Ficha de Trabalho nº 4

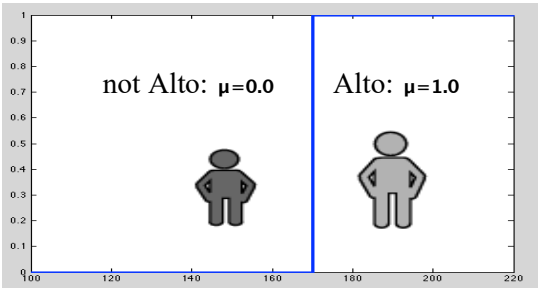
Lógica Difusa: fuzzy toolbox do Matlab

1. Introdução

A lógica difusa é uma extensão da lógica booleana. Enquanto na lógica booleana tudo se resume a Verdadeiro e Falso, na lógica difusa é possível, através da utilização de variáveis linguísticas, criar intervalos difusos, em que se pode introduzir um "Talvez".

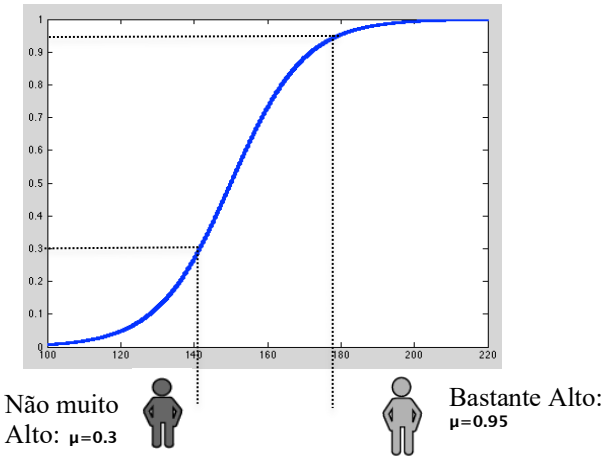
Um conjunto difuso permite definir graus de pertença: se um objeto tem um grau de pertença igual a 1 então pertence inteiramente a esse conjunto. Se um objeto tiver grau de pertença igual a zero, então não pertence ao conjunto. Qualquer valor entre 0 e 1 indicam o grau de pertença parcial de um objeto a um conjunto.

Por exemplo, na lógica binária, pode dizer-se que uma pessoa é **alta** se tiver mais do que 1.70m, caso contrário é uma pessoa **baixa**.

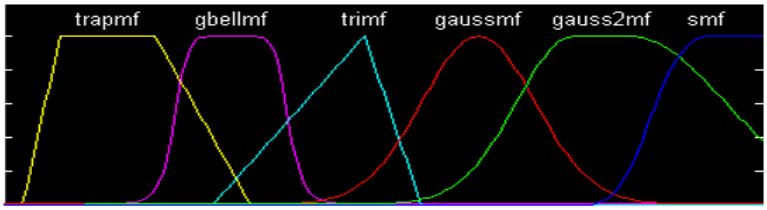


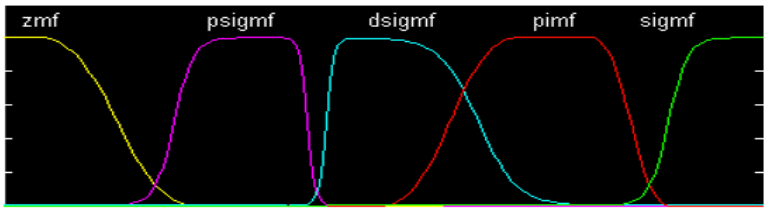
Assim, uma pessoa com 1.69 é considerada baixa, apesar da margem que a afasta do critério de separação ser muito pequena.

Um conjunto difuso permite uma abordagem mais correta para este caso. Através de valores de grau de pertença, é possível definir graus de altura diferentes: muito alto, pouco alto, etc.



O grau de pertença pode ser definido por diferentes funções de pertença:





Através da definição do domínio das variáveis, dos seus graus de pertença é possível fazer inferência difusa através da construção de regras IF...THEN...

