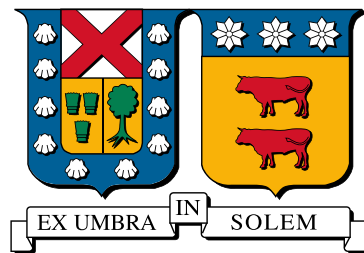


**UNIVERSIDAD TÉCNICA FEDERICO SANTA
MARÍA**

DEPARTAMENTO DE ELECTRÓNICA

VALPARAÍSO - CHILE



**“IMPLEMENTACIÓN DE UN SISTEMA
DE CONTROL PREDICTIVO PARA UNA
BARRA GIRATORIA FLEXIBLE”**

SEBASTIÁN LE BARÍ CÉSPEDES SEGOVIA

**MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL
ELECTRÓNICO.**

**PROFESOR GUÍA: DR. JUAN CARLOS AGÜERO
PROFESORA CORREFERENTE: DRA. MARÍA CORONEL
PROFESOR CORREFERENTE: DR. ÁNGEL CEDEÑO**

AGOSTO 2024

Agradecimientos

Agradecimientos

Quiero expresar mi profundo agradecimiento a mi familia: a mi mamá Verónica, a mi papá Leonardo y a mi abuelo Orlando, quienes me han apoyado desde mis primeros años en el colegio hasta esta etapa universitaria. Su dedicación y amor incondicional me han ayudado a formarme como una persona que aspira a retribuir lo recibido a su entorno. También agradezco a mi hermano Andrés, a mi hermana Mychelle y a mis mascotas, quienes, como parte fundamental de mi vida, me han acompañado en todo momento.

Agradezco sinceramente a mi novia Nahiara, cuyo amor, comprensión y compañía han sido un pilar fundamental durante este último año.

Extiendo mi agradecimiento a todos los amigos y compañeros que he conocido a lo largo de este camino, por su compañía, consejos y apoyo en diversas situaciones.

Quiero destacar el apoyo, guía y colaboración de mi compañero Ariel Parra en el desarrollo de esta memoria. Su contribución ha sido invaluable.

Agradezco también a mis profesores y ayudantes por el conocimiento y apoyo brindado durante mi tiempo en la universidad. En particular, a los profesores Juan Carlos Agüero, César Silva y Alejandro Suárez, por su comprensión, apoyo y las oportunidades que me ofrecieron, especialmente durante las etapas finales de mi carrera y en la elaboración de esta memoria.

Finalmente, agradezco el apoyo del Centro Basal "Advanced Center for Electrical and Electronic Engineering" (AC3E) - FB0008, así como de los proyectos ANID-FONDECYT 1211630, 3240181 y 3230398.

Resumen

En este proyecto se aplica un método de control avanzado denominado Control Predictivo por Modelo (*MPC*, por sus siglas en inglés) a la planta Barra Giratoria Flexible (RFL, por sus siglas en inglés). El objetivo principal es regular y controlar los movimientos rotacionales de la planta en torno a su eje de giro. La planta está sometida a vibraciones inherentes, excitaciones externas y perturbaciones que afectan su comportamiento.

Para lograr este objetivo, se inicia con la simulación del sistema utilizando los software MATLAB y Simulink, diseñando el controlador MPC mediante dos métodos de optimización. El primero es el método de Direcciones Alternadas de Multiplicadores de Lagrange (ADMM), y el segundo es el método *Active-set*, que utiliza el solver Quadprog en MATLAB. Ambos métodos emplean la Formulación Sparse para mejorar la eficiencia computacional.

Una vez obtenidos los resultados de la simulación, se comparan con los resultados de un controlador LQR (Linear Quadratic Regulator), diseñado por Quanser, la empresa que fabrica la planta RFL. La comparación tiene como objetivo mejorar los resultados de control para la vibración producida durante los movimientos rotacionales.

Para la implementación real del controlador, se utiliza la plataforma dSPACE, que facilita el desarrollo y la conexión con la planta RFL. Los datos del control MPC obtenidos se emplean para entrenar una red neuronal, la cual se utiliza tanto en dSPACE como en Quarc para optimizar el control de la planta.

Keywords: MPC; RFL; MATLAB; Simulink; ADMM; Active-Set; Quadprog; Formulación Sparse; LQR; Quanser; dSPACE; Red Neuronal; Quarc

Abstract

This project applies an advanced control method known as Model Predictive Control (MPC) to the Rotary Flexible Link (RFL) plant. The primary objective is to regulate and control the rotational movements of the RFL around its axis. The plant experiences inherent vibrations, external excitations, and disturbances that affect its performance.

To achieve this goal, the project begins with simulations using MATLAB and Simulink to design the MPC controller with two optimization methods. The first method is the Alternating Direction Method of Multipliers (ADMM), and the second is the *Active-set* method, utilizing the Quadprog solver in MATLAB. Both methods use Sparse Formulation to enhance computational efficiency.

After obtaining simulation results for the MPC controller design, these are compared with results from an LQR (Linear Quadratic Regulator) controller designed by Quanser, the manufacturer of the RFL plant. The comparison aims to improve control results for the vibrations produced during rotational movements.

For real-world implementation, the dSPACE platform is chosen, which facilitates development and integration with the RFL plant. Data from the MPC control implementation are used to train a neural network, which is then applied both in dSPACE and Quarc to optimize plant control.

Keywords: MPC; RFL; MATLAB; Simulink; ADMM; Active-Set; Quadprog; Sparse Formulation; LQR; Quanser; dSPACE; Neural Network; Quarc

Glosario

- **MPC**: Model Predictive Control o Control Predictivo por Modelo.
- **RHC**: Receding Horizon Control o Horizonte de Control Móvil.
- **RFL**: Rotary Flexible Link o Barra Giratoria Flexible.
- **DC**: Direct Current o Corriente Continua.
- **SIMO**: Single-Input & Multiple-Outpout o Única-Entrada % Múltiples Salidas.
- **ADMM**: Alternating Direction Method of Multipliers o Método de Multiplicadores de Lagrange con Direcciones Alternadas.
- **LQR**: Lineal Quadratic Regulator o Regulador Cuadrático Lineal.
- **QP**: Quadratic Programming o Programación Cuadrática
- **ADAM**: Adaptive Moment Estimation o Estimación de Momentos Adaptativos.
- **KKT**: Condiciones de Karush-Kuhn-Tucker para optimalidad.
- **RTI**: Real Time Interface o Interfaz en Tiempo Real.
- **FPGA**: Field Programmable Gate Array o Matriz de compuertas programable en campo.

Contents

1. Introducción	1
1.1. Estado del arte	1
1.1.1. Control MPC	1
1.1.2. Método de optimización ADMM	2
1.1.3. Método de optimización Active-Set de Quadprog	2
1.1.4. MicroLabBox dSPACE	2
1.1.5. Redes Neuronales	3
1.1.6. Software Quarc	4
1.2. Objetivos Específicos	4
2. Planta de interés	6
2.1. Descripción de la planta	6
2.2. Componentes de la planta	7
2.3. Modelo de ecuaciones de movimiento	10
2.4. Representación en variables de estado	13
2.5. Estabilidad del sistema	15
2.6. Controlabilidad y Observabilidad	17
2.7. Discretización	18
3. Controlador LQR para RFL	21
3.1. Formulación de LQR	21
3.2. Efectos de las ganancias en el controlador y sistema	22
4. Control predictivo por modelo	27
4.1. Metodología del control predictivo	28
4.2. Programación cuadrática (QP)	30
4.3. Formulación problema de control	30
4.4. Formulación Sparse y Restricciones	32
5. Redes Neuronales	37
5.1. Código de la red neuronal	40

6. Métodos de optimización	42
6.1. Método de optimización ADMM	42
6.1.1. Método "Dual Ascent"	42
6.1.2. Método de Descomposición Dual	43
6.1.3. Método de los Multiplicadores Aumentados de Lagrange	44
6.1.4. Método de los Multiplicadores de Lagrange con Direcciones Alternadas	45
6.2. Método de optimización Active-set de Quadprog	46
6.2.1. Metodología del Algoritmo Active-Set	46
7. Diseño de control	49
7.1. Prealimentación para inestabilidad de la planta	49
7.2. Seguimiento a referencia	50
7.3. Observador de estados	52
7.4. Observador de estados Extendido con perturbación modelada	54
7.5. Seguimiento a referencia con perturbación	55
8. Modelo en entorno de simulación	57
8.1. Bloques utilizados	59
8.2. Entorno MATLAB/Simulink	61
8.3. Aspectos Generales para simulación	61
8.3.1. Modelo de la RFL a simular	61
8.4. Active-Set de Quadprog para la RFL	63
8.4.1. Active-Set de Quadprog para estado cero y seguimiento a referencia . .	64
8.4.2. Incorporación de observadores de estado para Quadprog	71
8.5. ADMM para la RFL	76
8.5.1. ADMM para estado cero y seguimiento a referencia	77
8.5.2. Incorporación de observadores de estado para ADMM	81
9. Implementación Hardware-Software dSPACE y Quarc	86
9.1. Implementación MATLAB Simulink & MicroLabBox dSPACE	86
9.1.1. Hardware de dSPACE	87
9.1.2. Software de dSPACE	89

9.1.3. Implementación de MPC en dSPACE	97
9.1.4. Implementación de Redes Neuronales en dSPACE	101
9.2. Implementación MATLAB Simulink & Quarc	108
9.2.1. Hardware de Quanser	109
9.2.2. Software de Quanser	113
9.2.3. Implementación de Redes Neuronales en Quarc	117
10. Conclusiones y Trabajos Futuros	121
10.1. Conclusión	121
10.2. Trabajos Futuros	121
11. Anexos	123
11.1. Controlabilidad para la RFL	123
11.2. Observabilidad para la RFL	123
11.3. Control LQR sobre la Rotary Flexible Link	124
11.4. Control MPC sobre la Rotary Flexible Link	125
11.4.1. MPC para Estado Cero	125
11.4.2. MPC Seguimiento a Referencia	127
11.4.3. Código Observador de Estados	128
11.4.4. Código Bloque <i>mat_f</i> Estado Cero	129
11.4.5. Código Bloque <i>mat_f</i> y <i>mat_g</i> Seguimiento a Referencia	129
11.4.6. Código Quadprog	129
11.4.7. Código ADMM	129
11.5. Control con Redes Neuronales sobre la Rotary Flexible Link	130
11.5.1. Código de Entrenamiento de Redes Neuronales	130
11.5.2. Código Bloque Neural Network	131
11.5.3. Código Redes Neuronales	132
Referencias	134

List of Tables

2.1. Componentes de la Planta [1]	8
8.2. Parámetros de la RFL	58
9.3. Cables y Sensores [1]	112

List of Figures

2.1. Barra giratoria Flexible [2]	7
2.2. Diagrama de la RFL [2]	7
2.3. Barra Giratoria acoplada a Motor SRV02 [1]	8
2.4. Galga Extensométrica [1]	9
2.5. Diagrama de control digital-análogo	19
3.6. Lazo de control del estado del sistema	22
3.7. Esquema de control de RFL en Simulink	24
3.8. Voltaje del motor	25
3.9. Referencia deseada (verde), ángulo de rotación de la RFL (rojo)	26
3.10. Ángulo de flexión de la RFL	26
4.11. Esquema de control MPC	28
4.12. Horizonte de control móvil [3]	29
4.13. Cálculo de actuación u_{opt} en Horizonte de control móvil [3]	29
5.14. Red Neuronal	40
7.15. Lazo de control con prealimentación	49
7.16. Lazo de control con prealimentación	52
7.17. Diagrama del observador básico	52
7.18. Seguimiento a referencia con observador de estados	56
8.19. Bloque States Calculation	58
8.20. Interior bloque States Calculation	59
8.21. Bloques de Simulink	59
8.22. Diagrama RFL Estado Cero	63
8.23. Diagrama RFL Seguimiento a Referencia	63

8.24. Bloque MPC para Quadprog en estado cero	64
8.25. Quadprog Estado Cero: Actuación u_x en el Tiempo	65
8.26. Quadprog Estado Cero: Actuación u_{sat} en el Tiempo	65
8.27. Quadprog Estado Cero: Ángulo de rotación θ en el Tiempo	66
8.28. Quadprog Estado Cero: Ángulo de flexión α en el Tiempo	66
8.29. Código Seguimiento a Referencia: Lineas 70-74	67
8.30. Interior bloque Reference Tracking	67
8.31. Interior bloque MPC	68
8.32. Quadprog Seguimiento a Referencia: Referencia en el Tiempo	68
8.33. Quadprog Seguimiento a Referencia: Actuación u_x en el Tiempo	69
8.34. Quadprog Seguimiento a Referencia: Actuación u_{sat} en el Tiempo	69
8.35. Quadprog Seguimiento a Referencia: Ángulo de rotación θ en el Tiempo	70
8.36. Quadprog Seguimiento a Referencia: Ángulo de flexión α en el Tiempo	70
8.37. Observador de estados para RFL	71
8.38. Diagrama de estado cero con observador de estados	71
8.39. Quadprog Estado Cero Observador: Actuación u_x en el Tiempo	72
8.40. Quadprog Estado Cero Observador: Actuación u_{sat} en el Tiempo	72
8.41. Quadprog Estado Cero Observador: Ángulo de rotación θ en el Tiempo	72
8.42. Quadprog Estado Cero Observador: Ángulo de flexión α en el Tiempo	73
8.43. Diagrama de seguimiento a referencia con observador de estados	74
8.44. Quadprog Seguimiento a Referencia Observador: Actuación u_x en el Tiempo	74
8.45. Quadprog Seguimiento a Referencia Observador: Actuación u_{sat} en el Tiempo	75
8.46. Quadprog Seguimiento a Referencia Observador: Ángulo de rotación θ en el Tiempo	75
8.47. Quadprog Seguimiento a Referencia Observador: Ángulo de flexión α en el Tiempo	76
8.48. Interior bloque MPC con función ADMM	77
8.49. ADMM Estado Cero: Actuación u_x en el Tiempo	78
8.50. ADMM Estado Cero: Actuación u_{sat} en el Tiempo	78
8.51. ADMM Estado Cero: Ángulo de rotación θ en el Tiempo	79
8.52. ADMM Estado Cero: Ángulo de flexión α en el Tiempo	79

8.53. ADMM Seguimiento a Referencia: Actuación u_x en el tiempo	80
8.54. ADMM Seguimiento a Referencia: Actuación u_{sat} en el tiempo	80
8.55. ADMM Seguimiento a Referencia: Ángulo de rotación θ en el tiempo	81
8.56. ADMM Seguimiento a Referencia: Ángulo de flexión α en el tiempo	81
8.57. ADMM Estado Cero Observador: Actuación u_x en el tiempo	82
8.58. ADMM Estado Cero Observador: Actuación u_{sat} en el tiempo	82
8.59. ADMM Estado Cero Observador: Ángulo de rotación θ en el tiempo	82
8.60. ADMM Estado Cero Observador: Ángulo de flexión α en el tiempo	83
8.61. ADMM Seguimiento a Referencia Observador: Actuación u_x en el tiempo	83
8.62. ADMM Seguimiento a Referencia Observador: Actuación u_{sat} en el tiempo . .	84
8.63. ADMM Seguimiento a Referencia Observador: Ángulo de rotación θ en el tiempo	84
8.64. ADMM Seguimiento a Referencia Observador: Ángulo de flexión α en el tiempo	85
9.65. MicroLabBox dSPACE	87
9.66. Cable dSPACE Encoder	88
9.67. Cable RCA to BNC	88
9.68. Cable DIN 4 pin to DIN 6 pin	88
9.69. Interfaz RTI	89
9.70. Software ControlDesk	90
9.71. Bloque ADC_CLASS_1_BL1	90
9.72. Bloque DAC_CLASS_1_BL1	91
9.73. DS1202_SENSOR_SUPPLY	91
9.74. Bloque EMC_Encoder_BL1	92
9.75. Selección dSPACE RTI Platform Support	92
9.76. Verificación en ventana de comandos	93
9.77. Verificación de ruta de trabajo	93
9.78. Creación de esquema Simulink	93
9.79. Selección de Template DS1202	93
9.80. Ventana CCODE	94
9.81. Selección de Solver Discrete	94
9.82. Desmarcar opción Load application after build	95
9.83. Ventana File y creación de Experimento	95

9.84. Creación de experimento	96
9.85. Importar archivo .sdf	96
9.86. Selección de señales a medir	97
9.87. Instrumentos para medición de señales	97
9.88. Ventana Home de controlDesk	97
9.89. Esquema de Control MPC con Observador de Estados Planta Real	98
9.90. Control MPC dSPACE: Referencia en el Tiempo	99
9.91. Control MPC dSPACE: Ángulo de rotación θ en el Tiempo	99
9.92. Control MPC dSPACE: Ángulo de flexión α en el Tiempo	100
9.93. Control MPC dSPACE: Actuación u_x en el Tiempo	100
9.94. Control MPC dSPACE: Actuación u en el Tiempo	101
9.95. Control MPC dSPACE: Estimación de la perturbación d_z en el Tiempo	101
9.96. Magnitud de pérdida por Número de Épocas	102
9.97. Actuación u_x : Predicción con datos de entrenamiento	103
9.98. Actuación u_x : Datos de entrenamiento	103
9.99. Actuación u_x : Predicción con datos de validación	104
9.100 Actuación u_x : Datos de validación	104
9.101 Diagrama control con redes neuronales en dSPACE	105
9.102 Redes Neuronales dSPACE: Referencia en el Tiempo	106
9.103 Redes Neuronales dSPACE: Ángulo de rotación θ en el Tiempo	106
9.104 Redes Neuronales dSPACE: Ángulo de flexión α en el Tiempo	107
9.105 Redes Neuronales dSPACE: Actuación u en el Tiempo	107
9.106 Redes Neuronales dSPACE: Actuación u_x en el Tiempo	108
9.107 Redes Neuronales dSPACE: Estimación de la perturbación d_z en el Tiempo	108
9.108 Diagrama de conexión del sistema [1]	109
9.109 VoltPAQ-X1 Amplifier	110
9.110 Q2-USB Data Acquisition Device	110
9.111 Conexiones de la Rotary Flexible Link	111
9.112 Bloques de Quarc Utilizados	113
9.113 Interior bloque Bias Removal	114
9.114 Interior bloque HIL Write Digital	114

9.115	Interior bloque HIL Read Time	115
9.116	Interior bloque HIL Initialize	116
9.117	Diagrama del controlador con redes neuronales en Quarc	117
9.118	Diagrama del controlador con redes neuronales en Quarc	117
9.119	Implementación de RFL con software Quarc	118
9.120	Redes Neuronales Quarc: Ángulo de rotación θ y Referencia en el Tiempo . . .	118
9.121	Redes Neuronales Quarc: Ángulo de flexión α en el Tiempo	119
9.122	Redes Neuronales Quarc: Actuación u_{sat} en el Tiempo	119
9.123	Redes Neuronales Quarc: Actuación u_x en el Tiempo	120
9.124	Redes Neuronales Quarc: Estimación de la perturbación d_z en el Tiempo . . .	120

1. Introducción

En el presente trabajo de titulación, titulado "*Implementación de un sistema de control predictivo para una barra giratoria flexible*", se utilizará la *Rotary Flexible Link* (RFL) o Barra Giratoria Flexible como planta de estudio. El objetivo principal de este trabajo es desarrollar un sistema de control que gestione eficazmente esta planta. Para lograrlo, se analizarán las propiedades de controlabilidad y observabilidad del sistema. Se implementarán estrategias de control empleando un controlador *Linear Quadratic Regulator* (LQR) para evaluar el comportamiento del sistema en términos de actuación y salida.

Con base en los resultados obtenidos del controlador LQR, se diseñará un controlador utilizando *Model Predictive Control* (MPC), un método avanzado de control automático. El MPC permite tomar decisiones óptimas en tiempo real al utilizar un modelo del proceso para predecir la salida de la planta y anticipar posibles eventos mediante la optimización de las acciones de control futuras.

Para abordar el problema planteado, se implementarán el método de multiplicadores de Lagrange con direcciones alternadas (ADMM) y el método *Active-set* de la herramienta *Quadprog* como técnicas de optimización. Estos métodos serán desarrollados en MATLAB y complementados con esquemas de simulación en el software Simulink.

Una vez realizadas las simulaciones, se procederá a implementar las soluciones en la planta real utilizando el software ControlDesk y el hardware MicroLabBox 1202 de dSPACE. Este entorno permite la ejecución en tiempo real mediante la construcción de un archivo a partir de Simulink. Finalmente, se compararán los resultados de ambos controladores para evaluar su desempeño y efectividad en la planta real.

1.1. Estado del arte

1.1.1. Control MPC

La técnica de *Model Predictive Control* (MPC) es un método avanzado de control caracterizado por ser un algoritmo de predicción. Consiste en encontrar la acción de control óptima, respetando siempre las restricciones establecidas dentro de la ventana de predicción. Esta

técnica tiene un alto potencial en plantas con comportamientos dinámicos complejos, ya sea por la rapidez o alta variación de sus respuestas y señales a medir, o por los retrasos generados por distintas acciones [4].

Una de las formas de trabajar con *Model Predictive Control* (MPC) es considerar una función de costo cuadrático junto a restricciones lineales. Esto se puede llevar a cabo mediante un algoritmo de optimización cuadrática (QP), que se enfoca en minimizar la función de costo planteada. Este enfoque se diferencia de otros métodos conocidos, como PID o LQR, que suelen implementarse de manera general.

1.1.2. Método de optimización ADMM

El método de multiplicadores de Lagrange con direcciones alternadas (ADMM, por sus siglas en inglés) es un método de optimización utilizado para resolver problemas en los que se busca minimizar una función objetivo sujeta a ciertas restricciones. Este método es especialmente útil para problemas que pueden dividirse en subproblemas más pequeños y que presentan restricciones lineales.

La idea básica detrás de ADMM es descomponer el problema original en subproblemas más simples que pueden resolverse de forma más eficiente. Luego, se alterna entre la optimización de cada subproblema y la actualización de los multiplicadores de Lagrange, con el fin de garantizar que las soluciones parciales converjan hacia la solución óptima del problema completo [5].

1.1.3. Método de optimización Active-Set de Quadprog

Quadprog es una función en MATLAB que se utiliza para resolver problemas de optimización cuadrática con restricciones lineales. Este tipo de problemas de optimización es común en el control avanzado y en la teoría de control, ya que permite encontrar la mejor manera de controlar un sistema sujeto a ciertas restricciones y considerando un objetivo específico [6].

1.1.4. MicroLabBox dSPACE

MicroLabBox es un dispositivo compacto y potente diseñado por dSPACE, una empresa alemana líder en tecnología de sistemas de control. Esta empresa se destaca por ofrecer

soluciones innovadoras y de alta calidad para diversas industrias, incluyendo automotriz, aeroespacial, eléctrica y más.

MicroLabBox de dSPACE es una herramienta versátil que combina E/S analógicas y digitales, capacidad de procesamiento de señales y conectividad en un único dispositivo. Su diseño compacto lo hace ideal para aplicaciones en entornos móviles o de laboratorio donde se requiere un sistema de control eficiente y fácil de usar.

MicroLabBox se integra perfectamente con el entorno de desarrollo de dSPACE, lo que permite a los ingenieros desarrollar, simular y probar sistemas de control en un entorno virtual antes de implementarlos en el dispositivo para su ejecución en tiempo real. Esto garantiza un desarrollo rápido y eficiente de sistemas de control precisos y confiables [7].

1.1.5. Redes Neuronales

Las redes neuronales, inspiradas en el funcionamiento del cerebro humano, son modelos de aprendizaje automático que imitan cómo las neuronas procesan y transmiten información. Estas redes están formadas por neuronas interconectadas, donde cada neurona recibe señales de entrada, procesa esa información y genera una salida. La red se organiza en capas: una capa de entrada que recibe los datos, una o más capas ocultas que procesan la información, y una capa de salida que produce el resultado final [8].

Cada conexión entre neuronas tiene un peso asociado que determina la importancia de la señal transmitida. Durante el entrenamiento, estos pesos se ajustan para minimizar la diferencia entre las salidas predichas y las salidas reales. Este ajuste se realiza utilizando ejemplos de entrada y salida en un proceso conocido como aprendizaje supervisado [8]. Las neuronas aplican funciones de activación a la combinación lineal de las señales de entrada y los pesos. Estas funciones introducen no linealidades en la red, permitiendo que las redes neuronales aprendan a modelar relaciones complejas en los datos.

Existen diferentes tipos de aprendizaje: supervisado, que utiliza ejemplos etiquetados de entrada y salida; no supervisado, en el que la red intenta descubrir patrones en los datos sin etiquetas; y por refuerzo, donde la red aprende a través de la interacción con un entorno dinámico [8].

Las aplicaciones de las redes neuronales son amplias y abarcan desde el reconocimiento de imágenes y el procesamiento del lenguaje natural hasta el diagnóstico médico, la conducción autónoma y mucho más. Su capacidad para adaptarse y aprender de los datos las convierte en una herramienta poderosa en diversos campos.

1.1.6. Software Quarc

QUARC Real Time Control Software, conocido como QUARC, es un software desarrollado por Quanser, una empresa líder en sistemas de control y educación en ingeniería. QUARC se utiliza principalmente en combinación con los productos de hardware de Quanser, como los sistemas de control en tiempo real y los experimentos virtuales. Su propósito es facilitar el diseño, la implementación y la verificación de algoritmos de control en tiempo real.

Una de las principales características de QUARC es su capacidad para permitir a los usuarios diseñar y probar algoritmos de control de manera eficiente en un entorno de simulación antes de implementarlos en hardware real. Esto ayuda a reducir el tiempo y los costos asociados con el desarrollo de sistemas de control, ya que permite a los ingenieros realizar pruebas exhaustivas y ajustes previos a la implementación final. QUARC también ofrece una interfaz gráfica intuitiva que facilita la configuración de experimentos de control y la visualización de resultados en tiempo real. Además, proporciona herramientas de análisis y depuración que ayudan a los usuarios a identificar y corregir posibles problemas en sus algoritmos de control [9].

1.2. Objetivos Específicos

Se plantean los siguientes objetivos principales de trabajo:

- Definir un modelo matemático y los parámetros para la planta.
- Estudiar el controlador LQR propuesto por Quanser.
- Construir un modelo en variables de estado en tiempo discreto para su implementación en un control predictivo por modelo (MPC).

- Formular el problema de optimización QP en su Formulación Sparse para resolver el problema de control de MPC.
- Diseñar un controlador MPC que incluya:
 - Variables de decisión.
 - Matrices de peso, estado y actuación.
 - Función de costo.
 - Horizonte de predicción.
 - Método de optimización.
- Desarrollar el código en MATLAB y el esquema de simulación en Simulink.
- Implementar la Formulación Sparse.
- Utilizar Active-set de la herramienta Quadprog como método de optimización.
- Aplicar *ADMM* como método de optimización.
- Analizar los métodos utilizados en simulación.
- Implementar el controlador simulado en el software y hardware de dSPACE.
- Recopilar datos y realizar el entrenamiento de redes neuronales.
- Implementar el control con redes neuronales en dSPACE y en QUARC.

2. Planta de interés

En este capítulo se estudiará la estructura de la planta de interés, que es del tipo electromecánica y corresponde a la Rotary Flexible Link (RFL) o Barra Giratoria Flexible. La RFL se compone de dos partes principales: la primera es la unidad base servo giratoria, que es un servomotor. Este dispositivo de accionamiento regula la corriente y controla la posición angular de giro y la velocidad angular de la planta. Está conectado a un sistema de control que, mediante el envío de señales a través del servomotor, denominado SRV02 Rotary Servo Base Unit, permite controlar la posición angular. Es idóneo para introducir conceptos y teoría básica de control debido a su sencilla manipulación. El motor DC del servomotor empuja un engranaje de piñón pequeño conectado a uno de tamaño medio que gira sobre el eje de carga. La posición se mide empleando un codificador óptico de alta resolución (encoder) o un potenciómetro [10].

2.1. Descripción de la planta

Junto al servomotor opera la barra giratoria, que se conecta a la carga del servomotor y permite experimentar con los movimientos rotativos generados. Mediante las señales enviadas hacia y desde el servomotor, se producen efectos sobre la barra, ajustando la velocidad y la dirección para que se mueva hacia una posición objetivo propuesta. El brazo giratorio actúa como un enlace flexible, lo que le permite doblarse o ajustarse para adaptarse a las condiciones del sistema, siendo útil en casos de desalineación o variación en la carga. Posee varias aplicaciones, como en robótica, sistemas de automatización industrial y control de posición, entre otros [10].

La figura 2.1 muestra la Rotary Flexible Link, extraída del datasheet de su sitio oficial [2], donde se describen detalladamente los módulos mencionados anteriormente, así como los tópicos de modelado y control y sus especificaciones.

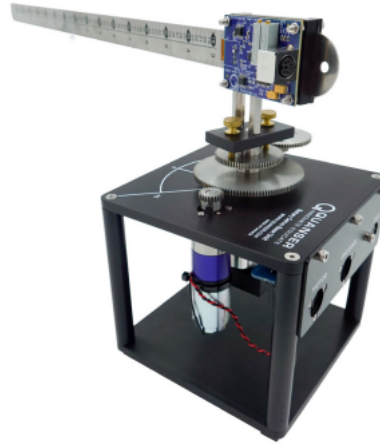


Figure 2.1: Barra giratoria Flexible [2]

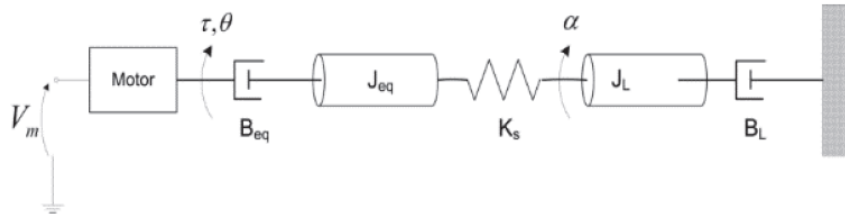


Figure 2.2: Diagrama de la RFL [2]

Al observar la Figura 2.1 se puede modelar un diagrama que modela el dinamismo del sistema, donde se muestra el módulo SRV02 junto a V_m que corresponde al voltaje del motor, τ que corresponde al torque generado por el motor, α que corresponde al ángulo de flexión de la RFL, θ que corresponde al ángulo de rotación en torno al eje del motor, B_{eq} y B_l que corresponden a los coeficientes de fricción viscosa, el primero se opone al torque generado por el motor y el segundo al que actúa sobre la barra, y por último la RFL se modela como un sistema con constante elástica K_s .

2.2. Componentes de la planta

La RFL representada en la Figura 2.2 se compone de los siguientes elementos que muestran en las Figuras 2.3 y 2.4 identificados en el manual de usuario de la RFL [1] y serán mencionados a continuación.

En la Tabla 2.1 se mencionan los componentes de la planta y luego se define su funcionalidad.

Table 2.1: Componentes de la Planta [1]

ID	Componente
1	Planta SRV02
2	Módulo FLEXGAGE
3	Enlace FLEXGAGE
4	Galga Extensométrica
5	Circuito de Galga Extensométrica
6	Tornillos de Mariposa
7	Conector del Sensor
8	Potenciómetro de OFFSET
9	Potenciómetro de GAIN

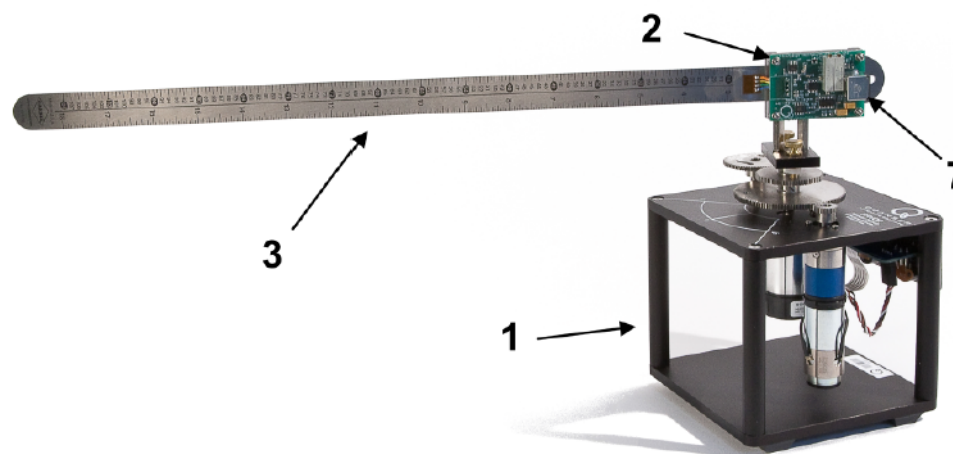


Figure 2.3: Barra Giratoria acoplada a Motor SRV02 [1]

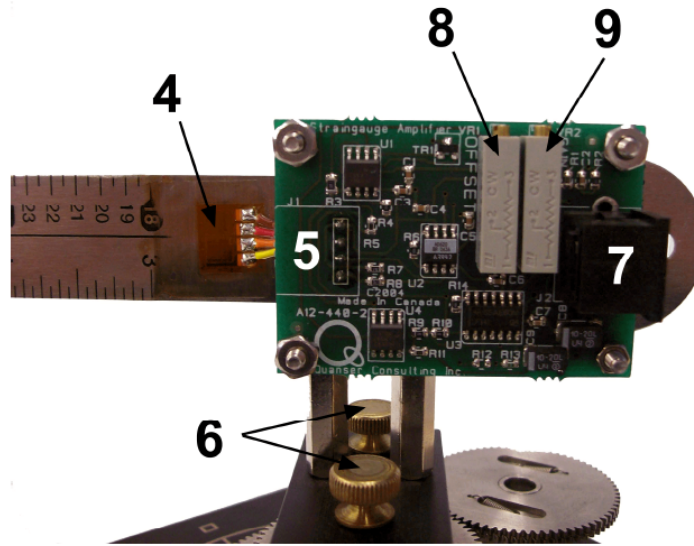


Figure 2.4: Galga Extensométrica [1]

- **1.- Planta SRV02 :** Sistema de brazo rotativo que gira en torno a su eje y permite los movimientos angulares.
- **2.- Módulo FLEXGAGE :** Proporciona información precisa sobre la deformación del enlace, permitiendo al sistema de control ajustar y mantener la posición deseada del enlace flexible durante su movimiento.
- **3.- Enlace FLEXGAGE :** Barra giratoria que rota por los movimientos del servomotor SVR02 y a razón de los desplazamientos se generan vibraciones que son medidas por la Galga Extensométrica.
- **4.- Galga Extensométrica :** Sensor utilizado para medir la deformación en un objeto cuando se le aplica una fuerza o se somete a una carga.
- **5.- Circuitos de Galga Extensométrica :** Mide los cambios en la resistencia eléctrica de la galga extensométrica cuando se deforma el objeto al que está adherida.
- **6.- Tornillos de Mariposa :** Fijan la galga extensométrica a la base del servomotor SRV02.
- **7.- Conector del Sensor :** Se conecta con el amplificador VoltPAQ-X1, X2 o X4 para transmitir la información que luego es enviada al dispositivo de adquisición de datos.

- **8.-Potenciómetro de OFFSET** : Ajusta el nivel de voltaje de salida cuando la galga extensométrica no está sometida a ninguna fuerza o deformación. Permite establecer un punto de referencia o "cero" para la medición de la deformación, asegurando que la señal de salida sea cero cuando no hay deformación en el objeto.
- **9.- Potenciómetro de GAIN** : Ajusta la sensibilidad de la señal de salida en función de la deformación del objeto. Permite calibrar la relación entre la deformación del objeto y el cambio en la resistencia de la galga extensométrica, ajustando así la escala de medición de la señal de salida.

2.3. Modelo de ecuaciones de movimiento

El sistema RFL tiene como respuesta en su salida α , como el ángulo de flexión de la barra, es un modelo oscilatorio, se debe de modelar las ecuaciones que describan el movimiento del sistema para poder realizar simulaciones adecuadas y estudiar el comportamiento para poder efectuar acciones de control sobre la planta.

Se decide emplear un modelo Lagrangiano, la ecuación de Euler-Lagrange [11] para describir tanto los movimientos del servomotor como la barra con respecto al voltaje del motor en base a las energías de este, siendo las cinéticas y potenciales tanto lineales como rotacionales en base a las coordenadas generalizadas, que corresponden a los ángulos de rotación y flexión del sistema θ y α , considerando las fuerzas no conservativas que interactúan con el sistema y se le asocia la siguiente ecuación de Euler-Lagrange:

$$\frac{\partial^2 L}{\partial t \partial \dot{q}_i} - \frac{\partial L}{\partial q_i} = Q_i \quad (2.1)$$

Donde q_i son las coordenadas generalizadas y se representa de la siguiente manera:

$$q(t) = \begin{bmatrix} \theta(t) \\ \alpha(t) \end{bmatrix} \quad (2.2)$$

Y las velocidades $\dot{Q}_i$ se expresan de la siguiente manera:

$$\dot{q}(t) = \begin{bmatrix} \dot{\theta}(t) \\ \dot{\alpha}(t) \end{bmatrix} \quad (2.3)$$

Cabe señalar que se emplea la convención de que el termino $\dot{x} = \frac{\partial x}{\partial t}$ representa a la derivada de una variable, en este caso de la posición con respecto al tiempo.

