Aprendizagem Computacional

Relatório do Trabalho 4

"Fuzzy and Neuro-Fuzzy Systems"

André Gouveia – 2014216306
Joel Pires - 2014195242

Universidade de Coimbra

Departamento de Engenharia Informática

Mestrado em Engenharia Informática

Índice

Introdução

Este trabalho visa aplicar os conhecimentos adquiridos previamente em aulas teóricas sobre sistemas difusos, nomeadamente:

Sistemas Neuro-Difusos - de acordo com determinada entrada produz um certo valor de saída. Útil especialmente para modelar processos e sistemas,

Controladores Difusos - Representam um processo que faz parte do sistema. Entre outros, podem ser do tipo Mamdani e Sugeno. Estes dois últimos serão implementados e analisados durante este trabalho, cada um com 9 e 25 regras lógicas.

A implementação dos controladores difusos em questão far-se-á em Matlab e recorrendo às ferramentas de Simulink e Fuzzy Logic Toolbox.

Estes controladores, que se regem por uma lógica difusa, são constituídos por 3 fases: entrada, processamento, saída. Há então inicialmente um mapeamento das funções de pertença e valores verdade; há a chamada das regras lógicas adequadas em função da entrada; por fim dá-se a saída do resultado desse processamento.

Parte A

Implementação

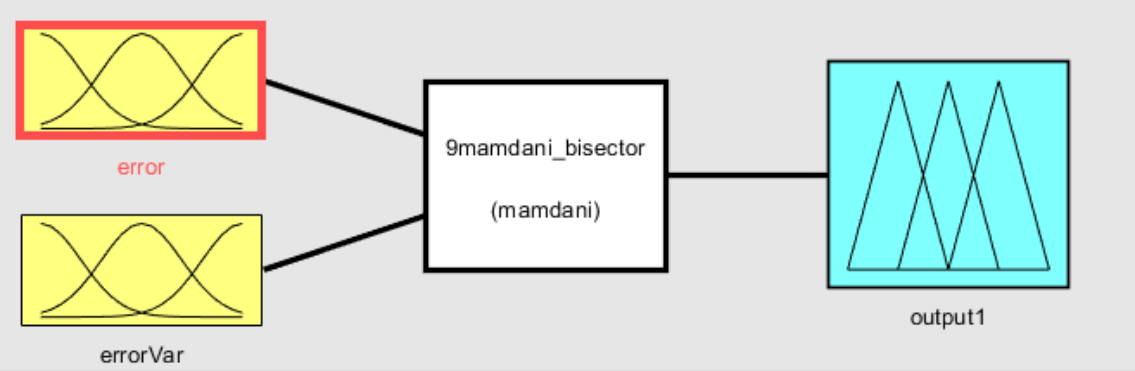
Controladores:

Foram implementados ao todo **14 controladores** diferentes, recorrendo ao *fuzzyLogicDesigner*. Foram todos criados previamente e colocados na pasta *controllers*. Esses controladores tem como possíveis combinações o seguinte:

- **Tipo de Controlador:** Mamdani ou Sugeno
- **Quantidade de Regras Lógicas:** 9 ou 25
- **Função de Defuzificação:** *Mom* ou *Centroid* no caso de ser *Mamdani*; *wtaver* ou *wtsum* no caso de ser *Sugeno*.

Todos estes controladores têm uma coisa em comum – o número de variáveis. Foram usadas **3 variáveis** em cada controlador:

- **2 inputs:** o erro e a sua variação
- **1 output**



Função de Transferência:

A função de transferência que calhou ao nosso grupo analisar é a seguinte:

$$\frac{2}{s^3+2s^2+2.5s+1.5}$$

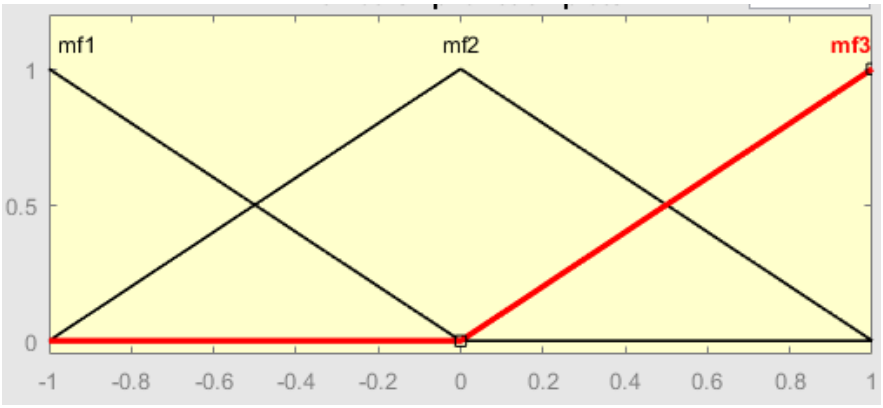
Funções de Pertença:

Como já foi mencionado para cada tipo de controlador (sugeno ou mamdani) foram desenvolvidas 9 e 25 regras. As nossas funções de pertença chamam-se:

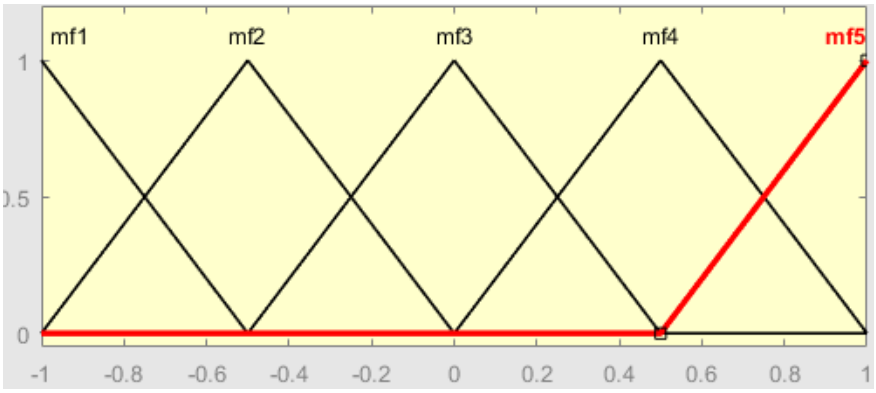
- **9 regras:** mf1, mf2, mf3
- **25 regras:** mf1, mf2, mf3, mf4, e mf5

Usou-se sempre funções do tipo ***trimf***. Os seus parâmetros são, respetivamente:

9 regras	
Mf1	[-2 -1 0]
Mf2	[-1 0 1]
Mf3	[0 1 2]



25 regras	
Mf1	[-1.5 -1 -0.5]
Mf2	[-1 -0.5 0]
Mf3	[-0.5 0 0.5]
Mf4	[0 0.5 1]
Mf4	[0.5 1 1.5]



Regras:

Quando utilizamos 9 regras, implementámo-las recorrendo ao operador "And" da seguinte forma:

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

Já no caso das 25 regras, guiámo-nos pela seguinte tabela:

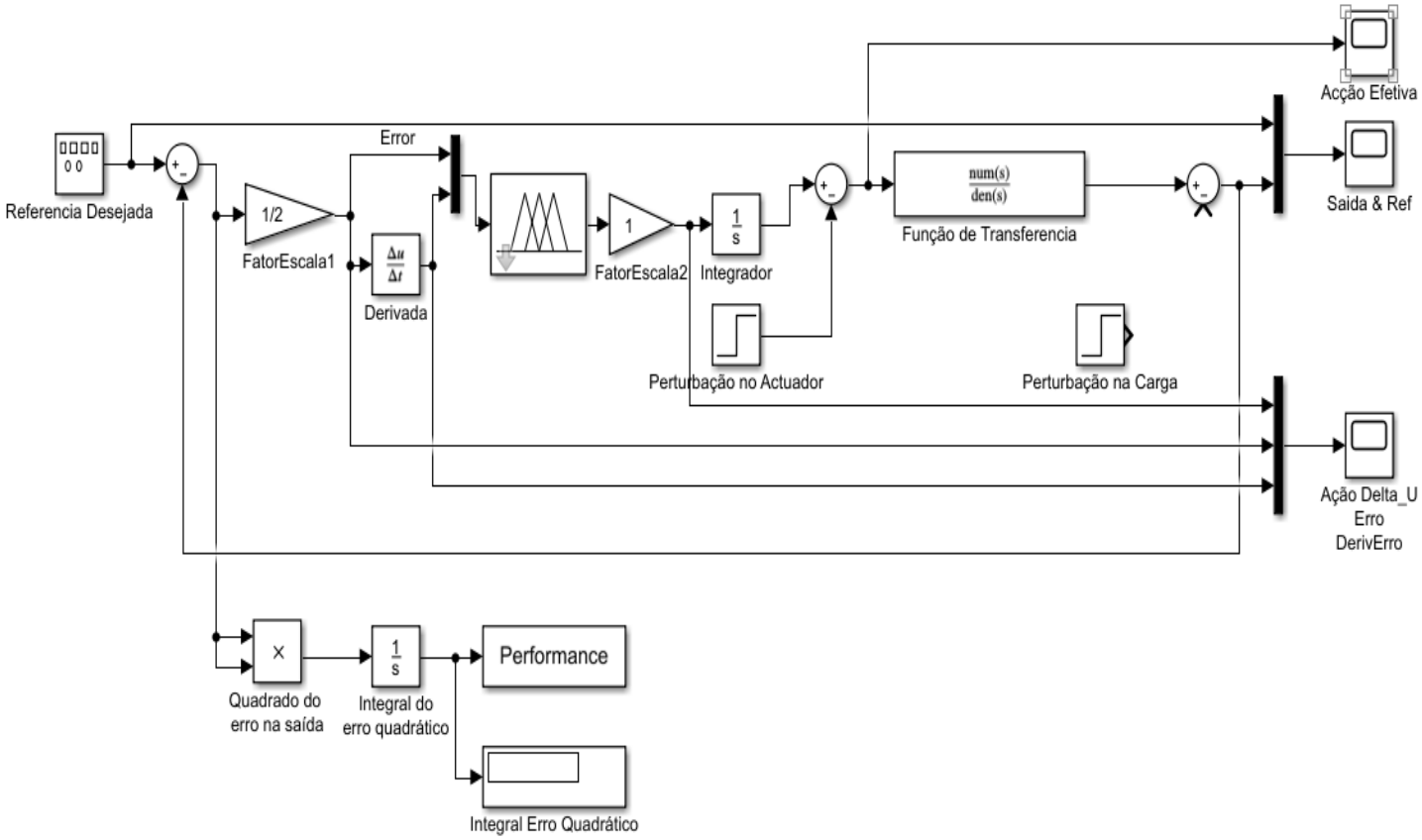
e_k \ Δe_k	NB	NS	ZE	PS	PB
NB	NB	NB	NB	NS	ZE
NS	NB	NB	NS	ZE	PS
ZE	NB	NS	ZE	PS	PB
PS	NS	ZE	PS	PB	PB
PB	ZE	PS	PB	PB	PB

Apresentámos abaixo o mapeamento que fizemos desses acrónimos para as nossas funções de pertença:

- N – Negativo -> mf1
- ZE – Zero -> mf2 no caso das 9 regras e mf3 no caso de serem 25
- P – Positivo -> mf3
- NB - Negativo Grande -> mf1
- NS - Negativo Pequeno -> mf2
- PS - Positivo Pequeno -> mf4
- PB - Positivo Grande -> mf5

Modelos:

Foram implementados ao todo **64 modelos** diferentes. Cada um deles foi desenhado e simulado recorrendo à ferramenta Simulink. A sua estrutura é algo comum em todos modelos, a saber a que se encontra na imagem abaixo:



Foram todos criados previamente e colocados na pasta *models*. Esses modelos tem como possíveis combinações o seguinte:

- **Tipo de Controlador:** dentre os 14 criados
- **Tipo de Perturbação:** Sem perturbação; com perturbação da carga, com perturbação do atuador; com perturbação do atuador e da carga
- **Tipo de Função de Referência:** a função de referência pode ser *Square* ou *Sine*

Vários **fatores de escala** foram testados e aqueles em que verificámos melhores resultados foram: 0.5 para o primeiro e 1 para o segundo.

Para os modelos poderem ser testados tem que se correr previamente o ***controllersScript.m*** para que os controladores possam estar no *workplace* do matlab e serem acessíveis pelos modelos a simular.

Testes e Observações

Foram simulados todos os 64 modelos acima mencionados através do Simulink e com T = 500.0. Obtiveram-se os seguintes resultados:

Função de Ref.	Nº de Regras	Tipo	Defuzificação	Perturbação	Performance
sine	25	mamdani	centroid	atuador	3,206
sine	25	mamdani	centroid	both	6,979
sine	25	mamdani	centroid	carga	2,188
sine	25	mamdani	centroid	nothing	0,1083
sine	25	mamdani	mom	atuador	126
sine	25	mamdani	mom	both	48,42
sine	25	mamdani	mom	carga	27,93
sine	25	mamdani	mom	nothing	34,25
sine	25	sugeno	wtaver	atuador	4,605
sine	25	sugeno	wtaver	both	7,67
sine	25	sugeno	wtaver	carga	2,282
sine	25	sugeno	wtaver	nothing	0,05322
sine	25	sugeno	wtsum	atuador	4,605
sine	25	sugeno	wtsum	both	7,67
sine	25	sugeno	wtsum	carga	2,282
sine	25	sugeno	wtsum	nothing	0,05322
sine	9	mamdani	centroid	atuador	19,2
sine	9	mamdani	centroid	both	24,7
sine	9	mamdani	centroid	carga	16,95
sine	9	mamdani	centroid	nothing	14,11
sine	9	mamdani	mom	atuador	186
sine	9	mamdani	mom	both	89,61
sine	9	mamdani	mom	carga	104,1
sine	9	mamdani	mom	nothing	263,3
sine	9	sugeno	wtaver	atuador	3,555
sine	9	sugeno	wtaver	both	7,356
sine	9	sugeno	wtaver	carga	2,32
sine	9	sugeno	wtaver	nothing	0,05322
sine	9	sugeno	wtsum	atuador	3,555
sine	9	sugeno	wtsum	both	7,356
sine	9	sugeno	wtsum	nothing	0,05323
sine	9	sugeno	wtsum	carga	2,32
square	25	mamdani	centroid	atuador	16,24
square	25	mamdani	centroid	both	19,97
square	25	mamdani	centroid	carga	15,22
square	25	mamdani	centroid	nothing	13,13
square	25	mamdani	mom	atuador	108,4
square	25	mamdani	mom	both	66,09

square	25	mamdani	mom	carga	40,88
square	25	mamdani	mom	nothing	60,51
square	25	sugeno	wtaver	atuador	21,44
square	25	sugeno	wtaver	both	24,58
square	25	sugeno	wtaver	carga	19,17
square	25	sugeno	wtaver	nothing	16,87
square	25	sugeno	wtsum	atuador	21,84
square	25	sugeno	wtsum	both	24,99
square	25	sugeno	wtsum	carga	19,57
square	25	sugeno	wtsum	nothing	17,27
square	9	mamdani	centroid	atuador	25,34
square	9	mamdani	centroid	both	30,56
square	9	mamdani	centroid	carga	23,24
square	9	mamdani	centroid	nothing	17,5
square	9	mamdani	mom	both	133,6
square	9	sugeno	wtsum	nothing	15,62
square	9	sugeno	wtsum	carga	17,94
square	9	sugeno	wtsum	both	23,03
square	9	sugeno	wtsum	atuador	17,81
square	9	sugeno	wtaver	nothing	15,26
square	9	mamdani	mom	carga	228,7
square	9	sugeno	wtaver	carga	17,58
square	9	sugeno	wtaver	both	22,66
square	9	mamdani	mom	nothing	141,3
square	9	sugeno	wtaver	atuador	15,94
square	9	mamdani	wtaver	atuador	67,1

