

Complete los siguientes datos

- Número de grupo: Grupo 2
- Nombre y Apellidos de los integrantes: Jon Ochoa, Xabier Lopez
- Fecha: 23/10/2025
- Tiempo dedicado a la entrega: 6 Dias

Notas generales

- Dos o más prácticas iguales o con un índice de similitud muy alto serán consideradas plagio por todos los implicados.
- Todas las prácticas son susceptibles de ser revisadas minuciosamente, incluyendo una defensa por parte de sus autores de forma individual. En caso de suspender esta defensa se suspendería la práctica.
- La práctica supone la realización de todos los apartados y el cumplimiento de todas las condiciones, si falta alguno, la práctica estará incompleta y no será evaluada.
 - **Las prácticas que no cumplan este requisito serán consideradas no válidas y obtendrán una calificación de 0.**
- Las prácticas han de ser funcionales y no dar error en su ejecución.
 - **Las prácticas que no cumplan este requisito serán consideradas no válidas y obtendrán una calificación de 0.**

Instrucciones

Este es el cuaderno que utilizarás como **plantilla** para entrega de la práctica correspondiente.

Por favor, **lee atentamente** el enunciado. Si tienes alguna duda, utiliza el foro o pregunta en clase (pero no compartas código).

Consideraciones generales sobre el código:

- Utilizar tantas celdas (de texto o de código) como considere para dar una respuesta legible y clara a las preguntas planteadas.
- Mostrar resultados intermedios o finales (tablas, contenido de las variables...) que demuestren que la solución es correcta si fuera necesario.
- No olvidéis importar todas las librerías necesarias para la correcta ejecución del código.

En las indicaciones, hay instrucciones extra indicadas con el símbolo . No son requisitos para superar la práctica, sino acciones que puedes realizar para mejorar la

calificación.

La calificación de los ejercicios se hará con los siguientes criterios:

- [9 a 10] puntos: La solución aportada cumple con los requisitos de forma excelente y existen aspectos positivos o extras (💡) que avalan su excelencia.
- [7 a 9] puntos: La solución aportada cumple con los requisitos, pero algunas cuestiones menores son susceptibles de ser mejoradas, tales como: presentación de la solución, justificación de la decisión tomada, eficiencia del código...
- [5 a 7] puntos: Las soluciones aportadas no cumplen alguno de los requisitos, como por ejemplo: no se responde a una pregunta, no se aporta la solución a una parte...
- [1 a 5] puntos: La solución aportada no cumple con varios requisitos.
- [0] puntos: solución no aportada o solución plagiada. **Cabecera no completada, ficheros con nombre erróneo, etc.**

La nota de la práctica dependerá de:

1. Cumplimiento de las condiciones.
2. Sofisticación del sistema.
3. Realismo del sistema.
4. Limpieza y eficiencia del código.
5. Comentarios aportados.

Desarrollo a realizar

En esta práctica tendrás que desarrollar un sistema experto relacionado con algún problema relacionada con la [temática seleccionado](#) al inicio del curso.

El problema a resolver ha de ser solucionable a través de un sistema difuso, recuerda que un sistema difuso se utiliza cuando la lógica tradicional no es suficiente para modelar la incertidumbre o la ambigüedad del problema. Por lo tanto, si el problema implica tomar decisiones en situaciones donde los límites no son claros o donde se requiere manejar información parcial o ambigua, un sistema difuso sería adecuado para abordarlo.

Entrega

Validación del problema

Descripción del problema a resolver:

En un almacén automatizado con múltiples robots móviles (AMR), las decisiones operativas suelen depender de variables inciertas o ambiguas: la congestión real de un pasillo cambia con el tiempo, los sensores reportan distancias con ruido, la urgencia percibida de un pedido depende de ventanas de tiempo flexibles, y la "seguridad" de circular a cierta velocidad no es un umbral nítido. Proponemos un sistema difuso que,

dados nivel de batería, distancia al objetivo, congestión del pasillo y urgencia del pedido (y opcionalmente peso del pedido), calcula tres salidas:

Velocidad recomendada del robot (muy-baja, baja, media, alta, muy-alta).

- Confirmación del docente (fecha y comentarios):

Confirmado a dia 16/10/2025 Comentarios : No hacen falta, valorar:

Prioridad de asignación (baja–alta) para el dispatcher (sirve para desempatar entre robots/pedidos).

Necesidad de desvío/espera (baja–alta) para evitar colisiones cuando la congestión es alta.

Introducción

Breve descripción del sistema experto y el problema que resuelve.

El sistema experto desarrollado aplica lógica difusa para recomendar la velocidad óptima de desplazamiento de un robot móvil autónomo (AMR) dentro de un almacén automatizado. A diferencia de un sistema de control clásico basado en umbrales fijos, este modelo gestiona situaciones ambiguas o imprecisas, como la congestión variable en los pasillos o la urgencia flexible de los pedidos.

El sistema recibe como entradas:

Nivel de batería (%): refleja la autonomía disponible.

Distancia al destino (m): mide cuánto le falta para completar la tarea.

Congestión del pasillo (%): estima la densidad de tráfico o posibles bloqueos.

Urgencia del pedido (0–10): indica la prioridad operativa del traslado.

A partir de estas variables, el sistema difuso calcula la velocidad recomendada del robot con cinco posibles salidas: muy baja, baja, media, alta y muy alta. El modelo busca equilibrar seguridad (evitar colisiones o maniobras bruscas) y eficiencia (cumplir los pedidos urgentes con rapidez), ofreciendo una salida continua y explicable incluso ante información incierta.

Contexto y finalidad del sistema.