Ya que se tienen dos coordenadas generalizadas, θ y α , se plantean dos ecuaciones de Euler-Lagrange y se expresan de la siguiente manera:

$$\begin{aligned} \frac{\partial^2 L}{\partial t \partial \dot{\theta}} - \frac{\partial L}{\partial \theta} &= Q_1 \\ \frac{\partial^2 L}{\partial t \partial \dot{\alpha}} - \frac{\partial L}{\partial \alpha} &= Q_2 \end{aligned} \quad (2.4)$$

Por otra parte, el sistema lagrangiano que equivale a la diferencia entre la energía cinética (T) y la energía potencial (V) del sistema a modelar:

$$L = T - V \quad (2.5)$$

Donde la energía cinética (T), puede referirse a la energía cinética traslacional en función de la velocidad lineal (v) y la masa (m) del elemento. La energía cinética rotacional en función del momento de inercia (J) y la velocidad angular (ω) se representan como:

$$\begin{aligned} T &= \frac{1}{2}mv^2 \\ T &= \frac{1}{2}J\omega^2 \end{aligned} \quad (2.6)$$

En base a las ecuaciones planteadas se puede determinar para el sistema RFL la energía cinética rotacional, ω corresponde a la velocidad angular del ángulo de rotación y se denota como $\dot{\theta}$ junto a la inercia rotacional (J_{eq}), a la cual se le suma la energía cinética rotacional de la deformación de la barra giratoria y resultando en:

$$T = \frac{1}{2}J_{eq}\dot{\theta}^2 + \frac{1}{2}J_l(\dot{\alpha}^2 + \dot{\theta}^2) \quad (2.7)$$

Por otra parte la energía potencial (V), se puede referir a la energía potencial gravitatoria en función de la masa (m), la altura (h) a la que se encuentra con respecto a la referencia cero y la gravedad (g). También se puede denominar como energía potencial elástica, la cual corresponde a la energía almacenada en un resorte como lo es el caso que se plantea el modelo de la RFL mediante una constante de elasticidad (K) y un diferencial de distancia que equivale a la distancia que se deforme el resorte con respecto a su posición natural (Δx) y se describen mediante la siguiente expresión respectivamente:

$$\begin{aligned} V &= mgh \\ V &= \frac{1}{2}K\Delta x^2 \end{aligned} \quad (2.8)$$

Luego, se puede realizar el mismo procedimiento para determinar la energía potencial elástica para el sistema RFL, la distancia deformada Δx corresponde al ángulo de flexión α , se modela como:

$$V = \frac{1}{2}K_s\alpha^2 \quad (2.9)$$

Y se emplean las fuerzas generalizadas Q_i para describir a las fuerzas no conservativas que influyen y afectan a cada coordenada, estas se representan en base a parámetros e interacciones hacia las coordenadas de la siguiente manera:

$$\begin{aligned} Q_1 &= \tau - B_{eq}\dot{\theta} \\ Q_2 &= -B_l\dot{\alpha} \end{aligned} \quad (2.10)$$

Y el torque [12] que interactúa sobre la base de la barra giratoria se modela en base a las característica extraídas del manual de usuario del servo motor [13] se puede definir como:

$$\tau = \frac{\eta_g K_g \eta_m k_t (V_m - K_g K_m \dot{\theta})}{R_m} \quad (2.11)$$

Con las expresiones anteriores se puede sustituir en la ecuación (2.4) con el objetivo de determinar un sistema en variables de estado que represente de manera adecuada al sistema en función de los ángulos de rotación (θ) y de flexión (α) de la RFL.

Teniendo en cuenta como entrada el voltaje correspondiente a la actuación $u(t)$ y considerando

el ángulo de rotación $\theta(t)$ y el ángulo de flexión $\alpha(t)$ como las salidas del sistema, se puede clasificar este sistema como de Única Entrada y Múltiples Salidas (SIMO).

2.4. Representación en variables de estado

Con lo mencionado en el sub-capítulo anterior, se puede determinar un sistema en variables de estado que represente de manera apropiada el comportamiento de la planta siguiendo el modelo planteado en un comienzo, para aquello se determina que el lagrangiano corresponde a la diferencia entre las expresiones (2.7) y (2.9):

$$L = T - V$$

$$L = \frac{1}{2}J_{eq}\dot{\theta}^2 + \frac{1}{2}J_l(\dot{\alpha}^2 + \dot{\theta}^2) - \frac{1}{2}K_s\alpha^2 \quad (2.12)$$

En primera instancia se calculan las derivadas parciales respectivas para el ángulo de rotación θ :

$$\frac{\partial L}{\partial \theta} = 0$$

$$\frac{\partial L}{\partial \dot{\theta}} = J_{eq}\dot{\theta} + J_l(\dot{\alpha} + \dot{\theta}) \quad (2.13)$$

$$\frac{\partial^2 L}{\partial t \partial \dot{\theta}} = J_{eq}\ddot{\theta} + J_l(\ddot{\alpha} + \ddot{\theta})$$

En segunda instancia se calculan las derivadas parciales respectivas para el ángulo de flexión α resultando en:

$$\frac{\partial L}{\partial \alpha} = K_s\alpha$$

$$\frac{\partial L}{\partial \dot{\alpha}} = J_l(\dot{\alpha} + \dot{\theta}) \quad (2.14)$$

$$\frac{\partial^2 L}{\partial t \partial \dot{\alpha}} = J_l(\ddot{\alpha} + \ddot{\theta})$$

Considerando las expresiones calculadas en las ecuaciones (2.13) y (2.14) se sustituyen en la ecuación (2.4) y se iguala cada una a la ecuación de fuerzas generalizadas (2.10) para plantear un sistema de ecuaciones en función de θ , α y sus derivadas para luego determinar el sistema

en variables de estado.

$$\begin{aligned}(J_{eq} + J_l)\ddot{\theta} + J_l\ddot{\alpha} - \tau + B_{eq}\dot{\theta} &= 0 \\ J_l(\ddot{\alpha} + \ddot{\theta}) + K_s\alpha + B_l\dot{\theta} &= 0\end{aligned}\tag{2.15}$$

Se restan las ecuaciones en (2.15) para eliminar la variable $\ddot{\alpha}$ y luego se despeja $\ddot{\theta}$ resultando en :

$$\ddot{\theta} = -\frac{B_{eq}}{J_{eq}}\dot{\theta} + \frac{K_s}{J_{eq}}\alpha + \frac{\tau}{J_{eq}}\tag{2.16}$$

Ahora para determinar $\ddot{\alpha}$ se sustituye lo obtenido anteriormente de la ecuación (2.16) en (2.15) y se obtiene que $\ddot{\alpha}$ corresponde a:

$$\ddot{\alpha} = \frac{B_{eq}}{J_{eq}}\dot{\theta} - \frac{K_s(J_l + J_{eq})}{J_l J_{eq}}\alpha - \frac{\tau}{J_{eq}}\tag{2.17}$$

Con todo lo anterior se procede a determinar el sistema en variables de estado [14] de la RFL, el cual sigue la siguiente estructura para sistemas lineales e invariantes en el tiempo:

$$\begin{aligned}\dot{x} &= Ax + Bu \\ y &= Cx + Du\end{aligned}\tag{2.18}$$

En el cual x corresponde al vector de estados del sistema, u corresponde a la entrada o actuación de la planta, las matrices de estado del sistema son: A , B , C y D . En la salida del sistema solo se medirán los ángulos θ y α , por lo que solo estos serán 1 para cada estado x_1 y x_2 respectivamente. Por otra, parte para la matriz D se cumple que es nula debido a que la entrada no afecta de manera directa en la salida sino a través de los estados. En base a lo anterior se plantea que el vector de estado y la salida del sistema son:

$$\begin{aligned}x^T &= [\theta \quad \alpha \quad \dot{\theta} \quad \dot{\alpha}] \\ y^T &= [x_1 \quad x_2]\end{aligned}\tag{2.19}$$

Se define $x_1 = \theta$, $x_2 = \dot{\theta}$, $x_3 = \alpha$, $x_4 = \dot{\alpha}$ los cuales se sustituyen en las ecuaciones 2.16 y 2.17 generándose el sistema en variables de estado a continuación:

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{K_s}{J_{eq}} & -\frac{B_{eq}}{J_{eq}} & 0 \\ 0 & -\frac{K_s(J_l + J_{eq})}{J_l J_{eq}} & \frac{B_{eq}}{J_{eq}} & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 0 \\ \frac{1}{J_{eq}} \\ -\frac{1}{J_{eq}} \end{bmatrix}, \quad (2.20)$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \quad D = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Además, se tiene que la velocidad angular del servomotor $\dot{\theta}$ y la velocidad del ángulo de flexión $\dot{\alpha}$ se pueden calcular de manera computacional por el controlador empleando las derivadas y luego aplicando un filtro respectivo.

2.5. Estabilidad del sistema

Mediante información de la matriz A del sistema en variables de estado se puede determinar la estabilidad del sistema analizando la ubicación de los polos de lazo abierto, que corresponden a los autovalores de la matriz A, para aquello se debe de resolver la ecuación característica y estudiar la naturaleza de los polos que se obtienen y provienen de:

$$\det(sI - A) = 0 \quad (2.21)$$

Al aplicar el criterio de la ecuación característica al sistema en variables de estado de la RFL se llega a:

$$\left| \begin{bmatrix} s & 0 & 0 & 0 \\ 0 & s & 0 & 0 \\ 0 & 0 & s & 0 \\ 0 & 0 & 0 & s \end{bmatrix} - \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{K_s}{J_{eq}} & -\frac{B_{eq}}{J_{eq}} & 0 \\ 0 & -\frac{K_s(J_l + J_{eq})}{J_l J_{eq}} & \frac{B_{eq}}{J_{eq}} & 0 \end{bmatrix} \right| = 0 \quad (2.22)$$

Para encontrar los polos se desarrolla la ecuación característica y se calcula el determinante aplicando el método de los cofactores y se plantea de la siguiente manera aplicando los sigu-

ientes cambios de variable:

$$\begin{aligned}\beta &= \frac{K_s}{J_{eq}} \\ \gamma &= \frac{B_{eq}}{J_{eq}} \\ \delta &= \frac{(J_l + J_{eq})}{J_l}\end{aligned}\tag{2.23}$$

El sistema resulta en el siguiente:

$$\begin{vmatrix} s & 0 & -1 & 0 \\ 0 & s & 0 & -1 \\ 0 & -\beta & s + \gamma & 0 \\ 0 & \beta\delta & -\gamma & s \end{vmatrix} = 0\tag{2.24}$$

La expresión del determinante de $[sI - A]$ resulta en:

$$\begin{aligned}|sI - A| &= s \cdot \begin{vmatrix} s & 0 & -1 \\ -\beta & s + \gamma & 0 \\ \beta\delta & -\gamma & s \end{vmatrix} - 0 \cdot \begin{vmatrix} 0 & 0 & -1 \\ 0 & s + \gamma & 0 \\ 0 & -\gamma & s \end{vmatrix} + (-1) \cdot \begin{vmatrix} 0 & s & -1 \\ 0 & \beta & 0 \\ 0 & \beta\delta & s \end{vmatrix} + 0 \cdot \begin{vmatrix} 0 & s & 0 \\ 0 & \beta & s + \gamma \\ 0 & \beta\delta & -\gamma \end{vmatrix} \\ |sI - A| &= s \cdot (s \cdot \begin{vmatrix} s + \gamma & 0 \\ -\gamma & s \end{vmatrix} - 0 \cdot \begin{vmatrix} \beta & 0 \\ \beta\delta & s \end{vmatrix} + -1 \cdot \begin{vmatrix} -\beta & s + \gamma \\ \beta\delta & -\gamma \end{vmatrix}) \\ |sI - A| &= s \cdot ((s^3 + \gamma s^2) - (\beta\gamma - \beta\delta s - \beta\delta\gamma)) \\ 0 &= s^4 + \gamma s^3 + \beta\gamma s^2 + (\beta\delta\gamma - \beta\gamma)s\end{aligned}\tag{2.25}$$

Se puede apreciar un polo en $s = 0$, ya que los coeficientes resultan positivo al sustituirse por los parámetros de la planta en los cambios de variable efectuados, se afirma que los polos restantes pertenecen al semiplano izquierdo ya que poseen parte real negativa, según el criterio de Routh-Hurwitz [15], por todo lo anterior se concluye que el sistema es marginalmente estable, esto se puede asociar a la respuesta oscilatoria del sistema.

2.6. Controlabilidad y Observabilidad

En base a las matrices del sistema en variables de estado de la RFL, se puede estudiar la controlabilidad y la observabilidad [15]. Lo primero que se analizará será la controlabilidad, lo cual es fundamental ya que es la capacidad que se tiene para llevar el sistema desde cualquier estado inicial hacia cualquier estado deseado, ya que es posible encontrar una entrada de control que permita dirigir el sistema hacia un estado en específico sin importar el valor de la condición inicial que se tenga. Esto se relaciona directamente con la matriz de controlabilidad del sistema que se forma por la matriz de coeficientes de un sistema dinámico y se determina de la siguiente manera:

Para un sistema en variables de estado de la forma:

$$\dot{x} = Ax + Bu \quad (2.26)$$

La matriz de controlabilidad se determina como:

$$CNT = \begin{bmatrix} B & AB & A^2B & \dots & A^{n-1}B \end{bmatrix} \quad (2.27)$$

Donde n corresponde al orden del sistema y la condición que se debe cumplir para que el sistema sea controlable, es que el rango de la matriz de controlabilidad CNT sea igual a la cantidad de estados del sistema.

La controlabilidad se evalúa mediante el Código *Controlabilidad para la RFL* de la Sección 11.1 de los Anexos.

Al ejecutarlo con los valores respectivos para los parámetros de la planta se cumple que el sistema es completamente controlable.

Por otra parte para analizar la observabilidad del sistema (2.18) que consiste en la capacidad de determinar el estado interno de un sistema a partir de observar sus salidas, se cumple que un sistema es completamente observable si es que es posible conocer el estado actual y estados pasados del sistema empleando las mediciones de las salidas. A través de la matriz

de observabilidad:

$$OBS = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \end{bmatrix} \quad (2.28)$$

Donde n corresponde al orden del sistema y se debe de cumplir que el rango de la matriz de observabilidad OBS debe ser igual a la cantidad de estados de la matriz A y se puede utilizar el siguiente Código *Observabilidad para la RFL* de la Sección 11.2 de los Anexos.

Al ejecutar el código anterior y considerando los parámetros de la planta *RFL* se logra afirmar que el sistema es completamente observable.

2.7. Discretización

El control digital es una técnica que se utiliza ampliamente en una variedad de aplicaciones, desde sistemas de control de procesos industriales hasta sistemas de control de vehículos y dispositivos electrónicos de consumo. La capacidad de procesar señales de forma digital ofrece varias ventajas sobre los métodos analógicos tradicionales [16].

Uno de los aspectos clave del control digital es la conversión de señales analógicas a digitales y viceversa. El proceso comienza con el muestreo de la señal analógica mediante un conversor analógico-digital (A-D). Este conversor toma muestras de la señal a intervalos regulares, lo que permite la representación discreta de la señal continua. La precisión del muestreo es crucial para garantizar que la señal digital resultante capture con precisión la información de la señal analógica original.

Una vez que la señal se ha convertido en digital, puede ser procesada por un sistema de control digital, que puede incluir algoritmos de control, lógica de procesamiento y otros componentes electrónicos. Estos componentes operan en tiempo discreto, lo que significa que realizan cálculos y toman decisiones basadas en las muestras digitales de la señal en intervalos de tiempo discretos.

Después de que el sistema de control digital ha procesado la señal, puede ser necesario volver

a convertirla a formato analógico para su salida al sistema físico que se está controlando. Esto se logra mediante un conversor digital-analógico (D-A), que reconstruye la señal analógica a partir de los datos digitales muestreados. Para garantizar una reconstrucción precisa de la señal, se suele utilizar un retentor de orden cero (ZOH), que mantiene constante cada muestra digital durante un intervalo de tiempo antes de pasar a la siguiente muestra [16].

El esquema más común de control digital incluye estos dos conversores junto con otros componentes como el procesador de señales digitales (DSP), la memoria digital y las interfaces de entrada y salida y se aprecia en la Figura 2.5. Este enfoque de control ofrece ventajas como una mayor precisión, flexibilidad y capacidad de procesamiento, lo que lo hace indispensable en el mundo moderno de la automatización y el control. Además, el control digital permite la implementación de algoritmos avanzados de control y la integración con sistemas informáticos para análisis y supervisión en tiempo real [16].

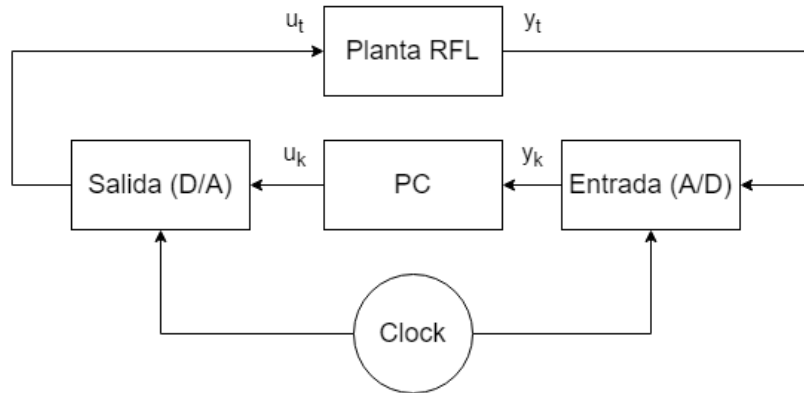


Figure 2.5: Diagrama de control digital-análogo

Para poder discretizar un modelo de un sistema en variables de estado de tiempo continuo (2.18) se puede expresar un sistema discreto genérico de la siguiente manera:

$$x_{k+1} = A_d x_k + B_d u_k \quad (2.29)$$

$$y_k = C_d x_k \quad (2.30)$$

La sincronización perfecta entre un conversor A-D y un D-A, según se menciona en la referencia [16], permite obtener las entradas y salidas en un mismo instante t_k luego del muestreo,

lo que resulta en:

$$x(t_k + 1) = \Phi x(t_k) + \Gamma u(t_k) \quad (2.31)$$

$$y(t_k) = Cx(t_k) + Du(t_k) \quad (2.32)$$

Aquí, $\Phi = A_d$ y $\Gamma = B_d$ siendo constantes en todo instante, mientras que C no se ve afectada ya que es igual para las salidas, y para sistemas causales, D es nula.

Para obtener las matrices discretizadas, se pueden establecer las siguientes relaciones:

$$A_d = \mathcal{L}^{-1}\{(sI - A_c)^{-1}\}\Big|_{t=T_s} = e^{A_c T_s} \quad (2.33)$$

$$B_d = \int_0^{T_s} A_d(\tau) d\tau \cdot B_c \quad (2.34)$$

Aquí, se emplea la transformada inversa de Laplace evaluada en un instante T_s , que corresponde al tiempo de muestreo. La integral definida se utiliza para calcular la matriz discreta B_d a partir de las matrices discreta y continua A_d y B_c respectivamente.

Todo esto se explica debido a que la metodología de control que se plantea utilizar es en tiempo discreto, y se verá mas adelante en el Capítulo 4.

3. Controlador LQR para RFL

3.1. Formulación de LQR

Para implementar acciones de control mediante LQR [15] como lo realiza la empresa Quanser en el controlador que diseñan para trabajar con la RFL el cual es estudiado para el entendimiento del problema de control, se debe trabajar con un sistema lineal que pueda ser representado en variables de estado. Luego se debe verificar la controlabilidad del sistema para asegurar la capacidad de llegar a cualquier estado, habiendo estudiado previamente la estabilidad del sistema analizando los polos de este. Para el caso de la RFL, en el capítulo anterior se estudió estos requerimientos por lo que se procede a implementar LQR.

El control LQR tiene como objetivo minimizar la función de costo cuadrático que combina el estado y las entrada a lo largo del tiempo:

$$J(x(t), u(t)) = \int_{t=0}^{t=\infty} (x(t)^T Q x(t) + u(t)^T R u(t)) dt \quad (3.35)$$

Donde Q y R corresponden a las matrices de ponderación, Q pondera la importancia de minimizar los estados y R pondera la importancia de minimizar las entradas en sí de los efectos que generan al minimizar la función de costo $J(t)$.

Se emplea la ley de control óptimo 3.36 que indica la relación matemática entre el vector de entrada $u(t)$ y el vector de estado $x(t)$ que minimiza la función de costo cuadrático asociada al sistema lineal:

$$u = -Kx \quad (3.36)$$

Donde K corresponde a la ganancia de retroalimentación de estado, la cual se determina al emplear la ecuación de Riccati, calculando la matriz P que es simétrica y positiva definida y corresponde a la solución de la ecuación, donde A , B , Q y R son las matrices del sistema y las matrices de ponderación asociadas a los estados y a las entradas respectivamente:

$$A^T P + P A - P B R^{-1} B^T P + Q = 0 \quad (3.37)$$

Y la ganancia del controlador K se define como:

$$K = R^{-1}B^T P \quad (3.38)$$

al sustituir en la ecuación 3.36 se indica que la entrada u es proporcional y opuesta a los estados del sistema, y dicha proporcionalidad corresponde a la ganancia K , utilizándola para minimizar el costo cuadrático proporcionando un control óptimo y eficiente para los sistemas lineales.

En la Figura 3.6 se muestra el esquema de control LQR implementado para la planta RFL, en donde se tiene como objetivo alcanzar una posición deseada para el ángulo θ por lo que el estado al cual se busca llegar sería el siguiente $x_d = [\theta_d \ 0 \ 0 \ 0]^T$:

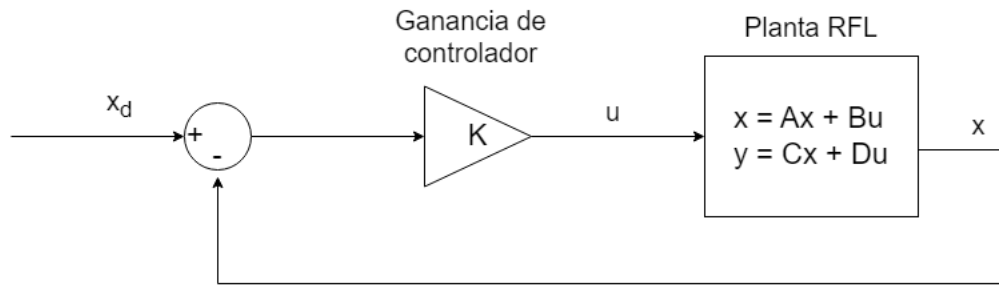


Figure 3.6: Lazo de control del estado del sistema

La actuación del sistema es la diferencia entre el estado deseado x_d y el estado actual del sistema x en producto con la ganancia del controlador K y eso se refleja en el esquema de lazo cerrado.

3.2. Efectos de las ganancias en el controlador y sistema

La empresa Quanser diseñó un controlador LQR para implementar en la RFL, el cual se presenta en el *Student WorkBook de la RLF* [10]. El cual genera un código en MATLAB y emplea diagrama de Simulink junto a sus componentes de QUARC para poder representar de manera óptima las simulaciones y la implementación física con la planta. Para ello se emplea un código de MATLAB mediante la función $(K, S, P) = lqr(A, B, Q, R)$ [17] que permite determinar la ganancia K del controlador, S como la solución de la ecuación de Ricatti y P

como los polos de lazo cerrado del sistema.

Con Q como matriz diagonal con valores q_i desde $i = 1$ hasta $i = 4$, en un inicio son todos iguales y unitarios, R de orden 1 y coeficiente R y también se tiene una matriz con coeficientes k_i desde $i = 1$ hasta $i = 4$, todo esto se representa como:

$$Q = \begin{bmatrix} q_1 & 0 & 0 & 0 \\ 0 & q_2 & 0 & 0 \\ 0 & 0 & q_3 & 0 \\ 0 & 0 & 0 & q_4 \end{bmatrix} \quad (3.39)$$

$$K = \begin{bmatrix} k_1 & k_2 & k_3 & k_4 \end{bmatrix}$$

Y junto con las matrices A y B del sistema en variables de estado mediante el siguiente código se determina la ganancia K , mediante pruebas se logra determinar que los coeficientes de la matriz Q , generan los siguientes efectos en la salida:

- Aumentar q_1 genera que la respuesta sea más rápida.
- Aumentar q_2 no genera muchos efectos.
- Aumentar q_3 minimiza el overshoot en la respuesta pero provoca que el sistema sea más lento.
- Aumentar q_4 minimiza la desviación de la barra.

El código en MATLAB permite simular las acciones de control sobre la RFL, cuyo esquema se muestra en la Figura 3.7. El funcionamiento es el siguiente: mediante un generador de señales se ingresa una señal que se amplifica por una constante, determinando la posición de θ deseada como referencia. Esta posición se encuentra en grados, por lo que debe convertirse de grados a radianes. Esta entrada se convierte en un vector para el sistema y se ingresa al sumador.

La señal se resta con la realimentación del estado (decidiendo si se emplean todos los elementos del estado o solo los relacionados con el ángulo de rotación) y se multiplica por la ganancia K del controlador. A este producto se le aplica un bloque de saturación para el voltaje del motor,

que por condiciones de funcionamiento se encuentra entre $\pm 10V$. Este voltaje limitado por la saturación actúa sobre el sistema, obteniéndose como salida las posiciones y determinándose las velocidades para cada una mediante un filtro respectivo.

Luego se escoge qué señal observar, siendo estas θ y α , las cuales se convierten de radianes a grados y se observan mediante el bloque *scope* y su señal.

Además, se establecen los siguientes requerimientos que debe cumplir el sistema y que serán considerados para el diseño del controlador:

- El voltaje del motor debe encontrarse entre $\pm 10[V]$ para su correcto funcionamiento.
- El ángulo de rotación de la RFL θ debe encontrarse entre $-90[^\circ]$ y $90[^\circ]$.
- El ángulo de flexión de la RFL α debe encontrarse entre $\pm 8[^\circ]$.
- El tiempo para que se estabilice $T_{set}[s]$ del sistema debe ser inferior a $0.7[s]$.

En los Anexos en la sección 11.3 se encuentra el código del control LQR, analizando la controlabilidad, observabilidad y estabilidad de la planta.

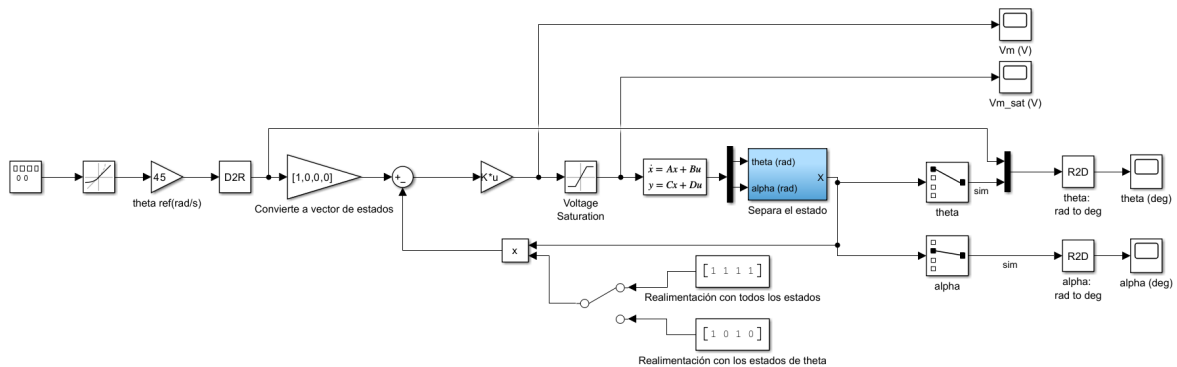


Figure 3.7: Esquema de control de RFL en Simulink

El código en MATLAB permite simular las acciones de control sobre la RFL, cuyo esquema se muestra en la Figura 3.7. El funcionamiento es el siguiente: mediante un generador de señales, se ingresa una señal que se amplifica por una constante, determinando la posición de θ deseada como referencia. Esta posición se encuentra en grados, por lo que debe convertirse de grados a radianes utilizando el bloque *Degrees 2 Radians* (D2R). Esta entrada se convierte en un vector

para el sistema y se ingresa al sumador. La señal se resta con la realimentación del estado (decidiendo si se emplean todos los elementos del estado o solo los relacionados con el ángulo de rotación) y se multiplica por la ganancia K del controlador. A este producto se le aplica un bloque de saturación para el voltaje del motor, que por condiciones de funcionamiento se encuentra entre -10 [V] y 10 [V]. Este voltaje limitado por la saturación actúa sobre el sistema, obteniéndose como salida las posiciones y determinándose las velocidades para cada una mediante un filtro respectivo. Luego se escoge qué señal observar, siendo estas θ y α , las cuales se convierten de radianes a grados utilizando el bloque *Radians 2 Degrees* (R2D) y se observan mediante el bloque *scope* y su señal. Se emplean también un seleccionador de estados para decidir qué estados se consideran en la realimentación y el bloque *Separa el estado*, que permite calcular las salidas y_3 e y_4 correspondientes a las velocidades.

Se observa mediante los bloques *Scope*, el voltaje del motor, el ángulo de rotación junto a la referencia y el ángulo de flexión en las siguientes figuras.

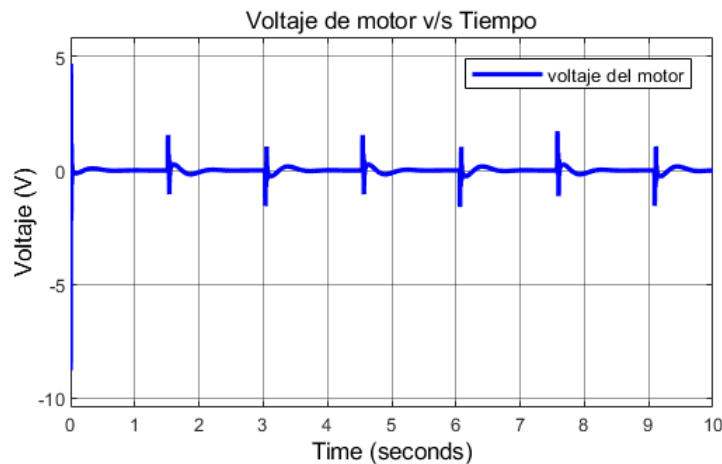


Figure 3.8: Voltaje del motor

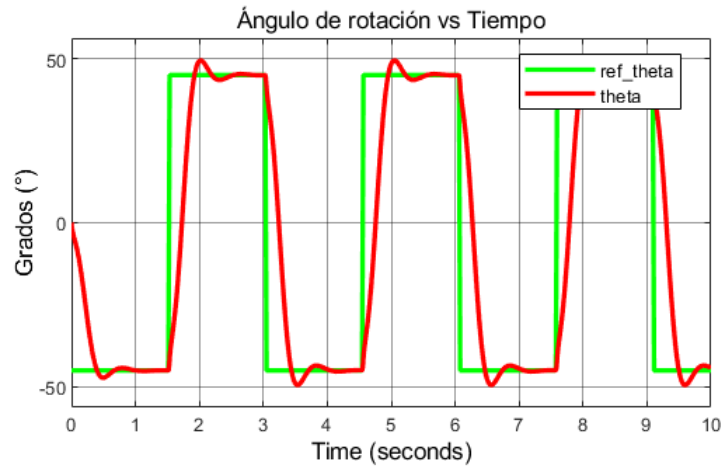


Figure 3.9: Referencia deseada (verde), ángulo de rotación de la RFL (rojo)

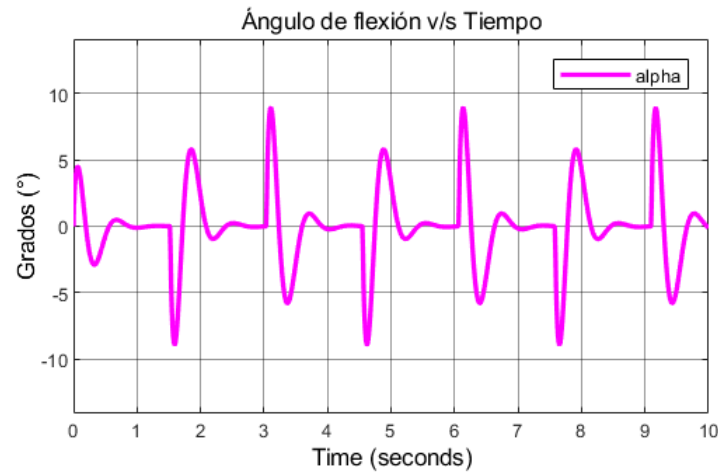


Figure 3.10: Ángulo de flexión de la RFL

De la Figura 3.8 se observa que la actuación se encuentra y se mantiene en el margen permitido. Al comienzo se observa una fuerte actuación de arranque y luego pequeños impulsos que efectúan los cambios en el ángulo de rotación $\theta = y_1$. En la Figura 3.9 se observa la referencia (verde) y el ángulo de rotación (rojo); se sigue de manera adecuada, teniendo un pequeño sobreimpulso permitido según las condiciones propuestas, y el tiempo de ajuste también es el apropiado. Por último, con base en la Figura 3.10, se observa que el valor máximo no supera los $\pm 8^\circ$, que se indicaron como condiciones de funcionamiento, y es periódica debido al tipo de referencia que actúa sobre el sistema.

4. Control predictivo por modelo

El Control Predictivo por Modelo o MPC corresponde a un método de control avanzado, en el cual se implementa la optimización en base al modelo de una planta. De este modelo se predice el comportamiento que tendrá el sistema en el futuro. Empleando las predicciones del modelo de la planta, se procede a calcular las acciones de control óptimas con el propósito de minimizar la función de costo cuadrático y seguir una referencia durante el tiempo.

En la metodología empleada para MPC se pueden distinguir cuatro componentes principales para ejercer control MPC, y corresponden a los siguientes:

- **Modelo matemático del sistema:** Se requiere que el sistema sea lineal y que se encuentre discretizado para determinar las dinámicas del sistema.
- **Horizonte de predicción de N muestras:** Se escoge uno apropiado según las dinámicas de la planta para decidir la cantidad de acciones de control futuras que se deberán calcular para ejercer control.
- **Conjunto de restricciones de estados y de la actuación del sistema:** Corresponden a las limitaciones necesarias requeridas para lograr una optimización coherente de acuerdo a parámetros físicos de la planta.
- **Función de costo:** Se define como el problema de optimización que suele ser con normalidad cuadrático, este penaliza las desviaciones presentes en los estados y en las actuaciones según se requiera para el control.

A continuación se presenta el esquema de MPC empleado con regularidad que se expone en la Figura 4.11.

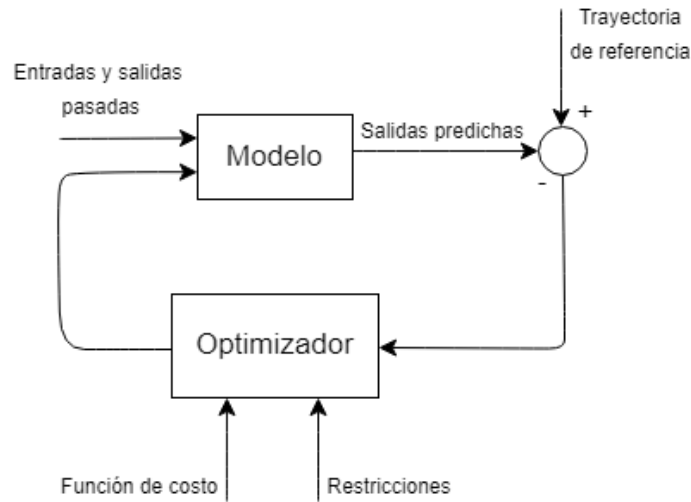


Figure 4.11: Esquema de control MPC

Se observa en la Figura 4.11 el bloque Modelo y el bloque Optimizador, esto suele incluir adicionalmente observadores para medir el estado actual de la planta, que se corresponde a una entrada para el bloque Optimizador.

4.1. Metodología del control predictivo

El control predictivo por modelo (MPC) se basa en la optimización del modelo de la planta de manera iterativa dentro de un horizonte de predicción finito (N). Además, en un período de muestra discreto para un instante k , se mide o estima el estado actual de la planta x_k utilizando observadores. Al medir el estado y el modelo matemático de la planta, es posible predecir la secuencia de acciones futuras de control. Empleando estas predicciones de control, se aplica solo la primera entrada $u_{opt}(k)$ al sistema y se calcula de nuevo el valor óptimo siguiente para el instante $k + 1$ dentro del horizonte de control ya definido, lo que se conoce como el método de horizonte móvil. Las acciones de control deben minimizar el problema de optimización y cumplir las restricciones impuestas para la actuación y los estados.

Para complementar la comprensión de la metodología de MPC, se presenta el método de horizonte móvil en la Figura 4.12 y la Figura 4.13.

