Aprendizagem Computacional - Trabalho Prático 4

João Tiago Márcia do Nascimento Fernandes - 2011162899
Joaquim Pedro Bento Gonçalves Pratas Leitão - 2011150072

9 de Dezembro de 2014

Índice

1 Introdução

O presente trabalho visa a implementação de controladores difusos de um dado sistema, dos tipos *Mamdani* e *Sugeno*, e que representam um hipotético processo real, tendo como base do seu funcionamento regras da *lógica difusa*.

A *lógica difusa* extende o conceito de *lógica booleana*, admitindo valores lógicos intermediários entre os valores de *FALSO* e *VERDADEIRO*. Assim, a lógica difusa permite-nos extender o conceito de conjunto, permitindo que um elemento passe a possuir um *grau de pertença* num dado conjunto, cujo valor varia entre *0* e *1*.

Um controlador que se rege por essa lógica, ou seja, um *controlador difuso*, possuí um conjunto de regras lógicas, e é constituído por três fases:

- *Fase entrada*
- *Fase de processamento*
- *Fase de saída*

A fase de entrada mapeia a entrada recebida pelo controlador nas funções de pertença e valores de verdade apropriados. Já na fase de processamento, as regras lógicas apropriadas à entrada recebida são invocadas, combinando o seu resultado, que é devolvido na fase de saída, sendo codificado num valor específico.

Relativamente aos controladores disponíveis na aplicação, estes foram desenvolvidos no ambiente *Simulink*, em conjunto com a *Fuzzy Logic Toolbox*, ambos pertencentes à plataforma *Matlab*. Numa fase inicial do trabalho apenas foram desenvolvidos controladores de 9 regras, tendo esse valor aumentado para 25 numa fase posterior do trabalho.

No presente documento pretendemos apresentar de forma mais detalhada os controladores implementados, discutindo alguns detalhes relativos à sua implementação e apresentando uma reflexão crítica sobre o desempenho e performance dos diferentes controladores implementados (*Mamdani* e *Sugeno*, de *9* e *25* regras).

2 Aplicação Desenvolvida

A aplicação desenvolvida visa implementar os controladores difusos referidos anteriormente, representando um hipotético processo real, cujo funcionamento pode ser simulado, através da aplicação.

Para isso, a aplicação foi implementada em *Matlab*, recorrendo à *Fuzzy Logic Toolbox* do *Simulink*, de forma a permitir a simulação do processo.

A aplicação faz ainda uso de uma interface por linha de comandos (*CLI - Command Line Interface*), que permite ao utilizador escolher as características do processo que pretende simular: tipo de controlador, número de regras e tipos de perturbações, referência, entre outros.

2.1 Processo

Para a aplicação em questão, foram atribuídos a cada grupo diferentes processos a simular, sendo que cada processo é caracterizado por uma função de transferência diferente.

O processo que nos foi atribuído é caracterizado pela seguinte função de transferência:

$$\frac{3 \times (s+1)}{s \times (s+2) \times (s+4)}$$

2.2 Referências

Nesta aplicação encontram-se disponíveis dois tipos diferentes de referências: Uma correspondente à *onda quadrada*, e outra a uma *onda senoide (seno)*, ambas de amplitude 1.

Como podemos observar nas figuras seguintes, a onda senoide apresenta-se como a mais suave das duas ondas, contendo variações menos pronunciadas, quando comparada com a onda quadrada.

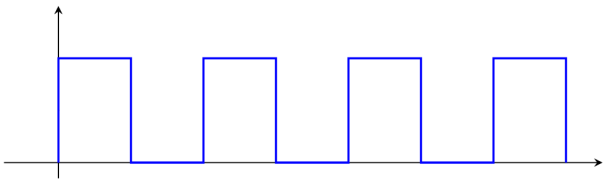

Figura 1: Exemplo de uma onda quadrada

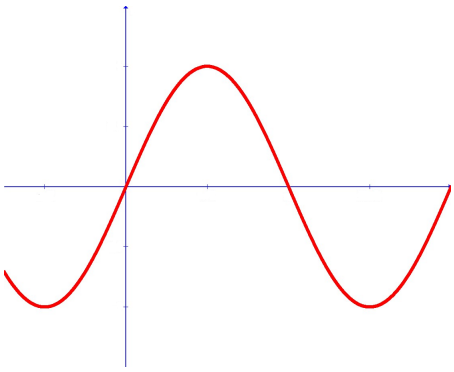

Figura 2: Exemplo de uma onda senoide

Esta última é caracterizada por períodos em que o seu valor se mantém constante, que alternam com pontuais momentos onde esta muda bruscamente o seu valor.