2. Sistemas de Inferência *Fuzzy*

Os sistemas de inferência efetuam o mapeamento entre as entradas e as saídas, utilizando lógica fuzzy para lidar com afirmações às quais estão associados determinados graus de verdade. Recorrem a funções de pertença, operações lógicas fuzzy e regras fuzzy para determinar o valor da saída. O processamento decorre em 5 passos:

i) **Fuzzificação dos inputs**: todas as entradas são convertidas para valores entre [0, 1] (o seu grau de verdade), de acordo com o especificado nas funções de pertença;

ii) **Aplicação de operadores lógicos aos antecedentes das regras**: O grau de verdade dos antecedentes das regras constituídos por várias componentes é calculado utilizando os operadores lógicos habituais, expandidos para lidar com lógia fuzzy. Por defeito:
   a. AND(A, B): min(A,B)
   b. OR(A,B): max(A, B)
   c. NOT(A): 1-A

iii) **Aplicação das regras fuzzy**: todas as regras fuzzy são ativadas e cada conjunto fuzzy resultante é ajustado em função do grau de verdade do respetivo antecedente. Por defeito, é aplicada a função *min* que trunca o resultado.

iv) **Agregação dos resultados**: Todos os conjuntos fuzzy que resultam da aplicação das regras são agrupados, resultando num único conjunto fuzzy para cada output.

v) **Defuzzificação do output**: O conjunto que resulta do ponto anterior é transformado num único valor, representando o seu ponto médio. Por defeito, é utilizada a função que calcula o centroide.

3. Toolbox *fuzzy* do Matlab

Em Matlab, a toolbox fuzzy possui um conjunto de funções que permite modelar todo o sistema de Lógica difusa.
Também possui um interface gráfico para construção de um modelo fuzzy que poderá explorar usando o comando: *fuzzyLogicDesigner*.

4. Descrição do problema

Para exemplificar o uso da *toolbox fuzzy*, considere o seguinte problema. Como determinar a gorjeta adequada face ao serviço prestado por um restaurante nos EUA?
Vamos assumir dois critérios para a decidir qual o valor a atribuir:

- Qualidade do serviço (medido entre 0 e 10)
- Qualidade da comida (medida entre 0 e 10)

A variável **serviço** será usada com 3 valores: fraco, bom, excelente
A variável **comida** será usada com 2 valores: má, deliciosa
A variável **gorjeta** será usada com 3 valores: fraca, média, generosa

A definição destes conceitos será concretizada com as funções de pertença escolhidas e respetivos parâmetros.

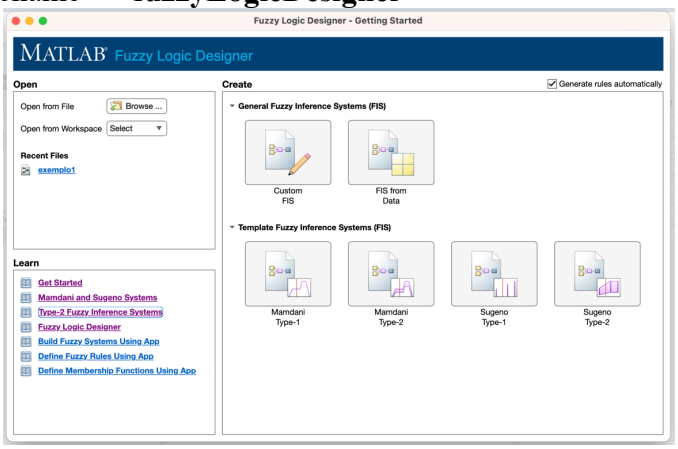
Analisando a opinião de vários especialistas, obtiveram-se as seguintes regras:

- Se o **serviço** é fraco ou a **comida** má, então a gorjeta é fraca
- Se o **serviço** é bom, então a gorjeta é média
- Se o **serviço** é excelente ou a **comida** é deliciosa, então a gorjeta é generosa

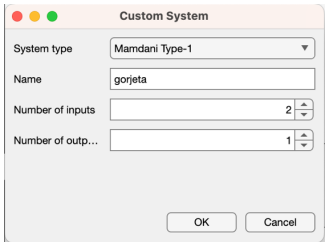
Assume-se que uma **gorjeta fraca** é de 5%, uma **gorjeta média** é de 15% e uma **gorjeta generosa** é de 25% (% relativas ao valor da conta)