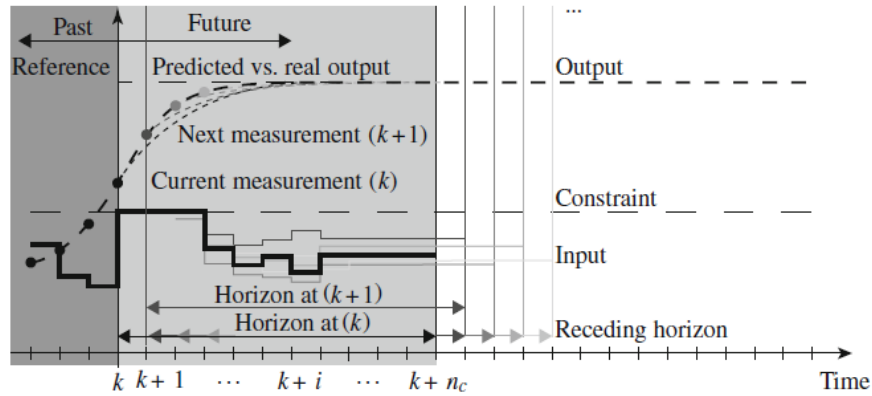


Figure 4.12: Horizonte de control móvil [3]

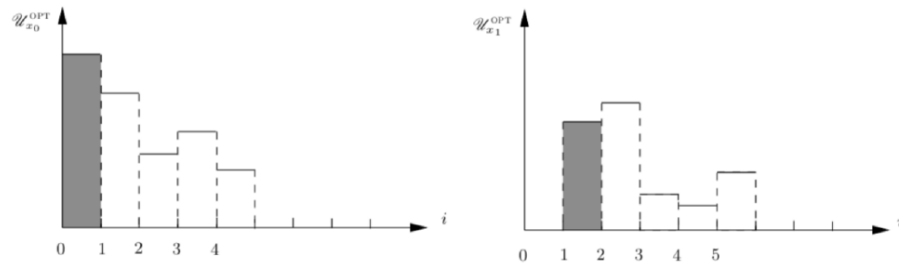


Figure 4.13: Cálculo de actuación u_{opt} en Horizonte de control móvil [3]

En la Figura 4.12 se muestra la metodología de funcionamiento del MPC con horizonte móvil de predicción, también conocido por sus siglas en inglés RHC. En la sección de entrada se representan las cantidades de actuaciones $u(k)$ que interactúan con la planta, así como los vectores de predicción óptimos que varían en el tiempo. La secuencia aplicada corresponde a la primera entrada.

Se muestra en la Figura 4.13 el uso de un horizonte de predicción $N = 5$. Inicialmente, en el instante $t = 0$, se aplica la actuación $u_{opt}(0)$ correspondiente al área sombreada entre los instantes $t = 0$ y $t = 1$. Sin embargo, en el instante $t = 1$ se observa un cambio en la actuación $u_{opt}(1)$ calculada previamente, así como en los siguientes vectores de predicción. Por lo tanto, se vuelve a aplicar solo la primera entrada y se repite el proceso para los siguientes instantes en los horizontes establecidos.

Es importante destacar que la optimización del MPC es una respuesta en lazo abierto del

sistema. En esencia, el MPC no es un control realimentado, pero su estrategia lo simula al permitir la realimentación con errores futuros, lo que le da al controlador la capacidad de seguir referencias.

4.2. Programación cuadrática (QP)

Se pueden resolver problemas lineales y no lineales mediante el control predictivo. En general, cuando se trabaja con un sistema lineal o dentro de un rango de operación lineal, es posible formular problemas cuadráticos, también conocidos como Programación Cuadrática (QP por sus siglas en inglés), los cuales se utilizan en el desarrollo de la presente memoria.

El problema cuadrático general, expresado de manera matricial y sujeto a restricciones de igualdad y desigualdad, se presenta mediante las ecuaciones (4.40) y (4.41).

$$\min J(x) = \frac{1}{2}x^T Hx + h^T x \quad (4.40)$$

Sujeto a las restricciones lineales:

$$\begin{aligned} Fx &= f \\ Gx &\leq g \end{aligned} \quad (4.41)$$

Tanto H como h , las matrices del problema cuadrático, dependen de las matrices de predicción del sistema, como se desarrolla en las secciones siguientes. Es importante destacar que H debe ser semipositiva definida para que el problema sea convexo. La importancia de que el problema sea convexo radica en la capacidad de encontrar un mínimo global y, por ende, el valor óptimo real [3].

4.3. Formulación problema de control

Para formular el problema de control MPC como un problema de Programación Cuadrática (QP), se considera el sistema descrito en la ecuación (4.42), que es de naturaleza lineal, discreto e invariante en el tiempo.

$$\begin{aligned} x_{k+1} &= A_d x_k + B_d u_k \\ y_k &= C_d x_k \end{aligned} \quad (4.42)$$

donde $x_k \in \mathbb{R}^n$ representa el estado en el instante k , con n siendo el número de estados, $u_k \in \mathbb{R}^m$ representa la entrada en el instante k , con m siendo el número de entradas, y $y_k \in \mathbb{R}^p$ representa la salida en el instante k , con p siendo el número de salidas.

Para relacionar la función cuadrática QP de la ecuación (4.40) con el sistema de la ecuación (4.42) y así lograr un seguimiento de referencia, se define el vector de error e_k como:

$$e_k = r_k - y_k \quad (4.43)$$

donde r_k es la referencia en el instante k . Para simplificar, se asume inicialmente $r_k = 0$, lo que implica que $e_k = -y_k$, es decir, se busca llevar el sistema a su origen. Esto se logra penalizando cuadráticamente los estados.

Ya que el objetivo es llevar al sistema a su origen, se aplica la penalización cuadrática en función de los estados del sistema (4.40), y al reemplazar con el vector de error se obtiene:

$$\begin{aligned} J_k &= y_k^T y_k \\ J_k &= x_k^T C_k^T C_k x_k \\ \Omega &= C_k^T C_k \end{aligned} \quad (4.44)$$

De manera similar, se busca penalizar las entradas del sistema para evitar cambios bruscos que puedan afectar la actuación de la planta, como se muestra en la ecuación (4.45).

$$J_k = x_k^T \Omega x_k + u_k^T \Gamma u_k \quad (4.45)$$

Además, se introduce la ponderación de los estados finales x_N para aproximar el comportamiento a un horizonte de tiempo infinito, lo que da lugar a la función de costo cuadrático final mostrada en la ecuación (4.46).

$$J_N(x_k, u_k) = x_N^T \Omega_N x_N + \sum_{k=0}^{N-1} (x_k^T \Omega x_k + u_k^T \Gamma u_k) \quad (4.46)$$

Finalmente, el problema de control MPC se formula utilizando la función de costo cuadrático de la ecuación (4.46), junto con el sistema en variables de estado en tiempo discreto de la

ecuación (4.42), como se muestra en la ecuación (4.47).

$$u_k^{opt} = \arg \min J_N(x_k, u_k) \quad (4.47)$$

Sujeto a las restricciones mostradas en la ecuación (4.48).

$$\begin{aligned} x_0 &= x_k \\ x_{k+1} &= A_d x_k + B_d u_k \\ u^{min} &\leq u_k \leq u^{max} \\ x^{min} &\leq x_k \leq x^{max} \\ x_N^{min} &\leq x_N \leq x_N^{max} \end{aligned} \quad (4.48)$$

Es importante destacar que se utiliza la notación x_0 para referirse al estado actual medido.

4.4. Formulación Sparse y Restricciones

Para el planteamiento matricial del problema de control anteriormente presentado en (4.47) y (4.48), la representación de la función de costo dependerá de las variables de decisión para la optimización. En esta ocasión se desarrollará el planteamiento para la Formulación Sparse que considera tanto la entrada u_k como el estado x_k .

La formulación Sparse [4] se presenta mediante el siguiente problema de optimización:

$$u_t^{opt} = \min_{\theta} J_N(x_k, \bar{u}) \quad (4.49)$$

El cual se encuentra sujeto a las siguientes restricciones:

$$\begin{aligned} x_0 &= x(k) \\ x_{k+1} &= A_d x_k + B_d u_k \\ Jx_k + Eu_k &\leq d \\ J_N x_k^N &\leq d_N \end{aligned} \quad (4.50)$$

Las ecuaciones anteriores corresponden a las restricciones de igualdad y de desigualdad del planteamiento del problema, en base a ello se definen las matrices de restricciones de la siguiente manera:

$$J = \begin{bmatrix} I_n \\ -I_n \\ C_{mn} \\ C_{mn} \end{bmatrix}, \quad E = \begin{bmatrix} C_{nm} \\ C_{nm} \\ I_m \\ -I_m \end{bmatrix}, \quad d = \begin{bmatrix} x_{\text{máx}} \\ -x_{\text{mín}} \\ u_{\text{máx}} \\ -u_{\text{mín}} \end{bmatrix} \quad (4.51)$$

$$J_N = \begin{bmatrix} I_n \\ -I_n \end{bmatrix}, \quad d_N = \begin{bmatrix} x_N^{\text{máx}} \\ -x_N^{\text{mín}} \end{bmatrix} \quad (4.52)$$

En las ecuaciones de (4.50) se presentan los rangos inferior y superior para las restricciones de estado x_k por lo que se desarrollan como:

$$x^{\text{min}} \leq x_k \leq x^{\text{max}} \quad (4.53)$$

$$x^{\text{min}} \leq x_k \quad x_k \leq x^{\text{max}} \quad (4.54)$$

Las matrices de la ecuación anterior se acomodan con tal de plantear y expresar en una misma variable:

$$-x_k \leq -x^{\text{min}} \quad (4.55)$$

$$x_k \leq x^{\text{max}} \quad (4.56)$$

Lo que se expresa mediante matrices y resulta en lo siguiente:

$$\begin{bmatrix} I \\ -I \end{bmatrix} x_k \leq \begin{bmatrix} x^{\text{max}} \\ -x^{\text{min}} \end{bmatrix} \quad (4.57)$$

Se realiza el mismo procedimiento pero esta vez para la entrada u_k cumpliéndose lo siguiente:

$$u^{\min} \leq u_k \leq u^{\max} \quad (4.58)$$

$$-u_k \leq -u^{\min} \quad (4.59)$$

$$u_k \leq u^{\max} \quad (4.60)$$

Y se escribe de la siguiente forma matricial:

$$\begin{bmatrix} I \\ -I \end{bmatrix} u_k \leq \begin{bmatrix} u^{\max} \\ -u^{\min} \end{bmatrix} \quad (4.61)$$

Al agrupar las expresiones matriciales calculadas de la ecuación (4.57) y de la ecuación (4.61), se puede representar matricialmente el sistema resultando en lo siguiente:

$$\underbrace{\begin{bmatrix} I \\ -I \\ 0 \\ 0 \end{bmatrix}}_J x_k + \underbrace{\begin{bmatrix} 0 \\ 0 \\ I \\ -I \end{bmatrix}}_E u_k \leq \underbrace{\begin{bmatrix} x^{\max} \\ -x^{\min} \\ u^{\max} \\ u^{\min} \end{bmatrix}}_d \quad (4.62)$$

Donde $J \in \mathbb{R}^{(2n+2m) \times n}$ y n corresponde al número de estados, m al número de entradas, $E \in \mathbb{R}^{(2n+2m) \times n}$ se asocia a la matriz de entradas y finalmente la matriz $d \in \mathbb{R}^{2(n+m) \times 1}$ para las restricciones.

Se realiza el mismo procedimiento para los estados finales:

$$x_N^{\min} \leq x_N \leq x_N^{\max} \quad (4.63)$$

A continuación se separa nuevamente:

$$-x_N \leq -x_N^{\min} \quad (4.64)$$

$$x_N \leq x_N^{\max} \quad (4.65)$$

Llevando las relaciones directamente a su forma matricial resulta en:

$$\underbrace{\begin{bmatrix} I \\ -I \end{bmatrix}}_{J_N} x_N \leq \underbrace{\begin{bmatrix} x_N^{\max} \\ -x_N^{\min} \end{bmatrix}}_{d_N} \quad (4.66)$$

Donde $J_N \in \mathbb{R}^{2n \times 1}$ es equivalente para $d_N \in \mathbb{R}^{2n \times 1}$.

Para la Formulación Sparse del problema QP, se debe considerar que el vector de decisión se conforma en base a las entradas y a los estados del sistema de forma alternada, como se puede ver a continuación:

$$\theta = \begin{bmatrix} x_0^T & u_0^T & x_1^T & u_1^T & \cdots & x_{N-1}^T & u_{N-1}^T & x_N^T \end{bmatrix}^T \quad (4.67)$$

La función de costo del problema se reescribe en función del vector de decisión y resulta en lo siguiente:

$$J_N(\theta) = \begin{bmatrix} x_0 \\ u_0 \\ x_1 \\ u_1 \\ \vdots \\ x_{N-1} \\ u_{N-1} \\ x_N \end{bmatrix}^T \underbrace{\begin{bmatrix} \Omega & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & \Gamma & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & \Omega & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \Gamma & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots & \Omega \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots & \Gamma \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \end{bmatrix}}_{\mathbb{H}_N} \begin{bmatrix} x_0 \\ u_0 \\ x_1 \\ u_1 \\ \vdots \\ x_{N-1} \\ u_{N-1} \\ x_N \end{bmatrix} \quad (4.68)$$

Donde Ω es la matriz de pesos para el estado, Γ es la matriz de pesos para la entrada, y $\mathbb{H}_N \in \mathbb{R}^{(n+m) \times (n+m)}$ es una matriz diagonal de tamaño $(N+1) \cdot (n+m) \times (N+1) \cdot (n+m)$.

Finalmente, el problema de optimización es:

$$\min_{\theta} J_N(\theta) \quad (4.69)$$

con restricciones:

$$\begin{bmatrix} J & E \end{bmatrix} \begin{bmatrix} x_0 \\ u_0 \\ x_1 \\ u_1 \\ \vdots \\ x_{N-1} \\ u_{N-1} \\ x_N \end{bmatrix} \leq d \quad (4.70)$$

y

$$\begin{bmatrix} J_N \\ 0 \end{bmatrix} \begin{bmatrix} x_0 \\ u_0 \\ x_1 \\ u_1 \\ \vdots \\ x_{N-1} \\ u_{N-1} \\ x_N \end{bmatrix} \leq d_N \quad (4.71)$$

El objetivo es minimizar la función de costo con las restricciones del sistema dinámico y las limitaciones de los estados y las entradas. Para resolver el problema, se pueden utilizar algoritmos de optimización adecuados para problemas cuadráticos restringidos.

5. Redes Neuronales

Para entrenar la red neuronal, se deben recolectar datos de entrenamiento. Se plantea grabar y registrar los datos del control MPC utilizando dSPACE. Se recopilan los datos de los tres vectores de entrada mencionados y el vector de salida del controlador. El modelo de la red neuronal propuesta consta de una capa de entrada con nueve neuronas, una capa oculta con dos neuronas, y una capa de salida con una neurona. Es importante considerar la cantidad de neuronas y capas ocultas, así como la función de activación utilizada en cada capa para obtener los pesos y los sesgos adecuados.

Tras las pruebas de entrenamiento, se ajustará la cantidad de neuronas y capas ocultas en función de los requerimientos para diseñar un modelo óptimo.

Las ecuaciones matemáticas que describen el funcionamiento de una red neuronal son las siguientes:

1. Función de activación en la capa oculta:

$$h_i = \sigma \left(\sum_{j=1}^n w_{ij} x_j + b_i \right) \quad (5.72)$$

Donde:

- h_i es la salida de la neurona i en la capa oculta.
- w_{ij} es el peso de la conexión entre la neurona j de la capa de entrada y la neurona i de la capa oculta.
- x_j es la entrada j .
- b_i es el sesgo de la neurona i .
- $\sigma(\cdot)$ es la función de activación, por ejemplo, la función sigmoide o ReLU.

2. Función de activación en la capa de salida:

$$\hat{u}_k = \phi \left(\sum_{i=1}^m w_i h_i + b \right) \quad (5.73)$$

Donde:

- $\hat{u}_k$ es la salida del controlador neuronal.
- w_i es el peso de la conexión entre la neurona i de la capa oculta y la neurona de la capa de salida.
- h_i es la salida de la neurona i en la capa oculta.
- b es el sesgo de la neurona de salida.
- $\phi(\cdot)$ es la función de activación de la capa de salida, que puede ser una función lineal o sigmoide, dependiendo del tipo de problema.

Estas ecuaciones definen cómo los datos de entrada se transforman a través de la red neuronal para producir la salida deseada. La función de activación introduce no linealidades en el modelo, permitiendo que la red neuronal aprenda relaciones complejas entre las entradas y las salidas.

Se realizarán pruebas para ajustar la cantidad de neuronas y la estructura de la red en base a los resultados obtenidos durante el proceso de entrenamiento.

$$\begin{aligned} S &= WX \\ Y &= f(S) \end{aligned} \tag{5.74}$$

Donde W corresponde a la matriz de los sesgos y pesos de la red neuronal, b corresponde a los sesgos de las neuronas y w_1 a w_k corresponden a los pesos de cada neurona. En cuanto a X , corresponde a la matriz de las entradas de la red neuronal y x_1 a x_k corresponden a las

neuronas de entrada. $f(S)$ corresponde a la función de activación de la neurona.

$$W = \begin{pmatrix} b & w_1 & w_2 & \dots & w_k \end{pmatrix}$$

$$X = \begin{pmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_k \end{pmatrix} \quad (5.75)$$

El proceso de replicación de conexiones y cálculos se extiende a lo largo de cada capa de la red neuronal, desde la capa de entrada hasta la capa de salida, donde se encuentran las neuronas que generan las salidas. Este proceso tiene lugar en todas las capas intermedias, conocidas como capas ocultas, donde cada neurona recibe señales de entrada ponderadas por sus respectivos pesos y las procesa mediante funciones de activación. Este proceso iterativo de cálculo y propagación se repite hasta que la señal alcanza la capa de salida, donde se produce el resultado final de la red neuronal [18]. A continuación, se presenta en la Figura 5.14 un diagrama que ilustra cómo se configura una red neuronal. Este diagrama proporciona una representación visual de las diferentes capas de la red, incluidas las conexiones entre las neuronas en cada capa y las salidas generadas por la capa de salida. Esta visualización es útil para comprender la estructura y el funcionamiento de la red neuronal, así como para identificar las interconexiones y la complejidad del modelo matemático subyacente [19].

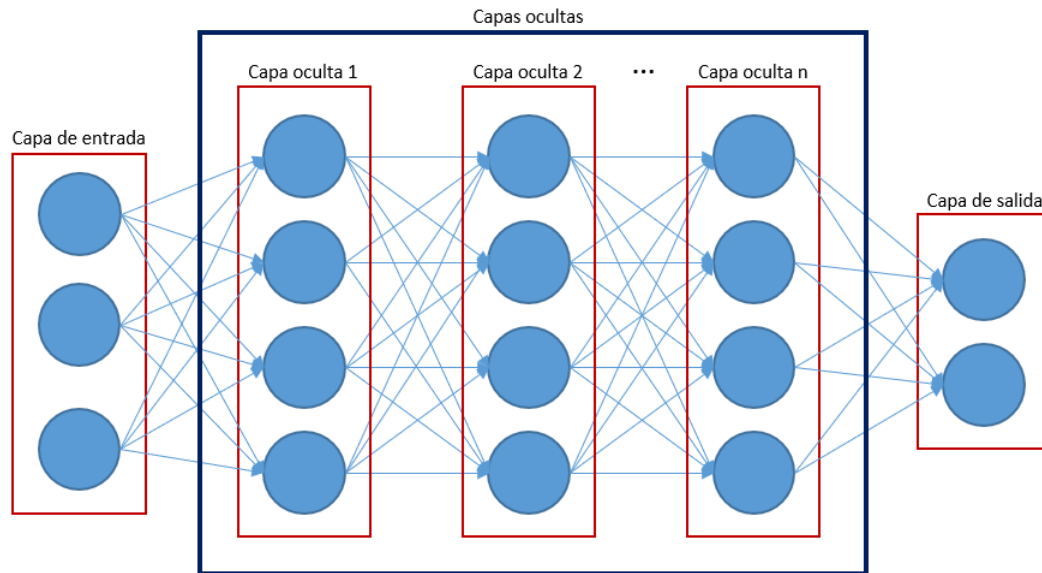


Figure 5.14: Red Neuronal

Para llevar a cabo el entrenamiento de la red neuronal, se optó por utilizar Python como lenguaje de programación.

Durante el proceso de entrenamiento, se hizo uso de varias bibliotecas de Python para realizar tareas específicas. Entre estas bibliotecas se incluyen Numpy, que son herramientas fundamentales para el manejo y procesamiento de datos. Además, se utilizó la biblioteca Tensorflow para implementar y entrenar la red neuronal en sí misma. Por último, la biblioteca Matplotlib se empleó para la visualización de resultados y análisis de datos, proporcionando herramientas gráficas para interpretar los resultados del entrenamiento. Juntas, estas bibliotecas permitieron llevar a cabo el proceso de entrenamiento de manera efectiva y obtener resultados significativos en el desarrollo de la red neuronal.

5.1. Código de la red neuronal

Se inició el proceso importando las librerías necesarias para el desarrollo del modelo en Python. Entre las librerías utilizadas se encuentran NumPy, TensorFlow y Matplotlib.

Posteriormente, se procedió a cargar el archivo de datos desde un archivo del programa Microsoft Excel. Utilizando la función *loadtxt* de NumPy, se leyeron los datos recopilados,

separándolos en dos listas correspondientes a las entradas y salidas respectivamente. Estas listas se convirtieron luego en arreglos NumPy para su procesamiento posterior.

A continuación, se creó la estructura de la red neuronal. La primera capa oculta se configuró con dos unidades neuronales y una función de activación ReLU. La capa de salida se estableció con una única unidad y una función de activación lineal. Todo esto se realizó mediante las funcionalidades proporcionadas por TensorFlow, utilizando su API de alto nivel Keras.

Una vez configurada la arquitectura de la red, se seleccionó el optimizador, la tasa de aprendizaje y la función de pérdida mediante el método *compile()* del modelo. Se optó por utilizar el optimizador Adaptive Moment Estimation o bien llamado *ADAM* con una tasa de aprendizaje de 0.1 y la función de pérdida que es una medida del error del modelo al hacer predicciones, calculada mediante una función que compara las predicciones con los valores reales. definió como el error cuadrático medio.

Finalmente, se dio inicio al entrenamiento de la red. Utilizando el método *'fit()'*, se proporcionaron los datos de entrada y salida junto con el número de épocas, que en este caso fueron 500. El entrenamiento se realizó en silencio, es decir, sin mostrar información detallada en cada época, y al finalizar se imprimió un mensaje indicando que el modelo había sido entrenado con éxito.

El código que ejecuta lo descrito anteriormente se encuentra en la Sección 11.5.1 de los Anexos.

6. Métodos de optimización

A continuación se presentarán los métodos de optimización que se consideraron para el trabajo de la memoria de titulación, estos consisten en el Método ADMM y el Método *Active-Set* de Quadprog los cuales serán explicados en los Subcapítulos siguientes.

6.1. Método de optimización ADMM

Al tratarse de un problema de optimización cuadrática (QP), se busca calcular las acciones de control más adecuadas de acuerdo al horizonte de predicción finito, para aquello se opta por utilizar el Método de los Multiplicadores Alternados de Lagrange (ADMM).

Para poder introducir el método ADMM y explicarlo, se debe conocer sobre los métodos que lo preceden con tal de respaldar y explicar de mejor manera, para aquello se refiere al documento [20]:

6.1.1. Método "Dual Ascent"

El método de "Dual Ascent" es un algoritmo de optimización utilizado para resolver ciertos tipos de problemas de optimización, especialmente problemas de programación lineal. También se conoce como el método de ascenso dual o el método de optimización de subgradientes. Este método es especialmente útil cuando se trata con problemas a gran escala o bien para problemas en los que el número de variables o restricciones es muy grande, lo que dificulta resolver directamente el problema primal por lo que se crea un problema dual, mediante la introducción de los Multiplicadores de Lagrange.

Al trabajar con el problema dual y utilizar subgradientes que proporcionan una generalización de las derivadas para las funciones que no sean suaves de manera iterativa. En cada iteración se verifica la factibilidad de las soluciones del problema primal y dual, donde se pueden ajustar los tamaños de los pasos para garantizar la convergencia, En resumen este método es capaz de manejar problemas a gran escala de manera más eficiente.

6.1.2. Método de Descomposición Dual

El método de descomposición dual es otra técnica utilizada para resolver problemas de optimización, especialmente aquellos que son grandes y se pueden descomponer en subproblemas más pequeños e independientes. Este enfoque busca abordar problemas de gran escala al dividirlos en partes más manejables y resolver cada parte de manera independiente. En base a las descomposiciones se formula un problema dual derivado de los subproblemas, estos se resuelven de manera independiente, y pueden resolverse empleando distintos métodos para cada uno de los subproblemas y se obtienen sus soluciones duales.

Luego se actualizan los multiplicadores o variables duales en función de las soluciones de los subproblemas, esto se realiza para mejorar la convergencia y garantizar contingencia, al igual que el método "Dual Ascent" se ejecuta de manera iterativa hasta lograr la convergencia o si es que se cumple un criterio de parada. Finalmente al lograrse la convergencia de los subproblemas se reconstruye la solución del problema original empleando las soluciones de los subproblemas trabajados.

En base a los dos puntos anteriores y del siguiente documento [5] se presentan resumidamente las ecuaciones que definen al ADMM, este se basa en [21]. Se puede explicar de mejor manera sobre que trata el tema, se define el siguiente problema de optimización para la minimización de una función $f(x)$ a restricciones:

$$\min_x f(x) \quad \text{sueto a: } Ax = b \quad (6.76)$$

donde $f(x)$ es estrictamente convexa y sus restricciones son lineales.

Para aquello se define el Lagrangiano de la función $f(x)$ como:

$$L(x, u) = f(x) + u^T(Ax - b) \quad (6.77)$$

El Lagrangiano de la función se puede separar en torno a x , de tal forma que el gradiente de la descomposición dual se repite de manera iterativa para $k = 1, 2, 3, \dots$ hasta N de la siguiente manera:

$$x^{(k)} = \arg \min_x L(x, u^{(k-1)}) \quad (6.78)$$

$$u^{(k)} = u^{(k-1)} + \tau^{(k)}(Ax^{(k)} - b) \quad (6.79)$$

6.1.3. Método de los Multiplicadores Aumentados de Lagrange

El método de los multiplicadores aumentados de Lagrange es una técnica que permite abordar problemas de optimización con restricciones. Cuando se enfrenta a una función objetivo que se desea optimizar, sujeta a restricciones, los multiplicadores aumentados de Lagrange permiten incorporar esas restricciones en el proceso de optimización.

En este método se introduce un nuevo conjunto de variables llamadas multiplicadores aumentados. Estos multiplicadores aumentados ayudan a manejar tanto las restricciones de igualdad como las de desigualdad en el problema de optimización.

La función Lagrangiana aumentada se construye sumando la función objetivo ponderada por los multiplicadores de Lagrange y multiplicada por las restricciones de igualdad, y luego los términos adicionales que incluyen las restricciones de desigualdad multiplicadas por los multiplicadores aumentados.

La optimización se lleva a cabo al encontrar los valores de las variables y los multiplicadores que hacen que todas las derivadas parciales de la Lagrangiana aumentada sean iguales a cero. Estos valores proporcionarán los puntos óptimos de la función objetivo bajo las restricciones dadas, incluyendo las restricciones de desigualdad.

A continuación se muestra el desarrollo al aumentar el Lagrangiano de 6.77 para un parámetro ρ el cual es $\rho > 0$:

$$\min_x \left(f(x) + \frac{\rho}{2} \|Ax - b\|_2^2 \right) \quad (6.80)$$

Al aplicar lo anterior, el Lagrangiano aumentado en función del parámetro ρ resulta en:

$$L_\rho(x, u) = f(x) + u^T(Ax - b) + \frac{\rho}{2} \|Ax - b\|_2^2 \quad (6.81)$$

Donde se puede repetir el proceso de manera iterativa para $k = 1, 2, 3, \dots$ hasta N :

$$x^{(k)} = \arg \min_x L_\rho(x, u^{(k-1)}) \quad (6.82)$$

$$u^{(k)} = u^{(k-1)} + \rho(Ax^{(k)} - b) \quad (6.83)$$

6.1.4. Método de los Multiplicadores de Lagrange con Direcciones Alternadas

El método de los multiplicadores alternados de Lagrange, considera lo más importante de todos los métodos explicados anteriormente y este consiste en que para un problema de optimización sujeto a restricciones de la forma:

$$\min_{x,z} f(x) + g(z) \quad \text{sujeto a: } Ax + Bz = c \quad (6.84)$$

En la siguiente ecuación se muestra la expresión para el Lagrangiano aumentado en función de un parámetro ρ donde se cumple que $\rho > 0$:

$$L_\rho(x, z, u) = f(x) + g(z) + u^T(Ax + Bz - c) + \frac{\rho}{2}\|Ax + Bz - c\|_2^2 \quad (6.85)$$

Al igual que para el método anterior se repite el proceso de forma iterativa ($k = 1, 2, 3, \dots, N$), por lo que el método ADMM resulta en:

$$x^{(k)} = \arg \min_x L_\rho(x, z^{(k-1)}, u^{(k-1)}) \quad (6.86)$$

$$z^{(k)} = \arg \min_z L_\rho(x^{(k-1)}, z, u^{(k-1)}) \quad (6.87)$$

$$u^{(k)} = u^{(k-1)} + \rho(Ax^{(k)} + Bz^{(k)} - c) \quad (6.88)$$

Para las funciones $f(x)$ y $g(x)$ de ADMM al cumplirse que $\rho > 0$ se aplican las siguientes propiedades de convergencia :

- **Convergencia Residual:** $r^{(k)} = Ax^{(k)} + Bz^{(k)} - c \rightarrow 0$ como $k \rightarrow \infty$.
- **Convergencia Objetivo:** $f(x^{(k)}) + g(z^{(k)}) \rightarrow f^* + g^*$, donde $f^* + g^*$ son las soluciones óptimas primales.
- **Convergencia Dual:** $u^{(k)} \rightarrow u^*$, donde u^* es una solución dual.

En base a lo anterior se puede apreciar que se depende directamente del valor del parámetro

ρ . Por aquello es fundamental ajustar el valor del parámetro ρ de manera adecuada, con tal de tener una optimización adecuada, sin embargo aquello puede ser complejo matemáticamente por lo que se puede obtener implementando un código en MATLAB.

6.2. Método de optimización Active-set de Quadprog

Existen varias estrategias para resolver problemas en MATLAB, y Quadprog es uno de los entornos más utilizados para resolver problemas cuadráticos. Ofrece varios solvers para la optimización cuadrática, entre los cuales destaca el método *Active-Set*.

El método *Active-Set* es una estrategia para resolver problemas cuadráticos que poseen un conjunto inicial de restricciones para encontrar un punto factible. Luego de ello, se itera en este conjunto para encontrar un punto factible que satisfaga las condiciones de optimalidad. Este se escoge por su compatibilidad con el software Simulink.

En base a la metodología descrita en las referencias [6] y [22], se explica detalladamente el procedimiento empleado por el método *Active-Set*.

6.2.1. Metodología del Algoritmo Active-Set

Para un problema de optimización cuadrático de la forma:

$$\min_x \frac{1}{2} x^T Q x + c^T x \quad (6.89)$$

sujeto a las restricciones

$$Ax = b \quad (6.90)$$

$$Dx \geq e \quad (6.91)$$

donde la matriz Q corresponde al Hessiano y es definida semipositiva, la función objetivo es cuadrática convexa y las restricciones describen ecuaciones e inecuaciones lineales.

El método *Active-Set* sigue los siguiente pasos:

- **Paso 1: Inicialización**

Se busca un punto factible de solución x_0 que cumpla todas las condiciones de igualdad y esté dentro de los límites de las condiciones de desigualdad. Además, se elige el conjunto A_0 como las posibles soluciones de desigualdad que se consideran **Activas**, es decir, donde la desigualdad se convierte en igualdad para el punto x_0 . Esto se expresa como:

$$A_0 = \{i \mid D_i x_0 = e_i, \text{ } i \text{ es un índice de todas las restricciones de desigualdad}\} \quad (6.92)$$

- **Paso 2: Resolución del subproblema cuadrático**

Se formula el subproblema de optimización cuadrático de la forma:

$$\min_d \frac{1}{2} d^T \cdot Q \cdot d + c^T \cdot d, \quad \text{sujeto a } A_{eq} d = 0 \quad (6.93)$$

donde la variable "d" corresponde a la dirección de búsqueda.

- **Paso 3: Determinación de dirección y tamaño del paso**

Para calcular la dirección d , se utiliza el método de los multiplicadores de Lagrange. El tamaño del paso se calcula como:

$$\alpha_k = \min \left(1, \min_{i \notin W_k, D_i d_k > 0} \frac{f_i - D_i x_k}{D_i d_k} \right) \quad (6.94)$$

Y se cumple que:

- α_k es el tamaño del paso en la iteración k .
- W_k es el conjunto activo en la iteración k y contiene los índices de las restricciones sospechosas de actividad.
- D_i es la fila i -ésima de la matriz D , que define las restricciones de igualdad.
- d_k corresponde a la dirección de búsqueda en la iteración k .
- x_k es el punto actual en la iteración k .
- f_i es el elemento i -ésimo del vector f que define las constantes de las restricciones de desigualdad.

- **Paso 4: Verificación de condiciones de optimalidad KKT**

Se verifican las condiciones de Karush-Kuhn-Tucker (KKT) que son necesarias y suficientes para demostrar la optimalidad:

- **Factibilidad primal:** Se verifica que la solución cumpla con todas las restricciones.
- **Estacionariedad:** El gradiente de la función objetivo en el punto óptimo es cero.
- **Factibilidad dual:** Los multiplicadores de Lagrange asociados con las restricciones de desigualdad son mayores o iguales a cero.
- **Holgura complementaria:** Se cumple la siguiente relación para cada restricción de desigualdad:

$$\lambda_i \cdot (D_i x - f_i) = 0 \quad \forall i \quad (6.95)$$

- **Paso 5: Actualización del conjunto activo e iteración**

Se ajusta el conjunto activo W_k añadiendo o eliminando restricciones según sea necesario.

- **Paso 6: Actualización del punto de solución e iteraciones**

Si se añade una restricción al conjunto activo, se recalcula la dirección de búsqueda y el tamaño del paso antes de actualizar x y repetir el proceso.

Esta metodología del método *Active-Set* garantiza encontrar un punto factible que también sea un punto estacionario de la función objetivo, cumpliendo con todas las restricciones y las condiciones de optimalidad.

7. Diseño de control

En el siguiente capítulo se analizará la solución para la inestabilidad de la planta en tiempo discreto. Para abordar este problema, se considerará la prealimentación de la planta. Luego, se buscará realizar un seguimiento a la referencia, incorporando observadores de estado y teniendo en cuenta las perturbaciones que pueden afectar al sistema.

7.1. Prealimentación para inestabilidad de la planta

Primero que todo, para este problema de control se debe considerar la inestabilidad determinada en el Capítulo 2. Para abordar la inestabilidad, se plantea utilizar un lazo de prealimentación hacia la actuación que proviene del *MPC* antes de ingresar a la *Planta RFL*, realizando una multiplicación con una ganancia L que se calcula mediante una ley de control, como se realizó en un trabajo anterior [23]. Esta planta posee un polo en cero en tiempo continuo que es marginalmente estable. El diseño del lazo de prealimentación se aprecia en la siguiente figura.

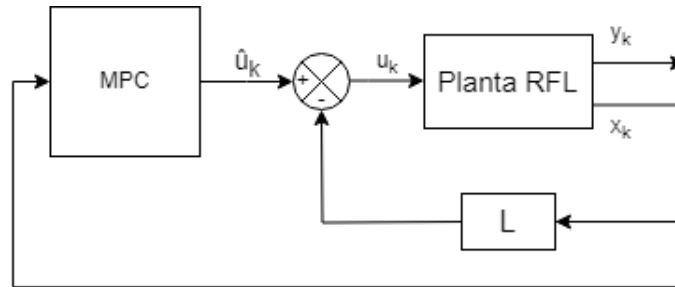


Figure 7.15: Lazo de control con prealimentación

Al discretizar el modelo de la planta formulado en el Subcapítulo 4.3, se reconoce que la magnitud de los polos de la matriz A_d es mayor a 1, es decir, sobrepasan el círculo unitario [15]. Para lograr aquello se utiliza la siguiente ley de control:

$$u_k = -Lx_k + \hat{u}_k \quad (7.96)$$

Donde al reemplazar en (4.42), se tiene que:

$$x_{k+1} = (A_d - B_d L)x_k + B_d \hat{u}_k \quad (7.97)$$

Donde se define que $A_x = A_d - B_d L$ y los polos de la matriz A_x se encuentran ubicados en $\lambda[A_x < 1]$, lográndose estabilizar el sistema. Por lo que se puede reescribir el sistema:

$$x_{k+1} = A_x x_k + B_d \hat{u}_k \quad (7.98)$$

Considerando las restricciones que resultan en:

$$u^{min} \leq \hat{u}_k - Lx_k \leq u^{max} \quad (7.99)$$

Finalmente se puede reescribir el problema de optimización QP como:

$$u_t^{opt} = \min_{\hat{u}} J_N(x_k, \hat{u}) \quad (7.100)$$

Sujeto a las siguientes restricciones:

$$\begin{aligned} u^{min} &\leq \hat{u}_k - Lx_k \leq u^{max} \\ x_0 &= x \\ x_{k+1} &= A_x x_k + B_d \hat{u}_k \\ x^{min} &\leq x_k \leq x^{max} \\ x_N^{min} &\leq x_N \leq x_N^{max} \end{aligned} \quad (7.101)$$

7.2. Seguimiento a referencia

Para que la salida sea capaz de seguir una referencia deseada, se requiere que el valor de ella en estado estacionario denotado como y_∞ debe tender a la referencia con tal que $y_\infty \rightarrow r$.