Estas constantes e abruptas variações têm um efeito negativo no controlador, resultando em erros mais pronunciados, e numa performance superior (o que na presente situação, em que a performance é medida em função o integral do erro quadrático, não é uma situação de todo desejada).

Assim, será de prever melhores desempenhos quando a referência fornecida ao controlador é a onda senoide, do que quando este tem como referência uma onda quadrada.

2.3 Controladores

Relativamente aos *controladores* implementados, encontram-se disponíveis dois tipos distintos: Controladores *Mamdani* e *Sugeno*, ambos com implementações de 9 e 25 regras.

Efetivamente, ao variarmos o número de regras em cada controlador estamos a alterar a ação por ele tomada, uma vez que o valor à saída do controlador está dependente da aplicação dessas regras lógicas à sua entrada.

Mais ainda, se aumentarmos o número de regras do controlador, mantendo todas as outras características constantes, seria de esperar uma melhoria na sua performance, uma vez que com um maior número de regras os ajustes realizados pelo controlador podem ser mais suaves e adaptados a cada situação em particular.

Assim, quanto menor for o número de regas, maior terá que ser a generalização de cada regra, culminando num maior erro do controlador, e numa performance inferior.

Nas figuras que se seguem apresentamos as tabelas com as regras implementadas nos controladores, onde:

- N - Negativo
- ZE - Zero
- P - Positivo
- NB - Negativo Grande
- NS - Negativo Pequeno

- PS - Positivo Pequeno
- PB - Positivo Grande

e_k \ Δe_k	N	ZE	P
N	N	N	Z
ZE	N	Z	P
P	Z	P	P

Figura 3: Tabela com as 9 regras implementadas por um controlador de 9 regras

e_k \ Δe_k	NB	NS	ZE	PS	PB
NB	NB	NB	NB	NS	ZE
NS	NB	NB	NS	ZE	PS
ZE	NB	NS	ZE	PS	PB
PS	NS	ZE	PS	PB	PB
PB	ZE	PS	PB	PB	PB

Figura 4: Tabela com as 25 regras implementadas por um controlador de 25 regras

Desta forma, implementámos controladores com as seguintes características:

- Tipo *Mamdani* ou *Sugeno*
- Função de pertença do tipo *trimf*
- Operadores *min-max*
- Funções de Pertença do tipo *gaussmf* e *trimf*
- Desfuzificação do tipo *Centróide* e *Medmax* (Mamdani) e *Sum* e *Med* (Sugeno)

2.4 Perturbações

Numa primeira fase do trabalho, pretendemos estudar o desempenho dos diferentes controladores ao seguirem uma dada entrada, ignorando qualquer perturbação das ações realizadas pelo controlador. Posteriormente, surge com naturalidade o estudo do comportamento e desempenho dos mesmos controladores quando limitados nas suas ações.

Assim, introduzimos perturbações nos sinais de referência gerados, que podem atuar ao nível dos *atuadores* e/ou da *carga* do controlador, com o objetivo de verificarmos como é que cada controlador compensa essas perturbações.

Uma vez que estamos a adulterar os sinais de entrada do controlador, facilmente prevemos que o seu desempenho diminuirá quando sujeito a entradas desta forma adulteradas. Mais ainda, quando estas ocorrem ao nível do atuador e da carga, em simultâneo, será de esperar que o controlador apresente desempenho inferior, do que quando as perturbações se dão apenas na carga ou no atuador. Isto deve-se ao facto de estas perturbações introduzirem variações na referência, que são, naturalmente, mais difíceis de seguir para o controlador.

