PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

Escuela de Posgrado

Desarrollo e implementación de un sistema de control avanzado no lineal de brazos y cabeza para un robot móvil

Tesis para obtener el grado académico de Maestro en Ingeniería de Control y automatización
que presenta:

Ing. Juan Manuel Gomez Quispe

Asesor:
Dr. Carlos Gustavo Pérez Zuñiga

Lima, 2023

Informe de Similitud

Yo, Carlos Gustavo Pérez Zuñiga, docente de la Escuela de Posgrado de la Pontificia Universidad Católica del Perú, asesor de la tesis titulada: "Desarrollo e implementación de un sistema de control avanzado no lineal de brazos y cabeza para un robot móvil", dejo constancia de lo siguiente:

- El mencionado documento tiene un índice de puntuación de similitud de 7%. Así lo consigna el reporte de similitud emitido por el software *Turnitin* el 11/06/2023.
- He revisado con detalle dicho reporte y la Tesis o Trabajo de Suficiencia Profesional, y no se advierte indicios de plagio.
- Las citas a otros autores y sus respectivas referencias cumplen con las pautas académicas.

Lima, 11 de Junio del 2023

Apellidos y nombres del asesor: Pérez Zuñiga, Carlos Gustavo	
DNI: 41864666	Firma:
ORCID: 0000-0001-5946-1395	

AGRADECIMIENTOS

El autor de este trabajo de tesis, Juan Manuel Gomez Quispe, agradece el apoyo del Proyecto de Investigación de Robot Móvil Teleoperado para Manejo de Afecciones de Salud Mental en Pacientes con Enfermedades Infecciosas a través de contrato E041-2020-02-01 de proyectos de investigación aplicada y desarrollo tecnológico de FONDECYT-CONCYTEC en el marco en el cual se ha desarrollado esta tesis.

De la misma forma, quiero agradecer al Dr. Carlos Gustavo Pérez Zuñiga por su ayuda y asesoramiento en la elaboración de la presente Tesis y a los profesores de la maestría en general que siempre estuvieron atentos a las consultas realizadas durante este tiempo de estudios e hicieron posible también la implementación de este proyecto.

Por último, quiero agradecer el apoyo incondicional que me brindaron durante todo este proceso de estudios de posgrado a mis padres Mauro y Eutropia, Mi hermana María, mis sobrinos Josselyn y Nicolás y mi novia Rosmery.

RESUMEN

En el contexto de la pandemia de COVID-19 entre los años 2020-2021 se ha podido observar un crecimiento en los casos de enfermedades mentales como la depresión y ansiedad producto del aislamiento obligatorio a los que se tuvieron que someter muchos pacientes por la infección viral lo cual no permitía el contacto directo con los especialistas de la salud mental para el tratamiento de estas enfermedades. Así es como el proyecto del robot teleoperado para el tratamiento de afecciones mentales busca llevar la consulta del especialista hacia los pacientes a través de una videoconferencia y a la vez llevar toda la comunicación incorporada del robot a partir de los gestos que pueda realizar y que entable una conexión emotiva entre robot y paciente que permita facilitar la consulta teleoperada.

En el marco de la tesis presentada, se resolverá el control necesario para la generación de los gestos que debe realizar el robot a través de sus brazos y cabeza y lograr de esta manera la conexión emocional con el paciente durante su consulta, lo cual conlleva a una realización de movimientos de manera natural y muy próximas a las que realizaría una persona cuando se comunica utilizando todo el cuerpo para expresar emociones.

Para lograr este control sofisticado será necesario el uso de un controlador no lineal diseñado a partir del modelo matemático de los brazos y cabeza del robot. Se realizará tanto el diseño a partir de simulaciones de movimientos del robot, así como la implementación desde un prototipo inicial donde se probarán todas las rutinas validades por especialistas de la salud mental.

Finalmente, la tesis obtendrá como resultado, una comparativa de los principales controladores no lineales como producto de la elección del controlador más adecuado para este proyecto. También explicará a detalle la implementación de los algoritmos necesarios para el control de las extremidades del robot y se mostrarán técnicas de programación que permitan el control y la obtención de datos de servomotores en tiempo real para su posterior análisis.

INDICE GENERAL

INDICE DE FIGURAS

INDICE DE TABLAS

INTRODUCCION

Hoy en día se observa la aplicación de sistemas robóticos en diferentes aplicaciones tales como industria, transporte, farmacéutica, salud e incluso en aplicaciones sociales. Es así que los sistemas robóticos se vienen volviendo parte cotidiana de nuestro día a día y cada vez toma mayor campo de acción sobre las actividades humanas.

Solo en las aplicaciones de salud en centros hospitalarios, se vienen desarrollando robots para distintas actividades basándose inicialmente en tareas humanas repetitivas como por ejemplo la toma de muestras para laboratorio, realizar alguna teleconsulta a algún paciente, toma de temperatura y hasta en algunas oportunidades medicación. Todas ellas llevan consigo una interacción directa con los pacientes que es un tema con grandes perspectivas en la robótica social. Es así, que en el contexto de enfermedades infecciosas como aun es el caso del COVID-19, en la PUCP se viene desarrollando un proyecto para el diseño e implementación de un robot humanoide teleoperado que posibilite el tratamiento de afecciones mentales en pacientes con enfermedades infecciosas, con el fin de minimizar las probabilidades de infección de los especialistas y de ampliar la atención a pacientes que padecen de estas enfermedades, además de las complicaciones que se tienen para atender pacientes con equipos de protección que dificultan la tarea y a la falta de medios para atención sin contacto. Estos son factores que hacen que (actualmente) haya muy poca atención para pacientes que presenten cuadros de enfermedades psicológicas previas o también para los pacientes, que, a raíz de una enfermedad infecciosa, y su posterior aislamiento, generaron perfiles de enfermedades mentales.

En este contexto, el robot propuesto (dentro de este proyecto de investigación) tiene el principal objetivo de movilizarse dentro de ambientes hospitalarios de manera autónoma llevando a cabo algunas tareas específicas como la consulta teleoperada de un especialista hacia un paciente y que ayude en las tareas de interlocución entre ambas partes evitando así un contacto directo con el paciente y posteriormente sea el inicio de una plataforma de ejercicios de rehabilitación y pruebas psicológicas que se puedan implementar.

De esta manera, es de vital importancia que el robot permita generar un dialogo corporal donde los movimientos se generen de una manera suavizada y natural que le

brinden al paciente naturalidad en su comunicación con el terapeuta que se encuentra a distancia. En ese sentido, la presente tesis se enfoca en desarrollar e implementar algoritmos de control que permitan el movimiento de los brazos y la cabeza del robot tal como lo haría una persona.

Específicamente, la presente tesis se enfoca en la integración de movimientos de 2 brazos robóticos con 4 grados de libertad cada uno y una cabeza robot de 2 grados de libertad que permitan generar movimientos controlados en posición y velocidad por controladores no lineales basados en sus modelos matemáticos, utilizando para esto componentes electrónicos como servomotores y microcontroladores que permitan la integración y programación de las rutinas de gestos a ejecutar.

El desarrollo del trabajo que se presenta a continuación, se ha divido en 4 capítulos referidos al avance progresivo de las actividades realizadas para su implementación.

En el Capítulo 1 podremos dar una revisión al estado de arte de la robótica en la salud y la sociedad, una revisión profunda de los sistemas robóticos actualmente implementados, así como el contexto en el cual se crea la necesidad de un robot para el tratamiento de las afecciones mentales. El capítulo termina describiendo algunos casos prácticos realizados en Perú, así como una breve explicación sobre los sistemas dinámicos y de control.

Dentro del capítulo 2 se podrá visualizar la etapa de diseño del sistema robótico, así como las pautas necesarias para realizar un correcto modelo matemático, empezando por el análisis cinemático para obtener los alcances y espacio de trabajo de cada extremidad. Posteriormente, se realiza el análisis dinámico que nos permita obtener la ley que gobierna el movimiento de los sistemas.

Con el modelo matemático obtenido, en el capítulo 3 se realiza el diseño y simulación del sistema de control. Se ha determinado una comparativa de las leyes de control más recurrentes en la industria tales como Control PD no lineal, Backstepping, Sliding Mode, etc. El resultado de esta comparativa nos brinda la elección del controlador que cumple mejor las necesidades del proyecto y que por su efectividad, nos brinda un método seguro para su implementación en código abierto. En este capítulo también se verá la forma de generar las trayectorias deseadas que seguirán los brazos y cabeza utilizando su controlador no lineal, así como las simulaciones respectivas en lazo cerrado para comprobar el funcionamiento en la implementación.

Finalmente, en el capítulo 4, se presenta el hardware a utilizar para la implementación, así como el protocolo de comunicación CANBUS para todos los equipos. Luego se brinda los alcances sobre el esquema de control utilizado, la producción del código abierto para el control inicialmente en lazo abierto y luego en lazo cerrado para cada extremidad. Debido a que el control PD no requiere la información del modelo de la planta, es el primer controlador implementado para que nos ayude a encontrar, en base a pruebas y regresiones lineales de los resultados, los últimos parámetros del modelo matemático. Finalmente, teniendo todos los resultados necesarios, se implementa el controlador final Backstepping y se procede a realizar las pruebas y análisis de los resultados obtenidos en la implementación de los movimientos con el controlador no lineal y la generación de trayectorias deseadas.

La memoria de tesis culmina brindado conclusiones en base a los resultados obtenidos en el prototipo inicial utilizado para el proyecto del robot para el tratamiento de afecciones mentales.

Como resultado del trabajo realizado en la presente memoria de tesis, el diseño, la implementación, las pruebas y los resultados han sido incluidos en un artículo científico donde se presentan los temas más relevantes del proyecto realizado.

1. ESTADO DEL ARTE

1.1. Motivación de la investigación

Los trastornos mentales son afecciones que impactan en el pensamiento, la percepción, las emociones, la conducta y las relaciones interpersonales. Entre las principales afecciones tenemos a la depresión, la bipolaridad, la esquizofrenia, demencia, discapacidades intelectuales y de desarrollo, así como el autismo. Según la OMS [1], hasta finales del 2019, solo los casos de depresión han afectado a más de 264 millones de personas en el mundo. Los trastornos mentales pueden generar problemas sociales y económicos tanto a la persona afectada, a sus cuidadores, y familiares. En el año 2015, la OMS realizó una estimación de los costos referidos a la atención de estas enfermedades, que ascendieron a US$ 818 000 millones, lo cual representaba el 1.1% del producto bruto interno (PBI) mundial como una media por país de ese año.

Muchos de los síntomas presentes en los trastornos mentales implican, para el caso de la depresión, la tristeza, pérdida de interés, no disfrutar, sentir culpa, baja autoestima, falta de sueño o apetito, cansancio, desconcentración. Para el caso de la esquizofrenia, son las anomalías del pensamiento, alucinaciones, delirios. Para el caso de la demencia, el deterioro de la función cognitiva (acto de procesar los pensamientos), pérdida de la memoria, orientación, aprendizaje y el juicio.

Es de vital importancia entonces, entender que las afecciones mentales es un problema latente y que muchas de las investigaciones tecnológicas podrían dar solución en este campo. Para esto, se requiere la comprensión sobre las causas que generan estas afecciones. Entre las principales causas se encuentran los factores biológicos, lesiones cerebrales, el sentimiento de soledad, la aislación y el estrés. Muchos de los expertos de la salud coinciden que, si bien no existe una cura definitiva para estas afecciones mentales, es posible brindar un tratamiento preventivo a que la enfermedad pueda ser desarrollada bajo las causas anteriormente descritas. Es necesario por lo tanto que los pacientes expuestos a estos factores, así como las personas designadas a su delicada atención, tengan un monitoreo constante de evaluación psicológica con el fin de evitar la complicación de síntomas o reincidencia de los mismos.

En el contexto de una pandemia como el COVID-19, las causas de las afecciones mentales se intensificaron aún más, debido a la aislación de personas ya sea en hospitales, mediante las unidades de cuidados intensivos, así como la cuarentena

obligatoria en sus domicilios para muchas de las personas con síntomas de la enfermedad. Este escenario también complica el llevar la atención médica psicológica de estos pacientes debido a la exposición del personal de salud lo cual agrava la situación de estas personas generando también un alto nivel de ansiedad y agotamiento de las personas encargadas de su tratamiento.

1.2. Antecedentes de la investigación

El campo de la robótica no es ajeno a esta problemática, tanto así, que muchos de los investigadores han orientado esfuerzos a aplicaciones que den soluciones en base a lo que podemos denominar tecnologías asistenciales. La respuesta de la robótica a la pandemia del COVID-19 ha sido tema de estudio en diferentes investigaciones. Por ejemplo, [2] (Robin Murphy et al., 2020) han descrito, en sus estudios hasta 6 categorías de robots (entre terrestres y aéreos) que dan apoyo eficiente a diferentes campos que luchan contra la pandemia. En la Figura 1.1 podemos ver un cuadro resumen del estudio respecto a la clasificación de estos tipos de robot.

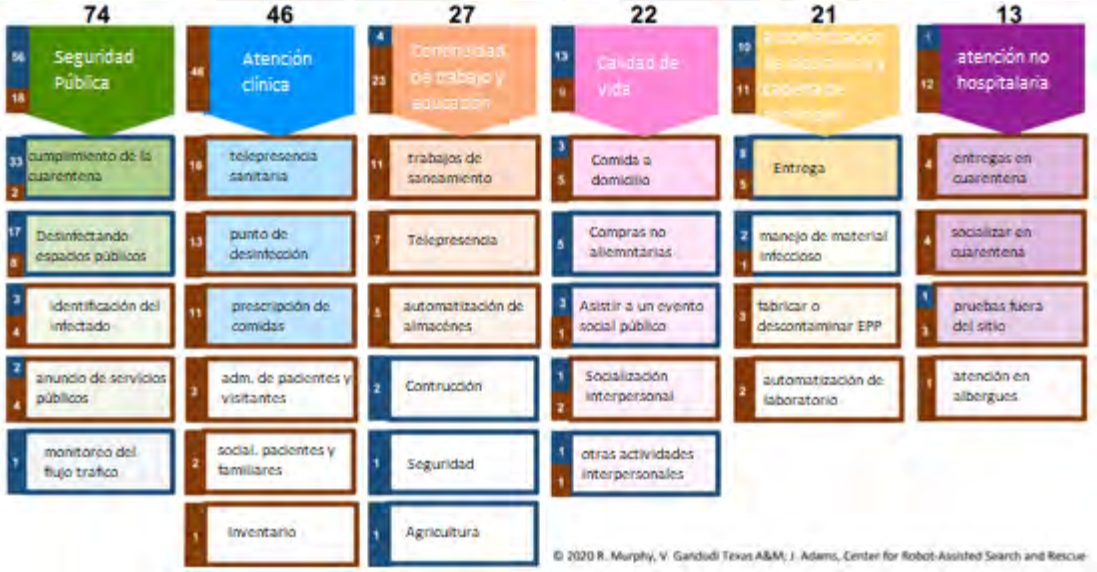


Figura 1.1. Aplicaciones Robóticas contra COVID-19 entre Marzo – Julio 2020. [2]

En la actualidad, la robótica ha alcanzado muchos campos de aplicación como la industria tecnológica, farmacéutica, transporte, monitoreo, etc. Es innegable que la comprensión de los sistemas robóticos demande un conocimiento profundo de muchas materias adicionales según sea el caso, como por ejemplo mecánica, informática, economía, matemáticas, física e incluso se han creado nuevas disciplinas que den soporte a las diversas aplicaciones que se van generando con el uso de la robótica.

El estudio profundo de la robótica ha conllevado que las aplicaciones con robots vayan más allá del ámbito industrial. El acoplamiento de la Robótica a la pandemia COVID-19 nos da prueba de este punto y que además los Robots cuentan con una flexibilidad para llevar a cabo diferentes actividades en comparación a las que inicialmente fueron creadas. Un ejemplo de este último punto es el "Robot Ninja" [3], el cual fue originalmente creado para el monitoreo de pacientes con accidente cerebrovasculares en recuperación y hoy en día se ha reprogramado para apoyar en diversas tareas a los doctores que trabajan como primera línea de defensa en la lucha contra la Pandemia, en diversos hospitales de Tailandia. Así como este caso, existen otras tecnologías como LHF connect en Italia [4], el Robot Eva [5] en Chile, Temi [6] en México los cuales son robots que se han distribuido para llevar a cabo fundamentalmente la consulta médica a los pacientes mediante la autonavegación y la telepresencia.

Por otro lado, en una rama de la robótica, se ha venido evaluando desde años atrás, la interacción que los robots pueden tener con personas en un entorno social, intentando que el robot acumule aprendizaje para tener respuestas naturales y enfrente diversas situaciones sociales tal como un ser humano lo haría.

La interacción entre robots y humanos (HRI – Human Robot Interface), es un campo de estudio que no está completamente abordado pero que en gran medida es una base en la cual se asienta la investigación de los robots asistenciales. [7] (Jenay M. Beer et al., 2017) concluye que la HRI se basó inicialmente en la interacción de los humanos con las computadoras (HCI – Human Computer Interface) a lo cual se le adiciona la compleja interacción social y su respuesta al entorno en el cual se desenvuelve. Aquí también definen 2 tipos de interacción, una es la interacción remota donde se enfatiza principalmente la funcionalidad para que el robot realice una tarea determinada como por ejemplo en rescate o búsqueda y por otro lado menciona a la interacción próxima como la interacción directa entre el humano y robot, y a diferencia de la interacción remota, se debe asociar los temas de afectos y sociabilidad a las funcionalidades del robot. [8] (Moon and Nass., 1996), en su estudio, comprueba que los seres humanos desarrollan, sin planearlo, ciertos afectos hacia los robots con lo que interactúan regularmente, sin haberles desarrollado algún algoritmo de interacción o afectividad, en algunos casos llegan a hablar con ellos y hasta ponerles un nombre. Estos precedentes nos indican que la apariencia física del robot termina siendo un primer paso para la HRI, como la primera impresión lo es en muchos casos para una relación

interpersonal entre seres humanos, y su desarrollo en el tiempo depende en gran parte de las reacciones del robot a los estímulos externos.

Estas reacciones del robot, dentro del marco de una HRI, se encuentran gobernadas principalmente por el mensaje de respuesta que expresa el robot a través de un dialogo programado, también, en la mayoría de ocasiones, a través de gestos corporales y faciales de robot. Nosotros como seres humanos somos capaces de percibir mensajes adicionales a partir de la comunicación no verbal, así como también al comunicarnos, realizamos distintos movimientos en el cuerpo de manera voluntaria e involuntaria. A estos movimientos corporales y faciales durante la transmisión de un mensaje, el estudio en [9] (Ly and Chignell, 2010), los denomina "**comunicación incorporada**" y es donde se basa el hecho de que los robots puedan mejorar la interacción con las personas, a partir de gestos, diferentes mensajes adicionales, así como crear un vínculo emocional y de confianza de forma más natural con la persona con la que interactúa.

Si bien es cierto, aún no existe como tal, un procedimiento establecido para el diseño de los gestos a realizar por un robot, en la mayoría de los casos, es suficiente con la implementación de mecanismos que permitan recrear los movimientos más recurrentes de las personas dando así, al robot una impresión de ente social inteligente. En el estudio de Ly and Chignell [9], concluye que las personas distinguen mejor las emociones de una información con movimientos cinemáticos y dinámicos en lugar de una información estática, además que las características de este movimiento pueden afectar incluso, en gran medida, la apariencia del robot. Estos movimientos deberán generar gestos que se han clasificado hasta en dos tipos:

- Los gestos expresivos para comunicar estados afectivos y emocionales a partir de movimientos de las manos mientras se hablan.
- Los Gestos simbólicos a partir de movimientos que expresen algún mensaje como la agitación de las manos para un HOLA o la guía gestual para indicar la forma y tamaño de un objeto.

Dentro de algunos trabajos importantes se encuentra ARI (PalRobotics – Spain [10]) el cual ha logrado realizar diversas pruebas sociales y aprende gracias a la inteligencia artificial que tiene programada, mejorando su adaptabilidad en la interacción con personas. Otro ejemplo es el robot MAGGIE [11] que pretende ser un módulo de aprendizaje de interacción social que abre las puertas a nuevas investigaciones. Un

último ejemplo a mencionar es el Robot Pepper [12] que, dentro de su aprendizaje a partir de la interacción social, también nos muestra la capacidad de reconocer si una persona está usando mascarilla. Estos últimos 3 ejemplos, que se muestran en la Figura 1.2, tienen un objetivo en común, que es relacionarse en un entorno social, y por supuesto, fuera de la inteligencia artificial que cada uno de ellos contiene, además poseen una apariencia amigable que imita a la de un ser humano, lo cual, invita a que la persona con la cual estén interactuando, sienta mayor comodidad de desenvolverse con estas tecnologías. Además, estos tipos de robots, en muchos de los casos, intentan siempre simular un lenguaje corporal para transmitir sus mensajes (sin dejar de lado la comunicación verbal) complementando así la información que desean llevar a la persona. El lenguaje corporal que practican estos robots lo realizarán a partir del movimiento inteligente de brazos y cabeza, así como expresiones faciales que responderán a estímulos externos durante la interacción.

(a) (b) (c)

Figura 1.2. (a) Robot ARI, (b) Robot Maggie, (c) Robot Pepper. [10], [11] y [12]

Estas cualidades en los robots presentados tienen como resultado final obtener un sentimiento de empatía hacia ellos por parte de los usuarios, haciendo más efectiva la interacción y comunicación entre el robot y las personas. Así también se ha demostrado en [13], como las expresiones en un robot mejoran la comprensión del afecto y emociones, incluso ayudan a atribuirles estados de ánimo.

La acción de la robótica en el marco de una pandemia y las afecciones mentales en conjunto con las investigaciones relacionadas a mejorar la interacción de los robots

con los pacientes presentado, son los puntos que se quieren desarrollar en el marco del proyecto denominado como “Robot Móvil teleoperado para Manejo de afecciones de Salud Mental en pacientes con enfermedades Infecciosas”. Este robot se encargará de llevar la consulta médica y monitoreo por teleoperación de un Psicólogo a los pacientes aislados en un entorno hospitalario mediante test psicológicos y juegos lúdicos, que han sido evaluados previamente por profesionales neuro-psicológicos. Para lograr este objetivo el robot tendrá las capacidades de autonavegación lo cual le permitirá llegar a la habitación o el espacio donde se encuentra el paciente y durante la consulta, el robot también será capaz de comunicarse con la persona por medio de expresiones corporales y gestos faciales que serán determinados de acuerdo a las indicaciones que esté brindando el Psicólogo durante la consulta médica generando una charla amigable con el paciente para que este pueda desarrollar la consulta de manera más confortable.

Este último punto que abarca la aplicación de gestos mediante los brazos y cabeza del robot, para lograr la interrelación del robot y el paciente, será tratada a profundidad en la memoria de tesis presentada para lo cual se iniciará presentando algunos puntos de importancia para el desarrollo del estado de arte.

1.3. Definición de un Robot

Al buscar el concepto en diferentes fuentes, siempre se tienen ciertas diferencias que permiten ver que “Robot” es un término que puede adaptarse a las necesidades y tareas que se requieran cumplir. Solo en el diccionario de la real academia española, la palabra robot toma hasta 4 diferentes significados que pueden complementarse entre sí. Uno de ellos, tal vez es el más acertado, es “Maquina programable que es capaz de manipular objetos y realizar diversas operaciones”. Para complementar esta definición es necesario saber que el robot puede ser una entidad autónoma compuesta por una mecánica artificial y sistemas electromecánicos.

Vamos a definir algunas expresiones relacionadas al uso de los robots.

- Grados de Libertad (GDL). - el grado de libertad en robots es una variable necesaria para definir los movimientos y la dirección del mismo. Por ejemplo, para definir un punto en el plano es necesario 2 variables, en el espacio es necesario tener 3 variables y estas variables vienen a ser en la mayoría de casos los GDL.

- Espacio de Trabajo. – Para un robot, el área de trabajo está definida como el alcance que tiene el robot para llegar a un punto desde su posición cero (0).
- Servomotores. – Accionadores de precisión que se encargan de realizar el movimiento de un sistema mecánico. Para efectos de la robótica, viene a ser en muchos casos el elemento de control que le brinda 1 grado de libertad al robot.

- Encoder. – Son sensores que son capaces, a partir de una electrónica detallada, de brindar la posición o velocidad de un sistema mecánico en rotación.

También es importante resaltar los proyectos que se han llevado a cabo hasta la actualidad en el país, como referencia y punto de partida para el proyecto a ser implementado. Es por eso que a continuación se mostrará algunas de estos sistemas robóticos aplicados en el Perú.

1.4. El Perú y la Robótica

Hay diferentes aplicaciones que se están llevando a cabo actualmente dentro del contexto de la robótica peruana. Todos ellos intentan brindar comodidad a los usuarios y relevar en algunos casos la acción humana en trabajos forzados y peligrosos. Tal es el caso de los robots Pterodáctilo y Spider minero en minera Cerro Verde [14] los cuales se pueden visualizar en la Figura 1.3. El primero se encarga de la seguridad del personal ante posibles caídas de mineral en el proceso de molienda, mientras que el segundo robot tiene como función principal la medición, mediante ondas de ultrasonido, del espesor de grandes tanques evitando así la exposición a alturas de los trabajadores.

(a) (b)

Figura 1.3. (a) Pterodáctilo extiende alas. (b) Spider robot con arnés [14].

Se debe considerar que estos robots están sometidos a ambientes agresivos. Por otro lado, también se tiene, en la minera Yanacocha, a Geobot mostrado en la Figura 1.4 quien ayuda en los trabajos de exploración a partir de la manipulación de explosivos y el alcance a zonas donde los trabajadores no tienen acceso [15].

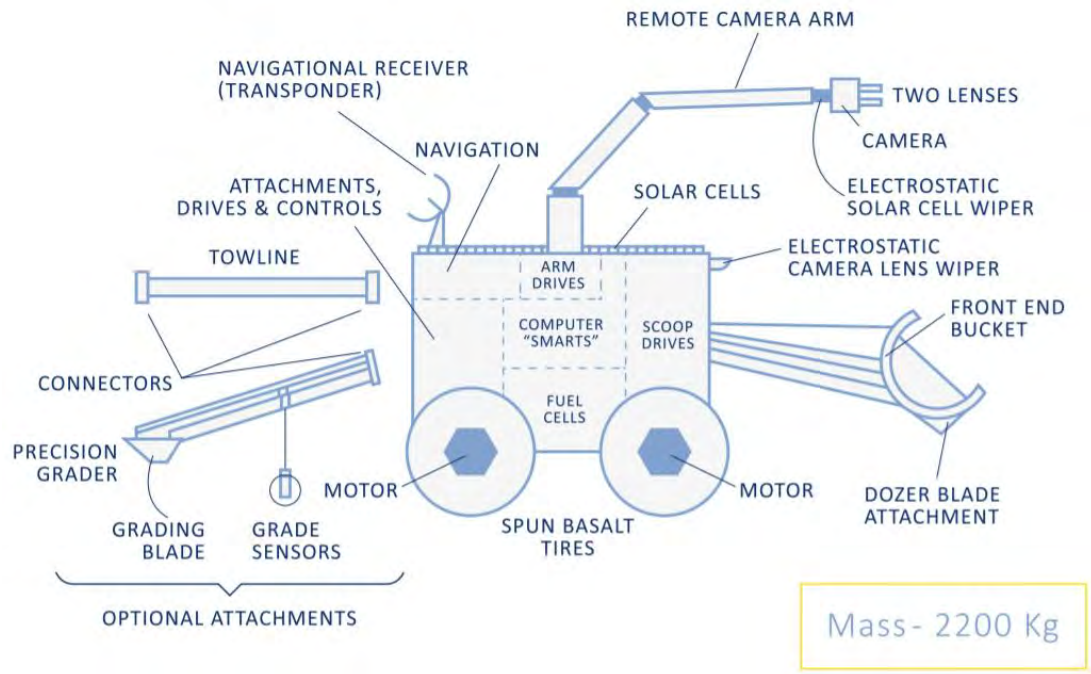


Figura 1.4. Esquema Geobot en trabajos de exploración minera [15].

Dentro del ámbito social también se cuentan con algunas aplicaciones robóticas como por ejemplo RobotMan [16] mostrado en la Figura 1.5.

Figura 1.5. Supervisor RobotMan. [16.]

Este robot tiene la capacidad de realizar rondas de videovigilancia con detección automática de personas, así como la interacción y ayuda como un guía a los usuarios dentro del centro comercial.

De esta manera podemos observar que, en la actualidad, dentro del contexto peruano, el proyecto "Robot Móvil teleoperado para Manejo de afecciones de Salud Mental en pacientes con enfermedades Infecciosas", vinculado a la Robótica asistencial, se está efectuando en vanguardia para el apoyo asistencial directo al personal médico, pese a que en el Perú ya se tiene varias aplicaciones robóticas dentro de un entorno social y de servicios.

Ahora que conocemos los intereses del proyecto del robot teleoperado para el tratamiento de las afecciones mentales, y en particular, se conoce que para lograr movimientos en los brazos y cabeza que colaboren a la empatía del paciente con el robot, se necesita de un sistema de control que logre obtener estos movimientos deseados.

Para esto, es necesario definir qué tipo de sistema y que control debemos realizar en el contexto de los sistemas dinámicos y la teoría de control.

1.5. Sistemas Dinámicos y su modelamiento

Se defina un sistema en general como la colaboración de distintos componentes para lograr un fin específico. Estos sistemas pueden ser de naturaleza física o abstracta donde se pueden encontrar diversas materias como la economía, la biología, etc. Desde este punto de vista, es posible realizar el estudio de un sistema, encontrando la relación que tiene sus distintos componentes entre sí.

Ahora, un sistema dinámico se caracteriza por que en todo momento la salida del sistema en el presente depende de la entrada en ese instante y sus valores en tiempos pasados. Si la presente salida de un sistema solo depende de la entrada instantánea, entonces estamos ante un sistema "estático" y en ese sentido podremos ver que la salida del sistema será constante si la entrada no sufre perturbaciones. En cambio, en los sistemas dinámicos, la salida siempre estará variando con el tiempo mientras no se llegue a un estado estable. Una forma de estudiar los sistemas dinámicos es a partir de los modelos matemáticos, los cuales nos ayudan a comprender como evoluciona un

sistema en el tiempo, incluso nos permite realizar predicciones sobre su salida por lo que el modelo se convierte en una herramienta valiosa para la ingeniería [17].

Dentro de los sistemas dinámicos podemos encontrar a los sistemas lineales y no lineales.

- Los sistemas lineales consideran una proporcionalidad entre la entrada y la salida, pero la característica más importante de un sistema lineal, se basa en el principio de superposición. Este principio establece que la salida de un sistema ante la aplicación en conjunto de más de una entrada puede ser encontrada a partir de la suma de las salidas que se generan por cada entrada. Esta ventaja permite entender de mejor manera a un sistema lineal cuando se le aplica distintas perturbaciones con solo analizar el efecto de cada una de las perturbaciones de manera independiente.

- En los sistemas no lineales, ya no aplica el principio de superposición y además estos sistemas están relacionados por ecuaciones diferenciales no lineales lo cual conlleva una seria complicación para su análisis. En muchos casos se aplica una linealización de este sistema no lineal donde se pueda observar solo un rango en el que se vea un comportamiento lineal aproximado con lo cual ya es posible realizar un mejor análisis. También es necesario recalcar que ya existe un estudio detallado para este tipo de sistemas no lineales desde el cual se puede tener cierto entendimiento de la evolución del sistema, sin embargo, este análisis está ligado directamente al nivel de la no linealidad del sistema y al modelo que se pueda encontrar.

Para el caso de los sistemas robóticos, están considerados como sistemas mecánicos no lineales donde el nivel de no linealidad se encuentra directamente relacionado a los grados de libertad que tenga el robot. Por ejemplo, el caso de un robot manipulador de 2 grados de libertad, es posible manejarlo con relativa facilidad mientras que para sistemas de 3 GDL o más ya vamos a obtener un sistema altamente no lineal.

1.6. Sistemas de Control

Dentro de los sistemas dinámicos, se debe entender que serán constantes en caso sean estables. En muchos sistemas, estos terminan siendo inestables. En cualquiera de los

casos, estable o inestable, el operador no tiene decisión sobre cómo responderá la planta, dado que esto es una naturaleza propia del sistema.

Para lograr cierta manipulación del sistema dinámico, el operador debe recurrir a un dispositivo adicional que le permita ingresar una señal, y luego la salida de este dispositivo pueda proporcionar a la planta una señal de entrada adecuada para llegar a un punto de interés. Cuando realizamos este procedimiento, estamos aplicando un sistema de control donde el dispositivo instalado viene a ser el controlador de la planta. En este esquema, existen dos formas de aplicar el control.

Control en Lazo abierto. – Aquí el operador será capaz de inyectar la señal de entrada directamente al controlador, la cual será escogida sobre todo por la experiencia que tiene este operador del proceso. Es decir, el operador sabe por experiencia que, al aplicar cierta entrada, la planta o sistema deberá responder con cierta salida. En muchos casos esto puede ser aplicado de buena manera, pero podemos notar que si la planta no responde como se espera (por posibles perturbaciones o cambios internos de la planta), no será posible llegar al punto de operación deseado y el operador deberá variar de manera empírica la señal de control esperando obtener una respuesta próxima a la deseada. La ventaja de un sistema en lazo abierto es que es de sencilla implementación y sobre todo muy económico. En la Figura 1.6 podemos visualizar un diagrama de bloques que nos explica cómo funciona el control en lazo abierto.

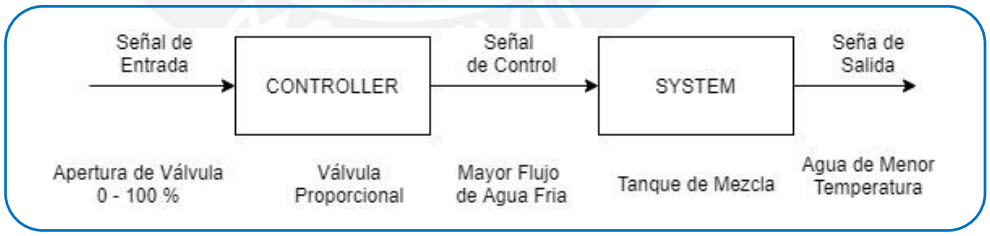


Figura 1.6. Sistema de control en lazo abierto (Elaboración propia).

- **Control en Lazo cerrado. –** Dentro de un sistema de lazo cerrado podemos encontrar mayor dominio del sistema dado que para su implementación nos apoyamos de un tercer elemento que es el sensor de medición de la salida del sistema. Esta medición la comparamos ahora con la salida que pretendemos obtener (set point) y el error resultante será el ingreso al controlador. De esta manera cada vez que haya una perturbación en la planta, los cambios de la

señal de salida serán comunicados a través del sensor hacia el controlador el cual determinará una nueva señal controlada para establecer y mantener siempre el punto de operación adecuada. En la Figura 1.7 encontramos un ejemplo de lazo de control cerrado.

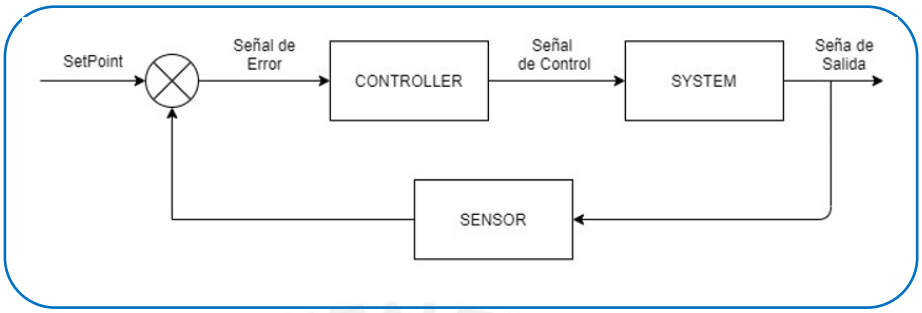


Figura 1.7. Sistema de Control en lazo cerrado (Elaboración propia).

Para mantener el punto de operación ante perturbaciones de la planta, el controlador deberá realizar ciertas oscilaciones en el sistema, las cuales pueden llevarlo a una inestabilidad por lo que es imperativo, que, al utilizar un controlador en lazo cerrado, este se diseñe de buena manera para evitar en primer lugar, llevar al sistema a una zona en la cual sea inestable y, en segundo lugar, llevar el sistema al punto de operación deseado.

Al igual que los sistemas dinámicos, los controladores pueden ser de naturaleza lineal y no lineal. Se debe entender que en la mayoría de casos el controlador y el sistema dinámico deberían ser de la misma naturaleza, pero esto está condicionado siempre a los requerimientos que se desean alcanzar.

Respecto a los controladores lineales, su eficacia para estabilizar un sistema lineal se ha estudiado continuamente al punto que en la actualidad se tiene toda una teoría para su diseño. En algunas ocasiones, incluso cuando se tiene un sistema no lineal, se procura realizar la linealización del sistema alrededor de un punto de operación deseado y de esta manera poder trabajar en el entorno seguro de un sistema lineal. El controlador más representativo de este tipo es el PID, sobre el cual existe una amplia teoría sobre su implementación y diseño. Además, incluso se han creado variantes de este controlador lineal para acoplarse de buena manera incluso a sistemas no lineales.

Por otro lado, los controladores no lineales, suelen llevar complicaciones al momento de realizar el diseño. Si bien es cierto, que ya existe una base teórica para los controladores no lineales, esto solo pueden ser aplicados a algunos sistemas dinámicos

que cumplen ciertos requerimientos por lo que complica aún más su implementación en muchos sistemas no lineales. La ventaja de utilizar estos controladores es que, al considerar las no linealidades del sistema, es posible llegar a tener, en muchas ocasiones, mejor performance y respuesta que los controladores lineales. La principal función de estos controladores no lineales es llevar al sistema a un estado estable, y evitar las inestabilidades que, para el caso de los sistemas no lineales, puede estar presente en más de un estado de operación. Al lograr la estabilización del sistema, es muy probable que el controlador no lineal pueda llevar al sistema al punto de operación deseado mediante algunas simples adaptaciones del control.

1.7. Generación de Gestos en sistemas robóticos.

Los sistemas de control tienen como objetivos el estabilizar, regular o seguir una trayectoria deseada. Incluso se puede considerar a la estabilización como un caso particular de la regulación con setpoint en el punto cero mientras que el seguimiento de trayectorias es considerado otro caso particular de la regulación con setpoint variable.

Los gestos que pueden realizar un sistema robótico pueden ser tomados como la ruta o trayectoria que se desea que siga el sistema. En el caso del proyecto realizado, dado que se tiene como objetivo entablar una relación emocional y de confianza con el paciente, no solo es suficiente tener un mínimo error en el seguimiento de la trayectoria sino también una afinidad en la forma en cómo se va desarrollando el movimiento hasta el punto de tener movimientos naturales similares a las de un humano. En ese sentido, la implementación de la ruta a seguir no solo sugiere seguimiento de una posición, sino además de la velocidad y hasta la aceleración con la cual se realiza dicho movimiento. De esta manera, es necesario implementar un control apropiado y que pueda realizar el seguimiento de posición, velocidad, aceleración y a la vez generar trayectorias especiales que cumplan con lo requerido.

La generación de trayectorias para sistemas robóticos es un tema muy estudiado y con vasta teoría, tanto que en la actualidad se tiene varios métodos para generar estas rutas de seguimiento que a su vez deben llevar la naturaleza en los movimientos seleccionados por un estudio paralelo que ayuden en la interacción con personas. En [18] podemos ver esta selección de movimientos que llevan el nombre de “gestos” los cuales son una colaboración en los movimientos de brazos y cabeza para transmitir un

solo mensaje que complementa la información que se brinda por medio de la voz. La intención de este estudio es implementar estos "gestos" que ayuden en la interacción que el paciente tiene con el especialista que lleva la consulta médica teleoperada.

1.8. Objetivos del Estudio

El propósito general del proyecto, del cual que forma parte la tesis presentada, es diseñar un robot médico asistencial que permita la consulta psicológica a los pacientes internos por enfermedades infecciosas que sufran afecciones mentales producto del aislamiento y que no permite que el personal médico pueda interactuar con él de manera directa. La idea nace a partir de la necesidad del personal médico de psicología de dar seguimiento a sus pacientes y nuevas personas afectadas durante el periodo 2020 – 2021 de la pandemia COVID-19 en el Perú sin exponerse directamente al contacto con pacientes infectados por lo que se considera un proyecto innovador en el país que involucra la tecnología robótica en entornos hospitalarios y se proyecta a ser un punto de partida para futuras aplicaciones dentro de la gama de robots médicos asistenciales.

El robot propuesto para estas actividades también está basado en la temática de la robótica social, con el fin de generar la empatía del paciente a realizar la consulta. Esto lo puede lograr a partir de un diseño estético que le permita tener rasgos humanos y la naturaleza de sus movimientos al interactuar con el paciente.

Así, el objetivo general del trabajo presentado es obtener movimientos coordinados y naturales de todas las extremidades del robot teleoperado (brazos y cabeza) los cuales sean controlados por sistemas realimentados.

La investigación desarrollada en esta memoria de tesis, estará enfocada justamente en el control de los movimientos de los brazos con 4 grados de libertad cada uno y cabeza de 2 grados de libertad del robot teleoperado, para mejorar la apariencia y coordinar los movimientos del robot y se realicen de formas más naturales, en términos de rapidez y suavidad, al punto de simular en gran magnitud los movimientos que realizaría un ser humano.

En ese sentido cada uno de los sistemas a controlar cuentan con 4 y 2 grados de libertad respectivamente, lo que convierte al robot en un sistema dinámico no lineal. De esta

manera nace la necesidad de utilizar algoritmos y sistemas de control avanzados que permitan el movimiento sincronizado de las extremidades del robot.

Para lograr este objetivo general del control realimentado de las extremidades del robot, entonces será necesario cumplir con los siguientes objetivos específicos que se muestran a continuación.

- Desarrollar las pruebas en lazo abierto para todos accionadores eléctricos de las extremidades del robot de tal forma de comprobar que estos sean capaces de llegar a las condiciones de operación, así como lograr la configuración interna necesaria de estos equipos para el control realimentado a utilizar posteriormente.
- De las pruebas realizadas, generar un modelo matemático para cada sistema mecánico del robot teleoperado que nos permita diseñar y desarrollar, en una siguiente etapa, un control no lineal avanzado basado en el modelo obtenido.
- Se podrá observar que, como parte de la implementación de este controlador avanzado no lineal, será necesario obtener pruebas a partir de controladores más sencillos como el PID y sus variantes, que será también un objetivo específico necesario para la continuidad y finalización del proyecto.
- La verificación y validación del control no lineal basado en el modelo matemático obtenido, que se ha seleccionado para el robot teleoperado se realizará a partir de pruebas simuladas e implementadas, que será otro de los objetivos a alcanzar para demostrar la efectividad del controlador finalmente propuesto.
- Durante el desarrollo del diseño del controlador a utilizar, se podrá observar diferentes opciones de control lo cual nos llevará a la elección de un controlador final que será llevada al a etapa de implementación. Esta selección del controlador a utilizar se realizará a partir del planteamiento de un índice de performance que deberá cumplir el sistema en conjunto y que nos garantice llegar a cumplir los requerimientos de proyecto. En [19] se muestra un camino para la comparativa de controladores que será tomada en cuenta para el desarrollo del presente trabajo en este punto.
- Otro objetivo del trabajo será que el controlador final obtenido sea sintonizado de acuerdo a las exigencias que requieran los movimientos que realizará el robot asistencial a implementar, con la finalidad de darle al paciente, las

condiciones adecuadas y la confianza de poder interactuar con él, llevando a cabo los movimientos necesarios para transmitir la "comunicación incorporada" esperada hacia el paciente.

El cumplimiento de los objetivos específicos presentados, nos dará como resultado el alcance del objetivo general del proyecto, que como ya se ha dicho, es controlar completamente los movimientos de los brazos y cabeza del robot, en términos de precisión y velocidad, lo cual contribuya a la interacción que tendrá el robot con los pacientes a ser tratados mediate la consulta teleoperada.

En los próximos capítulos de esta memoria de tesis, se presentará los tópicos de diseño e implementación de los brazos y cabeza del robot teleoperado enfocándose principalmente el en análisis de movimiento, modelos matemáticos del sistema físico, diseño del sistema central de control y el tópico final de implementación a nivel de hardware y software necesario para la funcionalidad autónoma y semi autónoma de las extremidades del robot.

2. MODELAMIENTO DE BRAZOS Y CABEZA DE ROBOT TELEOPERADO

El análisis y la comprensión de un sistema robótico, así como todo sistema mecánico, requiere de un modelo cinemático - dinámico que consta de ecuaciones matemáticas donde se relacionen los distintos componentes que conforman el sistema, así como también den una idea de cómo vienen interactuando entre sí en cada momento.

Para conseguir el modelo matemático, en robótica, se hacen uso de todas las leyes físicas de la cinemática y dinámica del movimiento los cuales serán aplicados en base a un procedimiento ya establecido para los sistemas de robots.

El análisis comienza con la cinemática del sistema, quien se encarga de explicar los movimientos que realiza el cuerpo sin tomar en cuenta que factores provocan que el cuerpo se mueva. De esta manera podremos tener una idea inicial de cómo será el movimiento del robot. Para poder tener un modelo más cercano al sistema real, ahora se deben involucrar las leyes de la dinámica que explican las razones por las que un cuerpo llega a tener movimiento. Este análisis será explicado a detalle a continuación de tal manera que podamos obtener las herramientas necesarias para realizar el modelo matemático de nuestro sistema, el cual será validado al realizar la comparación con el sistema real y posteriormente podamos diseñar el control tanto en lazo abierto como en lazo cerrado.