En un entorno de intralogística automatizada, los AMR comparten espacios de trabajo con otros robots y operarios, lo que hace que las condiciones de circulación sean dinámicas y difíciles de modelar de forma precisa. La finalidad del sistema es apoyar la toma de decisiones operativas del robot, proporcionando una recomendación de velocidad adaptativa en función de factores cambiantes.

Contexto industrial: almacén automatizado con tráfico variable, rutas compartidas y diferentes prioridades de pedidos.

Finalidad técnica: optimizar el movimiento del robot sin comprometer seguridad ni eficiencia energética.

Beneficio: sustituir umbrales fijos ("si congestión > 70%, velocidad = 0.5 m/s") por un razonamiento difuso más realista y flexible.

La salida del sistema puede integrarse en un módulo de planificación o control de flota, de modo que el sistema experto actúe como un "asistente inteligente" que ajusta la velocidad de cada robot según la situación real.

Documentación

Describa cada una de las fases necesarias para el desarrollo de un sistema experto contextualizadas en tu problema.

Describe cada una de las fases necesarias para el desarrollo de un [sistema experto basado en reglas](#), contextualizándolas con el problema que hayas elegido. Es decir, no solo menciones las fases, sino que expliques cómo se aplicarían específicamente a tu problema.

Recuerda, el conocimiento usado ha de basarse en una fuente fidedigna y ha de ser relevante para resolver el problema seleccionado.

El desarrollo del sistema experto se estructura en cinco fases clásicas, adaptadas al problema de la velocidad recomendada de un AMR:

1. Adquisición del conocimiento

Consiste en recopilar información de expertos humanos o fuentes técnicas sobre cómo deben comportarse los robots en distintas condiciones. En este caso, el conocimiento procede de:

Normas y buenas prácticas en la operación de AMR/AGV (p. ej., ISO 3691-4 sobre seguridad de vehículos industriales). Experiencia operativa en intralogística: reducir velocidad ante congestión, batería baja o proximidad; aumentar con urgencia alta y entorno libre. Fuentes académicas sobre control difuso en movilidad robotizada (Fragapane et al., 2021; Rema et al., 2025).

Este conocimiento se traduce en variables medibles (batería, distancia, congestión, urgencia) y en la relación cualitativa entre ellas y la velocidad deseada.

2. Representación del conocimiento

La información obtenida se modela mediante conjuntos difusos y reglas lingüísticas. Cada variable se define con funciones de pertenencia adecuadas:

battery:high → gaussiana (suaviza transición alta). congestion:high → sigmoide (riesgo crece rápido). urgency:critical → campana (transición más gradual). La salida velocidad usa cinco conjuntos (muy_baja → muy_alta) para cubrir todo el rango de velocidades posibles.

3. Inferencia

Es el razonamiento del sistema, donde se aplican las reglas a los hechos actuales. El sistema emplea un modelo Mamdani, con implicación min, agregación max y defuzzificación por centroide, lo que permite combinar varias reglas activas simultáneamente. Ejemplo de regla:

Si urgencia es alta y congestión es baja y batería es alta, entonces velocidad es alta.

De esta forma, el sistema puede emitir una recomendación intermedia y continua, no binaria.

4. Validación y ajuste

En esta fase se comprueba el comportamiento del sistema ante escenarios representativos:

Urgencia alta, pasillo despejado, batería alta → salida muy alta. Congestión alta y distancia corta → salida muy baja. Urgencia media y batería media → salida media.

Si los resultados no son coherentes, se ajustan los rangos o las pendientes de las funciones de pertenencia.

5. Mantenimiento y actualización

El sistema debe poder revisarse con nueva información de operación:

Ajustar funciones de pertenencia si se modifican las velocidades máximas o los criterios de seguridad. Añadir nuevas variables (p. ej., peso del pedido o riesgo de zona) si el almacén se complejiza. Actualizar las reglas según retroalimentación de operarios o datos históricos.

Fuentes de conocimiento

Fragapane, G. et al. (2021). Planning and control of autonomous mobile robots for intralogistics. *Transportation Research Part C*.

Rema, C. et al. (2025). Task Scheduling with Mobile Robots—A Systematic Literature Review. *Robotics* 14(6).

ULMA Handling Systems (2024). Automatización y seguridad en intralogística AMR/AGV.

Programa

Requisitos

- Utilizar al menos 4 variables y codificar al menos 3 conjuntos difusos.

- Incluir 8 reglas difusas.
- Realizar al menos 3 inferencias y explicar sus resultados.
- Explicar el motivo de elección de los conjuntos difusos, reglas y operadores lógicos basándose en documentación o en conocimiento explícito. Proporciona ejemplos si lo consideras necesarios.
- Al menos una función de pertenencia ha de ser trapezoidal.
- Explicar cómo crees que las decisiones anteriores afectaron el rendimiento del sistema.
- El sistema ha de ser realista y basarse en los datos utilizados para construir el sistema experto (2. Adquisición del conocimiento).

💡 Requisitos extra

- Existencia justificada de funciones de pertenencias no triangulares y no trapezoidales con al explicación pertinente.
- El sistema ha de tener un grado de sofisticación elevado.

Importaciones

```
In [22]: import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl
```

Universos y variables

```
In [23]: bateria = ctrl.Antecedent(np.arange(0, 100.1, 0.1), 'bateria')      # %
distancia = ctrl.Antecedent(np.arange(0, 50.1, 0.1), 'distancia')      # m
congestion= ctrl.Antecedent(np.arange(0, 100.1, 0.1), 'congestion')      # %
urgencia = ctrl.Antecedent(np.arange(0, 10.1, 0.1), 'urgencia')      # 0-10

velocidad = ctrl.Consequent(np.arange(0, 1.51, 0.01), 'velocidad')      # m/s
```

Funciones de pertenencia

Las zonas trapezoidales en los extremos representan zonas en las que no hay duda y que serán siempre así (p. ej., congestión muy alta \Rightarrow bajar a tope) esto trae menos sensibilidad a pequeños cambios.