Desta forma, no presente trabalho, incluímos os seguintes cenários relativos a existência de perturbações na referência do controlador:

- Ausência de perturbações na referência do controlador
- Perturbações ao nível do atuador
- Perturbações ao nível da carga
- Perturbações ao nível do atuador e da carga

Nas figuras seguintes apresentamos um exemplo da referência fornecida ao controlador, e a sua respetiva saída, com e sem perturbações:

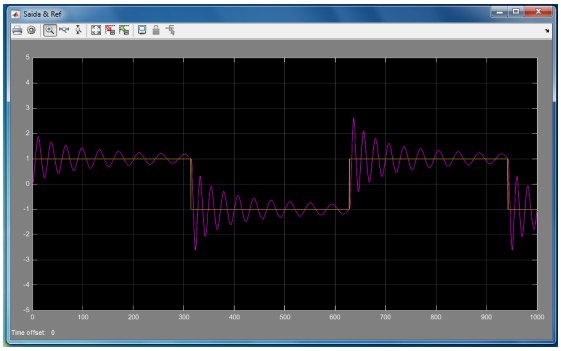

Figura 5: Exemplo da referência fornecida ao controlador (amarelo) e respetiva saída (roxo) sem perturbação da referência

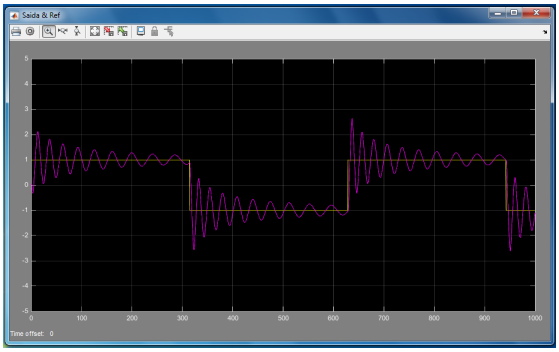

Figura 6: Exemplo da referência fornecida ao controlador (amarelo) e respetiva saída (roxo) com perturbação do atuador

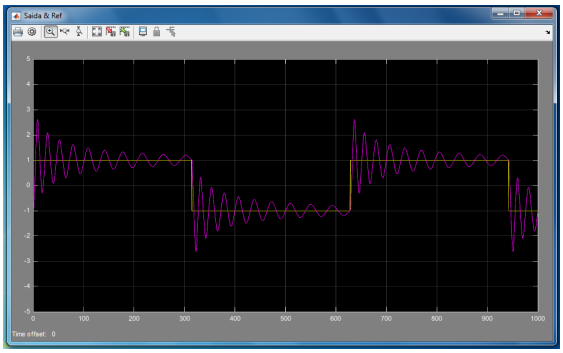

Figura 7: Exemplo da referência fornecida ao controlador (amarelo) e respetiva saída (roxo) com perturbação da carga

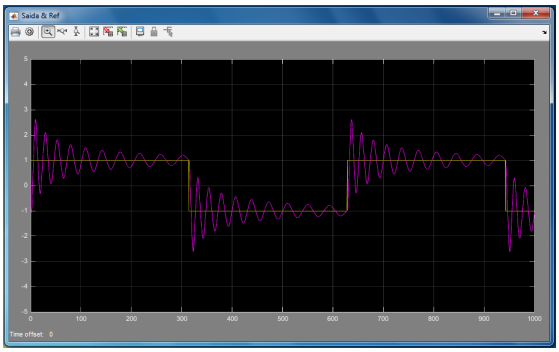

Figura 8: Exemplo da referência fornecida ao controlador (amarelo) e respetiva saída (roxo) com perturbação do atuador e da carga

2.5 Implementação

Tal como referimos no início do documento, a aplicação foi desenvolvida em *Matlab*, recorrendo ao seu ambiente de simulação, o *Simulink* e à sua biblioteca de lógica difusa: *Fuzzy Logic Toolbox*.

Assim, a nossa implementação consiste num pequeno ficheiro *Matlab* (a partir do qual a aplicação é iniciada, e onde são selecionados os seus parâmetros), e num conjunto dos controladores implementados e dos modelos de simulação do processo.

Estes últimos, permitem-nos implementar o sistema a simular, possibilitando-nos a sua execução de acordo com os parâmetros especificados. Na figura seguinte apresentamos um exemplo de um modelo implementado:

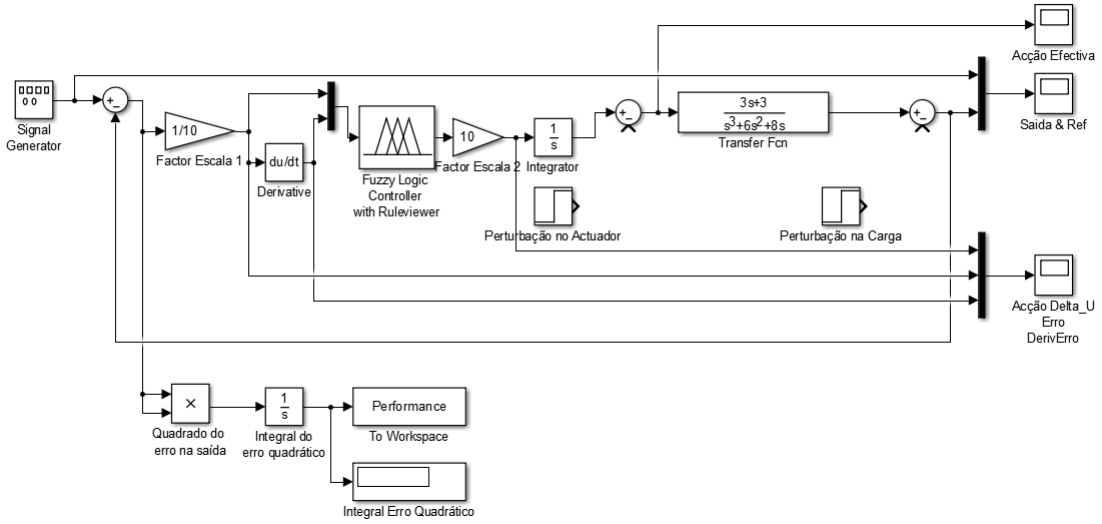

Figura 9: Exemplo de um modelo implementado em *Simulink*

O modelo em questão pretende simular a existência de um controlador do tipo *Mamdani* com 9 regras (embora fosse possível considerar outro controlador, sem necessitar de proceder a alterações significativas no modelo), possuindo uma referência quadrada (à semelhança do controlador, poderíamos considerar outra referência sem necessitarmos de proceder a alterações significativas no modelo) e com ausência de perturbações.

Assim, implementámos um modelo diferente para cada situação a simular, tendo em conta o controlador a adotar, referência e perturbações a considerar.

2.5.1 Modelos

No que respeita à implementação dos modelos existe um aspeto relevante que, desde já, gostaríamos de salientar. Este aspeto prende-se com os fatores de escala e com a sua determinação.

Uma vez que não possuímos nenhum método analítico para determinar os fatores de escala que nos permitem obter os melhores resultados, a determinação dos fatores de escala a considerar

foi realizada por *tentativa-erro*, até encontrarmos um par de valores que nos permitissem obter resultados aceitáveis (ou o mais próximo disso possível) nos diferentes cenários considerados.

Após alguns testes, optámos por utilizar um fator de escala de 0.5 à entrada do controlador, e um fator de escala de 2 à saída do controlador. No entanto, acreditamos ser possível a utilização de outros fatores de escala que nos permitam obter melhores resultados.

2.5.2 run.m

Este ficheiro contém um *script* que permite executar a aplicação. Neste *script* o utilizador poderá definir todas as propriedades do sistema a simular. O funcionamento da aplicação, tal como a estrutura deste ficheiro, serão abordados em maior detalhe na secção que se segue.

2.6 Execução

Para executar a aplicação o utilizador deverá executar o ficheiro *run.m*. Uma vez iniciada a execução, o utilizador deverá especificar as propriedades do sistema a simular, tais como:

- Referência do sistema (Onda senoide ou quadrada)
- Tipo de controlador a implementar (Mamdani ou Sugeno)
- Número de regras a implementar no controlador (9 ou 25 regras)
- Tipo de função de pertença a considerar
- Método de defuzificação a considerar
- Perturbações a considerar (Ausência de perturbações, perturbações no atuador e/ou na carga)

Uma vez definidas estas propriedades, será aberta uma janela contendo um modelo que representa o sistema escolhido (de acordo com as propriedades especificadas). O utilizador poderá executar este sistema clicando no botão de execução, presente na barra de opções no topo da janela.

Uma vez finda a execução do sistema, podemos consultar o seu desempenho consultando o campo *IntegralErroQuadrático*, que, tal como o nome sugere, corresponde ao integral do erro quadrático (quadrado da diferença entre a saída do controlador e a referência considerada). Para visualizar a saída, basta clicar sobre o ícone com o nome *Saída & Ref.*

3 Testes da Aplicação

Para podermos comparar o desempenho dos diferentes controladores implementados, nas situações consideradas, realizámos alguns testes à aplicação, que apresentamos no final da presente secção.

