UNIVERSIDAD DE GUANAJUATO

CAMPUS IRAPUATO - SALAMANCA
DIVISIÓN DE INGENIERÍAS

"MODELADO Y CONTROL DE UN BRAZO ROBÓTICO DE 3 GRADOS DE LIBERTAD"

T E S I S

QUE PARA OBTENER EL GRADO DE:

MAESTRO EN INGENIERÍA ELÉCTRICA

PRESENTA:

ING. CESAR EDUARDO CONEJO BENITEZ

ASESORES:

DR. JESÚS IXBALANK TORRES ZÚÑIGA
DR. EDMUNDO GABRIEL ROCHA CÓZATL

SALAMANCA, GUANAJUATO MARZO 2021

Salamanca, Gto., a 22 de enero del 2021.

M. en I. HERIBERTO GUTIÉRREZ MARTIN
JEFE DE LA UNIDAD DE ADMINISTRACIÓN ESCOLAR
PRESENTE-

Por medio de la presente, se otorga autorización para proceder a los trámites de impresión, empastado de tesis y titulación al alumno(a) Cesar Eduardo Conejo Benitez del ***Programa de Maestría en*** Ingeniería Eléctrica (Instrumentación y Sistemas Digitales) y cuyo número de ***NUA*** es: 146766 del cual soy director. El título de la tesis es: Modelado y control de un brazo robótico de 3 grados de libertad.

Hago constar que he revisado dicho trabajo y he tenido comunicación con los sinodales asignados para la revisión de la tesis, por lo que no hay impedimento alguno para fijar la fecha de examen de titulación.

ATENTAMENTE

NOMBRE Y FIRMA
DIRECTOR DE TESIS
SECRETARIO

Edmundo Gabriel Rocha Cózatl
NOMBRE Y FIRMA
DIRECTOR DE TESIS

Juan Gabriel Aviña Cervantes
NOMBRE Y FIRMA
PRESIDENTE

Juan Pablo I. Ramírez Paredes
NOMBRE Y FIRMA
VOCAL

Dedicado a Dios y a mi familia por
haber sido parte fundamental
de mi vida.

Agradecimientos Institucionales

Al Consejo Nacional de Ciencia y Tecnología (CONACYT) por haberme apoyado con la asignación de la beca nacional para el estudio de programas de posgrado, la cual me permitió poder llevar a cabo mis estudios de maestría en una institución académica de excelencia, teniendo como número de registro de CVU: 893291.

A la Universidad de Guanajuato por haberme permitido realizar mis estudios de maestría en la División de Ingenierías Campus Irapuato-Salamanca (DICIS) y por formar parte de la enseñanza en este país.

A la DICIS, al Departamento de Ingeniería Electrónica y al Posgrado de Ingeniería Eléctrica por las instalaciones, la atención y el apoyo que me proporcionaron durante la realización de mis estudios de maestría y durante el desarrollo de mi proyecto de investigación para la obtención del grado académico.

Al Instituto Tecnológico de Lázaro Cárdenas por haberme permitido realizar mis estudios de Ingeniería Electrónica y por las oportunidades que me brindó para realizar estancias de investigación científica en otras instituciones, ya que fue lo que despertó mi interés hacia la investigación.

Agradecimientos Personales

Primeramente a Dios, por haberme permitido llegar hasta este punto de mi vida y por conservar la vida de mi familia cercana, ya que han sido el pilar más grande de mi vida por todo el apoyo incondicional brindado.

A mis padres, Crispín y Roverta por haberme dado siempre su apoyo y ánimo en cada decisión que he tomado, y por haberme sacado adelante con mucho esfuerzo y dedicación. Me han demostrado que siempre contare con ellos.

A mi abue Ma. Rosa por haberme permitido alojarme en su hogar durante el transcurso de la maestría y por haberme brindado alegría, ánimo y todo su apoyo cuando más lo necesitaba, lo cual hizo muy amena mi estancia.

A mis hermanos, Osmar, Elisa y Mailo, que siempre me han aportado felicidad y diversas emociones a mi vida.

Al Dr. Ixbalank Torres Zúñiga, por haberme dado la oportunidad de trabajar con el como tesista, por las clases de control que me impartió durante la maestría, por los consejos dados y por el tiempo prestado. De igual manera agradezco al Dr. Edmundo Rocha Cózatl, por el material que nos compartió para la construcción del manipulador robótico, así como la ayuda y tiempo que me proporcionó durante la realización de este trabajo.

A mis amigos y compañeros de maestría: Jazmín, Giselle, Cecilia, Scarlett, Erika, Alejandro, Sigifredo, Ángel, Viridiana, Iván y Luis; por permitirme ser su amigo y por hacer tan amena mi estancia. Además, por el apoyo y paciencia que me brindaron cuando presentaba dificultades de entendimiento en alguna asignatura.

Finalmente, a todas las personas que también hicieron posible este trabajo.

Resumen

Este trabajo describe la construcción, modelado y simulación de un brazo robótico de tres grados de libertad con efector final, así como la implementación de técnicas de control robusto adecuadas.

Se investigó acerca de las características mecánicas y los modelos matemáticos (cinemático y dinámico) que pueden presentarse en el manipulador robótico. Para observar su comportamiento dinámico, se realizó una simulación en SciLab donde una ley de control robusto, en lazo cerrado, fue implementada para eliminar las incertidumbres del modelo dinámico y para reducir los efectos de perturbaciones externas.

En este trabajo se introducen los controladores H_∞, twisting y super-twisting de tercer orden para que el brazo robótico realice el seguimiento de una trayectoria definida dentro de su área de trabajo aun con la presencia de una fuerza externa y variaciones paramétricas. Con los servomotores Dynamixel: AX-12A y AX-18A, se encontró la forma de acoplarlos a la estructura del manipulador y la interfaz adecuada para enlazarlos a una computadora mediante una comunicación serial asíncrona tipo half duplex (8 bits de datos, 1 bit de paro y sin bit de paridad).

Como resultado se obtuvo la simulación del comportamiento del brazo robótico con la implementación de una estrategia de control super-twisting de tercer orden, obteniendo así, la estabilidad del sistema, una mejor precisión en cada articulación del manipulador y la generación de un movimiento suave durante una trayectoria parametrizada en el tiempo a partir de cualquier posición requerida dentro de su área de trabajo.

Abstract

This work describes the construction, modeling and simulation of a robotic arm of three degrees of freedom with a final effector, as well as the implementation of adequate robust control techniques.

The mechanical characteristics as well as kinematic and dynamic models of the robotic arm were first investigated. In order to observe its dynamic behavior, closed-loop simulations in SciLab were developed by considering three robust control laws to bypass the dynamic model uncertainties and to reduce the effect of external disturbances.

In this work, H_∞, twisting and third-order super-twisting controllers are introduced to track a predefined trajectory inside a work area even in presence of an external force and parametric uncertainties. Dynamixel actuators, AX-12A and AX-18A were attached to the manipulator structure and connected to a personal computer throughout half duplex asynchronous serial communication (8 data bits, 1 stop bit, no parity bit).

As a result, it was possible to obtain the simulation of the robotic arm behavior with the implementation of a third-order super-twisting control strategy, obtaining, system stability, a better precision in each articulation of the manipulator and the generation of a smooth movement during a parameterized trajectory in time from any position required inside its work area.

Índice general

Agradecimientos Institucionales II

Agradecimientos Personales III

Resumen IV

Abstract V

1. Introducción 1

1.1. Antecedentes 2
1.2. Planteamiento del problema 5
1.3. Objetivos 5
1.4. Justificación 6
1.5. Descripción de los capítulos 6
1.6. Resumen del primer capítulo 7

2. Marco teórico 8

2.1. Cinemática en brazos robóticos 8
2.1.1. Cinemática directa 9
2.1.2. Cinemática inversa 12
2.2. Planificación de trayectoria 13
2.2.1. Trayectoria lineal 13
2.2.2. Polinomio de quinto grado 14
2.3. Dinámica en brazos robóticos 16
2.3.1. Formulación de Euler-Lagrange 17
2.3.2. Modelo dinámico 18
2.4. Control de sistemas robóticos 19
2.4.1. Problema de control en sistemas robóticos 20

2.4.2. Modelo en espacio de estados 20
2.4.3. Definición de puntos de operación 22
2.4.4. Linealización de sistemas no lineales 25
2.4.5. Estabilidad de sistemas no lineales 27
2.4.6. Control por retroalimentación de estados 31
2.4.7. Regiones de estabilidad para sistemas LTI 34
2.5. Control robusto 36
2.5.1. Control H_∞ 36
2.5.2. Introducción al control por modos deslizantes 42
2.5.3. Control por modos deslizantes para regulación de salida 50
2.5.4. Controlador twisting 55
2.5.5. Controlador super-twisting de tercer orden 58
2.6. Resumen del segundo capítulo 62

3. Metodología **64**
3.1. Análisis cinemático del brazo robótico 64
3.1.1. Desarrollo de la cinemática directa 65
3.1.2. Desarrollo de la cinemática inversa 68
3.1.3. Simulación de la cinemática 71
3.1.4. Simulación de trayectoria con polinomio de 5to. grado 73
3.2. Análisis dinámico del brazo robótico 75
3.2.1. Lagrangiano del brazo robótico 76
3.2.2. Desarrollo de la ecuación de Euler-Lagrange 78
3.2.3. Modelo dinámico en la forma estándar 81
3.2.4. Modelo dinámico real para el brazo robótico 82
3.3. Estabilidad en lazo abierto 83
3.3.1. Modelado del manipulador en el espacio de estados 83
3.3.2. Puntos de operación 84
3.3.3. Linealización y estabilidad del sistema 86
3.3.4. Simulación de la estabilidad del sistema en lazo abierto 88
3.4. Análisis de un controlador lineal 90
3.4.1. Retroalimentación de estados 91
3.4.2. Consideración de regiones de estabilidad 96
3.4.3. Incertidumbres paramétricas 100
3.4.4. Implementación del control H_∞ 103

3.5. Resumen del tercer capítulo 109

4. Implementación y resultados 111

4.1. Construcción del brazo robótico 111

4.1.1. Piezas mecánicas 112

4.1.2. Servomotores Dynamixel 117

4.1.3. Dispositivo U2D2 121

4.1.4. Alimentación de servomotores 122

4.2. Comunicación y operación de los servomotores 125

4.2.1. Programación de los servomotores en Python 126

4.2.2. Ejecución del código de programación 128

4.3. Implementación de controladores por modos deslizantes 130

4.3.1. Implementación del algoritmo twisting 133

4.3.2. Implementación del 3-AST 140

4.4. Escritura de los valores obtenidos por el 3-AST 153

4.4.1. Escritura de valores en los servomotores Dynamixel 155

4.4.2. Resultados 157

4.5. Resumen del cuarto capítulo 166

5. Conclusiones 167

5.1. Conclusiones generales 167

5.2. Trabajo a futuro 170

Apéndice A. Modelo dinámico en el espacio de estados 171

Apéndice B. Simulación de la cinemática 173

Apéndice C. Simulación de la trayectoria 175

Apéndice D. Linealización del modelo 178

Apéndice E. Estabilidad en lazo abierto 181

Apéndice F. Simulación del controlador lineal H_∞ 186

Apéndice G. Instalación de las funciones de control 191

Apéndice H. Programación de los Dynamixel en Python 195

Apéndice I. Configuración de los Dynamixel en RoboPlus 205

Apéndice J. Simulación del controlador twisting 210

Apéndice K. Simulación del 3-AST anti-windup 218

Apéndice L. Escritura de valores obtenidos de SciLab 229

Bibliografía 236

Índice de figuras

1.1. Estrategia de control para regular las salidas de un brazo robótico . . 1
1.2. Obra teatral Rossum's Universal Robots 2
1.3. Primer telemanipulador para manejar elementos radioactivos 3
1.4. Manipulador industrial para soldadura con controlador automático . . 4

2.1. Relación de la cinemática directa e inversa 9
2.2. Sistema de referencias para robot planar de dos grados de libertad . . 9
2.3. Representación descompuesta del robot planar 12
2.4. Perfil de posición con polinomio de quinto grado 14
2.5. Perfil de velocidad con polinomio de quinto grado 15
2.6. Perfil de aceleración con polinomio de quinto grado 15
2.7. Brazo robótico de un eje 22
2.8. Linealización de un modelo no lineal 25
2.9. Estabilidad del brazo robótico de un eje 30
2.10. Sistema en lazo cerrado con dinámicas asintóticamente estables . . . 31
2.11. Ley de control propuesta para sistema linealizado 34
2.12. Ley de control propuesta para un sistema no lineal 34
2.13. Representación de la región de estabilidad 35
2.14. Sistema con presencia de incertidumbres paramétricas 37
2.15. Politopo formado a partir de los valores máximos y mínimos 38
2.16. Sistema que relaciona la salida y entrada con la función de transferencia 39
2.17. Control por retorno de estados H_∞ 40
2.18. Ley de control retroalimentada por modo deslizante 43
2.19. Señal del control u(t) con chattering 46
2.20. Estimación del control equivalente 48
2.21. Estimación de la perturbación 49

2.22. Convergencia a cero de las variables de estado 49
2.23. Convergencia a cero de las variables deslizantes σ y $\dot{\sigma}$ 50
2.24. Ley de control con regulación de salida 51
2.25. Comparación de la salida de referencia y_r con la salida medida y . . . 54
2.26. Señal del controlador $u(t)$ para regulación de salida 54
2.27. Señal de la variable deslizante σ 55
2.28. Retroalimentación del controlador twisting 57

3.1. Brazo robótico tipo antropomórfico 64
3.2. Representación cinemática del brazo robótico 65
3.3. Representación descompuesta del manipulador 68
3.4. Primer triángulo para el análisis geométrico 69
3.5. Segundo triángulo para el análisis geométrico 69
3.6. Tercer triángulo del análisis geométrico 70
3.7. Brazo robótico ubicado en la posición establecida 72
3.8. Trayectoria seguida por el manipulador robótico 74
3.9. Posiciones angulares para cada articulación 75
3.10. Representación del brazo robótico real 75
3.11. Representación del brazo robótico con masas distribuidas 76
3.12. Posiciones angulares estables para cada articulación 89
3.13. Posición cartesiana estable para el brazo robótico 90
3.14. Respuesta de la posición con control por retroalimentación estados . . 95
3.15. Pares motor debidos a la retroalimentación de estados 96
3.16. Plano complejo 97
3.17. Representación gráfica de las regiones de estabilidad 97
3.18. Respuesta de la posición con regiones de estabilidad 99
3.19. Pares motor debidos a las regiones de estabilidad 99
3.20. Politopo formado por los valores de gravedad y fricción 100
3.21. Respuesta de la posición con presencia de incertidumbres paramétricas 101
3.22. Pares motor debidos a las incertidumbres paramétricas 102
3.23. Perturbación provocada por una fuerza externa 104
3.24. Respuesta de la posición con presencia de la perturbación τ_d 107
3.25. Pares motor debido a la presencia de la perturbación τ_d 108

4.1. Brazo robótico construido 112

4.2. Representación virtual de la base 113
4.3. Vista frontal de la base 113
4.4. Vista superior de la base desdoblada 114
4.5. Representación virtual de pieza para primer eslabón 114
4.6. Medidas de los extremos laterales de la pieza para el primer eslabón . 115
4.7. Representación virtual de pieza para segundo eslabón 115
4.8. Vista frontal de pieza desdoblada para segundo eslabón 116
4.9. Medidas del extremo lateral izquierdo de la pieza del segundo eslabón 116
4.10. Servomotores Dynamixel AX-12A y AX-18A 117
4.11. Ensamblaje de servomotores 118
4.12. Conectores molex de tres pines 119
4.13. Conexión multipunto con servomotores 120
4.14. Conexión del dispositivo U2D2 121
4.15. Diseño del dispositivo U2D2 121
4.16. Configuración de pines del dispositivo U2D2 122
4.17. Fuente de alimentación de 12 V a 5 A 123
4.18. Circuito regulador de voltaje a 11.1 V 123
4.19. Regulador de voltaje en placa fenólica 124
4.20. Conexión para la alimentación y comunicación del brazo robótico . . 125
4.21. Diagrama de flujo para comunicación y movimiento de los servomotores 127
4.22. Pinza mecánica ubicada en las posiciones establecidas 128
4.23. Leyendas que indican el estado de la comunicación y los datos leídos . 129
4.24. Seguimiento de dos trayectorias por el manipulador robótico 131
4.25. Sistema de control en lazo cerrado para brazo robótico 132
4.26. Posiciones angulares dadas durante la trayectoria 132
4.27. Señales de perturbación para cada servomotor 136
4.28. Convergencia de las salidas debidas al controlador twisting 137
4.29. Señales de control debidas al controlador twisting 138
4.30. Vista amplificada de las señales de control twsiting 138
4.31. Variables deslizantes σ y $\dot{\sigma}$ debidas al controlador twisting 139
4.32. Errores de regulación debidos al controlador twisting 140
4.33. Convergencia de las salidas debidas al controlador 3-AST 143
4.34. Señales de control debidas al controlador 3-AST 144
4.35. Variables deslizantes σ y $\dot{\sigma}$ debidas al controlador 3-AST 145

4.36. Errores de regulación debidos al controlador 3-AST 145
4.37. Señales de control debidas al controlador 3-AST anti-windup 147
4.38. Respuestas de las salidas debidas al controlador 3-AST anti-windup . 148
4.39. Variables deslizantes σ y $\dot{\sigma}$ debidas al controlador 3-AST anti-windup 149
4.40. Errores de regulación debidos al controlador anti-windup 150
4.41. Nuevas señales de perturbación consideradas para cada servomotor . 150
4.42. Convergencia de salidas al considerar nuevas señales de perturbación . 151
4.43. Señales de control al considerar las nuevas señales de perturbación . . 152
4.44. Variables deslizantes σ y $\dot{\sigma}$ al considerar las nuevas perturbaciones . . 152
4.45. Errores de regulación al considerar las nuevas perturbaciones 153
4.46. Sintaxis para la creación del archivo CSV en SciLab 154
4.47. Archivo CSV en formato Excel con los datos requeridos 155
4.48. Diagrama de flujo para escritura de valores y control de la pinza . . . 156
4.49. Masas de los objetos que serán traslados por el brazo robótico 157
4.50. Primer compilación del programa 157
4.51. Posiciones angulares leídas durante la primer ejecución 158
4.52. Cargas aplicadas durante la primer ejecución 158
4.53. Errores de posición angular de la primer compilación 160
4.54. Segunda compilación del programa 161
4.55. Posiciones angulares leídas durante la segunda compilación 161
4.56. Cargas aplicadas durante la segunda compilación 162
4.57. Errores de posición angular de la segunda compilación 162
4.58. Aplicando fuerzas externas en la primera articulación 163
4.59. Aplicando fuerzas externas en la segunda articulación 163
4.60. Aplicando fuerzas externas en la tercera articulación 164
4.61. Posiciones angulares leídas durante la aplicación de fuerzas externas . 164
4.62. Cargas aplicadas durante la presencia de fuerzas externas 165
4.63. Errores de posición angular debido a la presencia de fuerzas externas 165

G.1. Repositorio para descargar las funciones de control 191
G.2. Dirección del archivo instalador 192
G.3. Pagina web para descargar el distribuidor Anaconda 192
G.4. Opciones de instalación avanzadas 193
G.5. Instalación de las funciones de control en Python 193
G.6. Funciones de control instaladas 194

H.1. Instrucción para leer caracteres individuales en Python 195
H.2. Importación de funciones y definición de la tabla de control 196
H.3. Protocolo de comunicación y configuraciones predeterminadas 196
H.4. Solución de la cinemática inversa para obtención de ángulos 198
H.5. Conversión a valores enteros y definición de vectores 199
H.6. Funciones básicas para la comunicación y el control simultáneo 200
H.7. Funciones para habilitar y asignar el par motor 201
H.8. Funciones para la operación simultánea de los servomotores 202
H.9. Funciones para leer la posición angular y la carga actual 203
H.10.Sentencia para cambiar posición objetivo 204

I.1. Sitio para descargar RoboPlus 205
I.2. Ventana de inicio de RoboPlus 206
I.3. Selección y habilitación del puerto de comunicación 207
I.4. Selección de la velocidad de transmisión y protocolo 207
I.5. Busqueda de los servomotores conectados al puerto 208
I.6. Modificación del ID para servomotor 209
I.7. Inhabilitación del puerto 209

L.1. Importación de funciones y definición de vectores 229
L.2. Función que permite la escritura de la posición angular y el par motor 230
L.3. Configuración de la posición inicial 231
L.4. Lectura del archivo CSV y asignación de valores 232
L.5. Condiciones para la pinza mecánica y la lectura de los servomotores . 233
L.6. Código para graficar las posiciones angulares leídas 234

Índice de tablas

2.1. Comparación de controladores por modo deslizante 58

3.1. Parámetros de Denavit-Hartenberg para robot diseñado 66
3.2. Medidas de los piezas que construyen al brazo robótico 72
3.3. Coordenadas cartesianas de las posiciones establecidas 74
3.4. Par y Velocidad manejados por los servomotores 82
3.5. Posibles puntos de operación para la estabilidad del sistema 86
3.6. Valores propios del polinomio característico $P(\lambda)$ 87
3.7. Parámetros considerados para el análisis de estabilidad 87

4.1. Características generales de los servomotores Dynamixel 119
4.2. Parámetros que involucra el modelo dinámico 136
4.3. Coordenadas cartesianas de las posiciones objetivo 136
4.4. Valores mínimos y máximos de las incertidumbres paramétricas . . . 137

H.1. Parámetros considerados para el código de programación 197
H.2. Funciones que permiten la comunicación y el control simultáneo . . . 200
H.3. Funciones para el control del par motor y la posición angular 202

Capítulo 1

Introducción

Un robot es un sistema electromecánico programable que puede realizar algunas funciones de manera autónoma mediante la ejecución de movimientos. A partir del término robot, se define a un brazo robótico como un brazo mecánico programable capaz de realizar actividades muy parecidas al brazo humano, pero con una mayor rapidez; como la manipulación y el desplazamiento de objetos.

Antes se tenía la idea de que los robots eran diseñados solo para sustituir al ser humano en la realización de actividades del uso doméstico, pero ahora los robots han ido innovándose cada vez más para su implementación en la industria manufacturera.

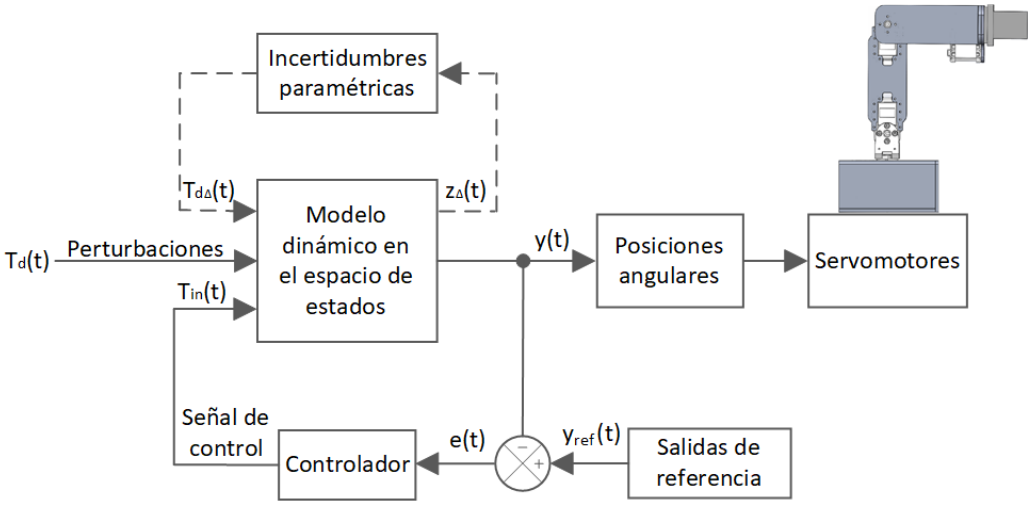


Figura 1.1: Estrategia de control para regular las salidas de un brazo robótico

La robótica es una rama de la ingeniería que se encarga de asegurar que los robots puedan realizar y desempeñar las tareas hechas por los seres humanos que requieran

del uso de la inteligencia, (Siciliano y Khatib, 2008). La robótica no solo es la creación de robots, sino también consiste en el diseño, análisis, programación y producción.

Todo sistema robótico debe de tener ciertas mejoras para poder trabajar sin ningún problema bajo la presencia de incertidumbres y perturbaciones que puedan presentarse al momento de su operación. Por lo tanto, se diseñará una estrategia de control en lazo cerrado para regular las salidas de un brazo robótico, ver Figura 1.1.

1.1. Antecedentes

Durante siglos, el ser humano ha construido máquinas que imitan las partes del cuerpo humano. Por ejemplo, los antiguos egipcios unían brazos mecánicos a las estatuas de sus dioses y los griegos construían estatuas que operaban con sistemas hidráulicos para fascinar a los adoradores de los templos, (UdeSantiago, 2021).

El término inglés "robot" fue derivado de la palabra checa robota que significa trabajo ejecutivo y fue introducido por primera vez por el dramaturgo checo Karel Čapek en su obra teatral Rossum's Universal Robots en 1921, en la Figura 1.2[1] se muestra una imagen de dicha obra teatral. Desde entonces el termino ha aplicado a prácticamente cualquier cosa que opere con cierto grado de autonomía, generalmente bajo control informático, (Hoifodt, 2011).

Figura 1.2: Obra teatral Rossum's Universal Robots

[1]NCLab. (2018). Ilustración de Karel Čapek y su obra R.U.R. https://community.nclab.com

Los primeros robots fueron diseñados con la capacidad de realizar movimientos generales, bajo la suposición de que se encontrarían dentro de un mercado más grande si pudieran realizar la mayor variedad de tareas; este énfasis demostró ser costoso tanto en recursos monetarios como en desempeño. Por lo cual, ahora los robots se diseñan teniendo en cuenta un conjunto especifico de tareas, (Siciliano y Khatib, 2008).

Con el objetivo de manipular elementos radioactivos sin riegos para el operador, en 1948 R.C. Goertz del Argonne National Laboratory, desarrolló al primer telemanipulador que consistía en un dispositivo mecánico maestro-esclavo, en donde el manipulador maestro era movido directamente por el operador mientras que el esclavo reproducía fielmente los movimientos de este. A través del dispositivo maestro, el operador sentía las fuerzas que el esclavo ejercía sobre el entorno, ver Figura 1.3 [2]. En 1954 Goertz hizo uso de la tecnología electrónica y del servocontrol, para sustituir la transmisión mecánica por una eléctrica y desarrollando así al primer telemanipulador con servocontrol bilateral, (Barrientos et. al, 1997).

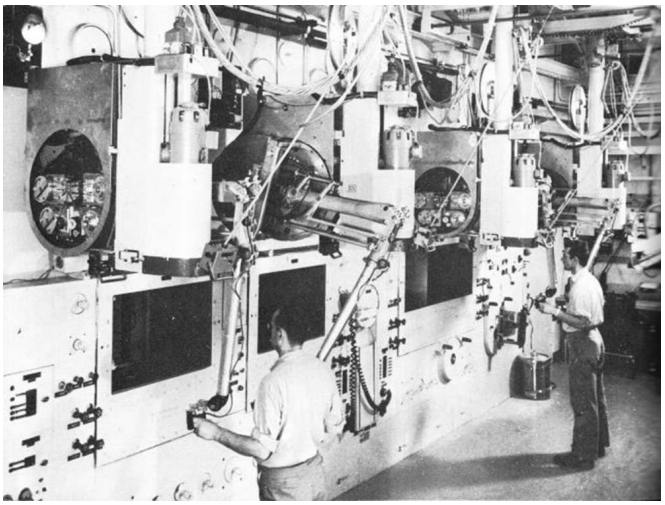


Figura 1.3: Primer telemanipulador para manejar elementos radioactivos

Los robots industriales surgen de las tecnologías del control automático, de los manipuladores teleoperados y de la aplicación de computadoras en tiempo real. Las actividades que llevan a cabo los robots industriales se están volviendo cada vez más sofisticadas, pero aún se da el caso de que solo se instalan para soldadura o el manejo de materiales, (Craig, 2006), ver Figura 1.4 [3].

[2]Cyberneticzoo. (2014). Ilustración de un primer modelo experimental. http://cyberneticzoo.com
[3]AVEX. (2017). Soldadura robótica. https://www.avexproducts.com

Para mejorar las prestaciones de los robots industriales, se han ido investigando técnicas para identificar modelos suficientemente fiables de la dinámica del robot y métodos para el control de las articulaciones, permitiendo optimizar el comportamiento dinámico mediante la compensación de las no linealidades y acoplamientos, (Atkenson y Hollerback, 1988).

Figura 1.4: Manipulador industrial para soldadura con controlador automático

El trabajo de Rivera et al. (2011) es una de las primeras publicaciones que presenta la solución del problema de control de posición en un sistema robótico de 2-GDL (Pendubot), mediante la implementación de un controlador por modo deslizante super-twisting. Sin embargo, el algoritmo super-twisting (AST) aplicado en el Pendubot, no logra la estabilización en tiempo finito y por lo tanto, existe una menor robustez ante incertidumbres y perturbaciones. Esto se debe a que el AST solo funciona para sistemas con grado relativo uno y no para sistemas con grado relativo dos como es el caso de los brazos robóticos.

Posteriormente, Ruíz y Fridman (2014) presentan resultados experimentales de diferentes algoritmos de control por modos deslizantes, aplicados a un robot paralelo de 2-GDL para lograr el seguimiento de una trayectoria deseada; en dicha publicación se muestran dos nuevos algoritmos: control integral discontinuo (CID) y super-twisting de tercer orden (3-AST), los cuales permitieron lograr la estabilidad en tiempo finito del sistema robótico con grado relativo dos. Recientemente Cheol-Su et al. (2018) y Javed et al. (2019), han presentado publicaciones donde proponen algoritmos super-twisitng mejorados para cumplir con el mismo objetivo de estabilidad y control.

1.2. Planteamiento del problema

Los brazos robóticos se describen dinámicamente mediante ecuaciones diferenciales no lineales que relacionan: las masas inerciales, las fuerzas de Coriolis y centrípetas, las incertidumbres paramétricas debidas a las fricciones viscosas y la fuerza gravitacional, los momentos de fuerzas y la presencia de perturbaciones. Por lo tanto, definir trayectorias de movimiento en un brazo robótico con el uso del modelo dinámico en lazo abierto, es una tarea complicada, ya que la estabilidad podría no existir o solamente existir en un cierto rango de posiciones que se encuentren fuera de su área de trabajo, siendo de esta manera difícil poder llevarlo a una posición requerida.

Por lo tanto, en este trabajo se resolverá el problema del control de la posición para un brazo robótico de tres grados de libertad con el uso de un controlador robusto en lazo cerrado, para lograr la estabilidad del sistema, evitar las incertidumbres del modelo y reducir el efecto de las perturbaciones.

1.3. Objetivos

El objetivo general de este trabajo es la construcción, modelado y simulación de un brazo robótico antropomórfico de tres grados de libertad, así como la implementación en simulación de técnicas de control robusto para el seguimiento de trayectorias definidas. Como objetivos particulares se tienen:

1. Construcción del brazo robótico con el uso de los servomotores Dynamixel AX-12A y AX-18A, para lograr una mejor precisión en los movimientos de las articulaciones.

2. Desarrollar el modelo cinemático del brazo robótico con el uso del algoritmo de Denavit-Hartemberg y el método geométrico.

3. Desarrollar el modelo dinámico de este manipulador mediante la formulación Lagrangiana.

4. Desarrollar la simulación del brazo robótico en SciLab, para observar el comportamiento dinámico del manipulador durante el seguimiento de una trayectoria lineal previamente definida.

5. Desarrollar un controlador robusto por retroalimentación de salida, para mantener la estabilidad del sistema durante el seguimiento de una trayectoria lineal en presencia de perturbaciones y a pesar de las incertidumbres del modelo.

1.4. Justificación

Desde comienzos del siglo XXI, el diseño, construcción y operación de robots está siendo cada vez más investigado debido a las diversas aplicaciones implementadas, como en la industria automotriz y manufacturera, medicina, vehículos teleoperados, simulaciones virtuales, entre otras.

En la actualidad es común la implementación de robots en la industria para la realización de tareas repetitivas y actividades que pueden ser riesgosas para el ser humano, por tal hecho, se busca profundizar la investigación y desarrollo de un manipulador robótico para promover el interés y seguimiento hacia el estudio del control en estos sistemas.

Antes de implementar una estrategia de control, es necesario obtener el modelo dinámico del brazo robótico, para posteriormente aplicar las técnicas de control robusto que permitan minimizar el efecto de incertidumbres y perturbaciones, estabilizando además las dinámicas del sistema en lazo cerrado. Con esto se espera obtener un buen funcionamiento en el manipulador durante el seguimiento de una trayectoria parametrizada.

1.5. Descripción de los capítulos

Esta tesis está dividida en 5 capítulos. El capítulo 2 describe la información teórica a utilizar para el desarrollo de este trabajo, como la cinemática y dinámica que presentan los brazos robóticos, la planificación de trayectorias, los conceptos básicos del control no lineal y los tipos y características de los controladores robustos que pueden ser implementados en sistemas robóticos.

El capítulo 3 describe la metodología que se llevó a cabo para el desarrollo de este trabajo, en el cual se incluye el análisis de la cinemática directa e inversa para el brazo robótico a analizar, la generación de una trayectoria lineal con el uso del modelo

cinemático, la obtención del modelo dinámico mediante el desarrollo de la ecuación de Euler-Lagrange, el análisis de estabilidad en lazo abierto para el sistema robótico y la implementación de un controlador lineal H_∞ para determinar la fiabilidad que presenta este tipo de controlador robusto en un sistema linealizado que originalmente es no lineal.

En el capítulo 4 se presenta la estructura física del brazo robótico de tres grados de libertad, la interfaz de comunicación que se utilizó para la transferencia de datos entre los servomotores y una PC, las simulaciones realizadas en SciLab para verificar el comportamiento del sistema robótico al implementar controladores por modos deslizantes, la prueba que fue realizada con el brazo robótico físico y los resultados obtenidos.

Finalmente, el capítulo 5 consiste en las conclusiones generales obtenidas durante el desarrollo de este trabajo. Además, se menciona el trabajo a futuro y las posibles mejoras que pueden realizarse. En el apartado bibliográfico se encuentran los datos de la literatura que sirvió de apoyo para la escritura de esta tesis.

1.6. Resumen del primer capítulo

En resumen, este primer capítulo consistió en dar a conocer los antecedentes más relevantes acerca de la evolución tecnológica que han ido desarrollando los manipuladores robóticos al pasar de los años, así como también, dar a conocer las investigaciones más recientes que se han estado realizando para la solución del problema de control de la posición. Además, se mencionó una descripción general acerca del planteamiento del problema, los objetivos y la justificación para este trabajo de tesis.

Capítulo 2

Marco teórico

En este capítulo se mencionarán los conceptos teóricos de las técnicas que fueron implementadas para el desarrollo de este trabajo.

2.1. Cinemática en brazos robóticos

La cinemática consiste en el estudio de los movimientos de un robot donde la posición, velocidad y aceleración de cada uno de sus elementos articulares son calculados sin considerar las fuerzas que causan dicho movimiento.

Para el estudio de los movimientos de un brazo robótico sin consideración de las fuerzas que lo originan, es necesario realizar ciertos cálculos matemáticos que toman en cuenta el número de grados de libertad, las dimensiones y propiedades geométricas que constituyen los eslabones de cada articulación y la posición requerida del manipulador robótico. Con la cinemática es posible obtener los valores de las coordenadas articulares para llevar al extremo del robot a diferente posición o viceversa. El análisis cinemático se deriva en dos estudios: cinemática directa y cinemática inversa.

Un problema muy básico en el estudio de la manipulación mecánica se conoce como cinemática directa, que es el problema geométrico estático de calcular la posición y orientación del efector final del brazo robótico con los valores de las coordenadas articulares conocidas, (Craig, 2006). Otro problema fundamental es el de la cinemática inversa, el cual calcula los valores de las coordenadas articulares para llevar al efector final del robot a una posición y orientación conocida.

Lo comentado anteriormente, puede representarse con un diagrama de relación como el que se muestra en la Figura 2.1.

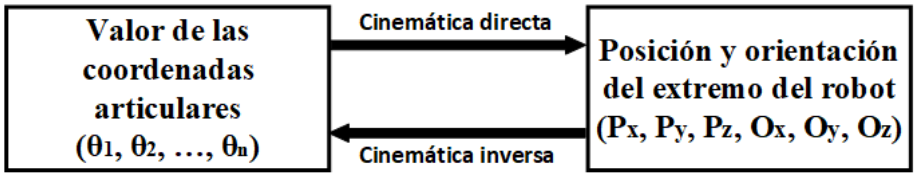


Figura 2.1: Relación de la cinemática directa e inversa

2.1.1. Cinemática directa

El álgebra vectorial y matricial se utiliza para representar y describir la localización de un objeto en el espacio tridimensional con respecto a un sistema de referencia fijo mediante una matriz de transformación homogénea. Como un brazo robótico puede considerarse como una cadena cinemática abierta formada por eslabones unidos entre sí mediante articulaciones, puede establecerse un sistema de referencia fijo situado en la base del robot y describir la localización de cada uno de sus eslabones con respecto a dicho sistema de referencia, (Barrientos 1997).

Para la obtención de las ecuaciones cinemáticas, se utiliza el algoritmo propuesto por Denavit-Hartenberg en 1955, descrito en Barrientos (1997). El cual mediante la matriz de transformación homogénea ${}^{i-1}T_i$, describe la geometría de la relación espacial entre los eslabones adyacentes del brazo robótico.

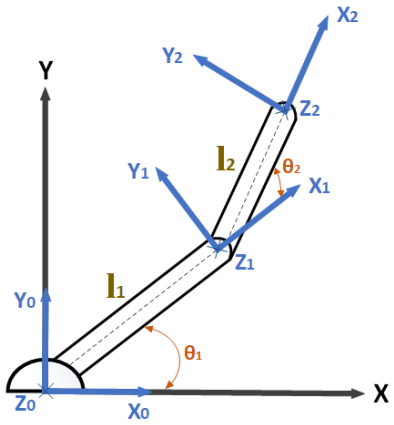


Figura 2.2: Sistema de referencias para robot planar de dos grados de libertad

Antes de implementar el método de Denavit-Hartenberg, es necesario tener una representación articulada ilustrada del brazo robótico que vaya a ser considerado para el análisis cinemático, donde a la base del robot se le asignará un sistema de referencia fijo y a cada articulación por eslabón un sistema de referencia coordenado, así como se muestra en la Figura 2.2. Los sistemas de referencia coordenados se nombran de acuerdo al número del eslabón al que son asignados (l_i). Esto es, el sistema de coordenadas i se le asigna al eslabón i, (Craig, 2006).

Los sistemas de referencia se asignan haciendo coincidir uno de los ejes del sistema de coordenadas, típicamente Z_i, con el eje de la articulación. El eje X_i se elige en la dirección de la perpendicular común entre el eje de la articulación y la siguiente. Para elegir el eje Y_i se sigue la regla de la mano derecha, (Ollero, 2001).

Mediante los sistemas de referencia de la Figura 2.2, es posible encontrar los cuatros parámetros cinemáticos de Denavit-Hartenberg, los cuales dependen de la geometría de los eslabones y la ubicación de las articulaciones que los unen, (Pérez, 2011). Estos parámetros están definidos como:

a_i = Es la distancia entre el eje Z_{i-1} hasta el eje Z_i medida a lo largo de X_{i-1}.
α_i = Es el ángulo de separación entre los ejes Z_{i-1} y Z_i medido alrededor de X_i.
d_i = Es la distancia entre el eje X_{i-1} hasta el eje X_i medida a lo largo de Z_{i-1}.
θ_i = Es el ángulo de separación entre los ejes X_{i-1} y X_i medido alrededor de Z_{i-1}.

La matriz de transformación homogénea ${}^{i-1}T_i$ es de 4x4 y representa a la posición y orientación de las articulaciones asociadas por los eslabones consecutivos. Esta matriz está dada por:

$$ {}^{i-1}T_i = \begin{bmatrix} R_{3x3} & P_{3x1} \\ f_{1x3} & w_{1x1} \end{bmatrix} = \begin{bmatrix} Rotacion & Traslacion \\ Perspectiva & Escalado \end{bmatrix} = \begin{bmatrix} {}^{i-1}R_i & {}^{i-1}P_i \\ 0\,0\,0 & 1 \end{bmatrix} \quad (2.1) $$

donde $R_{3x3} = {}^{i-1}R_i$, representa a una matriz de rotación de 3x3 que determina la orientación del sistema de coordenadas (i) con respecto al sistema de coordenadas $(i-1)$; $P_{3x1} = {}^{i-1}P_i$, es un vector de traslación de 3x1 que representa a la posición del sistema (i) con respecto al sistema $(i-1)$; f_{1x3}, es una submatriz que representa una transformación de perspectiva y w_{1x1} es una submatriz que representa a un escalado global, (Pérez, 2011).

De acuerdo a Shilling (2003), la forma general de la matriz de transformación homogénea ${}^{i-1}T_i$ que representa al sistema (i) con respecto al sistema $(i-1)$ queda representada por la ecuación (2.2).

$$ {}^{i-1}T_i = \begin{bmatrix} \cos\theta_i & -\cos\alpha_i \operatorname{sen}\theta_i & \operatorname{sen}\alpha_i \operatorname{sen}\theta_i & a_i\cos\theta_i \\ \operatorname{sen}\theta_i & \cos\alpha_i\cos\theta_i & -\operatorname{sen}\alpha_i\cos\theta_i & a_i\operatorname{sen}\theta_i \\ 0 & \operatorname{sen}\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.2} $$

En la ecuación (2.2) se sustituyen los cuatro parámetros de Denavit-Hartenberg correspondientes para obtener cada matriz de transformación homogénea ${}^{i-1}T_i$ que relaciona a dos sistemas de coordenadas vecinos: ${}^0T_1$, ${}^1T_2$, ..., ${}^{i-1}T_i$. Para el caso del manipulador robótico de dos grados de libertad de la Figura 2.2, se tendrían dos matrices de transformación homogénea: ${}^0T_1$ y ${}^1T_2$.

Para poder obtener una matriz que exprese a la posición y orientación del extremo final del último eslabón con respecto a la base del mismo, se tienen que considerar todos los grados de libertad del brazo robótico. Para lograr esto, se tienen que multiplicar entre sí cada una de las matrices de transformación homogénea; para el caso del brazo robótico de la Figura 2.2 se tiene:

$$ {}^0T_2 = {}^0T_1 \times {}^1T_2 \tag{2.3} $$

La matriz ${}^0T_2$ de la ecuación (2.3) que indica la localización del extremo final del brazo robótico con respecto al sistema de referencia de su base, tiene la siguiente forma:

$$ {}^0T_2 = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.4} $$

donde queda reflejado el valor de la posición (p_x, p_y, p_z) y los vectores de orientación (n, o, a) del extremo final del robot en función de las coordenadas articulares θ_1 y θ_2 para este caso.

2.1.2. Cinemática inversa

El uso de la cinemática inversa ayuda a encontrar los valores necesarios que deben de tomar las coordenadas articulares del brazo robótico para que el extremo del último eslabón pueda ser llevado a la posición y orientación requerida. La solución al problema de la cinemática inversa es útil incluso cuando no se emplean sensores externos como, por ejemplo, al hacer que el efector final o la herramienta sigan un camino en línea recta, (Shilling, 2003). Existen procedimientos ya desarrollados que ayudan a la solución del problema de la cinemática inversa.

Los métodos geométricos permiten obtener normalmente los valores articulares para brazos robóticos de pocos grados de libertad. El procedimiento consiste en utilizar relaciones trigonométricas y geométricas sobre los eslabones del brazo robótico y para ello, es necesario descomponer a la estructura robótica en triángulos geométricos para poder encontrar los valores articulares mediante el uso de la geometría plana.

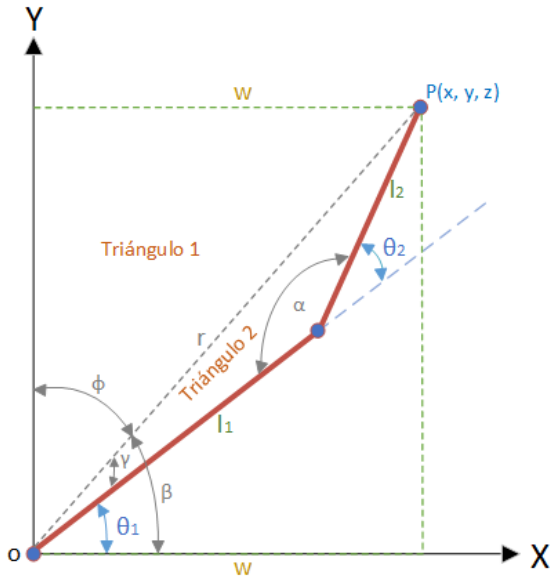


Figura 2.3: Representación descompuesta del robot planar

En la Figura 2.3 se muestra una descomposición trigonométrica para el brazo robótico de dos grados de libertad de la Figura 2.2, el cual permite facilitar la obtención de las ecuaciones cinemáticas inversas con el uso del método geométrico. Cabe señalar que, para este caso, existen dos posibles soluciones para la obtención de la cinemática inversa donde las soluciones no son únicas, ya que se pueden restringir por la geometría del robot o por la planeación de la trayectoria, el cual es el tema de la sección siguiente.

2.2. Planificación de trayectoria

El objetivo principal de cualquier robot es el poder moverse de un punto inicial a uno final mediante el seguimiento de una trayectoria predefinida. La planificación de trayectoria consiste en resolver continuamente las ecuaciones de la cinemática inversa para obtener los valores de las coordenadas articulares que lleven al extremo final del brazo robótico a la posición requerida, de acuerdo al tipo de trayectoria asignada.

En la planificación de la trayectoria, el usuario tiene que definir los parámetros que describan a la trayectoria deseada para posteriormente, generar una secuencia en el tiempo de sus valores mediante una función polinomial, (Pérez, 2015). Debido a que las articulaciones de un robot se encuentran representadas por actuadores mecánicos, la trayectoria planificada debe generar movimientos suaves en los actuadores para evitar el desgaste o daño de los mismos.

2.2.1. Trayectoria lineal

Una ruta es la descripción geométrica de un movimiento, la cual se denota por los puntos que tiene que seguir el brazo robótico en su espacio operacional, en cambio, una trayectoria es una ruta donde se especifica una ley en el tiempo, en términos, por ejemplo, de velocidades y aceleraciones en cada uno de los puntos, (Pérez, 2015).

Para este trabajo, se planificará una trayectoria que se encuentre dentro del espacio operacional del brazo robótico, la cual estará descrita por una ruta lineal basada en la ecuación paramétrica de una recta en tres dimensiones:

$$\begin{aligned} P_x(t) &= x_1 + (x_2 - x_1) * P(t) \\ P_y(t) &= y_1 + (y_2 - y_1) * P(t) \\ P_z(t) &= z_1 + (z_2 - z_1) * P(t) \end{aligned} \tag{2.5}$$

donde:

P_x: Puntos ubicados en x	P_y: Puntos ubicados en y	P_z: Puntos ubicados en z
x_1, y_1, z_1: Primer punto	x_2, y_2, z_2: Segundo punto	$P(t)$: Parámetro variente

De acuerdo a Pérez (2015), a la recta expresada en su forma paramétrica de la

ecuación (2.5) se le conoce como el lugar geométrico de la trayectoria, ya que está compuesta por el conjunto de puntos por donde va a pasar el extremo final del brazo robótico. El parámetro variante en el tiempo P(t) es el encargado de relacionar el lugar geométrico con el perfil de trayectoria, por lo que este parámetro es el polinomio que determina el perfil de posición y la suavidad del movimiento durante la trayectoria.

2.2.2. Polinomio de quinto grado

El parámetro P(t) tendrá la característica de un polinomio de quinto grado; esta función polinomial de tiempo es de los más implementados debido a que es sencilla de calcular y es una función que proporciona la suavidad requerida, (Corke, 2017). Este algoritmo queda expresado por la ecuación (2.6):

$$P(t) = At^5 + Bt^4 + Ct^3 + Dt^2 + Et + F \tag{2.6}$$

Y las primeras dos derivadas de la ecuación (2.6) quedan expresadas como:

$$\dot{P}(t) = 5At^4 + 4Bt^3 + 3Ct^2 + 2Dt + E \tag{2.7}$$

$$\ddot{P}(t) = 20At^3 + 12Bt^2 + 6Ct + 2D \tag{2.8}$$

donde t $\in [t_0, t_f]$. La primera derivada hace referencia a la velocidad y la segunda. derivada a la aceleración.

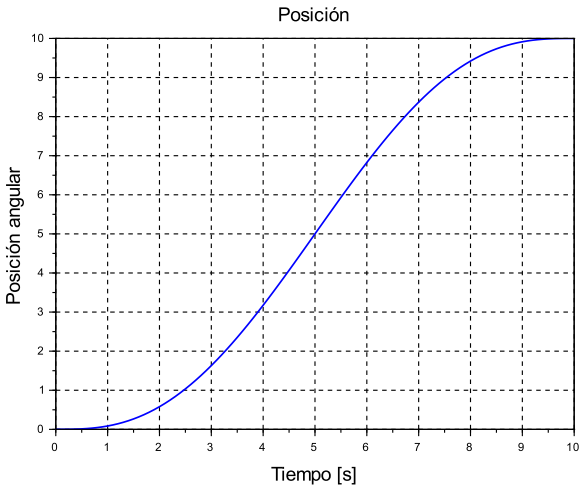


Figura 2.4: Perfil de posición con polinomio de quinto grado

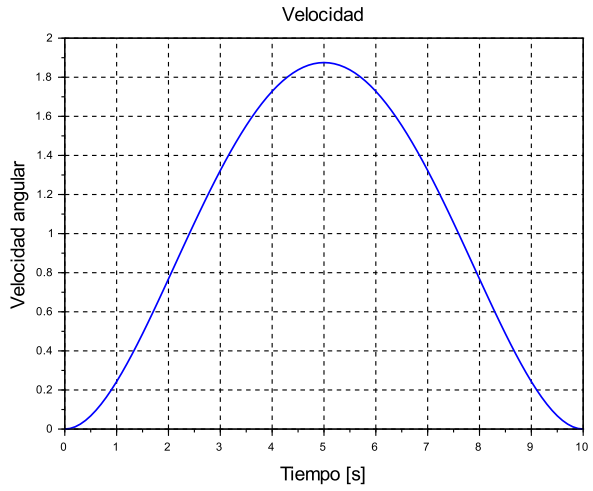


Figura 2.5: Perfil de velocidad con polinomio de quinto grado

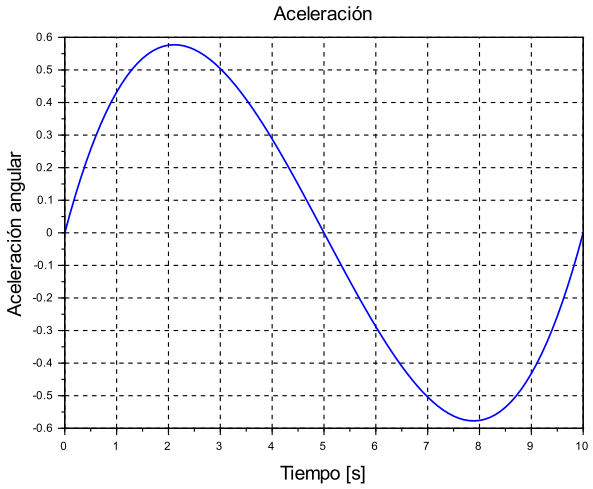


Figura 2.6: Perfil de aceleración con polinomio de quinto grado

En Craig (2006) se menciona que los polinomios de mayor orden (como el de quinto grado) se utilizan para especificar la posición, velocidad y aceleración al inicio y al final de cada segmento de ruta. Además, Pérez (2015) menciona que con este algoritmo polinomial el cambio de velocidad y aceleración es gradual al inicio y fin del recorrido, lo que hace que mejore las características dinámicas principalmente en una vecindad del límite del volumen de trabajo. Ver Figuras 2.4, 2.5 y 2.6.

El polinomio de quinto grado puede ser calculado mediante softwares que proporcionen métodos de álgebra lineal.

En la ecuación (2.9) se muestra al polinomio de quinto grado que permite generar la trayectoria de la Figura 2.4 (en el perfil de la posición), la cual depende del valor final al que se quiere llegar:

$$P(t) = 10\left(\frac{t^3}{t_f^3}\right) - 15\left(\frac{t^4}{t_f^4}\right) + 6\left(\frac{t^5}{t_f^5}\right) \tag{2.9}$$

donde:

t^n: Representa al tiempo actual en el que se va ejecutando la trayectoria.
t_f^n: Representa al tiempo establecido en el que se requiere que la trayectoria finalice.

Además, la función polinomial de la ecuación (2.9) permite obtener suavidad de movimiento en cada articulación durante la ejecución de la trayectoria. Está ecuación se encuentra descrita en Pérez (2015).

2.3. Dinámica en brazos robóticos

Al contrario que la cinemática, la dinámica en robótica se encarga de estudiar los movimientos del robot donde tanto la posición, velocidad y aceleración de cada uno de sus elementos articulares son calculados en relación con las fuerzas que son implicadas durante el movimiento del robot. De acuerdo a Barrientos (1997), la relación matemática quedaría expresada por:

a) La localización del robot (definida por sus variables articulares o por las coordenadas de localización de sus extremos) y sus derivadas: velocidad y aceleración.

b) Los momentos de fuerzas que son aplicados en las articulaciones.

c) Los parámetros dimensionales del robot, como longitud, masas e inercias de sus elementos.

La gran mayoría de los sistemas físicos se encuentran descritos dinámicamente por ecuaciones diferenciales no lineales, involucrando las variaciones en los parámetros del sistema y las perturbaciones externas.

Los métodos para la obtención del modelo dinámico, emplean formulaciones tales como la de Newton-Euler o la de Euler-Lagrange. El primer método se basa en fórmulas dinámicas elementales, en el análisis de fuerzas y en los momentos de restricción

que actuán entre los vínculos; el segundo método solamente involucra el balance de energías cinéticas y potenciales dentro de la dinámica; así como coordenadas y fuerzas generalizadas. Ambos métodos tienen diferente análisis matemático pero proporcionan resultados similares, (Schilling, 2003).

2.3.1. Formulación de Euler-Lagrange

La formulación de Euler-Lagrange, basada en conservaciones energéticas (Lagrangiano), será usada para la obtención del modelo dinámico debido a qué es más sistemático que el de Newton-Euler y facilita la obtención del modelo. Con el uso del Lagrangiano, las ecuaciones de movimiento pueden ser derivadas sistemáticamente en cualquier sistema de coordenadas.

El Lagrangiano de un sistema mecánico se define como la diferencia entre las energías cinética y potencial, expresado como:

$$L = K - P \tag{2.10}$$

donde K y P son respectivamente la energía cinética y energía potencial del sistema. Con la ecuación Lagrangiana, es posible desarrollar la formulación de Euler-Lagrange:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}_i}\right) - \frac{\partial L}{\partial \theta_i} = \tau_i \qquad i = 1, ..., n \tag{2.11}$$

donde $\dot{\theta}_i$ es la coordenada generalizada asociada con la fuerza generalizada τ_i, (Lewis y Munro, 2006).

Las fuerzas generalizadas (τ_i) vienen dadas por los pares que ejercen las articulaciones para mover a los eslabones (pares motor) y los pares que son inducidos por las fuerzas que produce el efector final en contacto con el ambiente (perturbaciones externas).

La ecuación (2.11) establece la relación existente entre las fuerzas generalizadas aplicadas al manipulador y las posiciones, velocidades y aceleraciones conjuntas. Por lo tanto, permiten la derivación del modelo dinámico del manipulador a partir de la determinación de la energía cinética y la energía potencial del sistema mecánico, (Siciliano et. al, 2009).

2.3.2. Modelo dinámico

Es importante mencionar que en la dinámica aparecerán fuerzas de Coriolis debidas al movimiento de sus elementos con respecto a un sistema en rotación; así como fuerzas centrípetas que dependen de la configuración instantánea del brazo robótico. Debido a esto, el modelo dinámico en su forma estándar quedaría expresado como:

$$M(\theta)\ddot{\theta} + C(\theta, \dot{\theta})\dot{\theta} + G(\theta) = \mathcal{T} \tag{2.12}$$

donde θ, $\dot{\theta}$, $\ddot{\theta} \in \mathbb{R}^n$ son la posición, velocidad y aceleración angular respectivamente. M(θ) es una matriz simétrica de nxn de masas inerciales en el espacio, C(θ,$\dot{\theta}$)$\dot{\theta}$ es un vector de nx1 que define a las fuerzas centrípetas y de Coriolis, G(θ) es un vector de nx1 que incluye a la gravedad y $\mathcal{T}$ es un vector de nx1 que involucra a los momentos de fuerzas (fuerzas generalizadas) que intervienen en el movimiento de las articulaciones (pares motor y perturbaciones externas).

De acuerdo a Craig (2006), un momento de fuerza puede ser positivo o negativo, donde los signos permiten indicar la dirección en la que se movería la articulación debido al momento de fuerza aplicado; por ejemplo, si el momento de fuerza es positivo, se indica que la articulación se movería en dirección del ángulo creciente (en sentido contrario a las manecillas del reloj); si es negativo, entonces, se movería en dirección del ángulo decreciente (en sentido a las manecillas del reloj).

En la realidad, un brazo robótico siempre se ve afectado por la presencia de fricciones articulares. Por lo tanto, considerando a las fricciones articulares, los pares motor y las perturbaciones externas, entonces, el modelo dinámico del brazo robótico de la ecuación (2.12) puede ser expresado como indica Lewis y Munro (2006):

$$M(\theta)\ddot{\theta} + C(\theta, \dot{\theta})\dot{\theta} + F(\dot{\theta}) + G(\theta) = \mathcal{T}_{in} + \mathcal{T}_d \tag{2.13}$$

donde el término F($\dot{\theta}$) es un vector de nx1 que incluye a las fuerzas de fricción que se pueden manifiestar en las articulaciones del brazo robótico, $\mathcal{T}_{in}$ es un vector de nx1 que incluye a los pares motor (entrada del sistema) y $\mathcal{T}_d$ es un vector de nx1 que incluye a las perturbaciones externas (entrada desconocida del sistema).

Para el caso de la fricción F($\dot{\theta}$) se considera la presencia de la fricción viscosa

F_b, ya que puede manifestarse por problemas de lubricación u otros defectos que pueden presentar los servomotores (articulaciones). En la fricción viscosa, el momento de torsión debido a la fricción es proporcional a la velocidad del movimiento de la articulación, (Craig, 2006). Por lo tanto, se tiene que:

$$F_b\dot{\theta} = diag[b_i]\dot{\theta}_i \tag{2.14}$$

donde b_i serán los coeficientes de la fricción viscosa.

La representación matricial de la fricción viscosa de acuerdo a la ecuación (2.14) estará dada por:

$$F_b\dot{\theta} = \begin{bmatrix} b_1 & 0 & 0 & 0 \\ 0 & b_2 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & b_n \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \vdots \\ \dot{\theta}_n \end{bmatrix} \tag{2.15}$$

La fricción no es un término fácil de modelar y, de hecho, puede ser el término más contrario a describir en el modelo dinámico del brazo robótico, (Lewis y Munro, 2006).

Finalmente, el modelo dinámico real para cualquier brazo robótico de n grados de libertad quedaría expresado por la siguiente ecuación de segundo orden:

$$M(\theta)\ddot{\theta} + C(\theta,\dot{\theta})\dot{\theta} + F_b\dot{\theta} + G(\theta) = \mathcal{T}_{in} + \mathcal{T}_d \tag{2.16}$$

2.4. Control de sistemas robóticos

La gran mayoría de los sistemas físicos se describen dinámicamente mediante ecuaciones diferenciales no lineales. Los brazos robóticos no son la excepción, sus modelos dinámicos relacionan las masas inerciales de cada elemento conformado, las fuerzas centrípetas y de Coriolis, las fricciones articulares, la fuerza gravitacional y los pares que son aplicados en cada articulación cuando el brazo robótico origina cualquier movimiento.

Para un brazo robótico de grado de libertad n, su modelo dinámico estará dado

por un sistema simultáneo de n ecuaciones diferenciales no lineales de segundo orden altamente acopladas. El desarrollo de algoritmos de control para brazos robóticos es actualmente un área de investigación activa, (Shilling, 2003).

Para el desarrollo teórico de esta sección, se usará el modelo dinámico de segundo orden de la ecuación (2.16) sin consideración de la entrada desconocida $\mathcal{T}_d$.

2.4.1. Problema de control en sistemas robóticos

Los modelos dinámicos de los brazos robóticos son muy complejos, por lo tanto, la planificación de trayectorias con el uso del modelo dinámico en lazo abierto puede ser complicado. Se dice que es complicado ya que, debido a la fuerza de gravedad, se podría presentar la estabilidad del sistema solamente en regiones de operación que tengan una coordenada de altura negativa, siendo difícil llevarlo a posiciones con coordenada de altura positiva.

La dificultad principal de aplicar técnicas de control a estos modelos para la regulación de la entrada del sistema ($\mathcal{T}_{in}$) es el conocimiento de los parámetros, que en algunos casos son difíciles de medir o calcular.

El problema del control de la posición angular en brazos robóticos consiste en proponer una ley de control adecuada en lazo cerrado que lleve a los estados del sistema a la estabilidad asintótica, en otras palabras, que los estados del sistema robótico no sólo permanecerán en las cercanías del punto de equilibrio, sino que llegarán a aproximarse al equilibrio a lo largo del tiempo: $\lim_{t\to+\infty} x(t) = x_0$.

2.4.2. Modelo en espacio de estados

Para facilitar el control de la posición angular en un manipulador robótico, se debe reformular la ecuación (2.16) como un sistema de primer orden de $2n$ ecuaciones; este sistema de ecuaciones se conoce como ecuaciones de estado, (Shilling, 2003).

Para reformular la ecuación (2.16) a la forma del espacio de estados, se tiene que aislar el término de aceleración $\ddot{\theta}$. De acuerdo a Shilling (2003), esto se puede hacer fácilmente porque la matriz del tensor de inercia $M(\theta)$ es positiva definida y por lo tanto no singular. El modelo en espacio de estados de un brazo robótico incluye tanto

una ecuación de estado dinámico como una ecuación de salida cinemática.

El lado izquierdo de un modelo en espacio de estados debe quedar expresado en términos de la velocidad y la aceleración angular, por lo tanto, el modelo reformulado de la ecuación (2.16), sin la presencia de la entrada desconocida $\mathcal{T}_d$, quedaría representado como:

$$\begin{aligned} \dot{\theta} &= v_\theta \\ \ddot{\theta} &= M(\theta)^{-1}\left[\mathcal{T}_{in} - C(\theta,\dot{\theta})\dot{\theta} - F_b\dot{\theta} - G(\theta)\right] \\ y &= \theta \end{aligned} \tag{2.17}$$

donde θ, $\dot{\theta}$, $\ddot{\theta} \in \mathbb{R}^n$ son los vectores de posición, velocidad y aceleración angular por articulación, v_θ hace referencia a la velocidad angular y y representa a la salida del sistema debida a la posición angular.

Posteriormente, se tienen que definir las variables de estado correspondientes, lo que resulta:

$$\begin{aligned} x_i &= \theta \\ \dot{x}_i &= \dot{\theta} = x_j \\ \dot{x}_j &= \ddot{\theta} \end{aligned}$$

Por lo tanto, la ecuación (2.17) estará representada por el siguiente modelo en espacio de estados:

$$\begin{aligned} \dot{x}_i &= x_j \\ \dot{x}_j &= M(x_i)^{-1}\left[\mathcal{T}_{in} - C(x_i,x_j)x_j - F_b x_j - G(x_i)\right] \\ y &= x_i \end{aligned} \tag{2.18}$$

donde x_i, $x_j \in \mathbb{R}^n$ son los vectores de estados de posición y velocidad angular por articulación. Además, $x_i = [x_1, x_3, x_5, \cdots, x_{i+1}]^T$ y $x_j = [x_2, x_4, x_6, \cdots, x_{j+1}]^T$.

Como ejemplo se considerará al brazo robótico de un solo eje (péndulo invertido) que es mostrado en la Figura 2.7, donde la masa m está concentrada en el extremo del eslabón l y su inercia es despreciada.

El modelo dinámico del brazo robótico de la Figura 2.7 estará representado como:

$$ml^2\ddot{\theta}_1 + mgl\cos(\theta)_1 + b\dot{\theta}_1 = \tau_{in} \tag{2.19}$$

donde los coeficientes de masa (m), gravedad (g), longitud del eslabón (l), fricción viscosa (b) y par motor (τ_{in}) son definidos como valores constantes.

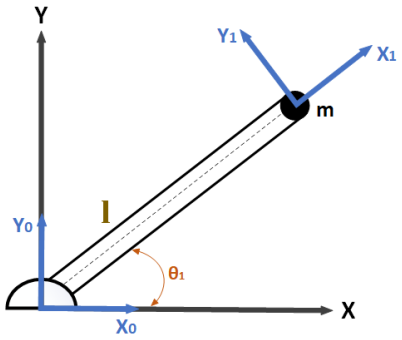


Figura 2.7: Brazo robótico de un eje

Como es notable observar, el modelo dinámico de la ecuación (2.19) es un sistema de segundo orden, por lo tanto, para representar a dicha ecuación en el espacio de estados se deben elegir las variables de estado correspondientes:

$$\begin{aligned} x_1 &= \theta_1 \\ \dot{x}_1 &= \dot{\theta}_1 = x_2 \\ \dot{x}_2 &= \ddot{\theta}_1 \end{aligned}$$

Obteniendo así, al siguiente modelo representado en el espacio de estados:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= \frac{\tau_{in} - mgl\cos(x_1) - bx_2}{ml^2} \\ y &= x_1 \end{aligned} \tag{2.20}$$

2.4.3. Definición de puntos de operación

El sistema de la ecuación (2.18) que representa al modelo general en el espacio de estados para cualquier brazo robótico, será referido como un sistema no lineal S; el cual tiene la siguiente forma genérica:

$$\dot{x} = f\left(x, u\right) \tag{2.21}$$

donde la variable de tiempo independiente, denotada t, se deja implícita. Para t ≥ 0, el vector de estados $x(t)$ siempre pertenecerá a S y el vector $u(t)$ dado por los pares motor ($\mathcal{T}_{in}$) será la entrada al sistema S en tiempo t.

Se asume que el sistema S tiene una única solución x(t) para t ≥ 0, satisfaciendo la condición inicial:

$$x(0) = x_0 \tag{2.22}$$

donde $x_0 \in \mathbb{R}^m$ es llamado el estado inicial.

El sistema S es un sistema no autónomo debido a que la entrada u(t) puede estar variando con respecto al tiempo. Si la entrada u(t) estuviese dada por entradas de par motor constantes (u^*), el sistema S llegaría a ser un sistema autónomo, por lo que el sistema presentaría soluciones constantes llamados puntos de operación, (Shilling, 2003).

Un punto de operación en cualquier sistema existe si y solo si:

$$f\left(x, u\right) = 0 \tag{2.23}$$

Un sistema autónomo no lineal puede tener un punto de operación, múltiples puntos de operación o ningún punto de operación, (Shilling, 2003).

Para determinar los puntos de operación que conlleva el modelo dinámico en espacio de estados de la ecuación (2.18), se tiene que aplicar lo definido en la ecuación (2.23). Por lo tanto, al establecer que el lado derecho de la ecuación (2.18) será igual a cero, se tiene:

$$\begin{aligned} 0 &= x_j \\ 0 &= M(x_i)^{-1}\left[\mathcal{T}_{in} - C(x_i, x_j)x_j - F_b x_j - G(x_i)\right] \end{aligned} \tag{2.24}$$

debido a que $x_j = 0$, entonces, la ecuación (2.24) quedaría expresada como:

$$M(x_i)^{-1}\left[\mathcal{T}_{in} - F_b(0) - G(x_i)\right] = 0 \tag{2.25}$$

Al multiplicar ambos lados de la ecuación (2.25) por el tensor de inercia $M(x_i)$ y resolviendo $F_b(0)$, se obtiene lo siguiente:

$$\begin{aligned} \mathcal{T}_{in} - F_b(0) - G(x_i) &= 0 \\ \mathcal{T}_{in} &= F_b(0) + G(x_i) \\ \mathcal{T}_{in} &= G(x_i) \end{aligned} \tag{2.26}$$

De la ecuación (2.26) se observa que los puntos de operación asociados con una entrada constante son determinados exclusivamente por el término de gravedad y los campos potenciales (si el sistema los tuviera) $G(x_i)$. La restricción que lo acompaña, $x_j = 0$, establece que el brazo robótico está inmóvil en los puntos de operación, ya que el vector x_j representa al vector de las velocidades conjuntas. Dependiendo de los valores de par aplicados, los puntos de operación pueden variar o incluso no existir, (Shilling, 2003).

Como ejemplo, consideremos al modelo en el espacio de estados de la ecuación (2.20); por lo tanto, de acuerdo a la ecuación (2.26), los puntos de operación para el péndulo invertido de la Figura 2.7 quedarían expresados por la siguiente ecuación:

$$\tau_{in} = mgl\cos(x_1) \tag{2.27}$$

Al despejar el estado x_1 y considerando que el brazo robótico puede tener posiciones negativas, se tiene:

$$\begin{aligned} x_1 &= \pm \arccos\left(\frac{\tau_{in}}{mgl}\right) \\ x_2 &= 0 \end{aligned} \tag{2.28}$$

Claramente, los dos puntos de operación existen si y solo si: $|\tau_{in}| \leq mgl$.

2.4.4. Linealización de sistemas no lineales

Una gran parte de la teoría de control desarrollada para el diseño de controladores automáticos emplea modelos lineales, aunque en realidad, la mayoría de los sistemas en su naturaleza describen dinámicas no lineales.

La teoría de control lineal se puede aplicar a sistemas no lineales mediante la linealización del modelo dinámico, la cual consiste en reemplazar la función no lineal del modelo en espacio de estados por una función lineal equivalente. Esta aproximación solamente está dada en una cierta región alrededor de un punto de operación x* y no en todo el dominio de la función. En la Figura 2.8 se muestra una descripción ilustrada de lo anteriormente mencionado.

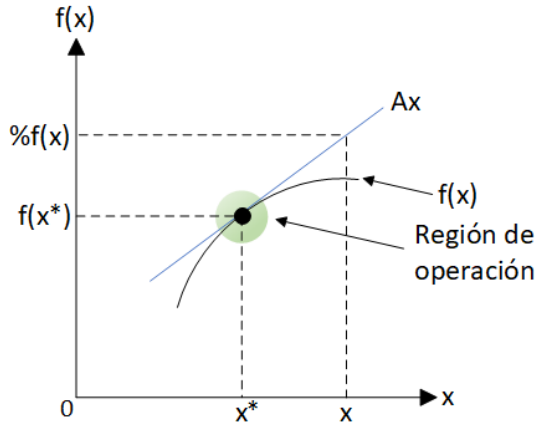


Figura 2.8: Linealización de un modelo no lineal

Considérese el siguiente modelo dinámico no lineal en espacio de estados:

$$\dot{x} = f(x) + Bu(t) \tag{2.29}$$

donde $f(x)$ representa a la función no lineal conformada por los términos de masas, fuerzas centrípetas y de Coriolis, gravedad y fricción; B representa a un vector lineal con tamaño de nxn, el cual se conforma por los valores que acompañan a la entrada $u(t)$; para el caso de un brazo robótico, la entrada $u(t)$ es el vector de pares $\mathcal{T}_{in}$.

Para aproximar un modelo no lineal a uno lineal alrededor de un punto de operación, se debe de expandir una función no lineal en series de Taylor. Al despreciar todos los términos de grado mayor a uno en la serie de Taylor precedente, se tiene:

$$f(x) \approx f(x^*) + \nabla f(x^*)(x - x^*) \tag{2.30}$$

donde $f(x^*)$ = -Bu^*, siendo u^* un vector con valores de par motor constantes; por lo tanto:

$$f(x) \approx -Bu^* + \nabla f(x^*)(x - x^*) \tag{2.31}$$

Al sustituir la ecuación (2.31) en la ecuación (2.29) y resolviendo la expresión se tiene:

$$\begin{aligned}
\dot{\bar{x}} &= -Bu^* + \nabla f(x^*)(x - x^*) + Bu(t) \\
\dot{\bar{x}} &= \nabla f(x^*)(x - x^*) + B(u - u^*) \\
\dot{\bar{x}} &= A(x - x^*) + B(u - u^*) \\
\dot{\bar{x}}(t) &= A\bar{x}(t) + B\bar{u}(t)
\end{aligned} \tag{2.32}$$

donde la ecuación (2.32) representa a la función linealizada del modelo dinámico original, siendo A la derivada de la función no lineal con respecto a sus estados y evaluado por los puntos de operación. Por lo que A es la matriz jacobiana:

$$A = J_f(x_1, x_2, \cdots, x_n) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix} \tag{2.33}$$

Para mostrar la linealización de un sistema no lineal, volveremos nuevamente al ejemplo del péndulo invertido de la Figura 2.7, donde su modelo dinámico en el espacio de estados está representado por la ecuación (2.20).

De acuerdo a la ecuación (2.29), el modelo dinámico no lineal en el espacio de estados del brazo robótico de un eje puede ser representado como:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ -\frac{mgl\cos(x_1)+bx_2}{ml^2} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{ml^2} \end{bmatrix} \tau_{in}(t) \tag{2.34}$$

donde $f(x) = \begin{bmatrix} x_2 \\ -\frac{mgl\cos(x_1)+bx_2}{ml^2} \end{bmatrix}$, es la función no lineal y $B = \begin{bmatrix} 0 \\ \frac{1}{ml^2} \end{bmatrix}$, es el vector lineal con valores constantes que acompañan a la entrada τ_{in}.

A partir de la ecuación (2.34), se puede linealizar dicho modelo del sistema no

lineal. Por tanto, siguiendo las ecuaciones (2.32) y (2.33) se tiene:

$$\begin{bmatrix} \dot{\bar{x}}_1 \\ \dot{\bar{x}}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \frac{mgl\,\mathrm{sen}(x_1)}{ml^2} & -\frac{b}{ml^2} \end{bmatrix} \begin{bmatrix} \bar{x}_1 \\ \bar{x}_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{ml^2} \end{bmatrix} \bar{\tau}_{in}(t) \tag{2.35}$$

Es así como la ecuación (2.35) representa a la función linealizada del modelo dinámico no lineal de la ecuación (2.18).

2.4.5. Estabilidad de sistemas no lineales

Considerando un sistema no lineal autónomo modelado en el espacio de estados como:

$$\dot{x}(t) = f(x) \tag{2.36}$$

donde x pertenece al dominio $D \subset \mathbb{R}^n$. Además, se supone que el sistema tiene un punto de equilibrio $x_e = 0$, es decir, $f(0) = 0$.

La estabilidad de puntos de equilibrio generalmente se caracteriza en el sentido de Lyapunov, un matemático e ingeniero ruso que estableció las bases de la teoría que hoy lleva su nombre. Un punto de equilibrio se dice estable si todas las soluciones que se inicien en las cercanías del punto de equilibrio permanecen en las cercanías del mismo; de otro modo resulta inestable. Un punto de equilibrio se dice asintóticamente estable si todas las soluciones además de permanecer en las cercanías del mismo, tienden hacia el equilibrio a medida que el tiempo se aproxima a infinito, (Khalil, 1996).

Lyapunov demostró que existen ciertas funciones que pueden ser usadas para determinar la estabilidad del punto de equilibrio de un sistema, a este teorema se le conoce como el método directo de Lyapunov.

Considerando nuevamente al sistema no lineal de la ecuación (2.36) y suponiendo que existe una función continuamente diferenciable V: $D \to \mathbb{R}$ tal que:

$$\begin{aligned} &V(0) = 0, \\ &V(x) > 0, \quad x \in D, \quad x \neq 0, \\ &\dot{V}(x) = \frac{\partial V}{\partial x} f(x) \leq 0, \quad x \in D. \end{aligned}$$

Entonces, el equilibrio $x_e = 0$ es estable en el sentido de Lyapunov. Si además se

cumple:

$$\dot{V}(x) = \frac{\partial V}{\partial x} f(x) < 0, \quad x \in D, \quad x \neq 0,$$

el equilibrio $x_e = 0$ es asintóticamente estable. Esta estabilidad será global si $D = \mathbb{R}^n$ y, además, la función V es radialmente no acotada, es decir, $\lim_{||x|| \to \infty} V(x) = \infty$. Finalmente, si existen escalares μ, ν, $\xi > 0$ y $\eta \geq 1$ tales que:

$$\mu ||x||^\eta \leq V(x) \leq \nu ||x||^\eta, \quad x \in D,$$
$$\dot{V}(x) = \frac{\partial V}{\partial x} f(x) \leq \xi V(x), \quad x \in D,$$

entonces el equilibrio $x_e = 0$ es exponencialmente estable. El concepto de estabilidad exponencial implica estabilidad asintótica y especifica la velocidad en que las soluciones convergen al equilibrio. La estabilidad exponencial será global si $D = \mathbb{R}^n$ y, además, la función V es radialmente no acotada. Lo anterior mencionado puede consultarse en Gordillo (2009).

Existe un segundo método que también permite determinar la estabilidad del punto de equilibrio de un sistema, el cual se le conoce como el método indirecto de Lyapunov. Dicho método puede ser aplicable en sistemas no lineales si se realiza la linealización del sistema alrededor de un punto de equilibrio.

De acuerdo a Seron y Braslavsky (2000), si se linealiza el sistema no lineal autónomo de la ecuación (2.36) en la vecindad del punto de equilibrio $x_e = 0$, se tiene:

$$\dot{\bar{x}} = A\bar{x}(t); \quad A = \frac{\partial f}{\partial x}(x) \mid_{x_e}, \quad \bar{x} = x - x_e \tag{2.37}$$

entonces, considerando a λ_i como los valores propios de A $(i = 1, ..., n)$, se tiene que:

a) El punto de equilibrio x_e es estable si los valores propios de A satisfacen $Re\{\lambda_i\} \leq 0$. En caso de que todos los valores propios de A tengan parte real negativa $(Re\{\lambda_i\} < 0)$, entonces, el punto de equilibrio x_e es asintóticamente estable, por lo que su respuesta natural tiende a cero cuando el tiempo tiende a infinito.

b) El punto de equilibrio x_e es inestable si uno o más valores propios de A tiene parte real positiva $(Re\{\lambda_i\} > 0)$, por lo que su respuesta natural crece ilimitadamente cuando el tiempo tiende a infinito.

c) El punto de equilibrio x_e es marginalmente estable si los valores propios de A tienen parte real negativa y al menos uno tenga un eje imaginario, por lo que no se puede concluir algo con respecto al punto de equilibrio (puede ser asintóticamente estable, estable o inestable).

En caso de que x_e sea asintóticamente estable para el sistema linealizado, entonces x_e es asintóticamente estable para el sistema no lineal original. En caso contrario, si x_e es inestable para el sistema linealizado, entonces x_e es inestable para el sistema no lineal original.

De acuerdo a Shilling (2003), para examinar la estabilidad de cualquier sistema dinámico, se necesitan conocer los posibles puntos de operación que presentan el sistema y la matriz Jacobiana A de la función linealizada. Por lo tanto, se volverá a considerar el ejemplo del brazo robótico de un eje para el desarrollo de esta sección.

Consideremos que al péndulo invertido de la Figura 2.7 se le asignará la mitad de su límite máximo del par motor: $\tau_{in} = \frac{mgl}{2}$; es así como a partir de la ecuación (2.28) se tendrían los siguientes dos puntos de operación:

$$x_1 = \pm \arccos\left(\frac{\frac{mgl}{2}}{mgl}\right), \; x_2 = 0$$

$$x = \left(\pm\frac{\pi}{3}, 0\right) \tag{2.38}$$

Posteriormente, los dos puntos de operación se evaluarán en la matriz Jacobiana A de la ecuación (2.35), por lo que se tienen dos matrices Jacobianas evaluadas:

$$Ae_{1,2} = \begin{bmatrix} 0 & 1 \\ \frac{mgl\,\mathrm{sen}(\pm\frac{\pi}{3})}{ml^2} & -\frac{b}{ml^2} \end{bmatrix} \tag{2.39}$$

Una vez obtenidas la matrices Jacobianas evaluadas por sus puntos de operación ($Ae_{1,2}$), es posible transformarlas en un polinomio característico $P(\lambda)$ mediante:

$$P(\lambda) = det(\lambda I - Ae_{1,2}) \tag{2.40}$$

donde I, en este caso, es una matriz identidad de 2x2. Al resolver la ecuación (2.40) se obtiene la siguiente representación para el polinomio característico:

$$P(\lambda) = det\left(\begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} - \begin{bmatrix} 0 & 1 \\ \frac{mgl\,\mathrm{sen}(\pm\frac{\pi}{3})}{ml^2} & -\frac{b}{ml^2} \end{bmatrix}\right)$$

$$P(\lambda) = det\left(\begin{bmatrix} \lambda & -1 \\ -\frac{mgl\,\mathrm{sen}(\pm\frac{\pi}{3})}{ml^2} & \lambda + \frac{b}{ml^2} \end{bmatrix}\right)$$

$$P(\lambda) = \lambda^2 + \frac{b}{ml^2}\lambda - \frac{mgl\,\mathrm{sen}(\pm\frac{\pi}{3})}{ml^2} \tag{2.41}$$

Una vez obtenido el polinomio característico $P(\lambda)$, se pueden calcular sus raices polinomiales (λ_i), las cuales se conocen como valores propios. Del polinomio característico de la ecuación (2.41) se tienen las siguientes raices:

$$(\lambda - \lambda_1)(\lambda - \lambda_2) = 0 \tag{2.42}$$

donde los dos valores propios quedan determinados por la siguiente ecuación:

$$\lambda_{1,2} = \frac{-\frac{b}{ml^2} \pm \left[\left(\frac{b}{ml^2}\right)^2 + \frac{4mgl\,\mathrm{sen}(\pm\frac{\pi}{3})}{ml^2}\right]^{1/2}}{2} \tag{2.43}$$

Si a la función seno del polinomio característico $P(\lambda)$ de la ecuación (2.41) se le asigna solamente el punto de operación: $x_1 = -\frac{\pi}{3}$, entonces, los dos valores propios de la ecuación (2.43) serán negativos. Por lo tanto, al tener parte real negativa ambos valores propios, se determina que el sistema es asintóticamente estable.

