Trabajo Fin de Grado
en Ingeniería Electrónica, Robótica y Mecatrónica

Control inteligente de grúas industriales: aplicación de lógica borrosa en la automatización de tareas de carga

Autor: Pablo Santervás Blanco

Tutor: Juan Manuel Escaño González

Dpto. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2024

Trabajo Fin de Grado
en Ingeniería Electrónica, Robótica y Mecatrónica

Control inteligente de grúas industriales: aplicación de lógica borrosa en la automatización de tareas de carga

Autor:
Pablo Santervás Blanco

Tutor:
Juan Manuel Escaño González
Profesor titular

Dpto. de Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2024

Trabajo Fin de Grado: Control inteligente de grúas industriales: aplicación de lógica borrosa en la automatización de tareas de carga

Autor: Pablo Santervás Blanco

Tutor: Juan Manuel Escaño González

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2024

El Secretario del Tribunal

Agradecimientos

Quiero expresar mi más profundo agradecimiento a mi familia, que ha estado siempre a mi lado a lo largo de este camino, brindándome apoyo, confianza y palabras de aliento, especialmente en los momentos en los que las cosas no salían como esperaba.

También agradezco a todos los compañeros que he tenido la suerte de conocer durante estos años. Han sido personas extraordinarias que no solo me han acompañado, sino que también me han ofrecido su ayuda, tanto en lo personal como en lo académico, siempre con generosidad y disposición. Del mismo modo, quiero recordar a mis viejos amigos, quienes, a pesar del paso del tiempo, han continuado apoyándome y animándome en cada etapa de este proceso.

Por último, mi gratitud se extiende a todos los profesores que han sido parte de mi formación, por compartir sus conocimientos y guiarme durante esta etapa. De manera especial, quiero destacar a mi tutor, un gran profesional que me ha orientado con paciencia y creatividad, aportándome ideas clave cuando enfrentaba dificultades en este proyecto.

Pablo Santervás Blanco

Sevilla, 2024

Resumen

El presente trabajo aborda el desarrollo de un sistema automático para la carga de materiales pesados, con el objetivo de reducir los riesgos para los operarios y aumentar la productividad en el entorno industrial. Inicialmente, el manejo manual de las cargas implicaba altos riesgos de accidentes y limitaciones en la eficiencia operativa. Como solución, se planteó la automatización del proceso mediante una grúa controlada por un PLC, una opción que destaca por su capacidad para cumplir con los requisitos de seguridad y eficiencia, además de ser una tecnología ampliamente utilizada y probada en aplicaciones industriales.

El sistema de control implementado se basa en la lógica borrosa, un enfoque reconocido por su capacidad para gestionar sistemas complejos con incertidumbre. La elección de esta metodología se fundamentó en la revisión de literatura científica que aborda problemáticas similares, aportando así una base sólida para la propuesta planteada.

Para evaluar el desempeño del sistema, se realizaron comparaciones con el método clásico de control basado en controladores PID. Los resultados evidenciaron la superioridad del controlador borroso, destacando no solo por ofrecer una mejor respuesta dinámica, sino también por su mayor adaptabilidad a cambios en las condiciones de operación, como el peso de la carga. Estas evaluaciones se llevaron a cabo en el entorno de simulación Simulink (MATLAB), empleando tanto los controladores de la herramienta Fuzzy Logic Toolbox como el modelo de controlador TM221 de Schneider Electric, disponible en el entorno Machine Expert Basic.

Cabe destacar que el desarrollo de este trabajo se basó en un caso real dentro de una fábrica donde se manipulan lotes de chapas de diferentes formatos y pesos. Sin embargo, la solución planteada puede ser aplicada a cualquier tipo de carga, no solo al caso particular en el que se basa este proyecto.

En conclusión, este proyecto demuestra la eficacia de soluciones tecnológicas consolidadas en la industria, contribuyendo a mejorar tanto la seguridad como la productividad en el manejo de materiales.

Abstract

This project focuses on the development of an automated system for handling heavy materials, aiming to reduce risks for operators and increase productivity in industrial environments. Initially, the manual handling of loads posed high accident risks and operational efficiency limitations. To address this issue, the automation of the process through a crane controlled by a PLC was proposed. This solution stands out for its ability to meet safety and efficiency requirements, as well as being a widely used and proven technology in industrial applications.

The implemented control system is based on fuzzy logic, a well-recognized approach for managing complex systems with uncertainty. The choice of this methodology was supported by a review of scientific literature addressing similar problems, providing a solid foundation for the proposed solution.

To evaluate the system's performance, comparisons were made with the classical control method using PID controllers. The results demonstrated the superiority of the fuzzy controller, excelling not only in delivering better dynamic response but also in greater adaptability to changes in operating conditions, such as load weight. These evaluations were conducted in the Simulink (MATLAB) simulation environment, using both the controllers provided by the Fuzzy Logic Toolbox and the TM221 controller model from Schneider Electric, available for simulation in the Machine Expert Basic platform.

It is important to note that this project was developed based on a real-world case in a factory that handles batches of metal sheets of varying sizes and weights. Nevertheless, the proposed solution can be applied to any type of load, not just the specific case on which this project is based.

In conclusion, this project confirms the effectiveness of well-established technological solutions in the industry, contributing to improved safety and productivity in material handling.

Índice

Agradecimientos **vii**

Resumen **ix**

Abstract **xi**

Índice **xii**

Índice de Tablas **xv**

Índice de Figuras **xvii**

Notación **xx**

1 Introducción **1**

2 Viabilidad Técnica **4**

2.1 Puente grúa automático *4*

2.1.1 Funcionamiento 4

2.1.2 Ventajas 5

2.1.3 Inconvenientes 5

2.2 Manipulador *5*

2.2.1 Funcionamiento 5

2.2.2 Ventajas 5

2.2.3 Desventajas 5

2.3 Automatic Guided Vehicle *5*

2.3.1 Funcionamiento 5

2.3.2 Ventajas 6

2.3.3 Desventajas 6

2.4 Conclusiones *6*

2.5 Solución planteada en planta *6*

3 Simulación del sistema **9**

3.1 Factory i/o *9*

3.1.1 Introducción 9

3.1.2 Bloques usados 11

3.1.3 Modelo de planta 14

3.1.4 Simulaciones 14

3.2 Machine Expert Basic *16*

3.2.1 Introducción 16

3.2.2 Programación y conexión con factory io 17

3.2.3 Programa 19

3.3 Conclusión y resumen *23*

4 Lógica Borrosa **26**

4.1 Estado del arte *26*

4.2 Marco teórico *27*

4.2.1 Conjuntos borrosos 27

4.2.2 Variables lingüísticas 27

4.2.3 Funciones de membresía 27

4.2.4 Reglas borrosas 28
4.3 Controlador borroso *30*
4.3.1 Fuzzificación 30
4.3.2 Base de reglas 30
4.3.3 Mecanismos de inferencia 31
4.3.4 Defuzzificación 31
4.4 Controlador borroso en MATLAB *33*
4.4.1 Creación del archive*.fis 33
4.4.2 I/O y funciones de membresía 34
4.4.3 Reglas de inferencia 35
4.4.4 Visualizador de reglas 35
4.4.5 Implementación del controlador en Simulink 37

5 Análisis del modelo e implementación **39**
5.1 Esquema del modelo *39*
5.2 Modelo dinámico en el dominio del tiempo *40*
5.2.1 Ecuaciones de Euler-Lagrange 40
5.2.2 Ecuaciones del sistema 41
5.3 Espacio de estados *42*
5.4 Modelo de Simulink *43*
5.5 Conclusión *44*

6 Control de la carga **46**
6.1 Implementación de controlador PID-PI en cascada *46*
6.1.1 Obtención de las funciones de transferencia 46
6.1.2 Obtención del control en cascada 48
6.2 FLC basado en PID *51*
6.2.1 Diseño de un FLC basado en un PID 51
6.2.2 FLC del sistema 56
6.2.3 Conclusión 57
6.3 FLC basado en conocimiento de expertos *57*
6.3.1 Inputs y output 57
6.3.2 Base de reglas 59
6.3.3 Comparativa entre métodos de control 61
6.4 Control frente al cambio de carga *62*
6.5 FLC en autómata *63*
6.5.1 Comunicación entre MATLAB y Machine Expert 63
6.5.2 Modelo empleado 65
6.5.3 Programación de FLC en autómata 66

7 Conclusiones **71**

Referencias **73**

ÍNDICE DE TABLAS

Tabla 3-1. Campos de selección de equipos de Factory i/o 10
Tabla 3-2. Asignación de i/o digitales y analógicos en Machine Expert 18
Tabla 3-3. Correspondecia entre input y outputs 19
Tabla 4-1. Operaciones lógicas del controlador 29
Tabla 4-2. Ecuaciones del centroide 32
Tabla 5-1. Coordenadas de ecuaciones E-L 41
Tabla 6-1. Valores de las ganancias mediante *pidtune* 49
Tabla 6-2. Ganancias del control en cascada 50
Tabla 6-3. MFs de FLC basado en PID 52
Tabla 6-4. Base de reglas de un FLC basado en PID 54
Tabla 6-5. MFs de las entradas y la salida 57
Tabla 6-6. Base de reglas heurísticas 59
Tabla 6-7. Reglas del ángulo θ 59
Tabla 6-8. Funciones usadas de "Industrial Comunication" Toolbox [18] 64
Tabla 6-9. Manejo de read y write 64

ÍNDICE DE FIGURAS

Figura 1-1. Tasa (%) de accidentes laborales en España a lo largo de la década (2009-18) [1] 1
Figura 2-1. Layout de la situación de origen 4
Figura 2-2. Layout de solución planteada 7
Figura 3-1. Menú de selección de escena de Factory i/o 9
Figura 3-2. Estados de funcionamiento de sensores y actuadores 10
Figura 3-3. Emitter con pallet y material 11
Figura 3-4. Remover 11
Figura 3-5. Pick & Place 11
Figura 3-6. Safety door 12
Figura 3-7. Safeguard 12
Figura 3-8. Diffuser sensor 12
Figura 3-9. Belt conveyor 12
Figura 3-10. Warning Light 13
Figura 3-11. Stack Light 13
Figura 3-12. Alarm Siren 13
Figura 3-13. Panel de control 13
Figura 3-14. Handrail 14
Figura 3-15. Platform 14
Figura 3-16. Seguimiento de la posición de X-Y-Z de la carga 17
Figura 3-17. Asignación de entradas salidas del protocolo modbus en Factory i/o 18
Figura 3-18. Primera etapa del grafcet 21
Figura 3-19. Segunda etapa del grafcet 22
Figura 3-20. Tercera etapa del grafcet 23
Figura 4-1. Ejemplo de funciones de membresía (MF) para el caso de la temperatura en una habitación 28
Figura 4-2. Operadores lógicos aplicados a conjuntos borrosos [11]. 29
Figura 4-3. Esquema de un controlador borroso [12]. 30
Figura 4-4. Ejemplo de evaluación de reglas [13]. 31
Figura 4-5. Aplicación del método del centroide [13]. 32
Figura 4-6. Proceso completo de operaciones de un controlador fuzzy [14]. 33
Figura 4-7. FLD 34
Figura 4-8. Editor de MFs 34
Figura 4-9. Editor de reglas de inferencia 35
Figura 4-10. Rule Inference 36
Figura 4-11. Control Surface 36

Figura 4-12. FLC 37
Figura 5-1. Esquema de fuerzas [15] 39
Figura 5-2. Modelo empleado en Simulink [15] 43
Figura 5-3. Modelo de planta de Simulink 43
Figura 5-4. Espacio de estados del sistema 44
Figura 6-1. Respuesta ante escalón del sistema 47
Figura 6-2. Funciones de transferencia de cada variable 47
Figura 6-3. Modelo para la comparación 48
Figura 6-4. Comparación entre modelo no lineal y TFs 48
Figura 6-5. Modelo de control clásico 49
Figura 6-6. Primer resultado de PID-PI 49
Figura 6-7. Aplicación del control en cascada 50
Figura 6-8. Diagrama de bloques de PID 51
Figura 6-9. Ecuaciones de las ganancias de un FLC tipo PID [16] 51
Figura 6-10. Superficie de control de un FLC basado en un PID [16] 53
Figura 6-11. Superficie de control del FLC creado 54
Figura 6-12. Comparativa entre FLC y PID 55
Figura 6-13. Modelo empleado en simulación 55
Figura 6-14. Modelo empleado con FLCs 56
Figura 6-15. Comparación entre PID-PI y FLCs 56
Figura 6-16. Superficie de control del error de posición y velocidad 60
Figura 6-17. Superficie de control del ángulo de balanceo 60
Figura 6-18. Modelo de FLC heurístico 61
Figura 6-19. Comparativa entre FLC heurístico y PID-PI 61
Figura 6-20. Control en cascada frente cambios en la masa de la carga 62
Figura 6-21. FLC frente a cambios en la masa de la carga 63
Figura 6-22. Inicialización de la comunicación modbus 64
Figura 6-23. Comunicación modbus con controlador 65
Figura 6-24. Modelo para FLC de un autómata 65
Figura 6-25. Gaussiana 66
Figura 6-26. Ejemplo de asignación de grado de pertenencia a la salida 67
Figura 6-27. Cálculo del grado de pertenencia del ángulo 67
Figura 6-28. Regla dominante para la MF NL 67
Figura 6-29. Bucle de cálculo 68
Figura 6-30. Cálculo del crisp 68
Figura 6-31. Control mediente FLC en autómata 69

Notación

AGV	Automatic Guided Vehicle
LD	Ladder Diagram
SFC	Sequential Function Chart
IL	Instruction List
POU	Program Organization Unit
FIS	Fuzzy Inference System
FLD	Fuzzy Logic Designer
MF	Membership Function
IR	Inference Rule
FLC	Fuzzy Logic Controller
COF	Coefficient Of Friction
E-L	Euler-Lagrange
MIMO	Multiple Input Multiple Output
SS	State Space
TF	Transfer Function
SCI	Sistema Compatible Indeterminado

1 INTRODUCCIÓN

En el ámbito industrial, los trabajadores enfrentan múltiples riesgos laborales que afectan su seguridad y salud. Según estadísticas recientes, sectores como la manufactura y la logística presentan tasas significativas de accidentes debido a factores como el uso de maquinaria pesada, el manejo manual de cargas y las condiciones de trabajo en entornos exigentes. Estos incidentes no solo comprometen la integridad física de los empleados, sino que también generan importantes pérdidas económicas y operativas para las empresas, acentuando la necesidad de adoptar medidas efectivas para prevenirlos.

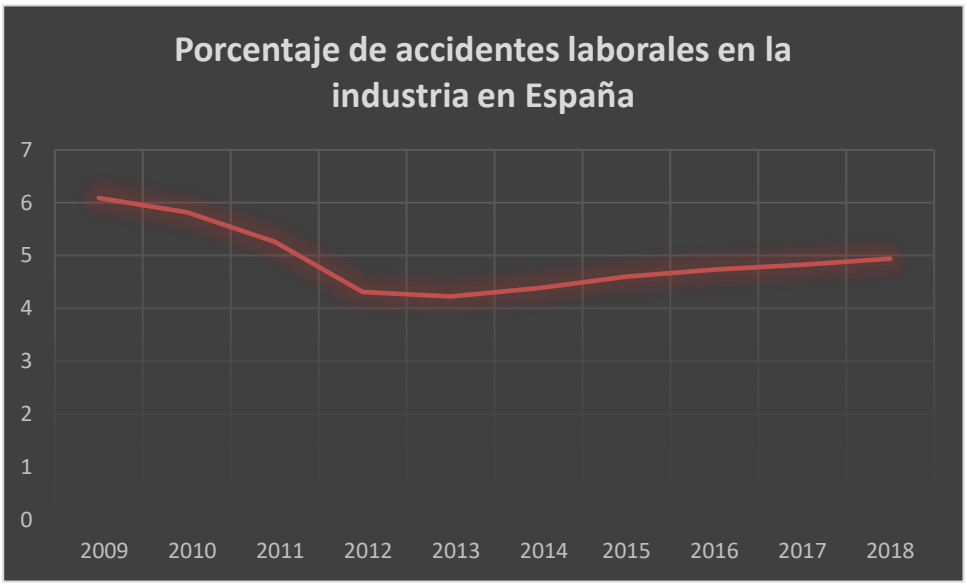


Figura 1-1. Tasa (%) de accidentes laborales en España a lo largo de la década (2009-18) [1]

En este contexto, el presente trabajo se enfoca en una problemática específica: el manejo de cargas pesadas. Actualmente, estas operaciones suelen realizarse con equipos manuales como transpaletas, lo que expone a los operarios a riesgos significativos. El proceso de carga y transporte, debido al peso y volumen de las cargas, incrementa las probabilidades de accidentes graves, creando puntos críticos de peligrosidad dentro de la planta. Este proyecto busca contribuir a la reducción de estos riesgos mediante la implementación de un sistema automatizado que no solo mejore la seguridad, sino que también optimice la productividad de los procesos industriales. Teniendo todo lo anterior en cuenta, la solución propuesta pretende minimizar la intervención humana en áreas críticas, reduciendo la exposición a riesgos y garantizando condiciones de trabajo más seguras. Al mismo tiempo, se busca optimizar los procesos de carga mediante una operación más eficiente y consistente, promoviendo una mayor productividad y alineándose con los estándares modernos de automatización industrial.

El diseño del sistema de control se basó en lógica borrosa, un método conocido por su capacidad de gestionar sistemas complejos y con incertidumbre. Para su desarrollo y simulación se utilizó MATLAB, empleando herramientas específicas como Simulink y el Fuzzy Logic Toolbox. Estas plataformas permitieron modelar el comportamiento del sistema y optimizar las reglas borrosas antes de su implementación en un PLC. Una de las ventajas principales de este enfoque radica en su flexibilidad, ya que permite incluir diversas variables que influyen en la operación, como la posición, velocidad y peso de la carga.

El proyecto también incluye la comparación del controlador borroso con un control en cascada PID+PI convencional, realizado en el entorno de simulación de Simulink. Estas comparaciones demostraron que la lógica borrosa no solo proporciona una mejor respuesta dinámica, sino que también se adapta más eficientemente a cambios en las condiciones de operación, como las variaciones en el peso de la carga.

Aunque el diseño del controlador borroso puede requerir un mayor tiempo de ajuste debido a la importancia de definir correctamente sus reglas, su flexibilidad y rendimiento justifican ampliamente su elección. Por otro lado, el uso de MATLAB y Simulink facilitó tanto la validación del modelo como la comparación con otras estrategias de control, destacando la versatilidad de estas herramientas para el desarrollo de proyectos industriales.

Adicionalmente, se desarrolló un modelo virtual de la planta utilizando el software Factory IO. Este modelo permitió recrear de manera precisa la casuística real del proceso de carga, incluyendo aspectos como la disposición de los elementos, el flujo de materiales y las restricciones operativas. Sobre esta base, se implementó la automatización del sistema mediante un PLC simulado en el entorno Machine Expert Basic, mostrando una prueba visual de la automatización del proceso. Esta metodología permitió validar la funcionalidad del sistema en un entorno seguro y controlado antes de su eventual aplicación en un entorno físico.

En conclusión, este trabajo no solo implementa una solución ampliamente validada en la industria, sino que también explora el uso de un enfoque de control intuitivo y adaptable para mejorar la seguridad y la productividad en el manejo de materiales pesados. Además, el desarrollo y la validación del sistema mediante simulaciones en Factory IO y Machine Expert Basic permitió replicar con precisión las condiciones reales de la planta, garantizando la fiabilidad y eficacia de la solución antes de su implementación física.

2 VIABILIDAD TÉCNICA

Como paso previo a la selección de una solución, es necesario plantear diferentes propuestas que puedan ser empleadas para abordar el problema presente. En este caso particular, se han considerado tres posibles planteamientos que utilizan diferentes tecnologías, cada una con sus propias ventajas e inconvenientes. Esta sección actúa como preámbulo y justificación de la solución expuesta en los apartados siguientes. Ya que la solución escogida se encuentra parcialmente condicionada por el problema de origen de la planta en la que se basa este trabajo, se presenta en este un pequeño esquema:

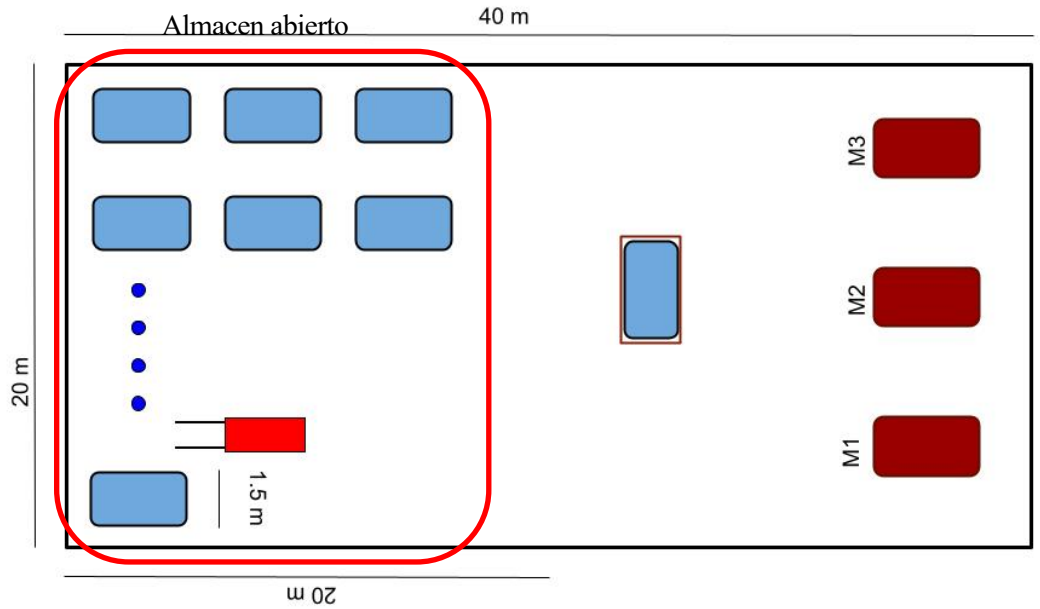


Figura 2-1. Layout de la situación de origen

Como se puede ver en la figura hay tres zonas diferenciadas, el almacén de lotes de chapas, la estación intermedia y los depósitos de las máquinas. El almacén está muy limitado debido a que se necesita espacio para el movimiento y maniobra de los transpaletas para el depositado y recogida de lotes. La estación intermedia esta descubierta pues los lotes se depositan mediante los transpaletas antes mencionados. Por último, el traslado de la carga libre desacoplada (lotes de chapas) se realiza de nuevo con transpaletas siendo una operación compleja e insegura para los operarios implicados. Teniendo en cuenta todo esto se procede a comparar las soluciones propuestas.

2.1 Puente grúa automático

2.1.1 Funcionamiento

El sistema propuesto utilizaría una grúa para realizar todas las operaciones. La secuencia se iniciaría cuando el operario seleccionase el formato y el proceso de transformación del material, o lo que es lo mismo, el depósito de la máquina que procesará el material. Tras ello, la grúa recogería un lote del formato seleccionado y lo trasladaría a la celda de operación manual, donde un operario cortaría las bridas metálicas que sujetan la carga el material al palé, desacoplándolos, para después asegurar la carga mediante cinchos. Toda la operación se haría con la grúa alejada de la celda y parada, esperando a la confirmación manual del operario de la finalización del

proceso. Posteriormente a un tiempo de seguridad después de la confirmación, la grúa trasladaría la carga a la estación seleccionada, esperando en su posición home hasta que hubiese una nueva orden.

2.1.2 Ventajas

El sistema presenta diversas ventajas, entre las que destaca la utilización de maquinaria genérica y sencilla, lo que facilita su implementación y mantenimiento. Esto también se traduce en un número reducido de sensores y actuadores, optimizando la simplicidad y los costos operativos. Asimismo, se incluye la posibilidad de implementar un modo manual, directo y fácil de aprender, lo que garantiza que los operarios puedan adaptarse rápidamente al sistema. Una de sus características más relevantes es su robustez ante fallos, ya que permite continuar la operación manualmente en caso de errores en el modo automático. Finalmente, el sistema cuenta con una gran durabilidad y un mantenimiento sencillo, lo que contribuye a su fiabilidad y eficacia a largo plazo.

2.1.3 Inconvenientes

Sin embargo, el sistema tiene un inconveniente importante: la carga se maneja suspendida en el aire, lo que implica que cualquier fallo puede ser extremadamente peligroso, representando un alto riesgo para la seguridad.

2.2 Manipulador

2.2.1 Funcionamiento

El sistema constaría de sendos depósitos de máquinas, una estación de operación y un manipulador móvil. La descarga en la plataforma de operación se realizaría mediante el método original de descarga. Posteriormente, y al igual que en el caso de la grúa el operario cortaría las bridas de acople entre la carga y el palé. El robot, una vez confirmada la ausencia de presencia humana en el entorno de trabajo, trasladaría la carga por unidad de la estación de operación a la estación de almacenaje de la máquina seleccionada. Al finalizar, el robot volvería a la posición home, esperando nuevas órdenes.

2.2.2 Ventajas

Este sistema ofrece un alto nivel de seguridad al manejar cargas de menor peso, lo que minimiza los riesgos asociados. Además, al usar un manipulador la obra es de instalación supone un menor coste que la opción previa. Otra ventaja destacada es su capacidad de adaptarse a diferentes condiciones de operación, lo que lo convierte en una solución versátil para distintas necesidades industriales.

2.2.3 Desventajas

Sin embargo, el sistema presenta algunas desventajas notables. Su complejidad radica en el uso de numerosos sensores y actuadores, lo que incrementa la dificultad mantenimiento. Además, requiere un alto nivel de utilización debido a la descarga por unidad, lo que puede generar costos operativos elevados. La instalación, a pesar de que menos voluminosa, resulta compleja, demandando más tiempo para su implementación. Asimismo, el mantenimiento exige mayor asiduidad, lo que puede afectar la disponibilidad del sistema. Finalmente, identificar el origen de los fallos puede ser complicado, lo que podría prolongar los tiempos de reparación.

2.3 Automatic Guided Vehicle

2.3.1 Funcionamiento

El sistema constaría de sendos depósitos de máquinas, una estación de operación y un AGV móvil. La descarga en la plataforma de operación lo realizaría el AGV recogiendo la carga. Posteriormente, y al igual que en los anteriores casos, el operario cortaría las bridas de acople entre la carga y el palé. El AGV, una vez confirmada la ausencia de presencia humana en el entorno de trabajo, trasladaría el lote de la estación de operación a la estación de almacenaje de la máquina seleccionada. Al finalizar, el AGV volvería a la posición home, esperando

nuevas órdenes.

2.3.2 Ventajas

Este sistema destaca por su alto nivel de seguridad al manejar los lotes a nivel del suelo de manera completamente autónoma, lo que reduce significativamente los riesgos operativos. Además, sus dimensiones reducidas permiten su instalación en espacios limitados, lo que favorece su versatilidad. Es modulable para adaptarse a diferentes tipos de carga, lo que amplía sus posibilidades de uso en diversas aplicaciones industriales. Asimismo, tiene la capacidad de integrarse con relativa facilidad en entornos industriales más grandes o complejos, ofreciendo una solución escalable y eficiente.

2.3.3 Desventajas

A pesar de sus ventajas, el sistema presenta ciertas desventajas. En primer lugar, su complejidad técnica, derivada del uso de numerosos sensores y actuadores, puede aumentar los costos y la dificultad de mantenimiento. La instalación resulta ser complicada y depende en gran medida de la presencia de otros objetos o personas en el entorno, lo que podría afectar su rendimiento. Por último, para que funcione correctamente, es necesario disponer de una infraestructura específica, lo que podría limitar su implementación en algunos entornos.

2.4 Conclusiones

Tras analizar las diferentes alternativas propuestas para la automatización del manejo de materiales pesados, se concluye que la opción del puente grúa automático es la más adecuada para esta aplicación específica. Esta elección se fundamenta en varios factores clave que hacen que esta solución sobresalga frente a las demás.

En primer lugar, es un sistema ampliamente validado por su uso en entornos como los puertos comerciales [2], donde se emplean para manejar cargas pesadas de forma eficiente y segura. Estos antecedentes demuestran la capacidad de esta tecnología para operar en condiciones diversas y su fiabilidad a lo largo del tiempo.

Además, el sistema de grúa presenta una complejidad significativamente menor en comparación con las otras alternativas, como el manipulador o el AGV. Su reducido número de sensores y actuadores no solo simplifica su diseño e instalación, sino que también minimiza los puntos de fallo potenciales, lo que incrementa la robustez del sistema. En caso de un fallo en el modo automático, la posibilidad de operar manualmente de manera directa permite garantizar la continuidad de las operaciones sin grandes interrupciones.