5. Resolver o problema usando o fuzzyLogicDesigner

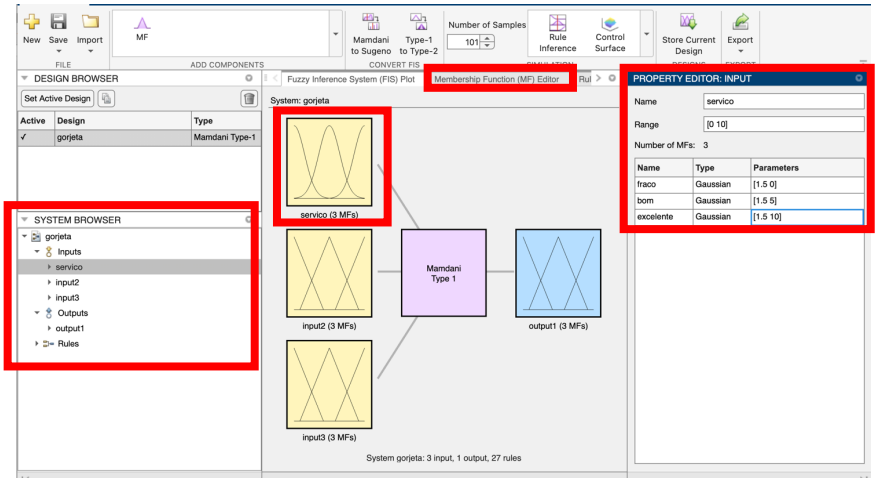
1. Na linha de comando chame >> **fuzzyLogicDesigner**



2. Escolha a opção: **Custom FIS** e preencha os campos para resolver o problema da gorjeta descrito anteriormente.



3. Usando a interface, explore o separador **Membership Function Editor** e defina as 2 variáveis de entrada e a variável de saída
   Na figura abaixo já se encontra definida a variável **serviço**.



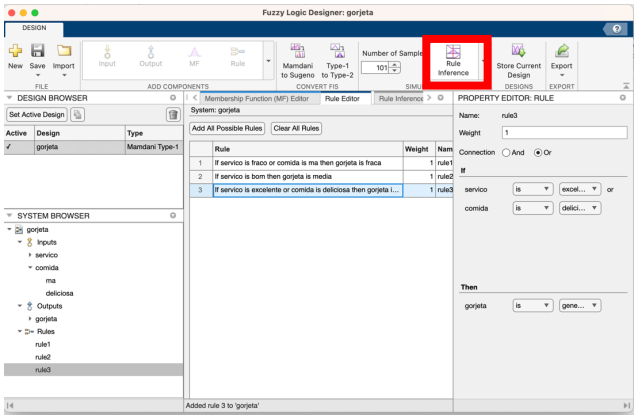
Complete o sistema para as restantes variáveis. Propõem-se os seguintes termos linguísticos, domínios, e funções de pertença:
serviço: domínio [0, 10]
: termos linguísticos – fraco, bom, excelente
: função ***gaussiana*** com os seguintes parâmetros
Fraco: [1.5 0], bom: [1.5 5], excelente: [1.5 10]

comida: domínio [0, 10]
: termos linguísticos – má, deliciosa
: função ***trapezoidal*** com os seguintes parâmetros
ma: [0 0 1 3], deliciosa: [7 9 11 19]

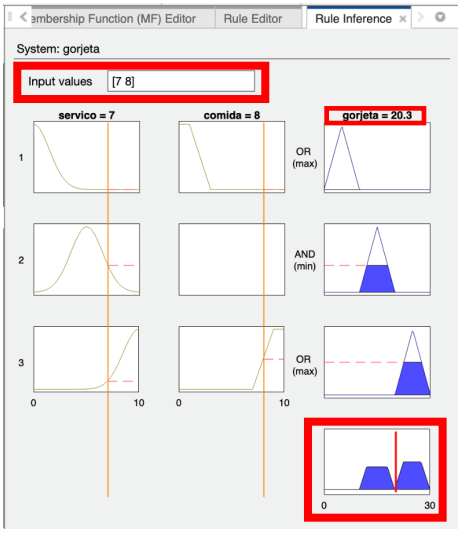
gorjeta: domínio [0, 30]
: termos linguísticos – fraca, media, generosa
: função **triangular** com os seguintes parâmetros
fraca: [0 5 10], media: [10 15 20], generosa: [20 25 30]

4. Usando a interface, explore o separador **Rule Editor**
   Limpe todas as regras usando ***Clear all rules***
   Adicione as seguintes regras
   - Se o **serviço** é fraco ou a **comida** má, então a gorjeta é fraca
   - Se o **serviço** é bom, então a gorjeta é média
   - Se o **serviço** é excelente ou a **comida** é deliciosa, então a gorjeta é generosa