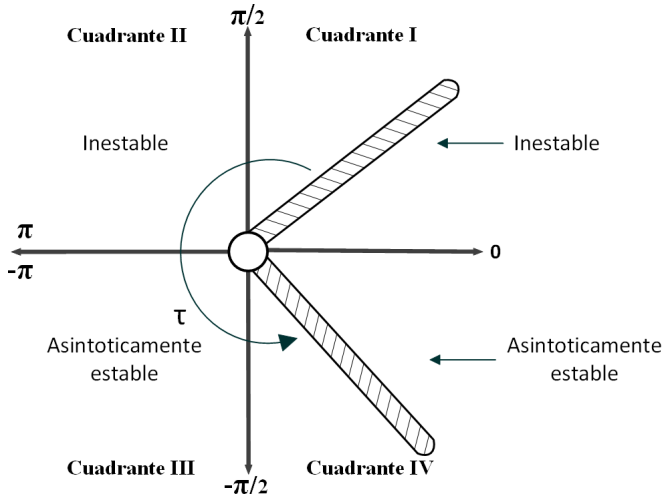


Figura 2.9: Estabilidad del brazo robótico de un eje

Con esto se determina que los puntos de operación pueden localizarse en el rango de posición: $-\pi < x_1 < 0$, para que el sistema pueda ser asintóticamente estable.

En la Figura 2.9 se muestra la representación de lo anteriormente mencionado.

Al usar el modelo dinámico en lazo abierto, el espacio de trabajo para el péndulo invertido quedaría limitado ya que solo podría operar en posiciones que estén dentro de los cuadrantes III y IV (región de estabilidad) y no para posiciones que estén dentro de los cuadrantes I y II de su espacio bidimensional, debido a que en esa región la estabilidad es nula. Esto representa un problema ya que un péndulo invertido debe presentar estabilidad para ambas regiones, por lo tanto, se tiene que encontrar una técnica de control adecuada que resuelva el problema de control de posición.

2.4.6. Control por retroalimentación de estados

Considérese un sistema lineal e invariante en el tiempo (LTI) con modelo en espacio de estados:

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) \end{aligned} \tag{2.44}$$

donde el par $(A,\ B)$ es controlable.

El control por retroalimentación de estados consiste en proponer la siguiente ley de control:

$$u(t) = Kx(t) \tag{2.45}$$

lo que permitirá que las dinámicas del sistema en lazo cerrado sean asintóticamente estables, donde $K \in \mathbb{R}^{mxn}$ es la ganancia del controlador por retroalimentación de estados. Esta retroalimentación queda representada por la Figura 2.10.

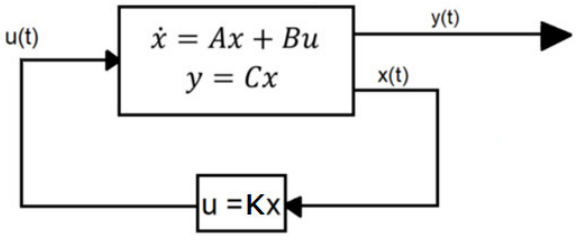


Figura 2.10: Sistema en lazo cerrado con dinámicas asintóticamente estables

Las dinámicas del sistema en lazo cerrado están dadas por:

$$\begin{aligned}\dot{x}(t) &= Ax(t) + BKx(t)\\ \dot{x}(t) &= (A + BK)x(t)\end{aligned} \tag{2.46}$$

donde la ganancia K puede ser propuesta para que los valores propios del sistema en lazo cerrado tengan parte real negativa. De la ecuación (2.46) se puede definir: $A_K = A + BK$, por lo que:

$$\dot{x}(t) = A_K x(t) \tag{2.47}$$

El problema de control por retorno de estados consiste en obtener una ganancia K adecuada para que el sistema en lazo cerrado sea asintóticamente estable.

De acuerdo a Dahleh et. al (2011), considérese la siguiente función de Lyapunov:

$$V(x) = x^T {W_1}^{-1} x \tag{2.48}$$

donde $V(x)$ será una función de Lyapunov candidata si:

1. $V(0) = 0$
2. $V(x) > 0; \qquad 0 \leq ||x|| \leq r$

Además, W^{-1} es simétrica y definida positiva. De esta forma $\dot{V}(x)$ estará dada por:

$$\dot{v}(x) = g(x)h(x)$$

siendo $g(x) = x^T$ y $h(x) = {W_1}^{-1}x$, entonces:

$$\dot{v}(x) = \frac{\partial g(x)}{\partial t}h(x) + \frac{\partial h(x)}{\partial t}g(x)$$

dando como resultado:

$$\dot{V}(x) = \dot{x}^T {W_1}^{-1} x + x^T {W_1}^{-1} \dot{x} \tag{2.49}$$

Al implementar la ecuación (2.47) en (2.49), se obtiene lo siguiente:

$$\begin{aligned}\dot{V}(x) &= (A_K x)^T {W_1}^{-1} x + x^T {W_1}^{-1}(A_K x)\\ \dot{V}(x) &= x^T {A_K}^T {W_1}^{-1} x + x^T {W_1}^{-1}(A_K x)\\ \dot{V}(x) &= x^T({A_K}^T {W_1}^{-1} + {W_1}^{-1} A_K)x\end{aligned} \tag{2.50}$$

$\dot{V}(x)$ es una función de Lyapunov si:

1. $\dot{V}(0) = 0$
2. $\dot{V}(x) < 0; \qquad 0 < ||x|| < r$

Ambas condiciones se cumplen si: $({A_K}^T {W_1}^{-1} + {W_1}^{-1} A_K) < 0$. Al sustituir la expresión de la matriz dinámica en la desigualdad negativa definida y haciendo las operaciones correspondientes, se tiene:

$$\begin{aligned} &(A + BK)^T {W_1}^{-1} + {W_1}^{-1}(A + BK) < 0 \\ &(A^T + (BK)^T){W_1}^{-1} + {W_1}^{-1}A + {W_1}^{-1}BK < 0 \\ &A^T {W_1}^{-1} + K^T B^T {W_1}^{-1} + {W_1}^{-1}A + {W_1}^{-1}BK < 0 \end{aligned} \tag{2.51}$$

La ecuación (2.51) queda representada como una desigualdad matricial lineal (LMI, del inglés: Linear Matrix Inequality). Al multiplicar la LMI por W_1 a la izquierda y a la derecha, se tiene:

$$W_1 A^T + W_1 K^T B^T + A W_1 + BK W_1 < 0$$

Puesto que ${W_1}^{-1} = [{W_1}^{-1}]^T$, entonces $W_1 = {W_1}^T$. Considerando un cambio de variable, $W_2 = KW_1$, se tiene:

$$W_1 A^T + {W_2}^T B^T + A W_1 + B W_2 < 0$$

Además, puesto que ${W_1}^{-1} > 0$, entonces, $W_1 > 0$.

Por lo tanto, la función $V(x) = x^T {W_1}^{-1} x$ es una función de Lyapunov candidata y el sistema en lazo cerrado $A_K = A + BK$ es asintóticamente estable si y solo si:

$$\begin{aligned} &{W_1}^T = W_1 \\ &W_1 > 0 \\ &A W_1 + B W_2 + W_1 A^T + {W_2}^T B^T < 0 \end{aligned} \tag{2.52}$$

Mientras que la ganancia del controlador se calcula como $K = W_2 {W_1}^{-1}$. Esta ganancia se puede implementar tanto en el modelo linealizado como en el no lineal.

Para el sistema linealizado de la ecuación (2.32), la ley de control queda como:

$$\bar{u} = K\bar{x} \tag{2.53}$$

por lo que el sistema en lazo cerrado estará representado por la figura 2.11.

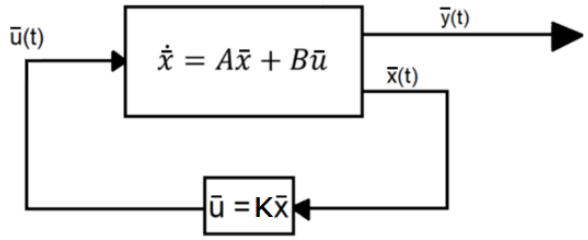


Figura 2.11: Ley de control propuesta para sistema linealizado

Para un sistema no lineal como el de la ecuación (2.29), se puede obtener una ley de control por retroalimentación de estados a partir de la ecuación (2.53), resultando:

$$\begin{aligned} u - u^* &= K(x - x^*) \\ u &= u^* + K(x - x^*). \end{aligned} \tag{2.54}$$

La Figura 2.12 muestra la representación del sistema no lineal en lazo cerrado.

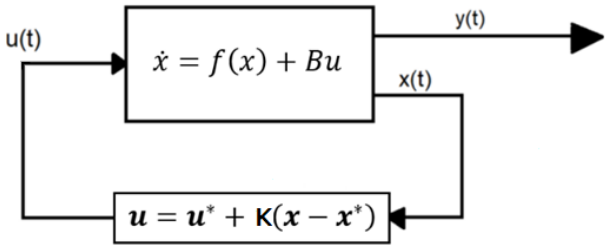


Figura 2.12: Ley de control propuesta para un sistema no lineal

Cabe señalar que la estabilidad asintótica del sistema se puede presentar únicamente en una región cercana al punto de operación x^*.

2.4.7. Regiones de estabilidad para sistemas LTI

Un sistema LTI autónomo es asintóticamente estable si las raíces de la matriz A_K tienen parte real negativa. Para muchas aplicaciones en ingeniería de control, se consideran regiones de estabilidad más específicas, (Scherer y Weiland, 2005).

Una región de estabilidad es un subconjunto $\mathbb{C}_{estab} \subseteq \mathbb{C}$ con las siguientes propiedades:

1) Si $\lambda \in \mathbb{C}_{estab}$, entonces, $\bar{\lambda} \in \mathbb{C}_{estab}$.

2) Si $\mathbb{C}_{estab}$ es convexa.

Los ejemplos típicos de regiones de estabilidad incluyen:

a) $\mathbb{C}_{estab1} = \mathbb{C}^{-}$: Semiplano imaginario izquierdo.

b) $\mathbb{C}_{estab2} = \{S \in \mathbb{C} \mid \mathbb{R}(s) < -\varphi\}$: Barra vertical.

c) $\mathbb{C}_{estab3} = \{S \in \mathbb{C} \mid |s| < \chi\}$: Circulo con centro en el origen.

d) $\mathbb{C}_{estab4} = \{S \in \mathbb{C} \mid \mathbb{R}(s)\tan(\psi) < |\mathbb{I}(S)|\}$: Región de estabilidad canónica.

Se define la región de estabilidad S como la intersección de $\mathbb{C}_{estab2}$, $\mathbb{C}_{estab3}$ y $\mathbb{C}_{estab4}$, es decir:

$$S(\varphi, \chi, \psi) = \mathbb{C}_{estab2} \cap \mathbb{C}_{estab3} \cap \mathbb{C}_{estab4} \tag{2.55}$$

La región de estabilidad S queda representada como se muestra en la Figura 2.13.

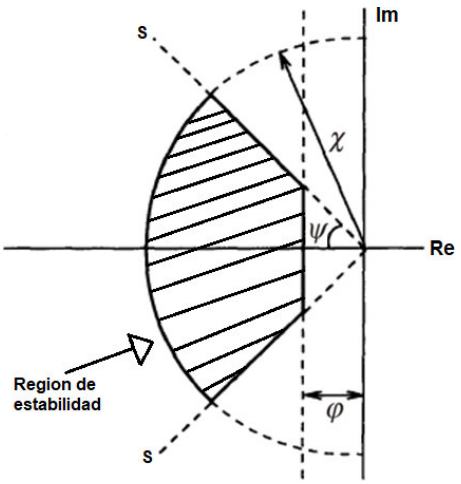


Figura 2.13: Representación de la región de estabilidad

Para agregar el concepto de región de estabilidad a un sistema de control por retroalimentación de estados, se resuelven las siguientes LMI's, (Chilai y Gahinet, 1996):

$$\begin{aligned}
&\mathbb{C}_{estab2} : A_K W_1 + W_1 {A_K}^T + 2\varphi W_1 < 0 \\
&\mathbb{C}_{estab3} : \begin{bmatrix} -\chi W_1 & A_K W_1 \\ W_1 {A_K}^T & -\chi W_1 \end{bmatrix} < 0 \\
&\mathbb{C}_{estab4} : \begin{bmatrix} \operatorname{sen}\psi[A_K W_1 + W_1 {A_K}^T] & \cos\psi[A_K W_1 - W_1 {A_K}^T] \\ \cos\psi[W_1 {A_K}^T - A_K W_1] & \operatorname{sen}\psi[A_K W_1 + W_1 {A_K}^T] \end{bmatrix} < 0
\end{aligned} \tag{2.56}$$

donde las LMI's de la ecuación (2.52) se siguen cumpliendo y además, se considera que los límites para los valores de las regiones de estabilidad son:

$$\varphi \in [0, \infty), \quad \chi > \varphi, \quad \psi = (0, \pi/2) \tag{2.57}$$

Aplicar las regiones de estabilidad son de gran importancia en sistemas de control, debido a que ψ ayuda a limitar las oscilaciones y, φ y χ modifican el tiempo de convergencia.

2.5. Control robusto

Además de las ecuaciones diferenciales no lineales que presentan los modelos dinámicos de manipuladores robóticos; se involucran también, las variaciones inciertas en los parámetros del sistema y las perturbaciones debidas a entradas desconocidas. Por lo tanto, el modelo en espacio de estados de la ecuación (2.18) quedaría modificado como:

$$\begin{aligned} \dot{x}_i &= x_j \\ \dot{x}_j &= M(x_i)^{-1} \left[\mathcal{T}_{in} - C(x_i, x_j) - F_b x_j - G(x_i) - \mathcal{T}_d \right] \\ y &= x_i \end{aligned} \tag{2.58}$$

El problema de control de posición presentado en manipuladores robóticos puede solucionarse al proponer una ley de control robusta en lazo cerrado que lleve a las dinámicas del sistema a ser asintóticamente estables y además, elimine las incertidumbres del modelo y reduzca el efecto de las perturbaciones externas, logrando que los manipuladores robóticos puedan posicionarse en cualquier coordenada requerida.

El controlador lineal H_∞ y los controladores de modo deslizante: Twisting y Super-Twisting de tercer orden, fueron propuestos para minimizar el efecto de las perturbaciones externas y las incertidumbres paramétricas.

2.5.1. Control H_∞

Un controlador lineal robusto consiste en emplear una ley de control lineal por retorno de estados para que:

a) El sistema sea asintóticamente estable.

b) La ganancia K lleve a los valores paramétricos a estar dentro de la región de estabilidad $S(\varphi, \chi, \psi)$.

c) La ganancia K minimice el efecto que pueda provocar la presencia de alguna perturbación en el sistema físico.

Por tal motivo, se describirán las LMI's y los procedimientos correspondientes para llevar a cabo lo anteior mencionado.

A continuación, se considera un sistema LTI de estructura y orden conocidos donde existen algunos parámetros que son inciertos pero acotados como se muestra en la ecuación (2.59), a esto se le conoce como incertidumbres paramétricas. En la Figura 2.14 se muestra el diagrama de un sistema en lazo cerrado con ganancia K para eliminar las incertidumbres paramétricas.

$$\begin{aligned} \dot{x}(t) &= A(p)x(t) + B(q)u(t) \\ y(t) &= C(r)x(t) \end{aligned} \tag{2.59}$$

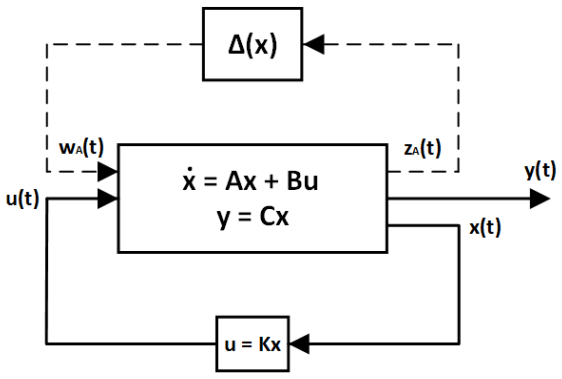


Figura 2.14: Sistema con presencia de incertidumbres paramétricas

donde las matrices A, B y C dependen y son lineales de p, q y r respectivamente, siendo estos últimos, vectores de parámetros en el sistema.

Como las incertidumbres paramétricas son acotadas, de acuerdo a Arzelier et. al (1993), es posible seleccionar los valores máximos y mínimos de las mismas, por lo tanto:

$$
\begin{aligned}
P &= \{p \in \mathbb{R}^p \mid \underline{p}_k \leq p_k \leq \overline{p}_k; \quad \underline{p}_k, \overline{p}_k : Constantes; \quad k = 1, 2, \cdots, p\} \\
Q &= \{q \in \mathbb{R}^q \mid \underline{q}_k \leq q_k \leq \overline{q}_k; \quad \underline{q}_k, \overline{q}_k : Constantes; \quad k = 1, 2, \cdots, q\} \\
R &= \{r \in \mathbb{R}^r \mid \underline{r}_k \leq r_k \leq \overline{r}_k; \quad \underline{r}_k, \overline{r}_k : Constantes; \quad k = 1, 2, \cdots, r\}
\end{aligned}
$$

Formando de tal manera un modelo politópico como el que se muestra en la Figura 2.15:

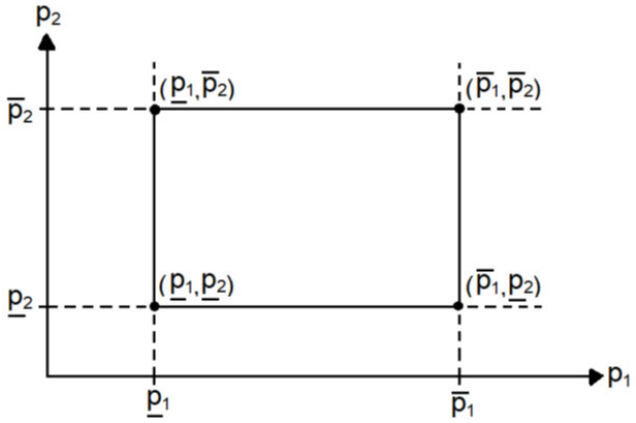


Figura 2.15: Politopo formado a partir de los valores máximos y mínimos

Además, las matrices A, B y C pertenecen a los siguientes dominios convexos:

$$
\begin{aligned}
D_A &= \{A \in \mathbb{R}^{nxn} \mid A = \sum_{i=1}^{N_p} \delta_i A_i; \quad \sum_{i=1}^{N_p} \delta_i = 1; \quad \delta_i \in (0, 1)\} \\
D_B &= \{B \in \mathbb{R}^{nxm} \mid B = \sum_{i=1}^{N_q} \varepsilon_i B_i; \quad \sum_{i=1}^{N_q} \varepsilon_i = 1; \quad \varepsilon_i \in (0, 1)\} \\
D_C &= \{C \in \mathbb{R}^{pxn} \mid C = \sum_{i=1}^{N_r} \zeta_i C_i; \quad \sum_{i=1}^{N_r} \zeta_i = 1; \quad \zeta_i \in (0, 1)\}
\end{aligned}
$$

Tal modelo politópico puede ser generado a partir de la interpolación convexa de un conjunto de modelos (A_i, B_i, C_i) identificados alrededor de diferentes puntos de operación. También puede ser obtenido a partir de modelos dependientes de los parámetros descritos por los vectores p, q y r.

Por lo tanto, el problema de control por retorno de estados de un sistema LTI incierto consiste en calcular una ganancia K para forzar a todos los valores propios de cada una de las matrices de A_K a localizarse dentro de la región de estabilidad $S(\varphi, \chi, \psi)$ para todos los valores admisibles de A y B. Es decir, para todo $A \in D_A$,

$B \in D_B$. Este problema se resuelve calculando la solución de las LMI's dadas por la ecuación (2.52) del controlador por retroalimentación de estados y la ecuación (2.56) de las regiones de estabilidad, para cada uno de los vértices del politopo convexo definido para el par incierto (A, B).

En caso de tener presencia de una entrada desconocida, considérese el siguiente sistema LTI:

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bw(t); \quad x(0) = x_0 \\ z(t) &= Cx(t) \end{aligned} \tag{2.60}$$

donde $x \in \mathbb{R}^n$ es el estado, $w \in \mathbb{R}^m$ es la entrada y $z \in \mathbb{R}^p$ es la salida. Sea además,

$$T(s) = C[sI - A]^{-1}B$$

la matriz de funciones de transferencia del sistema, que gráficamente estaría representada por la Figura 2.16.

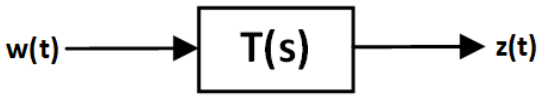


Figura 2.16: Sistema que relaciona la salida y entrada con la función de transferencia

En este caso, w es una perturbación cuyo efecto en la salida z (indicador de error) se desea minimizar. Existen muchas maneras de minimizar el efecto de w sobre z. Por ejemplo, para una entrada w dada y para la norma de una señal, el cociente $||z||/||w||$ indica la ganancia relativa que tiene la entrada w sobre la salida z. La ganancia del peor caso del sistema es:

$$||T|| = \sup_{0<||w||<\infty} \frac{||z||}{||w||} \tag{2.61}$$

la cual depende de la norma seleccionada. Con esto se puede saber el valor más grande (ganancia) del efecto w reflejado en la salida z. De preferencia se ocupa que el valor de la ganancia sea muy pequeño para que la entrada desconocida no le afecte a la salida y así la perturbación sea muy pequeña. Una medida de desempeño muy popular de

sistemas LTI estables es la norma H-infinito (H_∞) de la función de transferencia.

El problema de control H_∞ consiste en proponer una ley de control por retorno de estados donde:

a) El sistema en lazo cerrado sea asintóticamente estable.

b) La norma H_∞ de $T(s)$ (máxima ganancia de w a z) sea minimizada y además, $||T(s)||_\infty < \upsilon < 1$, donde $\upsilon \in \mathbb{R} > 0$.

c) El sistema LTI tiene que estar considerado con entrada desconocida w y salida controlada z.

Si al modelo del sistema LTI de la ecuación (2.44) se le añade la presencia de una entrada desconocida w, el modelo en espacio de estados estará dado por:

$$\begin{aligned} \dot{x}(t) &= Ax(t) + B_u u(t) + B_w w(t); \quad x(0) = x_0 \\ z(t) &= Cx(t) \end{aligned} \tag{2.62}$$

donde el sistema en lazo cerrado con ganancia K estará representado por la Figura 2.17.

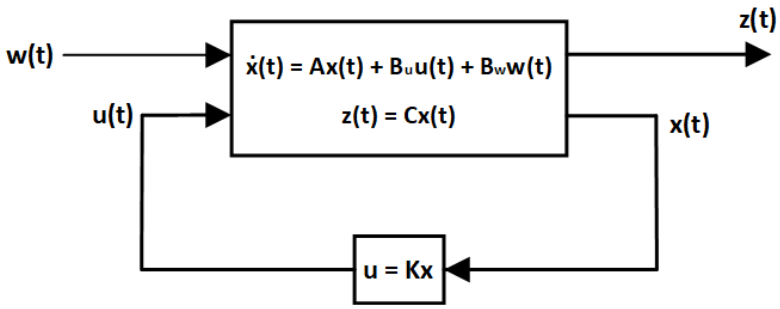


Figura 2.17: Control por retorno de estados H_∞

y la función de transferencia $T(s)$ es tal que:

$$z(s) = T(s)w(s)$$

Por lo tanto, las dinámicas del sistema en lazo cerrado están dadas por:

$$\begin{aligned}\dot{x} &= Ax + B_u K x + B_w w \\ \dot{x} &= (A + B_u K)x + B_w w \\ \dot{x} &= A_K x + B_w w \end{aligned} \tag{2.63}$$

Al aplicar la transformada de Laplace a la ecuación (2.63), es posible tener:

$$\begin{aligned} sX(s) &= A_K x(s) + B_w w(s) \\ [sI - A_K]\, x(s) &= B_w w(s) \\ x(s) &= [sI - A_K]^{-1} B_w w(s) \end{aligned} \tag{2.64}$$

y además, al aplicar la transformada de Laplace a la salida $z(t)$ de la ecuación (2.62) y sustituyendo la ecuación (2.64), se tiene:

$$\begin{aligned} z(s) &= Cx(s) \\ z(s) &= C[sI - A_K]^{-1} B_w w(s) \end{aligned} \tag{2.65}$$

Por lo tanto, a partir de la ecuación (2.65) se puede obtener la siguiente función de transferencia:

$$T(s) = C[sI - A_K]^{-1} B_w. \tag{2.66}$$

Lema Real Acotado, (Gahinet, 1996). Enuncia el lema como aparece en la bibliografía:

$$\min_{\upsilon, W_1} \quad \upsilon$$

tal que:

$$\begin{gathered} \upsilon > 0 \\ {W_1}^T = W_1 > 0 \\ \begin{bmatrix} {A_K}^T W_1 + W_1 A_K & B_w & W_1 C^T \\ {B_w}^T & -\upsilon I & 0 \\ C W_1 & 0 & -\upsilon I \end{bmatrix} < 0. \end{gathered} \tag{2.67}$$

Se propone entonces un problema de optimización semidefinida para calcular K y minimizar la norma H_∞ de $T(s)$ utilizando el Lema Real Acotado como:

$$\min_{\upsilon, W_1, W_2} \quad \upsilon$$

tal que:

$$
\begin{gathered}
\upsilon > 0 \\
{W_1}^T = W_1 > 0 \\
\begin{bmatrix}
W_1 A^T + {W_2}^T {B_u}^T + W_1 A + A W_1 + B_u W_2 & B_w & W_1 C^T \\
{B_w}^T & -\upsilon I & 0 \\
C W_1 & 0 & -\upsilon I
\end{bmatrix} < 0
\end{gathered}
\tag{2.68}
$$

donde $K = W_2 {W_1}^{-1}$ y $||T(s)||_\infty < \upsilon$.

Por lo que a partir del problema de optimización se podrá minimizar la ganancia máxima $||T(s)||_\infty$ para posteriormente minimizar el efecto de la perturbación w en la salida z.

2.5.2. Introducción al control por modos deslizantes

Considérese el siguiente modelo diferencial invariante en el tiempo:

$$\dot{x}(t) = f(x, u, w) \tag{2.69}$$

donde $x \in \mathbb{R}^n$ es el vector de estados, $u \in \mathbb{R}^m$ es la entrada de control y $w \in \mathbb{R}^q$ es la entrada desconocida debida a las perturbaciones e incertidumbres del sistema.

Las perturbaciones modifican las dinámicas de cualquier sistema y además, suelen ser desconocidas pero se asumen acotadas, es decir:

$$|w(x,t)| \leq L, \qquad L > 0$$

El problema de control es diseñar una ley de control $u(t)$ retroalimentada que lleve al sistema a la estabilidad asintótica, en otras palabras, la ley de control debe llevar a las variables de estado a cero ($x = 0$).

El presente problema de control es complicado debido a la presencia de la perturbación, aunque si la perturbación $w(x,t)$ fuese igual a cero, la ley de control llevaría al sistema a la estabilidad asintótica y a los estados a un dominio acotado alrededor del origen para $|w(x,t)| \leq L$. Por lo tanto, se implementará un controlador por modo deslizante para minimizar a la perturbación $w(x,t)$ que presenta el sistema; ver el diagrama de la Figura 2.18.

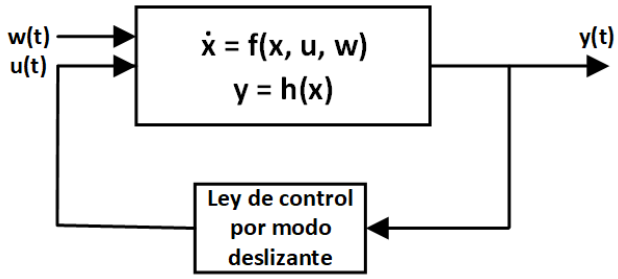


Figura 2.18: Ley de control retroalimentada por modo deslizante

En seguida, se dará una breve introducción al control por modos deslizantes; esta información se basa en el estudio de Shtessel et al. (2014). Considere que el modelo diferencial de la ecuación (2.69) presenta el siguiente modelo en espacio de estados:

$$\begin{aligned} \dot{x}_1(t) &= x_2(t); & x_1(0) &= x_{10} \\ \dot{x}_2(t) &= u(t) + w(x_1, x_2, t); & x_2(0) &= x_{20} \end{aligned} \tag{2.70}$$

Defínase las dinámicas compensadas para el caso de estudio como:

$$\begin{aligned} cx_1(t) + x_2(t) &= 0; \qquad c > 0 \\ cx_1(t) + \dot{x}_1(t) &= 0 \end{aligned} \tag{2.71}$$

donde la solución de la ecuación diferencial homogénea (2.71) está dada por:

$$\begin{aligned} x_1(t) &= x_1(0) \exp(-ct) \\ x_2(t) &= \dot{x}_1(t) = -cx_1(0) \exp(-ct) \end{aligned}$$

Defínase ahora una nueva variable en el espacio de estados del sistema como:

$$\sigma = \sigma(x_1, x_2) = cx_1(t) + x_2(t) = 0; \qquad c > 0 \tag{2.72}$$

Entonces, para lograr la convergencia asintótica de x_1 y x_2 a cero en presencia de la perturbación acotada $w(x_1, x_2, t)$, la variable σ deberá converger a cero; esto puede lograrse a partir de la entrada de control u(t). Para lograr el objetivo de control, se tienen que aplicar técnicas de la función de Lyapunov a las dinámicas de σ, las cuales están dadas por:

$$\dot{\sigma} = cx_2(t) + u(t) + w(x_1, x_2, t); \qquad \sigma(0) = \sigma_0$$

Para las dinámicas de σ se introduce una función de Lyapunov candidata que toma la forma:

$$V(\sigma) = \frac{1}{2}\sigma^2 \tag{2.73}$$

Entonces, para asegurar que las dinámicas de σ sean asintóticamente estables en el punto de equilibrio $\sigma = 0$, se deben satisfacer las siguientes condiciones:

a) $\dot{V}(\sigma) < 0$ para $\sigma \neq 0$

b) $\lim_{|\sigma|\to\infty} V(\sigma) = \infty$

La condición (b) se cumple sin problema para $V(\sigma)$ en la ecuación (2.73). La derivada de $V(\sigma)$ es calculada como:

$$\dot{V}(\sigma) = \sigma\dot{\sigma} = \sigma[cx_2(t) + u(t) + w(x_1, x_2, t)],$$

para este caso, se propone que $u(t) = -cx_2(t) + v(t)$, entonces:

$$\dot{V}(\sigma) = \sigma[v(t) + w(x_1, x_2, t)] = \sigma v(t) + \sigma w(x_1, x_2, t)$$

Puesto que $w(x_1, x_2, t)$ está acotada por L, es decir, $|w(x_1, x_2, t)| < L$ para $L > 0$, entonces:

$$\begin{aligned} \dot{V}(\sigma) &\leq \sigma v(t) + |\sigma w(x_1, x_2, t)| \\ \dot{V}(\sigma) &\leq \sigma v(t) + |\sigma||w(x_1, x_2, t)| \\ \dot{V}(\sigma) &\leq \sigma v(t) + |\sigma| L \end{aligned} \tag{2.74}$$

Proponiendo $v(t) = -\omega sign(\sigma)$ con $\omega > 0$, donde:

$$sign(x) = \begin{cases} 1 & \text{si } x > 0 \\ -1 & \text{si } x < 0 \end{cases}$$

y

$$sign(0) \in [\ -1, \quad 1\ ],$$

entonces, sustituyendo en la ecuación (2.74) se tiene:

$$\dot{V}(\sigma) \leq -|\sigma|\omega + |\sigma|L = -|\sigma|(\omega - L) \tag{2.75}$$

Claramente, si $\omega > L$, $\dot{V}(\sigma) < 0$ y por lo tanto, la estabilidad asintótica de σ estará asegurada.

Para alcanzar la convergencia $\sigma = 0$ en tiempo finito, la condición $\dot{V}(\sigma) < 0$ puede ser modificada como:

$$\dot{V}(\sigma) \leq -\kappa V^{1/2}(\sigma); \qquad \kappa > 0 \tag{2.76}$$

Al separar variables e integrando la desigualdad de la ecuación (2.76) en el intervalo de tiempo $0 \leq T \leq t$, se obtiene:

$$V^{1/2}(\sigma) \leq -\frac{1}{2}\kappa t + V^{1/2}(0)$$

En consecuencia, $V(\sigma)$ llega a cero en un tiempo finito t_r que está limitado por:

$$t_r \leq \frac{2V^{1/2}(0)}{\kappa}.$$

Por lo tanto, un control $u(t)$ que satisfaga la ecuación (2.76), llevará a la variable σ a cero en un tiempo finito y permanecerá en cero para tiempos posteriores.

Teniendo en cuenta la ecuación (2.73), la condición (2.76) se puede reescribir como:

$$\dot{V}(\sigma) \leq -\kappa V^{1/2}(\sigma) = -\frac{\kappa}{\sqrt{2}}|\sigma|; \qquad \kappa > 0 \tag{2.77}$$

Combinando las ecuaciones (2.75) y (2.77) se obtiene:

$$\dot{V}(\sigma) \leq -|\sigma|(\omega - L) \leq -\frac{\kappa}{\sqrt{2}}|\sigma|,$$

por lo que la ganancia de control ω es calculada como:

$$\omega \leq L + \frac{\kappa}{\sqrt{2}} \tag{2.78}$$

Por lo tanto, la ley de control u(t) que conduce a la variable deslizante σ a cero en tiempo finito y además, compensa a la perturbación, está dada por:

$$u(t) = -cx_2(t) - \omega sign(\sigma) \tag{2.79}$$

En base al estudio anterior, se observa que $\dot{\sigma}$ debe de estar en función de $u(t)$ para poder diseñar un controlador que estabilice a las dinámicas de σ. Además, el componente L de la ecuación (2.78) se diseña para compensar la perturbación $w(x_1, x_2, t)$, mientras que el componente $\frac{\kappa}{\sqrt{2}}$ es responsable de determinar el tiempo de convergencia a las dinámicas deseadas.

La variable σ es llamada como variable deslizante y es la que lleva a los estados del sistema a ser asintóticamente estables por medio de la superficie deslizante de la ecuación (2.72), manteniéndolos ahí en tiempos posteriores.

El control $u(t)$ de la ecuación (2.79) que lleva a los estados del sistema a la superficie deslizante en tiempo finito t_r y lo mantiene en la superficie a pesar de la perturbación $w(x_1, x_2, t)$ es llamado controlador de modos deslizantes.

Como la ley de control de la ecuación (2.79) incluye a la función signo, entonces, se presentará en ella una dinámica de zig zag de amplitud y frecuencia finita, debido a la naturaleza discreta de la función. Este efecto es llamado chattering, el cual significa oscilación; ver la Figura 2.19.

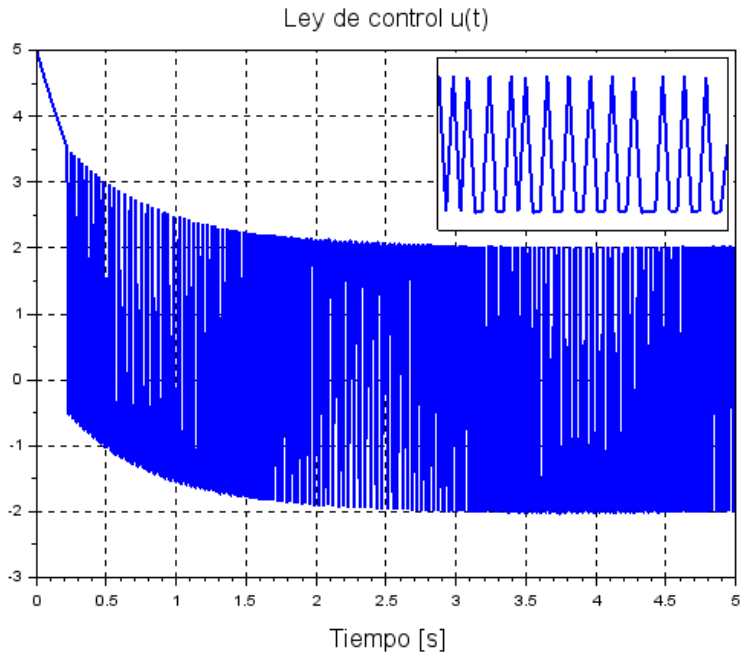


Figura 2.19: Señal del control u(t) con chattering

La ley de control anteriormente diseñada es discontinua debido a la función signo que se incluye y, por lo tanto, provoca el efecto chattering. En los sistemas de control se debe evitar la presencia de este efecto ya que, para el caso de motores de corriente continua, este tipo de oscilaciones podrían dañarlos. Por lo tanto, es conveniente diseñar controladores continuos que puedan minimizar el efecto de las perturbaciones.

Supóngase que en el tiempo t_r se alcanza la superficie deslizante $\sigma = x_2 + cx_1 = 0$ y a partir de entonces las trayectorias x_1 y x_2 permanecerán en dicha superficie por medio del controlador por modos deslizantes u(t) de la ecuación (2.79). Esto significa que $\sigma = \dot{\sigma} = 0$ para todo $t \geq t_r$. La condición $\dot{\sigma} = 0$ origina:

$$\dot{\sigma} = cx_2(t) + u(t) + w(x_1, x_2, t) = 0; \qquad \sigma(t_r) = 0$$

Una señal de control que satisface a la ecuación anterior es:

$$u_{eq}(t) = -cx_2(t) - w(x_1, x_2, t), \tag{2.80}$$

dicha función de control es llamada control equivalente y teóricamente se supone que necesita ser aplicada al sistema de la ecuación (2.70) después de que se alcanza la superficie deslizante $\sigma = 0$ para asegurar que las trayectorias del sistema permanezcan sobre la superficie deslizante para tiempos posteriores, pero en la realidad, el control equivalente no puede ser implementado debido a que depende de la perturbación desconocida.

La acción del control equivalente describe el efecto promedio del control $u(t)$ de la ecuación (2.79), una señal de conmutación de alta frecuencia. El promedio puede ser obtenido por medio de un filtro pasabajas (LPF, del inglés Low-Pass Filter) del término de conmutación de alta frecuencia $\omega sign(\sigma)$ de la ley de control $u(t)$ de la ecuación (2.79). Por lo tanto, el control equivalente puede ser estimado como:

$$\hat{u}_{eq}(t) = -cx_2(t) - \omega LPF\left(sign(\sigma)\right); \qquad t \geq t_r$$

El LPF puede implementarse como una ecuación diferencial de primer orden:

$$\dot{z}(t) = \frac{1}{\iota}\left[-z(t) + sign(\sigma)\right], \tag{2.81}$$

por lo que:

$$\hat{u}_{eq}(t) = -cx_2(t) - \omega z(t) \tag{2.82}$$

donde ι es un pequeño escalar positivo que representa la constante de tiempo del filtro. Además, la perturbación puede ser estimada como:

$$\hat{w}(t) = \omega LPF\left(sign(\sigma)\right) = \omega z(t); \qquad t \geq t_r \tag{2.83}$$

donde ω está diseñada para minimizar el efecto de la perturbación.

Ejemplo. Considere al sistema de la ecuación (2.70) con el control de modo deslizante (2.72), (2.79), las condiciones iniciales $x_1(0) = 1$, $x_2(0) = -2$, la ganancia de control $\omega = 2$, el parámetro $c = 1.5$, y la perturbación $w(x1, x2, t) = \text{sen}(2t)$; las cuales han sido utilizadas únicamente para fines de simulación. El control equivalente y la perturbación se pueden estimar usando las ecuaciones (2.81), (2.82) y (2.83) con $\iota = 0.1$. Los resultados de la simulación se presentan de la Figura 2.20 a la 2.23.

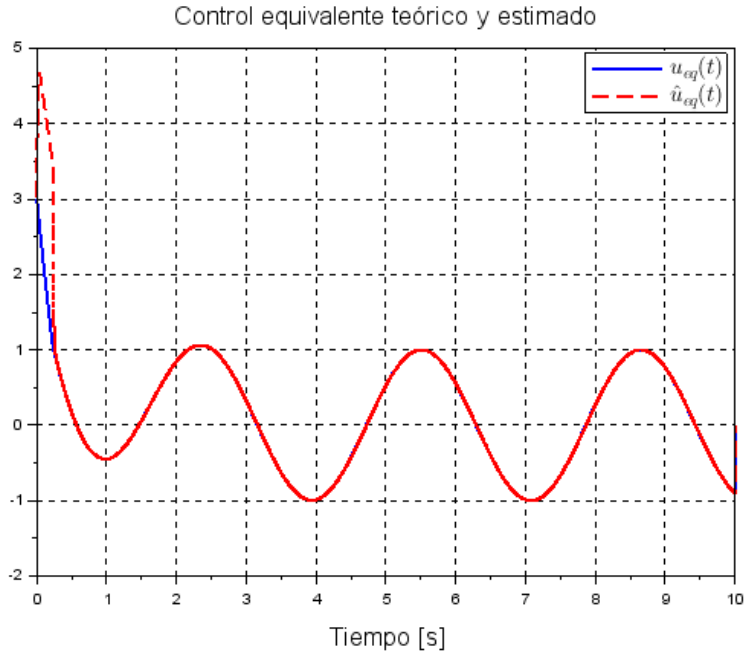


Figura 2.20: Estimación del control equivalente

En la Figura 2.20 se muestra la comparación de las señales del control equivalente; la señal de color azul representa al control equivalente de la ecuación (2.80), donde la perturbación es conocida, la señal de color roja representa al control equivalente

estimado de las ecuaciones (2.81) y (2.82). Como es notable observar, ambas señales tienen similitud y el efecto chattering se encuentra atenuado.

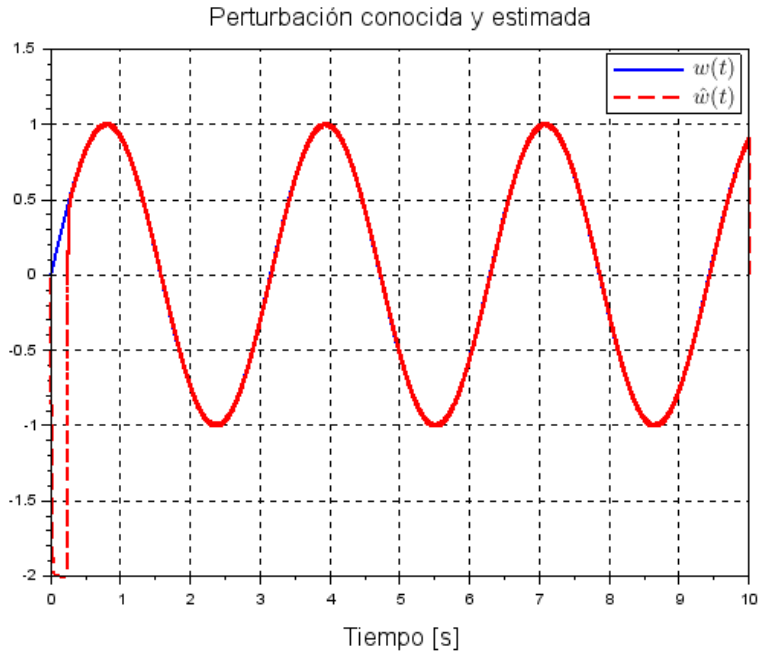


Figura 2.21: Estimación de la perturbación

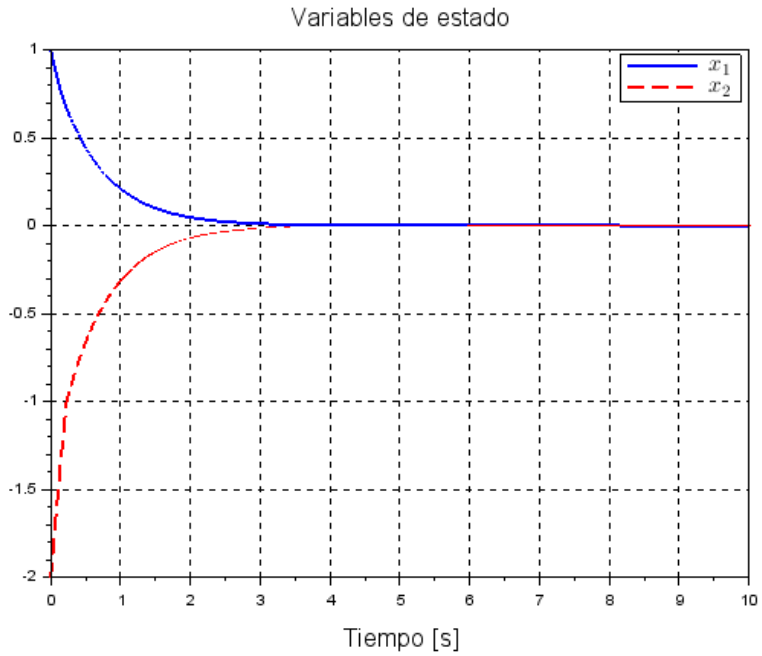


Figura 2.22: Convergencia a cero de las variables de estado

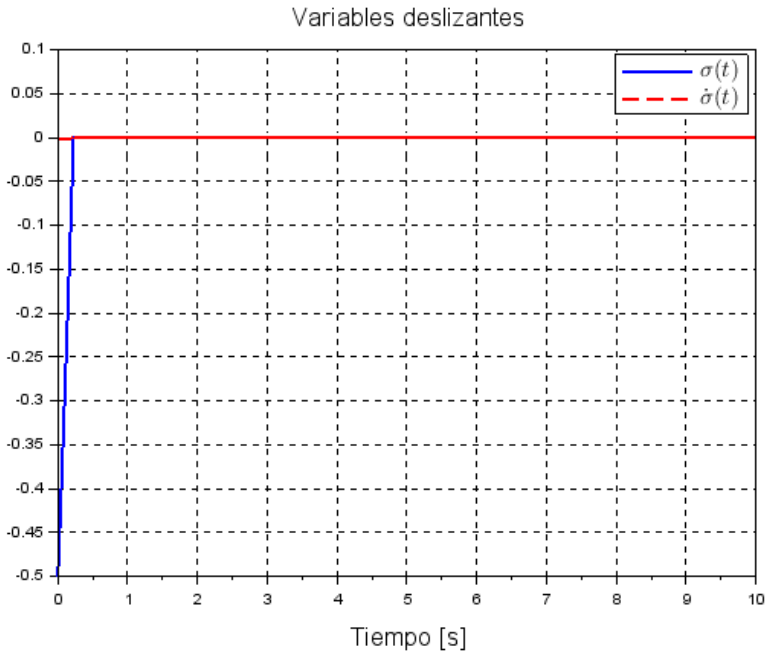


Figura 2.23: Convergencia a cero de las variables deslizantes σ y $\dot{\sigma}$

En la Figura 2.21 se muestra la comparación de las perturbaciones, la señal de color azul representa a la perturbación conocida: $w(x_1, x_2, t) = sen(2t)$ y la señal de color roja representa a la perturbación estimada por la ecuación (2.83). Ambas señales presentan el mismo comportamiento.

En la Figura 2.22 se muestra que las variables de estado exhiben convergencia a cero a medida que aumenta el tiempo, además, en la Figura 2.23 se observa que la variable deslizante σ converge a cero al igual que sus dinámicas, es por eso que los estados del sistema logran la convergencia asintótica y se mantienen ahí para tiempos posteriores.

2.5.3. Control por modos deslizantes para regulación de salida

Para este análisis nos basaremos al estudio que se muestra en Shtessel et al. (2014). Considérese nuevamente el sistema de la ecuación (2.70) donde la salida medida será el estado x_1, por lo tanto:

$$\begin{aligned} \dot{x}_1(t) &= x_2(t); & x_1(0) &= x_{10} \\ \dot{x}_2(t) &= u(t) + w(x_1, x_2, t); & x_2(0) &= x_{20} \\ y(t) &= x_1(t) \end{aligned} \tag{2.84}$$

de la ecuación (2.84), $x_1(t)$ y $x_2(t)$ son la posición y la velocidad del sistema respectivamente, $u(t)$ es la entrada de control, $w(x_1, x_2, t)$ es un término de perturbación acotada y $y(t)$ es la salida medida a controlar. Puesto que el término de perturbación es acotada, cumple con $|w(x_1, x_2, t)| \leq L$ para $L > 0$.

El problema a resolver es diseñar una ley de control por modos deslizantes en donde aun con la presencia de la perturbación $w(x_1, x_2, t)$, la salida $y(t)$ siga asintóticamente un perfil de referencia $y_r(t)$ a lo largo del tiempo.

En otras palabras, el controlador $u(t)$ lleva al error de regulación, $e(t) = y_r(t) - y(t)$, a cero asintóticamente aun con la presencia de la perturbación $w(x_1, x_2, t)$, es decir:

$$\lim_{t\to\infty} y_r(t) - y(t) = 0$$

La implementación de esta ley de control en el sistema de la ecuación (2.84) quedaría representada por el sistema en lazo cerrado que se muestra en la Figura 2.24.

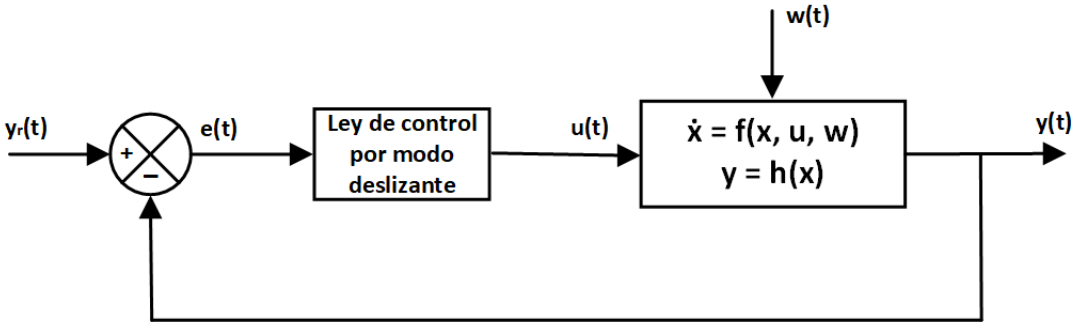


Figura 2.24: Ley de control con regulación de salida

Antes de implementar un algoritmo de control que logre la estabilidad asintótica del sistema, es importante conocer su grado relativo ρ. El grado relativo de un sistema se obtiene a partir de la n-esima derivada de la salida $y(t)$, la cual tiene que estar relacionada con la entrada $u(t)$ y de ser así, el sistema puede ser controlable.

Las dinámicas de entrada - salida del sistema analizado tienen grado relativo $\rho = 2$, debido a que la segunda derivada de y(t) relaciona a la entrada u(t):

$$\dot{y}(t) = \dot{x}_1(t) = x_2(t)$$
$$\ddot{y}(t) = \dot{x}_2(t) = u(t) + w(x_1, x_2, t)$$

Las dinámicas compensadas del error de regulación de entrada - salida para el sistema de la ecuación (2.84) pueden ser definidas como una ecuación diferencial lineal de orden $\rho - 1$ con respecto al error de regulación $e(t) = y_r(t) - y(t)$. Específicamente, se puede definir:

$$\sigma = \dot{e}(t) + ce(t); \qquad c > 0 \tag{2.85}$$

De esta manera, se puede diseñar una ley de control por modos deslizantes que lleve a la variable deslizante σ a cero en tiempo finito y mantenerla ahí para $t = \infty$. Claramente, una vez que la variable deslizante alcance el valor de cero, el modelo deslizante comienza y el error de regulación $e(t)$ obedecerá las dinámicas deseadas:

$$\sigma = \dot{e}(t) + ce(t) = 0; \qquad c > 0$$

Sí las dinámicas son asintóticamente estables, el error de regulación convergerá a cero y $y(t)$ convergerá a $y_r(t)$.

Las dinámicas de la variable deslizante σ de la ecuación (2.85) estarán dadas por:

$$\begin{aligned}
\dot{\sigma} &= \ddot{e}(t) + c\dot{e}(t) = \ddot{y}_r(t) - \ddot{y}(t) + c\left[\dot{y}_r(t) - \dot{y}(t)\right] \\
\dot{\sigma} &= \ddot{y}_r(t) - \ddot{x}_1(t) + c\left[\dot{y}_r(t) - \dot{x}_1(t)\right] \\
\dot{\sigma} &= \ddot{y}_r(t) - \dot{x}_2(t) + c\dot{y}_r(t) - cx_2(t) \\
\dot{\sigma} &= \ddot{y}_r(t) + c\dot{y}_r(t) - cx_2(t) - u(t) - w(x_1, x_2, t)
\end{aligned} \tag{2.86}$$

sea $\Psi(x,t) = \ddot{y}_r(t) + c\dot{y}_r(t) - cx_2(t) - w(x_1, x_2, t)$, entonces:

$$\dot{\sigma} = \Psi(x,t) - u(t) \tag{2.87}$$

Puesto que $y_r(t)$ es una función acotada, $\dot{y}_r(t)$ y $\ddot{y}_r(t)$ son acotados, además si el sistema compensado es asintóticamente estable, $x_2(t)$ también es acotada, por lo tanto, $\Psi(x,t)$ puede ser asumida acotada, es decir $|\Psi(x,t)| \leq M$ para $M > 0$.

En base a lo anterior, se puede diseñar una ley de control por modos deslizantes usando la condición de estabilidad en tiempo finito de la ecuación (2.76):

$$\dot{V}(\sigma) \leq -\kappa V(\sigma)^{1/2}; \qquad \kappa > 0$$

y la función de Lyapunov candidata de la ecuación (2.73):

$$V(\sigma) = \frac{1}{2}\sigma^2$$

Por lo tanto:

$$\dot{V}(\sigma) \leq -\frac{\kappa}{\sqrt{2}}|\sigma|$$

el cual ya había sido obtenido en la ecuación (2.77).

Dado que $|\Psi(x,t)| \leq M$, entonces, la derivada de la ecuación (2.73) es:

$$\begin{aligned}
\dot{V}(\sigma) &= \sigma\dot{\sigma} = \sigma\left[\Psi(x,t) - u(t)\right] \\
\dot{V}(\sigma) &\leq |\sigma\Psi(x,t)| - \sigma u(t) \\
\dot{V}(\sigma) &\leq |\sigma||\Psi(x,t)| - \sigma u(t) \\
\dot{V}(\sigma) &\leq |\sigma|M - \sigma u(t)
\end{aligned}$$

al proponer $u(t) = \varepsilon sign(\sigma)$, se tiene:

$$\begin{aligned}
\dot{V}(\sigma) &\leq |\sigma|M - \varepsilon\sigma sign(\sigma) \\
\dot{V}(\sigma) &\leq |\sigma|(M - \varepsilon)
\end{aligned} \tag{2.88}$$

Para que $\dot{V}(\sigma) < 0$, se tiene que cumplir que $\varepsilon > M$. Al igualar las ecuaciones (2.77) y (2.88) se tiene que:

$$|\sigma|(M - \varepsilon) \leq -\frac{\kappa}{\sqrt{2}}|\sigma|$$

por lo tanto, la ganancia de control ε es calculada como:

$$\varepsilon \geq M + \frac{\kappa}{\sqrt{2}} \tag{2.89}$$

Ejemplo. Considere al sistema de la ecuación (2.84) con el control de modo deslizante convencional $u(t) = \varepsilon sign(\sigma)$ y las ecuaciones (2.85), (2.89), las condiciones iniciales $x_1(0) = 1$, $x_2(0) = -2$, la ganancia de control $\varepsilon = 6$, el parámetro $c = 1.5$, la salida del perfil de referencia $y_r(t) = 2\cos(t)$, y la perturbación $w(x1, x2, t) = \text{sen}(2t)$; las cuales han sido utilizadas únicamente para fines de simulación. Los resultados de la simulación se presentan de la Figura 2.25 a la 2.27.

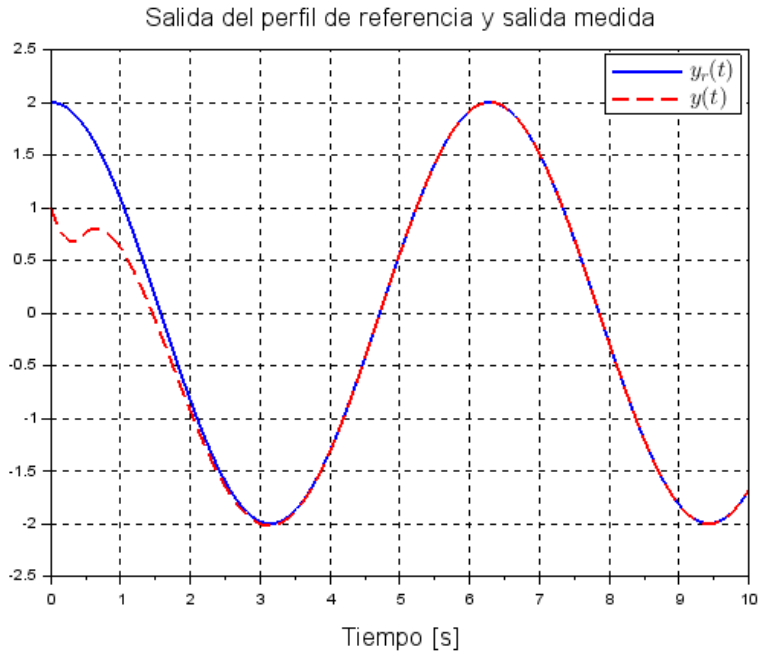


Figura 2.25: Comparación de la salida de referencia y_r con la salida medida y

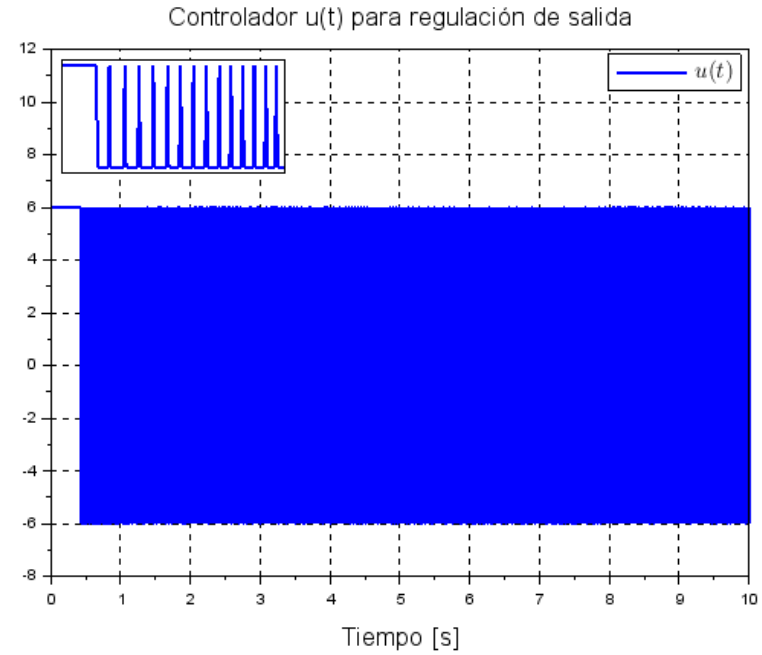


Figura 2.26: Señal del controlador $u(t)$ para regulación de salida

En la Figura 2.25 se muestra la comparación de la salida de referencia $y_r(t)$ con la salida real $y(t)$. La señal de color azul representa a la salida de perfil de referencia y la señal de color roja representa a la salida regulada (posición) por el control de

modo deslizante convencional $u(t)$ a pesar de la presencia de la perturbación.

En la Figura 2.26 se muestra la señal del controlador u(t) que compensa a las dinámicas en tiempo finito a pesar de la perturbación $w(x_1, x_2, t)$. Como es notable observar, el chattering está presente en la señal de control debido a la función signo empleada. Algo interesante de analizar sería resolver el mismo problema de control pero tratando de eliminar el Chatering producido.

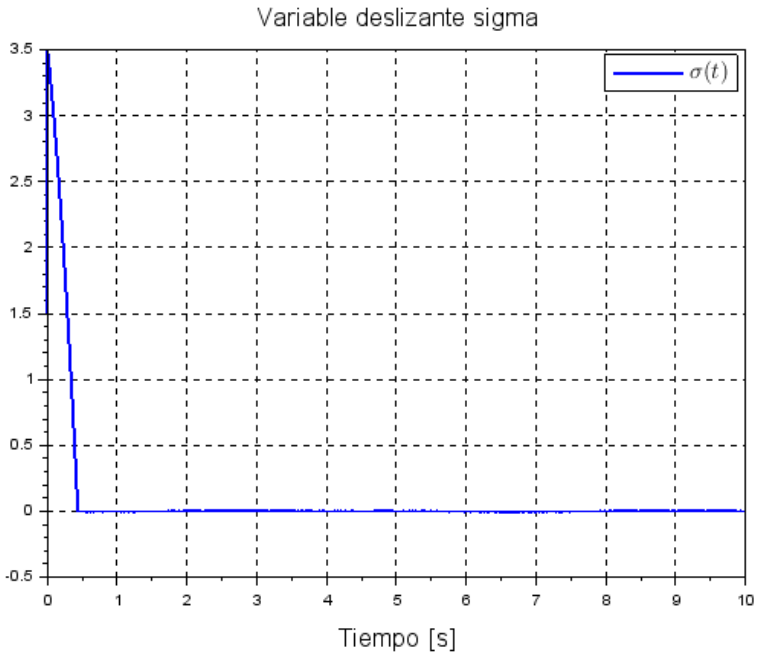


Figura 2.27: Señal de la variable deslizante σ

En la Figura 2.27 se muestra la señal de la variable deslizante σ, la cual alcanza el valor cero en el tiempo finito $t_r \leq 0.7$ y desde entonces el modo deslizante en el sistema seguirá presente.

2.5.4. Controlador twisting

Considérese el sistema afín modelado en el espacio de estados como:

$$\dot{x}(t) = a(t, x) + b(t, x)u(t) \tag{2.90}$$

donde $x \in \mathbb{R}^n$ y con salida:

$$y(t) = \sigma(t, x) \tag{2.91}$$

por lo que se asume que las funciones a, b y σ tienen todas las derivadas posibles y además, u, $\sigma \in \mathbb{R}$. La derivada de σ con respecto al tiempo está dada por:

$$\dot{\sigma}(t,x) = \frac{\partial \sigma}{\partial t} + \frac{\partial \sigma}{\partial x} a(t,x) + \frac{\partial \sigma}{\partial x} b(t,x) u(t)$$

En caso de que $\frac{\partial \sigma}{\partial x} b(t,x) \neq 0$, entonces, el grado relativo $\rho = 1$. En caso contrario, si $\frac{\partial \sigma}{\partial x} b(t,x) = 0$, entonces, $\rho > 1$, por lo que al derivar nuevamente $\dot{\sigma}$ con respecto al tiempo se tiene:

$$\begin{aligned} \ddot{\sigma} =& \frac{\partial^2 \sigma}{\partial t^2} + 2\frac{\partial^2 \sigma}{\partial t \partial x} a(t,x) + \left[\frac{\partial^2 \sigma}{\partial x^2}(a(t,x) + b(t,x)u(t))\right] a(t,x) \; + \\ & \frac{\partial \sigma}{\partial x}\frac{\partial a}{\partial t} + \frac{\partial \sigma}{\partial x}\left[\frac{\partial a}{\partial x}(a(t,x) + b(t,x)u(t))\right] \end{aligned}$$

La ecuación anterior se puede definir como:

$$\ddot{\sigma} = f(t,x) + g(t,x)u(t) \tag{2.92}$$

donde $f(t,x)$ es una función definida que está dada por:

$$f(t,x) = \frac{\partial^2 \sigma}{\partial t^2} + 2\frac{\partial^2 \sigma}{\partial t \partial x} a(t,x) + \frac{\partial \sigma}{\partial x}\frac{\partial a}{\partial t} + \left[\frac{\partial^2 \sigma}{\partial x^2} a(t,x)\right] a(t,x) + \frac{\partial \sigma}{\partial x}\frac{\partial a}{\partial x} a(t,x)$$

y

$$g(t,x) = \left[\frac{\partial^2 \sigma}{\partial x^2} b(t,x)\right] a(t,x) + \frac{\partial \sigma}{\partial x}\left[\frac{\partial a}{\partial x} b(t,x)\right].$$

En caso de que:

$$g(t,x)u(t) \neq 0$$

entonces, $\rho = 2$.

Suponiendo que existe el grado relativo en el sistema y la función de control $u(t)$ está definida por retroalimentación discontinua, entonces, con grado relativo 1 la función $\dot{\sigma}$ es discontinua, mientras σ es continua; en el caso del grado relativo 2, las funciones σ, $\dot{\sigma}$ serán continuas, mientras que $\ddot{\sigma}$ es discontinua. De acuerdo a Shtessel et. al (2014), es así como un modo deslizante de segundo orden requiere tener grado relativo 2 con respecto al control discontinuo.

El controlador twisting fue el primer controlador de modo deslizante de segundo orden propuesto, el cual tiene como propósito garantizar la aparición de un modo deslizante de segundo orden $(\sigma, \dot{\sigma})$ para llevar al origen a las trayectorias de las dinámicas de segundo orden de la variable deslizante σ en tiempo finito y así estabilizar el sistema, (Shtessel et. al, 2014). Este controlador está definido como:

$$\begin{aligned} u(t) &= -\left(r_1 sign(\sigma) + r_2 sign(\dot{\sigma})\right); \quad r_1 > r_2 > 0 \\ \sigma(t,x) &= e(t) = y_r(t) - y(t) \end{aligned} \tag{2.93}$$

donde la función $\sigma(t,x)$ satisface a la inclusión diferencial:

$$\ddot{\sigma}(t,x) \in [-C, C] + [K_m, K_M]\, u(t) \tag{2.94}$$

donde C, K_m, $K_M \in \mathbb{R}$ y además, r_1 y r_2 son constantes que deben satisfacer las siguientes condiciones:

$$\begin{gathered} (r_1 + r_2)K_m - C > (r_1 - r_2)K_M + C \\ (r_1 - r_2)K_m > C \end{gathered} \tag{2.95}$$

De acuerdo a las ecuaciones (2.90) y (2.91) del sistema afín modelado, el controlador twisting de la ecuación (2.93) estaría representado por la Figura 2.28.

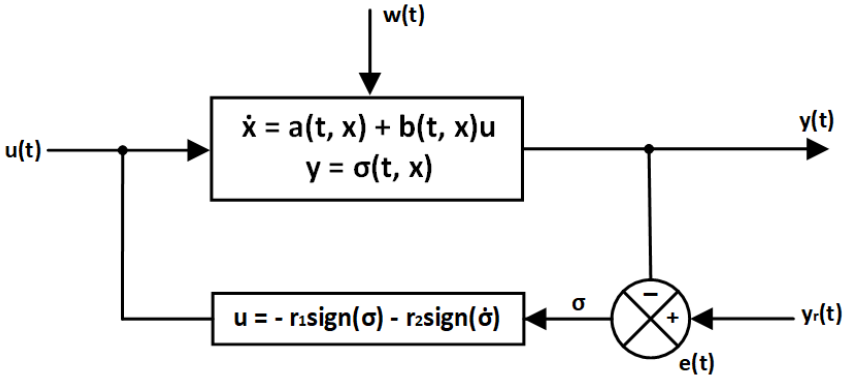


Figura 2.28: Retroalimentación del controlador twisting

Debido a que la entrada de control de la ecuación (2.93) está definida por dos funciones signo, se presentarán oscilaciones de switcheo de alta frecuencia (chattering); por lo tanto, la señal de control será discontinua.

2.5.5. Controlador super-twisting de tercer orden

El algoritmo de control super-twisting de tercer orden (3-AST) tiene características combinadas de los algoritmos twisting y super-twisting de segundo orden, por lo que el algoritmo 3-AST es aplicable a sistemas cuyas dinámicas de entrada - salida tienen grado relativo igual a 2. En la Tabla 2.1 se muestra una comparación del algoritmo 3-AST con respecto al resto de los algoritmos de modo deslizante:

Algoritmo	Caracteristicas		
	Convergencia	Control	Sistemas
Primer Orden	Exponencial	Discontinuo	Grado Relativo 2
Twisting	Tiempo Finito	Discontinuo	Grado Relativo 2
Super-Twisting	Exponencial	Continuo	Grado Relativo 1
3-STA	Tiempo Finito	Continuo	Grado Relativo 2

Tabla 2.1: Comparación de controladores por modo deslizante

De acuerdo a la Tabla 2.1, se indica que el control 3-AST proporciona una señal continua y además, los estados convergen hacía la superficie deslizante en tiempo finito en sistemas de grado relativo dos, (Ruiz y Fridman, 2014). Por lo tanto, la ley de control propuesta que soluciona el problema de regulación de salida es:

$$\begin{aligned} u &= -k_1|\sigma|^{\frac{1}{2}} sign(\sigma) + L \\ \dot{L} &= -k_3 sign(\sigma) \\ \sigma &= \dot{e} + k_2|e|^{\frac{2}{3}} sign(e) \end{aligned} \tag{2.96}$$

donde los valores de las ganancias de control k_1, k_2 y k_3 tienen que ser positivas y deben escogerse apropiadamente, además, el error de regulación $e(t)$ está definido por la diferencia de la salida del perfil de referencia $y_r(t)$ y la salida medida $y(t)$, es decir:

$$e(t) = y_r(t) - y(t)$$

La ley de control de la ecuación (2.96) con retroalimentación de salida, permite que las salidas $y(t)$ de un brazo robótico (posiciones angulares) puedan seguir asintóticamente las salidas de referencia $y_r(t)$ a lo largo de un tiempo t establecido para lograr ubicar al extremo final del robot en una posición requerida, aun con la presencia de las perturbaciones y variaciones paramétricas en el sistema; por lo que el controlador llevará al error de regulación $e(t)$ a cero asintóticamente. La salidas de referencia $y_r(t)$

se pueden generar mediante el desarrollo de una planificación de trayectorias.

Por lo tanto, de acuerdo a Franco (2014), las ganancias de control tienen la siguiente función en un sistema robótico:

k_1: Regula más rápido la posición angular cuando su error tiende a ser grande.

k_2: Tiene efectos sobre la velocidad; en caso de que se requiera tener un mejor control de ella, la ganancia debe incrementarse.

k_3: Sirve para el rechazo de perturbaciones, por lo que el valor tiene que elegirse mayor a la cota de la derivada de la perturbación Δ.

Para seleccionar el valor correcto de la ganancia k_3, es necesario conocer la cota de la derivada de la perturbación (Δ). La cota es calculada como el valor absoluto de la derivada de dicha perturbación:

$$\Delta = |\dot{w}_{max}|$$

Por ende, el valor de la ganancia k_3 debe de cumplir la siguiente condición:

$$k_3 > \Delta \tag{2.97}$$

Para realizar el análisis de estabilidad como lo indica Franco (2014), se va a considerar el siguiente sistema SISO (Single-Input Single-Output):

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= u + f(x_1, x_2, w) \\ y &= x_1 \end{aligned} \tag{2.98}$$

donde los estados x_1 y x_2 representan a la posición y a la velocidad respectivamente, u $\in \mathbb{R}$ representa la entrada de control, $f(x_1, x_2, w)$ es la función no lineal con perturbaciones e incertidumbres acopladas que pueden afectar al sistema, y la posición será la salida a medir.

El objetivo es implementar un controlador continuo para regulación de salida en lazo cerrado que estabilice en tiempo finito al sistema dinámico, aún con la presencia

de perturbaciones e incertidumbres paramétricas.

Al implementar un algoritmo de control super-twisting de tercer orden al sistema dinámico de la ecuación (2.98), se obtiene el siguiente sistema en lazo cerrado:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -k_1|\sigma|^{\frac{1}{2}} sign(\sigma) + L + f(x_1, x_2, w) \\ \dot{L} &= -k_3 sign(\sigma) \end{aligned} \tag{2.99}$$

donde $\sigma = \dot{e} + k_2|e|^{\frac{2}{3}} sign(e)$ y los términos k_1, k_2 y k_3 son ganancias positivas diseñadas adecuadamente.

Al suponer que $x_3 = \mathrm{L} + f(x_1, x_2, w)$, el sistema de la ecuación (2.99) puede reescribirse como:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -k_1|\sigma|^{\frac{1}{2}} sign(\sigma) + x_3 \\ \dot{x}_3 &= -k_3 sign(\sigma) + \psi \end{aligned} \tag{2.100}$$

donde $\psi = \dot{f}(x_1, x_2, w)$ y se satisface que:

$$|\psi| \leq \Delta$$

Por lo tanto, el sistema diferencial con lado derecho discontinuo de la ecuación (2.100), puede estabilizar al sistema en el origen en tiempo finito aun en presencia de la perturbación acotada: $|\psi| \leq \Delta$.

Además, de acuerdo a Kamal et. al (2014), para poder llevar a cabo el análisis de estabilidad de Lyapunov, se utiliza la siguiente notación: Para una variable real $z \in \mathbb{R}$ elevada a una potencia real $p \in \mathbb{R}$, $\lfloor z \rceil^p = |z|^p sign(z)$, por ejemplo, $\lfloor z \rceil^2 = |z|^2 sign(z) \neq z^2$. Si p es un número non, esta notación no cambia el significado de la ecuación, i.e., $\lfloor z \rceil^p = z^p$. Por lo tanto:

$$\begin{aligned} &\lfloor z \rceil^0 = sign(z), \;\; \lfloor z \rceil^0 z^p = |z|^p, \;\; \lfloor z \rceil^0 |z|^p = \lfloor z \rceil^p \\ &\lfloor z \rceil^p \lfloor z \rceil^q = |z|^p sign(z) |z|^q sign(z) = |z|^{p+q} \end{aligned} \tag{2.101}$$

Una vez definida la notación de la ecuación (2.101), Kamal et. al, (2014) propone la siguiente función de Lyapunov para el sistema en lazo cerrado:

$$\begin{aligned} V(x) = p_1|x_1|^{\frac{4}{3}} - p_{12}\lfloor x_1 \rceil^{\frac{2}{3}} \left(x_2 + k_2\lfloor x_1 \rceil^{\frac{2}{3}}\right) \\ +p_2|x_2 + k_2\lfloor x_1 \rceil^{\frac{2}{3}}|^2 + p_{13}\lfloor x_1 \rceil^{\frac{2}{3}}\lfloor x_3 \rceil^2 \\ -p_{23}\left(x_2 + k_2\lfloor x_1 \rceil^{\frac{2}{3}}\right)\lfloor x_3 \rceil^2 + p_3|x_3|^4 \end{aligned} \tag{2.102}$$

la cual puede expresarse en forma cuadrática con el vector:

$$\Xi^T = [|x_1|^{\frac{2}{3}} sign(x_1) \quad \phi \quad |x_3|^2 sign(x_3)]$$

por lo que:

$$V(x) = \Xi^T P \Xi, \quad P = \begin{bmatrix} p_1 & -\frac{1}{2}p_{12} & \frac{1}{2}p_{13} \\ -\frac{1}{2}p_{12} & p_2 & -\frac{1}{2}p_{23} \\ \frac{1}{2}p_{13} & -\frac{1}{2}p_{23} & p_3 \end{bmatrix} \tag{2.103}$$

A partir de la ecuación (2.103) se entiende que V(x) es positiva definida y radialmente no acotada sí y solo sí P > 0, cuyos elementos deben cumplir lo siguiente:

$$p_1 > 0$$
$$p_1 p_2 > \frac{1}{4}p_{12}^2$$
$$p_1\left(p_2 p_3 - \frac{1}{4}p_{23}^2\right) + \frac{p_{12}}{2}\left(-\frac{p_{12}p_3}{2} + \frac{p_{13}p_{23}}{4}\right) + \frac{p_{13}}{2}\left(\frac{p_{12}p_{23}}{4} - \frac{p_2 p_{13}}{2}\right) > 0$$

Por otro lado, la derivada de la función de Lyapunov $\dot{V}(x)$ estaría dada por:

$$\begin{aligned} \dot{V}(x) = q_1\lfloor x_1 \rceil^{\frac{1}{3}} x_2 - q_2|x_1|^{-\frac{1}{3}}x_2^2 - 2k_1 p_2|\phi|^{\frac{3}{2}} - p_{23}|x_3|^3 \\ -q_3|x_1|^{-\frac{1}{3}}x_2\lfloor x_3 \rceil^2 + k_1 p_{12}\lfloor x_1 \rceil^{\frac{2}{3}}\lfloor \phi \rceil^{\frac{1}{2}} - \bar{q_4}\lfloor x_1 \rceil^{\frac{2}{3}} x_3 \\ +\bar{q_5}x_3\phi + p_{23}k_1\lfloor \phi \rceil^{\frac{1}{2}}\lfloor x_3 \rceil^2 - \bar{q_6}\lfloor x_3 \rceil^3\lfloor \phi \rceil^0 \end{aligned} \tag{2.104}$$

donde:

$$q_1 = \left(\frac{4p_1}{3} - \frac{4k_2 p_{12}}{3} + \frac{4p_2 k_2^2}{3}\right)$$

$$
\begin{aligned}
q_2 &= \left(\frac{2p_{12}}{3} - \frac{4p_2k_2}{3}\right) \\
q_3 &= \left(\frac{2p_{23}k_2}{3} - \frac{2p_{13}}{3}\right) \\
\bar{q}_4 &= \left(p_{12} + 2p_{13}k_3\lfloor\phi\rceil^0\lfloor x_3\rceil^0 - 2p_{13}\lfloor x_3\rceil^0\rho\right) \\
\bar{q}_5 &= \left(p_{12} + 2p_{23}k_3\lfloor\phi\rceil^0\lfloor x_3\rceil^0 - 2p_{23}\lfloor x_3\rceil^0\rho\right) \\
\bar{q}_6 &= \left(4k_3p_3 - 4p_3\rho\lfloor\phi\rceil^0\right)
\end{aligned}
$$

La función $\dot{V}(x)$ dada por la ecuación (2.104), será negativa definida si las ganancias cumplen con lo siguiente:

$$
\begin{aligned}
p_1 + p_2k_2^2 &> k_2p_{12} \\
p_{12} &= 2p_2k_2 \\
p_{23}k_2 &= p_{13} \\
p_{12} &> 2p_{13}k_3 \\
2p_2 &> 2p_{23}k_3 \\
k_3 &> 0
\end{aligned}
$$

2.6. Resumen del segundo capítulo

Para estudiar los movimientos de un brazo robótico es importante realizar el análisis cinemático, el cual permite calcular la posición, velocidad y aceleración de cada junta articular del robot sin considerar las fuerzas que originan el movimiento. El análisis cinemático se deriva en cinemática directa y cinemática inversa, donde el primero plantea conocer las posiciones y orientaciones del efector final con base a las posiciones angulares de cada junta, y el segundo calcula las posiciones angulares de cada junta para ubicar al efector final en una posición requerida.

Como el objetivo principal de un brazo robótico es moverse de un punto inicial a un punto final, entonces, es necesario planificar una trayectoria, donde se especifica una ruta con un algoritmo polinomial y se implementan las ecuaciones de la cinemática inversa. La ruta se compone por el conjunto de puntos por donde tiene que pasar el extremo final del brazo robótico para llegar a su destino y el algoritmo polinomial con la cinemática inversa permiten relacionar a cada punto de la ruta con un cam-

bio gradual de velocidad y aceleración al inicio y final del recorrido, lo cual brinda suavidad de movimiento en cada junta articular.

Opuesto a la cinemática, la dinámica calcula la posición, velocidad y aceleración de cada junta articular de un brazo robótico con relación a las fuerzas que son aplicadas. Los brazos robóticos pueden describirse mediante modelos dinámicos complejos, los cuales pueden contener un sistema simultáneo de ecuaciones diferenciales no lineales de segundo orden que involucren a las masas inerciales, las fuerzas de Coriolis y centrípetas, las incertidumbres paramétricas debidas a las fricciones articulares y la fuerza gravitacional, los momentos de fuerzas y la presencia de perturbaciones.

Debido a que los modelos dinámicos de este tipo de robots son muy complejos, entonces, la planificación de trayectorias con el uso del modelo dinámico en lazo abierto es una tarea complicada, ya que la estabilidad de estos sistemas robóticos puede presentarse solamente en regiones de operación que se encuentren fuera del espacio de trabajo requerido; siendo este caso el problema de control que se trata de resolver en este tipo de sistemas.

El problema del control de la posición puede resolverse al proponer una ley de control robusta en lazo cerrado, tal que los estados del sistema robótico puedan ser estables en tiempo finito y que el efecto de las incertidumbres del modelo y de las perturbaciones sea reducido. El controlador lineal H_∞ y los controladores por modo deslizante twisting y super-twsiting de tercer orden, pueden introducirse en lazo cerrado con el modelo dinámico del brazo robótico (representado en el espacio de estados) para que el robot pueda moverse a una posición objetivo dentro de su espacio de trabajo aun en presencia de perturbaciones e incertidumbres paramétricas.

Dado a que el controlador lineal H_∞ solo se puede implementar en sistemas dinámicos lineales, entonces, un modelo dinámico no lineal tiene que ser linealizado previamente para obtener un modelo lineal aproximado alrededor de un punto de operación. Los controladores twisting y super-twisting de tercer orden son controladores ideales para sistemas dinámicos no lineales, estos controladores dependen de una variable deslizante que deberá converger a cero para que los estados del sistema dinámico sean estables en tiempo finito.

Capítulo 3

Metodología

En este capítulo se muestra el desarrollo y la simulación de la cinemática y dinámica de un brazo robótico de tres grados de libertad (3-GDL), además, se realiza la simulación de un controlador lineal aplicado en un péndulo invertido simple para determinar la fiabilidad que presenta este tipo de control en un sistema linealizado que originalmente es no lineal.

3.1. Análisis cinemático del brazo robótico

El brazo robótico a analizar es del tipo antropomórfico, ya que está conformado por tres juntas rotacionales y una prismática (gripper); la primera junta rotacional que gira alrededor de un eje vertical representa al tronco humano, la segunda junta rotacional que eleva al primer eslabón representa al hombro, la tercera junta rotacional que eleva al segundo eslabón representa al codo y finalmente, la junta prismática que permite abrir y cerrar una pinza mecánica representa a la mano. Ver Figura 3.1.

