Ficha de Trabalho nº 9

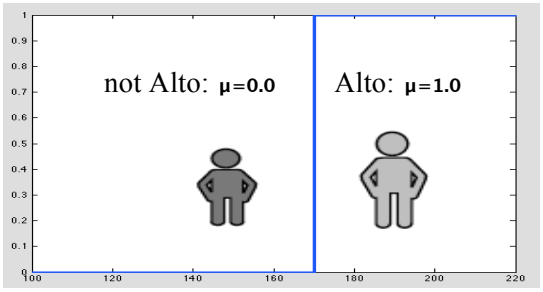
Lógica Difusa: fuzzy toolbox do Matlab

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".

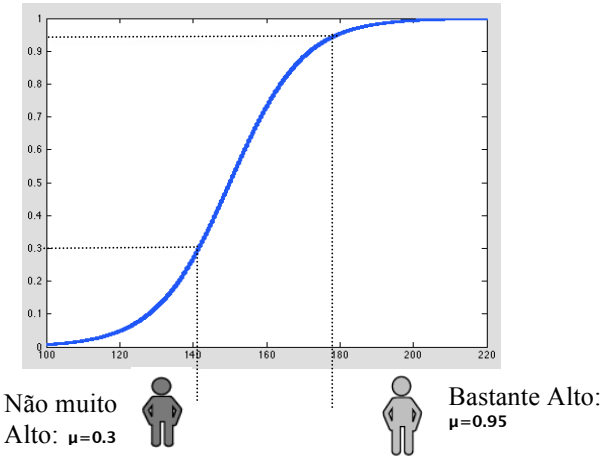
Um conjunto difuso permite definir graus de pertença: se um objeto tem um grau de pertença igual a 1 então pertence inteiramente a esse conjunto. Se um objeto tiver grau de pertença igual a zero, então não pertence ao conjunto. Qualquer valor entre 0 e 1 indicam o grau de pertença parcial de um objeto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

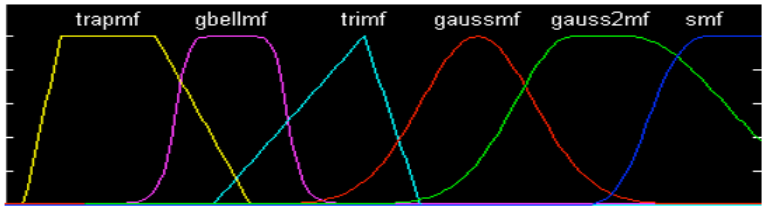


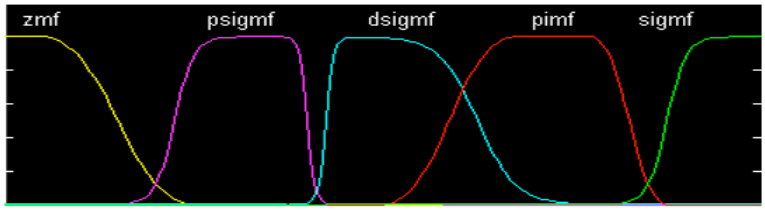
Assim, uma pessoa com 1.69 é considerada baixa, apesar da margem que a afasta do critério de separação ser muito pequena.

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 respectivo 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 centróide.

3. Toolboz *fuzzy* do Matlab

Em Matlab, a toolbox fuzzy possui o programa *fuzzy* como editor gráfico de sistemas de inferência fuzzy. Esta interface gráfica possui 5 módulos para definir o problema de forma interativa e para visualizar o processo de inferência:

- Os módulos *Fuzzy Inference System (FIS) Editor* e *Membership Function Editor* permitem definir as variáveis de input e output, especificando quais as funções de pertença mais adequadas e qual a interligação entre as variáveis linguísticas e os graus de verdade;
- O *Rule Editor* permite especificar as regras fuzzy que ligam os inputs aos outputs;
- Os módulos *Rule Viewer* e *Surface Viewer* permitem visualizar o processo de inferência.