Por otro lado, la grúa destaca por su mantenimiento sencillo y durabilidad, factores que reducen los costos operativos a largo plazo. A diferencia del manipulador o el AGV, que requieren instalaciones más complejas y un mantenimiento más frecuente, la grúa ofrece un ciclo de vida más prolongado con menores intervenciones técnicas.

Finalmente, si bien el manejo de cargas suspendidas representa un riesgo en caso de fallos graves, este aspecto puede mitigarse mediante un diseño adecuado de los sistemas de seguridad, como sensores redundantes y procedimientos estrictos de operación. La experiencia acumulada en su uso en otros sectores respalda esta afirmación.

En resumen, el puente grúa automático se posiciona como la mejor opción por su simplicidad, confiabilidad, menor coste operativo y respaldo en aplicaciones similares, ofreciendo una solución robusta y eficiente para la automatización del manejo de materiales en este proyecto.

2.5 Solución planteada en planta

Como se ha comentado al principio de este capítulo, la solución escogida se encuentra parcialmente condicionada por el caso específico a solucionar en una planta donde la carga son lotes de chapas metálicas con diferentes formatos y pesos. Como es obvio tras escoger una de las alternativas expuestas es conveniente exponer como quedaría el layout de la solución propuesta.

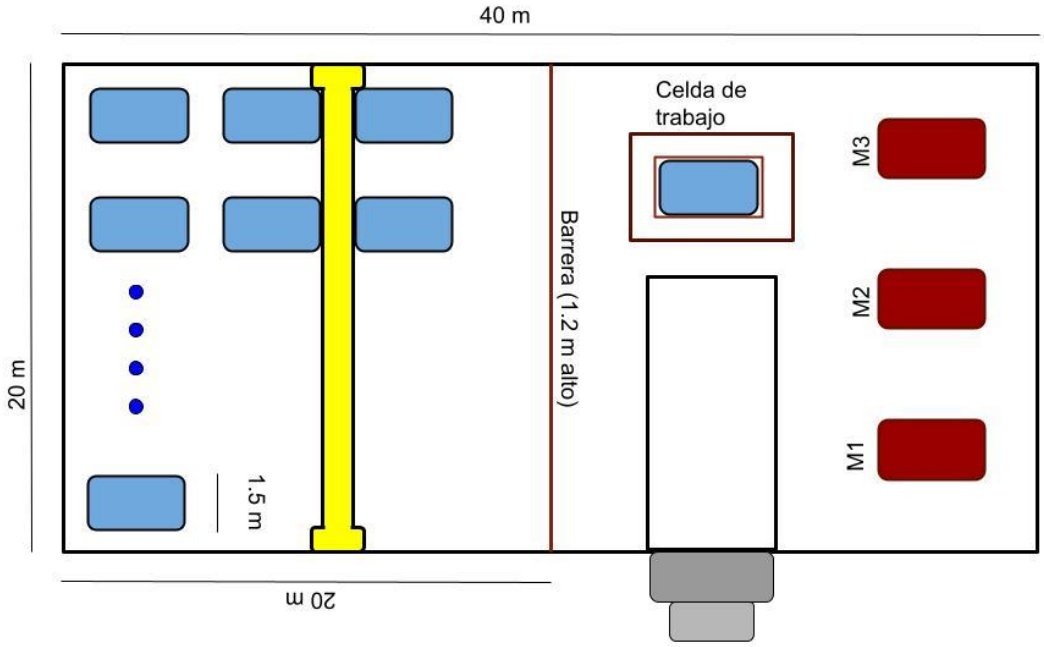


Figura 2-2. Layout de solución planteada

Como se puede ver se solucionan todos los problemas de los cuales padecía la situación de partida, aislando la zona destinada a almacén y haciendo más eficiente el espacio al no necesitar de grandes espacios para la maniobra de transpaletas, cerrando la zona de trabajo manual al introducir una celda posibilitando así protocolos de seguridad para la entrada y salida de esta. Lo único cuestionable es que es necesario que los camiones que entran con el material han de entrar en la sección de operación de la grúa para poder usar esta para el almacenado de lotes, haciendo algo compleja la instalación en otro tipo de plantas.

Con todo lo expuesto, se puede ver que la solución planteada mejora la situación actual, pero queda realizar la tarea de simular el sistema y verificar cuan bueno puede llegar a ser como alternativa al proceso actual. De todo esto es de lo que nos encargaremos en el siguiente capítulo.

3 SIMULACIÓN DEL SISTEMA

La simulación de sistemas automáticos es una herramienta esencial en el desarrollo y validación de soluciones industriales. Antes de implementar un sistema en un entorno físico, es fundamental evaluar su comportamiento en condiciones virtuales, lo que permite identificar posibles fallos, optimizar el diseño y garantizar su funcionalidad. Este enfoque reduce los riesgos asociados a la puesta en marcha y minimiza los costos de ajustes posteriores.

En este proyecto, se emplearon dos herramientas clave para la simulación: Factory IO, para modelar el entorno físico y las interacciones del sistema, y Machine Expert Basic, para simular la lógica de control del PLC. La integración de ambas herramientas permitió realizar pruebas del sistema planteado con el problema del balanceo de la carga solucionado, permitiendo centrar el foco en este apartado en la automatización del proceso.

En el caso particular en el que se ha basado el trabajo, es necesario realizar una operación manual sobre la carga previamente para poder introducirla en los depósitos de las máquinas, lo que lo hace más complejo que un caso en el que simplemente sea necesario trasladar la carga entre dos puntos deseados. Es por ello por lo que cobra todavía más relevancia la simulación, pues es una situación de colaboración hombre-máquina en la que se debe de asegurar la integridad del operario.

3.1 Factory i/o

3.1.1 Introducción

Factory IO es una plataforma de simulación diseñada para crear entornos virtuales que emulan procesos industriales reales. Utiliza un motor gráfico basado en Unity 3D, para ofrecer una representación visual detallada y realista de los sistemas modelados. Este software está orientado tanto a estudiantes como a profesionales en el ámbito de la automatización industrial, brindando una herramienta versátil para el diseño, la simulación y la enseñanza de sistemas automatizados.

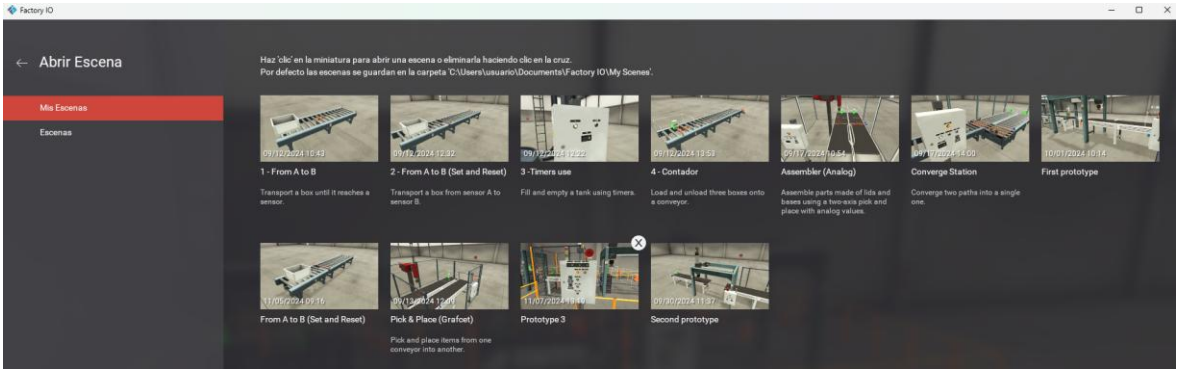
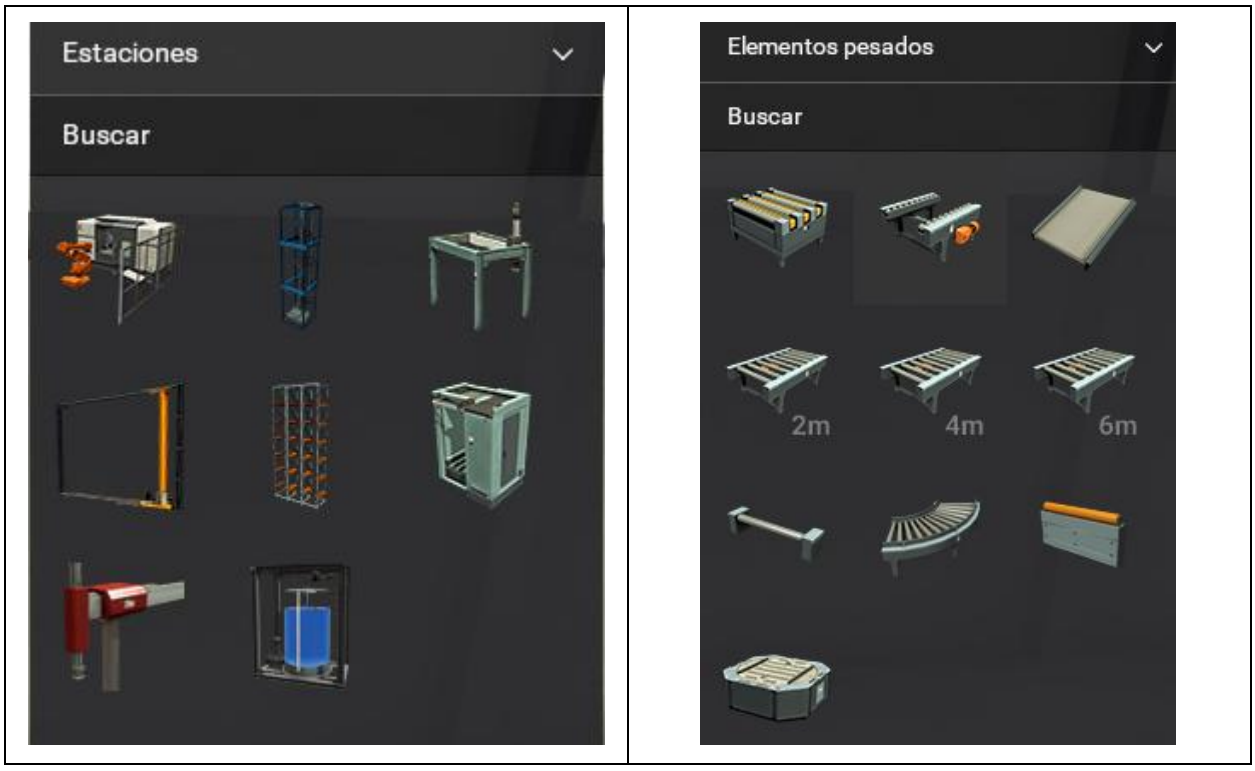


Figura 3-1. Menú de selección de escena de Factory i/o

Entre sus funcionalidades básicas, Factory IO permite crear modelos de plantas industriales mediante una biblioteca extensa de componentes predefinidos, que incluye sensores, actuadores, transportadores, robots y otros elementos típicos de entornos automatizados. Estos componentes se pueden configurar y conectar para reproducir flujos de trabajo, probar secuencias de operación y validar soluciones de control. Además, el software es compatible con diversos controladores lógicos programables (PLC), sistemas SCADA y software de programación, lo que facilita la integración de simulaciones con entornos de control reales o virtuales.

Tabla 3-1. Campos de selección de equipos de Factory i/o



Otra característica destacada es su capacidad para simular fallos en los componentes, lo que resulta útil para evaluar la robustez de los sistemas de control y preparar estrategias de respuesta ante eventos inesperados. Asimismo, permite realizar mediciones en tiempo real, como el conteo de piezas, tiempos de ciclo y eficiencia del sistema, proporcionando datos valiosos para optimizar procesos.

Figura 3-2. Estados de funcionamiento de sensores y actuadores

Esto lo ha convertido en una herramienta clave en el proyecto para diseñar y validar sistemas de automatización sin necesidad de contar con una instalación física, reduciendo costos y riesgos. Su enfoque intuitivo y su

versatilidad han reducido el problema de diseño del entorno a mínimos, propiciando un mejor desarrollo del automatismo, sin necesidad de preocuparse de problemas relacionados con el modelo empleado.

3.1.2 Bloques usados

En el diseño del modelo se emplearon diversos bloques disponibles en Factory IO, cada uno desempeñando funciones específicas para emular las operaciones del sistema automático. A continuación, se describen los bloques utilizados y su papel dentro de la simulación:

1. **Emitter & Remover:**

   El bloque Emitter genera los lotes de material necesarios para el sistema, representados como raw metal y square pallet. El bloque Remover, por su parte, elimina los lotes de la simulación una vez completadas las operaciones asociadas, manteniendo el flujo de trabajo dentro del modelo.

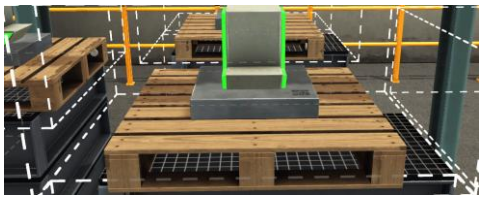


Figura 3-3. Emitter con pallet y material

Figura 3-4. Remover

2. **Pick and Place (Analog):**

   Este bloque emula el funcionamiento del puente grúa mediante movimientos controlados en los ejes X, Y y Z, además de la acción de "coger" (grab) el material. Debido a las limitaciones de tamaño en Factory IO, se utilizaron dos bloques de Pick and Place: uno para la recogida del material y otro para la descarga en el depósito de la máquina seleccionada.

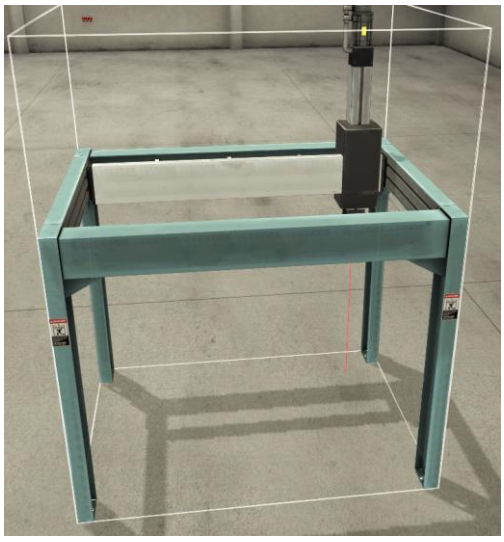


Figura 3-5. Pick & Place

3. **Safety Door & Safeguards:**

   Estos bloques emulan la celda de trabajo donde la puerta, equipada con un sensor, detecta si hay alguien dentro o no. Se utiliza para garantizar la seguridad en la estación de operación, emulando el control de acceso del operario durante el proceso de corte.

Figura 3-6. Safety door

Figura 3-7. Safeguard

4. Diffuser Sensor & Belt Conveyor (2m):

Estos bloques se añadieron como elementos extra para trasladar la carga desde el Emitter al Remover dentro de la estación de operación, en el sistema real estos o existirían.

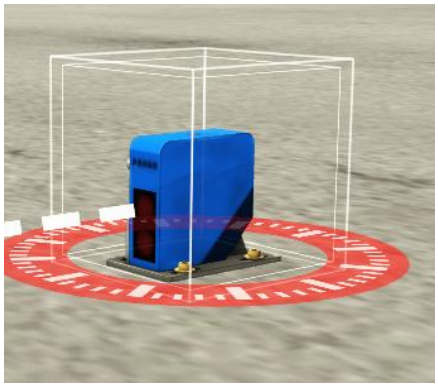


Figura 3-8. Diffuser sensor

Figura 3-9. Belt conveyor

5. Warning Devices:

Este conjunto de bloques incluye tres elementos:

- Warning Light, que se activa para avisar de la próxima activación del pick and place para la recogida del material de la estación de trabajo.
- Stack Light, que indica el estado del sistema mediante colores: rojo para emergencia, amarillo para espera o parada, y verde para funcionamiento normal.
- Alarm Siren, que solo se activa si el botón de parada de emergencia está presionado.

Figura 3-10. Warning Light

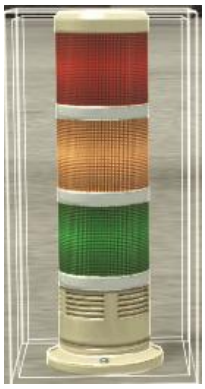


Figura 3-11. Stack Light

Figura 3-12. Alarm Siren

6. **Panel de Control:**

Este bloque centraliza todos los botones, selectores y contadores necesarios para interactuar con la simulación. Permite al operario seleccionar formatos, activar procesos y supervisar el estado del sistema.

Figura 3-13. Panel de control

7. **Handrails & platforms:**

Los primeros sirven para delimitar la zona de operación, al igual que se haría en una planta real y los segundos sirven para permitir establecer los emitters a una altura suficiente para que el bloque pick and place pueda recoger la carga.

Figura 3-14. Handrail

Figura 3-15. Platform

Estos bloques permiten replicar de manera funcional las operaciones y la lógica del sistema descrito, garantizando un entorno de simulación realista y eficiente para probar y ajustar el control del PLC antes de su implementación.

3.1.3 Modelo de planta

Debido a las limitaciones de Factory IO, que no permite modificar las dimensiones de sus elementos, se ha diseñado una escena que, aunque no es idéntica a la planta descrita en el sistema físico, resulta funcionalmente equivalente desde la perspectiva del PLC. Este modelo incluye tres secciones principales que emulan las operaciones fundamentales del sistema: la recogida del material, la estación intermedia y la zona de descarga. A continuación, se describen en detalle estas secciones y la funcionalidad que cumple el PLC en cada una de ellas.

- **Zona de recogida del material:** Esta sección simula el almacenamiento de los lotes de material, representados por tres formatos distintos para simplificar el modelo. El sistema utiliza un pick and place que sustituye al puente grúa, encargado de recoger el formato solicitado por el operario a través de los selectores del cuadro de control. Una vez seleccionado el formato, el pick and place traslada el lote a la estación intermedia, donde continuará el proceso.
- **Estación intermedia:** En esta sección, se simula el proceso manual de corte de bridas mediante la apertura de una puerta con detector. La puerta permanece abierta durante todo el proceso de corte, y el operario debe pulsar el botón de "finish" para confirmar la finalización de la tarea. Tras esta acción, se activa una luz rotatoria y se inicia un temporizador de seguridad. Al finalizar este tiempo, el sistema activa la recogida del lote por el pick and place para continuar con el traslado. Si al finalizar el tiempo de seguridad la puerta sigue abierta, el sistema asume que hay presencia humana en la celda, y será necesario confirmar nuevamente la finalización de la tarea manual para garantizar la seguridad.
- **Zona de descarga:** Esta sección es funcionalmente análoga a la zona de recogida, pero en este caso, el pick and place traslada el lote desde la estación intermedia hasta el depósito de la máquina seleccionada por el operario mediante los selectores del cuadro de control. El sistema finaliza el proceso cuando el lote es colocado correctamente en el destino seleccionado.

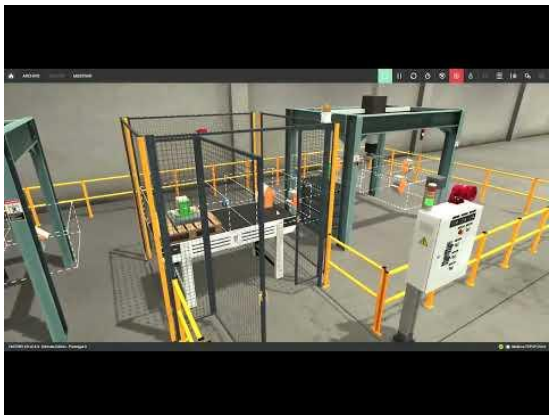
Para gestionar las operaciones de traslado entre secciones, así como la creación y eliminación de lotes dentro de la simulación, se utilizaron los bloques emitters y removers de Factory IO. El PLC, por su parte, supervisa y controla toda la lógica del sistema, incluyendo la selección de formatos, la activación de los elementos mecánicos y las comprobaciones de seguridad, garantizando el correcto funcionamiento de todo el proceso simulado.

3.1.4 Simulaciones

Todos los videos presentes en este apartado son grabados haciendo uso de las dos plataformas mencionadas, con Factory i/o siendo la cara visible en los vídeos, mientras que Machine Expert trabaja debajo haciendo uso del programa diseñado, permitiendo así realizar las simulaciones expuestas.

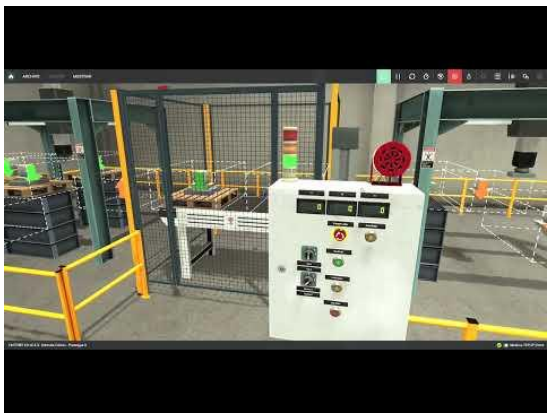
3.1.4.1 Funcionamiento en Automático

En este vídeo se puede ver un ciclo completo de operación tal y como se encuentra descrito en el apartado 3.2.3.2 del proceso automatizado de recogida, procesado manual y depositado en las máquinas.



3.1.4.2 Reset & Stop

En este video se puede ver en funcionamiento las funciones de los botones reset y stop. El primero devuelve al estado de reposo el sistema y el segundo cancela la consecución entre estados dejando el sistema en el estado en el que se encontrase. Estas operaciones pueden ser complementarias, es decir, si el sistema se ha parado por un problema con el tratamiento del material, siendo este inútil, se puede reinicia el sistema para empezar de nuevo; en cambio sí le problema es otro que no involucre la inutilidad del material procesado, el sistema puede simplemente quedarse en "stand by" para continuar por donde se quedó una vez las condiciones de parada desaparezcan. Además, el botón reset reinicia los contadores de piezas procesadas., algo que solo hace si se esta en caso de estar en un estado no anómalo de procesado.



3.1.4.3 Emergency

Este vídeo muestra el funcionamiento de la activación del protocolo de emergencia que comienza con la pulsación de la seta de emergencia y que acaba tras volver a pulsarla. Durante el proceso de parada de emergencia no se completa la operación como en el caso anterior, sino que se sistema se queda en la situación en la que estuviese. Una vez solucionada la contingencia que provocó la emergencia, al volver a pulsar la seta, el sistema vuelve al estado de reposo.

3.2 Machine Expert Basic

3.2.1 Introducción

Machine Expert Basic es un software desarrollado por Schneider Electric para la programación y configuración de sus controladores lógicos programables (PLCs), específicamente de la serie Modicon TM221. Este entorno está diseñado para aplicaciones industriales de automatización, ofreciendo una interfaz intuitiva y herramientas versátiles que facilitan la programación y simulación de sistemas automatizados.

Uno de los aspectos más destacados de Machine Expert Basic es su compatibilidad con diversos lenguajes de programación estándar definidos en la norma IEC 61131-3. Entre los lenguajes soportados se encuentran:

- **LD:** Ideal para usuarios con experiencia en diagramas de escalera, común en sistemas eléctricos.
- **IL:** Un lenguaje basado en instrucciones, compacto y eficiente, similar al ensamblador.
- **SFC:** Diseñado para programar procesos secuenciales, permite organizar el control en pasos y transiciones, facilitando la visualización y estructuración lógica de sistemas complejos.

El software también incluye herramientas básicas de simulación que permiten probar la lógica de programación sin necesidad de hardware físico, reduciendo así el tiempo y los costos asociados al desarrollo.

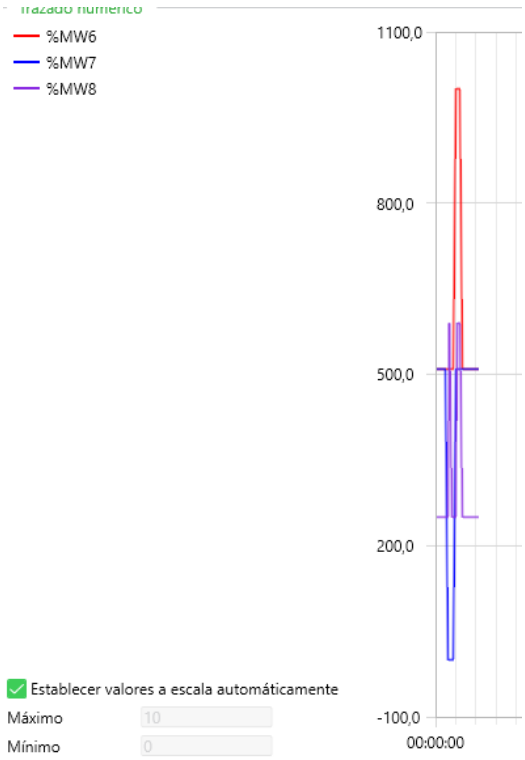


Figura 3-16. Seguimiento de la posición de X-Y-Z de la carga

Entre sus funcionalidades adicionales destacan:

- **Diagnóstico y depuración:** Permite identificar errores en tiempo real y realizar ajustes sobre la marcha.
- **Bibliotecas predefinidas:** Incluye bloques de funciones y plantillas que facilitan la programación de tareas comunes.
- **Compatibilidad con hardware de Schneider Electric:** Garantiza una integración fluida con dispositivos como sensores, actuadores y módulos de expansión.

En resumen, Machine Expert Basic está diseñado para ser una herramienta accesible pero poderosa, adecuada para profesionales de la automatización industrial que necesitan desarrollar y mantener sistemas de control eficientes y seguros. Su enfoque en la simplicidad y la compatibilidad lo convierte en una opción popular en entornos donde la optimización de tiempo y recursos es esencial.

3.2.2 Programación y conexión con factory io

Factory IO se conecta con Machine Expert Basic a través del protocolo Modbus TCP/IP, uno de los métodos más comunes y eficientes para la comunicación entre dispositivos en sistemas de automatización industrial. Este protocolo permite el intercambio de datos en tiempo real, asegurando que las señales generadas en la simulación de Factory IO sean correctamente interpretadas y gestionadas por el PLC configurado en Machine Expert Basic.

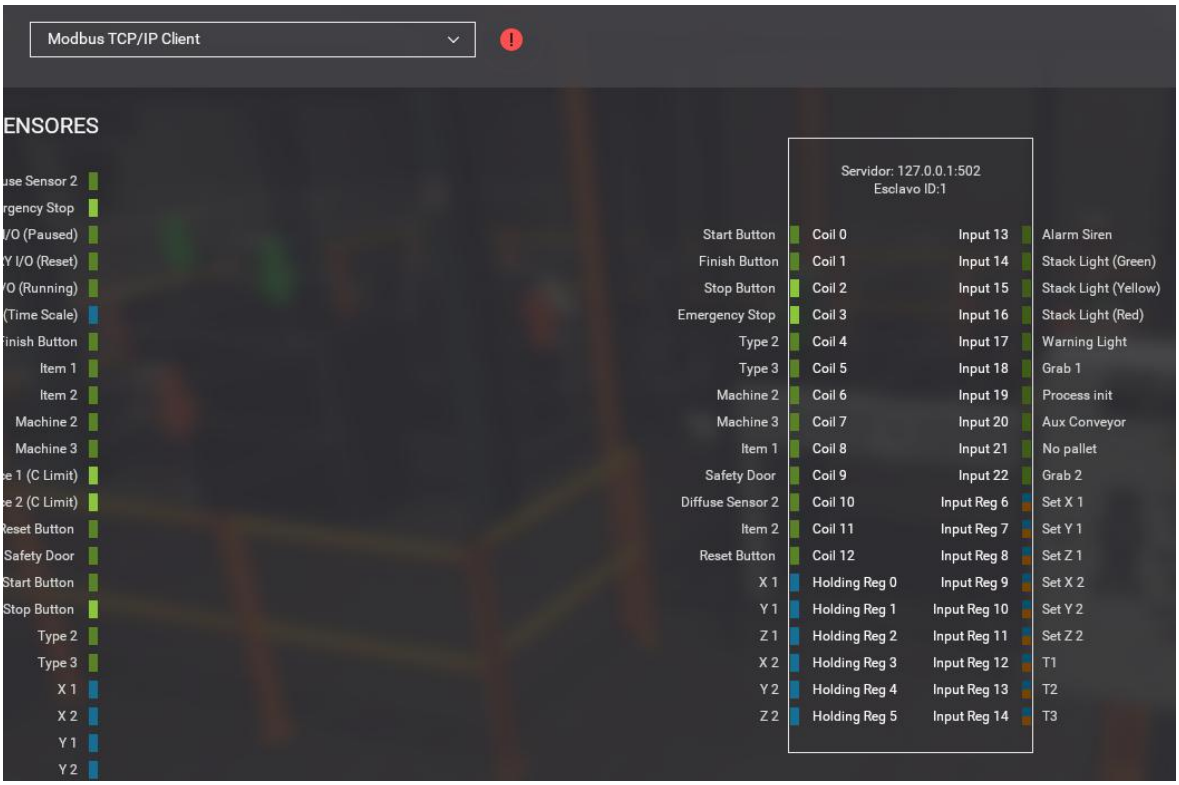


Figura 3-17. Asignación de entradas salidas del protocolo modbus en Factory i/o

Tabla 3-2. Asignación de i/o digitales y analógicos en Machine Expert

Utiliz...	Dirección	Símbolo
☑	%M0	START_BUTTON
☑	%M1	FINISH_BUTTON
☑	%M2	STOP_BUTTON
☑	%M3	EMERGENCY_STOP
☑	%M4	TYPE_2
☑	%M5	TYPE_3
☑	%M6	MACHINE_2
☑	%M7	MACHINE_3
☑	%M8	ITEM_1
☑	%M9	DOOR
☑	%M10	EXIT_PROCESS_SENSOR
☑	%M11	ITEM_2
☑	%M12	RESET_BUTTON
☑	%M13	ALARM_SIREN
☑	%M14	GREEN_LIGHT
☑	%M15	YELLOW_LIGHT
☑	%M16	RED_LIGHT
☑	%M17	WARNING_LIGHT
☑	%M18	GRAB_1
☑	%M19	PROCESS_INIT
☑	%M20	AUX_CONVEYOR
☑	%M21	PROCESS_EXIT
☑	%M22	GRAB_2

Utiliz...	Equ utili...	Dirección	Símbolo
☑	☐	%MW0	X_1
☑	☐	%MW1	Y_1
☑	☐	%MW2	Z_1
☑	☐	%MW3	X_2
☑	☐	%MW4	Y_2
☑	☐	%MW5	Z_2
☑	☐	%MW6	SET_X_1
☑	☐	%MW7	SET_Y_1
☑	☐	%MW8	SET_Z_1
☑	☐	%MW9	SET_X_2
☑	☐	%MW10	SET_Y_2
☑	☐	%MW11	SET_Z_2
☑	☐	%MW12	M1
☑	☐	%MW13	M2
☑	☐	%MW14	M3

La relación entre la figura y la tabla expuestas se muestra en el siguiente apartado.