Para ello se reescribe el sistema en variables de estado que considera el lazo de prealimentación

para tratar la estabilidad en estado estacionario:

$$\begin{aligned}x_{\infty} &= A_x x_{\infty} + B_d \hat{u}_{\infty} \\y_{\infty} &= C_d x_{\infty}\end{aligned}\tag{7.102}$$

De lo anterior se despeja x_{∞} y u_{∞} para expresar el sistema en base a la referencia:

$$\begin{aligned}x_{\infty} &= (I - A_x)^{-1} B_d \hat{u}_{\infty} \\y_{\infty} &= C_d (I - A_x)^{-1} B_d \hat{u}_{\infty}\end{aligned}\tag{7.103}$$

$\hat{u}_{\infty}$ resulta en:

$$\hat{u}_{\infty} = [C_d (I - A_x)^{-1} B_d]^{-1} r\tag{7.104}$$

Se desplaza el punto de operación restando los vectores de entrada y salida estacionarios:

$$\begin{aligned}\bar{x} &= x - x_{\infty} \\\bar{u} &= \hat{u} - \hat{u}_{\infty}\end{aligned}\tag{7.105}$$

Tras aplicar el cambio de variable, resta hacerlo también en las restricciones, por lo que se deben realizar arreglos en ellas y resultan en:

$$\underbrace{\begin{bmatrix} \mathcal{O}_N \\ -\mathcal{O}_N \\ \mathcal{O} \\ -\mathcal{O} \\ I \\ -I \end{bmatrix}}_G \vec{u} \leq \underbrace{\begin{bmatrix} x_N^{max} - x_{\infty} - \mathcal{A}_N x_0 \\ -x_N^{min} + x_{\infty} - \mathcal{A}_N x_0 \\ \bar{x}^{max} - x_{\infty} - \mathcal{A} x_0 \\ -\bar{x}^{min} + x_{\infty} + \mathcal{A} x_0 \\ u^{max} - \hat{u}_{\infty} \\ -u^{min} + \hat{u}_{\infty} \end{bmatrix}}_g\tag{7.106}$$

Y las dimensiones son las siguientes tanto $G \in \mathbb{R}^{(2 \cdot n + 2 \cdot n \cdot N + 2 \cdot m \cdot N) \times (m \cdot N)}$ como para $g \in \mathbb{R}^{(2 \cdot n + 2 \cdot n \cdot N + 2 \cdot m \cdot N) \times 1}$.

Finalmente debido al cambio de variable realizado al desviar el punto de operación, se debe sumar nuevamente antes de la salida para tener la actuación óptima como entrada para la

planta.

$$\hat{u}^{opt} = \bar{u}_0^{opt} + \hat{u}_\infty \quad (7.107)$$

Finalmente el diagrama que representa el sistema se observa en la Figura

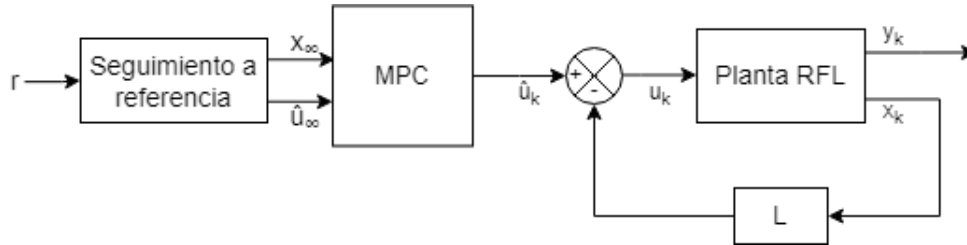


Figure 7.16: Lazo de control con prealimentación

7.3. Observador de estados

Un observador de estados, también conocido como estimador de estados, desempeña un papel fundamental en los sistemas de control al estimar el estado interno de un sistema dinámico. Este proceso se basa en medidas disponibles y en el modelo matemático del sistema [24]. Su función primordial es proporcionar una estimación precisa del estado del sistema, incluso cuando no todas las variables de estado son directamente observables o medibles.

A continuación se presenta un diagrama de bloques que ilustra la función básica del observador.

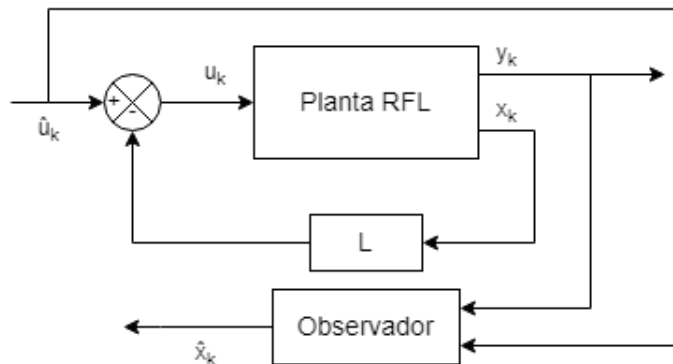


Figure 7.17: Diagrama del observador básico

En la Figura 7.17, se muestra el diagrama de un observador básico que utiliza la entrada $\hat{u}_k$ y

la salida del sistema y_k para estimar los estados $\tilde{x}$. En este esquema, K representa la matriz de ganancia de realimentación de estado.

Las ecuaciones que describen la dinámica y el comportamiento del observador son las siguientes:

$$\begin{aligned}\tilde{x}_{k+1} &= A_x \tilde{x}_k + B_d \hat{u}_k \\ y_k &= C_d x_k\end{aligned}\tag{7.108}$$

El observador se aplica al modelo del sistema cuyas matrices son A_x , B_d , C_d y $D_d = 0$. Con base en la ecuación anterior, es posible reconstruir el sistema como sigue:

$$\tilde{x}_{k+1} = A_d \tilde{x}_k + B_d \hat{u}_k\tag{7.109}$$

Utilizando la matriz de realimentación de estado mostrada en la Figura 7.17, se obtiene:

$$\tilde{x}_{k+1} = A_d \tilde{x}_k + B_d \hat{u}_k + L_z(y_k - C_d \tilde{x}_k)\tag{7.110}$$

Donde L_z es equivalente a la matriz K de ganancia de retroalimentación de estados mencionada anteriormente.

Para la estimación de estado, se busca que el término $L_z(y_k - C_d \tilde{x}_k)$ no tenga efecto, es decir, forzar que la salida y_k sea igual a la estimación. Esto se logra expresando el vector de error como:

$$\bar{x}_k = x_k - \tilde{x}_k\tag{7.111}$$

Reescribiendo la ecuación (7.111) en la ecuación (7.110), se obtiene:

$$\bar{x}_{k+1} = A_d \bar{x}_k - L_z(y_k - C_d \tilde{x}_k)\tag{7.112}$$

De lo anterior, se deduce que:

$$A_{obs} = (A_x - L_z C_d)\tag{7.113}$$

Para finalizar se debe escoger los polos de L_z de manera adecuada con el objetivo de que la matriz A_{obs} sea estable y, por lo tanto, se puedan reconstruir los estados con el propósito de

disminuir el error de la la señal lo mas cercano a cero.

7.4. Observador de estados Extendido con perturbación modelada

Se busca incorporar al modelo del sistema perturbaciones que puedan afectar a este, planteando el siguiente modelo:

$$\begin{aligned}x_{k+1} &= A_x x_k + B_d \hat{u}_k + B_d d_k \\y_k &= C_d x_k + e_k\end{aligned}\tag{7.114}$$

Donde d_k corresponde a la perturbación a estimar y e_k al error de medición.

Para realizar la estimación, se procede a aumentar el grado del sistema y se define un nuevo vector de estado z_k de la forma:

$$z_k = \begin{bmatrix} x_k \\ d_k \end{bmatrix}\tag{7.115}$$

Se reescribe el sistema extendido de forma matricial:

$$\begin{aligned}\underbrace{\begin{bmatrix} x_{k+1} \\ d_{k+1} \end{bmatrix}}_{z_{k+1}} &= \underbrace{\begin{bmatrix} A_x & B_d \\ 0 & I \end{bmatrix}}_{A_z} \underbrace{\begin{bmatrix} x_k \\ d_k \end{bmatrix}}_{z_k} + \underbrace{\begin{bmatrix} B_d \\ 0 \end{bmatrix}}_{B_z} u_k \\y_k &= \underbrace{\begin{bmatrix} C_d & 0 \end{bmatrix}}_{C_z} \begin{bmatrix} x_k \\ d_k \end{bmatrix}\end{aligned}\tag{7.116}$$

Se diseña la ley de realimentación de estado considerando una ganancia L_z :

$$\hat{z}_{k+1} = A_z \hat{z}_k + B_z \hat{u}_k + L_z (y_k - C_z \hat{z}_k)\tag{7.117}$$

En base a lo anterior el sistema estimado observado resulta en:

$$\begin{aligned}\bar{x}_{k+1} &= A_{obs} \bar{x}_k + B_{obs} \bar{\hat{u}}_k \\ \bar{y}_k &= C_{obs} \bar{x}_k\end{aligned}\tag{7.118}$$

$$A_{obs} = A_z - L_z C_z, \quad B_{obs} = \begin{bmatrix} B_z & L_z \end{bmatrix}, \quad C_{obs} = I \quad (7.119)$$

Y los vectores de entrada y salida resultan en:

$$\text{entrada : } \bar{u}_k = \begin{bmatrix} \hat{u}_k & y_k \end{bmatrix}^T, \quad \text{salida : } \bar{y}_k = \begin{bmatrix} x_k & d_k \end{bmatrix}^T \quad (7.120)$$

7.5. Seguimiento a referencia con perturbación

Como se vio anteriormente, se busca implementar seguimiento a referencia. Para que la salida en estado estacionario logre llegar al valor correspondiente $y_\infty \rightarrow r$, se planea considerar los efectos de la perturbación en el sistema.

Se debe efectuar el mismo cambio de variable realizado en la ecuación (7.105)

Para aquello se estudiarán las señales del sistema en estado estacionario considerando la perturbación.

$$\begin{aligned} x_\infty &= A_x x_\infty + B_d \hat{u}_\infty + B_d \hat{d} \\ y_\infty &= C_d x_\infty \end{aligned} \quad (7.121)$$

Lo primero que se debe realizar es despejar x_∞ en estado estacionario y resulta en:

$$\begin{aligned} x_\infty - A_x x_\infty &= B_d(\hat{u}_\infty + \hat{d}) \\ (I - A_x)x_\infty &= B_d(\hat{u}_\infty + \hat{d}) \\ x_\infty &= (I - A_x)^{-1} B_d(\hat{u}_\infty + \hat{d}) \end{aligned} \quad (7.122)$$

luego la salida y_∞ resulta en:

$$y_\infty = C_d(I - A_x)^{-1} B_d(\hat{u}_\infty + \hat{d}) \quad (7.123)$$

Considerando lo anterior para el estado y la salida en estado estacionario se puede despejar

la entrada u_∞ siguiendo los pasos:

$$\begin{aligned}
 \hat{u}_\infty + \hat{d} &= [C_d(I - A_d)^{-1}B_d]^{-1}y_\infty \\
 \hat{u}_\infty &= [C_d(I - A_d)^{-1}B_d]^{-1}y_\infty - \hat{d} \\
 \hat{u}_\infty &= [C_d(I - A_d)^{-1}B_d]^{-1}r - \hat{d}
 \end{aligned} \tag{7.124}$$

En base a lo anterior se puede representar el diagrama del sistema que contiene la prealimentación para la estabilización de la planta, el seguimiento a referencia considerando una perturbación y el observador de estados como el de la Figura 7.18

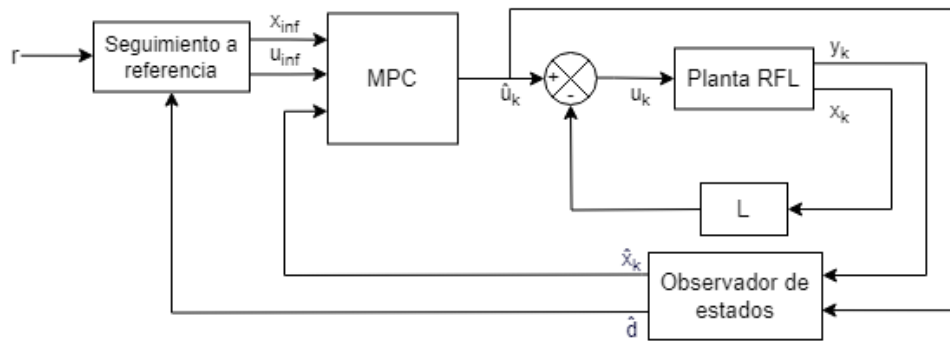


Figure 7.18: Seguimiento a referencia con observador de estados

8. Modelo en entorno de simulación

Como se mencionó en los objetivos de esta memoria, se busca implementar acciones de control utilizando MPC sobre la planta RFL. El problema de optimización QP planteado en el Capítulo 4 se puede resolver de dos formas distintas empleando la Formulación Sparse. Se busca utilizar dos métodos de optimización distintos.

Primero, se desarrollará un script base en MATLAB para analizar el caso del estado cero, es decir, sin excitación presente, respondiendo a la pregunta: *¿Es capaz la planta de lograr seguimiento a referencia?*. Para ambos casos, se busca medir y analizar las señales antes y después de añadir observadores al sistema. Junto al script, se desarrollarán esquemas de simulación en Simulink, empleando los bloques necesarios para realizar las simulaciones de manera adecuada.

Además, se considerará el uso de dos métodos de optimización. El primero es el método *Active-Set* de la herramienta Quadprog, que requiere pocas líneas de código y menos parámetros a ajustar para controlar el sistema.

El segundo método a utilizar es el solver ADMM, cuyo funcionamiento se explicó en el Capítulo 6. Este método es importante para la implementación física en la planta RFL, y es crucial ajustar cuidadosamente los parámetros asociados, como la cantidad de iteraciones y el coeficiente ρ (*rho*).

Para la correcta implementación del sistema, se emplean los parámetros de la planta utilizados para el cálculo de matrices y parámetros del sistema, como se muestra en la Tabla 8.2. También se consideran las frecuencias de corte utilizadas para desarrollar los filtros que permiten calcular las velocidades necesarias para los estados x_3 y x_4 .

Table 8.2: Parámetros de la RFL

Parámetro	Valor	[Unidad]
B_{eq}	0.004	$N \cdot m$
J_{eq}	2.07e-3	$kg \cdot m^2$
m_l	0.065	Kg
L_l	0.419	m
ω_n	19.4495	rad/s
J_l	0.0038	$kg \cdot m^2$
K_s	1.4389	$N \cdot m/rad$
ω_{cf1}	314.1592	rad/s
ω_{cf2}	62.8318	rad/s

Se muestra en la Figura 8.19 el bloque *subsystem* utilizado para calcular los estados x_3 e x_4 a través de las velocidades mediante filtros en tiempo discreto, como se ilustra en la Figura 8.20. En este bloque, las salidas y_1 y y_2 se ingresan como entradas y se obtienen los estados x_3 y x_4 en la salida.

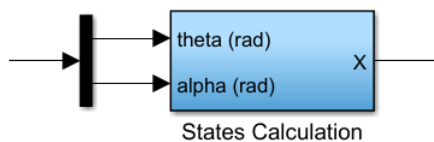


Figure 8.19: Bloque States Calculation

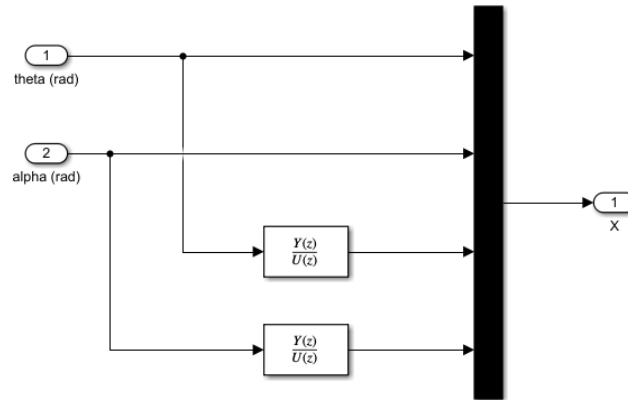


Figure 8.20: Interior bloque States Calculation

8.1. Bloques utilizados

A continuación se presentan los bloques de Simulink empleados para la construcción de los diagramas de simulación:

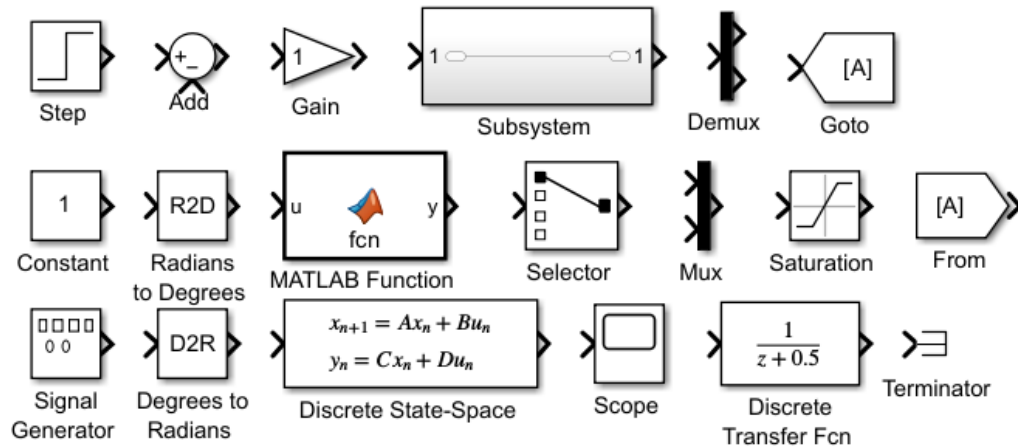


Figure 8.21: Bloques de Simulink

- **Step:** Este bloque genera una señal escalón.
- **Constant:** Este bloque añade señales o parámetros al esquema Simulink provenientes del Workspace de MATLAB.
- **Signal Generator:** Este bloque genera señales periódicas, como cuadrada, diente de sierra y sinusoidal.

- **Add:** Este bloque realiza la suma o resta de señales.
- **Product:** Este bloque realiza la multiplicación de señales.
- **Radians to Degrees:** Este bloque convierte valores de radianes a grados.
- **Degrees to Radians:** Este bloque convierte valores de grados a radianes.
- **Gain:** Este bloque amplifica o reduce una señal, conocido como ganancia.
- **Matlab Function:** Este bloque permite diseñar funciones necesarias para el cálculo de matrices o variables. Posee entradas (u) y salidas (y), y se escribe mediante código de MATLAB.
- **Discrete State Space:** Este bloque implementa la representación en variables de estado de tiempo discreto, ingresando las matrices y el tiempo de muestreo.
- **Subsystem:** Este bloque agrupa una sección de diagramas en un conjunto o sub-sistema.
- **Selector:** Este bloque selecciona una señal específica en un arreglo de señales.
- **Demux:** Este bloque desagrega señales de un conjunto.
- **Mux:** Este bloque agrupa señales formando un conjunto.
- **Scope:** Este bloque permite medir y observar señales.
- **Discrete Transfer Function:** Este bloque crea una función de transferencia discreta, utilizada como filtro.
- **Saturation:** Este bloque limita las señales entre valores mínimos y máximos.
- **To Workspace:** Este bloque exporta valores o señales al Workspace de MATLAB.
- **Goto:** Este bloque recibe una señal de origen del bloque *From*.
- **From:** Este bloque indica el origen de una señal y la conecta con el bloque *Goto*.
- **Terminator:** Este bloque descarta señales.

8.2. Entorno MATLAB/Simulink

El presente problema de optimización se busca resolver utilizando dos métodos de optimización distintos ADMM y *Active-Set* de Quadprog empleando la Formulación Sparse.

Se decide estudiar el problema de optimización por método de optimización, el primer método utilizado será *Active-Set* de Quadprog con Formulación Sparse. El segundo método será el de ADMM con Formulación Sparse.

8.3. Aspectos Generales para simulación

8.3.1. Modelo de la RFL a simular

Para el desarrollo de las simulaciones, se utilizaron los parámetros de la planta especificados en la Tabla 8.2, los cuales se sustituyeron en las matrices del sistema en variables de estado de (2.20), resultando en las matrices del sistema simulado dadas por (8.125). Se incluyeron también las dinámicas del motor DC, de manera similar a lo utilizado por Quanser en las simulaciones de control LQR, como se detalla en el Capítulo 3.

$$\begin{aligned} A_c &= \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 623.7468 & -7.1971 & 0 \\ 0 & -965.3452 & 7.1971 & 0 \end{bmatrix}, \quad B_c = \begin{bmatrix} 0 \\ 0 \\ 480.7692 \\ -480.7692 \end{bmatrix}, \\ C_c &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \quad D = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \end{aligned} \quad (8.125)$$

Como se analizó en el Capítulo 2, la controlabilidad, observabilidad y estabilidad de la planta justifican la aplicación de prealimentación para estabilizar el sistema.

La discretización del sistema se realiza mediante el comando *c2d* de MATLAB, considerando un tiempo de muestreo $T_s = 0.002$ s. Esto permite generar las matrices discretas necesarias para desarrollar la prealimentación y definir un horizonte de predicción $N = 5$.

El bloque MPC calcula la actuación óptima, a la cual se le debe restar la ganancia de prealimentación. Es esencial verificar que el solver utilizado en el sistema opere en tiempo discreto.

Para las simulaciones, se define el estado inicial x_0 , considerando las condiciones iniciales para los cuatro estados del sistema. El vector se convierte de grados a radianes usando la función `deg2rad` debido a que la planta RFL opera en radianes:

$$x_0 = \begin{bmatrix} 60 \\ 5 \\ 60 \\ -3 \end{bmatrix} \quad (8.126)$$

Las restricciones de los estados y de la actuación también se ajustan considerando la conversión de grados a radianes. Las restricciones de los estados y de la actuación se establecen como se detalla en(8.127):

$$x_{\min} = \begin{bmatrix} -360 \\ -10 \\ -1000 \\ -360 \end{bmatrix}, \quad x_{\max} = \begin{bmatrix} 360 \\ 10 \\ 1000 \\ 360 \end{bmatrix}, \quad (8.127)$$

$$u_{\min} = -10, \quad u_{\max} = 10$$

Las simulaciones se dividen en dos partes principales:

- **Estado Cero:** El diagrama de simulación mostrado en la Figura 8.22 representa el comportamiento de la planta RFL en ausencia de excitación. El objetivo es observar la respuesta del sistema en estado estacionario y verificar si los estados convergen a cero. Se examinarán la actuación saliente del MPC, la actuación saturada y las salidas y_1 y y_2 .

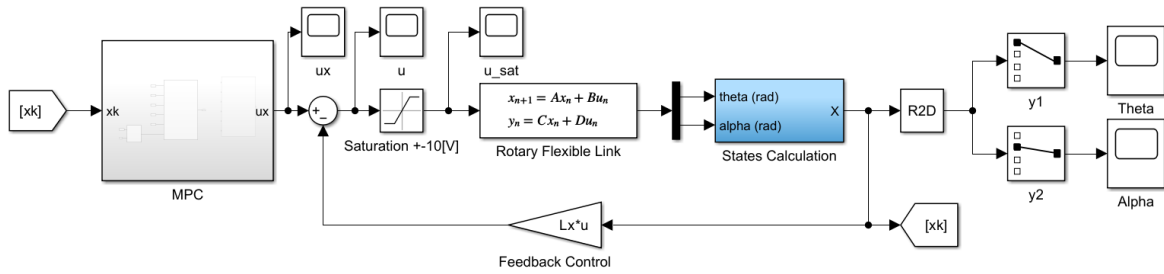


Figure 8.22: Diagrama RFL Estado Cero

- **Seguimiento a Referencia:** El diagrama de simulación en la Figura 8.23 se utiliza para observar y comparar la respuesta del sistema frente a una referencia y una perturbación. Se analiza cómo el sistema sigue la señal de entrada y se observan la actuación saliente del MPC, la actuación saturada y las salidas y_1 y y_2 .

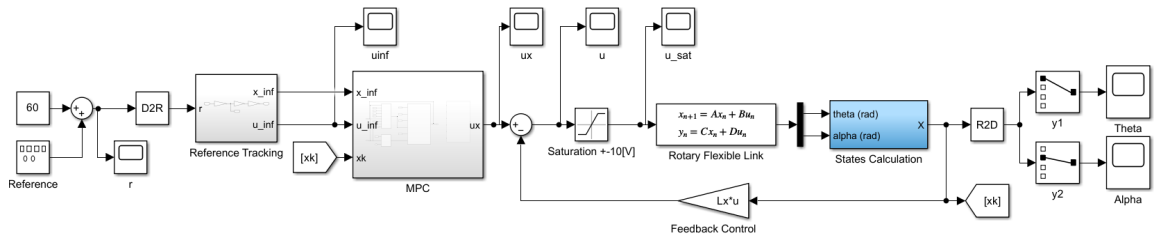


Figure 8.23: Diagrama RFL Seguimiento a Referencia

Las variaciones en los diagramas se presentan dentro del bloque MPC, el cual es un subsistema que incluye las matrices de predicción. Estas matrices varían dependiendo del tipo de formulación utilizada para resolver el problema de optimización y del método de optimización aplicado, ya sea *Active-Set* de Quadprog o ADMM.

8.4. Active-Set de Quadprog para la RFL

Para este caso se decide emplear el método de optimización *Active-Set* de Quadprog para controlar la RFL mediante un control MPC, donde se verá el comportamiento para el caso del estado cero y luego para el caso del seguimiento a referencia.

8.4.1. Active-Set de Quadprog para estado cero y seguimiento a referencia

Se decide desarrollar el caso del estado cero en primera instancia, donde se consideran las condiciones iniciales (8.126) y las restricciones (8.127), implementando el diagrama visto en la Figura (8.22), donde se aprecia la implementación del bloque *MPC* que recibe como entrada el estado del sistema y entrega la actuación óptima que actúa sobre la planta la entrada de la planta se presente un bloque de saturación preventivo para la actuación de $\pm 10[V]$.

Al interior del bloque *MPC*, se considera la aplicación del método *Active-Set* de Quadprog en su respectivo bloque cuyo código se encuentra en la Sección 11.4.6 de los Anexos. Además se observa la función *mat_f* que se encarga de construir la matriz *f* cuyo código se encuentra en la Sección 11.4.4 de los Anexos, junto a la función *quadprog* que se encarga de calcular la actuación óptima utilizando las matrices calculadas, como se aprecia en la Figura 8.24 en el bloque *MPC*.

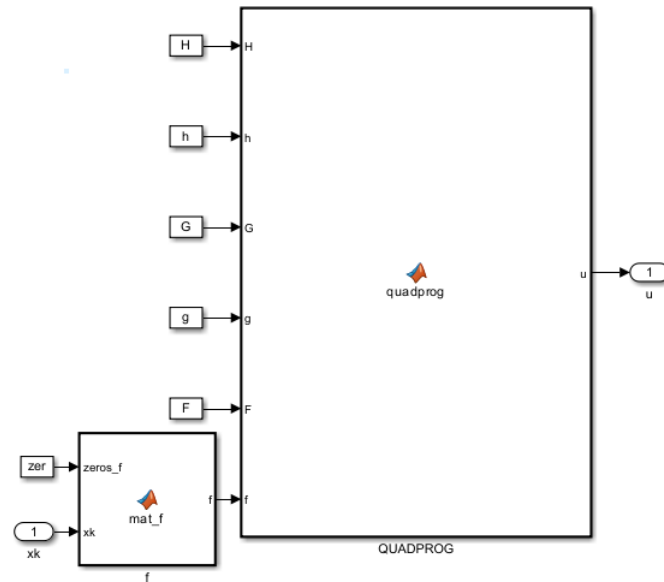


Figure 8.24: Bloque MPC para Quadprog en estado cero

Se considera también la implementación de la prealimentación para poder estabilizar la planta, considerando la ganancia L_{pre} calculada mediante la función *dlqr*, reubicando los polos del sistema en un entorno estable para que la magnitud de cada uno de ellos sea inferior a uno.

Con un tiempo de simulación de $30[s]$, un tiempo de muestreo $T_S = 0.002[s]$ y un horizonte

de predicción de $N = 5$, se obtienen los siguientes resultados para la actuación saliente del *MPC*, la actuación saturada que ingresa a la planta, las salidas y_1 e y_2 .

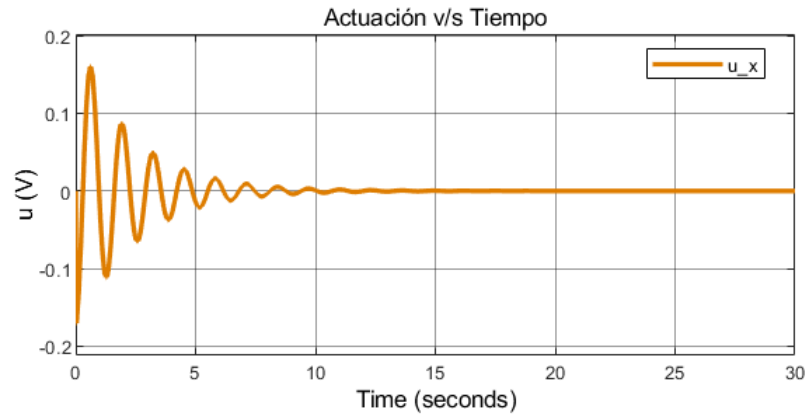


Figure 8.25: Quadprog Estado Cero: Actuación u_x en el Tiempo

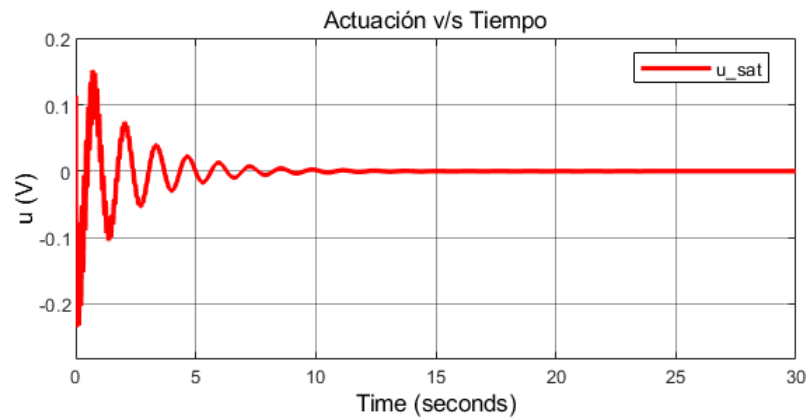


Figure 8.26: Quadprog Estado Cero: Actuación u_{sat} en el Tiempo

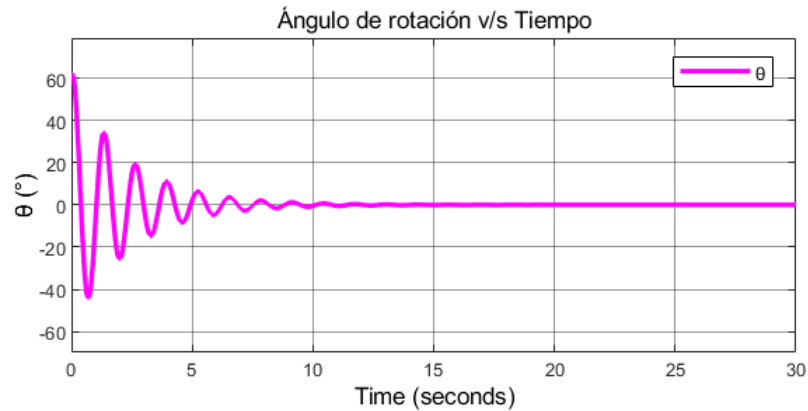


Figure 8.27: Quadprog Estado Cero: Ángulo de rotación θ en el Tiempo

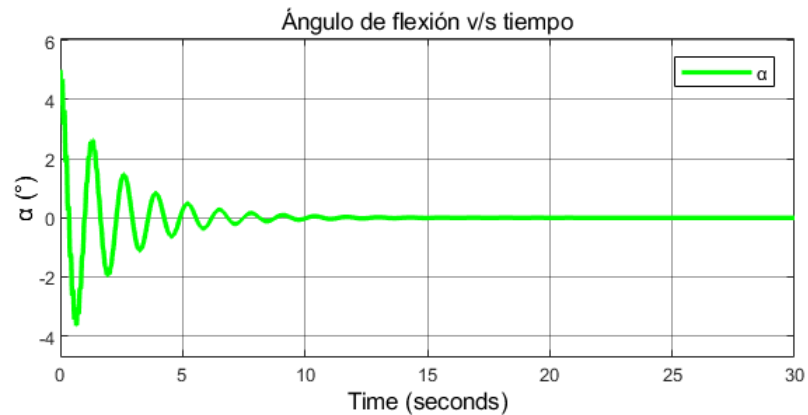


Figure 8.28: Quadprog Estado Cero: Ángulo de flexión α en el Tiempo

En ellos se observa que las señales comienzan con los valores indicados en las condiciones iniciales propuestas y convergen su valor hacia cero siguiendo la forma de una señal del tipo exponencial amortiguada, llegando a estabilizar el sistema oscilando en torno al origen de este, por este comportamiento medido se busca implementar observadores de estado para mejorar el rendimiento del sistema en estado estacionario.

Para el caso del seguimiento a referencia se consideran las mismas condiciones iniciales, pero se tuvieron que ajustar los valores de la ganancia de la prealimentación, esto se realiza cambiando los valores de la matriz Q de la función `dlqr` que se implementa, los cambios se pueden ver en el código *MPC Seguimiento a Referencia* ubicado en la Sección 11.4.2 de los Anexos.

En la Figura 8.23, se observa la incorporación del bloque *Reference Tracking*, que describe

el comportamiento de la señal para seguir una referencia no nula junto a una perturbación, este bloque que se aprecia en la Figura 8.30 recibe la referencia y la perturbación, con ellas se realizan las operaciones matriciales y entrega en su salida el estado y la actuación a seguir.

```

%% Seguimiento a referencia
i1 = eye(4)-Ax;
72 inv1 = inv(i1);
i2 = C(1,:)*inv1*B;
inv2 = inv(i2);
  
```

Figure 8.29: Código Seguimiento a Referencia: Líneas 70-74

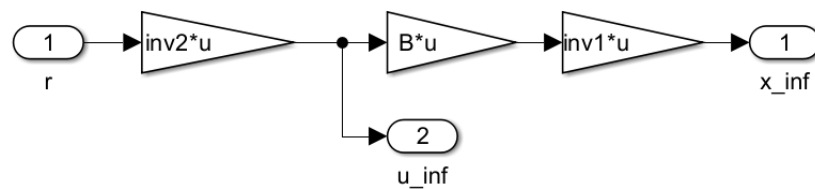


Figure 8.30: Interior bloque Reference Tracking

Para este caso el bloque *MPC* recibe las referencia proveniente del bloque *Reference Tracking* y los estados provenientes de la realimentación, en su interior se calculan las matrices mediante las funciones *mat_g*, *mat_f* y *quadprog* cuyos códigos se encuentran en las Secciones 11.4.5 y 11.4.6 de los Anexos, como se puede ver en la Figura 8.31.

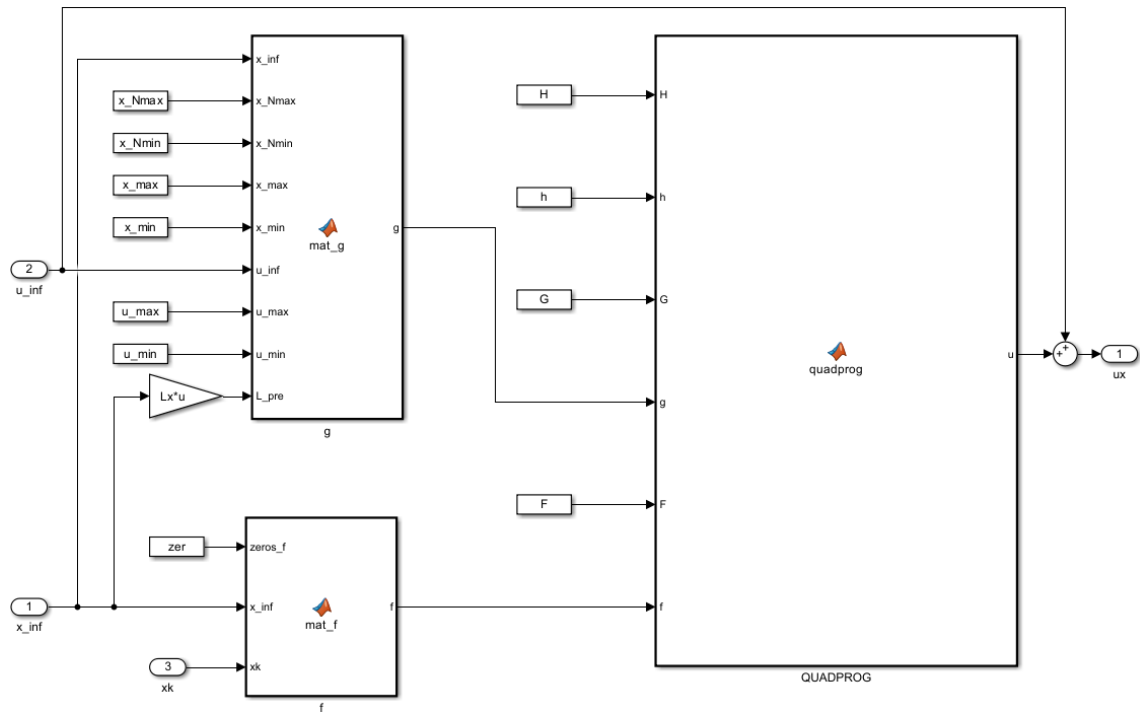


Figure 8.31: Interior bloque MPC

Se considera aumentar el tiempo de simulación a 60[s], y se tiene como referencia la señal cuadrada de la Figura 8.32, de amplitud 50[°] con un desfase de 60[°] y frecuencia $\frac{1}{20}[Hz]$ de la cual se pueden observar 3 ciclos.