Para fines del proyecto, se ha realizado un diseño CAD inicial de los brazos y cabeza en el software INVENTOR el cual es mostrado en la Figura 2.1. A partir de este CAD es posible iniciar el análisis de movimientos que pueden tener cada GDL del sistema.

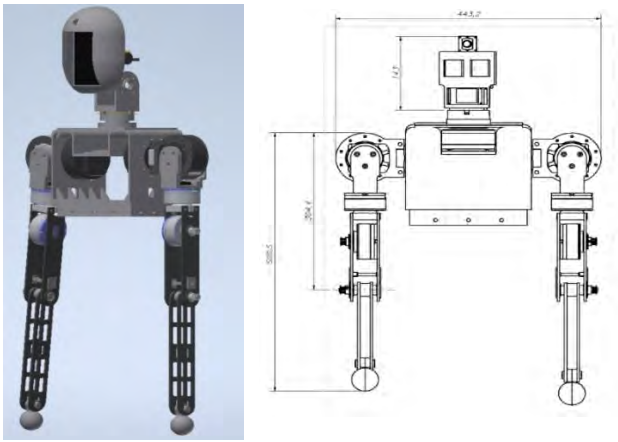


Figura 2.1. Diseño inicial CAD de robot (Elaboración propia).

2.1. Cinemática Directa

El procedimiento a realizar dentro de la cinemática directa requiere de algunos conceptos generales que se detallan en [18] y se muestra de forma resumida a continuación.

El movimiento de un cuerpo se puede entender mejor si analizamos como se mueve una partícula en el espacio. Los puntos en el espacio se mueven de dos maneras las cuales se presentan a continuación:

Forma Traslacional: Es el movimiento que realiza un cuerpo de un punto a otro tomando siempre un marco de referencia inercial. La distancia que el cuerpo se mueve, se puede hallar con la suma vectorial de la distancia de centros de los dos marcos de referencia **[A]** y **[B]** y la posición del punto respecto al segundo marco de referencia

$$\boldsymbol{P^A = P^A_{BORG} + P^B} \qquad (2.1)$$

Forma Rotacional: Se realiza una rotación del marco de referencia de **[A]** a **[B]** para un punto determinado. La rotación se puede denotar a partir de una Matriz de Rotación **R** con la siguiente expresión.

$$\boldsymbol{R^A_B = R^{B^{-1}}_A = R^{B^T}_A} \qquad (2.2)$$

Donde $\boldsymbol{R^A_B}$ es la matriz de rotación de **A** a **B.** Podemos notar que, para esta matriz de rotación, su inversa y su transpuesta son iguales lo cual podría simplificar algunos cálculos matriciales que puedan ser necesarios en el sistema a analizar.

El movimiento en conjunto, es decir, cuando un punto realiza una traslación y una rotación al mismo tiempo, se conoce como una **"transformación homogénea"** y se denota $\boldsymbol{T^A_B}$ que es la transformación homogénea de A hacia B.

$$\boldsymbol{P^A = T^A_B P^B} \quad \text{ó} \quad \boldsymbol{T_{AB}} = \begin{bmatrix} \boldsymbol{P^A} \\ \mathbf{1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{R^A_B} & \boldsymbol{P^A_{BORG}} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \qquad (2.3)$$

Debemos notar que esta transformación homogénea es una matriz de 4x4. Es así como se puede describir cualquier movimiento en el espacio de una partícula.

Para mayor entendimiento de los conceptos explicados, se muestra la Figura 2.2 donde se puede visualizar el punto **P** dentro de los sistemas de referencia **[A]** y **[B]** así como la traslación y rotación que realiza el punto P entre ambos sistemas de referencia.

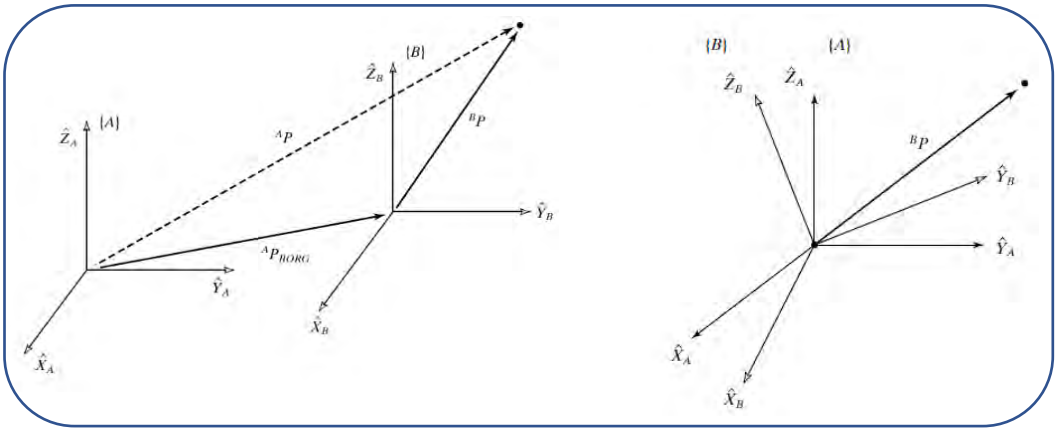


Figura 2.2. (a) Traslación de A a B. (b) Rotación de A a B. [18]

Es necesario tener en cuenta que, dentro de los sistemas robóticos, específicamente en el contexto de los manipuladores robóticos, se pueden definir los vínculos o segmentos como los cuerpos o partes del sistema que realizarán el movimiento. Estos segmentos se encuentran unidos mediante los eslabones o articulaciones que por lo general son del mismo número que los grados de libertad que tiene un sistema.

En algunos casos, estas articulaciones reciben el nombre de junturas. Algunos ejemplos los podemos visualizar en la Figura 2.3 donde los grados de libertad de cada articulación, por lo general es 1 pero hay algunos casos especiales como las articulaciones esféricas donde se podrá generar más de un grado de libertad.

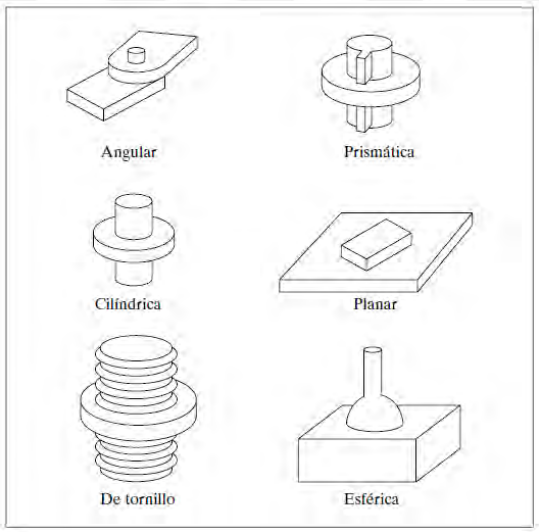


Figura 2.3. Distintos de tipos de articulación (juntura). [18]

En los sistemas robóticos donde se tiene grados de libertad conectados en serie, es importante definir el movimiento de un vínculo en base al marco de referencia de la articulación anterior y de esta manera podemos llegar a tener una relación de

movimiento entre el último vínculo del sistema con el marco de referencia de la base del sistema lo cual nos permitirá relacionar las posiciones angulares de cada articulación con la posición que tiene el último vínculo en el espacio.

Para lograr esta relación de movimientos entre vínculos de un sistema robóticos se utiliza la convención de DENAVIT-HARTENBERG (D-H) donde encontraremos un procedimiento para efectuar las diferentes rotaciones y traslaciones que se están efectuando en cada articulación a partir de la definición de transformadas homogéneas.

Convención Denavit - Hartenberg

En 1955, los físicos Jacques Denavit (1930 – 2012) y Richard Hartenberg (1907-1997) propusieron una metodología [20] para relacionar los sistemas de coordenadas de cada vínculo a partir de transformaciones homogéneas (matrices de 4x4). La metodología explica el paso a paso a seguir en una cadena articulada donde cada vínculo ha sido parametrizado mediante los valores $a_i, \alpha_i, d_i, \theta_i$ donde *i* es el número de articulación.

Un procedimiento para desarrollar el análisis cinemático, tomando en cuenta la metodología DH se detalla a continuación [22]:

Paso 1: Identificar los eslabones, articulaciones y ejes articulares:

- Número de eslabones de 0 hasta *n*.
- Número de articulaciones de 1 hasta *n* e identificar su eje

Paso 2: Asociar una trama $\{\boldsymbol{S_i}\}$ a cada eslabón *i*:

- Colocar el eje $\boldsymbol{Z_i}$ colineal con el eje de articulación *i+1*.
- Para cada trama identificar entre los siguientes casos:
    - Caso1: $\boldsymbol{Z_i}$ intercepta a $\boldsymbol{Z_{i-1}}$
    - Caso2: $\boldsymbol{Z_i}$ es paralelo a $\boldsymbol{Z_{i-1}}$

Paso 3: Asignar el origen de la trama $\{\boldsymbol{S_i}\}$- según sea el caso:

- Caso1: En el punto de intersección entre $\boldsymbol{Z_i}$ y $\boldsymbol{Z_{i-1}}$.
- Caso2: En cualquier punto a lo largo de $\boldsymbol{Z_i}$.

Paso 4: Asignar el sentido de $\boldsymbol{X_i}$ y $\boldsymbol{Y_i}$ según sea el caso:

- Caso1: $\boldsymbol{X_i}$ será siempre perpendicular al plano formado por $\boldsymbol{Z_i}$ y $\boldsymbol{Z_{i-1}}$.
- Caso2: $\boldsymbol{X_i}$ será siempre perpendicular a $\boldsymbol{Z_i}$ y $\boldsymbol{Z_{i-1}}$.

- $\boldsymbol{Y_i}$ siempre se escoge para completar el sistema dextrógiro.

Paso 5: Obtener los parámetros D-H para cada trama teniendo en cuenta la Figura 2.4.

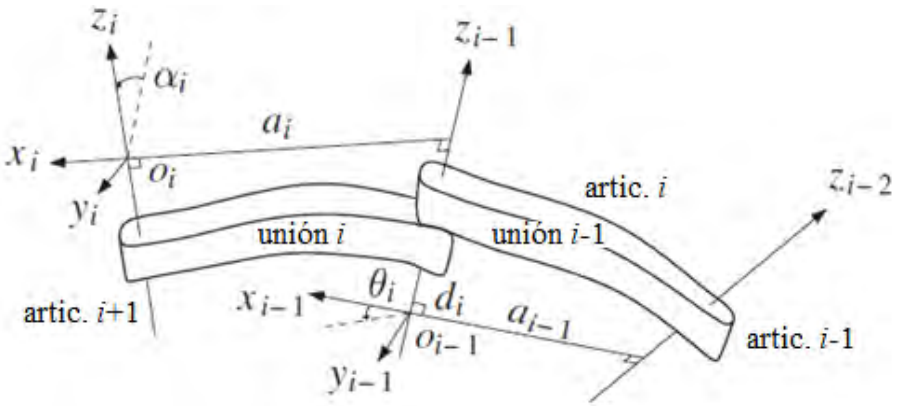


Figura 2.4. Asignaciones de los marcos Denavit-Hartenberg. [19]

Paso 6: Para cada par de tramas $\{\boldsymbol{S_i}\}\{\boldsymbol{S_{i-1}}\}$ obtener la matriz de transformación homogénea $\boldsymbol{A_i^{i-1}}$ que las relacionen.

Paso 7: Operar las matrices de transformación homogénea de cada par de tramas para obtener la matriz resultante que relacione la trama *n* con la trama 0.

$$\boldsymbol{A_n^0 = A_1^0 A_2^1 A_3^2 \dots A_n^{n-1}} \qquad (2.4)$$

Paso 8: Obtener las ecuaciones cinemáticas directa del vector de traslación contenido en la matriz $\boldsymbol{A_n^0}$.

Se debe tener en cuenta para el paso 5 que cada matriz de transformación homogénea es una secuencia de traslaciones y rotaciones los cuales dan como resultado la matriz de transformación para cada frame *i*.

$$\boldsymbol{A_i^{i-1}} = \begin{vmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{vmatrix} \qquad (2.5)$$

Así de esta manera, si conocemos los ángulos de cada articulación, podremos obtener la posición del vínculo final en el espacio cartesiano solo tomando los 3 primeros elementos de la última columna de la matriz de transformación $\boldsymbol{A_n^0}$.

2.1.1. Desarrollo de Cinemática Directa en Brazos Robótcos

Gracias a la metodología D-H y el procedimiento mostrado anteriormente se ha podido realizar el modelamiento, partiendo de la posición de los sistemas de referencia para cada grado de libertad y sus correspondientes parámetros DH mostrados en la figura 2.5.

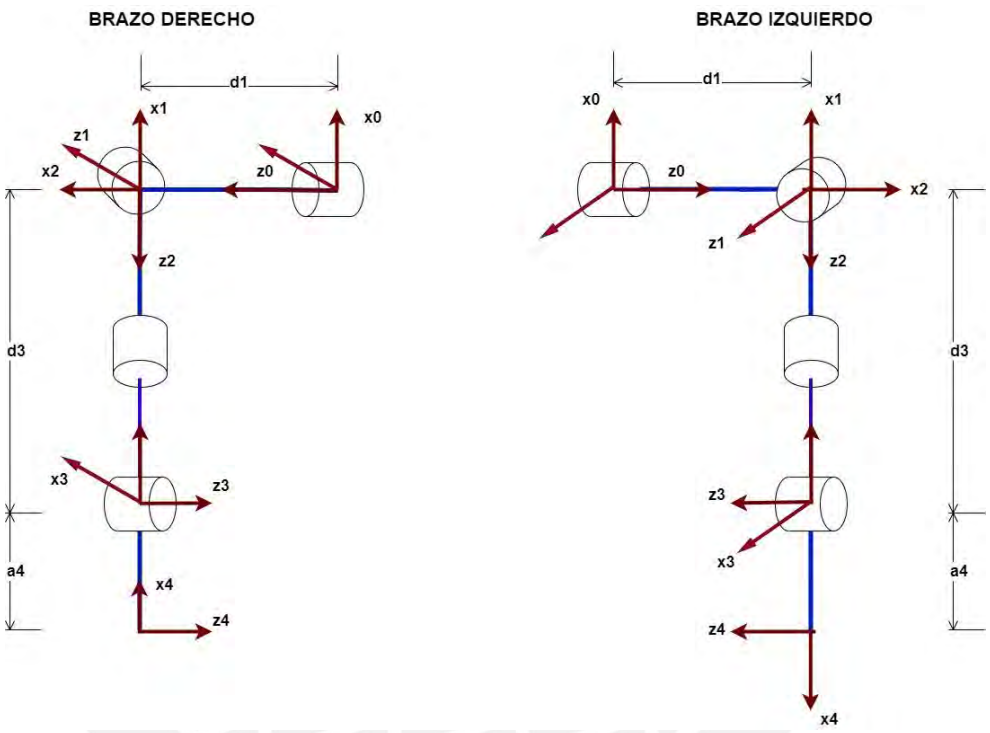


Figura 2.5. Análisis Denavit–Hartenberg Brazos Robot. (Elaboración propia).

El análisis base de la cinemática directa se está realizando, teniendo en cuenta que la posición HOME (ángulos en 0°) corresponde a la posición de descanso de los brazos de una persona.

Los brazos tienen 4 GDL cada uno con las mismas medidas para tener un sistema simétrico. Los análisis son completamente independientes uno del otro. Así mismo, los sistemas de referencias son similares, pero se han planteado en la disposición mostrada en la figura anterior para obtener parámetros D-H mostrados en la tabla 2.1.

Esta tabla corresponde al análisis de ambos brazos lo cual nos permite tener un único resultado y por lo tanto una solución que represente el movimiento de los dos brazos.

Tabla 2.1. Parámetros D-H para brazo Izquierdo y derecho.

i	a_i	α_i	d_i	θ_i
1	0	-90°	d_1	θ_1
2	0	90°	0	$\theta_2 - 90°$
3	0	-90°	d_3	$\theta_3 + 90°$
4	a_4	0	0	$\theta_4 - 90°$

Al obtener estos parámetros podemos obtener ahora las transformadas homogéneas respecto a cada grado de libertad utilizando la ecuación (2.5) descrita anteriormente. Estas matrices son mostradas a continuación:

$$\boldsymbol{A_1^0} = \begin{vmatrix} \cos\theta_1 & 0 & -\sin\theta_1 & 0 \\ \sin\theta_1 & 0 & \cos\theta_1 & 0 \\ 0 & -1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{vmatrix} \quad \boldsymbol{A_2^1} = \begin{vmatrix} \cos\theta_2 & 0 & -\cos\theta_2 & 0 \\ -\cos\theta_2 & 0 & \sin\theta_2 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

$$\boldsymbol{A_3^2} = \begin{vmatrix} -\sin\theta_3 & 0 & -\cos\theta_3 & 0 \\ \cos\theta_3 & 0 & -\sin\theta_3 & 0 \\ 0 & -1 & 0 & d_3 \\ 0 & 0 & 0 & 1 \end{vmatrix} \quad \boldsymbol{A_4^3} = \begin{vmatrix} \sin\theta_4 & \cos\theta_4 & 0 & a_4\sin\theta_4 \\ -\cos\theta_4 & \sin\theta_4 & 0 & -a_4\cos\theta_4 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \tag{2.6}$$

Es a partir de estas matrices que se puede obtener la transformación homogénea $\boldsymbol{A_4^0}$ haciendo uso de la ecuación (2.4) que vincula la muñeca del brazo robótico con el sistema de referencia base, la cual se muestra en la siguiente expresión:

$$\boldsymbol{A_4^0} = \boldsymbol{A_1^0 A_2^1 A_3^2 A_4^3} = \begin{vmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{23} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{vmatrix} \tag{2.7}$$

En donde:

$r_{11} = -sin\theta_4(\cos\theta_3\sin\theta_1 + \cos\theta_1\sin\theta_2\sin\theta_3) - \cos\theta_1\cos\theta_2\cos\theta_4,$

$r_{21} = -sin\theta_4(\cos\theta_3\cos\theta_1 - \sin\theta_1\sin\theta_2\sin\theta_3) - \sin\theta_1\cos\theta_2\cos\theta_4,$

$r_{31} = \cos\theta_4\sin\theta_2 - \cos\theta_2\sin\theta_3\sin\theta_4,$

$r_{12} = cos\theta_1\cos\theta_2\sin\theta_4 - cos\theta_4(\cos\theta_3\sin\theta_1 + \cos\theta_1\sin\theta_2\sin\theta_3),$

$r_{22} = sin\theta_1\cos\theta_2\sin\theta_4 - cos\theta_4(\cos\theta_3\cos\theta_1 - \sin\theta_1\sin\theta_2\sin\theta_3),$

$$r_{23} = -\sin\theta_4 \sin\theta_2 - \cos\theta_2 \sin\theta_3 \cos\theta_4,$$

$$r_{13} = \sin\theta_1 \sin\theta_3 - \cos\theta_1 \sin\theta_2 \cos\theta_3,$$

$$r_{23} = -\cos\theta_1 \sin\theta_3 - \cos\theta_1 \sin\theta_2 \cos\theta_3,$$

$$r_{32} = -\cos\theta_2 \cos\theta_3,$$

$$p_x = -a_4 \sin\theta_4 (\sin\theta_1 \cos\theta_3 - \cos\theta_1 \sin\theta_2 \sin\theta_3) - d_3 \cos\theta_1 \cos\theta_2 - a_4 \cos\theta_1 \cos\theta_2 \cos\theta_4,$$

$$p_y = -a_4 \sin\theta_4 (\cos\theta_1 \cos\theta_3 - \sin\theta_1 \sin\theta_2 \sin\theta_3) - d_3 \sin\theta_1 \cos\theta_2 - a_4 \sin\theta_1 \cos\theta_2 \cos\theta_4,$$

$$p_z = d_1 + d_3 \sin\theta_2 + a_4 \sin\theta_2 \cos\theta_4 - a_4 \cos\theta_2 \sin\theta_3 \sin\theta_4, \qquad (2.8)$$

Las ecuaciones descritas en (2.8) nos permiten representar a los brazos derecho e izquierdo a la vez. Aquí se debe analizar que las ecuaciones nos permiten obtener la posición y orientación de la muñeca para un determinado juego de valores para los ángulos $\theta_1, \theta_2, \theta_3$ *y* θ_4.

Para comprobar que el análisis se haya realizado de manera correcta, se han realizado simulaciones de los movimientos de los brazos utilizando la librería de Peter Corke [23]. Esta librería nos permite construir los brazos simulados en base a la información de los parámetros D-H encontrados y a partir de esto, mediante una GUI seleccionar las posiciones angulares deseadas y como resultados podremos ver la posición y orientación de la muñeca a la cual se ha llegado con el brazo robótico y mediante un algoritmo en un script Matlab independiente que utilizará la ecuaciones (2.8) y comandos de la librería "PeterCorke" se va a comprobar que ,coincida con los resultados obtenidos en simulación.

El resultado de la simulación y los valores de la posición con el script realizado se muestran en la Figura 2.6 donde se puede comprobar que los resultados coinciden.

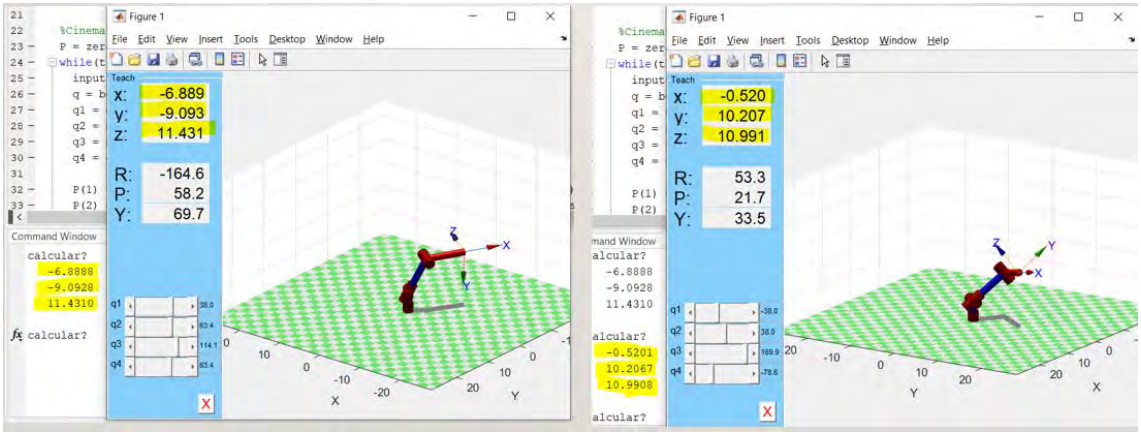


Figura 2.6. Simulaciones de cinemática directa para Brazos robóticos.

Para propósitos del proyecto, este análisis nos ayudará a validar la posición deseada a la que queremos llevar las muñecas de los brazos durante la implementación.

2.1.2. Desarrollo de cinemática directa en Cabeza del Robot.

De una manera similar al análisis de los brazos, se ha realizado el análisis de los movimientos de la cabeza teniendo los sistemas de referencia dispuestos como se muestra la figura 2.8.

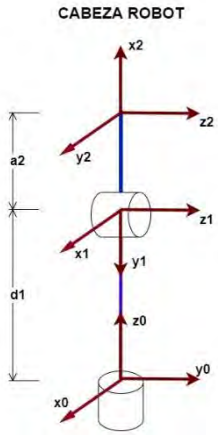


Figura 2.7. Análisis D-H de los GDL de cabeza robot (Elaboración propia).

De esta manera podemos obtener los parámetros D-H (tabla 2) y las matrices de transformación correspondiente para cada eslabón.

Tabla 2.2. Parámetros D-H para cabeza Robot.

i	a_i	α_i	d_i	θ_i
1	0	-90°	d_1	θ_1
2	a_2	0°	0	$\theta_2 - 90°$

$$A_1^0 = \begin{vmatrix} \cos\theta_1 & 0 & -\sin\theta_1 & 0 \\ \sin\theta_1 & 0 & \cos\theta_1 & 0 \\ 0 & -1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{vmatrix} \quad A_2^1 = \begin{vmatrix} \sin\theta_2 & \cos\theta_2 & 0 & a_2\sin\theta_2 \\ -\cos\theta_2 & \sin\theta_2 & 0 & -a_2\cos\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \tag{2.9}$$

Así obtenemos la matriz de transformación general A_2^0

$$A_2^0 = A_1^0 A_2^1 = \begin{vmatrix} \cos\theta_1\sin\theta_2 & \cos\theta_1\cos\theta_2 & -\sin\theta_1 & a_2\cos\theta_1\sin\theta_2 \\ \sin\theta_1\sin\theta_2 & \sin\theta_1\cos\theta_2 & \cos\theta_1 & a_2\sin\theta_1\sin\theta_2 \\ \cos\theta_2 & -\sin\theta_2 & 0 & d_1 + a_2\cos\theta_2 \\ 0 & 0 & 0 & 1 \end{vmatrix} \tag{2.10}$$

Teniendo en cuenta que los tres primeros elementos de la columna 4 de la matriz en la ecuación (2.10) nos dan las coordenadas XYZ del punto de interés en la cabeza, se realizan las simulaciones en la librería de Peter Corke tal como se muestra en la figura 2.8 donde también podemos apreciar además que en la posición home, los sistemas de referencia coinciden con el análisis realizado anteriormente.

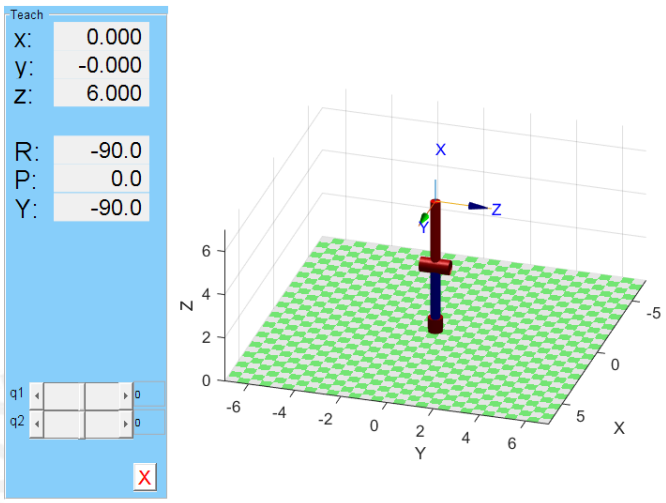


Figura 2.8. Posición HOME de cabeza robot en GUI Matlab.

Al asignar posiciones angulares desde la GUI de simulación, vemos que obtenemos una posición final XYZ la cual coincide con el script de ecuaciones de movimiento realizado en paralelo tal como se muestra en la figura 2.9.

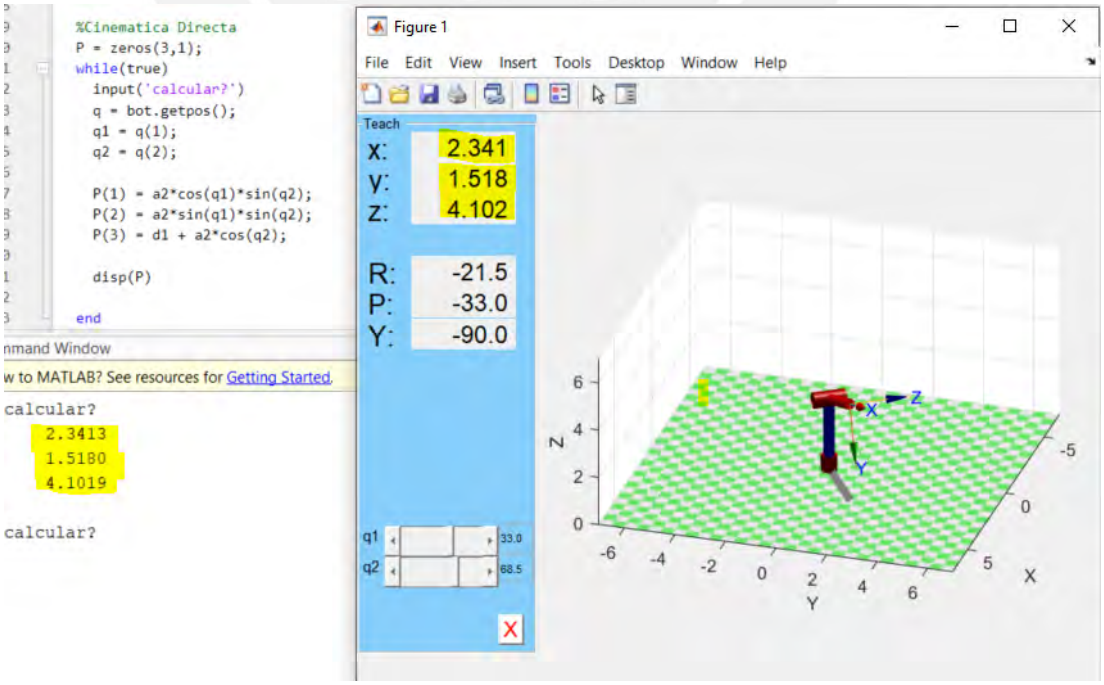


Figura 2.9. Simulación de movimientos de Cabeza Robot.

La formulación de la Cinemática directa de las extremidades del Robot nos ayuda a validar los resultados de parametrización matemática de los movimientos. Lo que se requiere ahora es con la ayuda de este análisis, hallar las ecuaciones que nos permitan definir las posiciones angulares que se requieren para una posición XYZ deseada. Esto involucra un nuevo análisis nombrado como cinemática inversa el cual vamos a desarrollar para cada extremidad del sistema robótico.

2.2. Cinemática Inversa

En este punto se tiene como objetivo que, al tener una posición definida en el espacio cartesiano, la cinemática inversa sea capaz de calcular los ángulos que debe tener cada articulación para lograr ese posicionamiento. Nos podemos dar cuenta que para lograr esto, debemos encontrar unas expresiones para cada ángulo de articulación en función de los demás parámetros del sistema robótico y la información de la posición cartesiana deseada.

También podemos observar que a partir de la cinemática directa donde se halló una expresión para la posición deseada en función de los ángulos de articulación deberíamos poder despejar los ángulos en las ecuaciones (2.8) y (2.10) y encontrar las expresiones necesarias para la cinemática inversa, sin embargo, nos daremos cuenta que el desarrollo de estas ecuaciones de posición son en realidad ecuaciones trascedentes que no permiten despejar las variables angulares de forma sencilla. Es por esto, que se recurren a distintos métodos que nos ayuden a resolver esta cinemática inversa.

En la mayoría de casos, la aplicación de métodos geométricos será suficiente para resolver esta problemática, sobre todo en aquellos problemas que involucren movimientos dentro de un plano geométrico, es decir, a partir de un análisis gráfico será posible determinar los ángulos necesarios para una posición del vínculo final en el espacio cartesiano. En otras oportunidades se hará uso de un análisis computacional para hallar estas posiciones angulares en tiempo real, lo cual obviamente es una solución práctica, pero en contraparte puede volver nuestro algoritmo complicado y presentar algunas deficiencias en la implementación como por ejemplo la lentitud en la respuesta.

Es necesario tener en cuenta que, como comentamos anteriormente, para definir una posición en el espacio, un sistema robótico tiene que tener como mínimo 3 GDL. En

el caso que se tengan más grados de libertad, y no sea un requisito la orientación, estos pasan a llamarse grados redundantes dado que no adiciona alguna característica de movimiento al vínculo final. Por el contrario, si los GDL es menor a 3, no será posible llevar al efector final a cualquier punto en el espacio de trabajo del sistema robótico.

De esta forma se han realizado diversos análisis para buscar una solución al análisis de la cinemática inversa en brazos y cabeza descrito a continuación.

2.2.1. Cinemática Inversa de brazos robóticos de 4 GDL

En el caso de los brazos robots implementados, y con la intención de hallar una solución cerrada (única) en [24] se propone una solución analítica a esta problemática para los 4 primeros GDL. En la figura 2.10, observaremos la analítica realizada previo a los cálculos necesarios para la solución cerrada.

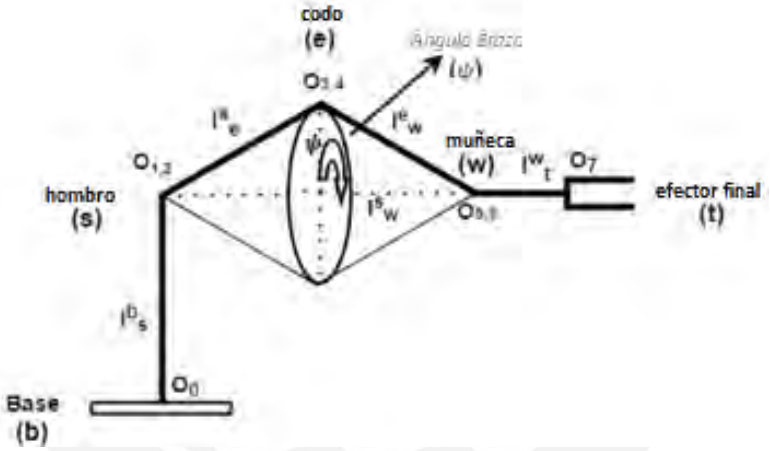


Figura 2.10. Movimiento parametrizado del manipulador redundante. [24]

En esta figura se puede observar que efectivamente, se pueden tener diferentes ternas de soluciones angulares para una única posición cartesiana de la muñeca. Sin embargo, estas infinitas soluciones, hacen que el codo del brazo (“Elbow” “e”) describa una circunferencia, donde cada punto es provocado por cada solución que existe para la cinemática inversa.

De esta manera se encuentra una relación entre una solución especifica y las infinitas soluciones adicionales que existen para los brazos robóticos.

Ahora, cada solución se lleva a cabo respecto a un plano, y en el caso de la solución a analizar, en [25] proponen llamarlo plano referencial mientras que las demás soluciones se estarán dando dentro de planos llamados planos de brazo tal como se muestra en la figura 2.11.

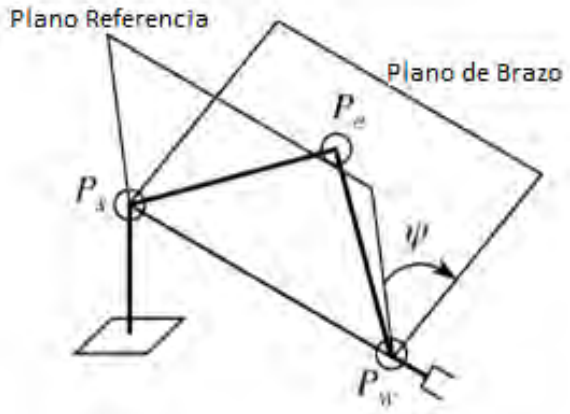


Figura 2.11. Definición de Planos y ángulos de brazo [24]

A partir de estas conclusiones el procedimiento para la resolución de la cinemática inversa descrita en [24] se resumen en los siguientes pasos:

- PASO 1: Dada la posición deseada $\boldsymbol{{}^{0}_{4}P}$ respecto a la base, se debe hallar la posición de la muñeca $\boldsymbol{w}$. Debido a que en el proyecto se está considerando con una mano estática se puede considerar que la posición deseada será la posición de la muñeca sumando un offset que viene a ser el tamaño de la mano. De la figura 2.12, nos podemos dar cuenta que al realizar una resta de vectores posición de hombro y muñeca vamos a obtener el vector $\boldsymbol{{}^{s}L_{w}}$ que coincide con el vector por el cual pasan todos los planos soluciones de la cinemática inversa.
- PASO 2: Existe una ecuación que relaciona el ángulo que gira un plano respecto a un vector. En algunas ocasiones se llama "Fórmula de Rodriguez". Este vector de giro debe ser el vector unitario de $\boldsymbol{{}^{s}L_{w}}$ descrito como $\boldsymbol{{}^{s}u_{w}}$ y el ángulo de giro del plano $\boldsymbol{\varphi}$ que ha sido definido como el "ángulo de brazo"

$$R_{\varphi} = I + (1 - \cos(\varphi))^{s}k_{w} + sin(\varphi)^{s}k_{w} \qquad (2.11)$$

$$^{0}R_{3} = R_{\varphi}\,^{0}R'_{3} \qquad (2.12)$$

Donde $\boldsymbol{{}^{s}k_{w}}$ denota a la matriz antisimétrica generado por el vector $\boldsymbol{{}^{s}u_{w}}$

$$^{s}u_{w} = \begin{vmatrix} 0 & -{}^{s}u_{w(z)} & {}^{s}u_{w(y)} \\ -{}^{s}u_{w(z)} & 0 & -{}^{s}u_{w(x)} \\ -{}^{s}u_{w(y)} & {}^{s}u_{w(x)} & 0 \end{vmatrix} \qquad (2.13)$$

Además, podemos ver que la matriz de transformación para cualquier solución la podemos obtener con la multiplicación de $\boldsymbol{R_{\varphi}}$ (fórmula de Rodríguez) y la

matriz ${}^{0}\boldsymbol{R'}_{3}$ que será la matriz de rotación para el plano referencial encontrada con los ángulos θ'_1, θ'_2 y θ'_3 determinados en un paso posterior.

Hay que notar que la matriz ${}^{0}\boldsymbol{R}_{3}$ ya la obtenemos en la cinemática directa en función de θ_1, θ_2 y θ_3 por lo que solo debemos realizar la comparación de la matriz ${}^{0}\boldsymbol{R}_{3(\theta_1,\theta_2,\theta_3)}$ y la matriz ${}^{0}\boldsymbol{R}_{3(\varphi)}$ encontrada con este procedimiento.

- PASO 3: En la Figura 2.12, podemos apreciar el resultado del ángulo 4 que no depende del ángulo de brazo (plano solución asignada) lo cual nos asegura que el ángulo 4 siempre será el mismo para cualquier solución de la cinemática inversa.

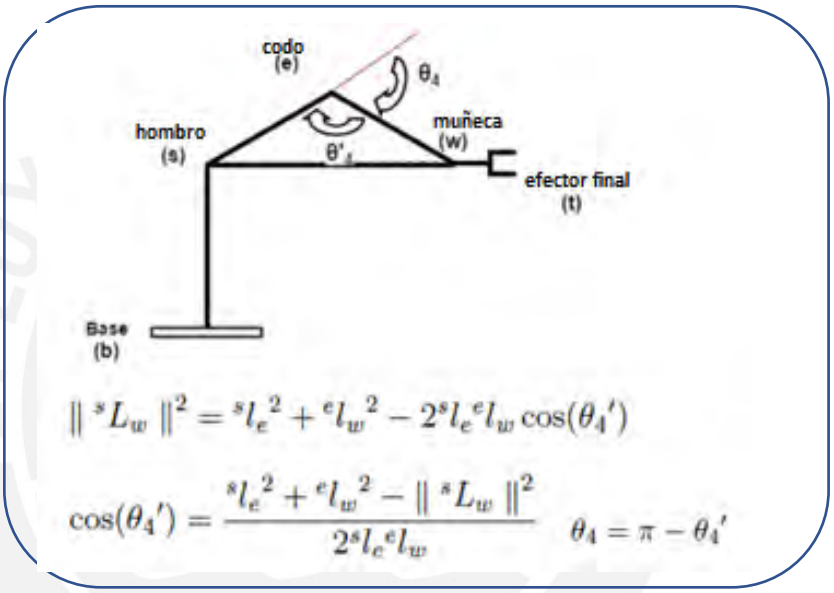


Figura 2.12. Análisis para solución del ángulo 4 del brazo [24]

- PASO 5: Aquí se debe asumir un ángulo θ'_3 para el plano referencia. En este caso y por la disposición de los brazos del proyecto se ha tomado $\theta_3 = 90°$. Ahora, para encontrar los ángulos θ'_1 y θ'_2 es necesario considerar la figura 2.13.

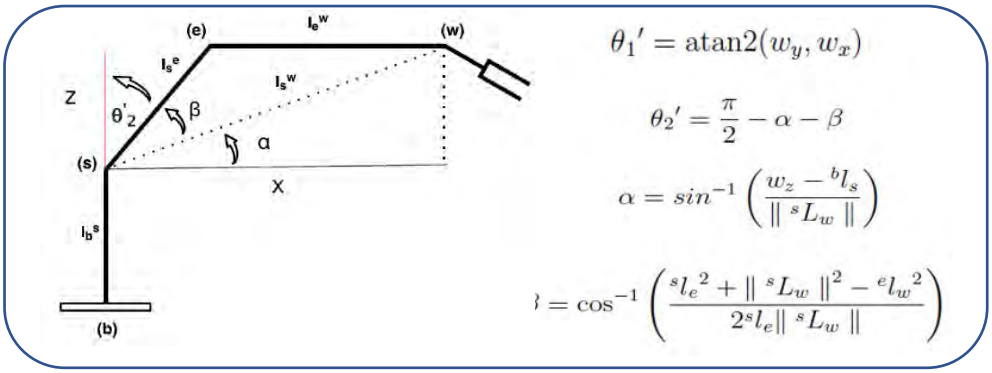


Figura 2.13. Análisis geométrico para los ángulos θ'_1, θ'_2 y θ'_3.[24]

De esta figura podemos ver que el ángulo θ'_1 es encontrado como la elevación que sufre el plano solución del brazo y el ángulo θ'_2 en hallado geométricamente encontrando el valor de los ángulos α y β.

- PASO 6: Con los ángulos θ'_1, θ'_2 y θ'_3 podemos obtener la matriz $^{0}\boldsymbol{R'_3}$ que al multiplicarla por la matriz $\boldsymbol{R_\varphi}$ nos dará la matriz $^{0}\boldsymbol{R_3}$ en función de **φ.**

  Esta matriz se debe igualar con la matriz $^{0}\boldsymbol{R_3}$ que se encuentra en función de los ángulos θ_1, θ_2 y θ_3 desde la cinemática directa y que deseamos encontrar.

  La comparación término a término de las matrices nos darán como resultado los ángulos θ_1, θ_2 y θ_3 en función de ángulo de brazo parametrizado.

$$\theta_1 = \tan^{-1}\left(\frac{-sin\varphi X_{s22} - cos\varphi Y_{s22} - Z_{s22}}{-sin\varphi X_{s12} - cos\varphi Y_{s12} - Z_{s12}}\right)$$

$$\theta_2 = \cos^{-1}(-sin\varphi X_{s32} - cos\varphi Y_{s32} - Z_{s32})$$

$$\theta_3 = \tan^{-1}\left(\frac{sin\varphi X_{s33} + cos\varphi Y_{s33} - Z_{s33}}{-sin\varphi X_{s31} - cos\varphi Y_{s31} - Z_{s31}}\right) \quad (2.14)$$

Donde:

$$^{0}R_3 = sin\varphi X_s + cos\varphi Y_s + Z_s$$

$$X_s = {}^{s}k_w{}^{0}R'_3 = \begin{vmatrix} X_{s11} & X_{s12} & X_{s13} \\ X_{s21} & X_{s22} & X_{s23} \\ X_{s31} & X_{s32} & X_{s33} \end{vmatrix}$$

$$Y_s = {}^{s}k_w{}^{0}R'_3 = \begin{vmatrix} Y_{s11} & Y_{s12} & Y_{s13} \\ Y_{s21} & Y_{s22} & Y_{s23} \\ Y_{s31} & Y_{s32} & Y_{s33} \end{vmatrix}$$

$$Z_s = {}^{s}k_w{}^{0}R'_3 = \begin{vmatrix} Z_{s11} & Z_{s12} & Z_{s13} \\ Z_{s21} & Z_{s22} & Z_{s23} \\ Z_{s31} & Z_{s32} & Z_{s33} \end{vmatrix} \quad (2.15)$$

De esta forma se ha encontrado una solución cerrada para la cinemática inversa de los brazos robóticos teniendo como parámetros de entrada la posición deseada **[x, y, z]** y el ángulo de brazo **φ.**

2.2.2. Orientación de la Palma de las manos de los brazos robóticos

Como se ha mencionado anteriormente, los GDL de brazo implementado solo nos ayudan al posicionamiento de la muñeca, pero algunos movimientos a realizar con el sistema, involucran que el robot muestre la palma de la mano o su reverso. Es así que en este caso hay una necesidad de poner la mano del robot al menos en 2 orientaciones (palma o su reverso).

Para la solución de esta problemática se ha recurrido a un algoritmo de cinemática inversa dual el cual se reconoce como dos análisis implementados para llegar a dos orientaciones definidas para una sola posición deseada.

De acuerdo al análisis dual, la orientación de la mano solo va depender de los motores 3 y 4, donde el motor 3 cambiará de signo en el ángulo y el motor 4 tomará ángulos suplementarios para los dos casos de orientación analizados. El motor 1 y 2 no tendrán variación en ángulos a alcanzar. En la figura 2.14 podemos ver la implementación de la cinemática inversa y el cómo se ha desarrollado para englobar a ambas orientaciones de la palma de la mano. Aquí también podemos observar que las posiciones angulares dependen de los datos de la posición cartesiana solicitada y el parámetro llamado "ángulo de brazo".

```
void InverseK(float *Xik, int32_t *Jik, float ang, uint8_t invert)
{
    float Jik_f[4];
    float x = Xik[0];
    float y = Xik[1];
    float z = Xik[2];

//Cálculo de el angulo q4 (angulo de codo)
Jik_f[3] = acos(( d3*d3 + a4*a4 - mL*mL)/(2*a4*d3));
if      (invert==0)         {Jik_f[3] = PI - Jik_f[3];}
else if (invert==1)         {Jik_f[3] = Jik_f[3] - PI;}

Jik_f[0] = atan2f((sinf(ang)*Xs[4]+cosf(ang)*Ys[4]+Zs[4]),(sinf(ang)*Xs[1]+cosf(ang)*Ys[1]+Zs[1]));
Jik_f[1] = asinf(-sinf(ang)*Xs[7] - cosf(ang)*Ys[7] - Zs[7]);

if      (invert==0)         {Jik_f[2] = atan2f((-sinf(ang)*Xs[6]-cosf(ang)*Ys[6]-Zs[6]),( -sinf(ang)*Xs[8]-cosf(ang)
else if (invert==1)         {Jik_f[2] = atan2f(( sinf(ang)*Xs[6]+cosf(ang)*Ys[6]+Zs[6]),(  sinf(ang)*Xs[8]+cosf(ang)
```

Figura 2.14. Implementación de la cinemática inversa para 2 orientaciones de la mano.