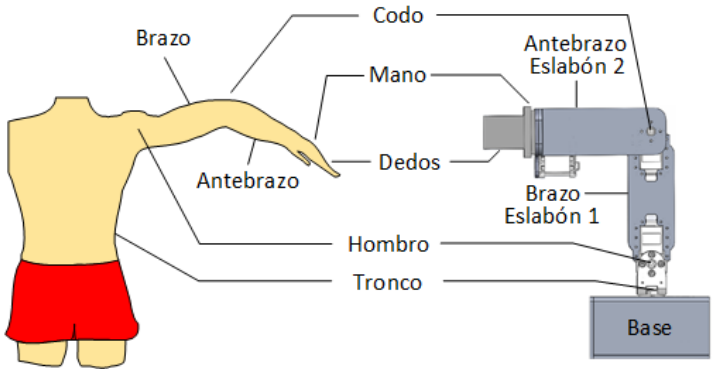


Figura 3.1: Brazo robótico tipo antropomórfico

Debido a que la junta prismática solo consiste en abrir y cerrar la pinza mecánica, entonces, para la obtención del modelo cinemático de este brazo robótico solo se van a tomar en cuenta las tres juntas rotacionales, ya que son las encargadas de llevar al extremo final del robot a la posición requerida. Por dicha razón, el análisis matemático estará basado en un brazo robótico de tres grados de libertad (tres articulaciones).

3.1.1. Desarrollo de la cinemática directa

Para la solución de la cinemática directa se necesita dibujar un diagrama que represente a la cinemática del brazo robótico de tres grados de libertad (3-GDL), y en base al diagrama se desarrolla el método de Denavit-Hartenberg. A la base del robot se le asigna un sistema de referencia fijo (O_0) y a cada extremo de los eslabones un sistema de referencia coordenado ($O_{1,2,3}$), así como se muestra en la Figura 3.2,

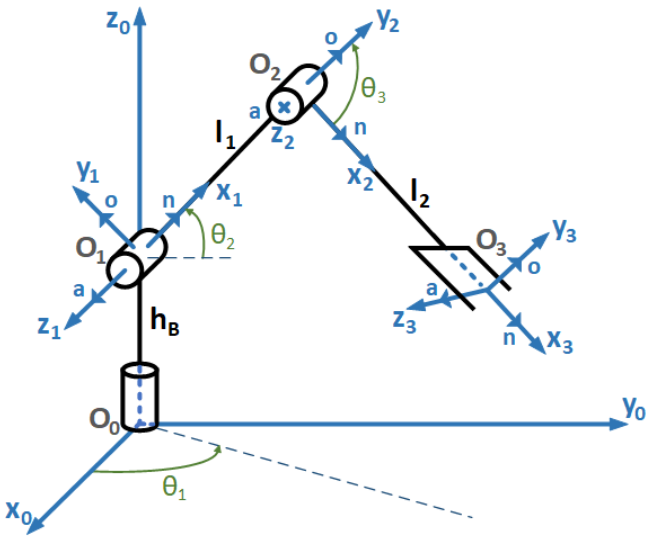


Figura 3.2: Representación cinemática del brazo robótico

donde:

h_B: Altura de la base

l_1: Longitud del primer eslabón

l_2: Longitud del segundo eslabón

θ_1: Movimiento angular de la primer articulación

θ_2: Movimiento angular de la segunda articulación

θ_3: Movimiento angular de la tercera articulación

Tomando como referencia al diagrama de la Figura 3.2, los parárametros cinemáticos de Denavit-Hartenberg que relacionan a los eslabones y a las articulaciones del brazo robótico de 3-GDL, quedan descritos por la Tabla 3.1:

	Parámetros D-H			
i	a_i	α_i	d_i	θ_i
1	0	$\pi/2$	h_B	θ_1
2	l_1	0	0	θ_2
3	l_2	0	0	θ_3

Tabla 3.1: Parámetros de Denavit-Hartenberg para robot diseñado

donde,

a_i es la distancia entre el eje Z_{i-1} hasta el eje Z_i medida a lo largo de X_{i-1}.

α_i es el ángulo de separación entre los ejes Z_{i-1} y Z_i medido alrededor de X_i.

d_i es la distancia entre el eje X_{i-1} hasta el eje X_i medida a lo largo de Z_{i-1}.

θ_i es el ángulo de separación entre los ejes X_{i-1} y X_i medido alrededor de Z_{i-1}.

A partir de los parámetros cinemáticos de Denavit-Hartenberg, es posible obtener la matriz de transformación homogénea ${}^{i-1}T_i$ que relaciona dos eslabones vecinos: ${}^0T_1$, ${}^1T_2$, ${}^2T_3$. La expresión matemática de dicha matriz se definió en la ecuación (2.2).

Para i = 1 se tiene:

$$a_1 = 0; \quad \alpha_1 = \pi/2; \quad d_1 = h_B; \quad \theta_1 = \text{algún valor angular;}$$
$$cos\theta_1 = c_1; \quad sin\theta_1 = s_1; \quad cos\alpha_1 = 0; \quad sin\alpha_1 = 1;$$

por lo tanto,

$$ {}^0T_1 = \begin{bmatrix} c_1 & 0 & s_1 & 0 \\ s_1 & 0 & -c_1 & 0 \\ 0 & 1 & 0 & h_B \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

Para i = 2 se tiene:

$$a_2 = l_1; \quad \alpha_2 = 0; \quad d_2 = 0; \quad \theta_2 = \text{algún valor angular;}$$
$$cos\theta_2 = c_2; \quad sin\theta_2 = s_2; \quad cos\alpha_2 = 1; \quad sin\alpha_2 = 0;$$

por lo tanto,

$$ {}^{1}T_{2} = \begin{bmatrix} c_2 & -s_2 & 0 & l_1c_2 \\ s_2 & c_2 & 0 & l_1s_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Para i = 3 se tiene:

$$ a_3 = l_2; \quad \alpha_3 = 0; \quad d_3 = 0; \quad \theta_3 = \text{algún valor angular}; $$
$$ cos\theta_3 = c_3; \quad sin\theta_3 = s_3; \quad cos\alpha_3 = 1; \quad sin\alpha_3 = 0; $$

por lo tanto,

$$ {}^{2}T_{3} = \begin{bmatrix} c_3 & -s_3 & 0 & l_2c_3 \\ s_3 & c_3 & 0 & l_2s_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

La matriz ${}^{0}T_{3}$ que expresa a la posición y orientación del extremo final del último eslabón con respecto a la base, esta dada por la multiplicación de las matrices de transformación homogénea que relacionan dos eslabones vecinos:

$$ {}^{0}T_{3} = {}^{0}T_{2} \times {}^{2}T_{3} $$

donde,

$$ {}^{0}T_{2} = {}^{0}\,T_{1} \times {}^{1}\,T_{2} $$

$$ {}^{0}T_{2} = \begin{bmatrix} c_1c_2 & -c_1s_2 & s_1 & l_1c_1c_2 \\ s_1c_2 & -s_1s_2 & -c_1 & l_1s_1c_2 \\ s_2 & c_2 & 0 & l_1s_2 + h_B \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.2} $$

entonces, se tiene que:

$$
{}^{0}T_{3} = \begin{bmatrix} c_1c_{23} & -c_1s_{23} & s_1 & (l_2c_{23} + l_1c_2)c_1 \\ s_1c_{23} & -s_1s_{23} & -c_1 & (l_2c_{23} + l_1c_2)s_1 \\ s_{23} & c_{23} & 0 & l_2s_{23} + l_1s_2 + h_B \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.3}
$$

donde $c_{23} = cos(\theta_2 + \theta_3)$ y $s_{23} = sin(\theta_2 + \theta_3)$.

Cada matriz de transformación homogénea de las ecuaciones (3.1), (3.2) y (3.3) indicán un vector de posición (p_x, p_y, p_z) y una matriz de orientación (n, o, a) para cada sistema de referencia coordenado con respecto al sistema de referencia fijo, tal como se definió en la ecuación (2.4) del capítulo 2.

3.1.2. Desarrollo de la cinemática inversa

En este análisis se va a considerar que el extremo final del brazo robótico de 3-GDL está ubicado en una posición donde sus coordenadas p_x, p_y y p_z son conocidas. En la Figura 3.3 se muestra la descomposición trigonométrica del brazo robótico analizado para la obtención de las ecuaciones de la cinemática inversa mediante el uso del método geométrico.

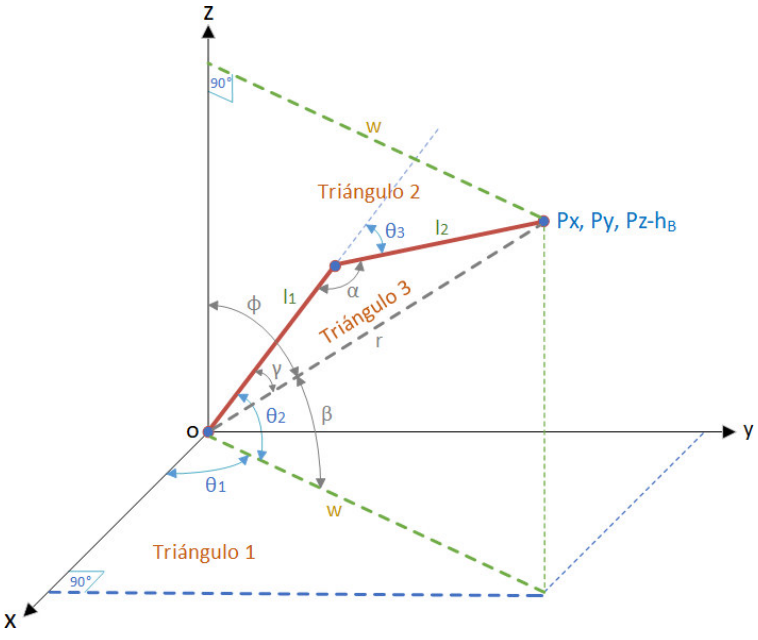


Figura 3.3: Representación descompuesta del manipulador

Como se observa en el diagrama de la Figura 3.3, la altura h_B que presenta la base del brazo robótico no fue considerara, debido a que su omisión facilita el desarrollo

este análisis, por lo tanto, $h_z = p_z - h_B$; siendo h_z la altura que presenta el extremo final del brazo robótico a lo largo del eje Z al omitir la altura de la base.

Tomando como referencia a la Figura 3.3, el desarrollo del método geométrico iniciará con el análisis del triángulo 1; en la Figura 3.4 se muestran los elementos que componen a dicho triángulo.

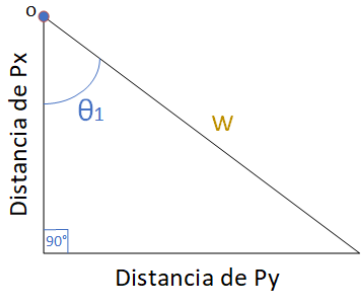


Figura 3.4: Primer triángulo para el análisis geométrico

Al aplicar el teorema de pitágoras, es posible obtener las siguientes ecuaciones para el ángulo θ_1:

$$w = \sqrt{{p_x}^2 + {p_y}^2} \tag{3.4}$$

$$\theta_1 = \text{arc tg}\left(\frac{p_y}{p_x}\right) \tag{3.5}$$

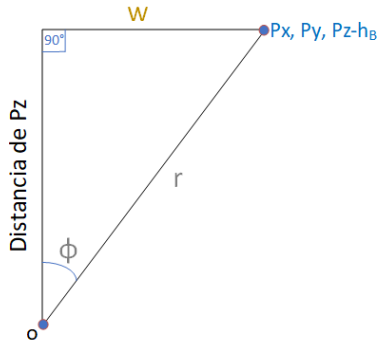


Figura 3.5: Segundo triángulo para el análisis geométrico

El triángulo 2 de la Figura 3.3 se trata también de un triángulo rectángulo, por lo que se puede aplicar nuevamente el teorema de pitágoras para obtener el ángulo ϕ y la distancia r. En la Figura 3.5 se muestran los elementos que componen a dicho triángulo.

Hay que considerar que el triángulo 2 se encuentra en un plano en el espacio tridimensional X, Y, Z que contiene al origen, y donde además la altura h_B fue omitida, por lo tanto:

$$\phi = \text{arc tg}\left(\frac{w}{h_z}\right) \tag{3.6}$$

$$r = \sqrt{{p_x}^2 + {p_y}^2 + {h_z}^2} \tag{3.7}$$

Una vez obtenido el ángulo ϕ y visualizando con detalle la Figura 3.3, se puede observar que el ángulo β puede ser obtenido mediante:

$$\beta = 90 - \phi \tag{3.8}$$

Debido a que el triángulo 3 es un triángulo irregular, entonces, es posible aplicar la ley de cosenos y con el uso de la Figura 3.6 como referencia, se puede obtener la siguiente ecuación con respecto al ángulo γ:

$${l_2}^2 = r^2 + {l_1}^2 - 2l_1 r\cos\gamma$$

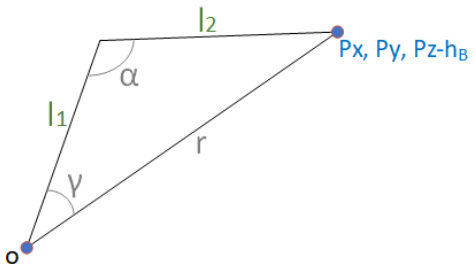


Figura 3.6: Tercer triángulo del análisis geométrico

Al despejar $\cos\gamma$ de la ecuación anterior y aplicando identidades trigonométricas, es posible obtener el ángulo γ. Así entonces:

$$\cos\gamma = \frac{r^2 + {l_1}^2 - {l_2}^2}{2l_1 r} \tag{3.9}$$

$$\text{sen}\,\gamma = \sqrt{1 - \cos^2\gamma} \tag{3.10}$$

$$\gamma = \text{arc tg}\left(\frac{\text{sen}\,\gamma}{\cos\gamma}\right) \tag{3.11}$$

De la Figura 3.3 es notable observar que el ángulo θ_2 estará dado por la suma de los ángulos β y γ, por lo tanto:

$$\theta_2 = \beta + \gamma \tag{3.12}$$

Nuevamente, se aplicará la ley de cosenos al triángulo de la Figura 3.6 pero ahora con respecto al ángulo α, obteniendo así la siguiente ecuación:

$$r^2 = {l_1}^2 + {l_2}^2 - 2l_1 l_2 \cos\alpha$$

De la ecuación anterior, es posible despejar $\cos\alpha$ y aplicar identidades trigonométricas, por lo que el ángulo α se obtiene mediante las siguientes ecuaciones:

$$\cos\alpha = \frac{{l_1}^2 + {l_2}^2 - r^2}{2l_1 l_2} \tag{3.13}$$

$$\operatorname{sen}\alpha = \sqrt{1 - \cos^2\alpha} \tag{3.14}$$

$$\alpha = \operatorname{arc\,tg}\left(\frac{\operatorname{sen}\alpha}{\cos\alpha}\right) \tag{3.15}$$

Una vez obtenido el ángulo α y visualizando la Figura 3.3, se deduce que ángulo θ_3 puede obtenerse mediante la siguiente ecuación:

$$\theta_3 = -\left(180 - \alpha\right) \tag{3.16}$$

Por lo tanto, los ángulos articulares θ_1, θ_2 y θ_3 se obtienen mediante las ecuaciones (3.5), (3.12) y (3.16) respectivamente, y en conjunto de sus ecuaciones auxiliares de ángulos complementarios.

3.1.3. Simulación de la cinemática

Una forma para comprobar si las ecuaciones anteriores de la cinemática directa e inversa son correctas, es mediante el desarrollo de un simulador en algún lenguaje de programación que permita construir virtualmente al brazo robótico para visualizar si es posible ubicar a su extremo final en una posición que sea establecida. Para llevar a cabo está simulación, se tienen que realizar los siguientes pasos:

a) Definir las medidas de las piezas que constituyen al brazo robótico de 3-GDL. Las medidas se muestran en la Tabla 3.2.

b) Establecer el punto donde se quiera ver posicionado al extremo final del brazo robótico, dicho punto debe estar definido por sus coordenadas cartesianas:

$$p_x = 0.15\ m; \quad p_y = -0.15\ m; \quad p_z = 0.05\ m.$$

c) Escribir las ecuaciones de la cinemática inversa para la obtención de las coordenadas articulares, dadas a partir de la ecuación (3.4) a la (3.16).

d) Escribir las ecuaciones del vector de posición dadas por las matrices de transformación homogenéa de las ecuaciones (3.1), (3.2) y (3.3) para la ubicación cartesiana de cada articulación.

d) Finalmente, determinar la dimensión del espacio de trabajo.

Descripción	**Símbolo**	**Valor**	**Unidades**
Altura de la base	h_B	0.079	m
Longitud del eslabón 1	l_1	0.113	m
Longitud del eslabón 2	l_2	0.14	m

Tabla 3.2: Medidas de los piezas que construyen al brazo robótico

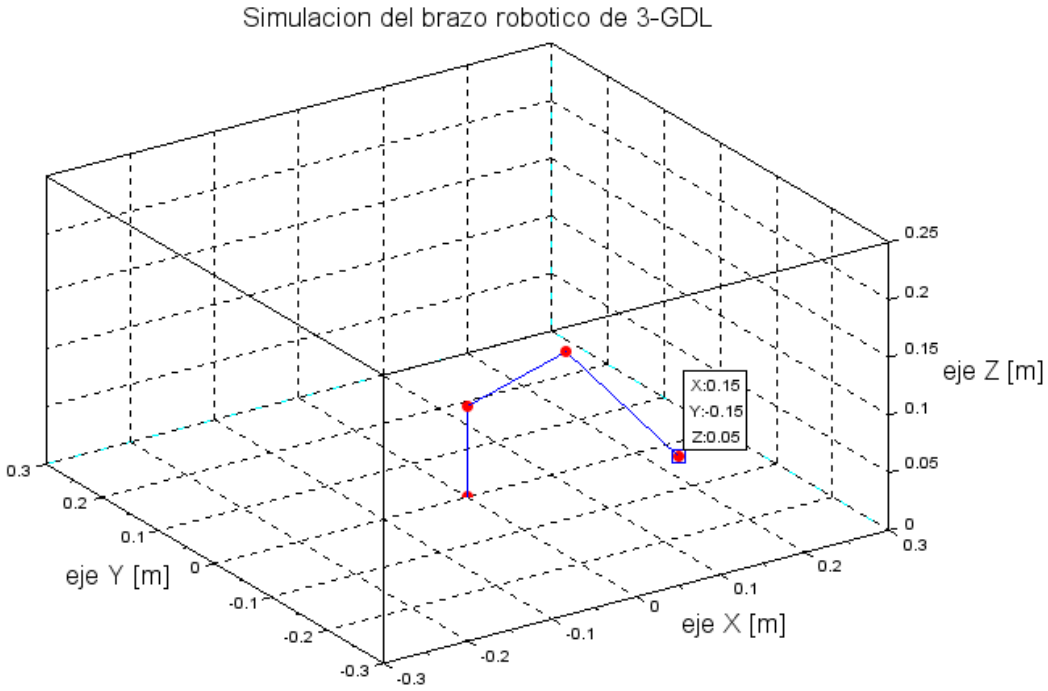


Figura 3.7: Brazo robótico ubicado en la posición establecida

En la Figura 3.7 se muestra el resultado obtenido de la simulación que fue realizada en SciLab, donde se observa la construcción simple del brazo robótico y a su extremo final ubicado en la posición que le fue establecida. Por lo tanto, las ecuaciones cinemáticas calculadas anteriormente son correctas.

El código de programación correspondiente a esta simulación se muestra en el apéndice B de este documento.

3.1.4. Simulación de trayectoria con polinomio de 5to. grado

Una vez que el extremo final del brazo robótico haya sido ubicado en la primera posición que fue establecida (P_1), se requiere ahora generar una trayectoria lineal para que el extremo final del robot pueda seguirla y así lograr ubicarse en una segunda posición (P_2). Por lo tanto, es necesario definir las coordenadas cartesianas de ambas posiciones para planificar la trayectoria con el uso del polinomio de quinto grado.

A continuación, se presentan los pasos que fueron llevados a cabo para realizar la simulación en SciLab de lo anteriormente comentado:

a) Definir las medidas de las piezas que construyen al brazo robótico de 3-GDL. Vea la Tabla 3.2.

b) Establecer las coordenadas cartesianas para ambas posiciones. Vea la Tabla 3.3.

c) Definir el tiempo t en el que se efectuará la simulación y la cantidad de puntos intermedios deseados.

d) Escribir el polinomio de quinto grado de la ecuación (2.9) y la fórmula paramétrica de una recta en tres dimensiones de la ecuación (2.5) del capítulo 2.

e) Dentro de un ciclo *for*, se crea una tabla que estará registrando las posiciones intermedias que se vayan generando durante la trayectoria lineal. Además, se escriben las ecuaciones de la cinemática inversa dadas a partir de la ecuación (3.4) a la (3.16) para la obtención de las coordenadas articulares: θ_1, θ_2 y θ_3. Finalmente, se definen las ecuaciones (3.1), (3.2) y (3.3) de la cinemática directa para graficar la ubicación cartesiana de cada articulación.

f) Determinar la dimensión del espacio de trabajo y escribir el código adecuado para visualizar el seguimiento de trayectoria lineal por el brazo robótico.

Posición	$p(x)$	$p(y)$	$p(z)$
P_1	0.15 m	-0.15 m	0.05 m
P_2	0.15 m	0.15 m	0.20 m

Tabla 3.3: Coordenadas cartesianas de las posiciones establecidas

En la Figura 3.8 se observa la trayectoria que tuvo que seguir el brazo robótico de 3-GDL para llegar a la nueva posición que le fue establecida.

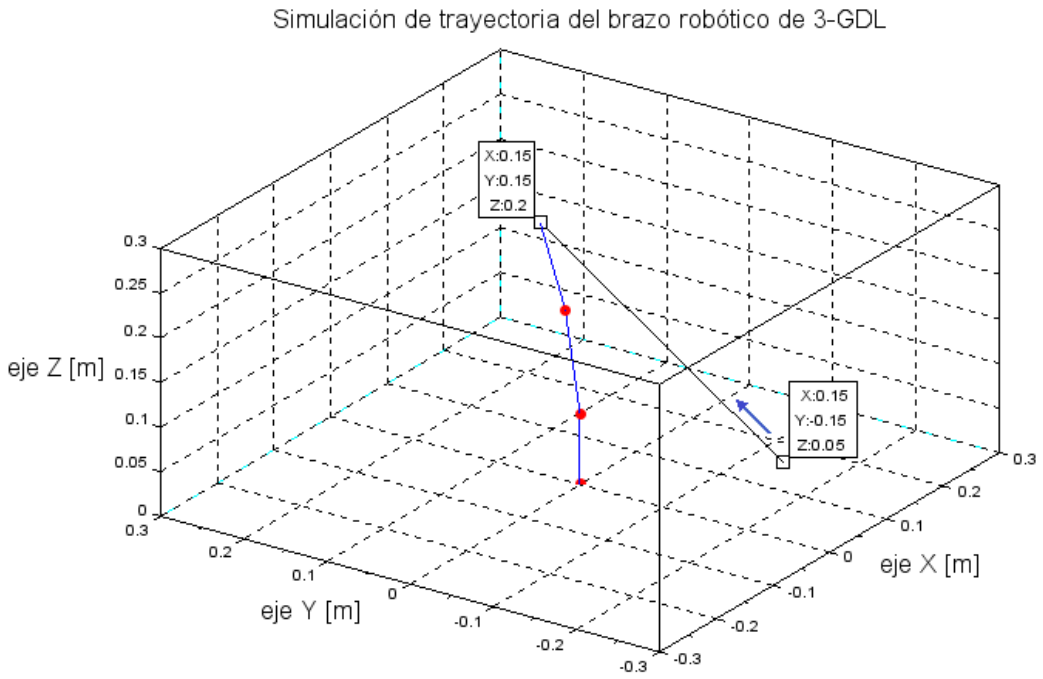


Figura 3.8: Trayectoria seguida por el manipulador robótico

En la Figura 3.9 se muestran las gráficas de los valores angulares que fueron tomando cada articulación del brazo robótico durante el seguimiento de la trayectoria lineal; observándose la presencia de movimientos suaves en las articulaciones.

El código de programación correspondiente a la simulación de trayectoria se muestra en el apéndice C de este documento.

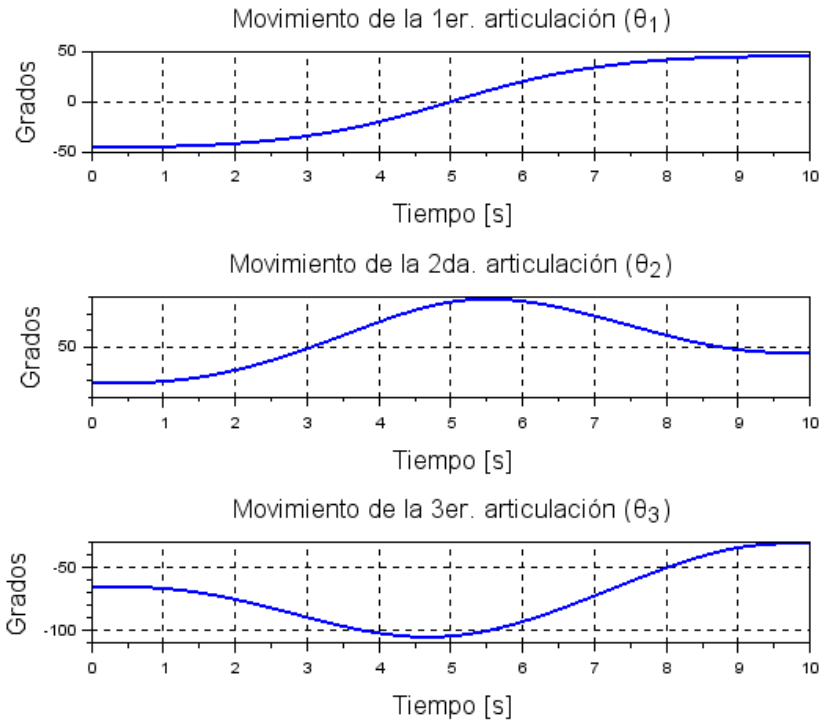


Figura 3.9: Posiciones angulares para cada articulación

3.2. Análisis dinámico del brazo robótico

El análisis dinámico se va a realizar con base en las características fisícas que presenta el brazo robótico de la Figura 3.10, el cuál volverá a ser simulado en SciLab para aplicar las técnicas de control robusto que fueron descritas en el capítulo 2.

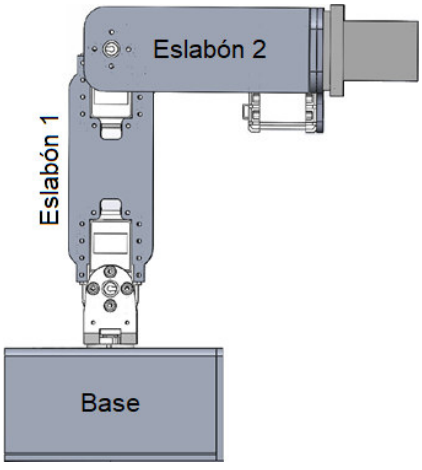


Figura 3.10: Representación del brazo robótico real

Los eslabones del brazo robótico de la Figura 3.10 están conformados por dos

láminas que tienen un grosor de 0.3 cm y que son hechas del material de la impresora $3D$; dichas láminas dan soporte a los servomotores que se ubican en los extremos de las mismas. Como las láminas son delgadas y forman eslabones huecos, entonces, los momentos de inercia quedan despreciados. Por otro lado, la masa total considerada para cada uno de los eslabones es $m_1 = 0.147\ kg$ y $m_2 = 0.116\ kg$, respectivamente.

3.2.1. Lagrangiano del brazo robótico

Debido a la ubicación de los servomotores y a la poca masa de las láminas, vamos a considerar que los momentos de inercia se desprecian por las consideraciones descritas en el párrafo anterior y que las masas se encuentran concentradas en los extremos de los eslabones; vea la Figura 3.11. Esta última consideración se hace por simplicidad y a que el modelo obtenido describe adecuadamente al robot.

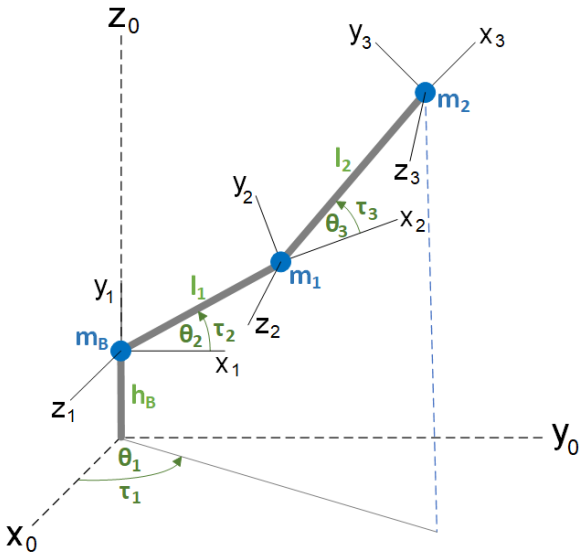


Figura 3.11: Representación del brazo robótico con masas distribuidas

Además, se considera también que el movimiento angular de cada articulación del robot es provocado por la aplicación de momentos de fuerzas, por lo tanto, las coordenadas generalizadas θ_1, θ_2 y θ_3 están asociadas respectivamente con los momentos de fuerzas τ_1, τ_2 y τ_3; vea nuevamente la Figura 3.11.

Para formular la ecuación Lagrangiana, recurriremos a las matrices de transformación homogénea ($^0T_1$, $^0T_2$ y $^0T_3$) de las ecuaciones (3.1), (3.2) y (3.3), ya que cada una contiene un vector de posición (P_{3x1}) que es compatible con la ubicación de las ma-

sas que se encuentran concentradas en los extremos de los eslabones. En la ecuación (3.17) se indican las posiciones globales de las masas distribuidas.

Una vez identificadas las posiciones globales, es posible obtener las ecuaciones de las velocidades angulares mediante la derivada con respecto al tiempo de la posición global, tal como se muestra en la ecuación (3.18).

$$\begin{aligned}
\begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} &= \begin{bmatrix} 0 \\ 0 \\ h_B \end{bmatrix} \\
\begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} &= \begin{bmatrix} l_1 c_1 c_2 \\ l_1 s_1 c_2 \\ l_1 s_2 + h_B \end{bmatrix} \\
\begin{bmatrix} x_3 \\ y_3 \\ z_3 \end{bmatrix} &= \begin{bmatrix} c_1(l_1 c_2 + l_2 c_{23}) \\ s_1(l_1 c_2 + l_2 c_{23}) \\ l_1 s_2 + l_2 s_{23} + h_B \end{bmatrix}
\end{aligned} \tag{3.17}$$

$$\begin{aligned}
\begin{bmatrix} \dot{x}_1 \\ \dot{y}_1 \\ \dot{z}_1 \end{bmatrix} &= \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \\
\begin{bmatrix} \dot{x}_2 \\ \dot{y}_2 \\ \dot{z}_2 \end{bmatrix} &= \begin{bmatrix} -l_1(\dot{\theta}_1 s_1 c_2 + \dot{\theta}_2 c_1 s_2) \\ l_1(\dot{\theta}_1 c_1 c_2 - \dot{\theta}_2 s_1 s_2) \\ l_1 \dot{\theta}_2 c_2 \end{bmatrix} \\
\begin{bmatrix} \dot{x}_3 \\ \dot{y}_3 \\ \dot{z}_3 \end{bmatrix} &= \begin{bmatrix} -l_1(\dot{\theta}_1 s_1 c_2 + \dot{\theta}_2 c_1 s_2) - l_2((\dot{\theta}_2 + \dot{\theta}_3) c_1 s_{23} + \dot{\theta}_1 s_1 c_{23}) \\ l_1(\dot{\theta}_1 c_1 c_2 - \dot{\theta}_2 s_1 s_2) + l_2(\dot{\theta}_1 c_1 c_{23} - (\dot{\theta}_2 + \dot{\theta}_3) s_1 s_{23}) \\ l_1 \dot{\theta}_2 c_2 + l_2(\dot{\theta}_2 + \dot{\theta}_3) c_{23} \end{bmatrix}
\end{aligned} \tag{3.18}$$

La energía cinética total en el brazo robótico estaría definida como:

$$K = K_1 + K_2 + K_3$$

donde K_1, K_2 y K_3 son las energías cinéticas que se manifiestan en cada una de las masas distribuidas, por lo tanto,

$$K = \frac{1}{2}m_1\left(\dot{x_1}^2 + \dot{y_1}^2 + \dot{z_1}^2\right) + \frac{1}{2}m_2\left(\dot{x_2}^2 + \dot{y_2}^2 + \dot{z_2}^2\right) + \frac{1}{2}m_3\left(\dot{x_3}^2 + \dot{y_3}^2 + \dot{z_3}^2\right)$$

y al resolver la sumatoria de las velocidades globales al cuadrado para cada masa, se tiene,

$$K = \frac{1}{2}m_2 v_2 + \frac{1}{2}m_3 v_3 \tag{3.19}$$

donde,

$v_2 = {l_2}^2{\dot{\theta_1}}^2{c_2}^2 + {l_2}^2{\dot{\theta_2}}^2$

$v_3 = {l_2}^2\left({\dot{\theta_1}}^2{c_2}^2 + {\dot{\theta_2}}^2\right) + {l_3}^2\left((\dot{\theta_2}+\dot{\theta_3})^2 + {\dot{\theta_1}}^2{c_{23}}^2\right) + 2l_2l_3\left({\dot{\theta_1}}^2c_2c_{23} + {\dot{\theta_2}}^2(\dot{\theta_2}+\dot{\theta_3})c_3\right)$

La energía potencial total en el brazo robótico estaría definida como:

$$P = P_1 + P_2 + P_3$$

donde P_1, P_2 y P_3 son las energías potenciales individuales que se manifiestan en las masas distribuidas, las cuales estan relacionadas con su altura en z_i, por lo tanto,

$$P = m_1gz_1 + m_2gz_2 + m_3gz_3$$

y con esto se tiene que,

$$P = m_1gl_1 + m_2g(l_2s_2 + l_1) + m_3g(l_2s_2 + l_3s_{23} + l_1) \tag{3.20}$$

El lagrangiano del manipulador robótico estará dado por: $L = K - P$, donde K y P representan a las ecuaciones (3.19) y (3.20) respectivamente.

3.2.2. Desarrollo de la ecuación de Euler-Lagrange

Con respecto a la coordenada θ_1, la ecuación de Euler-Lagrange quedaría expresada como:

$$\tau_1 = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}_1}\right) - \frac{\partial L}{\partial \theta_1}, \tag{3.21}$$

donde las derivadas parciales están dadas por:

$$\frac{\partial L}{\partial \dot{\theta}_1} = {l_1}^2\left(m_2 + m_3\right)\dot{\theta}_1{c_2}^2 + m_3\left({l_2}^2\dot{\theta}_1{c_{23}}^2 + 2l_1l_2\dot{\theta}_1c_2c_{23}\right) \tag{3.22}$$

$$\frac{\partial L}{\partial \theta_1} = 0 \tag{3.23}$$

y la derivada de la ecuación (3.22) con respecto al tiempo es:

$$\begin{aligned}\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}_1}\right) = {l_1}^2 (m_2 + m_3)\left(-2\dot{\theta}_1\dot{\theta}_2 c_2 s_2 + \ddot{\theta}_1 {c_2}^2\right) + \\ m_3 {l_2}^2\left(\ddot{\theta}_1 {c_{23}}^2 - 2\dot{\theta}_1\left(\dot{\theta}_2 + \dot{\theta}_3\right) c_{23} s_{23}\right) + \\ 2m_3 l_1 l_2\left(\ddot{\theta}_1 c_2 c_{23} - \dot{\theta}_1\dot{\theta}_2 s_2 c_{23} - \dot{\theta}_1\left(\dot{\theta}_2 + \dot{\theta}_3\right) c_2 s_{23}\right)\end{aligned} \tag{3.24}$$

Por lo tanto, al sustituir las ecuaciones (3.23) y (3.24) en la ecuación (3.21), se logra obtener la ecuación del momento de fuerza aplicado en la articulación 1.

Con respecto a la coordenada θ_2, la ecuación de Euler-Lagrange quedaría expresada como:

$$\tau_2 = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}_2}\right) - \frac{\partial L}{\partial \theta_2}, \tag{3.25}$$

donde las derivadas parciales están dadas por:

$$\frac{\partial L}{\partial \dot{\theta}_2} = {l_1}^2 (m_2 + m_3)\dot{\theta}_2 + m_3\left({l_2}^2\left(\dot{\theta}_2 + \dot{\theta}_3\right) + l_1 l_2\left(2\dot{\theta}_2 + \dot{\theta}_3\right) c_3\right) \tag{3.26}$$

$$\begin{aligned}\frac{\partial L}{\partial \theta_2} = & -{l_1}^2 (m_2 + m_3)\dot{\theta}_1^2 c_2 s_2 - m_3\dot{\theta}_1^2 c_{23} s_{23} {l_2}^2 - m_2 g l_1 c_2 + \\ & m_3\left(l_1 l_2\dot{\theta}_1^2\left(-s_2 c_{23} - c_2 s_{23}\right) - g\left(l_1 c_2 + l_2 c_{23}\right)\right)\end{aligned} \tag{3.27}$$

y la derivada de la ecuación (3.26) con respecto al tiempo es:

$$\begin{aligned}\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}_2}\right) = {l_1}^2 (m_2 + m_3)\ddot{\theta}_2 + l_2^2 m_3\left(\ddot{\theta}_2 + \ddot{\theta}_3\right) + \\ l_1 l_2 m_3\left(\left(2\ddot{\theta}_2 + \ddot{\theta}_3\right) c_3 - \dot{\theta}_3\left(2\dot{\theta}_2 + \dot{\theta}_3\right) s_3\right)\end{aligned} \tag{3.28}$$

Por lo tanto, al sustituir las ecuaciones (3.27) y (3.28) en la ecuación (3.25), se logra obtener la ecuación del momento de fuerza aplicado en la articulación 2.

Con respecto a la coordenada θ_3, la ecuación de Euler-Lagrange sería:

$$\tau_3 = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}_3}\right) - \frac{\partial L}{\partial \theta_3}, \tag{3.29}$$

donde las derivadas parciales están dadas por:

$$\frac{\partial L}{\partial \dot{\theta}_3} = m_3\left({l_2}^2\left(\dot{\theta}_2 + \dot{\theta}_3\right) + l_1 l_2 \dot{\theta}_2 c_3\right) \tag{3.30}$$

$$\frac{\partial L}{\partial \theta_3} = -m_3 l_2 c_{23}\left(l_2 \dot{\theta}_1^2 s_{23} + g\right) - m_3 l_1 l_2\left(\dot{\theta}_1^2 c_2 s_{23} + \left(\dot{\theta}_2^2 + \dot{\theta}_2\dot{\theta}_3\right) s_3\right) \tag{3.31}$$

y la derivada de la ecuación (3.30) con respecto al tiempo es:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}_3}\right) = m_3 {l_2}^2\left(\ddot{\theta}_2 + \ddot{\theta}_3\right) + m_3 l_1 l_2\left(\ddot{\theta}_2 c_3 - \dot{\theta}_2\dot{\theta}_3 s_3\right) \tag{3.32}$$

Por lo tanto, al sustituir las ecuaciones (3.31) y (3.32) en la ecuación (3.29), se logra obtener la ecuación del momento de fuerza aplicado en la articulación 3.

Una vez obtenidas las ecuaciones de los momentos de fuerza (τ_i) de cada articulación, pueden reorganizarse dichas ecuaciones en una forma más sistemática, quedando como:

$$\begin{aligned}
\tau_1 &= \left[C{c_2}^2 + A{c_{23}}^2 + 2Ec_2c_{23}\right]\ddot{\theta}_1 - 2\dot{\theta}_2 D - 2B\left(\dot{\theta}_2 + \dot{\theta}_3\right) - \\
&\quad 2\dot{\theta}_1 E\left(\dot{\theta}_2 s_{223} + \dot{\theta}_3 c_2 s_{23}\right) \\
\tau_2 &= \left[C + A + 2Ec_3\right]\ddot{\theta}_2 + \left[A + Ec_3\right]\ddot{\theta}_3 + \left(D + B\right)\dot{\theta}_1 + F + \\
&\quad E\left(\dot{\theta}_1^2 s_{223} - \left(2\dot{\theta}_2\dot{\theta}_3 + \dot{\theta}_3^2\right) s_3\right) + \left(m_2 + m_3\right) g l_1 c_2 \\
\tau_3 &= \left[A + Ec_3\right]\ddot{\theta}_2 + A\ddot{\theta}_3 + B\dot{\theta}_1 + E\left(\dot{\theta}_1^2 c_2 s_{23} + \dot{\theta}_2^2 s_3\right) + F
\end{aligned} \tag{3.33}$$

donde,

$A = m_3 {l_2}^2$ $\quad B = m_3 {l_2}^2 \dot{\theta}_1 c_{23} s_{23}$ $\quad C = \left(m_2 + m_3\right){l_1}^2$

$D = \left(m_2 + m_3\right){l_1}^2 \dot{\theta}_1 c_2 s_2$ $\quad E = m_3 l_1 l_2$ $\quad F = m_3 g l_2 c_{23}$

$s_{223} = sen\left(2\theta_2 + \theta_3\right)$

3.2.3. Modelo dinámico en la forma estándar

La ecuación (3.33) representa a las dinámicas del brazo robótico debidas a los momentos de fuerzas aplicadas, donde las fricciones no son consideradas. Por lo tanto, de acuerdo a la ecuación (2.12) del capítulo 2, se tendría la siguiente forma estándar:

$$M(\theta)\ddot{\theta} + C(\theta,\dot{\theta})\dot{\theta} + G(\theta) = \mathcal{T} \tag{3.34}$$

donde,

$$M(\theta)\ddot{\theta} = \begin{bmatrix} Cc_2{}^2 + Ac_{23}{}^2 + 2Ec_2c_{23} & 0 & 0 \\ 0 & C + A + 2Ec_3 & A + Ec_3 \\ 0 & A + Ec_3 & A \end{bmatrix} \begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \\ \ddot{\theta}_3 \end{bmatrix}$$

$$C(\theta,\dot{\theta})\dot{\theta} = \begin{bmatrix} -2\dot{\theta}_2 D - 2B\left(\dot{\theta}_2 + \dot{\theta}_3\right) - 2\dot{\theta}_1 E\left(\dot{\theta}_2 s_{223} + \dot{\theta}_3 c_2 s_{23}\right) \\ (D+B)\,\dot{\theta}_1 + E\left(\dot{\theta}_1^2 s_{223} - \left(2\dot{\theta}_2\dot{\theta}_3 + \dot{\theta}_3^2\right) s_3\right) \\ B\dot{\theta}_1 + E\left(\dot{\theta}_1^2 c_2 s_{23} + \dot{\theta}_2^2 s_3\right) \end{bmatrix} \tag{3.35}$$

$$G(\theta) = \begin{bmatrix} 0 \\ (m_2 + m_3)\, g l_1 c_2 + m_3 g l_2 c_{23} \\ m_3 g l_2 c_{23} \end{bmatrix}$$

$$\mathcal{T} = \begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{bmatrix}$$

y además, θ, $\dot{\theta}$, $\ddot{\theta} \in \mathbb{R}^3$, representan la posición, velocidad y aceleración angular respectivamente; $M(\theta)$ es una matriz simétrica de 3x3 de masas inerciales en el espacio; $C(\theta,\dot{\theta})\dot{\theta}$ es un vector de 3x1 que define a las fuerzas centrípetas y de Coriolis; $G(\theta)$ es un vector de 3x1 que incluye a la gravedad y $\mathcal{T}$ es un vector de 3x1 que involucra a los momentos de fuerzas aplicadas, siendo así la entrada del sistema.

3.2.4. Modelo dinámico real para el brazo robótico

Para que el modelo matemático de la ecuación (3.34) se asemeje más a la realidad de un brazo robótico físico, de acuerdo a la lectura de la subsección 2.3.2 del capítulo 2, deben de considerarse la presencia de fricciones viscosas (debidas a las articulaciones) y entradas desconocidas (alguna fuerza externa). Por lo tanto, el modelo dinámico de la ecuación (3.34) puede reformularse como:

$$M(\theta)\ddot{\theta} + C(\theta,\dot{\theta})\dot{\theta} + F_b\dot{\theta} + G(\theta) = \mathcal{T}_{in} + \mathcal{T}_d \tag{3.36}$$

donde $F_b \in \mathbb{R}^3$, es la fricción viscosa, $\mathcal{T}_{in} \in \mathbb{R}^3$, incluye a los pares motor (entrada del sistema) y $\mathcal{T}_d \in \mathbb{R}^3$, involucra incertidumbres y perturbaciones que pudiese presentar el sistema (entrada desconocida).

El brazo robótico de la Figura 3.10 usará como articulaciones a los servomotores Dynamixel AX-18A y AX-12A, donde el servomotor AX-18A es el que provoca la rotación del manipulador en el plano XY (θ_1) y los servomotores AX-12A provocan el movimiento de sus eslabones l_1 y l_2 con respecto al eje Z (θ_2 y θ_3).

De acuerdo a Hernández (2015), una manera matemática para estimar los coeficientes máximos de fricción viscosa (b_{imax}) para cada articulación, es mediante la relación del par máximo con respecto a la velocidad máxima que maneja cada servomotor:

$$b_{imax} = \frac{Max.Par\,[Nm]}{Max.Velocidad\,[rad/s]} \tag{3.37}$$

donde los valores de par y velocidad máximos que presentan los servomotores Dynamixel son mostrados en la Tabla 3.4.

	Caracteristicas	
Servomotores	Max. Par [Nm]	Max. Velocidad [rad/s]
$AX-12A$	1.5	6.178
$AX-18A$	1.8	10.158

Tabla 3.4: Par y Velocidad manejados por los servomotores

Por lo tanto, al sustituir los valores de la Tabla 3.4 en la ecuación (3.37), se tiene que la matriz de fricciones viscosas máximas quedaría representada como:

$$F_b(\dot{\theta}) = \begin{bmatrix} 0.1772 & 0 & 0 \\ 0 & 0.2428 & 0 \\ 0 & 0 & 0.2428 \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{bmatrix} \tag{3.38}$$

donde los coeficientes de la ecuación (3.38) son solo una estimación matemática de las fricciones viscosas máximas que pudierán presentarse en cada servomotor. Dichos valores serán considerados más adelante para la realización de simulaciones.

Los coeficientes de fricción son difíciles de obtener experimentalmente ya que, generalmente, presentan incertidumbre; sin embargo, como el objetivo de este trabajo es implementar un controlador suficientemente robusto, sería posible que dicha incertidumbre pueda ser compensada.

3.3. Estabilidad en lazo abierto

En esta sección se va a exponer la representación del modelo en espacio de estados del brazo robótico analizado, los puntos de operación, la linealización del modelo dinámico y la estabilidad que presenta el sistema en lazo abierto. Para este caso, no se tomará en cuenta la presencia del vector de entradas desconocidas $\mathcal{T}_d$.

3.3.1. Modelado del manipulador en el espacio de estados

El modelo dinámico del brazo robótico de 3-GDL, formulado por la ecuación (3.36), en conjunto con las ecuaciones (3.35), (3.38) y sin la presencia del vector $\mathcal{T}_d$, es un sistema de segundo orden con 3 ecuaciones diferenciales no lineales. Para facilitar el control del manipulador robótico, se debe reformular la ecuación (3.36) en un modelo representado en el espacio de estados.

Para poder representar al modelo dinámico de segundo orden del brazo robótico en un modelo representado en el espacio de estados, es necesario elegir las variables de estado correspondientes:

$$\begin{array}{lll} x_1 = \theta_1 & x_3 = \theta_2 & x_5 = \theta_3 \\ \dot{x}_1 = \dot{\theta}_1 = x_2 & \dot{x}_3 = \dot{\theta}_2 = x_4 & \dot{x}_5 = \dot{\theta}_3 = x_6 \\ \dot{x}_2 = \ddot{\theta}_1 & \dot{x}_4 = \ddot{\theta}_2 & \dot{x}_6 = \ddot{\theta}_3 \end{array}$$

Con las variables de estado definidas, y de acuerdo a la lectura de la subsección 2.4.2 del capítulo 2, el manipulador robótico estará representado por el siguiente modelo en espacio de estados:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_3 \\ \dot{x}_5 \end{bmatrix} = \begin{bmatrix} x_2 \\ x_4 \\ x_6 \end{bmatrix}$$
$$\begin{bmatrix} \dot{x}_2 \\ \dot{x}_4 \\ \dot{x}_6 \end{bmatrix} = M(x)^{-1} \left[\mathcal{T}_{in} - V(x, \dot{x}) - F_b\dot{x} - G(x)\right] \tag{3.39}$$

donde $x = [x_1,\ x_3,\ x_5]^T$, es el vector de posiciones angulares y $\dot{x} = [x_2,\ x_4,\ x_6]^T$, es el vector de velocidades angulares. En el apéndice A se encuentran los términos dinámicos de la ecuación (3.39), expresados en variables de estado.

3.3.2. Puntos de operación

El modelo dinámico de la ecuación (3.39) tiene la forma genérica $\dot{x} = f(x, u)$, donde el punto de operación es el estado estacionario del sistema dinámico, es decir, la solución del sistema de ecuaciones $f(x, u) = 0$.

Por lo tanto, para obtener los puntos de operación del sistema dinámico del brazo robótico de 3-GDL, se establecerá que el lado derecho del modelo en espacio estados de la ecuación (3.39) será igual a cero, teniendo entonces:

$$0 = \begin{bmatrix} x_2 \\ x_4 \\ x_6 \end{bmatrix}$$
$$0 = M(x)^{-1} \left[\mathcal{T}_{in} - C(x, \dot{x}) - F_b\dot{x} - G(x)\right]$$

dado que los estados que representan a las velocidades angulares son igual a cero, entonces, la ecuación anterior queda expresada como:

$$0 = M(x)^{-1} \left[\mathcal{T}_{in} - C(x, 0) - F_b(0) - G(x)\right]$$

por tanto, al multiplicar ambos lados por el tensor de inercia $M(x)$ y resolviendo los vectores $C(x,0)$ y $F_b(0)$ en el modelo dinámico en espacio de estados que se encuentra en el apéndice A, se tiene lo siguiente:

$$\begin{aligned} 0 &= \mathcal{T}_{in} - C(X,0) - F_b(0) - G(X) \\ \mathcal{T}_{in} &= C(X,0) + F_b(0) + G(X) \\ \mathcal{T}_{in} &= G(X) \end{aligned} \tag{3.40}$$

Al sustituir los vectores $\mathcal{T}_{in}$ y $G(X)$ del apéndice A en la ecuación (3.40), obtenemos:

$$\begin{bmatrix} \tau_{in_1} \\ \tau_{in_2} \\ \tau_{in_3} \end{bmatrix} = \begin{bmatrix} 0 \\ (m_2 + m_3)\, gl_2 cos(x_3) + m_3 gl_3 cos(x_3 + x_5) \\ m_3 gl_3 cos(x_3 + x_5) \end{bmatrix} \tag{3.41}$$

De la ecuación (3.41) se observa que el par motor τ_{in_1} siempre será igual a cero; por lo tanto, el equilibrio del sistema existirá siempre y cuando la primer articulación permanezca inmóvil en cualquier instante t ≥ 0.

De la ecuación (3.41) es posible resolver los puntos de operación para los estados de posición angular x_3 y x_5. Por lo tanto:

$$\begin{aligned} x_1 &= 0 \\ x_3 &= \pm\, arccos\left(\frac{\tau_{in_2} - \tau_{in_3}}{(m_2 + m_3)\, gl_2}\right) \\ x_5 &= \pm\, arccos\left(\frac{\tau_{in_3}}{m_3 gl_3}\right) - x_3 \end{aligned} \tag{3.42}$$

Es necesario mencionar que los puntos de operación para los estados x_3 y x_5 van a existir si y solo si: $|\tau_{in_2} - \tau_{in_3}| \leq (m_2 + m_3)\, gl_2$ y $|\tau_{in_3}| \leq m_3 gl_3$.

De la ecuación (3.42) se determina que el sistema robótico tiene cuatro combinaciones de puntos de operación, donde solamente alguna podría permitir la estabilidad del sistema en lazo abierto. Dichos puntos de operación se muestran en la Tabla 3.5 y serán utilizados más adelante para analizar la estabilidad del sistema robótico en lazo abierto.

Posibles puntos de operación					
x_1	x_2	x_3	x_4	x_5	x_6
0	0	$arccos\left(\frac{\tau_{in_2}-\tau_{in_3}}{(m_2+m_3)gl_2}\right)$	0	$arccos\left(\frac{\tau_{in_3}}{m_3gl_3}\right) - x_3$	0
0	0	$arccos\left(\frac{\tau_{in_2}-\tau_{in_3}}{(m_2+m_3)gl_2}\right)$	0	$-arccos\left(\frac{\tau_{in_3}}{m_3gl_3}\right) - x_3$	0
0	0	$-arccos\left(\frac{\tau_{in_2}-\tau_{in_3}}{(m_2+m_3)gl_2}\right)$	0	$arccos\left(\frac{\tau_{in_3}}{m_3gl_3}\right) - x_3$	0
0	0	$-arccos\left(\frac{\tau_{in_2}-\tau_{in_3}}{(m_2+m_3)gl_2}\right)$	0	$-arccos\left(\frac{\tau_{in_3}}{m_3gl_3}\right) - x_3$	0

Tabla 3.5: Posibles puntos de operación para la estabilidad del sistema

3.3.3. Linealización y estabilidad del sistema

Para demostrar la estabilidad del sistema robótico en lazo abierto, consideremos que se requiere llevar al brazo robótico a la posición: $Px = 0.20\ m$, $Py = 0.0\ m$ y $Pz = 0.10\ m$, por lo tanto, con el uso de la cinemática inversa se obtienen las posiciones angulares: $x_1 = 0°$, $x_3 = 49.63°$ y $x_5 = -79.68°$.

Al sustituir en la ecuación (3.41) los estados de posiciones angulares en conjunto con sus paramétros dinámicos conocidos, se determina que los pares de fuerza requeridos son: $\tau_{in_1} = 0$ Nm, $\tau_{in_2} = 0.335$ Nm y $\tau_{in_3} = 0.143$ Nm.

Debido a que el modelo no lineal de la ecuación (3.39) está conformado por tres ecuaciones no lineales complejas y seis variables de estado, entonces, la matriz Jacobiana A tendrá tamaño de 6x6. Dicha matriz se obtiene mediante la realización de un programa en SciLab con el uso de las ecuaciones dinámicas del sistema, por lo tanto, es posible evaluarla con las cuatro posibles combinaciones de puntos de operación descritas en la Tabla 3.5.

Una vez obtenida la matriz Jacobiana evaluada con sus puntos de operación (A_e), puede transformarse en el polinomio característico:

$$P(\lambda) = det(\lambda I - A_e) = 0$$

donde I en este caso, es una matriz identidad de 6x6.

Con el polinomio característico $P(\lambda)$ de la matriz Jacobiana A_e, pueden calcularse los valores propios λ_i, que representan a las soluciones de la ecuación característica.

En la la Tabla 3.6 se presentan los valores propios obtenidos por SciLab con el uso de la matriz Jacobiana A_e y evaluada en el orden que se mostraron los diferentes puntos de operación en la Tabla 3.5.

Puntos de equilibrio						Valores propios					
x_1	x_2	x_3	x_4	x_5	x_6	λ_1	λ_2	λ_3	λ_4	λ_5	λ_6
0	0	49.63°	0	-19.58°	0	-543.5	-21.0	0.004	0.024	0.0	-32.5
0	0	49.63°	0	-79.68°	0	-232.4	-30.3	-0.008	0.012	0.0	-32.5
0	0	-49.63°	0	79.68°	0	-232.4	-30.3	0.008	-0.012	0.0	-32.5
0	0	-49.63°	0	19.58°	0	-543.5	-21.0	-0.004	-0.024	0.0	-32.5

Tabla 3.6: Valores propios del polinomio característico $P(\lambda)$

En la Tabla 3.7 se muestran los valores de los parámetros que fueron considerados para el análisis de estabilidad en lazo abierto. Dichos valores son los que presenta el brazo robótico real de la Figura 3.10.

Descripción	**Símbolo**	**Valor**	**Unidades**
Altura de la base	h_B	0.079	m
Longitud del eslabón 1	l_1	0.113	m
Longitud del eslabón 2	l_2	0.14	m
Masa debida al eslabón 1	m_1	0.147	kg
Masa debida al eslabón 2	m_2	0.116	kg
Aceleración gravitacional	g	9.81	m/s^2
Fricción viscosa max. en art. 1	b_{1max}	0.1772	$Nm/rad/s$
Fricción viscosa max. en art. 2	b_{2max}	0.2428	$Nm/rad/s$
Fricción viscosa max. en art. 3	b_{3max}	0.2428	$Nm/rad/s$

Tabla 3.7: Parámetros considerados para el análisis de estabilidad

Al relacionar el cuarto renglón de la Tabla 3.5 con respecto al cuarto renglón de la Tabla 3.6, se puede observar que el sistema del brazo robótico de 3-GDL puede ser un sistema estable, siempre y cuando la variable de estado x_1 sea igual a cero y las variables de estados x_3 y x_5 tengan puntos de operación que hayan sido afectados por un signo negativo.

El código de programación realizado en SciLab para la obtención de los valores propios, se encuentra en el apéndice D de este documento.

3.3.4. Simulación de la estabilidad del sistema en lazo abierto

Para comprobar que el modelo dinámico en espacio de estados del brazo robótico de 3-GDL es correcto, y que la estabilidad del sistema en lazo abierto solo se lleva a cabo con los puntos de operación definidos en el cuarto renglón de la Tabla 3.6, es necesario diseñar un programa en SciLab que simule a los algoritmos matemáticos.

En la simulación realizada en SciLab, se definieron los valores paramétricos de la Tabla 3.7 y las coordenadas cartesianas de la posición donde se ubica inicialmente el extremo final del brazo robótico ($Px = 0.253\ m$, $Py = 0.0\ m$ y $Pz = 0.079\ m$).

A continuación se muestran los pasos que fueron realizados para llevar a cabo la simulación:

a) Definir una función que involucre a la matriz simétrica de 3x3 de masas inerciales M(X) y a los vectores de 3x1 de las fuerzas centrípetas y de Coriolis C(X,$\dot{X}$), las fuerzas de fricciones viscosas $F_b(\dot{X})$, los términos gravitatorios G(X) y los pares motor $\mathcal{T}_{in}$, que son dados por el apéndice A.

b) En la misma función, definir el modelo en espacio de estados de la ecuación (3.39).

c) Definir las coordenadas cartesianas de la posición donde se requiere llevar al manipulador robótico ($Px = 0.20\ m$, $Py = 0.0\ m$ y $Pz = 0.10\ m$), y escribir las ecuaciones de la cinemática inversa para la obtención de las posiciones angulares, dadas a partir de la ecuación (3.4) a la (3.16).

d) Escribir la ecuación (3.41) para la obtención de los pares motor, los cuales son requeridos para observar la estabilidad del sistema en lazo abierto.

e) Realizar la solución de las ecuaciones diferenciales de primer orden, dadas por el modelo en espacio de estados.

f) Finalmente, escribir las ecuaciones de la cinemática directa para conocer en que coordenadas de posición se estabilizo el extremo final del manipulador robótico, ya que el resolvedor de SciLab toma automáticamente los puntos de operación que hacen al sistema estable y, por lo tanto, llevaría al robot a una posición diferente a la deseada.

En la Figura 3.12 se muestran las gráficas que fueron obtenidas a partir de la simulación realizada, donde cada una representa a la posición angular estable para cada articulación. Los puntos de operación de posición angular que permitieron la estabilidad del sistema robótico fueron: $x_1 = 0°$, $x_3 = -49.63°$ y $x_5 = 19.58°$.

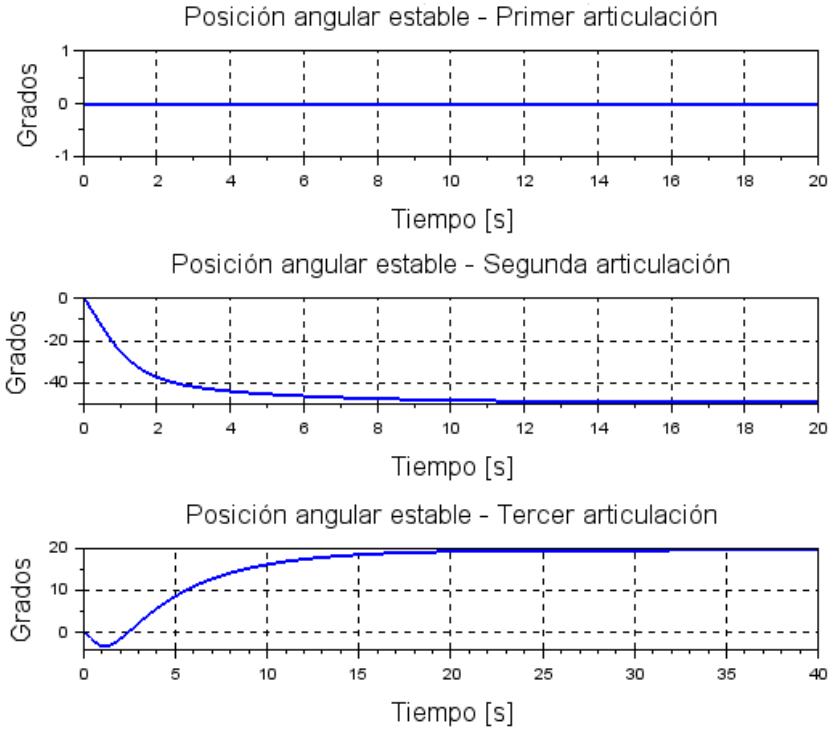


Figura 3.12: Posiciones angulares estables para cada articulación

Dichas posiciones angulares llevaron al extremo final del brazo robótico a la posición cartesiana estable: $Px = 0.20\ m$, $Py = 0.0\ m$ y $Pz = -0.075\ m$, tal como se muestra en la Figura 3.13.

Por lo tanto, se comprueba que efectivamente, el sistema en lazo abierto solo puede presentar estabilidad con los puntos de operación que hayan sido afectados por un signo negativo; como lo indicaba el cuarto renglón de la tabla 3.5

De acuerdo a lo indicado en el cuarto renglón de la Tabla 3.6 y a lo observado en la Figura 3.13, se determina que el uso del modelo dinámico en lazo abierto no es una buena idea, ya que el brazo robótico solo podría posicionarse en ciertas coordenadas cercanas a la región de estabilidad (con coordenada z negativa) y por ende, este movimiento nunca podrá presentarse en el manipulador robótico físico de la Figura 3.10 debido a la presencia de los servomotores y a la forma física que presentan los

eslabones. Lo ideal es que el manipulador robótico pueda ubicarse en posiciones que tengan coordenada z positiva.

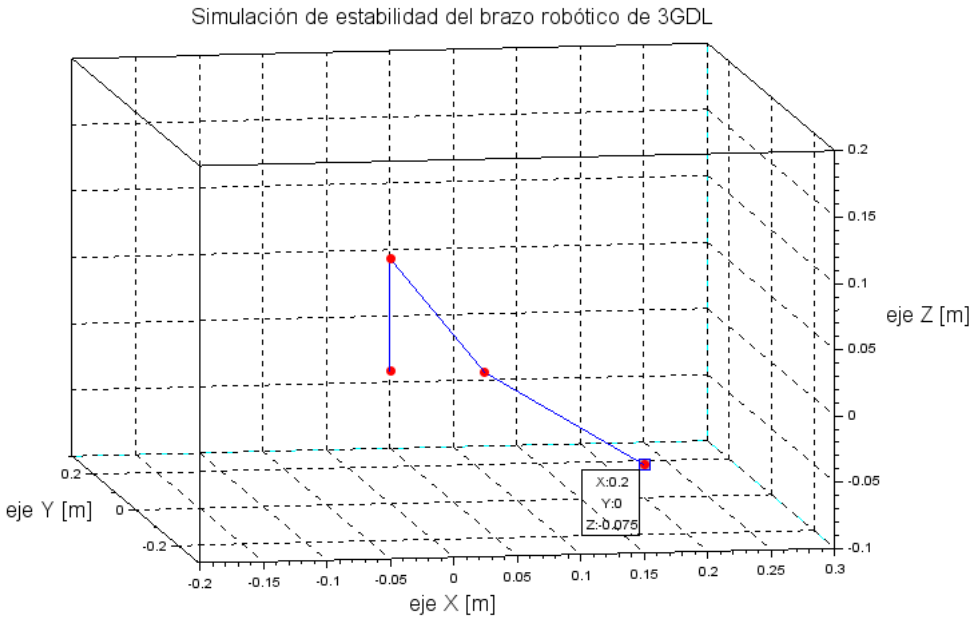


Figura 3.13: Posición cartesiana estable para el brazo robótico

El código de programación correspondiente a la simulación de estabilidad en lazo abierto se muestra en el apéndice E de este documento.

3.4. Análisis de un controlador lineal

Debido a las condiciones de estabilidad no deseables que presenta el modelo dinámico del brazo robótico de 3-GDL en lazo abierto; se tiene entonces que implementar una ley de control en lazo cerrado que permita a las dinámicas del sistema ser asintóticamente estables para cualquier punto de operación.

En esta sección se describe un primer análisis que fue llevado a cabo mediante simulaciones para conocer la posibilidad de implementar una ley de control lineal a un sistema con características no lineales. Para este análisis se consideró el modelo dinámico que presenta un péndulo invertido simple:

$$ml^2\ddot{\theta}_1 + mgl\cos(\theta_1) + b\dot{\theta}_1 = \tau_{in}$$

donde los valores de la masa (m), gravedad (g), longitud del eslabón (l), fricción

viscosa (b) y par motor (τ_{in}) son definidos como valores constantes.

3.4.1. Retroalimentación de estados

El modelo dinámico del péndulo invertido se transforma en el siguiente modelo representado en el espacio de estados:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ -\frac{mgl\cos(x_1)+bx_2}{ml^2} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{ml^2} \end{bmatrix} \tau_{in}$$
$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{3.43}$$

el cual presenta la forma $\dot{x}(t) = f(x,t) + B\tau_{in}(t)$ y $y(t) = Cx(t)$.

Tal como se mencionó en la subsección 2.4.6 del capítulo 2, para poder implementar control por retroalimentación de estados a un sistema que originalmente es no lineal, se tiene que linealizar dicho sistema para posteriormente evaluarlo alrededor de un punto de operación (x^*, u^*).

Por lo tanto, las dinámicas del sistema no lineal de la ecuación (3.43) pueden ser descritas por el siguiente modelo linealizado:

$$\begin{bmatrix} \dot{\bar{x}}_1 \\ \dot{\bar{x}}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \frac{mgl\,\mathrm{sen}(x_1{}^*)}{ml^2} & -\frac{b}{ml^2} \end{bmatrix} \begin{bmatrix} \bar{x}_1 \\ \bar{x}_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{ml^2} \end{bmatrix} \bar{\tau}_{in}$$
$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \bar{x}_1 \\ \bar{x}_2 \end{bmatrix} \tag{3.44}$$

el cual tiene la forma $\dot{\bar{x}}(t) = A\bar{x}(t) + B\bar{\tau}_{in}(t)$ y $y(t) = C\bar{x}(t)$, que será evaluado alrededor de un punto de operación (x^*, ${\tau_{in}}^*$), tomando en cuenta que:

$$\bar{x} = \begin{bmatrix} \bar{x}_1 \\ \bar{x}_2 \end{bmatrix} = \begin{bmatrix} x_1 - {x_1}^* \\ x_2 - {x_2}^* \end{bmatrix}$$
$$\bar{\tau}_{in} = \tau_{in} - {\tau_{in}}^*. \tag{3.45}$$

Para el sistema linealizado de la ecuación (3.44), se puede proponer la siguiente ley de control por retroalimentación de estados:

$$\bar{\tau}_{in}(t) = K\bar{x}(t) \tag{3.46}$$

donde las dinámicas del sistema en lazo cerrado estarán representadas por la expresión $\dot{\bar{x}}(t) = (A + BK)\bar{x}(t)$, siendo $A + BK$ la matriz dinámica A_K del sistema en lazo cerrado, por lo que:

$$\dot{\bar{x}}(t) = A_K\bar{x}(t) \tag{3.47}$$

Por lo tanto, el problema de control consiste en encontrar una ganancia K adecuada que lleve a las dinámicas del sistema en lazo cerrado a la estabilidad asintótica.

Para aplicar la estabilidad de Lyapunov, considérese la siguiente función:

$$V(\bar{x}) = \bar{x}^T {W_1}^{-1}\bar{x}$$

donde $V(\bar{x})$ será una función de Lyapunov candidata si:

1. $V(0) = 0$
2. $V(\bar{x}) > 0; \qquad 0 \leq ||\bar{x}|| \leq r$

además, W^{-1} es simétrica y definida positiva. De esta forma $\dot{V}(\bar{x})$ estará dada por:

$$\dot{V}(\bar{x}) = g(\bar{x})h(\bar{x})$$

siendo $g(\bar{x}) = \bar{x}^T$ y $h(\bar{x}) = {W_1}^{-1}\bar{x}$, entonces:

$$\dot{V}(\bar{x}) = \frac{\partial g(\bar{x})}{\partial t}h(\bar{x}) + \frac{\partial h(\bar{x})}{\partial t}g(\bar{x})$$

dando como resultado:

$$\dot{V}(\bar{x}) = \dot{\bar{x}}^T {W_1}^{-1}\bar{x} + \bar{x}^T {W_1}^{-1}\dot{\bar{x}} \tag{3.48}$$

Al implementar la ecuación (3.47) en (3.48), se obtiene lo siguiente:

$$\begin{aligned}
\dot{V}(\bar{x}) &= (A_K\bar{x})^T {W_1}^{-1}\bar{x} + \bar{x}^T {W_1}^{-1}(A_K\bar{x}) \\
\dot{V}(\bar{x}) &= \bar{x}^T {A_K}^T {W_1}^{-1}\bar{x} + \bar{x}^T {W_1}^{-1}(A_K\bar{x}) \\
\dot{V}(\bar{x}) &= \bar{x}^T({A_K}^T {W_1}^{-1} + {W_1}^{-1}A_K)\bar{x}
\end{aligned} \tag{3.49}$$

$\dot{V}(\bar{x})$ es una función de Lyapunov si:

1. $\dot{V}(0) = 0$
2. $\dot{V}(\bar{x}) < 0; \qquad 0 < ||\bar{x}|| < r$

Ambas condiciones se cumplen si: $({A_K}^T {W_1}^{-1} + {W_1}^{-1} A_K) < 0$. Al sustituir la expresión $A+BK$ en la matriz dinámica A_K de la desigualdad negativa definida y haciendo las operaciones correspondientes, se tiene:

$$\begin{aligned} &(A+BK)^T {W_1}^{-1} + {W_1}^{-1}(A+BK) < 0 \\ &(A^T + (BK)^T){W_1}^{-1} + {W_1}^{-1}A + {W_1}^{-1}BK < 0 \\ &A^T {W_1}^{-1} + K^T B^T {W_1}^{-1} + {W_1}^{-1}A + {W_1}^{-1}BK < 0 \end{aligned} \tag{3.50}$$

La ecuación (3.50) queda representada como una desigualdad matricial lineal (LMI, del inglés: Linear Matrix Inequality). Al multiplicar la LMI por W_1 a la izquierda y a la derecha, se tiene:

$$W_1 A^T + W_1 K^T B^T + A W_1 + BK W_1 < 0$$

Puesto que ${W_1}^{-1} = [{W_1}^{-1}]^T$, entonces $W_1 = {W_1}^T$. Considerando un cambio de variable, $W_2 = KW_1$, se tiene:

$$W_1 A^T + {W_2}^T B^T + A W_1 + B W_2 < 0$$

además, puesto que ${W_1}^{-1} > 0$, entonces, $W_1 > 0$.

Por lo tanto, la función $V(\bar{x}) = \bar{x}^T {W_1}^{-1} \bar{x}$ es una función de Lyapunov candidata y el sistema en lazo cerrado $A_K = A + BK$ es asintóticamente estable si y solo si:

$$\begin{aligned} &{W_1}^T = W_1 \\ &W_1 > 0 \\ &A W_1 + B W_2 + W_1 A^T + {W_2}^T B^T < 0 \end{aligned} \tag{3.51}$$

donde la ganancia del controlador por retroalimentación de estados puede ser obtenida mediante: $K = W_2 {W_1}^{-1}$.

En caso de que querer implementar control por retroalimentación de estados al

sistema no lineal de la ecuación (3.43), sin haber realizado linealización previamente, es posible proponer la siguiente ley de control a partir de la ecuación (3.46):

$$\tau_{in}(t) = \tau_{in}(t)^* + K\bar{x}(t) \tag{3.52}$$

Para verificar la respuesta del sistema en lazo cerrado, se realizó una simulación en SciLab con el uso de los modelos dinámicos de las ecuaciones (3.43) y (3.44) para la implementación de la ley de control por retroalimentación de estados.

Para poder llevar a cabo dicha simulación, se realizaron los siguientes pasos:

a) Definir los valores paramétricos considerados para las dinámicas del sistema:

$$l = 0.115\ m; \quad m = 0.147\ kg; \quad b = 0.2428\ \tfrac{Nm}{rad/s}; \quad g = 9.81\ m/s^2\ .$$

b) Definir el punto de operación requerido para los estados y la entrada del sistema:

$$x_1{}^* = \pm 60°; \qquad x_2{}^* = 0\ m/s; \qquad \tau_{in}{}^* = mgl\cos x_1{}^*.$$

c) Escribir las matrices A y B del modelo linealizado de la ecuación (3.44).

d) Resolver las desigualdades matriciales lineales (LMI's) de la ecuación (3.51) mediante el uso de la función *lmisolver* que proporciona SciLab.

e) Establecer el tiempo en el que se efectuará la simulación y la cantidad de puntos intermedios deseados.

f) Definir una función que describa las dinámicas del sistema linealizado de la ecuación (3.44) en conjunto con la ecuación (3.45) como ecuación diferencial.

g) Establecer los valores de condiciones iniciales para los estados y la entrada del sistema: $x_1(0)$, $x_2(0)$, $\tau_{in}(0)$.

j) Dentro de un ciclo *for*, definir la ley de control de la ecuación (3.46) para realizar la retroalimentación de estados en el sistema linealizado; posteriormente, resolver la ecuación diferencial de primer orden dada por su modelo dinámico.

g) Finalmente, gráficar los resultados obtenidos de dicha simulación.

Para diseñar y simular el controlador por retorno de estados con el modelo dinámico no lineal de la ecuación (3.43); se realizan nuevamente los pasos anteriores, reemplazando la ley de control de la ecuación (3.46) por la ecuación (3.52).

En la Figura 3.14 se muestran las gráficas del comportamiento de la posición angular con el uso de la ley de control por retroalimentación de estados. En el lado derecho se observa la respuesta de posición angular con el modelo linealizado, y del lado izquierdo se observa la respuesta de posición angular con el modelo no lineal original; ambos evaluados en el mismo punto de operación.

De las Figuras 3.14a) y 3.14b) se observa que tanto con el modelo linealizado como con el modelo no lineal del péndulo invertido; el controlador estabiliza a la posición angular en el mismo punto de operación asignado ($-60°$), aunque la estabilidad asintótica se logra más rápido con el uso del modelo linealizado. Además, la estabilización del sistema no lineal se logra solamente si la condición inicial de la posición $x_1(0)$ se encuentra cerca de la región de operación.

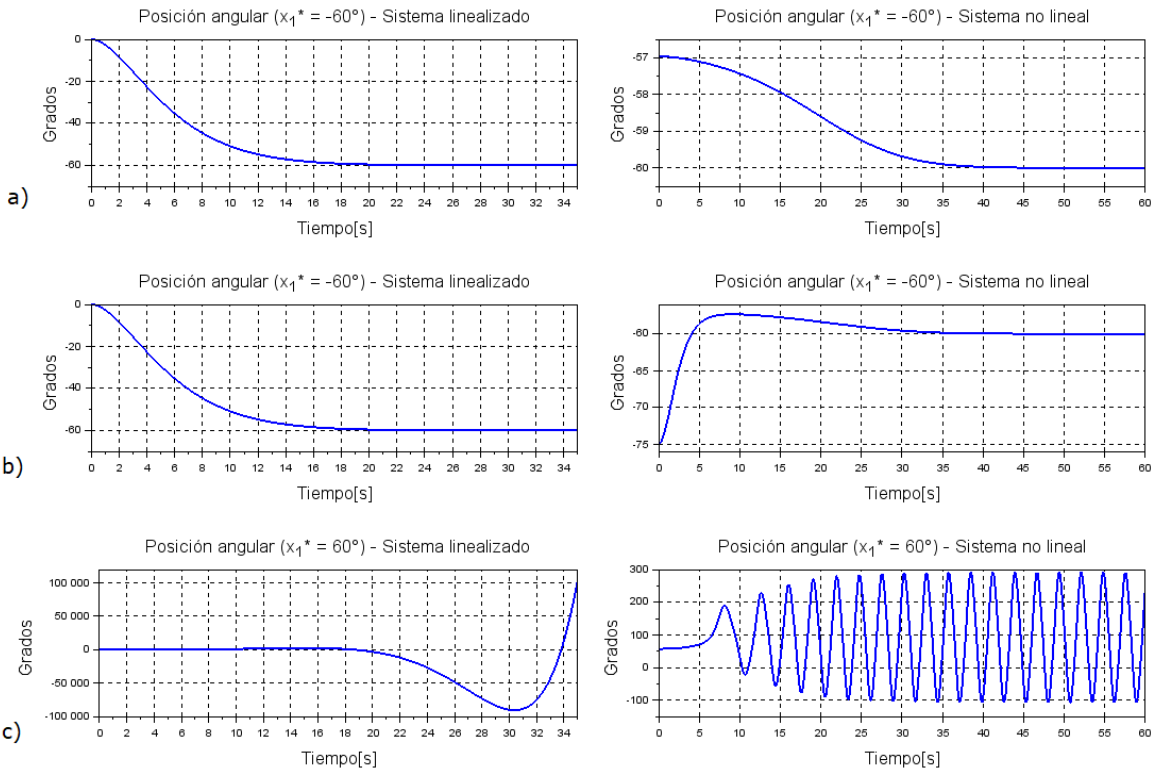


Figura 3.14: Respuesta de la posición con control por retroalimentación estados

De la Figura 3.14c) se observa que aun con la implementación del controlador en ambos modelos dinámicos, el sistema sigue siendo inestable para $x_1{}^* > 0°$.

En la Figura 3.15 se muestran los valores de los pares motor $\bar{\tau}_{in}$ y τ_{in} (para el sistema linealizado y no lineal respectivamente), los cuales fueron obtenidos por el controlador

para llevar a la salida del sistema a la posición angular que fue definida por el punto de operación, logrando así, la estabilización asintótica. Los incisos a) y b) de la Figura 3.15 corresponden respectivamente a las gráficas a) y b) de la Figura 3.14.

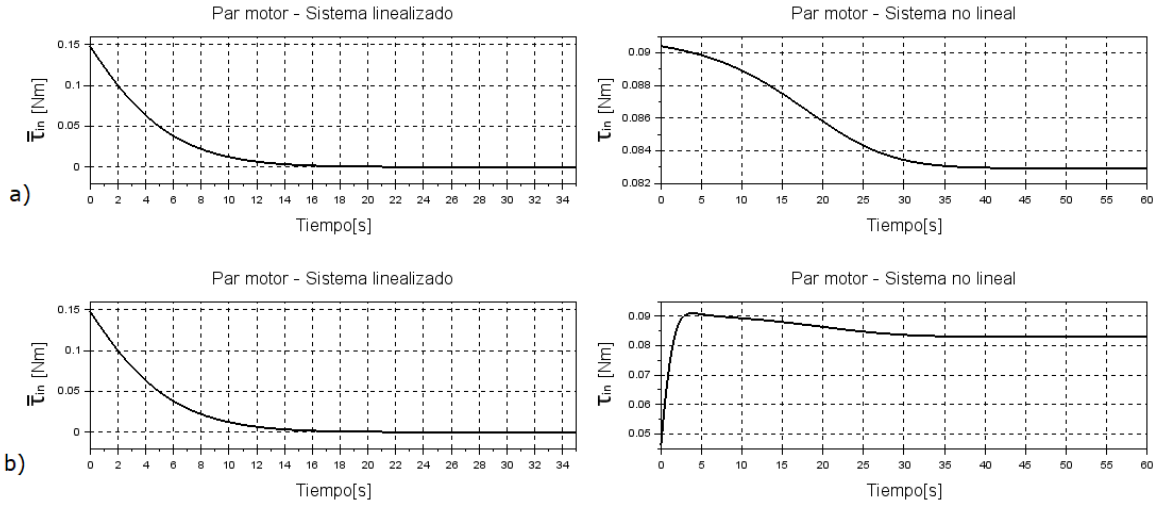


Figura 3.15: Pares motor debidos a la retroalimentación de estados

Los pares motor $\bar{\tau}_{in}$ y τ_{in} presentan valores que si pueden ser alcanzados por los servomotores Dynamixel $AX-18A$ y $AX-12A$, ya que producen una salida de par motor de 0 a 1.8 Nm y de 0 a 1.5 Nm, respectivamente.

3.4.2. Consideración de regiones de estabilidad

Para que el sistema del péndulo invertido llegue al estado estacionario más rápido y además las oscilaciones puedan ser reducidas; los valores propios de la matriz dinámica A_k deben alejarse lo más que se pueda del origen del semiplano imaginario izquierdo y así mismo, su parte imaginaria debe ser eliminada. Ver Figura 3.16.

Para cumplir lo anteriormente mencionado, se debe de aplicar el concepto de regiones de estabilidad. Una región de estabilidad es un subconjunto $\mathbb{C}_{estab} \subseteq \mathbb{C}$, el cual tiene las siguientes propiedades:

1) Si $\lambda \in \mathbb{C}_{estab}$, entonces, $\bar{\lambda} \in \mathbb{C}_{estab}$.