4. Problema 1

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

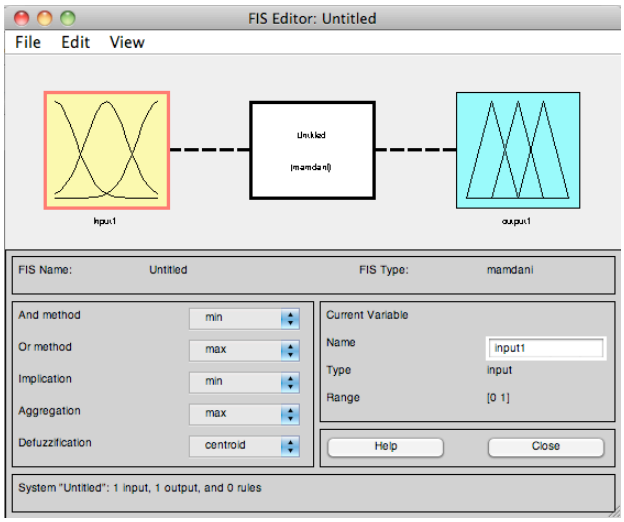
Assume-se que uma **gorjeta fraca** é de 5%, uma **gorjeta média** é de 15% e uma **gorjeta generosa** é de 25% (% relativas ao valor da conta)

4.1 Interface gráfica - Resolução Interactiva Utilizando a ToolBox Fuzzy

Na linha de comando do Matlab inicie o FIS Editor escrevendo o comando

>>**fuzzy**

Por defeito a interface gráfica é gerada com uma entrada (input1) e uma saída (output1):

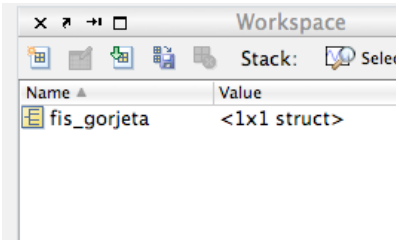


No nosso problema precisamos de duas entradas (serviço e comida) e uma saída (gorjeta). Para criar mais uma entrada escolha: **`Edit-Add Variable-Input`**

Altere o nome das variáveis:

- Clique em cima da variável ***input1*** e no campo *Name* escreva **`serviço`**. Pressione o **`ENTER`**
- Clique em cima da variável ***input2*** e no campo *Name* escreva **`comida.`** Pressione o **`ENTER`**
- Clique em cima da variável ***output1*** e no campo *Name* escreva **`gorjeta.`** Pressione o **`ENTER`**

Neste momento, o sistema criado pode ser exportado para o workspace: **`File-Export-To Workspace`**
Deve dar um nome ao modelo: **fis_gorjeta** e pressione o botão **`OK`**
No seu workspace deve surgir a variável criada:



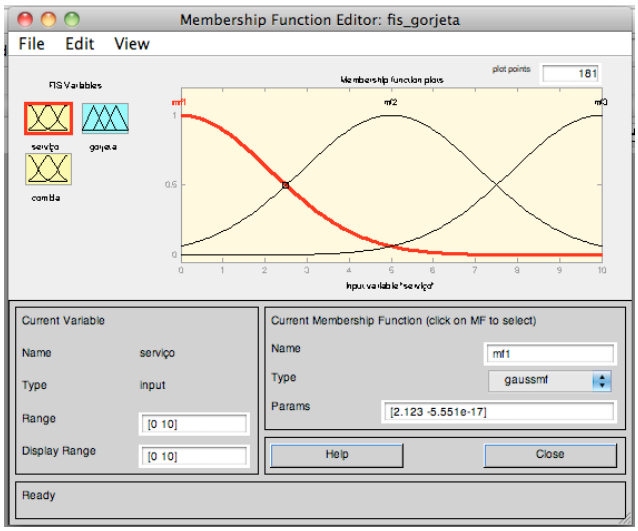
Definir funções de pertença para cada variável:

Esta etapa é feita usando o Membership Function Editor, que se acede em **`Edit - Membership Functions`**