3.2.2.1 Asignación de registros modbus

La conexión se establece asignando las variables de Factory IO a registros específicos del PLC mediante las siguientes convenciones:

Señales digitales:

Las entradas y salidas digitales de Factory IO se asignan a los registros del tipo %Mx en Modbus. Estos registros se relacionan directamente con:

- Entradas digitales (%Ix), para señales provenientes de sensores u otros dispositivos de entrada.
- Salidas digitales (%Qx), para señales destinadas a actuadores como luces o motores.

Señales analógicas:

Las entradas y salidas analógicas de Factory IO se asignan a registros del tipo %MWx en Modbus. Estos registros se relacionan con:

- Entradas analógicas (%IWx), para señales de sensores como potenciómetros o medidores.
- Salidas analógicas (%QWx), utilizadas para controlar dispositivos como servomotores o variadores de frecuencia.

Tabla 3-3. Correspondecia entre input y outputs

	Input	**Output**	**Register**
Digital	%Ix	%Qx	%Mx
Analog	%IWx	%QWx	%MWx

3.2.2.2 Configuración de la Conexión

En Factory IO:

- Habilitar el protocolo Modbus TCP/IP desde el panel de conectividad.
- Definir la dirección IP del PLC o simulador (127.0.0.1).
- Asignar los registros Modbus correspondientes a cada entrada y salida digital o analógica.

En Machine Expert Basic:

- Declarar las variables de entrada y salida en el programa (%Mx y %MWx).
- Configurar la tabla de asignación para asegurar la correcta correspondencia con los dispositivos simulados.

Este enfoque no solo simplifica la integración, sino que también permite probar y depurar la lógica del sistema de manera eficiente antes de llevarla al entorno físico. La equivalencia en los registros asegura que la simulación sea una réplica fiel de las operaciones reales, optimizando el desarrollo y reduciendo posibles errores.

3.2.3 Programa

3.2.3.1 Estructuración

El programa desarrollado está dividido en varios POUs para estructurar y organizar mejor la lógica de control. Cada uno de estos módulos tiene una función específica, lo que facilita la comprensión, el mantenimiento y la ampliación del sistema. Los POUs son los siguientes:

- **Grafcet:** Este programa, implementado en SFC contiene toda la lógica secuencial del sistema. Define los diferentes estados de la máquina (asociados a los registros %Xi del PLC) y las transiciones entre

ellos, organizando el flujo del proceso de manera clara y estructurada.

- **Actions:** Un programa en LD que gestiona las condiciones necesarias para la activación de los actuadores del sistema. Aquí se define cuándo deben activarse elementos como motores, puertas o luces, dependiendo de las señales de entrada y las condiciones del sistema.
- **Init Cond:** También implementado en LD, este programa asegura que todos los actuadores del sistema se mantengan en su estado de reposo inicial cuando las condiciones lo requieran, garantizando la seguridad y estabilidad del sistema antes de iniciar cualquier operación.
- **Buttons:** Este programa en LD contiene la lógica asociada al panel de control. Gestiona las señales de los botones, selectores y demás dispositivos de entrada del operador, permitiendo la interacción directa con el sistema.
- **Lights:** Otro programa en LD dedicado a la gestión de los dispositivos de advertencia visual, como luces de advertencia y estados (warning lights y stack lights). Aquí se controla la activación de las luces dependiendo de las condiciones del sistema, indicando estados como emergencia, stand-by o funcionamiento normal.
- **Counters and Timers:** Este programa en LD incluye todos los contadores y temporizadores utilizados en el sistema. Su función es medir tiempos de espera, gestionar retardos y contar eventos, elementos críticos para el correcto funcionamiento del proceso secuencial.

Esta división modular permite abordar cada parte del sistema de manera independiente, lo que facilita la depuración y mejora la claridad del programa.

3.2.3.2 Programa principal

El GRAFCET diseñado, que comprende los POUs de 'grafcet', 'actions' e 'init cond' antes mencionados, sigue una secuencia lógica para garantizar el manejo adecuado del material mediante el sistema automatizado. Por comodidad se ha dividido en tres secciones que son las siguientes:

3.2.3.2.1 Movimiento de recogida

1. **Estado inicial (Reposo):** En este estado, todos los actuadores digitales están reseteados, asegurando que no haya actividad accidental. Mientras que los actuadores analógicos (ejes X, Y, y Z) están posicionados en Home, lo que garantiza un punto de referencia conocido y seguro.
2. **Selección del formato:** Según la selección realizada por el operario, el sistema determina el tipo de formato (1, 2, o 3) y calcula la posición correspondiente en el plano XY. Esto activa la transición al estado siguiente.
3. **Movimiento al punto de recogida:** El sistema mueve el eje X e Y a la posición designada según el formato seleccionado. La transición ocurre únicamente cuando ambos ejes alcanzan su posición objetivo, confirmada por sus sensores de posición.
4. **Descenso para recogida:** Una vez alcanzada la posición en XY, el eje Z desciende al nivel de recogida de la carga. En este estado, el sistema activa el actuador de agarre (set 'grab') para sujetar el material.
5. **Confirmación de agarre:** Un sensor en el mecanismo de agarre verifica que el material ha sido correctamente sujetado. Al recibir esta confirmación, el sistema procede a elevar el eje Z a la altura previa, garantizando un traslado seguro.
6. **Traslado a estación de trabajo:** Tras alcanzar la altura inicial, el sistema traslado la carga a la posición de la estación de trabajo en el plano XY.
7. **Depositado:** Análogo al movimiento **4** tras alcanzar la posición designada para los ejes X e Y, se procede a depositar la carga en la estación intermedia.

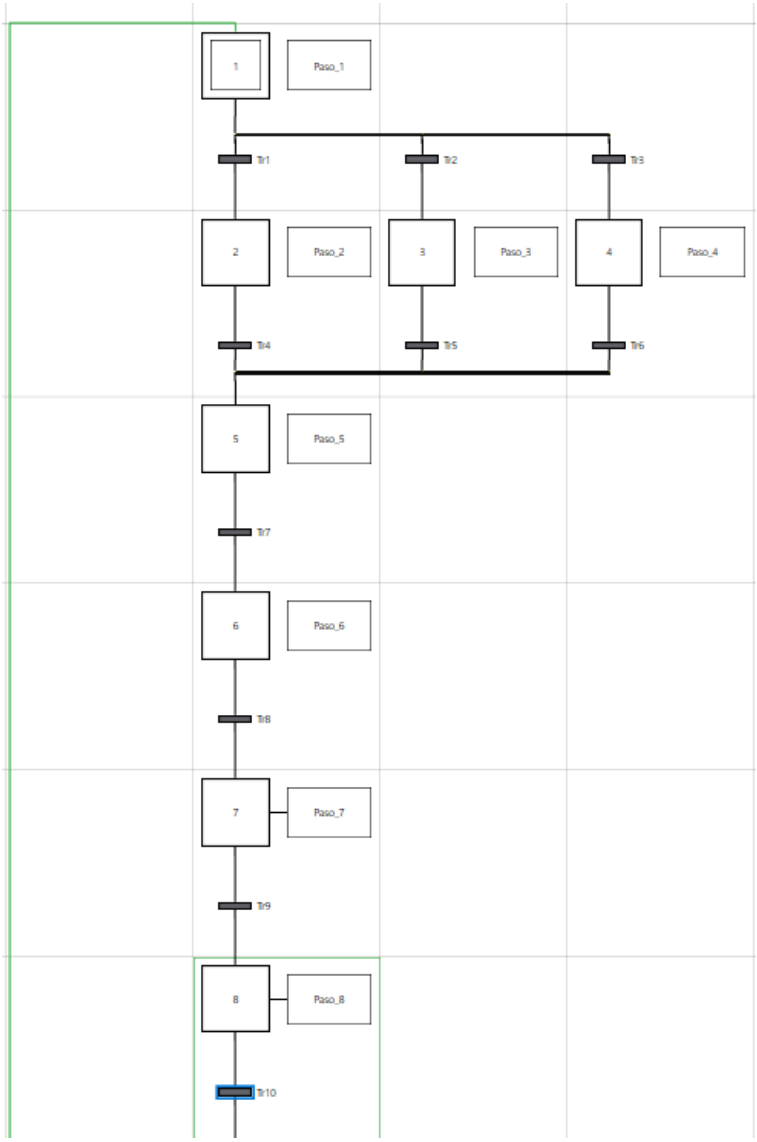


Figura 3-18. Primera etapa del grafcet

3.2.3.2.2 Operación manual

8. **Retorno a posición home:** Una vez depositada la carga en la estación intermedia, el sistema se mueve a la posición Home, alejándose completamente de la zona de trabajo para garantizar la seguridad del operario.
9. **Operación manual del operario:** El operario entra en la celda de trabajo y mantiene la puerta abierta mientras realiza la operación necesaria, verbigracia, cortar las bridas y asegurar la carga, ahora libre, con cinchos. Al finalizar, cierra la puerta y confirma manualmente en el panel de control que la tarea ha concluido.
10. **Activación del tiempo de seguridad:** Se enciende la luz rotatoria como advertencia de que la grúa se moverá tras el tiempo de seguridad (TOF). Si al finalizar el tiempo la puerta está abierta, el sistema asume la presencia del operario y solicita una nueva confirmación tras cerrar la puerta. En cambio, si la puerta está cerrada, se confirma la transición al siguiente estado automáticamente.

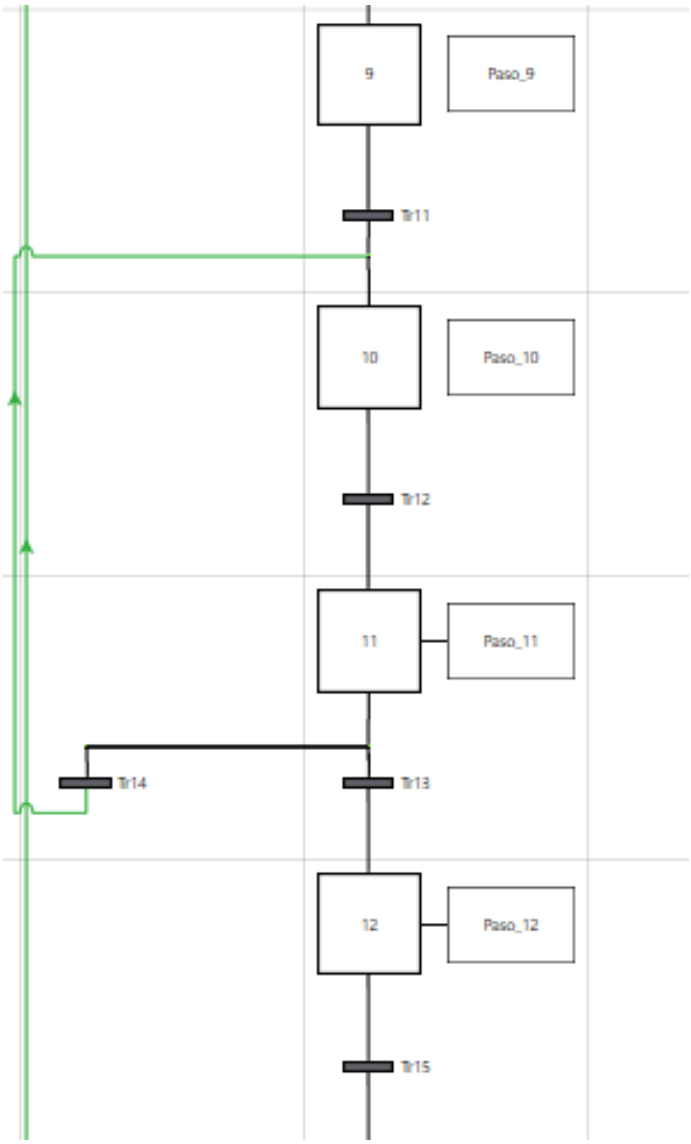


Figura 3-19. Segunda etapa del grafcet

3.2.3.2.3 Movimiento de depositado

11. **Traslado a la estación de trabajo:** El sistema se mueve en el plano XY hacia la posición correspondiente a la celda de trabajo. Pasando al siguiente estado cuando llegue a la posición objetivo.
12. **Recogida de la carga:** El sistema desciende y activa el sistema de agarre (set 'grab') para sujetar la carga. Un sensor confirma que la carga está asegurada, permitiendo que la grúa eleve el material nuevamente a la posición Z previa.
13. **Traslado al depósito de la máquina seleccionada:** Según la selección realizada en el panel de control, sistema se desplaza en el plano XY hacia el depósito de la máquina de procesado correspondiente.
14. **Depósito de la carga:** La grúa desciende y deposita la carga en el almacén de la máquina seleccionada. Una vez alcanzada la altura de depósito, el sistema de agarre se desactiva (reset 'grab') para liberar el material.
15. **Retorno a posición home:** La grúa vuelve a la posición Home, quedando lista para recibir nuevas órdenes, completando así el ciclo.

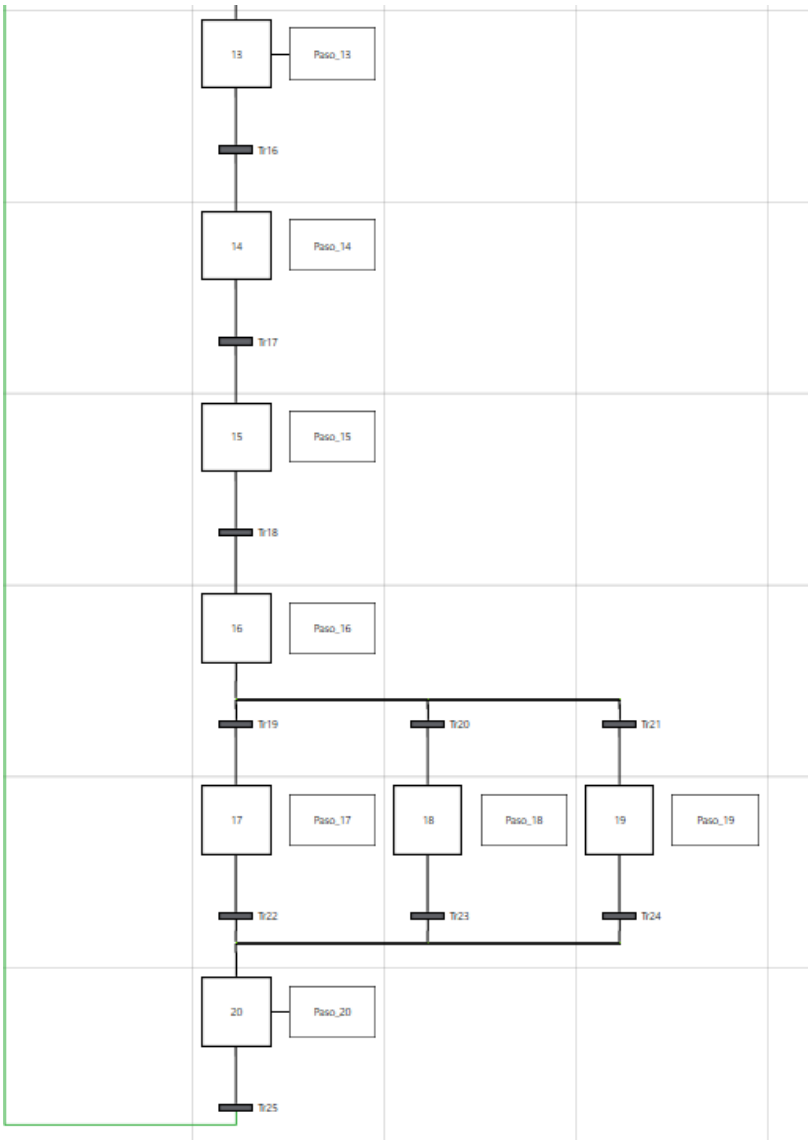


Figura 3-20. Tercera etapa del grafcet

3.3 Conclusión y resumen

A lo largo de este apartado, se ha presentado el desarrollo de un sistema automatizado mediante las herramientas Factory IO y Machine Expert Basic, destacando sus características principales y su papel complementario en el diseño, simulación y validación de sistemas industriales.

Factory IO ha permitido crear un entorno virtual en el que se emula de forma realista el comportamiento de una planta industrial. Aunque cuenta con limitaciones en cuanto a la personalización de algunos elementos, ofrece una plataforma flexible e intuitiva para experimentar con distintos escenarios y realizar pruebas antes de implementar la solución en un entorno físico. Su integración con protocolos estándar como Modbus TCP/IP facilita la comunicación con controladores reales o simulados.

Por su parte, Machine Expert Basic ha proporcionado el entorno ideal para el diseño y programación del controlador lógico programable (PLC). Con su compatibilidad con lenguajes de programación estándar como LD y SFC, ha sido posible implementar la lógica del sistema de manera estructurada, utilizando diversas POU para organizar las funciones clave.

La combinación de ambas herramientas ha permitido modelar, probar y ajustar un sistema automatizado

completo, desde la lógica de control hasta la simulación del comportamiento físico, asegurando la viabilidad y la fiabilidad del diseño. Esto destaca la importancia de utilizar herramientas integradas en la industria para optimizar tanto los procesos de desarrollo como los recursos empleados en la implementación de soluciones reales.

4 LÓGICA BORROSA

En el entorno que nos rodea, existen numerosos sistemas que son o pueden ser controlados mediante diferentes métodos. Un ejemplo común de ello es el control en lazo cerrado usando un controlador PID. Con el paso del tiempo, se ha incrementado el número de sistemas y aplicaciones que los seres humanos buscan automatizar para lograr mayores niveles de eficiencia y optimización. Además de esto, la automatización ofrece la ventaja de reducir costos y aumentar la producción, ya que los sistemas pueden realizar tareas de forma continua, sin necesidad de descansos, a diferencia de las personas, que necesitan reponer su fuerza de trabajo. Esta capacidad de operar durante largos períodos hace que la automatización sea cada vez más utilizada en diferentes sectores, como en fábricas de alimentos procesados o en sistemas logísticos de distribución. Sin embargo, surgen dificultades cuando se tratan de sistemas reales cuya modelización mediante una función de transferencia resulta compleja o incluso imposible, ya sea porque presentan comportamientos no lineales, variables a lo largo del tiempo, o son demasiado complejos para ser descritos con una función matemática precisa. Por esta razón, se han desarrollado métodos alternativos para abordar estos casos. Uno de ellos es el control mediante lógica borrosa, o controlador borroso, que será el foco de este estudio.

Este tipo de lógica se acerca más al razonamiento humano que la lógica tradicional binaria, en la que los resultados son siempre claros y definidos. En la vida cotidiana, muchas situaciones no están fácilmente determinadas. Un buen ejemplo de esto es la temperatura en una habitación, donde no hay un valor exacto que se pueda clasificar como fría, templada o cálida. Cada persona puede tener una sensación distinta sobre qué temperatura corresponde a cada categoría. Sin embargo, lo que sí se puede hacer es establecer rangos para clasificar las temperaturas, como considerar debajo de 18°C como fría, entre 18°C y 22°C como templada, y por encima de 22°C como cálida. Estos rangos, por supuesto, no tienen límites estrictos y pueden variar según la percepción individual. Si la temperatura es de 21°C, siguiendo los criterios tradicionales, se clasificaría como templada, ya que cae dentro del rango estipulado. No obstante, la diferencia con una temperatura de 22°C es mínima, lo que hace que la clasificación no sea tan clara. La lógica borrosa, en lugar de dar una respuesta absoluta, ofrece un grado de pertenencia a cada categoría, algo que se abordará en las siguientes secciones.

Este apartado introduce y explica el método de control que será empleado, destacando su relevancia frente a los métodos tradicionales, especialmente en escenarios donde la precisión matemática no es suficiente o no es viable debido a la naturaleza de los sistemas a controlar. La lógica borrosa se presenta como una solución adaptada a la complejidad y a la flexibilidad requeridas en este tipo de casos.

4.1 Estado del arte

Las bases del concepto de lógica borrosa (fuzzy logic, en inglés) o lógica borrosa, tal como lo conocemos hoy en día, fueron establecidas en 1965 por el profesor Lotfi Asker Zadeh de la Universidad de California en Berkeley, en su artículo “Fuzzy Sets” [13]. De dicho artículo se puede inferir que Zadeh discrepaba del paradigma de la lógica clásica, lo que lo llevó a proponer un nuevo concepto: los conjuntos borrosos. En esta teoría, un elemento puede pertenecer parcialmente a un conjunto, asignándole un valor numérico denominado grado de pertenencia, que va de 0 a 1 [3].

Años después, Zadeh profundizó aún más en su teoría con la publicación de otros dos artículos [4], [5], en los cuales introdujo conceptos adicionales como las variables lingüísticas y las reglas del tipo "If-then". Estos conceptos forman parte fundamental de la lógica borrosa actual y serán explicados en detalle en el siguiente apartado, para su correcta comprensión.

No sería hasta 1974 cuando Ebrahim H. Mamdani diseñó el primer controlador borroso, cuyo propósito era controlar una máquina de vapor [6]. Este trabajo sentó las bases de uno de los métodos de inferencia más utilizados en la lógica borrosa en la actualidad, además del más simple de entender. En este proyecto, el modelo usado es de este tipo.

Una década más tarde, en 1985, Tomohiro Takagi y Michio Sugeno propusieron un método alternativo de control borroso al desarrollado por Mamdani [7], siendo más eficiente computacionalmente pero más complejo

de entender y usar.

Desde entonces, se han realizado numerosas investigaciones y trabajos en los que se ha empleado el control borroso, allí donde le control clásico no consigue llegar.

4.2 Marco teórico

Haciendo un repaso de los conceptos mencionados previamente, se procederá a explicar en detalle cada uno de ellos y el funcionamiento de la lógica borrosa. Como se ha señalado, esta lógica es una herramienta que permite resolver problemas donde el objeto de estudio no puede describirse de forma precisa mediante una función de transferencia, pues es muy no lineal; o presenta una gran complejidad. Esto se logra utilizando un lenguaje más cercano al razonamiento humano, que no asigna valores absolutos en situaciones determinadas, como el ejemplo de la temperatura en la habitación. Por esta razón, se denomina lógica borrosa o borrosa, ya que existen elementos que no pueden clasificarse dentro de los conjuntos clásicos debido a la falta de límites bien definidos. Este desafío fue abordado mediante la creación de los conjuntos borrosos, la base de la lógica que abordaremos.

4.2.1 Conjuntos borrosos

Los conjuntos borrosos se caracterizan por permitir que un elemento pertenezca a ellos en cierto grado, definido por un valor llamado grado de pertenencia (μ). A diferencia de los conjuntos clásicos, que tienen límites estrictos y solo aceptan pertenencia total (1) o ausencia total (0), los conjuntos borrosos asignan a cada elemento un valor numérico dentro del intervalo [0,1]. Este valor indica el nivel de pertenencia del elemento al conjunto: un grado de 1 implica pertenencia completa, mientras que un grado de 0 indica que no pertenece en absoluto. Esta flexibilidad es la principal diferencia entre los conjuntos clásicos y los borrosos, ya que estos últimos permiten describir situaciones más complejas donde cada elemento puede tener un determinado grado parcial de pertenencia a varios conjuntos [8].

4.2.2 Variables lingüísticas

Lotfi Zadeh definió las variables lingüísticas en uno de sus artículos [5] como: “A linguistic variable is defined as a variable whose values are sentences in a natural or artificial language. Thus, if tall, not tall, very tall, etc. are values of height, then height is a linguistic variable” [Una variable lingüística se define como una variable cuyos valores son sentencias en un lenguaje natural o artificial. Así, si alto, no alto, muy alto, muy muy alto, etc. son valores de la altura, entonces la altura es una variable lingüística].

De esta definición se puede concluir que las variables lingüísticas son aquellas en las que los valores que pueden tomar no son numéricos, sino palabras o expresiones que representan un intervalo cuantitativo borroso, es decir, un conjunto borroso definido. Este enfoque se asemeja más al lenguaje utilizado por las personas para describir características específicas. Por ejemplo, recordando el caso de la temperatura en la habitación, la variable lingüística sería la temperatura, cuyos valores eran fría, templada o cálida.

4.2.3 Funciones de membresía

Las funciones de pertenencia son las que definen a los conjuntos borrosos, ya que establecen los límites imprecisos de estos conjuntos y determinan el grado de pertenencia de un elemento al mismo. Entre los diferentes tipos de funciones de pertenencia se encuentran la triangular, trapezoidal, gaussiana, sigmoidal y la de forma de campana. Las funciones triangulares y trapezoidales son las más utilizadas debido a su simplicidad y naturaleza lineal, aunque en ciertos casos es más adecuado recurrir a otras funciones, dependiendo de los requisitos específicos [9].

Para una variable lingüística, será necesario definir tantas funciones de pertenencia como conjuntos borrosos sean necesarios para abarcar todos los posibles valores del universo de discurso de esa variable. Esto significa que cada función representará un rango de valores cuantitativos asociado a esa variable en el contexto del caso de estudio.

La mejor manera de mostrar la idea es mediante un ejemplo, por eso, haremos uso del caso de la temperatura en la habitación.

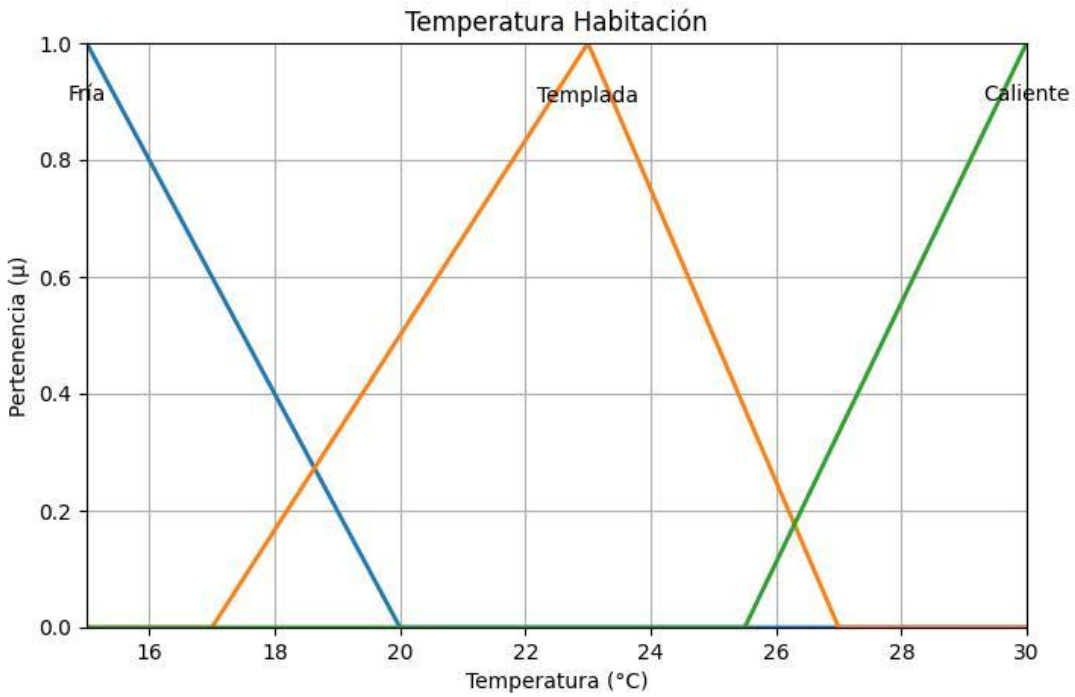


Figura 4-1. Ejemplo de funciones de membresía (MF) para el caso de la temperatura en una habitación

Como se puede observar, se han definido tres funciones de pertenencia que corresponden a los valores de la variable lingüística temperatura en el ejemplo. En este caso, se han utilizado funciones de pertenencia triangulares, aunque las funciones correspondientes a frío y cálido no presentan una arista definida en sus extremos. Esto se debe a que se ha querido otorgar el mayor grado de pertenencia a los valores más extremos: todas las temperaturas cercanas a 15°C son consideradas frías, mientras que las superiores a 30°C se clasifican como cálidas. También se puede apreciar cómo hay regiones en las que dos funciones de pertenencia se solapan, reflejando el hecho de que no existe un valor exacto que marque dónde termina un rango y comienza el otro.