Para cada um dos controladores implementados testámos o seu desempenho para diferentes números de regras, referências e perturbações, mantendo todos os outros fatores iguais (nomeadamente os fatores de escala). Assim, testámos as seguintes configurações, para cada controlador:

- Considerar todas as combinações das propriedades mencionadas para cada controlador (número de regras, função de pertença, método de defuzificação)
- Utilizar como referência a onda quadrada e a senoide (seno)
- Considerar a ausência de perturbações
- Considerar a existência de perturbações a nível do atuador
- Considerar a existência de perturbações a nível da carga
- Considerar a existência de perturbações a nível do atuador e da carga
- Utilizar como fator de escala à entrada do controlador o valor 0.5, e o valor 2 à saída do controlador

Controller	Number Rules	Signal Generator	Membership Function	Defuzzification Method	Perturbation Actuator	Perturbation Charge	Performance
Mamdani	9	Square	gaussmf	centroid	No	No	252.1
Mamdani	9	Square	gaussmf	centroid	Yes	No	278.3
Mamdani	9	Square	gaussmf	centroid	No	Yes	299.3
Mamdani	9	Square	gaussmf	centroid	Yes	Yes	305
Mamdani	9	Square	gaussmf	medmax	No	No	449.8
Mamdani	9	Square	gaussmf	medmax	Yes	No	476.3
Mamdani	9	Square	gaussmf	medmax	No	Yes	499.9
Mamdani	9	Square	gaussmf	medmax	Yes	Yes	504.8
Mamdani	9	Square	trimf	centroid	No	No	188.1
Mamdani	9	Square	trimf	centroid	Yes	No	200.2
Mamdani	9	Square	trimf	centroid	No	Yes	217.2
Mamdani	9	Square	trimf	centroid	Yes	Yes	222.3
Mamdani	9	Square	trimf	medmax	No	No	619.4
Mamdani	9	Square	trimf	medmax	Yes	No	461.2
Mamdani	9	Square	trimf	medmax	No	Yes	472.5
Mamdani	9	Square	trimf	medmax	Yes	Yes	480.6
Mamdani	9	Sin	gaussmf	centroid	No	No	3.698
Mamdani	9	Sin	gaussmf	centroid	Yes	No	71.01
Mamdani	9	Sin	gaussmf	centroid	No	Yes	57.47
Mamdani	9	Sin	gaussmf	centroid	Yes	Yes	79.3
Mamdani	9	Sin	gaussmf	medmax	No	No	414.3
Mamdani	9	Sin	gaussmf	medmax	Yes	No	420.8
Mamdani	9	Sin	gaussmf	medmax	No	Yes	349.3
Mamdani	9	Sin	gaussmf	medmax	Yes	Yes	423
Mamdani	9	Sin	trimf	centroid	No	No	0.9305
Mamdani	9	Sin	trimf	centroid	Yes	No	40.26
Mamdani	9	Sin	trimf	centroid	No	Yes	29.62
Mamdani	9	Sin	trimf	centroid	Yes	Yes	48.64
Mamdani	9	Sin	trimf	medmax	No	No	413.1
Mamdani	9	Sin	trimf	medmax	Yes	No	414.5
Mamdani	9	Sin	trimf	medmax	No	Yes	347.9
Mamdani	9	Sin	trimf	medmax	Yes	Yes	423.1
Mamdani	25	Square	gaussmf	centroid	No	No	29
Mamdani	25	Square	gaussmf	centroid	Yes	No	29.79
Mamdani	25	Square	gaussmf	centroid	No	Yes	33.23
Mamdani	25	Square	gaussmf	centroid	Yes	Yes	35.9
Mamdani	25	Square	gaussmf	medmax	No	No	143.3
Mamdani	25	Square	gaussmf	medmax	Yes	No	149.6
Mamdani	25	Square	gaussmf	medmax	No	Yes	145.7
Mamdani	25	Square	gaussmf	medmax	Yes	Yes	147.8
Mamdani	25	Square	trimf	centroid	No	No	30.99
Mamdani	25	Square	trimf	centroid	Yes	No	32.07
Mamdani	25	Square	trimf	centroid	No	Yes	36.94
Mamdani	25	Square	trimf	centroid	Yes	Yes	38.29
Mamdani	25	Square	trimf	medmax	No	No	134.7
Mamdani	25	Square	trimf	medmax	Yes	No	140.1
Mamdani	25	Square	trimf	medmax	No	Yes	150.6
Mamdani	25	Square	trimf	medmax	Yes	Yes	147.6
Mamdani	25	Sin	gaussmf	centroid	No	No	0.001557
Mamdani	25	Sin	gaussmf	centroid	Yes	No	0.993
Mamdani	25	Sin	gaussmf	centroid	No	Yes	2.125
Mamdani	25	Sin	gaussmf	centroid	Yes	Yes	4.448
Mamdani	25	Sin	gaussmf	medmax	No	No	110.5