Para a variável **`serviço:`**

- Definir o domínio (Range) e o range Display de [0 10]
- Remova todas as funções de pertença: **`Edit - Remove All MFs`**
- Defina a função de pertença: **`Edit - Add MFs`**
    - Escolha a função **`gaussmf`** e escreva **3** no campo **`Number of MFs`**

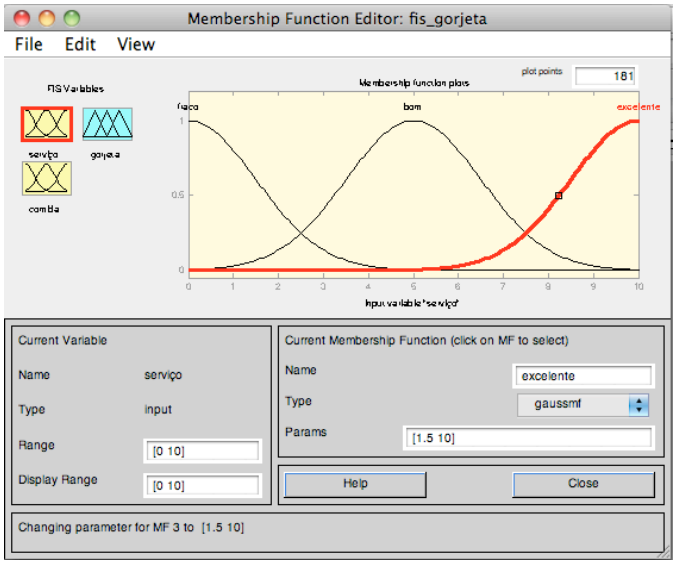
O resultado será semelhante a:



Cada uma das curvas classifica os diferentes conjuntos para a variável **serviço**: fraco, bom, excelente.

- Edite a curva **mf1**, altere o seu nome para **fraco**. No campo ***params*** escreva **[1.5 0].** Estes valores correspondem ao desvio padrão e ao centro da função gaussiana, respetivamente. Como pode ver pela curva, se o serviço for classificado com 1, o seu grau de pertença ao conjunto fraco é muito elevado. Por outro lado, um serviço classificado acima de 5 já não pertence a este conjunto.
- Edite a curva **mf2**, altere o seu nome para **bom**. No campo ***params*** escreva **[1.5 5]**.
- Edite a curva **mf3**, altere o seu nome para **excelente**. No campo ***params*** escreva **[1.5 10]**

Deve obter o seguinte resultado:



Para a variável **`comida:`**

- Definir o domínio (Range) e o range Display de [0 10]
- Remova todas as funções de pertença: **`Edit - Remove All MFs`**
- Defina a função de pertença: **`Edit - Add MFs`**
  - Escolha a função **`trapmf`** e escreva **2** no campo **`Number of MFs`**

Cada uma das curvas classifica os diferentes conjuntos para a variável **comida**: má, deliciosa

- Edite a curva **mf1**, altere o seu nome para **ma**. No campo ***params*** escreva **[0 0 1 3].** Estes valores definem a forma do trapézio, de tal forma que comida classificada com

valores inferiores ou iguais a 1 tem forte grau de pertença ao conjunto da comida má. Acima de 3, já não pertence a este conjunto.

- Edite a curva **mf2**, altere o seu nome para **deliciosa**. No campo ***params*** escreva **[7 9 11 19]**.

Para a variável **`gorjeta:`**

- Definir o domínio (Range) e o range Display de [0 30], de forma a cobrir os valores possíveis para a gorjeta (5% a 25%)
- Remova todas as funções de pertença: **`Edit - Remove All MFs`**
- Defina a função de pertença: **`Edit - Add MFs`**
  - Escolha a função **`trimf`** e escreva **3** no campo **`Number of MF`**