2) Si $\mathbb{C}_{estab}$ es convexa.

Los ejemplos típicos de regiones de estabilidad incluyen:

a) $\mathbb{C}_{estab1} = \mathbb{C}^{-}$: Semiplano imaginario izquierdo.

b) $\mathbb{C}_{estab2} = \{S \in \mathbb{C} \mid \mathbb{R}(s) < -\varphi\}$: Barra vertical.

c) $\mathbb{C}_{estab3} = \{S \in \mathbb{C} \mid |s| < \chi\}$: Circulo con centro en el origen.

d) $\mathbb{C}_{estab4} = \{S \in \mathbb{C} \mid \mathbb{R}(s)\tan(\psi) < |\mathbb{I}(S)|\}$: Región de estabilidad canónica.

Se define la región de estabilidad S como la intersección de $\mathbb{C}_{estab2}$, $\mathbb{C}_{estab3}$ y $\mathbb{C}_{estab4}$, es decir:

$$S(\varphi, \chi, \psi) = \mathbb{C}_{estab2} \cap \mathbb{C}_{estab3} \cap \mathbb{C}_{estab4}$$

La región de estabilidad S queda representada por la gráfica de la Figura 3.17.

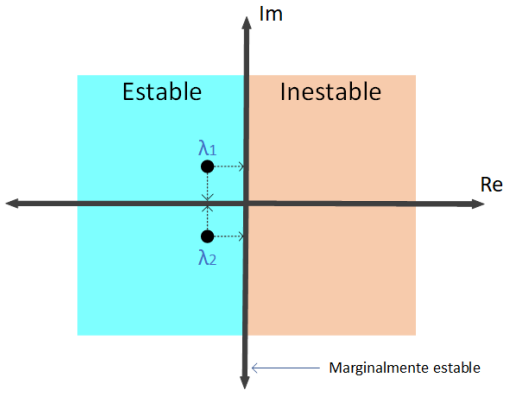


Figura 3.16: Plano complejo

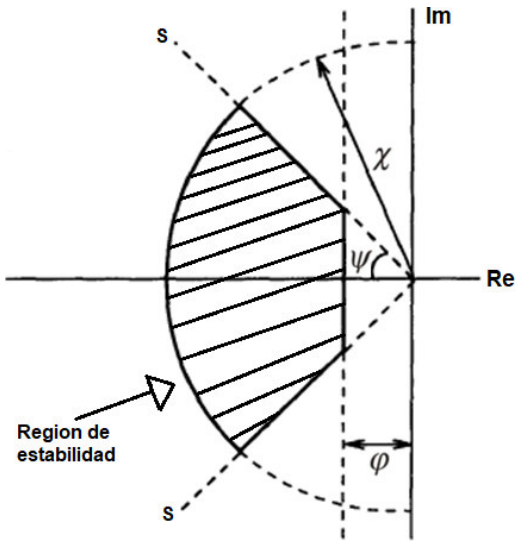


Figura 3.17: Representación gráfica de las regiones de estabilidad

Para aplicar el concepto de regiones de estabilidad al controlador por retroalimentación de estados de la subsección anterior, deben de agregarse las siguientes LMI's:

$$\begin{aligned}
&\mathbb{C}_{estab2} : A_K W_1 + W_1 {A_K}^T + 2\varphi W_1 < 0 \\
&\mathbb{C}_{estab3} : \begin{bmatrix} -\chi W_1 & A_K W_1 \\ W_1 {A_K}^T & -\chi W_1 \end{bmatrix} < 0 \\
&\mathbb{C}_{estab4} : \begin{bmatrix} \operatorname{sen}\psi[A_K W_1 + W_1 {A_K}^T] & \cos\psi[A_K W_1 - W_1 {A_K}^T] \\ \cos\psi[W_1 {A_K}^T - A_K W_1] & \operatorname{sen}\psi[A_K W_1 + W_1 {A_K}^T] \end{bmatrix} < 0
\end{aligned} \tag{3.53}$$

donde los valores de φ, χ y ψ deben cumplir con las siguientes condiciones:

$$\varphi \in [0, \infty); \quad \chi > \varphi; \quad \psi = (0, \pi/2). \tag{3.54}$$

Para verificar en SciLab la respuesta del controlador por retroalimentación de estados donde las regiones de estabilidad son consideradas, al resolvedor de LMI's de la simulación anterior se le agregan las ecuaciones (3.53).

Los valores que definen a la región de estabilidad se propnen como:

$$\varphi = 7; \quad \chi = 16; \quad \psi = \pi/8;$$

donde dichos valores se obtuvieron a partir de diferentes pruebas realizadas en la simulación hasta visualizar resultados óptimos en la salida del sistema.

En la Figura 3.18 se muestran las respuestas de salida que proporciona el controlador por retroalimentación de estados con regiones de estabilidad consideradas. En el lado derecho, se observa la respuesta de posición angular con el modelo linealizado y del lado izquierdo, la respuesta de posición angular con el modelo no lineal original; ambos evaluados en el mismo punto de operación.

De las Figuras 3.18a) y 3.18b) se observa que tanto con el modelo linealizado como con el modelo no lineal del péndulo invertido; la posición se estabiliza en el mismo punto de operación asignado ($-60°$) y mucho más rápido que en la simulación anterior para ambos casos. Además, la estabilización del sistema con el modelo no lineal ya puede lograrse si la condición inicial de la posición se encuentra alejada de la región de operación. De la Figura 3.18c) se observa que, para ambos modelos dinámicos, el sistema ya logra estabilizarse para posiciones angulares positivas. Los valores propios obtenidos de la matriz dinámica A_K fueron: $-10.397787 \pm 0.2921561i$.

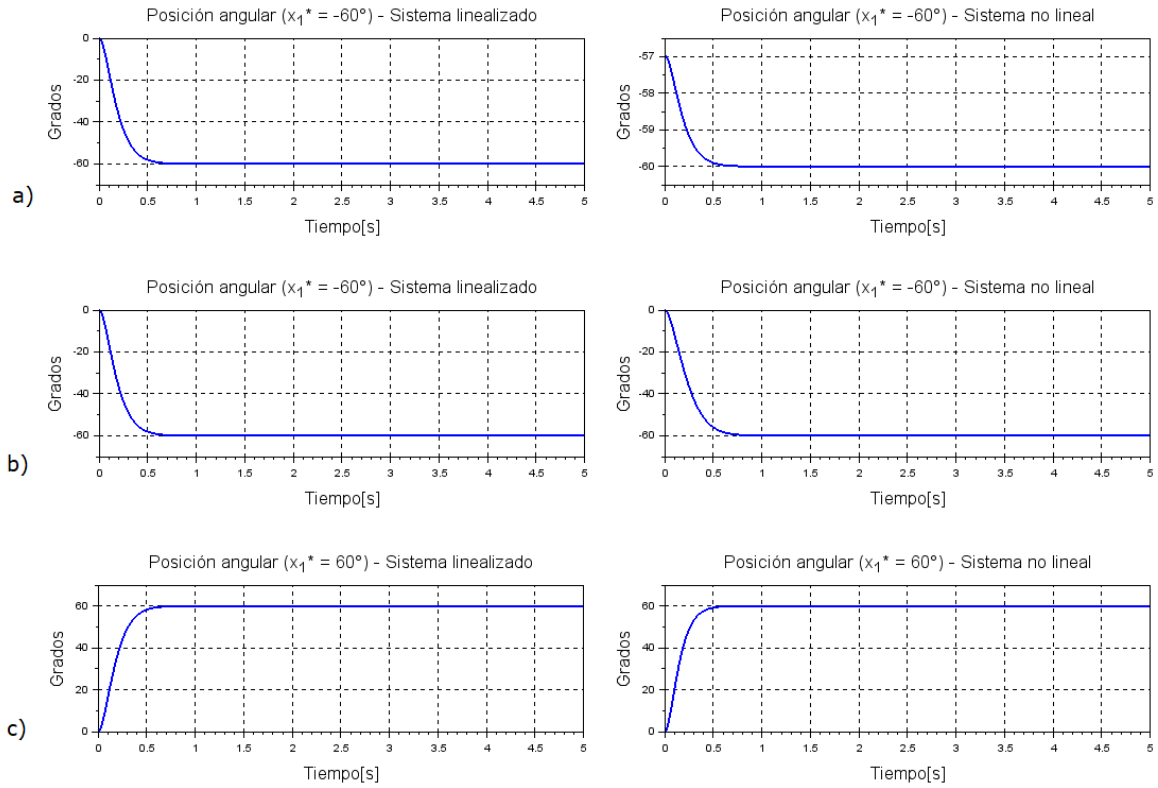


Figura 3.18: Respuesta de la posición con regiones de estabilidad

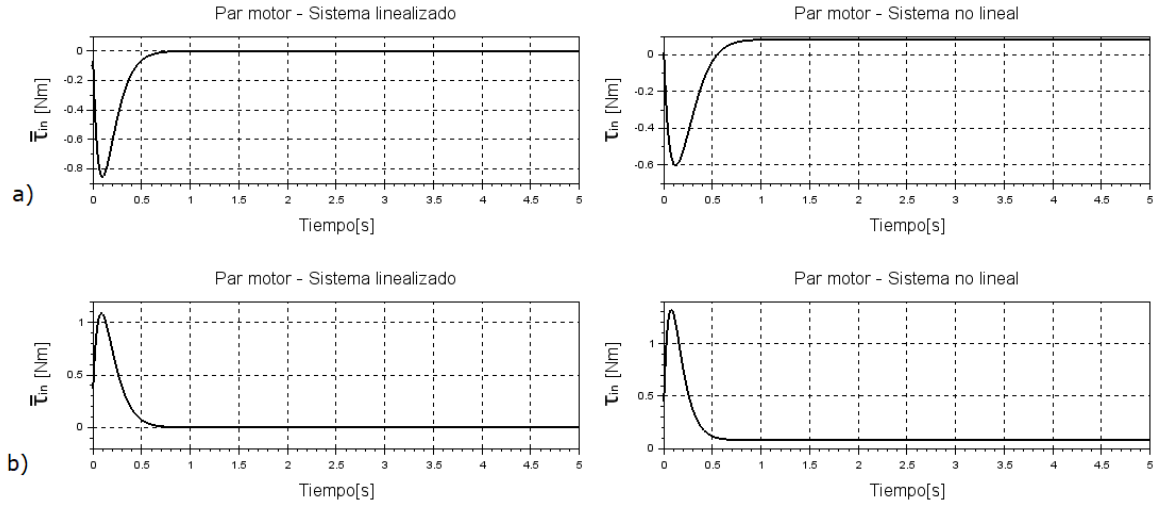


Figura 3.19: Pares motor debidos a las regiones de estabilidad

En la Figura 3.19 se muestran las gráficas de los pares motor $\bar{\tau}_{in}$ y τ_{in}, los cuales se obtuvieron por el controlador para llevar a la salida del sistema a la posición definida por el punto de operación y lograr la estabilización asintótica. Los incisos a) y b) de la Figura 3.19 corresponden respectivamente a las gráficas b) y c) de la Figura 3.18.

3.4.3. Incertidumbres paramétricas

La mayoría de los sistemas físicos se encuentran descritos por modelos dinámicos no lineales donde algunos de sus parámetros tiene valores inciertos, ya que estos parámetros presentan variaciones durante la operación del sistema; a esto se le conoce como incertidumbres paramétricas.

Para el sistema en lazo cerrado del brazo robótico de un eje, se tomará en cuenta a la fricción viscosa y a la gravedad como los únicos términos capaces de presentar incertidumbres paramétricas en el modelo dinámico del sistema, por lo que solamente se verá afectada la función no lineal de la ecuación (3.43) y la matriz A de la ecuación (3.44). Además, se considera que un objeto de 0.260 kg (por definir un valor) fue agregado en el extremo final del brazo robótico de un eje para ser trasladado.

Para que las incertidumbres paramétricas no afecten a la estabilidad del sistema en lazo cerrado, se va a acotar a la fricción viscosa y a la gravedad con los valores mínimos y máximos que serán considerados en la simulación, para posteriormente, formar un modelo politópico y en base a ello reformular una nueva ley de control.

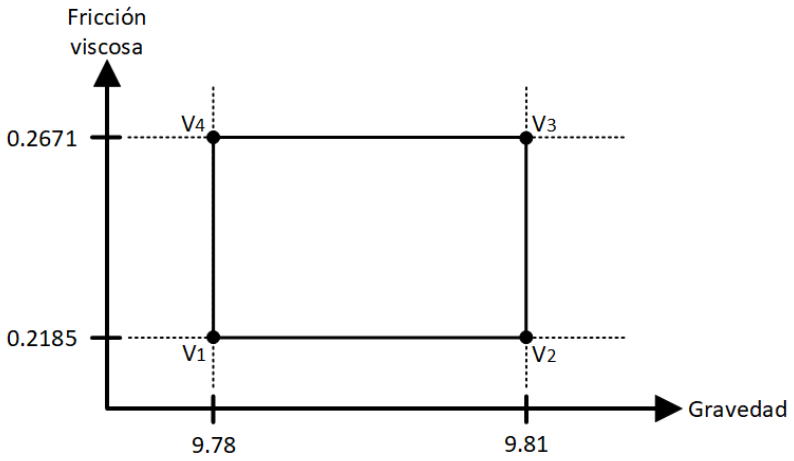


Figura 3.20: Politopo formado por los valores de gravedad y fricción

En la simulación se tomará en cuenta que la fricción viscosa varía en $\pm 10\,\%$ del valor que fue establecido en las simulaciones anteriores, y para la gravedad se considera que varía entre el valor nominal y el valor que presenta Moreno (2014), el cual fue obtenido mediante mediciones precisas que se realizaron en la Ciudad de México a 2283 metros sobre el nivel del mar; por lo tanto:

$$9.78 \leq \text{gravedad} \leq 9.81\ [m/s^2]$$
$$0.2185 \leq \text{fricción viscosa} \leq 0.2671\ [\tfrac{Nm}{rad/s}]$$

teniendo así al modelo politópico que se muestra en la Figura 3.20.

El programa de la simulación anterior fue nuevamente utilizado, pero ahora, con la consideración de incertidumbres paramétricas; por lo tanto, las ecuaciones (3.51) y (3.53) se tendrán que repetir cuatro veces debido a que la matriz A de la ecuación (3.44) tomara el valor de cada uno de los vertices que conforma al politopo convexo de la Figura 3.20. Para mejores resultados, los valores de las regiones de estabilidad tuvierón que ser modificados como:

$$\varphi = 10; \quad \chi = 28; \quad \psi = \pi/5.5.$$

En la Figura 3.21 se muestran las gráficas del comportamiento de la posición con la presencia de incertidumbres paramétricas. Del lado derecho se observa la respuesta de posición angular con el modelo linealizado y del lado izquierdo, la respuesta de posición angular con el modelo no lineal original; ambos evaluados en: $x_1{}^* = 60°$.

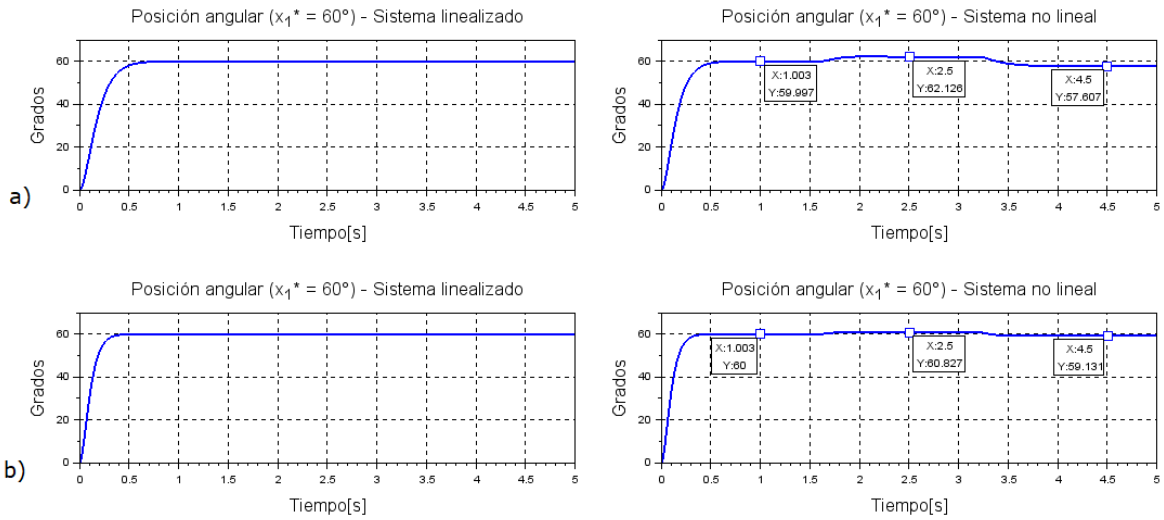


Figura 3.21: Respuesta de la posición con presencia de incertidumbres paramétricas

Las gráficas de la Figura 3.21a) representan a la respuesta de la posición angular debida a la presencia de incertidumbres paramétricas y sin tomar en cuenta el valor de cada uno de los vértices que conforma el modelo politópico de la Figura 3.20. Es notable observar que aun así, con el modelo linealizado, la salida del sistema no se ve afectada por las incertidumbres paramétricas; pero con el modelo no lineal, la posición si alcanza a desviarse un poco de su punto de operación.

Las gráficas de la Figura 3.21b) representan a la respuesta de la posición angular con la presencia de incertidumbres paramétricas, donde el controlador ya tiene implementado la acción de reducir el efecto de las variaciones paramétricas en la salida del sistema mediante el uso de los vértices del modelo politópico. Como puede observarse, con el modelo linealizado, la posición angular se sigue manteniendo estable en el punto de operación y además, con el modelo no lineal, la posición no se aleja mucho del punto de operación.

En la Figura 3.22 se muestran gráficamente los valores de los pares motor $\bar{\tau}_{in}$ y τ_{in}, los cuales fueron obtenidos por el controlador para llevar a la salida del sistema a la posición angular requerida. Los incisos a) y b) de la Figura 3.22 corresponden respectivamente a las gráficas de los casos a) y b) de la Figura 3.21. En el caso b) del sistema no lineal puede observarse que después de 0.5 s, el par de fuerza τ_{in} trata de mantenerse constante aun en presencia de las incertidumbres paramétricas del modelo.

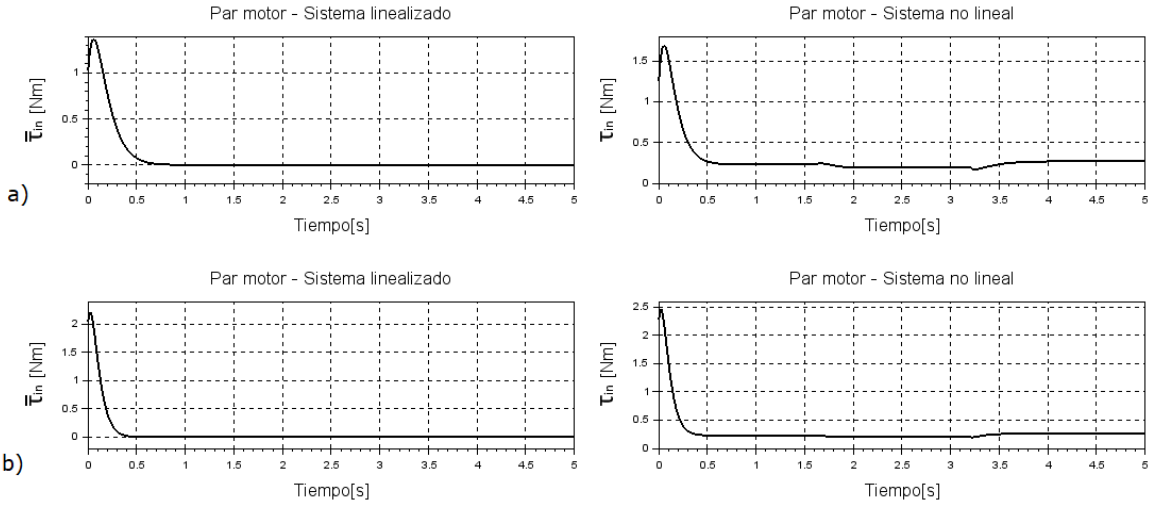


Figura 3.22: Pares motor debidos a las incertidumbres paramétricas

3.4.4. Implementación del control H_∞

El control H_∞ es una técnica de control robusto que permite minimizar el efecto de perturbaciones que pueden presentarse en un sistema dinámico, logrando así su estabilidad asintótica.

Para llevar a cabo este análisis, se va a considerar que el sistema en lazo cerrado del caso anterior, se ve afectado por la presencia de una entrada desconocida (par de fuerza τ_d), por lo tanto, el modelo dinámico no lineal de la ecuación (3.43) queda reformulado como:

$$\begin{aligned} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} &= \begin{bmatrix} x_2 \\ -\frac{mgl\cos(x_1)+bx_2}{ml^2} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{ml^2} \end{bmatrix} \tau_{in} + \begin{bmatrix} 0 \\ \frac{1}{ml^2} \end{bmatrix} \tau_d \\ y &= \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \end{aligned} \tag{3.55}$$

el cual presenta la forma $\dot{x}(t) = f(x,t) + B_u\tau_{in}(t) + B_w\tau_d(t)$ y $y(t) = Cx(t)$.

Al linealizar el modelo dinámico de la ecuación (3.51) se tiene:

$$\begin{aligned} \begin{bmatrix} \dot{\bar{x}}_1 \\ \dot{\bar{x}}_2 \end{bmatrix} &= \begin{bmatrix} 0 & 1 \\ \frac{mgl\,\mathrm{sen}(x_1{}^*)}{ml^2} & -\frac{b}{ml^2} \end{bmatrix} \begin{bmatrix} \bar{x}_1 \\ \bar{x}_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{ml^2} \end{bmatrix} \bar{\tau}_{in} + \begin{bmatrix} 0 \\ \frac{1}{ml^2} \end{bmatrix} \bar{\tau}_d \\ y &= \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \bar{x}_1 \\ \bar{x}_2 \end{bmatrix}, \end{aligned} \tag{3.56}$$

el cual presenta la forma $\dot{\bar{x}}(t) = A\bar{x}(t) + B_u\bar{\tau}_{in}(t) + B_w\bar{\tau}_d(t)$ y $y(t) = C\bar{x}(t)$,

donde la función de transferencia $T(s)$ es tal que:

$$y(s) = T(s)\bar{\tau}_d(s).$$

En este caso, $\bar{\tau}_d$ es una perturbación cuyo efecto en la salida y se desea minimizar. El cociente $||y||/||\bar{\tau}_d||$ indica la ganancia relativa que tiene la entrada desconocida $\bar{\tau}_d$ sobre la salida y. La ganancia del peor caso del sistema es:

$$||T||_\infty = \sup_{0<||\bar{\tau}_d||<\infty} \frac{||y||}{||\bar{\tau}_d||}.$$

De preferencia se requiere que el valor de la ganancia de $||T||_\infty$ (norma H_∞ de $T(s)$) sea muy pequeño para que la entrada desconocida no le afecte a la salida y así, la perturbación sea muy pequeña.

Si se considera al modelo dinámico linealizado de la ecuación (3.56), entonces, el problema de control H_∞ consiste en proponer una ley de control por retorno de estados:

$$\bar{\tau}_{in}(t) = K\bar{x}(t)$$

tal que:

a) El sistema en lazo cerrado sea asintóticamente estable.

b) La norma H_∞ de $T(s)$ (máxima ganancia de $\bar{\tau}_d$ a y) sea minimizada y además, $||T(s)||_\infty < \upsilon < 1$, donde $\upsilon \in \mathbb{R} > 0$.

b) El sistema de la ecuación (3.56) tiene que estar considerado con entrada desconocida $\bar{\tau}_d$ y salida controlada y.

Para fines de simulación, se contemplará a la perturbación τ_d como una señal senoidal con amplitud de 1 Nm, la cual representa al momento de torsión provocado por una fuerza variante externa que llega hasta los 8.7 N (τ_d = fuerza externa x distancia del eslabón); dicha señal de perturbación se muestra en la Figura 3.22.

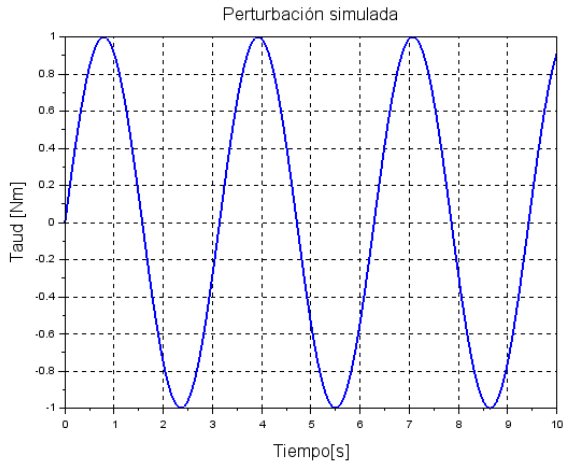


Figura 3.23: Perturbación provocada por una fuerza externa

Las dinámicas del sistema en lazo cerrado estarán representadas por la expresión $\dot{\bar{x}}(t) = (A + B_u K)\bar{x}(t) + B_w \bar{\tau}_d(t)$, siendo $A + B_u K$ la matriz dinámica A_K del sistema en lazo cerrado, por lo tanto:

$$\begin{aligned} \dot{\bar{x}}(t) &= A_K \bar{x}(t) + B_w \bar{\tau}_d(t) \\ y(t) &= C\bar{x}(t) \end{aligned} \tag{3.57}$$

Al aplicar la transformada de Laplace a las dinámicas de la ecuación (3.57) se tiene:

$$\bar{x}(s) = [sI - A_K]^{-1} B_w \bar{\tau}_d(s) \tag{3.58}$$

además, al aplicar la transformada de Laplace a la salida $y(t)$ de la ecuación (3.57) y sustituyendo la ecuación (3.58), se tiene:

$$y(s) = C[sI - A_K]^{-1} B_w \bar{\tau}_d(s) \tag{3.59}$$

Por lo tanto, a partir de la ecuación (3.59), se determina que la función de transferencia está dada por:

$$T(s) = C[sI - A_K]^{-1} B_w \tag{3.60}$$

Se propone entonces un problema de optimización semidefinida para minimizar la norma H_∞ de $T(s)$ utilizando el Lema Real Acotado (Gahinet, 1996) como:

$$\min_{\upsilon, W_1} \quad \upsilon$$

tal que:

$$\begin{gathered} \upsilon > 0 \\ {W_1}^T = W_1 > 0 \\ \begin{bmatrix} {A_K}^T W_1 + W_1 A_K & B_w & W_1 C^T \\ {B_w}^T & -\upsilon I & 0 \\ C W_1 & 0 & -\upsilon I \end{bmatrix} < 0 \end{gathered}$$

Por lo tanto, el problema de optimización se redefine como:

$$\min_{\upsilon, W_1, W_2} \quad \upsilon$$

tal que:

$$\begin{gathered} \upsilon > 0 \\ {W_1}^T = W_1 > 0 \\ \begin{bmatrix} W_1 A^T + {W_2}^T {B_u}^T + W_1 A + A W_1 + B_u W_2 & B_w & W_1 C^T \\ {B_w}^T & -\upsilon I & 0 \\ C W_1 & 0 & -\upsilon I \end{bmatrix} < 0 \end{gathered} \tag{3.61}$$

donde $K = W_2 {W_1}^{-1}$ y $||T(s)||_\infty < \upsilon$.

A partir del problema de optimización semidefinida se podrá minimizar la norma H_∞ de la función de trasferencia entre la perturbación y la salida.

Para crear la simulación del sistema en lazo cerrado con control robusto H_∞, se va a considerar el código de programación anterior que permitía eliminar las incertidumbres paramétricas del sistema analizado, agregando las LMI's de la ecuación (3.61) y tomando de nuevo en cuenta que se tienen cuatro vértices en el politopo de la Figura 3.19 para ser evaluados en la matriz Jacobiaba A de la ecuación (3.56).

Para tener una mejor robustez en la simulación del sistema dinámico, se va a considerar un politopo convexo adicional que estará conformado por dos vértices que representan a las posiciones angulares ${x_{1min}}^*$ y ${x_{1max}}^*$, las cuales se obtienen a partir del efecto que se manifiesta en la salida del sistema debido a la presencia del valor minímo y máximo de la señal de perturbación τ_d de la Figura 3.23. Por lo tanto, dichas posiciones tienen que ser también evaluadas en la matriz Jacobiana A.

Los valores que representan a las regiones de estabilidad fueron modificados como:

$$\varphi = 36; \quad \chi = 66; \quad \psi = \pi/4;$$

para que la norma H_∞ fuera la más mínima posible.

En la Figura 3.24 se muestran las gráficas del comportamiento de la posición angular con la presencia de la perturbación τ_d. Del lado izquierdo se observa la respuesta de la posición con el modelo linealizado, y del lado derecho se observa la respuesta de la posición con el modelo no lineal original.

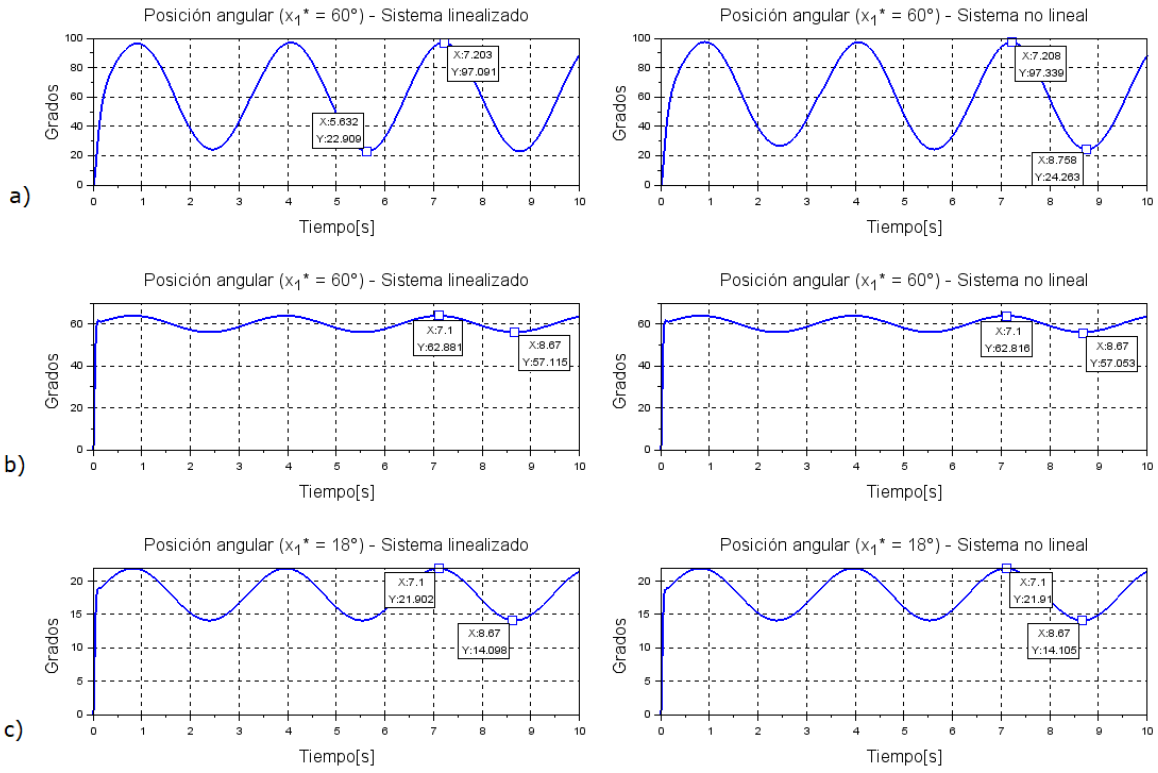


Figura 3.24: Respuesta de la posición con presencia de la perturbación τ_d

Las gráficas de la Figura 3.24a) representan a la respuesta de la posición angular debida a la presencia de incertidumbres paramétricas y la perturbación τ_d sin haber implementado el control lineal H_∞. De dichas gráficas se observa el valor mínimo y máximo que presenta la posición angular debido al efecto de la perturbación τ_d.

Las gráficas de la Figura 3.24b) representan a la respuesta de la posición angular con la presencia de incertidumbres paramétricas y la perturbación τ_d, donde el sistema en lazo cerrado ya tiene implementado la acción de eliminar el efecto de las variaciones paramétricas y reducir el efecto de la perturbación τ_d en la salida del sistema mediante el uso del control H_∞. Para ambos modelos, la posición angular se mantiene cerca del punto de operación x_1* con una diferencia de $\pm$ 2.9° aproximadamente.

Las gráficas de la Figura 3.24c) representan al caso donde se modifica el valor del punto de operación $x_1{}^*$, donde es notable observar que la posición angular obtenida ya no presenta la misma aproximación al punto de operación como en el caso anterior. Dado a esto, se tendrían que cambiar nuevamente los valores de las regiones de estabilidad.

En la Figura 3.25 se muestran las gráficas de los pares motor $\bar{\tau}_{in}$ y τ_{in}, los cuales fueron obtenidos por el controlador para llevar a la salida del sistema a la posición angular requerida. Los incisos a) y b) de la Figura 3.25 corresponden respectivamente a las gráficas de los casos a) y b) de la Figura 3.24.

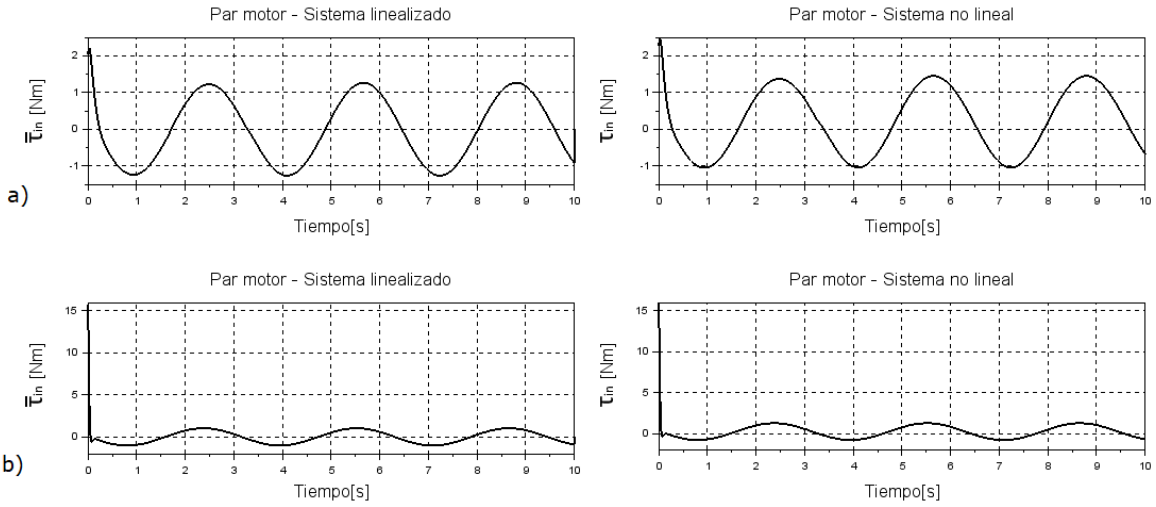


Figura 3.25: Pares motor debido a la presencia de la perturbación τ_d

Para este caso, los pares motor $\bar{\tau}_{in}$ y τ_{in} presentan valores que no pueden ser alcanzados por los servomotores Dynamixel $AX-18A$ y $AX-12A$, ya que el máximo par motor que pueden producir son 1.8 Nm y 1.5 Nm, respectivamente. Para solucionar este problema se tendría que limitar cada señal de control de la Figura 3.25 a los valores mínimos y máximos de par motor que ejercen los servomotores, aunque esto podría quitarle robustez al sistema y por ende, se tendría que volver a realizar todo el análisis anterior.

En base a los resultados que se obtuvieron de este análisis se determina que la implementación de un controlador lineal H_∞ a un sistema con características no lineales queda limitada, ya que solo es robusto para posiciones que se encuentren cerca del punto de operación y además, la implementación de este controlador puede ser muy complejo para un sistema dinámico de 3 ecuaciones diferenciales no lineales, como es el caso del brazo robótico de 3-GDL. Por lo tanto, se implementará una ley de control por modos deslizantes en lazo cerrado al sistema del brazo robótico de 3-GDL, para asegurar la estabilidad del sistema en tiempo finito aun con la presencia de las incertidumbres del modelo y de las perturbaciones aplicadas.

El código de programación realizado en SciLab que contempla a la retroalimentación de estados, regiones de estabilidad, eliminación de incertidumbres paramétricas y control H_∞, es mostrado en el apéndice F de este documento.

En la sección 4.3 del siguiente capítulo, se muestra la implementación de la ley de control por modos deslizantes al sistema robótico de 3-GDL.

3.5. Resumen del tercer capítulo

En este capítulo se considera un brazo robótico tipo antropomórfico, constituido por tres juntas rotacionales y una prismática (gripper); cada articulación está conformada por un servomotor Dynamixel de la serie AX.

En el análisis cinemático y dinámico sólo se consideran las tres juntas rotacionales del brazo robótico, ya que son las encargadas de llevar al extremo final del robot a la posición requerida.

Para la solución de la cinemática se dibuja un diagrama que represente al brazo robótico de 3 grados de libertad (3-GDL). En la base del robot se asigna un sistema de referencia fijo y a cada junta un sistema de referencia coordenado, posteriormente se desarrolla el algoritmo de Denavit-Hartenberg para obtener el modelo de la cinemática directa. Para obtener las ecuaciones de la cinemática inversa se implementa el método geométrico, el cual consiste en descomponer al diagrama del robot en triángulos geométricos para aplicar el análisis de la geometría plana.

Una vez obtenidas las ecuaciones de la cinemática directa e inversa se simula en SciLab la planificación de una trayectoria lineal, la cual se basa en la ecuación paramétrica de una recta de tres dimensiones.

En el análisis dinámico se toman en cuenta las características físicas que presenta el brazo robótico considerado. Para obtener el modelo dinámico se calcula la ecuación Lagrangiana y después se desarrolla la formulación de Euler-Lagrange, la cual permite encontrar a las ecuaciones dinámicas que relacionan a las fuerzas generalizadas que intervienen en el movimiento de las articulaciones. El modelo dinámico del brazo robótico consiste en un sistema de 2do. orden con 3 ecuaciones diferenciales no lineales.

El modelo dinámico obtenido se reformula en un modelo representado en el espacio de estados para observar la estabilidad del sistema robótico en lazo abierto. Posteriormente, se calculan los puntos de operación y se evalúan en la matriz jacobiana A del modelo linealizado para obtener el polinomio característico $P(\lambda)$ y sus valores propios, los cuales permiten determinar que el sistema será estable solamente cuando el primer servomotor permanezca inmóvil y el extremo final del robot se posicione en un punto ubicado en el eje negativo Z de su espacio de trabajo.

Como el sistema robótico en lazo abierto presenta condiciones de estabilidad no deseables, entonces, el problema de control puede solucionarse al aplicar una ley de control en lazo cerrado.

Finalmente se realiza un último análisis, el cual se lleva a cabo mediante simulaciones en SciLab para determinar la viabilidad de implementar una ley de control lineal en lazo cerrado con un sistema que presenta características no lineales. Para dicho análisis se considera el modelo dinámico de un péndulo invertido, al cual se le aplican técnicas de control lineal, como la retroalimentación de estados y el control lineal H_∞, considerando regiones de estabilidad e incertidumbres paramétricas.

En base a los resultados se determina que la implementación de un controlador lineal a un sistema con características no lineales queda limitada, ya que solo presenta estabilidad en regiones cercanas al punto de operación. Por lo que no es apto para aplicarlo en el modelo dinámico de un brazo robótico de 3-GDL.

Capítulo 4

Implementación y resultados

En este capítulo se menciona una breve descripción de los componentes mecánicos y electrónicos que se usaron para la construcción del brazo robótico de 3-GDL, así como el tipo de comunicación que se utilizó para lograr el control de los servomotores mediante una PC. Además, se presenta mediante simulaciones, la implementación del controlador twisting y super-twisting de tercer orden en el modelo dinámico del brazo robótico. Finalmente, se muestra una prueba que fue realizada con el brazo robótico fisíco para verificar su funcionamiento mediante el seguimiento de trayectorias.

4.1. Construcción del brazo robótico

La estructura física del brazo robótico de tres grados de libertad está compuesta por una base rectangular hueca y dos eslabones que se encuentran representados por dos láminas de plástico paralelas, tal como se muestra en la Figura 4.1; la base rectangular permite dar soporte al servomotor (Dynamixel AX-18A) que hace rotar al manipulador robótico, y los dos eslabones, dan soporte a los dos servomotores (Dynamixel AX-12A) que se encargan de elevar a los mismos. Además, en el extremo final del último eslabón, se encuentra un cuarto servomotor que permite abrir o cerrar una pinza mecánica que fue añadida en el manipulador robótico.

Las medidas de la base rectangular hueca son: 10 cm de largo, 10 cm de ancho y 5.2 cm de alto; las medidas de las láminas que forman al primer y segundo eslabón, respectivamente, son: 10.6 y 10.9 cm de largo y 4.8 y 4.0 cm de ancho. Para la construcción de las piezas mecánicas se utilizó el material que usan las impresoras 3D y se verificó en la práctica que si resisten al peso de los motores y a los movimientos.

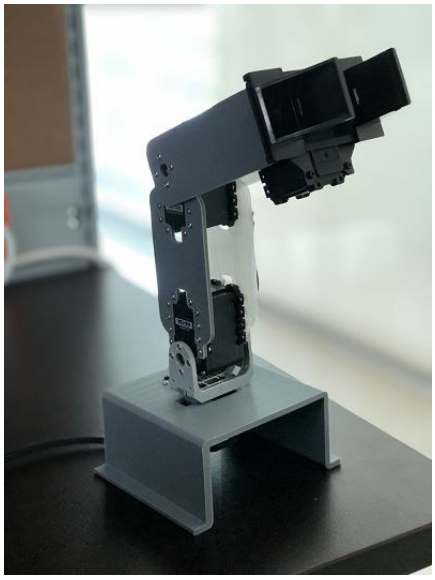


Figura 4.1: Brazo robótico construido

Con los servomotores de la serie AX Dynamixel, se tiene la ventaja de poder llevarlos a posiciones angulares específicas con tan sólo enviar una señal codificada; además, se puede obtener la lectura de la posición angular actual aun cuando los servomotores se encuentren en movimiento. La lectura de la posición angular es muy precisa, debido a que cuentan con una resolución de 0.29°.

A continuación, se mencionarán las características mecánicas y/o electrónicas de todas las piezas que se involucran en la construcción y comunicación del robot.

4.1.1. Piezas mecánicas

Los softwares de diseño asistido por computadora (CAD), como SolidWorks, brindan la facilidad de diseñar piezas mecánicas virtuales en segunda y tercera dimensión para la extracción de planos o cualquier otro tipo de información necesaria que permitan visualizar las medidas que presentan las piezas para su elaboración física.

A continuación, se muestra el diseño de las piezas mecánicas (diseñadas en SolidWorks) que permiten la construcción del brazo robótico. El diseño mecánico del robot fue desarrollado por el alumno Luis Eduardo Navarrete Ramos, miembro del grupo de trabajo del Dr. Edmundo G. Rocha Cózatl del Departamento de Ingeniería Mecatrónica de la Facultad de Ingeniería de la UNAM, como parte de un proyecto de innovación y mejoramiento de la enseñanza (Rocha, 2017).

En la Figura 4.2 se muestra el diseño virtual de la pieza mecánica que representa a la base del brazo robótico, la cual le da soporte al servomotor Dynamixel AX-18A.

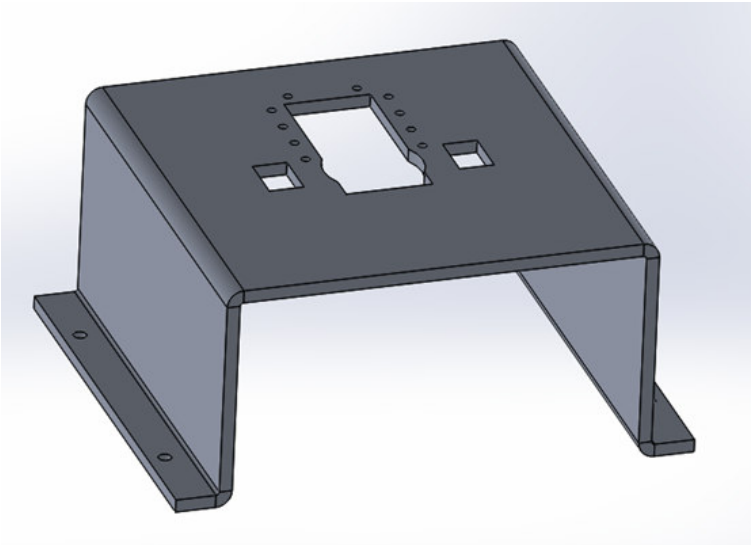


Figura 4.2: Representación virtual de la base

Una vez diseñada la pieza de la base del robot en SolidWorks, es posible obtener las medidas que la describen; tal como se muestran en las Figuras 4.3 y 4.4. Las medidas están expresadas en milímetros.

Figura 4.3: Vista frontal de la base

Cabe mencionar que el orificio rectangular central y los que se encuentran a su al-

rededor en la parte superior de la base, fueron diseñados de acuerdo a las medidas físicas que presenta el servomotor que irá acoplado.

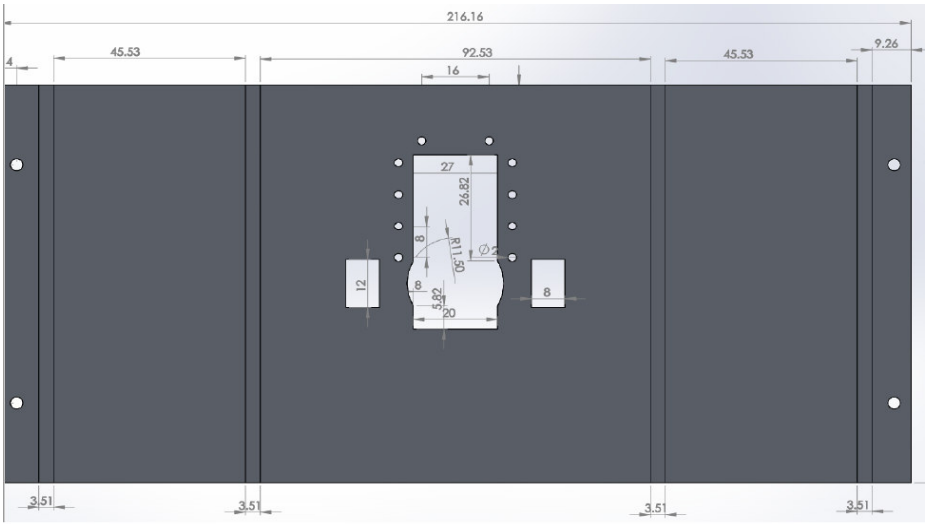


Figura 4.4: Vista superior de la base desdoblada

En la Figura 4.5 se muestra el diseño virtual y las medidas de la pieza mecánica que en conjunto con otra similar y dos servomotores Dynamixel AX-12A, forman parte del primer eslabón del brazo robótico.

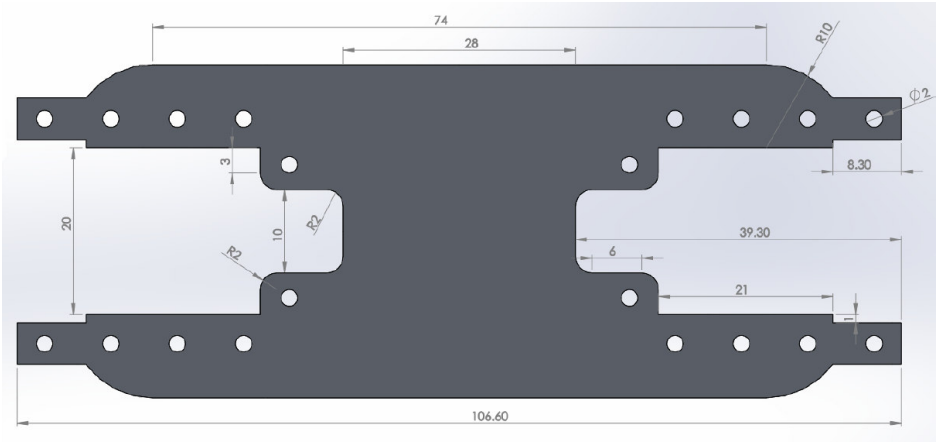


Figura 4.5: Representación virtual de pieza para primer eslabón

Para el diseño de los extremos laterales, se consideró la forma física de los servomotores que irán acoplados en ella; por lo que en la Figura 4.6, se presentan las medidas que describen a dichos extremos laterales.

En la Figura 4.7 se muestra el diseño virtual de la pieza mecánica que, en conjunto

con otra similar, dan soporte a un cuarto servomotor que permitirá abrir y cerrar la pinza mecánica ubicada en el extremo final del manipulador robótico; formando así, al segundo y último eslabón del mismo.

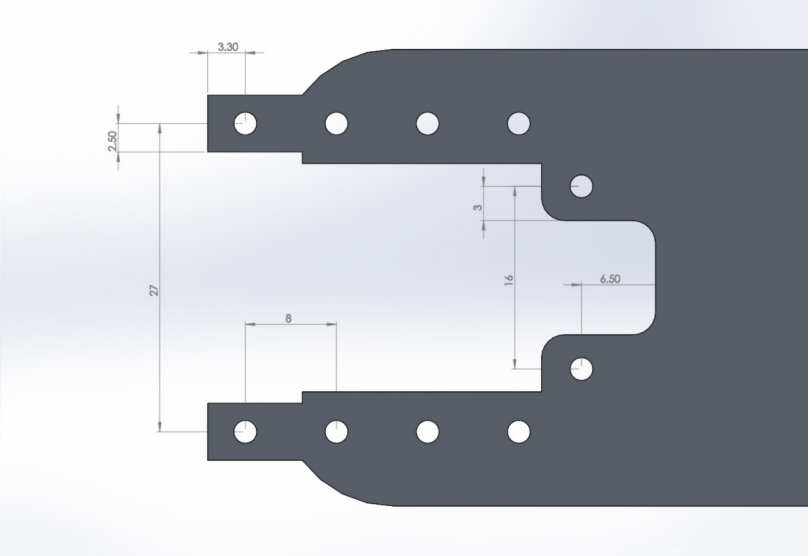


Figura 4.6: Medidas de los extremos laterales de la pieza para el primer eslabón

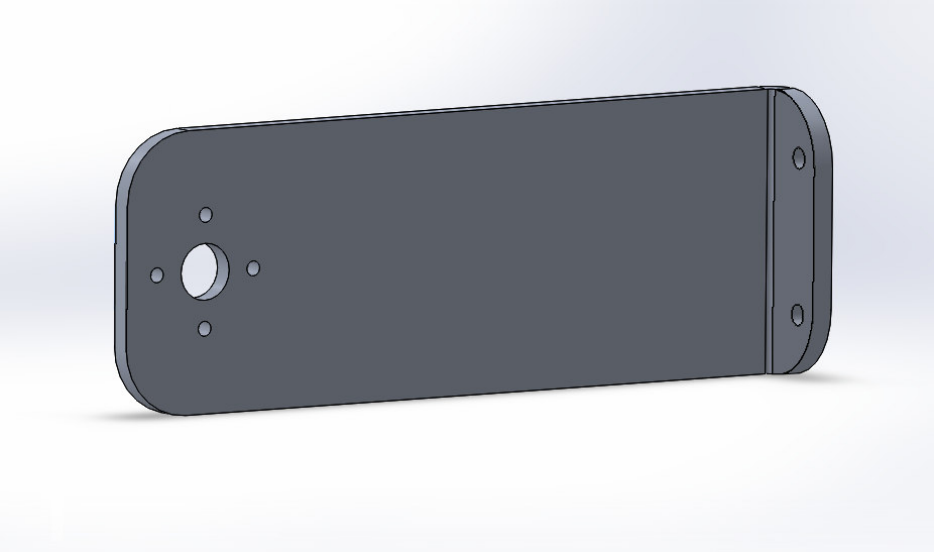


Figura 4.7: Representación virtual de pieza para segundo eslabón

En la Figura 4.8 se muestra a la pieza mecánica anterior en forma desdoblada, con el fin de poder indicar más fácilmente sus medidas que la describen. Además, en la Figura 4.9 se muestra una vista amplificada de su extremo lateral izquierdo con medidas más específicas que serán necesarias para la construcción física de la pieza.

Al igual que en el diseño de las piezas anteriores, se consideró también, la forma física de los servomotores que irán acoplados.

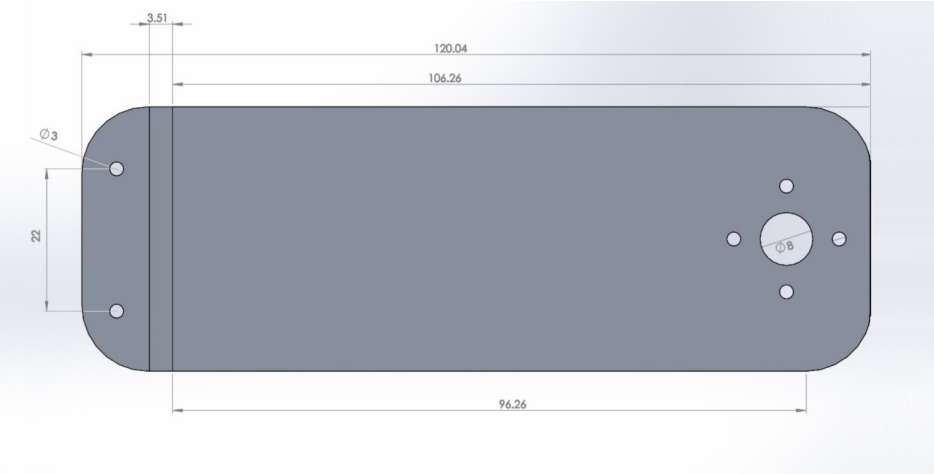


Figura 4.8: Vista frontal de pieza desdoblada para segundo eslabón

Una vez diseñadas todas las piezas mecánicas en SolidWorks y conociendo detalladamente sus medidas, pueden ser construidas con el uso de una impresora 3D. Con la colaboración del Dr. José Ángel Diosdado de la Peña, perteneciente al Departamento de Ingeniería Mecánica de la DICIS, se fabricaron físicamente las piezas que anteriormente fueron ilustradas.

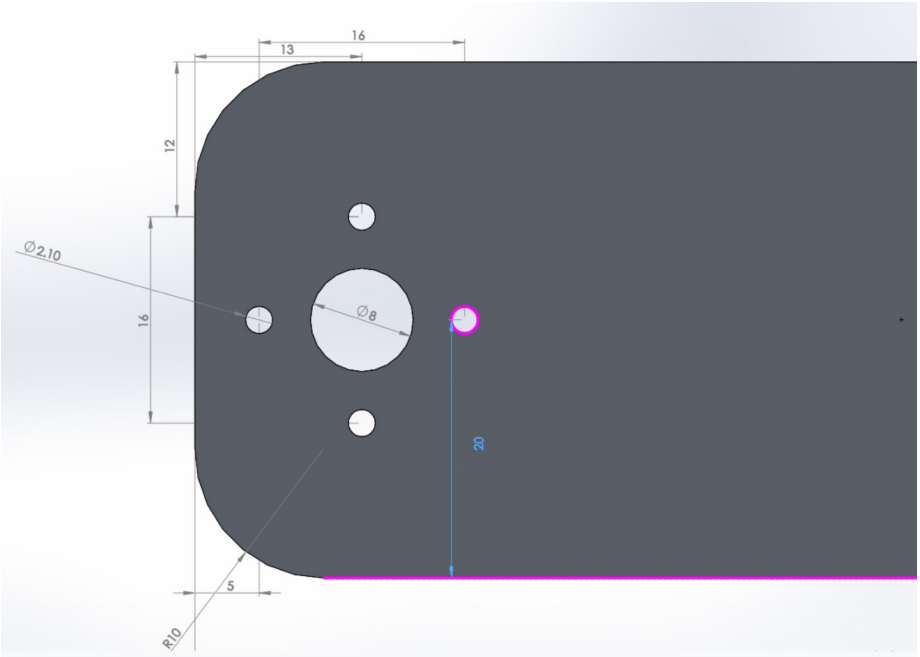


Figura 4.9: Medidas del extremo lateral izquierdo de la pieza del segundo eslabón

Cabe señalar que la pinza mecánica que va acoplada al cuarto servomotor, ubicado en el último eslabón del manipulador robótico, fue adquirida por un proveedor.

Una vez construidas y adquiridas todas las piezas mecánicas, fue posible armar la estructura física del brazo robótico de tres grados de libertad, la cual fue mostrada anteriormente en la Figura 4.1.

4.1.2. Servomotores Dynamixel

El servomotor es un dispositivo digital similar al motor de corriente continua, con la ventaja de poder ubicarse en cualquier posición que este dentro de su rango de operación y posteriormente, manteniéndose constante en la posición asignada. Además, está conformado por un motor, una caja reductora y un circuito de control.

Como se ha ido mencionando a lo largo de esta sección, la estructura física del brazo robótico de la Figura 4.1 está conformada por los servomotores Dynamixel AX-12A y AX-18A que se muestran en la Figura 4.10.

Figura 4.10: Servomotores Dynamixel AX-12A y AX-18A

Debido a la característica física que presentan los servomotores Dynamixel, se tiene la ventaja de poder facilitar su acoplamiento con alguna pieza mecánica u otro servomotor similar. Para el acoplamiento de dos servomotores, el fabricante (ROBOTIS) incluye dos piezas que son compatibles tanto con la rueda del servomotor como con cualquiera de los extremos laterales del mismo; ver la Figura 4.11.

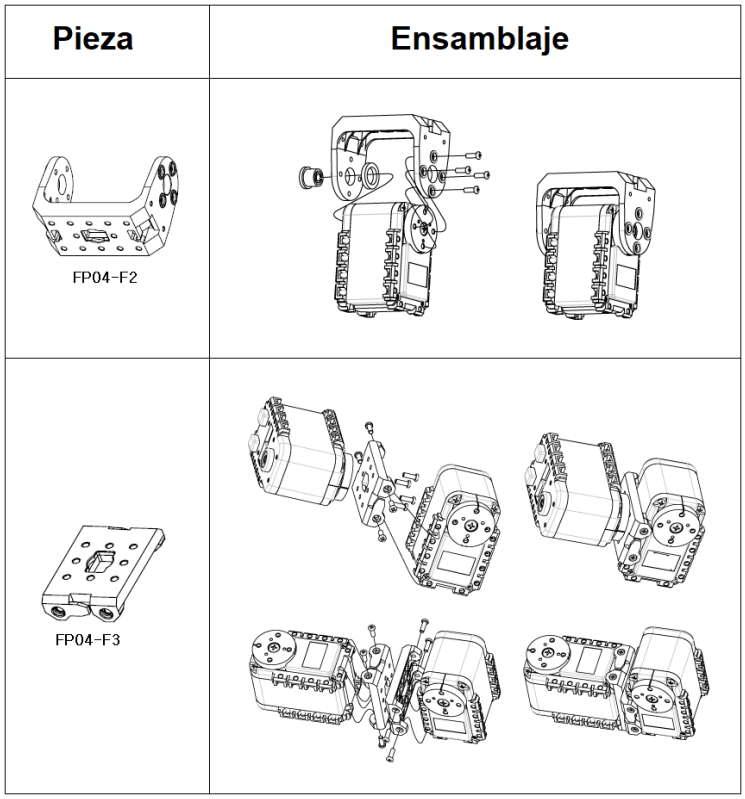


Figura 4.11: Ensamblaje de servomotores

A pesar del tamaño compacto que presentan los servomotores Dynamixel, son dispositivos que pueden producir un alto momento de fuerza (par motor) y además, debido a la alta calidad que presentan sus materiales de fabricación, cuentan con la suficiente resistencia para soportar grandes fuerzas externas.

Cabe señalar que los servomotores Dynamixel cuentan con la función de poder configurar ciertos parámetros (posición, velocidad, par motor, etc.) con tan sólo usar un paquete de comandos al momento de programarlos (vea el apéndice H). Otra característica con la que cuentan estos servomotores, es la presencia de sensores internos que permiten al usuario leer la medición de su posición, velocidad, carga, tensión y temperatura actual.

Además, los servomotores Dynamixel pueden alertar cuando sus parámetros (carga, tensión, temperatura, etc.) se desvían de los rangos que fueron previamente asignados por el usuario y automáticamente, se enciende un LED (diodo emisor de luz) para indicar que el servomotor fue desactivado debido a la presencia de un error. Para obtener más información consulte las hojas de especificación en ROBOTIS (2020).

A continuación, en la Tabla 4.1, se mencionan las especificaciones de algunas propiedades que presentan los servomotores Dynamixel AX-12A y AX-18A.

Propiedades:	**Especificaciones:**
Masa	54.6 g (AX-12A) y 55.9 g (AX-18A)
Dimensión	32 mm x 50 mm x 40 mm
Movimiento	En modo articulación: 0° a 300° En modo rueda: Giro sin fin
Par máximo	1.5 Nm (AX-12A) y 1.8 Nm (AX-18A)
Velocidad angular sin carga	59 RPM (AX-12A) y 97 RPM (AX-18A)
Voltaje de entrada	7 V a 12 V (el fabricante recomienda 11.1 V)
Temperatura de operación	-5 °C a 70 °C
Tipo de protocolo	Comunicación serial asíncrona semidúplex (8 bits, 1 parada, sin paridad)
Conexión física	Bus multipunto de nivel TTL
Velocidad de transmisión	7843 bps a 1 Mbps
Asignación de número ID	0 a 253 (Para 254 servomotores conectados en serie)

Tabla 4.1: Características generales de los servomotores Dynamixel

Cada servomotor cuenta con dos conectores molex de tres pines que se encuentran conectados entre sí, donde el primer pin representa la tierra (GND), el segundo pin representa la alimentación del voltaje y el tercer pin representa la entrada o salida de datos, tal como se muestra en la Figura 4.12.

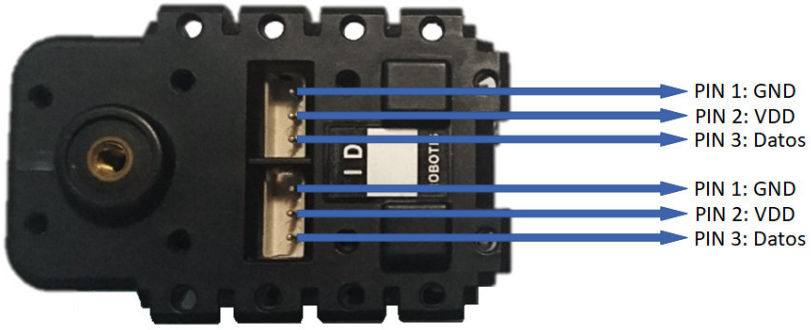


Figura 4.12: Conectores molex de tres pines

El servomotor Dynamixel puede ser operado con el uso de un solo conector, pero en caso de requerir conectar y comunicar más de un solo servomotor, el segundo conector es indispensable para la conexión multipunto, tal como se observa en la Figura 4.13. El cableado para la conexión multipunto es sencilla, debido a que el

fabricante proporciona unos cables que son compatibles con los conectores y admiten velocidades de comunicación de hasta 1 Mbps.

Los servomotores Dynamixel pueden ser operados mediante el uso de una computadora (PC) a través de un convertidor de comunicación USB, que permita cambiar las señales de nivel lógico del UART (Transmisor-Receptor Asíncrono Universal) a señales del tipo semidúplex de nivel TTL (Lógica Transistor a Transistor). Para el caso de este trabajo, se utilizó el convertidor de comunicación U2D2 que se observa en la Figura 4.13, el cuál es recomendado por el mismo fabricante, pero puede reemplazarse por algún microcontrolador que permita dicha conversión de señales.

Figura 4.13: Conexión multipunto con servomotores

Para el funcionamiento de cada servomotor, el fabricante proporciona una Tabla de Control que consiste en una estructura de datos que le permiten al usuario leer el estado o las características del dispositivo, así como poder asignarles algún valor para lograr el control del mismo. En ROBOTIS (2020) se encuentra una descripción más completa de dicha Tabla de Control, donde se indica que datos tienen la propiedad de solo lectura ("R") o la propiedad de lectura y escritura ("RW").

Los datos de la Tabla de Control se dividen en dos memorias; en una memoria RAM y en una memoria EEPROM, las cuales se encuentran interconectadas dentro del servomotor. A los datos que pertenecen al área RAM, se les pueden restablecer sus valores iniciales predeterminados siempre y cuando se encuentre presente la alimentación (volátil); por otro lado, los valores de los datos que pertenecen al área EEPROM, se mantienen guardados incluso cuando el servomotor está apagado (no volátil).

4.1.3. Dispositivo U2D2

El dispositivo U2D2 es un convertidor de comunicación USB que permite el control y operación de los servomotores Dynamixel con el uso de una PC. Dicho dispositivo de comunicación cuenta con un conector de tres pines para la conexión de servomotores que trabajan con tecnología TTL, y dos conectores de cuatro pines para la conexión de servomotores o dispositivos ROBOTIS que admitan comunicación UART o RS-485.

Cabe mencionar que el Dispositivo U2D2 no suministra el voltaje de entrada a los servomotores, por lo tanto, se necesita una fuente de alimentación externa para suministrar el voltaje de entrada a los mismos. En la Figura 4.14 se muestra la adecuada conexión del dispositivo U2D2 con la PC y los servomotores.

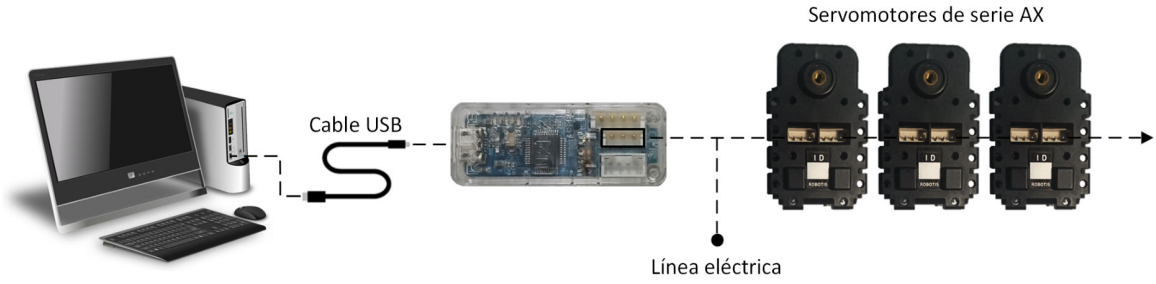


Figura 4.14: Conexión del dispositivo U2D2

Este convertidor de comunicación consta también de tres indicadores LED de estado, como se muestra en la Figura 4.15. El LED rojo indica el estado de la alimentación del dispositivo U2D2, el LED azul indica el estado de la lectura de datos (RxD) y el LED verde indica el estado de la escritura de datos (TxD).

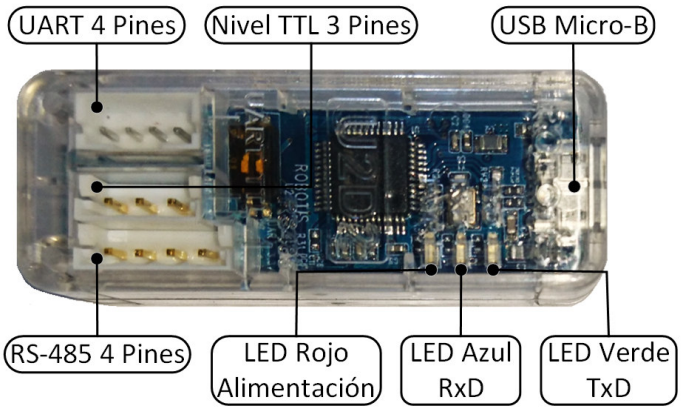


Figura 4.15: Diseño del dispositivo U2D2

En la Figura 4.16 se muestra la configuración de los pines para cada conector de comunicación en el dispositivo U2D2.

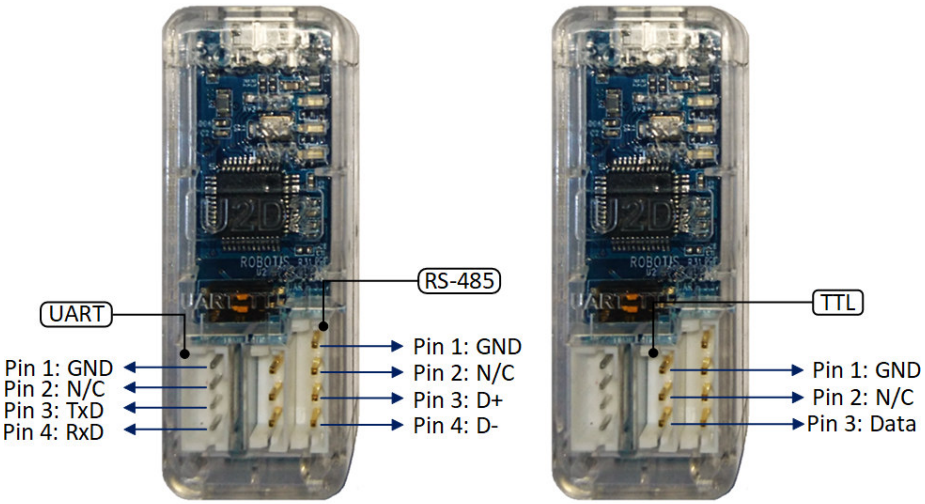


Figura 4.16: Configuración de pines del dispositivo U2D2

Con el uso del dispositivo U2D2 es posible desarrollar un software de control para los servomotores Dynamixel mediante varios lenguajes de programación como C, C++, C#, MATLAB, Python o Java. Para ello el fabricante recomienda el uso de un kit de desarrollo de software llamado DYNAMIXEL SDK, el cual proporciona un conjunto de funciones y métodos para la operación de los servomotores Dynamixel mediante la comunicación por paquetes digitales.

Para obtener más información acerca de este convertidor de comunicación USB, consulte el manual de usuario en ROBOTIS (2020).

4.1.4. Alimentación de servomotores

Como se mencionó en la subsección anterior, los servomotores Dynamixel necesitan tener una fuente de alimentación externa que les suministre el voltaje de entrada. En el manual de usuario que presenta el fabricante, (ROBOTIS, 2020), se recomienda que el voltaje de entrada para los servomotores sea de 11.1 V y además, se indica que los servomotores AX-12A y AX-18A pueden consumir como máximo 0.9 y 2.2 A respectivamente. Por lo tanto, se utilizó una fuente de alimentación tipo cargador adaptador de 12 V a 5 A, ver la Figura 4.17.

Figura 4.17: Fuente de alimentación de 12 V a 5 A

Como la fuente de alimentación seleccionada es de 12 V, fue necesario construir el circuito electrónico de la Figura 4.18 para reducir el voltaje de la fuente a 11.1 V.

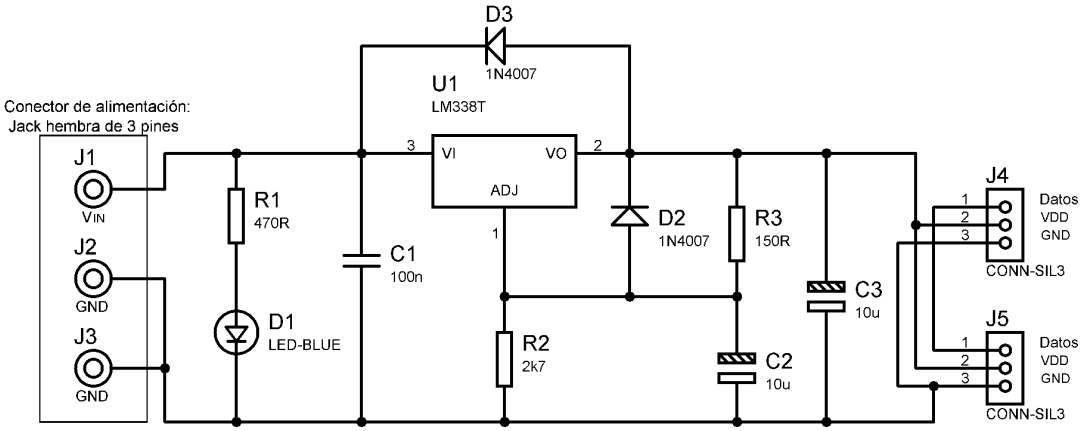


Figura 4.18: Circuito regulador de voltaje a 11.1 V

Las terminales ubicadas en el lado izquierdo del circuito (Figura 4.18), representan al conector de la alimentación. Los dos conectores de tres pines que se ubican en el lado derecho del mismo, permiten proveer el adecuado voltaje de entrada a los servomotores y él envío de datos por medio del dispositivo de comunicación U2D2.

El circuito regulador de voltaje fue diseñado en Proteus 8, ya que el software diseña automáticamente la placa de circuito impreso (Printed Circuit Board, PCB).

Los materiales y componentes electrónicos utilizados para la construcción del circuito regulador de voltaje son los siguientes:

- 1 Regulador de voltaje LM338T.
- 1 Resistencia de 470 Ω, 150 Ω y 2.7 $k\Omega$.
- 2 Capacitores electrolíticos de 10 μF - 25 V.
- 1 Capacitor cerámico de 100 nF - 50 V.
- 1 Diodo emisor de luz (LED).
- 1 Jack hembra de 3 pines (5.5 x 2.1 mm).
- 2 Header macho ángulo recto de 3 pines.
- 1 Placa fenólica de 5 x 5 cm.
- Cloruro férrico, soldadura de estaño y cautín.

En la Figura 4.19 se muestra al regulador de voltaje construido en la placa fenólica, donde se especifica la configuración de sus pines para la adecuada conexión con el brazo robótico y el dispositivo de comunicación U2D2.

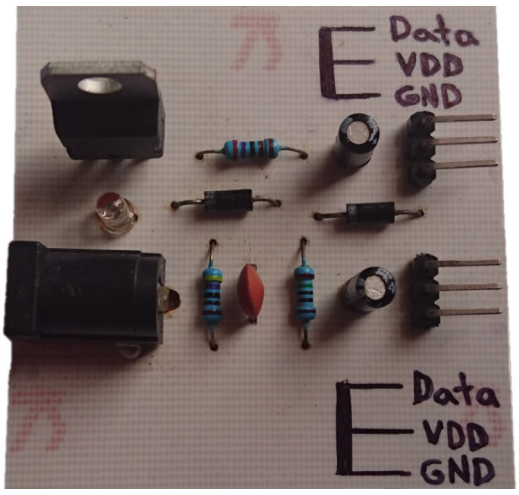


Figura 4.19: Regulador de voltaje en placa fenólica

En la Figura 4.20 se muestra la conexión del brazo robótico con el regulador de voltaje, el cual va conectado con la fuente de alimentación y el dispositivo de comunicación.

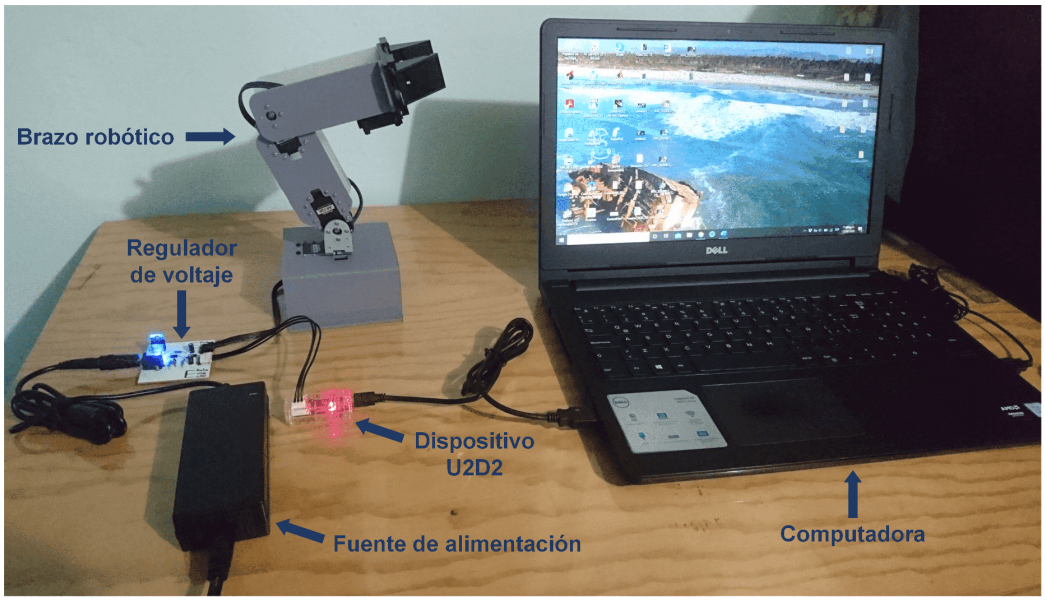


Figura 4.20: Conexión para la alimentación y comunicación del brazo robótico

4.2. Comunicación y operación de los servomotores

La comunicación de los servomotores Dynamixel se realizó a través del dispositivo U2D2, el cual va conectado al puerto USB de una PC y permite convertir las señales del UART al tipo semidúplex de nivel TTL.

Cuando los datos se transmiten desde el puerto USB de la PC al dispositivo U2D2 (Tx), todos los servomotores conectados recibirán datos idénticos. Por otro lado, cuando se reciben datos de uno de los servomotores conectados (Rx), se convertirán automáticamente como comunicación USB y se enviarán a la PC; en caso de recibir datos de dos o más servomotores al mismo tiempo, los datos recibidos pueden estar corruptos. Es por eso que todos los servomotores Dynamixel deben tener un valor único en la red (ID) para permitir la lectura sincronizada de datos.

Para la operación y control de los servomotores, el fabricante facilita un kit de desarrollo de software llamado DYNAMIXEL SDK, el cual proporciona funciones de control mediante comunicación por paquetes digitales. La Interfaz de Programación

de Aplicaciones (API, del inglés: Application Programming Interface) de DYNAMIXEL SDK está diseñada para servomotores Dynamixel y plataformas basadas en Dynamixel. Para el uso correcto del software, el usuario debe estar familiarizado con el lenguaje de programación C y C++.

DYNAMIXEL SDK es compatible con los sistemas operativos: Windows, Linux y MacOS. Además, admite varios lenguajes de programación: C, C++, C#, Python, Java, MATLAB y LabVIEW. Para obtener más información consulte el manual de usuario en ROBOTIS (2020).

4.2.1. Programación de los servomotores en Python

Como parte del desarrollo de este trabajo, se escribió un código de programación en lenguaje Python para lograr la comunicación y operación de los servomotores Dynamixel, por lo que fue necesario instalar previamente, las funciones de control que proporciona el kit DYNAMIXEL SDK para dicho lenguaje de programación. La instalación de las funciones de control se realizó con ayuda del distribuidor Anaconda, vea el apéndice G.

El distribuidor Anaconda tiene incluido al software Spyder, el cual es un entorno de desarrollo integrado y multiplataforma de código abierto (IDE, del inglés: Integrated Development Environment) para programación científica en el lenguaje Python. Por lo tanto, la programación de los servomotores Dynamixel se realizó en Spyder.

En la Figura 4.21 se muestra mediante un diagrama de flujo la representación gráfica del algoritmo de programación que fue desarrollado en Python para lograr la comunicación de los servomotores y el movimiento sincronizado de los mismos, permitiendo así, la ubicación de la pinza mecánica del brazo robótico de 3-GDL en dos posiciones cartesianas establecidas por el usuario.

En el apéndice H se describe una breve explicación del código de programación desarrollado y de las funciones de control que fueron utilizadas para la operación de los servomotores Dynamixel.

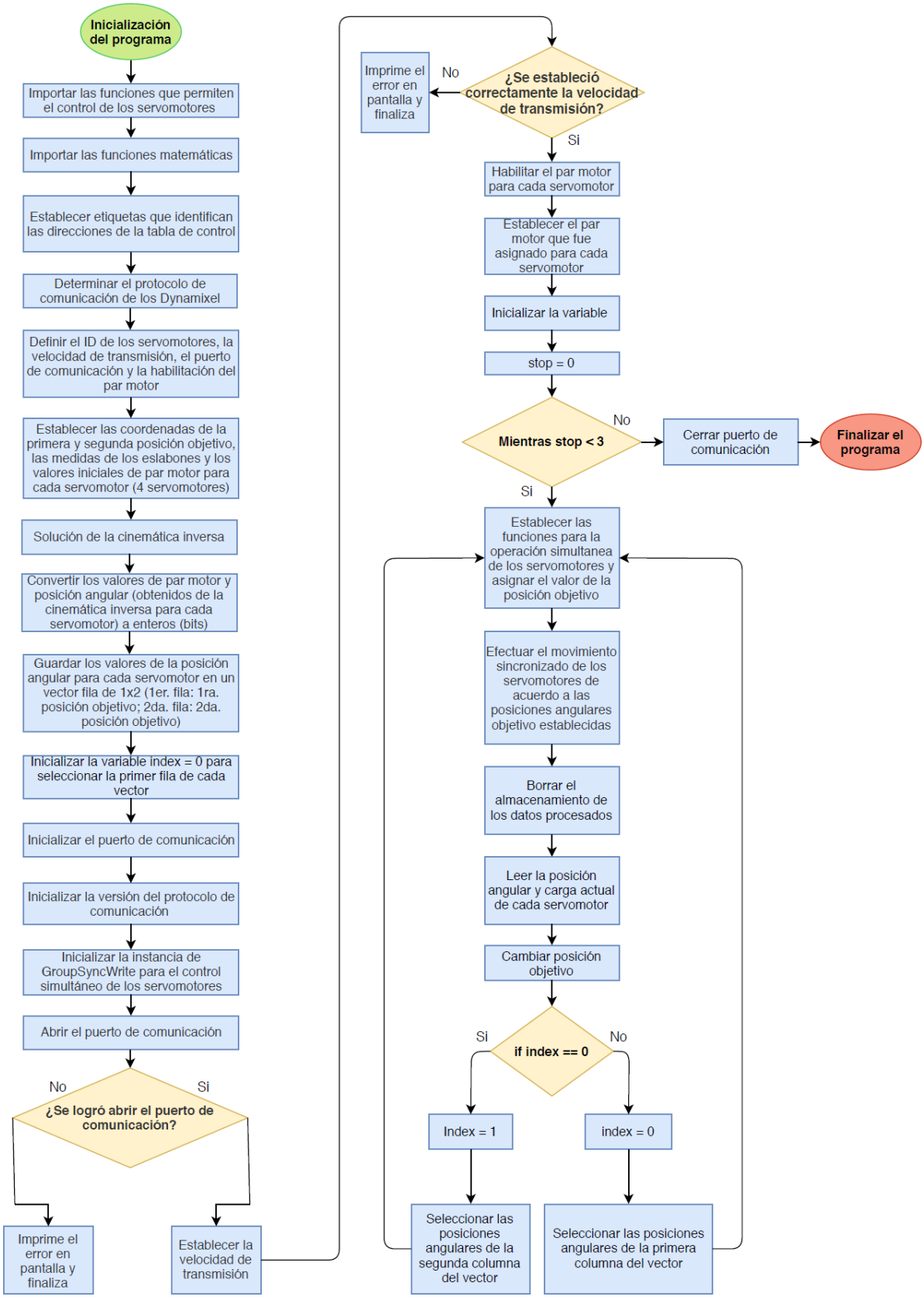


Figura 4.21: Diagrama de flujo para comunicación y movimiento de los servomotores

4.2.2. Ejecución del código de programación

En la Figura 4.22 se observa que, durante la ejecución del programa, la pinza mecánica del brazo robótico logra ubicarse (aparentemente) tanto en la primera como en la segunda posición.