Las triangulares en medio para una transición lineal y fácil de interpretar.

```
In [24]: # bateria (%)
bateria['muy_baja'] = fuzz.trapmf(bateria.universe, [0, 0, 10, 20])
bateria['baja']      = fuzz.trimf(bateria.universe, [10, 25, 40])
bateria['media']     = fuzz.trimf(bateria.universe, [35, 55, 75])
bateria['alta']      = fuzz.trapmf(bateria.universe, [70, 85, 100, 100])

# distancia (m)
distancia['cerca']  = fuzz.trapmf(distancia.universe, [0, 0, 4, 8])
distancia['media']   = fuzz.trimf(distancia.universe, [5, 18, 30])
distancia['lejos']   = fuzz.trapmf(distancia.universe, [25, 35, 50, 50])

# congestion (%)
congestion['baja']  = fuzz.trapmf(congestion.universe, [0, 0, 15, 30])
```

```

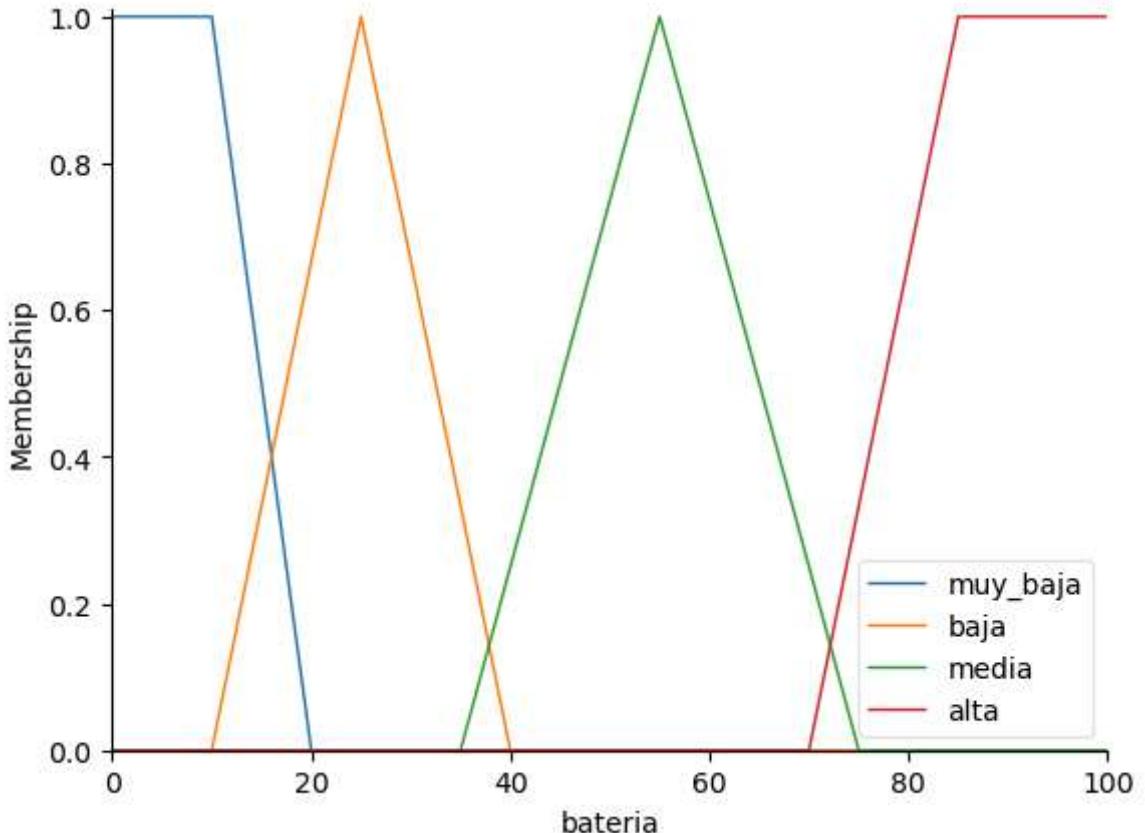
congestion['media'] = fuzz.trimf(congestion.universe, [20, 45, 70])
congestion['alta'] = fuzz.trapmf(congestion.universe, [60, 75, 100, 100])

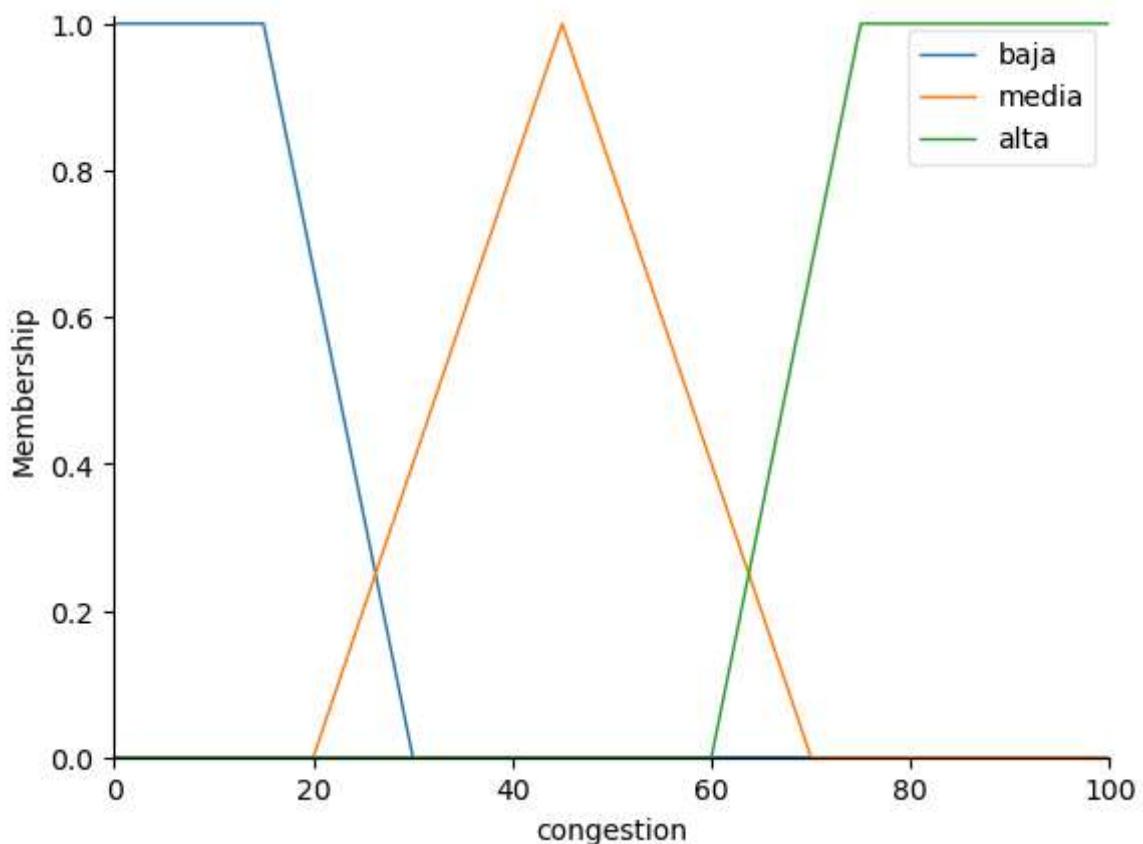
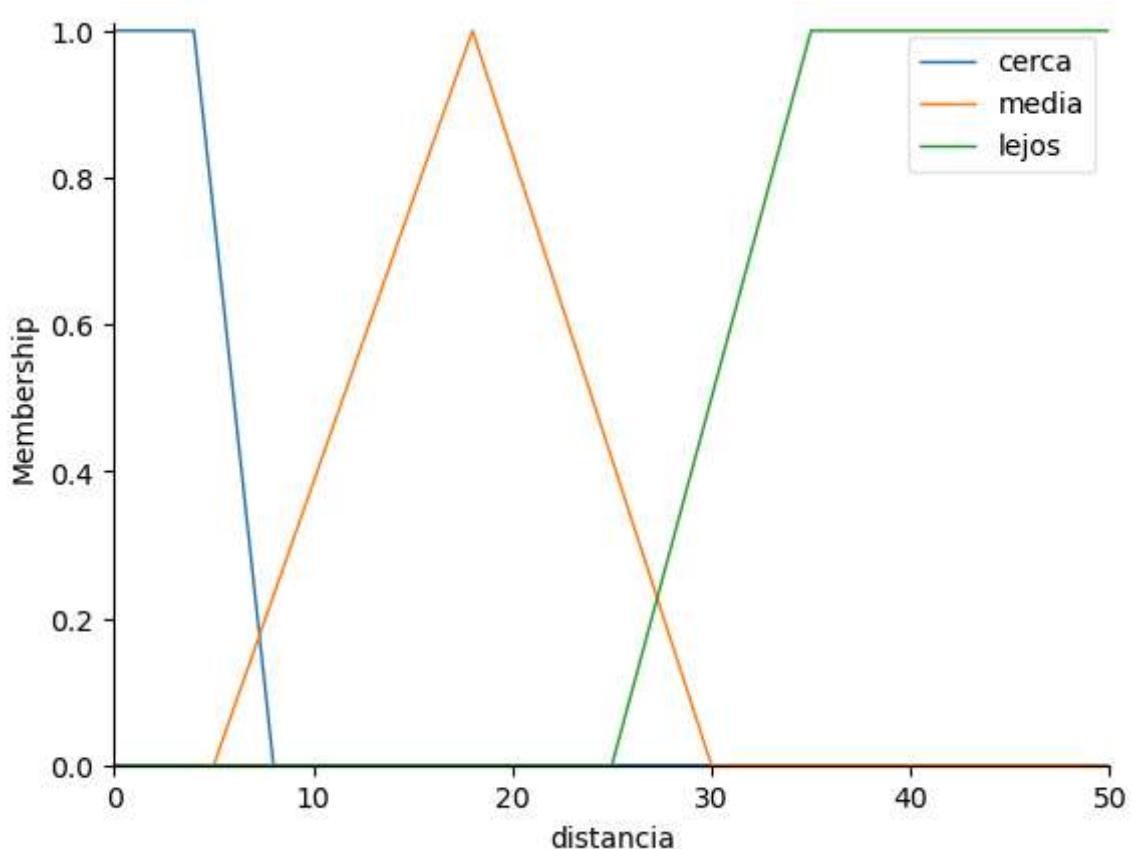
# urgencia (0-10)
urgencia['baja'] = fuzz.trapmf(urgencia.universe, [0, 0, 2, 4])
urgencia['media'] = fuzz.trimf(urgencia.universe, [3, 5.5, 7])
urgencia['alta'] = fuzz.trimf(urgencia.universe, [6, 7.5, 9])