Como vemos los parámetros de entrada serán la posición deseada en XYZ, el ángulo de brazo **φ** y la solicitud de inversión de mano.

En la Figura 2.15 podemos ver las simulaciones para comprobar que el análisis de cinemática inversa es válido. En esta simulación se realiza el posicionamiento del brazo robótico para dos casos donde la posición deseada es la misma, pero con distintos ángulos de brazo.

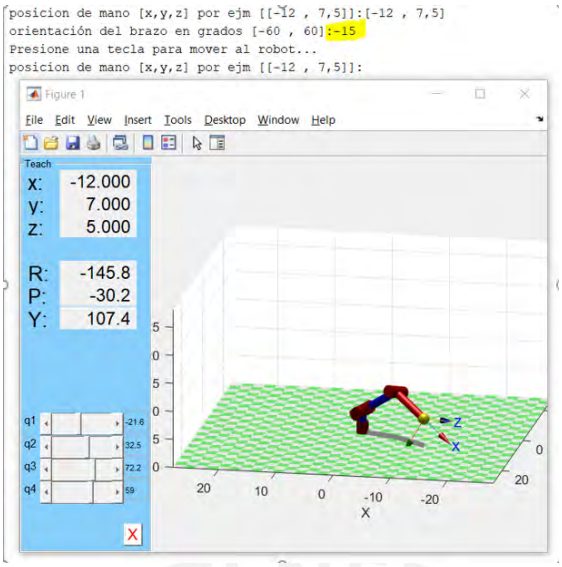


Figura 2.15. Simulación de cinemática inversa para un ángulo **φ.**

2.2.3. Cinemática Inversa de cabeza del robot de 2 GDL

Dado que se trata de 2 GDL, se puede realizar un análisis algebraico a partir de las ecuaciones de cinemática directa obtenidas en (2.10)

De estas ecuaciones ya es fácil resolver que:

$$\theta_2 = \cos^{-1}\left(\frac{z - d_1}{a_2}\right) \qquad \theta_1 = \tan^{-1}\left(\frac{y}{x}\right) \qquad (2.16)$$

Si vemos a las ecuaciones desde un punto geométrico, nos daremos cuenta que el efector final solo podrá moverse por todo el contorno de una esfera de radio a_2 y con centro en el punto $(0,0,d_1)$.

Esta última conclusión podría hacer que, encontrar un posicionamiento deseado no sea el correcto para el sistema. Es por esto que, para el caso de la cabeza, sería conveniente utilizar los 2 GDL libertad para orientar y no para posicionar.

En este caso, mencionando a los ángulos RPY [26]. Podemos encontrar las siguientes relaciones comparando la matriz de rotación RPY con la matriz de rotación como parte de la ecuación (2.9). Así obtenemos lo siguiente:

$$Roll = \theta_1 \qquad Pitch = \theta_2 - 90 \qquad Yaw = 0° \qquad (2.17)$$

En otras palabras, mencionando los ángulos de Euler tenemos libertad para cambiar el ángulo de azimut e inclinación de la cabeza.

2.3. Dinámica

La dinámica de un sistema mecánico relaciona las fuerzas que actúan en un cuerpo con el movimiento que se está originando en él. De esta manera podremos obtener un modelo matemático que involucre los siguientes puntos:

- Posición, velocidades y aceleración de cada vínculo en el robot.
- Las fuerzas y pares aplicados en cada articulación.
- Los parámetros del sistema como masas, inercias, longitudes.

Para sistemas que involucran hasta dos grados de libertad, será relativamente sencillo aplicar las leyes de newton o las ecuaciones de Lagrange para la rotación. En cambio, para sistemas de 3 o más GDL, ya no será muy intuitivo aplicar estas leyes por lo que en algunos casos se omite la obtención de un modelo matemático y se realiza directamente el análisis computacional. Sin embargo, la obtención del modelo nos da ciertas ventajas que podemos utilizar como, por ejemplo:

- Dimensionar los actuadores a utilizar.
- Simulación de los movimientos del robot.
- Diseño de la estructura mecánica del robot.
- Diseño del control dinámico del robot.

Este último punto sería el más importante, dado que, al generar un correcto modelo del sistema, seremos capaces de diseñar el controlador que nos permita cumplir con los requerimientos establecidos por la aplicación.

2.3.1. Modelo dinámico de brazos robóticos

Entendiendo la importancia del realizar el modelo del sistema, se han generado algunos procedimientos para aplicar las leyes físicas a sistemas de muchos grados de libertad considerando para eso algunas suposiciones como por ejemplo que el sistema sea un cuerpo rígido libre de deformaciones. Una forma alternativa de hallar el modelo matemático es mediante la formulación lagrangiana [26] que se explica a continuación.

Teniendo en cuenta la ecuación de Lagrange aplicada para cada articulación:

$$\frac{\partial y}{\partial x}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = \tau_i \;\; i = 1,2,\dots,n \qquad (2.18)$$

Dado que en este caso, esta ecuación involucra la velocidad, es necesario obtener una expresión para la velocidad en función de las transformadas homogéneas.

$$v_i^0 = v_i = \frac{d}{dt}\left(r_i^0\right) = \frac{d}{dt}\left(A_i^0 r_i^i\right) = \dot{A}_1^0 A_2^1 \dots A_i^{i-1} r_i^i + A_1^0 \dot{A}_2^1 \dots A_i^{i-1} r_i^i + A_1^0 \dots \dot{A}_i^{i-1} r_i^i + A_i^0 \dot{r}_i^i =$$

$$\left(\sum_{j=1}^{i} \frac{\partial A_i^0}{\partial q_j} \dot{q}_J\right) r_i^i \qquad (2.20)$$

En esta última expresión, debemos hallar la derivada de las transformadas homogéneas y dado que estas matrices son una relación entre senos y cosenos, nos daremos cuenta que esta derivada se consigue de manera simple multiplicando la matriz de transformación con una matriz Q_i.

$$Q_{i_r} = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad Q_{i_p} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad (2.21)$$

La primera matriz se aplica cuando se trata de una articulación de revolución mientras que la segunda matriz es aplicada cuando se trata de una articulación prismática. Ahora debemos aplicar el procedimiento aplicado en [27] y mostrado a continuación:

Paso 1: Se asigna un sistema de referencia a cada eslabón (paso 1 de D-H)

Paso 2: Obtener las matrices de transformación **A** para cada elemento *i*.

Paso 3: Obtener las matrices U_{ij}:

$$U_{ij} = \frac{\partial A_i^0}{\partial q_j} \qquad U_{ij} = \begin{cases} A_{j-1}^0 Q_j A_i^{j-1}, & j \le i \\ 0, & j > i \end{cases} \qquad (2.22)$$

Paso 4: Obtener las matrices U_{ijk}:

$$\frac{\partial U_{ij}}{\partial q_k} = U_{ijk} \begin{cases} A_{j-1}^0 Q_j A_{k-1}^{j-1} Q_k A_i^{k-1} & i \ge k \ge j \\ A_{k-1}^0 Q_k A_{j-1}^{k-1} Q_j A_i^{j-1} & i \ge j \ge k \\ 0 & i < j \ \text{ó} \ i < k \end{cases} \qquad (2.23)$$

Paso 5: Obtener las matrices de pseudo inercia.

$$J_i = r_i^i r_i^{i^T} dm = \begin{bmatrix} \int x_i^2 dm & \int x_i y_i dm & \int x_i z_i dm & \int x_i dm \\ \int x_i y_i dm & \int y_i^2 dm & \int y_i z_i dm & \int y_i dm \\ \int x_i z_i dm & \int y_i z_i dm & \int z_i^2 dm & \int z_i dm \\ \int x_i dm & \int y_i dm & \int z_i dm & \int dm \end{bmatrix} \quad (2.24)$$

Paso 6: Obtener la matriz de inercias $D = [d_{ij}]$, i,j=0,1,2…n / n=GDL

$$d_{ij} = \sum_{k=\max(i,j)}^{n} Traza(U_{kj} J_k U_{ki}^T \quad (2.25)$$

Paso 7: Se obtiene los términos h_{ikm}: i, k, m = 1, 2, … n.

$$h_{ikm} = \sum_{k=\max(i,k,m)}^{n} Traza(U_{jkm} J_j U_{ji}^T) \quad (2.26)$$

Paso 8: Se obtienen las matrices de fuerzas de Coriolis y Centrípeta $\boldsymbol{H} = [h_i]^T$

$$h_i = \sum_{k=1}^{n} \sum_{m=1}^{n} h_{ikm}\, \dot{q_k} \dot{q_m} \quad (2.27)$$

Paso 9: Se obtiene la matriz de Gravedad $\boldsymbol{C} = [c_i]^T$

$$c_i = \sum_{j=1}^{n} -m_j g U_{ji} r_j \quad (2.28)$$

Paso 10: Se obtiene la ecuación de movimiento

$$\tau_i = \sum_{k=1}^{n} D_{ik} \ddot{q_k} + \sum_{k=1}^{n} \sum_{m=1}^{n} h_{ikm} \dot{q_k} \dot{q_m} + c_i \quad i = 1,2,\dots,n \quad (2.29)$$

O en su forma matricial:

$$\tau_{(t)} = D_{q_{(t)}} \ddot{q_{(t)}} + h_{(q_{(t)}, \dot{q_{(t)}})} + c_{q_{(t)}} \quad (2.30)$$

El procedimiento mostrado anteriormente conlleva numerosos cálculos por lo que se recurre al software Matlab para procesarlo desde un script en offline de manera simbólica mostrado en el ANEXO A. De esta manera podremos obtener las

expresiones para los elementos de cada una de las matrices que intervienen en la ecuación Robótica.

La tarea de este script es obtener las matrices mostradas en la ecuación 2.31:

$$\boldsymbol{D} = \begin{vmatrix} d_{11} & d_{12} & d_{13} & d_{14} \\ d_{21} & d_{22} & d_{23} & d_{24} \\ d_{31} & d_{23} & d_{33} & d_{34} \\ d_{41} & d_{42} & d_{43} & d_{44} \end{vmatrix}$$

$$\boldsymbol{H} = \begin{vmatrix} h_{12} \\ h_{22} \\ h_{23} \\ h_{42} \end{vmatrix} \qquad \boldsymbol{C} = \begin{vmatrix} c_{12} \\ c_{22} \\ c_{23} \\ c_{42} \end{vmatrix} \qquad (2.31)$$

Estos elementos serán hallados en base a los ángulos $\theta_1, \theta_2, \theta_3, \theta_4$ y los parámetros del sistema como masas, centros de masas, tensores de inercia, distancias, etc. Esta información puede ser obtenida del modelo CAD inicial de los brazos robóticos y serán utilizados posteriormente en línea para el modelamiento del sistema en Matlab. También se está considerando un modelo ideal de barras metálicas para el brazo con pesos ideales a fin de comprobar que los resultados del modelo sean adecuados. Estos datos para el modelo ideal de brazo y el modelo implementado en CAD se muestran en las tablas 2.3 y 2.4.

Tabla 2.3. Parámetros de brazo ideal.

Datos de brazo ideal
Distancias: d1=0.1 m, d3=0.2 m, a4=0.2 m Gravedad: g=9.8 m/s2 masas: m1=0.2 kg, m2=0.1 kg, m3=0.1 kg, m4=0.1 kg centros de masa: rx1=0 , ry1=0.05, rz1=0 rx2=0 , ry2=0 , rz2=0.05 rx3=0 , ry3=0.05, rz3=0 rx4=0.05, ry4=0 , rz4=0 tensores de inercia: Ixx1=m1*ry1^2/3, Iyy1=0 , Izz1=m1*ry1^2/3 Ixy1=0 , Ixz1=0 , Iyz1=0 Ixx2=m2*rz2^2/3, Iyy2= m2*rz2^2/3 , Izz2=0 Ixy2=0 , Ixz2=0 , Iyz2=0; Ixx3=m3*ry3^2/3, Iyy3= 0 , Izz3= m3*ry3^2/3 Ixy3=0 , Ixz3=0 , Iyz3=0; Ixx4=0 , Iyy4= m4*(-ry4)^2/3, Izz4= m4*(-ry4)^2/3 Ixy4=0 , Ixz4=0 , Iyz4=0;

Tabla 2.4. Parámetros de brazo izquierdo en prototipo.

Datos de brazo izquierdo
Distancias:
d1=0.06462m, d3=0.2502m, a4=0.165 m
Gravedad:
g=9.8 m/s2
masas:
m1=0.83 kg, m2=0.442 kg, m3=0.542 kg, m4=0.0653 kg
centros de masa:
rx1=0.003418E-03 , ry1=3.8033E-03 , rz1=2.8773E-03
rx2=0.4234E-03 , ry2=-1.16037E-03 , rz2=72.7471E-03
rx3=-0.003182E-03 , ry3=79.8192E-03 , rz3=-0.82169E-03
rx4=-82.2175E-03 , ry4=0 , rz4=-0.957E-03
tensores de inercia:
Ixx1=0.000238593 , Iyy1=0.0001222 , Izz1=0.000254932
Ixy1=-8.11047E-09 , Ixz1=-2.41847E-09 , Iyz1=-9.12273E-06
Ixx2=0.000352458 , Iyy2= 0.000276223 , Izz2=0.000101278
Ixy2=-1.99745E-07 , Ixz2=-1.74948E-06 , Iyz2=4.4374E-05
Ixx3=0.000794265 , Iyy3= 0.000143176 , Izz3= 0.000805757
Ixy3=-1.0748E-08 , Ixz2=2.34356E-09 , Iyz3=-2.71813E-05
Ixx4=2.83506E-05 , Iyy4= 0.000231657 , Izz4=0.000245434
Ixy4=0 , Ixz4=-2.90289E-08 , Iyz4=0;

Estos brazos ideales con barras metálicas guardan mayor simetría respecto al sistema de referencia base por lo que se espera un comportamiento más intuitivo y fácil de comprender, lo que nos servirá como base para el análisis a desarrollar posteriormente con el modelo de Brazos en CAD.

Se debe tomar en cuenta que en la ecuación robótica es necesario considerar las fricciones que se están generando en el sistema mecánico, dado que, de no ser consideradas estas fricciones, obtendremos un sistema muy vibrante y hasta inestable para una mínima perturbación en el sistema.

Una buena estimación de estas fricciones es mediante el modelo de fricción de Coulomb (estática) con el modelo de fricción viscosa (dinámica) [29].

$$Df = sign(d\theta) * (Gain * abs(d\theta) + Offset) \quad (2.33)$$

Donde $\boldsymbol{d\theta}$ representa la matriz de las velocidades angulares, **Gain** y **Offset** son matrices con elementos que deben ser encontrados de manera experimental.

Esta matriz de fricciones complementará a la ecuación robótica de la siguiente manera:

$$\tau = D_{(q_{(t)})}\ddot{q_{(t)}} + h_{(q_{(t)},\dot{q_{(t)}})} + c_{(q_{(t)})} + Df \quad (2.34)$$

Para una mejor visualización de los resultados a obtener durante las simulaciones, se ha trabajado el modelado de las extremidades del robot en el software SIMULINK y las ecuaciones de movimiento en tiempo discreto se han trabajado a partir de bloque de funciones Matlab como se muestra en la figura 2.16.

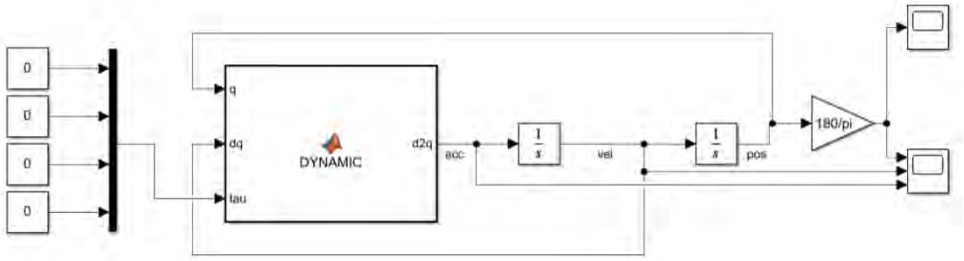


Figura 2.16. Modelado dinámico de brazo robótico en Simulink.

En este caso se desea mostrar la dinámica del cuerpo libre (DCL) donde al aplicar Torque igual a cero podamos visualizar cual es el comportamiento del brazo. La simulación de este caso la podemos visualizar en la Figura 2.17.

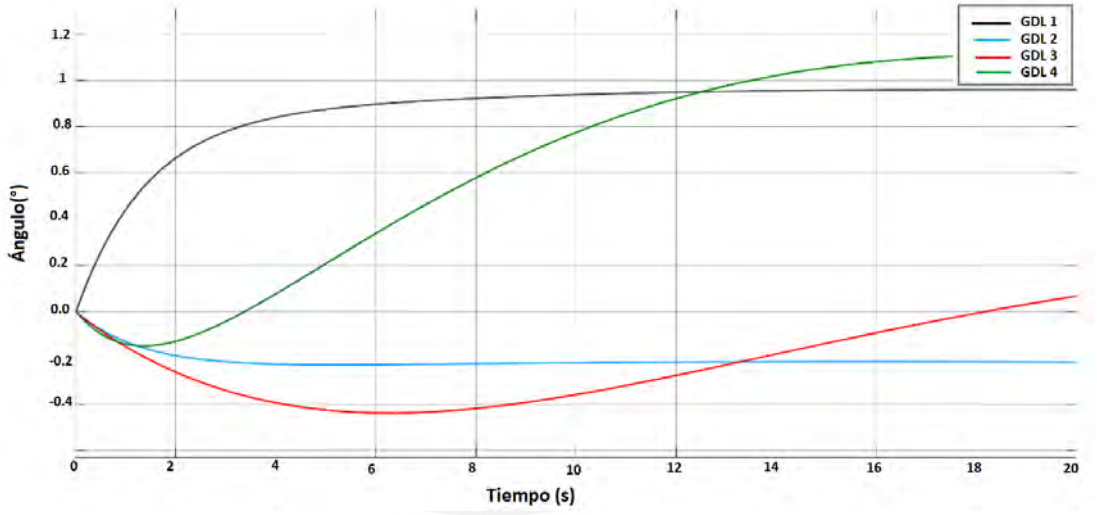


Figura 2.17. Posición ZERO de Brazo en DCL.

Se puede ver que al dejar al brazo libre y partiendo de posiciones angulares iguales a cero, el brazo hará mínimos movimientos y se estabilizará cada GDL en valores muy cercanos a 0° tal como se muestra en la figura anterior. Este último resultado es lógico dado que el brazo modelado tiene asimetrías que perturban ligeramente la posición ZERO (posición de reposo donde los ángulos son 0). En el caso de un modelo ideal de brazos con barras metálicas y completamente simétrico se logrará mantener la posición ZERO sin perturbaciones tal como se muestra en la figura 2.18 donde se realizó una simulación en MATLAB de este modelo ideal.

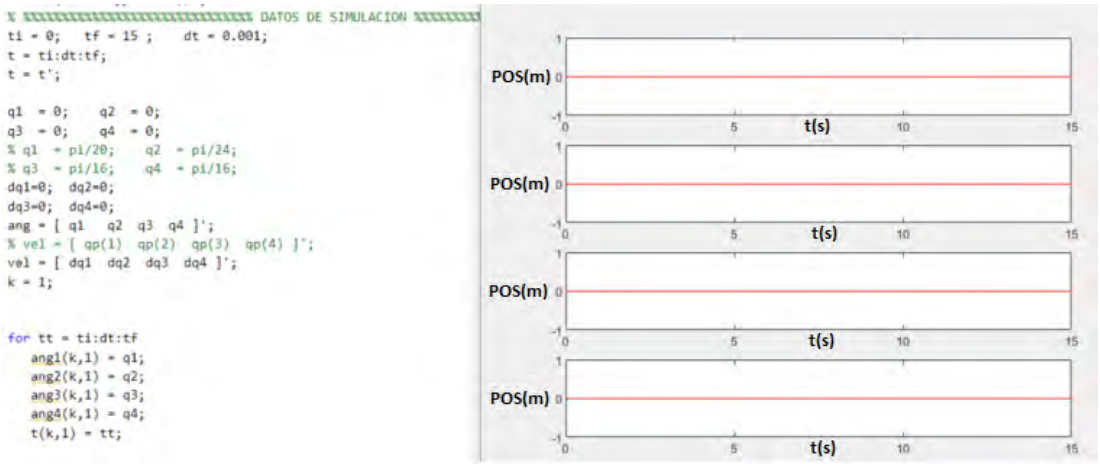


Figura 2.18. Simulación del brazo ideal para posición inicial 0 y Torque 0.

Se ha realizado la prueba de dinámica de cuerpo libre para una posición inicial diferente de 0 ([-60°,45°,30°, 60°]) para cada grado de libertad respectivamente obteniendo los resultados mostrados en la figura 2.19.

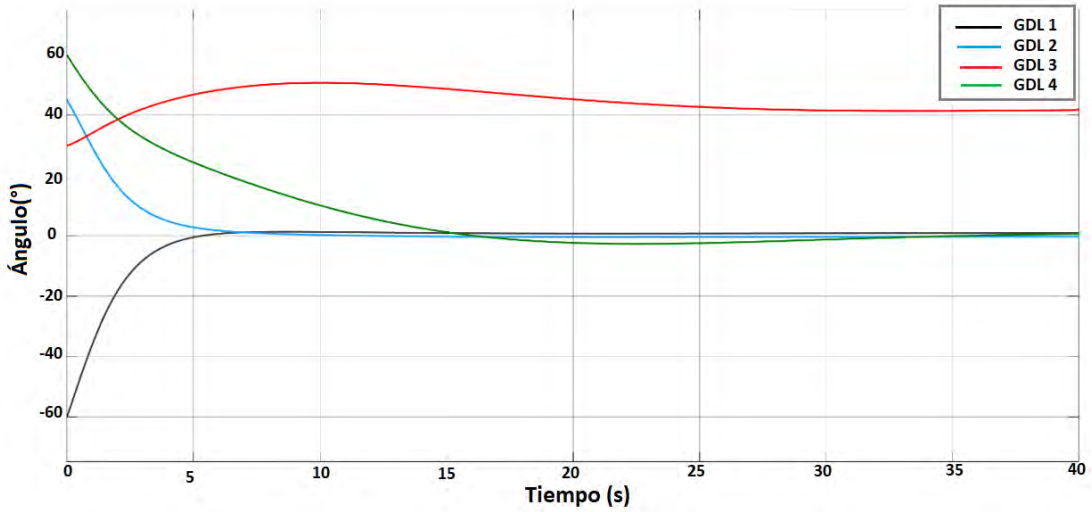


Figura 2.19. DCL para posiciones iniciales -60°,45°,30°, 60°.

Este es un resultado esperado ya que los grados de libertad 1, 2 y 4 se ven afectados directamente por la gravedad y en ese caso su posición de reposo debe ser un valor muy cercano 0° mientras que el movimiento del grado 3 es perpendicular a la gravedad y solo se estabilizará en un valor diferente de 0° por la acción de las fuerzas del modelo de fricciones supuestos anteriormente.

Estos resultados nos permiten asegurar, hasta cierto punto, que nuestro modelamiento simbólico realizado en MATLAB y SIMULINK para los brazos se hizo de manera correcta. Sin embargo, para esto, se ha supuesto valores para las ganancias y offset del

modelo de fricciones, tal como se muestra en la Tabla 2.5, los cuales deberán ser validados durante las pruebas experimentales con los brazos robóticos implementados.

Tabla 2.5. Tabla para modelo de fricción Viscosa y de Coulomb para Brazo.

Modelo de la fricción viscosa y Coulumb

off1 = 0;
off2 = 0;
off3 = 0;
off4 = 0;
 	
gain1 = 2;
gain2 = 2;
gain3 = 0.3;
gain4 = 0.2;
 	
offset = [off1; off2; off3; off4]
Gain = [gain1; gain2; gain3; gain4]
 	
$D_f = sign(dq).*(Gain.*abs(dq) + offset)$

Una última prueba del modelo dinámico obtenido se puede presentar en la figura 2.20 donde ahora se aplica un torque creciente al ángulo 2.

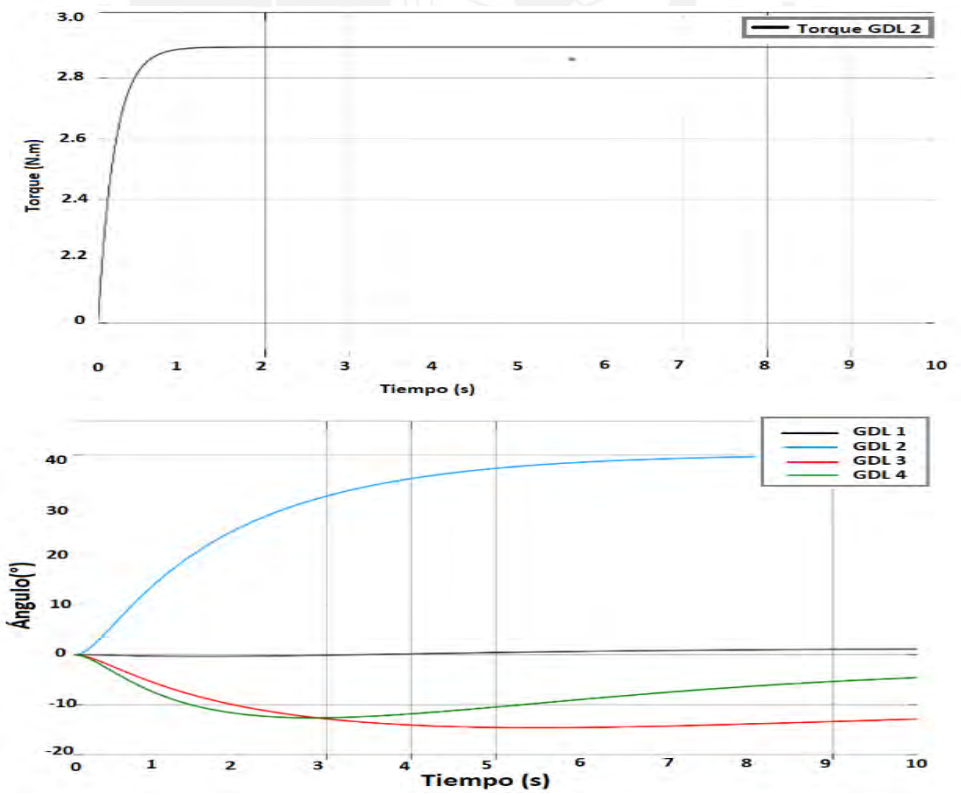


Figura 2.20. (a) Torque aplicado a GDL 2. (b) Respuesta de posiciones angulares.

Para el caso de este torque, vemos que es una entrada que se estabiliza en 0.9 y a la vez hace que el ángulo 2 se estabilice en un punto donde el torque y la fuerza de gravedad se equilibren. El ángulo 1, dado que es perpendicular al movimiento del ángulo 2, sufre cambios mínimos, mientras que los ángulos 3 y 4 si varían con este torque aplicado.

En conclusión, el modelo dinámico obtenido nos brinda resultados coherentes en las simulaciones de movimiento en cuerpo libre y con la aplicación de torques mínimos.

Este modelo debe ser corroborado experimentalmente con los brazos implementados, mejorando los coeficientes de fricción que más se adecuen a la respuesta transitoria del sistema.

2.3.2. Modelo dinámico de cabeza robot

Para el modelo dinámico de la cabeza es necesario el diseño CAD de donde se podrá calcular las masas, distancias y momentos de inercia asociados a cada grado de libertad. En la Figura 2.21 se muestra el CAD de este sistema con los respectivos grados de libertad.

Figura 2.21. Modelo CAD de cabeza robot con 2 grados de libertad.

Ahora, implementando el procedimiento explicado en [25] respecto al modelo dinámico de manipuladores robóticos mediante Euler-Lagrange se obtiene el modelo matemático de la cabeza robot que se muestran en las ecuaciones 2.35.

$$d_{11} = Ix_1 + \frac{Ix_2}{2} + \frac{Iy_2}{2} + Iz_1 + Iz_2 + \frac{a_2^2 m_2}{2} - \frac{Ix_2\cos(2q_2)}{2} + \frac{Iy_2\cos(2q_2)}{2} + Ixy_2\sin(2q_2) + a_2^2 m_2 rx_2 - \frac{a_2^2 m_2\cos(2q_2)}{2} - a_2 m_2 rx_2\cos(2q_2) + a_2 m_2 ry_2\sin(2q_2) + 1;$$

$$d_{12} = Iyz_2\sin(q_2) - Ixz_2\cos(q_2) - a_2 m_2 rz_2\cos(q_2) + 1;$$

$$d_{21} = Iyz_2\sin(q_2) - Ixz_2\cos(q_2) - a_2 m_2 rz_2\cos(q_2) + 1;$$

$$d_{22} = Ix_2 + Iy_2 + a_2^2 m_2 + 2a_2 m_2 rx_2 + 1;$$

$$h_1 = dq_2(2Ixy_2 dq_1 \cos(2q_2) + Ix_2 dq_1 \sin(2q_2) - Iy_2 dq_1 \sin(2q_2) + Iyz_2 dq_2 \cos(q_2) + Ixz_2 dq_2 \sin(q_2) + a_2^2 dq_1 m_2 \sin(2q_2) + 2a_2 dq_1 m_2 ry_2 \cos(2q_2) + 2a_2 dq_1 m_2 rx_2 \sin(2q_2) + a_2 dq_2 m_2 rz_2 \sin(q_2))$$

$$-2h_2 = dq_1^2(2Ixy_2 \cos(2q_2) + Ix_2 \sin(2q_2) - Iy_2 \sin(2q_2) + a_2^2 m_2 \sin(2q_2) + 2a_2 m_2 ry_2 \cos(2q_2) + 2a_2 m_2 rx_2 \sin(2q_2))$$

$$c_1 = 0;$$

$$c_2 = -gm_2(a_2 \sin(q_2) + ry_2 \cos(q_2) + rx_2 \sin(q_2)) \qquad (2.35)$$

Aplicando los datos numéricos obtenidos del CAD podemos obtener las simulaciones de diagrama de cuerpo libre necesarios para validar el modelo encontrado. El modelo realizado en Simulink lo encontramos en la Figura 2.22 muy similar al brazo robot, pero para 2 grados de libertad.

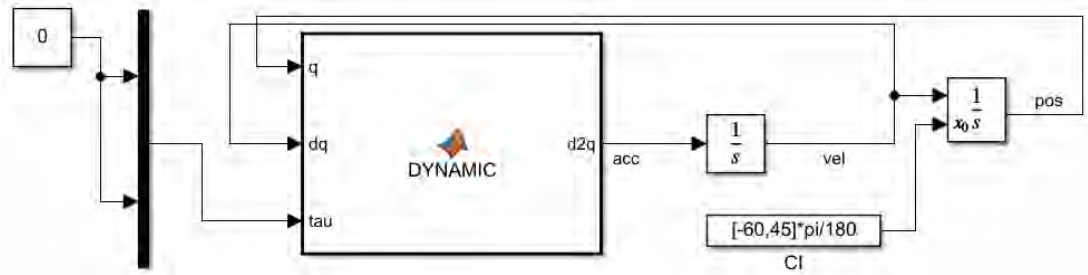


Figura 2.22. Simulación DCL para cabeza robot con CI de -60°, 45°.

Al realizar el análisis de cuerpo libre (torques ceros) se espera que el robot converja a su estado de menor energía, el cual, analizando podemos ver que el ángulo 2 deberá converger a un valor cercano a 180° mientras que el ángulo 1 realizará algunas oscilaciones debido al movimiento de la articulación 2 pero finalmente convergerá a un valor cercano al ángulo inicial (-60°) dado que no es afectado por la gravedad. Estos resultados los podemos ver reflejado en la simulación presentada en la figura 2.23.

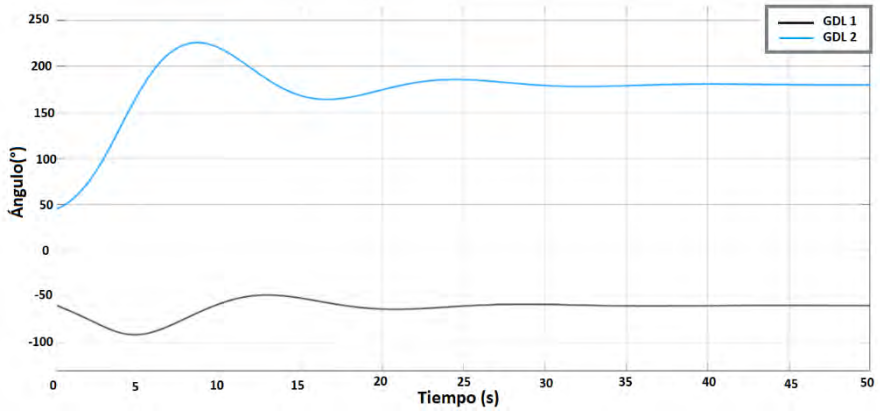


Figura 2.23. Movimientos libres de la cabeza robot para posición inicial [-60°, 45°].

De esta manera podemos validar el modelo matemático obtenido. Tener en cuenta que dentro del modelo dinámico existen coeficientes de fricción que por el momento se están asumiendo para brindarnos una respuesta promedio pero que tendrán que ser ajustados mediante experimentación posterior.

Como resultado del trabajo presentado en este capítulo, hemos obtenido modelos matemáticos que reflejan un comportamiento muy similar al prototipo de los brazos y cabeza del robot teleoperado. Es importante concluir que los sistemas mecánicos en su mayoría pueden ser abordados por criterios físicos de modelados para representar sus movimientos como es el caso de este trabajo teniendo como ventaja ahora el uso de un controlador basado en este modelo encontrado que se encargue de ejecutar los movimientos deseados a partir de un algoritmo de generación de trayectorias. Estos son temas que se van a abordar en el siguiente capítulo.

3. DISEÑO DEL SISTEMA DE CONTROL AVANZADO

Los sistemas de control para sistemas robóticos tienen un vasto campo de aplicación y esto se refleja en la proporción de publicaciones que se realizan para este tema con respecto a otros puntos relacionados con este campo.

En el capítulo anterior se ha realizado el modelo matemático considerando un análisis tanto cinemático y dinámico de las extremidades del robot. Tal y como hemos podido observar, los modelos dinámicos encontrados son sistemas no lineales por lo que la consideración de un sistema de control lineal no podría, hasta cierto punto, cumplir con los requerimientos solicitados. El uso de controladores no lineales serían los más apropiados para el control de los sistemas robóticos modelados para este proyecto. La intención de este capítulo es estudiar algunas propuestas de control no lineal y diseñar el controlador que mejor se adapte a los movimientos que se requieren para este proyecto.

Con el diseño del sistema de control, se debatirá la mejor forma de planear y generar las trayectorias de las extremidades del Robot, mediante algoritmos que aseguren o cumplan con los requerimientos de movimientos naturales para los Brazos y cabeza del robot.

3.1.Control en lazo cerrado

Muchos de los controladores utilizados para sistemas mecánicos utilizan la información de su modelo matemático, es por eso, la importancia de hallar previamente dicho modelo antes de plantear una estrategia de control. Vamos a tomar como modelo la ecuación diferencial robótica planteada en el capítulo anterior.

$$\boldsymbol{\tau = D(q)\ddot{q} + h(q,\dot{q}) + C(q)} \qquad \boldsymbol{(3.1)}$$

Cuando requerimos llevar a las articulaciones a una cierta posición, velocidad y aceleración $(\boldsymbol{q_d, \dot{q}_d, \ddot{q}_d})$, se puede pensar que dichas trayectorias deben cumplir en todo momento con la ecuación (3.1) con lo que al implementar nuestro sistema robótico solo bastaría con aplicar el torque del modelo de la planta en cada instante de tiempo para generar el torque necesario y cuando esto es realizado, se dice que estamos realizando un control sin realimentación, debido a que se aplica un torque estimado por el modelo, pero el control no verifica si se están cumpliendo las condiciones deseadas. Lo negativo de realizar este control no realimentado, es que muchas veces,

el torque calculado por el modelo no tiene una buena estimación dado que este tiene imperfecciones respecto al sistema real o por otro lado se pueden estar generando perturbaciones que no están siendo consideradas en el modelo lo que provoca que la condición deseada está muy lejos del punto real de operación. Es por eso que en estos casos se utiliza un sistema como el mostrado en la figura 3.1, donde se hace uso de retroalimentación que comunican en todo momento, el punto de operación, al sistema de control y esto hace que se tomen decisiones basándose en esta información actualizada.

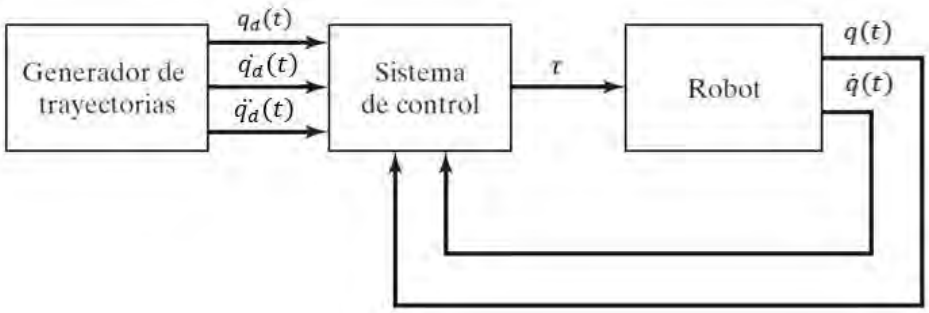


Figura 3.1. Sistema de Control en Lazo cerrado.

En este apartado nos corresponde evaluar el mejor control a considerar para poder cumplir con las condiciones deseadas que nos solicita un generador de trayectorias previo.

3.1.1. Controlador PID

El control PID es un controlador lineal, y aunque se ha mencionado que es requerido un controlador no lineal, es difícil no proponer un control PID como primera opción de control debido a que la sintonización de sus parámetros no requiere la información del modelo de la planta y solo se basa en el conocimiento experimental que se tenga del sistema. Este controlador podrá funcionar siempre y cuando se realicen algunas aproximaciones al sistema a controlar y utilizará como información de entrada el error de posicionamiento y velocidades.

Una buena forma de aproximar el sistema no lineal, como las extremidades del robot, para fines de poder utilizar el control PID, es poder realizar el control de cada grado de libertad de manera independiente, lo que en práctica sabemos que no es posible realizar dado que los grados de libertad están todos acoplados entre sí.

La ley de control PID a utilizar para cada posicionamiento angular es la siguiente:

$$\tau = K_p e_{(t)} + Ki \int_0^t e_{(t)} + K_d \frac{de_{(t)}}{dt} \qquad (3.2)$$

Donde $\boldsymbol{e_{(t)} = q_d - q}$. Si se quiere considerar la ecuación (3.2) para el conjunto completo de un sistema robótico, se deben manejar a $K_p, K_i \, y \, K_d$ como matrices diagonales que contienen los parámetros de sintonización para cada posicionamiento angular.

En algunos sistemas robóticos no se considera el termino integral dado que este a pesar de cancelar el error en estado estable, tiende a hacer a un sistema robot más lento para las aplicaciones a las cuales ha sido destinado.

Como se puede deducir de la ecuación (3.2) y también, como ya se ha comentado, no es necesario conocer el modelo matemático del sistema, solo se requiere afinar las constantes a unos valores que mejoren la respuesta y nos lleven a un punto deseado.

En algunos casos, cuando el desacoplamiento de las posiciones angulares no es posible, ya es necesario realizar alteraciones a la ley de control PID para que pueda aún permitir un control aceptable.

Vamos a considerar un control Proporcional Derivativo PD, Como primer controlador a proponer dada su práctica implementación. El lazo cerrado de control para este controlador en SIMULINK los podremos ver en la figura 3.2

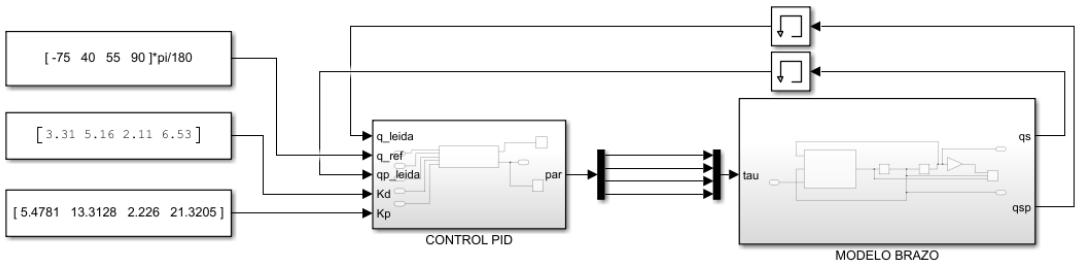


Figura 3.2. Lazo de control PD para Brazo Robot.

Para esta simulación se ha utilizado la estructura paralela del control PD y se ha llevado a una ecuación recursiva dentro de un bloque MATLAB FUNCTION que nos permita poder realizar el algoritmo para cada grado de libertad en forma matricial. Posteriormente se ha llevado este control a un bloque SUBSISTEMA que permita solo presentar las entradas y salidas necesarias, incluidos los valores de sintonización de cada controlador. Así, el controlador pide como entradas la posiciones y velocidades

leídas, las cuales se van a comparar con la posiciones y velocidades deseadas para hallar los errores respectivos a utilizar en la ecuación (3.2).

La ganancia proporcional y derivativa mostradas en matrices diagonales han sido hallados en base a pruebas realizadas siguiendo el método de sintonización de ganancia limite. Este método ha sido utilizado sintonizando cada GDL de manera independiente comenzando por el GDL 4 y manteniendo en 0 los demás torques. De esta manera se fue encontrando los parámetros de cada controlador PD mostrado en la Figura 3.2. El punto de operación deseado para las posiciones angulares es [-75°,40°,55°,90°] con el cual se obtuvo los resultados mostrados en la figura 3.3.

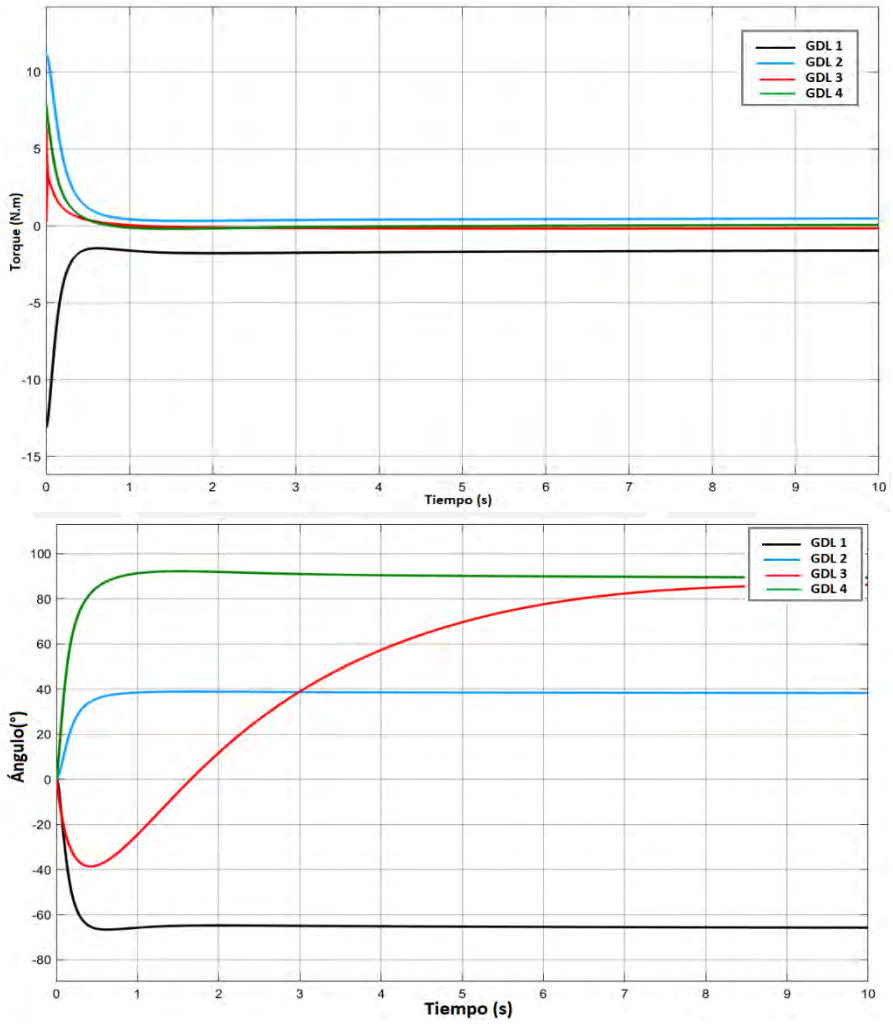


Figura 3.3. Grafica de Toques y posiciones con control PD.

En esta figura podemos observar una convergencia al punto [-65°,39°,65°,73°] donde ciertamente se puede ver un error en estado estacionario comparado a los valores deseados. En este caso, se podría usar una parte integral en el controlador para anular

el error en estado estacionario, pero esto provocará que el sistema se comporte de manera más lenta, siendo un percance para la operación solicitada.

De esta manera, utilizar un controlador lineal PD no sería lo más óptimo, sin embargo, se pueden realizar algunas adecuaciones, considerando, en parte, algunas no linealidades del sistema, para mejorar la performance alcanzada con este controlador PD lo cual se verá a continuación.