5. Usando a interface, explore o botão **Rule Inference**



Teste as três regras com algumas entradas e veja a forma como é feita a *fuzzificação*, *agregação* e *defuzifficação*. Abaixo encontra-se o exemplo de entrada [serviço: 7 ; comida: 8].
As regras 2 e 3 disparam, veja a agregação das áreas obtidas por cada uma das regras anteriores. A defuzzificação é feita pelo método do centroide e dá o valor de gorjeta de **20,3.**



Agregação das áreas dadas pelas regras 2 e 3

6. Resolver o problema usando as Funções da toolbox

Funções Matlab

A forma algorítmica para execução de um programa de lógica difusa é o seguinte:

- **Passo 1:** Criar a estrutura *fuzzy* usando a função **mamfis**

  `a = mamfis;` cria uma estrutura FIS do tipo Mamdani de nome `a`

- **Passo 2:** Criar as variáveis linguísticas de entrada e de saída usando as funções **addInput/addOutput**

  ```
  a = addInput(a, varBounds,'Name', varName);
  a = addOutput(a varBounds,'Name', varName);
  ```

  - **a: nome da estrutura FIS**
  - **varName: nome da variável (por exemplo, 'comida')**
  - **varBounds: limites do domínio da variável (por exemplo, [0 5])**

  ```
  a = addInput(a,[0,10],'Name',"comida");
  a = addOutput(a,[0,30],'Name',"gorjeta");
  ```

  Os domínios das variáveis de entrada são iguais: [0 10]

- **Passo 3:** Escolher as funções de pertença e determinar os domínios de entrada e saída usando a função **addmf**

  ```
  a = addMF(a, varName, mfType,mfParams, 'Name', mfName)
  ```

  - **a: nome da estrutura FIS**
  - **varName: nome da variável (por exemplo, 'comida')**
  - **mfType: tipo de função de pertença ('gaussmf', 'trapmf', 'trimf', etc)**
  - **mfParams: parâmetros para a função de pertença**
  - **mfName: nome da função de pertença para as variáveis linguísticas (por exemplo, 'boa', 'fraca', 'excelente')**

  ```
  a = addMF(a,"servico", "gaussmf",[1.5 0],'Name', "fraco");
  ```

  Utilize as seguintes funções de pertença:

  Variável serviço: *gaussmf*

  Parâmetros:

  ```
  "fraco": [1.5 0], "bom": [1.5 5], "excelente": [1.5 10]
  ```

  Variável comida: *trapmf*

  Parâmetros:

  ```
  "ma": [0 0 1 3], "deliciosa: [7 9 11 19]
  ```

  Variável gorjeta: *trimf*

  Parâmetros:

  ```
  "fraca": [0 5 10], "media": [10 15 20], "generosa": [20 25 30]
  ```

- **Passo 4:** Criar as regras fuzzy para as funções de pertença e usar a função **addrule** para inserir as regras no programa.

  regra1 = "entrada1 == MF1 | entrada2 == MF2 => saída = MF3";

exemplo: regra1 = "servico==fraco | comida==ma => gorjeta=fraca";

Depois de criada a matriz de regras chamar a função addrule:

```
regras =[regra1 regra2 … ];
a = addrule(a,regras);
```

- **Passo 5:** Avaliar a resposta fuzzy através da função **`evalfis`**

```
out = evalfis(a, entrada);
```

  - **`a: nome da estrutura FIS`**
  - **`entrada: vetor com valores para as variáveis de entrada`**

Programar o exemplo da gorjeta

Crie uma nova função de nome gorjeta e grave-a no disco com o mesmo nome
O esqueleto da função é o seguinte (disponível no moodle):