De referir desde já que a performance medida aqui diz respeito à integração dos quadrados dos erros a cada instante. Assim, quanto maior for o valor de performance, menos eficiente será o nosso sistema. Pela mera observação da tabela poder-se-á concluir que:

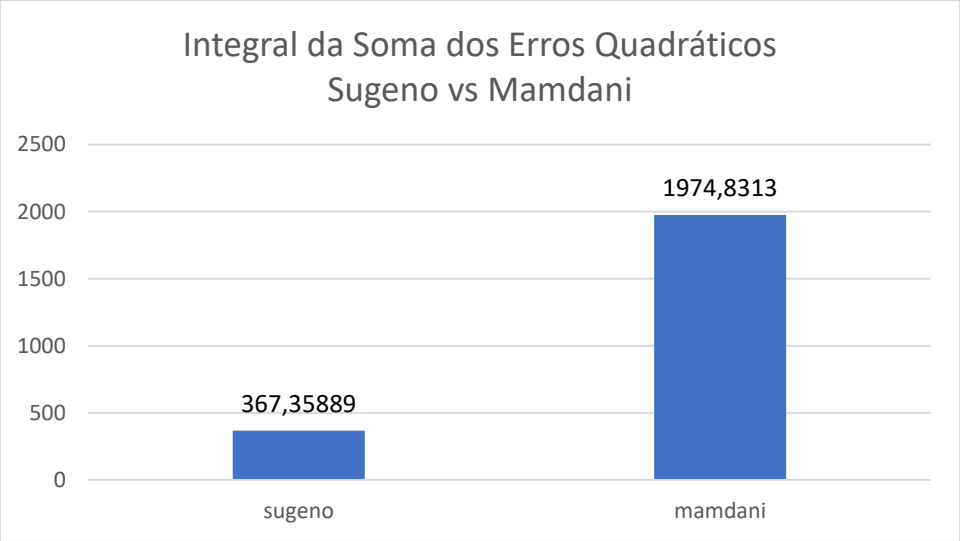
- O melhor desempenho ocorreu quando se usou a função de referência sinusoidal, do tipo sugeno e sem perturbações (independentemente do número de regras e da função de defuzificação)

- O pior desempenho ocorreu manifestamente quando se usou a função de referência sinusoidal, do tipo mamdani, sem perturbações, com 9 regras e com a função de defuzificação *mom*.

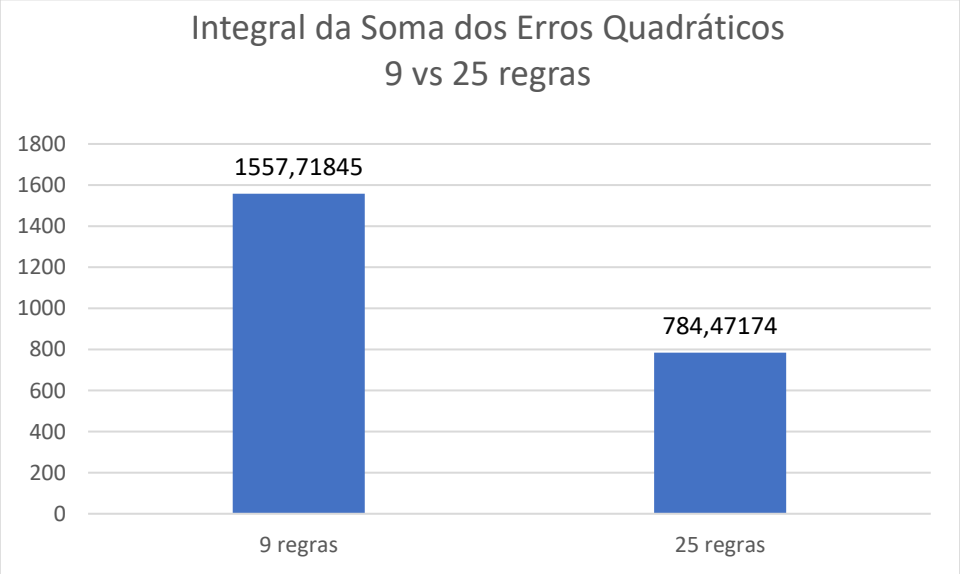
Os resultados gráficos da saída vs referência desejada encontram-se no Anexo A.

Prossigamos agora a analisar com mais detalhe a e confrontar as performances de:

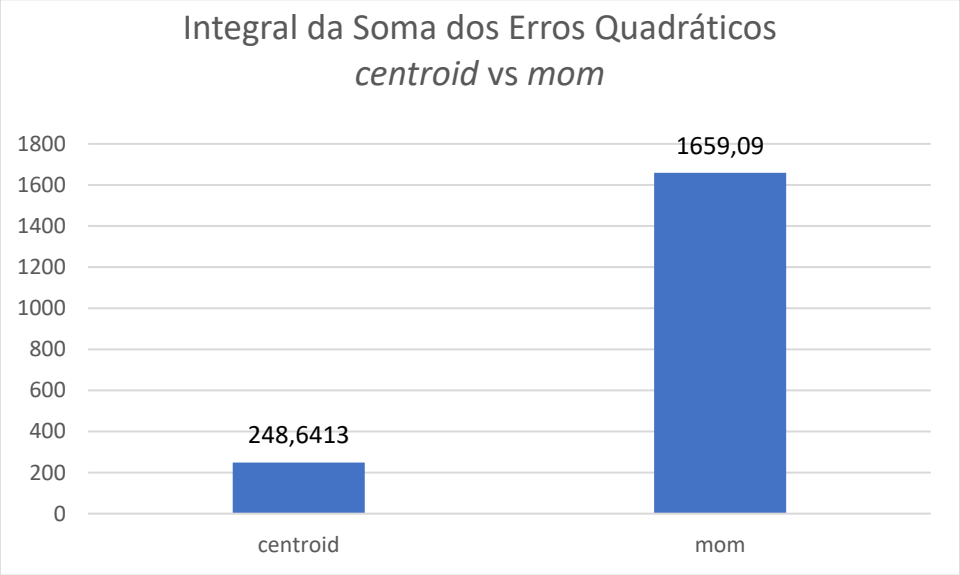
- Sugeno vs Mamdani
- 9 regras vs 25 regras
- *cetroid* vs *mom*
- *wtaver* vs *wtsum*
- Sem perturbações vs perturbação na carga vs perturbação no atuador vs ambas as perturbações
- *Sine* vs *Square*



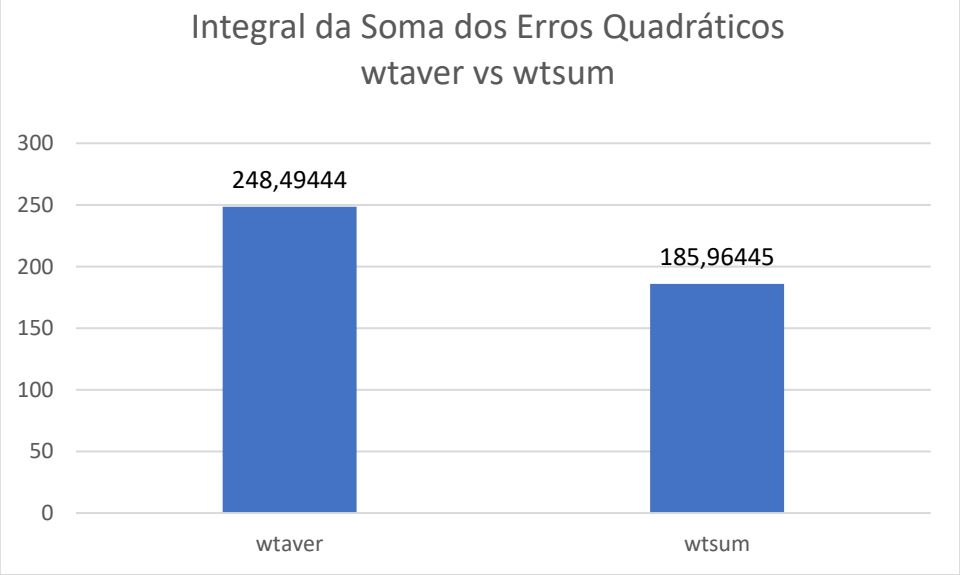
Através do gráfico acima podemos concluir de forma bastante intuitiva que o foi em regra geral obtido um melhor desempenho quando se usou um controlador do tipo Sugeno. Notemos que essa diferença de performance ainda é bastante significativa, sendo quase cerca de 5x maior face ao Mamdani.



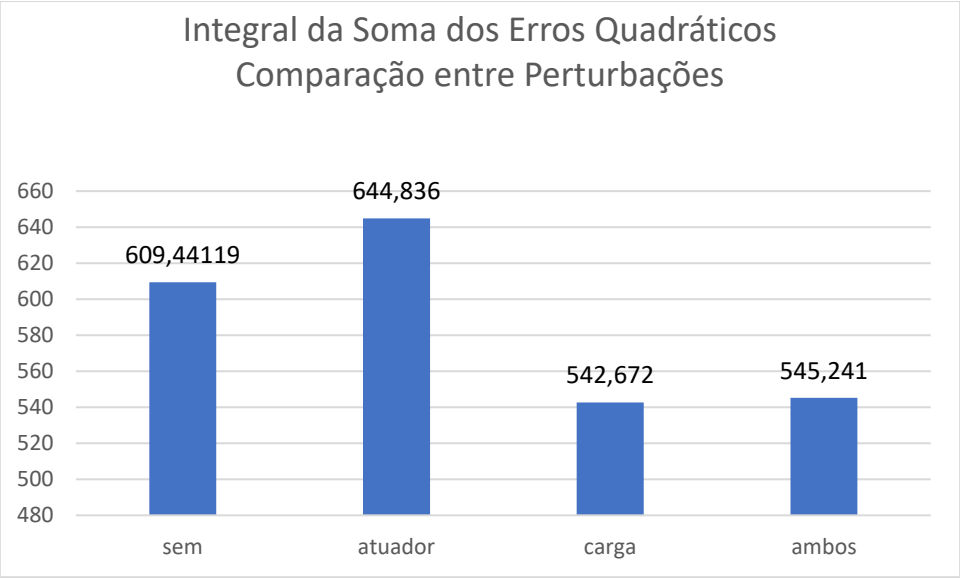
Deste gráfico deduzimos que o **uso de mais regras beneficia a performance** do sistema. Pelos valores acima concluímos que ao aumentar as regras de 3^2 para 5^2, obtivemos praticamente o **dobro da performance**.



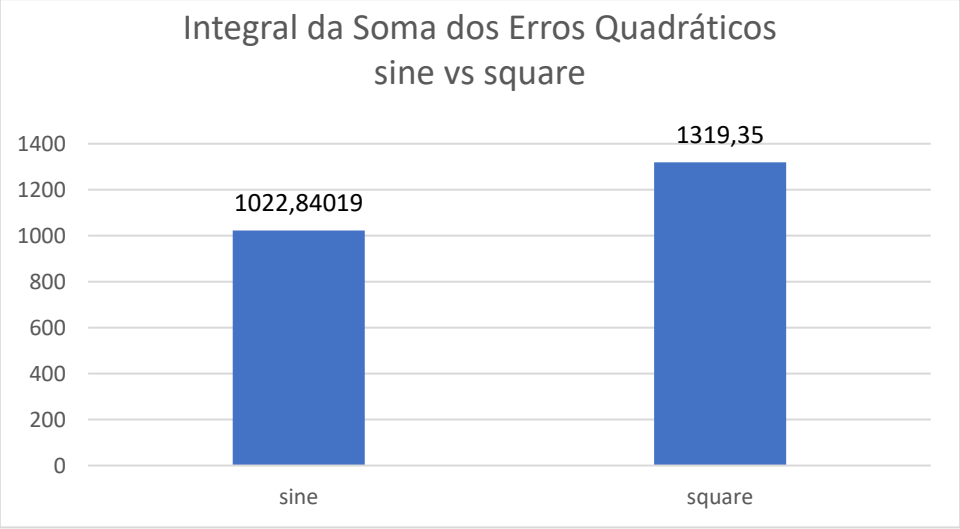
Aqui observamos que sem dúvida a escolha da função de defuzificação influencia muito a nossa performance. No caso de usarmos um controlador do tipo Mamdani, então a **função *centroid* é aquela que nos vair permitir ter melhor performance, quase 7x superior ao *mom*.**



Mais uma vez se conclui que a função de defuzificação influencia a performance, embora neste caso a diferença não seja tão notória. Contudo, o uso de ***wtsum* é a que nos permite alcançar melhor resultados para um controlador do tipo *Sugeno***.



Aqui os resultados podem ir um pouco contra o espectável uma vez que o nosso sistema apresenta uma melhor performance quando há perturbação na carga, depois quando ocorrem as duas perturbações depois quando não há perturbações e por fim quando há perturbações apenas no atuador. Teoricamente, o sistema deveria apresentar melhor performance quando não há perturbações. Uma análise mais detalhada à tabela da página 9 verificamos que **regra geral quando não há perturbações o resultado é melhor**. No entanto, quando se usa funções de defuzificação como *mom* o sistema tem um desempenho muito fraco quando não tem perturbações.



Através deste gráfico notamos que o sistema **consegue-se aproximar muito mais de uma referência desejada que assuma um comportamento sinusoida**l. Isso é natural dado que as variações abruptas da função *Square* resulta em erros mais significativos.

Parte B

Contexto:

Queremos descrever através de equações diferenciais um sistema dinâmico com memória. Admitamos que o sistema tem inércia e que y representa a saída num instante k, então temos:

$$y(k) = f(y(k-1), y(k-2), \ldots, u(k-1), u(k-2), \ldots)$$

Através de técnicas de clustering conseguimos deduzir as regras lógicas de forma a minimizar o erro entre a saída desejada e a saída do sistema difuso.