urgencia['critica'] = fuzz.trapmf(urgencia.universe, [8.5, 9.2, 10, 10])

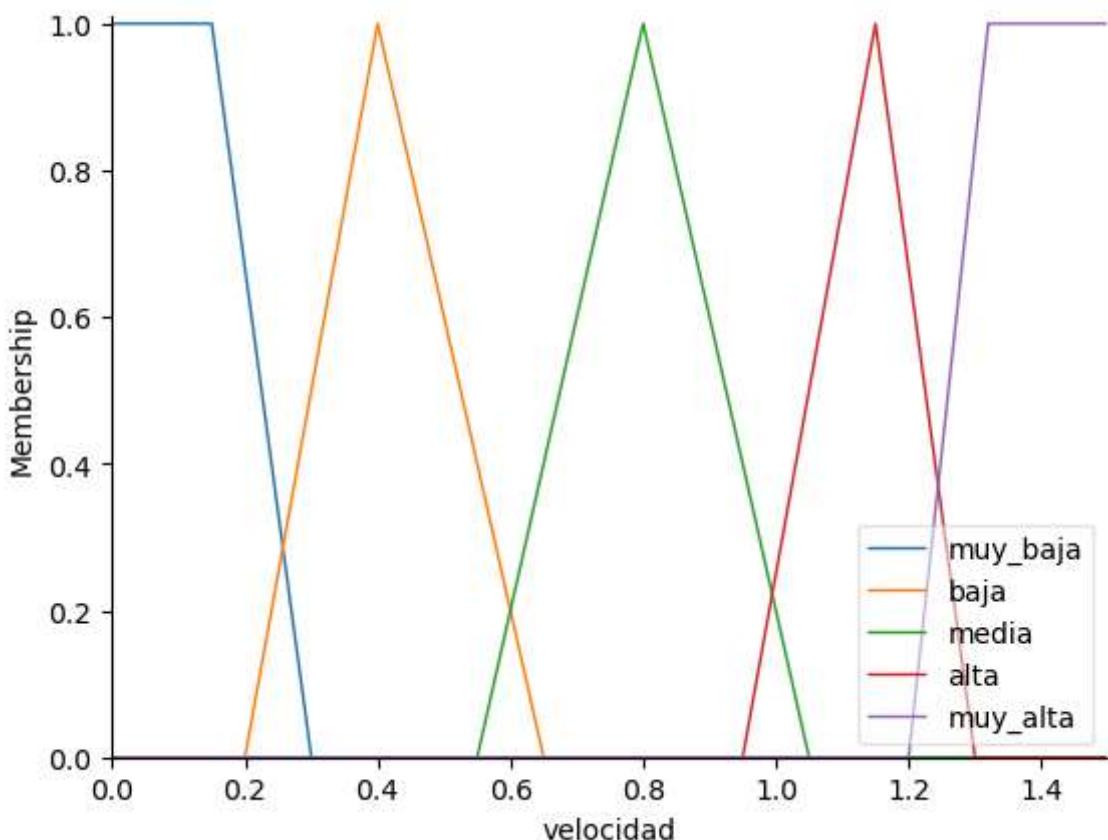
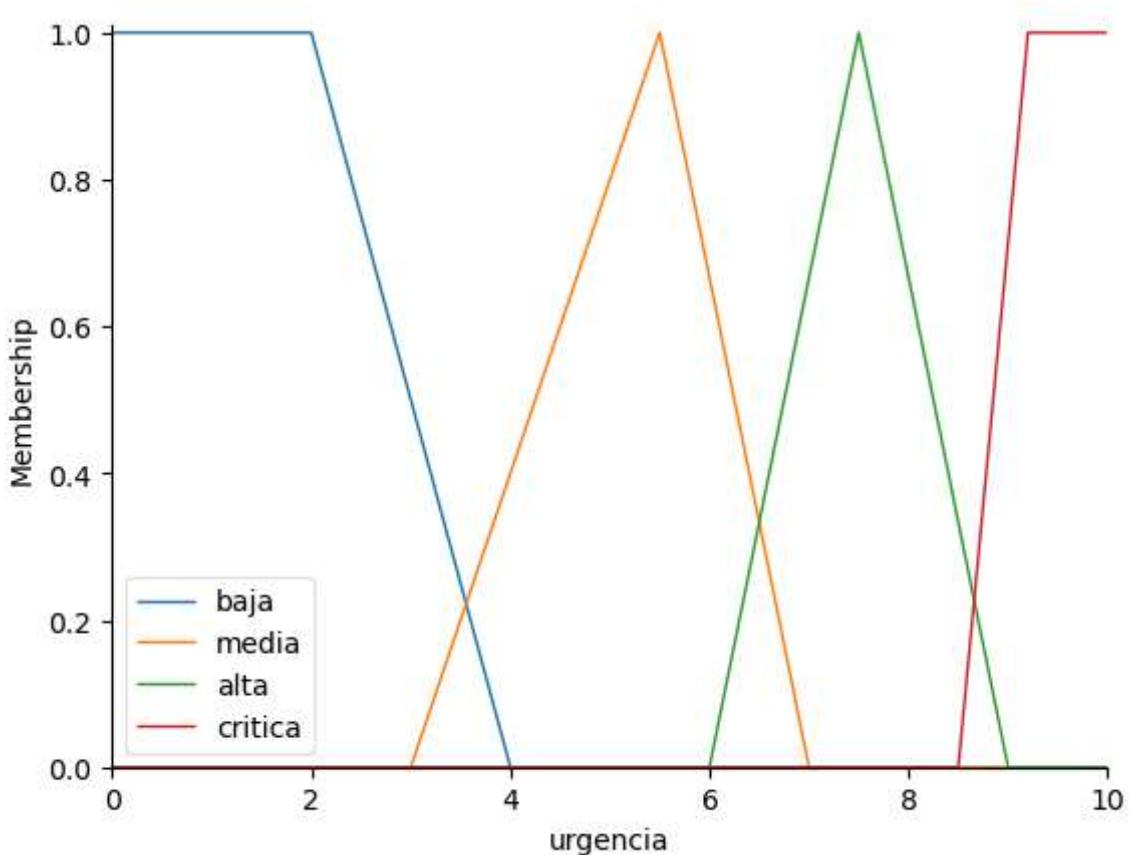
# velocidad (m/s)
velocidad['muy_baja'] = fuzz.trapmf(velocidad.universe, [0.00, 0.00, 0.15, 0.30])
velocidad['baja'] = fuzz.trimf(velocidad.universe, [0.20, 0.40, 0.65])
velocidad['media'] = fuzz.trimf(velocidad.universe, [0.55, 0.80, 1.05])
velocidad['alta'] = fuzz.trimf(velocidad.universe, [0.95, 1.15, 1.30])
velocidad['muy_alta'] = fuzz.trapmf(velocidad.universe, [1.20, 1.32, 1.50, 1.50])

bateria.view()
distancia.view()
congestion.view()
urgencia.view()
velocidad.view()