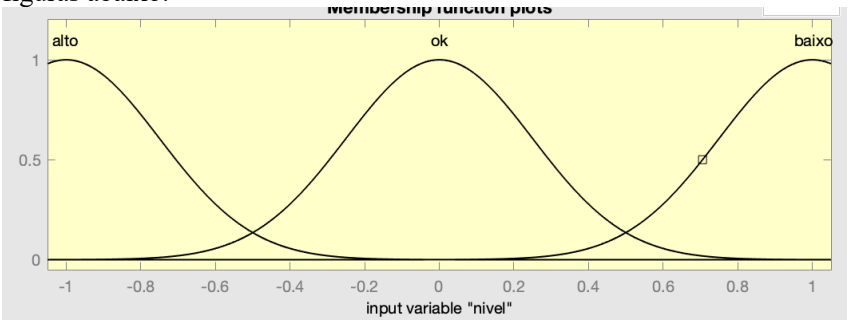
```
function [fis] = gorjeta()

          %PASSO 1 criar sistema fis
          fis = mamfis;

          %PASSO 2 VARIÁVEIS
          fis = addInput(fis,[0,10],'Name',"servico");
        %COMPLETAR

          %PASSO 3 FUNÇÕES DE PERTENÇA
          fis = addMF(fis,"servico", "gaussmf",[1.5 0],'Name', "fraco");
          %COMPLETAR para as três variáveis

          %PASSO 4 REGRAS
          regra1 = "servico==fraco | comida==ma => gorjeta=fraca";
          regra2 = %COMPLETAR
          regra3 = %COMPLETAR
          regras=[regra1 regra2 regra3];
          fis = addRule(fis,regras);

          %PASSO 5: avaliar para vários valores de servico e comida com evalfis
          for servico=0:10
            for comida=0:10
            entrada=[servico comida];
            out = evalfis(fis, entrada);
           fprintf('serviço = %d\nComida = %d\nGorjeta = %f\n\n',servico, comida, out);
          end
        end
end
```

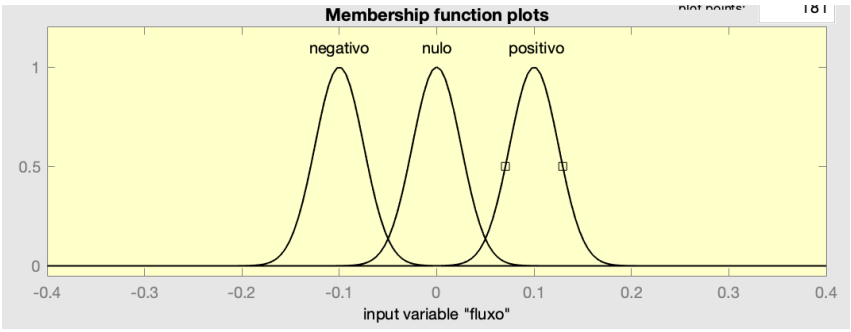
Complete a função, usando a informação fornecida. Execute a função e analise os resultados

Tarefa: Adicione mais uma variável de entrada: **tempo de espera**.
Escolha os termos linguísticos, domínios e funções de pertença adequadas. Crie regras adicionais e/ou altere as regras existentes incluindo esta variável. Faça alguns testes.

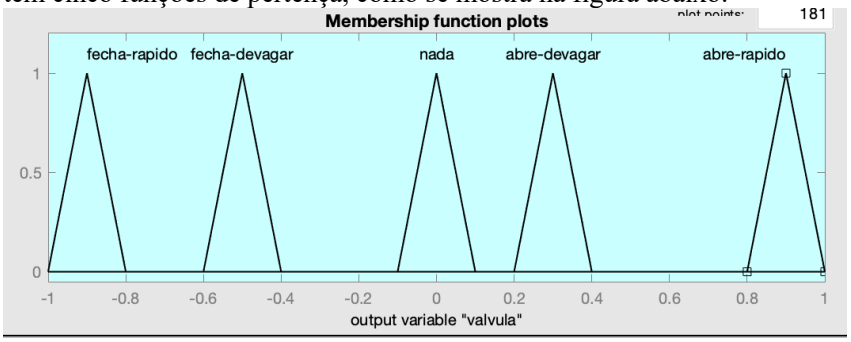
Exercício 1 - Controlador da água de um tanque

Neste exercício pretende-se utilizar um sistema de lógica difusa para a implementação de um modelo que controla o nível de água em um tanque usando um sistema de inferência *fuzzy*.
As duas entradas para o sistema difuso são o erro do nível da água (variável nível de domínio [-1, 1]) e a taxa de variação do nível da água (variável fluxo de domínio [-0.4, 0.4]). Cada entrada possui três termos linguísticos como se mostra nas figuras abaixo:





A saída do sistema *fuzzy* é a taxa na qual a válvula de controle abre ou fecha (variável válvula de domínio [-1, 1]), que tem cinco funções de pertença, como se mostra na figura abaixo:



Devido ao diâmetro do tubo de escoamento, o tanque de água neste sistema esvazia mais lentamente do que enche. Para compensar esse desequilíbrio, as funções de associação de válvula ***fecha_devagar*** e ***abre_devagar*** não são simétricas.
O sistema fuzzy tem cinco regras. As três primeiras regras ajustam a válvula com base apenas no erro do nível de água.