Modelo Inicial:

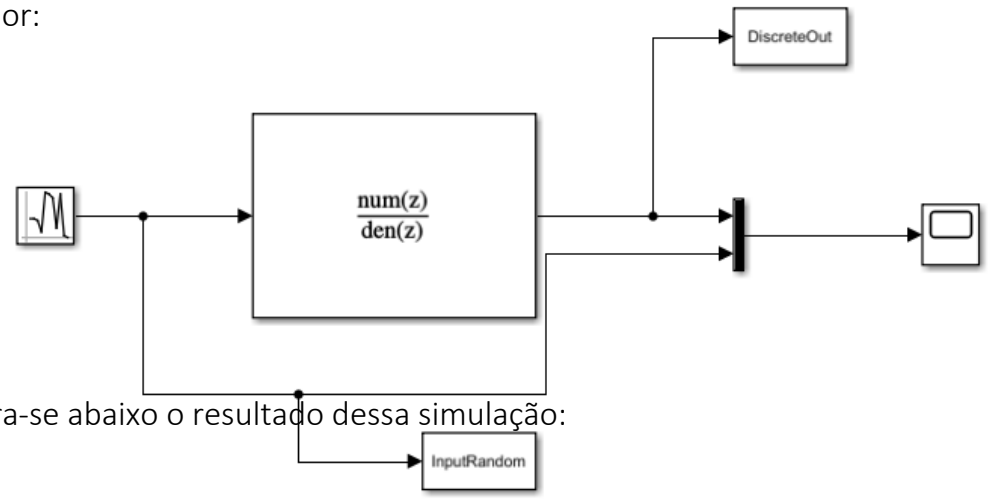
Precisávamos de gerar um conjunto de dados input/output representativo para a partir daí deduzir as regras necessárias. Para isso geramos uma entrada aleatória para a nossa função de transferência discreta. Esta última foi calculada da seguinte forma no ficheiro *run.m*:

```
numerator = [2];
denominator = [1 2 2.5 1.5];
[numerator, denominator]= c2dm(numerator,denominator, 1,'zoh');
```

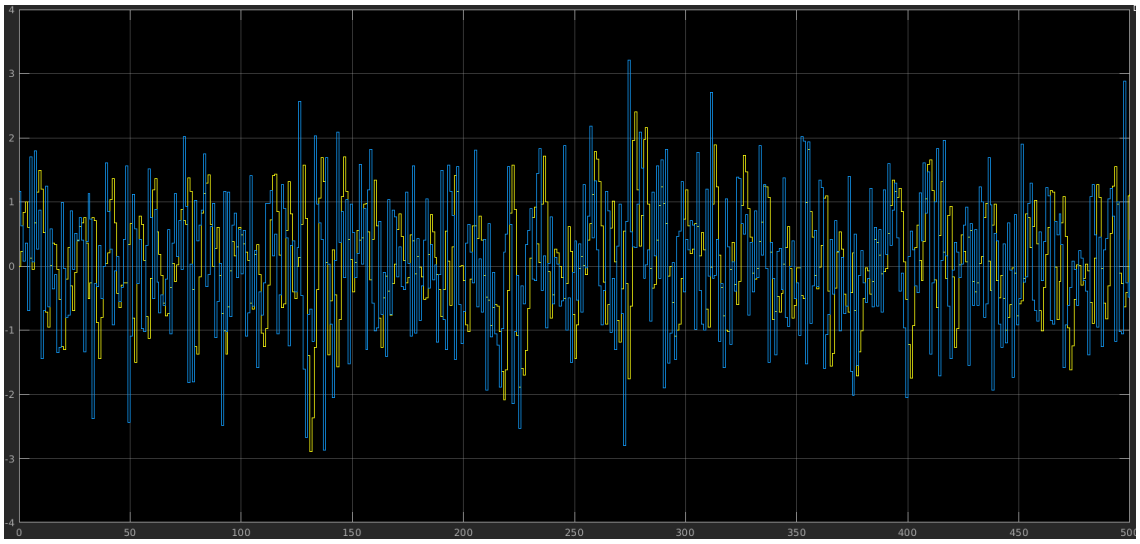
Obtiveram-se os seguintes coeficientes:

- Numerador = [0 0.1923 0.4424 0.0707]
- Denominador = [1 -0.8985 0.5630 -0.1353]

A partir daí foi só preciso construir o seguinte modelo no Simulink e correr o simulador:



Encontra-se abaixo o resultado dessa simulação:



A linha amarela representa a entrada aleatória que foi posteriormente guardada em ***InputRandom.mat*** e a saída está representada a azul e foi posteriormente guardada em ***DIscreteOut.mat***.

Matriz de Dados:

A partir destes ficheiros *.mat* vamos criar a nossa matriz de dados para clustering que será guardada em *matrix.dat.* Conforme explicado no enunciado, esta matriz terá o tamanha de (N-k)x7 onde N representa o tamanho dos vetores de entrad e saída; k é o instante em que se inicializa o alvo. Para **k = 3 temos**:

> 1ª coluna: série temporal da saída deslocada duas linhas para cima ($i = 2..(N-1)$). 2ª coluna: a mesma coisa só que desloca-se apenas uma linha para cima ($i = 1..(N-2)$). A 3ª coluna contém a série temporal da saída como se obteve na simulação ($i = 0..(N-3)$). A quarta coluna contém a série temporal da entrada deslocada duas linhas para cima ($i = 2..(N-1)$). Na 5ª coluna desloca-se uma linha para cima ($i = 1..(N-2)$); na 6ª teremos a série temporal da entrada como se obteve na simu- lação ($i = 0..(N-3)$). Na 7ª desloca-se 3 linhas para cima, ela e o *target* (($i = 3..N$).

O cálculo desta matriz é feito no **script *run.m*** e traduzir-se-á em código no seguinte:

```
load('DiscreteOut.mat');
load('InputRandom.mat');


DataSet = [];
for i=4:length(DiscreteOut)
    aux = [DiscreteOut(i-1) DiscreteOut(i-2) DiscreteOut(i-3) InputRandom(i-1) InputRandom(i-2) InputRandom(i-3) DiscreteOut(i)];
    DataSet = [DataSet; aux];
end
```

Regras Difusas:

Neste trabalho, é necessário aplicar regras difusas na seguinte equação:

$y(k) = f\,(y(k-1),\, y(k-2),\, y(k-3),\, u(k-1),\, u(k-2),\, u(k-3)$ Assim,

obtiveram-se as seguintes regras genéricas para a equação anterior:

IF $y(k-1)$ is A_1 AND $y(k-2)$ is A_2 AND $y(k-3)$ is A_3 AND

$u(k-1)$ is A_4 AND $u(k-2)$ is A_5 AND $u(k-3)$ is A_6 THEN $y(k)$ is α

O nosso *target* está localizado no instante 3 ($k = 3$), tal como explicado na secção anterior, logo obtém-se as seguintes regras:

IF $y(2)$ is A_1 AND $y(1)$ is A_2 AND $y(0)$ is A_3 AND

$u(2)$ is A_4 AND $u(1)$ is A_5 AND $u(0)$ is A_6 THEN $y(3)$is α

Geração das Regras e Clustering:

Tendo matriz de dados, podemos agora fazer o clustering. Usámos duas técnicas:

- Fcm
- Subtractive Clustering

Com recurso à função *findcluster* do Matlab obtivemos os seguintes clusters (assinalados com pontos pretos) usando o método ***fcm*** e ***subtractive clustering*** respetivamente:

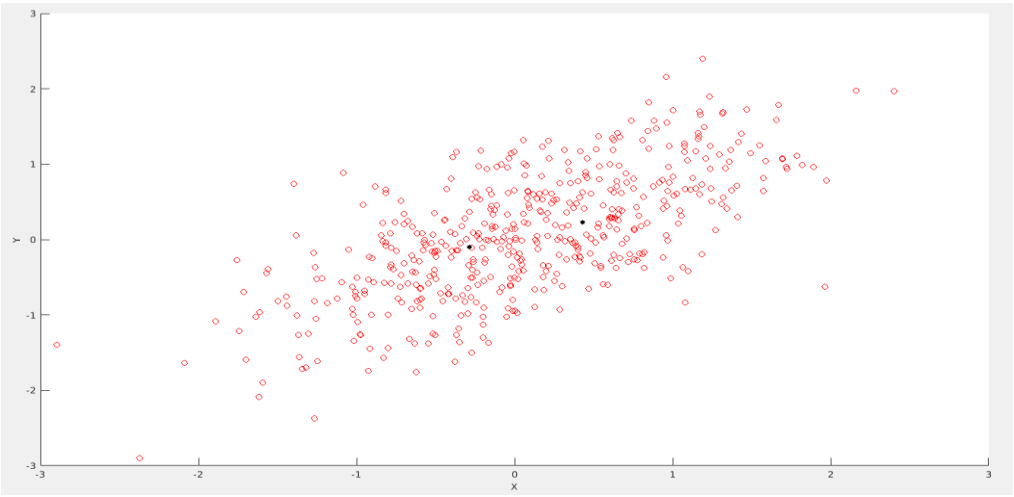


Figura: gráfico do *findclusters* usando *fcm*

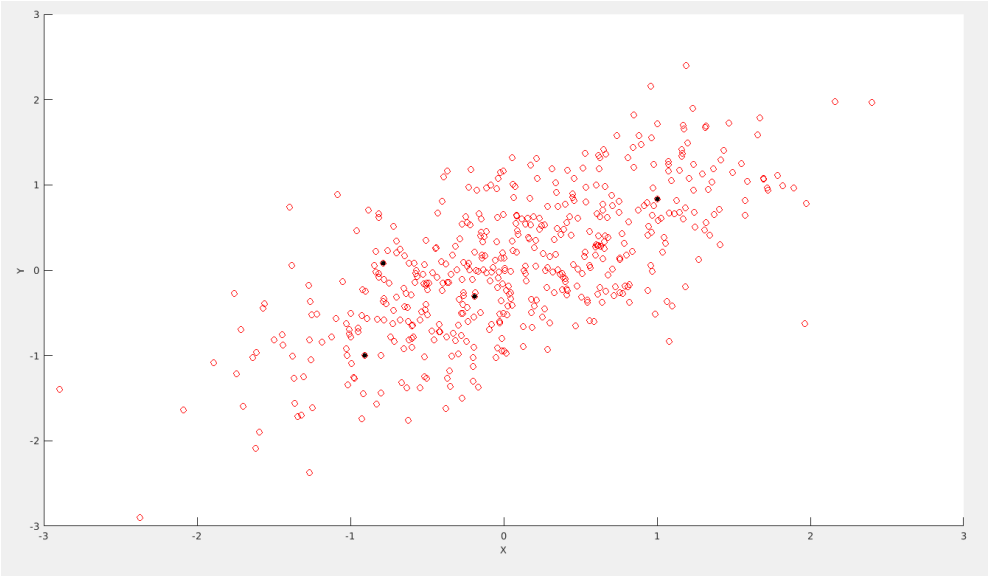


Figura: gráfico do *findclusters* usando *subtractive clustering*

A presença de vários clusters evidencia a dispersão dos dados.

Subtractive Clustering

Para aplicar esta técnica de clustering decidimos recorrer à função *neuroFuzzyDesigner*:

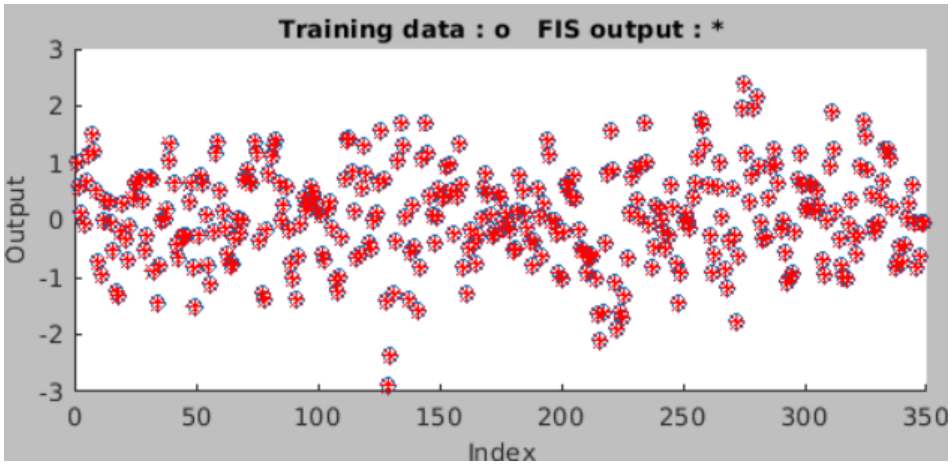
1. Formámos os dados de treino e de teste da seguinte forma:

```
testingSet = DataSet(round(length(DataSet)*0.7)+1:length(DataSet),:);
trainingSet = DataSet(1:round(length(DataSet)*0.7),:);
```

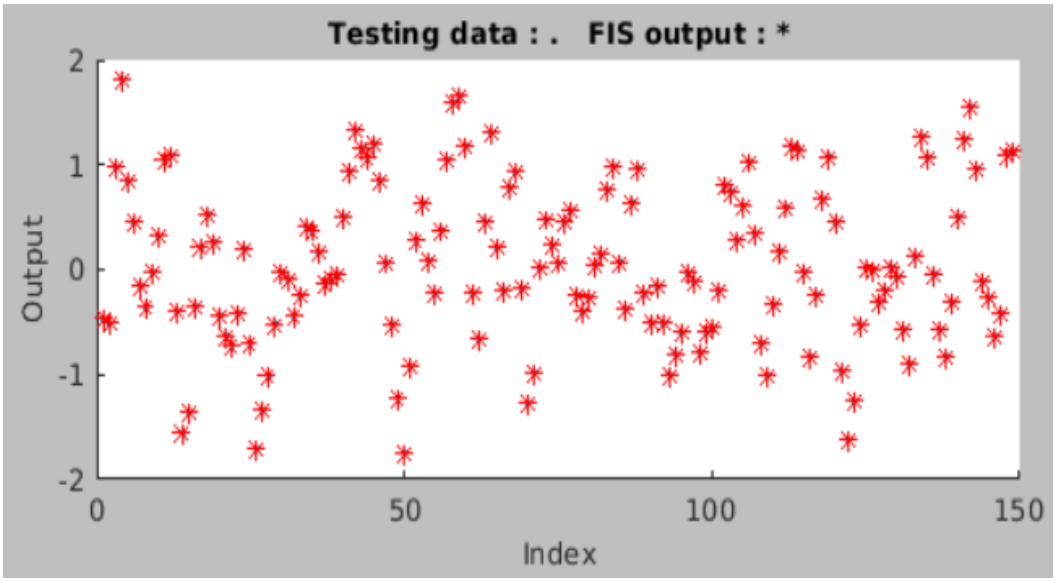
2. Fizemos o *load* desses dados através da GUI

3. Gerámos automaticamente as regras usando a técnica de *subtractive clustering* recorrendo à opção **"generate fis"**.

4. Otimizámos recorrendo ao **treino híbrido** com **500 épocas**:
    - O MSE no final do treino para os resultados do treino foi de **2.1014e-07**. O gráfico representa os **resultados do treino**:

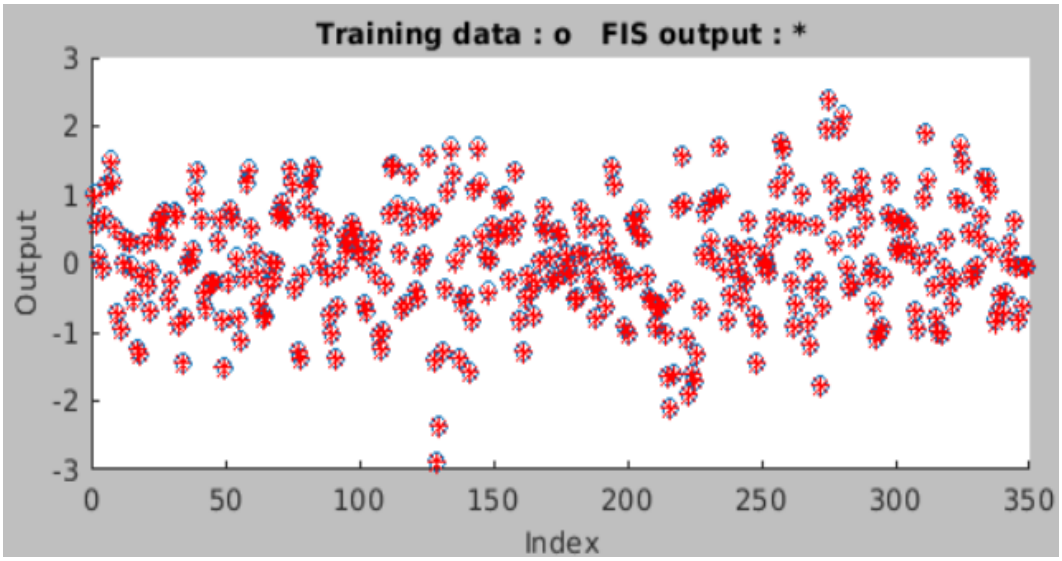


- O MSE no final do treino para os resultados do teste foi de 2.2455e-07. O gráfico representa os resultados do teste:

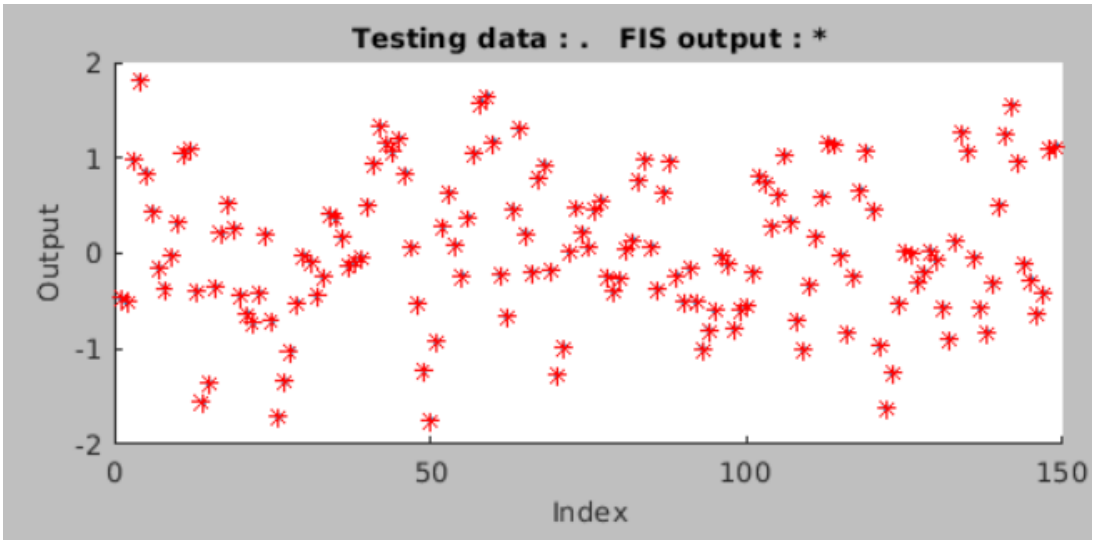


5. Otimizámos recorrendo ao treino de backpropagation com 500 épocas:

- MSE no final do treino para os resultados do treino foi de 0.0044666. O gráfico representa os resultados do treino:

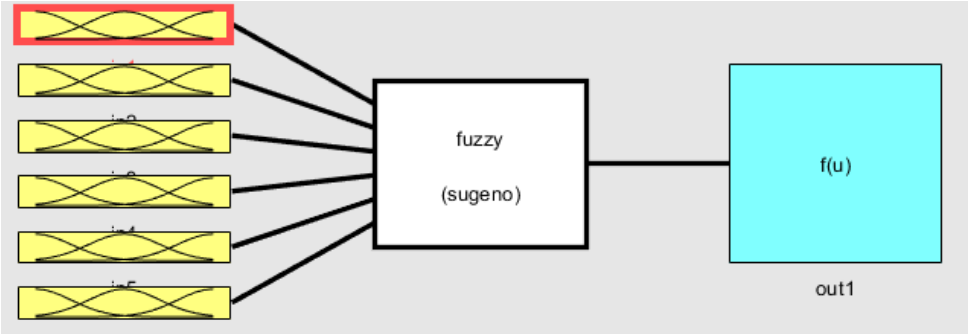


- MSE no final do treino para os resultados do teste foi de 0.0040541.
  O gráfico representa os resultados do teste:



6. Guardámos o controlador gerado como *subtractiveClustering.fis*. Este possui a seguinte arquitetura e funções de pertença (todas elas gaussianas - *gaussfm*):

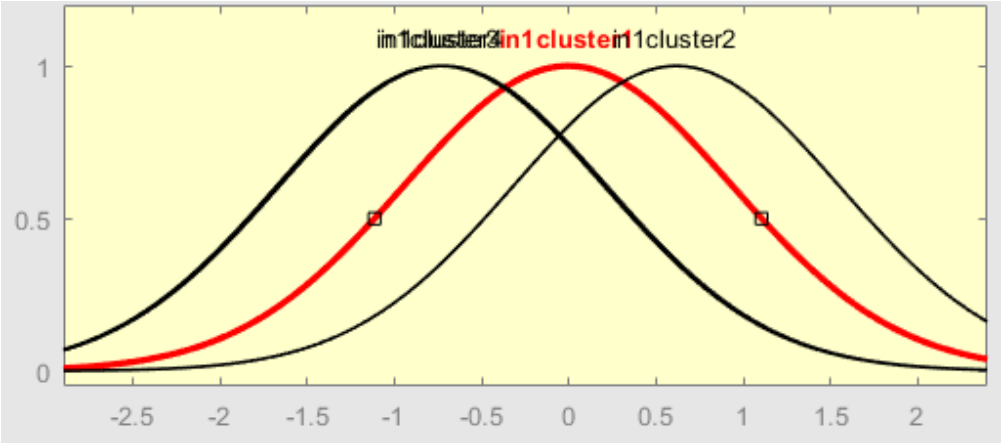
   a. Arquitetura



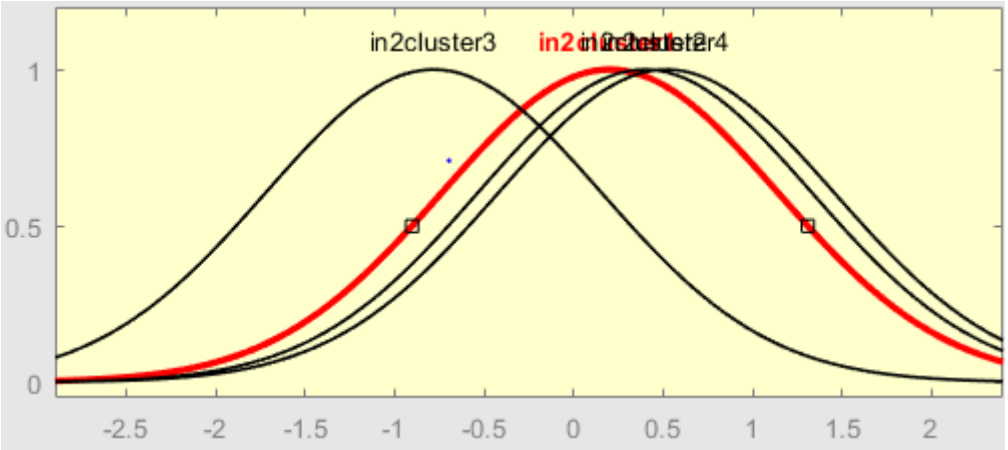
   b. Regras

1. If (in1 is in1cluster1) and (in2 is in2cluster1) and (in3 is in3cluster1) and (in4 is in4cluster1) and (in5 is in5cluster1) and (in6 is in6cluster1) then (out1 is out1cluster1) (1)
2. If (in1 is in1cluster2) and (in2 is in2cluster2) and (in3 is in3cluster2) and (in4 is in4cluster2) and (in5 is in5cluster2) and (in6 is in6cluster2) then (out1 is out1cluster2) (1)
3. If (in1 is in1cluster3) and (in2 is in2cluster3) and (in3 is in3cluster3) and (in4 is in4cluster3) and (in5 is in5cluster3) and (in6 is in6cluster3) then (out1 is out1cluster3) (1)
4. If (in1 is in1cluster4) and (in2 is in2cluster4) and (in3 is in3cluster4) and (in4 is in4cluster4) and (in5 is in5cluster4) and (in6 is in6cluster4) then (out1 is out1cluster4) (1)

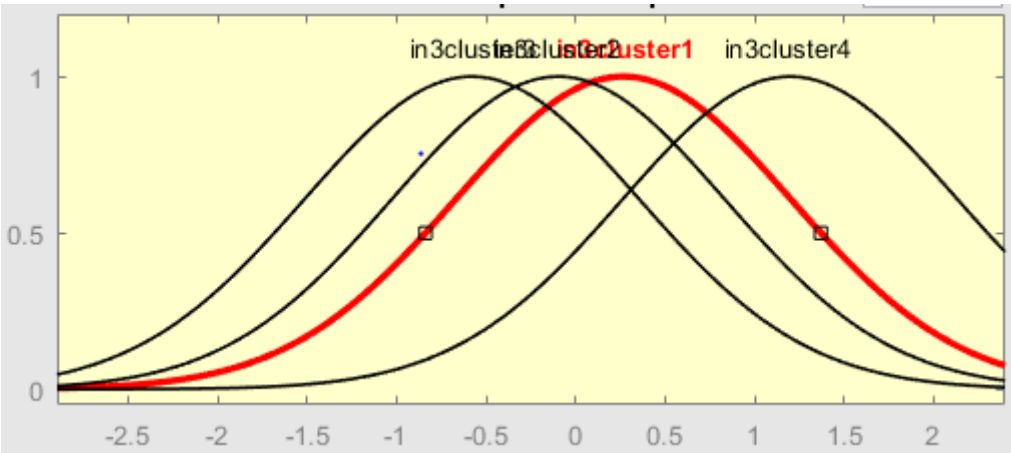
c. Funções de pertença do Input1



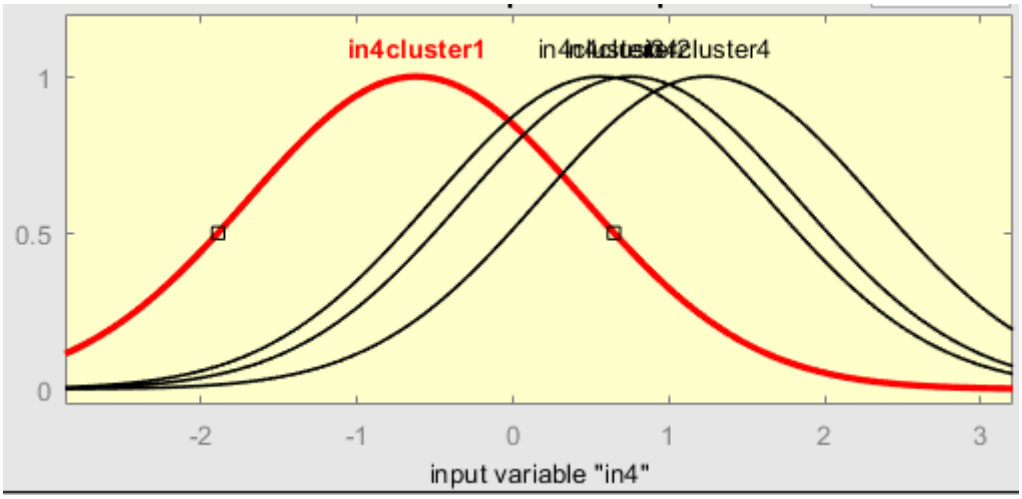
d. Funções de pertença do Input2



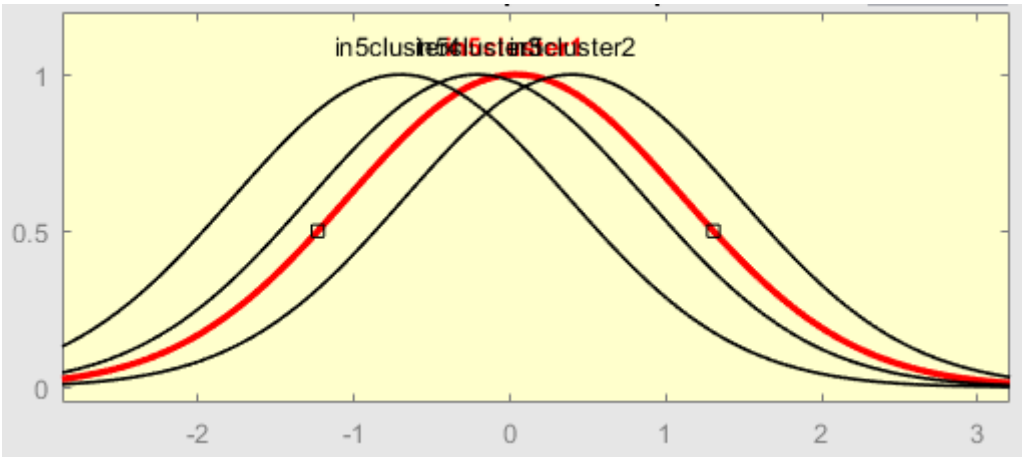
e. Funções de pertença do Input3



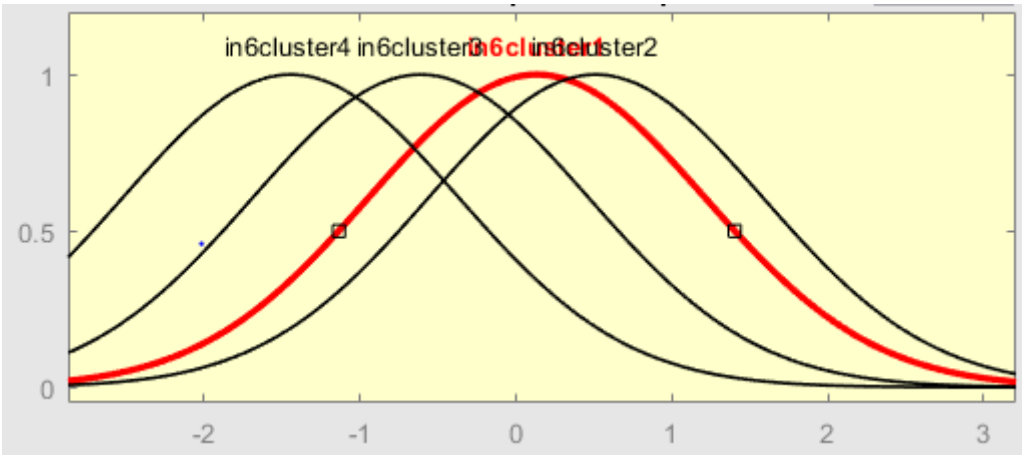
f. Funções de pertença do Input4



g. Funções de pertença do Input5



h. Funções de pertença do Input6



Clustering Fcm:

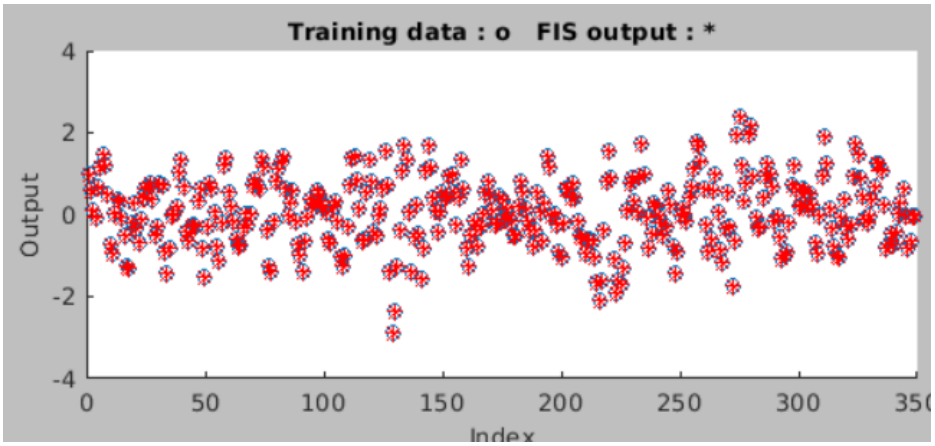
Para adotar a técnica de clustering fcm poderíamos fazer o .fis de forma manual conforme o enunciado aconselha ou poderíamos recorrer à funçãi **genfis3**. Optámos pela segunda opção conforme mostra a seguinte linha de código retirada do script ***run.m***.

```
fismat = genfis3(trainingSet(:,1:6), trainingSet(:,7), 'sugeno',5);
```

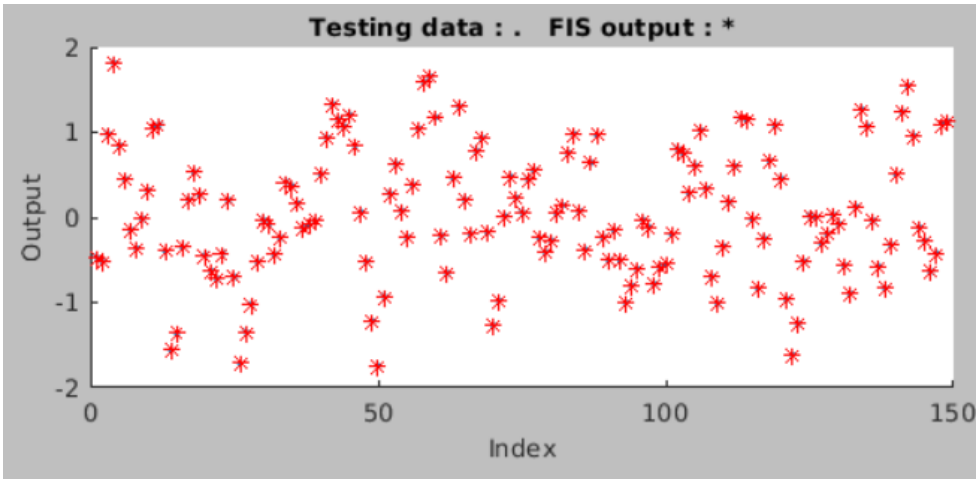
De forma a otimizar voltamos a recorrer ao *neuroFuzzyDesigner*:

1. Voltámos a fazer o **load** dos dados de teste e treino
2. Em vez de fazermos "generate fis" fizemos load do nosso fcmFis.fis
3. Otimizámos recorrendo ao **treino híbrido** com **500 épocas**:

- O MSE no final do treino para os resultados do treino foi de **2.5521e-07**. O gráfico representa os **resultados do treino**:

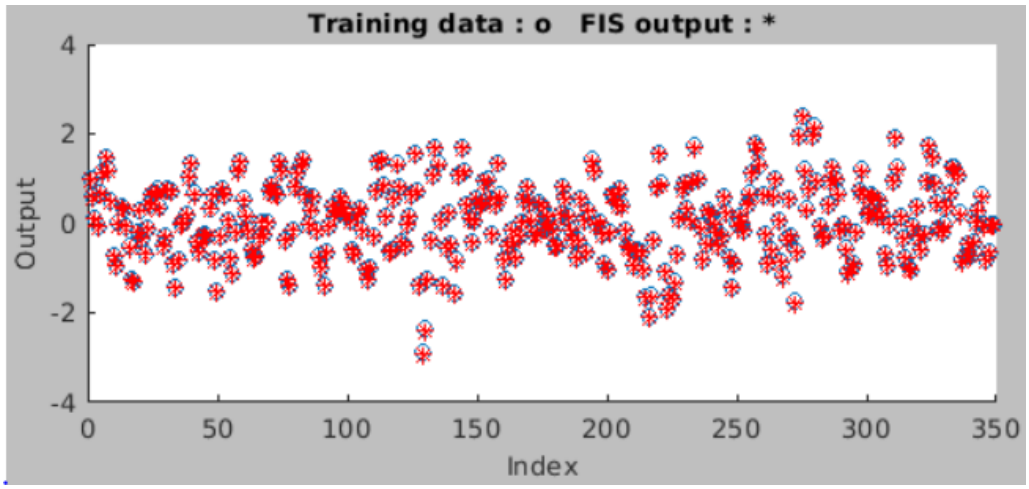


- O MSE no final do treino para os resultados do teste foi de **2.3668e-07**. O gráfico representa os **resultados do teste**:

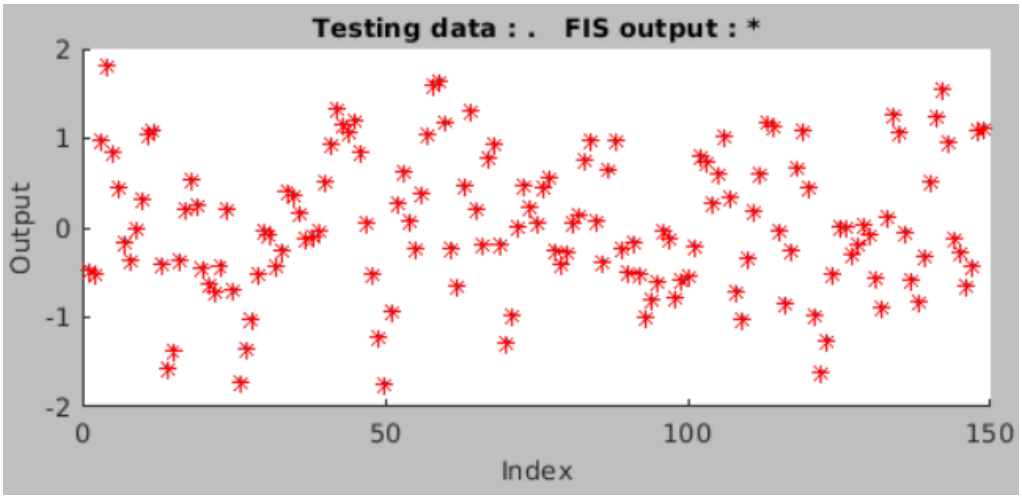


4. Otimizámos recorrendo ao **treino de backpropagation** com **500 épocas**:

   - MSE no final do treino para os resultados do treino foi de **0.0053692**.
     O gráfico representa os **resultados do treino**:

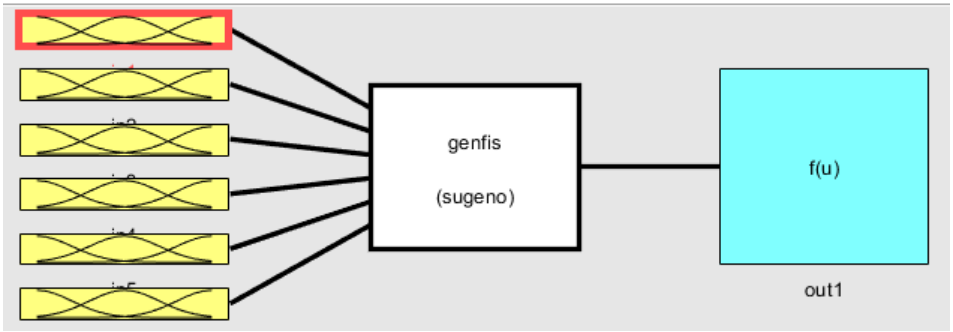


   - MSE no final do treino para os resultados do teste foi de **0.0048412**.
     O gráfico representa os **resultados do teste**:



5. Guardámos o controlador gerado como ***fcmFis.fis.*** Este possui a seguinte arquitetura e funções de pertença (todas elas gaussianas - *gaussfm*):

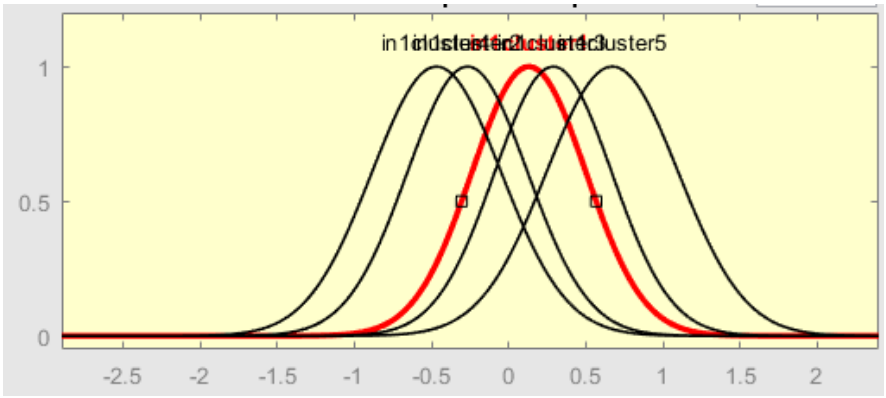
   a. Arquitetura



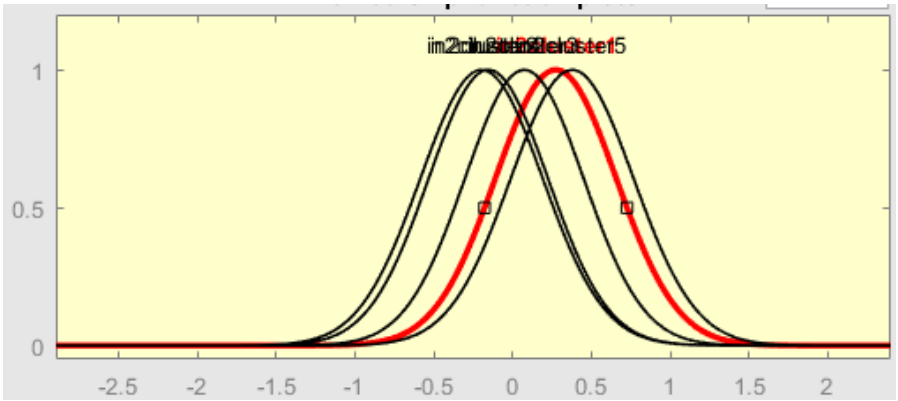
   b. Regras

1. If (in1 is in1cluster1) and (in2 is in2cluster1) and (in3 is in3cluster1) and (in4 is in4cluster1) and (in5 is in5cluster1) and (in6 is in6cluster1) then (out1 is out1cluster1) (1)
2. If (in1 is in1cluster2) and (in2 is in2cluster2) and (in3 is in3cluster2) and (in4 is in4cluster2) and (in5 is in5cluster2) and (in6 is in6cluster2) then (out1 is out1cluster2) (1)
3. If (in1 is in1cluster3) and (in2 is in2cluster3) and (in3 is in3cluster3) and (in4 is in4cluster3) and (in5 is in5cluster3) and (in6 is in6cluster3) then (out1 is out1cluster3) (1)
4. If (in1 is in1cluster4) and (in2 is in2cluster4) and (in3 is in3cluster4) and (in4 is in4cluster4) and (in5 is in5cluster4) and (in6 is in6cluster4) then (out1 is out1cluster4) (1)
5. If (in1 is in1cluster5) and (in2 is in2cluster5) and (in3 is in3cluster5) and (in4 is in4cluster5) and (in5 is in5cluster5) and (in6 is in6cluster5) then (out1 is out1cluster5) (1)