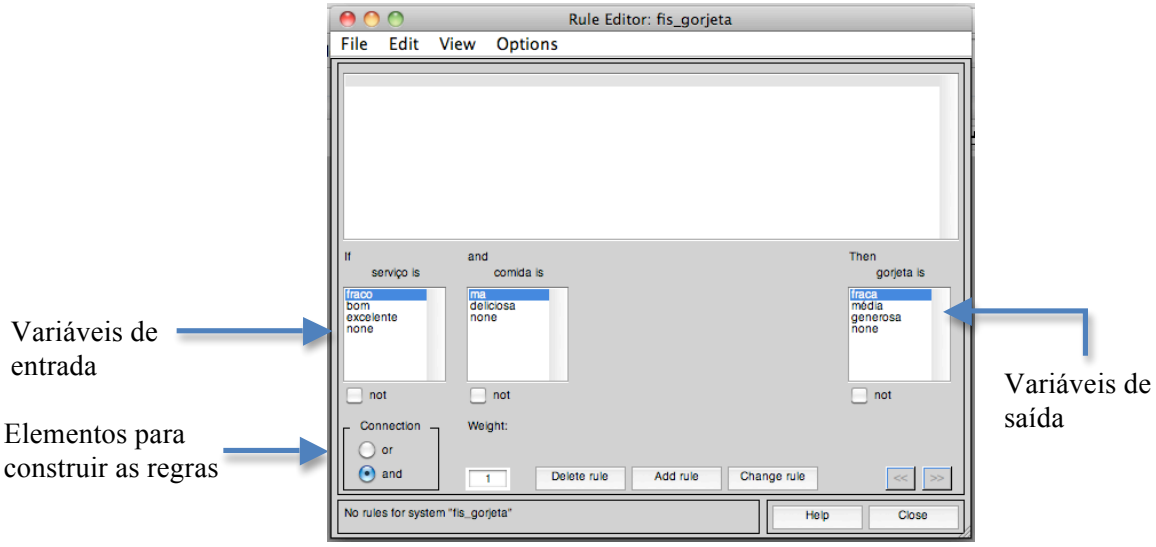
Cada uma das curvas classifica os diferentes conjuntos para a variável **gorjeta**: fraca, média, generosa.

- Edite a curva **mf1**, altere o seu nome para **fraca**. No campo ***params*** escreva **[0 5 10].** Estes valores definem a forma do triângulo, e consequentemente os graus de pertença para valores diferentes de gorjeta
- Edite a curva **mf2**, altere o seu nome para **media**. No campo ***params*** escreva **[10 15 20]**.
- Edite a curva **mf3**, altere o seu nome para **generosa**. No campo ***params*** escreva **[20 25 30]**.

Termine selecionando CLOSE.

Definir as regras:

- A definição das regras é feita usando o *Rules Editor*, que se acede em **`Edit - Rules`**



Defina a primeira regra:

`serviço = fraco`
`comida = ma`
`gorjeta = fraca`
`Conection = or`

- Clicar em **Add Rule**

Este procedimento permitiu criar a seguinte regra:

1. If (serviço is fraco) or (comida is ma) then (gorjeta is fraca) (1)

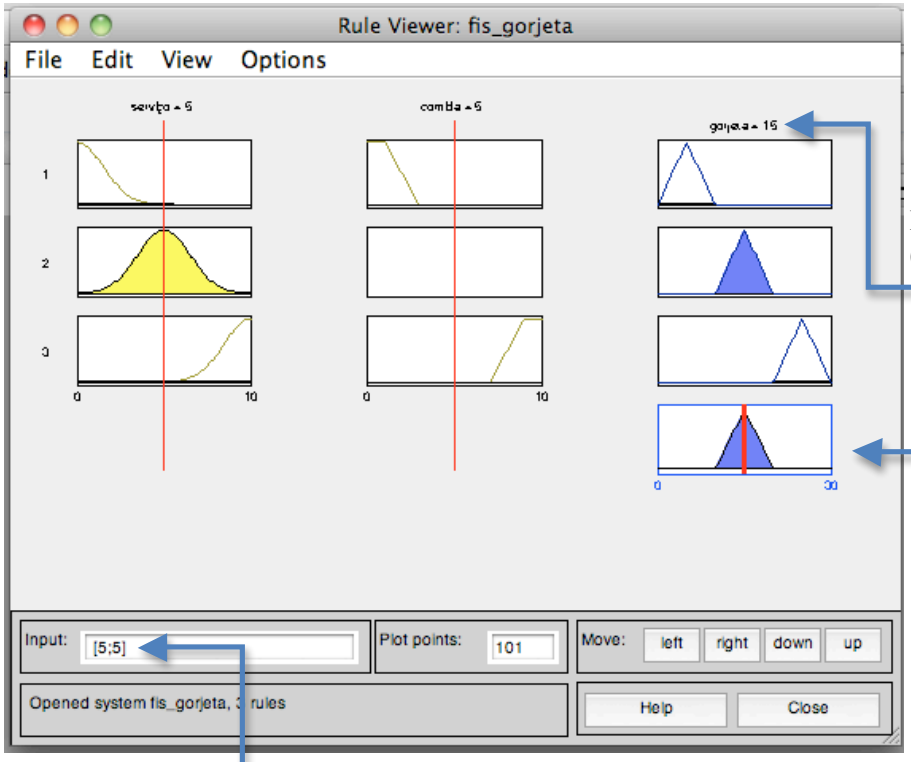
Repita o procedimento anterior de forma a criar as duas regras:

2. If (serviço is bom) then (gorjeta is média) (1)
3. If (serviço is excelente) or (comida is deliciosa) then (gorjeta is generosa) (1)

Termine selecionando CLOSE.

Visualizar as regras:

- A visualização das regras é feita usando o *Rules Viewer*, que se acede em **`View - Rules`**



Altere os valores do campo **input** [serviço, comida] e analise como a gorjeta é alterada.

Exercícios:

1. Identifique no exemplo anterior as 5 etapas de inferência fuzzy descritas no ponto 3.

2. Aplique o sistema de inferência fuzzy e verifique qual a gorjeta proposta para as seguintes entradas:
   a. {Serviço: 0, Comida: 0}
   b. {Serviço: 0, Comida: 5}
   c. {Serviço: 5, Comida: 0}
   d. {Serviço: 5, Comida: 5}
   e. {Serviço: 7.5, Comida: 7.5}
   f. {Serviço: 10, Comida: 7.5}
   g. {Serviço: 7.5, Comida: 10}
   h. {Serviço: 10, Comida: 10}

3. Altere as funções de pertença dos inputs para outras à sua escolha e verifique se existem alterações no output. Pode consultar as funções disponíveis aqui: http://www.mathworks.com/help/fuzzy/foundations-of-fuzzy-logic.html

4. Adicione a variável de input *Espera* ao sistema. Esta variável mede o tempo de espera no restaurante e pode ser classificada como {baixo, médio, elevado}. Efectue todas as definições necessárias relacionadas com a adição desta variável.

5. Crie algumas novas regras que utilizem a nova variável nos antecedentes. Teste o sistema fuzzy resultante.

4.2 Resolução Através de Programação de Funções Matlab

As tarefas anteriores podem ser implementadas numa função Matlab usando funções da toolbox fuzzy.

A forma algorítmica para execução do programa seria:

- **Passo 1:** Criar a estrutura *fuzzy* usando a função **`newfis`**
  `a = newfis('nome do fis')` cria uma estrutura FIS do tipo Mamdani de nome `a`

- **Passo 2:** Criar as variáveis linguísticas de entrada e de saída usando a função **`addvar`**
  `a = addvar(a,varType,varName,varBounds)`
  - `a: nome da estrutura FIS`
  - `varType: tipo de variável (pode ser 'input' ou 'output')`
  - `varName: nome da variável (por exemplo, 'comida')`
  - `varBounds: limites do domínio da variável (por exemplo, [0 5])`

- **Passo 3:** Escolher as funções de pertença e determinar os domínios de entrada e saída usando a função **`addmf`**
  `a = addmf(a,varType,varIndex,mfName,mfType,mfParams)`
  - `a: nome da estrutura FIS`
  - `varType: tipo de variável (pode ser 'input' ou 'output')`
  - `varIndex: índice da variável de input ou de output (1, 2, ..., conforme a ordem de criação da variável)`
  - `mfName: nome da função de pertença para as variáveis linguísticas (por exemplo, 'boa', 'fraca', 'excelente')`
  - `mfType: tipo de função de pertença ('gaussmf', 'trapmf', 'trimf', etc)`
  - `mfParams: parâmetros para a função de pertença`

```
a=addmf(a,'input',1,'fraco','gaussmf',[1.5 0]);
```