3.1.2. Controlador PD con compensación de Gravedad

Se sabe que los controladores PD son bastante simples de implementar en diversos sistemas ya que no requieren un conocimiento previo del sistema. En el caso de sistemas robóticos, como se mencionó, aplicar un integrador que nos permita asegurar el error 0 en estado estacionario, podría causar que el control limite el sistema a movimientos más lentos de lo que se puedan requerir. Para no depender de un integrador, una solución es utilizar un compensador de gravedad que proporcione al sistema este sesgo variable en cada movimiento para obtener error estacionario cero. En [30] se demuestra la convergencia de este controlador si aplicamos la siguiente ley de control:

$$\tau = C_{(q)} + K_d \dot{e_p} + K_p e \qquad (3.3)$$

Donde K_p y $K_d > 0 \in \mathbb{R}^{nxn}$ son matrices diagonales de ganancias proporcionales y derivativas respectivamente. Además, el error de posición $\tilde{q}$ y la posición deseada q_d está definido de la siguiente manera:

$$\tilde{q} = q_d - q \quad y \quad \dot{q_d} = 0 \qquad (3.4)$$

Así la expresión 3.3 se puede reescribir de la siguiente manera:

$$\tau = K_p \tilde{q} - K_d \dot{q} + C_{(q)} \qquad (3.5)$$

Ahora reemplazando la expresión (3.5) en la ecuación robótica (3.1) obtenemos el siguiente sistema:

$$D_{(q_{(t)})} \ddot{q_{(t)}} + h_{(q_{(t)}, \dot{q_{(t)}})} = K_p \tilde{q} - K_d \dot{q} \qquad (3.6)$$

Esta última expresión puede ser representada por el siguiente sistema matricial:

$$\frac{d}{dt}\begin{bmatrix}\tilde{q} \\ \dot{q}\end{bmatrix} = \begin{bmatrix}-\dot{q} \\ D^{-1}_{(q_{(t)})}(K_p \tilde{q} - K_d \dot{q} - h_{(q_{(t)}, \dot{q_{(t)}})})\end{bmatrix} \qquad (3.7)$$

Considerando la siguiente expresión como una función de Lyapunov:

$$V(\tilde{q},\dot{q}) = \frac{1}{2}\dot{q}^T D_{(q_{(t)})}\dot{q} + \frac{1}{2}\tilde{q}^T K_p \tilde{q} \qquad (3.8)$$

Y su derivada temporal:

$$\dot{V}(\tilde{q},\dot{q}) = \dot{q}^T D_{(q_{(t)})}\ddot{q} + \frac{1}{2}\dot{q}^T \dot{D}_{(q_{(t)})}\dot{q} + \tilde{q}^T K_p \dot{\tilde{q}} \qquad (3.9)$$

Y teniendo en cuenta las propiedades de las matrices de la ecuación robótica [31] se puede llegar al siguiente resultado:

$$\dot{V}(\tilde{q},\dot{q}) = -\dot{q}^T K_d \dot{q} \leq 0 \qquad (3.10)$$

Con lo cual se comprueba que el origen es estable y utilizando el principio de invariancia de Lasalle [28] se puede demostrar la estabilidad asintótica Global utilizando la siguiente función omega:

$$\Omega = \left\{[\tilde{q}^T, \dot{q}^T] \in \mathbb{R}^{2n} : \dot{V}(\tilde{q},\dot{q}) = 0\right\} \qquad (3.11)$$

Dado que para **Ω** se cumple que $\dot{V}(\tilde{q},\dot{q}) = 0$ sí y solo si $\dot{q} = 0$ entonces $\ddot{q} = 0$. De (3.7) nos podremos dar cuenta que:

$$0 = D_{(q_{(t)})} K_p \tilde{q} \qquad (3.12)$$

Por lo que $\tilde{q} = 0$, lo que asegura que $\dot{V}(\tilde{q},\dot{q}) < 0$ para todo $[\tilde{q}^T, \dot{q}^T]^T \neq 0$ y de esta manera el sistema es global asintóticamente estable.

De este mismo procedimiento, en [28] nos muestran algunos métodos para la sintonización de este controlador. El método 3 y 4 se basan por ejemplo en la obtención de la matriz de gravedad de modelo matemático, a partir del cual obtendremos los autovalores máximos de su matriz gradiente. Para una primera aproximación de los parámetros de sintonización se utilizará el método 2 que se basa en el procedimiento mostrado para demostrar su estabilidad y que nos permite elegir las ganancias derivativas con la condición que sean mayor o igual a 2 para no obtener respuestas muy sobreamortiguadas y a partir de cual podremos obtener las ganancias proporcionales aplicando la siguiente expresión:

$$k_{p_i} = \frac{k_{d_i}^2}{2} \quad i = 1, 2, \ldots n. \qquad (3.13)$$

De esta manera se han elegido los siguientes valores de ganancias:

$$K_p = diag\{5.4781 \quad 13.3128 \quad 2.226 \quad 21.3205\}$$

$$K_d = diag\{3.31 \quad 5.16 \quad 2.11 \quad 6.53\} \qquad (3.14)$$

La implementación en SIMULINK de este control lo podemos ver en la siguiente figura 3.4 siendo el controlador una función de MATLAB también mostrado en esta figura.

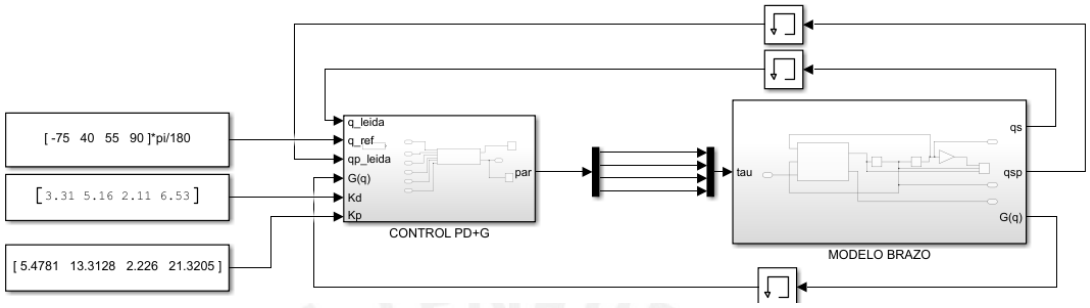


Figura 3.4. Lazo de control con compensador de gravedad.

Podemos notar que además de la posición y velocidad, el control requiere de la matriz de gravedad de modelo en cada instante de tiempo el cual se ha enviado con un valor inicial a través del bloque "Memory".

Las pruebas se realizaron en las mismas condiciones en el anterior controlador lineal PD haciendo una comparación de los resultados obtenidos, los cuales se muestran en la figura 3.5.

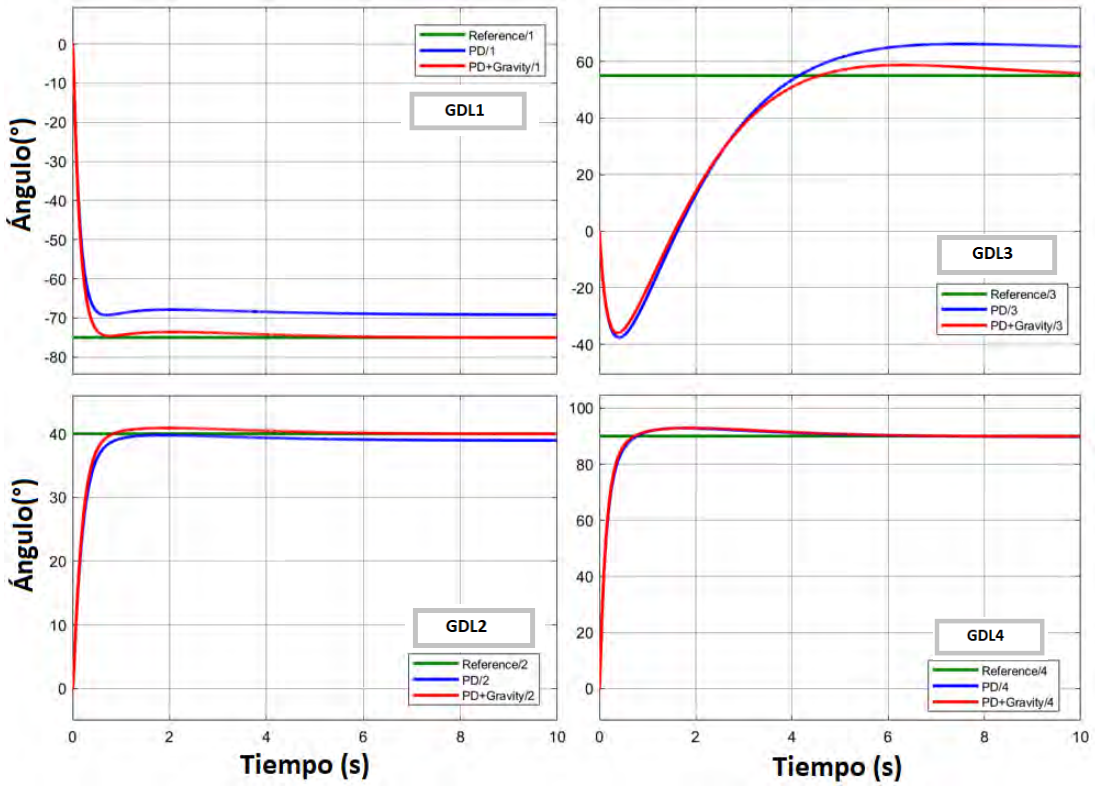


Figura 3.5. Respuesta de Control PD (azul) y PD-Gravedad (Rojo).

Se puede observar de la comparación realizada, que el control PD no lineal compensa los errores en estado estacionario del primer controlador por lo que se convierte en un controlador más preciso. En términos de velocidad de respuesta, podemos notar una ligera mejora para el caso del PD con compensador. Una observación adicional de los controladores vistos hasta el momento, es que se posicionan en un tiempo prudente a los ángulos 1, 2 y 4 pero en el caso del ángulo 3 no es posible generar un control que reduzca el tiempo de estabilización de esta señal por lo que se evaluará algún controlador no lineal adicional.

Así, hasta el momento se ha desarrollado el control PD no lineal como primera alternativa de control utilizando para estos experimentos el modelo matemático obtenido anteriormente. Se podrá observar posteriormente que el control PD con compensación de gravedad será un paso previo para el diseño de controlador final para obtener algunos parámetros del modelo que hasta el momento no se han determinado.

3.1.3. Control No Lineal BACKSTEPPING

Es un controlador basado en el modelo matemático del sistema mostrado en la ecuación (3.1) por lo que la eficiencia de este controlador radica principalmente en la elección de un modelo del sistema muy acertado. Este controlador, al igual que otros controladores no lineales, basa su diseño en los criterios de estabilidad y convergencia del sistema en lazo cerrado a partir del método de Lyapunov. El procedimiento para encontrar la ley de control inicia desde la remodelación de la ecuación robótica (3.1) a un sistema en espacio estado de forma vectorial de la manera siguiente:

$$x_1 = q$$

$$x_2 = \dot{x_1} = \dot{q}$$

$$\dot{x_2} = \ddot{q} = D_{(q_{(t)})}{}^{-1}\left(\tau - h_{(q_{(t)},\dot{q_{(t)}})} - c_{(q_{(t)})} - D_f\right) \quad (3.15)$$

Por lo que las ecuaciones de espacio estado serían:

$$\dot{x_1} = x_2$$

$$\dot{x_2} = D_{(q_{(t)})}{}^{-1}\left(\tau - h_{(q_{(t)},\dot{q_{(t)}})} - c_{(q_{(t)})} - D_f\right) = w \quad (3.16)$$

Notar que se ha utilizado la variable w para representar toda la expresión de $\dot{x_2}$ y considerando la variable de control virtual $x_2 = v$ obtenemos que:

$$\dot{x_1} = v \qquad (3.17)$$

Para este sistema se considera la primera función de Lyapunov y su derivada siguiente:

$$V_1 = \frac{1}{2}x_1^2 \; ; \; V_1(0) = 0; V_1(x) > 0 \; x \neq 0; \qquad \dot{V_1} = x_1\dot{x_1} \qquad (3.18)$$

Para demostrar que $\dot{V_1} < 0$ debemos tomar $v = -K_1x_1$ donde $K_1 > 0$ y reemplazando obtenemos:

$$\dot{V_1} = -K_1x_1^2 \qquad K_1 > 0 \qquad (3.19)$$

Ahora debemos demostrar la estabilidad del sistema que asegure que nuestra variable virtual cumpla con lo requerido. Este nuevo sistema viene dado por:

$$z = x_2 - v \qquad \lim_{t\to\infty}(z) = 0 \qquad (3.20)$$

De la ecuación (3.21) podemos obtener los siguiente:

$$z = x_2 + K_1x_1 \quad => \quad x_2 = \dot{x_1} = z - K_1x_1 \qquad (3.21)$$

Al derivar la ecuación (3.21) y dado que del sistema original de la ecuación (3.16) sabemos que $\dot{x_1} = x_2$ obtenemos que:

$$\dot{z} = \dot{x_2} + K_1\dot{x_1} \quad => \quad \dot{z} = w + K_1x_2 \qquad (3.22)$$

Al reemplazar la ecuación (3.21) en (3.22) vamos a obtener la expresión final del sistema a analizar:

$$\dot{z} = w + K_1(z - K_1x_1) \qquad (3.23)$$

Para lograr la estabilidad de este sistema vamos a considerar la siguiente función de Lyapunov y su derivada:

$$V = V_1 + \frac{1}{2}z^2 = \frac{1}{2}x_1^2 + \frac{1}{2}z^2; V(x) > 0 \; x \neq 0; \qquad \dot{V} = z\dot{z} + x_1\dot{x_1} \qquad (3.24)$$

Para demostrar que $\dot{V} < 0$ vamos a reemplazar los resultados de las ecuaciones (3.22) y (3.23) en (3.24) obteniendo:

$$\dot{V} = x_1(z - K_1x_1) + z\big(w + K_1(z - K_1x_1)\big) \qquad (3.25)$$

Al reordenar la última expresión vamos a obtener lo siguiente:

$$\dot{V} = -K_1x_1^2 + z(w + K_1(z - K_1x_1) + x_1) \qquad (3.26)$$

Podemos darnos cuenta que para obtener que $\dot{V} < 0$ solo debemos realizar lo siguiente:

$$w + K_1(z - K_1x_1) + x_1 = -K_2z \qquad K_2 > 0 \qquad (3.27)$$

De esta manera obtenemos:

$$\dot{V} = -K_1x_1^2 - K_2z^2 \qquad (3.28)$$

Así, este último resultado nos permite demostrar que $\dot{V} < 0$ y obtener que nuestro sistema será global asintóticamente estable.

De la ecuación (3.26) reemplazando $\boldsymbol{w}$ por su expresión original dada en la ecuación (3.16) y resolviendo:

$$w + K_1(z - K_1x_1) + x_1 = -K_2z$$

$$D_{(q_{(t)})}{}^{-1}\left(\tau - h_{(q_{(t)},\dot{q_{(t)}})} - c_{(q_{(t)})} - Df\right) + K_1(z - K_1x_1) + x_1 = -K_2z$$

$$D_{(q_{(t)})}{}^{-1}\left(\tau - h_{(q_{(t)},\dot{q_{(t)}})} - c_{(q_{(t)})} - Df\right) = -(K_1 + K_2)z + (K_1^2 - 1)x_1 \qquad (3.29)$$

De la ecuación (3.21) vamos a reemplazar z y reordenamos la ecuación anterior:

$$D_{(q_{(t)})}{}^{-1}\left(\tau - h_{(q_{(t)},\dot{q_{(t)}})} - c_{(q_{(t)})} - Df\right) = -(K_1 + K_2)(x_2 + K_1x_1) + (K_1^2 - 1)x_1$$

$$D_{(q_{(t)})}{}^{-1}\left(\tau - h_{(q_{(t)},\dot{q_{(t)}})} - c_{(q_{(t)})} - Df\right) = -(K_1K_2 + 1)x_1 - (K_1 + K_2)x_2 \qquad (3.30)$$

Ahora vamos a regresar a las variables iniciales $x_1 = q; x_2 = \dot{q}$ y despejamos τ de la última expresión:

$$\tau = h_{(q_{(t)},\dot{q_{(t)}})} + c_{(q_{(t)})} + Df - D_{(q_{(t)})}(K_1K_2 + 1)q - D_{(q_{(t)})}(K_1 + K_2)\dot{q} \qquad (3.31)$$

Dado que se ha demostrado que el sistema es global asintóticamente estable podemos obtener, para un $q^*, \dot{q}^*$ deseados, la expresión final de control.

$$\boldsymbol{\tau = h_{(q_{(t)},\dot{q_{(t)}})} + c_{(q_{(t)})} + Df - D_{(q_{(t)})}(K_1K_2 + 1)(q - q^*) - D_{(q_{(t)})}(K_1 + K_2)(\dot{q} - \dot{q}^*)} \qquad \mathbf{(3.32)}$$

Así se obtiene la expresión final del control Backstepping a aplicar al sistema. Se observa que esta expresión depende fuertemente del conocimiento de toda la expresión asignada al modelo matemático de sistema robótico. Además, la única condición para las ganancias K_1, K_2 es que ambas sean matrices positivas por lo que inicialmente hemos considerado las ganancias siguientes:

$$K_1 = diag\{10 \quad 10 \quad 10 \quad 10\}$$

$$K_2 = diag\{10 \quad 10 \quad 5 \quad 5\} \qquad (3.33)$$

Eventualmente, durante las pruebas, podremos observar que a medida que aumentemos los valores de las ganancias, seguiremos obteniendo un sistema estable

más rápido pero el τ necesario será mayor así que una limitante, durante la implementación, para estas ganancias será la capacidad máxima de cada actuador a utilizar.

Además, también se puede notar que la ley de control no hace uso de la inversa de una matriz, lo cual va a favorecer a la implementación del control porque por lo general, las inversas de matrices, conllevan un grado computacional alto en la implementación.

Teniendo entonces la ley de control a utilizar, vamos a plantear el lazo de control tal como se muestra en la figura 3.6 en la cual se confirma el uso de la medición de la posición y la velocidad en todo momento como entradas de controlador, así como también la posición y velocidad deseada.

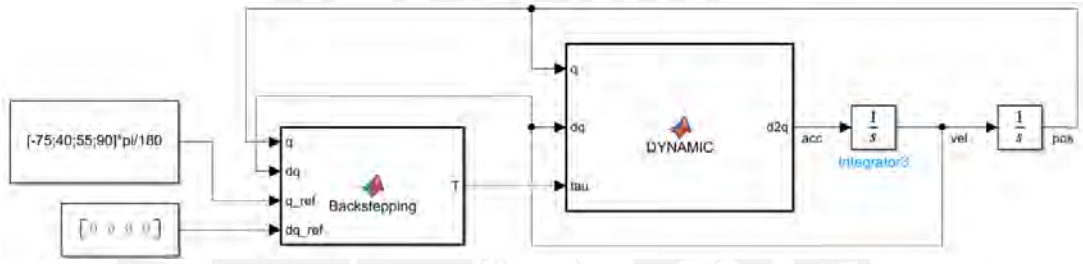


Figura 3.6. Lazo de control BACKSTEPPING para brazos robóticos.

Es importante resaltar que a diferencia de los controladores PD inicialmente propuestos, las ganancias del controlador Backstepping han sido escogidas aleatoriamente iniciando con valores bajos, además, mientras mayor sean estas ganancias, mayor será el torque inicial por lo que nuestro sistema puede tener mayor velocidad de respuesta en tanto lo permitan los servo-actuadores utilizados para el movimiento de las articulaciones. Para nuestro caso hemos limitado el torque máximo a 30 N-m. (valor promedio para motores a utilizar) con lo cual podremos obtener las respuestas mostradas en la figura 3.7 donde se realiza una comparación con las respuestas del controlador PD con compensación de gravedad.

Es importante notar que, aunque el controlador PD con compensación de gravedad realice cambios a mayor velocidad, es el controlador Backstepping que se estabiliza y produce un error cero de manera más rápida. Lo más importante en todo caso es que este controlador no lineal es capaz de llevar una correcta respuesta rápidamente para todas las articulaciones, lo que no pasaba con el anterior controlador donde se tenían dificultades para la articulación 3.

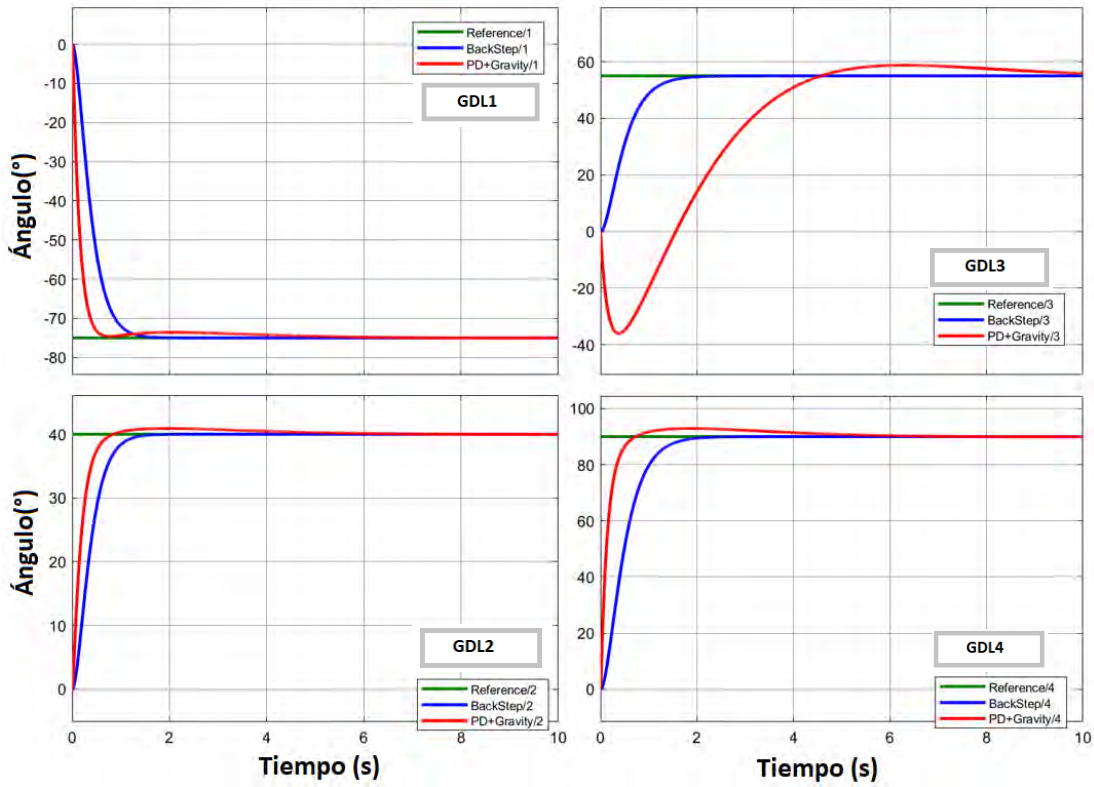


Figura 3.7. Comparativa de respuesta de controladores.

3.1.4. Control Sliding Mode

El principal objetivo de este controlador no lineal es llevar al sistema a un punto de operación deseado a partir de la acción de una función polinomial. Cuando el punto de operación es alcanzado, el controlador inicia una serie de oscilaciones en la variable de control para mantener el punto de operación. Pero estas oscilaciones, como cambios discontinuos denominadas "chattering", son perjudiciales para el actuador, por lo que existen varios métodos para evitar este fenómeno durante la etapa de control.

El detalle, como en otros controladores no lineales, es hallar la ley de control a utilizar. En [32] y [33] se muestra una manera de llegar a esta ley de control a partir del análisis de un sistema de orden "n". Para nuestro caso utilizamos este análisis para su aplicación en un sistema de segundo orden que viene a ser el modelo matemático encontrado. Este sistema de segundo orden se puede representar por la siguiente expresión:

$$\ddot{x} = f(x,t) + u(t) \qquad (3.33)$$

Y teniendo en cuenta que el error se define de la siguiente manera:

$$e = x_{set} - x \qquad (3.34)$$

Donde x_{set} es el punto de operación deseado, se define la siguiente región deslizante y su derivada:

$$s = \dot{e} + \lambda e; \qquad \dot{s} = \ddot{e} + \lambda \dot{e} \qquad (3.35)$$

Reemplazando la ecuación (3.34) en la última expresión obtenemos:

$$\dot{s} = \ddot{x}_{set} - \ddot{x} + \lambda(\dot{x}_{set} - \dot{x})$$

$$\dot{s} = \ddot{x}_{set} - f(x,t) - u(t) + \lambda(\dot{x}_{set} - \dot{x}) \qquad (3.36)$$

Para demostrar la estabilidad del sistema vamos a utilizar la siguiente función de Lyapunov y su derivada:

$$V = \frac{1}{2}s^2;\ V(x) > 0\ x \neq 0; \qquad \dot{V} = s\dot{s} \qquad (3.37)$$

Ahora en [33] se escoge lo siguiente:

$$\dot{s} = -Ksign(s) \qquad K > 0 \qquad (3.38)$$

Para así obtener lo siguiente:

$$\dot{V} = s\big(-Ksign(s)\big) = -K|s| < 0 \qquad (3.39)$$

Con lo que se demuestra que el sistema el global asintóticamente estable

Ahora, la obtención de la ecuación (3.37) solo es posible si $u(t)$ es igual a lo siguiente:

$$u(t) = -f(x,t) + \ddot{x}_{set} + \lambda e + Ksign(s) \qquad (3.40)$$

De esta manera, se obtiene la ley de control a utilizar. Se debe tener en cuenta que la aplicación de este último resultado debe considerar que el sistema es de dimensión 4x1 y las siguientes relaciones de variables con la ecuación del sistema robótico:

$u(t) = \tau.$	Torque aplicado
$\ddot{x}_{set} = \ddot{q^*}$	Aceleración deseada
$e = q^* - q$	Error.
λ, K	matrices diagonales de 4x4
$s = s = \dot{e} + \lambda e.$	relación de S y el error.
$f(x,t) = D_{(q_{(t)})}\ddot{q_{(t)}} + h_{(q_{(t)},\dot{q_{(t)}})} + c_{(q_{(t)})} + Df$	modelo matemático

Así. La ley de control final que definida de la siguiente manera:

$$\boldsymbol{\tau = -D_{(q_{(t)})}\ddot{q}_{(t)} - h_{(q_{(t)},\dot{q}_{(t)})} - c_{(q_{(t)})} - Df + \ddot{q}^{*} + \lambda e + Ksign(s)} \qquad (3.41)$$

Existen diversas formas de evitar o minimizar el "chattering" en el control SMC. Algunos autores utilizan condicionales para efectuar diferentes controladores en cada región del SMC y luego utilizan una fusión de estos controladores mediante un control adicional como Fuzzy. Otros autores utilizan una función continua como la tangente hiperbólica para evitar las discontinuidades y generar un control continuo. Por otro lado, en [33] se utiliza la función condicional de saturación para evitar las discontinuidades. Esa función de saturación presenta la siguiente forma mostrada en la expresión.

$$\dot{s}_i = K_i sat(s_i) = \begin{cases} K_i sign(s_i) & if \quad |S_i| > 0 \\ K_i \dfrac{s_i}{d} & if \quad |S_i| \leq d \end{cases} \qquad (3.42)$$

De esta manera, los parámetros a ser usados en el control SMC son: $\boldsymbol{\lambda, K}$ and $\boldsymbol{delay}$ donde $\boldsymbol{\lambda}$ apoya directamente a la velocidad con la que el sistema alcanza el punto de operación deseado, $\boldsymbol{K}$ será la ganancia permitida que el actuador se moverá para mantener el punto de operación y $\boldsymbol{delay}$ es el parámetro que evitará las discontinuidades de la función signo.

Los valores iniciales para estos parámetros serán los siguientes:

$$\begin{aligned} \lambda &= diag\{3 \quad 3 \quad 3 \quad 3\} \\ K &= diag\{15 \quad 15 \quad 15 \quad 15\} \\ delay &= diag\{1 \quad 1 \quad 1 \quad 0.1\} \end{aligned} \qquad (3.43)$$

El SMC diseñado en Simulink se puede ver en la Figura 3.8.

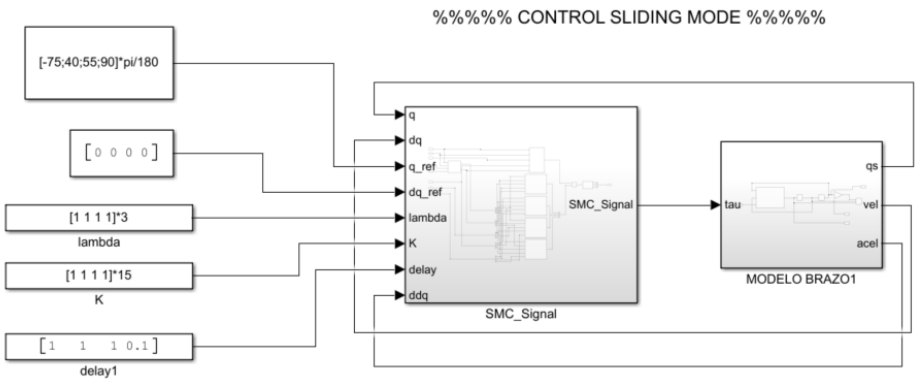


Figura 3.8. Control SMC para manipulador Robótico.

Como se muestra en la figura 3.8. El SMC ha sido aplicado mediante un subsistema que contiene distintas funciones Matlab donde se implementa por código las

ecuaciones que la ley de control necesita desarrollar para su aplicación. Aquí se realiza la prueba para posicionar al sistema en el mismo punto de operación probado para los anteriores controladores obteniendo los resultados mostrados en la figura 3.9.

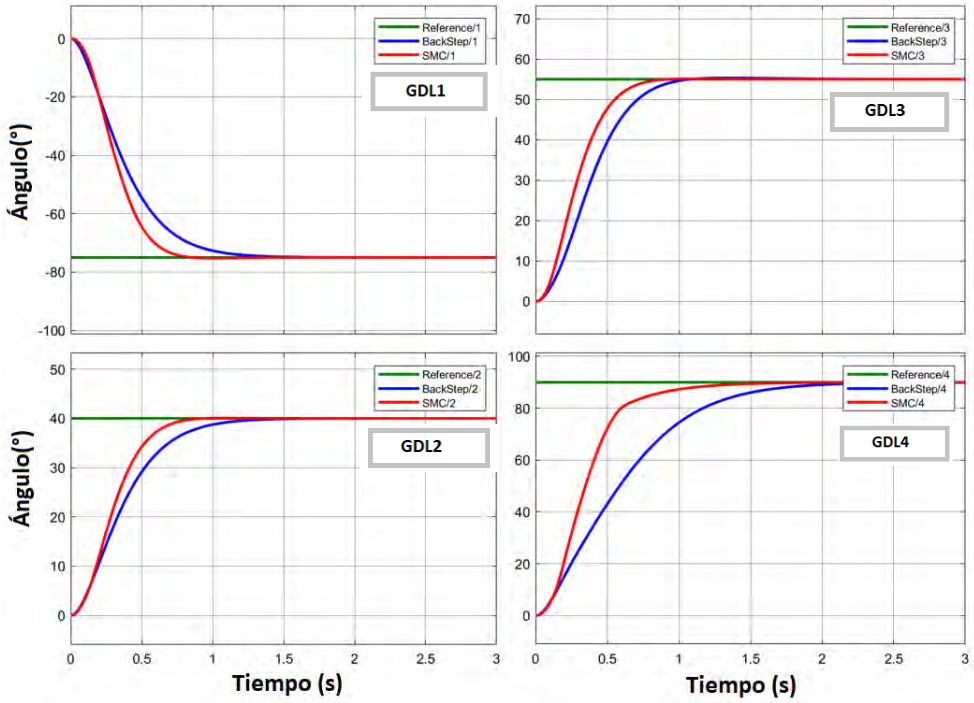


Figura 3.9. Comparativa de control SMC y Backstepping.

Como se puede apreciar, Para las mismas condiciones, el control SMC presenta mejores resultados que el Backstepping. Sin embargo, para lograr estos resultados es necesario una mayor exigencia del sistema como se puede apreciar en la Figura 3.10.

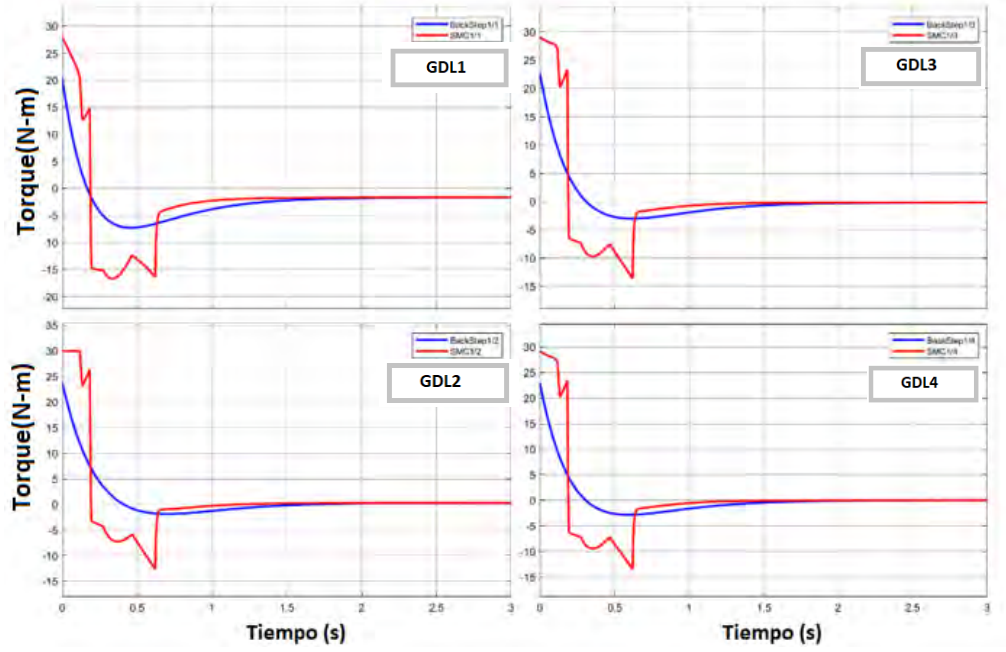


Figura 3.10. Comparativa de Torques aplicados a cada motor.

Como se puede ver, el SMC requiere de mayores torques picos, así como también se presentan cambios bruscos durante el movimiento de los motores, mientras que el sistema Backstepping mantiene torques suaves para el control.

Vale indicar que si no se aplica la saturación (Delay=0) los torques presentarán el "chattering" analizado, esto se muestra en la figura 3.11.

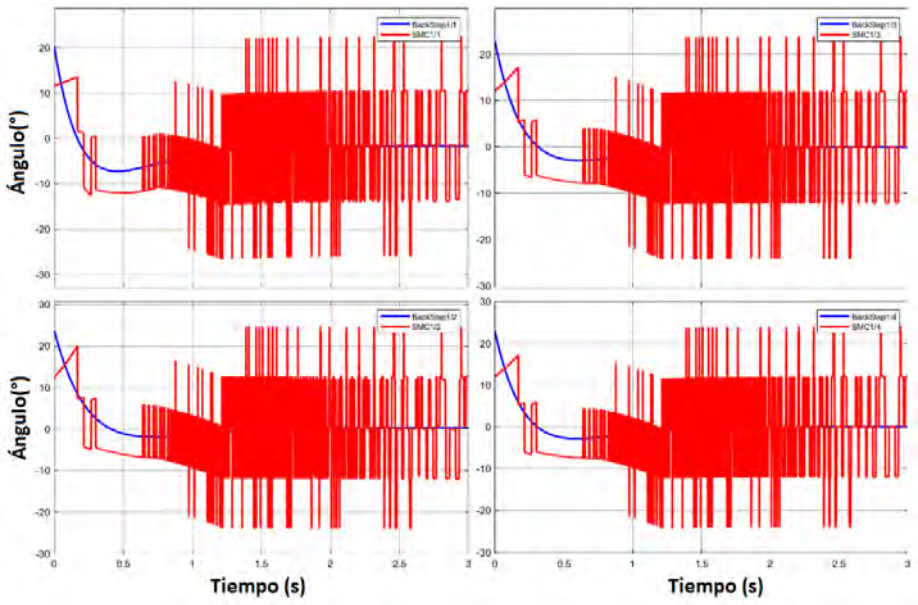


Figura 3.11. Chattering ocasionado por el control SMC sin saturación.

Se debe recordar que el Backstepping nos permite mejorar el control mediante el incremento de sus ganancias, pero de acuerdo a las pruebas de simulación realizadas se ha determinado que, para los requerimientos del sistema, la respuesta obtenida en el Backstepping se considera suficiente. Es cierto que el controlador SMC nos brinda mejores respuestas, pero en contraparte existe una exigencia mayor de los actuadores lo cual puede no ser propicio y hasta perjudicial para estos equipos.

Además, las pruebas de simulación nos indican un mayor cómputo realizado para el SMC que para el control Backstepping y por otro lado vemos que para el algoritmo de SMC es importante contar con los feedback de posición, velocidad y aceleración mientras que con el Backstepping solo es necesario la posición y velocidad y aunque es posible obtener todas estas retroalimentaciones, no cabe duda que las comunicaciones en la implementación estarán más sobrecargadas en el SMC que con el Backstepping.

3.1.5. Simulaciones de perturbaciones de los controles propuestos

Al culminar el diseño todos los tipos de controladores, se realizan una serie de pruebas que nos permitan concluir el uso del controlador más apropiado para la aplicación.

La primera prueba realizada es la comparación de las respuestas de cada controlador a una entrada escalón (SP: -75°, 40°, 55°, 90°), incluso se ha agregado una perturbación de torque en estado estacionario a cada controlador, como se observa en la Figura 3.12.

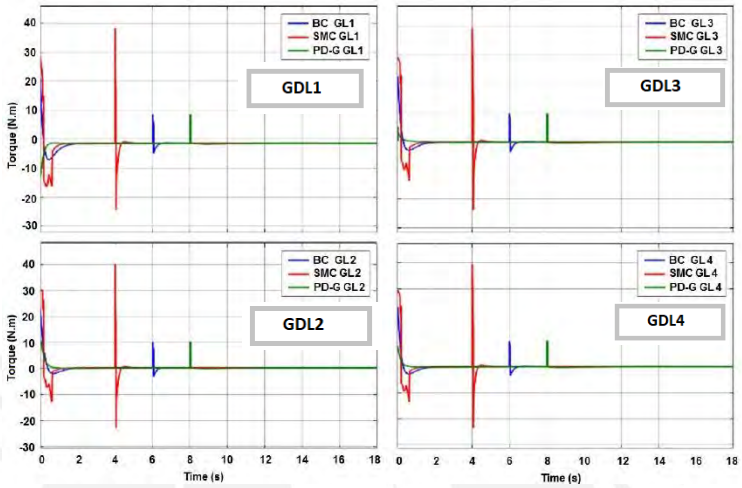


Figura 3.12. Torques con los controladores propuestos a los 4GDL del manipulador.

Como se puede apreciar, se ha aplicado una perturbación de la misma magnitud en diferentes tiempos para cada controlador siendo el control SMC el que requiere mayor valor de torque para regular de manera adecuada la posición solicitada de cada servomotor. En la Figura 3.13 se puede visualizar las posiciones generadas con esta variable de control.

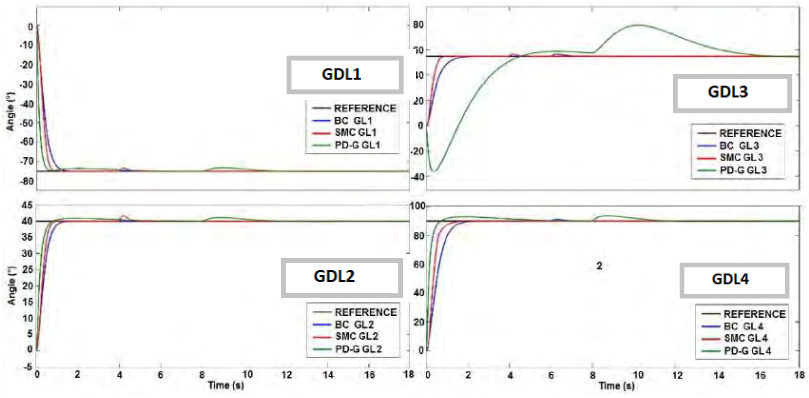


Figura 3.13. Posiciones de los 4 GDL del manipulador obtenidos por los controladores propuestos aplicando una perturbación de torque en SMC (tiempo 4s), Backstepping (tiempo 6s), PD+G (tiempo 8s).

Podemos apreciar, que la regulación es más rápida para los controladores Backstepping y Sliding Mode teniendo la ventaja que el control Backstepping solicita cambios de torque más suavizados que el control Sliding Mode. Por otro lado, vemos que el control Sliding Mode es mucho más rápido y más robusto a las perturbaciones que el control Backstepping.

Dado los requerimientos del Robot asistencial, se puede concluir que es importante mantener una velocidad promedio de los movimientos sin enfocarse tanto en la exactitud de los mismos. De acuerdo a esto, hasta el momento, el mejor controlador que se adapta a las necesidades del proyecto es el control BACKSTEPPING para lo cual ahora vamos a validar sus respuestas respecto a una trayectoria generada por un algoritmo adicional el cual va a ser explicado a continuación.

3.2. Generación de Trayectorias

Existe literatura diversa sobre este tema aplicado a los manipuladores robóticos. La intención de un generador de trayectorias es brindar la posición, velocidad y aceleración que debe tener el manipulador en cada instante de tiempo por lo que la información de salida de un generador de trayectorias será comunicada al controlador como el punto de operación deseado o set point.

Existen diversas formas de generar una trayectoria, en este documento se va a utilizar 2 formas principalmente, una mediante curvas parametrizadas matemáticamente y la otras a partir de polinomios de tercer y quinto orden.

3.2.1. Trayectorias por curvas parametrizadas

Según la geometría diferencial, existe la posibilidad de parametrizar cualquier tipo de curva continua en el espacio. El parámetro más recurrente es el tiempo con lo que se pueden hallar expresiones, respecto a este parámetro, de las coordenadas en cada punto de la curva. En la actualidad, dado el amplio estudio matemático realizado, se tienen curvas paramétricas notables, por ejemplo, en un plano podemos encontrar a la circunferencia, elipse, cardioide, etc. En el espacio se puede encontrar curvas paramétricas tales como la curva helicoidal, curva satelital, solenoide tórico, etc.

La intención de tener una curva paramétrica como un generador de trayectorias, es que de manera directa vamos a poder obtener puntos en cada instante de tiempo, que es

justamente el objetivo de un generador de trayectorias. Para las pruebas de validación a realizar, vamos a utilizar la curva paramétrica de una circunferencia planar.

$$x = -0.1;$$

$$y = r.\sin(wt) + 0.1;$$

$$z = r.\cos(wt) + 0.1; \qquad (3.44)$$

De la ecuación (3.44), se puede notar que corresponde a una circunferencia en el plano YZ a una altura de -0.1 en el eje X. El parámetro "w" nos permitirá variar la velocidad con la cual se genera la circunferencia. Esta curva paramétrica será utilizada para posteriormente realizar las pruebas de validación de los controladores propuestos.

3.2.2. Trayectorias por polinomios

Como se sabe, el movimiento ideal entre 2 puntos es una recta de primer orden. Esto hará que, al derivar la posición, obtengamos un valor de velocidad constante lo cual no nos va a permitir tener control sobre la suavidad y forma en la cual se ejecuta este movimiento recto del efector final del manipulador robótico. Es por eso que en práctica se plantea que el movimiento entre 2 puntos se realice a través de una curva suave polinomial tal como se muestra en la figura 3.14 y de esta manera si se pueda tener control sobre la velocidad y hasta la aceleración como curvas suaves continuas.

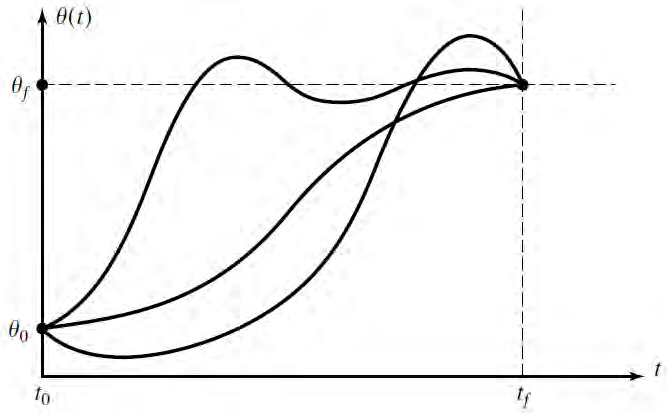


Figura 3.14. Ruta polinómica de una articulación.

Si se desea que el movimiento sea lo más similar a un movimiento recto, será necesario interpolar algunos puntos adicionales entre los puntos de inicio y llegada teniendo así, el control en todo momento de los parámetros del movimiento.