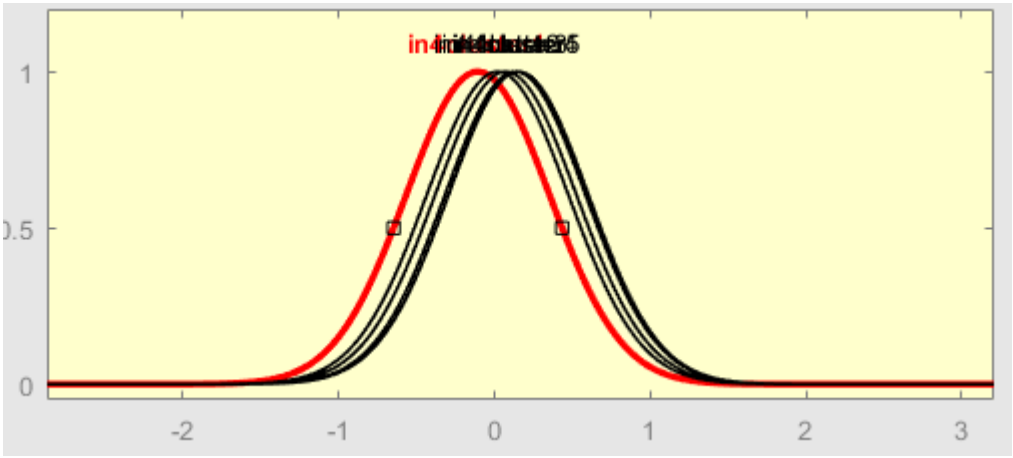
   c. Funções de pertença do Input1



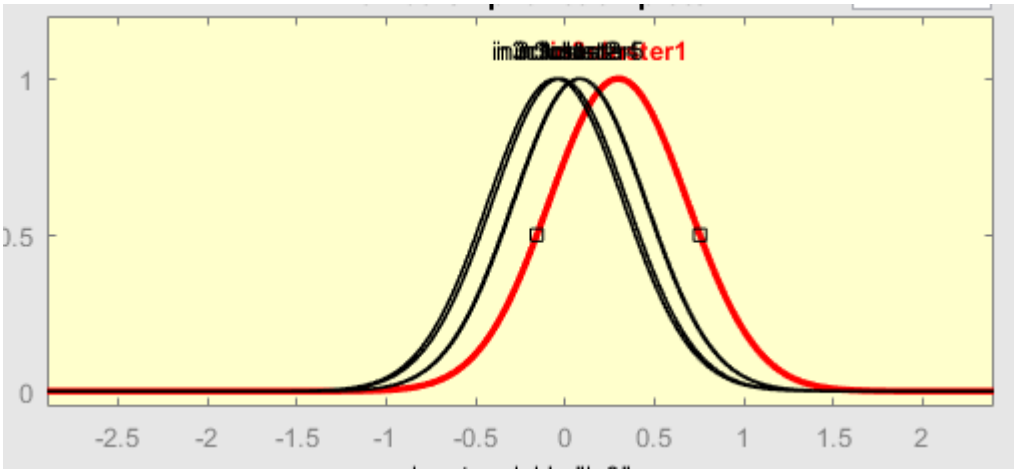
   d. Funções de pertença do Input2



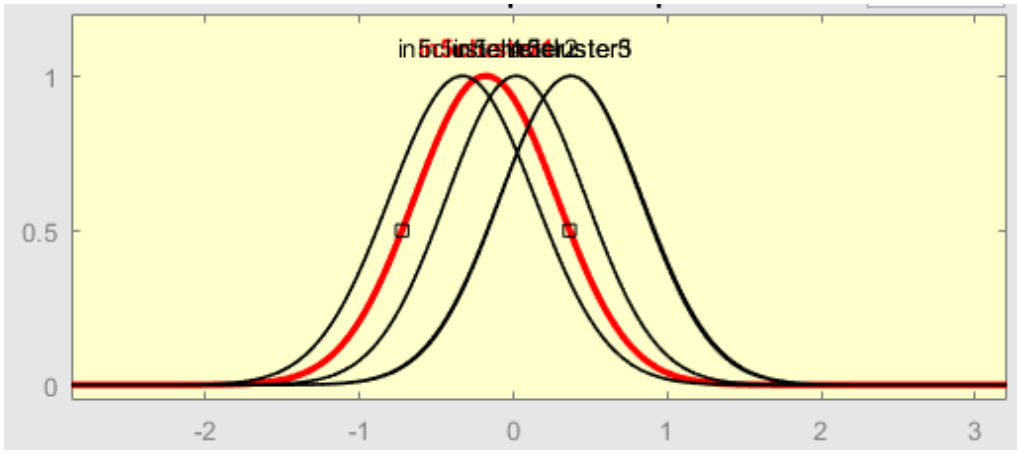
e. Funções de pertença do Input3



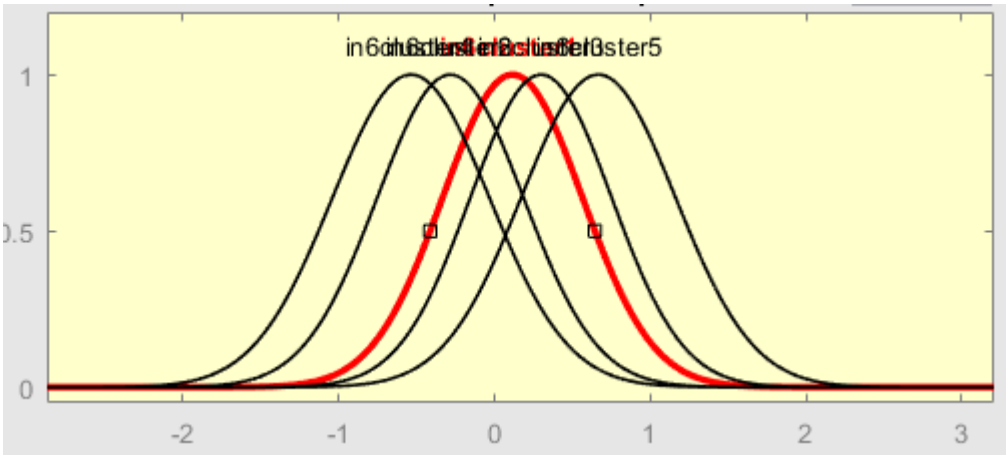
f. Funções de pertença do Input4



g. Funções de pertença do Input5



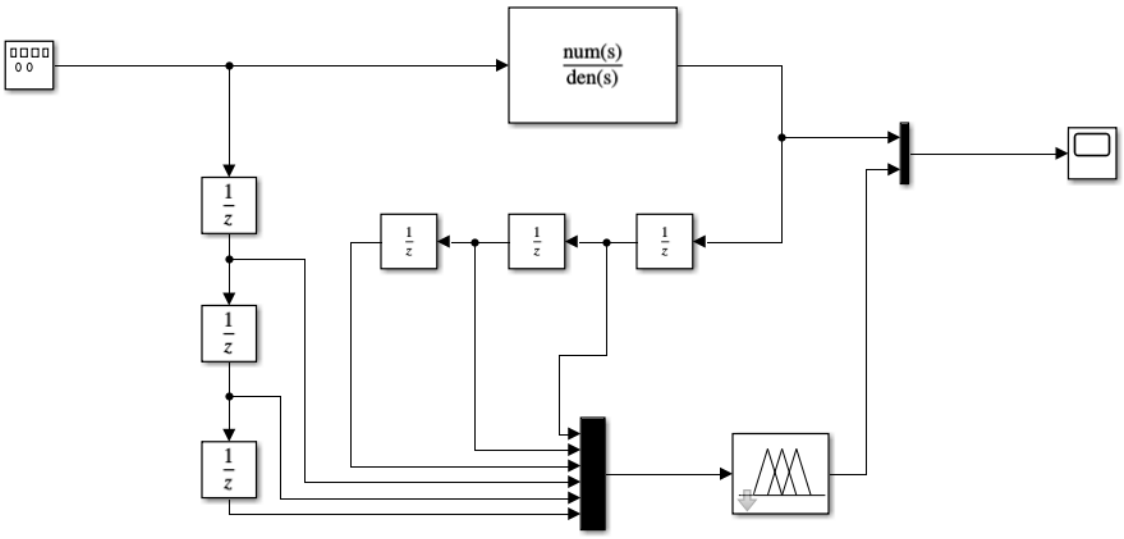
h. Funções de pertença do Input6



Testes e Observações:

Analisando para já os controladores criados podemos já concluir que o **treino híbrido dos controladores dá um MSE muito melhor** (mais pequeno).

Para fazer a simulação no Simulink usou-se o seguinte modelo:



Este modelo, para alem de ter a nossa função de transferência já revelada nas páginas acima, faz também uso de unidades de atraso. **Foram construídos ao todo 6 modelos diferentes que variam:**

- **Controlador:** pode ser aquele calculado a partir do *clustering fcm* ou por *subtractive clustering*

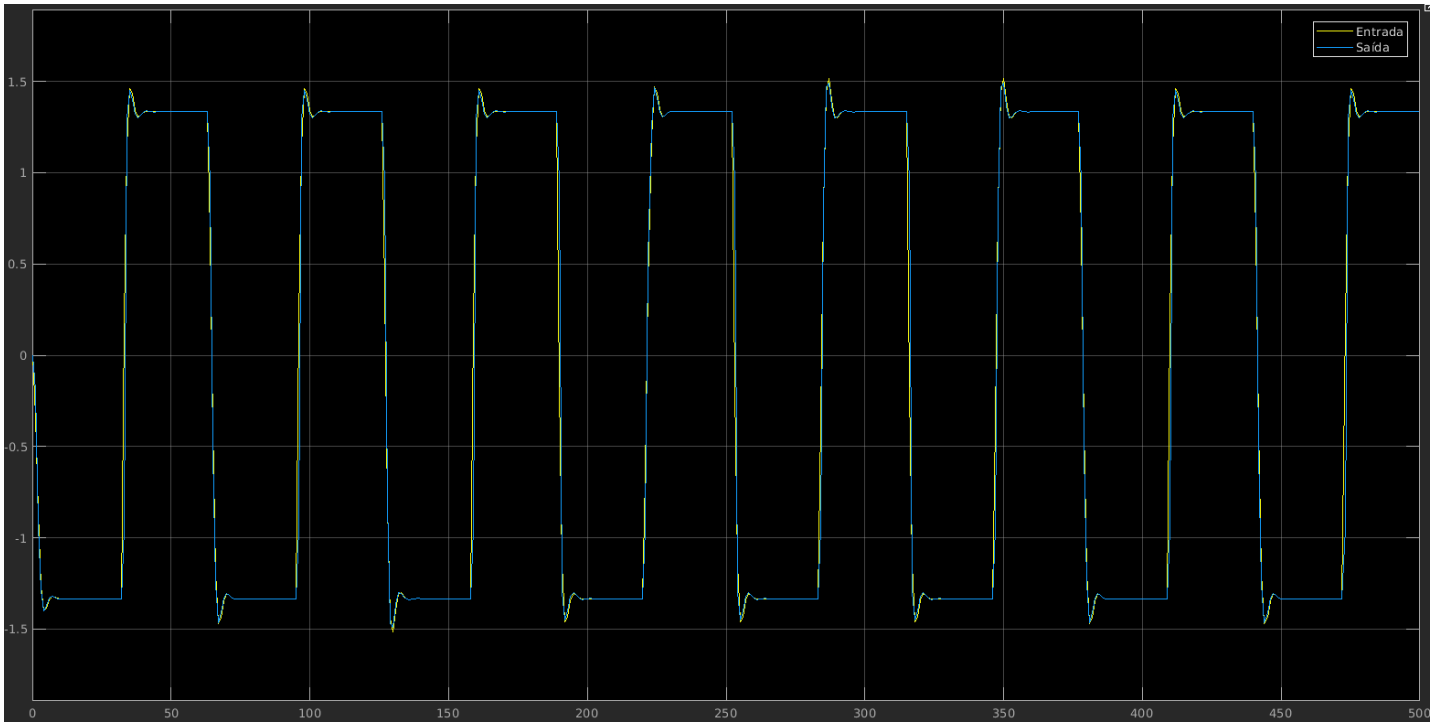
- **Referência desejada:** sawtooth, square ou sine.

Teve-se também o cuidado de ajustar os sample times para **um quarto da menor constante temporal**. No script ***run.m*** o seguinte excerto de código implementa o raciocínio necessário para tal:

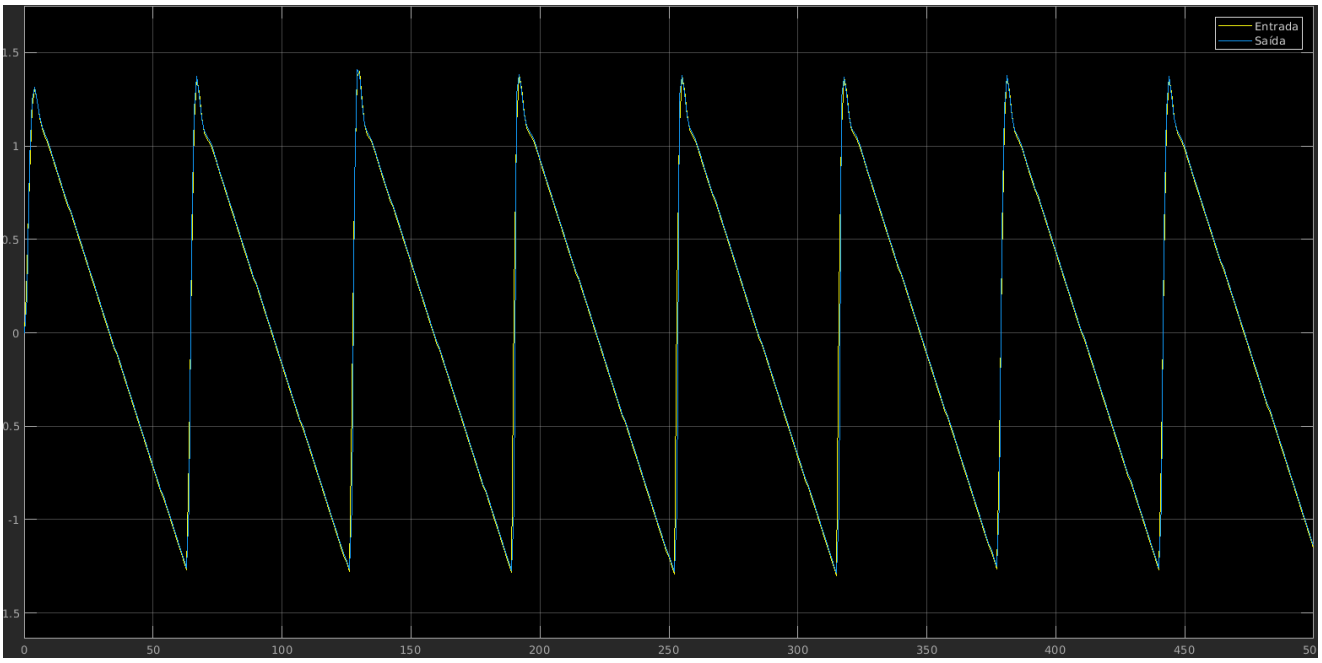
```
Roots = roots(denominator);
time = zeros(length(Roots),1);
for i=1:length(Roots)
    time(i)=-1/Roots(i)
end
```

Vejamos então os **gráficos** obtidos na simulação:

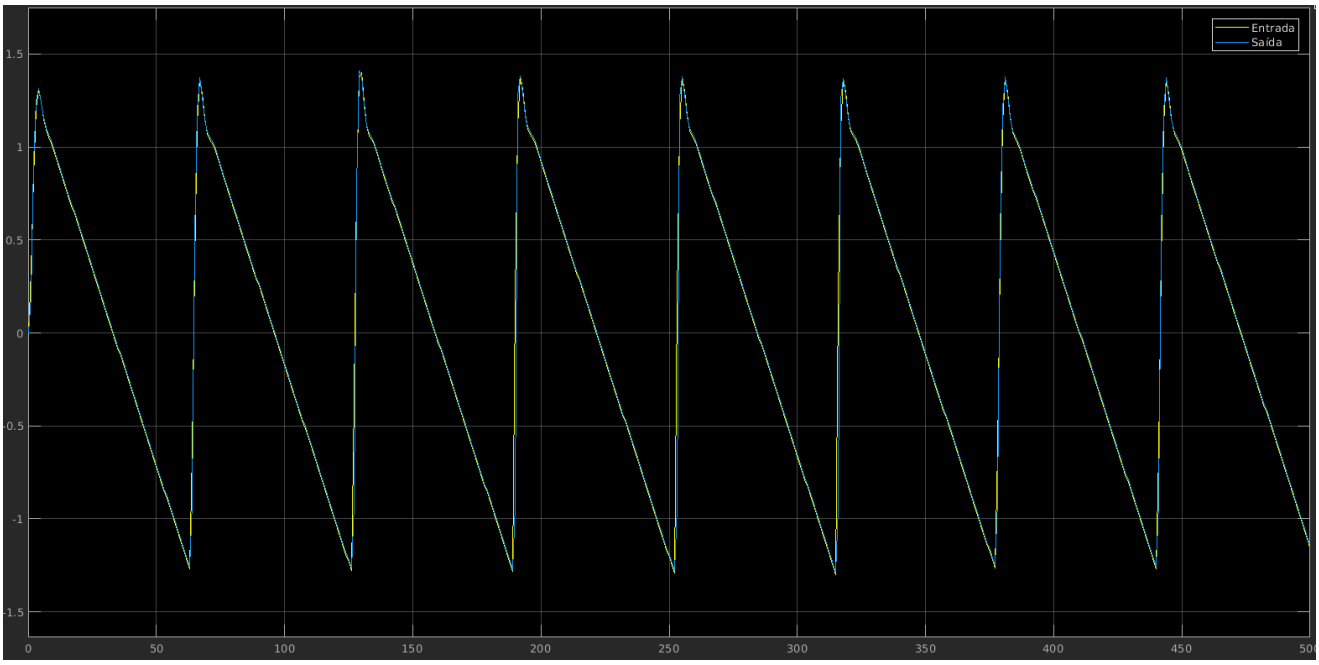
- Função de Referência: *Square*; subtractive clustering