- **Passo 4:** Criar as regras fuzzy para as funções de pertença usando uma matriz de entradas e saídas conforme definição do Matlab e usar a função **`addrule`** para inserir as regras no programa.
  - As regras escrevem-se numa matriz de regras
  - A matriz deve ter m + n + 2 colunas (m: nº de variáveis de entrada, n: nº e variáveis de saída)
  - A matriz deve ter tantas linhas quantas as regras pretendidas
  - Colunas 1 – m (indicar para cada variável de entrada qual a função de pertença a usar na regra)

- Colunas m+1 – n (indicar qual a função de pertença para a variável de saída)
- Coluna seguinte: peso da regra, valor entre zero e um
- Última coluna (operador a usar: 1 (AND); 2 (OR); -1 (NOT); 0 (Nenhum))

- Exemplo, uma linha da matriz de regras com: `1 2 3 1 2` pode ser traduzida em:

  `Se entrada1 = MF1 OR entrada2 = MF2 then saida = MF3`

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

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

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

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

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

4.2.1 Programar o exemplo da gorjeta

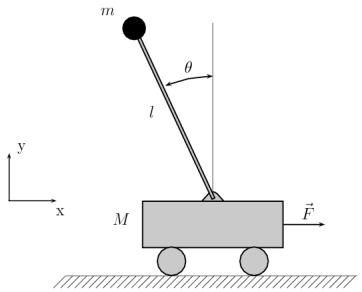
Crie uma nova função de nome gorjeta e grave-a no disco com o mesmo nome
O esqueleto da função é o seguinte (disponível no moodle):

```
function [ fis_gorjeta ] = gorjeta()

%PASSO 1: crie a estrutura FIS de nome fis_gorjeta
%COMPLETAR


%PASSO 2: criar variaveis linguisticas 'servico', 'comida' e 'gorjeta'
fis_gorjeta=addvar(fis_gorjeta,'input','servico',[0 10]);
%COMPLETAR

%PASSO 3: funções de pertença para cada variável criada anteriormente
fis_gorjeta=addmf(fis_gorjeta,'input',1,'fraco','gaussmf',[1.5 0]);
%COMPLETAR

fis_gorjeta=addmf(fis_gorjeta,'input',2,'ma','trapmf',[0 0 1 3]);
%COMPLETAR

fis_gorjeta=addmf(fis_gorjeta,'output',1,'fraca','trimf',[0 5 10]);
%COMPLETAR

%PASSO 4: criar matriz de regras e adicionar com addrule

regras=[];%COMPLETAR


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

end
```

Complete a função, usando as funções de pertença, domínios, etc, que foram usadas no tutorial da interface gráfica.
Execute a função e analise os resultados

5. Problema 2 - 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

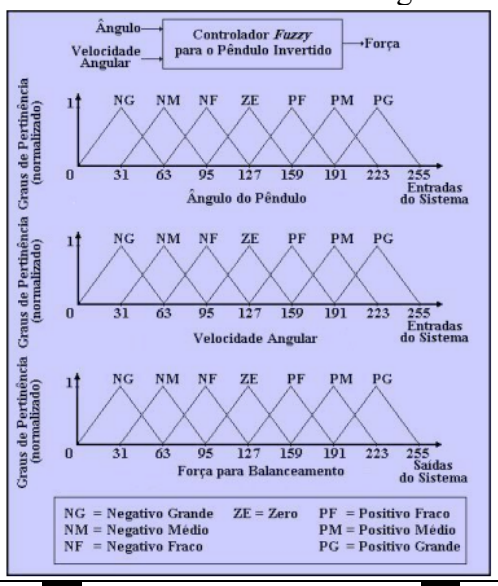
- duas entradas: o ângulo e a velocidade angular
- uma saída: a força.

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	
2	0	0	
3	157	223	