Existen diferentes métodos para definir las funciones de pertenencia, siendo uno de ellos la intuición o el conocimiento de un experto en el sistema, siendo estos los más comunes. Otros métodos incluyen el uso de redes neuronales o algoritmos genéticos [10].

4.2.4 Reglas borrosas

Para diseñar un controlador borroso, es necesario definir un conjunto de reglas borrosas basadas en el conocimiento experto sobre el ámbito del problema en estudio. Estas reglas conforman una base de conocimiento que guía el funcionamiento de la lógica borrosa. Las reglas se expresan en formato IF-THEN, estableciendo relaciones entre los conjuntos borrosos que aparecen en las premisas de la regla y el conjunto borroso que determina su conclusión [8]. La estructura general de estas reglas es la siguiente:

$$IF < proposición\ borrosa > THEN < proposición\ borrosa > \quad (5\text{-}1)$$

Las proposiciones borrosas pueden clasificarse como atómicas o compuestas [11]. Una proposición atómica consiste en una asignación simple, como “x es A”, donde x representa una variable lingüística y A es un conjunto borroso definido mediante una función de pertenencia, por ejemplo, una función triangular. Por otro lado, las proposiciones borrosas compuestas combinan varias proposiciones atómicas utilizando operadores lógicos

como *and, or y not.* Un ejemplo de regla con una proposición compuesta podría ser: "Si X es A y Y es B, entonces Z es C", donde X, Y , Z son variables lingüísticas (X e Y son entradas, mientras que Z es la salida), y A, B, C son conjuntos borrosos definidos para dichas variables.

Estas reglas permiten establecer relaciones entre las entradas del controlador y las salidas que este genera, formando así la base de conocimiento mencionada previamente. Todas las reglas están formuladas lingüísticamente, sin emplear valores numéricos, lo que hace bastante simple plantearlas para una persona. Por ejemplo, si se busca ajustar la intensidad de la iluminación en una habitación dependiendo de la cantidad de luz natural, una posible regla sería: "Si Luz_natural es Baja, entonces Intensidad_luz es Alta".

4.2.4.1 Operaciones lógicas

Como se ha mencionado, las proposiciones en este tipo de reglas pueden formarse mediante operaciones lógicas entre conjuntos borrosos, por lo que es importante entender cómo funcionan dichas operaciones. Las tres operaciones lógicas más comunes son *and, or y not*. También se puede comprender el propio *then* (implicación) al que se le suelen aplicar métodos semejantes que a *and.*

Existen diferentes métodos que pueden emplearse para implementar las operaciones lógicas en lógica borrosa, y su elección depende del caso de estudio o las especificaciones del sistema. Entre los métodos más comunes para la operación *and* se encuentran el mínimo *(min)* y el producto *(prod)*, mientras que para *or* se utilizan el máximo *(max)* y la suma acotada *(bsum)*. Por su parte, para la operación *not* se suele emplear el complemento estándar $1 - \mu(x)$. En el caso de las reglas *then*, los métodos más utilizados son el producto y el mínimo, dependiendo del enfoque que se quiera dar al cálculo del grado de pertenencia en la inferencia borrosa. En este proyecto, nos centraremos en los métodos específicos seleccionados para implementar estas operaciones, los cuales corresponden a la siguiente tabla:

Tabla 4-1. Operaciones lógicas del controlador

And	***Or***	***Not***	***Then***
Min	***Max***	$1 - \mu(x)$	***Min***

Una vez aclarado esto, en la siguiente figura se explican cada una de ellas:

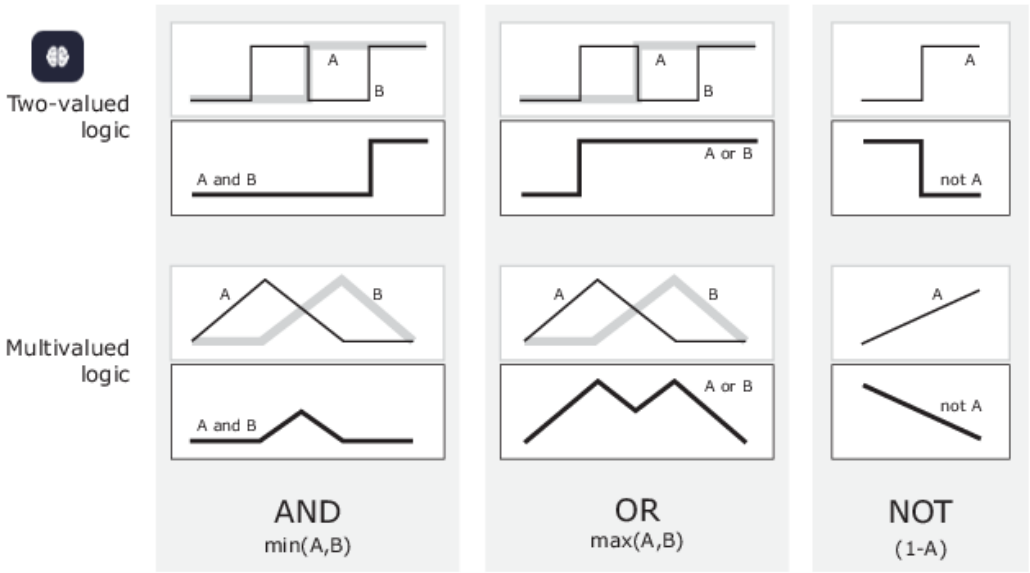


Figura 4-2. Operadores lógicos aplicados a conjuntos borrosos [12].

4.3 Controlador borroso

Haciendo un repaso de los conceptos mencionados previamente, se procederá a explicar en detalle cada uno de ellos y el funcionamiento de la lógica borrosa. Como se ha señalado, esta lógica es una herramienta que permite resolver problemas donde el objeto de estudio no puede describirse de forma precisa mediante una función de transferencia, pues es muy no lineal (como es nuestro caso); o presenta una gran complejidad. Esto se logra utilizando un lenguaje más cercano al razonamiento humano, que no asigna valores absolutos en situaciones determinadas, como el ejemplo de la temperatura en la habitación. Por esta razón, se denomina lógica borrosa o borrosa, ya que existen elementos que no pueden clasificarse dentro de los conjuntos clásicos debido a la falta de límites bien definidos. Este desafío fue abordado mediante la creación de los conjuntos borrosos, la base de la lógica que abordaremos.

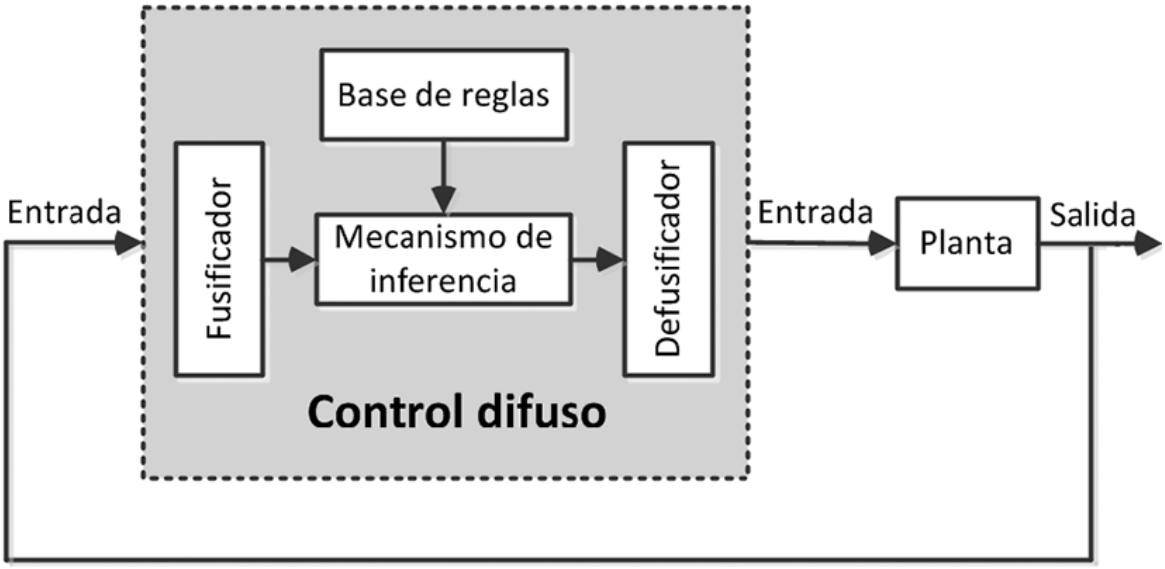


Figura 4-3. Esquema de un controlador borroso [13].

Para aclarar el funcionamiento de controlador se hablará de cada una de las etapas por las que se pasa en él expuesta en la figura.

4.3.1 Fuzzificación

El proceso de fuzzificación consiste en transformar las señales de entrada, que son valores numéricos, en señales borrosas. Esto se logra definiendo los conjuntos borrosos asociados a las variables lingüísticas que representan las entradas del sistema. A través de las funciones de pertenencia correspondientes, se determina el grado de pertenencia de cada entrada a los conjuntos borrosos, lo cual es fundamental para la evaluación de las reglas borrosas en el sistema [14].

4.3.2 Base de reglas

La base de reglas consiste en establecer un conjunto de reglas del tipo IF-THEN, las cuales, como se mencionó anteriormente, se construyen a partir del conocimiento o experiencia de un experto en el ámbito de estudio, utilizando razonamiento borroso.

Dependiendo de la forma de la proposición en el consecuente, las reglas pueden clasificarse en dos tipos principales. Si el consecuente utiliza conjuntos borrosos, se trata de reglas del tipo **Mamdani**. Por otro lado, si el consecuente está definido mediante una ecuación lineal o una función matemática, corresponde a reglas del tipo **Takagi-Sugeno** [11].

4.3.3 Mecanismos de inferencia

El mecanismo de inferencia es el proceso mediante el cual se evalúan las entradas del sistema en función de todas las reglas definidas, generando una salida en formato borroso.

Dentro de este proceso, la implicación de las reglas consiste en generar el consecuente de una regla a partir de su antecedente. Para ello, se utilizan distintos métodos de implicación (*then*), antes mencionados.

El método del mínimo, que es el empleado en el presente trabajo, recorta o trunca el conjunto borroso de salida, limitando los grados de pertenencia según el valor más bajo indicado por el antecedente. Por otro lado, el método del producto preserva la forma original de la función de pertenencia escalándola; esto se logra multiplicando cada grado de pertenencia de la función de salida por el grado de pertenencia indicado por el antecedente [14].

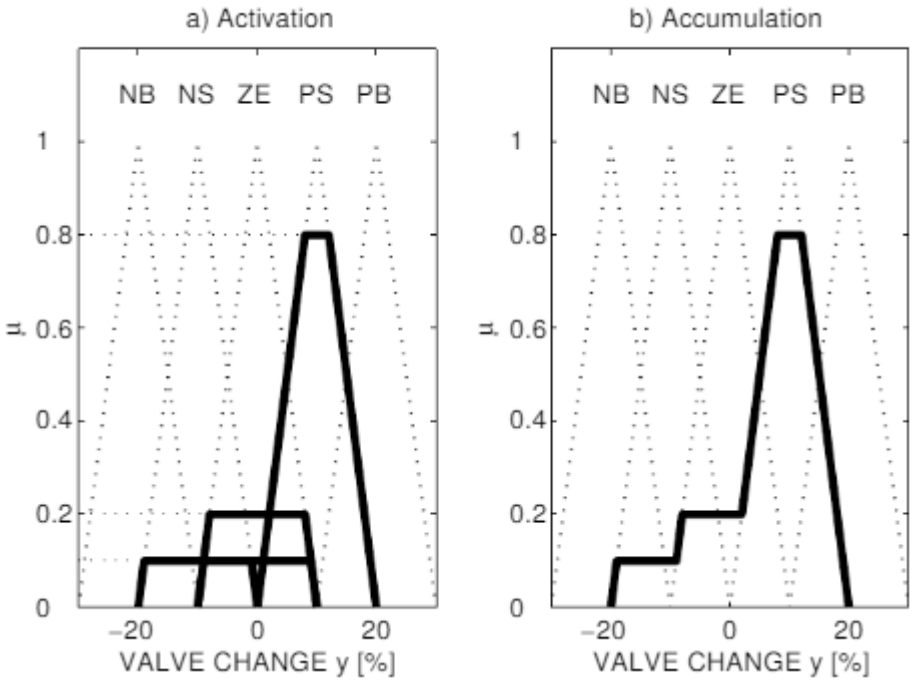


Figura 4-4. Ejemplo de evaluación de reglas [14].

En la Figura 5-4 se pueden ver dos pasos:

- **Activación de la regla:** Es el paso ya mencionado previamente, en el cual, el consecuente es truncado con respecto al mínimo de los antecedentes.
- **Acumulación:** Este paso depende del método de agregación seleccionado, en el caso que nos concierne, se aplica el método *max*, que como se puede apreciar es la selección de la regla con mayor grado de pertenencia.

4.3.4 Defuzzificación

La defuzzificación es el proceso inverso a la fuzzificación, en el cual la salida borrosa generada por el mecanismo de inferencia se convierte en un valor numérico que representa la salida final del controlador (*crisp)*. Existen diversos métodos para llevar a cabo este proceso, entre los más destacados se encuentran los siguientes:

- **Centroide o centro de gravedad *(center of areas)*:** Este método calcula el valor numérico de salida como el centroide del conjunto borroso resultante tras la agregación. Es el más utilizado en aplicaciones de control debido a su precisión y consistencia [14] [8]. Este método es el que ha sido empleado en el presente trabajo.

- **Bisectriz o áreas iguales *(equal areas)*:** Determina el valor numérico como aquel que divide el conjunto borroso de salida en dos partes con áreas iguales [8]
- **Máximo inferior *(lower maximum)*:** Se selecciona el primer valor del conjunto de salida en el que se alcanza el máximo grado de pertenencia, en caso de haber varios picos máximos [8]
- **Máximo superior *(upper maximum)*:** Similar al anterior, pero se elige el último valor en el que se produce el máximo grado de pertenencia [8]
- **Media de máximos o centro de máximos *(mean of maxima)*:** Este método toma como resultado el punto medio entre el máximo inferior y el máximo superior del conjunto de salida [8].

Cabe aclarar que independientemente del método empleado, el valor del *crisp* se obtiene por una discretización de los puntos de la salida. El caso más simple de exponer es el del centroide:

Tabla 4-2. Ecuaciones del centroide

$$c = \frac{\int x\mu(x)\,dx}{\int \mu(x)\,dx} \tag{5-3}$$

$$c = \frac{\Sigma_i x_i \mu(x_i)}{\Sigma_i \mu(x_i)}$$

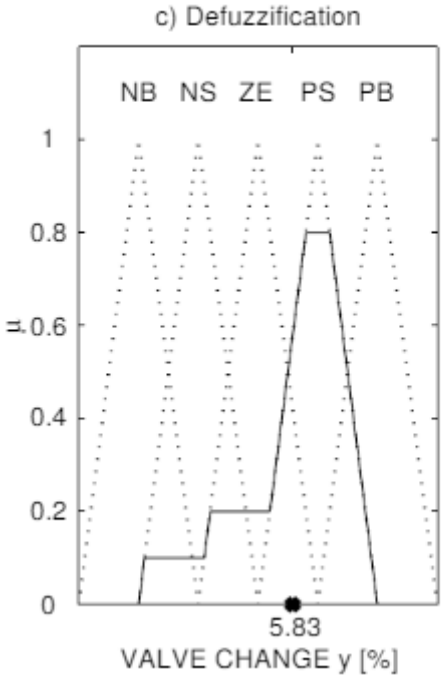


Figura 4-5. Aplicación del método del centroide [14].

Tras todo esto se muestra el resultado de aplicar todos los pasos en un caso particular, el del pago de una propina en base al servicio y la comida:

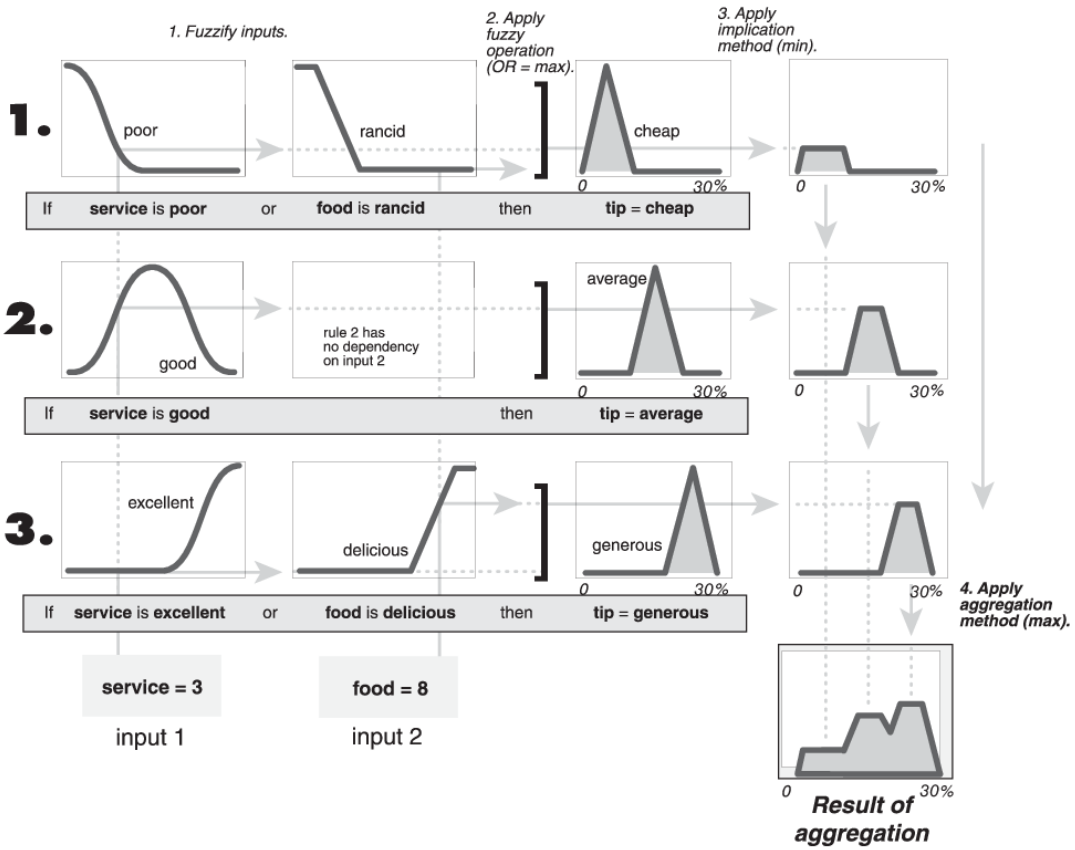


Figura 4-6. Proceso completo de operaciones de un controlador fuzzy [15].

4.4 Controlador borroso en MATLAB

4.4.1 Creación del archive*.fis

Tras haber explicado todos los conceptos necesarios para comprender esta metodología y sus aplicaciones actuales, es pertinente mencionar brevemente cómo trabajar con lógica borrosa en MATLAB. El primer paso consiste en hacer uso de la app FLD. Este editor facilita la creación y gestión de sistemas borrosos, permitiendo configurar entradas, salidas, reglas y funciones de pertenencia de forma visual.

Una vez completado el diseño, se puede guardar el sistema borroso como un archivo con extensión *.fis en el menú, simplemente guardándolo como archivo dentro de la carpeta de trabajo. El archivo es en esencia un archivo de texto plano en el que se contiene toda la información del controlador, como el tipo, las reglas, las MFs, etc.

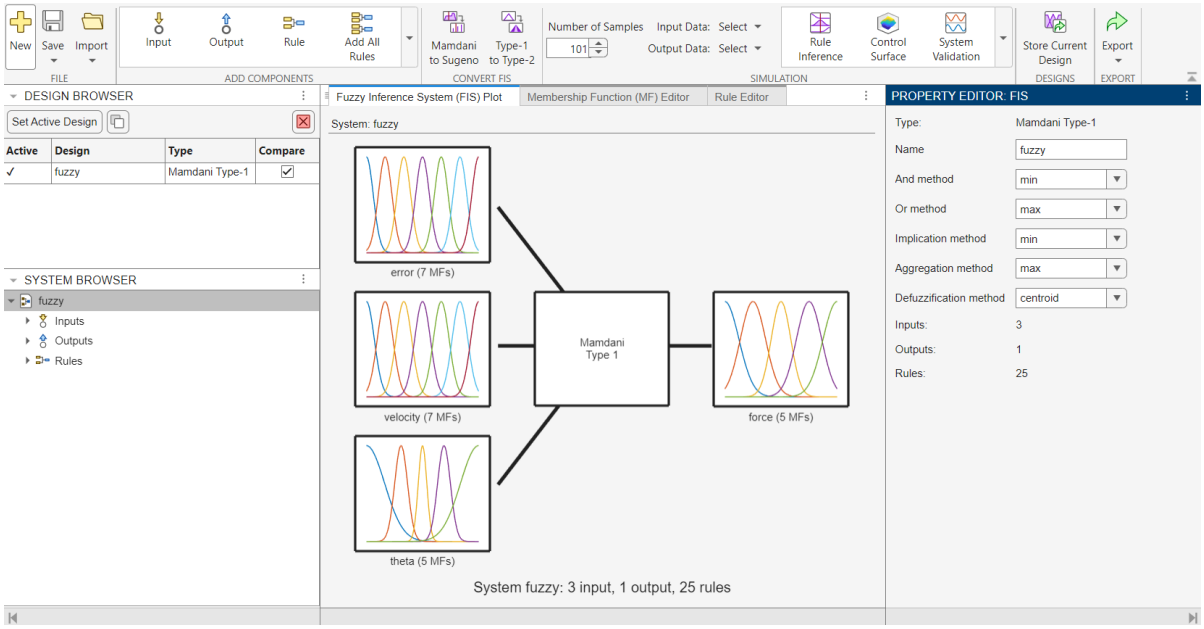


Figura 4-7. FLD

4.4.2 I/O y funciones de membresía

El siguiente paso consiste en definir las variables de entrada (inputs) y salida (Output) del controlador borroso, así como sus correspondientes MFs. Esto se realiza seleccionando la variable deseada en el FLD, lo que abrirá automáticamente el editor de funciones de membresía (Figura 5-8).

En este editor, se puede configurar el tipo de función de membresía que mejor se adapte al comportamiento de las variables del sistema. Entre las opciones disponibles se encuentran las funciones triangular, trapezoidal, sigmoidal y gaussiana, entre otras. La elección de la función dependerá de las características y requisitos específicos del controlador que se esté diseñando.

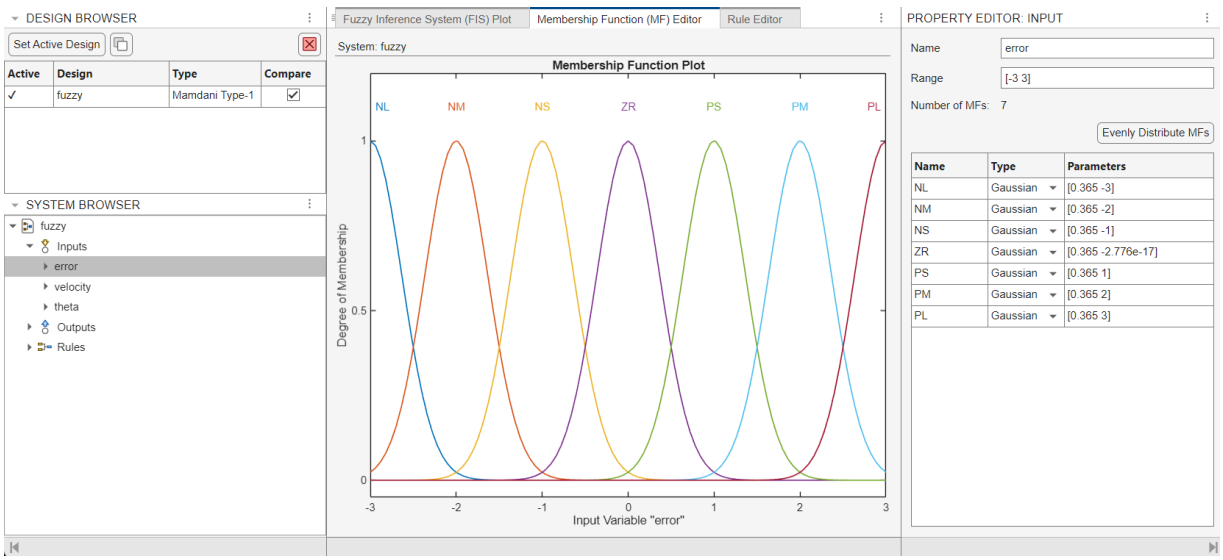
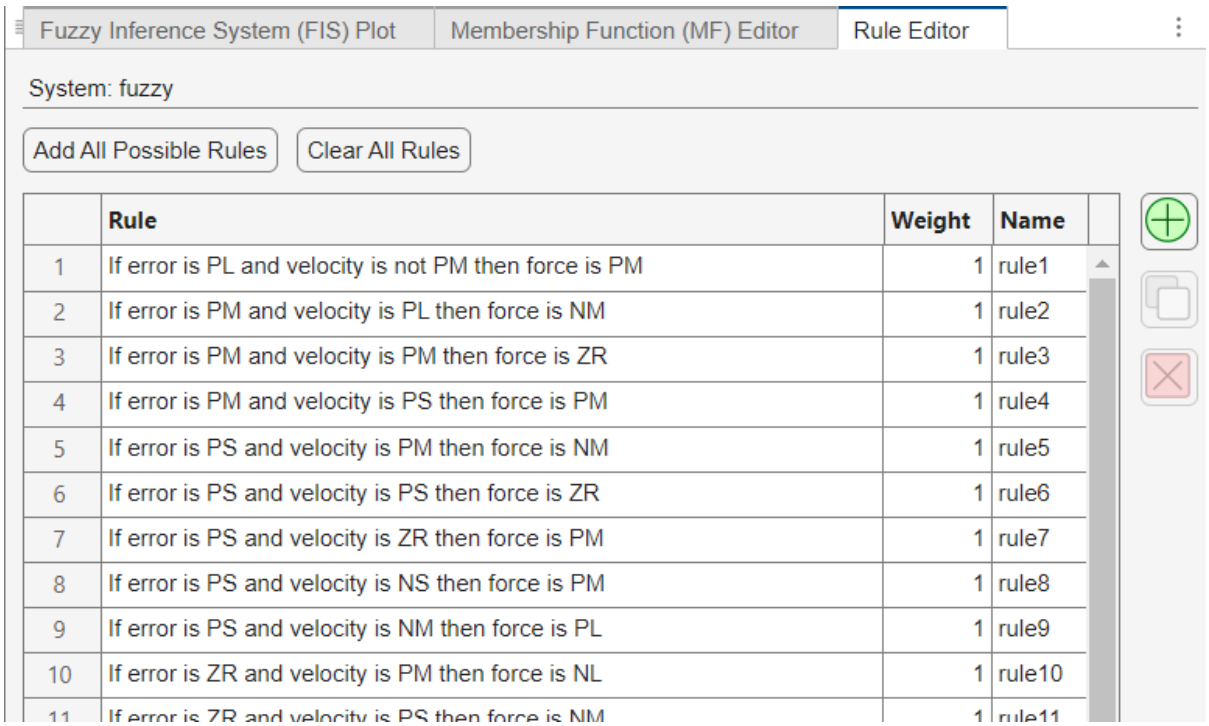


Figura 4-8. Editor de MFs

4.4.3 Reglas de inferencia

Una vez configuradas todas las variables del controlador borroso, el siguiente paso es definir las IRs que determinarán su comportamiento. Para ello, se debe hacer clic en el cuadro de "Rule Editor" de la ventana del FLD (Figura 3-8).