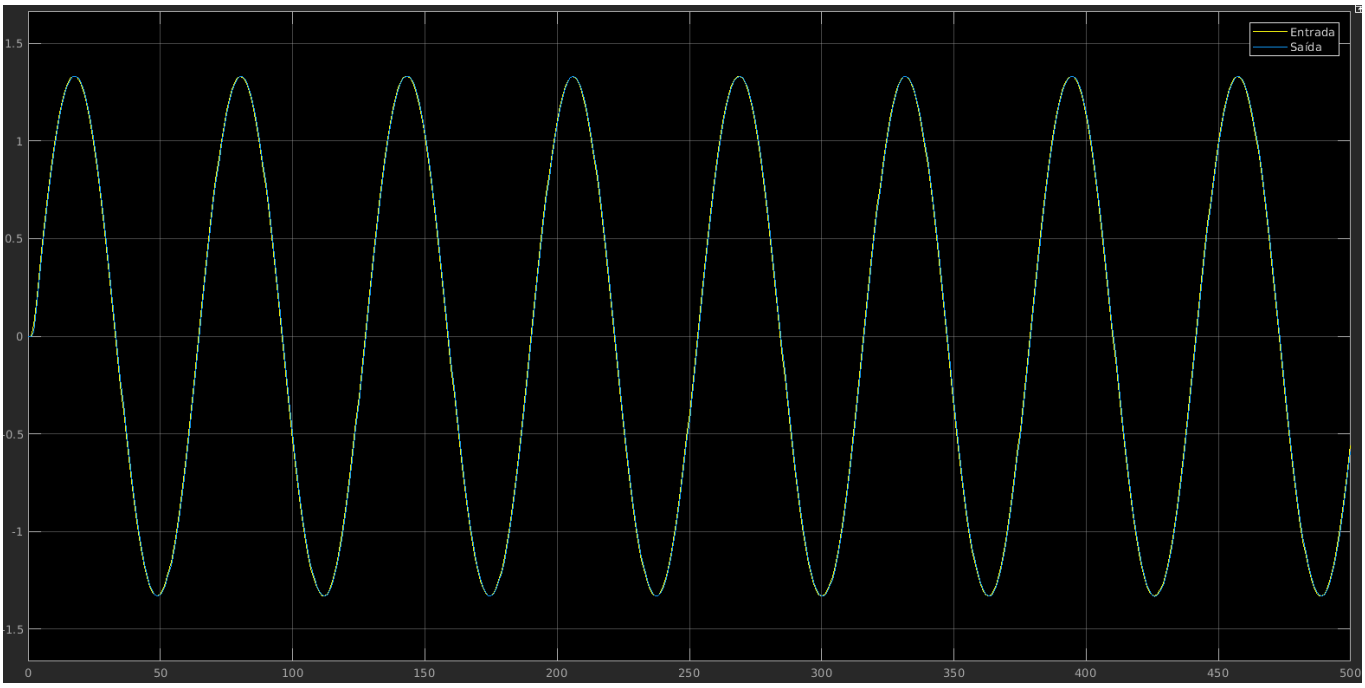
- Função de Referência: *Sawtooth*; Fcm



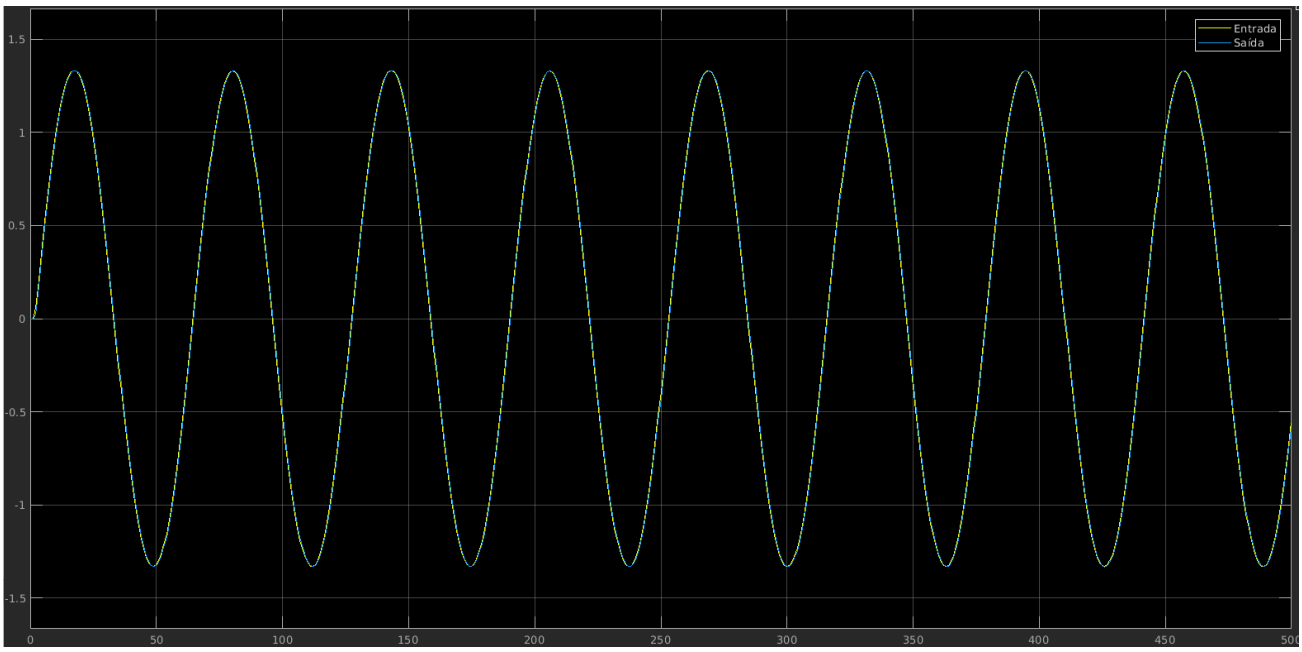
- Função de Referência: *Sawtooth*; subtractive clustering



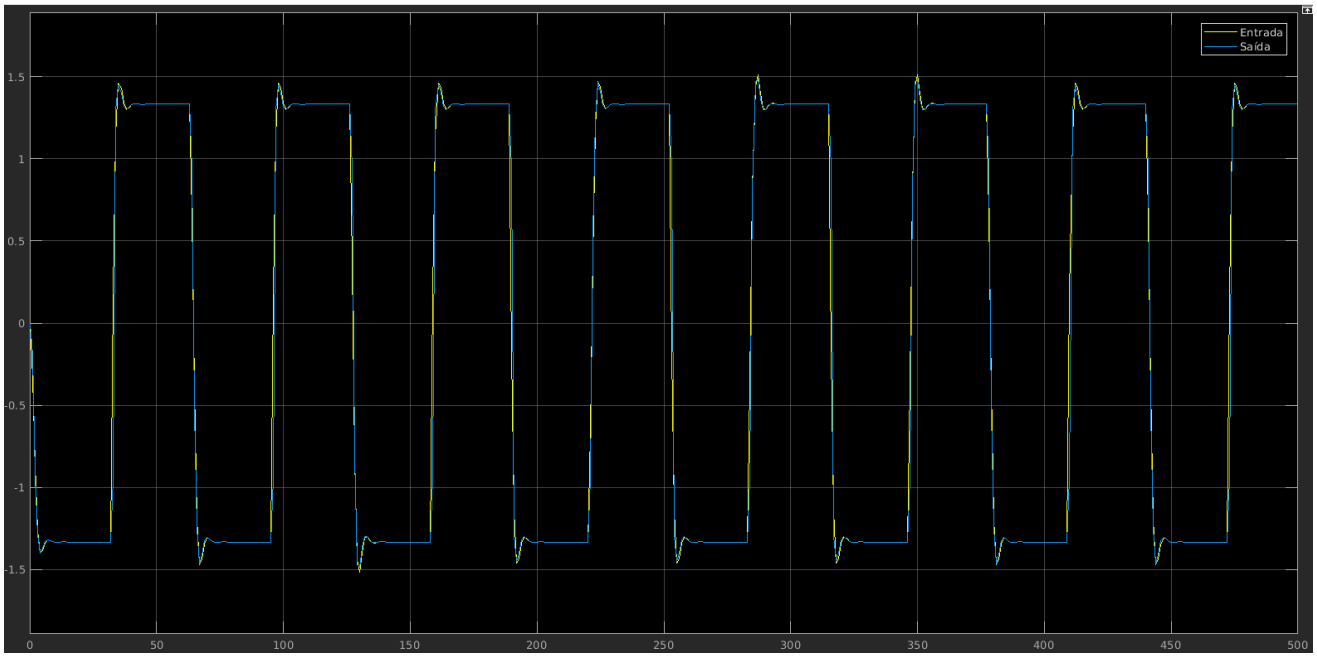
- Função de Referência: *Sine*; Fcm



- Função de Referência: *Sine*; subtractive clustering



- Função de Referência: *Square*; Fcm



Conclusões:

Da análise destes gráficos poderemos concluir que as **duas técnicas de clustering** são bastante eficientes produzindo resultados muito bons e indiferenciáveis a olho nu.

Vejamos também que se verifica uma melhoria de desempenho nas funções de Sawtooth e Sine pois as suas transição são menos abruptas que a função de Square.