```







Reglas

Las reglas 1,2,3 se usaran como medida de seguridad, usando una velocidad muy baja o baja. La regla 5 aun teniendo las características como para usar una velocidad alta, se usara una media debido a la precaucion cerca del destino. Las demás estan pensadas para todo lo eficiente posible mientras el entorno lo permita.

```
In [25]: regla1 = ctrl.Rule(congestion['alta'], velocidad['muy_baja'])
regla2 = ctrl.Rule(distancia['cerca'] & (congestion['media'] | congestion['alta']), velocidad['muy_baja'])

regla3 = ctrl.Rule(bateria['muy_baja'], velocidad['baja'])

regla4 = ctrl.Rule(urgencia['media'] & congestion['baja'] & distancia['media'], velocidad['media'])
regla5 = ctrl.Rule(distancia['cerca'] & congestion['baja'] & (urgencia['alta'] | velocidad['media']))
regla6 = ctrl.Rule(bateria['baja'] & urgencia['baja'], velocidad['baja'])
regla7 = ctrl.Rule(bateria['alta'] & congestion['baja'] & distancia['lejos'] & velocidad['alta'], velocidad['media'])

regla11 = ctrl.Rule(bateria['media'] & congestion['baja'] & distancia['lejos'] & velocidad['media'])

regla8 = ctrl.Rule(urgencia['alta'] & congestion['media'] & distancia['media'] & velocidad['alta'])
regla9 = ctrl.Rule(bateria['media'] & congestion['baja'] & distancia['lejos'] & velocidad['alta'])

regla10 = ctrl.Rule(urgencia['critica'] & congestion['baja'] & distancia['lejos'] & velocidad['muy_alta'])

velocidad_ctrl = ctrl.ControlSystem([
    regla1, regla2, regla3, regla4, regla5,
    regla6, regla7, regla8, regla9, regla10, regla11])

#Defuzzificacion por defecto
funcion_velocidad = ctrl.ControlSystemSimulation(velocidad_ctrl)
```

Inferencias: (Bateria, Distancia, Congestion, Urgencia)

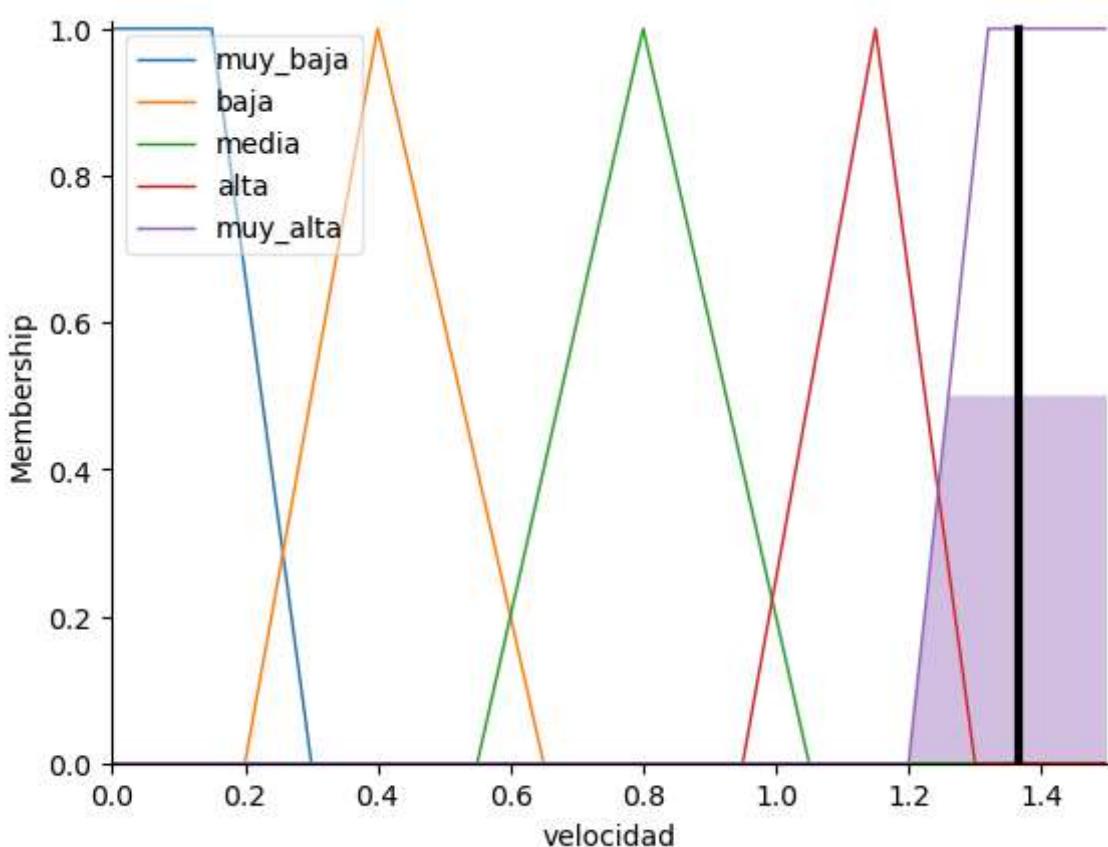
Caso A (90, 30, 10, 9.5): (batería alta, lejos, despejado, urgente) Velocidad esperada muy_alta