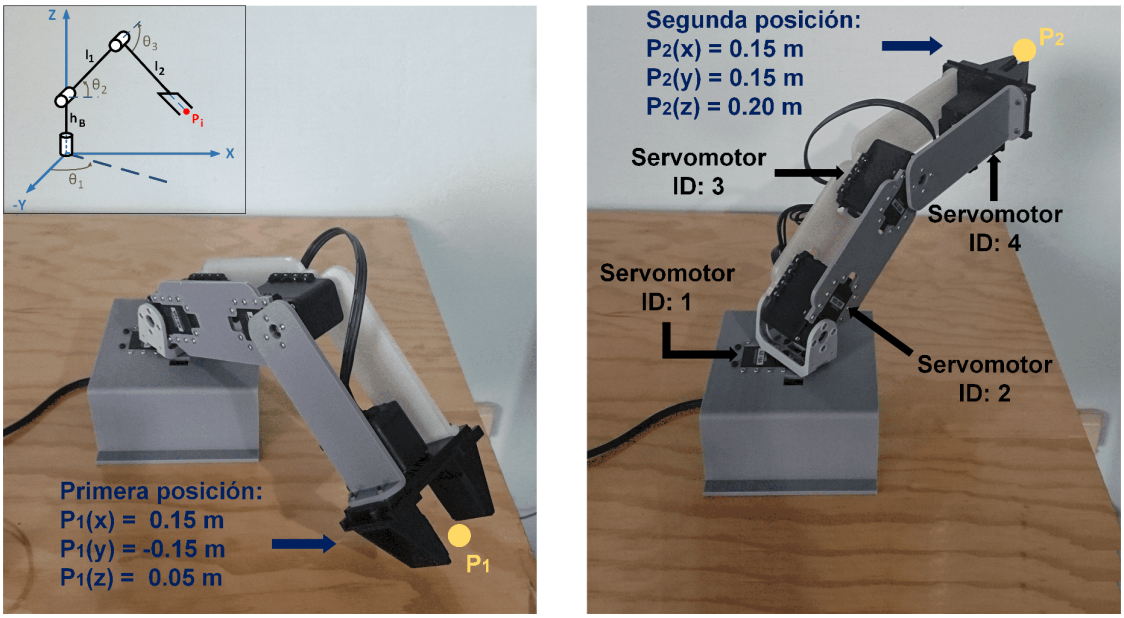


Figura 4.22: Pinza mecánica ubicada en las posiciones establecidas

En la Figura 4.23 se muestran las leyendas que aparecen en la ventana de la consola de Python cuando el programa empieza a ejecutarse, las cuales indican que la comunicación y conexión de los servomotores fue realizada correctamente. Además, más adelante se despliega la lectura de la posición angular y de la carga que presenta cada servomotor (indicando su ID) cuando la pinza mecánica se encuentra ubicada en la primera y segunda posición.

Para verificar la posición angular de cada servomotor, se realizó una comparación de la posición angular requerida (obtenida por la solución de la cinemática inversa) con la posición angular leída, lo que permitirá conocer el porcentaje del error de precisión que presenta cada servomotor por cada cambio de posición, este error se visualiza en la ventana de la consola, vea nuevamente la Figura 4.23.

Del error de precisión que se observa en la ventana de la consola, se puede determinar que la pinza mecánica logra ubicarse correctamente en las dos posiciones que le fueron

establecidas, ya que la máxima diferencia de error que se presenta es de $\pm$ 3.9 %, la cual no es desfavorable para el caso de este trabajo.

```
Python 3.8.3 (default, Jul  2 2020, 17:30:36) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.16.1 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/cesar/Dropbox/MAC DICIS/python servos/3GDL_Servos/Para tesis/
Mover articulaciones - Oficial Tesis/MovArtic_Oficial.py', wdir='C:/Users/cesar/Dropbox/
MAC DICIS/python servos/3GDL_Servos/Para tesis/Mover articulaciones - Oficial Tesis')
Se logro abrir el puerto
Se logro ajustar la velocidad de transmision
Se logro conectar al Dynamixel #1
Se logro conectar al Dynamixel #2
Se logro conectar al Dynamixel #3
Se logro conectar al Dynamixel #4
--------------------------------------------------------------------------------
La pinza mecánica se encuentra ubicada en la posición 1.

[ID: 1] Posición angular actual: -45.31 grados -----  error de precisión: -0.689 %
[ID: 2] Posición angular actual: 27.39 grados  -----  error de precisión: 3.860 %
[ID: 3] Posición angular actual: -65.25 grados -----  error de precisión: -0.696 %
[ID: 4] Posición angular actual: 150.15 grados

[ID: 1] Carga actual: 0.00000 Nm
[ID: 2] Carga actual: 0.23460 Nm
[ID: 3] Carga actual: 0.09384 Nm
[ID: 4] Carga actual: 0.00000 Nm
--------------------------------------------------------------------------------
La pinza mecánica se encuentra ubicada en la posición 2.

[ID: 1] Posición angular actual: 44.72 grados  -----  error de precisión: 0.622 %
[ID: 2] Posición angular actual: 45.28 grados  -----  error de precisión: 2.826 %
[ID: 3] Posición angular actual: -31.52 grados -----  error de precisión: -3.466 %
[ID: 4] Posición angular actual: 0.00 grados

[ID: 1] Carga actual: 0.01408 Nm
[ID: 2] Carga actual: 0.23460 Nm
[ID: 3] Carga actual: 0.09384 Nm
[ID: 4] Carga actual: 0.00000 Nm
--------------------------------------------------------------------------------
La pinza mecánica se encuentra ubicada en la posición 1.

[ID: 1] Posición angular actual: -45.31 grados -----  error de precisión: -0.689 %
[ID: 2] Posición angular actual: 27.39 grados  -----  error de precisión: 3.860 %
[ID: 3] Posición angular actual: -65.54 grados -----  error de precisión: -1.143 %
[ID: 4] Posición angular actual: 150.15 grados

[ID: 1] Carga actual: 0.00000 Nm
[ID: 2] Carga actual: 0.23460 Nm
[ID: 3] Carga actual: 0.09384 Nm
[ID: 4] Carga actual: 0.00000 Nm
```

Figura 4.23: Leyendas que indican el estado de la comunicación y los datos leídos

De esta forma se comprueba que el código de programación funciona correctamente, permitiendo llevar la pinza del robot a cualquier posición que este dentro de su espacio de trabajo. Además, se comprueba que la lectura de la posición angular en cada servomotor es lo suficientemente precisa.

4.3. Implementación de controladores por modos deslizantes

Una ley de control por modos deslizantes con retroalimentación de salida, permite que la salida medida $y(t)$ de un sistema dinámico siga en tiempo finito una salida de referencia $y_r(t)$ a lo largo de un tiempo establecido, aun con la presencia de perturbaciones e incertidumbres paramétricas en el sistema, llevando al error de regulación a cero en tiempo finito. El control por modo deslizante es ideal para ser aplicado en sistemas con dinámicas no lineales, por lo que es más práctico de implementar en modelos dinámicos complejos, como es el caso del brazo robótico de 3-GDL.

Para este trabajo se considera que el modelo dinámico del brazo robótico de 3-GDL se verá afectado por la presencia de incertidumbres paramétricas y por la entrada de fuerzas externas, por lo tanto, el modelo del sistema en espacio de estados estará representado por:

$$\begin{aligned}
\begin{bmatrix} \dot{x}_1 \\ \dot{x}_3 \\ \dot{x}_5 \end{bmatrix} &= \begin{bmatrix} x_2 \\ x_4 \\ x_6 \end{bmatrix} \\
\begin{bmatrix} \dot{x}_2 \\ \dot{x}_4 \\ \dot{x}_6 \end{bmatrix} &= M(x)^{-1}\left[\mathcal{T}_{in} - C(x,\dot{x}) - F_b\dot{x} - G(x) + \mathcal{T}_d\right] \\
\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} &= \begin{bmatrix} x_1 \\ x_3 \\ x_5 \end{bmatrix},
\end{aligned} \tag{4.1}$$

donde x_1, x_3, x_5 representan a las posiciones angulares de los servomotores, las cuales vendrían siendo las salidas del sistema, y x_2, x_4, x_6 representan a las velocidades angulares de los servomotores. Los términos dinámicos de la ecuación (4.1) se muestran en el apéndice A.

Para este caso se requiere que aun con la presencia de incertidumbres y perturbaciones en el sistema, el extremo final del brazo robótico pueda moverse de manera lineal hacia una posición requerida, y una vez que lo logre, regrese a la posición inicial de la misma manera, tal como se muestra en la Figura 4.24.

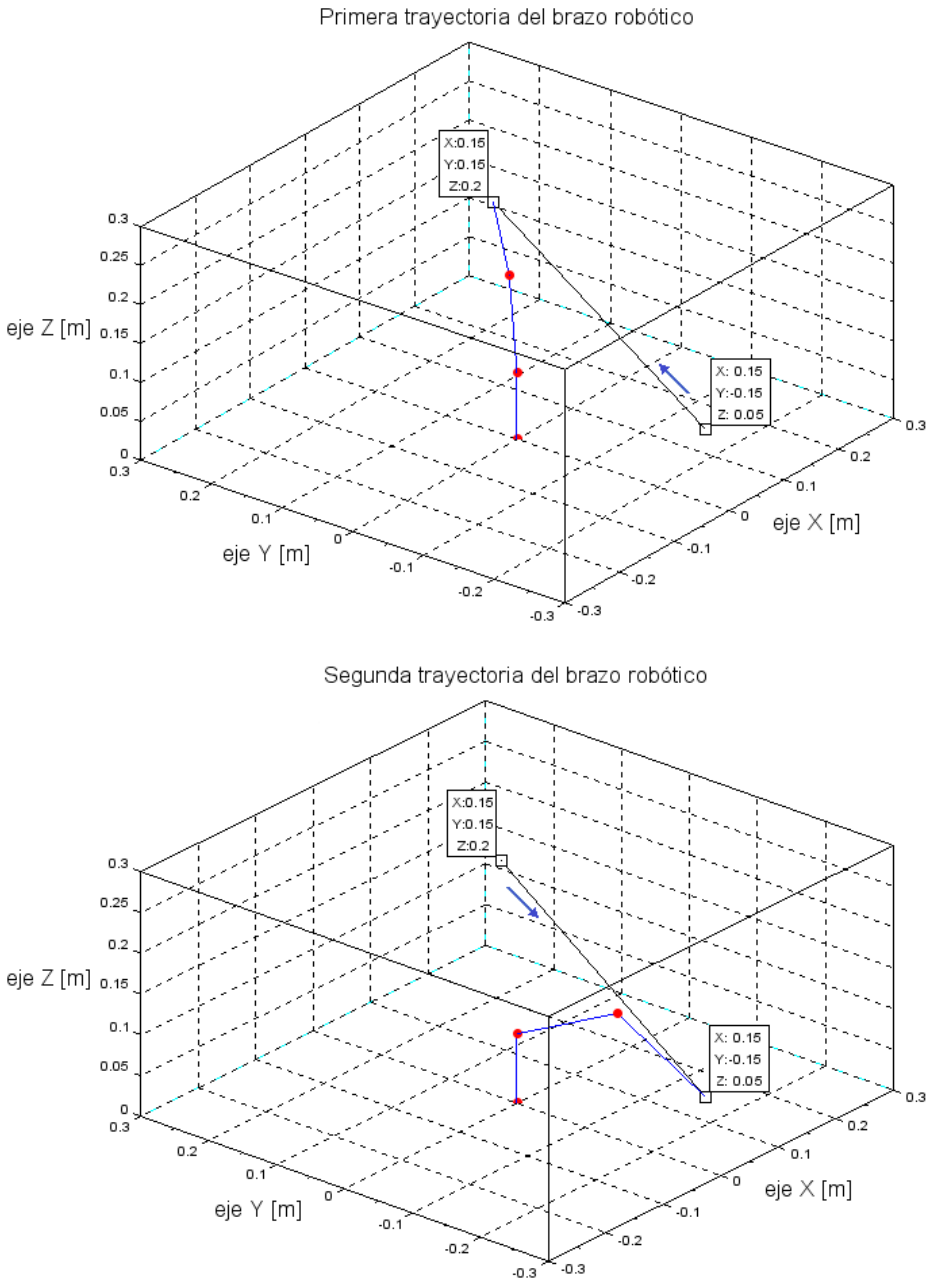


Figura 4.24: Seguimiento de dos trayectorias por el manipulador robótico

Por lo tanto, el problema del control de la posición en el brazo robótico de 3-GDL puede resolverse al proponer una ley de control por modos deslizantes en lazo cerrado, tal que el efecto de las incertidumbres del modelo y de las perturbaciones sea reducido; permitiendo que el robot pueda moverse a las dos posiciones objetivo mediante el seguimiento de dos trayectorias lineales previamente generadas con el uso de las ecuaciones cinemáticas; vea la Figura 4.25.

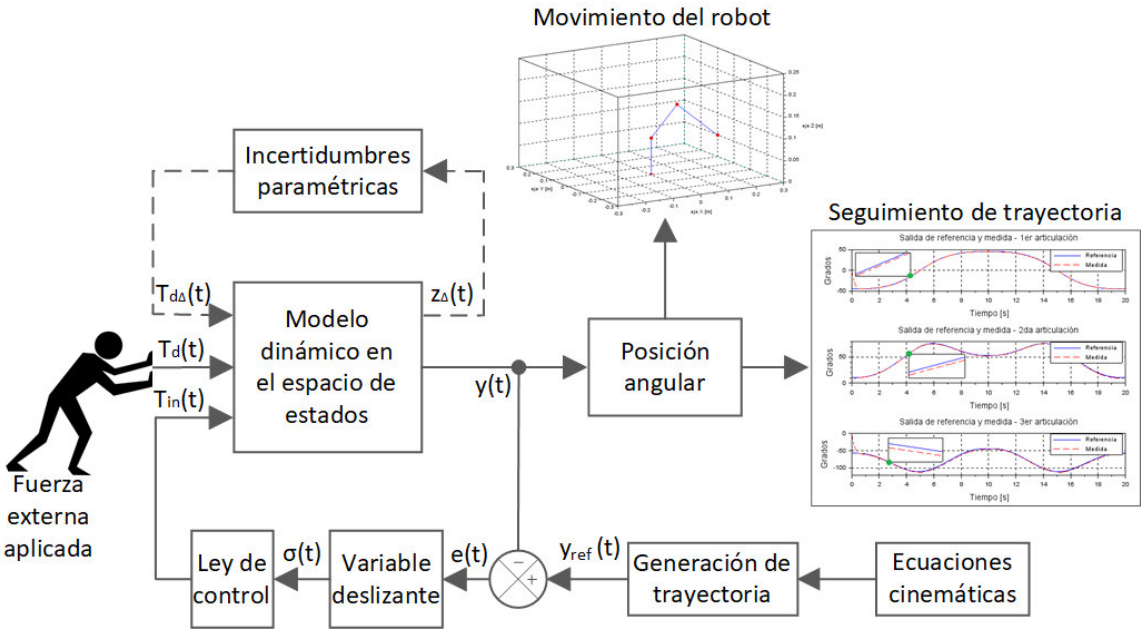


Figura 4.25: Sistema de control en lazo cerrado para brazo robótico

Las trayectorias lineales estarán basadas en la ecuación paramétrica de la recta (de tres dimensiones) y el polinomio de 5to. grado, las cuales fueron definidas en el capítulo 2 por las ecuaciones (2.5) y (2.9) respectivamente.

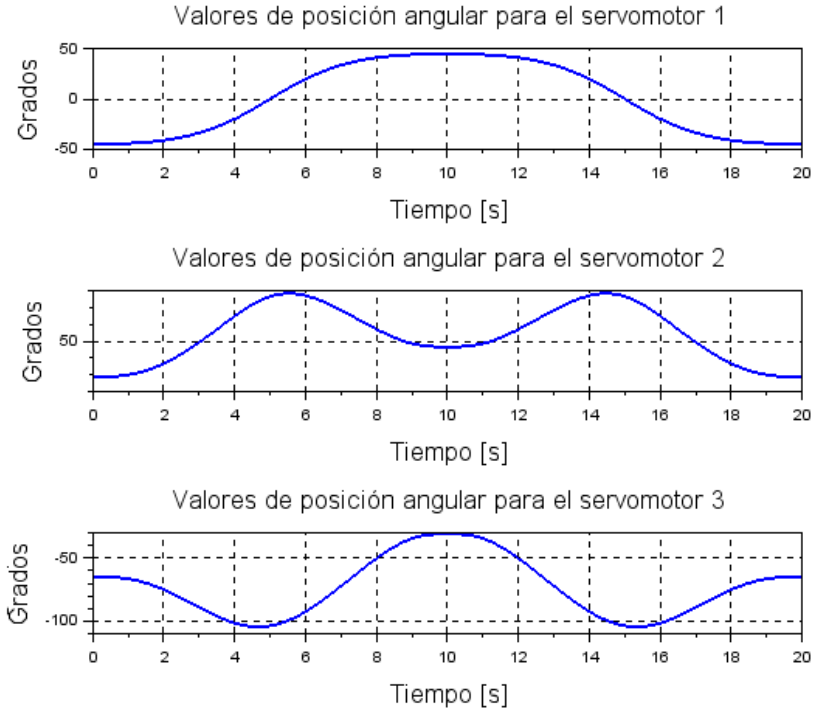


Figura 4.26: Posiciones angulares dadas durante la trayectoria

Una vez planificadas las trayectorias, se generan los valores angulares que deben tomar cada servomotor del brazo robótico para que su extremo final se mueva de acuerdo a las dos trayectorias lineales, estos valores deben ser guardados en un vector.

En la Figura 4.25 se muestran las gráficas de los valores angulares que fueron generados, los cuales serán las respectivas salidas de referencia (y_{1r}, y_{2r}, y_{3r}) para las salidas medidas del sistema (y_1, y_2, y_3).

Para determinar al adecuado algoritmo de control (por modos deslizantes) que permita la estabilidad del sistema robótico en tiempo finito durante el seguimiento de las trayectorias lineales, se debe conocer su grado relativo (ρ); el cual se obtiene mediante la n-esima derivada de la salida que relacione a la entrada $\mathcal{T}_{in}$ del sistema:

$$\begin{aligned} Y(t) &= [x_1 \;\; x_3 \;\; x_5]^T \\ \dot{Y}(t) &= [x_2 \;\; x_4 \;\; x_6]^T \\ \ddot{Y}(t) &= M(x)^{-1}\left[\mathcal{T}_{in} - C(x,\dot{x}) - F_b\dot{x} - G(x) + \mathcal{T}_d\right], \end{aligned}$$

siendo $Y = [y_1 \;\; y_2 \;\; y_3]^T$, por lo tanto, las dinámicas de entrada−salida del sistema robótico considerado tienen grado relativo $\rho = 2$.

4.3.1. Implementación del algoritmo twisting

El algoritmo twisting es una ley de control de modo deslizante de segundo orden que se propone como:

$$\mathcal{T}_{in} = -J_1 sign(\sigma) - J_2 sign(\dot{\sigma}); \quad J_1 > J_2 > 0 \tag{4.2}$$

donde $\mathcal{T}_{in} \in \mathbb{R}^3$, será la entrada de control para el sistema robótico descrito por el modelo no lineal de la ecuación (4.1); $\sigma \in \mathbb{R}^3$, es una variable deslizante que permite la convergencia de los estados a cero para asegurar la estabilidad del sistema; y $J_1, J_2 \in \mathbb{R}^3$, son las ganancias del controlador.

La variable deslizante $\sigma(t,x)$ estará definida por el error de regulación, el cual está dado por la comparación de la salida de referencia $Y_r(t)$ y la salida medida $Y(t)$ del sistema robótico: $\sigma(t) = Y_r(t) - Y(t)$.

Por lo tanto, la ley de control de la ecuación (4.2) con retroalimentación de salida, permitirá reducir el efecto de las perturbaciones e incertidumbres paramétricas en la salida del sistema robótico, logrando que las articulaciones puedan mover sin problema al extremo final del brazo robótico de acuerdo al perfil de trayectoria asignado.

Debido a que el modelo dinámico de la ecuación (4.1) está conformado por tres ecuaciones diferenciales no lineales, entonces, se considera una ley de control twisting para cada ecuación diferencial contenida en el modelo, por lo tanto, los tres controladores twisting se definen como:

$$\begin{aligned} \tau_{in_1}(t) &= -j_{1_1} sign(\sigma_1) - j_{2_1} sign(\dot{\sigma}_1) \\ \tau_{in_2}(t) &= -j_{1_2} sign(\sigma_2) - j_{2_2} sign(\dot{\sigma}_2) \\ \tau_{in_3}(t) &= -j_{1_3} sign(\sigma_3) - j_{2_3} sign(\dot{\sigma}_3) \end{aligned} \tag{4.3}$$

donde:

$$\begin{aligned} \sigma_1(t,x) &= e_1(t) = y_{r_1}(t) - y_1(t) = y_{r_1}(t) - x_1(t) \\ \sigma_2(t,x) &= e_2(t) = y_{r_2}(t) - y_2(t) = y_{r_2}(t) - x_3(t) \\ \sigma_3(t,x) &= e_3(t) = y_{r_3}(t) - y_3(t) = y_{r_3}(t) - x_5(t) \end{aligned} \tag{4.4}$$

y además:

$$\begin{aligned} \dot{\sigma}_1(t,x) &= \dot{e}_1(t) = \dot{y}_{r_1}(t) - \dot{x}_1(t) = \dot{y}_{r_1}(t) - x_2(t) \\ \dot{\sigma}_2(t,x) &= \dot{e}_2(t) = \dot{y}_{r_2}(t) - \dot{x}_3(t) = \dot{y}_{r_2}(t) - x_4(t) \\ \dot{\sigma}_3(t,x) &= \dot{e}_3(t) = \dot{y}_{r_3}(t) - \dot{x}_5(t) = \dot{y}_{r_3}(t) - x_6(t) \end{aligned} \tag{4.5}$$

Los valores propuestos para las ganancias J_1 y J_2 del controlador twisting son:

$$J_1 = [2 \;\; 3 \;\; 2]^T; \qquad J_2 = [1 \;\; 1 \;\; 1]^T;$$

los cuales se determinaron a partir de diferentes pruebas realizadas en la simulación del controlador hasta visualizar los resultados adecuados.

Para efectuar la simulación en SciLab de la implementación del algoritmo de control twsiting con el modelo dinámico del brazo robótico en lazo cerrado, se realizaron los siguientes pasos:

a) Definir una función que describa al modelo dinámico de la ecuación (4.1), en conjunto con los términos dinámicos que están descritos en el apéndice A.

b) Establecer los valores de los parámetros que involucra el modelo dinámico del brazo robótico, vea la Tabla 4.2.

c) Establecer las coordenadas cartesianas de las posiciones objetivo que serán consideradas para la planificación de las dos trayectorias lineales, vea la Tabla 4.3.

d) Establecer los valores de las ganancias J_1 y J_2 para el controlador twisting, además, indicar las condiciones iniciales de los pares motor, posiciones y velocidades angulares de cada articulación.

e) Definir el tiempo en el que se efectuará la simulación y la cantidad de puntos intermedios deseados.

f) Escribir el polinomio de quinto grado de la ecuación (2.9) y la fórmula paramétrica de la recta de la ecuación (2.5) del capítulo 2.

g) Para fines de simulación, definir señales continuas (libres de seleccionar) que servirán como representación de las perturbaciones que pueden afectar a los servomotores del manipulador robótico. Para el desarrollo de esta simulación se consideraron las siguientes señales de perturbación (vea la Figura 3.28):

$$\mathcal{T}_d = \begin{bmatrix} \tau_{d_1} \\ \tau_{d_2} \\ \tau_{d_3} \end{bmatrix} = \begin{bmatrix} 1.8\,\mathrm{sen}(3t^{\frac{4}{6}}) \\ 1.5\,\mathrm{sen}(3t^{\frac{4}{7}}) \\ 1.5\,\mathrm{sen}(t^{\frac{4}{5}}) \end{bmatrix} \tag{4.6}$$

h) En un ciclo *for*, variar los valores de la gravedad y las fricciones viscosas dentro de los límites que se muestran en la Tabla 4.4 para representar la presencia de incertidumbres paramétricas en el sistema. Además, con el uso de la cinemática inversa, generar la planificación de trayectorias para la obtención de las salidas de referencia: y_{r_1}, y_{r_2} y y_{r_3}. Después, realizar la regulación de salida con el uso de las ecuaciones (4.3), (4.4) y (4.5) del controlador twisting y finalmente, resolver la ecuación diferencial de primer orden dada por el modelo dinámico (4.1) del brazo robótico.

i) Realizar las gráficas correspondientes.

Cabe señalar que las perturbaciones aplicadas en los servomotores, representan a las cargas que son producidas por las fuerzas externas aplicadas en los eslabones.

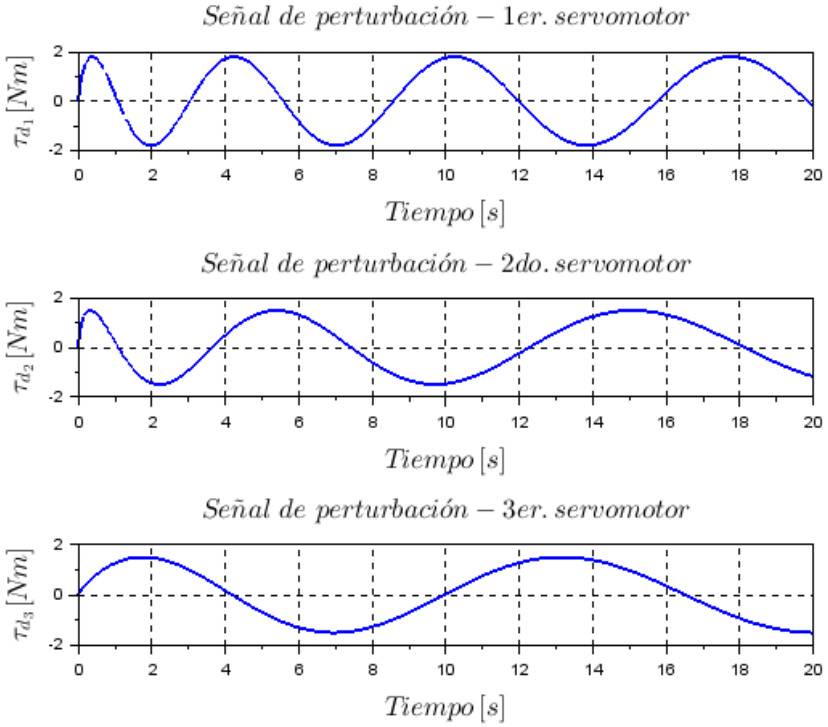


Figura 4.27: Señales de perturbación para cada servomotor

Descripción	**Símbolo**	**Valor**	**Unidades**
Altura de la base	h_B	0.079	m
Longitud del eslabón 1	l_1	0.113	m
Longitud del eslabón 2	l_2	0.140	m
Masa de la base	m_B	0.146	kg
Masa del eslabón 1	m_1	0.147	kg
Masa del eslabón 2	m_2	0.116	kg

Tabla 4.2: Parámetros que involucra el modelo dinámico

Posición	$p(x)$	$p(y)$	$p(z)$
Punto 1	0.15 m	-0.15 m	0.05 m
Punto 2	0.15 m	0.15 m	0.20 m

Tabla 4.3: Coordenadas cartesianas de las posiciones objetivo

Parámetro	Símbolo	Min. valor	Max. valor	Unidades
Aceleración gravitacional	g	9.78	9.81	m/s^2
Fricción viscosa en art. 1	b_1	0	0.1772	$Nm/rad/s$
Fricción viscosa en art. 2	b_2	0	0.2428	$Nm/rad/s$
Fricción viscosa en art. 3	b_3	0	0.2428	$Nm/rad/s$

Tabla 4.4: Valores mínimos y máximos de las incertidumbres paramétricas

En la Figura 4.28 se muestran las convergencias que tuvieron las salidas medidas (y_1, y_2, y_3) con respecto a las salidas de referencia (y_{r_1}, y_{r_2}, y_{r_3}) aun con la presencia de las perturbaciones y las incertidumbres de la gravedad y de las fricciones viscosas. Las señales de color azul representan a las salidas del perfil de referencia, mientras que las señales de color rojo representan a las salidas medidas que fueron reguladas por el controlador twisting.

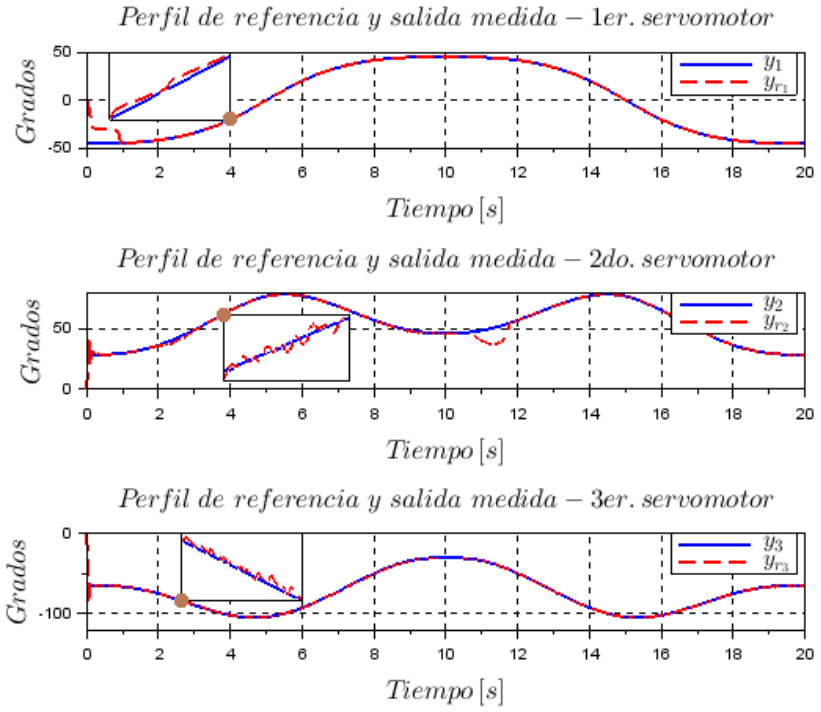


Figura 4.28: Convergencia de las salidas debidas al controlador twisting

En la Figura 4.29 se muestran las señales de control que son entregadas a cada articulación (servomotores) del brazo robótico para que la convergencia de las salidas se logre aun con la presencia de las perturbaciones y las incertidumbres paramétricas. Como es notable observar, se presenta el efecto chattering en cada señal de control, debido a las funciones signo que incluyen los algoritmos twisting.

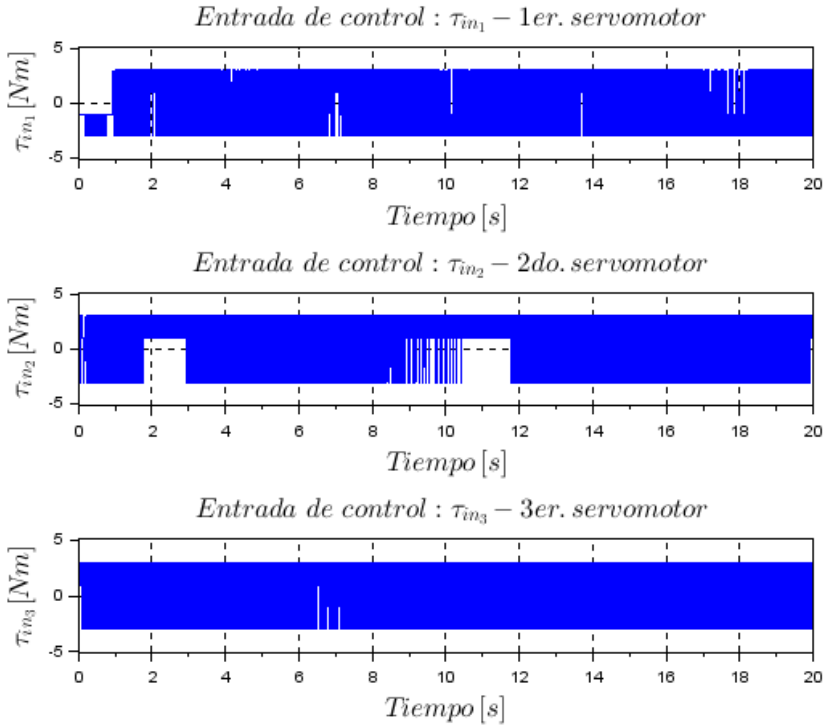


Figura 4.29: Señales de control debidas al controlador twisting

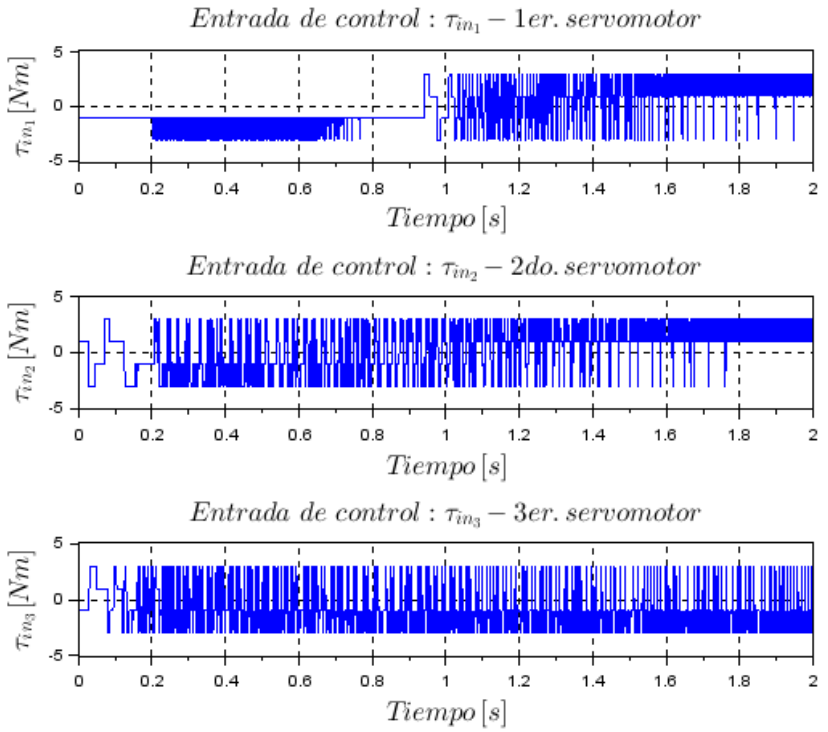


Figura 4.30: Vista amplificada de las señales de control twsiting

En la Figura 4.30 se muestra una vista amplificada de las señales de la Figura 4.29.

Debido a que la señal de control de la Figura 4.29 presenta un comportamiento de zig zag de amplitud y frecuencia finita, entonces, la implementación de un controlador twisting al sistema robótico real no sería una muy buena opción, ya que la presencia de este tipo de oscilaciones podría dañar a los servomotores.

En la Figura 4.31 se muestra la convergencia a cero en tiempo finito tanto de las variables deslizantes σ_i (azul) como de las derivadas $\dot{\sigma}_i$ (roja). Es así como desde entonces, sus dinámicas permanecerán asintóticamente en el origen.

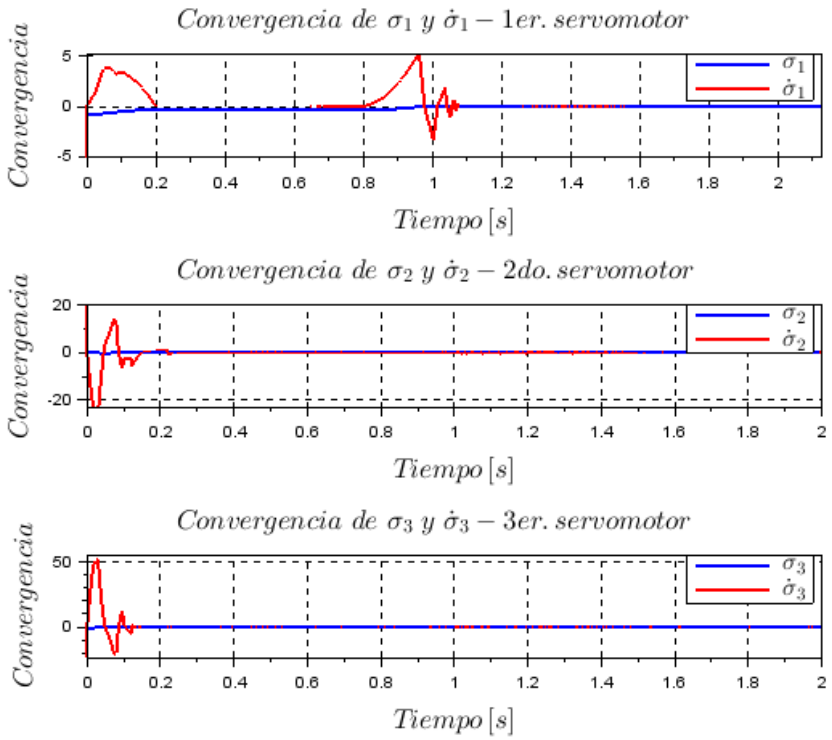


Figura 4.31: Variables deslizantes σ y $\dot{\sigma}$ debidas al controlador twisting

Finalmente en la Figura 4.32 se muestran las gráficas del error de regulación $e(t)$, las cuales están dadas por la comparación de las salidas de referencia (y_{r_1}, y_{r_2}, y_{r_3}) y las salidas medidas (y_1, y_2, y_3) de cada servomotor del brazo robótico de 3-GDL. Como es notable observar, el controlador lleva al error de regulación a cero en tiempo finito.

El código de programación que fue realizado en SciLab para simular al controlador twisting con el sistema robótico, se muestra en el apéndice J de este documento.

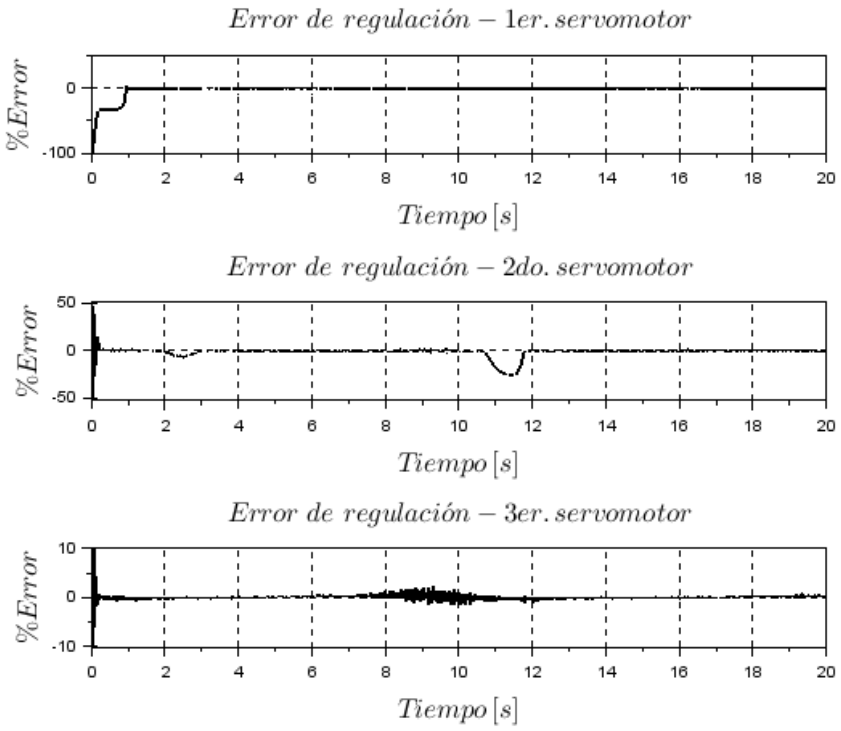


Figura 4.32: Errores de regulación debidos al controlador twisting

4.3.2. Implementación del 3-AST

El algoritmo super-twisting de tercer orden (3-AST) proporciona una señal de control continua (sin chattering) y hace que las dinámicas de un sistema con grado relativo 2 converjan hacia una superficie deslizante en tiempo finito para asegurar la estabilidad del sistema. Para el caso del sistema robótico, el 3-AST se propone como:

$$\begin{aligned} \mathcal{T}_{in} &= -K_1|\sigma|^{\frac{1}{2}} sign(\sigma) + L \\ \dot{L} &= -K_3 sign(\sigma) \\ \sigma &= \dot{e} + K_2|e|^{\frac{2}{3}} sign(e) \end{aligned} \tag{4.7}$$

donde $\mathcal{T}_{in} \in \mathbb{R}^3$, será la entrada de control para el sistema robótico descrito por el modelo no lineal de la ecuación (4.1); $\sigma \in \mathbb{R}^3$, es la variable deslizante que permite la convergencia de los errores de regulación a cero; $L \in \mathbb{R}^3$, es una entrada de control nominal; y $K_1, K_2, K_3 \in \mathbb{R}^3$, son las ganancias del controlador donde sus valores deben seleccionarse apropiadamente.

Por lo tanto, la ley de control de la ecuación (4.7) con retroaliemntación de salida,

permitirá que las salidas del sistema robótico ($Y(t) \in \mathbb{R}^3$, posiciones angulares), sigan en tiempo finito a las salidas de referencia ($Y_r(t) \in \mathbb{R}^3$, posiciones angulares de referencia) a lo largo de un tiempo t establecido, aun con la presencia de las perturbaciones e incertidumbres paramétricas en el sistema; por lo que el controlador llevará al error de regulación a cero asintóticamente.

Al igual que en el caso anterior, el error de regulación está dado por la comparación de la salida de referencia $Y_r(t)$ y la salida medida $Y(t)$ del sistema robótico: $e(t) = Y_r(t) - Y(t)$.

Debido a que el modelo dinámico de la ecuación (4.1) está descrito por tres ecuaciones diferenciales no lineales, entonces, se considera una ley de control 3-AST para cada ecuación diferencial contenida en el modelo, por lo tanto, los tres controladores 3-AST se definen como:

$$\begin{aligned}
\tau_{in_1}(t) &= -k_{1_1}|\sigma_1|^{\frac{1}{2}} sign(\sigma_1) - \int_0^t k_{3_1} sign(\sigma_1)\, dt \\
\tau_{in_2}(t) &= -k_{1_2}|\sigma_2|^{\frac{1}{2}} sign(\sigma_2) - \int_0^t k_{3_2} sign(\sigma_2)\, dt \\
\tau_{in_3}(t) &= -k_{1_3}|\sigma_3|^{\frac{1}{2}} sign(\sigma_3) - \int_0^t k_{3_3} sign(\sigma_3)\, dt,
\end{aligned} \tag{4.8}$$

donde:

$$\begin{aligned}
\sigma_1(t,x) &= \dot{e}_1 + k_{2_1}|e_1|^{\frac{2}{3}} sign(e_1) \\
\sigma_2(t,x) &= \dot{e}_2 + k_{2_2}|e_2|^{\frac{2}{3}} sign(e_2) \\
\sigma_3(t,x) &= \dot{e}_3 + k_{2_3}|e_3|^{\frac{2}{3}} sign(e_3),
\end{aligned} \tag{4.9}$$

Las ganancias de control k_{1_i}, k_{2_i} y k_{3_i} están contenidas en los vectores K_1, K_2 y K_3, respectivamente; además, cada error de regulación se define como:

$$\begin{aligned}
e_1(t) &= y_{r_1}(t) - x_1(t) \\
e_2(t) &= y_{r_2}(t) - x_3(t) \\
e_3(t) &= y_{r_3}(t) - x_5(t),
\end{aligned} \tag{4.10}$$

Para efectuar la simulación en SciLab de la implementación del 3-AST con el modelo dinámico del brazo robótico en lazo cerrado para la regulación de salida, se

realizaron los siguientes pasos:

a) Realizar los primeros seis incisos (excepto *d*) que fueron mencionados en la subsección anterior para la simulación del controlador twisting.

b) Definir tres funciones (una para 3-AST considerado) que describan a la derivada del término auxiliar L de la ecuación (4.8).

c) Seleccionar los valores adecuados de las ganancias de control k_{1_i}, k_{2_i} y k_{3_i} para cada algoritmo de control.

d) En un ciclo *for*, variar los valores de la gravedad y las fricciones viscosas dentro de los límites que se mostraron en la Tabla 4.4 para representar la presencia de incertidumbres paramétricas en el sistema. Además, con el uso de la cinemática inversa, generar la planificación de trayectorias para la obtención de las salidas de referencia: y_{r_1}, y_{r_2} y y_{r_3}; posteriormente, realizar la regulación de salida con el uso de las ecuaciones (4.8), (4.9) y (4.10) del controlador 3-AST; finalmente, resolver la ecuación diferencial de primer orden dada por el modelo dinámico del brazo robótico.

Los valores para las ganancias de control k_{1_i} y k_{2_i} se determinaron a partir de diferentes pruebas que fueron realizadas en la simulación del controlador hasta obtener los resultados adecuados, por lo tanto:

$$k_1 = \begin{bmatrix} 4 & 3 & 2 \end{bmatrix}^T$$
$$k_2 = \begin{bmatrix} 4 & 4 & 4 \end{bmatrix}^T.$$

Para determinar los valores adecuados de las ganancias de control k_{3_i}, es necesario conocer la cota de la derivada de cada perturbación (Δ_i) que afecta a cada articulación del manipulador robótico. Si consideramos nuevamente las perturbaciones que fueron usadas en la simulación del controlador twisting, entonces, las cotas estarán expresadas como:

$$\Delta = \begin{bmatrix} |max\ 1.8\,\mathrm{sen}(3t^{\frac{4}{6}})| \\ |max\ 1.5\,\mathrm{sen}(3t^{\frac{4}{7}})| \\ |max\ 1.5\,\mathrm{sen}(t^{\frac{4}{5}})| \end{bmatrix},$$

donde el valor máximo que puede tomar cada función es la unidad, entonces:

$$\Delta = \begin{bmatrix} 1.8 & 1.5 & 1.5 \end{bmatrix}^T.$$

Por lo tanto, el valor de la ganancia de control k_{3_i} para cada algoritmo de control deberá de cumplir la siguiente condición:

$$k_{3_1} > 1.8 \quad y \quad k_{3_{2,3}} > 1.5$$

A continuación, se muestran las gráficas de los resultados que fueron obtenidos en SciLab de la implementación del 3-AST con el modelo dinámico del brazo robótico en lazo cerrado.

En la Figura 4.33 se muestran las convergencias que tuvieron las salidas medidas (y_1, y_2, y_3) con respecto a las salidas de referencia (y_{r_1}, y_{r_2}, y_{r_3}) aun con la presencia de las perturbaciones y las incertidumbres paramétricas. Las señales de color azul representan a las salidas del perfil de referencia, mientras que las señales de color rojo representan a las salidas medidas que fueron reguladas por el 3-AST.

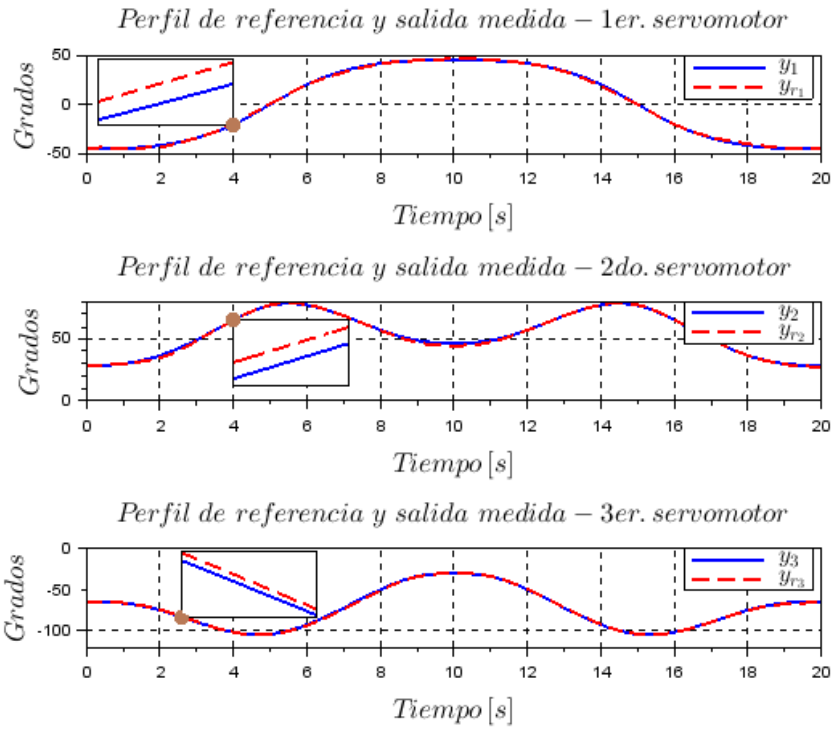


Figura 4.33: Convergencia de las salidas debidas al controlador 3-AST

En la Figura 4.34 se muestran las señales de control que fueron entregadas a cada servomotor (articulación) para que la convergencia de las salidas pudiera ser lograda, y a la vez, el sistema presente estabilidad en tiempo finito. Aquí se puede observar que cada señal de control presenta una señal inversa a la respectiva señal de perturbación, ya que la señal de control trata de entregar un par motor similar (o mayor) y con signo contrario al de la perturbación (carga aplicada) para compensarla, permitiendo reducir el efecto de desviación y así las salidas del sistema puedan seguir asintóticamente a sus salidas de referencia.

Cabe señalar que el signo que acompaña al par motor indica la dirección en la que se movería la articulación si se le aplicará dicho par; lo mismo indica el signo que acompaña a la carga aplicada (perturbación).

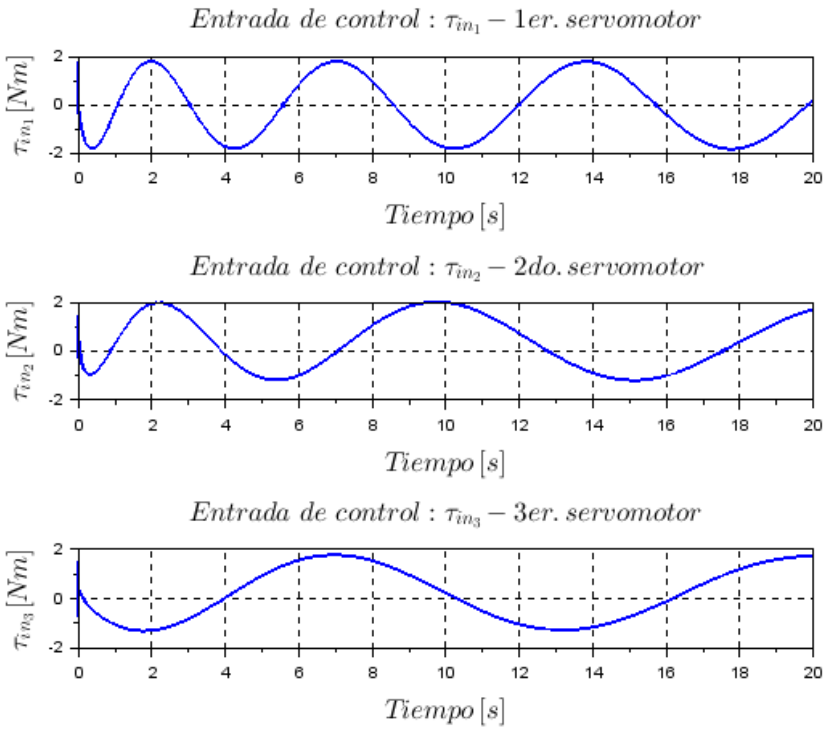


Figura 4.34: Señales de control debidas al controlador 3-AST

Como es notable observar, el efecto chattering no está presente en las señales de control, pero presentan valores de par motor que están fuera del alcance de los servomotores Dynamixel, ya que el par motor máximo que pueden proporcionar los servomotores son: 1.8 Nm para el AX-18A y 1.5 Nm para los AX-12A.

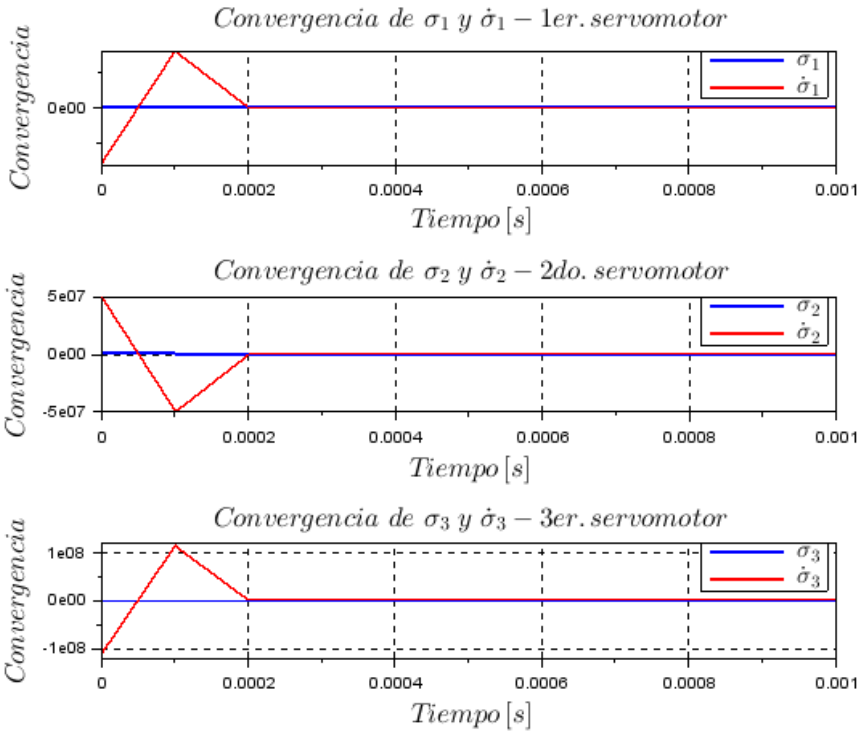


Figura 4.35: Variables deslizantes σ y $\dot{\sigma}$ debidas al controlador 3-AST

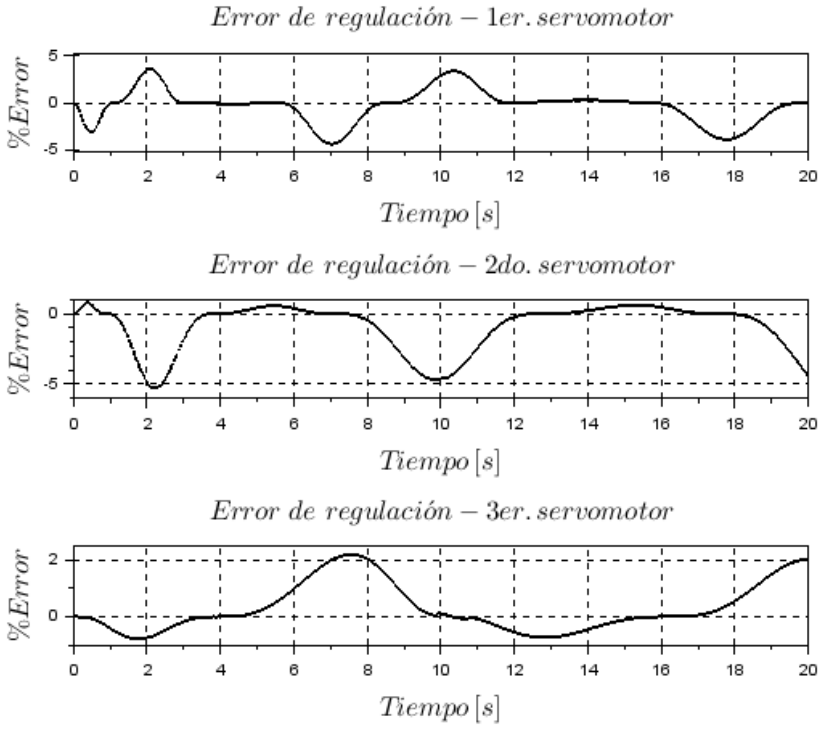


Figura 4.36: Errores de regulación debidos al controlador 3-AST

En la Figura 4.35 se muestran las convergencias a cero en tiempo finito tanto de las variables deslizantes σ_i (azul) como de las derivadas $\dot{\sigma}_i$ (roja). Es así como con las leyes de control de las ecuaciones (4.8) llevan a las variables deslizantes σ_i a cero en tiempo finito y las mantiene ahí para tiempos posteriores, permitiendo así la estabilidad del sistema robótico en tiempo finito.

En la Figura 4.36 se muestran las gráficas que hacen referencia al error de regulación $e(t)$, las cuales están definidas por la comparación de las salidas de referencia (y_{r_1}, y_{r_2}, y_{r_3}) y las salidas medidas (y_1, y_2, y_3) de cada servomotor del brazo robótico. Claramente, cuando las variables deslizantes converjan a cero, los errores de regulación $e(t)$ también lo harán y las salidas medidas convergerán a las salidas de referencia.

Debido a que las señales de control de la Figura 4.34 presentan valores de par motor que están fuera del alcance de los servomotores Dynamixel, entonces, se limitará a cada señal de control entre sus valores máximos y mínimos, por lo que al algoritmo de control 3-AST de la ecuación (4.7) se le aplicará la técnica de control anti-windup que presentan Torres et. al (2018) en su trabajo de investigación. Por lo tanto, el controlador 3-AST anti-windup para el caso del sistema robótico se propone como:

$$\begin{aligned} \mathcal{T}_{in} &= -K_1|\sigma|^{\frac{1}{2}}sign(\sigma) + L \\ \dot{L} &= \begin{cases} -G_{aw}(\mathcal{T}_{min} - \mathcal{T}_{in}); & \mathcal{T}_{in} < \mathcal{T}_{min} \\ -G_{aw}(\mathcal{T}_{max} - \mathcal{T}_{in}); & \mathcal{T}_{in} > \mathcal{T}_{max} \\ -K_3 sign(\sigma) \end{cases} \\ \sigma &= \dot{e} + K_2|e|^{\frac{2}{3}}sign(e) \end{aligned} \tag{4.11}$$

donde $G_{aw} \in \mathbb{R}^3$, es la ganancia del anti-windup, y además:

$$\mathcal{T}_{min} = \begin{bmatrix} -1.8 & -1.5 & -1.5 \end{bmatrix}^T,$$
$$\mathcal{T}_{max} = \begin{bmatrix} 1.8 & 1.5 & 1.5 \end{bmatrix}^T.$$

Los valores propuestos para la ganancia G_{aw} son:

$$G_{aw} = \begin{bmatrix} 20 & 400 & 400 \end{bmatrix}^T.$$

los cuales se determinaron a partir de diferentes pruebas realizadas en la simulación.

A continuación, se muestran los resultados simulados que se obtuvieron al implementar el controlador 3-AST anti-windup de la ecuación (4.11) en lazo cerrado con el sistema robótico descrito por el modelo de la ecuación (4.1).

En la Figura 4.37 se muestran las nuevas señales de control que son proporcionadas a los respectivos servomotores del brazo robótico de 3-GDL para que las salidas medidas (posiciones angulares) puedan seguir en tiempo finito a las salidas del perfil de referencia (posiciones angulares de referencia). Como puede observarse, los valores de par motor, que se presentan en cada señal de control, estan dentro del rango de valores que manejan los servomotores Dynamixel y además cada señal de control está libre del efecto chattering.

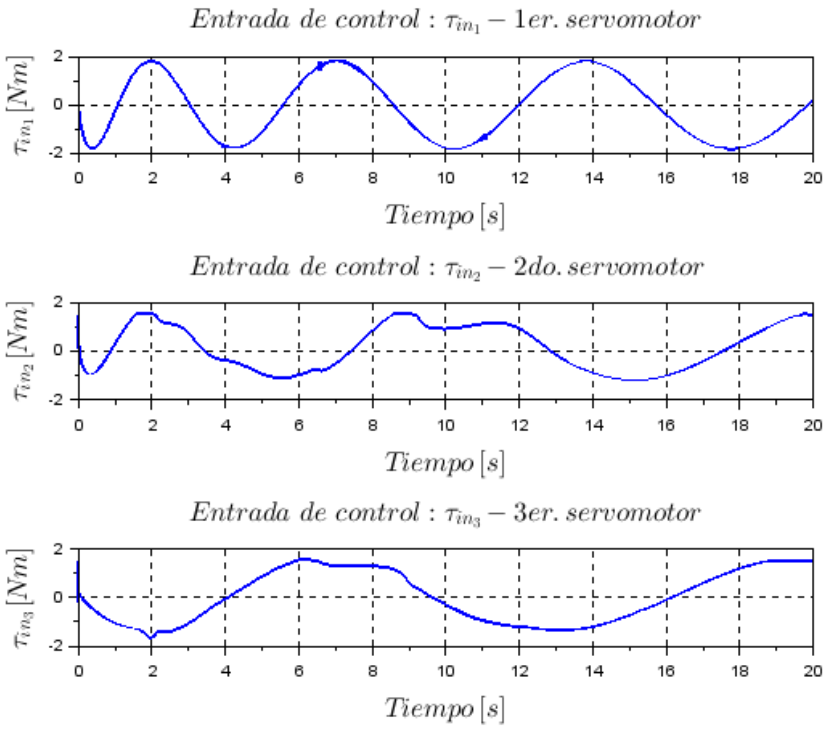


Figura 4.37: Señales de control debidas al controlador 3-AST anti-windup

En la Figura 4.38 se muestra el comportamiento que tuvieron las salidas medidas del sistema robótico mediante el controlador 3-AST anti-windup. Como puede observarse, la salida medida del primer servomotor logra converger adecuadamente con su salida de referencia; en cambio, las salidas medidas tanto del segundo como del tercer servomotor, llegan a desestabilizarse en el mismo tiempo cuando las señales de

control llegan a ser limitadas por sus valores máximos y mínimos, pero aun así, el controlador trata de volver a estabilizar el sistema.

A pesar de los resultados obtenidos, se puede deducir que el controlador 3-AST anti-windup funciona correctamente, ya que genera las señales de control de acuerdo a los límites establecidos; además, el comportamiento que presentaron las salidas medidas del segundo y tercer servomotor tienen cierta lógica, por ejemplo, cuando al segundo servomotor se le aplica una carga (perturbación) igual al par motor máximo que puede generar (1.5 Nm), entonces, para compensar el efecto de la carga aplicada, el servomotor deberá generar un par motor mayor a 1.5 Nm para vencer tanto a la carga aplicada como al peso que generan las masas de los eslabones del brazo robótico, por lo que al limitar la señal de control al par motor máximo que entregan los Dynamixel, este valor será insuficiente para vencer a las dos perturbaciones que se presentan; lo mismo pasaría con el tercer servomotor.

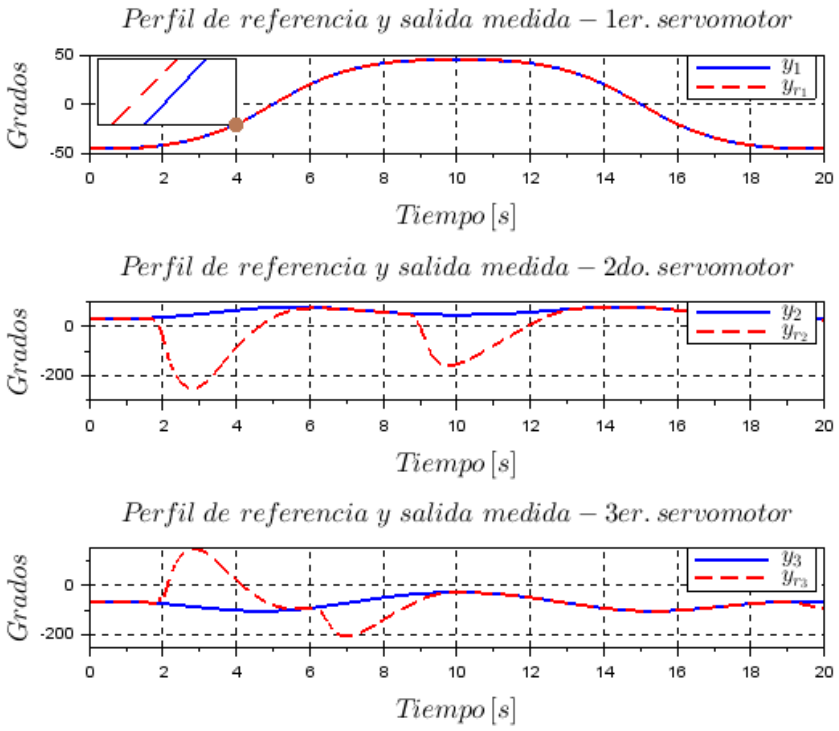


Figura 4.38: Respuestas de las salidas debidas al controlador 3-AST anti-windup

Debido a la ubicación que presenta el primer servomotor en el brazo robótico (vea la Figura 4.1), entonces, este no llegará a verse afectado por la masa de los eslabones;

es por eso que cuando se le aplica una carga de 1.8 Nm, la señal de control entrega un par motor con la misma magnitud, permitiendo que la salida medida converja adecuadamente a la salida del perfil de referencia. De la Figura 4.38 se observa que el segundo servomotor es al que más le afectan las perturbaciones, y esto se debe porque es el servomotor que sostiene a los dos eslabones del brazo robótico.

En la Figura 4.39 se muestran las gráficas que hacen referencia a las variables deslizantes σ_i (azul) y a las variables deslizantes auxiliares $\dot{\sigma}_i$ (roja). Para el caso del primer servomotor, las variables deslizantes permanecen cerca del origen (cero) para tiempos posteriores. Para el caso del segundo y tercer servomotor, se observa que cuando las variables deslizantes auxiliares $\dot{\sigma}_i$ se alejan del origen, las variables deslizantes σ_i también lo hacen y por consecuencia las salidas del sistema ya no logran la convergencia en tiempo finito con las salidas del perfil de referencia.

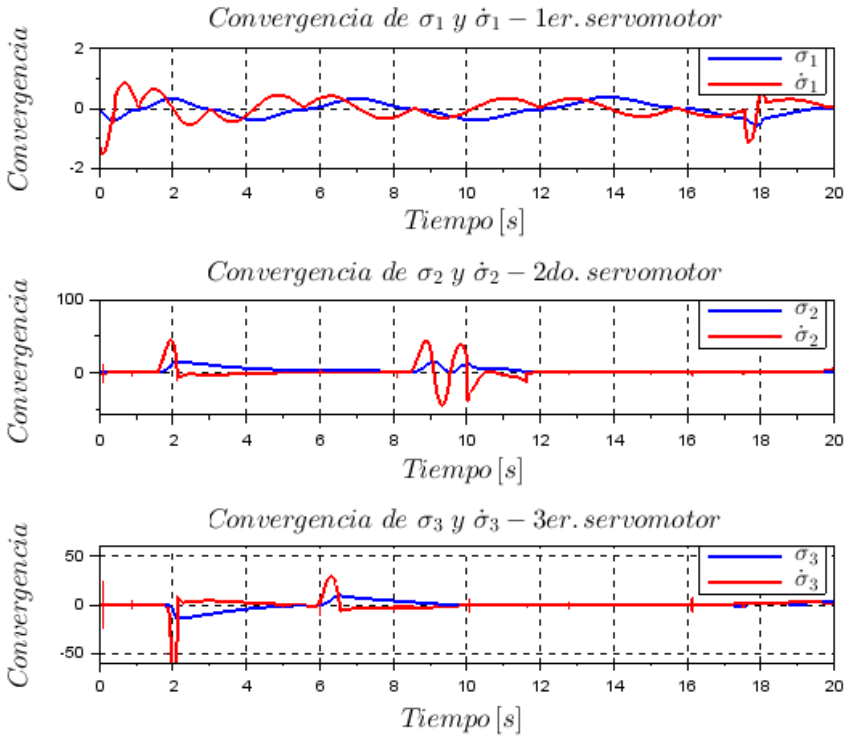


Figura 4.39: Variables deslizantes σ y $\dot{\sigma}$ debidas al controlador 3-AST anti-windup

En la Figura 4.40 se muestran las gráficas que hacen referencia al error de regulación $e(t)$ para cada servomotor.

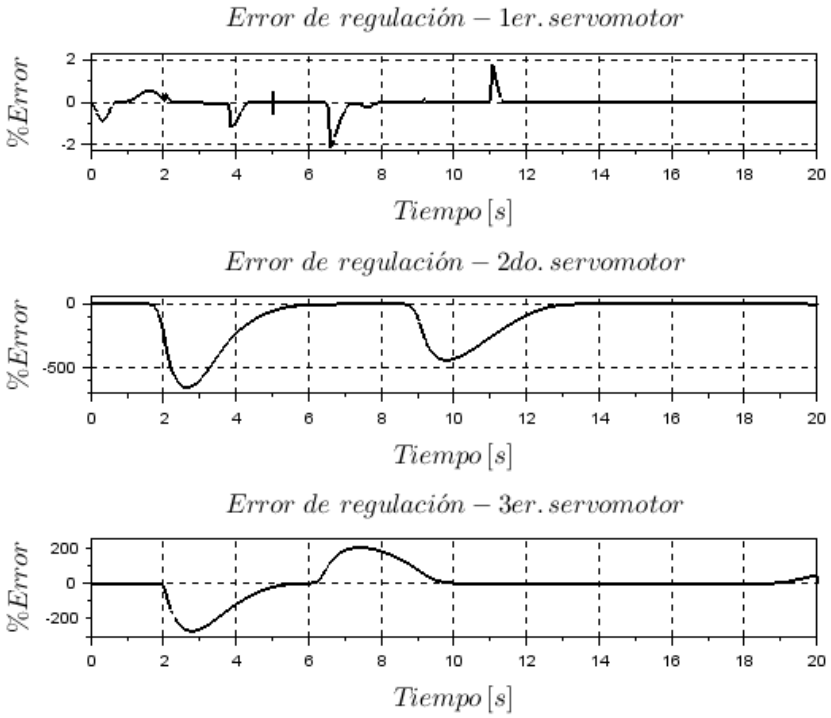


Figura 4.40: Errores de regulación debidos al controlador anti-windup

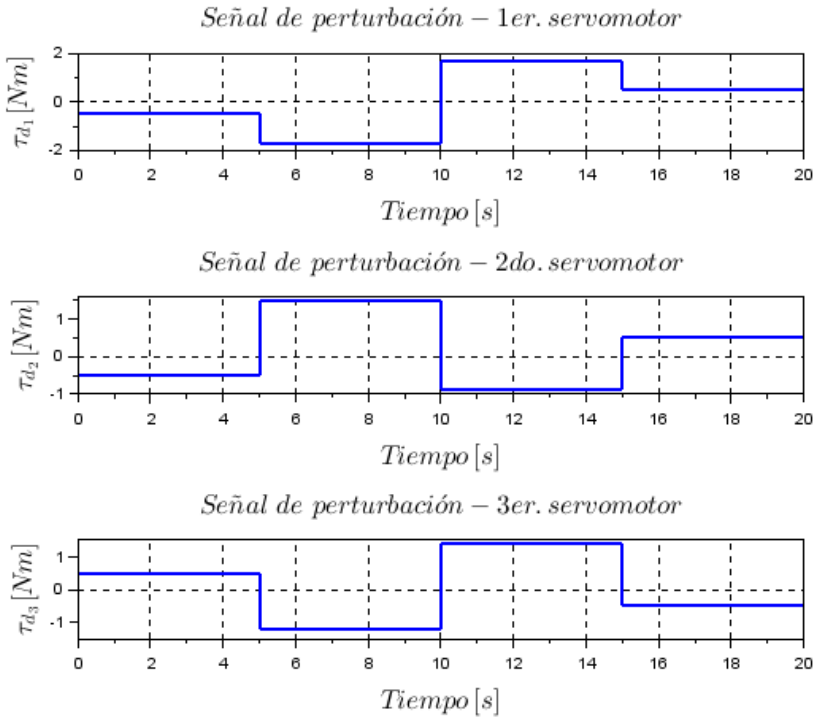


Figura 4.41: Nuevas señales de perturbación consideradas para cada servomotor

Dado a que se quiere realizar una prueba con el brazo robótico real para verificar su funcionamiento mediante el seguimiento de trayectorias lineales, entonces, se realizará una nueva simulación con el controlador 3-AST anti-windup donde se tomará en cuenta la presencia de las señales de perturbación que se muestran en la Figura 4.41.

En la Figura 4.42 se muestra la convergencia que tuvieron las salidas medidas (y_1, y_2, y_3) con respecto a las salidas de referencia (y_{r_1}, y_{r_2}, y_{r_3}) al considerar las nuevas señales de perturbación.

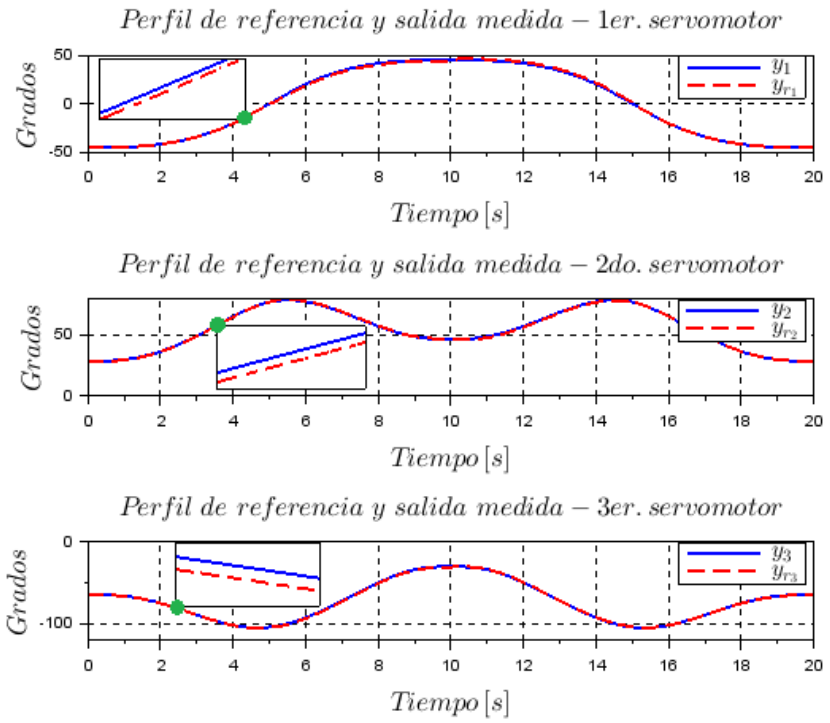


Figura 4.42: Convergencia de salidas al considerar nuevas señales de perturbación

En la Figura 4.43 se muestran las nuevas señales de control que son entregadas a los respectivos servomotores para que el brazo robótico pueda realizar las trayectorias que le fueron asignadas aun con la presencia de las perturbaciones e incertidumbres paramétricas. Los valores que presenta cada señal de control están dentro del rango de valores que manejan los servomotores Dynamixel y además, no presentan chattering.

En la Figura 4.44 se muestran las convergencias a cero en tiempo finito de las variables deslizantes σ_i (azul) y de las derivadas $\dot{\sigma}_i$ (roja), manteniendose ahí para tiempos posteriores.

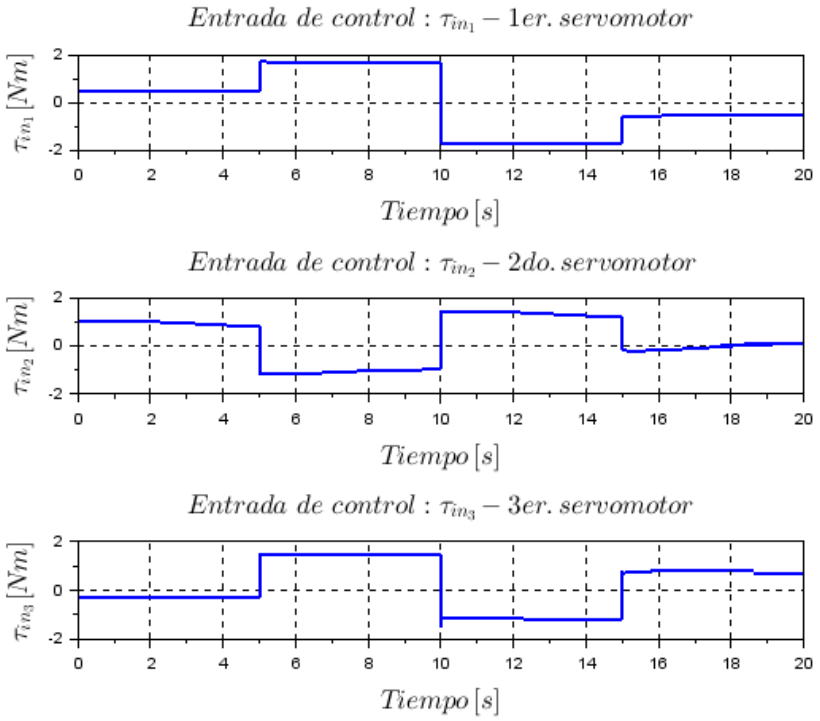


Figura 4.43: Señales de control al considerar las nuevas señales de perturbación

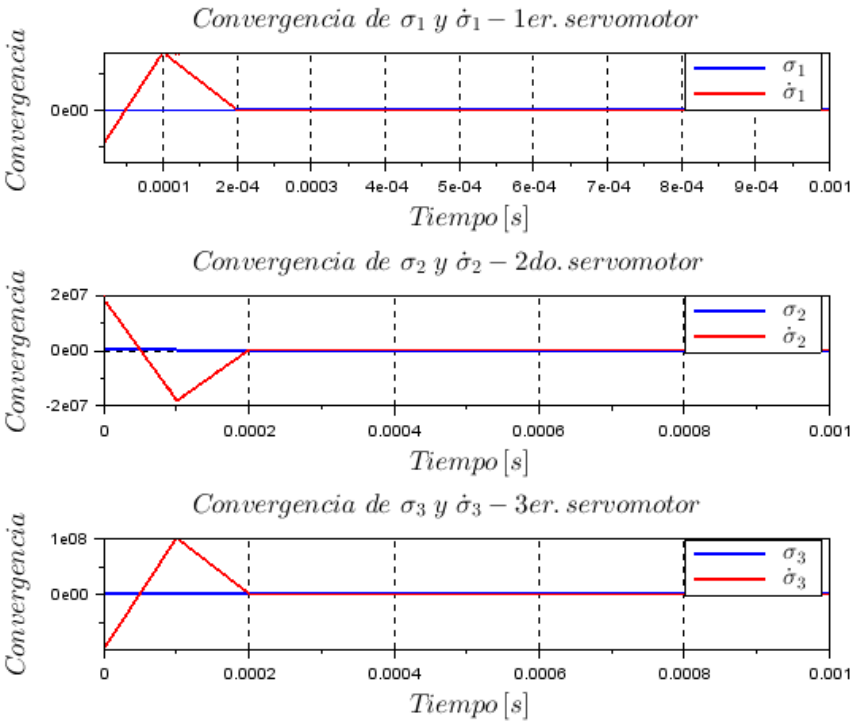


Figura 4.44: Variables deslizantes σ y $\dot{\sigma}$ al considerar las nuevas perturbaciones

En la Figura 4.45 se muestran las gráficas del error de regulación $e(t)$, las cuales están dadas por la comparación de las salidas de referencia (y_{r_1}, y_{r_2}, y_{r_3}) y las salidas medidas (y_1, y_2, y_3) de cada servomotor.

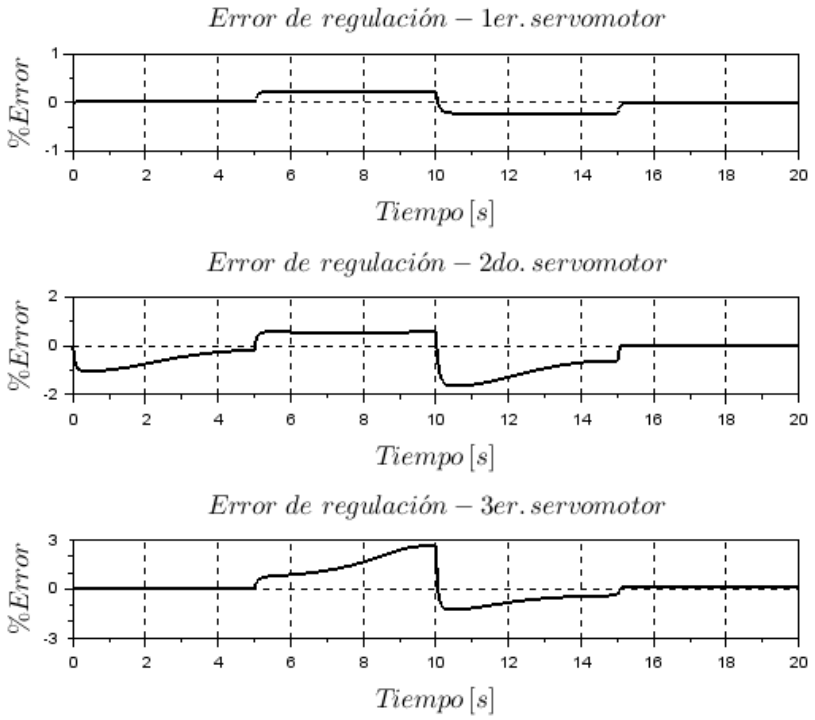


Figura 4.45: Errores de regulación al considerar las nuevas perturbaciones

El código de programación que fue realizado en SciLab para simular al controlador 3-AST anti-windup (4.11) con el modelo dinámico (4.1) del sistema robótico, se muestra en el apéndice K de este documento.