Mamdani	25	Sin	gaussmf	medmax	Yes	No	114.8
Mamdani	25	Sin	gaussmf	medmax	No	Yes	115.8
Mamdani	25	Sin	gaussmf	medmax	Yes	Yes	120.1
Mamdani	25	Sin	trimf	centroid	No	No	0.001043
Mamdani	25	Sin	trimf	centroid	Yes	No	0.6235
Mamdani	25	Sin	trimf	centroid	No	Yes	1.651
Mamdani	25	Sin	trimf	centroid	Yes	Yes	3.675
Mamdani	25	Sin	trimf	medmax	No	No	109.4
Mamdani	25	Sin	trimf	medmax	Yes	No	118.1
Mamdani	25	Sin	trimf	medmax	No	Yes	116.6
Mamdani	25	Sin	trimf	medmax	Yes	Yes	117.2
Sugeno	9	Square	gaussmf	med	No	No	24.88
Sugeno	9	Square	gaussmf	med	Yes	No	25.59
Sugeno	9	Square	gaussmf	med	No	Yes	28.89
Sugeno	9	Square	gaussmf	med	Yes	Yes	30.81
Sugeno	9	Square	gaussmf	sum	No	No	21.78
Sugeno	9	Square	gaussmf	sum	Yes	No	22.27
Sugeno	9	Square	gaussmf	sum	No	Yes	25.33
Sugeno	9	Square	gaussmf	sum	Yes	Yes	26.64
Sugeno	9	Square	trimf	med	No	No	24.71
Sugeno	9	Square	trimf	med	Yes	No	25.17
Sugeno	9	Square	trimf	med	No	Yes	29.43
Sugeno	9	Square	trimf	med	Yes	Yes	30.22
Sugeno	9	Square	trimf	sum	No	No	23.7
Sugeno	9	Square	trimf	sum	Yes	No	24.32
Sugeno	9	Square	trimf	sum	No	Yes	27.12
Sugeno	9	Square	trimf	sum	Yes	Yes	29.56
Sugeno	9	Sin	gaussmf	med	No	No	0.0006321
Sugeno	9	Sin	gaussmf	med	Yes	No	0.7773
Sugeno	9	Sin	gaussmf	med	No	Yes	1.896
Sugeno	9	Sin	gaussmf	med	Yes	Yes	3.597
Sugeno	9	Sin	gaussmf	sum	No	No	0.000277
Sugeno	9	Sin	gaussmf	sum	Yes	No	0.4134
Sugeno	9	Sin	gaussmf	sum	No	Yes	1.517
Sugeno	9	Sin	gaussmf	sum	Yes	Yes	2.708
Sugeno	9	Sin	trimf	med	No	No	0.002026
Sugeno	9	Sin	trimf	med	Yes	No	0.3688
Sugeno	9	Sin	trimf	med	No	Yes	1.568
Sugeno	9	Sin	trimf	med	Yes	Yes	3.101
Sugeno	9	Sin	trimf	sum	No	No	0.0002018
Sugeno	9	Sin	trimf	sum	Yes	No	0.2895
Sugeno	9	Sin	trimf	sum	No	Yes	1.494
Sugeno	9	Sin	trimf	sum	Yes	Yes	2.656
Sugeno	25	Square	gaussmf	med	No	No	21.77
Sugeno	25	Square	gaussmf	med	Yes	No	22.36
Sugeno	25	Square	gaussmf	med	No	Yes	24.94
Sugeno	25	Square	gaussmf	med	Yes	Yes	26.59
Sugeno	25	Square	gaussmf	sum	No	No	18.95
Sugeno	25	Square	gaussmf	sum	Yes	No	19.95
Sugeno	25	Square	gaussmf	sum	No	Yes	21.51
Sugeno	25	Square	gaussmf	sum	Yes	Yes	22.3
Sugeno	25	Square	trimf	med	No	No	24.57
Sugeno	25	Square	trimf	med	Yes	No	24.84
Sugeno	25	Square	trimf	med	No	Yes	27.88