Caso B (55, 5, 80, 6): (congestión alta y proximidad) Velocidad esperada muy_baja.

Caso C (60, 35, 15, 3): (sin prisa, lejos, despejado), Velocidad esperada media.

```
In [26]: #CASO A
# Urgente, despejado, lejos, batería alta
funcion_velocidad.input['bateria'] = 90
funcion_velocidad.input['distancia'] = 30
funcion_velocidad.input['congestion'] = 10
funcion_velocidad.input['urgencia'] = 9.5

funcion_velocidad.compute()
velocidad.view(sim=funcion_velocidad)
```

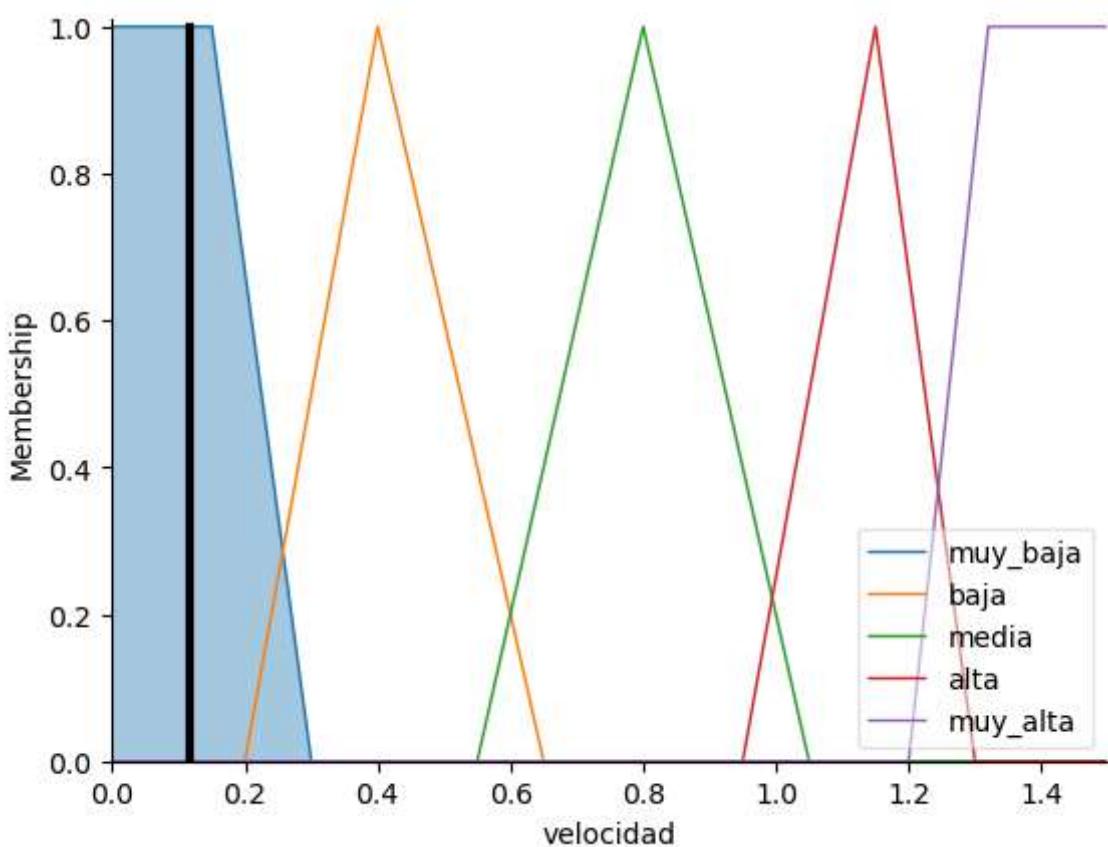


Escenario con urgencia crítica, pasillo fluido y batería alta. Se espera una velocidad muy alta ($\approx 1.3-1.5$ m/s) activada principalmente por la regla 3, ya que las condiciones son óptimas para priorizar eficiencia.

```
In [27]: # CASO B
# Congestión alta y cerca del destino

funcion_velocidad.input['bateria'] = 55
funcion_velocidad.input['distancia'] = 5
funcion_velocidad.input['congestion'] = 80
funcion_velocidad.input['urgencia'] = 6

funcion_velocidad.compute()
velocidad.view(sim=funcion_velocidad)
```