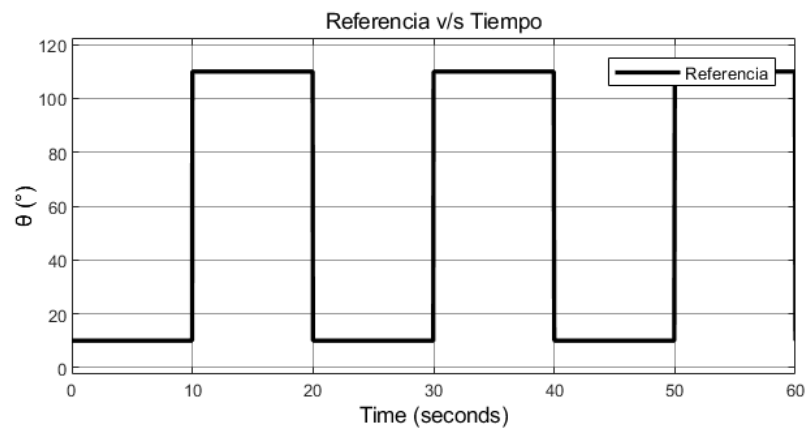


Figure 8.32: Quadprog Seguimiento a Referencia: Referencia en el Tiempo

Se ajustan los parámetros del controlador junto a la prealimentación vistos en el código *MPC Seguimiento a Referencia* de la Sección 11.4.2 de los Anexos para obtener señales de salida

apropiadas y estas resultan en las siguientes Figuras, para la actuación saliente del *MPC*, la actuación entrante a la planta RFL y las salidas y_1 e y_2 .

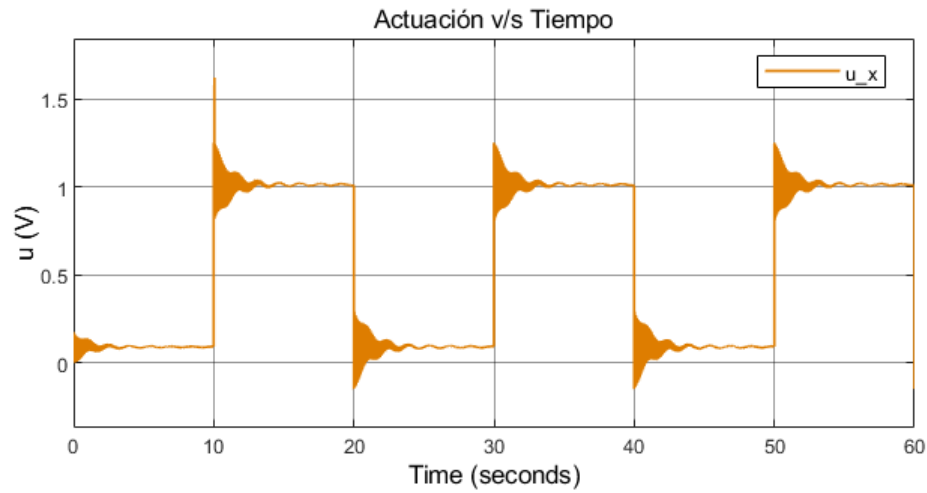


Figure 8.33: Quadprog Seguimiento a Referencia: Actuación u_x en el Tiempo

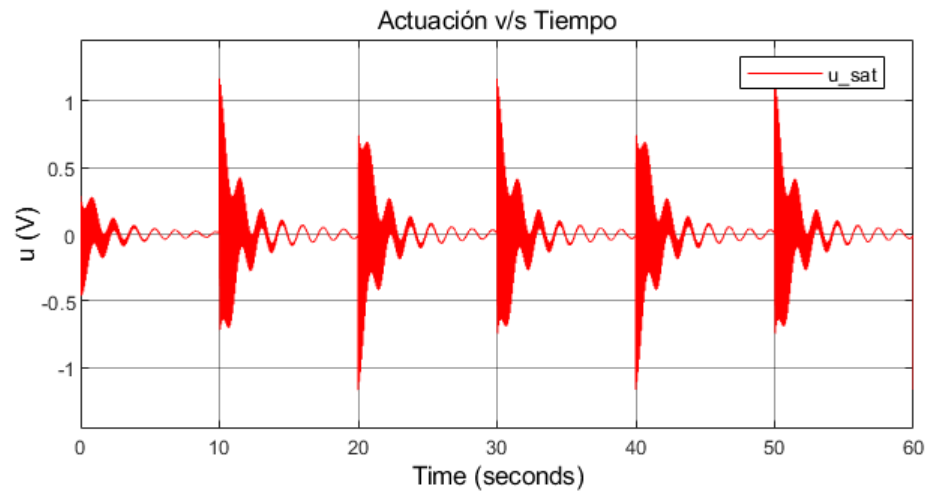


Figure 8.34: Quadprog Seguimiento a Referencia: Actuación u_{sat} en el Tiempo

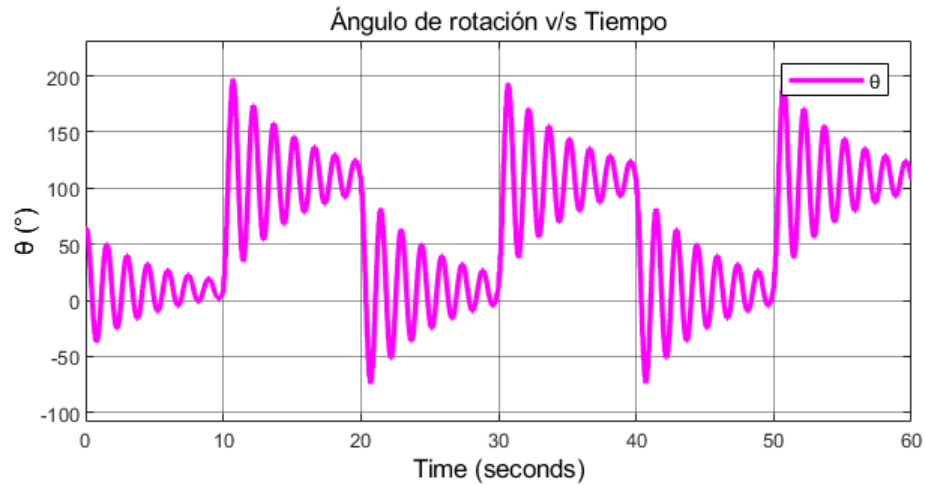


Figure 8.35: Quadprog Seguimiento a Referencia: Ángulo de rotación θ en el Tiempo

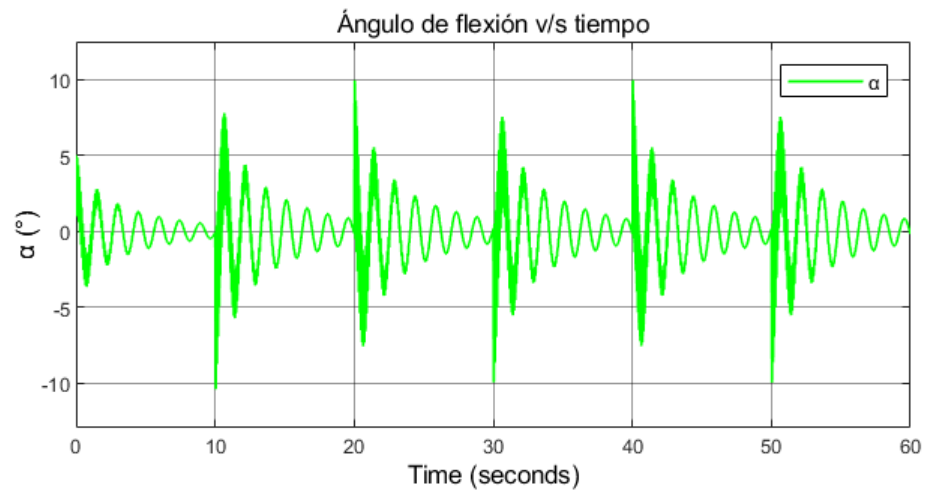


Figure 8.36: Quadprog Seguimiento a Referencia: Ángulo de flexión α en el Tiempo

De la Figura 8.34 se observa que la actuación se encuentra en valores apropiados para las restricciones establecidas.

De la Figura 8.35 se observa un sobre impulso con respecto al estado estacionario al seguir la referencia, y alcanza un valor muy alto. Esto no se encuentra apropiado sin la incorporación de observadores de estado.

De la Figura 8.36 se observa que los puntos de valor mas altos de valores para el ángulo de flexión, que de acuerdo a los cambios en la señal de referencia estos valores no alcanzan a

superar los $\pm 10[^\circ]$, lo cual es apropiado de acuerdo a las restricciones.

En base a lo anterior se considera incorporar observadores de estados al sistema para analizar su funcionamiento y obtener mejoras en las señales a medir.

8.4.2. Incorporación de observadores de estado para Quadprog

Para lograr mejoras en las mediciones de la señal de salida se decide incorporar un observador de estados, el cual fue descrito en el Capítulo 7, para este modelo se consideran las perturbaciones y se logra diseñar el diagrama de la Figura 8.37 junto al código *Observador de Estados* de la Sección 11.4.3, donde se consideran las matrices ampliadas y se calculan las matrices que modelan el observador de estados.

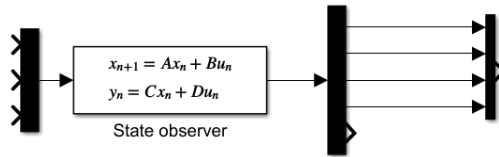


Figure 8.37: Observador de estados para RFL

Ingresa al observador de estados la actuación u_x proveniente de la salida del *MPC*, junto a las salidas medidas y_1 e y_2 . Y en sus salidas se obtienen los estados estimados junto a la estimación de perturbación, señales que hacen ingreso al *MPC* como se puede ver en la Figura 8.38

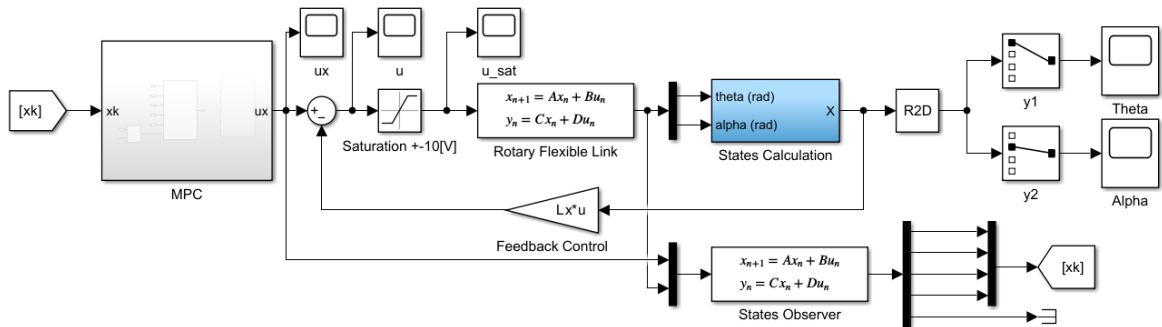


Figure 8.38: Diagrama de estado cero con observador de estados

Se consideran las mismas condiciones de simulación que para el caso del estado cero sin observador de estados y se obtienen los siguientes resultados para las señales de interés.

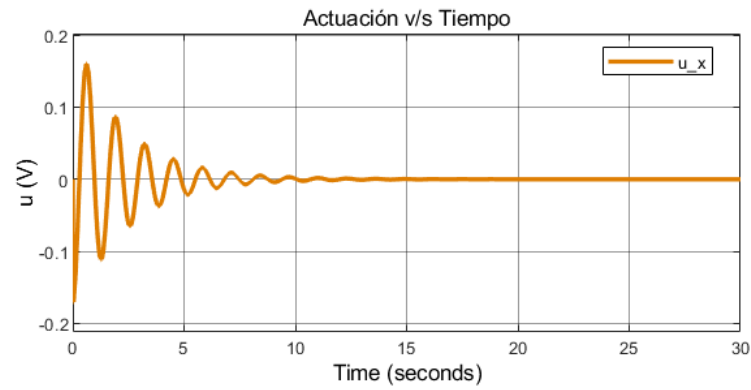


Figure 8.39: Quadprog Estado Cero Observador: Actuación u_x en el Tiempo

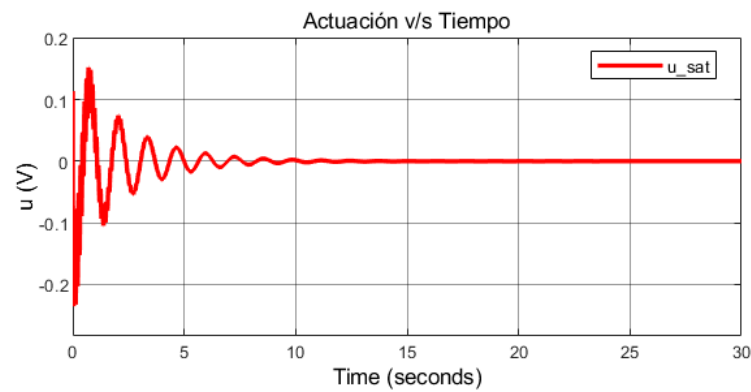


Figure 8.40: Quadprog Estado Cero Observador: Actuación u_{sat} en el Tiempo

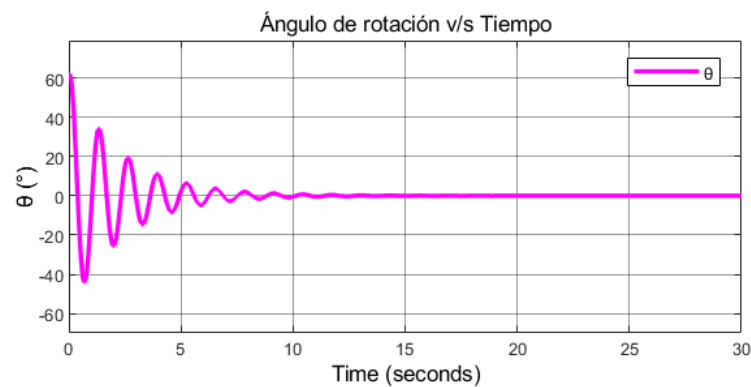


Figure 8.41: Quadprog Estado Cero Observador: Ángulo de rotación θ en el Tiempo

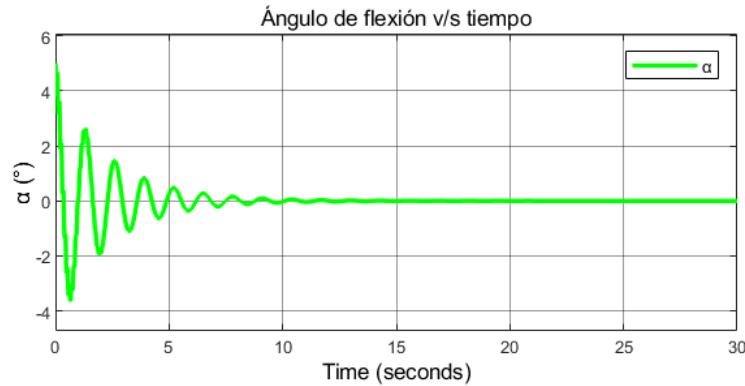


Figure 8.42: Quadprog Estado Cero Observador: Ángulo de flexión α en el Tiempo

En la figura 8.39 se observa la actuación calculada por el *MPC*.

En la Figura 8.40 se observa un peak al comienzo de la actuación producido por el movimiento inicial para llevar el sistema a la estabilidad, esta actuación no supera los $\pm 0.2[V]$, y en estado estacionario converge a cero.

En la Figura 8.41 se observa que parte en las condiciones iniciales establecidas y en un tiempo inferior a $10[s]$ donde converge al estado estacionario.

En la Figura 8.42 se observa la reducción de la oscilación en estado estacionario en comparación de la no aplicación de observadores de estado y se aprecia que se respetan las restricciones establecidas.

Luego de esto se procederá a estudiar el caso del seguimiento a referencia implementando observadores de estado.

Para ello se considera el diagrama de la Figura 8.43, que incorpora el observador de estados, cuya entrada son las salidas y_1 e y_2 junto a la actuación resultante de la prealimentación para la inestabilidad de la planta en tiempo discreto, los primeros cuatro estados de la salida se realimentan hacia el bloque *MPC*.

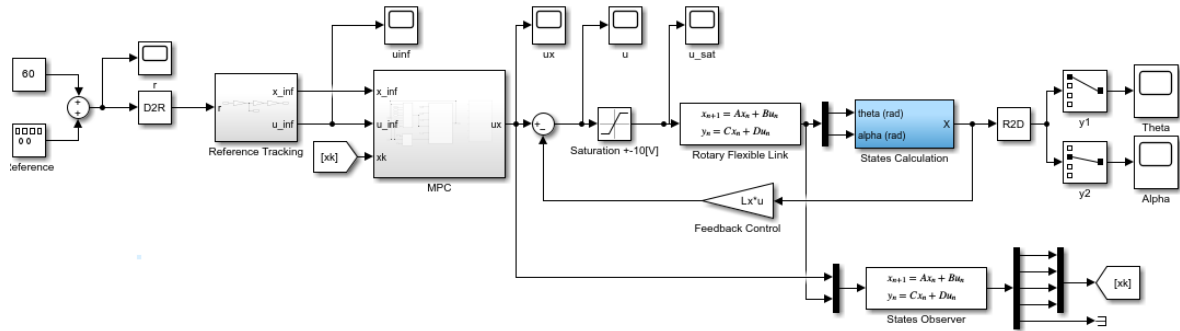


Figure 8.43: Diagrama de seguimiento a referencia con observador de estados

Se consideran las mismas condiciones iniciales y parámetros para la prealimentación de la inestabilidad obteniendo las siguientes señales medidas con bloques *scope*.

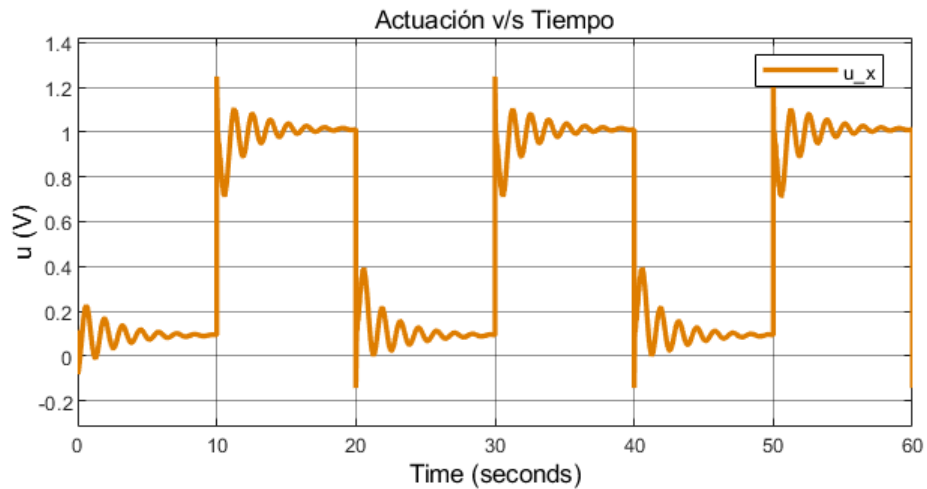


Figure 8.44: Quadprog Seguimiento a Referencia Observador: Actuación u_x en el Tiempo

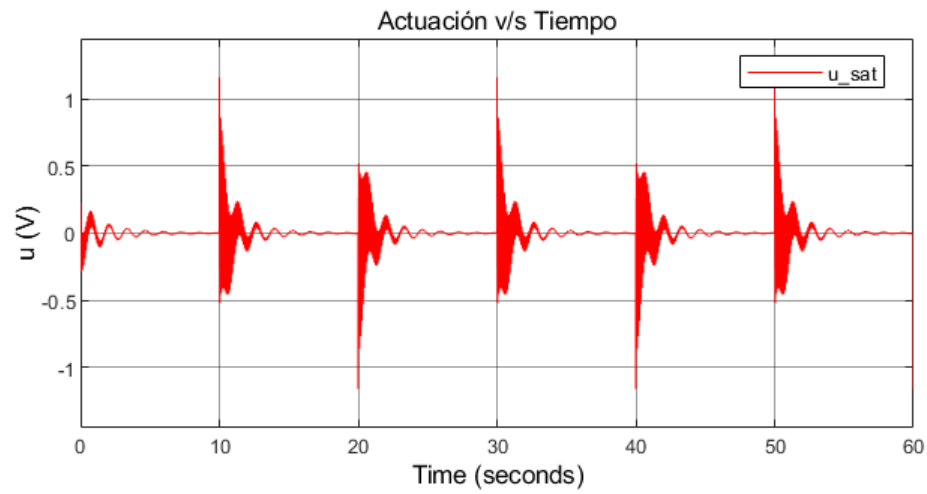


Figure 8.45: Quadprog Seguimiento a Referencia Observador: Actuación u_{sat} en el Tiempo

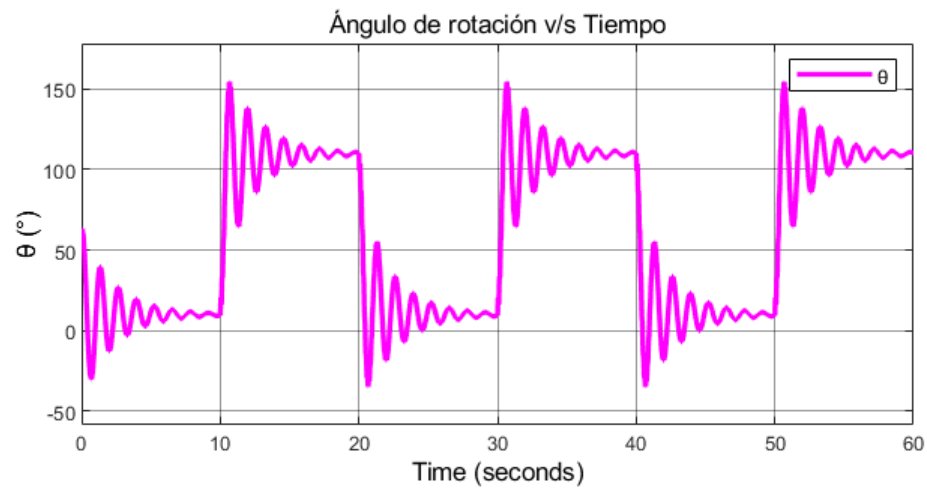


Figure 8.46: Quadprog Seguimiento a Referencia Observador: Ángulo de rotación θ en el Tiempo

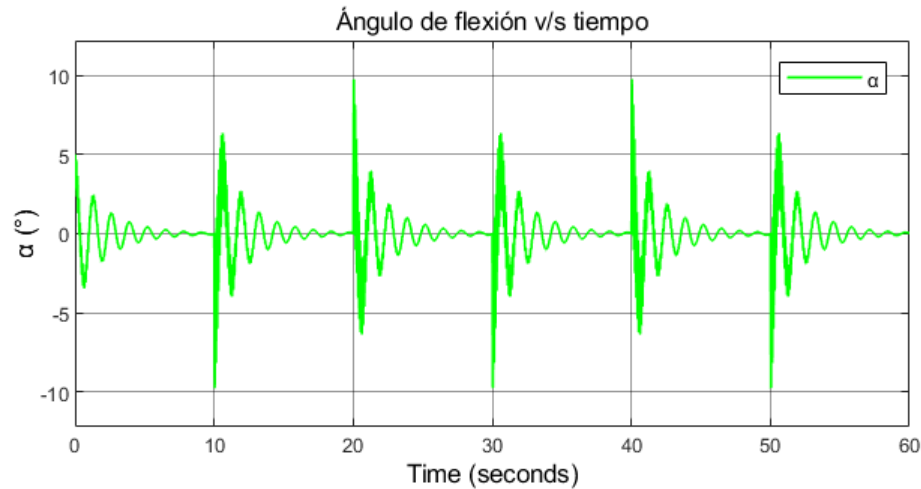


Figure 8.47: Quadprog Seguimiento a Referencia Observador: Ángulo de flexión α en el Tiempo

Se aprecia una gran mejoría en las señales, para el caso del ángulo de rotación este disminuye su sobre impulso logra llegar a la referencia en un tiempo de 10[s], respetando las restricciones como se puede ver en la Figura 8.46.

Para el caso del ángulo de flexión presente en la Figura 8.47 este posee puntos de valor mas altos al momento de conmutar la señal se referencia, estos no superan los $\pm 10^\circ$ de las restricciones.

Las actuaciones presentan mejorías en comparación del seguimiento a referencia sin observador de estados, mostrándose una señal mas limpia al comparar las Figuras 8.33 y 8.44 para la señal sin observador de estados y las Figuras 8.34 y 8.45 para la señal con observador de estados.

8.5. ADMM para la RFL

Ahora bien se abordará el diseño del control MPC utilizando el segundo método de optimización mencionado, que consiste en ADMM mencionado en el Capítulo 6, esto porque para trabajar en la planta real se requiere utilizar este método de optimización en QUARC, se espera obtener respuestas muy similares en comparación con utilizar *Active-Set* de Quadprog.

Se mantienen la misma base de código, diagrama de simulación y condiciones iniciales, pero lo

que cambia es el bloque *MCP* presente en la Figura 8.23. Dentro de él se sustituye la función *quadprog* por la función *ADMM* como se observa en la Figura 8.48, este bloque recibe las mismas matrices formuladas anteriormente, y en su interior se deben realizar ajustes como el parámetro de penalización ρ , que se encarga de controlar la estabilidad y convergencia del algoritmo o la cantidad de iteraciones k que debe realizar para obtener la actuación óptima u_x .

El *Código ADMM* representa el interior de este bloque y se encuentra la Sección 11.4.7 de los Anexos.

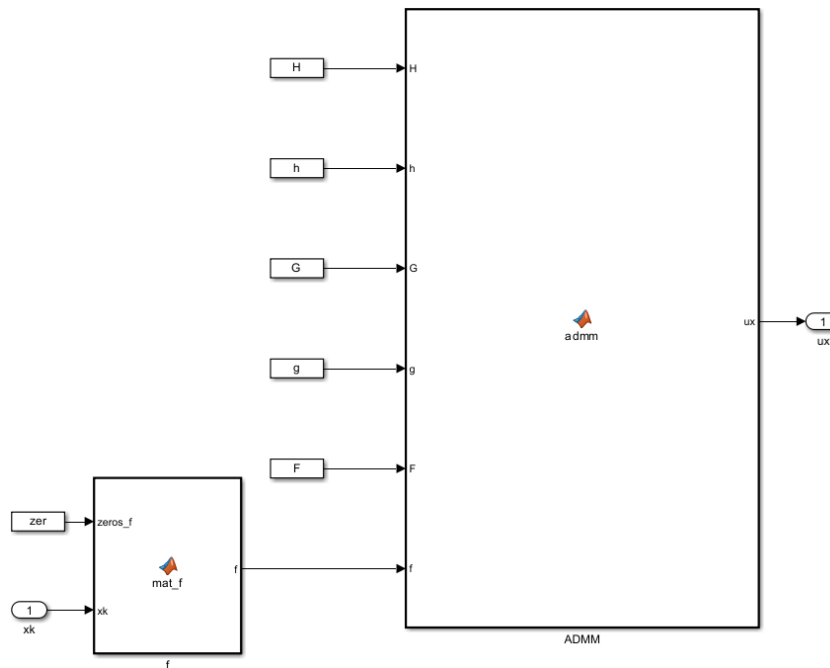


Figure 8.48: Interior bloque MPC con función ADMM

8.5.1. ADMM para estado cero y seguimiento a referencia

Se decide trabajar en primera instancia en el caso del estado cero, por lo que se utiliza el código *MPC para Estado Cero* de la Sección 11.4.1 de los Anexos junto con el esquema de simulación de la Figura 8.22. Para este caso se consideran las mismas condiciones iniciales y restricciones, donde el único cambio que se realiza, es el de reemplazar el bloque del método de optimización *Active-Set* de Quadprog por el presente en la Figura 8.48 el método ADMM.

Considerando lo anterior se procede a simular y se obtienen los siguientes resultados para la

variables de interés.

Mediante las Figuras 8.49, 8.50, 8.51 y 8.52 se puede observar que convergen rápidamente al equilibrio, llegando a estabilizarse en el valor cero en un tiempo de 5[s]. Todas las señales respetan las restricciones propuestas por lo que se tiene una respuesta óptima para el caso del estado cero.

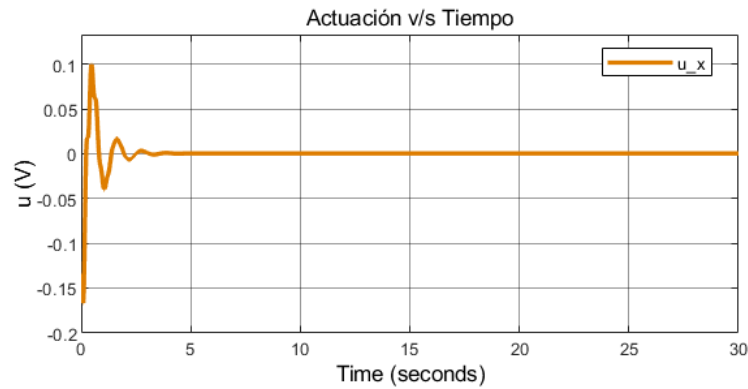


Figure 8.49: ADMM Estado Cero: Actuación u_x en el Tiempo

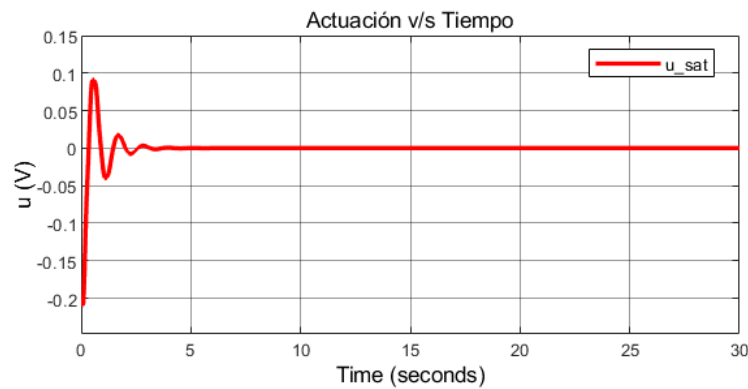


Figure 8.50: ADMM Estado Cero: Actuación u_{sat} en el Tiempo

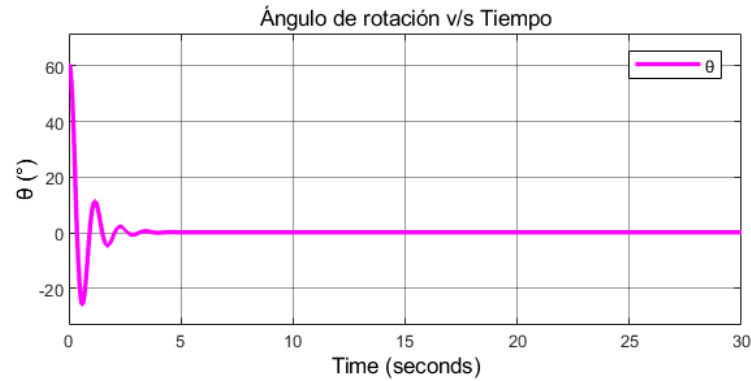


Figure 8.51: ADMM Estado Cero: Ángulo de rotación θ en el Tiempo

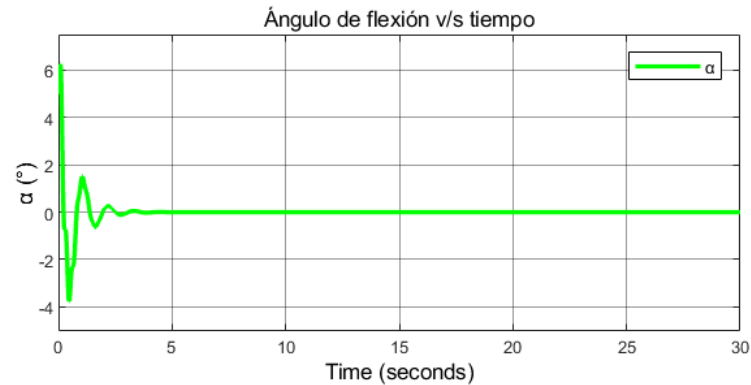


Figure 8.52: ADMM Estado Cero: Ángulo de flexión α en el Tiempo

Siguiendo los pasos realizados para el caso del estado cero, para desarrollar el caso del seguimiento a referencia se considera el esquema de simulación de la Figura 8.23, en el cual al interior del bloque *MPC* se ejecutan los mismos cambios presente en el caso del estado cero, donde se sustituye el bloque del método de optimización por el de la Figura 8.48 cuyo código se presenta en la Sección 11.4.7 de los Anexos.

Se decide utilizar la misma señal de referencia de la Figura 8.32 del método *quadprog* y el mismo tiempo de simulación. Tras ejecutar el script de MATLAB y el esquema de Simulink respectivo, se obtienen los siguientes resultados para las señales de interés, que consisten en la actuación de salida del *MPC*, la actuación luego de saturar que entra hacia la planta, el ángulo de rotación de la RFL y el ángulo de flexión de la RFL presentes en las Figuras a continuación.

La Figura 8.53 corresponde a la actuación de salida del *MPC*.

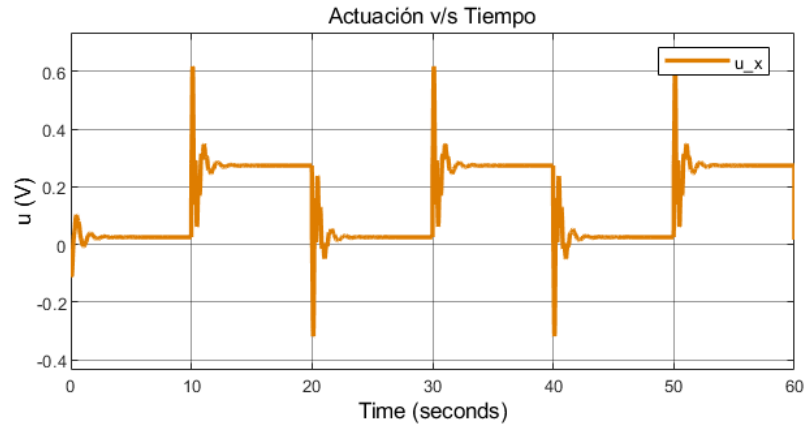


Figure 8.53: ADMM Seguimiento a Referencia: Actuación u_x en el tiempo

En la Figura 8.54 se aprecian puntos de valor mas altos en la actuación que consisten en la conmutación de la señal de referencia, la cual respeta las restricciones propuestas.

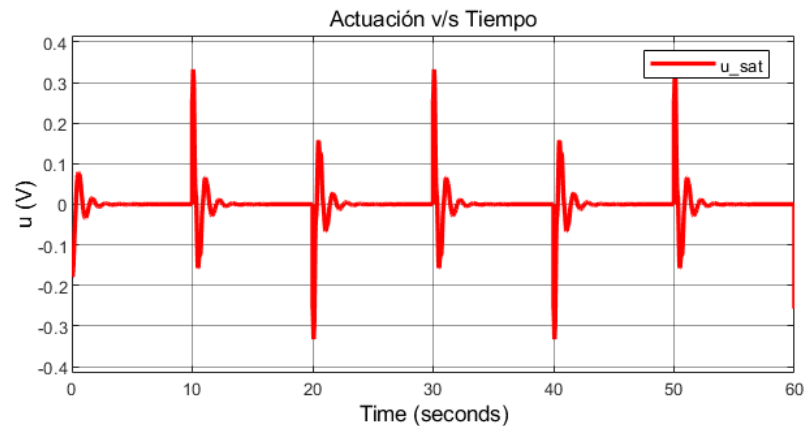


Figure 8.54: ADMM Seguimiento a Referencia: Actuación u_{sat} en el tiempo

En la Figura 8.55 se aprecia el ángulo de rotación que logra seguir la señal de referencia de buena manera, con mejor comportamiento que para el caso de implementar el método de optimización *Active-Set* de Quadprog, existe un pequeño sobre impulso en su respuesta y oscilaciones menores al llegar a la referencia debido a la naturaleza de la planta pero se respetan las restricciones en todo momento.

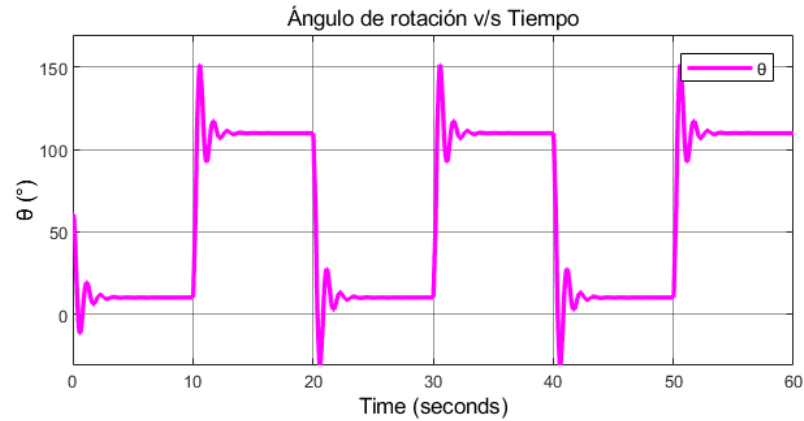


Figure 8.55: ADMM Seguimiento a Referencia: Ángulo de rotación θ en el tiempo

En la Figura 8.56 se observa que el máximo valor para el ángulo de flexión o bien llamado salida y_2 supera los $\pm 10^\circ$ por lo que no es apropiado según las restricciones propuestas para el problema de optimización.