Sugeno	25	Square	trimf	med	Yes	Yes	29.57
Sugeno	25	Square	trimf	sum	No	No	22.99
Sugeno	25	Square	trimf	sum	Yes	No	23.72
Sugeno	25	Square	trimf	sum	No	Yes	23.69
Sugeno	25	Square	trimf	sum	Yes	Yes	28.8
Sugeno	25	Sin	gaussmf	med	No	No	0.0003508
Sugeno	25	Sin	gaussmf	med	Yes	No	0.4425
Sugeno	25	Sin	gaussmf	med	No	Yes	1.618
Sugeno	25	Sin	gaussmf	med	Yes	Yes	3.154
Sugeno	25	Sin	gaussmf	sum	No	No	0.0001541
Sugeno	25	Sin	gaussmf	sum	Yes	No	0.2451
Sugeno	25	Sin	gaussmf	sum	No	Yes	1.302
Sugeno	25	Sin	gaussmf	sum	Yes	Yes	2.344
Sugeno	25	Sin	trimf	med	No	No	0.0001085
Sugeno	25	Sin	trimf	med	Yes	No	0.3371
Sugeno	25	Sin	trimf	med	No	Yes	1.539
Sugeno	25	Sin	trimf	med	Yes	Yes	2.898
Sugeno	25	Sin	trimf	sum	No	No	0.0001085
Sugeno	25	Sin	trimf	sum	Yes	No	0.2134
Sugeno	25	Sin	trimf	sum	No	Yes	1.459
Sugeno	25	Sin	trimf	sum	Yes	Yes	2.566

4 Conclusões

Após uma análise crítica dos resultados obtidos existem alguns pontos que consideramos importantes salientar.

Em primeiro lugar, tal como previmos no início deste documento, verificamos que quando utilizamos a função senoide (seno) como referência para o controlador, obtemos melhores valores de desempenho.

Tal como podemos constatar nas duas figuras que se seguem, os desajustes entre a saída obtida no controlador e a referência em questão são mais notórios em momentos de ocorrência de variações bruscas na referência (como é o caso da onda quadrada).

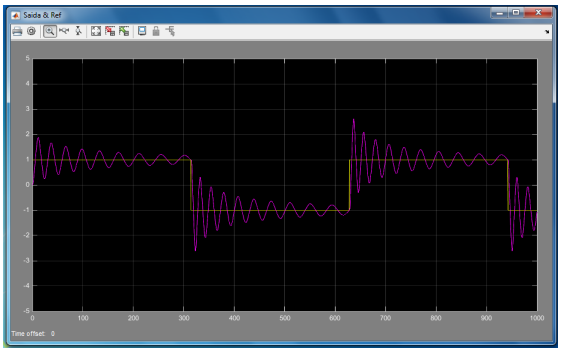

Figura 10: Saída obtida (a roxo) e esperada (a amarelo), utilizando a onda quadrada como referência e um controlador *Mamdani* de 9 regras

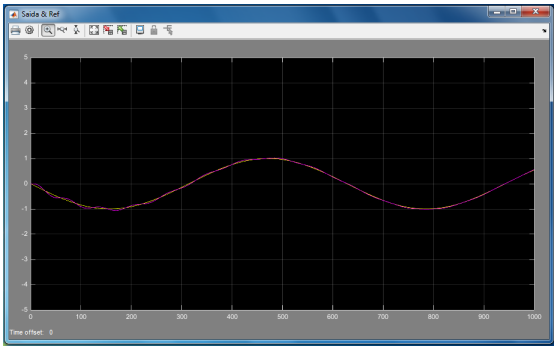

Figura 11: Saída obtida (a roxo) e esperada (a amarelo), utilizando a onda quadrada como referência e um controlador *Mamdani* de 9 regras

Quando a referência mantém uma tendência mais ou menos constante, apresentando variações mais suaves, é mais fácil para o controlador corrigir eventuais erros.