4.4. Escritura de los valores obtenidos por el 3-AST

Como desarrollo final se realizó una prueba que consiste en escribir, en cada servomotor Dynamixel del brazo robótico real, los correspondientes valores de posición angular y par motor que fueron obtenidos por el último controlador 3-AST anti-windup simulado en SciLab, con el objetivo de verificar si realmente el brazo robótico puede realizar las dos trayectorias lineales que fueron definidas en SciLab.

Para llevar a cabo la escritura de dichos valores en los servomotores Dynamixel, previamente en SciLab, se creó un archivo CSV (del inglés, comma-separated values) para guardar todos los valores de posición angular y par motor (θ_1, θ_2, θ_3, τ_{in_1}, τ_{in_2} τ_{in_3})

que se obtuvieron de la última simulación del controlador 3-AST anti-windup aplicado en el modelo dinámico del brazo robótico de 3-GDL en lazo cerrado. Posteriormente, el archivo CSV será leído en Python para escribir los correspondientes valores en los respectivos servomotores.

Los archivos CSV son un tipo de documento que permiten representar datos en forma de tabla, donde las columnas se separan por comas y los renglones por saltos de línea. En la Figura 4.46 se muestra la sintaxis que se escribió en SciLab para la creación del archivo CSV que contendrá los datos requeridos para este trabajo.

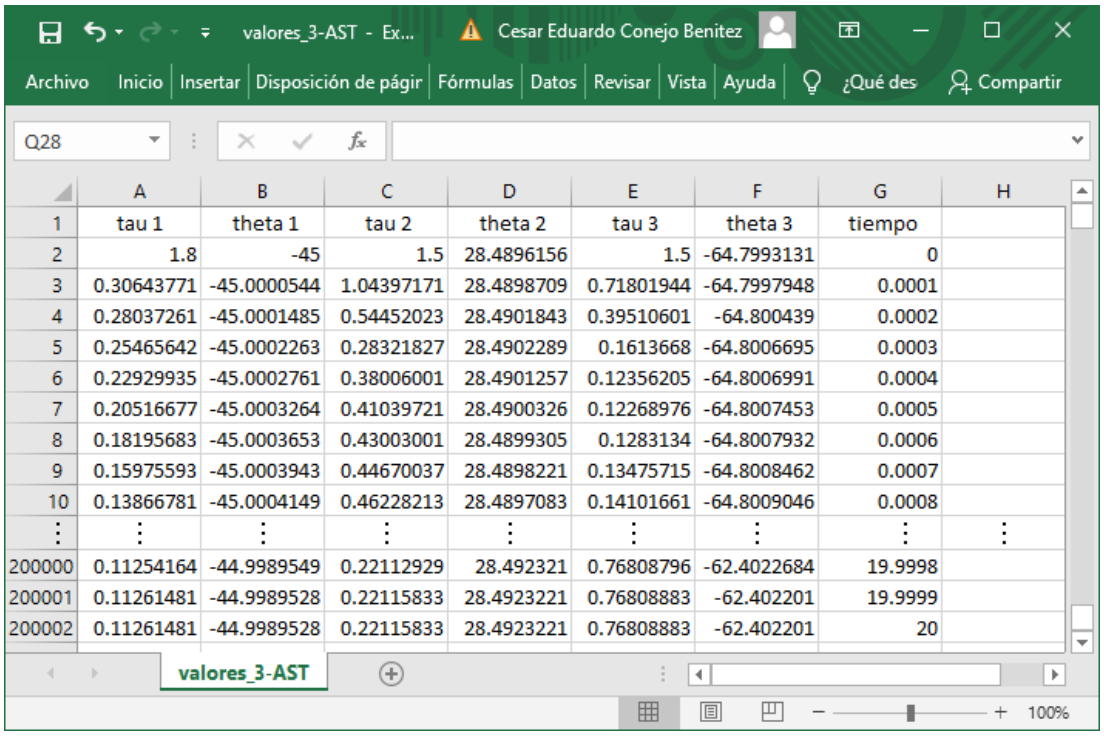
3GDL_3STA.sce

```
154 //·Definicion·del·tiempo:
155 t·=·0:1e-4:20;·····//·segundos
156 lt·=·length(t);
214 //·Definir·el·tamaño·de·la·tabla·que·contendrá·a·los·angulos·y·torques·
215 //·que·se·obtengan·del·controlador:
216 Tabla·=·zeros(lt,·7)
278 //·Implementación·del·controlador·en·lazo·cerrado:
279 for·k=1:lt-1
409 ····//·Guardar·los·valores·de·los·angulos·y·torques·que·se·vayan·obteniendo:
410 ····Tabla(k,1)·=·T1(k);·····//·Torque·para·el·servomotor·ID:·1
411 ····Tabla(k,2)·=·x10(k)*(180/%pi);···//·Pos.·ang.·para·el·servomotor·ID:·1
412 ····Tabla(k,3)·=·T2(k);·····//·Torque·para·el·servomotor·ID:·2
413 ····Tabla(k,4)·=·x30(k)*(180/%pi);···//·Pos.·ang.·para·el·servomotor·ID:·2
414 ····Tabla(k,5)·=·T3(k);·····//·Torque·para·el·servomotor·ID:·3
415 ····Tabla(k,6)·=·x50(k)*(180/%pi);···//·Pos.·ang.·para·el·servomotor·ID:·3
416 ····Tabla(k,7)·=·t(k);·····//·Tiempo·transcurrido
417 end
418
419 //·Crear·archivo·csv·con·los·valores·que·fueron·guardados·en·la·tabla:
420 filename·=·fullfile(TMPDIR,·"valores_3-AST.csv");
421 comments·=·["tau·1,·theta·1,·tau·2,·theta·2,·tau·3,·theta·3,·tiempo"];
422 csvWrite(Tabla,·filename,·[],·[],·[],·comments)
```

Figura 4.46: Sintaxis para la creación del archivo CSV en SciLab

En dicha sintaxis se definió a la variable **Tabla** como una matriz en $\mathbb{R}^{n\times 7}$ que permitirá guardar los datos requeridos. Además, se utilizaron unas funciones donde, **fillename** permite darle nombre al archivo CSV, **comments** permite escribir comentarios al principio de la tabla y **csvWrite** crea el archivo con la información requerida.

El archivo CSV se guarda como formato Excel y para este caso, estará conformado por n renglones (de acuerdo número de veces que se ejecutó el ciclo *for* en SciLab) y 7 columnas, tal como se muestra Figura 4.47.



	A	B	C	D	E	F	G	H
1	tau 1	theta 1	tau 2	theta 2	tau 3	theta 3	tiempo	
2	1.8	-45	1.5	28.4896156	1.5	-64.7993131	0	
3	0.30643771	-45.0000544	1.04397171	28.4898709	0.71801944	-64.7997948	0.0001	
4	0.28037261	-45.0001485	0.54452023	28.4901843	0.39510601	-64.800439	0.0002	
5	0.25465642	-45.0002263	0.28321827	28.4902289	0.1613668	-64.8006695	0.0003	
6	0.22929935	-45.0002761	0.38006001	28.4901257	0.12356205	-64.8006991	0.0004	
7	0.20516677	-45.0003264	0.41039721	28.4900326	0.12268976	-64.8007453	0.0005	
8	0.18195683	-45.0003653	0.43003001	28.4899305	0.1283134	-64.8007932	0.0006	
9	0.15975593	-45.0003943	0.44670037	28.4898221	0.13475715	-64.8008462	0.0007	
10	0.13866781	-45.0004149	0.46228213	28.4897083	0.14101661	-64.8009046	0.0008	
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
200000	0.11254164	-44.9989549	0.22112929	28.492321	0.76808796	-62.4022684	19.9998	
200001	0.11261481	-44.9989528	0.22115833	28.4923221	0.76808883	-62.402201	19.9999	
200002	0.11261481	-44.9989528	0.22115833	28.4923221	0.76808883	-62.402201	20	

Figura 4.47: Archivo CSV en formato Excel con los datos requeridos

4.4.1. Escritura de valores en los servomotores Dynamixel

Una vez creado el archivo CSV con los valores requeridos, se realiza un código de programación en Python que permita la lectura del archivo y la escritura de dichos valores en los servomotores del brazo robótico real, además, permitirá que la pinza mecánica (ubicada en la posición inicial) agarre un primer objeto y logre trasladarlo a la posición objetivo; una vez que llegue a dicha posición, la pinza mecánica soltará el objeto y agarrará un segundo objeto para trasladarlo a la posición anterior.

En este programa se usan las funciones de control que fueron implementadas en el código de programación que se mostró en el apéndice H.

En la Figura 4.48 se muestra el diagrama de flujo que representa al programa desarrollado en Python. En el apéndice L se redacta la sintaxis del programa.

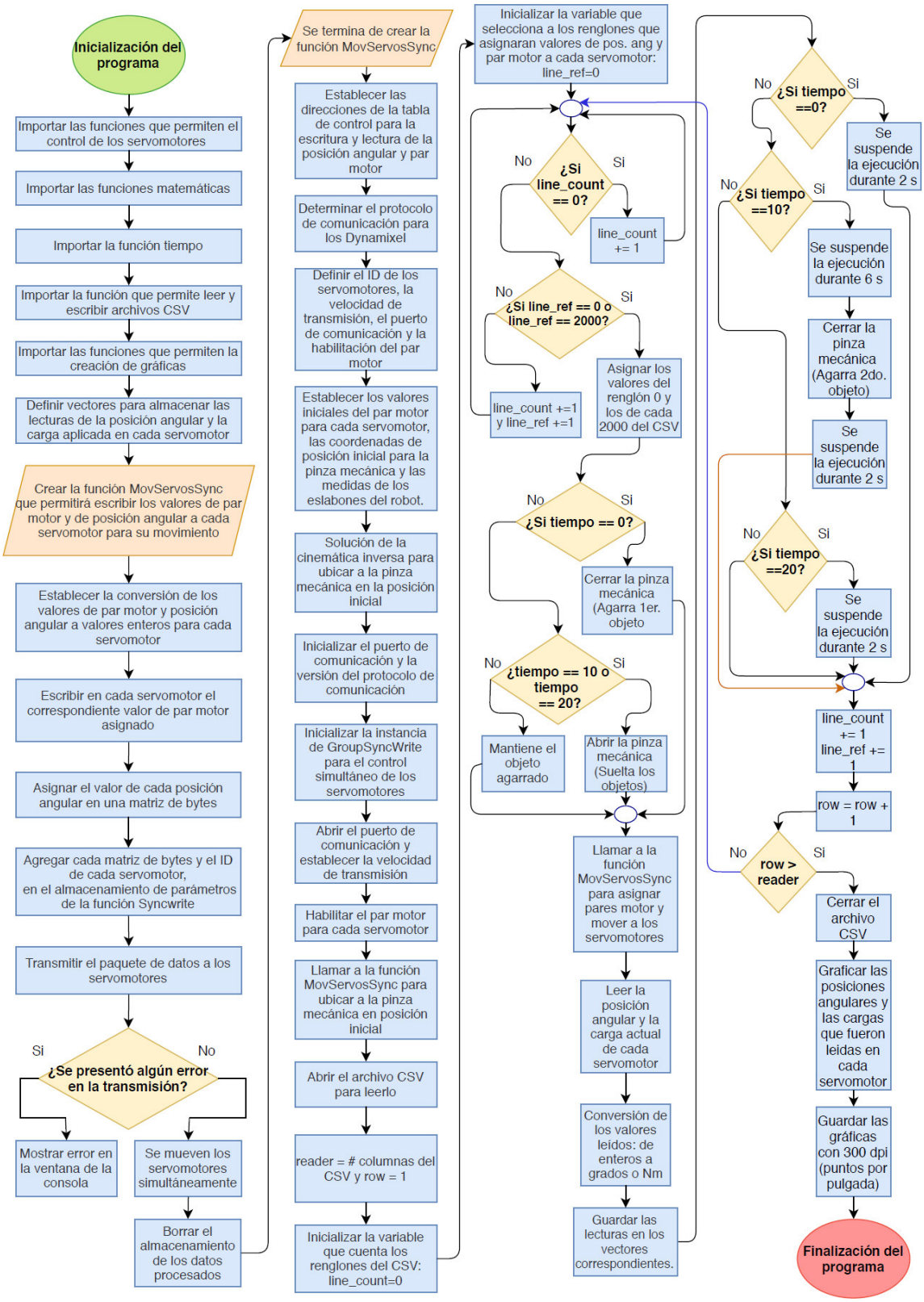


Figura 4.48: Diagrama de flujo para escritura de valores y control de la pinza

4.4.2. Resultados

Una vez finalizada la escritura del programa en Python, prosigue su compilación. Para este caso se usaron los objetos que se muestran en la Figura 4.49, los cuales servirán como perturbaciones (debido a sus masas). Estos objetos serán trasladados por el brazo robótico a las posiciones que anteriormente fueron establecidas.

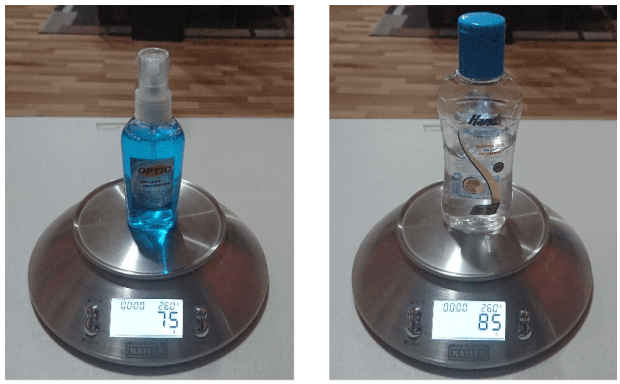


Figura 4.49: Masas de los objetos que serán traslados por el brazo robótico

En la Figura 4.50 se observa la operación que fue llevando a cabo el brazo robótico (seguimiento de trayectorias) durante la compilación del código de programación.

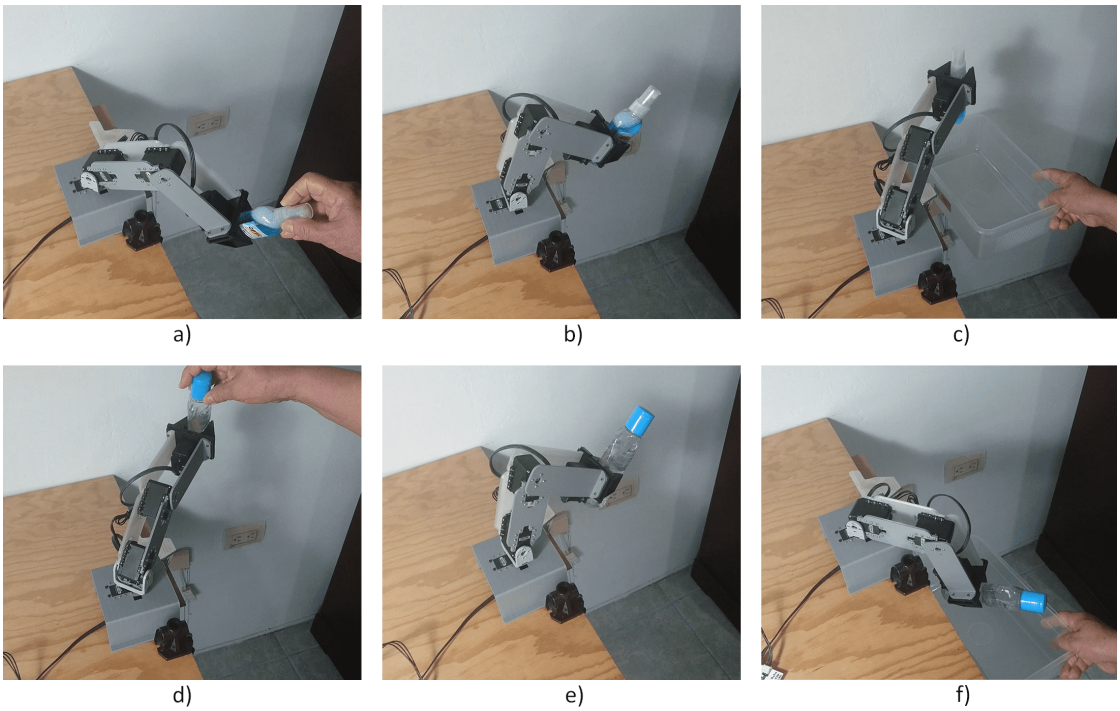


Figura 4.50: Primer compilación del programa

Después de que el brazo robótico finaliza con la realización de las trayectorias, se muestran en pantalla las gráficas de las lecturas que se obtuvieron de cada servomotor durante su operación. En las Figuras 4.51 y 4.52 se muestran respectivamente, las gráficas de las posiciones angulares y cargas leídas.

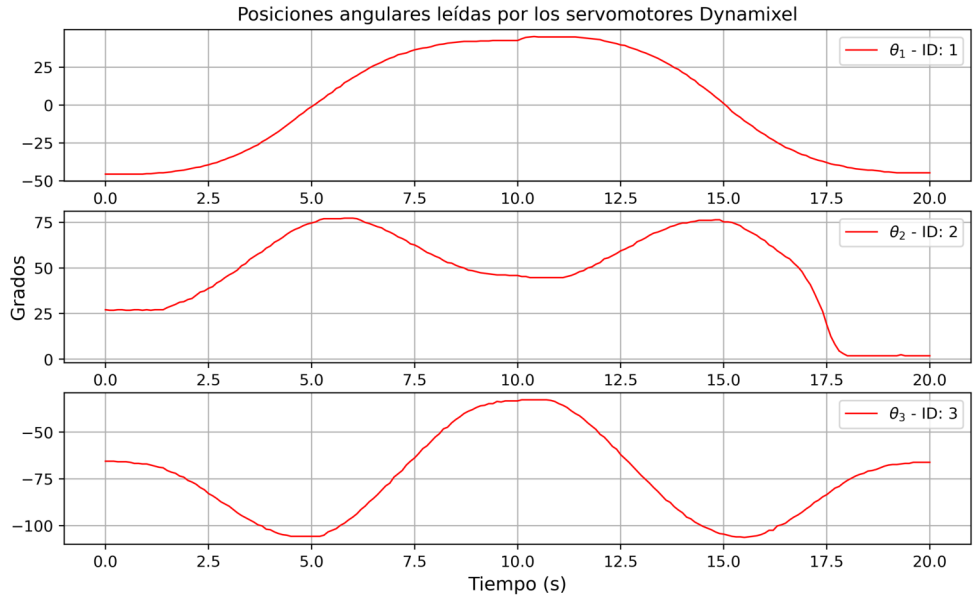


Figura 4.51: Posiciones angulares leídas durante la primer ejecución

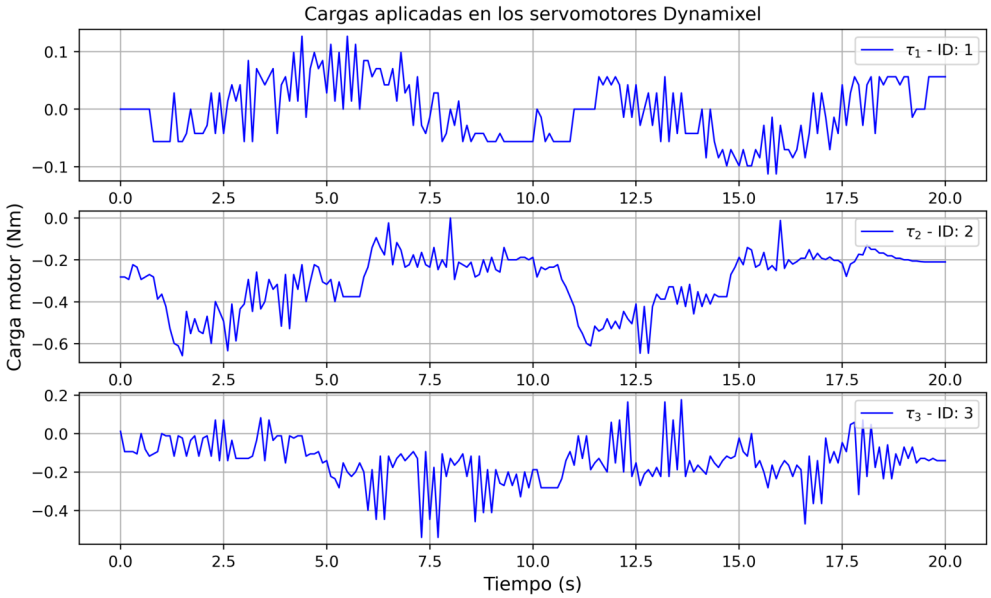


Figura 4.52: Cargas aplicadas durante la primer ejecución

De la Figura 4.50 en los incisos a), b) y c) se observa que la pinza mecánica del brazo

robótico logra ubicarse en la posición inicial y toma al primer objeto, posteriormente lo traslada a la posición objetivo (siguiendo una trayectoria lineal) y lo suelta una vez que haya logrado ubicarse en dicha posición; en los incisos d), e) y f) se observa que el brazo robótico toma al segundo objeto y trata trasladarlo a la posición donde inicialmente estaba ubicada la pinza mecánica. Del inciso f) se puede notar que el brazo robótico llega a desestabilizarse cuando el segundo objeto se acerca a la posición objetivo; esto se debe a que, en un cierto momento, la masa de los eslabones y la del segundo objeto llegaron a aplicar una carga mayor al par motor que suministraba la articulación (servomotor) afectada.

De las gráficas que muestran las posiciones angulares leídas (Figura 4.51), se observa que cada articulación sigue un perfil de trayectoria con movimientos angulares suaves (debido a la planificación de trayectoria que se implementó en SciLab), pero después de 16 s, la segunda articulación (servomotor ID: 2) ya no cumple con dicho perfil. Por lo tanto, la segunda articulación fue la que se vio afectada por la carga aplicada.

En la Figura 4.52 se observan las cargas que son aplicadas en cada articulación del brazo robótico en un cierto instante de tiempo debido a la propia masa de los eslabones del robot. Aquí se puede notar que la segunda articulación es a la que se le aplica mayor carga, ya que es la articulación que soporta a toda la estructura robótica. Además, se puede observar que después de 16 s, a la segunda articulación se le aplicaron cargas menores y cercanas a 0.2 Nm en el sentido a las manecillas del reloj (cargas con signo negativo), por lo tanto, se puede determinar que en ese momento dicha articulación suministraba un par motor menor a 0.2 Nm y por tal motivo, el brazo robótico no pudo terminar adecuadamente con el perfil de trayectoria.

Mediante una gráfica de errores se puede comprobar si las posiciones angulares leídas eran las adecuadas. Por lo tanto, se comparan las posiciones angulares leídas con respecto a las que se leyeron del archivo CSV (obtenidas de la simulación del controlador 3-AST anti-windup); estas gráficas se muestran en la Figura 4.53.

De la Figura 4.53 se observa que el error de posición angular para el primer y tercer servomotor (ID: 1, 3) se encuentran dentro de lo normal, ya que el error es mínimo. Este error puede deberse a la resolución que presentan los servomotores (0.39°). Para el caso del segundo servomotor (ID: 2) se nota que el error de posición angular es

mayor después de los 16 s, debido a la carga que llegó a afectar al servomotor en ese intervalo de tiempo.

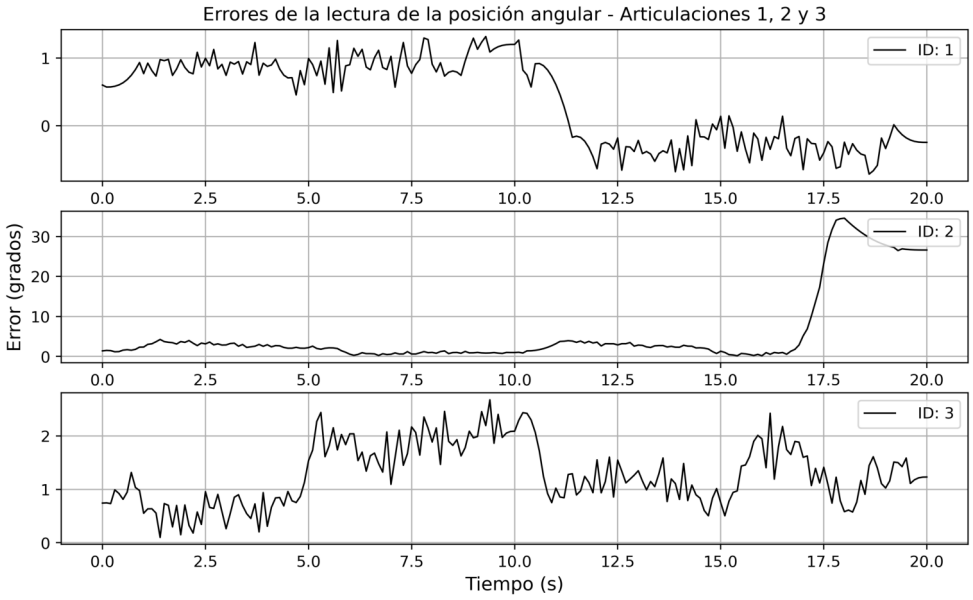


Figura 4.53: Errores de posición angular de la primer compilación

Una solución que podría implementarse para que el brazo robótico pueda finalizar la ejecución de sus trayectorias correctamente es limitando nuevamente al par motor, desde la simulación en SciLab se le puede establecer al controlador 3-AST anti-windup que entregue pares motor entre $\pm$ 0.3 Nm y $\pm$ 1.5 Nm para la segunda articulación, ya que esta articulación es débil cuando suministra pares motor menores a 0.3 Nm.

Una vez aplicada está técnica en SciLab, se volvió a compilar nuevamente el código de programación en Python. En la Figura 4.54 se observa la operación que fue llevando a cabo el brazo robótico al momento que el programa se compilaba por segunda vez.

Después de que el brazo robótico finaliza con su operación, se muestran en pantalla las gráficas de las lecturas que se obtuvieron de cada servomotor. En las Figuras 4.55 y 4.56 se muestran respectivamente, las gráficas que corresponden a las posiciones angulares y cargas leídas.

De la Figura 4.54 se observa que aparentemente la pinza mecánica logra ubicarse en las posiciones cartesianas establecidas, además, también se observa que los objetos fueron trasladados de acuerdo a las trayectorias lineales planificadas en SciLab.

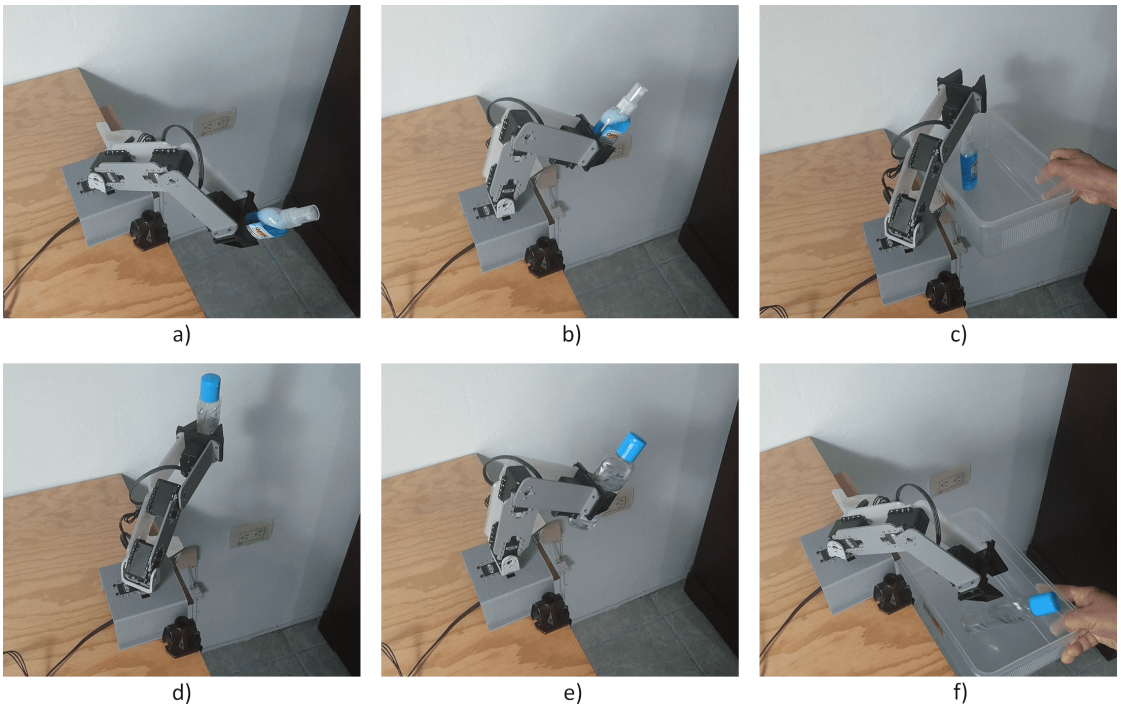


Figura 4.54: Segunda compilación del programa

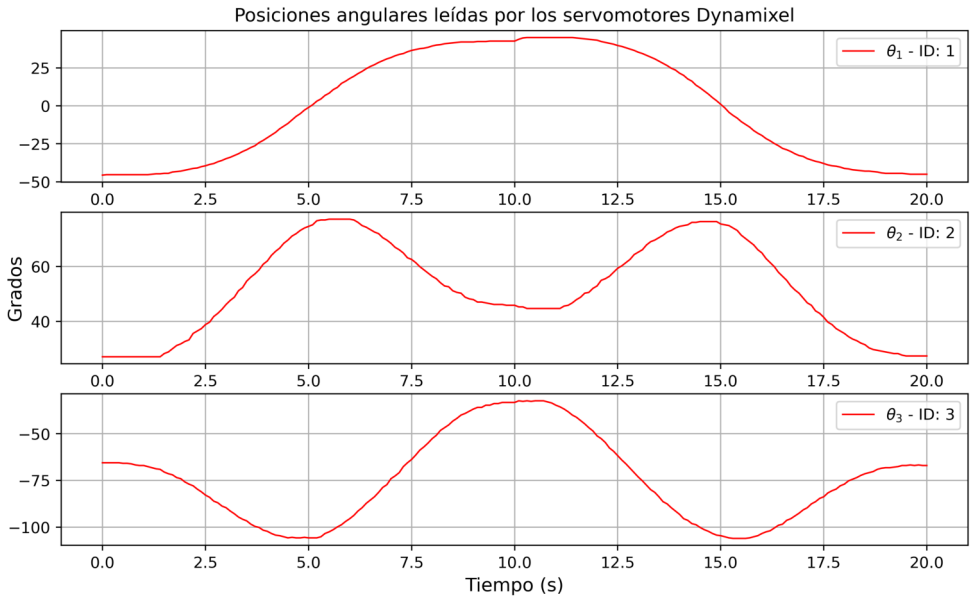


Figura 4.55: Posiciones angulares leídas durante la segunda compilación

De las gráficas que se muestran en la Figura 4.55 se observa que cada articulación sigue un perfil de trayectoria con movimientos angulares suaves desde el inicio hasta el final de la trayectoria. En las gráficas de la Figura 4.56 se observan las cargas que

fueron aplicadas en cada articulación del brazo robótico debido a la propia masa de sus eslabones y a la fuerza de gravedad.

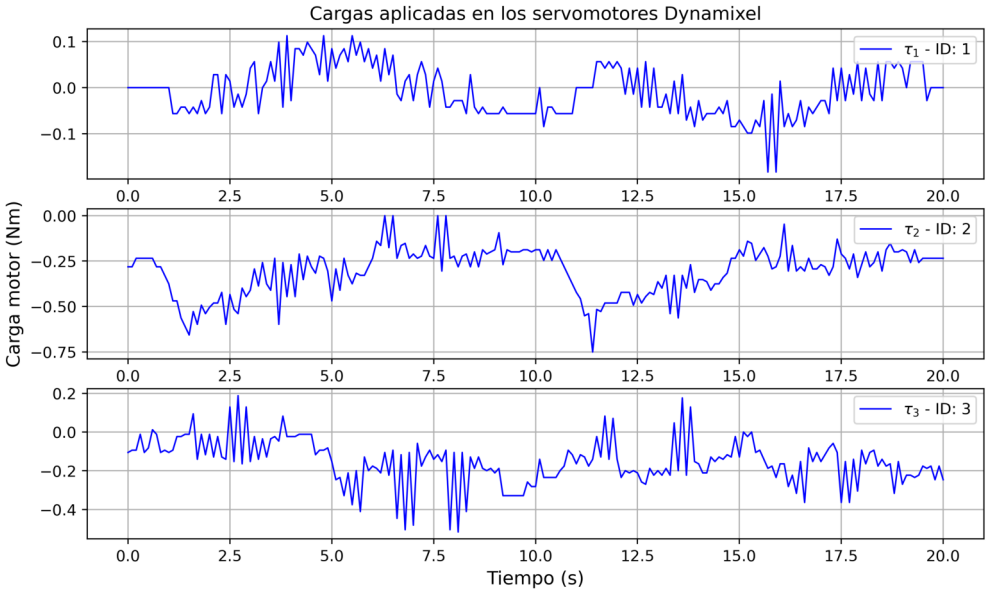


Figura 4.56: Cargas aplicadas durante la segunda compilación

En las gráficas de la Figura 4.57 se muestran los errores de posición angular que se presentaron durante la operación del robot, los cuales son aceptables.

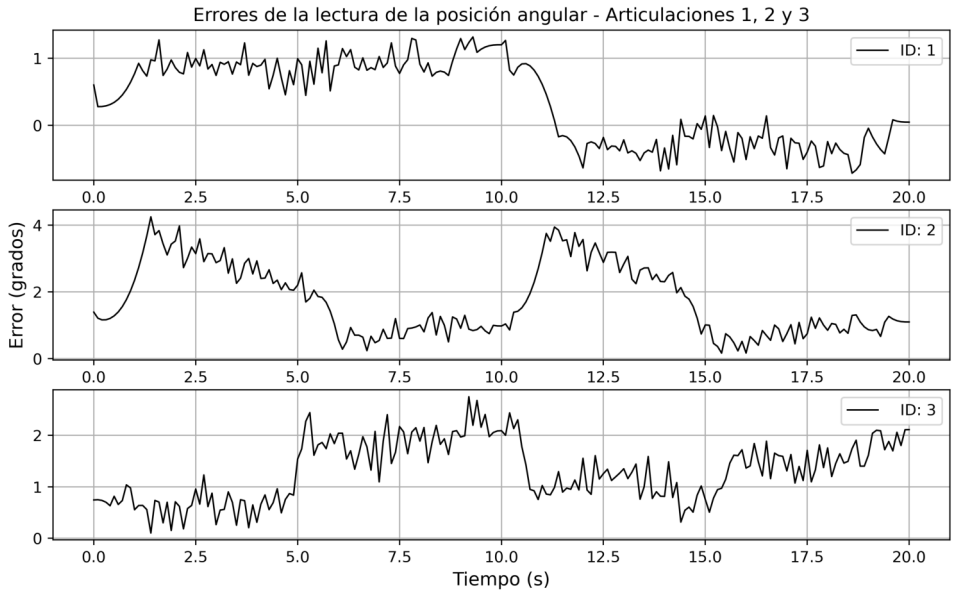


Figura 4.57: Errores de posición angular de la segunda compilación

Como prueba final, se volvió a compilar el código de programación en Python

para que el brazo robótico volviera a ejecutar su operación, pero en esta ocasión se aplicaron diferentes fuerzas externas en los eslabones del brazo robótico (en sentido contrario al movimiento), tal como se muestran en las Figuras 4.58, 4.59 y 4.60.

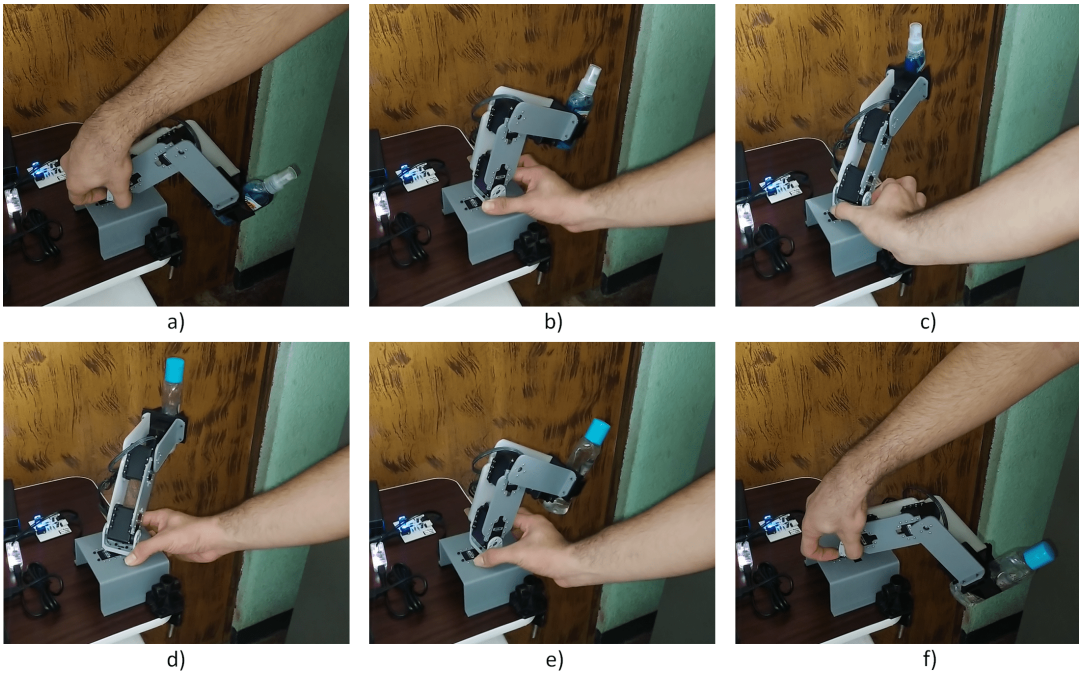


Figura 4.58: Aplicando fuerzas externas en la primera articulación

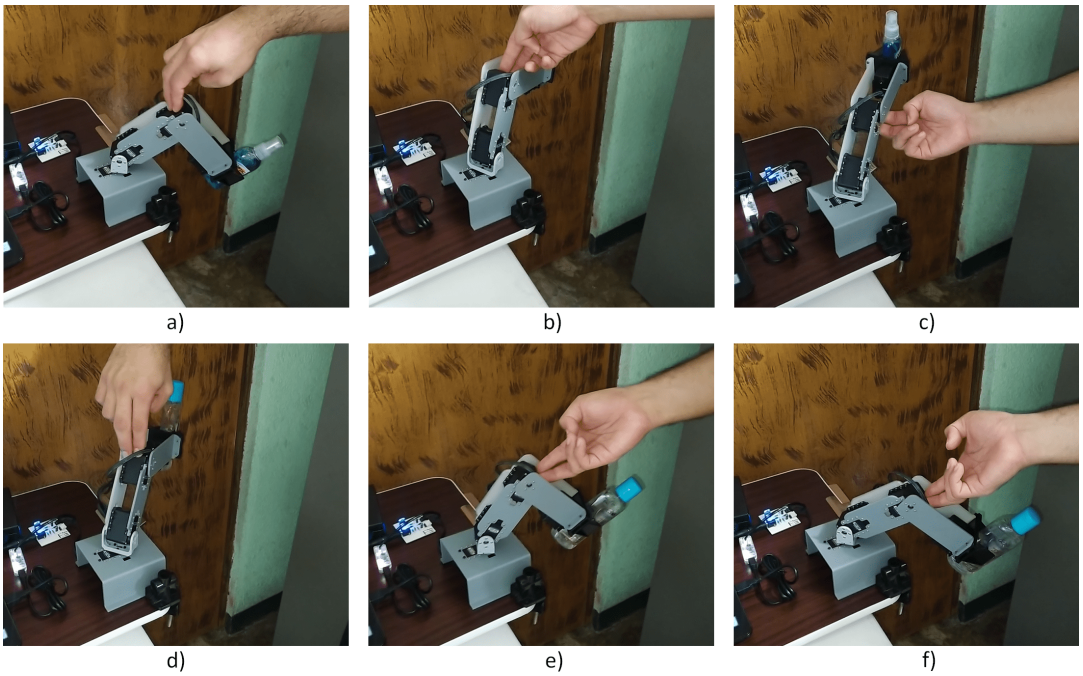


Figura 4.59: Aplicando fuerzas externas en la segunda articulación

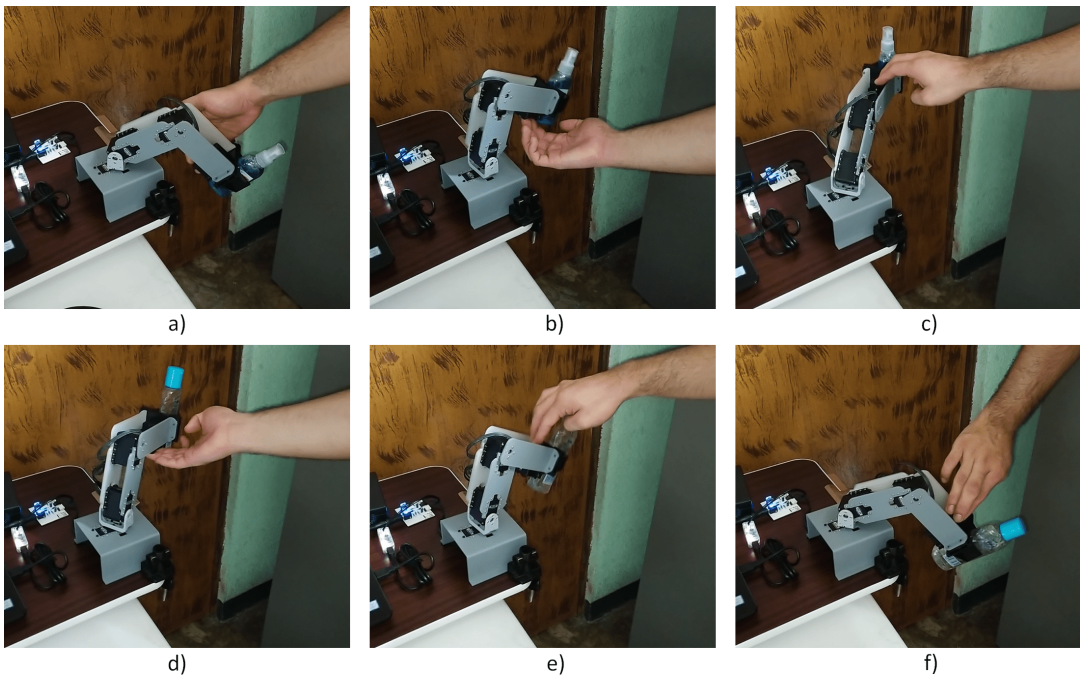


Figura 4.60: Aplicando fuerzas externas en la tercera articulación

En la Figura 4.61 se muestran las gráficas que permiten observar el comportamiento que presentaron las articulaciones del robot durante la aplicación de fuerzas externas.

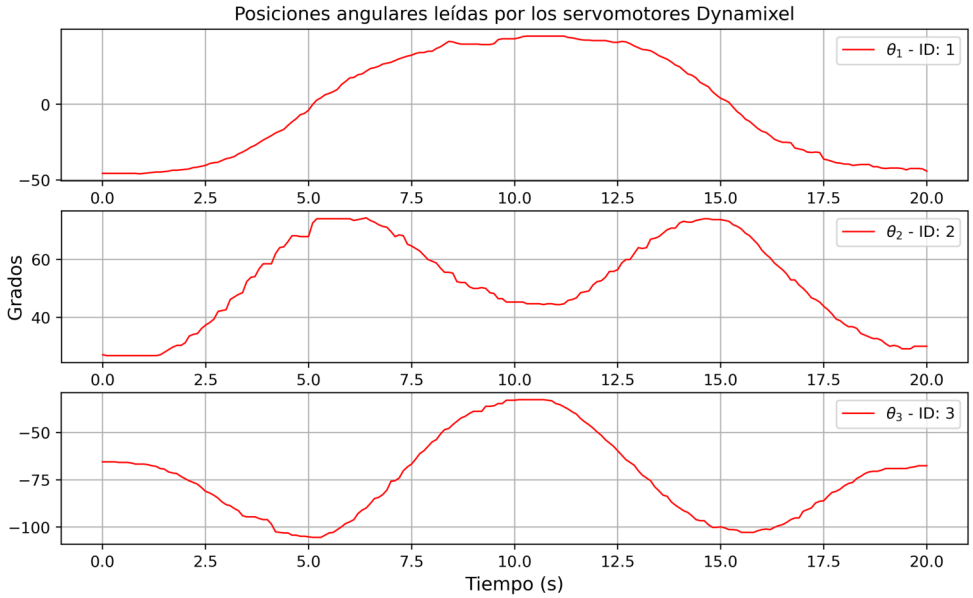


Figura 4.61: Posiciones angulares leídas durante la aplicación de fuerzas externas

En la Figura 4.62 se observan las cargas que fueron aplicadas en cada articulación del brazo robótico debidas a las fuerzas externas y a la propia masa de los eslabones.

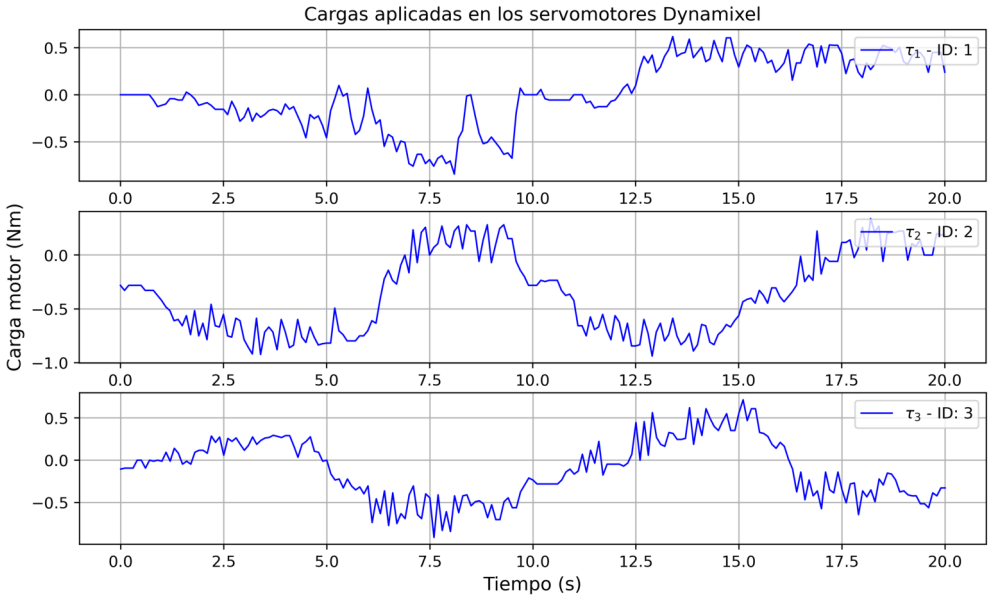


Figura 4.62: Cargas aplicadas durante la presencia de fuerzas externas

En las gráficas de la Figura 4.63 se muestran los errores de posición angular que se presentaron durante la aplicación de fuerzas externas.

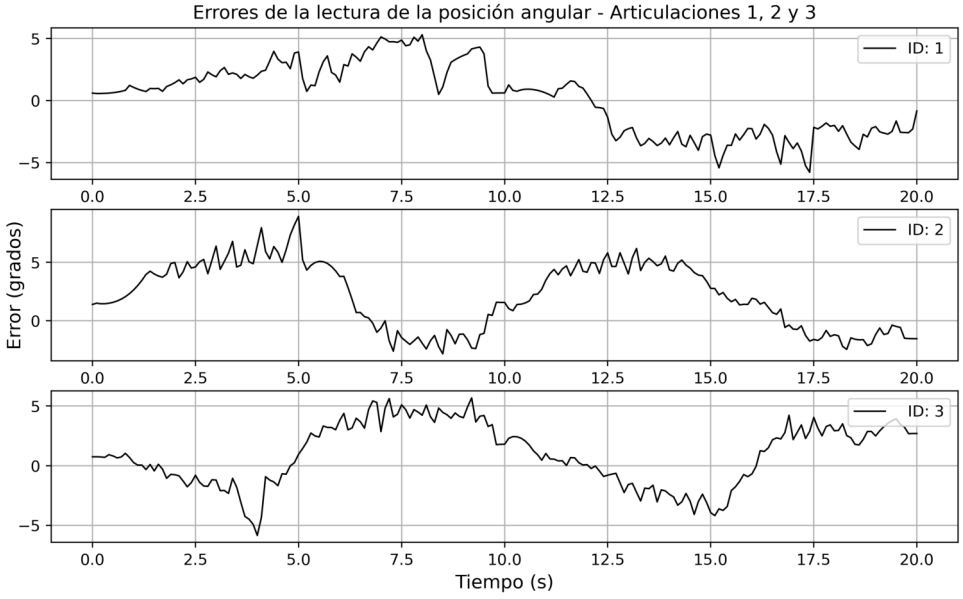


Figura 4.63: Errores de posición angular debido a la presencia de fuerzas externas

De las gráficas presentadas se observa que aun con la presencia de fuerzas externas, el brazo robótico trata de seguir el perfil de trayectoria sin llegar a desestabilizarse por completo, por lo tanto, el robot presenta una robustez adecuada.

4.5. Resumen del cuarto capítulo

En la primera sección se describe la estructura física del brazo robótico real y el diseño de las piezas mecánicas que permiten la construcción del robot. Además, se mencionan las características técnicas de los servomotores Dynamixel AX-12A y AX-18A, el dispositivo de comunicación U2D2, la fuente de alimentación y la construcción del regulador de voltaje.

En la segunda sección se describe detalladamente como se lleva a cabo la comunicación de los servomotores Dynamixel por medio de una computadora. Para la operación y control de los servomotores se debe instalar en la PC el software DYNAMIXEL SDK, el cual proporciona funciones de control mediante comunicación por paquetes digitales.

Una vez lograda la comunicación de los servomotores con la PC, se desarrolla un código de programación en Python que permita el movimiento sincronizado de los servomotores para ubicar al extremo final del brazo robótico en dos posiciones cartesianas previamente establecidas, con el objetivo de conocer a detalle algunas de las funciones de control que proporciona el software instalado. Posteriormente, se muestra la ejecución del código de programación y el error de precisión angular que presenta cada servomotor durante su funcionamiento.

En la tercera sección se muestra mediante simulaciones en SciLab, la implementación en lazo cerrado de los controladores por modo deslizante twisting y 3-AST con el modelo dinámico del brazo robótico, para que el efecto de las incertidumbres y de las perturbaciones sea reducido; permitiendo que el robot pueda moverse a la posición objetivo mediante el seguimiento de una trayectoria lineal previamente planificada (salidas de referencia) con el uso de las ecuaciones cinemáticas.

Ambos controladores permiten que las salidas del sistema robótico (posiciones angulares) sigan asintóticamente las salidas de referencia a lo largo de un tiempo establecido, aun con la presencia de las perturbaciones e incertidumbres del modelo; por lo que los controladores llevan al error de regulación a cero asintóticamente.

Finalmente, en la cuarta sección se muestra una prueba que consiste en escribir los valores obtenidos por el controlador 3-AST en cada servomotor del robot real.

Capítulo 5

Conclusiones

En el presente capítulo se mencionan las conclusiones generales que se obtuvieron con el desarrollo de este trabajo, las recomendaciones y las posibles mejoras que pueden implementarse como trabajo a futuro.

5.1. Conclusiones generales

Por una parte, se obtuvo la correcta comprobación de las ecuaciones cinemáticas al visualizar en SciLab que el extremo final del brazo robótico de 3-GDL, seguía adecuadamente la trayectoria lineal que le fue asignada para lograr ubicarse en el punto establecido dentro de su espacio de trabajo. Además, se demostró que el polinomio de quinto grado permite la generación de trayectorias con cambios de movimientos suaves en las articulaciones.

Con el desarrollo de la ecuación de Euler-Lagrange fue posible obtener el modelo dinámico del brazo robótico de 3-GDL, para posteriormente representarlo en el espacio de estados. Con dicho modelo, pudo demostrarse que el movimiento de los manipuladores robóticos se debe a los momentos de fuerzas que se generan en las articulaciones, aunque, también suelen estar afectados por las fricciones que suelen presentarse.

Para analizar la estabilidad del sistema robótico en lazo abierto, es necesario linealizar al modelo dinámico no lineal (que describe al sistema) alrededor de sus puntos de operación, obteniendo así a la matriz Jacobiana $A(x)$ del sistema, la cual puede transformarse en un polinomio característico para determinar sus valores propios y así

conocer el tipo de estabilidad que presenta el sistema de acuerdo al método indirecto de Lyapunov. Los puntos de operación para un brazo robótico quedan determinados exclusivamente por una entrada constante y un término de gravedad.

Con la simulación del modelo dinámico en lazo abierto se comprobó que la estabilidad del sistema solamente se cumplirá cuando el extremo final del brazo robótico opere en posiciones que tengan coordenada z negativa, por lo tanto, el espacio de trabajo para el brazo robótico de 3-GDL quedaría limitado. Este fue el problema de control que se requirió solucionar mediante la implementación de una ley de control en lazo cerrado.

Si se propone una ley de control robusta en lazo cerrado con el sistema robótico, entonces, las dinámicas del sistema serán llevadas a la estabilidad aun cuando se requiera ubicar al extremo final del brazo robótico en posiciones que tengan coordenada z positiva; además, se eliminarán las incertidumbres del modelo dinámico y se reducirán los efectos de las perturbaciones externas que pudieran presentarse durante la operación del robot. En este trabajo, se consideró el controlador lineal H_∞ y los controladores por modo deslizante twisting y super-twisting de tercer orden (3-AST).

Con la simulación del controlador lineal H_∞, se llegó a la conclusión de que su implementación en lazo cerrado solamente es viable para sistemas dinámicos que originalmente son lineales, ya que para un sistema dinámico linealizado (péndulo invertido) se demostró que el sistema en lazo cerrado se vuelve estable solamente para posiciones angulares que se encuentren cercanas al punto de operación que había sido considerado, sin embargo, en posiciones angulares alejadas del punto de operación, el sistema en lazo cerrado ya no cumple con el mismo comportamiento; por lo tanto, se tendrían que cambiar constantemente los valores de las regiones de estabilidad y esto no es conveniente para un brazo robótico que tiene que operar para diferentes posiciones angulares de manera instantánea.

Por otro lado, los algoritmos de control por modos deslizantes son controladores robustos ideales para ser aplicados en sistemas dinámicos no lineales, donde la variable deslizante σ se define en conjunto con el error de seguimiento de salida $e(t)$ dado por la diferencia entre la salida de referencia $y_r(t)$ y la salida medida $y(t)$ del sistema. Al conocer el grado relativo del sistema robótico, fue más sencillo determinar a la ley de

control por modo deslizante conveniente para llevar a cabo la regulación de salida.

Con la implementación simulada del controlador twisting al sistema robótico se obtuvo que las salidas medidas del sistema (posiciones angulares) convergen correctamente a las salidas de referencia (posiciones angulares dadas por la generación de trayectorias), pero las señales de control que compensan a las incertidumbres del modelo y a las perturbaciones externas aplicadas en cada articulación (servomotor) son discontinuas, esto se debe a que la ley de control twisting está definida por dos funciones signo que provocan oscilaciones de switcheo de alta frecuencia (chattering). Sin embargo, con la implementación del controlador 3-AST, el efecto chattering se atenúa mediante el uso de un filtro pasa bajas y la ley de control se vuelve continua.

Los controladores twisting y 3-AST, lograron llevar a las variables deslizantes σ y $\dot{\sigma}$ a cero en tiempo finito, manteniéndolas ahí para tiempos posteriores y permitiendo así la estabilidad del sistema robótico en tiempo finito aun con la presencia de las incertidumbres del modelo y las perturbaciones externas aplicadas.

De acuerdo a los resultados obtenidos por las simulaciones de los controladores twisiting y 3-AST, se concluye que el controlador twisitng no es una buena opción para implementar en el sistema robótico real, debido a que el efecto chattering que presenta cada señal de control puede dañar a los servomotores; por lo tanto, el controlador más viable a implementar para el sistema robótico real sería el 3-AST.

Al aplicar la técnica de control anti-windup al controlador 3-AST simulado, fue posible observar que el segundo servomotor es la articulación que más se verá afectada por la aplicación de perturbaciones externas, lo cual es lógico, ya que es la articulación a la que se le aplica la mayor cantidad de carga debido al peso que producen las masas de los eslabones que soporta.

Lo anteriormente mencionado puede comprobarse mediante las gráficas que muestran las cargas que fueron aplicadas en cada servomotor (sin considerar fuerzas externas) durante la operación del brazo robótico real. De acuerdo a dichas gráficas se corrobora que efectivamente, al segundo servomotor (ID: 2) se le aplica la mayor cantidad de carga aun cuando las perturbaciones externas son inexistentes. Para el caso del primer servomotor (ID: 1) se observa que las cargas aplicadas son insignificantes, ya que, como se encuentra ubicado en la base del brazo robótico y además siempre

se mantiene en reposo, entonces, las fuerzas gravitatorias no llegan a afectarlo.

En base a los resultados obtenidos por las simulaciones y por las pruebas que fueron realizadas con el brazo robótico real, se recomienda que el servomotor Dynamixel AX-18A sea propuesto como la segunda junta rotacional que permita la elevación de los eslabones y que el servomotor AX-12A sea propuesto como la primera junta rotacional que permita el giro del brazo robótico alrededor del eje vertical z. De esta manera el brazo robótico puede presentar una mejor robustez, ya que el servomotor AX-18A proporciona una mayor cantidad de par motor (1.8 Nm) que el servomotor AX-12A (1.5 Nm).

5.2. Trabajo a futuro

A continuación, se mencionan las posibles mejoras que pueden ser implementadas en el brazo robótico de 3-GDL como trabajo a futuro:

- Intercambiar de ubicación al servomotor AX-12A de la segunda junta rotacional con la ubicación del servomotor AX-18A de la primera junta rotacional, para obtener una mejor robustez en el sistema.
- Implementar en Python el algoritmo de control super-twisting de tercer orden anti-windup con el modelo dinámico del brazo robótico de 3-GDL en lazo cerrado, para analizar el funcionamiento de la estructura robótica en tiempo real ante la presencia de incertidumbres en el modelo y la aplicación de perturbaciones externas en las articulaciones.
- Reemplazar al dispositivo de comunicación U2D2 por un microcontrolador convencional, para que el brazo robótico pueda ejecutar sus trayectorias de movimiento sin necesidad de estar conectado a una computadora.
- Implementar sensores ultrasónicos para que el brazo robótico pueda conocer su espacio de trabajo y mediante un algoritmo de programación logré evadir los posibles obstáculos que puedan presentarse.

Apéndice A

Modelo dinámico en el espacio de estados

La ecuación dinámica en el espacio de estados para el manipulador robótico de tres grados de libertad estará representada por:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_3 \\ \dot{x}_5 \end{bmatrix} = \begin{bmatrix} x_2 \\ x_4 \\ x_6 \end{bmatrix}$$

$$\begin{bmatrix} \dot{x}_2 \\ \dot{x}_4 \\ \dot{x}_6 \end{bmatrix} = M(X)^{-1} \left[\mathcal{T}_{in} - C(X, \dot{X}) - F_b \dot{X} - G(X) + \mathcal{T}_d \right]$$

donde $M(X)$ es una matriz simetrica de 3x3 de masas inerciales, $C(X, \dot{X})$ es un vector de 3x1 de los términos de Coriolis y Centripetas, $F_b \dot{X}$ es un vector de 3x1 de los términos de fricción viscosa, $G(X)$ es un vector de 3x1 de la fuerza de gravedad, $\mathcal{T}_d$ es un vector de 3x1 que involucra a las incertidumbres y perturbaciones que pudiese presentar el sistema y $\mathcal{T}_{in}$ es un vector de 3x1 que involucra a los pares aplicados. Por lo tanto:

$$M(X) = \begin{bmatrix} Cc_3{}^2 + Ac_{35}{}^2 + 2Ec_3c_{35} & 0 & 0 \\ 0 & C + A + 2Ec_5 & A + Ec_5 \\ 0 & A + Ec_5 & A \end{bmatrix}$$

$$C(X,\dot{X}) = \begin{bmatrix} -2x_4D - 2B\left(x_4 + x_6\right) - 2x_2E\left(x_4 s_{235} + x_6 c_3 s_{35}\right) \\ \left(D + B\right)x_2 + E\left({x_2}^2 s_{235} - \left(2x_4x_6 + {x_6}^2\right)s_5\right) \\ Bx_2 + E\left({x_2}^2 c_3 s_{35} + {x_4}^2 s_5\right) \end{bmatrix}$$

$$F_b\dot{X} = \begin{bmatrix} 0.1772 & 0 & 0 \\ 0 & 0.2428 & 0 \\ 0 & 0 & 0.2428 \end{bmatrix} \begin{bmatrix} x_2 \\ x_4 \\ x_6 \end{bmatrix}$$

$$G(X) = \begin{bmatrix} 0 \\ \left(m_2 + m_3\right) g l_2 c_3 + m_3 g l_3 c_{35} \\ m_3 g l_3 c_{35} \end{bmatrix}$$

$$\mathcal{T}_d = \begin{bmatrix} \tau_{d1} \\ \tau_{d2} \\ \tau_{d3} \end{bmatrix}$$

$$\mathcal{T}_{in} = \begin{bmatrix} \tau_{in_1} \\ \tau_{in_2} \\ \tau_{in_3} \end{bmatrix}$$

además:

$A = m_3 {l_3}^2$
$B = m_3 {l_3}^2 x_2 \cos(x_3 + x_5)\,\mathrm{sen}(x_3 + x_5)$
$C = \left(m_2 + m_3\right){l_2}^2$
$D = \left(m_2 + m_3\right){l_2}^2 x_2 \cos(x_3)\,\mathrm{sen}(x_3)$
$E = m_3 l_2 l_3$
$F = m_3 g l_3 \cos(x_3 + x_5)$
$s_{235} = sen\left(2x_3 + x_5\right)$

y aclarando que:

$c_3 = \cos(x_3)$; $\quad c_5 = \cos(x_5)$; $\quad s_3 = \mathrm{sen}(x_3)$; $\quad s_5 = \mathrm{sen}(x_5)$;
$c_{35} = \cos(x_3 + x_5)$; $\quad s_{35} = \mathrm{sen}(x_3 + x_5)$; $\quad s_{235} = sen\left(2x_3 + x_5\right)$.

Apéndice B

Simulación de la cinemática

```
// UNIVERSIDAD DE GUANAJUATO - DIVISION DE INGENIERIAS CAMPUS IRAPUATO - SALAMANCA ///
////// MAESTRIA EN INGENIERIA ELECTRICA - INSTRUMENTACION Y SISTEMAS DIGITALES ////////
////////////////// COMPROBACION DE LA CINEMATICA DIRECTA E INVERSA //////////////////
////////////////// PROGRAMADO POR: CESAR EDUARDO CONEJO BENITEZ ////////////////////

clc;
clear;
xdel(winsid());

hB = 0.079;   // Base - Altura [m]
l1  = 0.113;   // Eslabón 1 - Longitud [m]
l2  = 0.140;   // Eslabón 2 - longitud [m]

x1 =   0.15;     // Posición cartesiana inicial en el eje X para la pinza
y1 = -0.15;     // Posición cartesiana inicial en el eje Y para la pinza
z1 =   0.05;     // Posición cartesiana inicial en el eje Z para la pinza

Pox1 = x1;
Poy1 = y1;
Poz1 = z1-hB;

//--------------------- Ecuaciones de la cinemática inversa ------------------//
w = sqrt(Pox1^2 + Poy1^2);
theta1 = atan(Poy1,Pox1)*(180/%pi);  //Primer ángulo

phi = atan(w,Poz1)*(180/%pi);
r = sqrt(Pox1^2 + Poy1^2 + Poz1^2);
beta = 90 - phi;

cg = ((r^2 + l1^2 – l2^2)/(2*l1*r));
sg = sqrt(1 - cg^2);
gama = atan(sg,cg)*(180/%pi);
theta2 = beta + gama;  //Segundo ángulo

ca = ((l1^2 + l2^2 - r^2)/(2*l1*l2));
sa = sqrt(1 - ca^2);
alfa = atan(sa,ca)*(180/%pi);
theta3 = -(180 - alfa);  //Tercer ángulo
//-------------------------------------------------------------------------------//
```

```
//----------------- Cinemática directa - vector de posición ----------------//
// Origen:
Px0 = 0;
Py0 = 0;
Pz0 = 0;

// Posición de la primera articulación:
Px1 = 0;
Py1 = 0;
Pz1 = hB;

// Posición de la segunda articulación:
Px2 = (l1*cosd(theta2))*cosd(theta1);
Py2 = l1*cosd(theta2)*sind(theta1);
Pz2 = l1*sind(theta2) + hB;

// Posición de la tercer articulación:
Px3 = (l1*cosd(theta2) + l2*(cosd(theta2 + theta3)))*cosd(theta1);
Py3 = (l1*cosd(theta2) + l2*(cosd(theta2 + theta3)))*sind(theta1);
Pz3 = l1*sind(theta2) + l2*sind(theta2 + theta3) + hB;

// Vector de articulaciones por coordenadas:
X=[Px0 Px1 Px2 Px3];
Y=[Py0 Py1 Py2 Py3];
Z=[Pz0 Pz1 Pz2 Pz3];
//--------------------------------------------------------------------------------//

scf(0);
clf(0);
param3d(X,Y,Z);
e=gce() //Se encarga de la lineal poligonal 3D (Dibujo de eslabones)
e.foreground=color('blue');
scatter3(X,Y,Z,'red',".");
set(gca(),"grid",[1 1 1]);
set(gca(),"data_bounds", matrix([-0.25,0.25,-0.25,0.25,0.0,0.25],2,-1));
title('Simulacion del brazo robotico de 3-GDL','FontSize',3)
   xlabel('eje X [m]','FontSize',3);
   ylabel('eje Y [m]','FontSize',3);
   zlabel('eje Z [m]','FontSize',3);
a=gca(); //Se encarga de agregar los ejes del plano 3D
a.rotation_angles=[70 235];
```

Apéndice C

Simulación de la trayectoria

```
// UNIVERSIDAD DE GUANAJUATO - DIVISION DE INGENIERIAS CAMPUS IRAPUATO - SALAMANCA ///
/////// MAESTRIA EN INGENIERIA ELECTRICA - INSTRUMENTACION Y SISTEMAS DIGITALES ///////
////////////////////////// TRAYECTORIA LINEAL DEL BRAZO ROBOTICO //////////////////////
//////////////////// PROGRAMADO POR: CESAR EDUARDO CONEJO BENITEZ ////////////////////

clc;
clear;
xdel(winsid());

hB = 0.079;  // Base - Altura [m]
l1 = 0.113;  // Eslabón 2 - Longitud [m]
l2 = 0.140;  // Eslabón 3 - longitud [m]

//--------------------- Asignando coordenadas cartesianas ---------------------//
X1 =  0.15;    // Primera posición cartesiana en el eje X para la pinza
Y1 = - 0.15;   // Primera posición cartesiana en el eje Y para la pinza
Z1 =   0.05;   // Primera posición cartesiana en el eje Z para la pinza

X2 = 0.15;     // Segunda posición cartesiana en el eje X para la pinza
Y2 = 0.15;     // Segunda posición cartesiana en el eje Y para la pinza
Z2 = 0.20;     // Segunda posición cartesiana en el eje Z para la pinza
//-----------------------------------------------------------------------------//

//---------------- Incluyendo polinomio de quinto grado -------------------//
qf = 1;
t0 = 0;       // Tiempo inicial
tf = 50;      // Tiempo final
Dt = 1e-1

t = t0:Dt:tf;
lt = length(t);

Pt = ((qf)*((10*(t.^3/tf^3)) - (15*(t.^4/tf^4)) + (6*(t.^5/tf^5))));

xx = ((X1) + ((X2-X1/qf)*(Pt)));
yy = ((Y1) + ((Y2-Y1/qf)*(Pt)));
zz = ((Z1) + ((Z2-Z1/qf)*(Pt)));
//-------------------------------------------------------------------//

theta1g = zeros(1,lt);   // Vectores para almacenar los valores de theta 1
```

```
theta2g = zeros(1,lt);   // Vectores para almacenar los valores de theta 2
theta3g = zeros(1,lt);   // Vectores para almacenar los valores de theta 3

// ---------------------- Siguiendo trayectoria -------------------------//
for i=1:length(t)
   Px = xx(i);
   Py = yy(i);
   Pz = zz(i)-hB;
//--------------------------- Cinemática inversa ----------------------//
   w = sqrt(Px^2 + Py^2);
   theta1 = atan(Py,Px)*(180/%pi); // Primer ángulo
   theta1g(i) = theta1;

   phi = atan(w,Pz)*(180/%pi);
   r = sqrt(Px^2 + Py^2 + Pz^2);
   beta = 90 - phi;

   cg = ((r^2 + l1^2 – l2^2)/(2*l1*r));
   sg = sqrt(1 - cg^2);
   gama = atan(sg,cg)*(180/%pi);
   theta2 = beta + gama;  // Segundo ángulo
   theta2g(i) = theta2;

   ca = ((l1^2 + l2^2 - r^2)/(2*l1*l2));
   sa = sqrt(1 - ca^2);
   alfa = atan(sa,ca)*(180/%pi);
   theta3 = -(180 - alfa);  // Tercer ángulo
   theta3g(i) = theta3;
//-------------------------------------------------------------------------//

//---------------------------- Cinemática inversa ----------------------//
   // Origen:
   Px0 = 0;
   Py0 = 0;
   Pz0 = 0;

   // Posición de la primera articulación:
   Px1 = 0;
   Py1 = 0;
   Pz1 = hB;

   // Posición de la segunda articulación:
   Px2 = (l1*cosd(theta2))*cosd(theta1);
   Py2 = l1*cosd(theta2)*sind(theta1);
   Pz2 = l1*sind(theta2)+hB;

   // Posición de la tercer articulación:
   Px3 = (l1*cosd(theta2) + l2*(cosd(theta2 + theta3)))*cosd(theta1);
   Py3 = (l1*cosd(theta2) + l2*(cosd(theta2 + theta3)))*sind(theta1);
   Pz3 = l1*sind(theta2) + l2*sind(theta2 + theta3)+hB;

   // Vector de articulaciones por coordenadas:
   X=[Px0 Px1 Px2 Px3];
   Y=[Py0 Py1 Py2 Py3];
   Z=[Pz0 Pz1 Pz2 Pz3];
```

```
//-------------------------------------------------------------------//

  // Puntos intermedios de la trayectoria
  vectorx(i)=xx(i)
  vectory(i)=yy(i)
  vectorz(i)=zz(i)

  scf(0);
  clf(0);
  drawlater();
  param3d(X,Y,Z);
  e=gce() //Se encarga de dibujar la lineal poligonal 3D (Dibujo de eslabones)
  e.foreground=color('blue');
  scatter3(X,Y,Z,'red',".");
  param3d(vectorx,vectory,vectorz);
  set(gca(),"grid",[1 1 1]);
  set(gca(),"data_bounds", matrix([-0.3,0.3,-0.3,0.3,0.0,0.3],2,-1));
  xtitle("Simulación de trayectoria del brazo robótico de 3GDL", "eje X [m]", "eje Y [m]", "eje Z [m]")
  a=gca(); //Se encarga de agregar los ejes del plano 3D
  a.rotation_angles=[70 235];
  drawnow();

  sleep(0.03,"s")
end
// -------------------------------------------------------------------//

scf(1); clf();
plot(t,theta1g,'-blue');
title('Movimiento de la 1er. articulación (θ1)','FontSize',3);
xlabel('Tiempo [s]','FontSize',3);
ylabel('Grados','FontSize',3);
xgrid;

scf(2); clf();
plot(t,theta2g,'-blue');
title('Movimiento de la 2da. articulación (θ2)','FontSize',3);
xlabel('Tiempo [s]','FontSize',3);
ylabel('Grados','FontSize',3);
xgrid;

scf(3); clf();
plot(t,theta3g,'-blue');
title('Movimiento de la 3er. articulación (θ3)','FontSize',3);
xlabel('Tiempo [s]','FontSize',3);
ylabel('Grados','FontSize',3);
xgrid;
```

Apéndice D

Linealización del modelo

```
// UNIVERSIDAD DE GUANAJUATO - DIVISION DE INGENIERIAS CAMPUS IRAPUATO - SALAMANCA ///
////// MAESTRIA EN INGENIERIA ELECTRICA - INSTRUMENTACION Y SISTEMAS DIGITALES ///////
////////////////////////// ANALISIS DE ESTABILIDAD DEL SISTEMA ////////////////////////////
////////////////////// PROGRAMADO POR: CESAR EDUARDO CONEJO BENITEZ //////////////////////

clc;
clear;
xdel(winsid());

// Parámetros
g    = 9.81;          // Gravedad [m/s^2]
mB = 0.146;        // Masa de la base [kg]
m1 = 0.147;        // Masa del eslabón 1 [kg]
m2 = 0.116;        // Masa del eslabón 2 [kg]
hB  = 0.079;       // Altura de la base [m]
l1   = 0.113;      // Longitud del eslabón 1 [m]
l2   = 0.140;      // Longitud del eslabón 2 [m]
v1  = 0.1772;     // Fricción viscosa 1 [N*m/(rad/s)]
v2  = 0.2428;     // Fricción viscosa 2 [N*m/(rad/s)]
v3  = 0.2428;     // Fricción viscosa 3 [N*m/(rad/s)]

//------------------------------ Cinemática Inversa ------------------------------//
// Posición requerida para el efector final:
XX = 0.2;      // Posición cartesiana requerida en el eje X
YY = 0.0;      // Posición cartesiana requerida en el eje Y
ZZ = 0.1;      // Posición cartesiana requerida en el eje Z

Px = XX;
Py = YY;
Pz = ZZ-hB;

w = sqrt(Px^2 + Py^2);
theta1 = atan(Py,Px)*(180/%pi);   // Primer ángulo

phi = atan(w,Pz)*(180/%pi);
r = sqrt(Px^2 + Py^2 + Pz^2);
betta = 90 - phi;

cg = ((r^2 + l1^2 – l2^2)/(2*l1*r));
sg = sqrt(1 - cg^2);
```

```
gama = atan(sg,cg)*(180/%pi);
theta2 = betta + gama;  // Segundo ángulo

ca = ((l1^2 + l2^2 - r^2)/(2*l1*l2));
sa = sqrt(1 - ca^2);
alfa = atan(sa,ca)*(180/%pi);
theta3 = -(180 - alfa);  // Tercer ángulo
//-------------------------------------------------------------------------------//

// Definición de momentos de fuerza:
T1 = 0;
T3 = m2*g*l2*cosd(theta2 + theta3);
T2 = (m1 + m2)*g*l1*cosd(theta2) + T3;

//---------------------------------- Modelo dinámico no lineal ----------------------------------//
function y=f(x)
  A = m2*(l2^2);
  B = m2*(l2^2)*x(2)*cosd(x(3)+x(5))*sind(x(3)+x(5));
  C = (m1 + m2)*(l1^2);
  D = (m1 + m2)*(l1^2)*x(2)*cosd(x(3))*sind(x(3));
  E = m2*l1*l2;
  F = m2*g*l2*cosd(x(3)+x(5));
  s223 = sind((2*x(3)) + x(5));

  p = 3;
  q = 1;

  //--------------------- MATRIZ M (Masas inerciales) ----------------------//
  M = zeros(p);
  M(1,1) = C*(cosd(x(3))^2) + A*(cosd(x(3)+x(5))^2) + 2*E*cosd(x(3))*cosd(x(3)+x(5));
  M(1,2) = 0;
  M(1,3) = 0;
  M(2,1) = 0;
  M(2,2) = C + A + 2*E*cosd(x(5));
  M(2,3) = A + E*cosd(x(5));
  M(3,1) = 0;
  M(3,2) = A + E*cosd(x(5));
  M(3,3) = A;

  //------------ VECTOR V (Términos de Coriolis and Centrípetas) ------------//
  V = zeros(p,q);
  V(1) = -2*x(4)*D - 2*B*(x(4)+x(6)) - 2*x(2)*E*(x(4)*s223 + x(6)*cosd(x(3))*sind(x(3)+x(5)));
  V(2) = (D+B)*x(2) + E*((x(2)^2)*s223 - (2*x(4)*x(6) + (x(6)^2))*sind(x(5)));
  V(3) = B*x(2) + (E*((x(2)^2)*cosd(x(3))*sind(x(3)+x(5))+(x(4)^2)*sind(x(5))));

  //-------------------- VECTOR F (Fuerzas de fricción) ---------------------//
  FV = zeros(p,q);
  FV(1) = v1*x(2);
  FV(2) = v2*x(4);
  FV(3) = v3*x(6);

  //------------------------- VECTOR G (Gravedad)--------------------------//
  G = zeros(p,q);
  G(1) = 0;
  G(2) = (m1+m2)*g*l1*cosd(x(3)) + m2*g*l2*cosd(x(3)+x(5));
```

```
  G(3) = m2*g*l2*cosd(x(3)+x(5));

  //---------------------- VECTOR T (Pares de fuerza) ------------------------//
  T = [T1; T2; T3];

  //--------------------- MATRIZ DE ACELERACION ANGULAR ---------------------//
  AA = inv(M)*(T - V - FV - G);

  //------------------- MATRIZ EN EL ESPACIO DE ESTADOS ---------------------//
  f1 = x(2);     // Velocidad angular de la masa 1
  f2 = AA(1);    // Aceleración angular de la masa 1
  f3 = x(4);     // Velocidad angular de la masa 2
  f4 = AA(2);    // Aceleración angular de la masa 2
  f5 = x(6);     // Velocidad angular de la masa 3
  f6 = AA(3);    // Aceleración angular de la masa 3
  y = [f1; f2; f3; f4; f5; f6]
endfunction
//-------------------------------------------------------------------------------------------------------------//

// Matriz Jacobiana con los puntos de equilibrio seleccionados:
x1 = 0;
x2 = 0;
x3 = -acosd((T2-T3)/((m1+m2)*g*l1));
x4 = 0;
x5 = -acosd((T3)/(m2*g*l2))-x3;
x6 = 0;
x = [x1; x2; x3; x4; x5; x6];
J = numderivative(f, x)
disp("La matriz Jacobiana A es:")
disp(J)

// Obtención de valores propios:
EV = spec(J);
disp("Los valores propios son:")
disp(EV)

if EV <= 0 then
  disp("El sistema puede ser estable con los puntos de equilibrio seleccionados!")
  disp("Las posiciones obtenidas, donde el sistema puede ser estable son:")
  //------------------------ Posiciones estables -------------------------//
  Pax = cosd(x1)*(l1*cosd(x3) + l2*cosd(x3+x5));
  disp("Px:")
  disp(Pax)
  Pay = sind(x1)*(l1*cosd(x3) + l2*cosd(x3+x5));
  disp("Py:")
  disp(Pay)
  Paz = l1*sind(x3) + l2*sind(x3 + x5) + hB;
  disp("Pz:")
  disp(Paz)
else
  disp("El sistema es inestable con los puntos de equilibrio seleccionados")
end
```