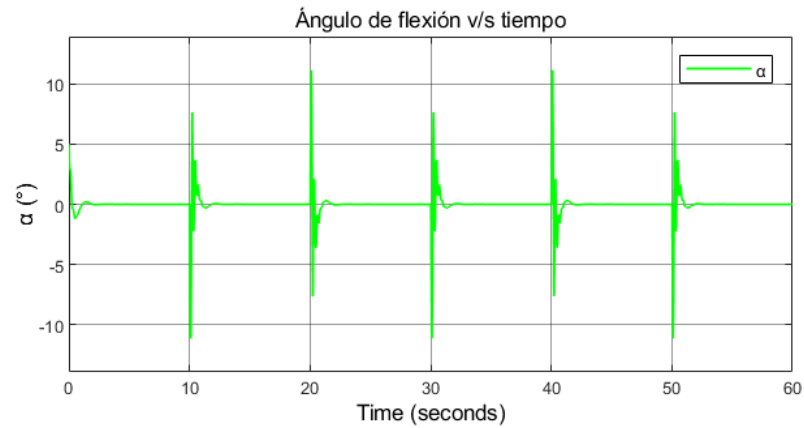


Figure 8.56: ADMM Seguimiento a Referencia: Ángulo de flexión α en el tiempo

8.5.2. Incorporación de observadores de estado para ADMM

Se decide implementar el observador de estados al igual que para el caso de la Figura 8.37, el cual se conecta al sistema como se aprecia resultando en el diagrama presente en la Figura 8.38. De este esquema se decide medir las mismas señales que para el caso de utilizar el método de optimización *quadprog*, por lo que se obtienen las siguientes respuestas.

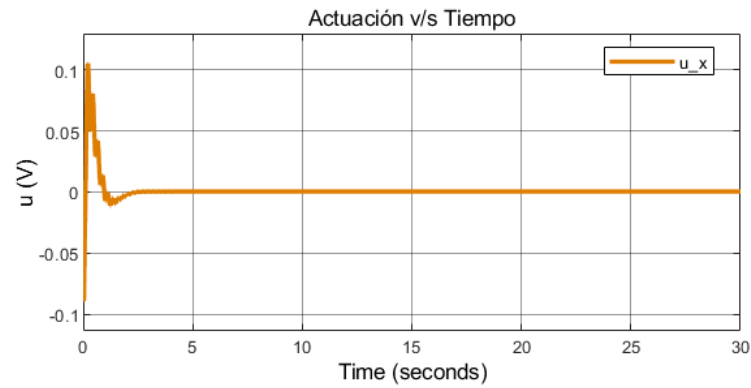


Figure 8.57: ADMM Estado Cero Observador: Actuación u_x en el tiempo

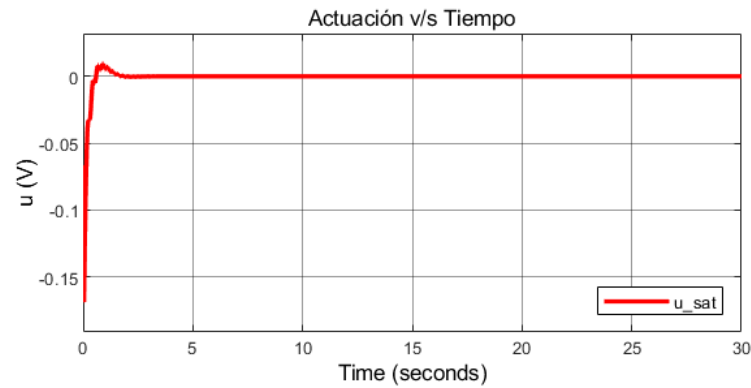


Figure 8.58: ADMM Estado Cero Observador: Actuación u_{sat} en el tiempo

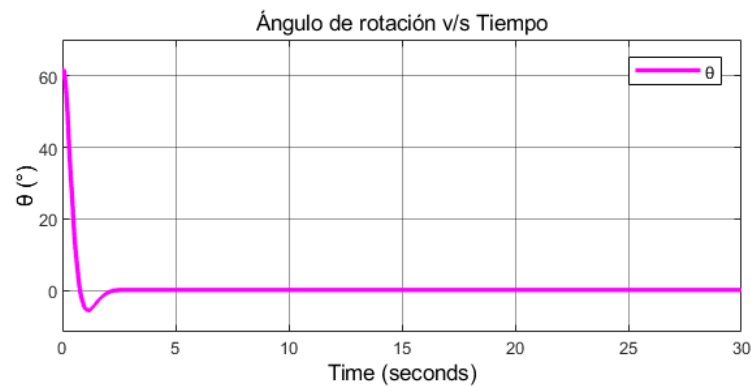


Figure 8.59: ADMM Estado Cero Observador: Ángulo de rotación θ en el tiempo

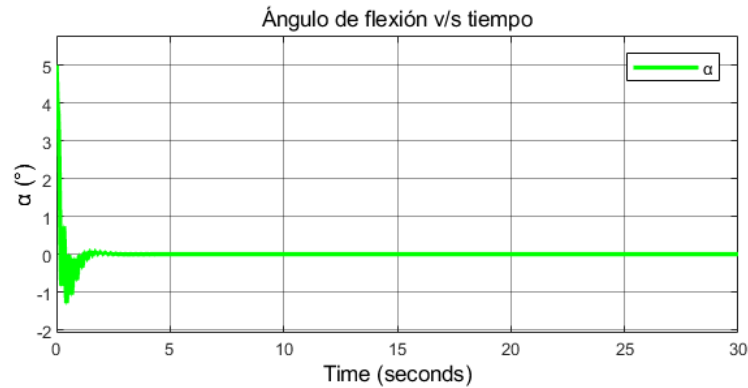


Figure 8.60: ADMM Estado Cero Observador: Ángulo de flexión α en el tiempo

Para finalizar con las simulaciones se procede a obtener la señales implementando el observador de estados, con tal de lograr mejorías en las señales al igual que el caso del estado cero. Se opta por utilizar el observador de estados de la Figura 8.37, con el propósito de realizar un mejor seguimiento que se evidencian en las siguientes imágenes.

A continuación se presenta la actuación saliente del *MPC* similar a la obtenida en el caso de no implementar observador de estados.

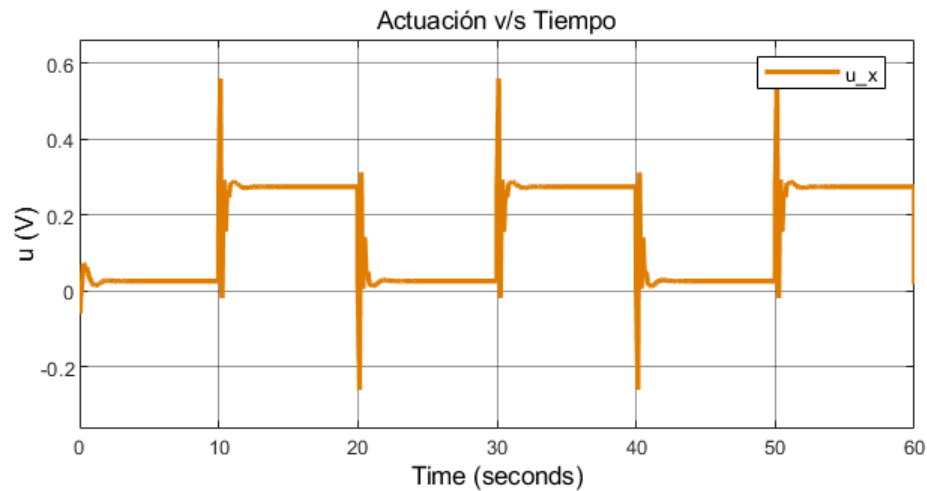


Figure 8.61: ADMM Seguimiento a Referencia Observador: Actuación u_x en el tiempo

En la Figura 8.62 la actuación obtenida para este caso es similar al del seguimiento a referencia sin observador de estados, respetando las restricciones establecidas.

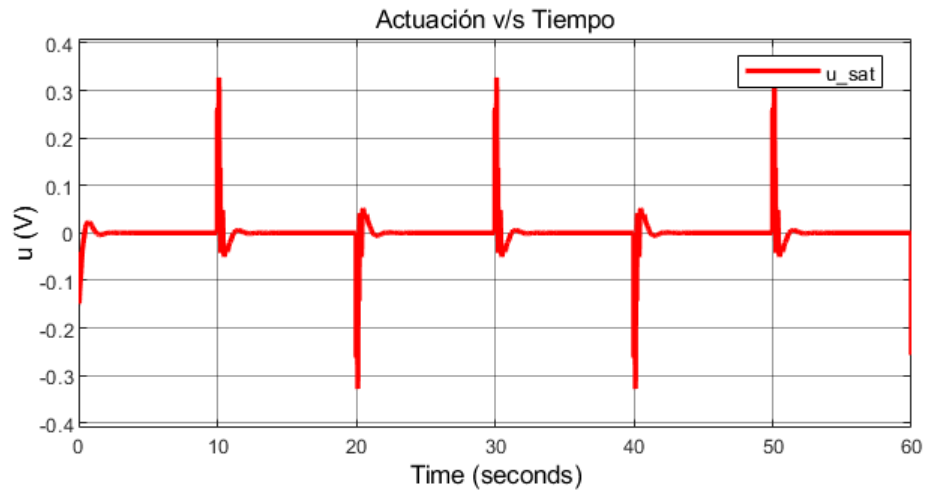


Figure 8.62: ADMM Seguimiento a Referencia Observador: Actuación u_{sat} en el tiempo

En la Figura 8.63 se aprecia un mejor seguimiento para la salida con respecto a la señal de referencia, presentando un menor overshoot y oscilaciones antes de alcanzar el estado estacionario en cada conmutación, se destaca que se respetan las restricciones establecidas.

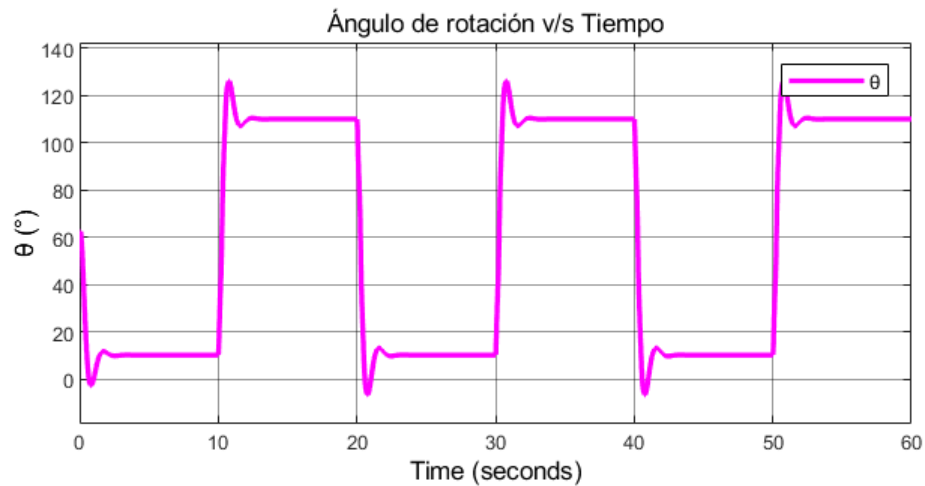


Figure 8.63: ADMM Seguimiento a Referencia Observador: Ángulo de rotación θ en el tiempo

Para finalizar en la Figura 8.64 se aprecia el ángulo de flexión de la RFL el cual se mantiene en valores apropiados para las restricciones establecidas en comparación con el caso de no utilizar el observador de estados, los puntos de valor mas altos presentes son ocasionados por la actuación para seguir la conmutación de la señal de referencia.

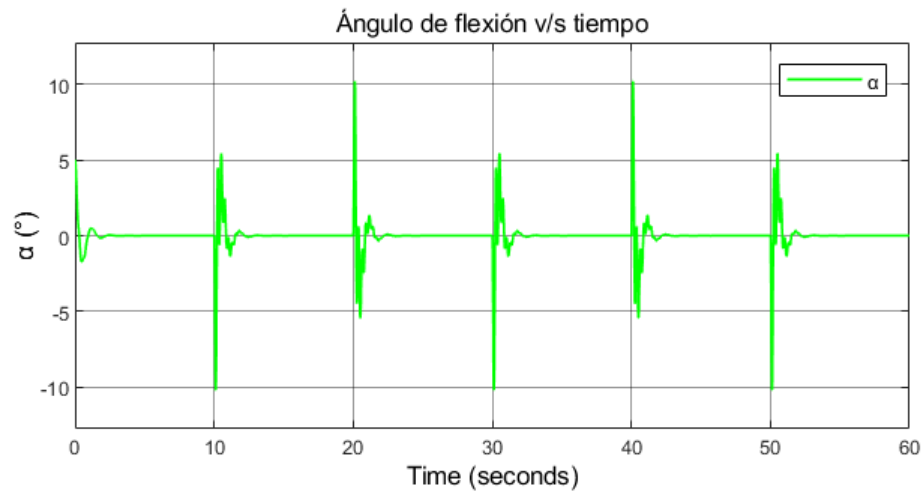


Figure 8.64: ADMM Seguimiento a Referencia Observador: Ángulo de flexión α en el tiempo

Finalmente se puede afirmar y concluir que la implementación del observador de estados genera mejoras en las señales de interés, logrando efectuar un mejor seguimiento de señales de referencias nulas y no nulas.

Sin embargo, se debe evaluar y considerar adecuadamente las ganancias del observador de estados para no ocasionar distorsiones en las señales de interés.

9. Implementación Hardware-Software dSPACE y Quarc

El objetivo de esta memoria es implementar control MPC mediante simulaciones, verificar la viabilidad y posibilidad de implementarlo físicamente en la planta RFL real. Para ello, se disponen de opciones mencionadas en el Capítulo 1.

Primero que todo, se mostrará el funcionamiento de ambos software con sus respectivos hardware. Luego, se procederá a la compilación del código y a las conexiones establecidas para realizar las mediciones, y finalmente se implementará el control sobre la planta.

Se abordará la implementación entre Simulink y dSPACE, y posteriormente se tratará la implementación entre Simulink y Quarc.

9.1. Implementación MATLAB Simulink & MicroLabBox dSPACE

dSPACE es una empresa líder en tecnología de software y hardware que se enfoca en proporcionar soluciones innovadoras para la simulación y validación de sistemas de control en una variedad de industrias. Con un énfasis particular en la conducción autónoma, la electromovilidad y la digitalización, dSPACE ofrece herramientas integrales que abordan desafíos específicos en campos como la automoción, la aeroespacial y la automatización industrial.

Uno de sus productos destacados es el *MicroLabBox 1202*, una solución compacta y rentable diseñada para entornos de laboratorio. Con más de 100 canales de entrada/salida y una combinación de potente procesamiento y FPGA, el *MicroLabBox* ofrece comunicación en tiempo real con baja latencia, lo que lo convierte en una herramienta esencial para prototipos y validación de sistemas de control en laboratorio.

El *MicroLabBox 1202* proporciona una plataforma versátil para configurar rápidamente aplicaciones de control, pruebas o mediciones en áreas que van desde la robótica hasta la ingeniería médica. Su capacidad para comunicarse en tiempo real entre sistemas simulados y físicos permite probar sistemas extremadamente rápidos y complejos, como el control de motores o la cancelación de ruido, con eficiencia y precisión. En resumen, el *MicroLabBox 1202* de dSPACE ofrece una solución integral para explorar una amplia gama de aplicaciones

tecnológicas, brindando a ingenieros y científicos las herramientas necesarias para desarrollar soluciones innovadoras de manera eficiente y confiable en el laboratorio [7].

9.1.1. Hardware de dSPACE

En la Figura 9.65 se presenta el equipo MicroLabBox mencionado anteriormente. Este se utilizará como microcontrolador para realizar mediciones digitales del encoder y del sensor de la galga extensiométrica, con el fin de captar las mediciones de la flexión generada en la barra de la RFL durante los movimientos provocados por la actuación.



Figure 9.65: MicroLabBox dSPACE

Este equipo se conecta vía Ethernet al computador que posee el software y configuraciones para implementar el control MPC.

Para el conexionado del sistema, se fabricó un cable esencial para captar las mediciones del encoder, el cual se alimenta con 5[V]. Por ello, se realiza una conexión hacia la fuente regulable del MicroLabBox, situada en la parte posterior de la imagen de la Figura 9.65, mediante cables BNC. Para trabajar en condiciones adecuadas, también posee un conector DB-50 pines para la conexión Digital I/O. El cable del tipo DIN 5 pin se conecta directamente con la salida

encoder de la Rotary Flexible Link de la Figura 9.111. El cable diseñado se observa en la Figura 9.66. Además, se considera en la implementación el amplificador de la Figura 9.109, cuya salida *to Load* se conecta con el cable tipo DIN de 6 pin a DIN de 4 pin que se conecta hacia el motor del SRV02, base de la RFL. El cable 6 pin-miniDIN a 6 pin-miniDIN conecta el módulo de la galga extensiométrica con la entrada del sensor análogo del VoltPAQ-X2 Amplifier. La entrada del amplificador se conecta con la salida análoga *A0-ch1* mediante el cable de la Figura 9.67. Finalmente, de la salida *to ADC* se conecta hacia la entrada análoga *AI ch1* mediante un adaptador BNC agregado al cable blanco RCA del DIN 5 pin a 4xRCA de la Tabla 9.3.

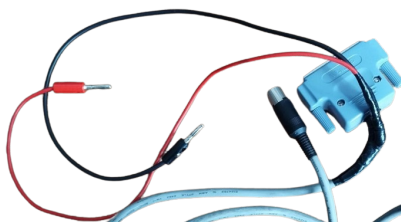


Figure 9.66: Cable dSPACE Encoder

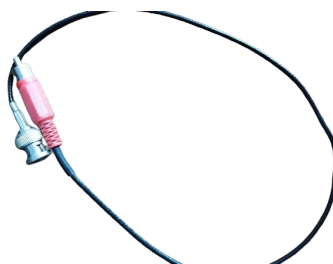


Figure 9.67: Cable RCA to BNC



Figure 9.68: Cable DIN 4 pin to DIN 6 pin

9.1.2. Software de dSPACE

Para poder desarrollar las actividades y objetivos propuestos en dSPACE se utilizaron los siguientes software:

- **Vitis Model Composer:** Es una herramienta de desarrollo de software proporcionada por Xilinx empresa reconocida por sus dispositivos y herramientas de alto rendimiento como lo son las FPGAs por sus siglas en ingles Field-Programmable Gate Arrays o Matriz de Puertas Programables. Al ejecutar el Vitis Model Composer, se ejecuta el software MATLAB junto a la librería Real-Time Interface o *RTI*.
- **Real-Time Interface:** Corresponde a un Software desarrollado por dSPACE, que permite la conexión MATLAB-Simulink & MicroLabBox.

Se emplean los bloques RTI DS1202, cuya disponibilidad se brinda al elegir en la creación del diagrama Simulink con el template RTI1202 en la Figura 9.69.

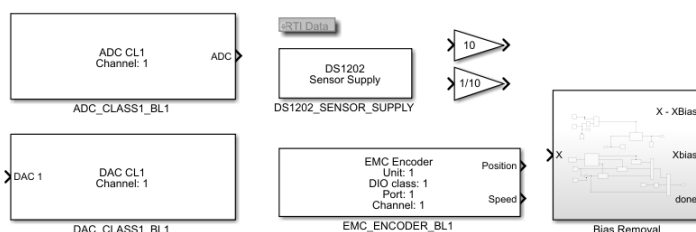


Figure 9.69: Interfaz RTI

- **ControlDesk:** ControlDesk es una herramienta de software proporcionada por dSPACE, diseñada para facilitar la visualización, simulación, análisis y calibración de sistemas de control en tiempo real. Permite la visualización de datos en tiempo real, ofrece una interfaz de usuario personalizable, herramientas de análisis, facilita la calibración de parámetros y permite la automatización de tareas. En la Figura 9.70 se muestra su interfaz.

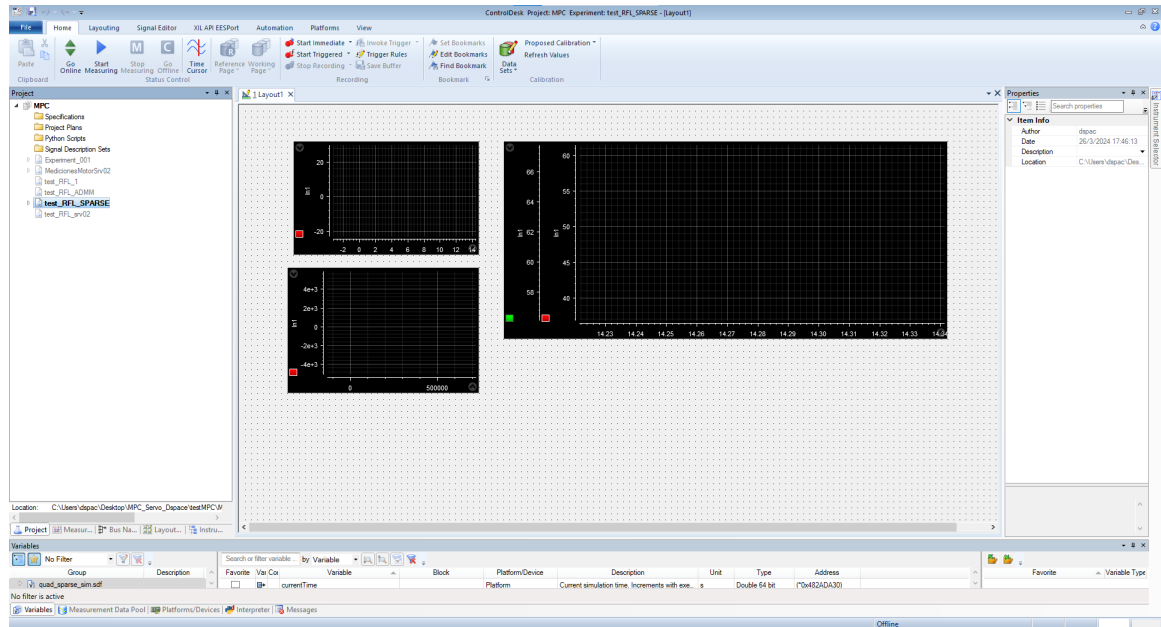


Figure 9.70: Software ControlDesk

Para desarrollar los diagramas de simulación e implementarlos en dSPACE se deben utilizar los bloques de la Figura 9.69.

A continuación se mostrará la configuración de los bloques utilizados.

- **ADC_CLASS_1_BL1:** Permite recibir una señal analógica a través de una entrada analógica del MicroLabBox. Se empleó el puerto AI1 ch1.

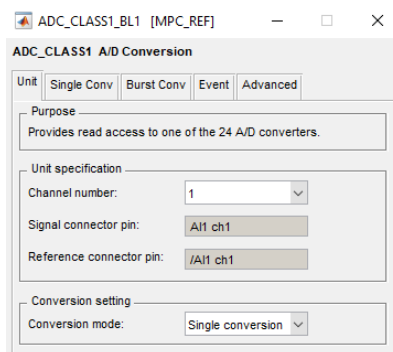


Figure 9.71: Bloque ADC_CLASS_1_BL1

- **DAC_CLASS_1_BL1:** Permite transmitir una señal digital a través de una salida analógica del MicroLabBox. Se empleó el puerto AO1 ch1.

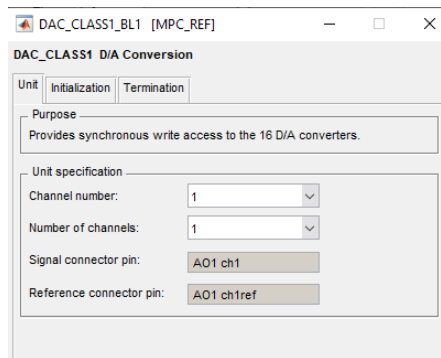


Figure 9.72: Bloque DAC_CLASS_1_BL1

- **RTI Data:** Permite compilar y construir el archivo empleando el toolbox RTI, se incluye al crear el esquema mediante el template RTI1202, no se debe copiar ni borrar.
- **DS1202_SENSOR_SUPPLY:** Permite ajustar el voltaje de la fuente programable que se encuentran en la cara posterior, se utiliza para alimentar el Encoder con 5[V].

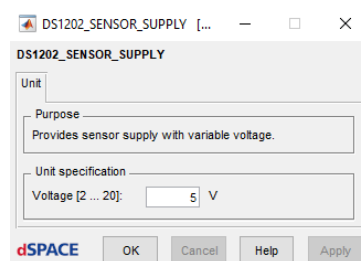


Figure 9.73: DS1202_SENSOR_SUPPLY

- **EMC_Encoder_BL1:** Permite recibir una señal digital del Encoder y decodificarla, entregando la posición angular de la barra en grados. Se empleó el puerto Digital I/O 1 y los canales 1-2, se configuró el incremento del Encoder en 1024, este registra mediciones entre 0[°] y 360[°], si excede los rangos la cuenta continúa descendiendo o aumentando desde los 360[°] o los 0[°].

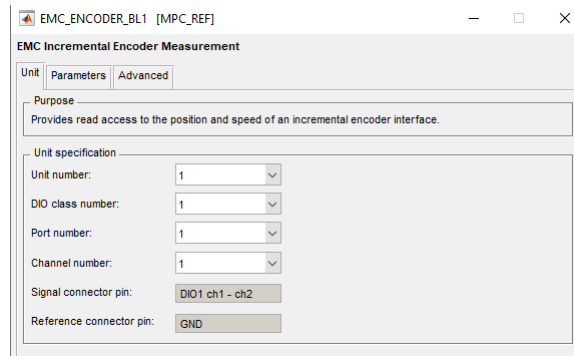


Figure 9.74: Bloque EMC_Encoder_BL1

- **Ganancia 1/10:** Se debe considerar al momento de transmitir.
- **Ganancia 10:** Se debe considerar al momento de recibir datos.
- **Bias Removal:** Este bloque permite eliminar el Bias o Sesgo de la señal proveniente del Sensor de la Galga Extensiométrica para ajustar las desviaciones en el valor captado.

Para poder realizar la conexión entre Simulink y el MicroLabBox se debe tener en cuenta las siguientes instrucciones:

- **Conexión:** Se debe verificar la conexión vía Ethernet entra el MicroLabBox con la PC en el que se utilice y encender el equipo.
- **Inicialización:** Se debe de ejecutar el software *Vitis Model Composer* con ningún proceso de MATLAB anteriormente, seleccionar la opción *RTI1202* y verificar en la ventana de comandos que esté listo el proceso como se ve en las Figuras 9.75 y 9.76 debido a que tarda en ejecutar las librerías.

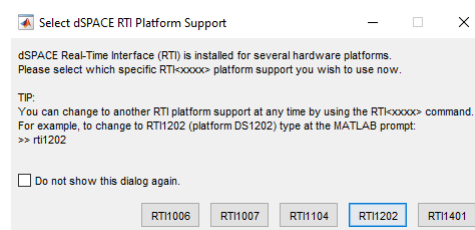


Figure 9.75: Selección dSPACE RTI Platform Support

```

Command Window
RTISTBM      RTI Synchronized Time Base Manager Bloc... 1.4.3      02-Nov-2021 okay
RTIUSBFlightRec RTI USB Flight Recorder Blockset      1.2.5      02-Nov-2021 okay
RTIWATCHDOG   RTI Watchdog Blockset      2.1.4      02-Nov-2021 okay
MIPS          Model Interface Package for Simulink      4.6        02-Nov-2021 okay
DSMLCON25     dSPACE MATLAB Connection 2.5 (win64)      2.5.2      02-May-2021 okay

*** RTI Platform Support RTI1202 activated.
*** Note: If you are missing ASM products they are probably not decrypted.
To start the decryption process please close MATLAB and
start the dSPACE Installation Manager.
Vitis Model Composer: Block Reference User Guide Tutorial Examples
Provide feedback for Vitis Model Composer.
fx >>
  
```

Figure 9.76: Verificación en ventana de comandos

- **Uso de Software Simulink:** Se debe de verificar la ruta de trabajo, ya que al compilar el diagrama se guardan los documentos ahí. Luego se debe crear un archivo Simulink y seleccionar el Template *RTI1202*.

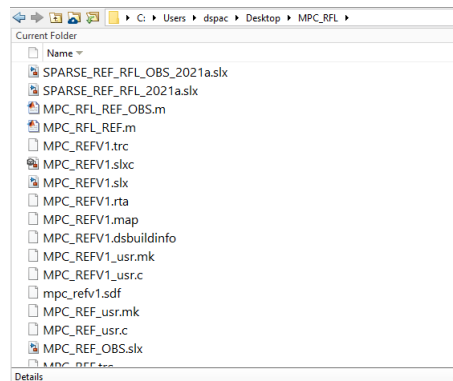


Figure 9.77: Verificación de ruta de trabajo

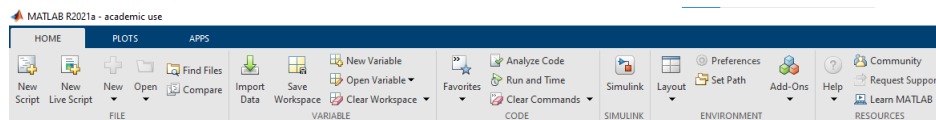


Figure 9.78: Creación de esquema Simulink

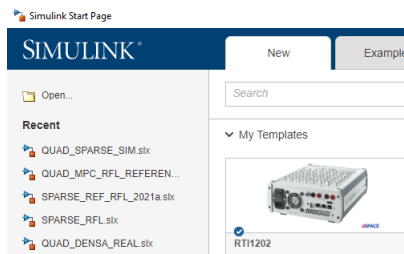


Figure 9.79: Selección de Template DS1202

Para poder desarrollar el esquema adecuadamente se debe configurar el esquema Simulink seleccionando la opción *Settings* de la Figura 9.80, en cuanto al método de resolución se debe configurar con la opción *discrete (no continuous states)* con el tiempo de muestro a elección.

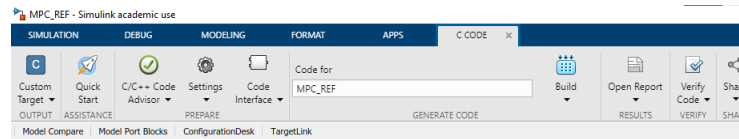


Figure 9.80: Ventana CCODE

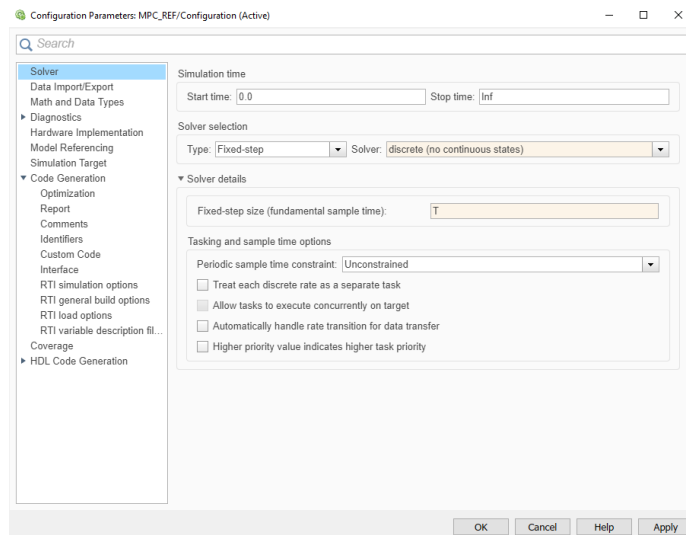


Figure 9.81: Selección de Solver Discrete

Además se debe verificar que no se encuentre seleccionada la opción *Load application after build* para que no aparezcan errores al momento de compilar.

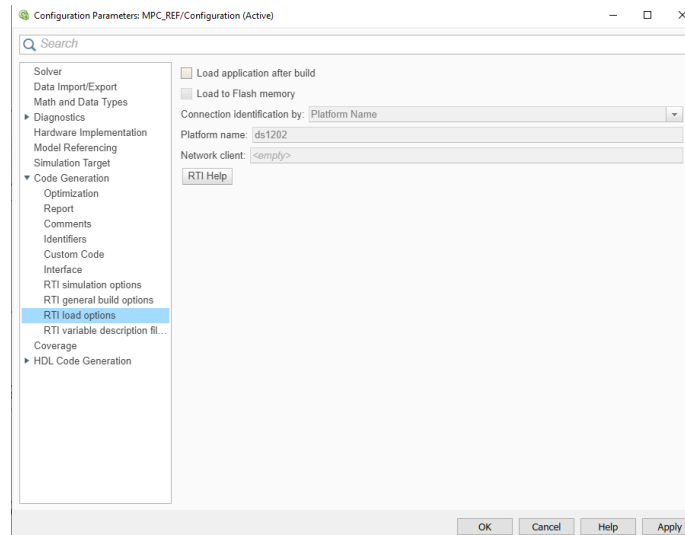


Figure 9.82: Desmarcar opción Load application after build

Se debe construir el diagrama Simulink utilizando los bloques necesarios, como se muestra en la Figura 9.69, y luego ejecutar el script de MATLAB para cargar los datos. Después, se compila y construye el archivo utilizando la opción *Build*. Este proceso genera un archivo *.sdf*, que significa *Structured Data File*, el cual almacena datos estructurados para facilitar su lectura y procesamiento en el software *ControlDesk*.

- **Uso de Software ControlDesk** Se debe ejecutar el software seleccionando la ventana 9.83 y la opción New para crear un proyecto y/o experimento.

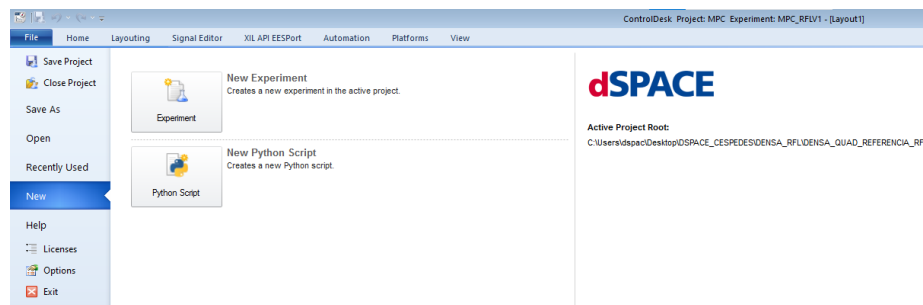


Figure 9.83: Ventana File y creación de Experimento

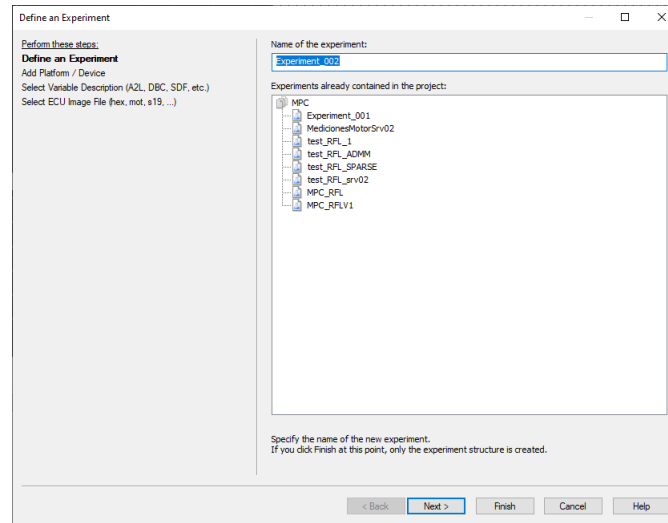


Figure 9.84: Creación de experimento

Tras crear el experimento se deben mantener las configuraciones por defecto e importar el archivo en formato *.sdf*

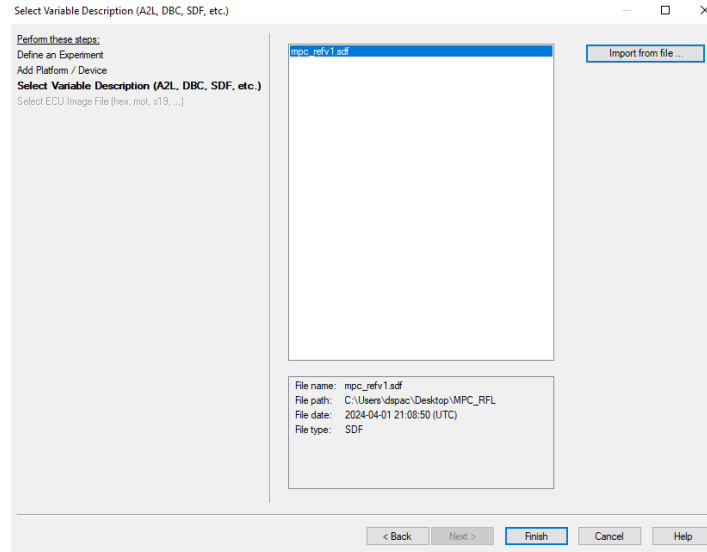


Figure 9.85: Importar archivo *.sdf*

Tras importar el archivo se pueden visualizar las señales de interés seleccionando la opción *Model Root* de la Figura 9.86 y añadiendo gráficos con la opción *time plotter* de la Figura 9.87.

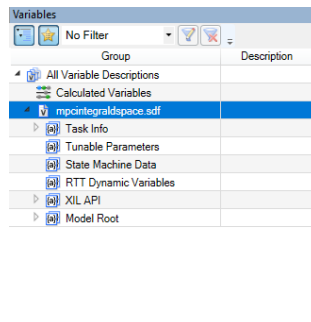


Figure 9.86: Selección de señales a medir

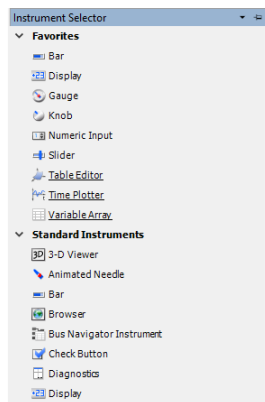


Figure 9.87: Instrumentos para medición de señales

Y tras configurar la interfaz se puede cargar el programa mediante la opción *go online* de la interfaz del software ControlDesk como se aprecia en la Figura 9.88, para realizar las mediciones se utiliza la opción *Start Measuring*, luego de ello se pueden observar en los diagramas *Time Plotter* que se añaden de la ventana de instrumentos.