- Se o nível da água estiver bom, não ajuste a válvula.
- Se o nível da água estiver baixo, abra a válvula rapidamente.
- Se o nível da água estiver alto, feche a válvula rapidamente.

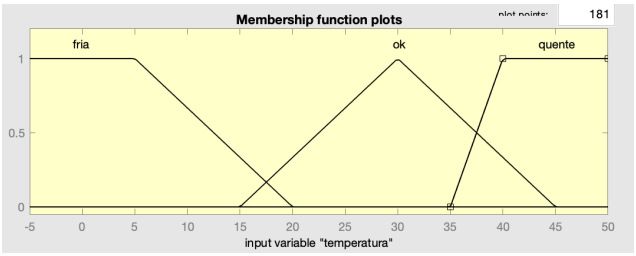
As outras duas regras ajustam a válvula com base na taxa de mudança do nível da água quando o nível da água está perto do ponto de ajuste.

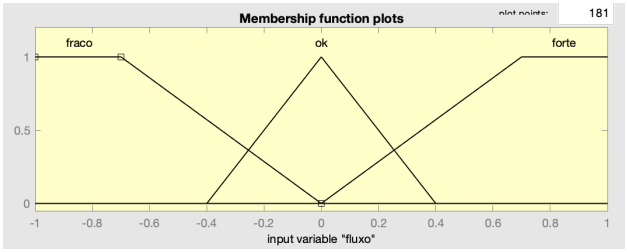
- Se o nível da água estiver bom e aumentando (positivo), feche a válvula lentamente.
- Se o nível da água estiver bom e diminuindo (negativo), abra a válvula lentamente.

Usando as funções da toolbox fuzzy do Matlab, implemente o sistema descrito anteriormente. Teste o sistema e avalie os resultados.

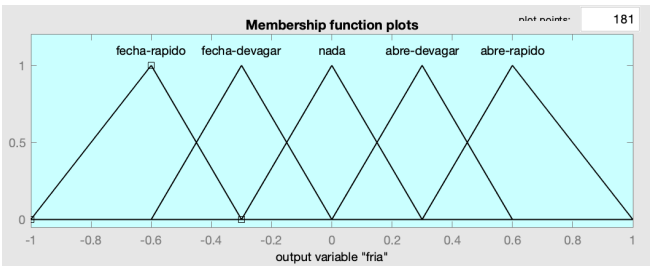
Exercício 2 - Controlador da temperatura de um chuveiro

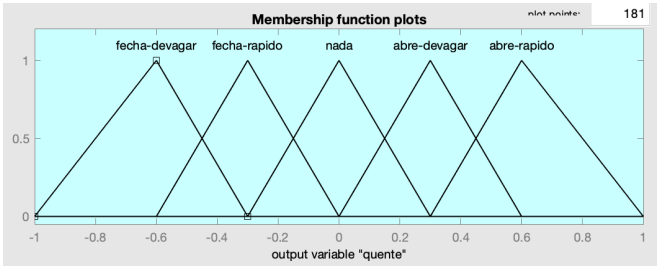
Neste exercício pretende-se utilizar um sistema de lógica difusa para a implementação de um modelo para controlo da temperatura de um chuveiro, através de um sistema de inferência fuzzy. Para tal sistema, o sistema ajusta a taxa de fluxo e a temperatura de um chuveiro ajustando as torneiras de água quente e fria.
As duas entradas para o sistema difuso são a temperatura (variável temperatura de domínio [-5, 50]) e o erro de taxa de fluxo (variável fluxo de domínio [-1, 1]). Cada entrada possui três funções de pertença como se mostra abaixo.





As duas saídas do sistema difuso são a taxa na qual as torneiras de água fria e quente abrem ou fecham (domínio [-1, 1]). Cada saída possui cinco funções de associação:





O sistema fuzzy possui nove regras para ajustar as torneiras de água quente e fria com base nos erros de fluxo e na temperatura. As regras ajustam a taxa de fluxo total com base no erro de fluxo e ajustam as taxas de fluxo quente e frio relativas com base na temperatura.