El algoritmo de generación de trayectorias inicia utilizando la información del punto inicial y final para encontrar los parámetros que se han planteado para el polinomio y sus derivadas.

$$\theta_{(t)} = a_o + a_1 t + a_2 t^2 + a_3 t^3;$$

$$\dot{\theta_{(t)}} = a_1 + 2a_2 t + 3a_3 t^2;$$

$$\ddot{\theta_{(t)}} = 2a_2 t + 6a_3 t\,; \qquad (3.45)$$

$$\theta_{(t)} = a_o + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5;$$

$$\dot{\theta_{(t)}} = a_1 + 2a_2 t + 3a_3 t^2 + 4a_4 t^3 + 5a_5 t^4;$$

$$\ddot{\theta_{(t)}} = 2a_2 + 6a_3 t + 12a_4 t^3 + 20a_5 t^3; \qquad (3.46)$$

Estos polinomios y sus derivadas nos representan las leyes del movimiento para polinomio de orden 3 (ecuación (3.45)) y polinomios de orden 5 (ecuación (3.46)).

$$\theta_{(t)} = \theta_0 \qquad \dot{\theta_{(t)}} = \dot{\theta}_0 \qquad \ddot{\theta_{(t)}} = \ddot{\theta}_0$$

$$\theta_{(t)} = \theta_f \qquad \dot{\theta_{(t)}} = \dot{\theta}_f \qquad \ddot{\theta_{(t)}} = \ddot{\theta}_f \qquad (3.47)$$

Las ecuaciones (3.47) representan la información del punto de inicio y final del movimiento, las cuales nos ayudarán a encontrar los parámetros de los polinomios propuestos. Cabe resaltar que para el caso del polinomio de orden 3 con 4 parámetros desconocidos, solo será necesario utilizar la información de posiciones y velocidades iniciales y finales y así obtener un sistema de ecuaciones 4x4 con solución única, mientras que para los polinomios de orden 5 con 6 parámetros serán necesario utilizar también la información de aceleración inicial y final con los que obtendremos un sistema de ecuaciones de 6x6.

Durante las simulaciones, se observa que la forma más adecuada de hallar los parámetros es de manera matricial. Estos sistemas de ecuaciones propuestos como matrices pueden ser resueltos, como se muestra en la figura 3.15, en un software matricial como MATLAB.

```
function [a0,a1,a2,a3] = polinomial3_path(pi,pip,pf,pfp,ti,tf)


%Coeficientes
M = [   1  ti  ti^2    ti^3
        0  1   2*ti    3*ti^2
        1  tf  tf^2    tf^3
        0  1   2*tf    3*tf^2];

P = [   pi
        pip
        pf
        pfp ];

%Coeficientes de F1d
a   =   M\P;
a0  =   a(1);
a1  =   a(2);
a2  =   a(3);
a3  =   a(4);

end
```

```
function [a0,a1,a2,a3,a4,a5] = polinomial5_path(pi,pip,pipp,pf,pfp,pfpp,ti,tf)


%Coeficientes
M = [   1  ti  ti^2    ti^3    ti^4     ti^5
        0  1   2*ti    3*ti^2  4*ti^3   5*ti^4
        0  0   2       6*ti    12*ti^2  20*ti^5
        1  tf  tf^2    tf^3    tf^4     tf^5
        0  1   2*tf    3*tf^2  4*tf^3   5*tf^4
        0  0   2       6*tf    12*tf^2  20*tf^3 ];

P = [   pi
        pip
        pipp
        pf
        pfp
        pfpp    ];

%Coeficientes de F1d
a   =   M\P;
a0  =   a(1);
a1  =   a(2);
a2  =   a(3);
a3  =   a(4);
a4  =   a(5);
a5  =   a(6);

end
```

Figura 3.15. Solución matricial de parámetros de polinomios generadores.

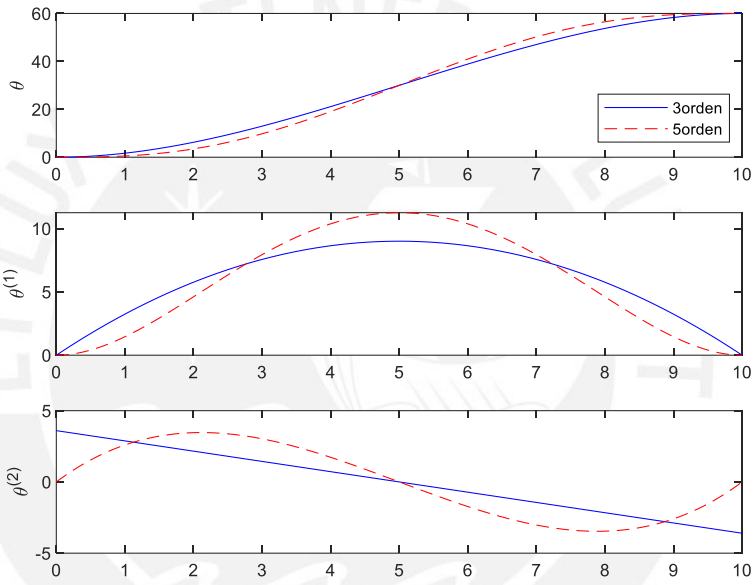


Figura 3.16. movimientos generados por polinomios orden 3 y 5.

Finalmente, se realiza una comparativa en la figura 3.16 sobre los movimientos generados con estos polinomios. Como se puede observar, ambos métodos dan un posicionamiento similar, pero se verá un movimiento más suave por parte de los polinomios de orden 5 dado que el movimiento partirá y llegará al reposo con una mínima velocidad ya que la aceleración en este punto parte de cero, diferente al polinomio de orden 3 donde se parte y se llega con una aceleración inicial y final. En los movimientos del manipulador, este detalle causará un leve estirón al inicio y final de los movimientos. El uso de trayectorias de tercer o quinto orden va a depender exclusivamente del movimiento que vamos a realizar con los brazos, ya que cada rutina necesitará movimientos más rápidos o lentos en la zonas intermedias o extremas de las rutinas a realizar.

3.3. Trayectorias en lazo de control cerrado

Ahora que se tienen todas las herramientas para iniciar los movimientos compuestos de los manipuladores robóticos, podemos completar el lazo cerrado con el algoritmo generador de trayectorias de acuerdo a las necesidades que se tengan.

En la figura 3.17 se presenta el diagrama completo implementado en SIMULINK.

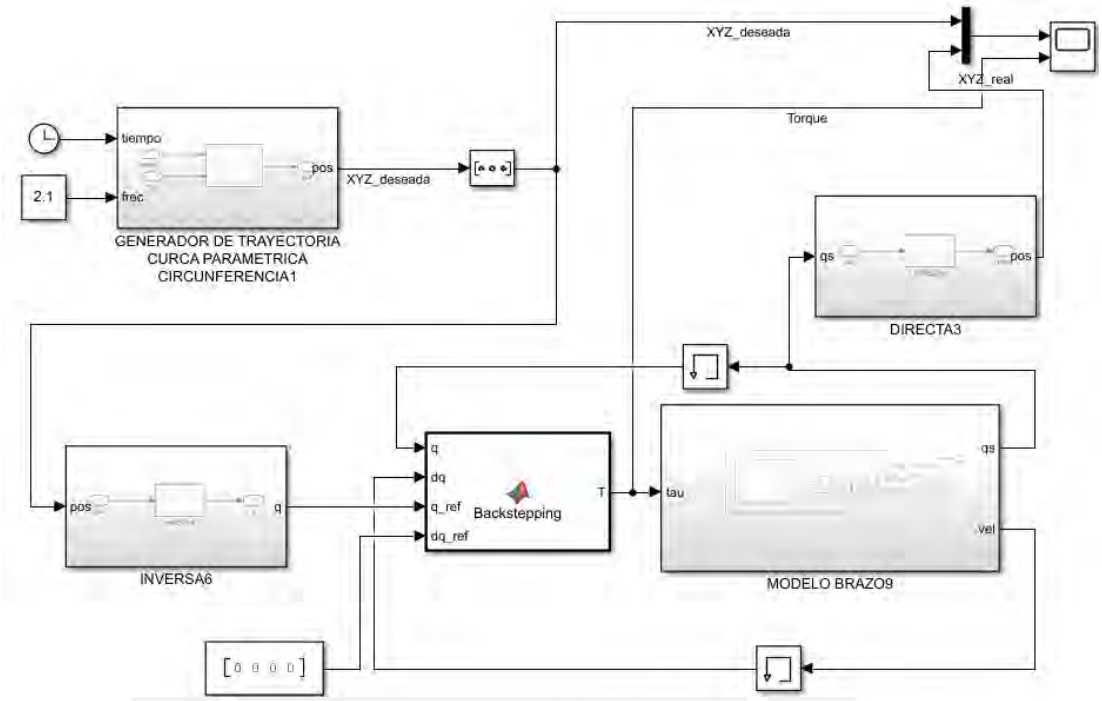


Figura 3.17. Lazo de control final para movimientos de manipulador.

En esta figura, se presenta el diagrama con un control Backstepping, pero este bloque puede ser reemplazado fácilmente por los controladores analizados anteriormente.

En el caso del bloque generador de trayectorias, vemos que las entradas son la frecuencia angular y el vector de tiempo necesarios para generar los puntos deseados a controlar.

Se puede observar también que la secuencia es:

- Generar los puntos en coordenadas cartesianas
- Realizar el algoritmo de cinemática inversa a cada punto para hallar las posiciones angulares deseadas para cada articulación en cada instante de tiempo.

- Generar el lazo de control con estas posiciones deseadas retroalimentando en cada momento el movimiento real al controlador.
- Se ejecuta la cinemática directa a las posiciones generadas por el modelo del manipulador para realizar la comparación de movimientos deseados y reales del efector final en el plano cartesiano.

Para una mejor comprensión de la simulación realizada, se ha iniciado los movimientos con una posición inicial del brazo en un punto de la circunferencia a generar, calculado previamente con la cinemática inversa. Este punto se ha reenviado a los bloques correspondientes del diagrama como una condición inicial.

En la figura 3.18 se muestran los resultados de simulación, comparando las posiciones de las articulaciones para cada controlador propuesto. Esto se ha realizado para un periodo completo, es decir, para el tiempo en el que el manipulador sigue a la circunferencia completa.

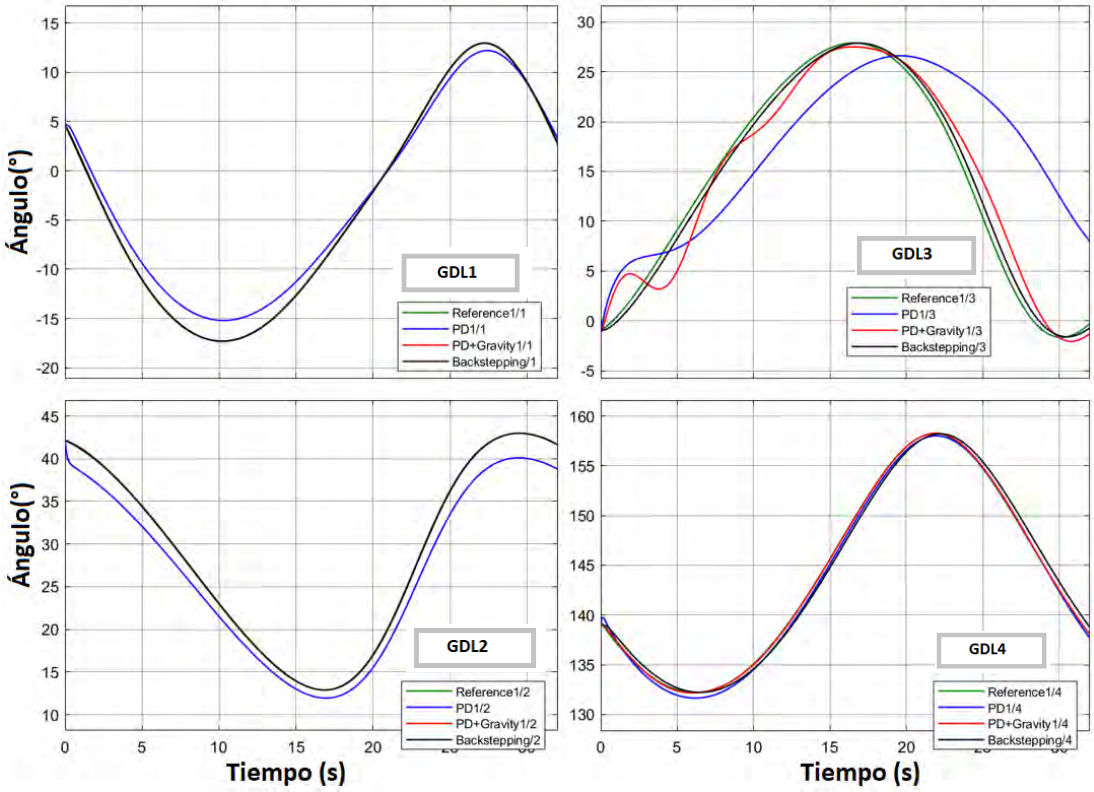


Figura 3.18. Comparativa de control de movimientos articulares.

Como se puede observar, El primer control lineal PD, responde con muchos errores para las 4 articulaciones, mientras que los controles PD no lineal y Backstepping tienen una mejor respuesta más cercana a la referencia dada, siendo el controlador

Backstepping el que mejor performance da para los movimientos de la articulación 3 donde el controlador PD no lineal no puede estabilizarse de manera rápida y el controlador Backstepping solo genera un ligero desfase respecto a la referencia.

En la Figura 3.19 podemos ver las respuestas en el plano YZ dibujado por el efector final del brazo para cada tipo de control.

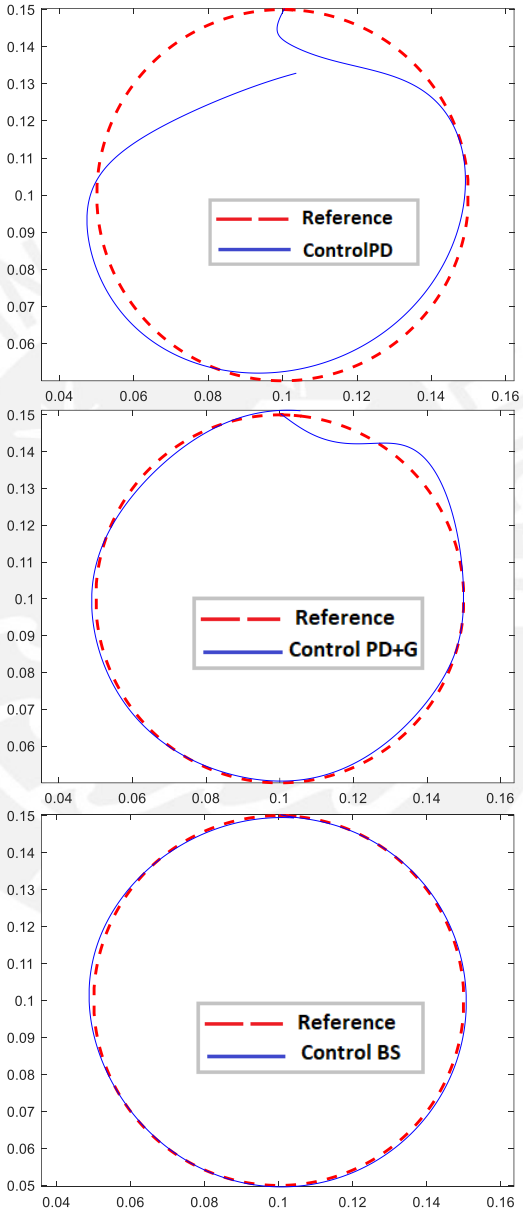


Figura 3.19. trayectorias cartesianas de Brazo para w=0.2 rad/s.

De las figuras 3.17 y 3.18 se puede concluir que tanto para las posiciones articulares y cartesianas, se da un mejor seguimiento para el controlador Backstepping.

Dado que la percepción humana fija la atención en los movimientos cartesianos más que en los articulares, se va a realizar el análisis de los errores obtenidos para el seguimiento en el espacio cartesiano utilizando para esto el siguiente criterio.

$$E = \sqrt{\frac{\sum(e_y^2 + e_z^2)}{r_y^2 + r_z^2}} \qquad (3.48)$$

Donde:

e_y: error entre referencia y curva generada en el eje y.

e_z: error entre referencia y curva generada en el eje z.

r_y: coordenada y de la curva generada.

r_z: coordenada z de la curva generada.

Considerando que para un vector de datos se cumple que: $error^2 = error' * error$.

Se ha realizado la exportación de los datos hacia MATLAB para realizar el análisis de estos errores mediante el script mostrado en la figura 3.20.

```
error_pd_y = (out.circ_ref(:,2)-out.circ_pd(:,2))'*(out.circ_ref(:,2)-out.circ_pd(:,2));
error_pd_z = (out.circ_ref(:,3)-out.circ_pd(:,3))'*(out.circ_ref(:,3)-out.circ_pd(:,3));
error_pd   = error_pd_y+error_pd_z;
ry_pd      = out.circ_pd(:,2)'*out.circ_pd(:,2);
rz_pd      = out.circ_pd(:,3)'*out.circ_pd(:,3);
rt_pd      = ry_pd+rz_pd;
desemp_pd  = sqrt(error_pd/rt_pd)*100

error_pd_g_y = (out.circ_ref(:,2)-out.circ_pd_g(:,2))'*(out.circ_ref(:,2)-out.circ_pd_g(:,2));
error_pd_g_z = (out.circ_ref(:,3)-out.circ_pd_g(:,3))'*(out.circ_ref(:,3)-out.circ_pd_g(:,3));
error_pd_g   = error_pd_g_y+error_pd_g_z;
ry_pd_g      = out.circ_pd_g(:,2)'*out.circ_pd_g(:,2);
rz_pd_g      = out.circ_pd_g(:,3)'*out.circ_pd_g(:,3);
rt_pd_g      = ry_pd_g+rz_pd_g;
desemp_pd_g  = sqrt(error_pd_g/rt_pd_g)*100

error_bs_y = (out.circ_ref(:,2)-out.circ_bs(:,2))'*(out.circ_ref(:,2)-out.circ_bs(:,2));
error_bs_z = (out.circ_ref(:,3)-out.circ_bs(:,3))'*(out.circ_ref(:,3)-out.circ_bs(:,3));
error_bs   = error_bs_y+error_bs_z;
ry_bs      = out.circ_bs(:,2)'*out.circ_bs(:,2);
rz_bs      = out.circ_bs(:,3)'*out.circ_bs(:,3);
rt_bs      = ry_bs+rz_bs;
desemp_bs  = sqrt(error_bs/rt_bs)*100
```

Figura 3.20. Script de análisis de errores para cada controlador.

De esta manera se ha conseguido estimar los índices de errores para cada controlador. Estos resultados se muestran en la tabla 3.1.

Tabla 3.1. Errores para seguimiento de circunferencia planar con w=0.2 rad/s.

	PD	PD+G	Backstepping
error	8.4707%	2.0078%	1.0693%

Aquí se puede comprobar matemáticamente que el menor error producido en el seguimiento de la trayectoria circular plana es utilizando el controlador Backstepping.

Los seguimientos de cada coordenada XYZ y los torques se puede visualizar en la figura 3.21.

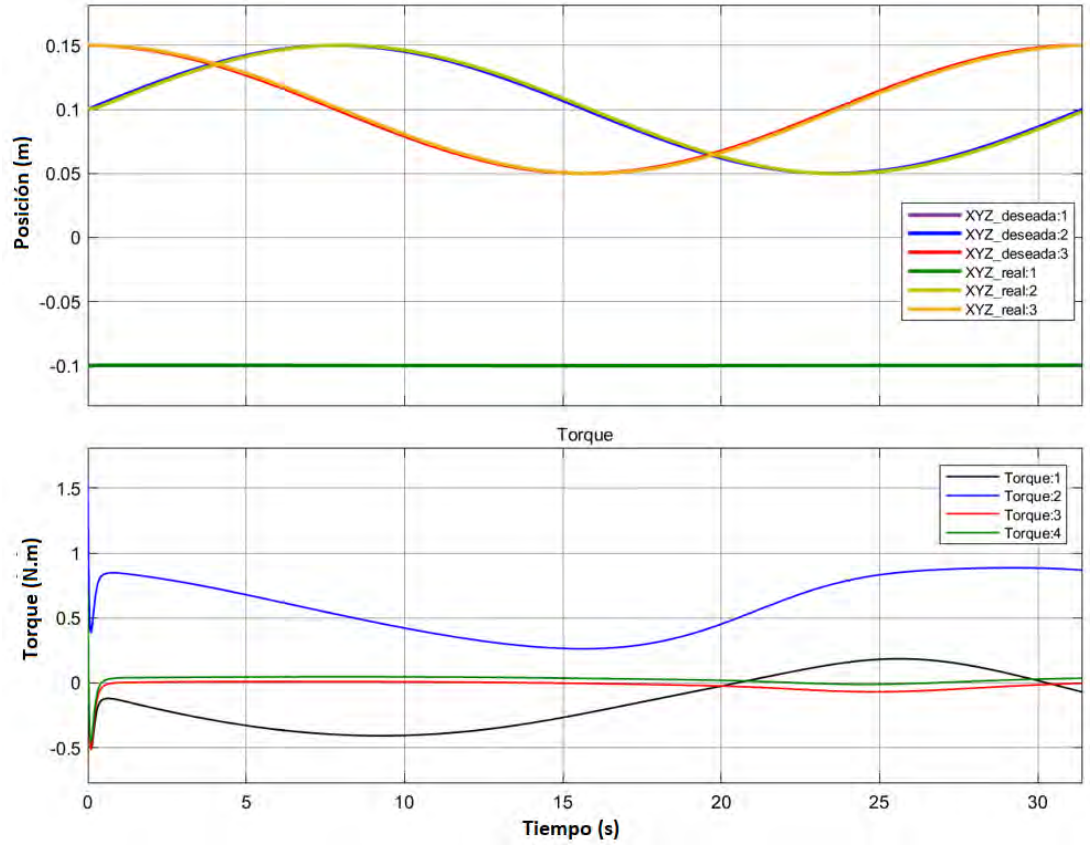


Figura 3.21. Posiciones XYZ y Torques con control Backstepping para w=0.2 rad/s.

Aquí se puede ver un correcto seguimiento con un ligero desfase de cada coordenada, además el torque utilizado no supera los 2 N-m. lo cual es propicio para la mayoría de servomotores que trabajan alrededor de esa fuerza.

Se debe tener en cuenta que esta primera prueba se ha realizado para una frecuencia angular "w" de 0.2 rad/s. De esta manera, la circunferencia será dibujada en su totalidad cuando se recorra 2π radianes por lo que el tiempo que demorará el brazo en realizar una circunferencia de 10 cm de radio será de **31.41 s**. que es justamente el tiempo que se le ha dado a la simulación y lo cual puede ser comprobado en la figura 3.18 donde se analiza un periodo de movimiento de las articulaciones.

Considerando el movimiento de un brazo humano, el movimiento para seguir a una trayectoria circular de 10 cm. lo debería poder realizar entre 2 y 3 segundos de manera natural, por lo que realizar esta trayectoria en más de 30 segundos no sería razonable

para el brazo robótico, donde lo que se intenta es tener movimientos naturales y lo más similar posible a los gestos humanos.

Es así que, si se requiere realizar la misma trayectoria circular en 3 segundos, por ejemplo, será necesario, por una regla de tres simple, adecuar nuestra frecuencia angular "w" a 2.1 rad/s. En la figura 3.22, justamente se realiza la simulación para esta nueva frecuencia angular y un tiempo de simulación de **2.9920 s.**

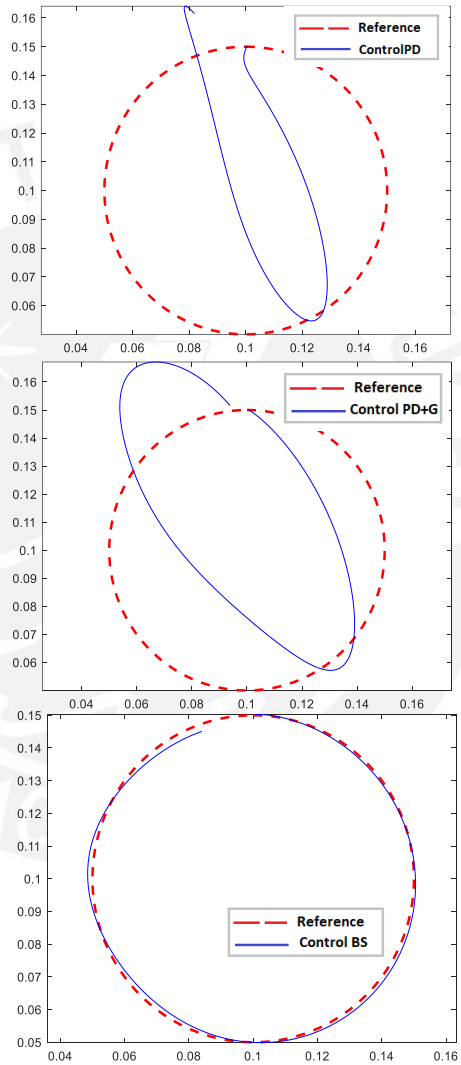


Figura 3.22. trayectorias cartesianas de Brazo para w=2.1 rad/s.

Con esta velocidad angular, el tiempo para efectuar el movimiento circular completo es de 3 segundos, pero vemos que los controladores ya comienzan a tener falencias en el seguimiento de la trayectoria. Solamente el controlador Backstepping mantiene un correcto seguimiento, pero el desfase se hace más evidente para esta velocidad. En la

figura 3.23 se puede comprobar el desfase que existe en el seguimiento de los pares YZ.

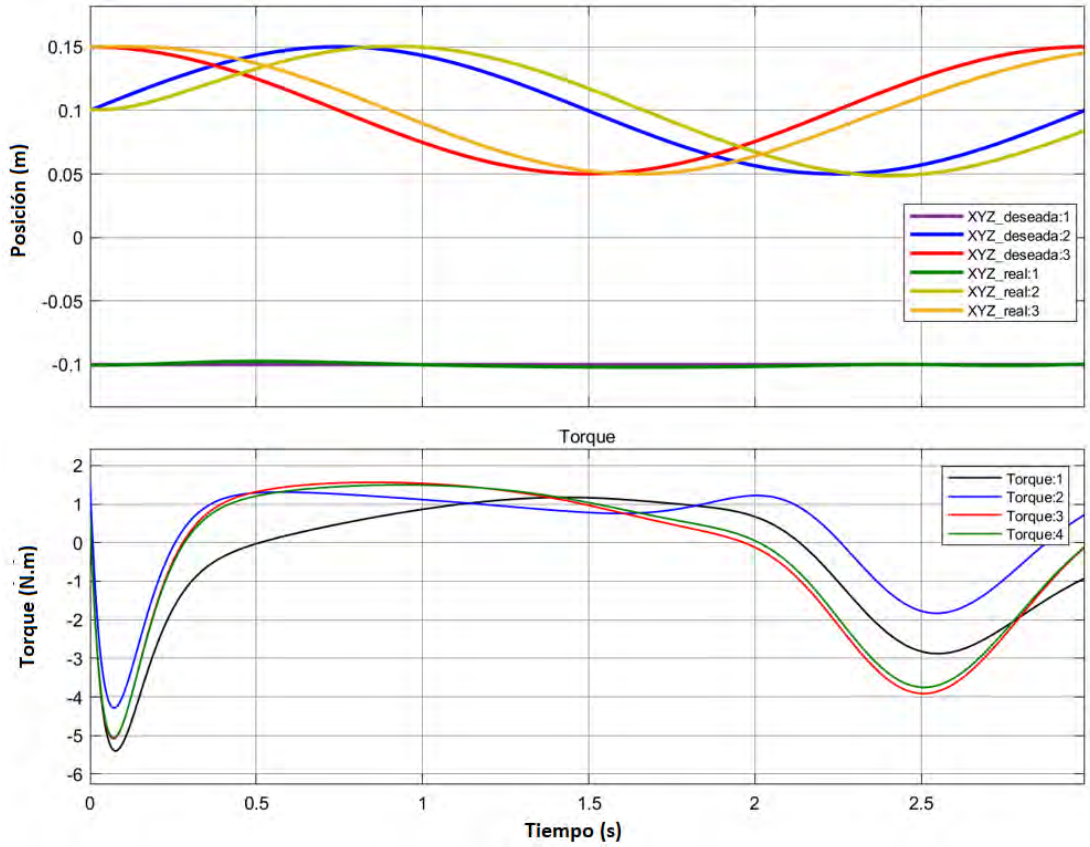


Figura 3.23. Posiciones XYZ y Torques con control Backstepping para w=2.1 rad/s.

Además, en esta misma figura, se observar un mayor esfuerzo del control dado que el torque pico llega hasta los 5 N-m. por lo que se debería seleccionar los servoactuadores que puedan trabajar en ese rango de torque.

Los errores de seguimiento de trayectoria para la velocidad angular de 2.1 rad/s se muestra en la tabla 3.2.

Tabla 3.2. Errores para seguimiento de circunferencia planar con w=2.1 rad/s.

	PD	PD+G	Backstepping
error	18.5030%	15.9374%	10.6776%

Se puede confirmar, en todo caso, que los errores de seguimiento serán mayores mientras mayor velocidad se exige al sistema, aunque en el caso del Backstepping este error se manifiesta más por el desfase provocado por el control. La ventaja del control Backstepping es que el problema puede solucionarse si se aumentan las ganancias de

controlador. Las pruebas anteriores se han realizado para los siguientes valores de ganancias:

$$K_1 = diag([15, 15, 15, 15]) \qquad K_2 = diag([15, 15, 8, 8])$$

Ahora se mostrarán los resultados para las siguientes ganancias.

$$K_1 = diag([40, 40, 40, 40]) \qquad K_2 = diag([40, 40, 30, 30])$$

En la figura 3.24. se muestran los resultados del seguimiento de trayectoria.

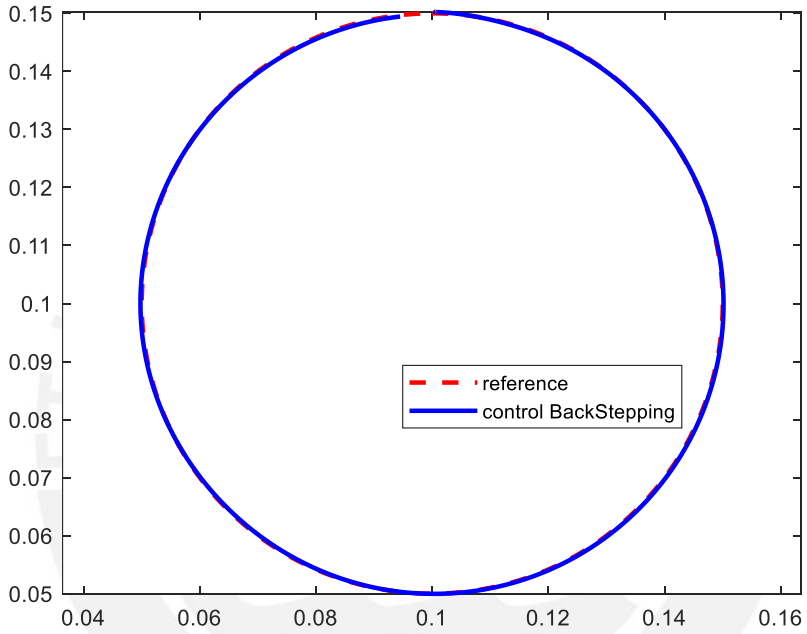


Figura 3.24. Seguimiento de trayectoria para altas ganancias de Backstepping.

Con estas nuevas ganancias se ha reducido el error anterior del seguimiento de **10.6% al 3.63%** donde también se han reducido aún más el desfase que se tenía en el controlador. De esta manera se puede comprobar que una de las ventajas del control Backstepping es que su rendimiento puede mejorar si trabajamos con mayores ganancias y podemos cumplir con los requerimientos solicitados por el sistema. La desventaja del aumento de estas ganancias es que el control será más exigido como se puede mostrar en la figura 3.25.

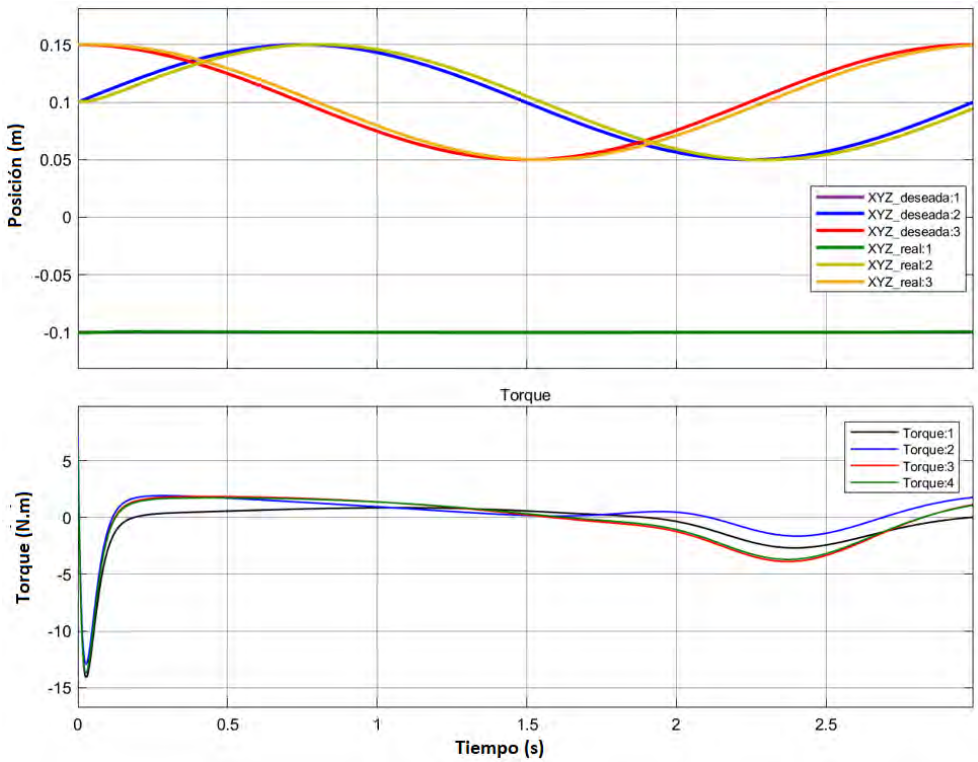


Figura 3.25. Posiciones XYZ y torques para Backstepping con altas ganancias.

Aquí se puede confirmar que el desfase se ha corregido significativamente, pero en contraparte los torques exigidos por el control son mayores, llegando a picos de control de más de 10 N-m por lo que habría que considerar servoactuadores que lleguen a estas capacidades de torques para efectuar el control diseñado.

Adicionalmente se han realizado estas pruebas con el controlador SMC y se ha realizado las comparativas con los anteriores controladores tal como se puede observar en la figura 3.26 para ambas frecuencias angulares.

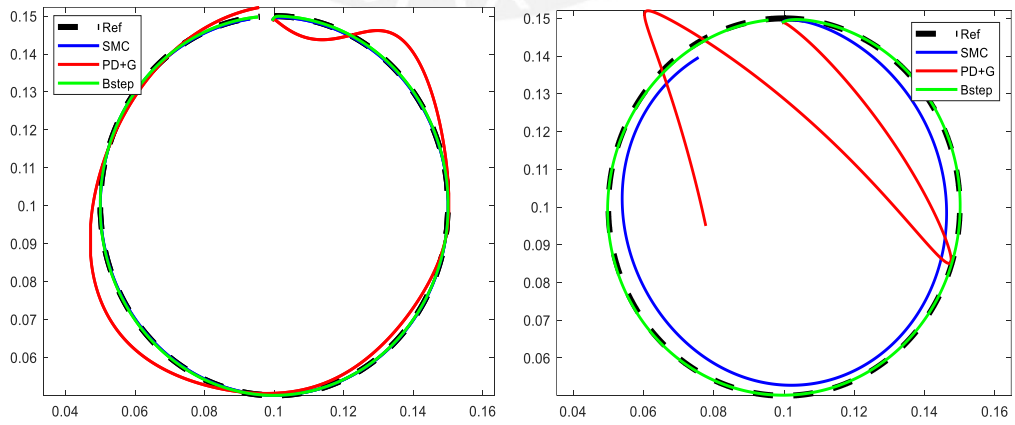


Figura 3.26. Movimiento cartesiano del efector final a diferentes frecuencias (izq) w=0.2, (der) w=2.1

Incluso en estas condiciones podemos notar mejor la efectividad de los controladores no lineales frente a los lineales dado que Backstepping y SMC presentan mejores resultados mientras que se aumenta la velocidad de sistema y tienen la ventaja de mejorar su performance a medida que se recalculan las ganancias de sus leyes de control. En la figura 3.27 podremos ver el resultado de estas pruebas en un esquema 3D donde se podrá observar mejor el movimiento del efector final para cada uno de los controles diseñados respecto a la circunferencia referencial.

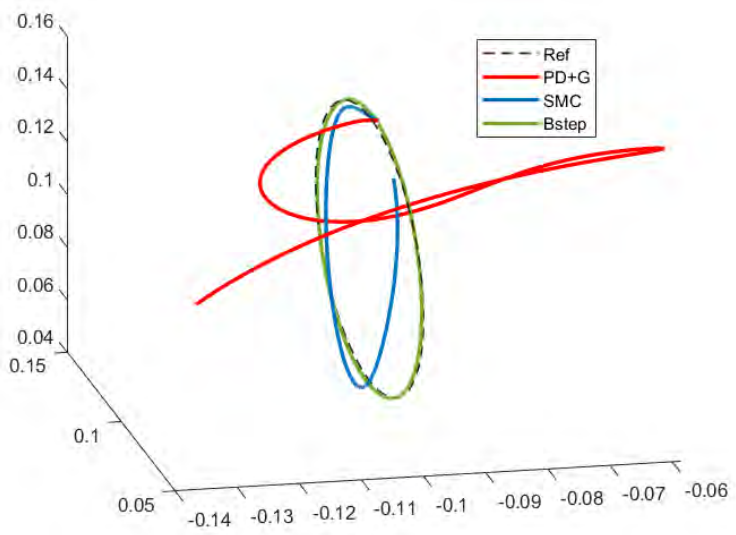


Figura 3.27. Comparación de controladores para el movimiento 3D del efector final.

Así, para las ganancias seleccionadas se puede ver un mejor seguimiento al utilizar el control Backstepping.

Ahora se realizarán las pruebas simuladas para una generación de trayectorias por polinomios para llegar de un punto a otro en las especificaciones de movimientos deseados.

En este caso el procedimiento para la generación de trayectorias, tanto para polinomio de orden 3 y 5 es el siguiente:

- Definir el tiempo para realizar el movimiento.
- Fijar las posiciones, velocidades, aceleraciones (solo para polinomio orden 5), iniciales y finales del movimiento.
- Utilizar la cinemática inversa para hallar las posiciones angulares de la articulaciones iniciales y finales.

- Generar las trayectorias angulares para cada articulación con los polinomios de orden 3 y 5.
- Enviar para cada instante de tiempo las trayectorias generadas al control como posiciones y velocidades deseadas.

Por simplicidad, en el ejemplo se están asignando puntos que inician y terminan en el reposo. En la figura 3.28 podremos ver el diagrama de bloques implementado para la generación de trayectorias en el caso de polinomio de orden 5. Para el caso de orden 3 solo se deben obviar las aceleraciones deseadas.

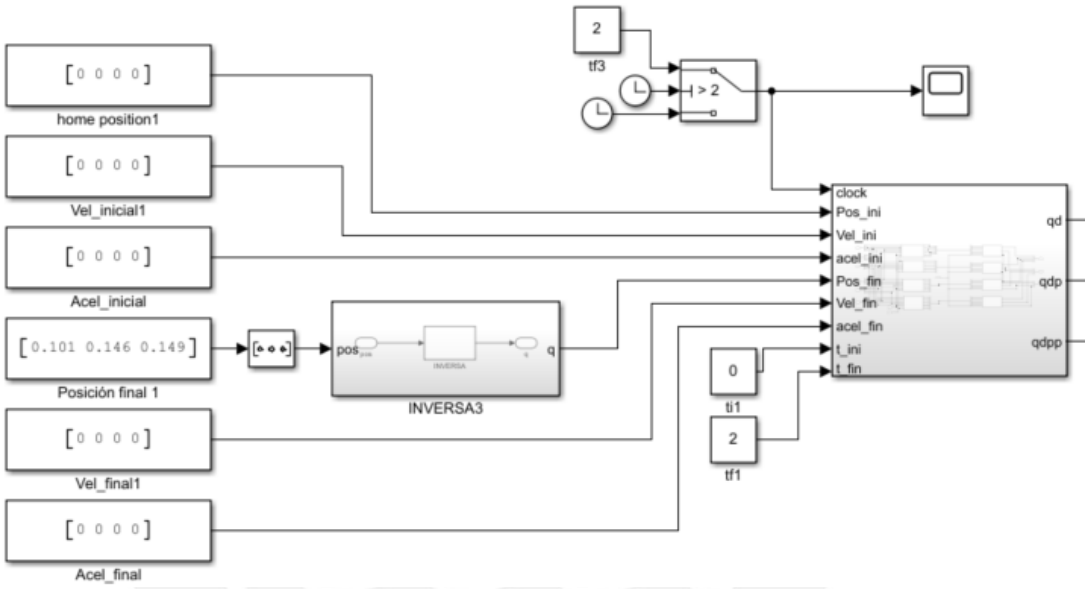


Figura 3.28. Implementación de generación de trayectoria con polinomio de orden 5.

Aquí se muestra el procedimiento descrito anteriormente y también se hace uso de un bloque switch para fines de simulación posterior al tiempo del movimiento y observar la estabilidad de la trayectoria.

El control Backstepping será trabajado para las primeras ganancias simuladas anteriormente.

$$K_1 = diag([15, 15, 15, 15]) \qquad K_2 = diag([15, 15, 8, 8])$$

La figura 3.29. muestran los resultados obtenidos para la generación de trayectoria con polinomio de orden 3 y 5.

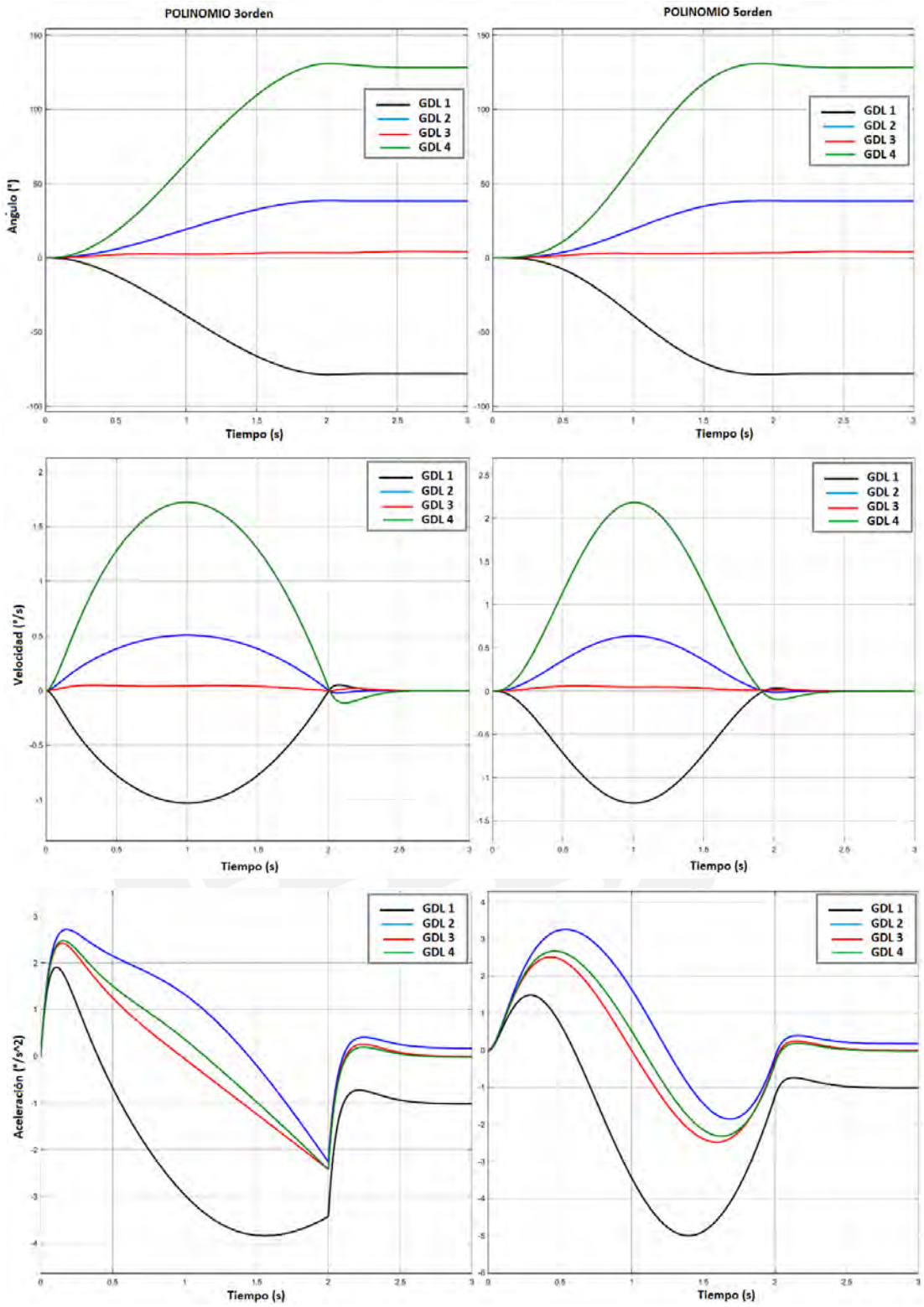


Figura 3.29. Posiciones, Velocidades y Torques para generación por polinomios.

Tal y como se había demostrado, la respuesta de trayectorias con polinomios de orden 5, permitirá obtener movimientos más suavizados al inicio y al final del movimiento.

En la generación por polinomios de orden 3, se puede notar el cambio brusco que se provocan en los torques para lograr los movimientos, mientras que con polinomios de orden 5, se tienen torques creciendo con suavidad y en contraparte se generan picos más altos por lo que, nuevamente, el uso de polinomios de mayor grado estará fuertemente ligados a los requisitos que soliciten para las rutinas de movimiento a realizar por los brazos y la capacidad que tienen sus actuadores.