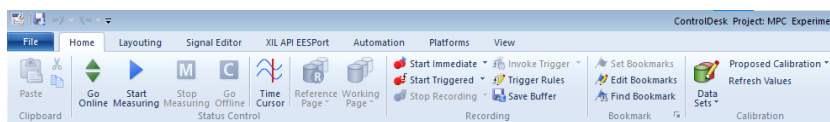


Figure 9.88: Ventana Home de controlDesk

9.1.3. Implementación de MPC en dSPACE

Se ejecutó el script desarrollado para las simulaciones y se diseñó el esquema de control *MPC* adaptado para su implementación en el software dSPACE, que se encuentra conectado a la

planta RFL real, como se muestra en la Figura 9.89. Para este propósito, se utilizaron los bloques de la interfaz RTI1202, detallados en la Figura 9.69. En el diseño, se incorporó un observador de estados para estimar los estados y las perturbaciones, los cuales se realimentan al *MPC* para el ajuste y el seguimiento de la referencia.

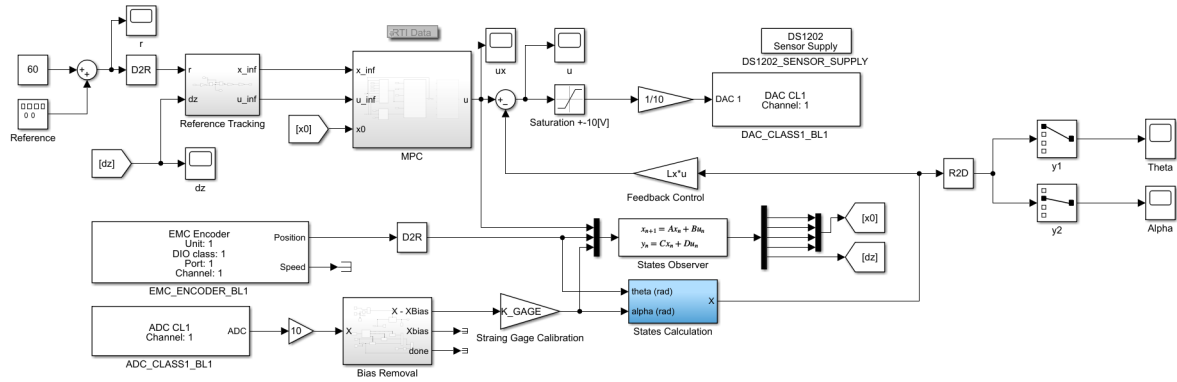


Figure 9.89: Esquema de Control MPC con Observador de Estados Planta Real

Para el control se consideraron como restricciones las presentes en 9.128, tanto para los estados como para la actuación.

$$\begin{aligned}
 x_{\min} &= \begin{bmatrix} -360 \\ -5 \\ -1000 \\ -180 \end{bmatrix}, & x_{\max} &= \begin{bmatrix} 360 \\ 5 \\ 1000 \\ 180 \end{bmatrix}, \\
 u_{\min} &= -10, & u_{\max} &= 10
 \end{aligned} \tag{9.128}$$

Al compilar el código utilizando el comando "*ctrl+b*", no se encontraron problemas. Se procedió a crear el experimento en ControlDesk, donde se añadieron *Time Plotter* para observar las señales de interés. Además, en el siguiente vídeo [25] se puede apreciar el funcionamiento del control MPC sobre la planta real.

- **Referencia:** Se consideró una señal de referencia de amplitud 50[°] centrada en 60[°], para obtener una buena respuesta por parte del Encoder y se consideró un periodo de 20[s] para la señal en un tiempo de medición de 60[s].

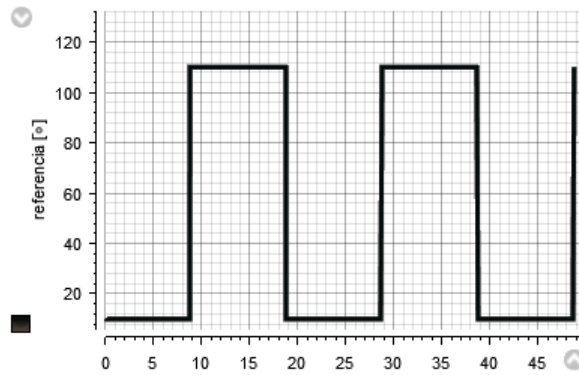


Figure 9.90: Control MPC dSPACE: Referencia en el Tiempo

- **Ángulo de rotación:** Se aprecia la salida y_1 parte en valores cercanos a $10[^\circ]$, respeta los tiempos de la señal de referencia y asciende su valor hacia $120[^\circ]$. Todo esto respetando las restricciones propuestas para el estado x_1 .

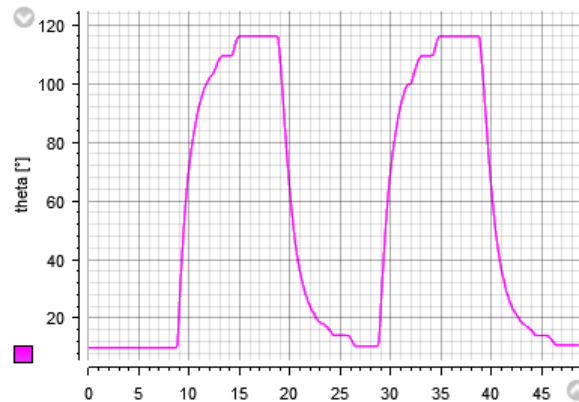


Figure 9.91: Control MPC dSPACE: Ángulo de rotación θ en el Tiempo

- **Ángulo de flexión:** Esta señal presenta peaks de valor al momento de conmutar desde los $10[^\circ]$ hasta $110[^\circ]$, los valores para la señal se mantienen entre $\pm 4.5[^\circ]$ respetando las restricciones para el estado x_2 .

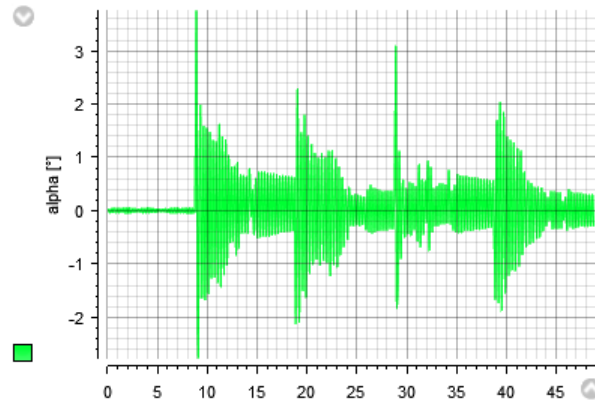


Figure 9.92: Control MPC dSPACE: Ángulo de flexión α en el Tiempo

- **Actuación u_x :** Corresponde a la actuación u_x calculada por la optimización y se suma con la señal u_{inf} siendo la salida del *MPC*.

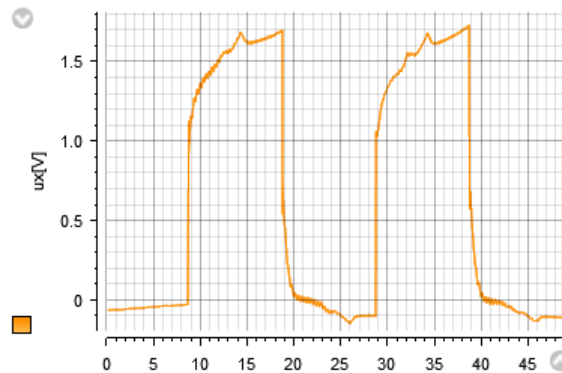


Figure 9.93: Control MPC dSPACE: Actuación u_x en el Tiempo

- **Actuación u :** Corresponde a la señal antes de ser saturada ente $\pm 10[^\circ]$, esta señal presenta peaks al momento de conmutar los cuales no superan los $\pm 1.5[V]$, también presenta amortiguaciones al momento de alcanzar la señal de referencia debido a que controla la vibración existente en la barra.

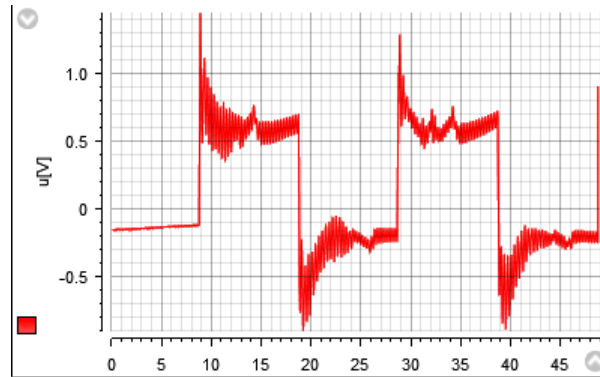


Figure 9.94: Control MPC dSPACE: Actuación u en el Tiempo

- **Estimación de la perturbación:** Corresponde a la perturbación estimada por el observador de estados.

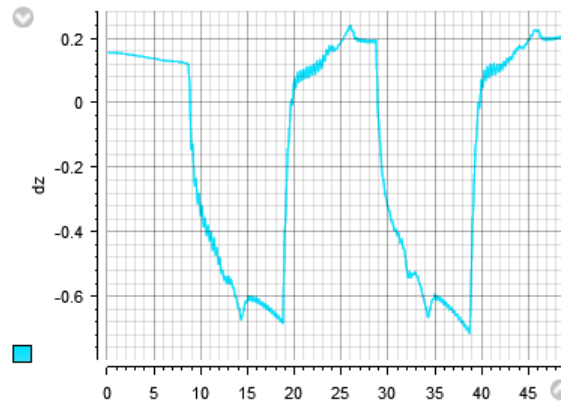


Figure 9.95: Control MPC dSPACE: Estimación de la perturbación d_z en el Tiempo

9.1.4. Implementación de Redes Neuronales en dSPACE

En base a la implementación del control MPC para la RFL con el software dSPACE, se grabaron los datos correspondientes al estado estimado, estado infinito y actuación infinita. Siguiendo las instrucciones del Capítulo 5.1, se obtuvieron los resultados del controlador basado en redes neuronales, ejecutados mediante el *Código Redes Neuronales* de la Sección 11.5.3 de los Anexos.

Para la red neuronal se utilizó una capa de entrada con 9 neuronas, una capa oculta con 2 neuronas utilizando la función de activación ReLU, y una capa de salida con 1 neurona con función de activación lineal. Se empleó el método de optimización ADAM [8] con un ratio

de aprendizaje de 0.1 para el entrenamiento, dado que es uno de los más populares en el entrenamiento de redes neuronales.

El dataset *train_rflmpc.csv*, que contiene datos medidos con el software ControlDesk de dSPACE, se utilizó como conjunto de entrenamiento.

En la Figura 9.96 se muestra la magnitud de la pérdida en función del número de épocas, utilizada para evaluar la eficacia del modelo entrenado. El cálculo del error cuadrático medio revela una pérdida leve, indicando que el modelo está aprendiendo de manera efectiva y mejorando su capacidad para realizar predicciones más precisas.

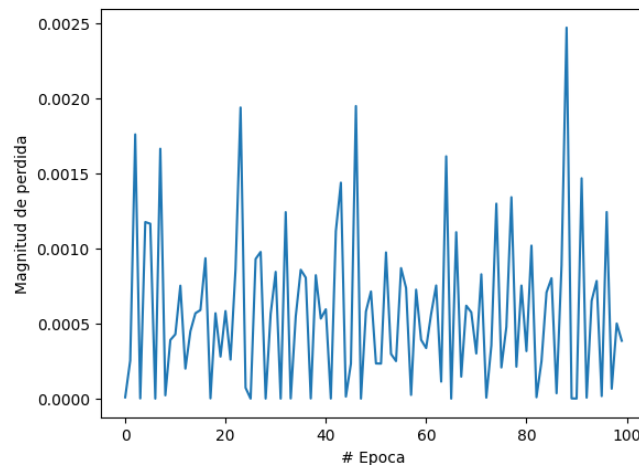


Figure 9.96: Magnitud de pérdida por Número de Épocas

En la línea 22 del código se grafica en la Figura 9.97 la actuación u_x predicha con los datos de entrenamiento que fueron medidos, y esta señal entra al lazo de prealimentación.

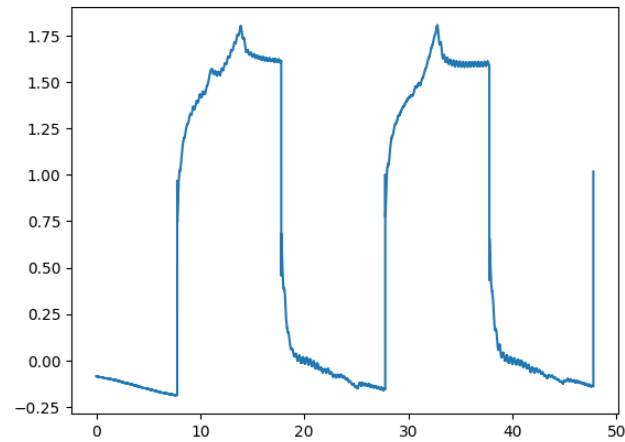


Figure 9.97: Actuación u_x : Predicción con datos de entrenamiento

A continuación se gráfica en la línea del código los datos de la actuación medida utilizados para entrenar la red neuronal.

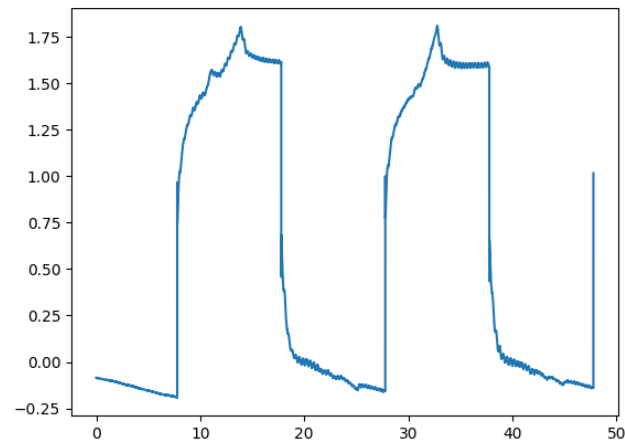


Figure 9.98: Actuación u_x : Datos de entrenamiento

Para evaluar y reforzar el modelo de la red neuronal entrenada, se utilizó el dataset *test_rflmpc.csv*, que fue grabado y recopilado mediante el software ControlDesk de dSPACE. Con estos datos, se procede a validar el modelo de la red neuronal, obteniendo una salida predicha que se muestra en la Figura 9.99.

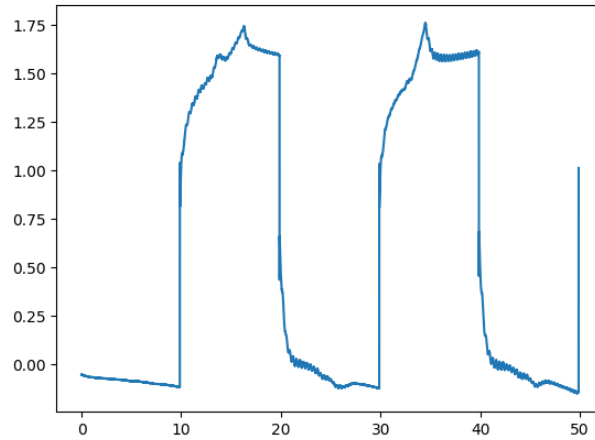


Figure 9.99: Actuación u_x : Predicción con datos de validación

Gráfico de u_x con los datos de validación presentes en la Figura 9.100.

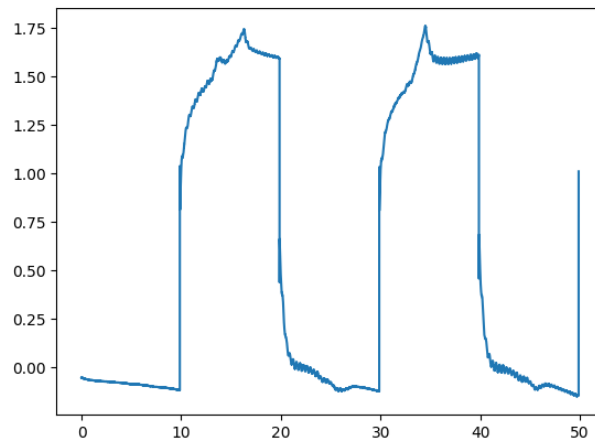


Figure 9.100: Actuación u_x : Datos de validación

Al observar las figuras, se puede apreciar que las señales son adecuadas para desarrollar un buen modelo del sistema utilizando los datos recopilados en ambas instancias.

Para el entrenamiento del modelo, se obtuvo un error promedio de -0.584% con una desviación estándar de 58.411. En cuanto a la validación, el error promedio fue de -0.246% y la desviación estándar de 23.475.

Se emplearon los mismos bloques para el control *MPC* en la RFL en dSPACE, sustituyendo el bloque *MPC* por el bloque *Neural Network*, como se muestra en la Figura 9.101. Este

[illegible]

El código del bloque *Neural Network* desarrollado para el controlador se encuentra en la Sección 11.5.2 de los Anexos.

La señal de referencia corresponde a una onda cuadrada con una amplitud de $50[^\circ]$, centrada en $60[^\circ]$, y una frecuencia de $\frac{1}{20}[Hz]$.

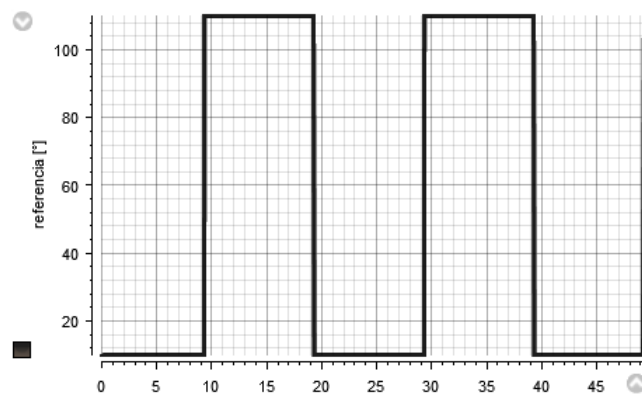


Figure 9.102: Redes Neuronales dSPACE: Referencia en el Tiempo

En el siguiente vídeo [26] se observa el funcionamiento en tiempo real de la planta RFL con el equipo MicroLabBox y el software ControlDesk de dSPACE.

En el vídeo se aprecia en color magenta que el ángulo de rotación θ sigue de manera apropiada la señal de referencia, mostrando solo una leve desviación. Esta desviación es consecuencia de la dinámica de la planta, que compensa la perturbación y la vibración generada durante el movimiento, mientras respeta las restricciones propuestas.

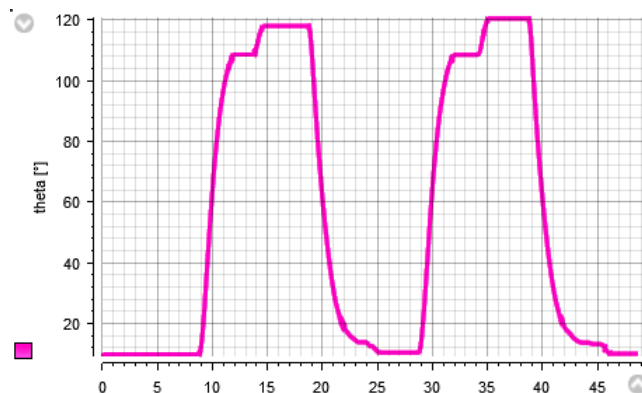


Figure 9.103: Redes Neuronales dSPACE: Ángulo de rotación θ en el Tiempo

Para el caso del ángulo de flexión α , se observa una pequeña magnitud que no supera los $1.5[^\circ]$, reflejando los picos que ocurren cuando el ángulo de rotación θ lo limita. Las restricciones establecidas se respetan de manera apropiada.

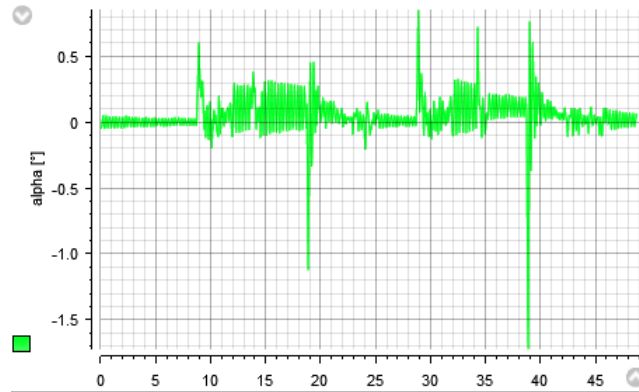


Figure 9.104: Redes Neuronales dSPACE: Ángulo de flexión α en el Tiempo

En el caso de la actuación, se observa que es opuesta a la estimación de la perturbación. Posee la misma forma que la señal u_x , atenuada por el lazo de prealimentación, y resulta adecuada para el control aplicado al sistema, respetando las restricciones propuestas.



Figure 9.105: Redes Neuronales dSPACE: Actuación u en el Tiempo

En naranja se aprecia la actuación u_x previa a la prealimentación para la estabilización del sistema, respeta también las restricciones establecidas.

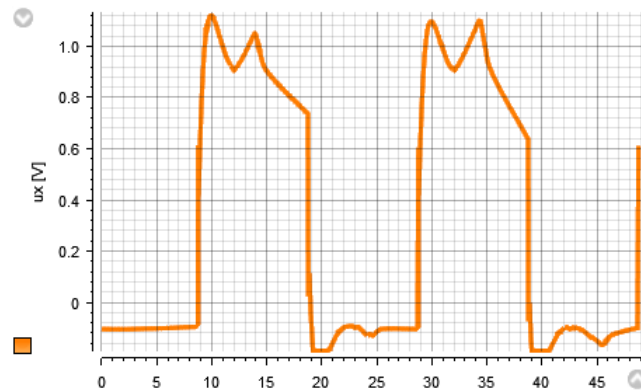


Figure 9.106: Redes Neuronales dSPACE: Actuación u_x en el Tiempo

En cian se puede apreciar la estimación de la perturbación medida por el observador de estados.

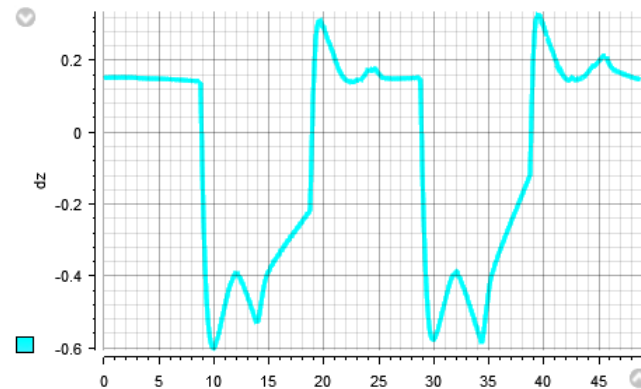


Figure 9.107: Redes Neuronales dSPACE: Estimación de la perturbación d_z en el Tiempo

Todas las señales identificadas anteriormente respetan adecuadamente las restricciones establecidas, logrando un mejor control en comparación con la implementación del *MPC* usando Quadprog sobre la RFL en dSPACE. Considerando lo anterior, se da el visto bueno para ejercer control en la interfaz MATLAB Simulink & Quarc.

9.2. Implementación MATLAB Simulink & Quarc

Quanser es una empresa canadiense que suministra software y hardware diseñados principalmente para el ámbito académico y de investigación en áreas como Control Automático, Robótica y Mecatrónica.

Ofrece una variedad de productos, incluidos plantas, amplificadores y el software *Quarc*, el cual está diseñado para combinar el modelado matemático de *Simulink* con aplicaciones prácticas. Esto permite probar diversos problemas de control, realizar mediciones para obtener datos en tiempo real, aprovechando el entorno gráfico y la alta capacidad de procesamiento proporcionada por *MATLAB* y *Simulink*.

Uno de los objetivos de esta memoria es compilar e implementar el Sistema de Control Predictivo en la planta RFL real, utilizando los bloques y conexiones del software y hardware *Quarc* de Quanser.

A continuación, se detallarán el conexionado, los equipos, los cables utilizados, el montaje y los elementos de calibración de la RFL.

9.2.1. Hardware de Quanser

Para lograr la correcta interacción con la RFL real, se mostrará el conexionado utilizado en la siguiente figura:

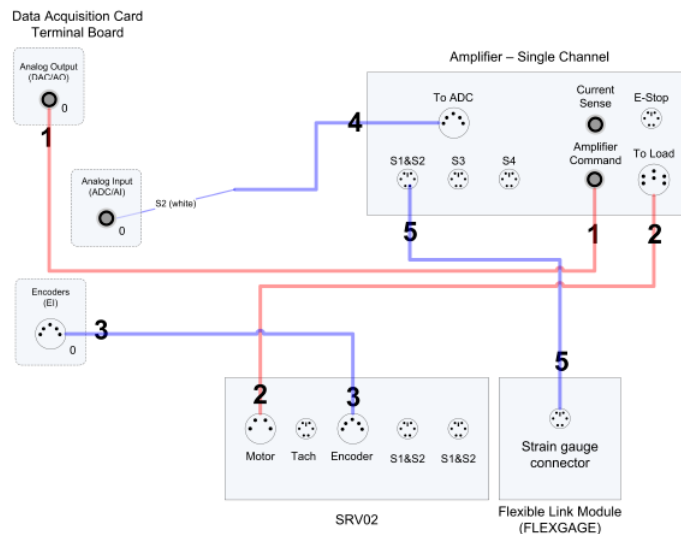


Figure 9.108: Diagrama de conexión del sistema [1]

Teniendo en cuenta el conexionado, se pueden identificar los equipos utilizados para las conexiones presentadas en la Figura 9.108, los cuales se mostrarán a continuación.

El elemento de la Figura 9.109 corresponde al amplificador, cuya función es amplificar señales

de entrada generadas por sensores o señales de referencia de control, y proporcionar una salida amplificada que se utilizará para alimentar actuadores o dispositivos de control.



Figure 9.109: VoltPAQ-X1 Amplifier

En la Figura 9.110 cuya función es adquirir datos de sensores y enviar comandos de control a actuadores en sistemas de control en tiempo real.



Figure 9.110: Q2-USB Data Acquisition Device

En la Figura 9.111 se observan los cables conectados al *motor*, al *encoder* y al *strain gage connector* del módulo de la RFL, los cuales se muestran detalladamente en la Figura 9.108.



Figure 9.111: Conexiones de la Rotary Flexible Link

Se muestran los cables utilizados en la Tabla 9.3, empleados para las conexiones del diagrama de la Figura 9.108. En la tabla se indica el nombre o tipo de cable, una imagen referencial de cada uno y la función que cumple en el conexionado de la planta RFL, para enlazar las plataformas MATLAB-Quanser.

Table 9.3: Cables y Sensores [1]

Tipo de Cable	Imagen	Descripción
2xRCA a 2xRCA		Conecta un canal de salida analógica de la tarjeta de adquisición de datos al amplificador.
4-pin-DIN a 6-pin-DIN		Conecta la salida del amplificador al motor de corriente continua en el servo.
5-pin-stereo-DIN a 5-pin-stereo-DIN		Transporta las señales del codificador entre un conector del codificador y la tarjeta de adquisición de datos (hacia el contador del codificador). Estas señales incluyen alimentación de $\pm 5\text{VDC}$, tierra, canal A y canal B.
6-pin-mini-DIN a 6-pin-mini-DIN		Transporta señales analógicas (desde un joystick, sensor de planta, etc.) al amplificador. Las señales pueden ser monitoreadas y/o utilizadas por un controlador. Además, el cable lleva una línea de $\pm 12\text{VDC}$ desde el amplificador para alimentar un sensor y/o circuitos de acondicionamiento de señales.
5-pin-DIN a 4xRCA		Transporta las señales analógicas sin cambios desde el amplificador hacia las entradas analógicas-digitales en la tarjeta de adquisición.

Se realizaron las conexiones de hardware correspondientes para los equipos y cableados utilizados, junto con la implementación del diagrama de Simulink que emplea los bloques y

conexiones especificadas.

9.2.2. Software de Quanser

Para utilizar de manera adecuada el software Quarc, se deben emplear los bloques mostrados en la Figura 9.112 para confeccionar el diagrama de implementación en tiempo real del controlador con redes neuronales, entrenado con datos grabados del controlador *MPC* en dSPACE.

El bloque *HIL-1 (q2-usb-0)* se encarga de conectar el dispositivo a la tarjeta de adquisición de Quarc, permitiendo el envío y la captura de los datos medidos por los sensores de la RFL.

El bloque *Enable VoltPAQR-X2-X4* habilita la conexión con el amplificador mostrado en la Figura 9.109.

Los bloques *Encoder Calibration* y *Strain Gage Calibration* aplican las ganancias respectivas para el encoder y el sensor de la galga extensiométrica, respectivamente.

El bloque *Actuator Dynamics* aplica las dinámicas del motor DC del SRV02, considerando las ganancias de amplificación, los bloques de saturación y el convenio de la mano derecha para el movimiento rotacional.

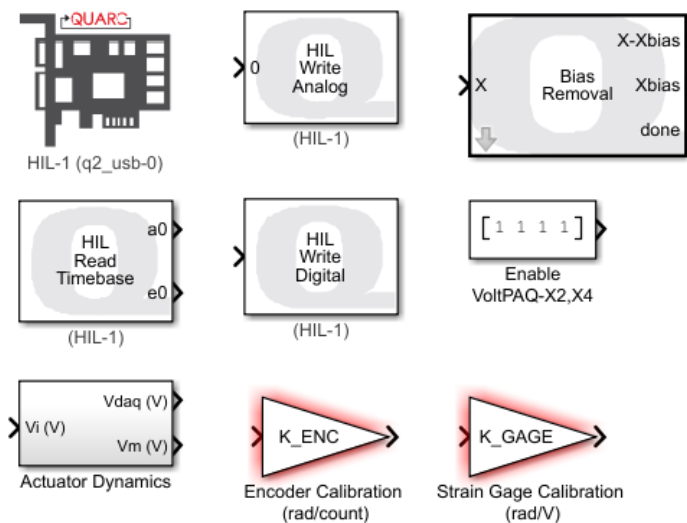


Figure 9.112: Bloques de Quarc Utilizados

- **Bloque Bias Removal:** En la siguiente imagen se muestra la configuración del bloque

Bias Removal, donde se selecciona el tiempo en el cual se considera eliminar el sesgo de la vibración del sensor de la galga extensiométrica, así como el tiempo de muestreo calculado por Quarc.

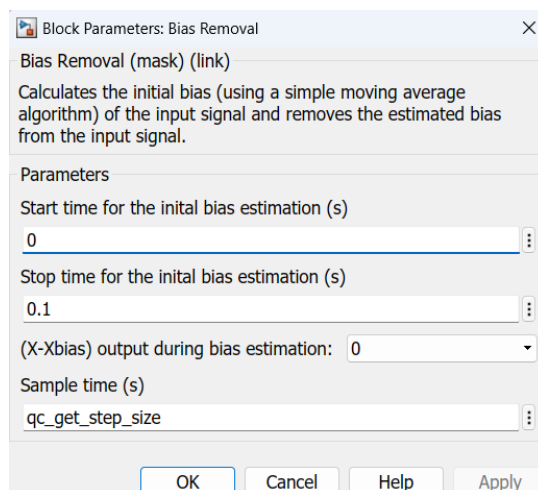


Figure 9.113: Interior bloque Bias Removal

- **Bloque HIL Write Digital:** En la siguiente imagen se muestra la configuración del bloque *HIL Write Digital*, donde se configuran los canales a utilizar para la escritura digital de la información recibida desde la tarjeta de adquisición.

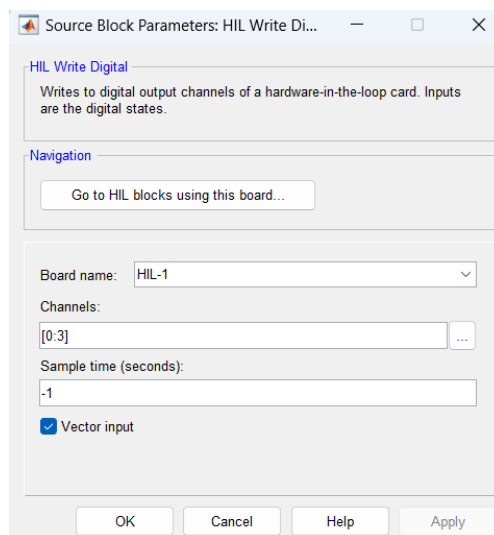


Figure 9.114: Interior bloque HIL Write Digital

- **Bloque HIL Read Time:** En la siguiente imagen se aprecia la configuración del

bloque *HIL Read Time* el cual se encarga de leer en tiempo real los datos captados y escritos que fueron recibidos.

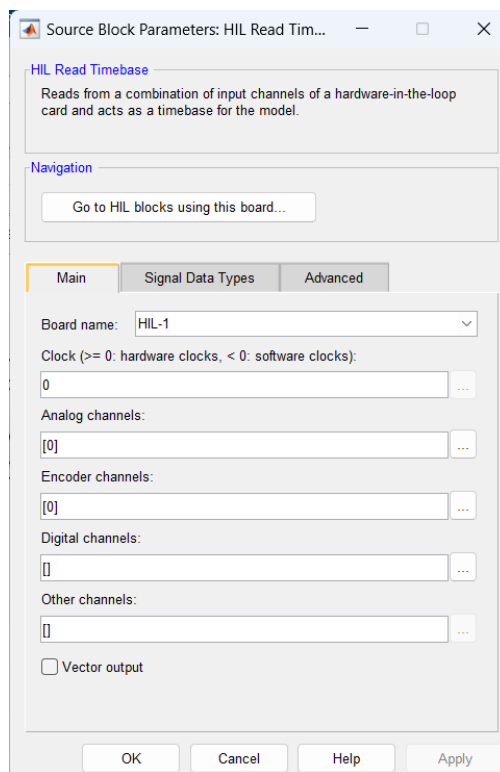


Figure 9.115: Interior bloque HIL Read Time

- **Bloque HIL Initialize:** En la siguiente imagen se aprecia la configuración del bloque *HIL Initialize* que se encarga configurar los canales análogos para el sensor de la Galga Extensiométrica y para el Encoder que capta la medición del ángulo de rotación.

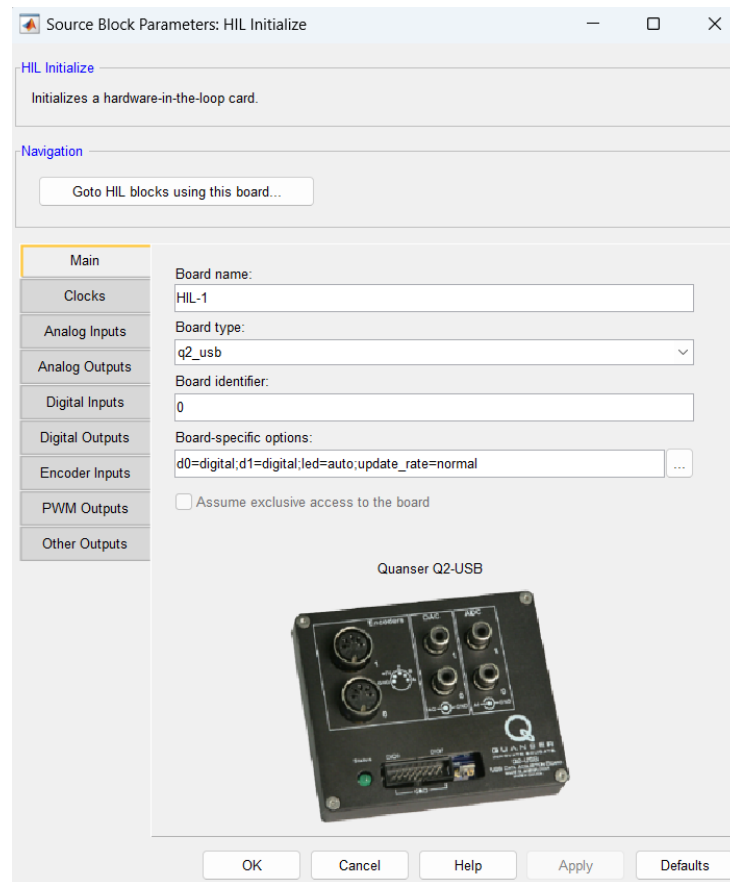


Figure 9.116: Interior bloque HIL Initialize

Se logró compilar el script que contiene el código de MATLAB y el diagrama de Simulink de manera adecuada, sin errores. Sin embargo, no se logró ejercer control correctamente en la planta real con el controlador *MPC*. Por lo tanto, se considerará utilizar la implementación de un controlador con redes neuronales, el cual se logró entrenar y ajustar adecuadamente en la interfaz MATLAB Simulink & MicroLabBox dSPACE.

El diagrama utilizado para controlar la RFL es el siguiente.