Apéndice E

Estabilidad en lazo abierto

```
// UNIVERSIDAD DE GUANAJUATO - DIVISION DE INGENIERIAS CAMPUS IRAPUATO - SALAMANCA ///
////// MAESTRIA EN INGENIERIA ELECTRICA - INSTRUMENTACION Y SISTEMAS DIGITALES ////////
/////////////////////////// SIMULACION DEL MODELO NO LINEAL ///////////////////////////
////////////////// PROGRAMADO POR: CESAR EDUARDO CONEJO BENITEZ //////////////////////

clc;
clear;
xdel(winsid());

//-------------------------------------- Sistema dinámico no lineal ---------------------------------------//
function dx=RobotSys2ODE(t, x, g, mB, m1, m2, hB, l1, l2, v1, v2, v3, T1, T2, T3)
  A = m2*(l2^2);
  B = m2*(l2^2)*x(2)*cos(x(3)+x(5))*sin(x(3)+x(5));
  C = (m1 + m2)*(l1^2);
  D = (m1 + m2)*(l1^2)*x(2)*cos(x(3))*sin(x(3));
  E = m2*l1*l2;
  F = m2*g*l2*cos(x(3)+x(5));
  s223 = sin((2*x(3)) + x(5));

  p = 3;
  q = 1;

  //---------------------- MATRIZ M (Masas inerciales) ------------------------//
  M = zeros(p);
  M(1,1) = C*(cos(x(3))^2) + A*(cos(x(3)+x(5))^2) + 2*E*cos(x(3))*cos(x(3)+x(5));
  M(1,2) = 0;
  M(1,3) = 0;
  M(2,1) = 0;
  M(2,2) = C + A + 2*E*cos(x(5));
  M(2,3) = A + E*cos(x(5));
  M(3,1) = 0;
  M(3,2) = A + E*cos(x(5));
  M(3,3) = A;

  //------------ VECTOR V (Términos de Coriolis y Centrípetas) ------------//
  V = zeros(p,q);
  V(1) = -2*x(4)*D - 2*B*(x(4)+x(6)) - 2*x(2)*E*(x(4)*s223 + x(6)*cos(x(3))*sin(x(3)+x(5)));
  V(2) = (D+B)*x(2) + E*((x(2)^2)*s223 - (2*x(4)*x(6) + (x(6)^2))*sin(x(5)));
  V(3) = B*x(2) + (E*((x(2)^2)*cos(x(3))*sin(x(3)+x(5))+(x(4)^2)*sin(x(5))));
```

```
  //-------------------- VECTOR F (Fuerzas de fricción) ---------------------//
  FV = zeros(p,q);
  FV(1) = v1*x(2);
  FV(2) = v2*x(4);
  FV(3) = v3*x(6);

  //------------------------- VECTOR G (Gravedad)-----------------------------//
  G = zeros(p,q);
  G(1) = 0;
  G(2) = (m1+m2)*g*l1*cos(x(3)) + m2*g*l2*cos(x(3)+x(5));
  G(3) = m2*g*l2*cos(x(3)+x(5));

  //--------------------- VECTOR T (Pares de fuerza) ------------------------//
  T = [T1; T2; T3];

  //--------------- MATRIZ DE ACELERACION ANGULAR ----------------//
  AA = inv(M)*(T - V - FV - G);

  //-------------- MODELO EN EL ESPACIO DE ESTADOS ---------------//
  dx(1) = x(2);      // Posición angular para del servomotor 1
  dx(2) = AA(1);   // Velocidad angular del servomotor 1
  dx(3) = x(4);      // Posición angular para del servomotor 2
  dx(4) = AA(2);   // Velocidad angular del servomotor 2
  dx(5) = x(6);     // Posición angular para del servomotor  3
  dx(6) = AA(3);   // Velocidad angular del servomotor 3
endfunction
//-------------------------------------------------------------------------------------------------------------------//

// Parámetros:
g = 9.81;          // Gravedad [m/s^2]
mB = 0.146;       // Masa de la base [kg]
m1 = 0.147;       // Masa del eslabón 1 [kg]
m2 = 0.116;       // Masa del eslabón 2 [kg]
hB = 0.079;       // Altura de la base [m]
l1 = 0.113;       // Longitud del eslabón 1 [m]
l2 = 0.140;       // Longitud del eslabón 2 [m]
v1 = 0.1772;     // Fricción viscosa 1 [N*m/(rad/s)]
v2 = 0.2428;     // Fricción viscosa 2 [N*m/(rad/s)]
v3 = 0.2428;     // Fricción viscosa 3 [N*m/(rad/s)]

// Posición cartesiana inicial para el extremo final del robot:
disp("POSICIONES CARTESIANAS INICIALES:")
disp("Pxi:")
disp("0.253 m")   // Posición inicial en el eje X
disp("Pyi:")
disp("0.0 m")      // Posición inicial en el eje Y
disp("Pzi:")
disp("0.079 m")  // Posición inicial en el eje Z

// Posición cartesiana requerida para el extremo final del robot:
Pxr = 0.2;    // Posición requerida en el eje X
Pyr = 0.0;    // Posición requerida en el eje Y
Pzr = 0.1;    // Posición requerida en el eje Z

disp("POSICIONES CARTESIANAS REQUERIDAS:")
```

```
disp("Pxr:")
disp(Pxr)
disp("Pyr:")
disp(Pyr)
disp("Pzr:")
disp(Pzr)

// Definición del tiempo
t0 = 0;        // segundos
tf = 40;       // segundos
Dt = 1e-1;  // segundos

t = t0:Dt:tf;
lt = length(t);

//--------------------------- Cinemática Inversa ----------------------------//
w = sqrt(Pxr^2 + Pyr^2);
theta1 = atan(Pyr,Pxr)*(180/%pi);  // Primer ángulo
theta1 = 0;

phi = atan(w,(Pzr-hB))*(180/%pi);
r = sqrt(Pxr^2 + Pyr^2 + (Pzr-hB)^2);
betta = 90 - phi;

cg = ((r^2 + l1^2 – l2^2)/(2*l1*r));
sg = sqrt(1 - cg^2);
gama = atan(sg,cg)*(180/%pi);
theta2 = betta + gama;  // Segundo ángulo

ca = ((l1^2 + l2^2 - r^2)/(2*l1*l2));
sa = sqrt(1 - ca^2);
alfa = atan(sa,ca)*(180/%pi);
theta3 = -(180 - alfa);  // Tercer ángulo
//-------------------------------------------------------------------------//

// Definición de momentos de fuerza:
TO1 = 0;
TO3 = m2*g*l2*cos(theta2*(%pi/180) + theta3*(%pi/180));
TO2 = (m1 + m2)*g*l1*cos(theta2*(%pi/180)) + TO3;

disp("PARES DE FUERZA NECESARIOS PARA EL MOVIMIENTO:")
disp("tau 1:")
disp(TO1)
disp("tau 2:")
disp(TO2)
disp("tau 3:")
disp(TO3)

T1 = TO1*ones(1,lt);   // Par del servomotor 1 [N*m]
T2 = TO2*ones(1,lt);   // Par del servomotor 2 [N*m]
T3 = TO3*ones(1,lt);   // Par del servomotor 3 [N*m]

// Posiciones y velocidades iniciales:
x10 = zeros(1,lt); // Posición angular 1 inicial - theta1(0)
x20 = zeros(1,lt); // Velocidad angular 1 inicial
```

```
x30 = zeros(1,lt); // Posición angular 2 inicial - theta2(0)
x40 = zeros(1,lt); // Velocidad angular 2 inicial
x50 = zeros(1,lt); // Posición angular 3 inicial - theta3(0)
x60 = zeros(1,lt); // Velocidad angular 3 inicial

disp("Por favor espere....")

// Calculo de la posición y velocidad angular:
for k=1:lt-1
  xOut = ode("stiff",[x10(k),x20(k),x30(k),x40(k),x50(k),x60(k)],t(k),t(k+1),[1e-5, 1e-6, 1e-7, 1e-8, 1e-
9, 1e-10],list(RobotSys2ODE,g,mB,m1,m2,hB,l1,l2,v1,v2,v3,T1(k),T2(k),T3(k)));
  x10(k+1) = xOut(1);
  x20(k+1) = xOut(2);
  x30(k+1) = xOut(3);
  x40(k+1) = xOut(4);
  x50(k+1) = xOut(5);
  x60(k+1) = xOut(6);
end

///------------------------- Cinemática Directa ----------------------------///
for k=1:lt-1
  // Punto de referencia:
  Px0 = 0;
  Py0 = 0;
  Pz0 = 0;

  // Posición de la primera articulación:
  Px1 = 0;
  Py1 = 0;
  Pz1 = hB;

  // Posición de la segunda articulación:
  Px2 = (l1*cos(x30(k)))*cos(x10(k));
  Py2 = l1*cos(x30(k))*sin(x10(k));
  Pz2 = l1*sin(x30(k))+hB;

  // Posición de la tercera articulación:
  Px3 = (l1*cos(x30(k)) + l2*(cos(x30(k) + x50(k))))*cos(x10(k));
  Py3 = (l1*cos(x30(k)) + l2*(cos(x30(k) + x50(k))))*sin(x10(k));
  Pz3 = l1*sin(x30(k)) + l2*sin(x30(k) + x50(k))+hB;

  // Vector de posiciones:
  X=[Px0 Px1 Px2 Px3];
  Y=[Py0 Py1 Py2 Py3];
  Z=[Pz0 Pz1 Pz2 Pz3];

  scf(0); clf();
  drawlater();
  param3d(X,Y,Z);
  e=gce() //Se encarga de dibujar la línea poligonal 3D (Dibujo de eslabones)
  e.foreground=color('blue');
  scatter3(X,Y,Z,'red',".");
  param3d(Px3,Py3,Pz3);
  set(gca(),"grid",[1 1 1]);
  set(gca(),"data_bounds", matrix([-0.25,0.25,-0.25,0.25,-0.10,0.25],2,-1));
```

```
  xtitle("Simulación de estabilidad del brazo robótico de 3GDL", "eje X [m]", "eje Y [m]", "eje Z [m]")
  a=gca(); //Se encarga de agregar los ejes del plano 3D
  a.rotation_angles=[70 235];
  drawnow();

  sleep(0.15,"s")
end
///--------------------------------------------------------------------///

disp("Proceso finalizado!   :D")

// Posición estable para el brazo robótico:
disp("POSICION CARTESIANA ESTABLE OBTENIDAD:")
disp("Pxe:")
disp(Px3)
disp("Pye:")
disp(Py3)
disp("Pze:")
disp(Pz3)

// Graficas para cada posición angular:
scf(1); clf();
plot(t,x10*(180/%pi),'-b');
title('Posición angular estable - Primer articulacion','FontSize',3);
xlabel('Tiempo [s]','FontSize',3);
ylabel('Grados','FontSize',3);
xgrid;

scf(2); clf();
plot(t,x30*(180/%pi),'-b');
title('Posición angular estable - Segunda articulacion','FontSize',3);
xlabel('Tiempo [s]','FontSize',3);
ylabel('Grados','FontSize',3);
xgrid;

scf(3); clf();
plot(t,x50*(180/%pi),'-b');
title('Posición angular estable - Tercer articulacion','FontSize',3);
xlabel('Tiempo [s]','FontSize',3);
ylabel('Grados','FontSize',3);
xgrid;
```

Apéndice F

Simulación del controlador lineal H_∞

```
// UNIVERSIDAD DE GUANAJUATO - DIVISION DE INGENIERIAS CAMPUS IRAPUATO - SALAMANCA ///
////// MAESTRIA EN INGENIERIA ELECTRICA - INSTRUMENTACION Y SISTEMAS DIGITALES ////////
/////// SIMULACION DE CONTROLADOR LINEAL H-INFINITO EN MANIPULADOR DE UN EJE ////////
////////////////// PROGRAMADO POR: CESAR EDUARDO CONEJO BENITEZ ///////////////////

clc;
clear;
xdel(winsid());

// Parámetros:
m = 0.147;          // Masa del eslabón [kg]
mo = 0.260;         // Masa de un objeto trasladado [kg]
L = 0.115;          // Longitud del eslabón [m]

x1p = (%pi/3); // Punto de operación para x1
x2p = 0;           // Punto de operación para x2
Tinp = (9.81*(m+mo)*L)*cos(x1p);  // Par de fuerza evaluado alrededor del punto de operación

// Politopo convexo:
p1me = 9.78;            // Mínimo valor para la gravedad [m/s^2]
p1ma = 9.81;            // Máximo valor para la gravedad [m/s^2]
p2me = 0.2185;          // Mínimo valor para la fricción viscosa [Nm/(rad/s)]
p2ma = 0.2671;          // Máximo valor para la fricción viscosa [Nm/(rad/s)]

// Posición x1 obtenida con presencia de la perturbación:
x1wmin = 0.9857*%pi;      // Posición del manipulador debida a la mínima perturbación [Nm]
x1wmax = -1.4468*%pi;     // Posición del manipulador debida a la máxima perturbación [Nm]

// Evaluar matrices con valores del politopo:
A = [0, 1; ((m+mo)*9.81*L*sin(x1p))/((m+mo)*(L^2)), -(0.2428)/((m+mo)*(L^2))]; //Matriz
linealizada A
A1 = [0, 1; ((m+mo)*p1me*L*sin(x1p))/((m+mo)*(L^2)), -(p2me)/((m+mo)*(L^2))]; //Vértice 1
A2 = [0, 1; ((m+mo)*p1ma*L*sin(x1p))/((m+mo)*(L^2)), -(p2me)/((m+mo)*(L^2))]; //Vértice 2
A3 = [0, 1; ((m+mo)*p1ma*L*sin(x1p))/((m+mo)*(L^2)), -(p2ma)/((m+mo)*(L^2))]; //Vértice 3
A4 = [0, 1; ((m+mo)*p1me*L*sin(x1p))/((m+mo)*(L^2)), -(p2ma)/((m+mo)*(L^2))]; //Vértice 4
Awmin = [0, 1; ((m+mo)*9.81*L*sin(x1wmin))/((m+mo)*(L^2)), -(0.2428)/((m+mo)*(L^2))]; //Matriz
linealizada A con perturbación mínima
Awmax = [0, 1; ((m+mo)*9.81*L*sin(x1wmax))/((m+mo)*(L^2)), -(0.2428)/((m+mo)*(L^2))];
//Matriz linealizada A con perturbación máxima
```

```
Bu = [0; (1)/((m+mo)*(L^2))]; // Matriz Bu
Bw = -[0; (1)/((m+mo)*(L^2))]; // Matriz Bw
C = [1, 0]; // Matriz C (Salida)
D = zeros(1,1);

// Regiones de estabilidad:
alpha = 36;
radio = 66;
theta = %pi/4;

// Resolvedor de LMI's:
function [LME, LMI, OBJ]=h2hinf_eval(XLIST)
  [w1,w2,gamma]=XLIST(:)
  LME=w1-w1'
  L1 = -(A1*w1)-(Bu*w2)-(w1*A1')-(w2'*Bu')  // Retroalimentación de estados
  // Regiones de estabilidad:
  E2_1 = -(A1*w1)-(Bu*w2)-(w1*A1')-(w2'*Bu')-(2*alpha*w1)
  E3_1 = [(radio*w1), (-(A1*w1)-(Bu*w2)); (-(w1*A1')-(w2'*Bu')), (radio*w1)]
  E4_1 = [-(sin(theta)*((A1*w1)+(Bu*w2)+(w1*A1')+(w2'*Bu'))), -(cos(theta)*((A1*w1)+(Bu*w2)-
(w1*A1')-(w2'*Bu'))); -(cos(theta)*((w1*A1')+(w2'*Bu')-(A1*w1)-(Bu*w2))), -
(sin(theta)*((A1*w1)+(Bu*w2)+(w1*A1')+(w2'*Bu')))]
  O1 = [(-(A1*w1)-(w1*A1')-(Bu*w2)-(w2'*Bu')), -(Bw), (-(w1*C')-(w2'*D')); -(Bw'),
(gamma*eye(1,1)), -(D'); (-(C*w1)-(D*w2)), -(D), (gamma*eye(1,1))]  // H-infinito
  ///
  L2 = -(A2*w1)-(Bu*w2)-(w1*A2')-(w2'*Bu')  // Retroalimentación de estados
  // Regiones de estabilidad:
  E2_2 = -(A2*w1)-(Bu*w2)-(w1*A2')-(w2'*Bu')-(2*alpha*w1)
  E3_2 = [(radio*w1), (-(A2*w1)-(Bu*w2)); (-(w1*A2')-(w2'*Bu')), (radio*w1)]
  E4_2 = [-(sin(theta)*((A2*w1)+(Bu*w2)+(w1*A2')+(w2'*Bu'))), -(cos(theta)*((A2*w1)+(Bu*w2)-
(w1*A2')-(w2'*Bu'))); -(cos(theta)*((w1*A2')+(w2'*Bu')-(A2*w1)-(Bu*w2))), -
(sin(theta)*((A2*w1)+(Bu*w2)+(w1*A2')+(w2'*Bu')))]
  O2 = [(-(A2*w1)-(w1*A2')-(Bu*w2)-(w2'*Bu')), -(Bw), (-(w1*C')-(w2'*D')); -(Bw'),
(gamma*eye(1,1)), -(D'); (-(C*w1)-(D*w2)), -(D), (gamma*eye(1,1))]  // H-infinito
  ///
  L3 = -(A3*w1)-(Bu*w2)-(w1*A3')-(w2'*Bu')  // Retroalimentación de estados
  // Regiones de estabilidad:
  E2_3 = -(A3*w1)-(Bu*w2)-(w1*A3')-(w2'*Bu')-(2*alpha*w1)
  E3_3 = [(radio*w1), (-(A3*w1)-(Bu*w2)); (-(w1*A3')-(w2'*Bu')), (radio*w1)]
  E4_3 = [-(sin(theta)*((A3*w1)+(Bu*w2)+(w1*A3')+(w2'*Bu'))), -(cos(theta)*((A3*w1)+(Bu*w2)-
(w1*A3')-(w2'*Bu'))); -(cos(theta)*((w1*A3')+(w2'*Bu')-(A3*w1)-(Bu*w2))), -
(sin(theta)*((A3*w1)+(Bu*w2)+(w1*A3')+(w2'*Bu')))]
  O3 = [(-(A3*w1)-(w1*A3')-(Bu*w2)-(w2'*Bu')), -(Bw), (-(w1*C')-(w2'*D')); -(Bw'),
(gamma*eye(1,1)), -(D'); (-(C*w1)-(D*w2)), -(D), (gamma*eye(1,1))]  // H-infinito
  ///
  L4 = -(A4*w1)-(Bu*w2)-(w1*A4')-(w2'*Bu')  // Retroalimentación de estados
  // Regiones de estabilidad:
  E2_4 = -(A4*w1)-(Bu*w2)-(w1*A4')-(w2'*Bu')-(2*alpha*w1)
  E3_4 = [(radio*w1), (-(A4*w1)-(Bu*w2)); (-(w1*A4')-(w2'*Bu')), (radio*w1)]
  E4_4 = [-(sin(theta)*((A4*w1)+(Bu*w2)+(w1*A4')+(w2'*Bu'))), -(cos(theta)*((A4*w1)+(Bu*w2)-
(w1*A4')-(w2'*Bu'))); -(cos(theta)*((w1*A4')+(w2'*Bu')-(A4*w1)-(Bu*w2))), -
(sin(theta)*((A4*w1)+(Bu*w2)+(w1*A4')+(w2'*Bu')))]
  O4 = [(-(A4*w1)-(w1*A4')-(Bu*w2)-(w2'*Bu')), -(Bw), (-(w1*C')-(w2'*D')); -(Bw'),
(gamma*eye(1,1)), -(D'); (-(C*w1)-(D*w2)), -(D), (gamma*eye(1,1))]  // H-infinito
  ///
  LMi = -(Awmin*w1)-(Bu*w2)-(w1*Awmin')-(w2'*Bu')  // Retroalimentación de estados
```

```
  // Regiones de estabilidad:
  E2_Mi = -(Awmin*w1)-(Bu*w2)-(w1*Awmin')-(w2'*Bu')-(2*alpha*w1)
  E3_Mi = [(radio*w1), (-(Awmin*w1)-(Bu*w2)); (-(w1*Awmin')-(w2'*Bu')), (radio*w1)]
  E4_Mi = [-(sin(theta)*((Awmin*w1)+(Bu*w2)+(w1*Awmin')+(w2'*Bu'))), -
(cos(theta)*((Awmin*w1)+(Bu*w2)-(w1*Awmin')-(w2'*Bu'))); -(cos(theta)*((w1*Awmin')+(w2'*Bu')-
(Awmin*w1)-(Bu*w2))), -(sin(theta)*((Awmin*w1)+(Bu*w2)+(w1*Awmin')+(w2'*Bu')))]
  OMi = [(-(Awmin*w1)-(w1*Awmin')-(Bu*w2)-(w2'*Bu')), -(Bw), (-(w1*C')-(w2'*D')); -(Bw'),
(gamma*eye(1,1)), -(D'); (-(C*w1)-(D*w2)), -(D), (gamma*eye(1,1))]  // H-infinito
  ///
  LMa = -(Awmax*w1)-(Bu*w2)-(w1*Awmax')-(w2'*Bu')  // Retroalimentación de estados
  // Regiones de estabilidad:
  E2_Ma = -(Awmax*w1)-(Bu*w2)-(w1*Awmax')-(w2'*Bu')-(2*alpha*w1)
  E3_Ma = [(radio*w1), (-(Awmax*w1)-(Bu*w2)); (-(w1*Awmax')-(w2'*Bu')), (radio*w1)]
  E4_Ma = [-(sin(theta)*((Awmax*w1)+(Bu*w2)+(w1*Awmax')+(w2'*Bu'))), -
(cos(theta)*((Awmax*w1)+(Bu*w2)-(w1*Awmax')-(w2'*Bu'))); -
(cos(theta)*((w1*Awmax')+(w2'*Bu')-(Awmax*w1)-(Bu*w2))), -
(sin(theta)*((Awmax*w1)+(Bu*w2)+(w1*Awmax')+(w2'*Bu')))]
  OMa = [(-(Awmax*w1)-(w1*Awmax')-(Bu*w2)-(w2'*Bu')), -(Bw), (-(w1*C')-(w2'*D')); -(Bw'),
(gamma*eye(1,1)), -(D'); (-(C*w1)-(D*w2)), -(D), (gamma*eye(1,1))]  // H-infinito
  ///

LMI=list(L1,E2_1,E3_1,E4_1,O1,L2,E2_2,E3_2,E4_2,O2,L3,E2_3,E3_3,E4_3,O3,L4,E2_4,E3_4,E4_4,O4,LMi,
E2_Mi,E3_Mi,E4_Mi,OMi,LMa,E2_Ma,E3_Ma,E4_Ma,OMa,w1-eye(),gamma-eye())
  OBJ = gamma;
endfunction

w1_init=zeros(Awmin);
w2_init=zeros(Bu');
gamma_init=zeros(D);
XLIST0=list(w1_init,w2_init,gamma_init);
XLISTF=lmisolver(XLIST0,h2hinf_eval);
[w1,w2,gamma]=XLISTF(:)

K = w2*inv(w1);
normHInf = gamma;

CK = C+(D*K);    // Matriz dinámica CK
AK = A+(Bu*K);  // Matriz dinámica AK
EV = spec(AK);  // Valores propios
disp(EV)

// Función de transferencia:
Ts = CK*inv((%s*eye(2,2))-(AK))*Bw+D;

// Time definition
t0 = 0;        // s
tf = 10;       // s
Dt = 1e-3;  // s

t = t0:Dt:tf;
lt = length(t);

//---------------------------------- Grafica del modelo linealizado ---------------------------------- //
// Modelo linealizado:
function dxL=Axis1LinealODE(t, x, m, mo, g, L, fv, TinL, Td)
```

```
  TdL = Td - Tdp;
  x1L = x(1) - x1p;
  x2L = x(2) - x2p;
  dxL(1) = x2L;
  dxL(2) = (((m+mo)*g*L*sin(x1p))/((m+mo)*(L^2)))*(x1L) - (fv/((m+mo)*(L^2)))*(x2L) +
(1/((m+mo)*(L^2)))*(TinL) - (1/((m+mo)*(L^2)))*(TdL);
endfunction

// Definición de los estados y entrada:
x10s = zeros(1,lt); // Valor inicial de la posición
x20s = zeros(1,lt); // Valor inicial de la velocidad
TinL = zeros(1,lt); //Valor inicial del par de fuerza

// Definición de la entrada desconocida:
Td = 1*sin(2*t);
Tdp = (0)/2

//Ley de control:
for k=1:lt-1
  if t(k)>=0.0 && t(k)<=1.6
    g = 9.81;       // Gravedad nominal
    fv = 0.2428;  // Fricción nominal
  elseif t(k)>1.6 && t(k)<=3.2
    g = 9.78;       // Gravedad mínima
    fv = 0.2185; // Fricción mínima
  elseif t(k)>3.2 && t(k)<=5.0
    g = 9.8;        // Variación  de gravedad
    fv = 0.2671; // Fricción máxima
  end

  TinL(k) = K*[x10s(k)-x1p; x20s(k)-x2p];
  xOut = ode("stiff",[x10s(k),x20s(k)],t(k),t(k+1),[1e-5, 1e-
6],list(Axis1LinealODE,m,mo,g,L,fv,TinL(k),Td(k)));
  x10s(k+1) = xOut(1);
  x20s(k+1) = xOut(2);
end

scf(0); clf();
subplot(2,1,1)
plot(t,x10s*(180/%pi),'-b');
set(gca(),"grid",[1 1 1]);
title('Posición angular (x1* = 18°) - Sistema linealizado','FontSize',3);
xlabel('Tiempo[s]','FontSize',3);
ylabel('Grados','FontSize',3);
xgrid;
subplot(2,1,2)
plot(t,TinL,'-k');
title('Par de fuerza obtenido - Sistema linealizado','FontSize',3);
xlabel('Tiempo[s]','FontSize',3);
ylabel('Tau [Nm]','FontSize',3);
xgrid;
scf(1); clf();
plot(t,Td,'-b');
title('Perturbación simulada','FontSize',3);
xlabel('Tiempo[s]','FontSize',3);
```

```
ylabel('Taud [Nm]','FontSize',3);
xgrid;
//-------------------------------------------------------------------------------------------------------------//

//---------------------------------- Grafica del modelo no lineal ---------------------------------- //
// Modelo no lineal:
function dx=Axis1NoLinealODE(t, x, g, m, mo, L, fv, Tin, Td)
   dx(1) = x(2);
   dx(2) = (Tin - Td - ((g*(m+mo)*L)*cos(x(1))) - fv*x(2))/((m+mo)*(L^2));
endfunction

// Definición de los estados y entrada:
x10 = zeros(1,lt); // Valor inicial de la posición
x20 = zeros(1,lt); // Valor inicial de la velocidad
Tin = zeros(1,lt); // Valor inicial del par de fuerza

// Definición de la entrada desconocida:
Td = 1*sin(2*t);

// Ley de control:
for k=1:lt-1
   if t(k)>=0.0 && t(k)<=1.6
      g = 9.81;       // Gravedad nominal
      fv = 0.2428; // Fricción nominal
   elseif t(k)>1.6 && t(k)<=3.2
      g = 9.78;      // Gravedad mínima
      fv = 0.2185; // Fricción mínima
   elseif t(k)>3.2 && t(k)<=5.0
      g = 9.8; // Variación de gravedad
      fv = 0.2671; // Fricción máxima
   end

   Tin(k) = Tinp + (K*[x10(k)-x1p; x20(k)-x2p]);
   xOut2 = ode("stiff",[x10(k),x20(k)],t(k),t(k+1),[1e-5, 1e-
6],list(Axis1NoLinealODE,g,m,mo,L,fv,Tin(k),Td(k)));
   x10(k+1) = xOut2(1);
   x20(k+1) = xOut2(2);
end
Tin(k+1) = Tin(k);

scf(2); clf();
subplot(2,1,1)
plot(t,x10*(180/%pi),'-b');
set(gca(),"grid",[1 1 1]);
title('Posición angular (x1* = 18°) - Sistema no lineal','FontSize',3);
xlabel('Tiempo[s]','FontSize',3);
ylabel('Grados','FontSize',3);
xgrid;
subplot(2,1,2)
plot(t,Tin,'-k');
title('Par de fuerza obtenido','FontSize',3);
xlabel('Tiempo[s]','FontSize',3);
ylabel('Tau [Nm]','FontSize',3);
xgrid;
//-------------------------------------------------------------------------------------------------------------//
```

Apéndice G

Instalación de las funciones de control

Las funciones de control del DYNAMIXEL SDK pueden descargarse libremente desde el repositorio que presenta el fabricante en GitHub; ver la Figura G.1.

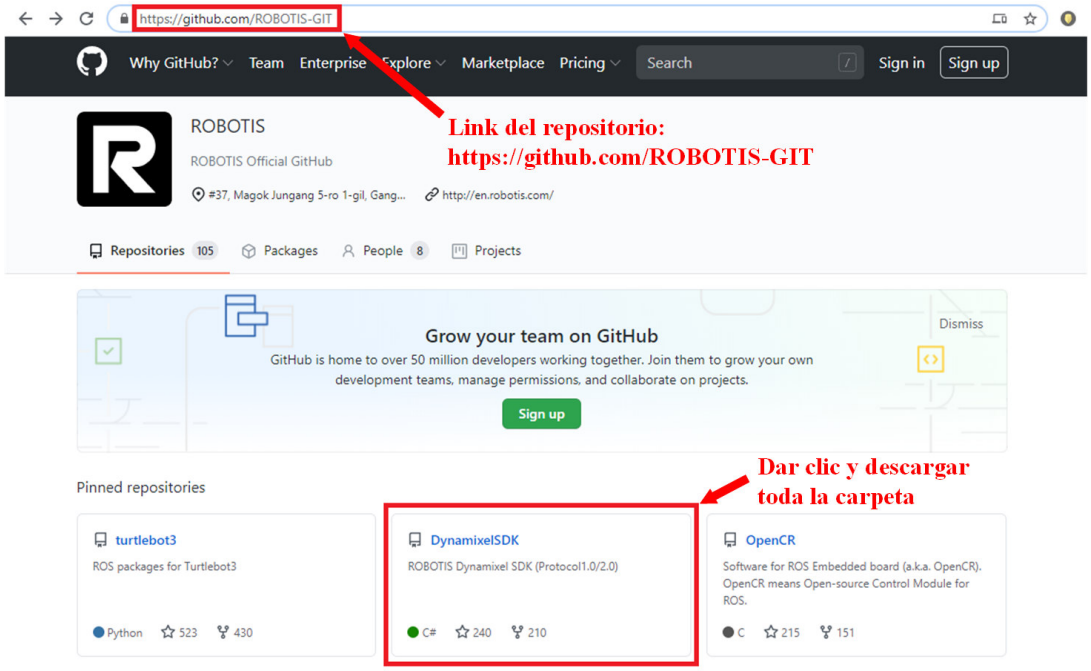


Figura G.1: Repositorio para descargar las funciones de control

Una vez finalizada la descarga se abre la carpeta DynamixelSDK-master, ahí se encontrarán otras carpetas que tienen el nombre de los lenguajes de programación que admiten las funciones de control del DYNAMIXEL SDK, para el caso de este trabajo se abrió la carpeta que tiene como nombre Python.

Dentro de la carpeta Python, se encuentra un archivo que tiene como nombre setup.py,

este archivo es el que se ejecutará desde la consola de comandos de Python (CMD, del inglés: Command Prompt) para instalar las funciones de control, por lo que es importante conocer la dirección de este archivo; ver la Figura G.2.

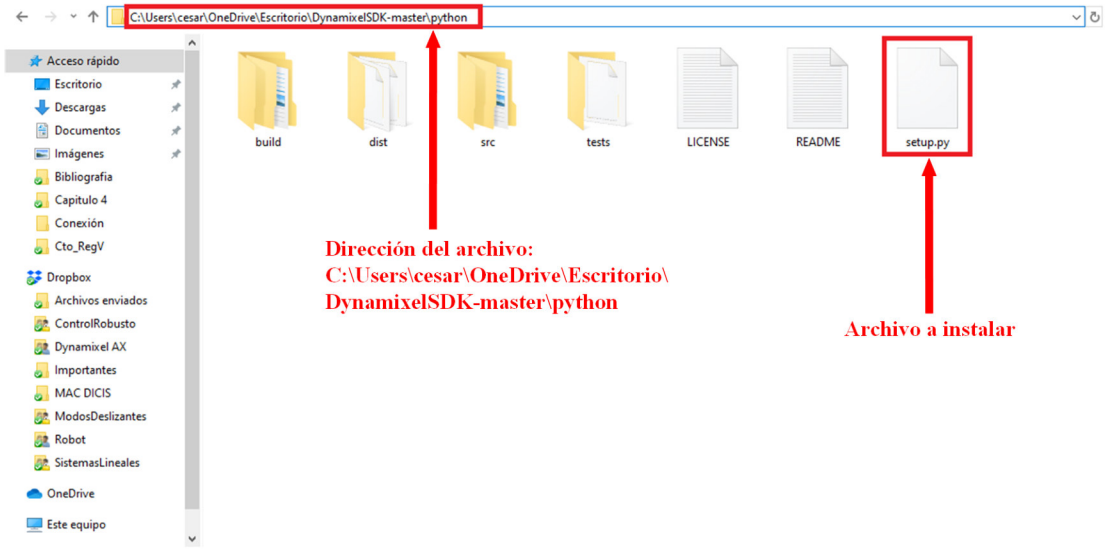


Figura G.2: Dirección del archivo instalador

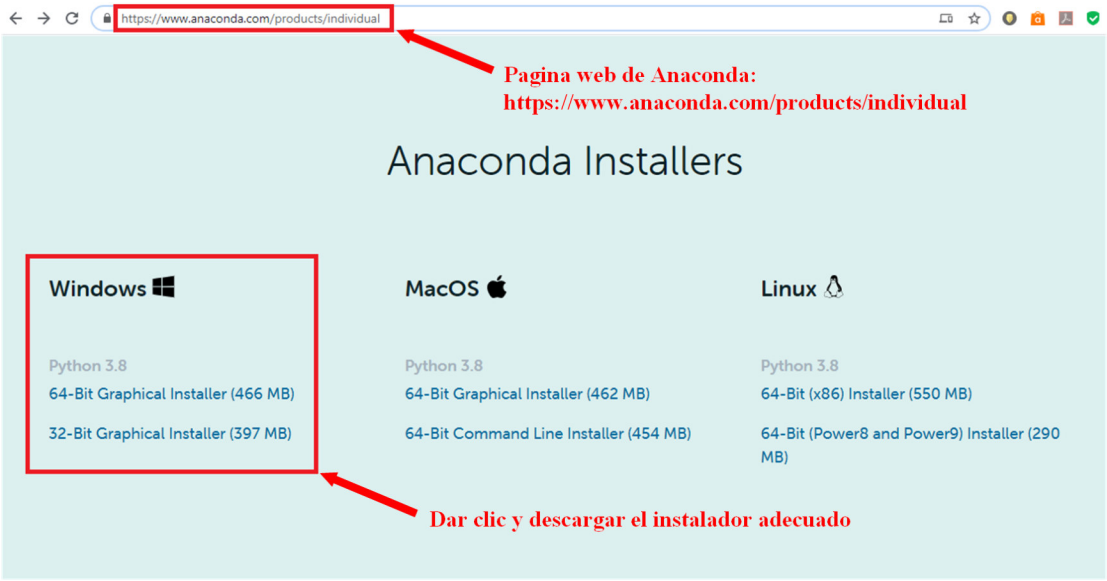


Figura G.3: Pagina web para descargar el distribuidor Anaconda

Para este trabajo se utilizó Anaconda, el cual es una distribución libre y abierta de Python; el distribuidor Anaconda puede descargarse gratuitamente desde su página web y se recomienda descargar la versión más reciente para el sistema operativo que

esté utilizando la PC. En el caso de este trabajo se descargó la versión más reciente para Windows, ver la Figura G.3.

Una vez que la descarga haya finalizado, se da doble clic al instalador de Anaconda para inicializar la ejecución. Durante el proceso de la instalación, aparecen dos opciones de instalación avanzada que permiten personalizar la integración de Anaconda con Windows, ambas opciones deben ser seleccionadas para que la instalación se realice correctamente, vea la Figura G.4. Posteriormente, la instalación queda finalizada.

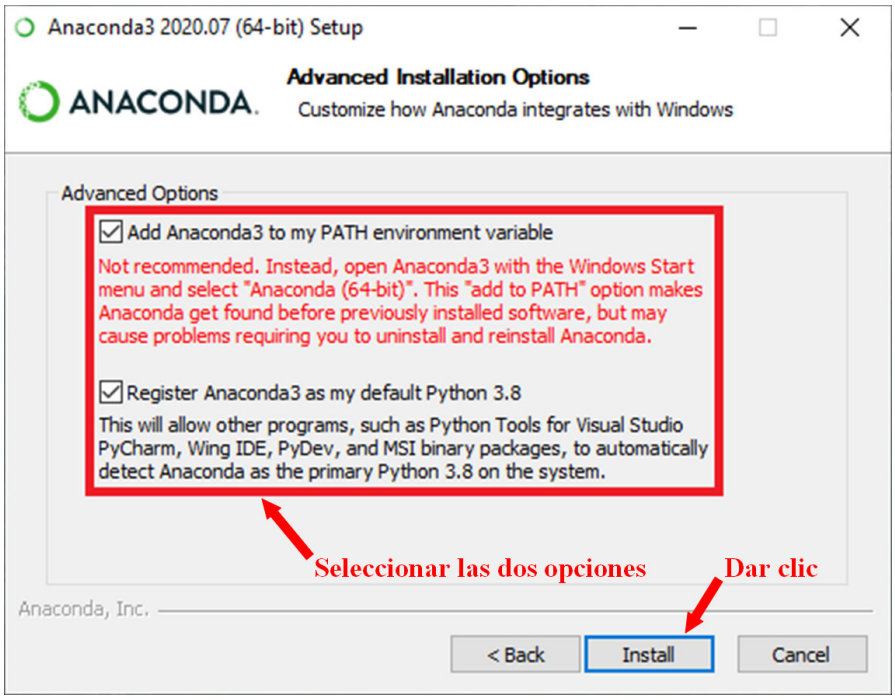


Figura G.4: Opciones de instalación avanzada

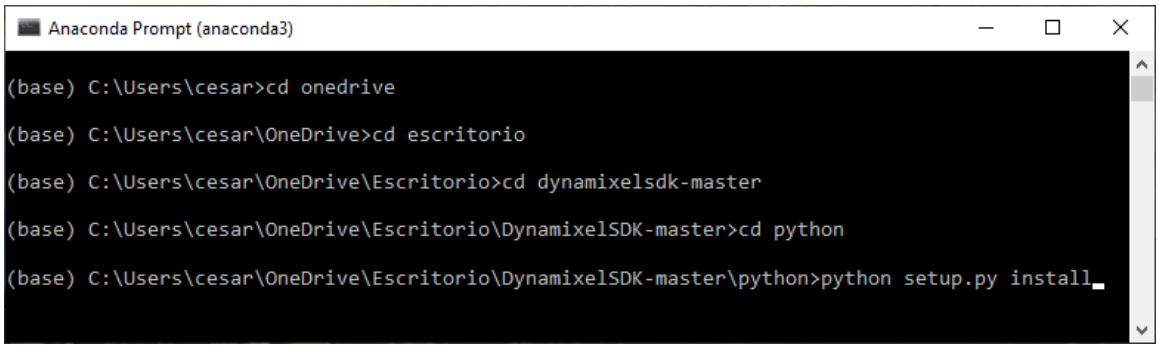


Figura G.5: Instalación de las funciones de control en Python

Una vez instalado el distribuidor Anaconda en la PC, escriba *Anaconda Prompt*

en la barra de búsqueda de Windows y de clic en el icono que tenga dicho nombre.

En Anaconda Prompt escriba la dirección donde se ubica el archivo setup.py, en seguida escriba *python setup.py install* y presione la tecla Enter para iniciar con la instalación de las funciones de control, tal como se muestra en la Figura G.5.

Una vez que la instalación haya finalizado exitosamente, aparecerá una leyenda donde indica que las funciones de control del DYNAMIXEL SDK pudieron ser instaladas en Anaconda; ver la Figura G.6.

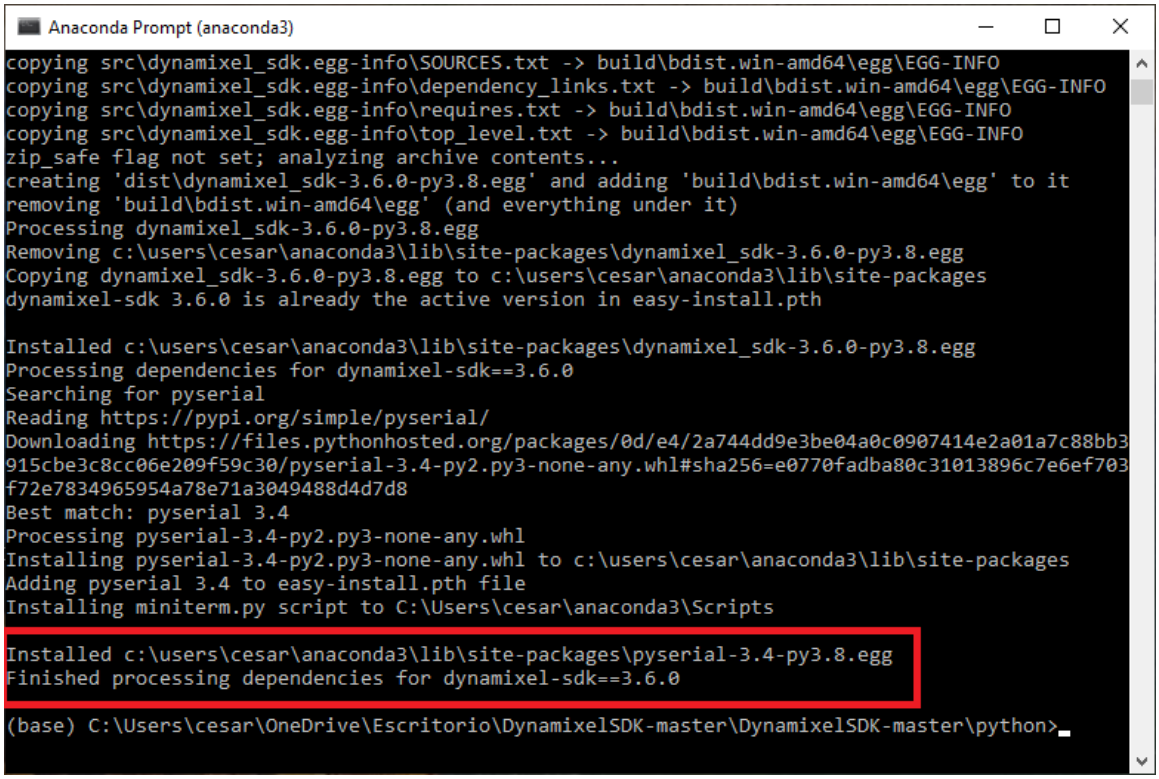


Figura G.6: Funciones de control instaladas

El distribuidor Anaconda tiene incluido al software Spyder, el cual es un entorno de desarrollo integrado y multiplataforma de código abierto (IDE, del inglés: Integrated Development Environment) para programación científica en el lenguaje Python. Por lo tanto, se usará Spyder para realizar la programación de los servomotores Dynamixel AX-12A y AX-18A. En caso de que el icono de acceso directo de Spyder no se encuentre en el escritorio, escriba *Spyder* en la barra de búsqueda de Windows.

Apéndice H

Programación de los Dynamixel en Python

El código de programación empieza con la instrucción que se muestra en la Figura H.1, la cual permite leer caracteres individuales de la información que es producida por el usuario para el control del programa.

```
import os

if os.name == 'nt':
    import msvcrt
    def getch():
        return msvcrt.getch().decode()
else:
    import sys, tty, termios
    fd = sys.stdin.fileno()
    old_settings = termios.tcgetattr(fd)
    def getch():
        try:
            tty.setraw(sys.stdin.fileno())
            ch = sys.stdin.read(1)
        finally:
            termios.tcsetattr(fd, termios.TCSADRAIN, old_settings)
        return ch
```

Figura H.1: Instrucción para leer caracteres individuales en Python

En seguida, se establece la importación de las funciones de control que permiten la operación de los servomotores Dynamixel, las funciones matemáticas y la función de tiempo. Posteriormente, se definen las direcciones y la longitud de bytes de los elementos que van a ser controlados en cada servomotor: escritura y lectura de la posición angular y el par motor; ver la Figura H.2. Las direcciones y la longitud de bytes se encuentran en la Tabla de Control que proporciona ROBOTIS (2020).

```
from dynamixel_sdk import *                    # Importación las funciones de control
import math as m                               # Importación de las funciones matemáticas
import time                                    # Importación de la función tiempo

# Direcciones de los elementos a controlar (tabla de control):
ADDR_MX_TORQUE_ENABLE      = 24                # Direccion para habilitar el torque
ADDR_MX_GOAL_POSITION      = 30                # Direccion para escribir la posición angular
ADDR_MX_PRESENT_POSITION   = 36                # Direccion para leer la posicion angular actual
ADDR_MX_TORQUE_LIMIT       = 34                # Direccion para escribir el par de fuerza
ADDR_MX_PRESENT_LOAD       = 40                # Dirección para leer la carga actual

LEN_MX_GOAL_POSITION       = 4                 # Longitud de bytes para leer posición angular
```

Figura H.2: Importación de funciones y definición de la tabla de control

Para la comunicación de los servomotores existen dos versiones de protocolo: 1.0 y 2.0, para este caso se define el uso del protocolo 1.0, el cual es apropiado para los servomotores de la serie AX. Además, se define el ID de cada servomotor para su identificación; el ID es un valor único y no debe repetirse en ningún otro servomotor. El ID puede personalizarse desde el software RoboPlus; vea el apéndice I.

Asimismo, se define que la tasa de baudios (Baud-Rate) será de 1,000,000 bits por segundo (bps); siendo la velocidad de comunicación en serie entre el Dispositivo U2D2 y los servomotores Dynamixel. También es importante definir, entre comillas, el puerto de comunicación (*DEVICENAME*) que utilizará el Dispositivo U2D2, por ejemplo, 'COM5' en Windows; esto puede visualizarse desde el administrador de dispositivos.

```
# Version del protocolo:
PROTOCOL_VERSION           = 1.0               # Protocolo que utilizan los Dynamixel AX

# Configuraciones predeterminadas:
DXL1_ID                    = 1                 # ID 1er. Servomotor: 1
DXL2_ID                    = 2                 # ID 2do. Servomotor: 2
DXL3_ID                    = 3                 # ID 3er. Servomotor: 3
DXL4_ID                    = 4                 # ID 4to. Servomotor: 4

BAUDRATE                   = 1000000           # Establecer velocidad de transmision (bps)
DEVICENAME                 = 'COM5'            # Puerto de comunicacion con windows

TORQUE_ENABLE              = 1                 # Valor para activar el par de fuerza
TORQUE_DISABLE             = 0                 # Valor para desactivar el par de fuerza
```

Figura H.3: Protocolo de comunicación y configuraciones predeterminadas

Por lo regular se habilita el par motor de los servomotores cuando estos van a rotar, permitiendo al usuario leer las cargas actuales; por otro lado, no se necesita tener habilitado el par motor cuando los servomotores dejan de rotar. Por lo tanto, debe definirse un valor para activar y desactivar el par motor en los servomotores. Esto y

lo anteriormente mencionado puede visualizarse en la Figura H.3.

Cabe señalar que este código de programación se realizó con la finalidad de conocer a las funciones de control (proporcionadas por Dynamixel SDK) que permiten la comunicación y operación de los servomotores Dynamixel. Por lo tanto, este programa hace que la pinza del brazo robótico pueda ubicarse en dos posiciones diferentes que son establecidas por el usuario (sin usar el polinomio de 5to. grado) y además, que el usuario pueda leer las posiciones angulares y los pares motor que presentan los servomotores durante su operación.

En el código de programación se establecen las coordenadas de la primera y segunda posición de ubicación, las medidas de los eslabones que componen al brazo robótico y los pares motor que ejercerán los servomotores, vea la Tabla H.1.

Descripción	**Simbolo**	**Valor**	**Unidades**
Coordenadas de la primera posición	$p_1(x)$	0.15	m
	$p_1(y)$	-0.15	m
	$p_1(z)$	0.05	m
Coordenadas de la segunda posición	$p_2(x)$	0.15	m
	$p_2(y)$	0.15	m
	$p_2(z)$	0.20	m
Pares de fuerza para servomotores	τ_1	0.15	Nm
	τ_2	0.15	Nm
	τ_3	0.20	Nm
	τ_4	0.20	Nm
Altura de la base del robot	h_B	0.079	m
Longitud del eslabón 1	l_1	0.113	m
Longitud del eslabón 2	l_2	0.14	m

Tabla H.1: Parámetros considerados para el código de programación

Posteriormente, se escriben las ecuaciones de la cinemática inversa (pertenecientes a este brazo robótico) para obtener los valores angulares (θ_1, θ_2, θ_3) que deben tomar cada servomotor para lograr ubicar a la pinza mecánica en las dos posiciones establecidas. Además, se define que la pinza mecánica se cierra cuando el cuarto servomotor gira a 0° y se abre cuando gira a 150°; ver la Figura H.4.

Es importante indicar que los servomotores Dynamixel tienen una resolución de 10 bits, es decir, aceptan valores enteros entre 0 y 1023 (0x000 y 0x3FF en forma he-

xadecimal). Cada servomotor tiene disponibilidad de rotar desde 0° igual a 0 (0x000), hasta 300° igual a 1023 (0x3FF), el resto de los valores se calculan de manera proporcional. El mismo caso sería para el par motor, por ejemplo, el valor 1023 (0x3FF) indica que cada servomotor usará el 100 % del par máximo que pueden producir, mientras que el valor 512 (0x200) indica que usaran el 50 % del par máximo.

```
#--------- Ecuaciones de la cinemática inversa: Obtención de ángulos ---------#
for i in range(1,3):
    if i == 1:
    # Establecer las coordenadas de la primer posición:
        Px = P1x              # Coordenada en X de la 1er. posición:  0.15 [m]
        Py = P1y              # Coordenada en Y de la 1er. posición: -0.15 [m]
        Pz = P1z              # Coordenada en Z de la 1er. posición:  0.05 [m]
    else:
    # Establecer las coordenadas de la segunda posición:
        Px = P2x              # Coordenada en X de la 2da. posición:  0.15 [m]
        Py = P2y              # Coordenada en X de la 2da. posición:  0.15 [m]
        Pz = P2z              # Coordenada en X de la 2da. posición:  0.20 [m]

    # Solución de la cinemática inversa:
    w = m.sqrt(Px**2 + Py**2)
    THETA1 = m.atan2(Py,Px)*(180/m.pi)         # Pos. ang. para 1er. servomotor
    if i == 1:
        THETA1_P1 = THETA1    # Valor de θ1 para ubicar a la pinza en la pos. 1
    else:
        THETA1_P2 = THETA1    # Valor de θ1 para ubicar a la pinza en la pos. 2

    phi = m.atan2(w,Pz-hB)*(180/m.pi)
    r = m.sqrt(Px**2 + Py**2 + (Pz-hB)**2)
    beta = 90 - phi
    cg = (r**2 + l1**2 - l2**2)/(2*l1*r)
    sg = m.sqrt(1 - cg**2)
    gama = m.atan2(sg,cg)*(180/m.pi)
    THETA2 = beta + gama                       # Pos. ang. para 2do. servomotor
    if i == 1:
        THETA2_P1 = THETA2    # Valor de θ2 para ubicar a la pinza en la pos. 1
    else:
        THETA2_P2 = THETA2    # Valor de θ2 para ubicar a la pinza en la pos. 2

    ca = (l1**2 + l2**2 - r**2)/(2*l1*l2)
    sa = m.sqrt(1 - ca**2)
    alfa = m.atan2(sa,ca)*(180/m.pi)
    THETA3 = -(180 - alfa)                     # Pos. ang. para 3er. servomotor
    if i == 1:
        THETA3_P1 = THETA3    # Valor de θ3 para ubicar a la pinza en la pos. 1
    else:
        THETA3_P2 = THETA3    # Valor de θ3 para ubicar a la pinza en la pos. 2

CERRAR_PINZA = 0                           # Valor en grados para cerrar pinza
ABRIR_PINZA = 150                          # Valor en grados para abrir pinza
#-----------------------------------------------------------------------------#
```

Figura H.4: Solución de la cinemática inversa para la obtención de ángulos

Por lo tanto, se convirtieron a valores enteros, los valores de par motor que fueron establecidos por el usuario y los valores angulares que se obtuvieron de la cinemática inversa; ver Figura H.5. Para la conversión de los valores angulares a enteros, se tomó

en cuenta la posición que presentan los servomotores en la estructura robótica.

```
#----------- Convertir valores de par de fuerza y pos. angular a enteros (bits) ----------#
DXL1_VAL_TORQ  = round(TAU1*(1023/1.8))                # Par de fuerza para 1er. servomotor
DXL1_VAL_POS1  = round((THETA1_P1 + 150)*(1023/300))   # Rotación del servo 1 para pos. 1
DXL1_VAL_POS2  = round((THETA1_P2 + 150)*(1023/300))   # Rotación del servo 1 para pos. 2

DXL2_VAL_TORQ  = round(TAU2*(1023/1.5))                # Par de fuerza para 2do. servomotor
DXL2_VAL_POS1  = round((240 - (THETA2_P1))*(1023/300)) # Rotación del servo 2 para pos. 1
DXL2_VAL_POS2  = round((240 - (THETA2_P2))*(1023/300)) # Rotación del servo 2 para pos. 2

DXL3_VAL_TORQ  = round(TAU3*(1023/1.5))                # Par de fuerza para 3er. servomotor
DXL3_VAL_POS1  = round((THETA3_P1 + 150)*(1023/300))   # Rotación del servo 3 para pos. 1
DXL3_VAL_POS2  = round((THETA3_P2 + 150)*(1023/300))   # Rotación del servo 3 para pos. 2

DXL4_VAL_TORQ  = round(TAU4*(1023/1.5))                # Par de fuerza para 4to. servomotor
DXL4_VAL_POS1  = round((ABRIR_PINZA)*(1023/300))       # Rot. del servo 4 para abrir pinza
DXL4_VAL_POS2  = round((CERRAR_PINZA)*(1023/300))      # Rot. del servo 4 para cerrar pinza
#-------------------------------------------------------------------------------------------#

# Vector que define entre que valores van a rotar los servomotores:
index = 0                                              # Para seleccionar columna del vector
dxl1_posiciones = [DXL1_VAL_POS1, DXL1_VAL_POS2]
dxl2_posiciones = [DXL2_VAL_POS1, DXL2_VAL_POS2]
dxl3_posiciones = [DXL3_VAL_POS1, DXL3_VAL_POS2]
dxl4_posiciones = [DXL4_VAL_POS1, DXL4_VAL_POS2]
```

Figura H.5: Conversión a valores enteros y definición de vectores

Después, se define un vector fila de $\mathbb{R}^2$ para cada servomotor y en cada columna se escriben los correspondientes valores angulares que llevarán al robot a la posición uno y dos (posiciones objetivo); además, se define un indicador (index) que será usado más adelante para seleccionar las columnas requeridas; ver nuevamente la Figura H.5.

En la Figura H.6 se muestran las funciones básicas que permiten establecer la comunicación del puerto USB, la velocidad de transmisión de datos y la construcción de paquetes de datos para el control simultáneo de los servomotores. La descripción de estás funciones básicas de control se muestran en la Tabla H.2.

En la Figura H.7 se muestra la parte del código que permite la habilitación y asignación del par motor para cada servomotor. Para este caso, la función *write1ByteTxRx* ordena que a los servomotores, usando el protocolo de comunicación 1.0, se les escribirá 1 byte del valor que permite habilitar el par motor (TORQUE_ENABLE) en la dirección correspondiente (ADDR_MX_TORQUE_ENABLE). Una vez dada la orden, la función comprueba el resultado y en caso de que no se reciba un error de comunicación o de hardware, se mostrará en consola la leyenda de conexión exitosa.

La función *write2ByteTxRx* es similar a la anterior, solo que ahora a cada servomotor

se le escribirá 2 bytes del valor del par motor (anteriormente establecido) en la dirección que corresponde a la asignación (ADDR_MX_TORQUE_LIMIT). El tamaño de bytes para la habilitación y la asignación del par motor se indica en la tabla de control que proporciona el fabricante en el manual de usuario, (ROBOTIS, 2020).

```
# Inicializar el puerto de comunicación:
portHandler = PortHandler(DEVICENAME)

# Inicializar el PacketHandler para construcción y almacenamiento de paquetes:
packetHandler = PacketHandler(PROTOCOL_VERSION)

# Inicializar la instancia de GroupSyncWrite para el control simultáneo:
groupSyncWrite = GroupSyncWrite(portHandler, packetHandler, ADDR_MX_GOAL_POSITION,
                                LEN_MX_GOAL_POSITION)

# Abrir el puerto de comunicación:
if portHandler.openPort():
    print("Se logro abrir el puerto")
else:
    print("Error al abrir el puerto")
    print("Presione cualquier tecla para terminar...")
    getch()
    quit()

# Ajustar la velocidad de transmision:
if portHandler.setBaudRate(BAUDRATE):
    print("Se logro ajustar la velocidad de transmision")
else:
    print("Error al ajustar la velocidad de transmision")
    print("Presione cualquier tecla para terminar...")
    getch()
    quit()
```

Figura H.6: Funciones básicas para la comunicación y el control simultáneo

Funciones	**Descripción**
PortHandler	Establece la ruta del puerto de comunicación e inicializa las funciones apropiadas para el control del puerto serie.
PacketHandler	Establece la versión del protocolo e inicializa las funciones para la construcción y el almacenamiento de paquetes.
GroupSyncWrite	Inicializa las funciones que permiten la construcción de paquetes para el control simultáneo de los servomotores al escribir datos de la misma longitud en la misma dirección de la tabla de control (ADDR_MX_GOAL_POSITION).
openPort	Abre la ruta del puerto que fue establecida, para realizar una comunicación en serie con los servomotores.
setBaudRate	Establece la velocidad de comunicación en baudios.

Tabla H.2: Funciones que permiten la comunicación y el control simultáneo

```
# Habilitacion del par motor para cada servomotor:
dxl_comm_result = packetHandler.write1ByteTxRx(portHandler, DXL1_ID, ADDR_MX_TORQUE_ENABLE,
                                               TORQUE_ENABLE)
if dxl_comm_result == COMM_SUCCESS:
    print("Se logro conectar al Dynamixel #%d" % DXL1_ID)    # Dynamixel #1

dxl_comm_result = packetHandler.write1ByteTxRx(portHandler, DXL2_ID, ADDR_MX_TORQUE_ENABLE,
                                               TORQUE_ENABLE)
if dxl_comm_result == COMM_SUCCESS:
    print("Se logro conectar al Dynamixel #%d" % DXL2_ID)    # Dynamixel #2

dxl_comm_result = packetHandler.write1ByteTxRx(portHandler, DXL3_ID, ADDR_MX_TORQUE_ENABLE,
                                               TORQUE_ENABLE)
if dxl_comm_result == COMM_SUCCESS:
    print("Se logro conectar al Dynamixel #%d" % DXL3_ID)    # Dynamixel #3

dxl_comm_result = packetHandler.write1ByteTxRx(portHandler, DXL4_ID, ADDR_MX_TORQUE_ENABLE,
                                               TORQUE_ENABLE)
if dxl_comm_result == COMM_SUCCESS:
    print("Se logro conectar al Dynamixel #%d" % DXL4_ID)    # Dynamixel #4

#Asignación del par motor requerido para cada servomotor:
packetHandler.write2ByteTxRx(portHandler, DXL1_ID, ADDR_MX_TORQUE_LIMIT, DXL1_VAL_TORQ)
packetHandler.write2ByteTxRx(portHandler, DXL2_ID, ADDR_MX_TORQUE_LIMIT, DXL2_VAL_TORQ)
packetHandler.write2ByteTxRx(portHandler, DXL3_ID, ADDR_MX_TORQUE_LIMIT, DXL3_VAL_TORQ)
packetHandler.write2ByteTxRx(portHandler, DXL4_ID, ADDR_MX_TORQUE_LIMIT, DXL4_VAL_TORQ)
time.sleep(0.1) # Suspende la ejecución durante 1 [ms]
```

Figura H.7: Funciones para habilitar y asignar el par motor

Para la operación simultánea de los servomotores, se desarrolló el código que se muestra en la Figura H.8. Esta parte del código está dentro de un ciclo while, en el cual se declara una variable (stop) para especificar el número de veces que el controlador va a escribir y leer los valores angulares en los servomotores a través de la transmisión/recepción de paquetes (Tx/Rx). El ciclo finaliza al cumplirse la condición.

Posteriormente, se asignan los valores angulares de la posición objetivo en una matriz de bytes, usando las funciones *DXL_LOBYTE* y *DXL_HIBYTE*, *DXL_LOWORD* y *DXL_HIWORD*, la descripción de estas funciones se muestra en la tabla H.3. Como la variable index se definió inicialmente como cero, entonces, a la matriz de bytes se le asignaron los valores angulares que corresponden a la primera posición.

En seguida se define la función *AddParam* para cada servomotor, la cual permitirá almacenar el ID de los servomotores y sus correspondientes posiciones objetivo (arreglada en una matriz de bytes) en la función *SyncWrite*, para que puedan ser leídas por la función *txPacket* y transmita al mismo tiempo, el paquete de datos a los servomotores para su operación simultánea. Después, la función *getTxRxResult* obtendrá el resultado de la comunicación y si hay algún error, este será mostrado en la consola.

```
stop = 0

while 1:
    if stop == 3:     # Especifica el número de veces que se ejecutará el ciclo
        break

    # Asignar el valor de la posición objetivo en la matriz de bytes:
    param1_posiciones = [DXL_LOBYTE(DXL_LOWORD(dxl1_posiciones[index])),
                         DXL_HIBYTE(DXL_LOWORD(dxl1_posiciones[index])),
                         DXL_LOBYTE(DXL_HIWORD(dxl1_posiciones[index])),
                         DXL_HIBYTE(DXL_HIWORD(dxl1_posiciones[index]))]
    param2_posiciones = [DXL_LOBYTE(DXL_LOWORD(dxl2_posiciones[index])),
                         DXL_HIBYTE(DXL_LOWORD(dxl2_posiciones[index])),
                         DXL_LOBYTE(DXL_HIWORD(dxl2_posiciones[index])),
                         DXL_HIBYTE(DXL_HIWORD(dxl2_posiciones[index]))]
    param3_posiciones = [DXL_LOBYTE(DXL_LOWORD(dxl3_posiciones[index])),
                         DXL_HIBYTE(DXL_LOWORD(dxl3_posiciones[index])),
                         DXL_LOBYTE(DXL_HIWORD(dxl3_posiciones[index])),
                         DXL_HIBYTE(DXL_HIWORD(dxl3_posiciones[index]))]
    param4_posiciones = [DXL_LOBYTE(DXL_LOWORD(dxl4_posiciones[index])),
                         DXL_HIBYTE(DXL_LOWORD(dxl4_posiciones[index])),
                         DXL_LOBYTE(DXL_HIWORD(dxl4_posiciones[index])),
                         DXL_HIBYTE(DXL_HIWORD(dxl4_posiciones[index]))]

    # Agregar el valor de la posición objetivo al almacenamiento de parámetros de Syncwrite:
    groupSyncWrite.addParam(DXL1_ID, param1_posiciones)
    groupSyncWrite.addParam(DXL2_ID, param2_posiciones)
    groupSyncWrite.addParam(DXL3_ID, param3_posiciones)
    groupSyncWrite.addParam(DXL4_ID, param4_posiciones)

    # Efectuar la sincronización:
    dxl_comm_result = groupSyncWrite.txPacket()
    if dxl_comm_result != COMM_SUCCESS:
        print("%s" % packetHandler.getTxRxResult(dxl_comm_result))

    # Borrar el almacenamiento de los datos procesados:
    groupSyncWrite.clearParam()

    time.sleep(3.0) # Suspende la ejecución durante 3 [s]
```

Figura H.8: Funciones para la operación simultánea de los servomotores

Funciones	**Descripción**
write1ByteTxRx	Transmite y recibe un paquete de 1 byte.
write2ByteTxRx	Transmite y recibe paquetes de 2 bytes.
DXL_LOWORD(l)	Obtiene un valor del tipo de palabra más bajo de l.
DXL_HIWORD(l)	Obtiene un valor del tipo de palabra más alto de l.
DXL_LOBYTE(w)	Obtiene un valor del tipo de byte más bajo de w.
DXL_HIBYTE(w)	Obtiene un valor del tipo de byte más alto de w.
addParam	Añade almacenamiento de parámetros para leer.
txPacket	Transmite el paquete datos al servomotor correspondiente (indicando su ID).
getTxRxResult	Obtiene el resultado de la comunicación.
clearParam	Borra el almacenamiento de parámetros.

Tabla H.3: Funciones para el control del par motor y la posición angular

Finalmente, se escribe la función *clearParam* para borrar los datos que fueron almacenados en la función *SyncWrite*; esto permitirá almacenar más adelante un nuevo paquete de datos con nueva información. La Tabla H.3 da una descripción de las funciones que permiten establecer el par motor y la posición angular de cada servomotor.

```
# Leer la posición actual de los cuatro servomotores:
DXL1_ACT_POS = packetHandler.read2ByteTxRx(portHandler, DXL1_ID, ADDR_MX_PRESENT_POSITION)
DXL2_ACT_POS = packetHandler.read2ByteTxRx(portHandler, DXL2_ID, ADDR_MX_PRESENT_POSITION)
DXL3_ACT_POS = packetHandler.read2ByteTxRx(portHandler, DXL3_ID, ADDR_MX_PRESENT_POSITION)
DXL4_ACT_POS = packetHandler.read2ByteTxRx(portHandler, DXL4_ID, ADDR_MX_PRESENT_POSITION)

# Conversión de valores enteros a grados:
THETA1_R = DXL1_ACT_POS[0]*(300/1023) - 150
THETA2_R = 240 - DXL2_ACT_POS[0]*(300/1023)
THETA3_R = DXL3_ACT_POS[0]*(300/1023) - 150
PINZA_R  = DXL4_ACT_POS[0]*(300/1023)

# Imprimir en consola la lectura de la pos. angular actual:
print("[ID: %01d] Posicion actual: %.2f grados" % (DXL1_ID, THETA1_R))
print("[ID: %01d] Posicion actual: %.2f grados" % (DXL2_ID, THETA2_R))
print("[ID: %01d] Posicion actual: %.2f grados" % (DXL3_ID, THETA3_R))
print("[ID: %01d] Posicion actual: %.2f grados" % (DXL4_ID, PINZA_R))

time.sleep(0.5) # Suspende la ejecución durante 0.5 [s]

# Leer la carga actual de los cuatro servomotores:
DXL1_ACT_CARGA = packetHandler.read2ByteTxRx(portHandler, DXL1_ID, ADDR_MX_PRESENT_LOAD)
DXL2_ACT_CARGA = packetHandler.read2ByteTxRx(portHandler, DXL2_ID, ADDR_MX_PRESENT_LOAD)
DXL3_ACT_CARGA = packetHandler.read2ByteTxRx(portHandler, DXL3_ID, ADDR_MX_PRESENT_LOAD)
DXL4_ACT_CARGA = packetHandler.read2ByteTxRx(portHandler, DXL4_ID, ADDR_MX_PRESENT_LOAD)

# Conversión de valores enteros a valores de momento de fuerza:
if DXL1_ACT_CARGA[0] <= 1023:
    CARGA_DXL1 = DXL1_ACT_CARGA[0]*(1.8/1023)
else:
    CARGA_DXL1 = (DXL1_ACT_CARGA[0]-1024)*(1.8/1023)

if DXL2_ACT_CARGA[0] <= 1023:
    CARGA_DXL2 = DXL2_ACT_CARGA[0]*(1.5/1023)
else:
    CARGA_DXL2 = (DXL2_ACT_CARGA[0]-1024)*(1.5/1023)

if DXL3_ACT_CARGA[0] <= 1023:
    CARGA_DXL3 = DXL3_ACT_CARGA[0]*(1.5/1023)
else:
    CARGA_DXL3 = (DXL3_ACT_CARGA[0]-1024)*(1.5/1023)

if DXL4_ACT_CARGA[0] <= 1023:
    CARGA_DXL4 = DXL4_ACT_CARGA[0]*(1.5/1023)
else:
    CARGA_DXL4 = (DXL4_ACT_CARGA[0]-1024)*(1.5/1023)

# Imprimir en consola la lectura de la carga actual:
print("[ID: %01d] Carga actual: %.5f Nm" % (DXL1_ID, CARGA_DXL1))
print("[ID: %01d] Carga actual: %.5f Nm" % (DXL2_ID, CARGA_DXL2))
print("[ID: %01d] Carga actual: %.5f Nm" % (DXL3_ID, CARGA_DXL3))
print("[ID: %01d] Carga actual: %.5f Nm" % (DXL4_ID, CARGA_DXL4))

time.sleep(0.5) # Suspende la ejecución durante 0.5 [s]
```

Figura H.9: Funciones para leer la posición angular y la carga actual

Una vez que los servomotores hayan rotado simultáneamente a las posiciones angulares asignadas, se tiene que comprobar si dichas rotaciones se realizaron correctamente. Por lo tanto, se desarrolló el siguiente código de programación que se muestra en la Figura H.9. En esta parte de código se utilizó la función *read2ByteTxRx*, la cual permite solicitar datos de 2 bytes de los servomotores para su lectura. Como en este caso se requiere leer los datos de la posición angular actual de cada servomotor, entonces, a dicha función se le definirá el ID del servomotor requerido, el protocolo y puerto de comunicación que se está utilizando y la dirección que permite la lectura de la posición angular actual (ADDR_MX_PRESENT_POSITION).

Para leer la carga actual de los servomotores, se usa nuevamente la función *read2ByteTxRx*, la cual quedará definida por el ID del servomotor requerido, el protocolo y puerto de comunicación que se está utilizando y la dirección que permite la lectura de la carga actual (ADDR_MX_PRESENT_LOAD); ver la Figura H.9.

Debido a que la función de lectura solicita datos de 2 bytes, entonces, los valores leídos (posición angular y carga actual) estarán representados por valores enteros, entre 0 y 1023 para la posición angular, y entre 0 y 2047 para la carga, por lo que fue necesario realizar un tipo de conversión contraria a la que se hizo al principio del programa. Posteriormente dichos valores serán mostrados en la ventana de la consola.

```
    # Cambiar posición objetivo:
    if index == 0:
        index = 1       # Llevará a la pinza mecánica a la segunda posición (P2)
    else:
        index = 0       # Llevará a la pinza mecánica a la primera posición (P1)

    stop += 1

    time.sleep(0.5)     # Suspende la ejecución durante 0.5 [s]

# Cerrar puerto de comunicación:
portHandler.closePort()
```

Figura H.10: Sentencia para cambiar posición objetivo

Por último, se cambia el valor de la variable index a uno, para que ahora los servomotores roten hacía los valores angulares que corresponden a la segunda posición y así la pinza logre ubicarse ahí; ya que lo haya logrado, volverá a ubicarse en la primera posición. Una vez que se haya ejecutado el programa tres veces, se rompe el ciclo while y finaliza el programa. Para cerrar el puerto de comunicación, se utiliza la función *closePort*, vea la Figura H.10.

Apéndice I

Configuración de los Dynamixel en RoboPlus

Para la configuración de los servomotores Dynamixel, se debe de instalar previamente, un software que proporciona el fabricante, el cual tiene como nombre RoboPlus. Para descargar RoboPlus, ingrese al sitio que se muestra en la Figura I.1.

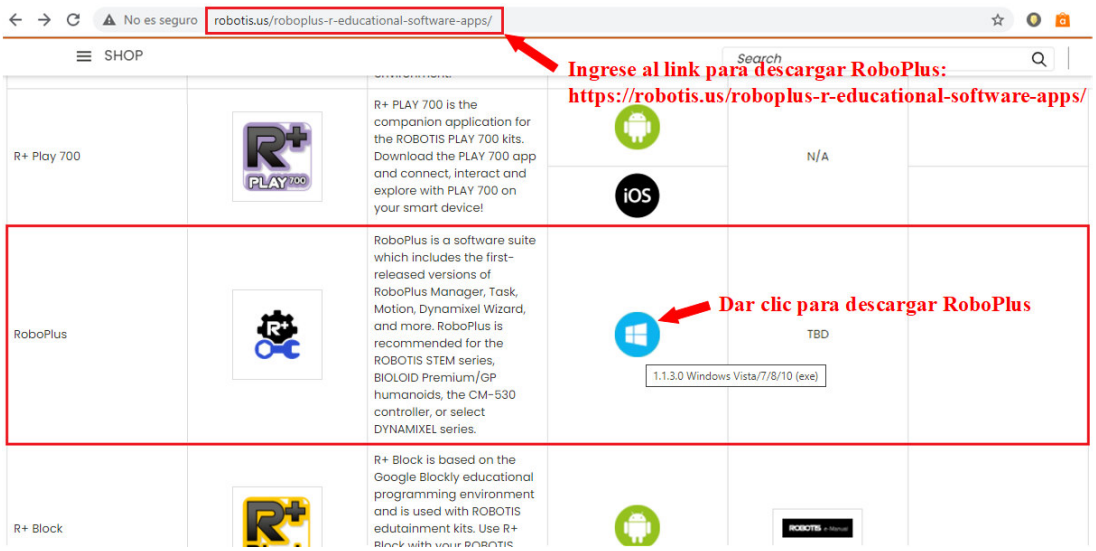


Figura I.1: Sitio para descargar RoboPlus

Una vez finalizada la descarga, dirijase a la opción de descargas y dé doble clic al icono de RoboPlus para proceder con la instalación.

Una vez instalado RoboPlus en la PC, abra el software y de clic en la pestaña *Expert*, después de clic en la opción que dice *Dynamixel Wizard*, vea la Figura I.2.

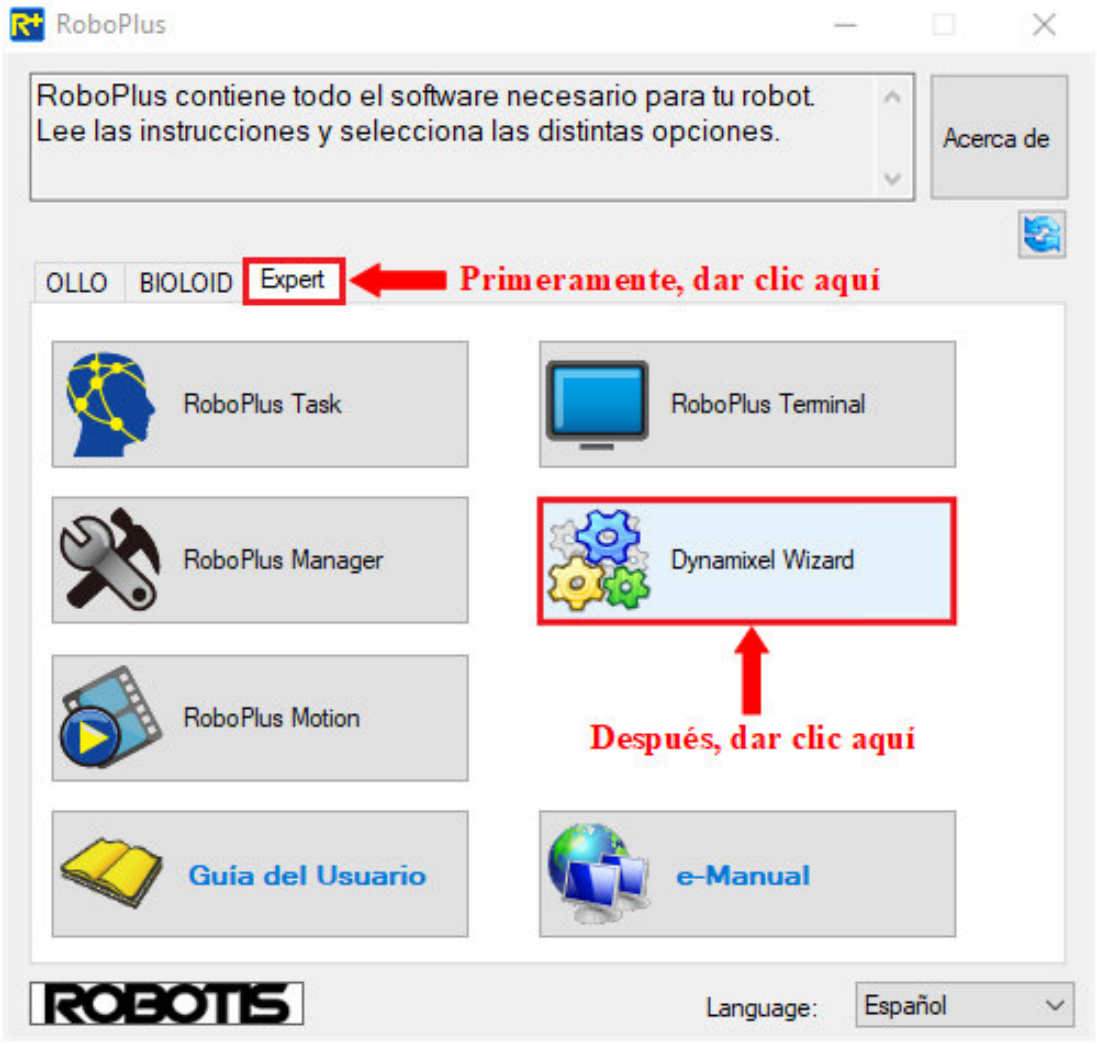


Figura I.2: Ventana de inicio de RoboPlus

Posteriormente, aparecerá una ventana como la que se muestra en la Figura I.3. En dicha ventana seleccione al puerto de comunicación donde se encuentre conectado el dispositivo U2D2 (para el caso de este trabajo) y luego de clic sobre el icono que permite abrir el puerto.

Enseguida, aparecerá una ventana con opciones para seleccionar la velocidad de transmisión de datos y el protocolo de comunicación que se requiera. Para el caso de los servomotores Dyanmixel de la serie AX se seleccionó una velocidad de 1000000 *bps* y el protocolo 1.0. Posteriormente, se da clic en *Empezar a buscar*. Vea la Figura I.4.

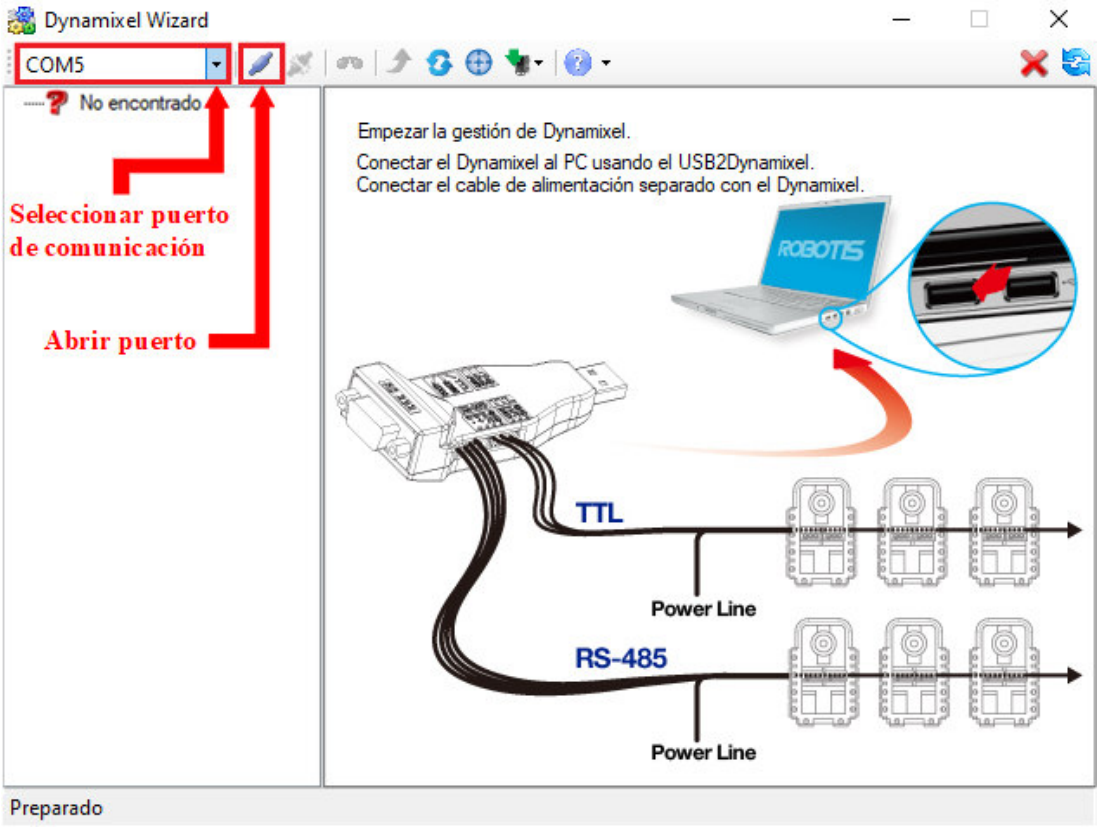


Figura I.3: Selección y habilitación del puerto de comunicación

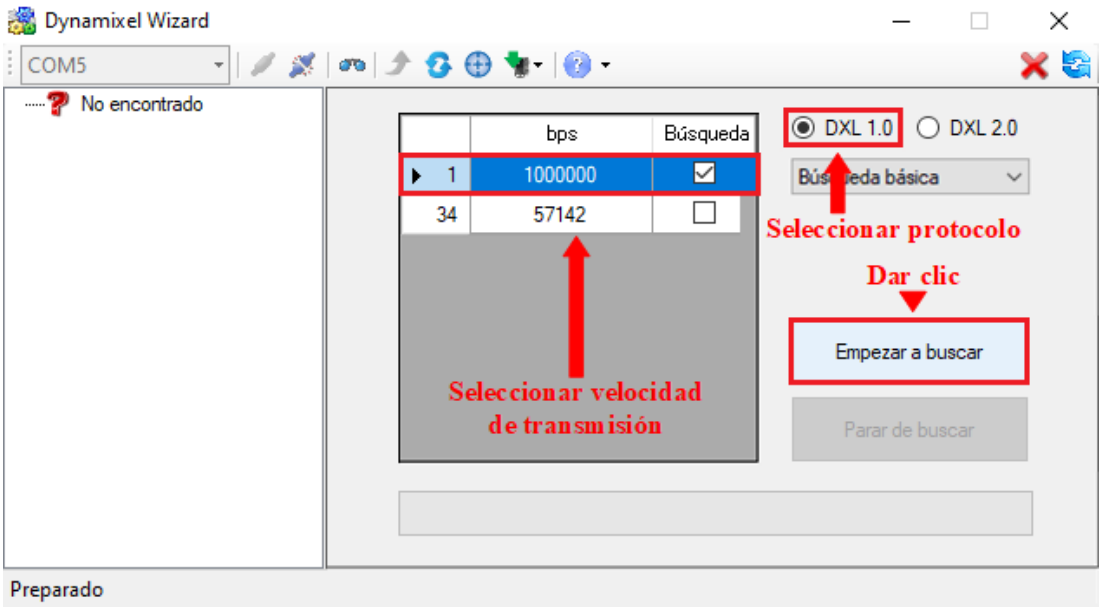


Figura I.4: Selección de la velocidad de transmisión y protocolo

Después, el software estará buscando a los servomotores que se encuentren conectados en el puerto de comunicación seleccionado, una vez que aparezcan los modelos de los servomotores en la ventana del lado izquierdo, puede parar la búsqueda y seleccionar al servomotor que quisiera configurar, vea la Figura I.5.