3.4. Simulación de rutinas de movimientos de robot teleoperado

Durante la implementación, podremos observar que los movimientos a los que se quiere llegar con los brazos del robot, son de acuerdo a puntos guía que se pondrán en el plano cartesiano para que siga la trayectoria que pase por estos puntos. En esta última prueba de comparación, se aplican los distintos controladores al modelo matemático de los brazos robóticos para obtener resultados que nos brindarán la conclusión del controlador más apropiado a utilizar para nuestra aplicación.

De esta forma se ha creado un diagrama de control en Simulink donde se ha agregado un bloque programado de generación de trayectorias que a partir de puntos cartesianos pueda brindar la trayectoria que debe recorrer el brazo en cada instante de tiempo. El diagrama de control se puede visualizar en la figura 3.30.

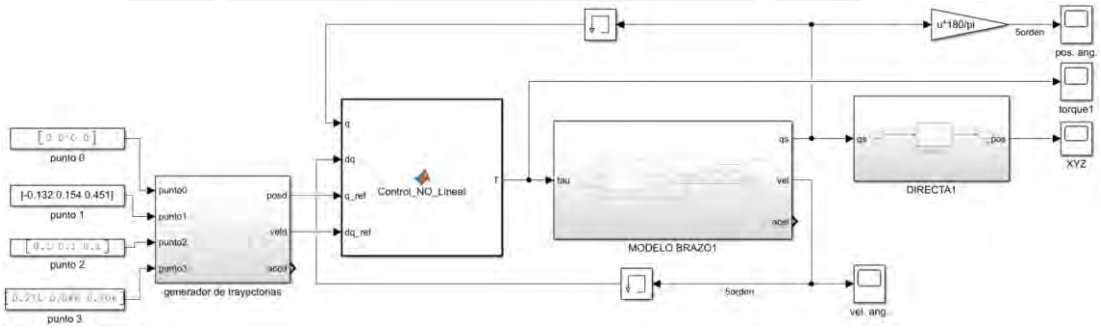


Figura 3.30. Diagrama de control con generador de trayectorias polinómicas.

Así, se puede notar que la referencia para el controlador estará dada por este bloque generador de trayectorias y a la vez siempre necesitará de la retroalimentación de la posición y velocidad en todo momento. (Para algunos controladores como el SMC será necesario incluso la aceleración). De esta manera se podrá evaluar los controladores para cambios de set point y a la vez ver su desempeño ante movimientos similares a los que deberá realizar el prototipo implementado y presentado en el siguiente capítulo.

En la figura 3.31 se podrá observar la referencia angular que debe seguir cada articulación del brazo y las trayectorias realizadas utilizando cada controlador hasta ahora estudiado.

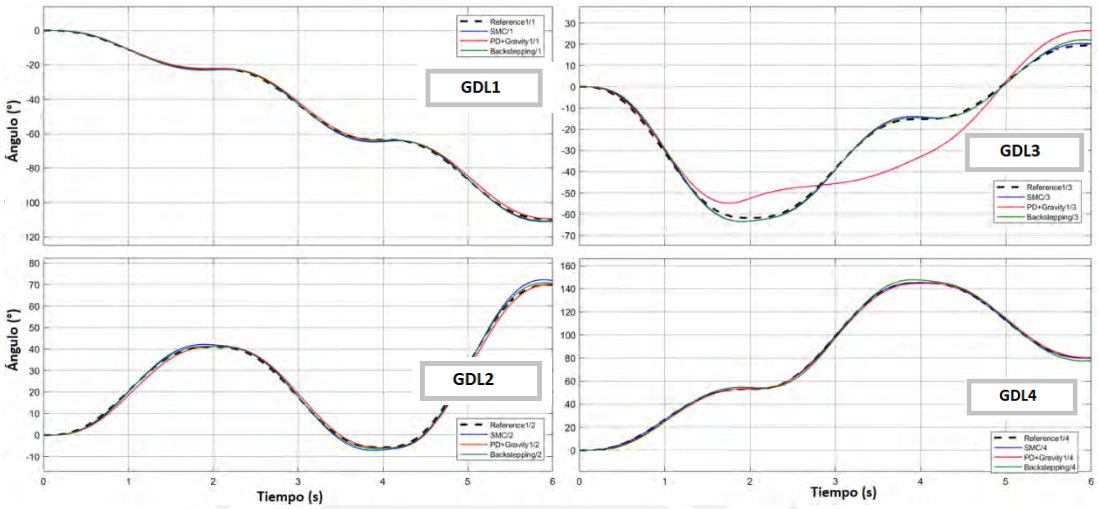


Figura 3.31. Gráficas de movimientos de cada articulación de brazo con generador de trayectoria polinómico.

Nuevamente, se puede observar la eficiencia que tienen los controladores no lineales puros como el Backstepping y el SMC ante el control no lineal PD con compensación de gravedad, pero también se debe resaltar la acción de los torques de cada articulación lo cual se muestra en la figura 3.32 lo cual es importante ya que los actuadores a elegir para la implementación trabajan hasta cierto rango de torque los cuales han sido limitados para esta simulación.

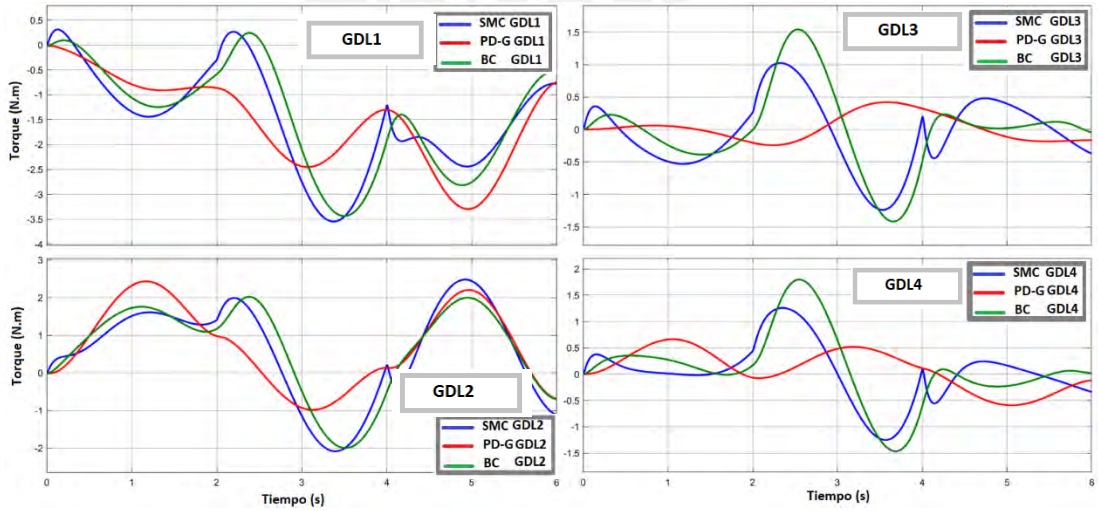


Figura 3.32. Torques aplicados para cada controlador con generador polinómico.

Es importante mencionar que los controladores pueden ser mejorados en caso se apliquen ganancias más altas, pero esto, como se sabe, haría necesario mayor torque en los motores por lo que las simulaciones presentadas han sido realizadas para un uso de torques similares y que no excedan la capacidad de los actuadores a utilizar en la implementación. Así, se puede notar que los controladores para este movimiento no superan el torque de 2N-m para cada articulación.

Algo adicional que se debe notar, es que, si bien el control SMC tiene mejor posicionamiento de las articulaciones respecto a la referencia, es el control Backstepping que requiere de cambios más suaves en los actuadores lo cual puede beneficiar y debe ser tomado en cuenta para el control interno que se debe realizar en la implementación.

En la figura 3.33. se muestra el recorrido que realizará el brazo en el espacio cartesiana XYZ.

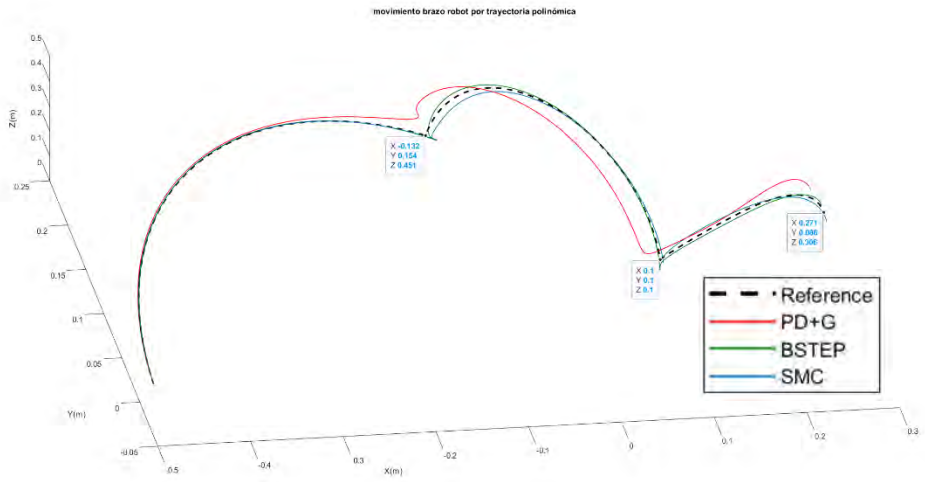


Figura 3.33. Movimiento cartesiano 3D del brazo robóticos para seguimiento de trayectoria polinómica.

Es importante notar en la figura 3.33, que la referencia pasa por los puntos solicitados en el diagrama de control y que el controlador PD con compensación de gravedad es quien más problemas da para el seguimiento de este tipo de trayectoria. Esto se puede notar en la Tabla 3.3 donde se muestran los errores del seguimiento definido con la ecuación 3.48.

Tabla 3.3. Errores de controladores para seguimiento de trayectoria polinómica.

	PD+G	Backstepping	SMC
error	8.0307 %	2.1541 %	2.0206 %

De esta manera, se puede demostrar que los controladores Backstepping y SMC tienen una performance similar para los torques generadores similares.

Como último punto de estas simulaciones, es importante mencionar que la ejecución en simulink para el control PD con compensación de gravedad tuvo una duración de 6.7 segundos mientras que el Backstepping duró 7.6 segundos y el control SMC, tuvo una duración final de 10.8 segundos. La diferencia de los tiempos de ejecución tiene que ver directamente con el desarrollo de los algoritmos que deben ser realizados para cada controlador siendo el control SMC el que mayor cómputo realiza durante su ejecución. Esto también es un punto importante a tomar en cuenta cuando se lleven todos estos algoritmos a su implementación en microcontrolador físico.

3.5. Conclusiones de la comparativa de controladores

A lo largo del capítulo, se ha presentado el diseño de los posibles controladores a utilizar para el proyecto, así como una serie de pruebas simuladas que nos permitan decidir por el mejor controlador que se acomode a las necesidades de la implementación. La comparación de las pruebas realizadas se presenta en la tabla 3.4 donde se puede ver punto a punto los resultados de cada controlador PD con compensación de gravedad, Backstepping y control por regímenes deslizantes SMC.

Tabla 3.4. Comparativa de resultados del uso de controladores no lineales

Comparativa controladores			
	PD+G	**Backstepping**	**SMC**
Error trayectoria circular	**15.93%**	3.63%	3.23%
Error trayect. polinomial	**8.0307%**	2.1541%	2.0206%
Tiempo ejecución	6.7 seg	7.6 seg.	**10.8 seg.**
Control perturbaciones	**Regular**	Alto	Alto
Cambio de Setpoint	**Regular**	Alto	Alto
Datos feedback	Pos, vel	Pos, vel	**Pos, Vel, Acc**
Cambios suave actuador	Si	Si	**No**
Control puede mejorar?	**No**	Si	Si

En esta tabla, se muestran resaltados en cada punto de comparación, el controlador más desfavorable. Se debe tener en cuenta que es el controlador SMC quien tiene la mejor regulación y seguimiento en el control en lazo cerrado sin embargo en la comparativa se toman en cuenta factores que pueden ser determinantes para la implementación del algoritmo en el controlador. Es así que el **control Backstepping**, dado que tiene resultados muy cercanos al controlador SMC y además considera algunas ventajas importantes para la implementación como el tiempo de ejecución y la suavidad de cambios del actuador, **es considerado como el controlador más apropiado para este trabajo**.

Es importante mencionar que, aunque se ha tomado como conclusión el uso del controlador Backstepping como controlador final del trabajo, será necesario el uso e implementación del control PD con compensación de gravedad quien nos ayudará a obtener datos experimentales para la identificación de los coeficientes de fricción que son los parámetros del modelo de los brazos robóticos que aún son necesarios para obtener el modelo completo a utilizar para el control final.

En conclusión, ya que se ha demostrado la eficiencia del controlador Backstepping frente a otros controladores, la implementación va a requerir el conocimiento más próximo del modelo del sistema por lo que la primera tarea para implementar el lazo cerrado de control será mediante pruebas experimentales continuas, validar que el modelo obtenido produzca respuestas muy similares al sistema real. Una vez realizado este punto, será posible implementar el controlador Backstepping y modificarlo según los alcances de los actuadores que se tienen en el sistema robótico afectando de esta manera la selección de las ganancias a utilizar para este controlador. Estas tareas son parte de la implementación que será tema a tratar en el capítulo siguiente.

4. IMPLEMENTACIÓN DEL SISTEMA

Dado que ya se tiene los diseños correspondientes al control de los movimientos de los brazos y cabeza del robot asistencial, se han realizado una serie de actividades como la selección y configuración de equipos electrónicos, pruebas de validación de modelos obtenidos, programación de microcontroladores, etc. para realizar la implementación del sistema y que este acorde al diseño realizado. En las siguientes páginas se podrá observar la forma de implementación, así como los resultados obtenidos.

4.1. Desarrollo tecnológico.

El desarrollo de la presente tesis estará enfocado en el control de expresiones corporales y faciales del sistema robótico realizando la implementación en el prototipo inicial del robot realizado como parte del proyecto "Robot teleoperado para el tratamiento de afecciones mentales". En ese sentido, se cuenta con los siguientes componentes generales que serán utilizados para el control del robot.

- Brazo derecho con cuatro grados de libertad compuesto de cuatro servomotores con caja reductora para la generación de mayor torque y estabilidad, alimentación de 24Vdc y señal de realimentación por encoder para la posición angular con comunicación sobre el protocolo CANBUS.
- Brazo Izquierdo con cuatro grados de libertad compuesto por cuatro servomotores con caja reductora para la generación de mayor torque y estabilidad, alimentación de 24Vdc y señal de realimentación por encoder para la posición angular con comunicación sobre el protocolo CANBUS.
- Cabeza con dos grados de libertad mediante dos servomotores con caja reductora para la generación de mayor torque y estabilidad, alimentación de 24Vdc y señal de realimentación por encoder para la posición angular con comunicación sobre el protocolo CANBUS.
- Controladores de servomotores basado en microprocesador STM32 High performance MCUs.
- Controlador de servomotores basado en microprocesador STM32 Mainstream MCUs.
- Módulo central de control CPU Jetson Nano.

Para mejor entendimiento de los equipos utilizados, en el ANEXO B se dará una breve referencia de cada componente tecnológico utilizado para el diseño del proyecto y como se logra la comunicación de todos estos elementos con el objetivo de lograr los movimientos coordinados y controlados del robot.

4.2.Esquema de Control

Ahora que se ha revisado de manera general los principales componentes a utilizar para el proyecto, podemos en todo caso, presentar el esquema de control utilizado y la interconexión de los equipos para lograr la autonomía que se requiere para las funciones motrices del robot teleoperado.

El diagrama de bloques presentado a continuación plantea una supervisión de control completa del robot desde el procesador Jetson Nano, en la cual se ingresarán los comandos de ejecución de robot por medio de sus periféricos o Wireless. Estos comandos serán enviados a cada uno de los microprocesadores STM32, los cuales tienen preparadas las rutinas de ejecución de movimiento controlado para las extremidades y la cabeza del robot. En la Figura 4.1 podremos observar este diagrama y veo como estarán interconectados todos los equipos involucrados en el control del robot.

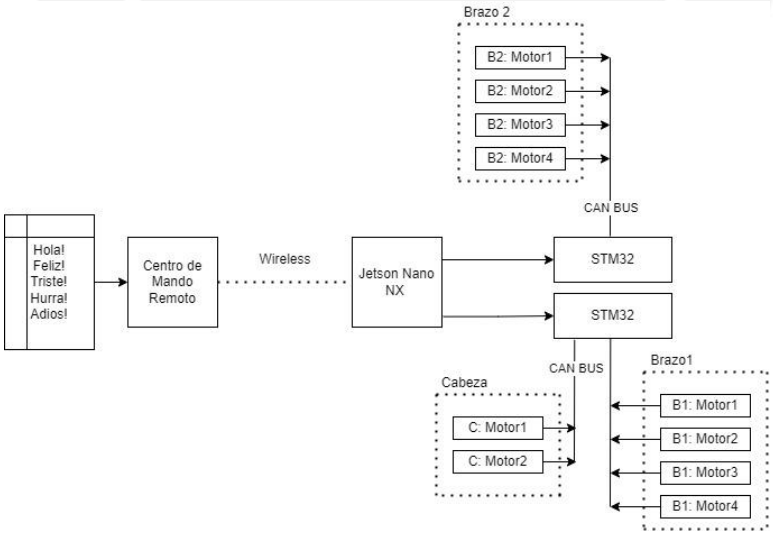


Figura 4.1. Esquema general de control de robot teleoperado (Elaboración propia).

Con este esquema, es posible realizar la configuración y programación de cada equipo que serán tópicos a tratar a continuación.

4.3.Implementación de sistema de Control descentralizado

Los motores GYEMS utilizados para el movimiento de los brazos cuentan con un control de posición proporcional integrativo PI interno y el protocolo CAN BUS para la comunicación con el controlador central. Estos controles han sido previamente configurados y están explicados en el ANEXO C.

De esta manera, el posicionamiento final de la muñeca en lazo abierto se podrá lograr con un correcto control de posición de cada motor independiente.

Teniendo en cuenta que el control no lineal será aplicado a cada extremidad del robot, se puede obtener un control en lazo cerrado representado por la figura 4.2 siguiente.

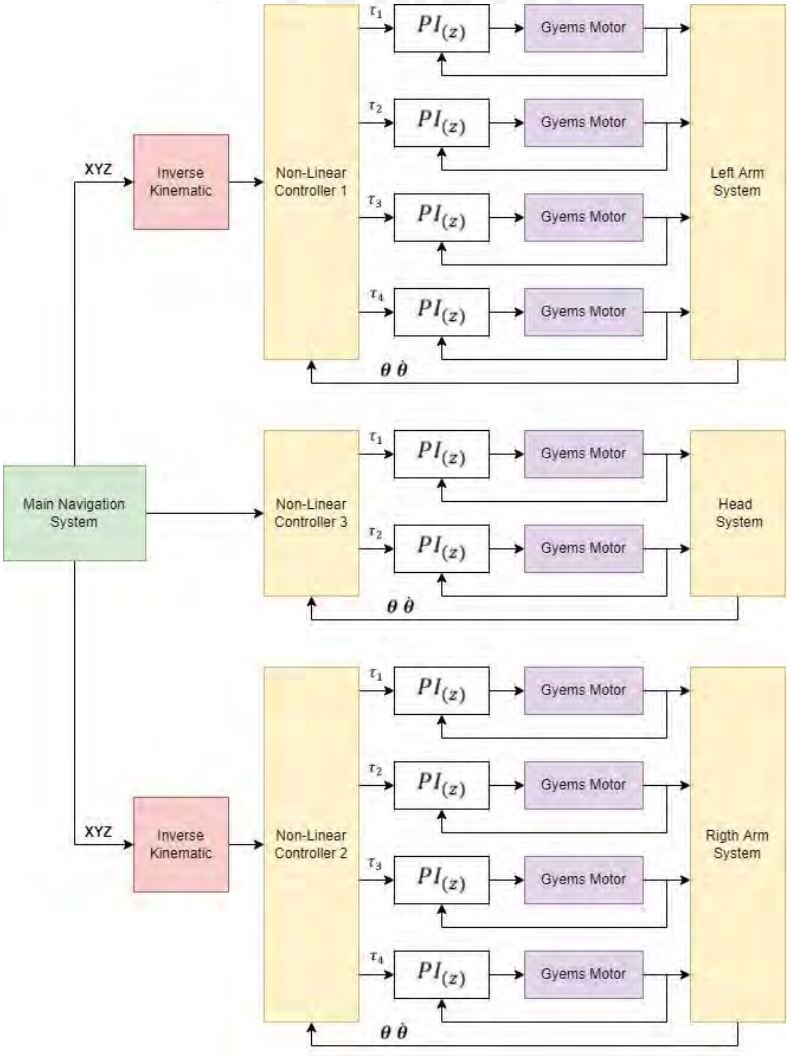


Figura 4.2. Esquema de control para los movimientos del robot teleoperado.

La trama CAN BUS para la posición en un motor GYEMS se muestra en la figura 4.3 la cual depende de los datos de la posición y la velocidad máxima a la que se moverá.

Data field	Description
DATA[0]	Command byte
DATA[1]	Motor temperature
DATA[2]	Torque current low byte
DATA[3]	Torque current high byte
DATA[4]	Speed low byte
DATA[5]	Speed high byte
DATA[6]	Encoder position low byte
DATA[7]	Encoder position high byte

```
void position_control(int32_t GenPos, int16_t velm)
{
                        GenPos =  36000*GenPos;
                        GenPos /= 60;
                        TxData[0] = 0xA4;
                        TxData[1] = 0x00;
                        TxData[2] = velm;
                        TxData[3] = velm >> 8;
                        TxData[4] = GenPos;
                        TxData[5] = GenPos >> 8;
                        TxData[6] = GenPos >> 16;
                        TxData[7] = GenPos >> 24;
}
```

Figura 4.3. Trama CAN para control de posición y su implementación en código abierto.

En esta imagen podemos observar que cuando mandemos la trama con los datos de posición y velocidad máxima, el motor responderá al control y enviará al controlador principal información retroalimentada de temperatura, torque, velocidad actual y posición angular actual mediante el encoder que tiene implementado. Para poder validar esta información, se ha hecho uso de un analizador lógico para visualizar las tramas CAN durante la comunicación. Estas pruebas las podemos visualizar en la Figura 4.4.

Figura 4.4. Trama CAN en Tx / Rx para el Motor 2.

A partir de esta prueba, podemos confirmar la comunicación de la STM32 con los motores GYEMS y la forma en como envían los comandos y se recibe la información. Podemos ver que la comunicación respeta la arquitectura de la trama de datos en CAN.

Otro punto importante es que con las pruebas del analizador lógico es posible medir el tiempo de comunicación por cada secuencia de datos. Por ejemplo, en la Figura 4.4 podemos tomar el tiempo que se demora en enviar el comando y recibir la respuesta de la siguiente manera:

- Tiempo de muestreo 20 ms
- Transmisión = 111.375 us.
- Delay TxRx = 21.75 us.

- Recepción = 109.333 us.
- **TOTAL = 242.455 us << 1ms**

De esta manera podemos asegurar que la comunicación en cada bucle de control entre la STM32 y un motor gyems no supera el tiempo de 500 us lo cual es un tiempo bastante bajo y que nos ayudará a realizar de mejor manera la implementación de la comunicación entre todos los motores y la STM32.

Así, la implementación e interconexión de los equipos para el control de los movimientos a realizar por el robot quedarán definidos tal como se presenta en la figura 4.5.

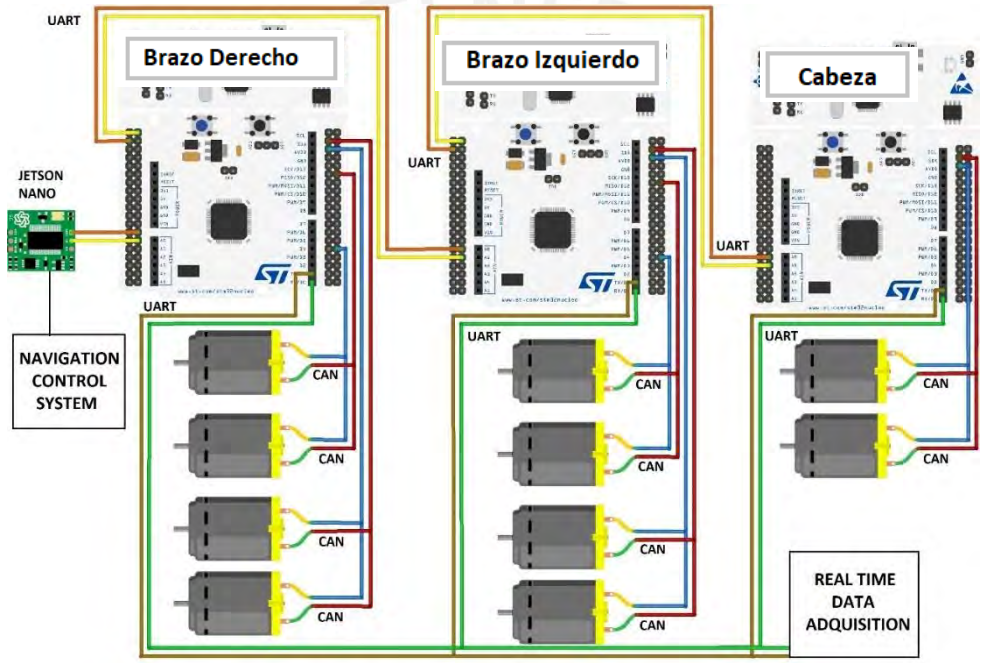


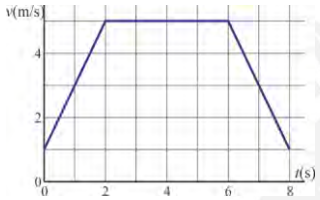
Figura 4.5. Esquema de conexiones para el control del sistema robótico.

Se debe aclarar que los algoritmos de control se están ejecutando en tiempo real dentro de cada uno de los microcontroladores STM32 en comunicación por CANBUS con los servoaccionadores. Por otro lado, para la validación y verificación de datos gráficos que serán presentados posteriormente en este documento de tesis, se está utilizando datos muestreados del lazo de control cerrado mediante el protocolo UART hacia una interfaz como MATLAB.

En el control de posición que tiene implementado los motores Gyems, luego de realizar la sintonización de su controlador PI, podremos obtener una respuesta suave para el posicionamiento del motor. La velocidad a la cual se moverán estos motores durante

el posicionamiento tiene un perfil trapezoidal mostrado en la figura 4.6, en la cual inicia con una aceleración máxima llegando a una velocidad fija y desacelerando rápidamente al llegar a la posición objetivo.

Debido a que esta aceleración es la máxima posible al inicio y final del posicionamiento, se puede aproximar que el control de posición moverá al motor con una velocidad constante igual a la velocidad máxima configurada en la trama CAN. De esta manera será posible encontrar la velocidad necesaria para cada motor en base a la información de la posición inicial, la posición final y el tiempo que se requiere para que el motor se mueva entre estas dos posiciones angulares. Este perfil de velocidad es mostrado en la figura 4.6.




```
void gogogo(int32_t *xfi, int32_t *xff, int32_t tf, int32_t *Vel)
{
    Vel[0] = fabs(xff[0]-xfi[0])/tf;
    Vel[1] = fabs(xff[1]-xfi[1])/tf;
    Vel[2] = fabs(xff[2]-xfi[2])/tf;
    Vel[3] = fabs(xff[3]-xfi[3])/tf;

}
```

Figura 4.6. Perfil y algoritmo de velocidad de motor.

Dado que ya tenemos definidas e implementadas todas las funciones necesarias para realizar los movimientos de brazo, es necesario crear el algoritmo que involucre estas funciones y envíe los comandos necesarios hacia los motores para un movimiento en particular. En la figura 4.7 podremos notar el bucle de control en lazo abierto que se va realizar para llevar al brazo de un punto cartesiano a otro

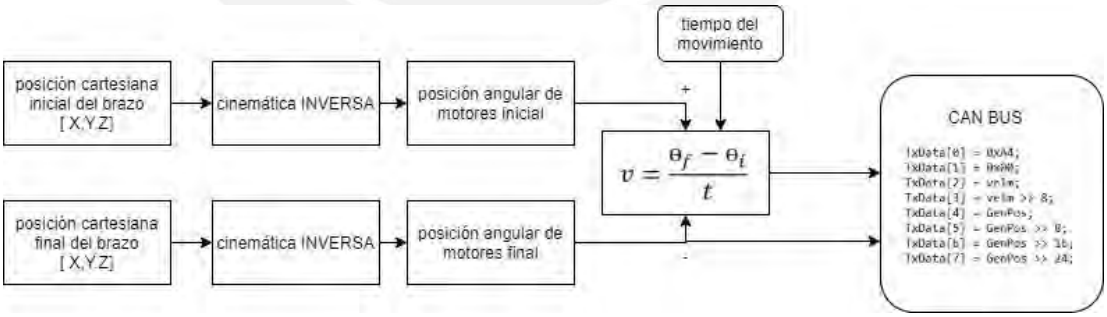


Figura 4.7. Bucle iterativo para el movimiento puntual del brazo robótico.

Se debe notar que el control de los actuadores es independiente e interno de cada motor mientras que el algoritmo en lazo abierto para definir un posicionamiento viene dado por el controlador principal STM32 de acuerdo al bucle iterativo presentado anteriormente y que puede ser aplicado para diferentes puntos cartesianos a los que se quiere llevar el robot.

Los movimientos del robot serán trabajados de acuerdo a las expresiones que se quieran realizar a continuación se muestran algunos ejemplos de algoritmos implementados para estos movimientos.

- **Movimientos punto a punto**

En este caso solo debemos definir los puntos de inicio y fin, así como el tiempo que necesitamos. En la figura 4.8 podemos ver un ejemplo de puntos establecidos para el movimiento, y el algoritmo que utiliza las funciones creadas previamente.

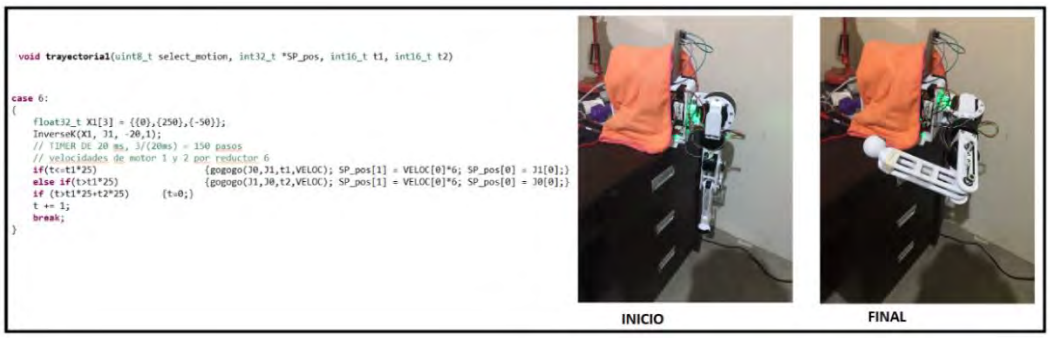


Figura 4.8. Implementación para el movimiento puntual del brazo

- **Movimiento de trayectorias – curvas parametrizadas**

En este caso se generan diversos puntos continuos a partir de un parámetro de frecuencia de las funciones trigonométricas que hará que los puntos generados sean más cercanos entre si haciendo que el movimiento sea más aproximado a la curva definida. En la figura 4.9 se puede visualizar la trayectoria implementada (circulo).

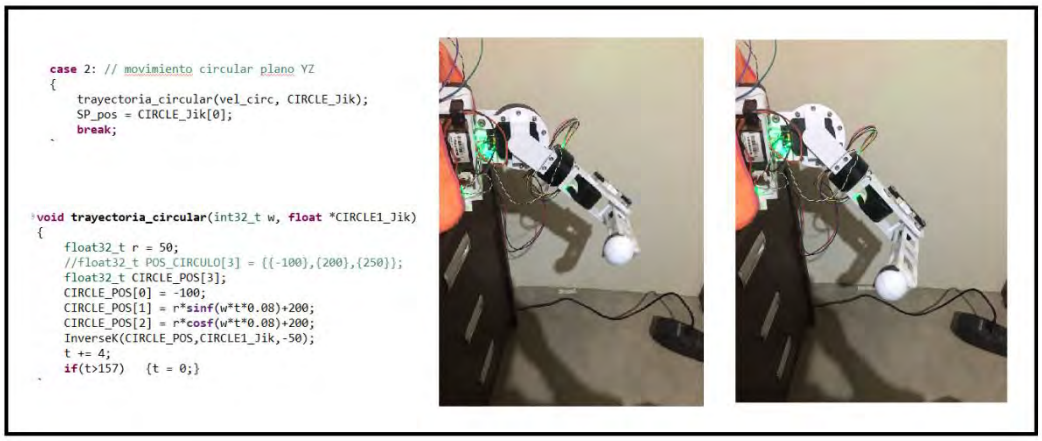
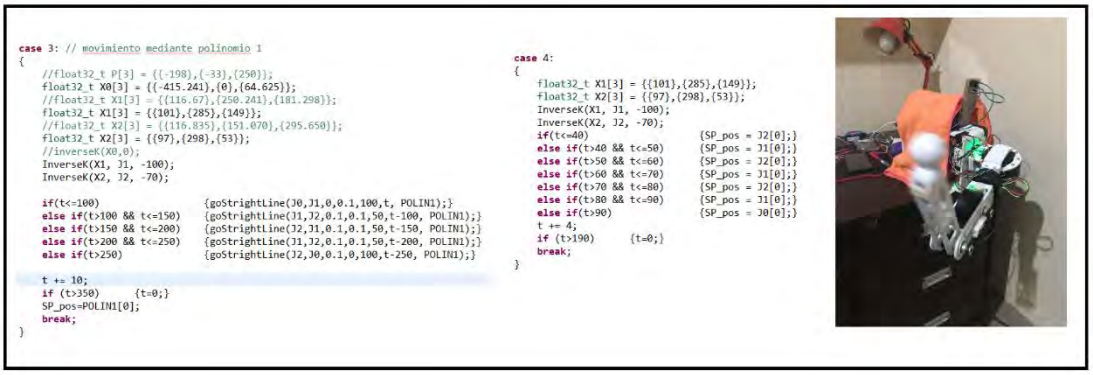


Figura 4.9. Movimiento circular del brazo a partir de una curva parametrizada.

- **Movimiento de trayectorias por polinomios**

En este caso se definen una serie de puntos que están generados a partir de polinomios cúbicos que permiten al brazo seguir trayectorias suaves definiendo puntos de partida y llegada. En la figura 4.10, veremos cómo se ha implementado este algoritmo y su ejecución en tiempo real.



```
case 3: // movimiento mediante polinomio 1
{
    //float32_t P[3] = {{-198},{-33},{250}};
    float32_t X0[3] = {{-415.241},{0},{64.625}};
    //float32_t X1[3] = {{116.67},{250.241},{181.298}};
    float32_t X1[3] = {{101},{285},{149}};
    //float32_t X2[3] = {{116.835},{151.070},{295.650}};
    float32_t X2[3] = {{97},{298},{53}};
    //inverseK(X0,0);
    InverseK(X1, J1, -100);
    InverseK(X2, J2, -70);

    if(t<=100)                  {goStrightLine(J0,J1,0,0.1,100,t, POLIN1);}
    else if(t>100 && t<=150)    {goStrightLine(J1,J2,0.1,0.1,50,t-100, POLIN1);}
    else if(t>150 && t<=200)    {goStrightLine(J2,J1,0.1,0.1,50,t-150, POLIN1);}
    else if(t>200 && t<=250)    {goStrightLine(J1,J2,0.1,0.1,50,t-200, POLIN1);}
    else if(t>250)              {goStrightLine(J2,J0,0.1,0,100,t-250, POLIN1);}

    t += 10;
    if (t>350)      {t=0;}
    SP_pos=POLIN1[0];
    break;
}
```

```
case 4:
{
    float32_t X1[3] = {{101},{285},{149}};
    float32_t X2[3] = {{97},{298},{53}};
    InverseK(X1, J1, -100);
    InverseK(X2, J2, -70);
    if(t<=40)                   {SP_pos = J2[0];}
    else if(t>40 && t<=50)      {SP_pos = J1[0];}
    else if(t>50 && t<=60)      {SP_pos = J2[0];}
    else if(t>60 && t<=70)      {SP_pos = J1[0];}
    else if(t>70 && t<=80)      {SP_pos = J2[0];}
    else if(t>80 && t<=90)      {SP_pos = J1[0];}
    else if(t>90)               {SP_pos = J0[0];}
    t += 4;
    if (t>190)      {t=0;}
    break;
}
```

Figura 4.10. Trayectorias suaves para brazos robóticos desde polinomios continuos.

Así de esta manera, podemos realizar algunos movimientos solo utilizando el análisis cinemático realizado y el control interno de cada motor Gyems el cual puede ser resumido en el esquema de la Figura 4.11.

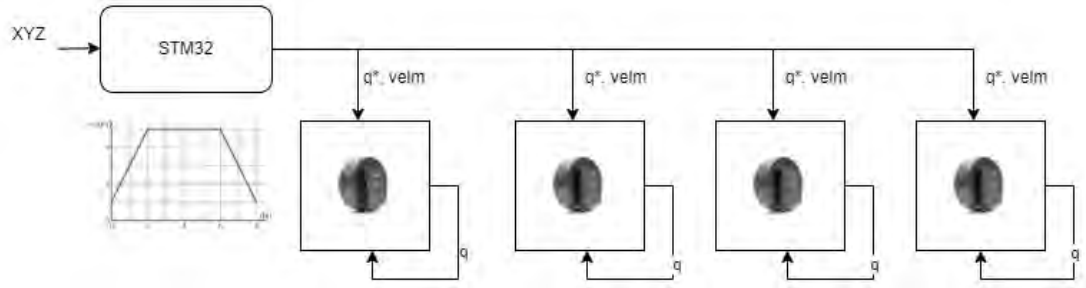


Figura 4.11. Lazo de control descentralizado para un brazo robot.

La desventaja de utilizar este tipo de esquema de control es que solo estamos realizando el control por posición por lo que no tendremos total control de la velocidad a la cual ocurren los movimientos lo cual nos trae algunas limitaciones como se muestra a continuación:

- Cambio de velocidad en los movimientos efectuados.
- lograr realizar trayectorias con puntos intermedios.
- Suavidad en el inicio y final del movimiento dado que en el control de posición se arranca y para a la máxima aceleración posible.

Hasta cierto punto es posible efectuar una solución de estas problemáticas mediante el algoritmo mostrado anteriormente, pero será más efectivo utilizar un control avanzado en lazo cerrado quien nos asegura un control de la posición, velocidad y aceleración en cada articulación.

4.4. Implementación de sistema de control central en lazo cerrado

En el capítulo anterior se ha realizado una comparativa de distintos controladores no lineales diseñados. Todos ellos tienen en común que requiere un conocimiento completo del modelo de sistema por lo que la primera tarea para esta implementación será obtener el modelo completo trabajado en el capítulo anterior.

Para completar el modelo tanto de brazos como de cabeza, es necesario encontrar los coeficientes de fricción adecuados que para el diseño de control se asumieron a valores que nos brinden respuestas muy similares a las reales.

4.4.1. Implementación de control no lineal en STM32

Aquí vamos a implementar un algoritmo de control basado en un código abierto que comande el movimiento de hasta 4 motores a la vez en un tiempo real de ejecución determinado por un muestreo. La interacción entre las variables de control en lazo cerrado se puede visualizar en la Figura 4.12.

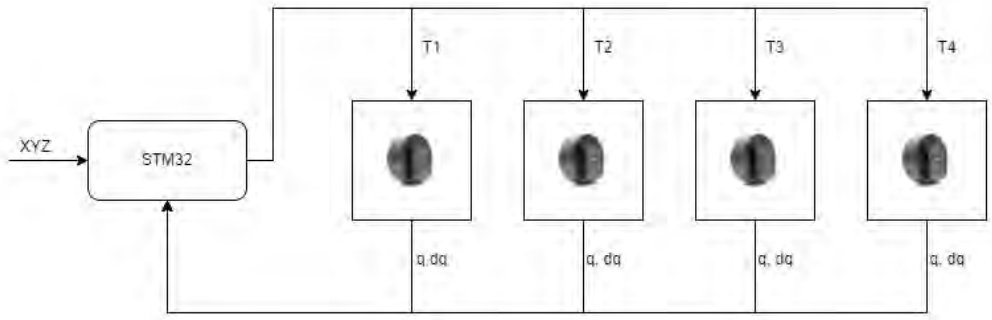


Figura 4.12. Esquema de control en lazo cerrado para un brazo robótico.

Así podemos observar que en cada bucle de control es necesario enviar los torques necesarios y a la vez realizar la recepción de la información de posición y velocidad de cada motor por el protocolo CAN en tiempo real para determinar las decisiones que tomará el control en cada instante de tiempo.

Bajo el esquema presentado anteriormente se ha realizado un algoritmo de control a implementar, mostrado en la figura 4.13, y que puede ser aplicado para todos los controladores no lineales diseñados anteriormente.

Algoritmo: Ejecución de Control No Lineal

Resultado: Señales de torque a aplicar por motores GYEMS.
Cada bucle está temporizado por interrupción a 10 ms
while
  if "writing" es igual a 1
    Generar perfiles de velocidad y posición
    Calcular ley de control y genera torque en N/m.
    Escalar Torque de N/m a Amp.
    Transformar torque eléctrico a señal de comm. CANBUS
    Poner "writing" a 0.
  else
    Solicitar información de posición y velocidad por CANBUS.
    Realizar escalamiento de señales de posición y velocidad
    Aplicar filtro DEMA a retroaliementación de velocidad.
    Poner "writing" a 1.
  endif
end

Figura 4.13. Algoritmo implementado para el control central de del sistema robot.

En el pseudocódigo presentado, se puede visualizar el uso de filtrado de señales para la mejora del control. Recordar que la velocidad de los servomotores está siendo retroalimentada por CANBUS a partir de la derivación de la información del posicionamiento por encoder lo cual conlleva un ruido de medición. Este algoritmo es presentado en el ANEXO D.

También, se puede entender que se utiliza un ciclo recurrente para ejecutar 2 acciones mediante el cambio del bit "writing". La primera acción cuando "writing" está en 0, ejecutará la ley de control no lineal y generará los torques a brindar a cada motor. La segunda acción, cuando "writing" esté en 1, se ejecutará la retroalimentación de la posición y velocidad actual de cada motor. Estas dos acciones se estarán ejecutando siempre en un bucle principal ejecutado por una interrupción por timer, que es una capacidad que brinda el microcontrolador para ejecutar las lógicas de control en un tiempo de muestreo definido que para nuestro caso es 20 ms.

Cada uno de los puntos mostrados en el algoritmo ha sido ejecutado en el microcontrolador, realizando funciones en macros para cada acción y variando para cada prueba, la ley de control a ejecutar. Es necesario entender que el escalamiento

tanto para los datos enviados como para los recibidos por CANBUS deben ser llevados a la misma escala para realizar las comparaciones y obtener los errores que se ingresarán a la ley de control. En la figura 4.14 se muestra el código realizado en la STM32 para el algoritmo utilizando por ejemplo el control Backstepping. Todos los demás controladores podrán ser ejecutados en este mismo algoritmo cambiando únicamente la f

```
if ( writing == 1 )
{
    {
    // Aqui se debe setear el algoritmo para la posiciones o torques
        lazo_cerrado(select_motion_cl, tff, tdd, J111, J11p_ref, J11pp_ref);
        scale_in(PV_ang, PV_vel, q, qp);
        Backstepping(q, qp, J111, J11p_ref, tau1);
        scale_out(tau1, torq);
        torque_control(1, torq[0], TxData1);
        torque_control(2, torq[1], TxData2);
        torque_control(3, torq[2], TxData3);
        torque_control(4, torq[3], TxData4);
    }
    // Envio de los comandos por CAN
    if(!HAL_CAN_AddTxMessage(&hcan1, &TxHeader1, TxData1, &TxMailbox1[0])== HAL_OK)
    if(!HAL_CAN_AddTxMessage(&hcan1, &TxHeader2, TxData2, &TxMailbox1[1])== HAL_OK)
    if(!HAL_CAN_AddTxMessage(&hcan2, &TxHeader3, TxData3, &TxMailbox2[0])== HAL_OK)
    if(!HAL_CAN_AddTxMessage(&hcan2, &TxHeader4, TxData4, &TxMailbox2[1])== HAL_OK)

    writing=0;

}else if(writing == 0)
{
    // Solicitud del feedback multiturn de los motores por CAN
    if(!HAL_CAN_AddTxMessage(&hcan1, &TxHeader1, TxData0, &TxMailbox11[0])== HAL_OK)
    if(!HAL_CAN_AddTxMessage(&hcan1, &TxHeader2, TxData0, &TxMailbox11[1])== HAL_OK)
    if(!HAL_CAN_AddTxMessage(&hcan2, &TxHeader3, TxData0, &TxMailbox22[0])== HAL_OK)
    if(!HAL_CAN_AddTxMessage(&hcan2, &TxHeader4, TxData0, &TxMailbox22[1])== HAL_OK)
    writing=1;
}
```

Figura 4.14. algoritmo implementado en STM32 para cada ciclo de 20ms.