De qualquer das formas, mesmo para referências quadradas, verificámos uma correta aplicação das regras dos controladores. Considerando, por exemplo, a situação seguinte:

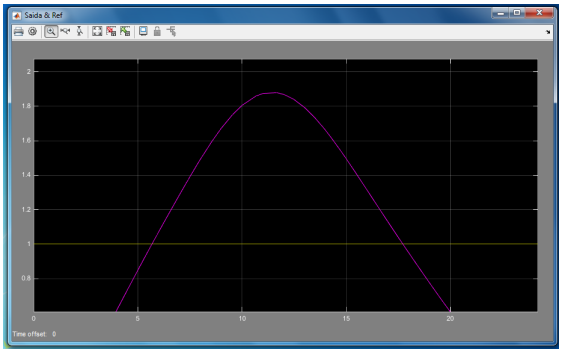

Figura 12: Exemplo de aplicação das regras do controlador *Mamdani* de 9 regras

Nesta situação a saída do controlador começa por ser inferior ao valor da sua referência, constituindo, portanto, um erro positivo. Uma vez que o valor da saída se está a aproximar da referência, concluímos que a sua variação é negativa (pois o erro está a diminuir, fruto da aproximação da saída à referência).

Assim, pela aplicação das regras apresentadas, o controlador não necessita de *"travar"* nem *"acelerar"* a saída, mantendo esta com a mesma tendência, caminhando no sentido de diminuição do erro, ou seja, da referência que pretendemos seguir.

De facto, tal situação é verificada, uma vez que o valor da saída do controlador aumenta, chegando até a igualar a referência. Nesse momento, estamos perante uma situação de erro nulo, e a sua variação é negativa (analisando os instantes imediatamente anteriores a esta situação verificamos que o erro se encontra a diminuir, o que corresponde a uma variação negativa). Assim, por aplicação das regras estudadas, o controlador deveria, nesta situação, *"travar"* a saída.

Efetivamente, tal ação é realizada, mas os seus efeitos não são sentidos no imediato, uma vez que a saída do controlador continua a aumentar, afastando-se do valor da referência, até que começa a diminuir, voltando a ir ao encontro da referência que se pretende seguir.

Para além destes factos, constatámos ainda, à semelhança do que previmos anteriormente neste documento, que ao aumentarmos o número de regras do controlador, este iria ter um melhor desempenho.

De facto, verificámos uma melhoria significativa desse desempenho, quer utilizando como referência a onda quadrada, quer utilizando a senoide, sendo que com esta última as melhorias são mais significativas.

Tal como referimos anteriormente, este resultado é esperado, uma que com um maior número de regras os ajustes realizados pelo controlador podem ser mais suaves e adaptados a cada situação em particular, resultando num melhor desempenho do mesmo.

Da mesma forma, ao introduzirmos perturbações nas referências fornecidas aos controladores podemos verificar uma baixa no seu desempenho, sendo por vezes bastante significativa.

De facto, este resultado também era por nós previsto, uma vez que ao considerarmos uma perturbação na referência estamos a adulterar os sinais de entrada do controlador, o que se vai repercutir na ação por ele tomada, nem sempre tomando a melhor ação para a situação em questão. Mais ainda, quando considerámos perturbações em simultâneo no atuador e na carga o desempenho dos controladores baixou ainda mais, tal como previsto.

Gostaríamos também de referir que, de uma forma geral, os controladores do tipo *Sugeno* apre-

sentaram melhores resultados do que os *Mamdani*, no entanto, estes últimos apresentam maior tolerância à presença de perturbações no atuador e na carga, uma vez que nessas situações obtiveram reduções relativas do desempenho inferiores às presenciadas pelos controladores do tipo *Sugeno*.

Mais ainda, nos controladores *Mamdani* que fazem uso do método de desfuzificação *centróide* obtivemos melhores resultados, do que nos que utilizaram o método *medmax*.

Já para os controladores *Sugeno*, o método de desfuzificação *sum* aparenta produzir melhores resultados do que o método *med*, embora a sua diferença seja bastante escassa.

Esta situação também se verifica no que diz respeito à função de pertença: os resultados obtidos sugerem que os controladores que fazem uso da função *trimf* possuem uma performance ligeiramente superior aos controladores que fazem uso da função *gaussmf*.

Assim, com base nos resultados obtidos e aqui apresentados, somos levados a concluir que, de uma forma geral, ao utilizarmos como referência a onda senoide, aliada a um controlador do tipo *Sugeno* com 25 regras, fazendo uso de um método de desfuzificação *sum* e uma função de pertença *trimf*, obteríamos a melhor performance.

Efetivamente, esta é a configuração que obteve melhores resultados para os diferentes cenários testados (ausência e presença de perturbações).