Como se mencionó en apartados anteriores, estas reglas siguen el formato "IF-THEN", y se pueden crear tantas como sean necesarias, considerando que el número máximo está limitado por la cantidad de variables y funciones de membresía definidas. Un aspecto relevante es la posibilidad de asignar un peso (*weight*) o importancia a cada regla. Este valor numérico debe estar entre 0 y 1, permitiendo que una regla tenga menos influencia que el resto si se asigna un peso inferior a 1. No obstante, es habitual que todas las reglas se configuren con el mismo nivel de importancia, siendo este el caso del controlador generado.



	Rule	Weight	Name
1	If error is PL and velocity is not PM then force is PM	1	rule1
2	If error is PM and velocity is PL then force is NM	1	rule2
3	If error is PM and velocity is PM then force is ZR	1	rule3
4	If error is PM and velocity is PS then force is PM	1	rule4
5	If error is PS and velocity is PM then force is NM	1	rule5
6	If error is PS and velocity is PS then force is ZR	1	rule6
7	If error is PS and velocity is ZR then force is PM	1	rule7
8	If error is PS and velocity is NS then force is PM	1	rule8
9	If error is PS and velocity is NM then force is PL	1	rule9
10	If error is ZR and velocity is PM then force is NL	1	rule10
11	If error is ZR and velocity is PS then force is NM	1	rule11

Figura 4-9. Editor de reglas de inferencia

4.4.4 Visualizador de reglas

Después de definir las reglas lógicas y guardar el archivo, la configuración del controlador borroso queda completa y está listo para implementarse en Simulink. No obstante, antes de llevarlo a cabo, es posible evaluar el comportamiento de las reglas mediante dos visualizadores ("Rule inference" y "Control Surface") integrados en el FLD. Estos permiten observar gráficamente cómo interactúan las variables de entrada con las reglas establecidas y cómo se genera la salida borrosa, lo que resulta útil para validar el diseño antes de su implementación final.

Como se puede observar, el visualizador presenta varios gráficos en función de las entradas y salidas del controlador. En cada columna correspondiente a las entradas, se visualizarán tantos gráficos como reglas se hayan configurado. Además, en cada columna de salidas, habrá un gráfico adicional que muestra el valor resultante de la salida borrosa, el cual estará representado por una línea roja más gruesa.

Esta herramienta es de gran utilidad, ya que permite verificar de forma aislada si el controlador funciona según lo esperado, sin interferencias de otros elementos del modelo en Simulink. Si el controlador no produce los resultados deseados al simularlo en el entorno de Simulink, el visualizador ayuda a determinar si el problema radica en la implementación técnica o en la validez de las reglas o funciones de membresía utilizadas. Esto facilita la corrección de errores en etapas tempranas del diseño y prueba del sistema.

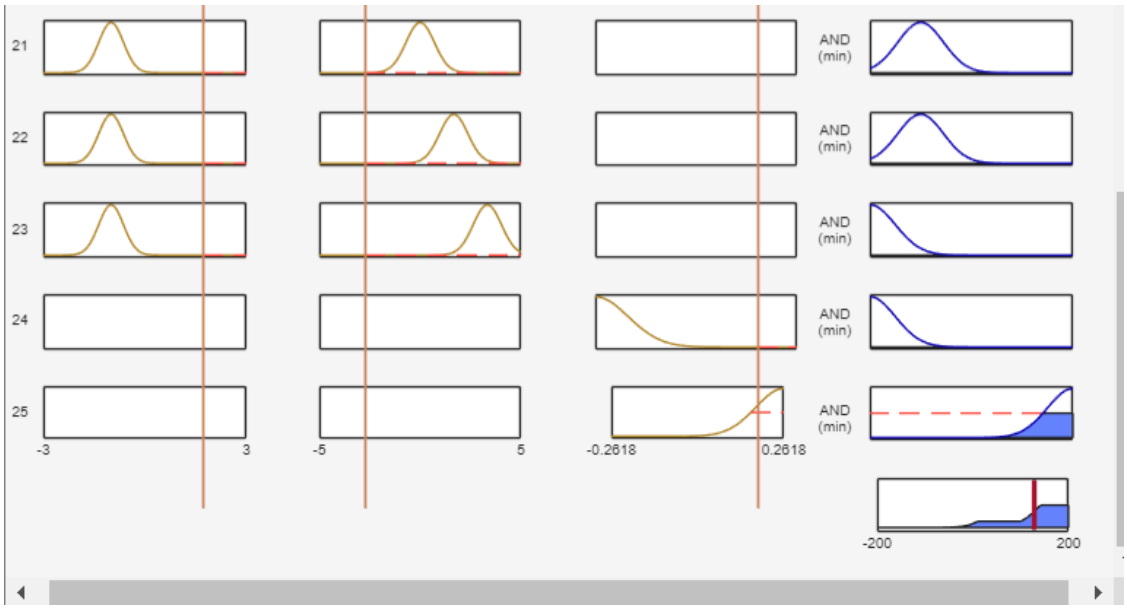


Figura 4-10. Rule Inference

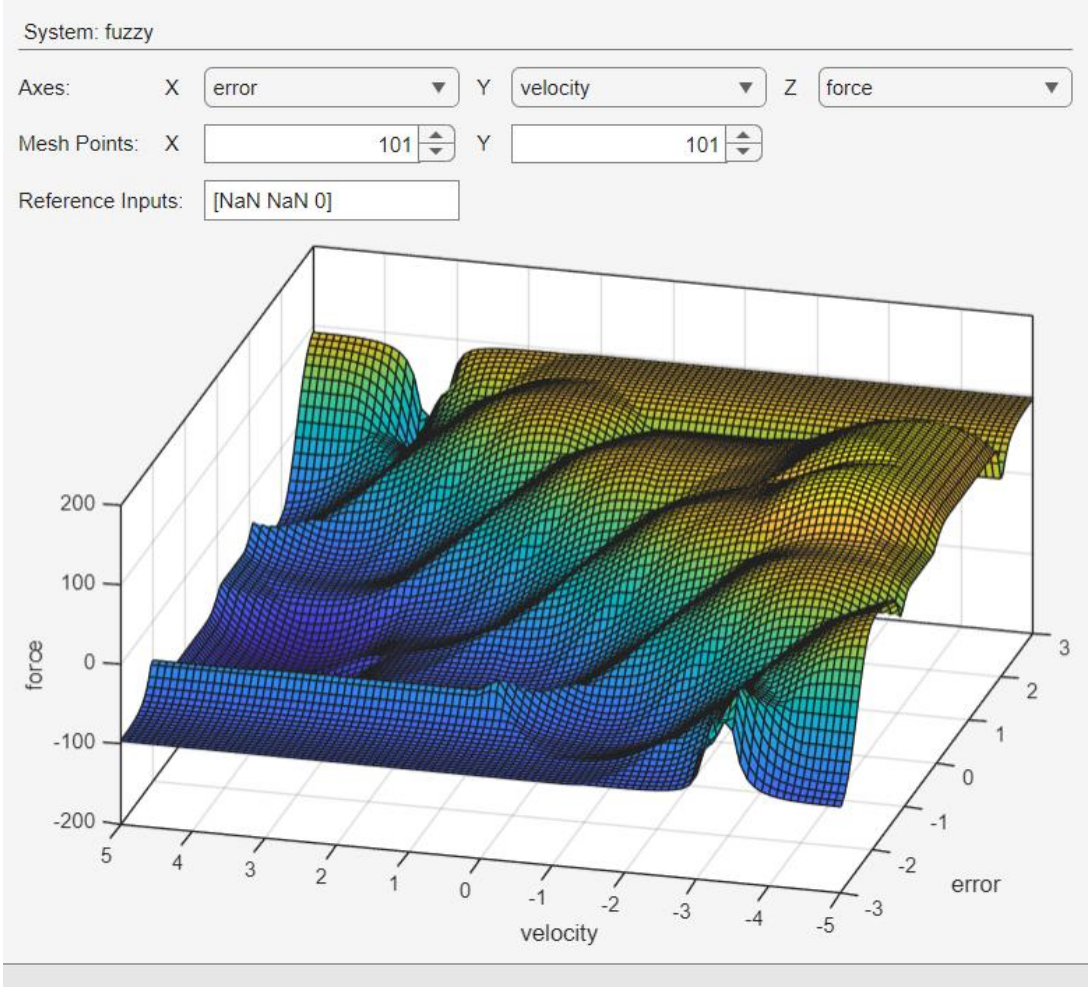


Figura 4-11. Control Surface

4.4.5 Implementación del controlador en Simulink

Para integrar el archivo de controlador borroso que ha sido creado previamente en FLD, simplemente hay que utilizar el bloque FLC. En el interior de este bloque, se debe especificar el nombre del archivo con su extensión *.fis entre comillas simples, incluyendo la ruta completa si es que no se encuentra en la misma carpeta que el modelo.

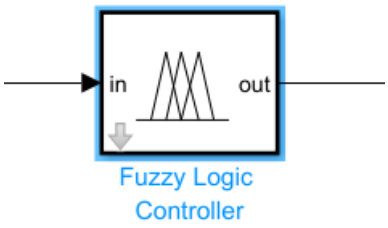


Figura 4-12. FLC

5 ANÁLISIS DEL MODELO E IMPLEMENTACIÓN

Antes de implementar un sistema de control en un entorno real, es crucial realizar simulaciones que permitan analizar su comportamiento y prever posibles problemas. Las simulaciones proporcionan un entorno seguro y controlado para probar estrategias de control sin arriesgar daños a los equipos ni comprometer la seguridad de las operaciones. Esto es especialmente importante cuando se trabaja con sistemas complejos, donde los errores pueden ser costosos o peligrosos.

La precisión en el modelado del sistema es un factor fundamental para garantizar que los resultados de las simulaciones sean representativos de la realidad. Un modelo que no capture correctamente las dinámicas del sistema puede llevar a conclusiones erróneas sobre su comportamiento y, por ende, a estrategias de control ineficaces. Por esta razón, es imprescindible dedicar tiempo y recursos a desarrollar un modelo que refleje lo más fielmente posible las características físicas y operativas del sistema. Además, un modelado preciso influye directamente en la elección de las estrategias de control. Los modelos permiten evaluar diferentes enfoques y determinar cuál se adapta mejor a las particularidades del sistema. Por ejemplo, en sistemas con comportamientos no lineales o dinámicas variables, métodos de control tradicionales como los PID pueden no ser suficientes, y es aquí donde estrategias como la lógica borrosa cobran relevancia. El modelo actúa como un laboratorio virtual en el que se pueden ajustar parámetros, identificar limitaciones y optimizar el rendimiento del controlador antes de trasladarlo al entorno real.

En este contexto, se desarrollará un modelo detallado del sistema, el cual servirá como base para diseñar, ajustar y validar el controlador borroso propuesto en este estudio. A través de este modelo, se podrá evaluar el desempeño del controlador, asegurando que las soluciones implementadas sean efectivas y seguras para su aplicación final.

5.1 Esquema del modelo

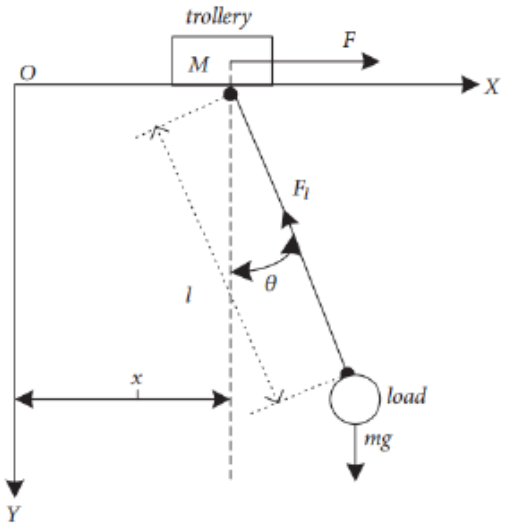


Figura 5-1. Esquema de fuerzas [16]

En la figura se muestra el esquema en el que se basa el modelo para, a través de las ecuaciones de Euler-Lagrange, establecer la dinámica del sistema a controlar. En el esquema se pueden diferenciar los siguientes elementos:

- **Trolley *(M)*:** Es el carro móvil de la grúa que se mueve para transportar la carga.
- **Rope *(l)*:** Es la cadena o cuerda que une la carga al carro. En este caso se asume inelástica.
- **Load *(m)*:** Es la carga a transporta de un punto *a*, a otro punto *b* del eje *x*.
- **Position *(x)*:** Es la posición relativa con respecto al eje *x* del sistema de coordenadas.
- **Angle *(θ)*:** Es el ángulo que forma la carga con respecto al eje *y* del sistema de coordenadas.
- **Force *(F)*:** Es la fuerza que induce el movimiento sobre el carro. Esta fuerza sería la que ejercería el controlador.
- **COF *(μ)*:** Es el coeficiente de rozamiento dinámico existente entre el carro y la superficie por la que se desplaza.

5.2 Modelo dinámico en el dominio del tiempo

Como se ha comentado previamente el método empleado para obtener las ecuaciones del sistema físico, son las ecuaciones de Euler-Lagrange. Se escogió este método pues resulta más esquemático y fácil de aplicar en un sistema complejo como este que la aplicación directa de las leyes de newton sobre los diferentes cuerpos del sistema.

5.2.1 Ecuaciones de Euler-Lagrange

En lugar de utilizar las fuerzas y leyes de Newton directamente, las ecuaciones de Lagrange trabajan con **energías**: la energía cinética *T* y la energía potencial *V*. Este enfoque es particularmente útil para sistemas con múltiples grados de libertad y restricciones porque permite describir el movimiento en términos de coordenadas generalizadas.

5.2.1.1 Principio fundamental

Las ecuaciones de Lagrange se derivan del **principio de Hamilton**, que establece que el sistema sigue una trayectoria que minimiza (o hace estacionaria) la acción *S*, definida como:

$$S = \int_{t_1}^{t_2} L\, dt \tag{5–1}$$

donde *L* es la **Lagrangiana**:

$$L = T - V \tag{5–2}$$

Donde *T* es la energía cinética y *V* es la energía potencial.

5.2.1.2 Fórmula general

Para un sistema con coordenadas generalizadas q_i y sus derivadas temporales $\dot{q}_i$, las ecuaciones de Lagrange se escriben como:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}i}\right) - \frac{\partial L}{\partial q_i} = 0, i = 1,2, \dots, n \tag{5–3}$$

Los términos son:

- $\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}i}\right)$: Momento generalizado asociado a q_i.
- $\frac{\partial L}{\partial q_i}$: Fuerza generalizada correspondiente.

5.2.1.3 Ventajas y razón de su uso

- **Flexibilidad**: Permite trabajar con coordenadas que no son necesariamente cartesianas, lo que es útil en sistemas con restricciones (por ejemplo, movimiento en superficies curvas).
- **Reducción de complejidad**: Evita calcular fuerzas de restricción explícitamente, pues estas se incorporan automáticamente en las ecuaciones.

5.2.2 Ecuaciones del sistema

Lo primero es establecer las coordenadas en las que nos basaremos para exponer las ecuaciones. Las coordenadas se encuentran en la siguiente tabla:

Tabla 5-1. Coordenadas de ecuaciones E-L

q_1	q_2
x	θ

Describimos la posición de ambos cuerpos:

$$\begin{cases} x_m = x + L\sin(\theta) \\ y_m = -L\cos(\theta) \end{cases} \qquad \begin{cases} \dot{x}_m = \dot{x} + L\dot{\theta}\cos(\theta) \\ \dot{y}m = L\dot{\theta}\sin(\theta) \end{cases} \tag{5–4}$$

$$x_M = x \qquad \dot{x}_M = \dot{x}$$

Establecemos la energía cinética y potencial del sistema para obtener la lagrangiana:

$$T = \frac{1}{2}(M\dot{v}_M^2 + m\dot{v}_m^2) = \frac{1}{2}\left(M\dot{x}^2 + m\left(\dot{x}^2 + 2L\dot{\theta}\dot{x}\cos(\theta) + L^2\dot{\theta}^2\right)\right) \tag{5–5}$$

$$V = -mgL\cos(\theta)$$

$$L = T - V = \frac{1}{2}\dot{x}^2(M+m) + mL\dot{\theta}\dot{x}\cos(\theta) + \frac{1}{2}mL^2\dot{\theta}^2 + mgL\cos(\theta)$$

Aplicamos la fórmula de la lagrangiana para cada coordenada:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}}\right) - \frac{\partial L}{\partial x} = F \qquad \ddot{x}(M+m) + mL\left(\ddot{\theta}\cos(\theta) - \dot{\theta}^2\sin(\theta)\right) + \mu\dot{x} = F \tag{5–6}$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) - \frac{\partial L}{\partial \theta} = 0 \qquad mL^2\ddot{\theta} + mL\ddot{x}\cos(\theta) + mLg\sin(\theta) = 0$$

Con esto ya poseemos las ecuaciones de la dinámica del sistema. Como se puede ver el sistema es muy no lineal (términos trigonométricos) y además existe acoplamiento, lo que lo hace más complejo de analizar. Además, es un sistema MIMO por lo que haciéndolo difícilmente representable a través de una función de transferencia

G(s). Por todo ello se ha elegido implementarlo como un SS.

5.3 Espacio de estados

La representación mediante espacio de estados de un sistema posee las siguientes ecuaciones:

$$\dot{x}(t) = Ax(t) + Bu(t) \qquad (5\text{–}7)$$
$$y(t) = Cx(t) + Du(t)$$

Pero esto solo es aplicable a sistemas linealizados, algo que en nuestro caso se puede hacer aplicando unas ciertas simplificaciones como las siguientes:

$$\cos(\theta) \sim 1 \qquad (5\text{–}8)$$

$$\sin(\theta) \sim \theta$$

$$\dot{\theta}^2 \sim 0$$

Estas aproximaciones surgen de la idea de que el ángulo en el sistema está limitado en un rango reducido $-\varepsilon \leq \dot{\theta} \leq \varepsilon$ pudiendo aproximarse los valores tal y como se puede ver en las ecuaciones (5-8). Esto dejaría un sistema como el siguiente:

$$\ddot{x}(M+m) + mL\ddot{\theta} + \mu\dot{x} = F \qquad (5\text{–}9)$$

$$mL^2\ddot{\theta} + mL\ddot{x} + mLg\theta = 0$$

De estas ecuaciones se obtiene fácilmente la expresión de espacio de estados de nuestro sistema:

$a_{22} = -\frac{\mu}{M}$	(5–10)
$a_{23} = \frac{m}{M}g$	
$a_{42} = \frac{\mu}{ML}$	
$a_{43} = -\frac{(m+M)}{ML}g$	
$b_2 = \frac{1}{M}$	
$b_4 = -\frac{1}{ML}$	
$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & a_{22} & a_{23} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & a_{42} & a_{43} & 0 \end{bmatrix} B = \begin{bmatrix} 0 \\ b_2 \\ 0 \\ b_4 \end{bmatrix}$	
$x(t)^T = [x, \dot{x}, \theta, \dot{\theta}]$	

Este modelo es perfectamente válido ya que la idea subyacente del control es provocar precisamente que el ángulo de la carga este contenido en un rango reducido, pudiendo aplicar así la linealización expuesta. Aun así, el modelo empleado en Simulink no es el expuesto previamente. El modelo empleado mantiene las no linealidades, haciéndolo más preciso que modelo lineal expuesto. La razón por la cual se expone este modelo es porque fue el usado originalmente hasta que se dio con este otro que mantenía las no linealidades.

$$\begin{bmatrix}\dot{\chi}_1\\ \dot{\chi}_2\\ \dot{\chi}_3\\ \dot{\chi}_4\end{bmatrix}=\begin{bmatrix}0 & 1 & 0 & 0\\ 0 & a_{11} & a_{12} & a_{13}\\ 0 & 0 & 0 & 1\\ 0 & a_{21} & a_{22} & a_{23}\end{bmatrix}\begin{bmatrix}\chi_1\\ \chi_2\\ \chi_3\\ \chi_4\end{bmatrix}+\begin{pmatrix}0\\ b_1\\ 0\\ b_2\end{pmatrix}F$$

$$a_{11}=\frac{-\mu}{M+m\sin^2\chi_3},a_{12}=\frac{mg\cos\chi_3\sin\chi_3}{M+m\sin^2\chi_3},$$

$$a_{13}=\frac{ml\chi_4\sin\chi_3}{M+m\sin^2\chi_3},a_{21}=\frac{\mu\cos\chi_3}{l\left(M+m\sin^2\chi_3\right)},$$

$$a_{22}=\frac{-\left(M+m\right)g\sin\chi_3}{l\left(M+m\sin^2\chi_3\right)\chi_3},a_{23}=\frac{-m\chi_4\sin\chi_3\cos\chi_3}{M+m\sin^2\chi_3},$$

$$b_1=\frac{1}{M+m\sin^2\chi_3},b_2=\frac{-\cos\chi_3}{l\left(M+m\sin^2\chi_3\right)}.$$

Figura 5-2. Modelo empleado en Simulink [16]

5.4 Modelo de Simulink

El modelo es simplemente la implementación de las ecuaciones del SS en un bloque de "MATLAB Function" de Simulink, teniendo el siguiente esquema:

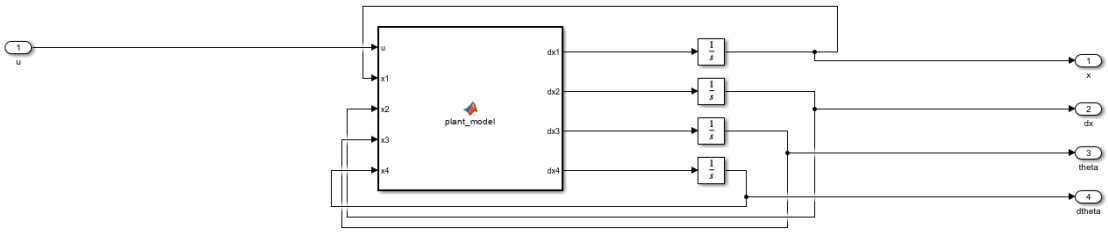


Figura 5-3. Modelo de planta de Simulink

Tiene como entradas el vector de estados $x(t)$ y el valor de salid del controlador $u(t)$, y como salida, la derivada del vector $\dot{x}(t)$, cuyas componentes son integradas para poder usarlas como inputs de los controladores.

```
function [dx1,dx2,dx3,dx4] = plant_model(u,x1,x2,x3,x4)
M = 10;g=9.8;m=5;l=1;mu=0.3;
q = [x1;x2;x3;x4];
a11 = -mu/(M+m*sin(x3)^2); a12 = m*g*cos(x3)*sin(x3)/((M+m*sin(x3)^2)*x3);
a13 = m*l*x4*sin(x3)/(M+m*sin(x3)^2); a21 = mu*cos(x3)/(l*(M+m*sin(x3)^2));
a22 = -(M+m)*g*sin(x3)/(l*x3*(M+m*sin(x3)^2)); a23=-m*x4*sin(x3)*cos(x3)/(M+m*sin(x3)^2);
b1 = 1/(M+m*sin(x3)^2); b2 = -cos(x3)/(l*(M+m*sin(x3)^2));
A = [0 1 0 0;0 a11 a12 a13;0 0 0 1;0 a21 a22 a23];
B = [0;b1;0;b2];

dq = A*q + B*u;
dx1 = dq(1);
dx2 = dq(2);
dx3 = dq(3);
dx4 = dq(4);
```

Figura 5-4. Espacio de estados del sistema

5.5 Conclusión

En resumen, hemos obtenido las ecuaciones de la dinámica del sistema en el dominio del tiempo utilizando las ecuaciones de Euler-Lagrange, lo que permitió describir el comportamiento físico del sistema de manera detallada. Posteriormente, representamos estas ecuaciones en el formato de espacio de estados, dado que el sistema analizado es de tipo MIMO (Multiple-Input Multiple-Output). Este paso fue esencial para preparar el modelo para su implementación en estrategias de control.

Para simplificar el análisis y diseño, se aplicaron diversas simplificaciones que nos permitieron linealizar el modelo. Esto resultó en la obtención de las matrices del sistema linealizado, que son cruciales para el diseño de controladores que asuman comportamientos lineales, como los clásicos PIDs o basados en el lugar de las raíces. Sin embargo, esta simplificación implica una pérdida de precisión en la descripción del comportamiento del sistema, especialmente en regímenes de operación alejados del punto de equilibrio considerado.

Finalmente, se recurrió a un modelo alternativo extraído de la literatura científica [16] que mantiene las no linealidades inherentes al sistema. Este modelo ofrece una representación más precisa y realista del comportamiento dinámico, especialmente útil para simulaciones y diseño de controladores que aprovechen características no lineales, como el caso de los controladores borrosos *(FLC)*.

6 CONTROL DE LA CARGA

A lo largo de este trabajo, hemos abordado todos los aspectos fundamentales para la correcta implementación de un FLC. Inicialmente, introdujimos los conceptos básicos de la **lógica borrosa**, explorando cómo esta técnica se inspira en el razonamiento humano para manejar incertidumbre y ambigüedad. Posteriormente, construimos un modelo matemático de la grúa, representándola como un **péndulo invertido** para describir su dinámica no lineal. Este modelo fue clave para comprender las fuerzas y momentos involucrados, permitiéndonos derivar las ecuaciones de estado que describen su comportamiento.

Para abordar la representación matemática de la planta, desarrollamos tanto un **modelo linealizado** basado en simplificaciones razonables, como un **modelo no lineal** más preciso tomado de la literatura, lo que nos permitió evaluar las ventajas y limitaciones de cada enfoque en términos de precisión y complejidad computacional.

En este apartado final, nos centraremos directamente en el diseño y comparación de estrategias de control para resolver el problema práctico del manejo de la carga. En particular, analizaremos dos estructuras diferentes:

1. **Control PID-PI en cascada**: Una técnica clásica ampliamente utilizada en la industria, que consiste en un controlador proporcional-integral-derivativo (PID) para la posición y un controlador proporcional-integral (PI) para la oscilación del ángulo del péndulo [16].
2. **Controlador borroso Mamdani tipo 1**: Una solución basada en lógica borrosa que, a través de un sistema de reglas lingüísticas, busca imitar el razonamiento humano para lograr un control preciso y robusto.

El objetivo de este análisis comparativo es evaluar las ventajas y limitaciones de cada estructura de control, destacando aspectos como la precisión, robustez frente a perturbaciones y facilidad de implementación. Esto nos permitirá no solo entender el desempeño de los controladores, sino también proporcionar una solución práctica que pueda implementarse en la grúa puente para optimizar su operación y seguridad.

6.1 Implementación de controlador PID-PI en cascada

Para diseñar un controlador clásico, como un PID, es común partir de la función de transferencia que describe el sistema a controlar. La función de transferencia es una herramienta fundamental en el análisis y diseño de sistemas en el dominio de la frecuencia *(s)*, ya que permite representar la dinámica del sistema de manera simplificada y estandarizada.

En muchas situaciones, sin embargo, no disponemos inicialmente de la función de transferencia, por lo que es necesario obtenerla a través de métodos específicos. Uno de ellos es someter el sistema a una respuesta a escalón, lo cual es especialmente útil cuando se trabaja con un modelo de "caja negra". En este caso, el sistema se trata como un objeto desconocido, y la respuesta obtenida se utiliza para ajustar un modelo aproximado en el dominio de la frecuencia.

Por otro lado, si disponemos de una representación matemática precisa del sistema, como es nuestro caso, podemos derivar directamente la función de transferencia a partir de esta. Esto se logra, por ejemplo, transformando la representación en espacio de estados del modelo al dominio de la frecuencia, utilizando herramientas específicas, como las que ofrece MATLAB con la función *tf*. Este enfoque garantiza una mayor fidelidad en la representación de las dinámicas del sistema, al conservar las características intrínsecas de su modelo original.