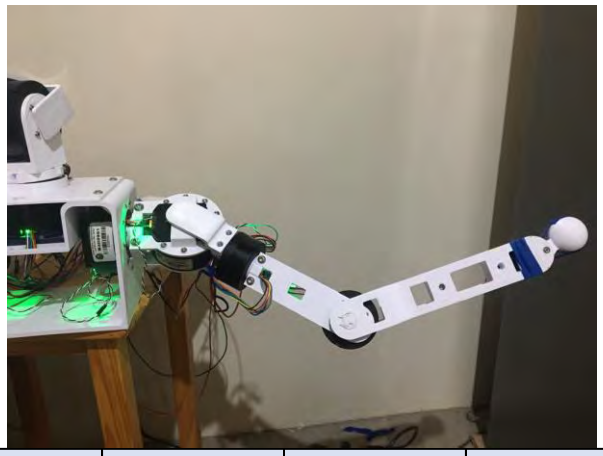
4.4.2. Control PD con compensación de gravedad

Dentro de la STM32 no existe una forma directa de trabajar con matrices, por lo que se están ejecutando primeramente funciones cíclicas para ejecutar operaciones matemáticas con matrices las cuales son recurrentemente utilizadas para la aplicación de la ley de control la cual se puede visualizar en la Figura 4.15.

```
void controlPD(float *q, float *qp, float *q_ref, float *q_refp, float *tau)
{
    float K11 = 0.05;
    float K12 = 0.01;
    float K13 = 0.6;
    float K14 = 0.06;

    float K21 = 10;
    float K22 = 16;
    float K23 = 0.3;
    float K24 = 5;

    float32_t Kd[16] =
    {
            K11, 0,   0,   0,
            0,   K12, 0,   0,
            0,   0,   K13, 0,
            0,   0,   0,   K14
    };

    float32_t Kp[16] =
    {
            K21, 0,   0,   0,
            0,   K22, 0,   0,
            0,   0,   K23, 0,
            0,   0,   0,   K24
    };

    float32_t e[4];
    matrix_sub(q_ref, q, 4, 1, e);

    float32_t ep[4];
    matrix_sub(q_refp, qp, 4, 1, ep);

    float32_t Kdep[4];
    matrix_mult(Kd, ep, 4, 4, 1, Kdep);

    float32_t Kpe[4];
    matrix_mult(Kp, e, 4, 4, 1, Kpe);

    float32_t torque[4];
    matrix_sum(Kdep, Kpe, 4, 1, torque);

    ////////////////////////////////////////////////////////
    // LIMITADOR DE TORQUE DE SALIDA
    float32_t Tmax[4] = { 5, 5, 1, 1};
    for(int i=0; i<=3; i++)
    {
        if          (torque[i]> Tmax[i])    {torque[i]= Tmax[i];
        else if     (torque[i]<-Tmax[i])    {torque[i]=-Tmax[i];
    }
    ////////////////////////////////////////////////////////

    tau[0]=torque[0];
    tau[1]=torque[1];
    tau[2]=torque[2];
    tau[3]=torque[3];

}
```

Figura 4.15. Algoritmo PD con compensación de gravedad.

Los valores para las ganancias del controlador son referenciales para el movimiento de prueba inicial a realizar y que se muestra en la figura 4.16 para el control PD lineal y el control PD con compensación de gravedad.



	Reference	only PD	PD w/comp
GDL1	-60	-54	-60
GDL2	60	53	59
GDL3	-60	-63	-58
GDL4	60	45	60

Figura 4.16. Comparativa de controlador PD lineal y no lineal.

La prueba realizada se hizo para una entrada escalón para cada grado de libertad. En las figuras 4.17 y 4.18 podremos observar la respuesta transitoria de cada articulación para los controladores PD lineal y no lineal respectivamente.

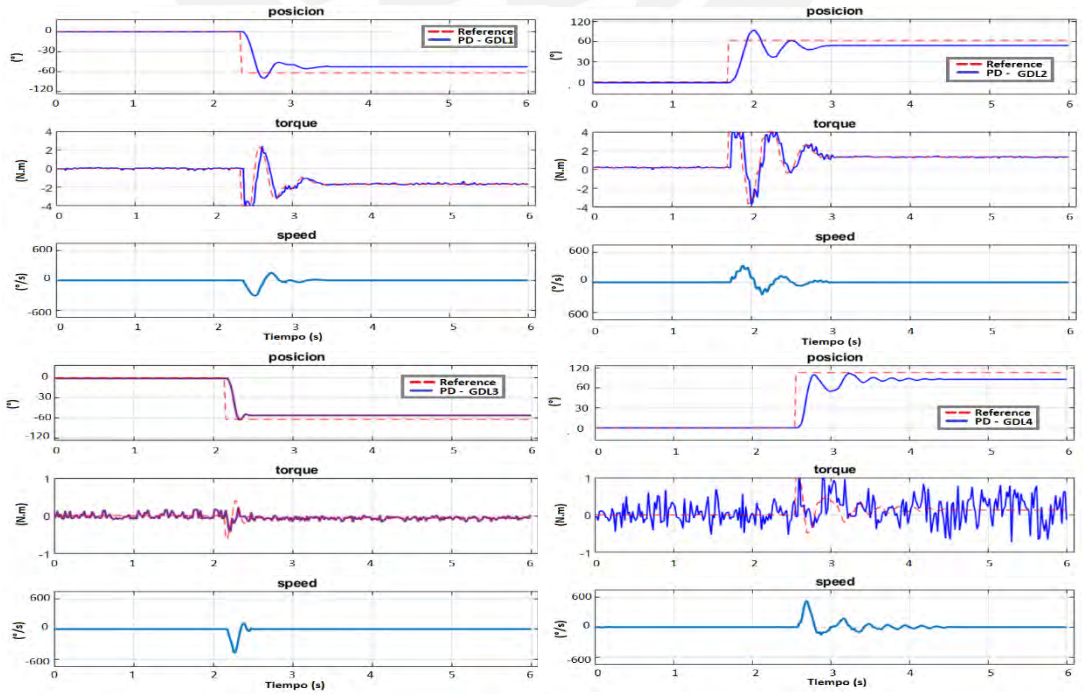


Figura 4.17. Respuesta transitoria ante una entrada escalón con control PD lineal.

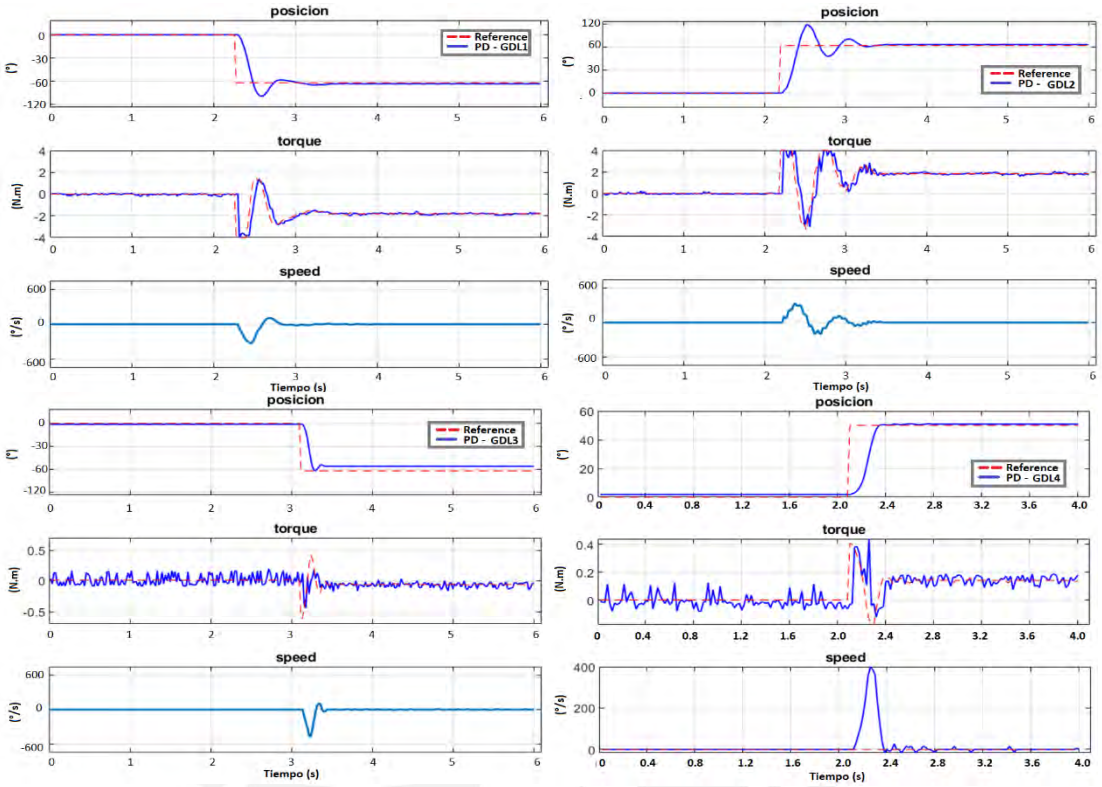


Figura 4.18. Respuesta transitoria ante una entrada escalón con control PD no lineal.

De esta manera se puede visualizar rápidamente que en el caso del controlador PD no lineal corrige de buena manera los errores en estado estacionario sin aportar alguna mejora en el estado transitorio.

Como se mencionó anteriormente, para el uso de este controlador PD con compensación de gravedad, solo es necesario el conocimiento parcial del modelo matemático del sistema lo cual nos facilita de alguna manera su implementación, pero en contraparte la complejidad radica en la selección de sus ganancias las cuales variaran significativamente mientras variamos el punto deseado de equilibrio para el efector final.

Para el uso del controlador Backstepping será necesario el conocimiento de todo el modelo matemático, de lo cual solo nos falta obtener el modelo de fricciones que será explicado a continuación.

4.4.3. Criterio experimental de los coeficientes de fricción.

Se aplicará un método experimental discutido en [37] para hallar los coeficientes de fricción recordando que se ha utilizado un modelo matemático para las fricciones de la forma:

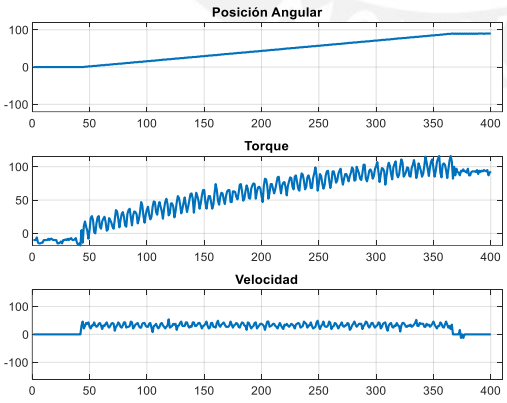
$$F(\dot{\theta}) = f_v\dot{\theta} + f_c sign(\dot{\theta}) \qquad (4.1)$$

La intención es hallar los coeficientes de los vectores f_v y f_c para las fricciones viscosas y de coulomb respectivamente.

El criterio para la experimentación parte de la idea que cuando el motor se mueve a velocidad constante genera un torque necesario para vencer apenas a las fricciones existentes por lo que se puede considerar que, durante el movimiento a velocidad constante, estos torques serán numéricamente iguales.

Es necesario entonces para esto, generar una velocidad constante en los motores y a partir de la toma de datos de velocidades y torques promedios para cada experimento hallar por regresiones lineales, la curva que se aproxime al modelo de fricciones.

Para realizar esto, se puede utilizar el control PD más compensación de gravedad implementado y agregar una generación de trayectoria con perfil trapezoidal que nos permita obtener una velocidad constante durante gran parte del movimiento de la extremidad y luego llevar estos datos a MATLAB para el respectivo análisis tal como se muestra en la Figura 4.19 donde se están captando los valores de posición, velocidad y torque y luego mediante un script adicional tomar las medias de estos 2 últimos datos los cuales nos generan un par cartesiano (velocidad, torque).




```
clc;
k=1;
d_vel=[];
d_tor=[];
while(k<300)
    if(data11(k)>5 && data11(k)<80)
       d_vel = [ d_vel data33(k)];
       d_tor = [ d_tor data22(k)];
    end
    k=k+1;
end
datos_vel = mean(d_vel)
datos_tor = mean(d_tor)

%kk=1; V(kk)=datos_vel; T(kk)=datos_tor;
```

Figura 4.19. Toma de datos de velocidad y torque para una prueba.

La prueba se repite hasta 10 veces para velocidades positivas y 10 veces para velocidad negativas logrando obtener los puntos necesarios para realizar la regresión lineal tal como se muestra en 4.20.

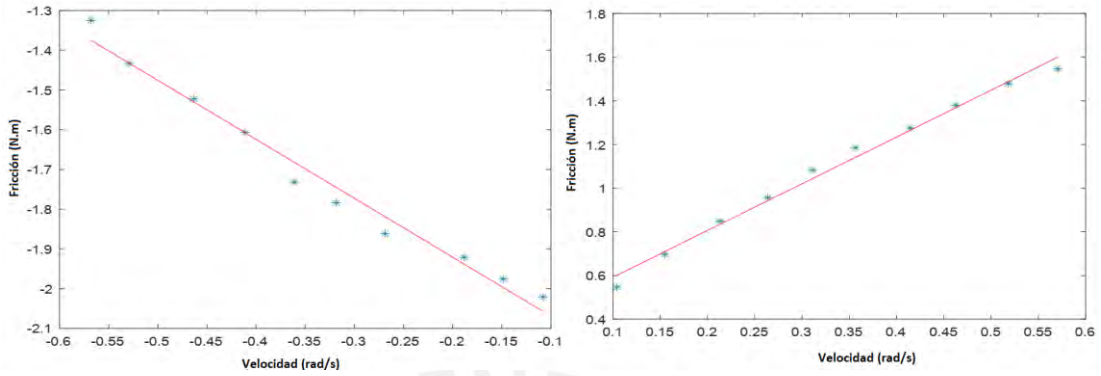


Figura 4.20. Regresiones lineales para datos experimentales.

Ahora con estas rectas al agregarlas a un solo gráfico, cumplirán con la ecuación del modelo de fricciones propuesto inicialmente y así podremos hallar directamente los parámetros f_v y f_c necesarios para obtener el modelo completo de brazos y cabeza. En la Figura 4.21 se muestran los mapas de fricciones obtenidos para cada motor de un brazo y desde donde podemos obtener fácilmente los coeficientes requeridos. Se entiende que los experimentos serán similares para el brazo siguiente y la cabeza.

Recodar que estos coeficientes variarán de acuerdo a si la velocidad es positiva o negativa tanto los coeficientes de fricción viscosa y coulomb. El detalle de los valores encontrados los podremos hallar también en la tabla 4.1.

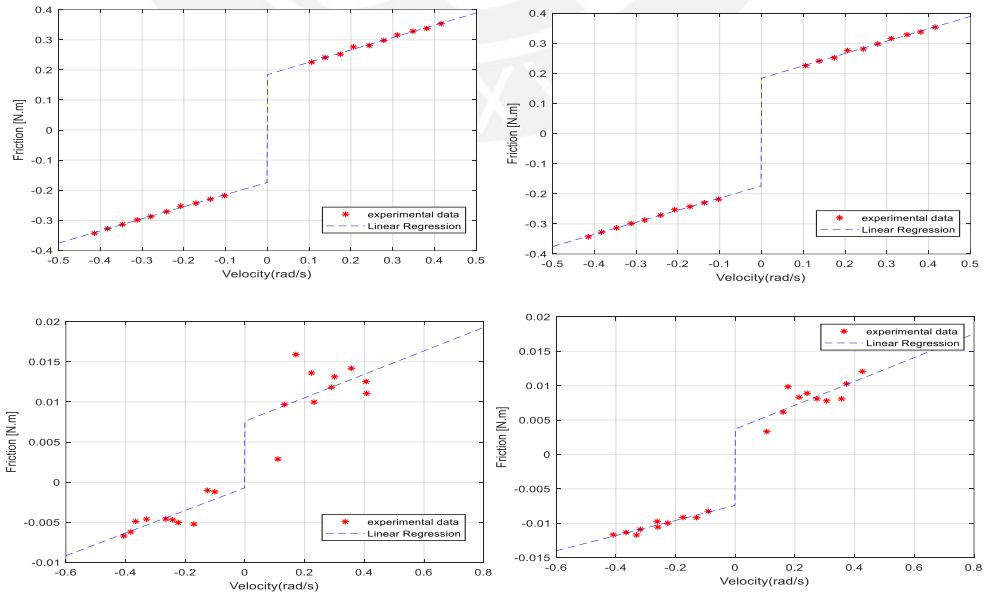


Figura 4.21. Mapa de fricciones experimentales para cada Motor GYEMS.

Tabla 4.1. Coeficientes de fricción experimentales del modelo de brazo robot.

V+/V-		V+/V-	
an	**an-**	**bn**	**bn-**
0.4106	0.4021	0.1842	-0.1742
0.3924	0.3834	0.2044	-0.3631
0.0146	0.0141	0.0076	-0.0007
0.0173	0.011	0.0037	-0.0074

En este punto de los experimentos, se puede apreciar que los motores 1 y 2 mantiene un perfil de velocidades muy similar y lo mismo para los motores 3 y 4. Esto se debe a que los primeros 2 motores son del mismo modelo de motores **Gyems RMD-X8 Pro** con reductor de 6:1 mientras que los motores 3 y 4 son motores Gyems del modelo RMD L-70 de menor capacidad de torque y sin reductor. Cabe mencionar, que este último modelo de motores trae consigo una dificultad mayor de ruido de medición para la lectura de la velocidad lo cual se ve reflejados en los puntos experimentales para la regresión realizada y demostró la dificultad que se podría tener para el control de estos motores.

Como se mencionó en el capítulo 2, la eficiencia del control a implementar radica en la proximidad de modelo matemático a la respuesta experimental del sistema. Debido a esto, se ha definido una metodología para asegurar los mejores coeficientes de fricción que complementan el modelo encontrado. Para la validación del modelo se está utilizando el índice de desempeño mostrado en la ecuación 4.2.

$$E = \sqrt{\frac{\sum(e_\theta^2)}{\sum r_\theta^2}} \qquad (4.2)$$

Donde $\boldsymbol{e_\theta}$ es el error o diferencia entre la respuesta simulada y la respuesta experimental y $\boldsymbol{r_\theta}$ es la respuesta experimental. Las pruebas realizadas fueron para la posición Zero, para una posición inicial diferente de 0, y para la aplicación de un torque de 2 N.m para el grado de libertad (GDL) 2. Los resultados de estas pruebas se pueden visualizar en la siguiente tabla 4.2.

Tabla 4.2. Porcentajes de error de modelo para pruebas realizadas con coeficientes ideales y experimentales.

Situación	Experimento	E_{GDL1}	E_{GDL2}	E_{GDL3}	E_{GDL4}
Coeficientes de fricción ideales	P. Inicial en [0°,0°,0°,0°]	9.2%	9.5%	23.1%	9.1%
	P. Inicial en [-60°,45°,30°, 60°]	8.4%	9.3%	21.7%	7.6%
	Torque de 2N.m en GDL2	9.2%	8.4%	33.3%	8.1%
Coeficientes de fricción experimentales	P. Inicial en [0°,0°,0°,0°]	2.4%	5.1%	9.6%	2.3%
	P. Inicial en [-60°,45°,30°, 60°]	3.2%	2.3%	8.5%	3.7%
	Torque de 2N.m en GDL2	4.1%	4.3%	9.1%	4.3%

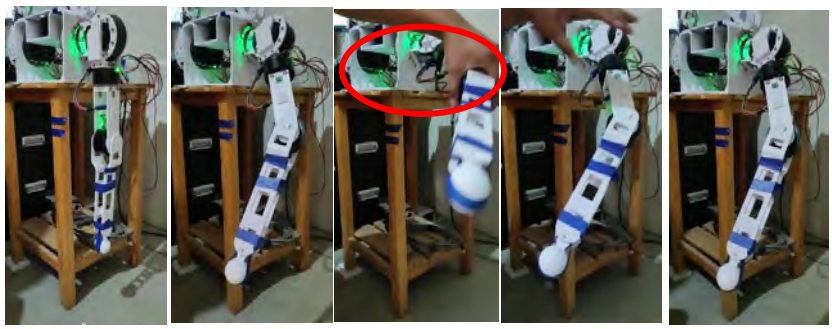
Se puede observar que los errores del modelo se reducen cuando tomamos los coeficientes de fricción calculados experimentalmente, mientras que para los coeficientes de fricción tomados inicialmente los errores son mayores. Esto es debido a que al principio solo se tomó en cuenta una respuesta que sea razonable para nuestras pruebas simuladas. Sin embargo, con las pruebas experimentales se encontró un modelo matemático en el cual hace que los errores no superen el 10% de error para cada grado de libertad y en distintas situaciones experimentales,

Ahora que hemos obtenido el modelo matemático completo es necesario llevar las ecuaciones tanto del modelo como del control diseñado a la STM32 en código abierto.

4.4.4. Implementación del Control Backstepping

Ahora que tenemos todos los parámetros validados para el modelo matemático es posible ejecutar el algoritmo Backstepping y en este caso, solo se reemplazó el algoritmo PD con gravedad por este nuevo algoritmo.

Para comprobar la efectividad de control implementado se realizó una primera prueba con cambios de ángulos con entrada escalón y luego aplicando una perturbación al sistema [38]. En la Figura 4.22 se observan las pruebas realizadas aplicando una perturbación en cierto momento de la ejecución del movimiento realizado por el brazo.



(a) (b) (c) (d) (e)

Figure 4.22. Movimiento de brazo izquierdo a [-30°; 30°; 60°; 30°]. (a) posición ZERO; (b) posición deseada; (c) perturbación aplicada y resaltada; (d) regulación posterior a la perturbación; (e) posición final del brazo.

Aquí se puede observar que el brazo llega a la posición deseada y segundos después se aplica una fuerza externa que cambia la posición y el algoritmo Backstepping se encarga, posterior a la perturbación, de reposicionar al brazo a la posición solicitada [39]. En la Figura 4.31 podremos observar, a través de los datos recopilados desde una interfaz UART en MATLAB, la posición en simultáneo de cada motor del brazo.

Es importante mencionar nuevamente que los resultados presentados son datos obtenidos mediante comunicación serial UART hacia MATLAB desde las STM32 y para evitar retardos de comunicación que afecten negativamente al control aplicado, se procedió a enviar datos de un motor a la vez por experimento. Así, los resultados presentados en el presente trabajo para cada motor, son parte de varios experimentos en las mismas condiciones que permitan obtener datos de los motores uno por uno.

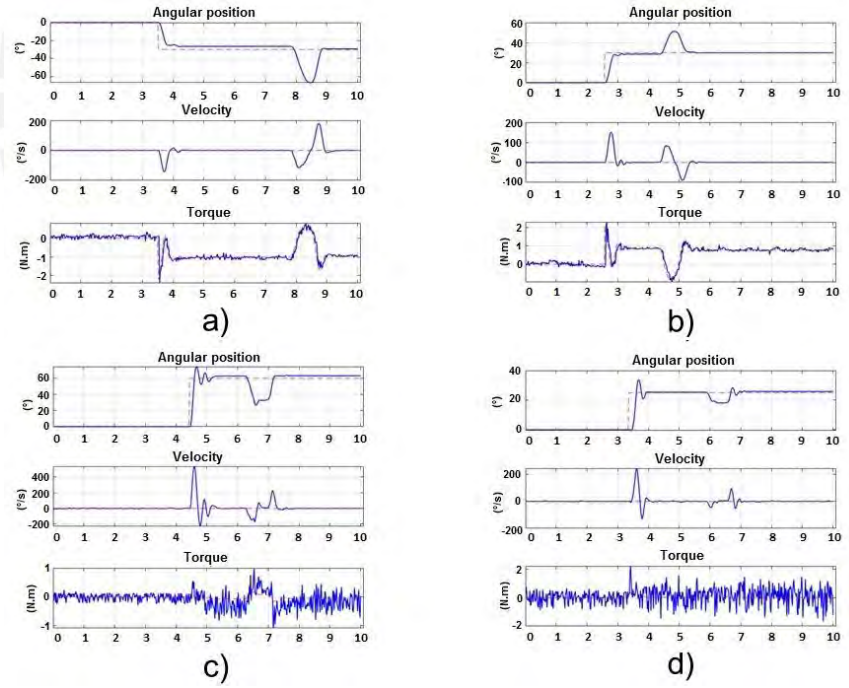


Figura 4.23. Regulación de posición de brazo izquierdo con perturbación. (a) Motor1 de 0° a -30°; (b) Motor2 de 0° a 30°; (c) Motor3 de 0° a 60°; (d) Motor4 de 0° a 30°.

Al comprobar la efectividad del control Backstepping en la regulación de la posición, se ha aplicado la generación de trayectorias con perfiles trapezoidales para llevar la muñeca del brazo hacia un nuevo punto diferente en el espacio cartesiano y regresarlo a la posición ZERO. Esta segunda prueba se realizó aplicando una perturbación

durante el estado de reposo en el punto deseado del movimiento. En la Figura 4.24 se puede observar los movimientos realizados por el brazo.

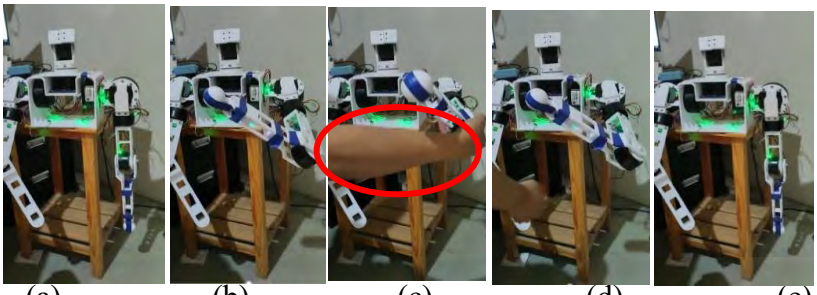


Figura 4.24. Regulación Backstepping con perfil trapezoidal de velocidad. (a) Posición ZERO inicial; (b) Posición deseada; (c) Perturbación aplicada y resaltada. (d) Regulación posterior a la perturbación. (e) Retorno a la posición ZERO.

De esta manera, al aplicar una generación de trayectoria, vemos que es posible controlar la forma en la cual el brazo puede llegar a la referencia deseada, realizando un movimiento más natural en términos de velocidad debido a que se está aplicando, gracias a la generación de trayectorias, una referencia de posición y velocidad al mismo tiempo y el controlador se encarga de regular estos valores de consigna los cuales se pueden visualizar en la Figura 4.25.

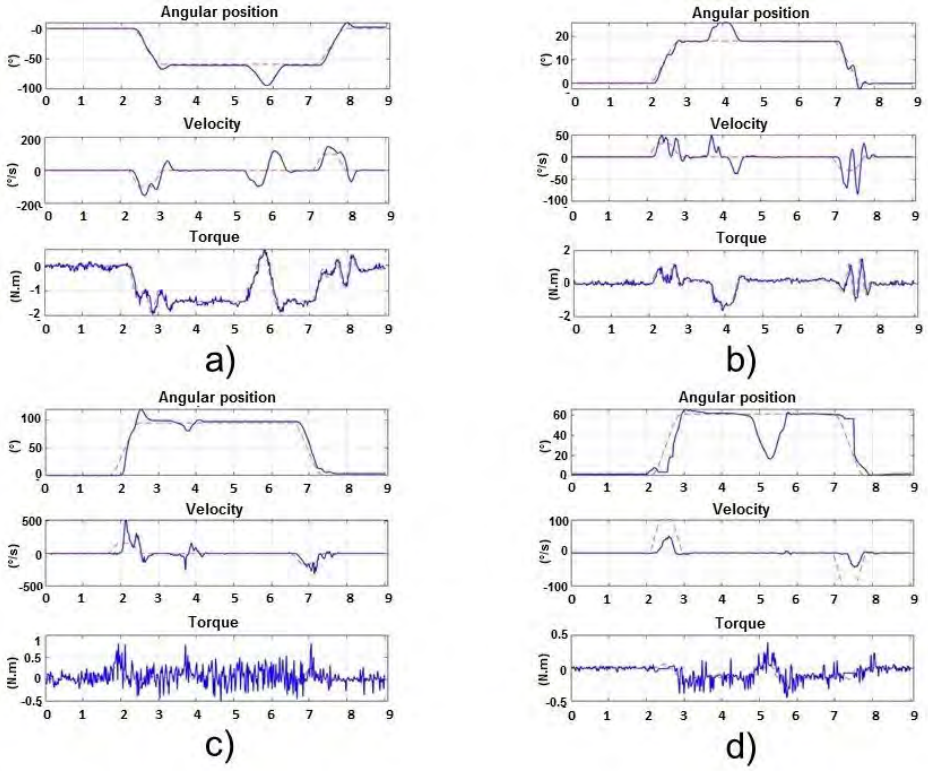
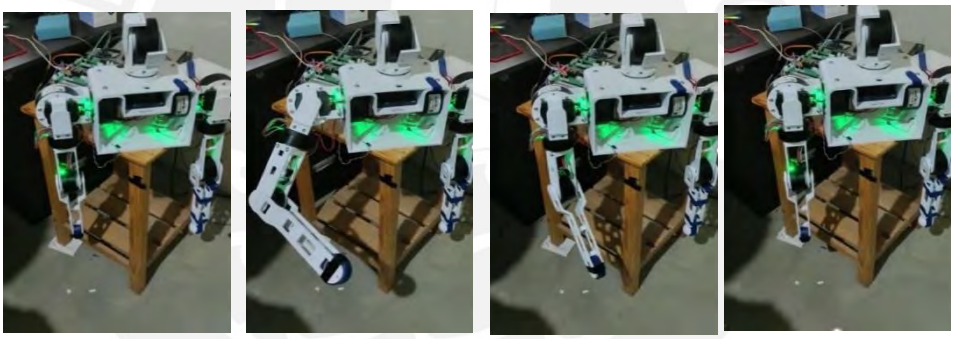


Figura 4.25. Regulación de posición y veloc. con posterior perturbación. (a) Motor 1 de 0° a -59°; (b) Motor 2 de 0° a 18°; (c) Motor 3 de 0° a 61°; (d) Motor 4 de 0° a 97°.

En este experimento se puede ver que para los motores 1 y 2 hay mejor control que para los motores 3 y 4, y esto por el ruido de medición que conllevan estos dos motores, pero de todas maneras se parecía cierto control. La perturbación aplicada en estado estable también nos permite confirmar que la regulación para todos los motores del brazo se está realizando siempre de manera exitosa.

Por otro lado, al realizar el experimento en otro punto cartesiano, estamos verificando la expansión del controlador en un espacio de trabajo con un ligero cambio en las ganancias del controlador. Esta última prueba motivó a realizar una nueva experiencia para un punto diferente pero esta vez aplicando movimientos más suaves mediante el uso de trayectorias polinómicas de grado 3 ahora al brazo derecho. De esta forma se pudo observar un movimiento más suave del brazo para llegar al punto deseado. La Figura 4.26 a 4.27 confirman estos resultados debido a que los torques solicitados son mucho menores que las trayectorias trapezoidales. De esta manera se ha podido comprobar la efectividad del controlador Backstepping para ambos brazos.



(a) (b) (c) (d)

Figura 4.26. Fotograma de movimiento del brazo derecho. (a) Posición ZERO; (b) posición deseada; (c) retorno de brazo a posición zero; (d) posición final.

Finalmente, debido a los resultados obtenidos con los brazos del robot teleoperador, se ha realizado este mismo control para el sistema de la cabeza que como hemos mencionado, tiene 2 grados de libertad. En la Figura 4.28 y 4.29 podremos visualizar las pruebas finales realizadas a la cabeza del robot con el control Backstepping y la generación de trayectorias trapezoidales.

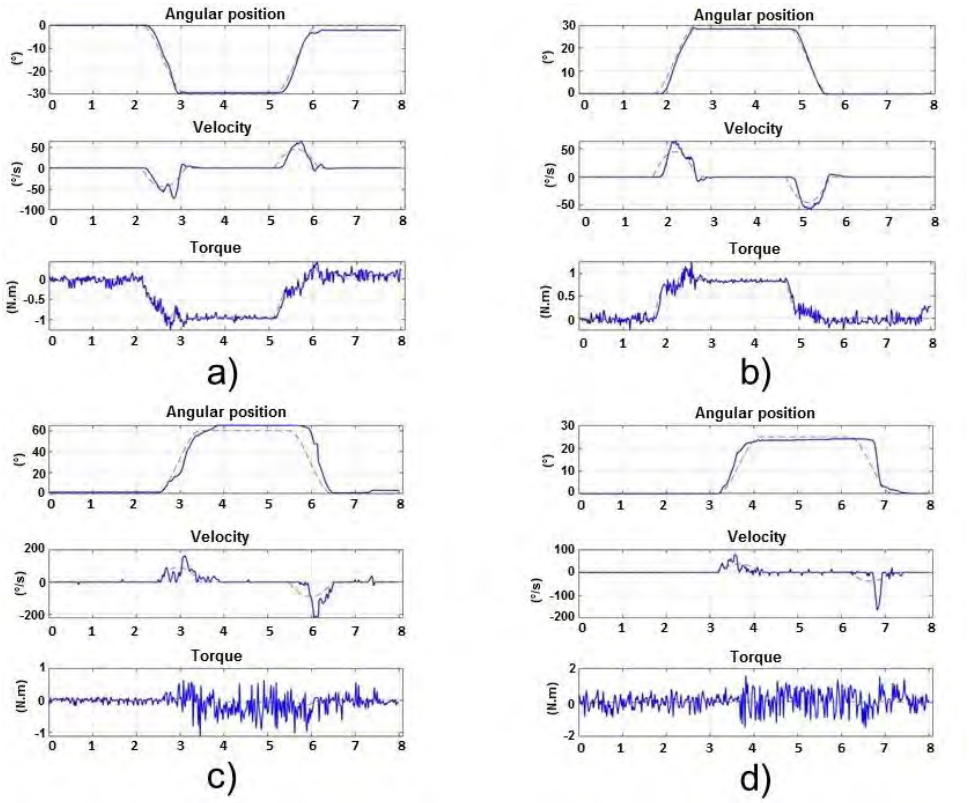
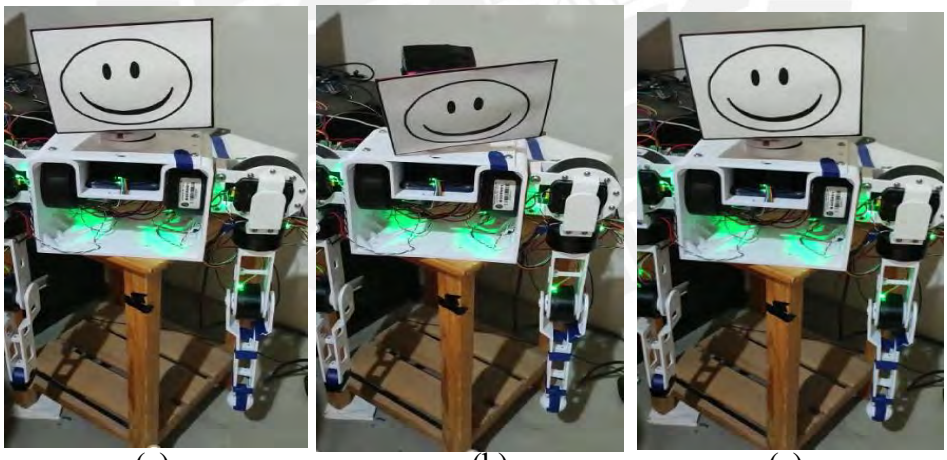


Figura 4.27. Mov. de brazo derecho mediante gen. de trayec. Polinomial. (a) Motor 1 de 0° a -30°; (b) Motor 2 de 0° a 30°; (c) Motor 3 de 0° a 60°; (d) Motor 4 de 0° a 25°.



(a) (b) (c)

Figura 4.28. Movimiento puntual y retorno del protipo de la cabeza del robot teleoperado. (a) Posición ZERO; (b) Posición deseada [20°;30°]; (c) Posición final del movimiento.

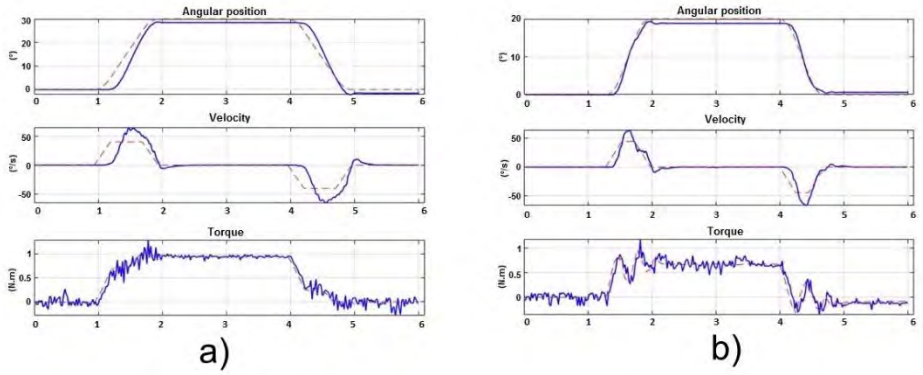


Figura 4.29. Posición, velocidad y torque de cada motor de la cabeza de robot teleoperador. (a) Motor 1 de 0° a 30°; (b) Motor 2 de 0° a 20°.

4.4.5. Análisis de los resultados obtenidos

A raíz de los distintos experimentos realizados en el prototipo de este proyecto se ha podido comprobar la efectividad de controladores no lineales como el Backstepping ante un controlador lineal adaptado parcialmente como el control PD con compensación de gravedad. Para este punto, además de los resultados ya presentados se muestra la Tabla 4.3 con una comparativa de las ganancias de los controladores para los movimientos realizados por el brazo Izquierdo, obteniéndose resultados similares para el brazo derecho y la cabeza.

Tabla 4.3. Valores de ganancias para los controladores no lineales en diferentes movimientos del brazo izquierdo.

	PD control with gravity			**Backstepping Control**		
	step	**spline**	**polinomial**	**step**	**spline**	**polinomial**
Kp11	5.2	9.6	3.2	15	15	15
Kp22	5.1	10.3	3.4	15	17	15
Kp33	0.5	0.87	0.22	120	118	120
Kp44	0.35	0.95	0.19	120	120	120
Kv11	1.1	2.2	0.6	6	13	8
Kv22	1.2	0.19	0.4	6	10	4
Kv33	0.04	0.07	0.05	6	4	2
Kv44	0.06	0.09	0.06	5	6	4

En el caso de control PD con compensación de gravedad se ha podido ver que un mínimo cambio en los valores de las ganancias, provocan respuesta de movimientos muy diferentes para las mismas condiciones del experimento hasta el punto de llevar al sistema, en algunos casos, a la inestabilidad. Esto complica aún más la selección de las ganancias para este controlador los cuales deben cambiar a menudo dependiendo

del movimiento que se quiera realizar. Esto nos hace pensar en una solución adaptativa para el cambio automático de estas ganancias y de esta manera generalizar el controlador para un espacio de trabajo.

En el caso del control Backstepping, las ganancias iniciales seleccionadas se han podido generalizar para todos los movimientos experimentados por el brazo, teniendo que realizar un ajuste mínimo sobre todo para los motores 3 y 4. Incluso, el rango de valores que se pueden utilizar para las ganancias de este controlador es grande, en el sentido que una mínima variación de las ganancias no afectará significativamente la respuesta del control, facilitando la selección e implementación de dichas ganancias.

De esta manera, la implementación también nos comprueba que la selección del controlador Backstepping, como controlador final, fue la más apropiada para los alcances del proyecto donde se solicitaba un control total sobre la naturaleza en los movimientos de las extremidades del robot teleoperado.

5. CONCLUSIONES

- Se ha desarrollado la conexión de hasta 4 servomotores por el protocolo CANBUS a los microcontroladores STM y se han realizado las pruebas de envío de comandos en simultáneo comprobando la eficiencia de estos motores para llegar a todos los puntos del área y condiciones de trabajo posteriormente propuestas.
- Se ha logrado obtener un modelo matemático aproximado de cada sistema mecánico del robot el cual se ha utilizado para la simulación de los controladores diseñados. Las pruebas posteriores ayudaron a realizar los ajustes necesarios a los modelos y así lograr una mejor aproximación de los resultados teóricos y de esta manera mejorar la repuesta del control no lineal basado en este modelo matemático.
- Con la implementación de un control PD no lineal, se ha logrado aplicar, mediante pruebas en lazo cerrado, un procedimiento para obtener los parámetros de fricción en cada articulación que complementa satisfactoriamente el modelo matemático y logra que nuestro controlador no lineal basado en este modelo, reduzca significativamente los errores de seguimiento de trayectorias.
- Se ha logrado establecer métodos para verificar y validar tanto los modelos como los controles propuestos en esta memoria de tesis lo cual permitió tomar decisiones, durante la etapa de diseño del control, para la selección de componentes a utilizar en la etapa de implementación.
- En la etapa de diseño se pudo observar la aplicación de diferentes controladores no lineales al modelo matemático del sistema mecánico del robot, lo cual nos permitió observar ventajas y desventajas de cada uno de estos controles, Se debe entender que los controladores Backstepping y SMC obtuvieron similar performance, pero su grado de complicación en la implementación fue lo que definió al control Backstepping como el control más apropiado para esta aplicación.
- Debido a que se ha propuesto distintos movimientos para la aplicación de esta memoria de tesis, se ha podido lograr implementar distintos tipos de generadores de trayectorias y realizar las comparaciones respectivas.

También nos permitió realizar sintonizaciones para cada controlador donde se pudo observar que solo en el caso del controlador Backstepping, los parámetros de sintonización no son tan sensibles a pequeños cambios como en el caso de los otros controladores y así nos permite llegar a una sintonización rápida de los controles. Esto refuerza también la selección de control Backstepping como el control final de nuestra aplicación.

- La implementación del control no lineal en el prototipo del proyecto nos ha permitido también comprobar la efectividad que tiene el uso de diferentes filtros de ruido en las señales de medición y una comparativa que nos asegure el uso de un correcto índice de filtrado. De esta manera, se pudo comprobar por experimentación que los motores 3 y 4 necesitaron de un filtrado de la señal de velocidad de hasta un 30% para mejorar las pruebas realizadas en lazo cerrado, así como también se mostró la aplicación e implementación de filtros de dos etapas (DEMA) para minimizar los retardos provocados por este proceso de filtración de señales.

6. RECOMENDACIONES

El desarrollo del control en lazo cerrado del sistema mecánico del robot presentado, mediante un control Backstepping y un generador de trayectorias nos ha permitido la generación de distintos movimientos que permitan incluso realizar una rutina de gestos. La variedad de estos movimientos ha demandado que los controladores usados deban cambiar constantemente sus ganancias para obtener un mejor performance en cada movimiento.

El presente trabajo de tesis permite pensar en un algoritmo adicional que se encargue de obtener los mejores valores de ganancias de controlador para cada tipo de rutina ejecutada. Por otro lado, ya que el modelo matemático hallado para nuestro sistema no siempre puede ser el más acertado, es interesante pensar también en otros tipos de algoritmos que permiten sobrellevar este sesgo que conlleva el modelo encontrado.

De esta manera es posible proponer un algoritmo Adaptativo o la combinación de los controles utilizados en esta memoria de tesis como un nuevo desarrollo que permita mejorar aún más la performance de movimientos del sistema robótico.

Por otro lado, el prototipo de robot realizado para esta memoria de tesis, será una base para diferentes pruebas adicionales que se quieran realizar para la propuesta de nuevas formas de generar curvas de seguimiento o para el tratamiento de señales analógicas que se usen para un futuro control.

BIBLIOGRAFÍA

[1] Organización Mundial de la Salud (28 de noviembre de 2019). *Trastornos Mentales.* https://www.who.int/es/news-room/fact-sheets/detail/mental-disorders.

[2] Murphy R., Gandudi M., Adams J. (2020). *Application of Robot for COVID-19 Response.* https://arxiv.org/pdf/2008.06976.pdf.

[3] Chulalongkorn University (27 de Marzo de 2020). *Thailand hospital use 'ninja robots' to fight coronavirus.* https://www.chula.ac.th/en/clipping/28858/.

[4] Servicio de Salud Metropolitano Sur Oriente – Chile (20 de mayo de 2020). *Hospital Padre Hurtado implementa el primer robot en Chile para apoyar atención de paciente en tiempo de Covid-19.* https://redsalud.ssmso.cl/hospital-padre-hurtado-implementa-el-primer-robot-en-chile-para-apoyar-atencion-de-pacientes-en-tiempos-de-covid-19/.

[5] Fossati M., Catalano M., Carbone M., Lentini G., Caporale D., Grioli G., Poggiani M., Maimeri M., Barbarossa M., Petrocelli C., Vivani M., Calderine C., Carrozi L., Ferrari M., Bicchi A. (2020). *LHF Connect: A DIY Telepresence Robot Against COVID-19.* Strategic Design Research Journal 13(3).

[6] Publication *User Manual The Personal Robot TEMI. (2019).* USA.

[7] J. M. Beer, K. R. Liles, X. Wu and S. Pakala, "Affective human-robot interaction" in Emotions and affect in human factors and human-computer interaction., Elsevier, pp. 359-381, 2017.

[8] MOON Y, NASS C. How "Real" Are Computer Personalities?: Psychological Responses to Personality Types in Human-Computer Interaction. Communication Research. 1996;23(6):651-674. doi:10.1177/009365096023006002

[9] Li, J., & Chignell, M. (2010). Communication of Emotion in Social Robots through Simple Head and Arm Movements [Ebook] (1st ed.). Retrieved from http://DOI 10.1007/s12369-010-0071-x

[10] Cooper S., Di Fava A., Vivas C., Marchionni L., Ferro F. (2020). *ARI: the Social Assitive Robot and Companion.* IEEE International Conference on Robot and Human Interactive Communication (RO-MAN). https://doi.org/10.1109/RO-MAN47096.2020.9223470.

[11] Salichs M., Barber R., Khamis A., Malfaz M., Gorostiza J., Pacheco R., Rivas R., Corrales A., Delgado A., García D. (2006)- *Maggie: A Robotic Platform for Human-Robot Social Interaction.* IEEE Conference on Robotics, Automation an Mechatronics. https://doi.org/10.1109/RAMECH.2006.252754.