- Se temperatura estiver fria e o fluxo fraco, então abre a torneira fria devagar e abre a torneira quente depressa.
- Se temperatura estiver fria e o fluxo ok, então fecha a torneira fria devagar e abre a torneira quente devagar.
- Se temperatura estiver fria e o fluxo forte, então fecha a torneira fria depressa e fecha a torneira quente devagar.
- Se temperatura estiver boa e o fluxo fraco, então abre a torneira fria devagar e abre a torneira quente devagar.
- Se temperatura estiver boa e o fluxo ok, então não altera nenhuma das torneiras.
- Se temperatura estiver boa e o fluxo forte, então fecha a torneira fria devagar e fecha a torneira quente devagar.
- Se temperatura estiver quente e o fluxo fraco, então abre a torneira fria depressa e abre a torneira quente devagar.
- Se temperatura estiver quente e o fluxo ok, então abre a torneira fria devagar e fecha a torneira quente devagar.
- Se temperatura estiver quente e o fluxo forte, então fecha a torneira fria devagar e fecha a torneira quente depressa.

Usando as funções da toolbox fuzzy do Matlab, implemente o sistema descrito anteriormente.
Teste o sistema e avalie os resultados.

Exercício 3 - Calculador para o consumo de um automóvel ligeiro

Neste exercício pretende-se utilizar um sistema de lógica difusa para a implementação do cálculo do consumo de combustível num automóvel.

Assuma que o consumo de um automóvel depende da aceleração, medida em rotações por minuto (RPM), do estado dos pneus (medida da altura do sulco do pneu) e da inclinação do terreno onde circula (medido em graus positivos ou negativos, consoante se é uma subida, ou descida, 0º corresponde a um terreno sem inclinação, uma inclinação acentuada pode considerar 30º/-30º para subidas ou descidas, respetivamente). Quanto maior for a aceleração, o pior estado dos pneus, ou a inclinação do terreno, maior será o consumo. O consumo deve ser medido em nº de litros gastos por cada 100 Km (L/100km)

Alguma informação relevante:

"A rotação de um automóvel ligeiro de maior cilindrada poderá chegar às 7000 RPM"

"Os automóveis ligeiros de 2020 mais económicos consomem desde 3,8L/100KM"

"Os pneus têm uma franja de utilização ideal entre os quatro e os oito milímetros, sendo que a legislação portuguesa aponta para uma profundidade mínima de 1,6 milímetros. Abaixo desse limite, arrisca-se a uma multa no caso de verificação pelas autoridades. Por conseguinte, também não passará na Inspeção Periódica Obrigatória (IPO)."

Implemente em Matlab o sistema de Lógica difusa descrito de seguida.

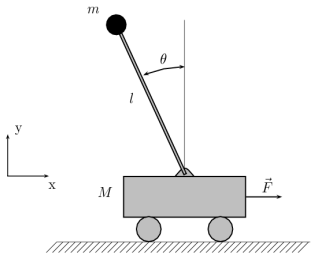
1) Defina as variáveis de **entrada** e de **saída**, e os respetivos **domínios**.
2) Para cada variável defina os **termos linguísticos**, com as **funções de pertença** que considerar mais apropriadas.
3) Defina as **regras** que achar apropriadas. Pode considerar que um consumo será mais baixo para situações em que tem baixa aceleração, pneus novos, terrenos sem inclinação, descidas, por exemplo. Pelo contrário com acelerações maiores, pneus velhos, terrenos com subidas o consumo será maior. Defina uma matriz de 8/10 regras.
4) Teste o sistema para vários valores de aceleração/inclinação e analise os resultados obtidos.

Por exemplo:

- Qual o consumo para uma aceleração de 100 RPM, pneus com 5mm, e terreno de 0º?
- Qual o consumo para uma aceleração de 5000 RPM, pneus com 1mm, e terreno de 20º?
- Qual o consumo para uma aceleração de 1000 RPM, pneus com 2.5 mm e terreno de -12º?
- ...