6.1.1 Obtención de las funciones de transferencia

De los dos métodos citados solo uno es aplicable a nuestro caso, no tanto porque sea más directo sino por las propias particularidades del sistema. Si sometemos nuestro sistema a una excitación mediante un escalón los resultados son los siguientes:

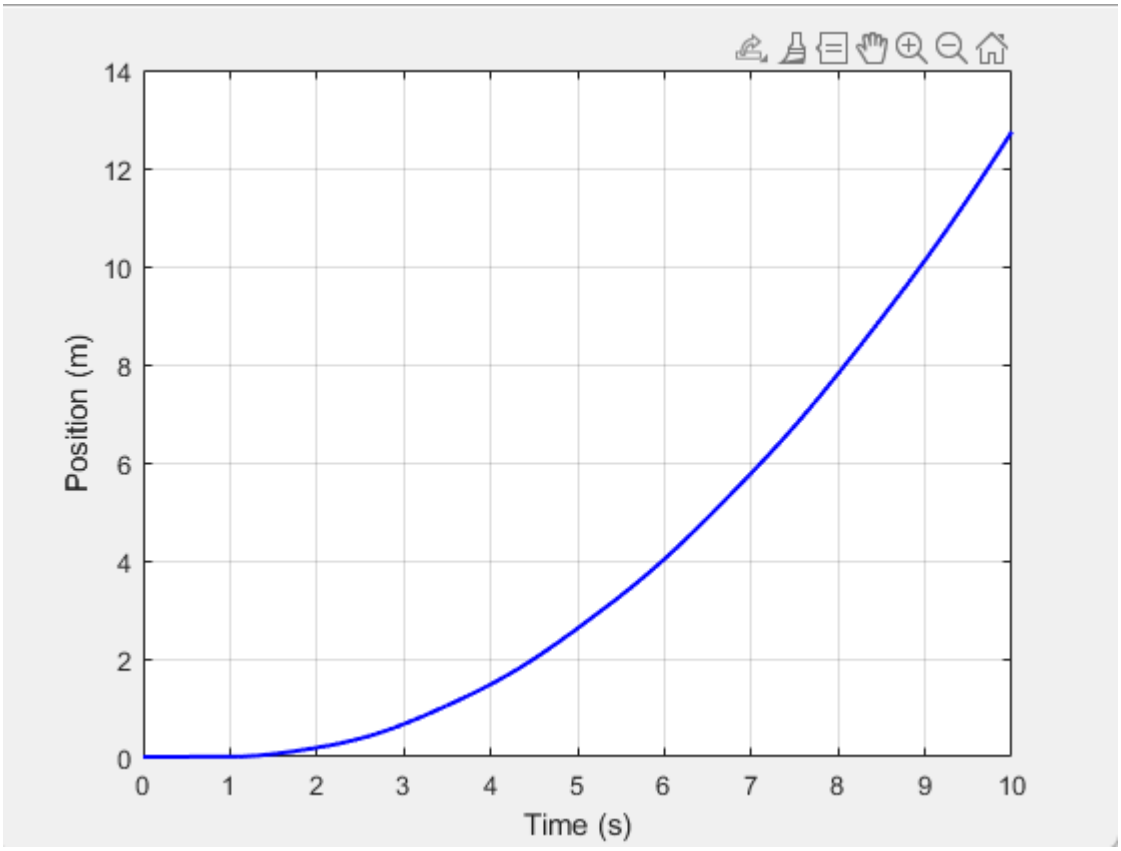


Figura 6-1. Respuesta ante escalón del sistema

Como se puede ver el sistema no llega nunca a estabilizarse en un valor de posición y eso es debido a como este ha sido descrito en la Figura 5-1 donde podemos ver claramente que si sometemos al carro *(trolley)* a una excitación *(force)* mantenida en el tiempo, el sistema se moverá indefinidamente hasta el infinito. Por lo tanto, la única opción que queda es transformar nuestro SS en sus respectivas funciones de transferencia.

```
>> G = tf(res)

G =

  From input to output...
           0.1 s^2 + 7.95e-34 s + 0.981
   1:  -------------------------------------
       s^4 + 0.03 s^3 + 14.71 s^2 + 0.2943 s

         0.1 s^2 - 7.704e-35 s + 0.981
   2:  --------------------------------
       s^3 + 0.03 s^2 + 14.71 s + 0.2943

               -0.1 s + 4.907e-19
   3:  --------------------------------
       s^3 + 0.03 s^2 + 14.71 s + 0.2943

       -0.1 s^2 + 4.907e-19 s + 3.057e-37
   4:  --------------------------------
       s^3 + 0.03 s^2 + 14.71 s + 0.2943
```

Figura 6-2. Funciones de transferencia de cada variable

Estas funciones de transferencia son las que se obtienen tras aplicar la función *tf* de MATLAB al sistema expresado en SS linealizado (ver las ecuaciones (5–10)), obteniendo una función de transferencia por cada uno de los términos del vector de estados $x(t) = [x, \dot{x}, \theta, \dot{\theta}]$. La cuestión es si la respuesta es semejante al modelo empleado que, como se expuso previamente, mantenía las no linealidades.

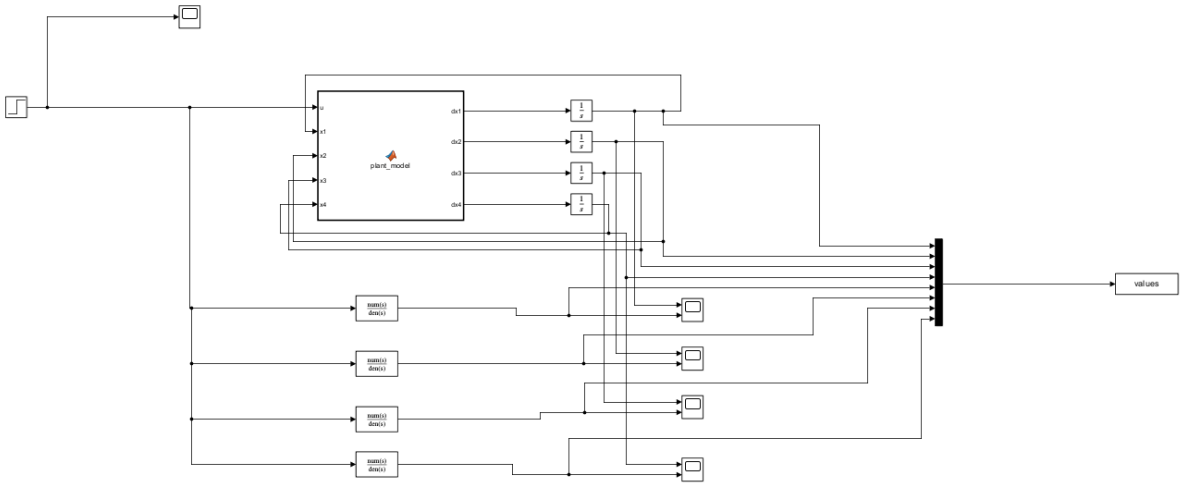


Figura 6-3. Modelo para la comparación

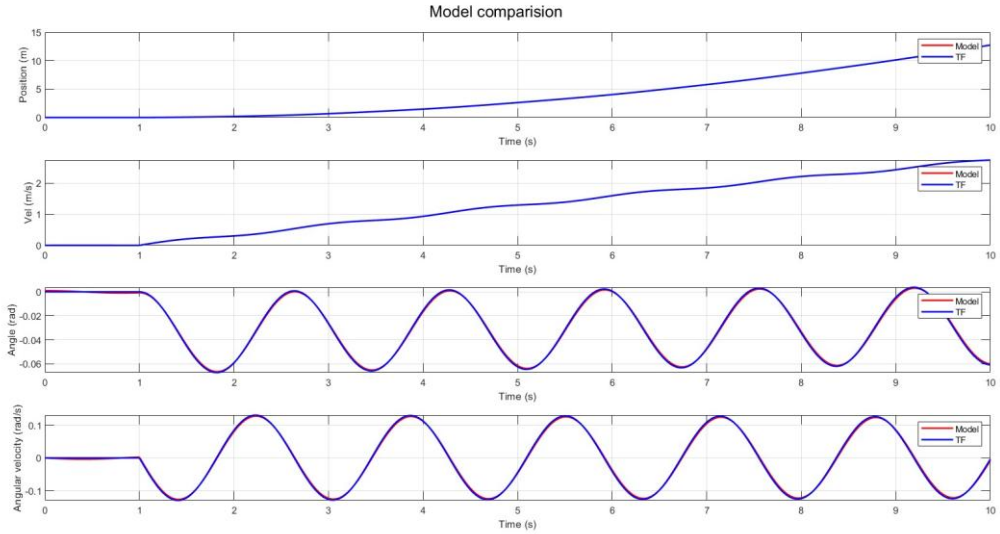


Figura 6-4. Comparación entre modelo no lineal y TFs

Como se puede ver en la figura, la comparativa entre el modelo no lineal (ver Figura 5-2. Modelo empleado en Simulink Figura 5-2) y las diferentes funciones de transferencia es tremendamente satisfactorio pues sus respuestas son prácticamente idénticas. Por todo esto es que se pueden usar para plantear el control en cascada PID-PI, ocupándose del control de la posición y velocidad respectivamente.

6.1.2 Obtención del control en cascada

La primera aproximación de los valores de estos controladores se puede hacer empleando la función *pidtune* de MATLAB, obteniendo lo siguiente:

Tabla 6-1. Valores de las ganancias mediante *pidtune*

$Kp + Ki * \frac{1}{s} + Kd * s$ with Kp = 0.0144, Ki = 8.23e-05, Kd = 0.627	$Kp + Ki * \frac{1}{s}$ with Kp = 0.376, Ki = 0.0228

Aplicando estos valores a los controladores del modelo obtenemos los siguientes resultados:

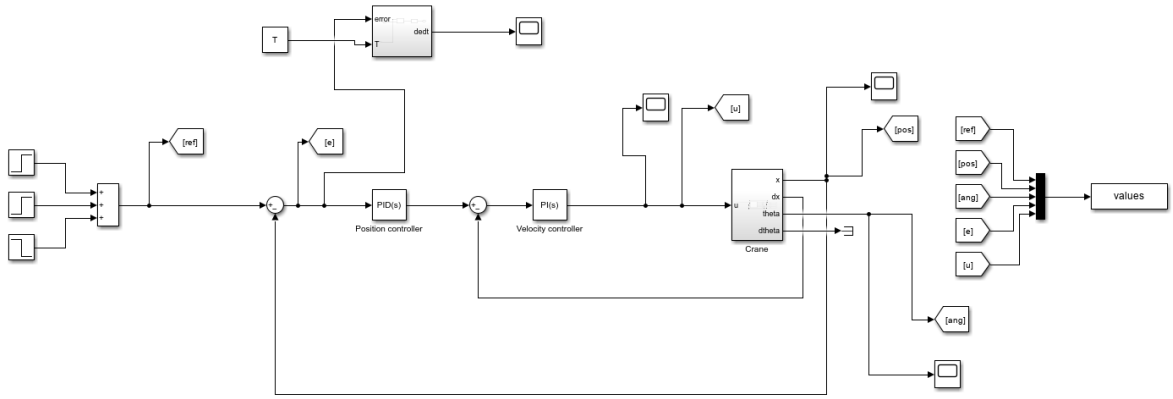


Figura 6-5. Modelo de control clásico

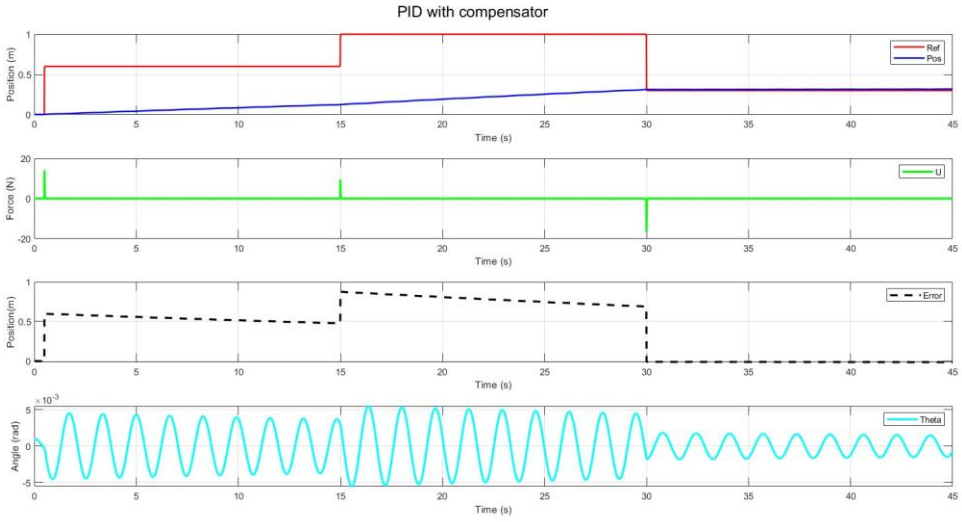


Figura 6-6. Primer resultado de PID-PI

Como se puede observar el control es tremendamente lento, aunque no imprime oscilación a la carga y su error en régimen permanente es prácticamente 0. Esto es debido a los valores tan pequeños de *Kp* y *Ki* de ambos controladores. Por lo tanto, se debe de aumentar dichos valores si se quiere mejorar los tiempos de respuesta. Tras un proceso de "fine-tuning", en el que se tuvo en cuenta los valores de las ganancias de los controladores

expuestos en el trabajo antes citado [16], se adquirieron las siguientes ganancias:

Tabla 6-2. Ganancias del control en cascada

CONTROLADOR	KP	KI	KD
PID	1.1	0.005	1.5
PI	9.46	0.282	

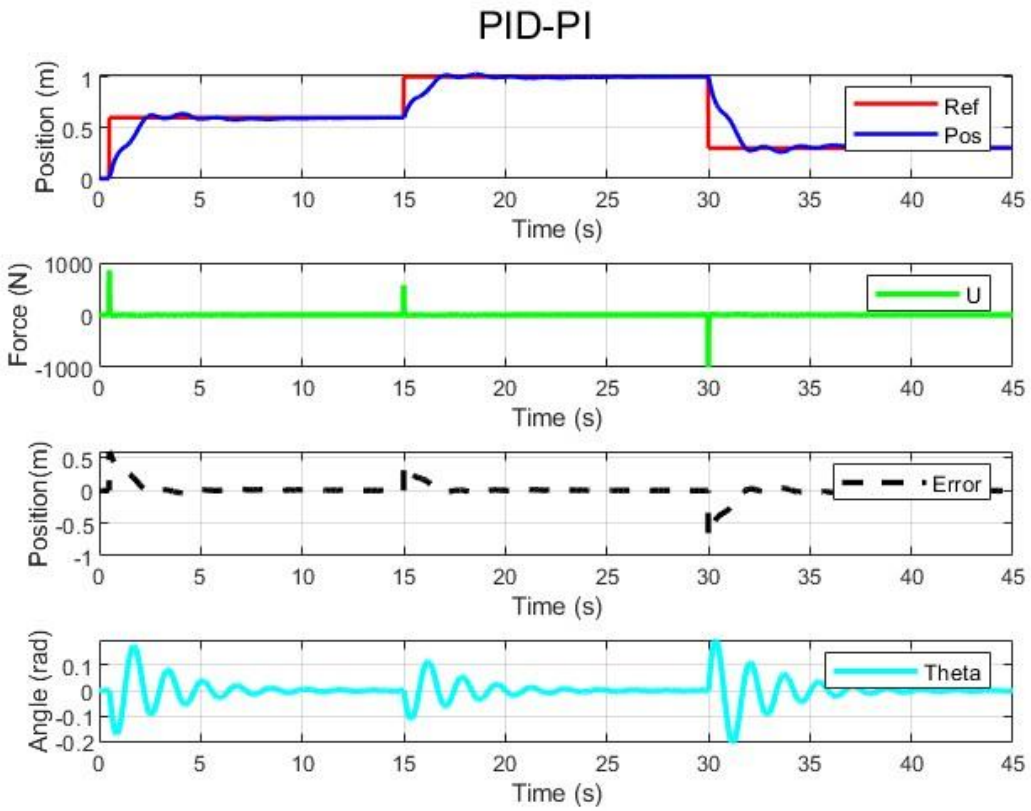


Figura 6-7. Aplicación del control en cascada

Con los cambios aplicados a ambos controladores se puede ver como la respuesta ha mejorado mucho en rapidez a costa de imprimir un mayor balanceo en la cara que llega a ser del orden de 0.2 rad, aunque se atenúa considerablemente a lo largo del tiempo. Además, se puede apreciar que el carro oscila alrededor del valor de referencia de posición en los primeros segundos en los que la alcanza, para luego estabilizarse también y confundirse con el valor de esta.

Con esto se puede dar por finalizado la creación del control en cascada y se puede continuar con la creación del FLC para poder establecer una comparativa.

6.2 FLC basado en PID

Para diseñar un controlador borroso se suele recurrir al conocimiento de alguien experimentado en el manejo del sistema para poder inducir la base de reglas necesarias o, aplicando soluciones más sofisticadas, haciendo uso de redes neuronales *(Neuro-Fuzzy)*. Empero, la primera aproximación del control se hizo empleando FLCs como sustitutos de los controladores convencionales como es el caso del archiconocido PID. La ventaja comparativa del FLC es su no linealidad, que, partiendo de un control lineal de referencia como un PID; puede ser empleada posteriormente para mejorar la respuesta aportada por su "molde" lineal.

6.2.1 Diseño de un FLC basado en un PID

Para poder crear un FLC en base a un PID o cualquiera de sus variantes (PI, PD, etc), se aplican una serie de pasos según el controlador deseado. Para nuestro caso se expondrá la generación del PID y se comentarán los cambios para el caso del controlador PI.

6.2.1.1 Diagrama de bloques del controlador

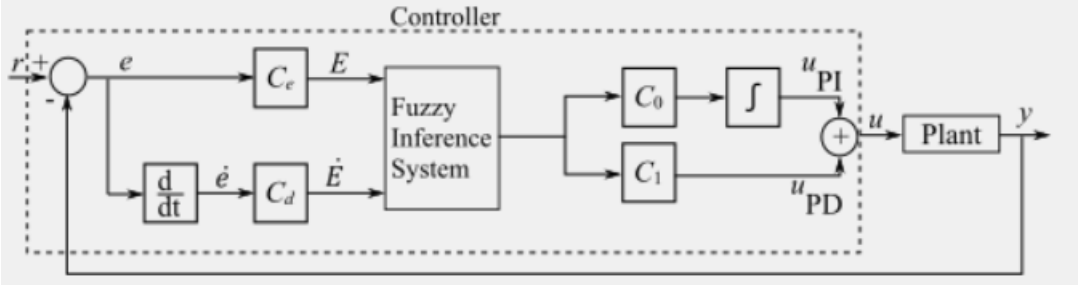


Figura 6-8. Diagrama de bloques de PID

En el diagrama se puede ver que los inputs del controlador son el error *($e(t)$)* y su derivada *($\dot{e}(t)$)* que son afectadas por unas ganancias a la entrada. Posteriormente la salida se desdobla en la componente integral *(PI)* y derivativa *(PD)*, que también poseen unas ganancias. El valor de salida del controlador *(u)* es la suma de ambas componentes. Este sería el caso de un PID completo si todas las ganancias tienen un valor distinto de cero. Para el caso del PI, el valor de la ganancia C1 es 0, dejando solo la acción integral.

6.2.1.2 Adecuación de las ganancias

Para este proceso hay que conocer de donde viene los valores de estas ganancias antes mencionadas, que sirven, entre otras cosas para poder normalizar los inputs y outputs del FLC.

- `Kp = C0 * |Cd| + C1 * |Ce|`
- `Ki = C0 * |Ce|`
- `Kd = C1 * |Cd|`

Figura 6-9. Ecuaciones de las ganancias de un FLC tipo PID [17]

Como se puede ver, partiendo de los parámetros de un PID conocido, se pueden extraer los valores de las 4 ganancias, pues poseemos un SCI, donde tenemos un parámetro libre. Este parámetro es el que establece la normalización de las entradas antes mencionado, como norma general se suele escoger la ganancia Ce, ya que es la que afecta al error que está acotado a un margen $e \in [-\varepsilon, \varepsilon]$ determinado previamente, obteniéndose así el valor de las ganancias mediante las siguientes ecuaciones:

$$Ce = \frac{U}{\max(e)} \tag{6–1}$$

$$C_d = K_i C_e \left(K_p - \sqrt{K_P^2 - 4K_i K_d} \right) / 2$$

$$C_0 = \frac{K_i}{C_e}$$

$$C_1 = \frac{K_d}{C_d}$$

Para el caso particular de este proyecto, las entradas se normalizaron con un valor $|E| = 10$ y en consecuencia una salida $|U| = 20$, siendo esta la agregación de ambas entradas. Para el caso del PI las ecuaciones se reducen a las siguientes:

$$C_e = \frac{U}{\max(e)} \tag{6–2}$$

$$C_d = C_e \frac{K_p}{K_i}$$

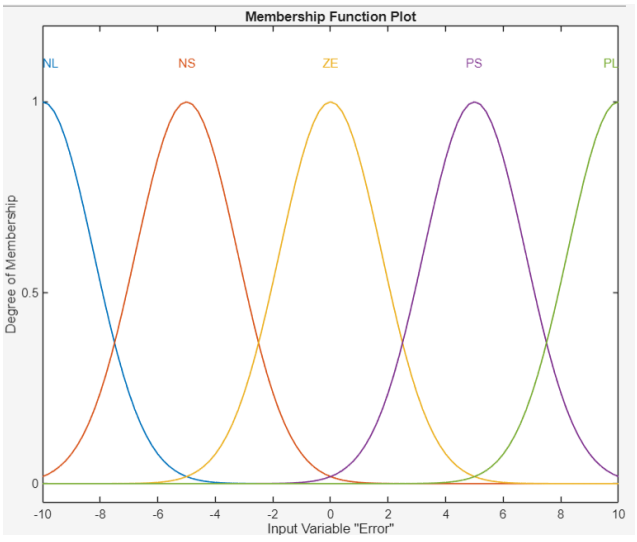
$$C_0 = \frac{K_i}{C_e}$$

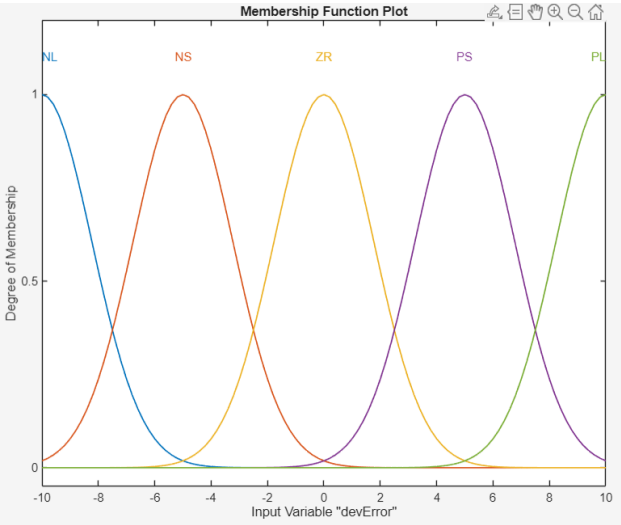
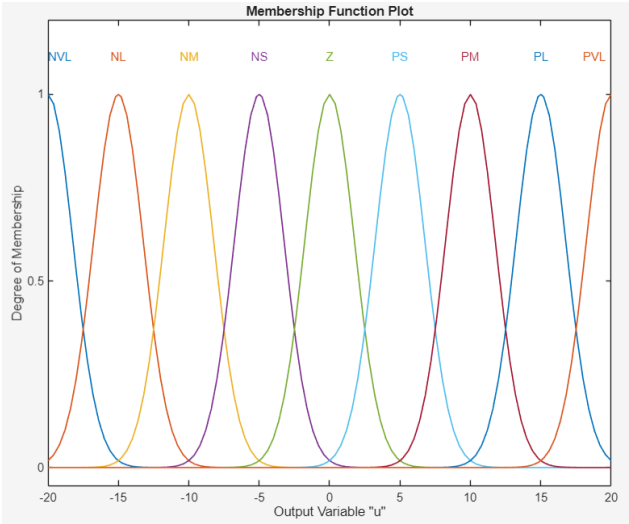
$$C_1 = 0$$

6.2.1.3 Inputs y output

Los valores empleados como entradas en el FLC diseñado son el error de posición *(e)* y su derivada *(ė)*. La salida es la fuerza *(F)* ejercida sobre el carro para realizar el desplazamiento.

Tabla 6-3. MFs de FLC basado en PID

	$\sigma = 1.769$ $\mu = [-10, -5, 0, 5, 10]$ Error

	$\sigma = 1.769$ $\mu = [-10, -5, 0, 5, 10]$ Derivada del error
	$\sigma = 1.769$ $\mu = [-20, -15, -10, -5, 0, 5, 10, 15, 20]$ Señal de control

Como se puede ver en la tabla se han usado gaussianas para las MFs. Su origen es el de aportar transiciones suaves en la superficie de control (ver Figura 4-11). Esto, acompañado de una base de reglas y de una distribución de las MFs determinada (generalmente equiespaciadas), hacen posible generar una superficie de control semejante a la de un PID, que es en esencia un plano.

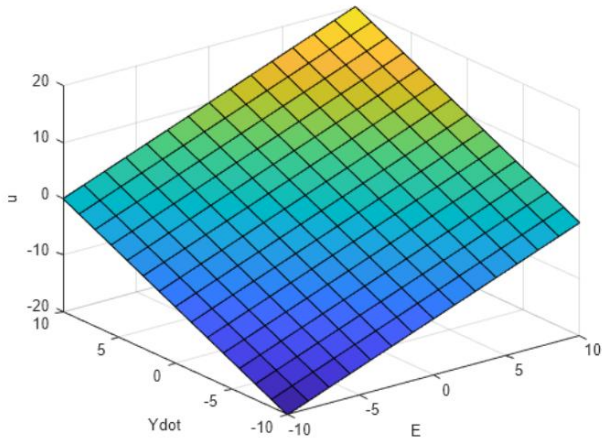


Figura 6-10. Superficie de control de un FLC basado en un PID [17]

6.2.1.4 Base de reglas

El planteamiento de la base de reglas para un FLC basado en un controlador PID es, en última instancia, generar la misma superficie de control que un PID, o lo que es lo mismo, que la relación entre la salida y las entradas sea lo más parecido a un plano.

Para conseguir esto, además de configurar las MFs de las salidas y entradas, tal y como se ha comentado en el apartado anterior, se necesita una base de reglas que emula la actuación propia de un PID. Para ello solo hay que tener en cuenta como influyen los términos proporcional y derivativo en la salida del controlador, pues el término integral se obtiene integrando la salida del FLC.

Tabla 6-4. Base de reglas de un FLC basado en PID

$e(t) \setminus \dot{e}(t)$	**NL**	**NM**	**ZR**	**PM**	**PL**
NL	NVL	NL	NM	NS	ZR
NM	NL	NM	NS	ZR	PS
ZR	NM	NS	ZR	PS	PM
PM	NS	ZR	PS	PM	PL
PL	ZR	PS	PM	PL	PL

Como se puede ver las entradas poseen 5 MFs (NL, NM, ZR, PM, PL) y la salida posee 9 MFs (NVL, NL, NM, NS, ZR, PS, PM, PL, PVL), que son las expuestas en la Tabla 6-3. Hay otra opción que permite generar este mismo efecto con MFs más simples de computar (triangulares), pero se hace implementando un FLC de tipo Takagi-Sugeno, por lo que no se ha empleado. Con todo esto la superficie de control obtenida es la siguiente:

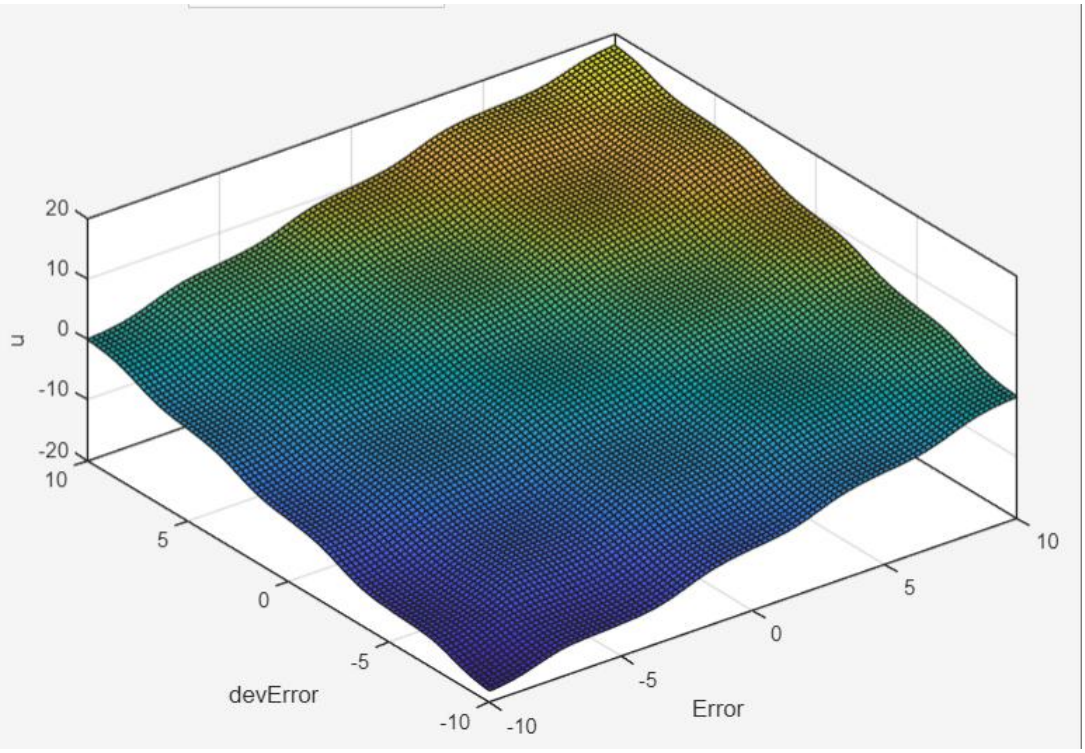


Figura 6-11. Superficie de control del FLC creado

6.2.1.5 Modelo de Simulink y comparativa

Hasta ahora se he expuesto como obtener un FLC en base a un PID conocido, por lo que queda es comprobar si el FCL diseñado es capaz de realizar una respuesta semejante al original. Para ello la simulación realizada ha sido hecha empleando simplemente el PID del controlador en cascada sobre el sistema y se ha comparado su respuesta con su homónimo borroso.