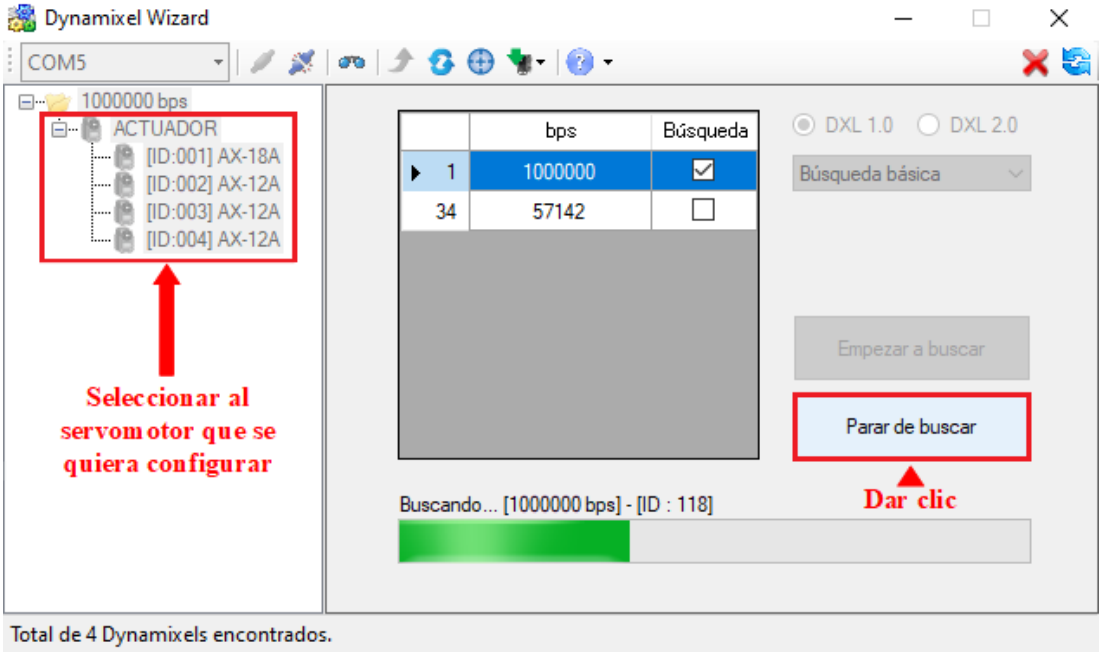


Figura I.5: Búsqueda de los servomotores conectados al puerto

Una vez seleccionado el servomotor, aparecerá una tabla de control con los parámetros que pueden ser modificados. En la Figura I.6 se muestra que para cambiar el ID del servomotor, basta con seleccionar dicho parámetro en la tabla de control donde del lado izquierdo aparecerá una lista con los posibles ID's que se pueden asignar.

Además del ID, se pueden modificar otros parámetros como, por ejemplo, el par máximo, la velocidad de movimiento, los límites mínimos y máximos para la tensión de alimentación y la temperatura, lo cual permitirá evitar el daño de los mismos en caso de que se presenten fallas en los dispositivos conectados o en los propios servomotores. También es posible conocer la lectura de la posición, velocidad, carga, tensión y temperatura actual del servomotor.

Una vez que haya terminado de modificar los parámetros de los servomotores requeridos, puede dar clic sobre el icono que aparece en la Figura I.7 para cerrar el puerto y posteriormente cerrar el software.

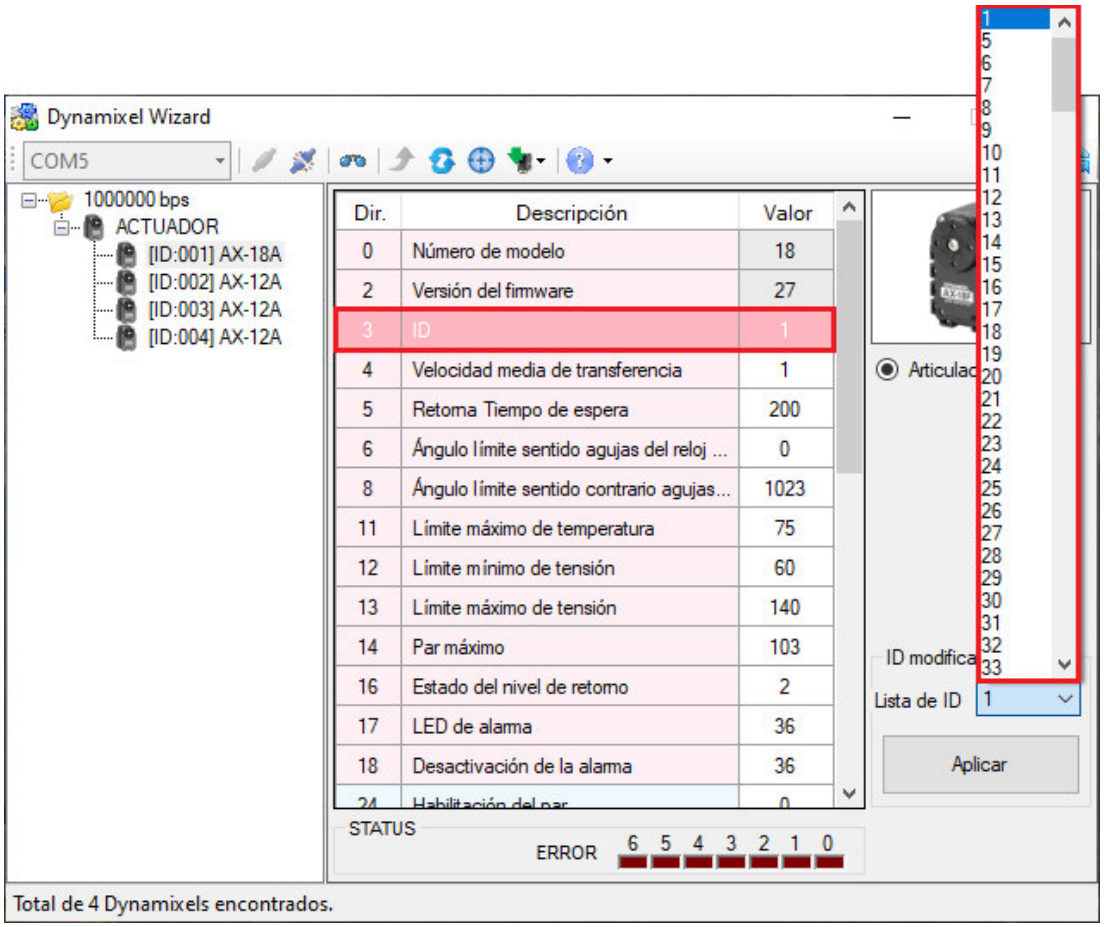


Figura I.6: Modificación del ID para servomotor

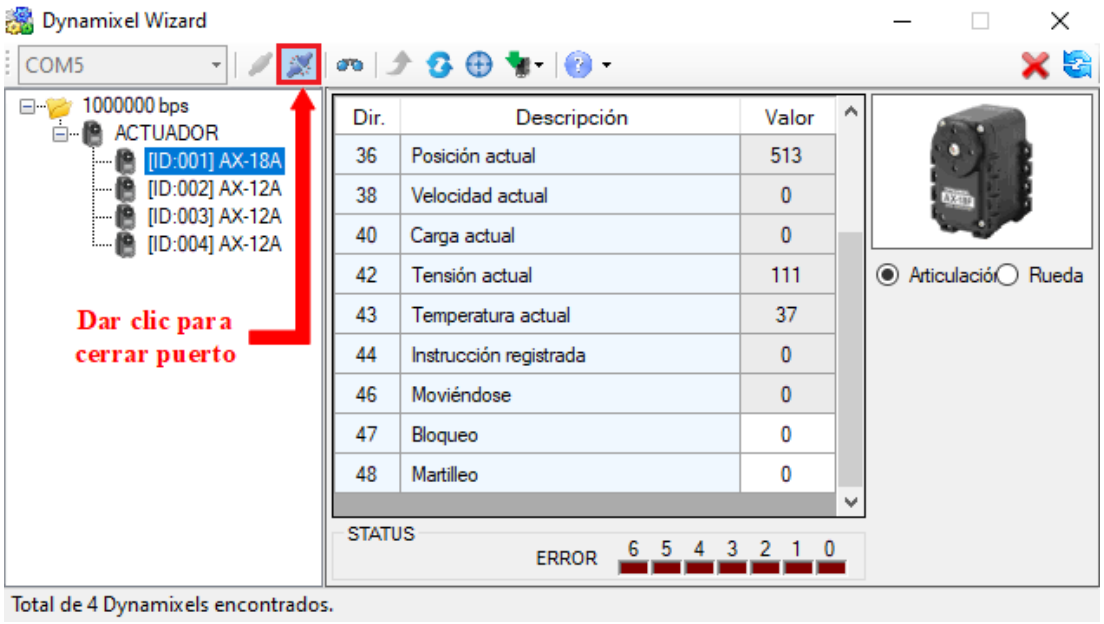


Figura I.7: Inhabilitación del puerto

Apéndice J

Simulación del controlador twisting

```
//// UNIVERSIDAD DE GUANAJUATO - DIVISION DE INGENIERIAS CAMPUS IRAPUATO - SALAMANCA ///
/////// MAESTRIA EN INGENIERÍA ELÉCTRICA - INSTRUMENTACIÓN Y SISTEMAS DIGITALES ///////
///// SIMULACIÓN DEL CONTROLADOR TWISTING CON MODELO DINÁMICO DEL BRAZO ROBÓTICO ///
////////////////// PROGRAMADO POR: CESAR EDUARDO CONEJO BENITEZ //////////////////

clc;
clear;
xdel(winsid());

//------------------------ Sistema dinámico no lineal ------------------------//
function dx=RobotSys2ODE(t, x, g, m, l, v, Tin1, Tin2, Tin3, Td1, Td2, Td3)
  A = m(3)*(l(3)^2);
  B = m(3)*(l(3)^2)*x(2)*cos(x(3)+x(5))*sin(x(3)+x(5));
  C = (m(2) + m(3))*(l(2)^2);
  D = (m(2) + m(3))*(l(2)^2)*x(2)*cos(x(3))*sin(x(3));
  E = m(3)*l(2)*l(3);
  F = m(3)*g*l(3)*cos(x(3)+x(5));
  s223 = sin((2*x(3)) + x(5));

  p = 3;
  q = 1;

  //---------------------- MATRIZ M (Masas inerciales) ------------------------//
  M = zeros(p);
  M(1,1) = C*(cos(x(3))^2) + A*(cos(x(3)+x(5))^2) + 2*E*cos(x(3))*cos(x(3)+x(5));
  M(1,2) = 0;
  M(1,3) = 0;
  M(2,1) = 0;
  M(2,2) = C + A + 2*E*cos(x(5));
  M(2,3) = A + E*cos(x(5));
  M(3,1) = 0;
  M(3,2) = A + E*cos(x(5));
  M(3,3) = A;

  //------------ VECTOR V (Términos de Coriolis and Centrípetas) ------------//
  V = zeros(p,q);
  V(1) = -2*x(4)*D - 2*B*(x(4)+x(6)) - 2*x(2)*E*(x(4)*s223 + x(6)*cos(x(3))*sin(x(3)+x(5)));
  V(2) = (D+B)*x(2) + E*((x(2)^2)*s223 - (2*x(4)*x(6) + (x(6)^2))*sin(x(5)));
  V(3) = B*x(2) + (E*((x(2)^2)*cos(x(3))*sin(x(3)+x(5))+(x(4)^2)*sin(x(5))));
```

```
  //-------------------- VECTOR F (Fuerzas de fricción viscosa) ----------------------//
  Fv = zeros(p,q);
  Fv(1) = v(1)*x(2);
  Fv(2) = v(2)*x(4);
  Fv(3) = v(3)*x(6);

  //------------------------- VECTOR G (Gravedad)----------------------------//
  G = zeros(p,q);
  G(1) = 0;
  G(2) = (m(2)+m(3))*g*l(2)*cos(x(3)) + m(3)*g*l(3)*cos(x(3)+x(5));
  G(3) = m(3)*g*l(3)*cos(x(3)+x(5));

  //--------------------- VECTOR T (Pares de fuerza) ------------------------//
  Tin = [Tin1; Tin2; Tin3];

  //------------------- VECTOR TD (Perturbación externa) ---------------------//
  Td = [Td1; Td2; Td3];

  //-------------------- MATRIZ DE ACELERACION ANGULAR ----------------------//
  AA = inv(M)*(Tin - Td - V - Fv - G);

  //------------------- MODELO EN EL ESPACIO DE ESTADOS --------------------//
  dx(1) = x(2);     // Posición angular para el eslabón 1
  dx(2) = AA(1);  // Velocidad angular de la masa 1
  dx(3) = x(4);     // Posición angular para el eslabón 2
  dx(4) = AA(2);  // Velocidad angular de la masa 2
  dx(5) = x(6);     // Posición angular para el eslabón 3
  dx(6) = AA(3);  // Velocidad angular de la masa 3
endfunction
//---------------------------------------------------------------------------//

// Valores paramétricos:
m = [0.146; 0.147; 0.116];       // masas de la base y eslabones [kg]
l = [0.079; 0.113; 0.140];       // Altura de la base y longitud de eslabones [m]

// Posiciones cartesianas donde la trayectoria va a comenzar:
Px1 =  0.15;     // Posición inicial en el eje X [m]
Py1 = -0.15;     // Posición inicial en el eje Y [m]
Pz1 =  0.0;      // Posición inicial en el eje Z [m]

disp("POSICIÓN INICIAL PARA LA TRAYECTORIA:")
disp("Pi(x):")
disp(Px1)
disp("Pi(y):")
disp(Py1)
disp("Pi(z):")
disp(Pz1)

// Posiciones cartesianas donde la trayectoria va a terminar:
Px2 = 0.15;      // Posición final en el eje X [m]
Py2 = 0.15;      // Posición final en el eje Y [m]
Pz2 = 0.20;      // Posición final en el eje Z [m]

disp("POSICIÓN FINAL PARA LA TRAYECTORIA:")
disp("Pf(x):")
```

```
disp(Px2)
disp("Pf(y):")
disp(Py2)
disp("Pf(z):")
disp(Pz2)

// Constantes seleccionadas para el controlador Twisting (r1 > r2):
r1 = [2; 3; 2];
r2 = [1; 1; 1];

// Condiciones iniciales de la posición y velocidad angular:
w = sqrt(Pix^2 + Piy^2);
x1i = atan(Piy,Pix); // Condición inicial del primer ángulo

phi = atan(w,Piz-l(1));
r = sqrt(Pix^2 + Piy^2 + (Piz-l(1))^2);
beta = (%pi/2) - phi;
cg = ((r^2 + l(2)^2 - l(3)^2)/(2*l(2)*r));
sg = sqrt(1 - cg^2);
gamma = atan(sg,cg);
x3i = beta + gamma;  // Condición inicial del segundo ángulo

ca = ((l(2)^2 + l(3)^2 - r^2)/(2*l(2)*l(3)));
sa = sqrt(1 - ca^2);
alfa = atan(sa,ca);
x5i = -(%pi - alfa);  // Condición inicial del tercer ángulo

x0 = [0; 0; 0; 0; 0; 0];

qf = 1;

// Definición del tiempo:
t0 = 0;          // segundos
tf = 20;         // segundos
Dt = 1e-3;       // segundos
t = t0:Dt:tf;
lt = length(t);
tt = t0:Dt:(tf/2);
ltt = length(tt);

// Polinomio de 5to. grado:
Pt = ((qf)*((10*(tt^3/(tf/2)^3)) - (15*(tt^4/(tf/2)^4)) + (6*(tt^5/(tf/2)^5))));

// Condiciones iniciales para pares de fuerza:
Tin1 = zeros(1,lt);   // Servomotor 1 [N*m]
Tin2 = zeros(1,lt);   // Servomotor 2 [N*m]
Tin3 = zeros(1,lt);   // Servomotor 3 [N*m]

// Señal de perturbaciones consideradas:
Td1 = 1.8*sin(3*t^(4/6));
Td2 = 1.5*sin(3*t^(4/7));
Td3 = 1.5*sin(1*t^(4/5));

// Condiciones para la salida de referencia y su derivada:
yr1 = zeros(1,lt); // Salida de referencia para posición angular 1
yr2 = zeros(1,lt); // Salida de referencia para posición angular 2
yr3 = zeros(1,lt); // Salida de referencia para posición angular 3
```

```
yr1p = zeros(1,lt); // Derivada para la salida de referencia 1
yr2p = zeros(1,lt); // Derivada para la salida de referencia 2
yr3p = zeros(1,lt); // Derivada para la salida de referencia 3

// Condiciones de la variable deslizante
sigma1 = zeros(1,lt);   // Variable sigma1
sigma2 = zeros(1,lt);   // Variable sigma2
sigma3 = zeros(1,lt);   // Variable sigma3
sigma1p = zeros(1,lt); // Sigma1 punto (Derivada)
sigma2p = zeros(1,lt); // Sigma2 punto (Derivada)
sigma3p = zeros(1,lt); // Sigma3 punto (Derivada)

// Condiciones del error:
e1 = zeros(1,lt);
e2 = zeros(1,lt);
e3 = zeros(1,lt);

// Definición de valores iniciales para la posición y velocidad:
x10 = x0(1)*ones(1,lt);   // Posición angular 1
x20 = x0(2)*ones(1,lt);   // Velocidad angular 1
x30 = x0(3)*ones(1,lt);   // Posición angular 2
x40 = x0(4)*ones(1,lt);   // Velocidad angular 2
x50 = x0(5)*ones(1,lt);   // Posición angular 3
x60 = x0(6)*ones(1,lt);   // Velocidad angular 3

disp("Por favor espere 8 minutos....:/")

//-------- Generando trayectoria para usar como salida de referencia ---------//
theta1r = zeros(1,ltt); // Vector theta1 para referencia
theta2r = zeros(1,ltt); // Vector theta2 para referencia
theta3r = zeros(1,ltt); // Vector theta3 para referencia

i = 2;

for i=1:2
  if i == 1 then
    xx = (Px1 + ((Px2-Px1)/qf)*Pt);
    yy = (Py1 + ((Py2-Py1)/qf)*Pt);
    zz = (Pz1 + ((Pz2-Pz1)/qf)*Pt);
  else
    xx = (Px2 + ((Px1-Px2)/qf)*Pt);
    yy = (Py2 + ((Py1-Py2)/qf)*Pt);
    zz = (Pz2 + ((Pz1-Pz2)/qf)*Pt);
  end

  for k=1:ltt
    Px = xx(k);
    Py = yy(k);
    Pz = zz(k)-l(1);

    // Solución de la cinemática inversa:
    w = sqrt(Px^2 + Py^2);
    theta1 = atan(Py,Px); // Primer ángulo
    if i == 1 then
      theta1r(k) = theta1;
```

```
      else
        theta1r(k+ltt) = theta1;
      end

      phi = atan(w,Pz);
      r = sqrt(Px^2 + Py^2 + Pz^2);
      beta1 = (%pi/2) - phi;
      cg = ((r^2 + l(2)^2 - l(3)^2)/(2*l(2)*r));
      sg = sqrt(1 - cg^2);
      gama = atan(sg,cg);
      theta2 = beta1 + gama;  // Segundo ángulo
      if i == 1 then
        theta2r(k) = theta2;
      else
        theta2r(k+ltt) = theta2;
      end

      ca = ((l(2)^2 + l(3)^2 - r^2)/(2*l(2)*l(3)));
      sa = sqrt(1 - ca^2);
      alfa = atan(sa,ca);
      theta3 = -(%pi - alfa);  // Tercer ángulo
      if i == 1 then
        theta3r(k) = theta3;
      else
        theta3r(k+ltt) = theta3;
      end
    end
end
//-------------------------------------------------------------------------//

//------------- Implementación del controlador en lazo cerrado ---------------//
for k=1:lt-1
   // Variando parámetros de gravedad y fricción viscosa:
   if t(k)>=0 && t(k)<=5.0
      g = 9.81;      // Gravedad: 9.77 <= g <= 9.81 [m/s^2]
      v(1) = 0.0;    // Fricción viscosa 1: 0.0 <= fv <= 0.1772 [N*m/(rad/s)]
      v(2) = 0.0;    // Fricción viscosa 2: 0.0 <= fv <= 0.2428 [N*m/(rad/s)]
      v(3) = 0.0;    // Fricción viscosa 3: 0.0 <= fv <= 0.2428 [N*m/(rad/s)]
   elseif t(k)>5.0 && t(k)<=10.0
      g = 9.77;      // Variación de la gravedad
      v(1) = 0.001;  // Variación de la fricción viscosa 1
      v(2) = 0.005;  // Variación de la fricción viscosa 2
      v(3) = 0.003;  // Variación de la fricción viscosa 3
   elseif t(k)>10.0 && t(k)<=15.0
      g = 9.78;      // Variación de la gravedad
      v(1) = 0.05;   // Variación de la fricción viscosa 1
      v(2) = 0.09;   // Variación de la fricción viscosa 2
      v(3) = 0.1;    // Variación de la fricción viscosa 3
   elseif t(k)>15.0
      g = 9.81;      // Variación de la gravedad
      v(1) = 0.1772; // Variación de la fricción viscosa 1
      v(2) = 0.2428; // Variación de la fricción viscosa 2
      v(3) = 0.2428; // Variación de la fricción viscosa 3
//       Td1(k) = 0.5;  //  0.5
//       Td2(k) = 0.5;  //  0.5
//       Td3(k) = -0.5; // -0.5
   end
```

```
  // Definición de la salida de referencia:
  yr1(k) = theta1r(k)
  yr2(k) = theta2r(k)
  yr3(k) = theta3r(k)

  // Derivada de la salida de referencia:
  if k <= 1 then
     yr1p(k) = (yr1(k)/Dt)
     yr2p(k) = (yr2(k)/Dt)
     yr3p(k) = (yr3(k)/Dt)
  else
     yr1p(k) = ((yr1(k)-yr1(k-1))/Dt)
     yr2p(k) = ((yr2(k)-yr2(k-1))/Dt)
     yr3p(k) = ((yr3(k)-yr3(k-1))/Dt)
  end

  // Error de regulación y su derivada:
  e1(k) = yr1(k) - x10(k);
  e2(k) = yr2(k) - x30(k);
  e3(k) = yr3(k) - x50(k);

  e1p(k) = yr1p(k) - x20(k);
  e2p(k) = yr2p(k) - x40(k);
  e3p(k) = yr3p(k) - x60(k);

  // Estableciendo valores a la variable sigma y su derivada
  sigma1(k) = e1(k);
  sigma2(k) = e2(k);
  sigma3(k) = e3(k);

  sigma1p(k) = e1p(k);
  sigma2p(k) = e2p(k);
  sigma3p(k) = e3p(k);

  // Obtención de pares motor mediante ley de control Twisting:
  Tin1(k) = r1(1)*sign(sigma1(k)) + r2(1)*sign(sigma1p(k));
  Tin2(k) = r1(2)*sign(sigma2(k)) + r2(2)*sign(sigma2p(k));
  Tin3(k) = r1(3)*sign(sigma3(k)) + r2(3)*sign(sigma3p(k));

  // Resolviendo ecuación diferencial de primer orden:
  xOut = ode("stiff",[x10(k),x20(k),x30(k),x40(k),x50(k),x60(k)],t(k),t(k+1),[1e-6, 1e-6, 1e-6, 1e-6, 1e-
6, 1e-6],list(RobotSys2ODE,g,m,l,v,Tin1(k),Tin2(k),Tin3(k),Td1(k),Td2(k),Td3(k)));
  x10(k+1) = xOut(1);
  x20(k+1) = xOut(2);
  x30(k+1) = xOut(3);
  x40(k+1) = xOut(4);
  x50(k+1) = xOut(5);
  x60(k+1) = xOut(6);
end

disp("Proceso terminado! :)")
//---------------------------------------------------------------------------//
```

```
// Gráficas:
scf(0); clf();
subplot(3,1,1);
plot(t,Tin1,'-blue');
set(gca(),"grid",[1 1 1]);
title('Entrada de control: T1','FontSize',3);
xlabel('Tiempo [s]','FontSize',3);
ylabel('T1 [Nm]','FontSize',3);
xgrid;
subplot(3,1,2);
plot(t,Tin2,'-blue');
title('Entrada de control: T2','FontSize',3);
xlabel('Tiempo [s]','FontSize',3);
ylabel('T2 [Nm]','FontSize',3);
xgrid;
subplot(3,1,3);
plot(t,Tin3,'-blue');
title('Entrada de control: T3','FontSize',3);
xlabel('Tiempo [s]','FontSize',3);
ylabel('T3 [Nm]','FontSize',3);
xgrid;

scf(1); clf();
subplot(3,1,1);
plot(t,sigma1,'-blue',t,sigma1p,'-red');
title('Convergencia de σ1 and σ1p','FontSize',3);
xlabel('Tiempo [s]','FontSize',3);
ylabel('Convergencia','FontSize',3);
legend('σ1','σ1p');
xgrid;
subplot(3,1,2);
plot(t,sigma2,'-blue',t,sigma2p,'-red');
title('Convergencia de σ2 and σ2p','FontSize',3);
xlabel('Tiempo [s]','FontSize',3);
ylabel('Convergencia','FontSize',3);
legend('σ2','σ2p');
xgrid;
subplot(3,1,3);
plot(t,sigma3,'-blue',t,sigma3p,'-red');
title('Convergencia de σ3 and σ3p','FontSize',3);
xlabel('Tiempo [s]','FontSize',3);
ylabel('Convergencia','FontSize',3);
legend('σ3','σ3p');
xgrid;

scf(2); clf();
subplot(3,1,1);
plot(t,yr1*(180/%pi),'-blue',t,x10*(180/%pi),'--red');
title('Salida de referencia y medida - 1er articulación','FontSize',3);
xlabel('Tiempo [s]','FontSize',3);
ylabel('Grados','FontSize',3);
legend('Referencia','Medida');
xgrid;
subplot(3,1,2);
plot(t,yr2*(180/%pi),'-blue',t,x30*(180/%pi),'--red');
```

```
title('Salida de referencia y medida - 2da articulación','FontSize',3);
xlabel('Tiempo [s]','FontSize',3);
ylabel('Grados','FontSize',3);
legend('Referencia','Medida');
xgrid;
subplot(3,1,3);
plot(t,yr3*(180/%pi),'-blue',t,x50*(180/%pi),'--red');
title('Salida de referencia y medida - 3er articulación','FontSize',3);
xlabel('Tiempo [s]','FontSize',3);
ylabel('Grados','FontSize',3);
legend('Referencia','Medida');
xgrid;

scf(3); clf();
subplot(3,1,1);
plot(t,e1,'-black');
title('Error de salida - 1er. articulación','FontSize',3);
xlabel('Tiempo [s]','FontSize',3);
ylabel('Error','FontSize',3);
xgrid;
subplot(3,1,2);
plot(t,e2,'-black');
title('Error de salida - 2da. articulación','FontSize',3);
xlabel('Tiempo [s]','FontSize',3);
ylabel('Error','FontSize',3);
xgrid;
subplot(3,1,3);
plot(t,e3,'-black');
title('Error de salida - 3er. articulación','FontSize',3);
xlabel('Tiempo [s]','FontSize',3);
ylabel('Error','FontSize',3);
xgrid;

scf(4); clf();
subplot(3,1,1);
plot(t,Td1,'-blue');
title('Señal de perturbación - 1er. articulación','FontSize',3);
xlabel('Tiempo [s]','FontSize',3);
ylabel('Tau [Nm]','FontSize',3);
xgrid;
subplot(3,1,2);
plot(t,Td2,'-blue');
title('Señal de perturbación - 2da. articulación','FontSize',3);
xlabel('Tiempo [s]','FontSize',3);
ylabel('Tau [Nm]','FontSize',3);
xgrid;
subplot(3,1,3);
plot(t,Td3,'-blue');
title('Señal de perturbación - 3er. articulación','FontSize',3);
xlabel('Tiempo [s]','FontSize',3);
ylabel('Tau [Nm]','FontSize',3);
xgrid;
```

Apéndice K

Simulación del 3-AST anti-windup

```
/// UNIVERSIDAD DE GUANAJUATO - DIVISION DE INGENIERIAS CAMPUS IRAPUATO - SALAMANCA ///
/////// MAESTRIA EN INGENIERÍA ELÉCTRICA - INSTRUMENTACIÓN Y SISTEMAS DIGITALES ///////
/ SIMULACIÓN DEL CONTROLADOR 3-AST ANTI-WINDUP CON MODELO DINÁMICO DEL BRAZO ROBÓTICO /
////////////////////// PROGRAMADO POR: CESAR EDUARDO CONEJO BENITEZ ///////////////////

clc;
clear;
xdel(winsid());

//------------------------------------------------- Sistema dinámico no lineal -------------------------------------------------//
function dx=RobotSys2ODE(t, x, g, m, l, v, Tin1, Tin2, Tin3, Td1, Td2, Td3)
  A = m(3)*(l(3)^2);
  B = m(3)*(l(3)^2)*x(2)*cos(x(3)+x(5))*sin(x(3)+x(5));
  C = (m(2) + m(3))*(l(2)^2);
  D = (m(2) + m(3))*(l(2)^2)*x(2)*cos(x(3))*sin(x(3));
  E = m(3)*l(2)*l(3);
  F = m(3)*g*l(3)*cos(x(3)+x(5));
  s223 = sin((2*x(3)) + x(5));

  p = 3;
  q = 1;

  //---------------------------------------- MATRIZ M (Masas inerciales) ----------------------------------------//
  M = zeros(p);
  M(1,1) = C*(cos(x(3))^2) + A*(cos(x(3)+x(5))^2) + 2*E*cos(x(3))*cos(x(3)+x(5));
  M(1,2) = 0;
  M(1,3) = 0;
  M(2,1) = 0;
  M(2,2) = C + A + 2*E*cos(x(5));
  M(2,3) = A + E*cos(x(5));
  M(3,1) = 0;
  M(3,2) = A + E*cos(x(5));
  M(3,3) = A;

  //------------------------- VECTOR V (Términos de Coriolis and Centrípetas) -------------------------//
  V = zeros(p,q);
  V(1) = -2*x(4)*D - 2*B*(x(4)+x(6)) - 2*x(2)*E*(x(4)*s223 + x(6)*cos(x(3))*sin(x(3)+x(5)));
  V(2) = (D+B)*x(2) + E*((x(2)^2)*s223 - (2*x(4)*x(6) + (x(6)^2))*sin(x(5)));
  V(3) = B*x(2) + (E*((x(2)^2)*cos(x(3))*sin(x(3)+x(5))+(x(4)^2)*sin(x(5))));
```

```
  //------------------------------------ VECTOR F (Fuerzas de fricción) ------------------------------------//
  FV = zeros(p,q);
  FV(1) = v(1)*x(2);
  FV(2) = v(2)*x(4);
  FV(3) = v(3)*x(6);

  //-------------------------------------------- VECTOR G (Gravedad) --------------------------------------------//
  G = zeros(p,q);
  G(1) = 0;
  G(2) = (m(2)+m(3))*g*l(2)*cos(x(3)) + m(3)*g*l(3)*cos(x(3)+x(5));
  G(3) = m(3)*g*l(3)*cos(x(3)+x(5));

  //------------------------------------ VECTOR Tin (Pares motor) ------------------------------------//
  Tin = [Tin1; Tin2; Tin3];

  //------------------------------------ VECTOR Td (Cargas aplicadas) ------------------------------------//
  Td = [Td1; Td2; Td3];

  //---------------------------- MATRIZ DE ACELERACION ANGULAR (AA) ----------------------------//
  AA = inv(M)*(Tin + Td - V - FV - G);

  //------------------------------ MODELO EN EL ESPACIO DE ESTADOS ------------------------------//
  dx(1) = x(2);    // Posición angular para el servomotor 1
  dx(2) = AA(1);  // Velocidad angular para el servomotor 1
  dx(3) = x(4);    // Posición angular para el servomotor 2
  dx(4) = AA(2);  // Velocidad angular para el servomotor 2
  dx(5) = x(6);    // Posición angular para el servomotor 3
  dx(6) = AA(3);  // Velocidad angular para el servomotor 3
endfunction
//-------------------------------------------------------------------------------------------------------------------------------//

//- Funciones para la derivada negativa definida: L1 punto -//
function dx=L1p_ODE(t, x, k3, sigma1)
  dx(1) = k3(1)*sign(sigma1);
endfunction

function dx=L1p_Gaw_ODE(t, x, Gaw, Tsel1, Tin1)
  dx(1) = Gaw(1)*(Tsel1 - Tin1);
endfunction
//-------------------------------------------------------------------------//

//- Funciones para la derivada negativa definida: L2 punto -//
function dx=L2p_ODE(t, x, k3, sigma2)
  dx(1) = k3(2)*sign(sigma2);
endfunction

function dx=L2p_Gaw_ODE(t, x, Gaw, Tsel2, Tin2)
  dx(1) = Gaw(2)*(Tsel2 - Tin2);
endfunction
//-------------------------------------------------------------------------//

//- Funciones para la derivada negativa definida: L3 punto -//
function dx=L3p_ODE(t, x, k3, sigma3)
  dx(1) = k3(3)*sign(sigma3);
endfunction
```

```
function dx=L3p_Gaw_ODE(t, x, Gaw, Tsel3, Tin3)
  dx(1) = Gaw(3)*(Tsel3 - Tin3);
endfunction
//---------------------------------------------------------------------------//

// Valores paramétricos:
m = [0.146; 0.147; 0.116];       // masa de la base y de los eslabones 1 y 2 [kg]
l = [0.079; 0.113; 0.140];       // Altura de la base y longitud de los eslabones 1 y 2 [m]

// Coordenadas del primer punto para la trayectoria:
P1x =  0.15;   // Coordenada en X del primer punto [m]
P1y = -0.15;   // Coordenada en Y del primer punto [m]
P1z =  0.05;   // Coordenada en Z del primer punto [m]

disp("PRIMER PUNTO:")
disp("P1(x):")
disp(P1x)
disp("P1(y):")
disp(P1y)
disp("P1(z):")
disp(P1z)

// Coordenadas del segundo punto para la trayectoria:
P2x = 0.15;    // Coordenada en X del segundo punto [m]
P2y = 0.15;    // Coordenada en Y del segundo punto [m]
P2z = 0.20;    // Coordenada en Z del segundo punto [m]

disp("SEGUNDO PUNTO:")
disp("P2(x):")
disp(P2x)
disp("P2(y):")
disp(P2y)
disp("P2(z):")
disp(P2z)

// Valores seleccionados para las ganancias del 3-AST:
k1 = [4; 3; 2];   // Para regular la posición más rápido
k2 = [4; 4; 4];   // Para tener un mejor control de la velocidad
k3 = [2; 2; 2];   // Para rechazar la perturbación

// Ganancias del Anti-windup:
Gaw = [20; 400; 400];

//------------ Condiciones iniciales de la posición y velocidad angular ------------//
w = sqrt(P1x^2 + P1y^2);
x1i = atan(P1y,P1x); // Condición inicial de posición angular para servo 1

phi = atan(w,P1z-l(1));
r = sqrt(P1x^2 + P1y^2 + (P1z-l(1))^2);
beta = (%pi/2) - phi;
cg = ((r^2 + l(2)^2 - l(3)^2)/(2*l(2)*r));
sg = sqrt(1 - cg^2);
gamma = atan(sg,cg);
x3i = beta + gamma; // Condición inicial de posición angular para servo 2
```

```
ca = ((l(2)^2 + l(3)^2 - r^2)/(2*l(2)*l(3)));
sa = sqrt(1 - ca^2);
alfa = atan(sa,ca);
x5i = -(%pi - alfa); // Condición inicial de posición angular para servo 3

x0 = [x1i; 0; x3i; 0; x5i; 0]; // Vector de condiciones iniciales (pos., vel.)
//-----------------------------------------------------------------------------------------------//

qf = 1;

// Definición del tiempo:
t0 = 0;        // Tiempo inicial [s]
tf = 20;       // Tiempo final [s]
Dt = 1e-4;   //Puntos intermedios
t = t0:Dt:tf;
lt = length(t);
tt = t0:Dt:(tf/2);
ltt = length(tt);  //Auxiliar para generación de trayectoria

// Polinomio de 5to. grado:
Pt = qf*(10*(tt^3/(tf/2)^3) - 15*(tt^4/(tf/2)^4) + 6*(tt^5/(tf/2)^5));

// Condiciones iniciales para los pares motor:
Tin1 = zeros(1,lt);    // Servomotor 1 [N*m]
Tin2 = zeros(1,lt);    // Servomotor 2 [N*m]
Tin3 = zeros(1,lt);    // Servomotor 3 [N*m]

// Condiciones iniciales para las señales de perturbación:
Td1 = zeros(1,lt);
Td2 = zeros(1,lt);
Td3 = zeros(1,lt);

// Condiciones iniciales para las salidas de referencia y sus derivadas:
yr1 = zeros(1,lt);     // Salida de referencia para posición angular 1
yr2 = zeros(1,lt);     // Salida de referencia para posición angular 2
yr3 = zeros(1,lt);     // Salida de referencia para posición angular 3
yr1p = zeros(1,lt);  // Derivada para la salida de referencia 1
yr2p = zeros(1,lt);  // Derivada para la salida de referencia 2
yr3p = zeros(1,lt);  // Derivada para la salida de referencia 3

// Condiciones iniciales para las variables deslizantes:
sigma1 = zeros(1,lt); // Variable deslizante: sigma1
sigma2 = zeros(1,lt); // Variable deslizante: sigma2
sigma3 = zeros(1,lt); // Variable deslizante: sigma3
sigma1p = zeros(1,lt); // Derivada de la variable deslizante sigma1
sigma2p = zeros(1,lt); // Derivada de la variable deslizante sigma2
sigma3p = zeros(1,lt); // Derivada de la variable deslizante sigma3

// Condiciones iniciales para los errores de regulación:
e1 = zeros(1,lt);
e2 = zeros(1,lt);
e3 = zeros(1,lt);
```

```
// Condiciones iniciales para los controles auxiliares Li:
L1 = zeros(1,lt)
L2 = zeros(1,lt)
L3 = zeros(1,lt)

// Definición de valores iniciales para la posición y velocidad angular:
x10 = x0(1)*ones(1,lt);   // Posición angular – servomotor 1
x20 = x0(2)*ones(1,lt);   // Velocidad angular – servomotor 1
x30 = x0(3)*ones(1,lt);   // Posición angular – servomotor 2
x40 = x0(4)*ones(1,lt);   // Velocidad angular – servomotor 2
x50 = x0(5)*ones(1,lt);   // Posición angular – servomotor 3
x60 = x0(6)*ones(1,lt);   // Velocidad angular – servomotor 3

// Definir el tamaño de la tabla que contendrá a las pos. angulares y torques
// que se obtengan del controlador:
Tabla = zeros(lt, 7)

disp("Por favor espere 14 minutos.....:/")

//-------- Generando trayectorias para usarlas como salidas de referencia ---------//
theta1r = zeros(1,ltt); // Vector theta1 para guardar las posiciones de referencia
theta2r = zeros(1,ltt); // Vector theta2 para guardar las posiciones de referencia
theta3r = zeros(1,ltt); // Vector theta3 para guardar las posiciones de referencia

i = 2;

for i=1:2
  if i == 1 then
    xx = (P1x + ((P2x-P1x)/qf)*Pt);
    yy = (P1y + ((P2y-P1y)/qf)*Pt);
    zz = (P1z + ((P2z-P1z)/qf)*Pt);
  else
    xx = (Px2 + ((Px1-Px2)/qf)*Pt);
    yy = (Py2 + ((Py1-Py2)/qf)*Pt);
    zz = (Pz2 + ((Pz1-Pz2)/qf)*Pt);
  end

  for k=1:ltt
    Px = xx(k);
    Py = yy(k);
    Pz = zz(k)-l(1);

    // Cinemática inversa:
    w = sqrt(Px^2 + Py^2);
    theta1 = atan(Py,Px); // Primer ángulo
    if i == 1 then
      theta1r(k) = theta1;
    else
      theta1r(k+ltt) = theta1;
    end

    phi = atan(w,Pz);
    r = sqrt(Px^2 + Py^2 + Pz^2);
    beta1 = (%pi/2) - phi;
    cg = ((r^2 + l(2)^2 - l(3)^2)/(2*l(2)*r));
```

```
    sg = sqrt(1 - cg^2);
    gama = atan(sg,cg);
    theta2 = beta1 + gama;  // Segundo ángulo
    if i == 1 then
      theta2r(k) = theta2;
    else
      theta2r(k+ltt) = theta2;
    end

    ca = ((l(2)^2 + l(3)^2 - r^2)/(2*l(2)*l(3)));
    sa = sqrt(1 - ca^2);
    alfa = atan(sa,ca);
    theta3 = -(%pi - alfa);  // Tercer ángulo
    if i == 1 then
      theta3r(k) = theta3;
    else
      theta3r(k+ltt) = theta3;
    end
  end
end
//-------------------------------------------------------------------------------------------------//

//------------------------------ Implementación del controlador en lazo cerrado -------------------------------//
for k=1:lt-1
  // Variando parámetros de gravedad, fricción viscosa y perturbaciones:
  if t(k)>=0 && t(k)<=5.0
    g = 9.81;          // Gravedad: 9.78 <= g <= 9.81 [m/s^2]
    v(1) = 0.0;        // Fricción viscosa 1: 0.0 <= fv <= 0.1772 [N*m/(rad/s)]
    v(2) = 0.0;        // Fricción viscosa 2: 0.0 <= fv <= 0.2428 [N*m/(rad/s)]
    v(3) = 0.0;        // Fricción viscosa 3: 0.0 <= fv <= 0.2428 [N*m/(rad/s)]
    Td1(k) = -0.5;
    Td2(k) = -0.5;
    Td3(k) = 0.5;
  elseif t(k)>5.0 && t(k)<=10.0
    g = 9.77;          // Variación de la gravedad
    v(1) = 0.001;      // Variación de la fricción viscosa 1
    v(2) = 0.005;      // Variación de la fricción viscosa 2
    v(3) = 0.003;      // Variación de la fricción viscosa 3
    Td1(k) = -1.7;
    Td2(k) = 1.5;
    Td3(k) = -1.2;
  elseif t(k)>10.0 && t(k)<=15.0
    g = 9.78;          // Variación de la gravedad
    v(1) = 0.05;       // Variación de la fricción viscosa 1
    v(2) = 0.09;       // Variación de la fricción viscosa 2
    v(3) = 0.1;        // Variación de la fricción viscosa 3
    Td1(k) = 1.7;
    Td2(k) = -0.9;
    Td3(k) = 1.4;
  elseif t(k)>15.0
    g = 9.81;          // Variación de la gravedad
    v(1) = 0.1772;     // Variación de la fricción viscosa 1
    v(2) = 0.2428;     // Variación de la fricción viscosa 2
    v(3) = 0.2428;     // Variación de la fricción viscosa 3
    Td1(k) = 0.5;
```

```
    Td2(k) = 0.5;
    Td3(k) = -0.5;
  end

  // Definición de la salida de referencia:
  yr1(k) = theta1r(k)
  yr2(k) = theta2r(k)
  yr3(k) = theta3r(k)

  // Derivada de la salida de referencia:
  if k <= 1 then
    yr1p(k) = (yr1(k)/Dt)
    yr2p(k) = (yr2(k)/Dt)
    yr3p(k) = (yr3(k)/Dt)
  else
    yr1p(k) = ((yr1(k)-yr1(k-1))/Dt)
    yr2p(k) = ((yr2(k)-yr2(k-1))/Dt)
    yr3p(k) = ((yr3(k)-yr3(k-1))/Dt)
  end

  // Error de regulación y su derivada:
  e1(k) = yr1(k) - x10(k);
  e2(k) = yr2(k) - x30(k);
  e3(k) = yr3(k) - x50(k);

  e1p(k) = yr1p(k) - x20(k);
  e2p(k) = yr2p(k) - x40(k);
  e3p(k) = yr3p(k) - x60(k);

  // Definición de la variable deslizante sigma:
  sigma1(k) = e1p(k)+(k2(1)*(abs(e1(k))^(2/3))*sign(e1(k)));
  sigma2(k) = e2p(k)+(k2(2)*(abs(e2(k))^(2/3))*sign(e2(k)));
  sigma3(k) = e3p(k)+(k2(3)*(abs(e3(k))^(2/3))*sign(e3(k)));

  // Obtención de la derivada de sigma:
  if k <= 1 then
    sigma1p(k) = (sigma1(k)/Dt)
    sigma2p(k) = (sigma2(k)/Dt)
    sigma3p(k) = (sigma3(k)/Dt)
  else
    sigma1(k-1) = e1p(k-1)+(k2(1)*(abs(e1(k-1))^(2/3))*sign(e1(k-1)));
    sigma2(k-1) = e2p(k-1)+(k2(2)*(abs(e2(k-1))^(2/3))*sign(e2(k-1)));
    sigma3(k-1) = e3p(k-1)+(k2(3)*(abs(e3(k-1))^(2/3))*sign(e3(k-1)));
    sigma1p(k) = ((sigma1(k)-sigma1(k-1))/Dt)
    sigma2p(k) = ((sigma2(k)-sigma2(k-1))/Dt)
    sigma3p(k) = ((sigma3(k)-sigma3(k-1))/Dt)
  end

  // Obtención de pares motor mediante la ley de control 3-AST:
  if t(k) == 0 then
    Tin1(k) = 1.8;
    Tin2(k) = 1.5;
    Tin3(k) = 1.5;
  else
    Tin1(k) = k1(1)*(abs(sigma1(k))^(1/2))*sign(sigma1(k)) + L1(k);
```

```
    Tin2(k) = k1(2)*(abs(sigma2(k))^(1/2))*sign(sigma2(k)) + L2(k);
    Tin3(k) = k1(3)*(abs(sigma3(k))^(1/2))*sign(sigma3(k)) + L3(k);
end

// Comparando la entrada Tin1 con los limites:
if Tin1(k) > Tmax(1) then
    Tsel1 = Tmax(1);
    L1pi = ode("stiff", L1(k), t(k), t(k+1), 1e-6, list(L1p_Gaw_ODE,Gaw,Tsel1,Tin1(k)));
elseif Tin1(k) < Tmin(1)
    Tsel1 = Tmin(1);
    L1pi = ode("stiff", L1(k), t(k), t(k+1), 1e-6, list(L1p_Gaw_ODE,Gaw,Tsel1,Tin1(k)));
elseif Tin1(k) >= Tmin(1) && Tin1(k) <= Tmax(1)
    L1pi = ode("stiff", L1(k), t(k), t(k+1), 1e-6, list(L1p_ODE,k3,sigma1(k)));
end
L1(k+1) = L1pi;

// Comparando la entrada Tin2 con los limites:
if Tin2(k) > Tmax(2) then
    Tsel2 = Tmax(2);
    L2pi = ode("stiff", L2(k), t(k), t(k+1), 1e-6, list(L2p_Gaw_ODE,Gaw,Tsel2,Tin2(k)));
elseif Tin2(k) < Tmin(2)
    Tsel2 = Tmin(2);
    L2pi = ode("stiff", L2(k), t(k), t(k+1), 1e-6, list(L2p_Gaw_ODE,Gaw,Tsel2,Tin2(k)));
elseif Tin2(k) >= Tmin(2) && Tin2(k) <= Tmax(2)
    L2pi = ode("stiff", L2(k), t(k), t(k+1), 1e-6, list(L2p_ODE,k3,sigma2(k)));
end
L2(k+1) = L2pi;

// Comparando la entrada Tin3 con los limites:
if Tin3(k) > Tmax(3) then
    Tsel3 = Tmax(3);
    L3pi = ode("stiff", L3(k), t(k), t(k+1), 1e-6, list(L3p_Gaw_ODE,Gaw,Tsel3,Tin3(k)));
elseif Tin3(k) < Tmin(3)
    Tsel3 = Tmin(3);
    L3pi = ode("stiff", L3(k), t(k), t(k+1), 1e-6, list(L3p_Gaw_ODE,Gaw,Tsel3,Tin3(k)));
elseif Tin3(k) >= Tmin(3) && Tin3(k) <= Tmax(3)
    L3pi = ode("stiff", L3(k), t(k), t(k+1), 1e-6, list(L3p_ODE,k3,sigma3(k)));
end
L3(k+1) = L3pi;

// Resolviendo ecuación diferencial de primer orden:
xOut = ode("stiff",[x10(k),x20(k),x30(k),x40(k),x50(k),x60(k)],t(k),t(k+1),[1e-6, 1e-6, 1e-6, 1e-6,
1e-6, 1e-6],list(RobotSys2ODE,g,m,l,v,Tin1(k),Tin2(k),Tin3(k),Td1(k),Td2(k),Td3(k)));
x10(k+1) = xOut(1);
x20(k+1) = xOut(2);
x30(k+1) = xOut(3);
x40(k+1) = xOut(4);
x50(k+1) = xOut(5);
x60(k+1) = xOut(6);

// Guardar los valores de las pos. angulares y los torques que se vayan obteniendo:
Tabla(k,1) = Tin1(k);                  // Torque para el servomotor ID: 1
Tabla(k,2) = x10(k)*(180/%pi);         // Pos. ang. para el servomotor ID: 1
Tabla(k,3) = Tin2(k);                  // Torque para el servomotor ID: 2
Tabla(k,4) = x30(k)*(180/%pi);         // Pos. ang. para el servomotor ID: 2
```

```
    Tabla(k,5) = Tin3(k);                  // Torque para el servomotor ID: 3
    Tabla(k,6) = x50(k)*(180/%pi);         // Pos. ang. para el servomotor ID: 3
    Tabla(k,7) = t(k);                     // Tiempo transcurrido
end

disp("Proceso terminado! :)")
//-----------------------------------------------------------------------------------------------------------------------------------//

// Crear archivo csv con los valores que fueron guardados en la tabla:
filename = fullfile(TMPDIR, "valores_3-AST_Puls.csv");
comments = [
"tau 1, theta 1, tau 2, theta 2, tau 3, theta 3, tiempo"
];
csvWrite(Tabla, filename, [], [], [], comments)

// Graficas:
scf(0); clf();
subplot(3,1,1);
plot(t,Tin1,'-blue');
title('$Entrada \hspace{0.2cm} de \hspace{0.2cm} control: \tau_{in_1}$','FontSize',3);
xlabel('$Tiempo \hspace{0.1cm} [s]$','FontSize',3);
ylabel('$\tau_{in_1} \hspace{0.1cm} [Nm]$','FontSize',3);
xgrid;
subplot(3,1,2);
plot(t,Tin2,'-blue');
title('$Entrada \hspace{0.2cm} de \hspace{0.2cm} control: \tau_{in_2}$','FontSize',3);
xlabel('$Tiempo \hspace{0.1cm} [s]$','FontSize',3);
ylabel('$\tau_{in_2} \hspace{0.1cm} [Nm]$','FontSize',3);
xgrid;
subplot(3,1,3);
plot(t,Tin3,'-blue');
title('$Entrada \hspace{0.2cm} de \hspace{0.2cm} control: \tau_{in_3}$','FontSize',3);
xlabel('$Tiempo \hspace{0.1cm} [s]$','FontSize',3);
ylabel('$\tau_{in_3} \hspace{0.1cm} [Nm]$','FontSize',3);
xgrid;

scf(1); clf();
subplot(3,1,1);
plot(t,sigma1,'-blue',t,sigma1p,'-red');
set(gca(),"grid",[1 1 1]);
set(gca(),"data_bounds", matrix([-t0,tf,-2,2],2,-1));
title('$Convergencia \hspace{0.2cm} de \hspace{0.2cm} σ_1 \hspace{0.2cm} y \hspace{0.2cm}
\dot{σ}_1$','FontSize',3);
xlabel('$Tiempo \hspace{0.1cm} [s]$','FontSize',3);
ylabel('$Convergencia$','FontSize',3);
legend('$\sigma_{1}$','$\dot{\sigma}_{1}$');
xgrid;
subplot(3,1,2);
plot(t,sigma2,'-blue',t,sigma2p,'-red');
set(gca(),"grid",[1 1 1]);
set(gca(),"data_bounds", matrix([-t0,tf,-100,100],2,-1));
title('$Convergencia \hspace{0.2cm} de \hspace{0.2cm} σ_2 \hspace{0.2cm} y \hspace{0.2cm}
\dot{σ}_2$','FontSize',3);
xlabel('$Tiempo \hspace{0.1cm} [s]$','FontSize',3);
ylabel('$Convergencia$','FontSize',3);
```

```
legend('$\sigma_{2}$','$\dot{\sigma}_{2}$');
xgrid;
subplot(3,1,3);
plot(t,sigma3,'-blue',t,sigma3p,'-red');
set(gca(),"grid",[1 1 1]);
set(gca(),"data_bounds", matrix([-t0,tf,-50,50],2,-1));
title('$Convergencia \hspace{0.2cm} de \hspace{0.2cm} σ_3 \hspace{0.2cm} y \hspace{0.2cm}
\dot{σ}_3$','FontSize',3);
xlabel('$Tiempo \hspace{0.1cm} [s]$','FontSize',3);
ylabel('$Convergencia$','FontSize',3);
legend('$\sigma_{3}$','$\dot{\sigma}_{3}$');
xgrid;

scf(2); clf();
subplot(3,1,1);
plot(t,yr1*(180/%pi),'-blue',t,x10*(180/%pi),'--red');
title('$Perfil \hspace{0.2cm} de \hspace{0.2cm} referencia \hspace{0.2cm} y \hspace{0.2cm} salida
\hspace{0.2cm} medida - 1ra. \hspace{0.1cm} articulación$','FontSize',3);
xlabel('$Tiempo \hspace{0.1cm} [s]$','FontSize',3);
ylabel('$Grados$','FontSize',3);
legend('$y_{1}$','$y_{r_1}$');
xgrid;
subplot(3,1,2);
plot(t,yr2*(180/%pi),'-blue',t,x30*(180/%pi),'--red');
title('$Perfil \hspace{0.2cm} de \hspace{0.2cm} referencia \hspace{0.2cm} y \hspace{0.2cm} salida
\hspace{0.2cm} medida - 2da. \hspace{0.1cm} articulación$','FontSize',3);
xlabel('$Tiempo \hspace{0.1cm} [s]$','FontSize',3);
ylabel('$Grados$','FontSize',3);
legend('$y_{2}$','$y_{r_2}$');
xgrid;
subplot(3,1,3);
plot(t,yr3*(180/%pi),'-blue',t,x50*(180/%pi),'--red');
title('$Perfil \hspace{0.2cm} de \hspace{0.2cm} referencia \hspace{0.2cm} y \hspace{0.2cm} salida
\hspace{0.2cm} medida - 3ra. \hspace{0.1cm} articulación$','FontSize',3);
xlabel('$Tiempo \hspace{0.1cm} [s]$','FontSize',3);
ylabel('$Grados$','FontSize',3);
legend('$y_{3}$','$y_{r_3}$');
xgrid;

scf(3); clf();
subplot(3,1,1);
plot(t,e1,'-black');
title('$Error \hspace{0.2cm} de \hspace{0.2cm} regulación - 1ra. \hspace{0.1cm}
articulación$','FontSize',3);
xlabel('$Tiempo \hspace{0.1cm} [s]$','FontSize',3);
ylabel('$Error$','FontSize',3);
xgrid;
subplot(3,1,2);
plot(t,e2,'-black');
title('$Error \hspace{0.2cm} de \hspace{0.2cm} regulación - 2da. \hspace{0.1cm}
articulación$','FontSize',3);
xlabel('$Tiempo \hspace{0.1cm} [s]$','FontSize',3);
ylabel('$Error$','FontSize',3);
xgrid;
subplot(3,1,3);
```

```
plot(t,e3,'-black');
title('$Error \hspace{0.2cm} de \hspace{0.2cm} regulación - 3ra. \hspace{0.1cm}
articulación$','FontSize',3);
xlabel('$Tiempo \hspace{0.1cm} [s]$','FontSize',3);
ylabel('$Error$','FontSize',3);
xgrid;

scf(4); clf();
subplot(3,1,1);
plot(t,Td1,'-blue');
title('$Señal \hspace{0.2cm} de \hspace{0.2cm} perturbación - 1ra. \hspace{0.1cm}
articulación$','FontSize',3);
xlabel('$Tiempo \hspace{0.1cm} [s]$','FontSize',3);
ylabel('$\tau_{d_1} \hspace{0.1cm} [Nm]$','FontSize',3);
xgrid;
subplot(3,1,2);
plot(t,Td2,'-blue');
title('$Señal \hspace{0.2cm} de \hspace{0.2cm} perturbación - 2da. \hspace{0.1cm}
articulación$','FontSize',3);
xlabel('$Tiempo \hspace{0.1cm} [s]$','FontSize',3);
ylabel('$\tau_{d_2} \hspace{0.1cm} [Nm]$','FontSize',3);
xgrid;
subplot(3,1,3);
plot(t,Td3,'-blue');
title('$Señal \hspace{0.2cm} de \hspace{0.2cm} perturbación - 3ra. \hspace{0.1cm}
articulación$','FontSize',3);
xlabel('$Tiempo \hspace{0.1cm} [s]$','FontSize',3);
ylabel('$\tau_{d_3} \hspace{0.1cm} [Nm]$','FontSize',3);
xgrid;
```

Apéndice L

Escritura de valores obtenidos de SciLab

Para este programa se usaron las mismas funciones de control que fueron explicadas en el apéndice H, por lo tanto, la sintaxis para este programa es similar al anterior.

Al inicio del programa se escribe la instrucción que fue mostrada en la Figura H.1, la cual permite leer caracteres individuales de la información que es producida por el usuario. Posteriormente, se establece la importación de las funciones de control que proporciona DYNAMIXEL SDK, así como de la función que permite realizar operaciones matemáticas, la función de tiempo, la función que permite leer y escribir archivos CSV y la función que permite la creación de gráficas. Para el diseño de las gráficas se definen a los vectores que permitirán almacenar las lecturas de posición angular y carga motor actual de cada servomotor. Vea la Figura L.1.

```
# Importación de funciones:
from dynamixel_sdk import *                 # Funciones para el control de los Dynamixel
import math as m                            # Funciones matemáticas
import csv                                  # Funciones que permite leer y escribir archivos.csv
import time                                 # Función tiempo
from matplotlib import pyplot as plt        # Funciones para crear gráficas

# Vectores para almacenar las lecturas de los servomotores:
theta1 = []
theta2 = []
theta3 = []
carga1 = []
carga2 = []
carga3 = []
carga4 = []
t      = []
```

Figura L.1: Importación de funciones y definición de vectores

Enseguida se define una función general que permitirá convertir los valores de posición angular y par motor (que hayan sido leídos del archivo CSV) a valores enteros (entre 0 y 1023); también permitirá escribir en cada servomotor los valores de par motor y de posición angular que deberán tomar en un instante de tiempo, permitiendo además la operación simultánea de los Dynamixel. Vea la Figura L.2.

```
# Función que permitirá la escritura del par motor y la posición angular en cada servomotor:
def MovServosSync(TAU1,THETA1,TAU2,THETA2,TAU3,THETA3,TAU4,PINZA):
    # Convertir valores del par motor y la pos. angular a enteros (0 a 1023):
    DXL1_VAL_TORQ  = round(abs(TAU1)*(1023/1.8))          # Par motor para 1er. servomotor
    DXL1_VAL_POS   = round((THETA1 + 150)*(1023/300))     # Pos. angular para 1er. servomotor

    DXL2_VAL_TORQ  = round(abs(TAU2)*(1023/1.5))          # Par motor para 2do. servomotor
    DXL2_VAL_POS   = round((240 - (THETA2))*(1023/300))   # Pos. angular para 2do. servomotor

    DXL3_VAL_TORQ  = round(abs(TAU3)*(1023/1.5))          # Par motor para 3er. servomotor
    DXL3_VAL_POS   = round((THETA3 + 150)*(1023/300))     # Pos. angular para 3er. servomotor

    DXL4_VAL_TORQ  = round(abs(TAU4)*(1023/1.5))          # Par motor para 4to. servomotor
    DXL4_VAL_POS   = round((PINZA)*(1023/300))            # Pos. angular para 4to. servomotor

    # Escritura del par motor asignado para cada servomotor:
    packetHandler.write2ByteTxRx(portHandler, DXL1_ID, ADDR_MX_TORQUE_LIMIT, DXL1_VAL_TORQ)
    packetHandler.write2ByteTxRx(portHandler, DXL2_ID, ADDR_MX_TORQUE_LIMIT, DXL2_VAL_TORQ)
    packetHandler.write2ByteTxRx(portHandler, DXL3_ID, ADDR_MX_TORQUE_LIMIT, DXL3_VAL_TORQ)
    packetHandler.write2ByteTxRx(portHandler, DXL4_ID, ADDR_MX_TORQUE_LIMIT, DXL4_VAL_TORQ)

    # Asignar el valor de cada posición angular en una matriz de bytes:
    param1_goal_position = [DXL_LOBYTE(DXL_LOWORD(DXL1_VAL_POS)), DXL_HIBYTE(DXL_LOWORD(DXL1_VAL_POS)),
                            DXL_LOBYTE(DXL_HIWORD(DXL1_VAL_POS)), DXL_HIBYTE(DXL_HIWORD(DXL1_VAL_POS))]
    param2_goal_position = [DXL_LOBYTE(DXL_LOWORD(DXL2_VAL_POS)), DXL_HIBYTE(DXL_LOWORD(DXL2_VAL_POS)),
                            DXL_LOBYTE(DXL_HIWORD(DXL2_VAL_POS)), DXL_HIBYTE(DXL_HIWORD(DXL2_VAL_POS))]
    param3_goal_position = [DXL_LOBYTE(DXL_LOWORD(DXL3_VAL_POS)), DXL_HIBYTE(DXL_LOWORD(DXL3_VAL_POS)),
                            DXL_LOBYTE(DXL_HIWORD(DXL3_VAL_POS)), DXL_HIBYTE(DXL_HIWORD(DXL3_VAL_POS))]
    param4_goal_position = [DXL_LOBYTE(DXL_LOWORD(DXL4_VAL_POS)), DXL_HIBYTE(DXL_LOWORD(DXL4_VAL_POS)),
                            DXL_LOBYTE(DXL_HIWORD(DXL4_VAL_POS)), DXL_HIBYTE(DXL_HIWORD(DXL4_VAL_POS))]

    # Agregar cada matriz de bytes en el almacenamiento de parámetros de Syncwrite:
    groupSyncWrite.addParam(DXL1_ID, param1_goal_position)
    groupSyncWrite.addParam(DXL2_ID, param2_goal_position)
    groupSyncWrite.addParam(DXL3_ID, param3_goal_position)
    groupSyncWrite.addParam(DXL4_ID, param4_goal_position)

    # Mover servomotores simultáneamente (Escritura de la pos. angular):
    dxl_comm_result = groupSyncWrite.txPacket()
    if dxl_comm_result != COMM_SUCCESS:
        print("%s" % packetHandler.getTxRxResult(dxl_comm_result))

    # Borrar el almacenamiento de los datos procesados:
    groupSyncWrite.clearParam()
```

Figura L.2: Función que permite la escritura de la posición angular y el par motor

Cabe señalar que los servomotores Dynamixel solamente aceptan la escritura de valores positivos de par motor, por lo que fue necesario establecer dentro de la conversión a números enteros, la obtención de su valor absoluto. Esto no afecta en la operación del brazo robótico, ya que el signo que acompaña al valor del par motor solamente nos indica en qué dirección podría moverse la articulación del brazo robótico debido al mismo par aplicado; positivo: en el sentido contrario de las manecillas del reloj, y

negativo: en el sentido de las manecillas del reloj.

Al igual que en el programa de la sección anterior (ver las Figuras H.2 y H.3), se definen las direcciones de los elementos que van a ser controlados en cada servomotor (tabla de control): escritura y lectura de la posición angular y el par motor. Posteriormente se definen las condiciones de la comunicación, como el uso del protocolo 1.0, el ID que tiene cada servomotor, la velocidad de comunicación (1,000,000 bps) y el puerto de comunicación que se estará utilizando ('COM5'). Además, deben definirse los valores que permiten la activación y desactivación del par motor en los servomotores (1 y 0 respectivamente).

```
# Valores iniciales del par motor para cada servomotor:
TAU1 = 0.35                                 # Par motor para 1er. servomotor [Nm]
TAU2 = 0.35                                 # Par motor para 2do. servomotor [Nm]
TAU3 = 0.35                                 # Par motor para 3er. servomotor [Nm]
TAU4 = 0.4                                  # Par motor para 4to. servomotor [Nm]

# Posición inicial de la pinza mecánica (como fue establecida en Scilab):
Pix =  0.15                                 # Coordenada en X de la posición inicial [m]
Piy = -0.15                                 # Coordenada en Y de la posición inicial [m]
Piz =  0.05                                 # Coordenada en Z de la posición inicial [m]

# Medidas de los eslabones que componen al brazo robótico:
hB = 0.079                                  # Altura de la base [m]
l1 = 0.113                                  # Longitud del eslabón 1 [m]
l2 = 0.14                                   # Longitud del eslabón 2 [m]

# Solución de la cinemática inversa para ubicar a la pinza en la posición inicial:
w = m.sqrt(Pix**2 + Piy**2)
THETA1 = m.atan2(Piy,Pix)*(180/m.pi)        # Pos. ang. para el 1er. servomotor (θ1)

phi = m.atan2(w,Piz-hB)*(180/m.pi)
r = m.sqrt(Pix**2 + Piy**2 + (Piz-hB)**2)
beta = 90 - phi
cg = (r**2 + l1**2 - l2**2)/(2*l1*r)
sg = m.sqrt(1 - cg**2)
gama = m.atan2(sg,cg)*(180/m.pi)
THETA2 = beta + gama                        # Pos. ang. para el 2do. servomotor (θ2)

ca = (l1**2 + l2**2 - r**2)/(2*l1*l2)
sa = m.sqrt(1 - ca**2)
alfa = m.atan2(sa,ca)*(180/m.pi)
THETA3 = -(180 - alfa)                      # Pos. ang. para el 3er. servomotor (θ3)

PINZA = 150                                 # Pos. ang. para el 4to. servomotor (Abrir pinza)
```

Figura L.3: Configuración de la posición inicial

Después, deben definirse las coordenadas de la posición inicial que fue establecida en SciLab, las medidas de los eslabones que componen al brazo robótico y los valores iniciales del par motor para cada servomotor (primera posición, vea la Tabla H.1). Luego, se escriben las ecuaciones de la cinemática inversa (pertenecientes a este brazo

robótico) para obtener los valores angulares (θ_1, θ_2, θ_3) que deberán tomar cada servomotor para ubicar a la pinza mecánica en la posición inicial establecida. Para este caso se definió que la pinza mecánica estará abierta. Vea la Figura L.3.

Una vez definidas todas las variables, se escriben las funciones de control que establecen la comunicación del puerto USB, la velocidad de transmisión de datos, la construcción de paquetes de datos para el control simultáneo de los servomotores y la habilitación y asignación del par motor para cada servomotor; vea las Figuras H.6 y H.7 que fueron mostradas en la sección anterior. Una vez establecidas las funciones de control correspondientes, se escribe la función general **MovServosSync** para ubicar a la pinza mecánica en la posición inicial que fue establecida.

```
# Abrir el archivo CSV para leerlo y escribir los valores de par motor y pos. ang.
# en cada servomotor:
with open('3STA_20s_NAW.csv', 'r') as csv_file:             # Abrir archivo CSV
    reader = csv.reader(csv_file, delimiter=',')            # Leer archivo CSV

    line_count = 0        # Variable que cuenta los renglones en el archivo CSV
    line_ref = 0          # Variable que selecciona a los renglones que asignarán los valores

    for row in reader:
        if line_count == 0:
            line_count += 1

        # Asignar los valores del renglón 0 y los de cada 2000 renglones:
        elif line_ref == 0 or line_ref == 2000:
            TAU1   = float(row[0])
            THETA1 = float(row[1])
            TAU2   = float(row[2])
            THETA2 = float(row[3])
            TAU3   = float(row[4])
            THETA3 = float(row[5])
            tiempo = float(row[6])

            # Mover servomotores y establecer valores del par motor:
            MovServosSync(TAU1,THETA1,TAU2,THETA2,TAU3,THETA3,TAU4,PINZA)

        # Ignorar 1999 renglones:
        else:
            line_count += 1
            line_ref += 1
    csv_file.close()                  # Cerar el archivo CSV
```

Figura L.4: Lectura del archivo CSV y asignación de valores

Como siguiente paso se tiene que realizar la lectura del archivo CSV que fue creado en SciLab, por lo tanto, se escribe la sintaxis que se muestra en la Figura L.4, donde la función **open()** permite abrir al archivo CSV como un archivo de texto y la función **reader()** permite leer todos los datos que están contenidos en el archivo. Es importante señalar que el archivo CSV debe estar dentro de la misma carpeta donde

se encuentre ubicado el programa que se está desarrollando en Python.

```
        # Asignar los valores del renglón 0 y los de cada 2000 renglones:
        elif line_ref == 0 or line_ref == 2000:
            TAU1   = float(row[0])
            THETA1 = float(row[1])
            TAU2   = float(row[2])
            THETA2 = float(row[3])
            TAU3   = float(row[4])
            THETA3 = float(row[5])
            tiempo = float(row[6])

            # Condiciones para la pinza mecánica:
            if tiempo == 0:
                time.sleep(3.0)             # Se suspende la ejecución durante 3 [s]
                PINZA = 88                  # Cerrar la pinza mecánica (Agarra 1er. objeto)
            elif tiempo == 10 or tiempo == 20:
                PINZA = 150                 # Abrir la pinza mecánica (Suelta los objetos)
            else:
                pass

            # Mover servomotores y establecer valores del par motor:
            MovServosSync(TAU1,THETA1,TAU2,THETA2,TAU3,THETA3,TAU4,PINZA)

            "Aquí van las funciones que permiten la lectura de la posición angular y carga
             actual de cada servomotor, al igual que la convesión de los valores leídos"

            # Guardar las lecturas en los vectores correspondientes:
            theta1.append(THETA1_R)
            theta2.append(THETA2_R)
            theta3.append(THETA3_R)
            carga1.append(CARGA1_R)
            carga2.append(CARGA2_R)
            carga3.append(CARGA3_R)
            carga4.append(CARGA4_R)
            t.append(tiempo)

            # Condiciones para la pinza mecánica:
            if tiempo == 0:
                time.sleep(2.0)             # Se suspende la ejecución durante 2 [s]
            elif tiempo == 10:
                time.sleep(6.0)             # Se suspende la ejecución durante 6 [s]
                PINZA = 95                  # Cerrar la pinza mecánica (Agarra 2do. objeto)
                MovServosSync(TAU1,THETA1,TAU2,THETA2,TAU3,THETA3,TAU4,PINZA)
                time.sleep(2.0)             # Se suspende la ejecución durante 2 [s]
            elif tiempo == 20:
                time.sleep(2.0)             # Se suspende la ejecución durante 1 [s]
            else:
                pass

            line_count += 1
            line_ref = 1
```

Figura L.5: Condiciones para la pinza mecánica y la lectura de los servomotores

Posteriormente se define un ciclo *for* y se establece que leerá todos los registros del archivo CSV, renglón por renglón (cada renglón contendrá 7 columnas), logrando asignar los correspondientes valores de posición angular y par motor a cada servomotor. Debido a que el archivo CSV está conformado por 200,002 renglones, puede que el programa tarde mucho en asignar (en cada servomotor) los valores de posición

angular y par motor que contiene cada renglón; por tal motivo se estableció que solamente se asignen los valores del primer renglón y los de cada 2000 renglones que se vayan contando en el archivo CSV (ignorar 1999 renglones); vea la Figura L.4.

Este programa va a permitir que la pinza mecánica (ubicada en la posición inicial) agarre un primer objeto (con masa de 0.075 kg) y logre trasladarlo a una posición objetivo (segunda posición, vea la Tabla H.1); una vez que llegue a dicha posición, la pinza mecánica se abrirá (soltará al objeto) y agarrará un segundo objeto (con masa de 0.085 kg) para trasladarlo a la posición anterior. Por lo tanto, en el código de programación de la Figura L.4 se agregaron las condiciones que permiten abrir y cerrar la pinza mecánica en el momento adecuado; vea la Figura L.5.

Al momento que se esté ejecutando el programa (el brazo robótico estará en movimiento), se deberán obtener las lecturas de la posición angular y carga actual de cada servomotor, por lo que fue necesario agregar las funciones de control que permiten realizar dichas lecturas. Debido a que las funciones de lectura entregan valores enteros, se realizó su conversión a valores de posición angular y carga motor; cada lectura se estará guardando en los vectores correspondientes que fueron definidos al principio del programa. Posteriormente se utilizó la función **close()** para cerrar el archivo CSV una vez que se haya terminado de leer todos los renglones. Para cerrar el puerto de comunicación, se utilizó la función **closePort()**.

```
# Gracficar las posiciones angulares que fueron leídas en cada servomotor:
plt.figure()
plt.subplot(3,1,1)
plt.title('Posiciones angulares leidas por los servomotores Dynamixel', fontsize = 12)
plt.plot(t, theta1, linestyle = ':', color = 'g', linewidth = 2, label = '$\\theta_1$ - ID: 1')
plt.legend(loc="upper right")

plt.subplot(3,1,2)
plt.plot(t, theta2, linestyle = ':', color = 'g', linewidth = 2, label = '$\\theta_2$ - ID: 2')
plt.ylabel('Grados', fontsize = 12)
plt.legend(loc="upper right")

plt.subplot(3,1,3)
plt.plot(t, theta3, linestyle = ':', color = 'g', linewidth = 2, label = '$\\theta_3$ - ID: 3')
plt.xlabel('Tiempo (s)', fontsize = 12)
plt.legend(loc="upper right")
plt.grid(True)
plt.savefig('PosAng.png', dpi = 300) # Guarda la gráfica con 300dpi (puntos por pulgada)
plt.show()
```

Figura L.6: Código para graficar las posiciones angulares leídas

La sintaxis que permite realizar la lectura y la conversión de los valores leídos, se mostró en la Figura H.9 del apéndice H.

Finalmente se escribe el código que permitirá construir las gráficas de las lecturas que se fueron obteniendo de cada servomotor en un cierto instante de tiempo. En la Figura L.6 se muestra el código que permite graficar las posiciones angulares leídas. La sintaxis que se muestra en la Figura L.6 puede usarse para gráficar las cargas que fueron aplicadas en cada servomotor.

Bibliografía

[Arzelier et. al, 1993]

Arzelier, D., Bernussou, J. y García, G.; «Pole Assignment of Linear Uncertain Systems in a Sector Via a Lyapunov-Type Approach»; *Problem statement*; **II**; p. 1128; IEEE Transactions on Automatic Control; Vol. 38, No. 7; 1993.

[Atkenson y Hollerback, 1988]

Atkenson, C. G. y Hollerback, J. M.; *Model-based control of a robot manipulator*; MIT Press; Cambridge; 1988.

[Barrientos et. al, 1997]

Barrientos, A., Peñin, L. F., Balaguer, C. y Aracil, R.; «Fundamentos de Robótica»; *Introducción, Cinemática del robot, Dinámica del robot*; **1, 4, 5**; pp. 5-8, 94-99, 131; Segunda edición; McGraw-Hill; 1996.

[Cheol-Su et al., 2018]

Cheol-Su, J., Jong-Shik, K. y Seong-Ik, H.; «Tracking error constrained supertwisting sliding mode control for robotic systems»; pp. 804-814; International Journal of Control, Automation and Systems; 2018.

[Chilai y Gahinet, 1996]

Chilai, M. y Gahinet, P.; «H_∞ Design with Pole Placement Constraints: An LMI Approach»; *LMI formulation of pole-placement objectives*; **II**; p. 361; IEEE Transactions on Automatic Control; Vol. 41, No. 3; 1996.

[Corke, 2017]

Corke, Peter; «Robotics, Vision and Control»; *Time and Motion*; **3**; pp. 70-71; Segunda edición; Springer; 2017.

[Craig, 2006]

Craig, John; «Robótica»; *Introducción, Cinemática de manipuladores, Generación de trayectorias, Dinámica de manipuladores*; **1, 3, 7, 6**; pp. 1-6, 67, 209-210, 188-189; Tercera edición; Prentice Hall México; 2006.

[Dahleh et. al, 2011]

Dahleh, M., Dahleh, M.A. y Verghese, G.; «Lectures on Dynamic Systems and Control»; *Internal Stability for LTI Systems*; **14**; p. 3; Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology; 2011.

[Franco, 2014]

Franco, Hugo; «Control por modos deslizantes de tercer orden para un helicóptero de prueba de tres grados de libertad»; *Diseño del Esquema de Control, Preliminares*; **4, 2**; pp. 31, 8-10; Tesis de Licenciatura en Ingeniería Eléctrica y Electrónica; UNAM; 2014.

[Gahinet, 1996]

Gahinet, Pascal; «Explicit Controller Formulas for LMI-Based H_∞ Synthesis»; *Preliminaries and Motivations*; **2**; pp. 3-4; Automatica; Vol. 32; 1996.

[Gordillo, 2009]

Gordillo, Francisco; «Estabilidad de Sistemas No Lineales Basada en la Teoría de Liapunov»; pp. 6-8; Revista Iberoamericana de Automática e Informática Industrial; Sevilla, España; 2009.

[Hernández, 2015]

Hernández, Luis; «Diseño y control de un brazo robótico de 3 grados de libertad»; *Análisis Cinemático y Dinámico del Robot de 3 GDL*; **2**; p. 24; Tesis de Licenciatura en Ingeniería Eléctrica y Electrónica; UNAM; 2015.

[Hoifodt, 2011]

Hoifodt, Herman; «Dynamic Modeling and Simulation of Robot Manipulators»; *Introduction*; **1**; pp. 1-2; Norwegian University of Science and Technology; Department of Engineering Cybernetics; 2011.

[Javed et al., 2019]

Javed, M., Liu, H., Nie, J. y Sun, J.; «Robust tracking control for robotic manipulators based on super-twisting algorithm»; pp. 129-136; AMMM International Conference; 2019.

[Kamal et. al, 2014]

Kamal, S., Chalanga, A., Moreno, J.A., Fridman, L. y Bandyopadhyay, B.; «Higher Order Super-Twisting Algorithm»; pp. 2-3; 13th IEEE Workshop on Variable Structure Systems; Nantes, France; 2014.

[Khalil, 1996]

Khalil, Hassan.; «Nonlinear Systems»; *Lyapunov Stability*; **3**; p. 97; Segunda edición; Prentice Hall; 1996.

[Lewis y Munro, 2006]

Lewis, F. y Munro, N.; «Robot Manipulator Control: Theory and Practice»; *Robot Dynamics*; **3**; pp. 111, 125; Segunda edición; Marcel Dekker Inc.; 2006.

[Moreno, 2014]

Moreno, M.A.; «Primeras mediciones precisas de la gravedad hechas en México»; *El experimento*; **6**; pp. 28-29; Revista Mexicana de Física, 60; 2014.

[Pérez, 2015]

Pérez, Waldemar; «Manual de prácticas de la materia de robótica I»; *Simulación de una Trayectoria en Linea Recta, usando Polinomio de Quinto Grado*; **8**; pp. 72-78; Academia de Electrónica; ITLAC; 2015.

[Pérez, 2011]

Pérez, Waldemar; «Planeación de trayectorias óptimas para robots manipuladores utilizando polinomios de octavo grado, algoritmos genéticos y técnicas de procesamiento en paralelo»; *Cinemática del robot manipulador*; **3**; pp. 25-30; Tesis de Doctarado en Ciencias en Ingeniería Eléctrica; UMSNH; 2011.

[Rivera, 2011]

Rivera, J., Garcia, L., Mora, C., Raygoza, J. y Ortega,S.; «Sliding Mo-

de Control»; *Super-twisting sliding mode in motioncontrol systems*; **13**; pp. 237-254; InTech; 2011.

[ROBOTIS, 2020]

ROBOTIS; «ROBOTIS e-Manual»; *AX-12A, AX-18A, Interface U2D2, DYNAMIXEL SDK*; 2020.
https://emanual.robotis.com/docs/en/dxl
https://emanual.robotis.com/docs/en/parts/interface//u2d2
https://emanual.robotis.com/docs/en/software/dynamixel/dynamixel_sdk

[Rocha, 2017]

Rocha Cózatl, Edmundo Gabriel; «Reporte técnico final del proyecto PE109815 "Bancos de prueba para prácticas de control automático en motores de corriente directa"»; Proyecto realizado en el marco del Programa de Apoyo a Proyectos para la Innovación y el Mejoramiento de la Enseñanza (PAPIME, Convocatoria 2015) de la Dirección General de Asuntos del Personal Académico; UNAM; Presentado el 5 de Mayo de 2017.

[Ruiz y Fridman, 2014]

Ruiz, M. y Fridman, L.; «Implementación de controladores por Modos Deslizantes a un Robot 2-GDL»; *Introducción*; **3**; pp. 367-372; Memorias del XVI Congreso Latinoamericano de Control Automático; 2014.

[Scherer y Weiland, 2005]

Scherer, C. y Weiland, S.; «Linear Matrix Inequalities in Control»; *Nominal Stability and nominal performance*; **3**; p. 65; Delft University of Technology, Eindhoven University of Technology; The Netherlands; 2005.

[Schilling, 2003]

Schiling, Robert; «Fundamentals of Robotics: Analysis and Control»; *Direct Kinematics: The Arm Equation, Inverse Kinematics: Solving The Arm Equation, Manipulator Dynamics, Robot Control*; **2, 3, 6, 7**; pp. 58, 83, 194-195, 235-236, 243-245, 249-250; Quinta edición; Prentice-Hall of India; 2003.

[Seron y Braslavsky, 2000]

Seron, M. y Braslavsky, J.; «Sistemas No Lineales»; *Estabilidad según Lyapunov - Sistemas Estacionarios*; **3**; p. 60; Universidad Nacional de Rosario, Departamento de Electrónica; 2000.

[Shtessel et. al, 2014]

Shtessel, Y., Edwards, C., Fridman, L. y Levant, A.; «Sliding Mode Control and Observation»; *Main Concepts of Sliding Mode Control, Concept of Equivalent Control, Conventional Sliding Mode Controller Design, Conventional Sliding Modes, Second-Order Sliding Mode Controllers and Differentiators*; pp. 1-9, 17-18, 28-30, 44, 146, 148; Primera edición; Birkhauser; 2014.

[Siciliano et. al, 2009]

Siciliano, B., Sciavicco, L., Luigi, V. y Oriolo, G.; «Robotics: Modeling, Planning and Control»; *Dynamics*; **7**; pp. 247-248; Novena edición; Springer; 2009.

[Siciliano y Khatib, 2008]

Siciliano, B. y Khatib, O.; «Handbook of Robotics»; *Mechanisms and Actuation*; **3**; pp. 67-68; Segunda edición; Springer; 2008.

[Torres et al., 2018]

Torres, I., Vargas, A., López-Caamal, F., y Hernández-Escoto, H.; «Preliminar ideas on a real-time optimization strategy based on the super-twisting algorithm»; *Real-time optimization based on the super-twisting algorithm*; **3**; p. 355; Memorias del Congreso Nacional de Control Automático; México; 2018.

[UdeSantiago, 2021]

Universidad de Santiago de Chile; «Robótica Industrial»; *Desarrollo Histórico y Evolución de la Robótica*; **1**; Edición virtual; 2021.
http://www.udesantiagovirtual.cl/moodle2/course/view.php?id=4