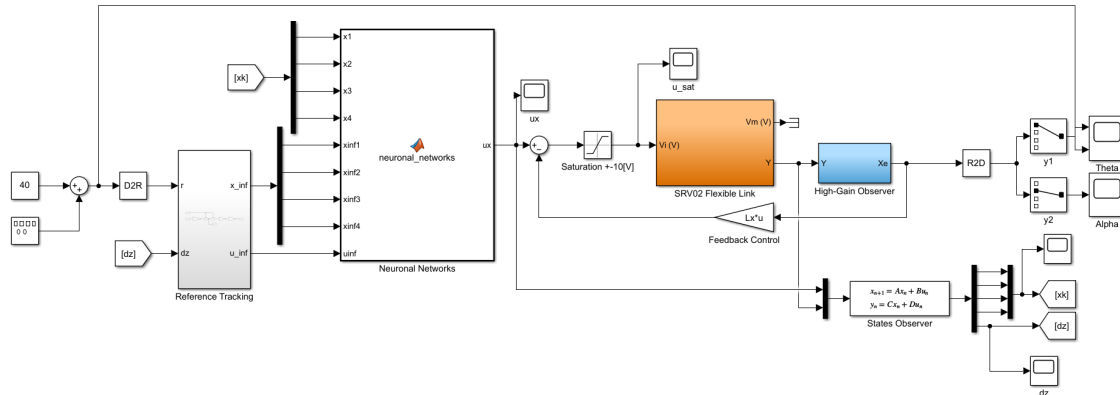


Figure 9.117: Diagrama del controlador con redes neuronales en Quarc

En la Figura 9.118 se observa el interior del bloque *SRV02 Flexible Link* que contiene los bloques de Quarc de la Figura 9.112.

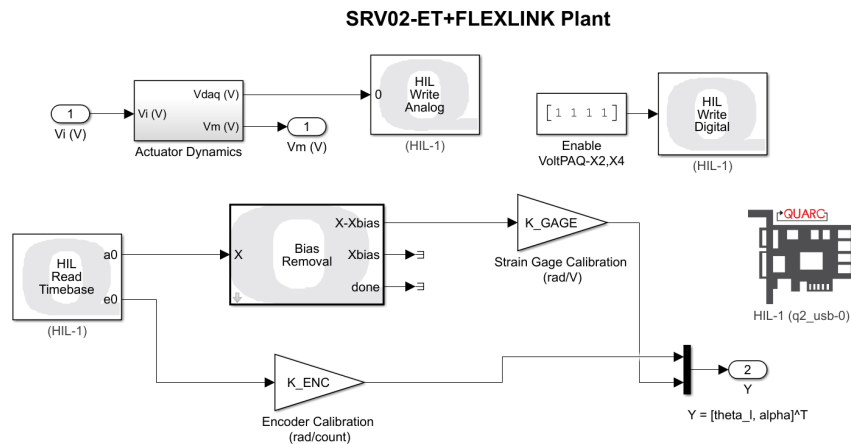


Figure 9.118: Diagrama del controlador con redes neuronales en Quarc

9.2.3. Implementación de Redes Neuronales en Quarc

Realizando las conexiones respectivas para la planta física, se tiene la siguiente configuración. En el vídeo [27], se observa el funcionamiento del controlador basado en redes neuronales para la RFL, demostrando un correcto seguimiento de la señal de referencia y vibraciones mínimas.

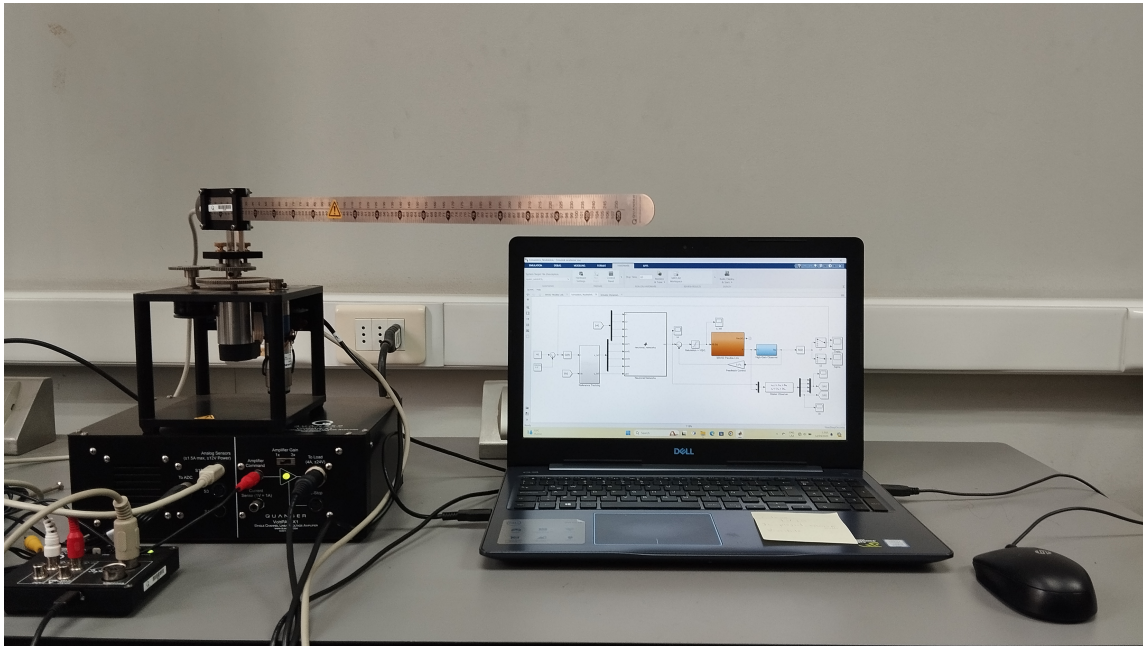


Figure 9.119: Implementación de RFL con software Quarc

Obteniendo los siguientes resultados: En la Figura 9.120 se aprecia que se logra realizar un seguimiento adecuado (en magenta) para la señal de referencia (en negro), que corresponde a una señal cuadrada centrada en 40° con una amplitud de 20° y una frecuencia de $\frac{1}{20}[Hz]$.

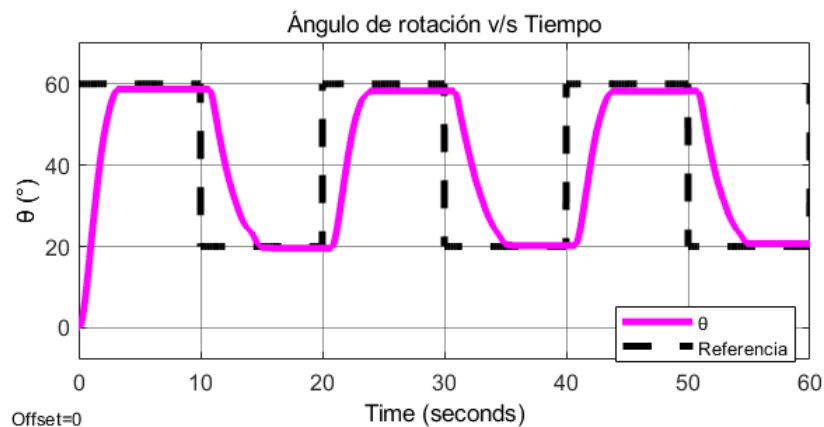


Figure 9.120: Redes Neuronales Quarc: Ángulo de rotación θ y Referencia en el Tiempo

En la Figura 9.121 se observa que el ángulo de flexión no supera los 0.5° de magnitud de vibración, lo que indica un control óptimo para la RFL.

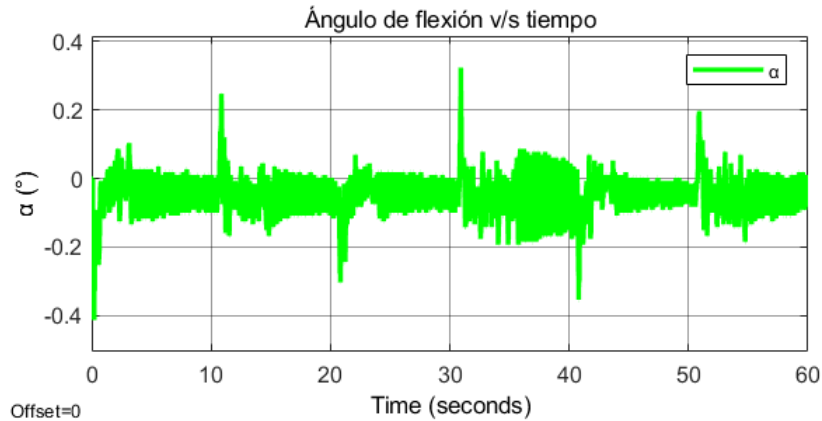


Figure 9.121: Redes Neuronales Quarc: Ángulo de flexión α en el Tiempo

En la Figura 9.122 se observa que la actuación presenta picos de magnitud cuando la señal de referencia cambia, y el valor máximo de esta no supera los $0.5[V]$.

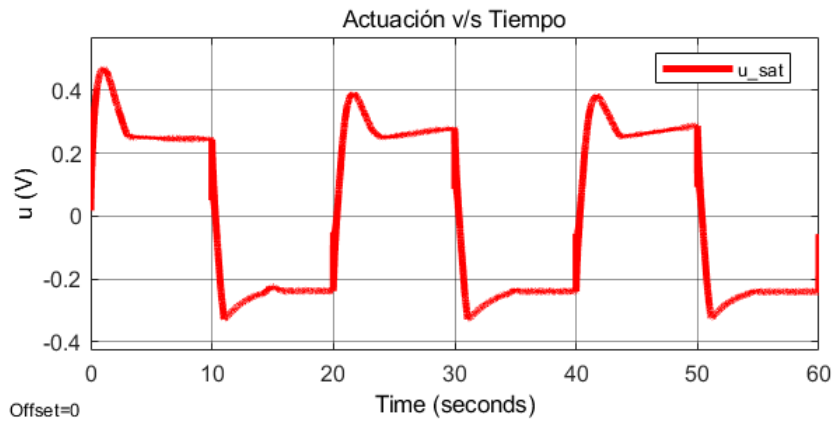


Figure 9.122: Redes Neuronales Quarc: Actuación u_{sat} en el Tiempo

En la Figura 9.123 se observa la actuación entrante a la prealimentación, que presenta la misma forma pero con mayor magnitud. Esto se debe a que el lazo de prealimentación atenúa la actuación que ingresa a la planta RFL.

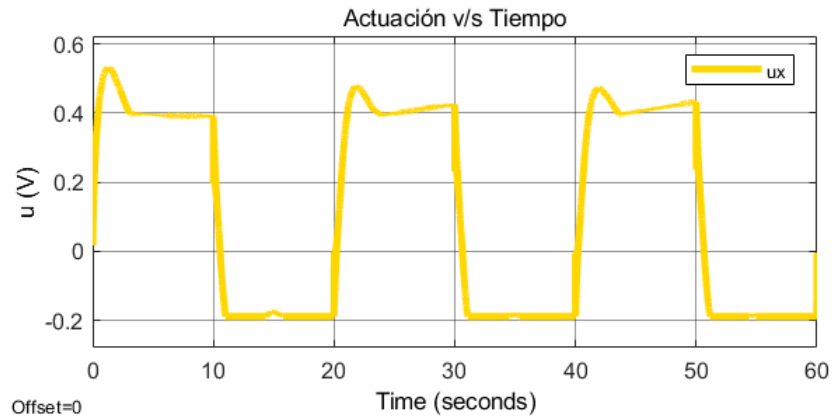


Figure 9.123: Redes Neuronales Quarc: Actuación u_x en el Tiempo

En la Figura 9.124 se observa la estimación de la perturbación, que se ajusta a la actuación entrante hacia la planta RFL.

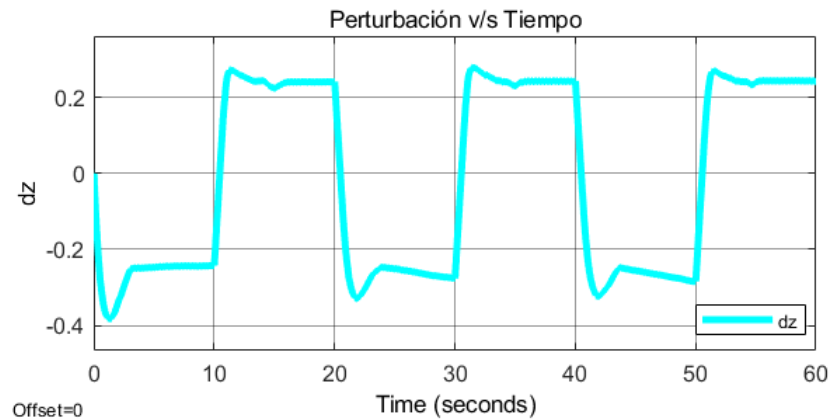


Figure 9.124: Redes Neuronales Quarc: Estimación de la perturbación d_z en el Tiempo

En todos los casos, se observa que se respetan las restricciones consideradas para el controlador *MPC*, cuyos datos fueron utilizados para entrenar la red neuronal con la que se diseñó el controlador implementado en Quarc.

Por lo tanto, se considera que el control realizado para la planta RFL es apropiado y óptimo.

10. Conclusiones y Trabajos Futuros

En este capítulo se presentan las conclusiones del trabajo realizado en la memoria y se proponen recomendaciones para futuros trabajos relacionados con el tema y la planta.

10.1. Conclusión

En este trabajo de titulación se ha abordado la implementación de un control *MPC* para la planta Rotary Flexible Link, cubriendo aspectos teóricos, simulados y prácticos.

Durante las simulaciones, se compararon dos métodos de optimización: *Active-Set* de Quadprog y ADMM. Aunque ADMM mostró mejores resultados en simulación, no pudo aplicarse eficazmente en la planta real, tanto en dSPACE como en QUARC. Esto se debió a la necesidad de ajustar parámetros como la penalización ρ , el número de iteraciones k , y el horizonte de predicción N . Por lo tanto, se optó por implementar el método Quadprog, obteniendo resultados significativamente mejores en la práctica.

El controlador *MPC* se desarrolló con éxito tanto en simulaciones como en la planta real en dSPACE, logrando resultados óptimos. A partir de estos resultados, se entrenó una red neuronal que fue aplicada en dSPACE y en Quarc, cumpliendo adecuadamente con las restricciones establecidas, que era el objetivo principal de esta memoria.

En resumen, se lograron cumplir los objetivos propuestos para este trabajo de titulación, como se evidencia en los vídeos que muestran el funcionamiento de la planta RFL con el control *MPC* y con la red neuronal en Quarc.

10.2. Trabajos Futuros

Para futuros trabajos, se proponen las siguientes acciones:

- Diseñar de manera profesional el cableado utilizado para el Encoder del MicroLabBox dSPACE, asegurando que los cables utilizados encajen correctamente en el laboratorio.
- Ajustar los parámetros del método de optimización ADMM para lograr un control efectivo en la planta real, tanto en dSPACE como en Quarc. En este último, se presentaron



problemas de overflow en los buffers, responsables de enviar y recibir información hacia y desde la tarjeta de adquisición.

- Implementar el sistema de control diseñado en este trabajo en la *Barra Giratoria Flexible*, utilizando el software y hardware MicroLabBox-dSPACE con tiempos de muestreo más cortos y diferentes horizontes de predicción para evaluar su funcionamiento.

11. Anexos

11.1. Controlabilidad para la RFL

```

1  %% Matrices del sistema
A = [0 0 1 0; 0 0 0 1; 0 Ks/Jeq -Beq/Jeq 0; 0 -Ks*(Jeq+Jl)/Jeq/Jl Beq/Jeq 0];
B = [0; 0; 1/Jeq; -1/Jeq];
C = eye(2,4);
D = zeros(2,1);

6  %%Se representa en variables de estado la planta RFL
sys_RFL=ss(A,B,C,D);
%% Controlabilidad
% Construir la matriz de controlabilidad
CNT = ctrb(A, B);

11 % Obtener el rango de la matriz de controlabilidad
rank_Control = rank(CNT);
% Obtener el numero de estados del sistema
num_states = size(A, 1);
% Verificar la controlabilidad

16 if rank_Control == num_states
    disp('El sistema es completamente controlable.');
```

11.2. Observabilidad para la RFL

```

%% Matrices del sistema
A = [0 0 1 0; 0 0 0 1; 0 Ks/Jeq -Beq/Jeq 0; 0 -Ks*(Jeq+Jl)/Jeq/Jl Beq/Jeq 0];
B = [0; 0; 1/Jeq; -1/Jeq];
C = eye(2,4);

5 D = zeros(2,1);

%%Se representa en variables de estado la planta RFL
sys_RFL=ss(A,B,C,D);
%% Observabilidad
% Construir la matriz de observabilidad

10 OBS = obsv(A, C);
% Obtener el rango de la matriz de observabilidad
rank_Observa = rank(OBS);
% Obtener el numero de estados del sistema
num_states = size(A, 1);

15 % Verificar la observabilidad
if rank_Observa == num_states
```

```

disp('El sistema es completamente observable.');
```

else

```

disp('El sistema no es completamente observable.');
```

20 end

11.3. Control LQR sobre la Rotary Flexible Link

```

%%%% LQR ROTARY FLEXIBLE LINK %%%%
%% Parametros de RFL
Beq=0.004; %N*m/(rad/s)
Jeq=2.08e-3; %kg*m^2
5 ml=0.065; %kg
Ll=0.419; %m
Jl=(ml*Ll^2)/3; %kg*m^2
wn=2*pi/0.34; %rad/s
Ks=Jl*wn^2; %N*m/rad
10 %% Parametros del filtro
wcf_1 = 2* pi * 50; %Velocidad angular
wcf_2 = 2* pi * 10; %Velocidad de flexión
%% Matrices del sistema
A= [0 0 1 0;0 0 0 1;0 Ks/Jeq -Beq/Jeq 0;0 -Ks*(Jeq+Jl)/Jeq/Jl Beq/Jeq 0];
15 B= [0; 0; 1/Jeq; -1/Jeq];
C= eye(2,4);
D= zeros(2,1);
x0= [deg2rad(45);deg2rad(4);deg2rad(0);deg2rad(0)];
%Representación en variables de estado
20 sys_RFL=ss(A,B,C,D);
%% Controlabilidad
CNT = ctrb(A, B); %Matriz de controlabilidad
rank_Control = rank(CNT); %Rango
num_states = size(A, 1); %Estados del sistema
25 % Verificar la controlabilidad
if rank_Control == num_states
disp('El sistema es completamente controlable.');
```

else

```

disp('El sistema no es completamente controlable.');
```

30 end

```

%% Observabilidad
OBS = obsv(A, C); %Matriz de observabilidad
rank_Observa = rank(OBS); %Rango
% Verificar la observabilidad
35 if rank_Observa == num_states
disp('El sistema es completamente observable.');
```

```

else
    disp('El sistema no es completamente observable.');
```

end

40 %% Estabilidad del sistema

% Se calculan los polos de lazo abierto del sistema, de la matriz A

P_RFL=eig(A);

%% Implementacion de LQR

Q=diag([125,1,1,10]);

45 R=1;

[K,S,Polos]=lqr(A,B,Q,R);

11.4. Control MPC sobre la Rotary Flexible Link

11.4.1. MPC para Estado Cero

```

%%%% MPC para Estado Cero %%%
clear all
close all

4 %% Filtros discretos
% Frecuencias de los filtros para las salidas y3 e y4
wcf_1= 2*pi*50;
wcf_2= 2*pi*10;
T= 0.002; %Tiempo de muestreo

9 tf1 = tf(1,[1 wcf_1]);
tf2 = tf(1,[1 wcf_2]);
tz1 = c2d(tf1,T);
[num_tz1, den_tz1] = tfdata(tz1);
numtz1 = cell2mat(num_tz1);
14 dentz1 = cell2mat(den_tz1);
tz2 = c2d(tf2,T);
[num_tz2, den_tz2] = tfdata(tz2);
numtz2 = cell2mat(num_tz1);
dentz2 = cell2mat(den_tz2);

19 %% Sistema en variables de estado
Ac = [0 0 1 0;0 0 0 1; 0 623.7468 -7.1971 0; 0 -965.3452 7.1971 0];
Bc = [0;0;479.8052;-479.8052];
C= [1 0 0 0;0 1 0 0];
D= [0;0];

24 sys_RFL= ss(Ac,Bc,C,D);
x0=[deg2rad(60);deg2rad(5);deg2rad(60);deg2rad(-3)];

%% Variables de estado en tiempo discreto
tz_RFL= c2d(sys_RFL,T); %Discretización
[Ad,Bd,C,D]= ssdata(tz_RFL); %Matrices discretizadas
```

```

29 %% Pre alimentación
Qx = diag([1 1 1 1]);
Rx = 1;
[Lx,Px,pol] = dlqr(Ad,Bd,Qx,Rx);
A = Ad;
34 B = Bd;
Ax=A-B*Lx;
%% MPC Constantes y Restricciones
N= 5; %Horizonte de prediccion
x_min= [deg2rad(-360);deg2rad(-10);deg2rad(-1000);deg2rad(-360)];
39 x_max= [deg2rad(360);deg2rad(10);deg2rad(1000);deg2rad(360)];
u_min= -10;
u_max= 10;
x_Nmin= x_min;
x_Nmax= x_max;
44 Omega= diag([1 1 1 1]); %Matriz de peso de estados
Gama= 1; %Matriz de peso de actuación
[Linf,OmegaN,~]= dlqr(Ax,B,Omega,Gama);
n= size(A,1); %n=4 para A de 2x2 %Dimensiones del sistema
m= size(B,2); %m=1 para B de 2x2
49 p= size(C,1); %p=2 para C de 2x2
J= [eye(n);-eye(n);-Lx;Lx]; %Construcción de matrices
Jn= [eye(n);-eye(n)];
E= [zeros(n,m);zeros(n,m);eye(m);eye(m)];
H= blkdiag(kron(eye(N),blkdiag(Omega,Gama)),OmegaN); %Matriz funcion de costo
54 H= (H+Transpose(H))/2;
h = zeros(size(H,1),1);
In= eye(n);
f= [Ax,B,-In];
F= [];
59 col_F= (N+1)*n+N*m;
for i= 1:1:N
    if i== 1
        F= [-In,zeros(n,col_F-n)];
    end
64 M= [zeros(n,(i-1)*(n+m)),f,zeros(n,col_F-((i-1)*(n+m)+(2*n+m)))];
F= [F;M];
end
ncero= size(F,1)-n; %Cantidad de ceros de f
zer= zeros(ncero,1); %Ceros para f[-x0,zer]
69 G = blkdiag(kron(eye(N),[J,E]),Jn); %Matriz de desigualdad
dx= [x_max;-x_min];
du= [u_max;-u_min];
dN= [x_Nmax;-x_Nmin];

```

```
g= [kron(ones(N,1),[dx;du]);dN];
```

11.4.2. MPC Seguimiento a Referencia

```

%%%% MPC para Seguimiento a Referencia %%%
2 clear all
close all

%% Filtros discretos
% Frecuencias de los filtros para las salidas y3 e y4
wcf_1= 2*pi*50;
7 wcf_2= 2*pi*10;
T= 0.002; %Tiempo de muestreo
tf1 = tf(1,[1 wcf_1]);
tf2 = tf(1,[1 wcf_2]);
tz1 = c2d(tf1,T);
12 [num_tz1, den_tz1] = tfdata(tz1);
numtz1 = cell2mat(num_tz1);
dentz1 = cell2mat(den_tz1);
tz2 = c2d(tf2,T);
[num_tz2, den_tz2] = tfdata(tz2);
17 numtz2 = cell2mat(num_tz1);
dentz2 = cell2mat(den_tz2);

%% Sistema en variables de estado
Ac = [0 0 1 0;0 0 0 1; 0 623.7468 -7.1971 0; 0 -965.3452 7.1971 0];
Bc = [0;0;479.8052;-479.8052];
22 C= [1 0 0 0;0 1 0 0];
D= [0;0];
sys_RFL= ss(Ac,Bc,C,D);
x0=[deg2rad(60);deg2rad(5);deg2rad(60);deg2rad(-3)];

%% Variables de estado en tiempo discreto
27 tz_RFL= c2d(sys_RFL,T); %Discretización
[Ad,Bd,C,D] = ssdata(tz_RFL); %Matrices discretizadas

%% Pre alimentación
Qx = diag([1 1 1 1]);
Rx = 1;
32 [Lx,Px,pol] = dlqr(Ad,Bd,Qx,Rx);
A = Ad;
B = Bd;
Ax=A-B*Lx;

%% MPC Constantes y Restricciones
37 N= 5; %Horizonte de prediccion
x_min= [deg2rad(-360);deg2rad(-10);deg2rad(-1000);deg2rad(-360)];
x_max= [deg2rad(360);deg2rad(10);deg2rad(1000);deg2rad(360)];

```

```

u_min= -10;
u_max= 10;
42 x_Nmin= x_min;
x_Nmax= x_max;
Omega= diag([1 1 1 1]); %Matriz de peso de estados
Gama= 1; %Matriz de peso de actuación
[Linf,OmegaN,~]= dlqr(Ax,B,Omega,Gama);
47 n= size(A,1); %n=4 para A de 2x2 %Dimensiones del sistema
m= size(B,2); %m=1 para B de 2x2
p= size(C,1); %p=2 para C de 2x2
J= [eye(n);-eye(n);-Lx;Lx]; %Construcción de matrices
Jn= [eye(n);-eye(n)];
52 E= [zeros(n,m);zeros(n,m);eye(m);eye(m)];
H= blkdiag(kron(eye(N),blkdiag(Omega,Gama)),OmegaN); %Matrices funcion de costo
H= (H+Transpose(H))/2;
h = zeros(size(H,1),1);
In= eye(n); %Matriz de restricción de igualdad
57 f= [Ax,B,-In];
F= [];
col_F= (N+1)*n+N*m;
for i= 1:1:N
    if i== 1
62 F= [-In,zeros(n,col_F-n)];
    end
    M= [zeros(n,(i-1)*(n+m)),f,zeros(n,col_F-((i-1)*(n+m)+(2*n+m)))];
    F= [F;M];
end
67 ncero= size(F,1)-n; %Cantidad de ceros de f
zer= zeros(ncero,1); %Ceros para f[-x0,zer]
G = blkdiag(kron(eye(N),[J,E]),Jn); %Matriz de desigualdad
%% Seguimiento a referencia
i1 = eye(4)-Ax;
72 inv1 = inv(i1);
i2 = C(1,:)*inv1*B;
inv2 = inv(i2);

```

11.4.3. Código Observador de Estados

```

1 % Matrices ampliadas
Az = [Ax B; zeros(size(B,2),size(Ax,1)) eye(size(B,2))];
Bz = [B; zeros(size(B,2))];
Cz = [C zeros(size(C,1),size(B,2))];
Q_obs = diag([1 1 1 1 1]);

```

```

6  R_obs = 1;
   % Matrices del observador
   [L,Pz,polz] = dlqr(Transpose(Az),Transpose(Cz),Q_obs,R_obs);
   Lz = Transpose(L);
   Aobs = Az - Lz*Cz;
11  Bobs = [Bz Lz];
   Cobs = eye(5);
   Dobs = zeros(5,3);

```

11.4.4. Código Bloque *mat_f* Estado Cero

```

function f = mat_f(zeros_f,xk)
2  f = [-xk;zeros_f];
   end

```

11.4.5. Código Bloque *mat_f* y *mat_g* Seguimiento a Referencia

```

function f = mat_f(zeros_f,x_inf,xk)
2  xk = xk-x_inf;
   f = [-xk;zeros_f];
   end

function g = mat_g(x_inf,x_Nmax,x_Nmin,x_max,x_min,u_inf,u_max,u_min,L_pre)
7  vectorx = [x_max-x_inf;-x_min+x_inf];
   vectoru = [u_max+ L_pre-u_inf;-u_min-L_pre+u_inf];
   vectorxN = [x_Nmax-x_inf;-x_Nmin+x_inf];
   g=[kron(ones(5,1),[vectorx;vectoru]);vectorxN];
   end

```

11.4.6. Código Quadprog

```

function ux = quadprog(H,h,G,g,F,f)
   options = optimoptions('quadprog','Algorithm','active-set');
   x = quadprog(H,h,G,g,F,f,[],[],zeros(29,1),options);
4  ux=x(5);
   end

```

11.4.7. Código ADMM

```

function [ux] = admm(H,h,G,g,F,f)
%=====

```

```
% Quadratic programing (QP) problem solution:
%
%           min 0.5 x'*H*x + h'*x
5 %           s.t      F*x  = f
%
%           G*x  <=  g
%
%  x       : Optimal solution of qp problem
%  val     : Value of the cost function in tk optimal
%=====
10  n=size(H,1);
    q=size(G,1);
    p=size(F,1);

    x=1*ones(n,1);
15  z=0.5*ones(q,1);
    tau=0.5*ones(q+p,1);

    A=[G; F];
    B=[eye(q);zeros(p,q)];
20  c=[g;f];
    ep=1e-9;  ed=1e-9;

    rho=2;%%Se tiene que escoger este valor
    rk=A*x+B*z-c;
25  sk=rho*Transpose(A)*B*(z); k=0;
    while norm(rk)>=ep || norm(sk)>=ed
        k=k+1;
        x=-(H+rho*Transpose(A)*A)\(h+rho*Transpose(A)*(B*z+tau-c));
        z=max(0,-G*x-tau(1:q)+g);
30  tau=tau+A*x+B*z-c;
        if k==500; break; end %%Se tiene que escoger este valor
    end
    ux = x(5);
end
```

11.5. Control con Redes Neuronales sobre la Rotary Flexible Link

11.5.1. Código de Entrenamiento de Redes Neuronales

```
1 import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
```

```
array_data = np.loadtxt('train_rflmpc.csv', skiprows=1, delimiter=',')

6
lista_input = list()
lista_output = list()
lista_tiempo = list()
for i in array_data:
11     lista_input.append(i[0:9])
        lista_output.append(i[9:10])
        lista_tiempo.append(i[10:11])

input_nn = np.array(lista_input, dtype = float)
16 output_nn = np.array(lista_output, dtype = float)
    tiempo_nn = np.array(lista_tiempo, dtype = float)

oculta1 = tf.keras.layers.Dense(units=2, activation="relu", input_shape=[9])
salida = tf.keras.layers.Dense(units=1, activation="linear")
21 modelo = tf.keras.Sequential([oculta1,salida])

modelo.compile(optimizer=tf.keras.optimizers.Adam(0.1), loss='mean_squared_error')

print("Comenzando entrenamiento...")
26 historial = modelo.fit(input_nn, output_nn, epochs=1000, verbose=False)
print("Modelo entrenado")
Comenzando entrenamiento...
Modelo entrenado
```

11.5.2. Código Bloque Neural Network

```
function ux = fcn(x1,x2,x3,x4,xinf1,xinf2,xinf3,xinf4,uinf)
b = [0.42553216;-0.6683681];
w1 = [-0.30759007;-1.2120334];
w2 = [5.261527;-0.22845627];
5 w3 = [-0.32257125;-0.45505166];
w4 = [0.1141999;-0.01958555];
w5 = [0.3080502;-1.1928272];
w6 = [-0.05670816;-0.14673078];
w7 = [0.6181058;-0.03464818];
10 w8 = [-0.25887203;-0.6465112];
w9 = [2.2610555;-0.15725341];

W = [b w1 w2 w3 w4 w5 w6 w7 w8 w9];
X = [1;x1;x2;x3;x4;xinf1;xinf2;xinf3;xinf4;uinf];
15 S = W*X;
```

```
R1 = max(0,S);  
R = [1;R1];  
  
c = -0.18857954;  
20 z1 = 0.44218156;  
z2 = 0.6941998;  
  
Z = [c z1 z2];  
  
25 uxã=ãZ*R;  
end
```

11.5.3. Código Redes Neuronales

```
import numpy as np  
import tensorflow as tf  
import matplotlib.pyplot as plt  
4  
  
plt.xlabel("# Epoca")  
plt.ylabel("Magnitud de pérdida")  
plt.plot(historial.history["loss"])  
  
9 print("Datos de la primera capa oculta:", oculta1.get_weights())  
print("Datos de la capa de salida:", salida.get_weights())  
  
out_predict = modelo.predict(input_nn)  
  
14 from statistics import mean  
err_por = abs(output_nn - out_predict)/output_nn*100  
err_prom = sum(err_por)/len(err_por)  
err_prom1 = np.mean(err_por)  
  
19 print("error promedio:", err_prom1,"%")  
print("desviación estándar:", np.std(err_por))  
  
plt.plot(tiempo_nn, out_predict)  
plt.plot(tiempo_nn, output_nn)  
  
24 array_data_verif = np.loadtxt('test_rflmpc.csv', skiprows=1, delimiter=',')  
  
lista_input_verif = list()  
lista_output_verif = list()  
29 lista_time_verif = list()
```

```
for i in array_data_verif:
    lista_input_verif.append(i[0:9])
    lista_output_verif.append(i[9:10])
    lista_time_verif.append(i[10:11])

34
input_nn_verif = np.array(lista_input_verif, dtype = float)
output_nn_verif = np.array(lista_output_verif, dtype = float)
time_nn_verif = np.array(lista_time_verif)
out_predict_verif = modelo.predict(input_nn_verif)
39 plt.plot(time_nn_verif, out_predict_verif)
plt.plot(time_nn_verif, output_nn_verif)

err_por_verif = abs(output_nn_verif - out_predict_verif)/output_nn_verif*100
err_prom1_verif = np.mean(err_por_verif)

44
print("error promedio:", err_prom1_verif,"%")
print("desviación estándar:", np.std(err_por_verif))
```

References

- [1] Quanser. (2012) User manual flexible link experiment. 02-11-2023. [Online]. Available: <https://www.made-for-science.com/de/quanser/?df=made-for-science-quanser-rotary-flexible-link-usermanual.pdf>
- [2] ——. (2011) Quanser innovate educate. 01-11-2023. [Online]. Available: <https://www.quanser.com/products/rotary-flexible-link/>
- [3] J. A. D. D. Graham C. Goodwin, María M. Seron, *Constrained Control and Estimation*. Springer, 2005.
- [4] A. L. Cedeño, “Control predictivo por modelo,” Universidad Técnica Federico Santa María, Tech. Rep., 2020, documento educativo o técnico.
- [5] R. Tibshirani. (2009) Alternating directions method of multipliers. [Online]. Available: <https://www.stat.cmu.edu/~ryantibs/convexopt/lectures/admm.pdf>
- [6] E. Wong, “Active-set method for quadratic programming,” Ph.D. dissertation, University of California, San Diego, 2011.
- [7] dSPACE. (2020) 01-04-2024. [Online]. Available: https://www.dspace.com/en/ltd/home/medien/product_info/microlabbox_product-brochure.cfm
- [8] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT press, 2016, vol. 1.
- [9] Quanser. (2012) 01-11-2023. [Online]. Available: <https://www.quanser.com/products/quarc-real-time-control-software/>
- [10] ——. (2012) student workbook flexible link experiment for labview users. 30-10-2023. [Online]. Available: <https://www.made-for-science.com/de/quanser/?df=made-for-science-quanser-rotary-flexible-link-coursewarestud-labview.pdf>
- [11] L. web de física. (2003) Ecuaciones de lagrange. 11-11-2023. [Online]. Available: <https://www.lawebdefisica.com/dicc/lagrange/>
- [12] M. Ázua, “Implementación de un sistema de control predictivo usando el toolbox quarc,” Universidad Técnica Federico Santa María, Valparaíso, Tesis de Pregrado, 2022.
- [13] Quanser. (2009) User manual srv02 rotary servo base unit. 02-11-2023. [Online]. Available: <https://nps.edu/documents/105873337/0/SRV02+Base+Unit+User+Manual.pdf/ce50a48e-d169-4652-a9df-ae979af10094?t=1441916178000>
- [14] M. Salgado, J. Yuz, and R. Rojas, *Análisis de Sistemas Lineales*. Pearson Educación, 2005, vol. 1.
- [15] K. Ogata, *Modern Control Engineering*. Prentice Hall, 2009, vol. 5.
- [16] B. W. Karl J. Astrom, “Computer-controlled systems theory and design,” in *Proceedings of a conference*, vol. 3, 1996, pp. 30–66.

- [17] MathWorks. (2006) lqr. 08-11-2023. [Online]. Available: <https://la.mathworks.com/help/control/ref/lti.lqr.html>
- [18] M. A. Nielsen, *Neural Networks and Deep Learning*. Determination Press, 2015.
- [19] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
- [20] E. C. B. P. J. E. Stephen Boyd, Neal Parikh, *Distributed Optimization and Statistical Learning via The Alternating Direction Method of Multipliers*. Foundations and Trends in Machine Learning, 2010, pp. 7–20.
- [21] L. V. Stephen Boyd, *Convex Optimization*. Cambridge University Press, 2004, pp. 10 – 725.
- [22] A. B. Francesco Borrelli and M. Morari, *Predictive Control for Linear and Hybrid Systems*. Cambridge: Cambridge University Press, 2017.
- [23] A. Parra Antillanca, “Implementación de la técnica de control predictivo utilizando admm para un sistema de estanques acoplados,” Universidad Técnica Federico Santa María, Valparaíso, Tesis de Pregrado, 2023.
- [24] R. J. Mantz, “Observadores de estados,” Ph.D. dissertation, Universidad Nacional de la Plata, Facultad de Ingeniería, Departamento de Electrotecnia, 2003.
- [25] S. C. Segovia. (2024) Rotary flexible link controlada por mpc en dspace. 15-04-2024. [Online]. Available: <https://youtu.be/tbiag8fisws>
- [26] ——. (2024) Rotary flexible link controlada con redes neuronales en dspace. 19-04-2024. [Online]. Available: <https://youtu.be/hzI9GEVaL58>
- [27] ——. (2024) Rotary flexible link controlada con redes neuronales en quarc. 12-04-2024. [Online]. Available: https://youtu.be/_YAg0bCi-4c