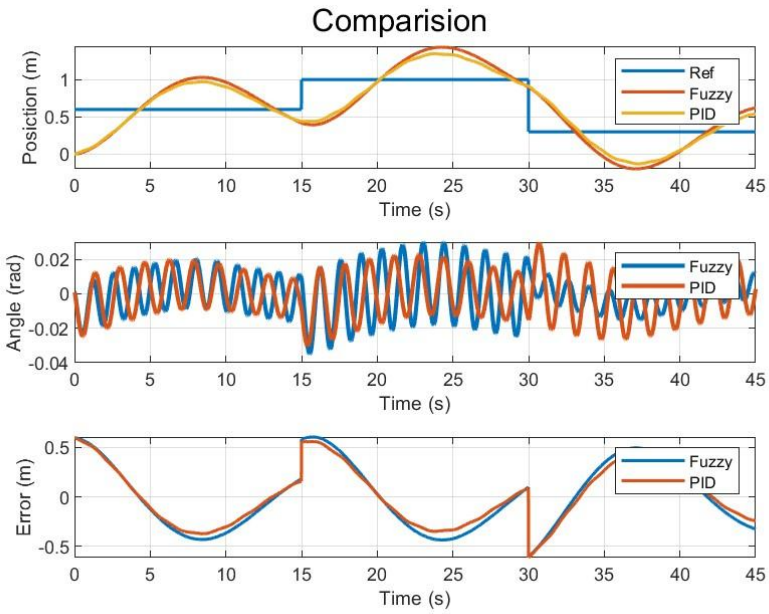


Figura 6-12. Comparativa entre FLC y PID

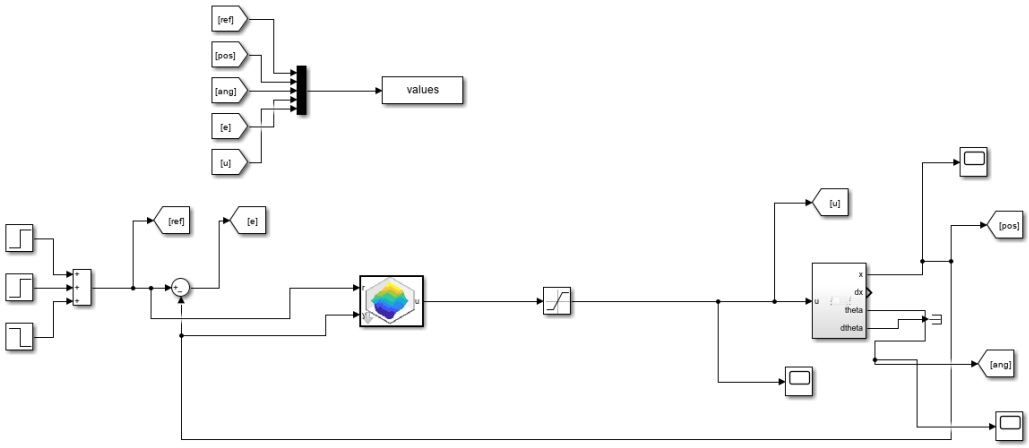


Figura 6-13. Modelo empleado en simulación

Como se puede ver los resultados son muy semejantes empleando ambos controladores, siendo un poco menos agresivo el FLC que el PID. Por simplicidad, para realizar el FLC basado en PID, se ha hecho uso del bloque "Fuzzy PID Controller" de Simulink, que ya posee la configuración expuesta en la Figura 6-8.

6.2.2 FLC del sistema

Una vez aclarado el diseño del FLC se procede a implementar una solución para el sistema planteado en cuestión.

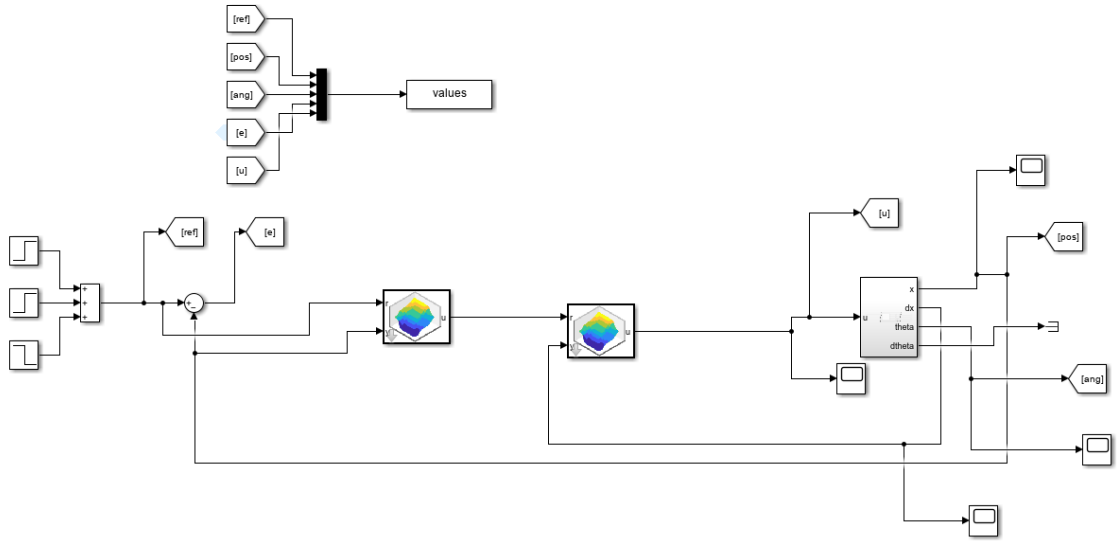


Figura 6-14. Modelo empleado con FLCs

En el modelo hay dos FLCs, el primero es el expuesto anteriormente (ver Figura 6-13) y el segundo es el homónimo borroso del PI. Con esta configuración y empleando los pasos de diseño comentados a lo largo del apartado 6.2.1 para el controlador PI, se obtienen los siguientes resultados.

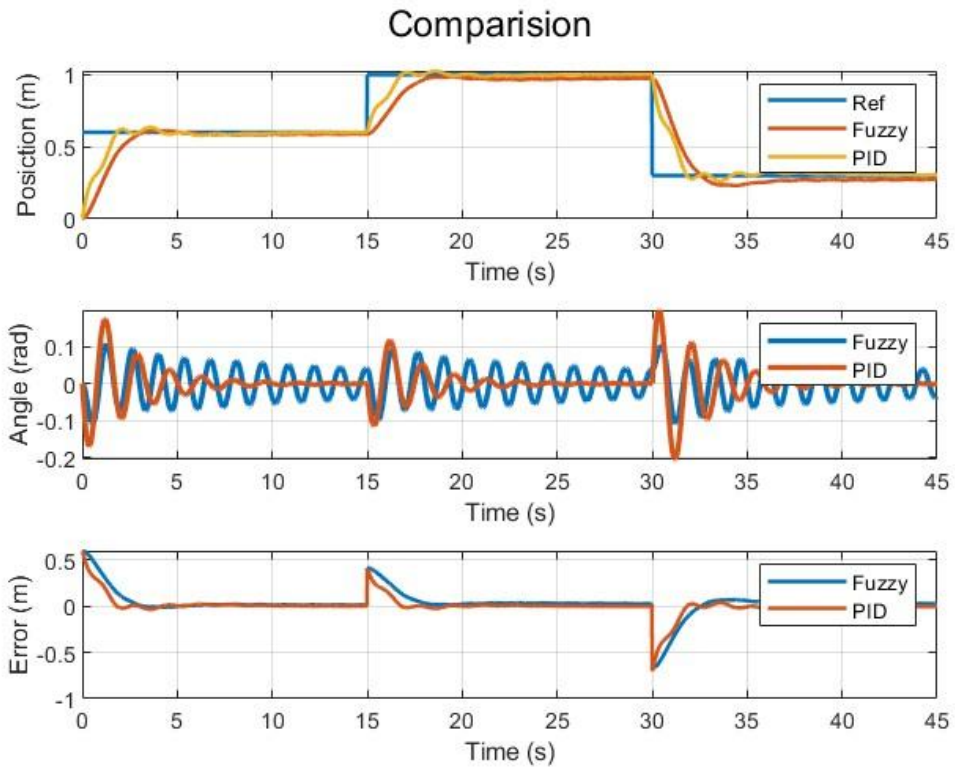


Figura 6-15. Comparación entre PID-PI y FLCs

Como se puede apreciar el conjunto de controladores borrosos producen una respuesta más suave, aunque algo más lenta que el control en cascada original, pero poseen un gran defecto; y es que, aunque la oscilación máxima que experimenta la carga es inferior al comienzo que, con el control en cascada, la atenuación efectuada es mucho menor.

6.2.3 Conclusión

Tras todo lo visto hasta ahora, podemos concluir que, aunque simple de implementar, diseñar un FLC en base a un PID para nuestro sistema no parece tener gran utilidad. Hay dos grandes razones:

- **Mismo número de controladores:** Al sustituir cada controlador por su equivalente no reducimos el número de controladores, algo que sería indiferente de no ser porque los primeros *(PID-PI)* son más simples de computar, diseñar e implementar.
- **Balanceo de la carga:** Aun poseyendo una respuesta más suave y precisa que la del control en cascada, la atenuación del balanceo de la carga deja mucho que desear, siendo claramente peor en este aspecto que control en cascada original.

Por todo ello, el diseño del FLC implementado finalmente no ha sido el realizado en apartado **¡Error! No se e ncuentra el origen de la referencia.**. Aun así, todo lo presentado en dicho apartado sirve de gran ayuda para profundizar en el diseño de los FLCs.

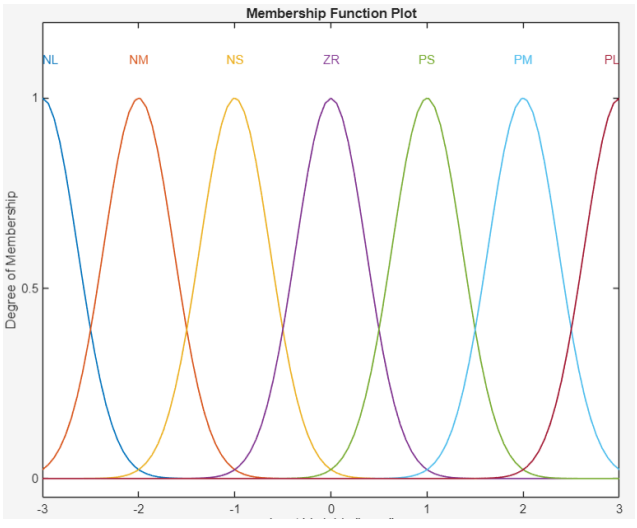
6.3 FLC basado en conocimiento de expertos

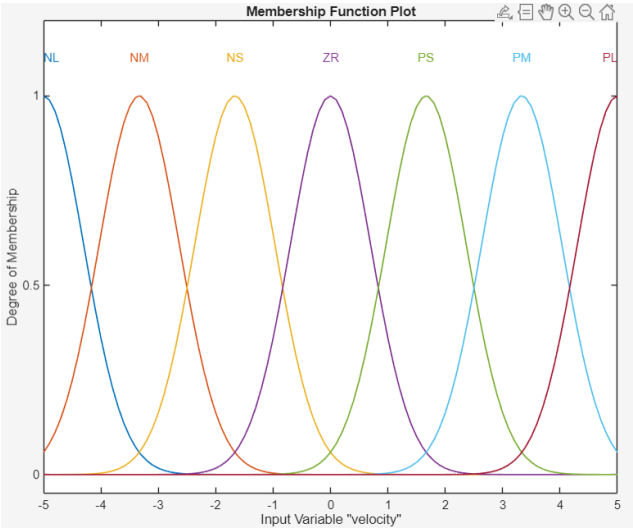
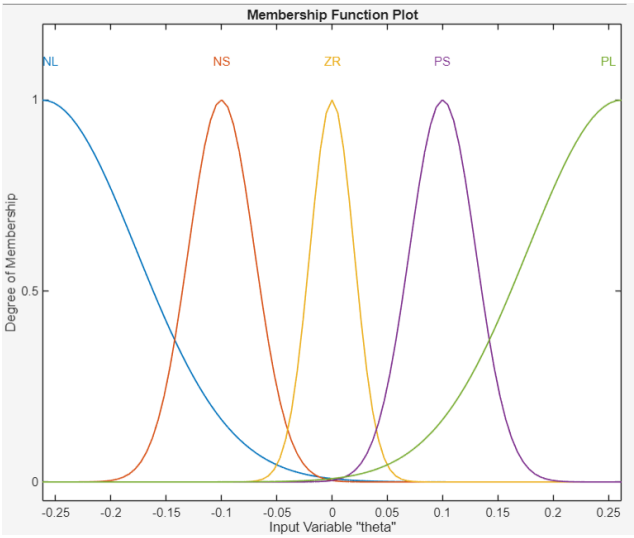
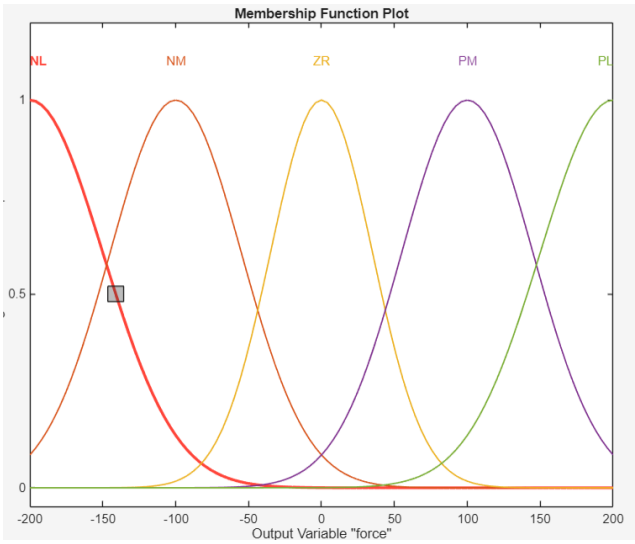
Tras comprobar las deficiencias del diseño anterior, se ha recurrido al método más común de diseño de un FLC, basarse en conocimiento de un experto. Al recurrir a este método los valores de entrada dejan de ser necesariamente el error y su derivada, pudiendo reducir el número de FLCs a uno, además de poder introducir el valor directo del ángulo de balanceo *(θ)* como parámetro de entrada añadiendo mayor precisión en el control de este.

6.3.1 Inputs y output

Los valores empleados como entradas en el FLC diseñado son el error de posición *(e)*, la velocidad del carro *(ẋ)* y el ángulo con respecto a la horizontal de la carga *(θ)*. La salida es la fuerza *(F)* ejercida sobre el carro para realizar el desplazamiento. Debido a que el sistema es muy no lineal y que se quieren establecer transiciones suaves, las MFs empleadas para las entradas y la salida han sido gaussianas. Todo ello ya ha sido expuesto en el apartado del diseño anterior por lo que no es necesario volver a incidir en ello.

Tabla 6-5. MFs de las entradas y la salida

	$\sigma = 0{,}365$ $\mu = [-3, -2, -1{,}0{,}1{,}2{,}3]$ Error de posición

	$\sigma = 0{,}7$ $\mu = [-5, -3.333, -1.667, 0, 1.667, 3.333, 5]$ Velocidad de carro
	$\sigma = [0.085, 0.03, 0.02]$ $\mu = [-0.2618, -0.1, 0, 0.1, 0.2618]$ Ángulo de balaceo de la carga
	$\sigma = [50, 45, 35]$ $\mu = [-200, -100, 0, 100, 200]$ Variable de control

En la tabla se exponen todas las variables antes mencionadas y sus respectivas MFs. Tanto el error de posición como la velocidad poseen 7, mientras que el ángulo y la variable de control poseen solo 5. Esto tiene que ver con la base de reglas que se expondrá posteriormente, aunque se puede inducir de ello que dichos inputs son de

gran relevancia para el establecimiento de esta.

6.3.2 Base de reglas

Para poder inducir una base de reglas para nuestro FLC es necesario partir del conocimiento de algún experto además de la propia pericia para convertir dicho conocimiento en las reglas aplicadas a nuestro FLC. Para arrancar se parten de las siguientes [2]:

- Arrancar a media potencia.
- Durante el recorrido y hasta próximo a la posición final, la cabeza de la grúa puede ir por delante del contenedor. Aceleraciones bruscas enlentecen el proceso porque para equilibrar el contenedor hay que realizar varios cambios del sentido de la marcha continuos.
- Cuando la cabeza de la grúa esté cerca del objetivo hay que enlentecer la marcha de modo que al final el contenedor llegue casi sin oscilaciones.
- Cuando la cabeza de la grúa está en el objetivo y el cable no tiene balanceo, se para el motor del carro.

Con toda esta información se pude generar la siguiente base de reglas:

Tabla 6-6. Base de reglas heurísticas

$e \setminus \dot{x}$	NL	NM	NS	ZR	PS	PM	PL
NL		NM					
NM	PM	ZR	NM				
NS		PM	ZR	NM	NM	NL	
ZR		PL	PM	ZR	NM	NL	
PS		PL	PM	PM	ZR	NM	
PM					PM	ZR	NM
PL						PM	

Además de esas 23 reglas hay otras 2, haciendo un total de 25, donde la única variable relevante es el ángulo de balanceo:

Tabla 6-7. Reglas del ángulo θ

IF θ is NL THEN F is NL
IF θ is PL THEN F is PL

Gracias al conocimiento experto, de un conjunto de $7 \cdot 7 \cdot 5 = 245$ posibles reglas, se ha reducido este número a 25, o lo que es lo mismo; la base es un 10.20% de su potencial tamaño. Esto es muy importante, pues a mayor número de reglas; más tiempo de cómputo es necesario para el controlador para poder generar el *crisp* de salida, es decir; el valor defuzzificado de la señal de control.

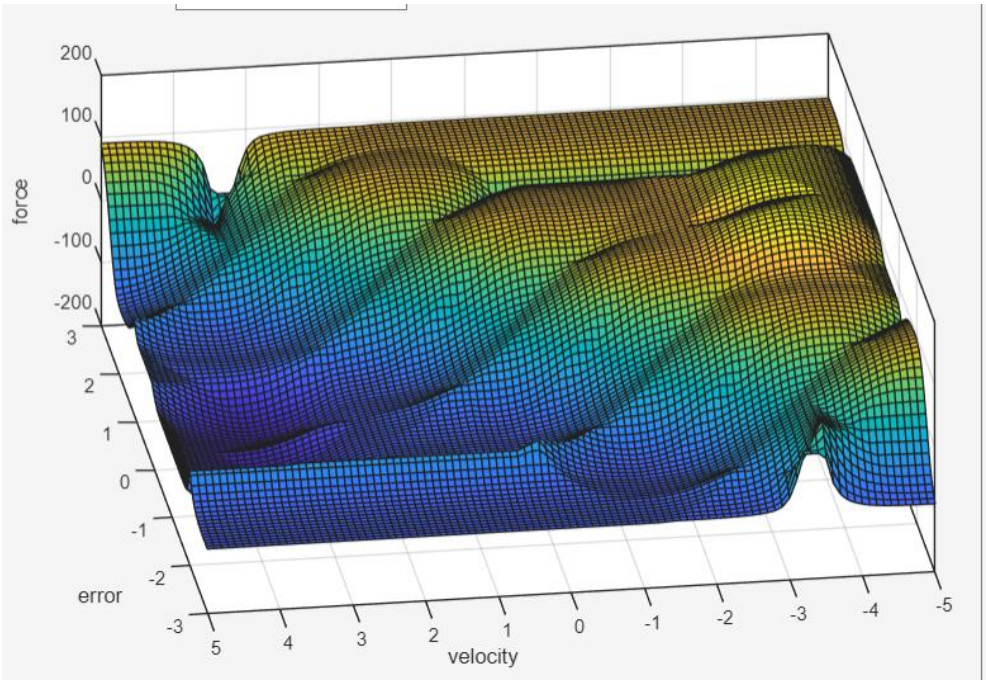


Figura 6-16. Superficie de control del error de posición y velocidad

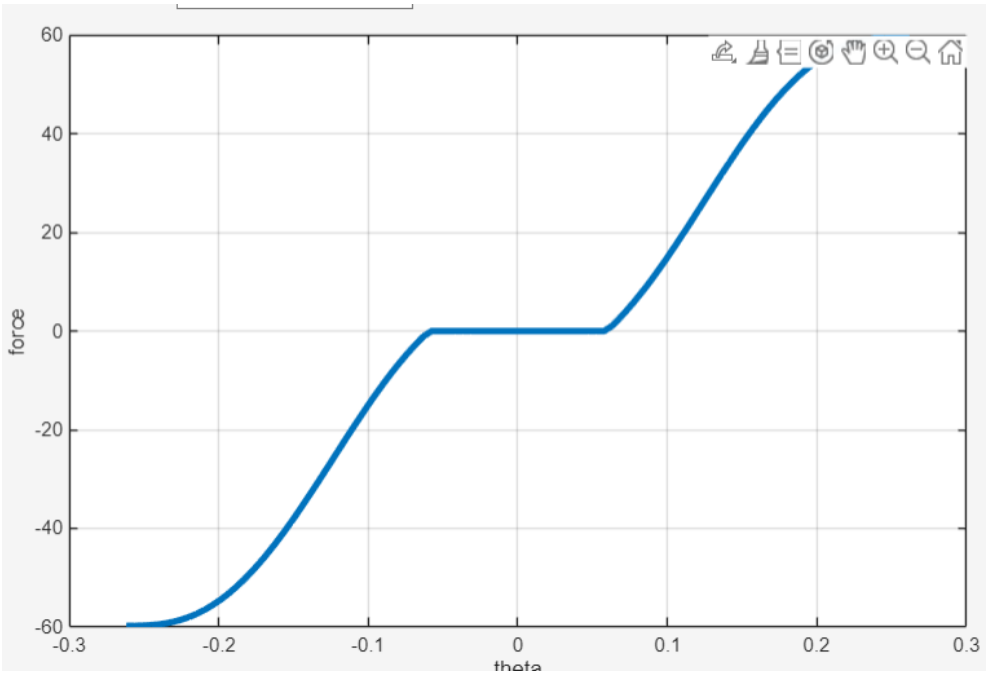


Figura 6-17. Superficie de control del ángulo de balanceo

Como se puede ver en ambas superficies de control, ninguna de ellas es lineal, ni en el plano (recta), ni en el espacio (plano). Aun así, sus transiciones son mayormente suaves, todo gracias la selección de las gaussianas como MFs.

6.3.3 Comparativa entre métodos de control

Lo que queda ahora es comparar el FLC basado en heurística con el control en cascada PID-PI y poder extraer ciertas conclusiones.

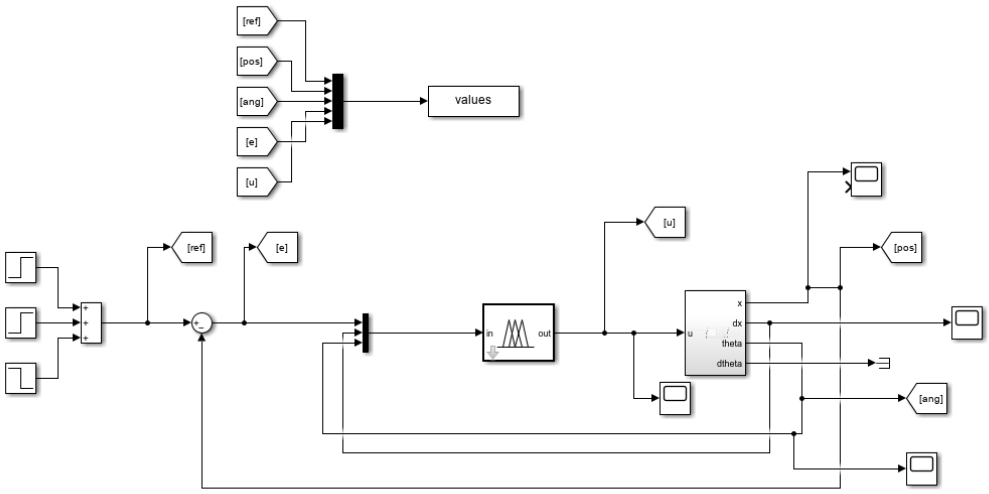


Figura 6-18. Modelo de FLC heurístico

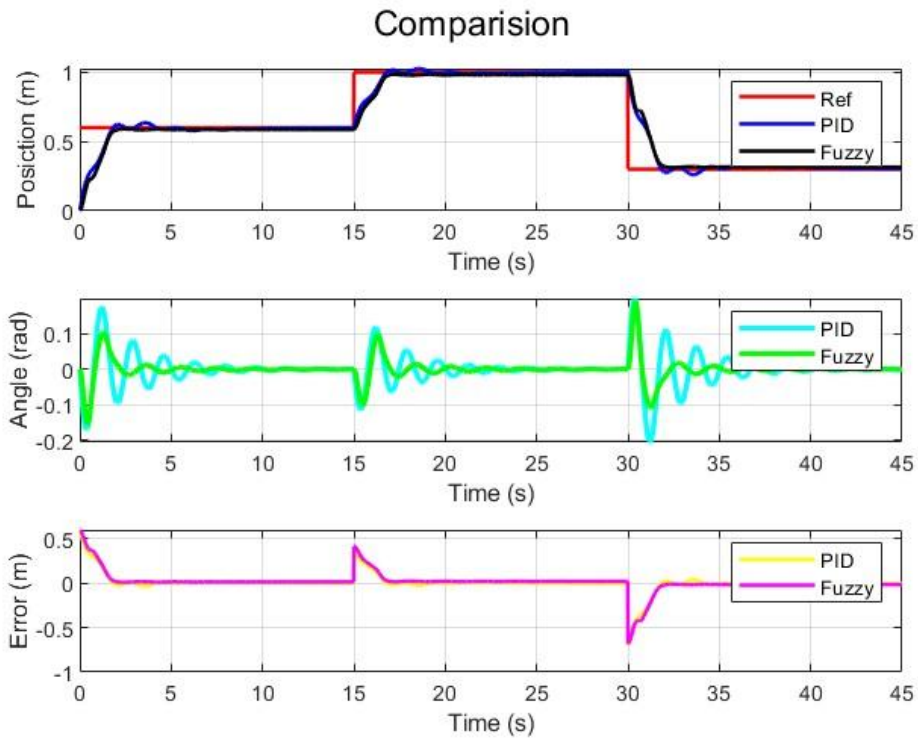


Figura 6-19. Comparativa entre FLC heurístico y PID-PI

A la luz de los resultados y dando un primer vistazo, pareciese que ambos son muy semejantes, pero hay dos claras diferencias según se pueden apreciar en las gráficas obtenidas:

- Suavidad de la respuesta: Al igual que en al caso anterior el controlador borroso obtiene una respuesta menos oscilante que el control en cascada clásico.
- Atenuación del balanceo: Aunque es cierto que el balaceo inicial de la carga es muy semejante entre ambas, la atenuación realizada por el FLC es muchísimo mayor que en el otro caso y todo esto teniendo una respuesta igual de rápida y sin oscilaciones con respecto a la referencia.

Puede aun así argüirse que el control anterior es mucho más simple de implementar y diseñar y que realizar todo lo anterior simplemente para obtener estas mejoras sea excesivo, y eso sería así de no ser por el hecho de que la carga no siempre es la misma. Hasta ahora todos los parámetros del modelo han sido fijados con los siguientes valores:

M (kg)	m (kg)	G (m/s^2)	L (m)	μ (kg/s)
10	5	9.81	1	0.3

Pero algo a lo que ha de enfrentarse el control diseñado es precisamente al cambio en los valores de la masa de la carga *(m)*, pues no siempre se trasladarán cargas del mismo peso. Por ello se han de someter ambos controladores a una última prueba.

6.4 Control frente al cambio de carga

Los controladores lineales como el PID son relativamente simples de plantear cuando se posee un modelo lineal de la planta que representa fielmente la dinámica del sistema, pero tienen un problema y es tienden a fallar cuando se varía de punto de operación, pues las acciones que realiza este sobre el sistema son incrementales. Un cambio en el peso de la carga es algo semejante a inducir un punto de operación diferente, pues ahora el offset es diferente.