Exercício 4 – Problema do pêndulo invertido

Considere o problema do pêndulo invertido. O sistema consiste num carro com um pêndulo invertido que é "empurrado" por uma força **F**. O objetivo é manter uma haste em equilíbrio na posição vertical. Esta haste tem um peso na sua extremidade superior está ligada a um carro. Se a haste se movimentar para a direita ou para a esquerda, o carro deve mover-se de forma a compensar este movimento e assim conseguir manter a haste em equilíbrio.
Analisando o **valor do ângulo** e da **velocidade angular** do pêndulo, um sistema difuso pode determinar a **força** necessária a ser aplicada ao carro de forma a manter o equilíbrio.



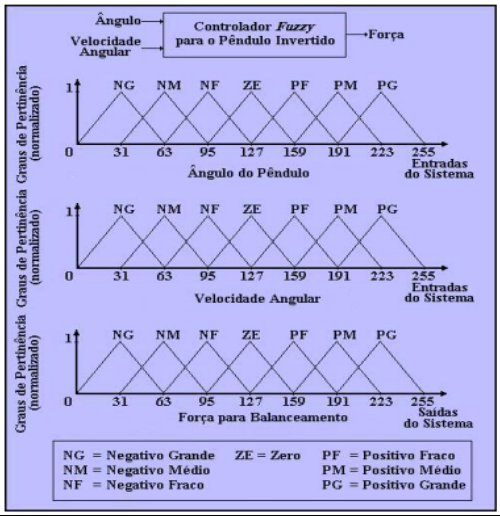
Na construção do nosso sistema FIS vamos considerar

- duas entradas: o ângulo e a velocidade angular
- uma saída: a força.

Os valores para cada variável difusa são:

- NG (Negativo Grande)
- NM (Negativo Médio)
- NF (Negativo Fraco)
- ZE (Zero)
- PF (Positivo Fraco),
- PM (Positivo Médio)
- PG (Positivo Grande)

Os domínios das 3 variáveis: [0 255]
As funções de pertença devem ser construídas de acordo com a figura/tabela:



Entrada: ângulo					Entrada: velocidade				Saída: força			
NG	000	031	063		**NG**	000	031	063	**NG**	000	031	063
NM	031	063	095		**NM**	031	063	095	**NM**	031	063	095
NF	063	095	127		**NF**	063	095	127	**NF**	063	095	127
ZE	095	127	159		**ZE**	095	127	159	**ZE**	095	127	159
PF	127	159	191		**PF**	127	159	191	**PF**	127	159	191
PM	159	191	223		**PM**	159	191	223	**PM**	159	191	223
PG	191	223	255		**PG**	191	223	255	**PG**	191	223	255

As regras de inferência são as seguintes:

```
Regra 01:  SE ângulo = NG E velocidade = ZE Então força = NG
Regra 02:  SE ângulo = NM E velocidade = ZE Então força = NM
Regra 03:  SE ângulo = NF E velocidade = ZE Então força = NF
Regra 04:  SE ângulo = NF E velocidade = PF Então força = NF
Regra 05:  SE ângulo = ZE E velocidade = NG Então força = NG
Regra 06:  SE ângulo = ZE E velocidade = NM Então força = NM
Regra 07:  SE ângulo = ZE E velocidade = NF Então força = NF
Regra 08:  SE ângulo = ZE E velocidade = ZE Então força = ZE
Regra 09:  SE ângulo = ZE E velocidade = PF Então força = PF
Regra 10:  SE ângulo = ZE E velocidade = PM Então força = PM
Regra 11:  SE ângulo = ZE E velocidade = PG Então força = PG
Regra 12:  SE ângulo = PF E velocidade = ZE Então força = PF
Regra 13:  SE ângulo = PF E velocidade = NF Então força = PF
Regra 14:  SE ângulo = PM E velocidade = ZE Então força = PM
Regra 15:  SE ângulo = PG E velocidade = ZE Então força = PG
```

Implemente o sistema difuso e grave-o com o nome **pendulo_fis.m**

Para testar, peça ao utilizador o valor do ângulo e da velocidade e registe os valores da força dados pelo sistema:

```
ang=input('ângulo (valor entre 0 e 255)');
vel=input('velocidade (valor entre 0 e 255)');
entrada=[ang vel]
out = evalfis(entrada,pendulo_fis);
fprintf(ângulo = %d\nvelocidade = %d\nforça = %f\n\n',ang, vel, out);
```

Situação	ângulo	velocidade	força?
1	63	127	63.00
2	5	126	31.48
3	100	223	222.9