[12] Al barakeh Z., Alkork S., Karar A., Said S. y Beyrouthy T. (2019). *Pepper Humanoid Robot as a Service Robot: a Customer Approach.* 3rd International Conference on Bio-engineering for Smart Technologies (BioSMART). https://doi.org/10.1109/BIOSMART.2019.8734250.

[13] Arce, D.; Jibaja, S.; Urbina, F. Maura, C.; Huanca, D.; Paredes, R.; Cuellar, F.; Perez-Zuñiga, G. Design and preliminary validation of social assistive humanoid robot with gesture expression features for mental health treatment of isolated patients in hospitals. In Proceedings of the 14th International Conference Social Robotics, ICSR, Florence, Italy, 13–16 December 2022.

[14] (Setiembre 2015). El Binomio Pterodáctilo y Spider Minero. *Revista de la sociedad nacional de minería, petróleo y energía "desde adentro",* (145), 104-105.

[15] Osinergmin. (2016). 10 MEJORES TRABAJOS DE INVESTIGACIÓN Y TECNOLOGÍA MINERA (pp. 116 - 118). Miraflores, Lima, Perú: Organismo Supervisor de la Inversión en Energía y Minería.

[16] Lopez A., Cuellar F. (2017). *ROBOTMAN: Security Robot for Human-robot Interaction Inside Malls.* Companion of the 2017 ACM/IEEE International Conference. https://doi.org/10.1145/3029798.3036653.

[17] Pumaricra Rojas, D., Noack, M., Reger, J., & Pérez-Zúñiga, G. (2022). State Estimation for Coupled Reaction-Diffusion PDE Systems Using Modulating Functions. Sensors, 22(13), 5008. MDPI AG.

[18] A. Tellaeche, J. Kildal, I. Maurtua, A flexible system for gesture based human–robot interaction Procedia CIRP, 72 (2018), pp. 57-62

[19] Gomez-Quispe JM, Pérez-Zuñiga G, Arce D, Urbina F, Gibaja S, Paredes R, Cuellar F. Non Linear Control System for Humanoid Robot to Perform Body Language Movements. Sensors. 2023; 23(1):552. https://doi.org/10.3390/s23010552

[20] Craig, J., & Vidal Romero Elizondo, A. (2006). Robótica. México: Pearson Educación.

[21] J. Denavit y R.S. Hartenberg, "A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices", Journal of Applied Mechanics, pp. 215-221, junio de 1955.

[22] SPONG. (2019). ROBOT MODELING AND CONTROL 2EDITION. JOHN WILEY.

[23] PETER CORKE (2015). Robotics Toolbok for MATLAB. Release 9.10.

[24] Mohammed, S. (2016). Kinematic Motion Planning for a 7-AxisRobotic Arm (LWA70 by Schunk) (Dissertation).

[25] Shimizu, M., Kakuya, H., Yoon, W., Kitagaki, K., Kosuge, K., October 2008. Analytical inverse kinematic computation for 7-dof redundant manipulator with joint limits and its application to redundancy resolution. IEEE Transactions on Robotics 24 (5), 1131 – 1142.

[26] M. Luzardo, M. Karppa, J. Laaksonen, T. Jantunen, "Head pose estimation for sign language video," in J.-K. Kamarainen and M. Koskela (eds.), Image Analysis. Springer, Lecture Notes in Computer Science, Vol. 7944, pp. 349–360, 2013.

[27] K.S Fu, R.C. Gonzáles y C.S.G. Lee, "Dinamica del Brazo del Robot", en: *Robótica, Control, Detección, Visión e Inteligencia,* McGraw-Hill/Interamericana, Madrid 1988.

[28] BARRIENTOS, ANTONIO (2007) Fundamentos de Robótica. Madrid: McGraw-Hill

[29] Coulomb and Viscous Friction. (2009). Simulink - MATLAB Simulink. https://la.mathworks.com/help/simulink/slref/coulombandviscousfriction.html

[30] A. Perrusquia, J. A. Flores-Campos, and C. R. Torres-San-Miguel, ''A novel tuning method of PD with gravity compensation controller for robot manipulators,'' IEEE Access, vol. 8, pp. 114773–114783, 2020.

[31] Khalil HK. Nolinear system. Upper Saddle River (NJ). Prentice-Hall, 2001.

[32] Huaman Loayza, A.; Pérez Zuñiga, C. Design of a fuzzy sliding mode controller for the autonomous path-following of a quadrotor. IEEE Lat. Am. Trans. 2019, 17, 962–971.

[33] T. T. Nguyen, "Sliding mode control-based system for the two-link robot arm," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 9, no. 4, pp. 2771-2778, 2019, doi: 10.11591/ijece.v9i4.pp2771-2778.

[34] ISO 11898-1:2015. (2015). Retrieved December 2015, from https://www.iso.org/standard/63648.html.

[35] Encinas, D., Meilan, P., Brava, A., & Naiouf, M. (2012). Protocolo de comunicaciones CAN aplicado a sistemas satelitales y vehículos lanzadores [Ebook] (pp. 1-5). La Plata.

[36] Product Center-GYEMS. (2021). Retrieved 1 July 2021, from http://www.gyems.cn/product.html.

[37] Z. Pineda Rico and A. Lecchini-Visintini and R. Quian Quiroga, Dynamic model of a 7-DOF Whole Arm Manipulator and validation from experimental data, 9th International Conference on Informatics in Control, Automation and Robotics, pp. 217, 2012.

[38] Aranda, I. A., & Pérez-Zúñiga, G. (2021). Highly Maneuverable Target Tracking Under Glint Noise via Uniform Robust Exact Filtering Differentiator With Intrapulse Median Filter. IEEE Transactions on Aerospace and Electronic Systems, 58(3), 2541-2559.

[39] Fenco, L., Pérez-Zuñiga, G., Quiroz, D., & Cuellar, F. (2021, September). Model Reference Adaptive Fuzzy Controller of a 6-DOF Autonomous Underwater Vehicle. In OCEANS 2021: San Diego–Porto (pp. 1-7). IEEE.

ET LUX IN TENEBRIS LUCET
MCMXVII

ANEXO A

A. Programas para simulación MATLAB

Listing 1 – Programa para hallar modelo matemático parametrizado

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% EULER - LANGRANGE PARA HALLAR MODELO MATEMÁTICO ROBOT
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
syms q1 real;
syms q2 real;
syms q3 real;
syms q4 real;
% Derivadas de las posiciones angulares:
syms dq1 real;
syms dq2 real;
syms dq3 real;
syms dq4 real;
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
syms d1 real;
syms d2 real;
syms d3 real;
syms a4 real;
% Gravedad
syms g;
syms Ix1 Iy1 Iz1 Ixy1 Ixz1 Iyz1 m1 rx1 ry1  rz1;
syms Ix2 Iy2 Iz2 Ixy2 Ixz2 Iyz2 m2 rx2 ry2  rz2;
syms Ix3 Iy3 Iz3 Ixy3 Ixz3 Iyz3 m3 rx3 ry3  rz3;
syms Ix4 Iy4 Iz4 Ixy4 Ixz4 Iyz4 m4 rx4 ry4  rz4;
sq1 = sin(q1);
sq2 = sin(q2);
sq3 = sin(q3);
sq4 = sin(q4);
cq1 = cos(q1);
cq2 = cos(q2);
cq3 = cos(q3);
cq4 = cos(q4);

A01 = [ cq1,  0, -sq1,  0;
        sq1,  0,  cq1,  0;
         0,  -1,    0, d1;
         0,   0,    0,  1];

A12 =  [ sq2,  0, -cq2, 0;
        -cq2,  0, -sq2, 0;
           0,  1,    0, 0;
           0, 0,    0, 1];

A23 =  [-sq3,   0, -cq3,  0;
         cq3,   0, -sq3,  0;
           0,  -1,    0, d3;
             0, 0,    0,  1];

A34 =  [  sq4,  cq4,  0,  a4*sq4;
         -cq4,  sq4,  0, -a4*cq4;
            0,    0,  1,       0;
            0,    0,  0,       1];
```

```
A00 = eye(4);
A11 = eye(4);
A22 = eye(4);
A33 = eye(4);

A02 = A01*A12;
A03 = A01*A12*A23;
A04 = A01*A12*A23*A34;
A13 = A12*A23;
A14 = A12*A23*A34;
A24 = A23*A34;

Qj = [ 0 -1 0 0;
       1  0 0 0;
       0  0 0 0;
       0  0 0 0];

%%%%%%%%%%%%%%%%%%%%%%%%%%%% L-E 3:%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Uij Matrixes
   U11=diff(A01,q1)   ;  U21=diff(A02,q1);      U31=diff(A03,q1);
   U12=zeros(4);         U22=diff(A02,q2);     U32=diff(A03,q2);
   U13=zeros(4);         U23=zeros(4);         U33=diff(A03,q3);
   U14=zeros(4);         U24=zeros(4);         U34=zeros(4);

   U41=diff(A04,q1);
   U42=diff(A04,q2);
   U43=diff(A04,q3);
   U44=diff(A04,q4);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% L-E 4:%%%%%%%%%%%%%%%%%%%%

%Uijk Matrixes

   %U1jk Matrixes
   U111=diff(U11,q1); U121=zeros(4)     ; U131=zeros(4);
   U112=zeros(4)     ; U122=zeros(4)     ; U132=zeros(4);
   U113=zeros(4)     ; U123=zeros(4)     ; U133=zeros(4);
   U114=zeros(4)     ; U124=zeros(4)     ; U134=zeros(4);

   U141=zeros(4);
   U142=zeros(4);
   U143=zeros(4);
   U144=zeros(4);

%U2jk Matrixes
   U211=diff(U21,q1); U221=diff(U22,q1); U231=zeros(4);
   U212=diff(U21,q2); U222=diff(U22,q2); U232=zeros(4);
   U213=zeros(4)     ; U223=zeros(4)     ; U233=zeros(4);
   U214=zeros(4)     ; U224=zeros(4)     ; U234=zeros(4);

   U241=zeros(4)     ;
   U242=zeros(4)     ;
   U243=zeros(4)     ;
   U244=zeros(4)     ;

%U3jk Matrixes
   U311=diff(U31,q1); U321=diff(U32,q1); U331=diff(U33,q1);
   U312=diff(U31,q2); U322=diff(U32,q2); U332=diff(U33,q2);
   U313=diff(U31,q3); U323=diff(U32,q3); U333=diff(U33,q3);
```

```
   U314=zeros(4)     ; U324=zeros(4)     ; U334=zeros(4);

   U341=zeros(4)     ;
   U342=zeros(4)     ;
   U343=zeros(4)     ;
   U344=zeros(4)     ;

 %U4jk Matrixes
   U411=diff(U41,q1); U421=diff(U42,q1); U431=diff(U43,q1);
   U412=diff(U41,q2); U422=diff(U42,q2); U432=diff(U43,q2);
   U413=diff(U41,q3); U423=diff(U42,q3); U433=diff(U43,q3);
   U414=diff(U41,q4); U424=diff(U42,q4); U434=diff(U43,q4);

   U441=diff(U44,q1);
   U442=diff(U44,q2);
   U443=diff(U44,q3);
   U444=diff(U44,q4);

%%%%%%%%%%%%%%%%%%%%%%5 L - E 5%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Matrices de pseudoinvercia Ji


J1 = [     Ix1 ,  Ixy1    ,  Ixz1    ,  m1*rx1 ;
          Ixy1 ,  Iy1     ,  Iyz1    ,  m1*ry1 ;
          Ixz1 ,  Iyz1    ,  Iz1     ,  m1*rz1 ;
        m1*rx1  ,  m1*ry1  ,  m1*rz1  ,      m1 ];

J2 = [     Ix2 ,  Ixy2    ,  Ixz2    ,  m2*rx2 ;
          Ixy2 ,  Iy2     ,  Iyz2    ,  m2*ry2 ;
          Ixz2 ,  Iyz2    ,  Iz2     ,  m2*rz2  ;
        m2*rx2  ,  m2*ry2  ,  m2*rz2  ,      m2 ];

J3 = [     Ix3 ,  Ixy3    ,  Ixz3    ,  m3*rx3 ;
          Ixy3 ,  Iy3     ,  Iyz3    ,  m3*ry3 ;
          Ixz3 ,  Iyz3    ,  Iz3     ,  m3*rz3  ;
        m3*rx3  ,  m3*ry3  ,  m3*rz3  ,      m3 ];


J4 = [     Ix4 ,  Ixy4    ,  Ixz4    ,  m4*rx4 ;
          Ixy4 ,  Iy4     ,  Iyz4    ,  m4*ry4 ;
          Ixz4 ,  Iyz4    ,  Iz4     ,  m4*rz4  ;
        m4*rx4  ,  m4*ry4  ,  m4*rz4  ,      m4 ];


%%%%%%%%%%%%%%%%%%%%%%%%%%%% L - E 6 %%%%%%%%%%%%%%%%%%%%%%%%
% Matriz de Inercias D
% sumatoria de la diagonal Ukj*Jk*Uki' para encontrar los dij

  %D matrix

  Uaux=zeros(4);
for i=1:4
   for j=1:4
       m=max([i j]);
       x=0;
       for k=m:4
       Uaux=eval(['U' num2str(k) num2str(i)]);
       Ud=Uaux';
       A=eval(['U' num2str(k) num2str(j)])*eval(['J' num2str(k)])*Ud;
```

```
       x=x+trace(A);
       end
       eval(['d' num2str(i) num2str(j) '=x']);
   end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% L - E 7
% Terminos hikm

%Hikm matrix
  for i=1:4
   for k=1:4
     for m=1:4
       mx=max([i m k]);
       x=0;
       for j=mx:4
       Uaux=eval(['U' num2str(j) num2str(i)]);
       Uh=Uaux';
       x=x+trace(eval(['U' num2str(j) num2str(k) num2str(m)])*eval(['J'
num2str(j)])*Uh);
       end
       eval(['h' num2str(i) num2str(k) num2str(m) '=x']);
     end
   end
  end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% L - E 8 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Obtener matriz columna de de fuerza de coriolis y centrípeta H = [h]'

%Coriolis column matrix

for i=1:4
      y=0;
      for k=1:4
          for m=1:4
          y=y+eval(['h' num2str(i) num2str(k) num2str(m)])*eval(['dq'
num2str(k)])*eval(['dq' num2str(m)]);
          end
      end
      eval(['h' num2str(i) '=y']);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% L - E 9
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Vector gravedad

vg = [-g, 0, 0, 0];
r1 = [  rx1 ; ry1  ; rz1 ; 1];
r2 = [  rx2 ; ry2  ; rz2   ; 1];
r3 = [  rx3 ; ry3  ; rz3  ; 1];
r4 = [  rx4 ; ry4  ; rz4   ; 1];

 for i=1:4
      x=0;
      for j=i:4
      x=x+(-eval(['m' num2str(j)])*vg*eval(['U' num2str(j)
num2str(i)])*eval(['r' num2str(j)]));
      end
      eval(['c' num2str(i) '=x']);
```

```
  end

%%%%%%%%%%%%%%%%%%%%%% L - E 10 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Obtenemos las matrices de la ecuación robótica
% D = [ d11 d12 d13 d14;
%       d21 d22 d23 d24;
%       d31 d32 d33 d34;
%       d41 d42 d43 d44];
%
% H = [ h1 ; h2 ; h3 ; h4 ];
%
% C = [ c1 ; c2 ; c3 ; c4 ];
%
% D = vpa(D,2);
% H = vpa(H,2);
% C = vpa(C,2);

%save matrices_4DOF D H C

d11=simplify(d11,100);
d12=simplify(d12,100);
d13=simplify(d13,100);
d14=simplify(d14,100);
d21=simplify(d21,100);
d22=simplify(d22,100);
d23=simplify(d23,100);
d24=simplify(d24,100);
d31=simplify(d31,100);
d32=simplify(d32,100);
d33=simplify(d33,100);
d34=simplify(d34,100);
d41=simplify(d41,100);
d42=simplify(d42,100);
d43=simplify(d43,100);
d44=simplify(d44,100);

h1=simplify(h1,100);
h2=simplify(h2,100);
h3=simplify(h3,100);
h4=simplify(h4,100);

c1=simplify(c1,100);
c2=simplify(c2,100);
c3=simplify(c3,100);
c4=simplify(c4,100);

fprintf('d11=%s;\n',d11)
fprintf('d12=%s;\n',d12)
fprintf('d13=%s;\n',d13)
fprintf('d14=%s;\n',d14)
fprintf('d21=%s;\n',d21)
fprintf('d22=%s;\n',d22)
fprintf('d23=%s;\n',d23)
fprintf('d24=%s;\n',d24)
fprintf('d31=%s;\n',d31)
fprintf('d32=%s;\n',d32)
fprintf('d33=%s;\n',d33)
fprintf('d34=%s;\n',d34)
fprintf('d41=%s;\n',d41)
```

```
fprintf('d42=%s;\n',d42)
fprintf('d43=%s;\n',d43)
fprintf('d44=%s;\n',d44)
fprintf('\n')
fprintf('h1=%s;\n',h1)
fprintf('h2=%s;\n',h2)
fprintf('h3=%s;\n',h3)
fprintf('h4=%s;\n',h4)
fprintf('\n')
fprintf('c1=%s;\n',c1)
fprintf('c2=%s;\n',c2)
fprintf('c3=%s;\n',c3)
fprintf('c4=%s;\n',c4)
```

ANEXO B

B. Componentes para la implementación del Robot

B.1. Protocolo CAN Bus

CAN (Controller Area Network), es un protocolo de comunicación originalmente creado para la comunicación de dispositivos relacionados a la industria automotriz. Por ejemplo, el primer auto en utilizar este protocolo CAN bus fue el modelo "850 Coupe" de BMW en 1986, reemplazando el uso de hasta 2 km de cable eléctrico de señales y sensores en el automóvil, esto significó una reducción de aproximadamente 50 kg de peso, lo cual permitió mayores ventajas para el vehículo. El uso del protocolo CAN bus se ha ampliado a otras industrias como Marina, Médica, Maquinaria, Militar, así como aplicaciones de elevadores, entre otras.

En 1993, El protocolo CAN Bus, se convierte en un estándar internacional que corresponde a la ISO-11898 [34]. Esto debido a las grandes ventajas que maneja el protocolo y por las cuales se mantiene vigente aún con la incorporación de redes industriales más sofisticadas. Algunas ventajas de este protocolo son:

- Posee herramienta para la detección de errores, así como volver a transmitir mensajes de tramas erróneas de manera automática.
- Priorización y tiempos de latencia garantizados para los mensajes lo cual lo vuelve una opción viable para los sistemas de tiempo real.
- Discriminación de errores provocados por un nodo y su posterior desconexión automática evitando así la saturación de la red.
- Flexible para agregar y quitar nodos de manera dinámica sin afectar el protocolo.
- Puede dirigir mensajes a varios nodos al mismo tiempo.

Existe hasta 4 tipos de tramas que gestionan los mensajes los cuales son: Tramas de Datos, Trama Remota (solicitud de datos), Trama de error y Trama de Sobrecarga. En la Figura 1 podemos ver la estructura de la Trama de Datos.

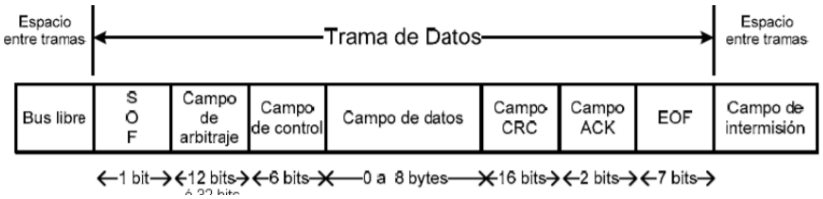


Figura 1. Campos de la Trama de datos [35]

La red CAN Bus se vuelve entonces determinista gracias al principio de arbitraje no destructivo dado que al momento que 2 o más nodos envíen un mensaje, estos serán priorizados mediante un identificador del mensaje asegurando que el más importante se envíe primero. Además, las Tramas de Datos se priorizan ante otras tramas enviadas al mismo tiempo. Para el bus, el nivel de monitoreo puede ser recesivo (1) o dominante (0) dependiente de la configuración de la alimentación física del bus. En la Figura 2 se podrá observar un ejemplo sobre el arbitraje de los mensajes entre nodos.

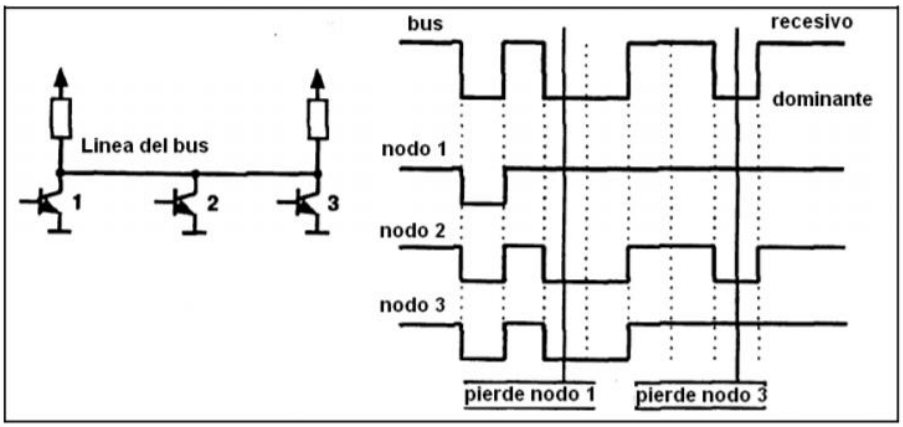


Figura 2. Arbitraje de los mensajes entre nodos.[34]

Para la comunicación de 2 nodos será necesario tener el formato de los mensajes que puede brindar un nodo, tanto para lectura como escritura.

B.2. Servoaccionadores GYEMS

Los servoaccionadores son elementos finales de control que permiten llevar una fuerza o par (torque) a un elemento mecánico, en este caso, a cada eslabón del brazo robótico. Este torque será ejecutado a partir de una señal de control que será enviada desde los controladores principales del Robot.

A continuación, se hará referencia a algunas de las ventajas por las que estos modelos de servos han sido seleccionados.

- Tiene incorporado un enconder que es de naturaleza magnética a 6, 14 o 18 bits dependiente del modelo específico del motor.
- Protocolos de comunicación desarrollados en CAN Bus o 485.
- Sensor de temperatura de placa incorporado.
- Específico para sistemas robóticos como brazos mecánicos, exoesqueleto, etc.
- Reductor, para el incremento de Torque asociado, de alta precisión.
- Diferentes modos de control: Posición, velocidad y Torque.

Estas características las puede realizar mediante un chip ARM de 32 bits que viene a ser un tipo de procesador que significa “Advance RISC Machine” donde RISC es el acrónimo de “Reduced Instruction Set Computer” lo cual le permite ser más eficiente en las tareas que tiene que realizar, en este caso, el servoactuador.

En el protocolo CAN Bus es posible conectar hasta 8 de estos motores en una misma red dado que cada motor cuenta con un conector de entrada y otro de salida para la red CAN Bus. Dentro del Modelo RMD del servo existen diferentes números de parte que básicamente se diferencian por el torque máximo que pueden ejecutar. En la Figura 3 podemos ver algunos de estos modelos específicos.

Figura 3. Diferentes tipos de motores RMD en la página oficial de GYEMS. [36]

B.3. Microcontroladores STM32

Para entender mejor la definición de estos microcontroladores se van a definir algunos conceptos previos que nos ayuden a mejorar el entendimiento de los términos a utilizar.

- Dentro del contexto de Hardware de procesamiento podemos encontrar a los MCU y los MPU. El MCU es un microcontrolador anexo a otros componentes como microprocesador, periféricos como ADC, puerto UART, etc. Mientras que los MPU son microprocesadores como tal, pero con mayor rendimiento que un microcontrolador.
- El concepto de ARM, es posible definirlo como un tipo de procesador, pero también como una arquitectura de procesamiento los cuales se definen hasta en 3 tipos denominados CORTEX.
  - Cortex A: Aplicaciones que requieren un sistema operativo.
  - Cortex M: Dispositivos embebidos, Bajo consumo de potencia.
  - Cortex R: Para Real-Time. Tiempo de respuesta mejorado.
- Como se mencionó anteriormente ARM significa “Advance RISC Machine” donde RISC se puede definir sobre todo como una ideología de arquitectura. Para entender mejor el concepto, se puede mencionar la Arquitectura CISC (Complex Instruction Set Computer) en donde podremos encontrar instrucciones más complejas y por lo tanto esto traerá programaciones más sencillas haciendo que el cargo computacional esté sobre el hardware en cambio RISC (Reduced Instruction Set Computer) utiliza instrucciones simples que hace una programación más compleja pero de esta manera la carga del procesamiento se encontrará en el software y ya no en el hardware lo que hace que esta ideología sea económicamente más rentable. Un ejemplo de CISC y RISC podría ser el uso de los microcontroladores PIC a través de Assembler (Programación a bajo nivel que utiliza RISC) pero se puede mitigar la complejidad de la programación a través del lenguaje de programación C.

Entonces, STM32 es una familia de microcontroladores basado en ARM Cortex -M que permiten alto rendimiento del procesador, ejecución de rutinas en tiempo real, procesamiento de señales digitales, entre otras funcionalidades. Para el uso de este microcontrolador es necesario utilizar una placa electrónica de desarrollo que nos permita interactuar físicamente con los periféricos del STM32.

Diferentes Proveedores nos brinda la opción de algunas de estas placas de desarrollo. Para el caso del control de los servoaccionadores RMD Gyems, se utilizado una placa de desarrollo Nucleo que podemos observar en la Figura 4.

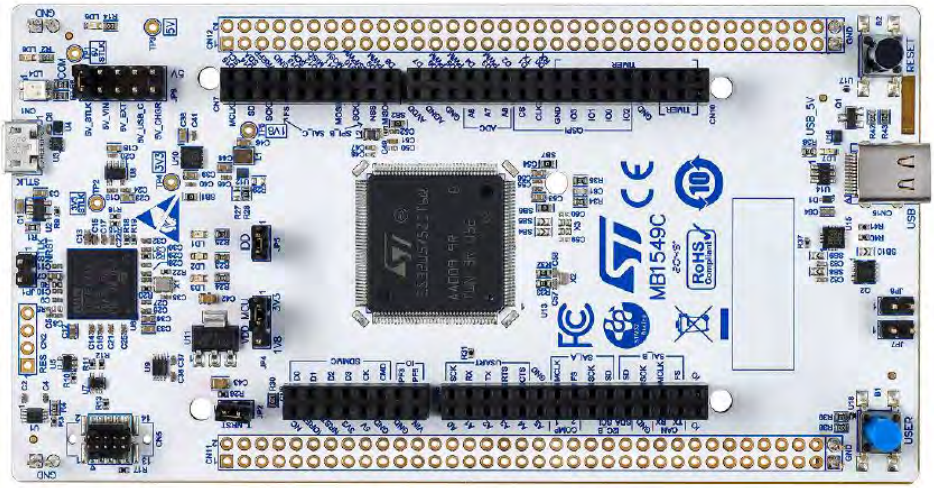


Figura 4. Placa de desarrollo Núcleo SMT32F446ZE

Con esta placa de desarrollo será posible conectar los servoaccionadores RMD con CANBUS y con el cual podremos generar la programación en base al lenguaje C.

B.4. Microcomputadoras Jetson

La empresa Multinacional NVIDIA, especializada en el desarrollo de tecnologías informáticas ha incursionado desde algunos años atrás en el área de los sistemas integrados apostando por una tecnología de pequeños computadores orientados a ser plataforma de desarrollo para la inteligencia artificial. Podemos ver en la Figura 5 algunos de estos sistemas.

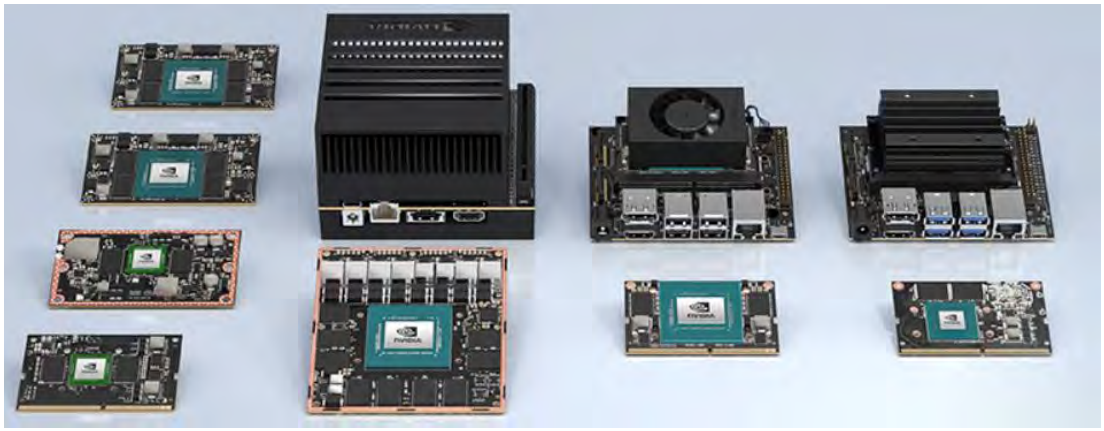


Figura 5. Kits de desarrollos para los procesadores NVIDIA JETSON. ("Sistemas Integrados NVIDIA para Máquinas Autónomas de Próxima Generación", 2012)

Dentro de los tipos de microcomputadoras Jetson tenemos:

- Jetson Nano.
- Jetson TX2.
- Jetson Xavier NX.
- Jetson AGX Xavier.

En cada uno de ellos se puede notar la diferencia en capacidad de procesamiento, memoria, periféricos siendo incluso la última de ellas, Jetson AGX Xavier, diseñada específicamente para maximizar rendimiento y la eficiencia energética del equipo lo que la convierte en una opción viable para aplicaciones de máquinas autónomas de larga duración energética. En la Figura 6 podemos ver el sistema integrado de una Jetson Xavier NX que se caracteriza entre otras cosas por su reducido tamaño y su alto rendimiento para aplicaciones de sistemas robóticos entre otros.

Figura 6. Kit de desarrollo para Jetson Nano. ("Sistemas Integrados NVIDIA para Máquinas Autónomas de Próxima Generación", 2012b)

La intención de utilizar Jetson Xavier NX en el proyecto de Robot teleoperado, es justamente poder obtener un procesamiento para las diferentes redes neuronales que se quieren ejecutar en paralelo en este procesador para brindarle al robot futuras capacidades de aprendizaje que son necesarias dentro del marco de la inteligencia artificial.

ANEXO C

C. Sintonización de los controladores locales PI de motores GYEMS

Los servoaccionadores GYEMS utilizados en el proyecto cuentan con lazos de control para posición, velocidad y torque. Para realizar el lazo de control abierto se ha dispuesto el uso del control de posición de este motor mientras que, para el control avanzado en lazo cerrado, se deberá hacer uso del control de torque del motor. Debido a esto, es necesario realizar previamente una correcta sintonización de los parámetros del controlador local que llevan estos servoaccionadores.

Los parámetros PI serán modificados reiterativamente desde la STM32 hasta conseguir las mejores respuestas de control a través del protocolo CAN y la TRAMA mostrada en la Figura 7.

Data field	Description
DATA[0]	Command byte
DATA[1]	NULL
DATA[2]	Position loop Kp
DATA[3]	Position loop Ki
DATA[4]	Speed loop Kp
DATA[5]	Speed loop Ki
DATA[6]	Torque loop Kp
DATA[7]	Torque loop Ki

```
void write_ROM_PID(uint8_t Motor_ID)
{
        SetTxHeader(Motor_ID);
        TxConfig[0] = 0x32;
        TxConfig[1] = 0x00;
        TxConfig[2] = k_pidp;
        TxConfig[3] = i_pidp;
        TxConfig[4] = k_pidv;
        TxConfig[5] = i_pidv;
        TxConfig[6] = k_pidt;
        TxConfig[7] = i_pidt;
}
```

Figura 7. Trama CAN para envío de parámetros PI de control local GYEMS.

En cada experimento habrá información que se recibe de estos motores como posición, velocidad y torques, los cuales serán llevados mediante uno de los puertos UART de la STM32 hacia MATLAB. donde se podrán realizar la síntesis de resultados experimentales para configurar los mejores parámetros PI de los controles locales.

Para la comunicación UART de diferentes datos hacia MATLAB. se debe realizar dos actividades en simultaneo:

- Envió de datos desde la STM32 separados por una coma y en una sola trama serial.
- Recepción de datos en MATLAB, donde se realizará un algoritmo para ubicar la posición de las comas que se encuentran en la trama. A partir de

estas posiciones se podrá realizar el desacoplamiento de los datos enviados para un posterior análisis gráficos y por tabla de cada dato por separado.

Un ejemplo de los datos recibidos en MATLAB por UART los podemos en la Figura 8.

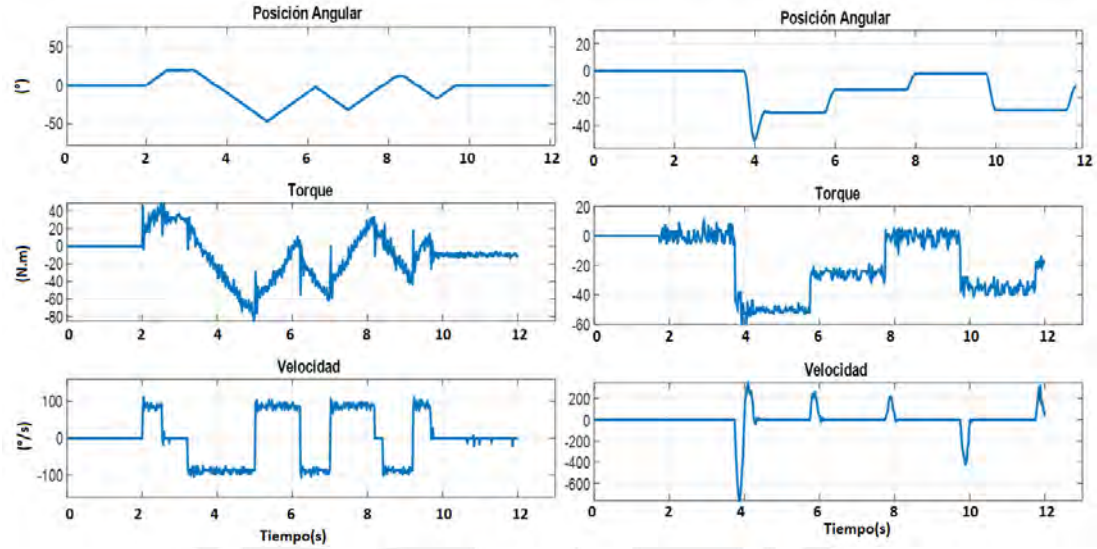


Figura 8. Respuesta en Matlab de motores GYEMS para control de posición y Torque.

Dado que no es posible abrir el lazo interno de los motores GYEMS, se ha realizado la sintonización de sus parámetros de manera heurística para diferentes posiciones como se muestra en la figura 4.9 hasta obtener resultados deseables y sin vibraciones. Una vez sintonizados los motores ya podemos pasar a la implementación de control.

ANEXO D

D. Filtro de ruido para las señales retroalimentadas en la implementación

Se puede observar que los valores analógicos de la velocidad angular de cada articulación, viene con un ruido de medición. Esto es debido a que el único sensor que tienen los motores GYEMS es el encoder para la medición de posición angular. Así, se puede concluir que internamente, estos motores están realizando una derivada de esta posición angular para obtener la velocidad lo cual conlleva un ruido en este proceso. En la figura 9 se puede evidenciar este ruido de medición cuando se tiene una velocidad constante de la articulación. Aquí se muestra la velocidad que se está enviando por comunicación desde el motor en comparación con la velocidad calculada en el microcontrolador STM32 a partir de la posición del encoder.

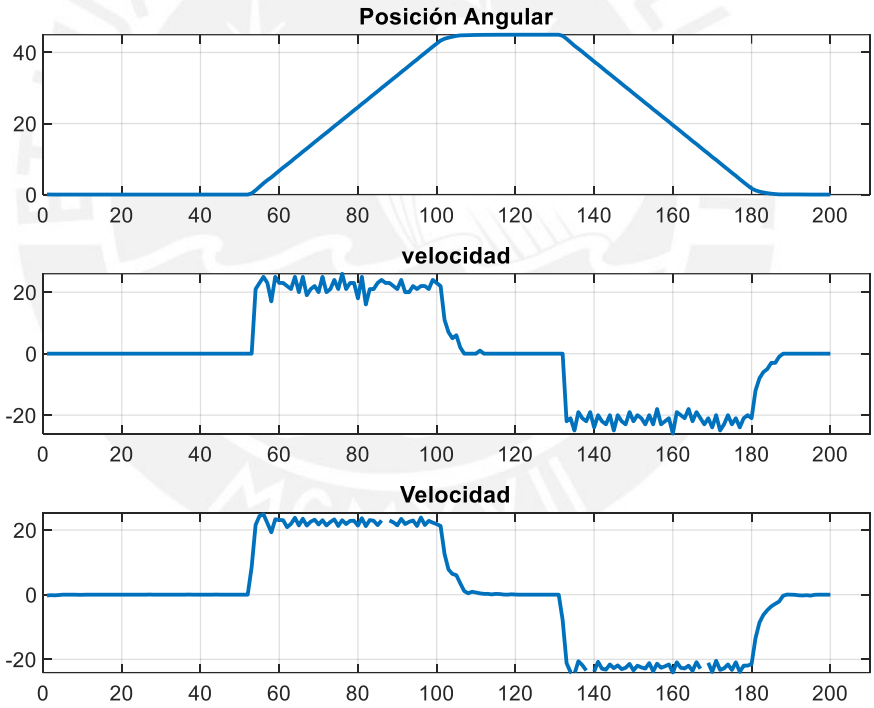


Figura 9. Comparación velocidades calculada del motor GYEMS.

En cualquiera de los casos, como se comentó, se puede observar que la velocidad conlleva el ruido de medición mencionado. Dado que esta velocidad es parte de las entradas de nuestro controlador principal es que se hace necesario el filtrado de esta señal.

Sin embargo, el uso de filtro, como sabemos siempre conlleva agregar un retardo a la señal lo cual puede afectar negativamente al controlador ya que no llegaría a tiempo la repuesta calculada para el control en determinado momento.

El primer filtro que se usó esta basado en la media móvil exponencial, debido a su sencilla implementación en código abierto a partir de la siguiente ecuación.

$$A_n = \alpha M + (1 - \alpha) A_{n-1}$$

Esta es una ecuación recursiva donde M es la señal analógica leída desde cada motor. A_n y A_{n-1} son lo valores calculados desde esta ecuación recursiva inicializando desde 0 y α es el índice de filtrado donde "1" corresponde a no aplicar filtro y "0" sígnica filtra al 100% la señal (mantener constante la señal calculada).

En la figura 10 se puede observar la aplicación de este filtro donde se podrá observar el correcto filtrado, pero en contraparte se comienza a visualizar un retardo en la lectura de esta señal.

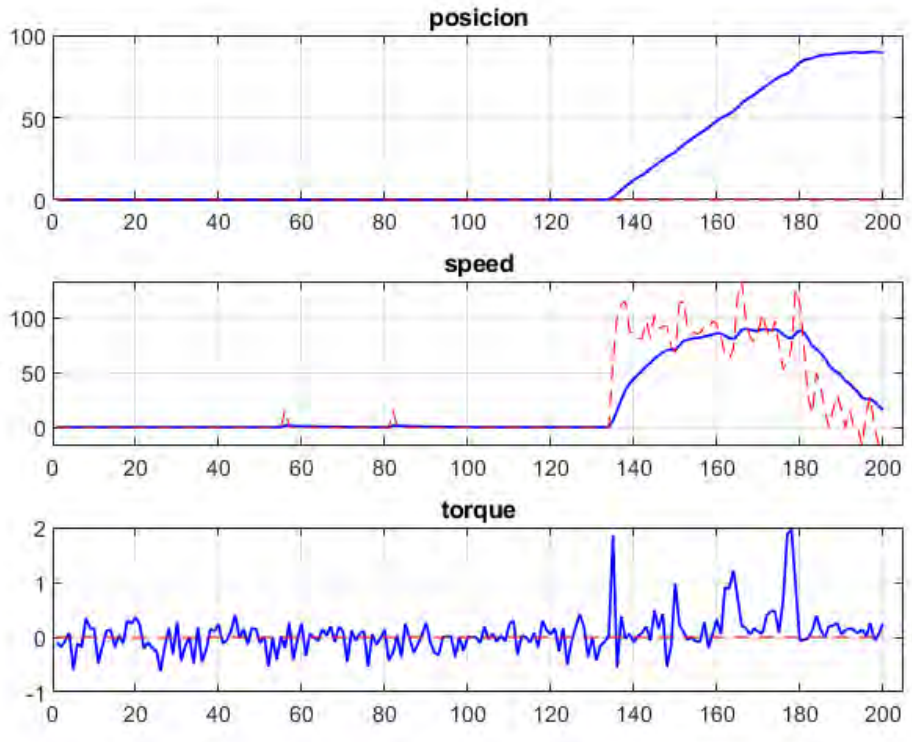


Figura 10. Filtro EMA aplicado a la velocidad del motor.

Este retardo como se comentó tendrá un efecto negativo en el control. Para poder aplicar este filtrado y minimizar este efecto retardo en su aplicación se ha pensado en doble filtrado EMA o que se conoce por sus siglas DEMA. Este doble filtrado ha

demostrado una minimización en el retardo presentado inicialmente a coste de un mayor computo, pero su implementación en código abierto viene dada por la siguiente expresión.

$$DEMA = 2\,x\,EMA - EMA\,(EMA)$$

Como se puede observar, la aplicación de este doble filtro viene dado por la aplicación consecutiva del primer filtro EMA. A continuación, se presenta el código en lenguaje estructurado realizado en la STM32 para la aplicación de este doble filtrado para cada señal.

Listing 2 – Doble filtro DEMA para señales analógicas en STM32

```
void scale_in (float *pos, int16_t *int_vel, float *q, float *qp)
{
	static float qp1[4];
	static float qp2[4];
//	filtro4 = (float)int_filt/100;
	float alpha[4] = {{0.2},{0.2},{1},{1}};
	float vel[4];
	for(int i=0; i<=3; i++) {q[i] = pos[i]*PI/180;}
///////////////Aplicamos la reducción de veloicidad para motores 1 y
2//////////////////
	vel[0] = (float)int_vel[0]*PI/(180*6);
	vel[1] = (float)int_vel[1]*PI/(180*6);
	vel[2] = (float)int_vel[2]*PI/180;
	vel[3] = (float)int_vel[3]*PI/180;


///////////////// usamos la señal que viene por CAN ///////////////////
//	for(int i=0; i<=3; i++) {qp[i] = vel[i]*PI/180;}

/////////////// Con los datos de posición, se deriva /////////////////
//	for(int j=0; j<=3; j++) {qp[j] = (pos[j]-pos_ant[j])/tm; qp[j] *=
PI/180; pos_ant[j] = pos[j];}

///////////////// Se filtra ña señal de velocidad que viene por CAN
//////////////
	for(int k=0; k<=3; k++)
	{
		qp1[k] = (alpha[k]*vel[k]) + ((1-alpha[k])*qp1[k]); // ema 1
		qp2[k] = (alpha[k]*qp1[k]) + ((1-alpha[k])*qp2[k]); // ema 2
		qp[k]  = 2*qp1[k] - qp2[k];
	//DEMA = 2EMA - EMA(EMA)
	}
}
```

La aplicación de este filtro posterior a la recepción de la señal de velocidad, nos brinda, como ya se indicó, una minimización en el retardo provocado por el filtrado. Esto se puede verificar la figura 11 para las misma condiciones de filtrado (igual α) que el filtro EMA anteriormente presentado.

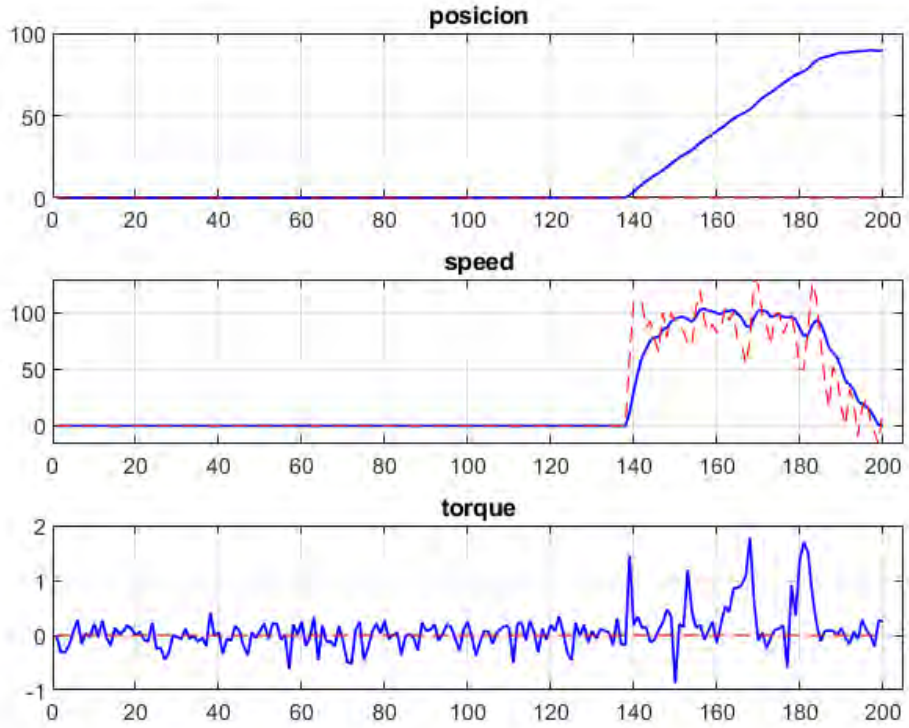


Figura 11. Filtro DEMA aplicado a la velocidad del motor.