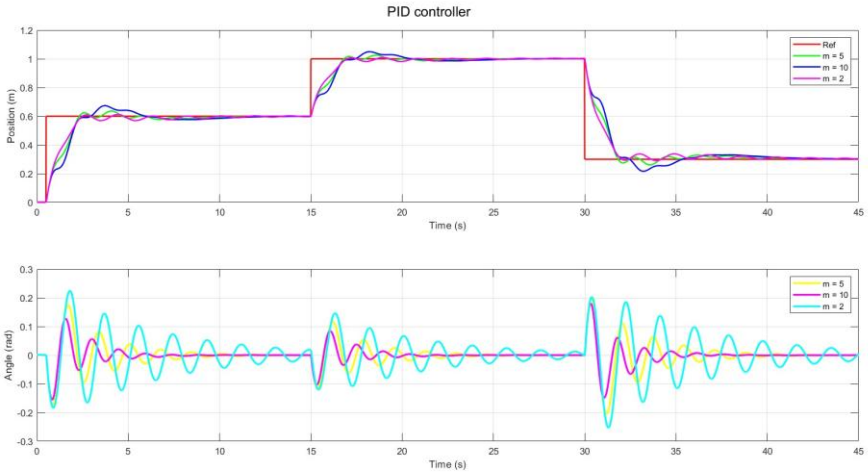


Figura 6-20. Control en cascada frente cambios en la masa de la carga

Como se ha adelantado, los cambios en la masa inducen peores resultados tanto en lo que respecta al seguimiento de la posición como a la atenuación del ángulo de la carga, siendo si acaso más sangrante en el segundo.

En lo que respecta al FLC, ya se ha comentado anteriormente que es más robusto frente a este tipo de variaciones debido a cómo funciona internamente para aportar una señal de control.

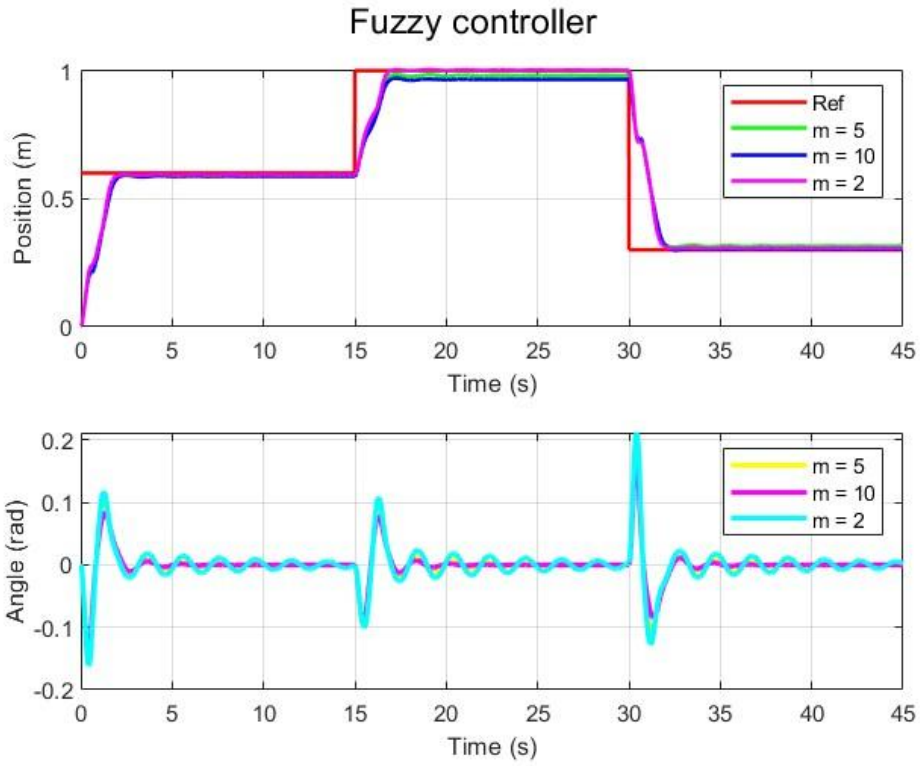


Figura 6-21. FLC frente a cambios en la masa de la carga

Tal y como se puede ver la respuesta es mucho más consistente que en el caso del control cásico en cascada, la atenuación del balanceo de la carga es mejor en todos los casos y la suavidad y precisión de la posición, aunque sufren algo, desde luego no llegan a nivel de irregularidad que presentaba el control clásico.

Hasta ahora todo lo mostrado ha sido simulado en el entorno de desarrollo de MATLAB que, aun siendo muy útil para sacar conclusiones sobre que estrategia de control es mejor, no nos aporta información sobre el funcionamiento de los controles planteados en un autómata. Para ello se ha desarrollado el posterior apartado que simula el FLC en un autómata de la serie TM221 de Schneider Electric disponible en el software de Machine Expert Basic.

6.5 FLC en autómata

Los autómatas de la serie TM221 estarían catalogados como autómatas de gama media-baja, teniendo un buen precio por las características que poseen. Debido a ello, estos autómatas no poseen funciones de lógica borrosa, algo que otras series como la S7-1500 de Siemens, si tienen. Por ello, en este apartado se creará desde cero un controlador borroso equivalente al planteado en MATLAB y se probará para comprobar, entre otras cosas su precisión y rapidez.

6.5.1 Comunicación entre MATLAB y Machine Expert

Antes de poder si quiera realizar las pruebas de un controlador borroso generado en Machine Expert, es necesario conocer y asegurarla comunicación entre MATLAB y este. Para ello es necesario el Toolbox conocido como “Industrial Communication”, que posee una serie de funciones necesarias para establecer la comunicación entre ambas plataformas.

Las funciones y su utilidad son las siguientes:

Tabla 6-8. Funciones usadas de "Industrial Comunication" Toolbox [18]

m = modbus(Transport, DeviceAddress, Port)	Esta función genera un objeto tipo modbus al cual se le especifica el método de trasporte *(Transport)*, la dirección IP del dispositivo *(DeviceAddres)* y el puerto de acceso *(Port)*.
moddata = read(m, target, address, count, precision)	La función permite leer lo recibido en el objeto modbus *(m)*, especificando el tipo de registro *(target)*, la dirección de este *(address)*, el número de datos a leer *(count)*, y el tipo de dato que se lee *(precision)*.
moddata = write(m, target, address, count, precision)	Esta función es equivalente a la función read, pero en vez de leer datos del objeto modbus, escribe sobre él.

Para poder hacer uso de la comunicación modbus, previamente se ha debido de iniciar la simulación y haber activado el controlador en Machine Expert, pues mientras esto no se haga no habrá ningún elemento virtual en nuestro ordenador para poder inicializar la comunicación modbus. Tras realizarlo siempre se debe de usar las siguientes líneas de código para poder establecer la comunicación:

```
global m;
m = modbus('tcpip', '127.0.0.1', 502);
```

Figura 6-22. Inicialización de la comunicación modbus

Con esto MATLAB establece una comunicación tipo 'tcpip' con el controlador con dirección IP '127.0.0.1' y puerto '502'. Tras ello las operaciones de *read* y *write* se pueden realizar sin problemas con dicho elemento mediante el objeto modbus *(m)*. Para explicar el uso de estas funcionan se expone la siguiente tabla:

Tabla 6-9. Manejo de read y write

Target	**'coils'→** Son los registros digitales "%Mx" **'holdingregs'→**Son los registros de palabras tipo "%MWx"
Address	Es la dirección exacta del registro al que se quiere acceder, es decir es el valor de 'x' de los registros
Count	Es el número de registros consecutivos al que queremos acceder para leer o un vector de elementos que contienen la información a escribir en los registros consecutivos, incluido el especificado en *address*
Precision	Es el nivel de precisión que se quiere en los datos. Para los registros tipo 'coil', no es necesario especificarlo y para los 'holdingregs', la precisión es de un entero con signo de 16 bits *(int16)*

Con toda esta información se puede entender el siguiente código, que es el utilizado para enviar y recibir datos del controlador generado en Machine Expert Basic:

```
function [u] = controller(error, vel, theta)
    global m;
    error = round(error * 1e3);
    vel = round(vel * 1e3);
    theta = round(theta * 1e3);
    write(m, 'holdingregs', 501, [error, vel, theta], 'int16');
    pause(0.05);
    u = read(m, 'holdingregs', 504, 1, 'int16') / 1e2;
end
```

Figura 6-23. Comunicación modbus con controlador

Lo que hace la función es convertir los datos de la simulación en MATLAB *(error, vel, theta)* mediante un factor de escala de 10^3 para enviarlo a los registros de palabra 500-502 *(MATLAB maneja los índices a partir de 1)*, espera 50 ms a que el controlador realice las operaciones necesarias para obtener posteriormente el valor de la señal de control *(u)* mediante la lectura del registro de palabra 503 alterado por un factor de escala de 10^2. La razón de dichos factores de escala es debido a las propias limitaciones de los registros que solo manejan enteros y si queremos tener precisiones superiores a la unidad debemos de usarlos.

6.5.2 Modelo empleado

Antes si quiera de realizar el controlador es necesario crear el que será el modelo de Simulink para la prueba de funcionamiento de este.

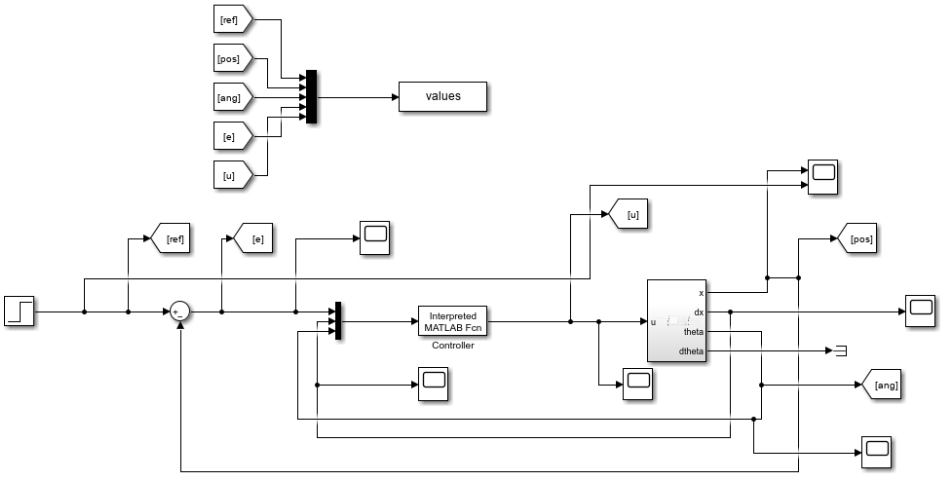


Figura 6-24. Modelo para FLC de un autómata

El bloque de 'Interpreted MATLAB Function' hace uso de la función antes expuesta (véase Figura 6-23), donde la salida es la señal de control y las entradas son el error de posición, la velocidad del carro y el ángulo con respecto a la horizontal de la carga. Esta configuración es exactamente igual que la expuesta con el bloque 'Fuzzy Logic Controller' expuesta anteriormente (véase Figura 6-18).

6.5.3 Programación de FLC en autómata

Machine Expert Basic al ser un programa sin licencia como sucede con otros del mercado, posee ciertas limitaciones para la programación de PLCs, como es el caso de no disponer de ST como lenguaje de programación. Aun así, los leguajes de IL y LD son más que suficientes para poder realizar el controlador borroso planteado. Debido a que se han generado ciertas funciones propias para simplificar el programa primero se explicaran cada una de ellas y posteriormente como se ha desarrollado el programa. También se han usado funciones internas como MAX_ARR(%MFx:L), que dado un registro y una longitud L, devuelve el mayor valor almacenado en el vector de L registros, así como también los registros de palabras constantes (%KFx) para las medias y varianzas entre otras constantes usadas para una mejor reprogramabilidad.

6.5.3.1 GAUSS

- **Inputs:** Variable leída procedente de MATLAB, media y varianza
- **Output:** Grado de pertenencia

Esta función devuelve el grado de pertenencia de la correspondiente MF definida por su media y varianza, con respecto al valor de entrada aportado por MATLAB. Las operaciones realizadas son las de la adecuación del primer input por un factor de escala *(10^3)* y obtener el grado de pertenencia mediante la ecuación de la gaussiana sin normalizar.

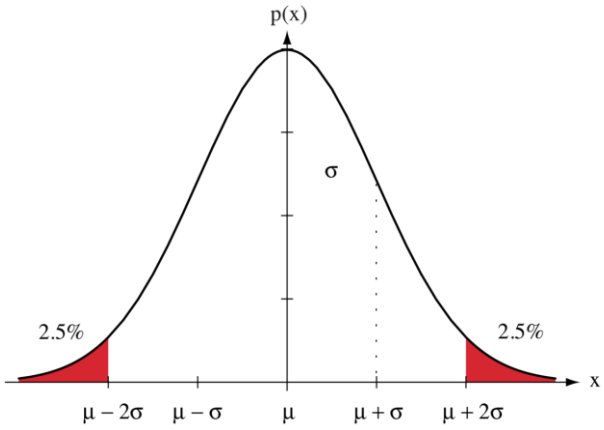


Figura 6-25. Gaussiana

(6–3)

$$p(x) = e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

6.5.3.2 MIN

- **Inputs:** Antecedente primero y antecedente segundo.
- **Output:** Mínimo de los antecedentes

Esta función recibe ambos antecedentes (grado de pertenencia) de cada regla y devuelve el mínimo de ambos. En resumen, es la aplicación del operador *and* para el caso de dos antecedentes.

6.5.3.3 MIN_VALUE

- **Inputs:** Valor de *x* de la gaussiana, grado de pertenencia a comparar, media y desviación típica.
- **Output:** Mínimo grado de pertenencia entre el introducido como input y el computado.

Esta función cumple una función crucial, pues determina el grado de pertenencia de cada uno de los puntos de muestreo, es decir, el μ_i de la correspondiente x_i de la ecuación (5–3). Para ello se compara el valor del grado de pertenencia obtenido por la evaluación de las reglas, es decir, el grado de pertenencia máximo de entre todas las reglas que afectan a una MF de la salida (operación *max*), con el grado de pertenencia real de dicho punto. Vamos a visualizarlo con un ejemplo:

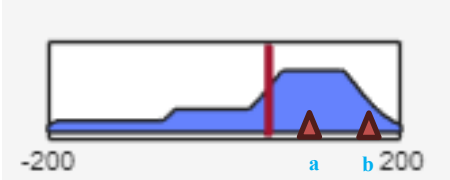


Figura 6-26. Ejemplo de asignación de grado de pertenencia a la salida

Como se puede ver en la imagen se señalan dos puntos *a* y *b*. Ambos tiene diferentes grados de pertenencia en la salida, siendo el grado de pertenencia de *a*, el aportado por el proceso de inferencia de las reglas, mientras que el de b es simplemente el que tendría aplicando la fórmula de la gaussiana antes expuesta (véase (6–3)). Esta función simplemente comprueba si el grado de pertenencia de un punto dado y lo compara con el obtenido por el proceso de inferencia, asignándole el menor de ambos.

6.5.3.4 Tarea Maestra

Antes de nada, se ha de aclarar que la tarea maestra funciona como una tarea periódica con intervalos de 50 ms, para coincidir con la pausa expresada previamente (véase Figura 6-23). Con esto en mente se puede explicar la secuencia seguida:

1. **Cálculo de los grados de pertenencia:** Primero se obtienen todos los grados de pertenencia de cada una de las MFs de los inputs, siendo un total de 19. Los valores recolectados son los pertenecientes a %MW500-2.

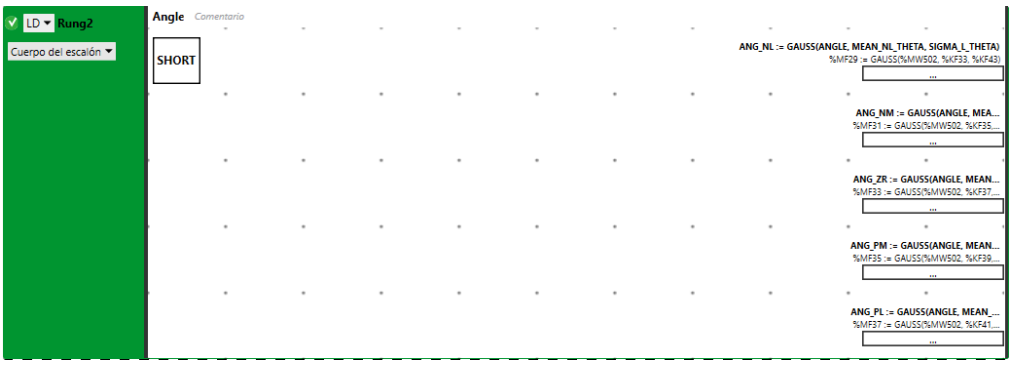


Figura 6-27. Cálculo del grado de pertenencia del ángulo

2. **Regla dominante:** En esta parte se obtiene el grado de pertenencia más grande de entre todas las reglas que afectan cada una de las MFs del output, siendo un total de 5. En esta parte se aplica tanto *min* para aplicar *and* como *max* para la agregación.

Figura 6-28. Regla dominante para la MF NL

3. **Bucle de cálculo de la MF de salida:** Después de obtener los valores de comparación para la función MIN_VALUE, se aplica dicha función a cada punto de muestreo de la MF de salida, para recolectar el mayor de todos los 5 obtenidos, pues se aplica a cada MF de la salida. Posteriormente se incrementa el numerador en $x_i\mu_i$ y el denominador en μ_i, siendo x_i el punto evaluado y μ_i el valor del grado de pertenencia obtenido tras las operaciones citadas. Esto ocurre en cada iteración por lo que al final en las variables NUM y DEN se obtiene el valor del numerador y el denominador de la ecuación (5–3).

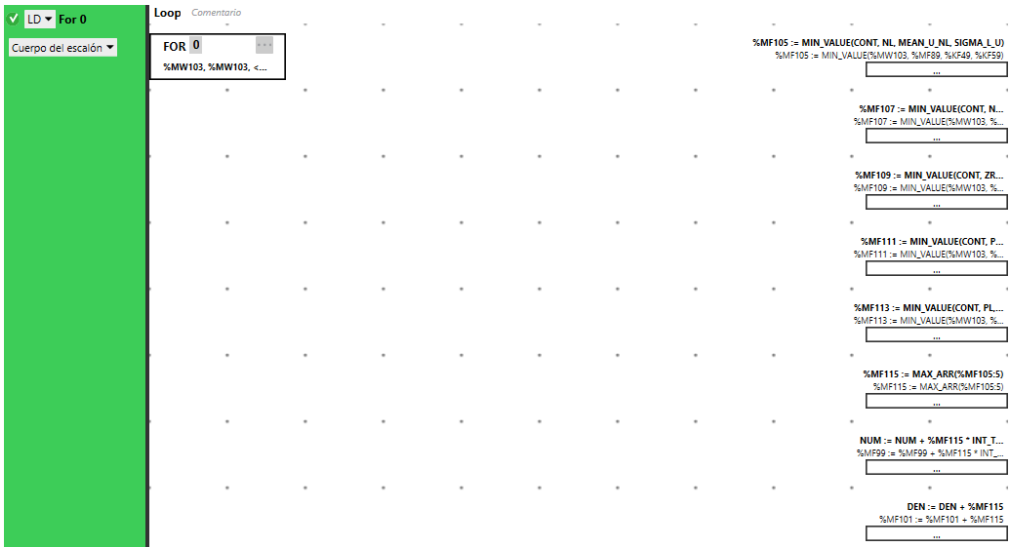


Figura 6-29. Bucle de cálculo

4. **Cálculo del crisp:** Tras salir del bucle simplemente se aplica la división de NUM entre DEN y se multiplica por un factor de escala 10^2, para convertirlo posteriormente a entero y guardarlo en el registro %MW503.

Figura 6-30. Cálculo del crisp

Para este caso el número de iteraciones realizadas es de 101, pues esos son el número de puntos a evaluar en la salida. Esto se ha hecho así pues en el bloque FLC de Simulink usado (véase Figura 4-12), el valor predeterminado de muestreo es ese.

6.5.3.5 Resultados

Tras realizar toda la programación anterior tanto en MATLAB como en Machine Expert solo queda probar los resultados del controlador en una simulación. Debido a que es comparativamente más lento que la simulación directa en MATLAB se ha reducido a comprobar la respuesta ante escalón del sistema controlado.

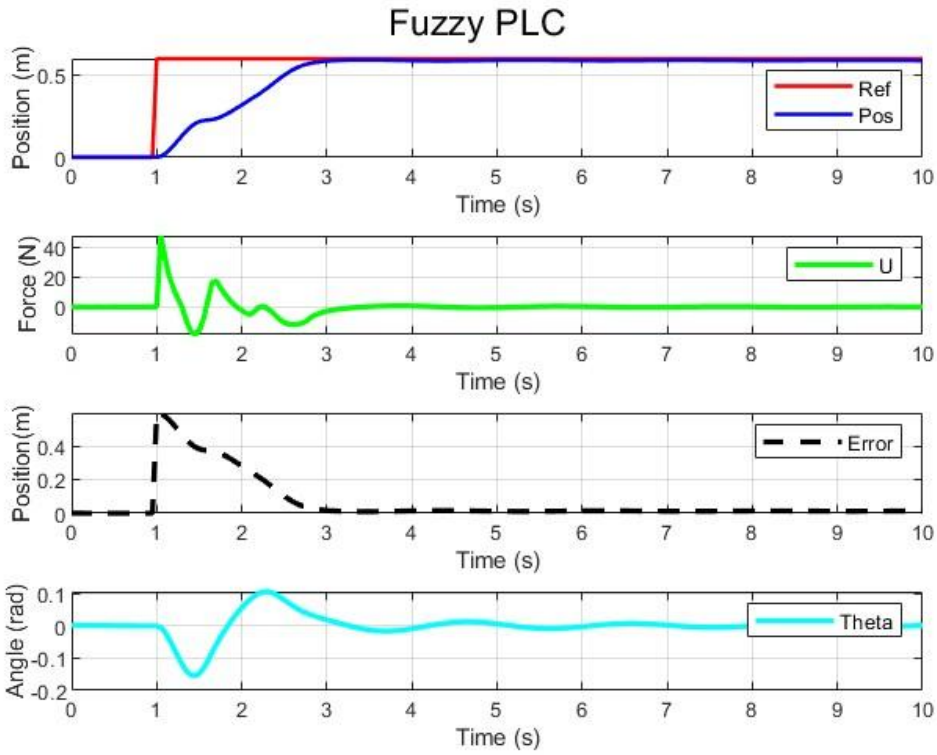


Figura 6-31. Control mediente FLC en autómata

Como se puede ver la respuesta es exactamente igual que la obtenida mediante simulación en MATLAB, el tiempo de subida es el mismo y la atenuación del ángulo de la carga es tan fuerte como en el caso del FLC de Simulink.

Con todo esto podemos dar por hecho que, aunque siendo tedioso de programar, un FLC con las características de diseño antes presentadas, puede ser aplicado no solo en autómatas de gama alta con funciones de lógica borrosa, sino que también se puede emplear en autómatas de menor gama que no las poseen, poseyendo muy buenos resultados con tiempos de muestreo más que suficientes *(50 ms)* para el sistema planteado.

7 CONCLUSIONES

En este Trabajo de Fin de Grado se ha demostrado la viabilidad y eficacia de un sistema automatizado basado en lógica borrosa para la manipulación de materiales pesados. A través de simulaciones detalladas y comparaciones con enfoques tradicionales, se evidenció la capacidad del controlador borroso para gestionar sistemas complejos con mayor adaptabilidad y precisión frente a variaciones en las condiciones de operación.

El desarrollo del sistema incluyó un modelo dinámico que permitió analizar y comprender las características físicas del problema, así como validar las estrategias de control en un entorno virtual antes de su implementación física. Herramientas como MATAB y Machine Expert Basic facilitaron este proceso, proporcionando un entorno seguro para pruebas y optimizaciones.

Los resultados obtenidos resaltan las ventajas de la lógica borrosa, especialmente su capacidad para ofrecer soluciones flexibles e intuitivas frente a sistemas no lineales o inciertos. Aunque la configuración inicial del controlador puede requerir un mayor esfuerzo, su rendimiento y versatilidad justifican ampliamente esta inversión.

En conclusión, el proyecto no solo proporciona una solución técnica para un caso práctico en el ámbito industrial, sino que también subraya el potencial de las tecnologías inteligentes para mejorar la seguridad, eficiencia y productividad en procesos de manejo de materiales. Este trabajo sienta las bases para futuras aplicaciones y mejoras en la automatización de tareas industriales complejas.

REFERENCIAS

[1] M. d. Trabajo, «Estadísticas de accidentes de trabajo,» [En línea]. Available: https://www.mites.gob.es/es/estadisticas/condiciones_trabajo_relac_laborales/EAT/welcome.htm. [Último acceso: 25 11 2024].

[2] J. Andújar, A. Barragán y M. Gegúndez, «CONTROL BORROSO MULTIVARIABLE BASADO EN HEURÍSTICA UN CASO PRÁCTICO: GRÚA PORTA CONTENEDORES.,» *Revista Iberoamericana de Automática e Informática Industrial,* vol. 4, nº 2, pp. 81-89, 2007.

[3] L. A. Zaddeh, «Fuzzy sets,» *Information and control,* vol. 8, nº 3, pp. 338-353, 1965.

[4] L. A. Zadeh, «Outline of a new approach to the analysis of complex systems and decision processes,» *IEEE Transactions on systems, Man, and Cybernetics,* vol. 3, nº 1, pp. 28-44, 1973.

[5] L. A. Zadeh, «The concept of linguistic variable and its application to approximate reasoning,» *Information sciences,* vol. 8, nº 3, pp. 199-249, 1975.

[6] E. H. Mamdani, «Application of fuzzy algorithms for control of simple dynamic plant,» *Proceedings of the institution of electrical engineers,* vol. 121, nº 12, pp. 1585-1588, 1974.

[7] T. Takagi y M. Sugeno, «Fuzzy identification of systems and its applications to modeling and control,» *IEEE transactions on systems, man, and cybernetics,* vol. 15, nº 1, pp. 116-132, 1985.

[8] I. H. Altas, Fuzzy Logic Control in Energy Systems: with Design Applications in MATLAB/Simulink, London: The Institution of Engineering and Technology, 2017.

[9] M. H. Hazam, M. H. Hasan, S. Hassan y S. J. Abdulkadir, «Fuzzy Type-1 Triangular Membership Function Approximation Using Fuzzy C-Means,» de *International Conference on Computational Intelligence (ICCI)*, Malaysia, 2020.

[10] P. N. Nwankwo, K. Akpado y C. C. Okezie, «Development of an Advanced Neuro-Fuzzy Algorithm for Intelligent Temperature Control System,» *International Journal of Advances in Engineering and Management (IJAEM) ,* vol. 6, nº 9, pp. 769-791, 2024.

[11] J. C. G. Infante, *Sistemas con Lógica Difusa,* México: Instituto Politécnico Nacional, 2009.

[12] MathWorks, «Foundations of Fuzzy Logic,» [En línea]. Available: https://es.mathworks.com/help/fuzzy/foundations-of-fuzzy-logic.html. [Último acceso: 2 12 2024].

[13] D. Pacheco Bautista, E. Cortes Rico y F. Aguilar Acevedo, «Diseño de un controlador de carga de tres etapas para sistemas fotovoltaicos usando lógica difusa,» *Ingeniare,* vol. 27, nº 4, pp. 540-550, 2019.

[14] J. Jaekel, R. Mikut y G. Bretthauer, «Fuzzy Control Systems,» de *Control Systems, Robotics, and Automation*, Karlsruhe, H. Unbehauen, 2004, pp. 1-30.

[15] MathWorks, «Fuzzy Inference Process,» [En línea]. Available:

https://es.mathworks.com/help/fuzzy/fuzzy-inference-process.html. [Último acceso: 2 12 2024].

[16] N. T. Nguyen, D. H. Nguyen, T. L. Nguyen y T. V. A. Nguyen, «Takagi-Sugeno Fuzzy Approach with Compressed Representation,» *Smart Systems and Devices,* vol. 32, nº 3, pp. 44-51, 2022.

[17] MathWorks, «Implement Fuzzy PID Controller in Simulink,» MATLAB, [En línea]. Available: https://es.mathworks.com/help/fuzzy/implement-fuzzy-pid-controller-in-simulink-using-lookup-table.html. [Último acceso: 10 12 2024].

[18] MathWorks, «Modbus Communication,» MATLAB, [En línea]. Available: https://es.mathworks.com/help/releases/R2024b/icomm/modbus.html?s_tid=CRUX_lftnav. [Último acceso: 13 12 2024].