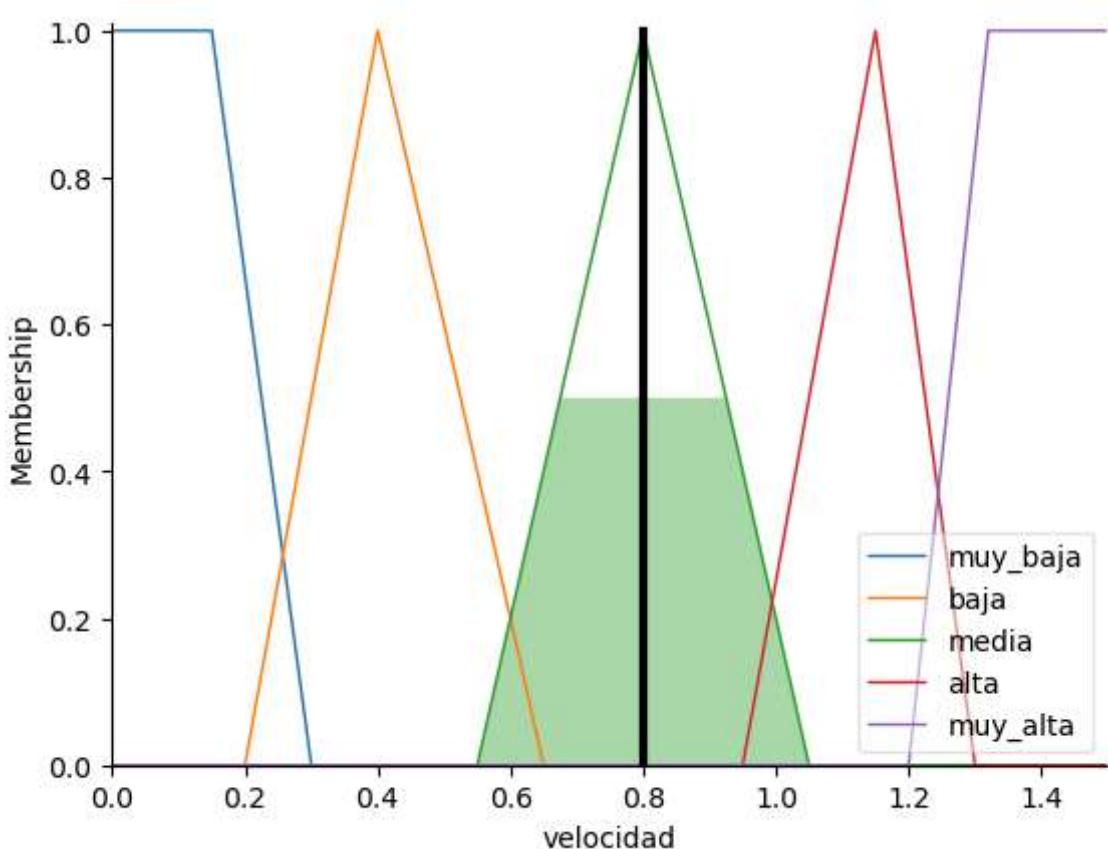
El robot está próximo a su destino y el pasillo presenta congestión alta. La lógica difusa aplica las reglas 1 y 2, priorizando seguridad → salida muy baja ($\approx 0.1\text{--}0.3$ m/s).

In [28]:

```
# CASO C
# Sin prisa, pasillo fluido, Lejos, batería media
funcion_velocidad = ctrl.ControlSystemSimulation(velocidad_ctrl)

funcion_velocidad.input['bateria'] = 60
funcion_velocidad.input['distancia'] = 35
funcion_velocidad.input['congestion'] = 15
funcion_velocidad.input['urgencia'] = 3

funcion_velocidad.compute()
velocidad.view(sim=funcion_velocidad)
```



Situación estable y sin urgencia. El robot mantiene un ritmo intermedio (media, $\approx 0.7\text{--}1.0$ m/s), equilibrando eficiencia y consumo energético.

Planificación

Reflexionad sobre si habéis seguido la planificación que definisteis en el Entregable CG.2. Explicad qué aspectos se cumplieron, cuáles no, y las razones que llevaron a cualquier desviación.

Aunque la planificación inicial establecía un trabajo distribuido entre el 6 y el 20 de octubre (definir variables, implementar reglas, probar e ir documentando), en la práctica comenzamos con retraso debido a la acumulación de entregas de otras asignaturas. No pudimos centrarnos plenamente en la práctica Fuzzy hasta el día 16, realizando en pocos días todas las fases: diseño de variables y conjuntos, implementación de las reglas en scikit-fuzzy, pruebas con tres inferencias y elaboración de las gráficas y la documentación.

Este retraso nos mostró que deberíamos haber ajustado mejor los márgenes de la planificación inicial, reduciendo los intervalos entre tareas y dedicando más tiempo a las pruebas y ajustes del sistema en lugar de dejarlo para el final. Aun así, el desarrollo se completó dentro del plazo. Para futuras entregas planeamos distribuir mejor el trabajo y prever tiempo extra por lo que pueda pasar.

Principales dudas y dificultades encontrados en el desarrollo

Durante el desarrollo surgieron principalmente dudas relacionadas con la activación de reglas y la visualización de los resultados. En algunos casos, al ejecutar las inferencias, el sistema no mostraba la línea vertical en el gráfico, lo que descubrimos que se debía a entradas que no activaban ninguna regla. Esto se solucionó ajustando los valores de los casos de prueba o añadiendo reglas complementarias para cubrir esos escenarios.

Ademas de que fue necesario invertir tiempo en ajustar los rangos de las funciones de pertenencia para evitar solapamientos excesivos o huecos sin cobertura.

También tuvimos algunas dificultades técnicas con la exportación del cuaderno a PDF, ya que requería TeX, por lo que optamos por generar el archivo en formato HTML y convertirlo después a PDF.