Anexo A

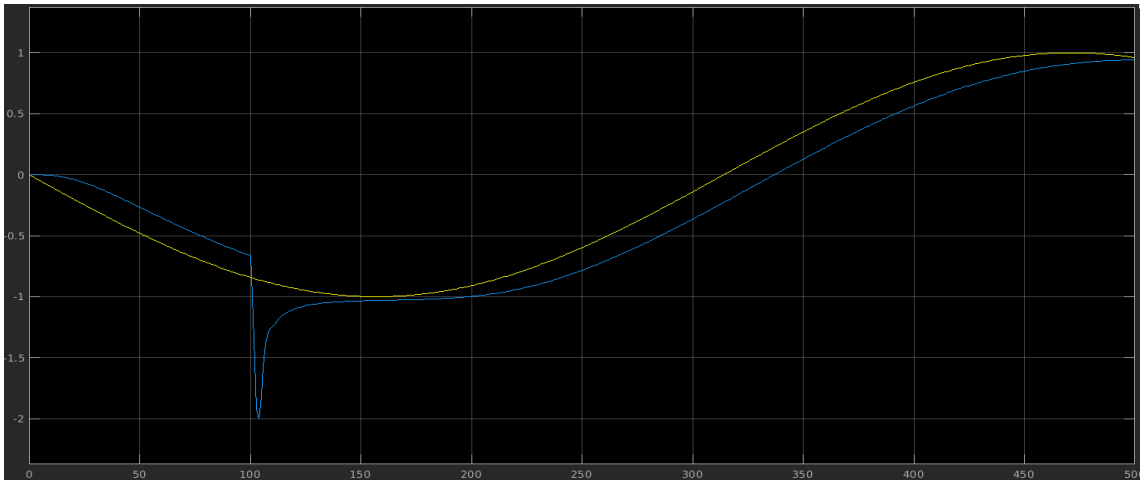


Figura 1: sine_9mamdani_centroid_atuador

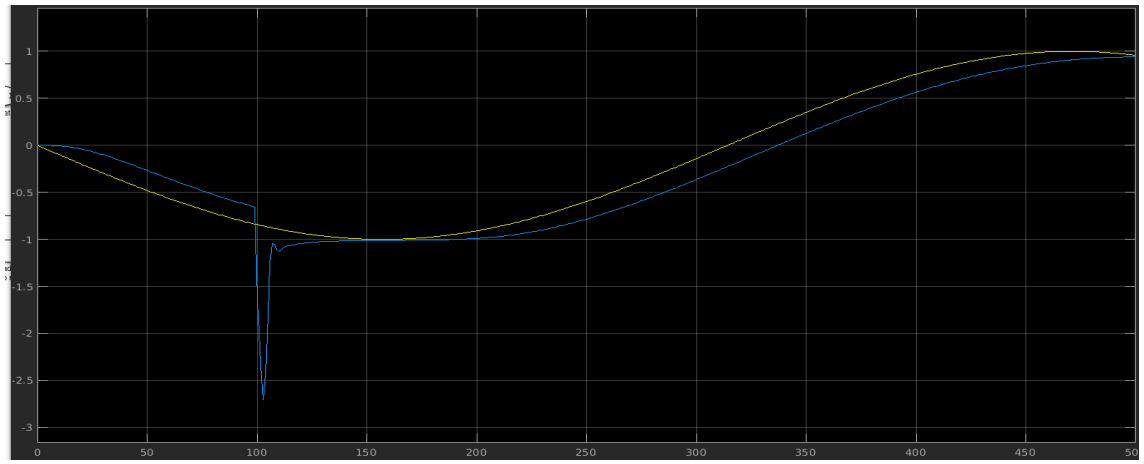


Figura 2: sine_9mamdani_centroid_both

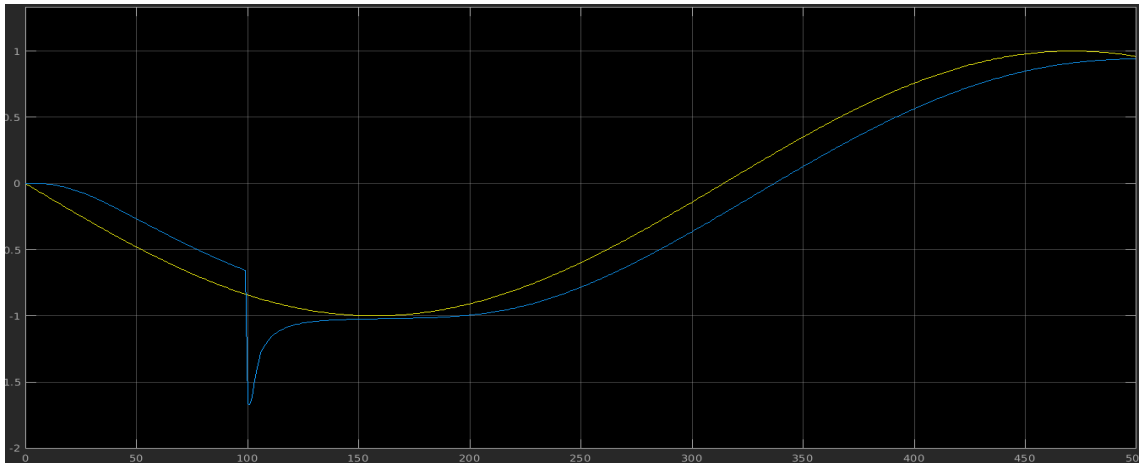


Figura 3: sine_9mamdani_centroid_carga

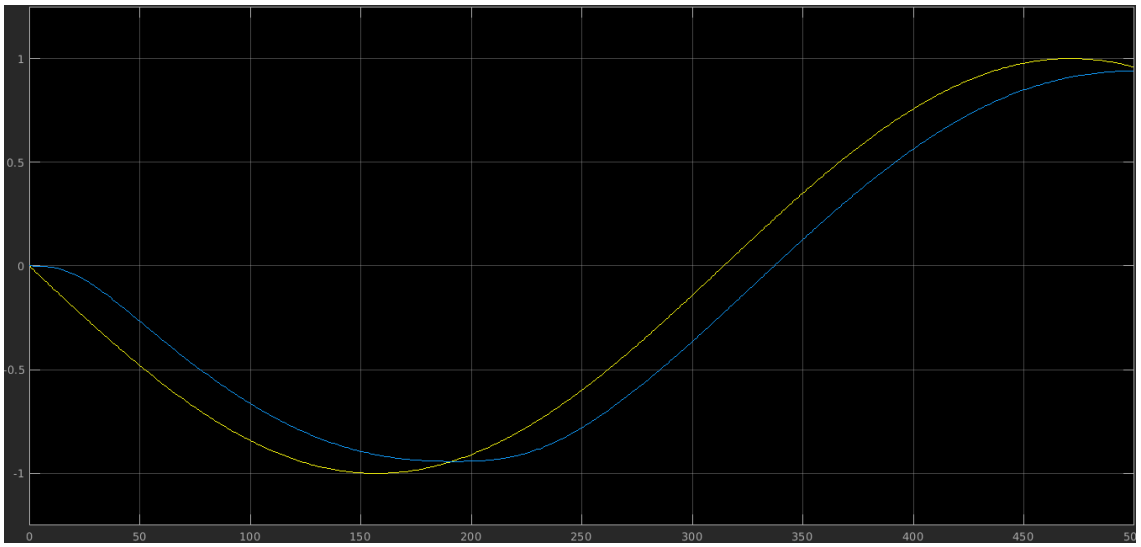


Figura 4: sine_9mamdani_centroid_nothing

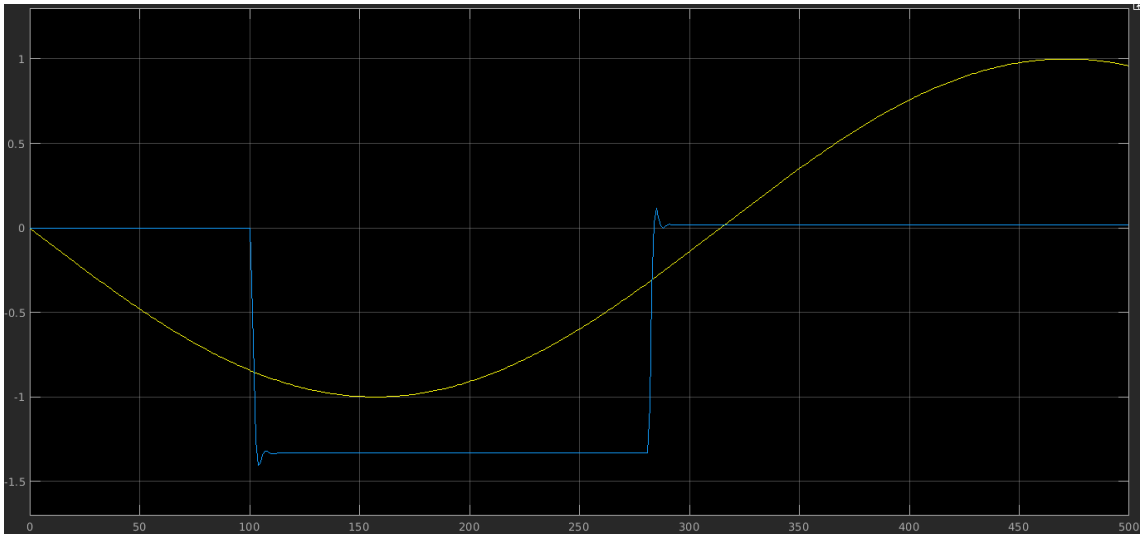


Figura 5: sine_9mamdani_mom_atuador

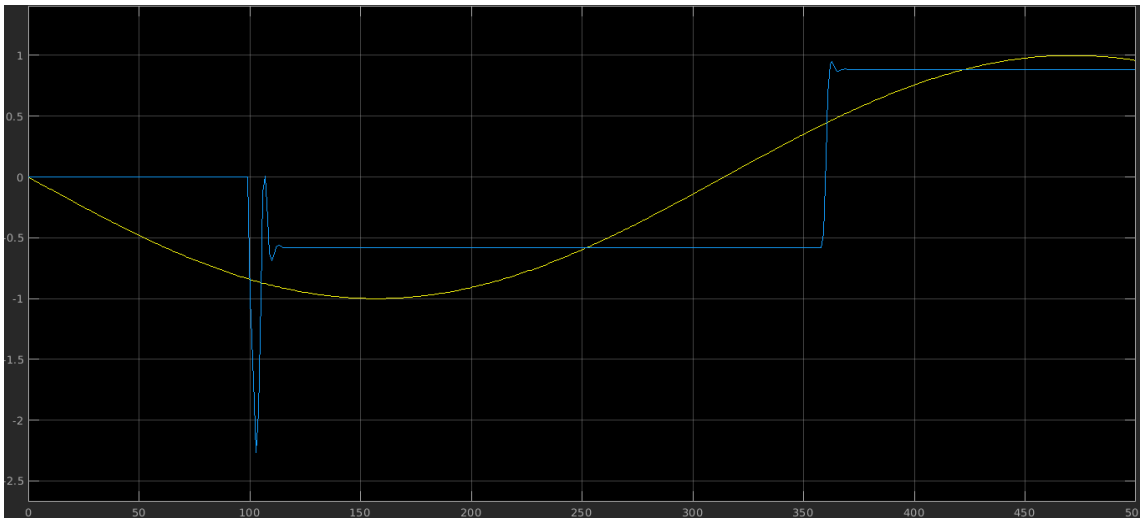


Figura 6: sine_9mamdani_mom_both

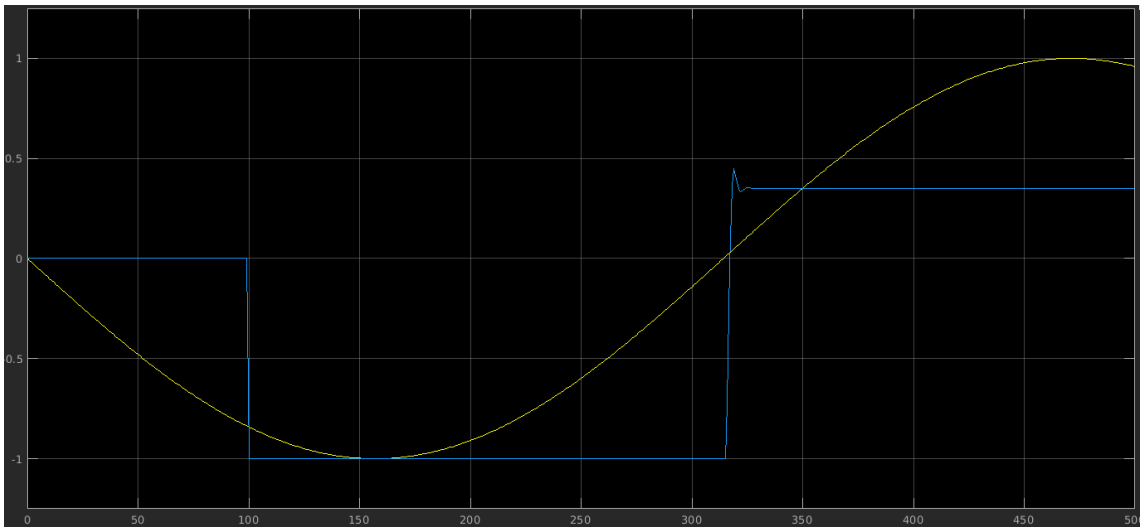


Figura 7: sine_9mamdani_mom_carga

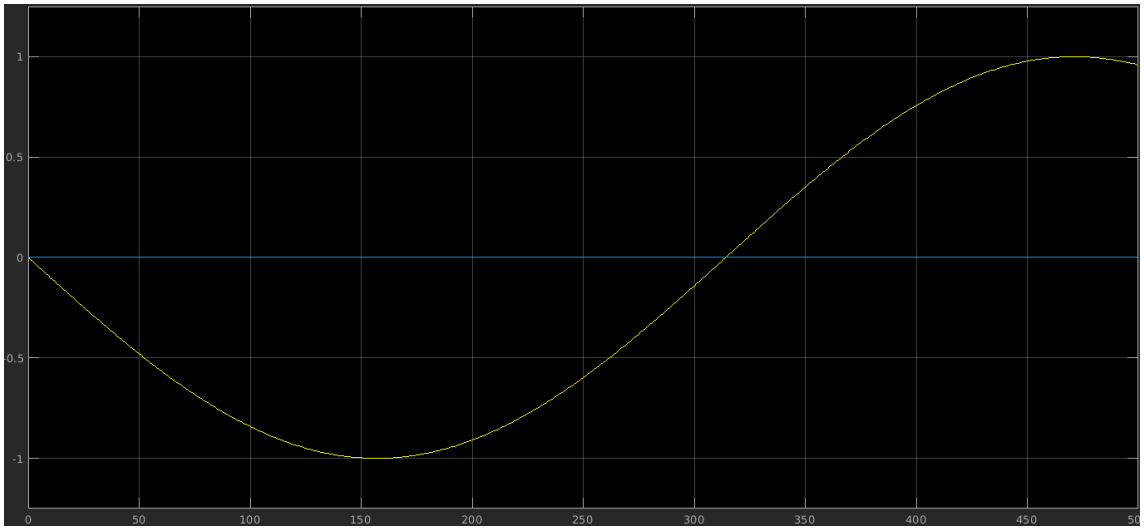


Figura 8: sine_9mamdani_mom_nothing

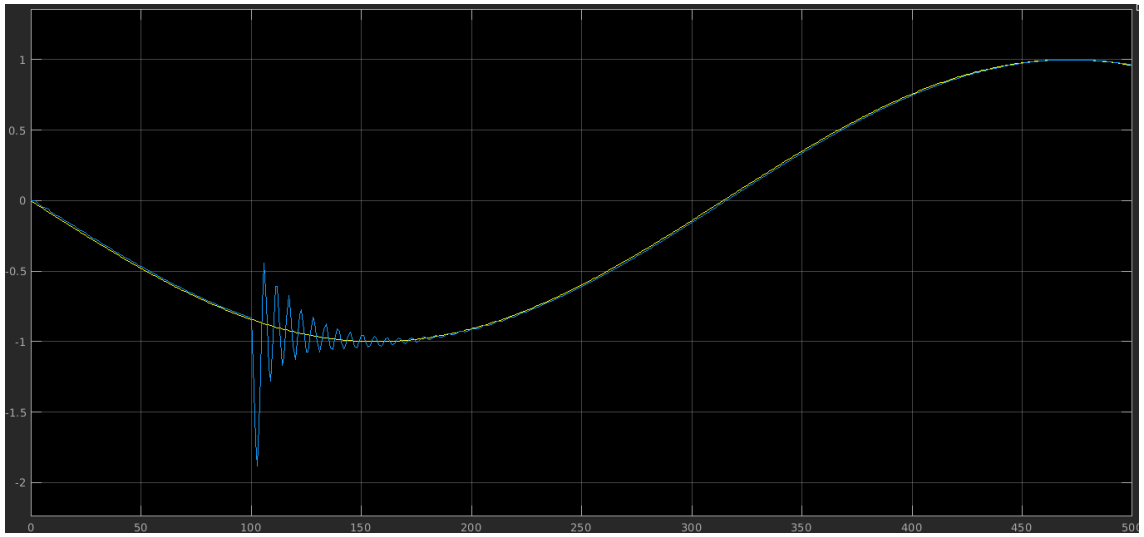


Figura 9: sine_9sugeno_wtaver_atuador

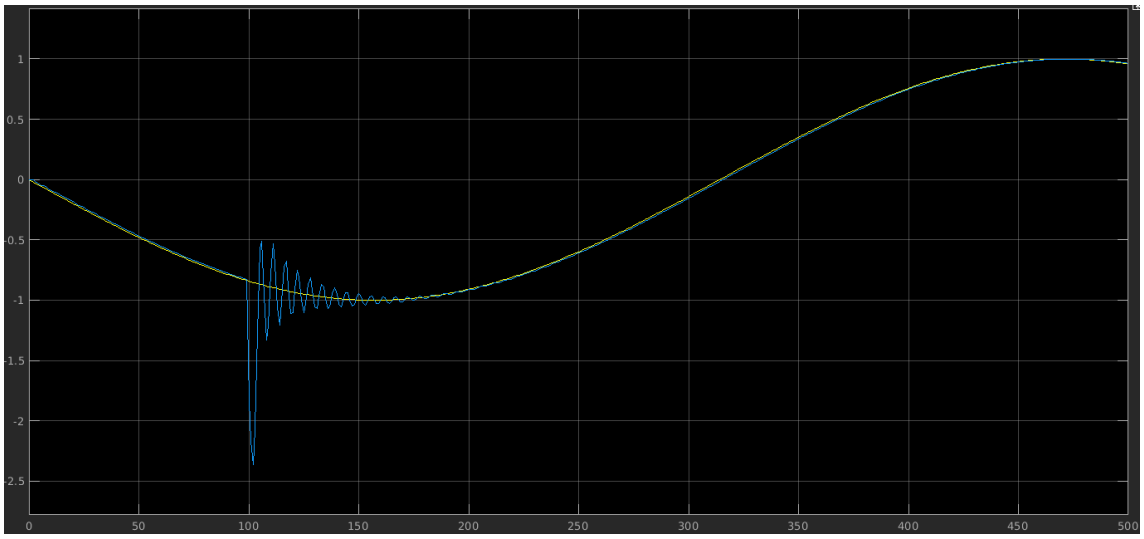


Figura 10: sine_9sugeno_wtaver_both

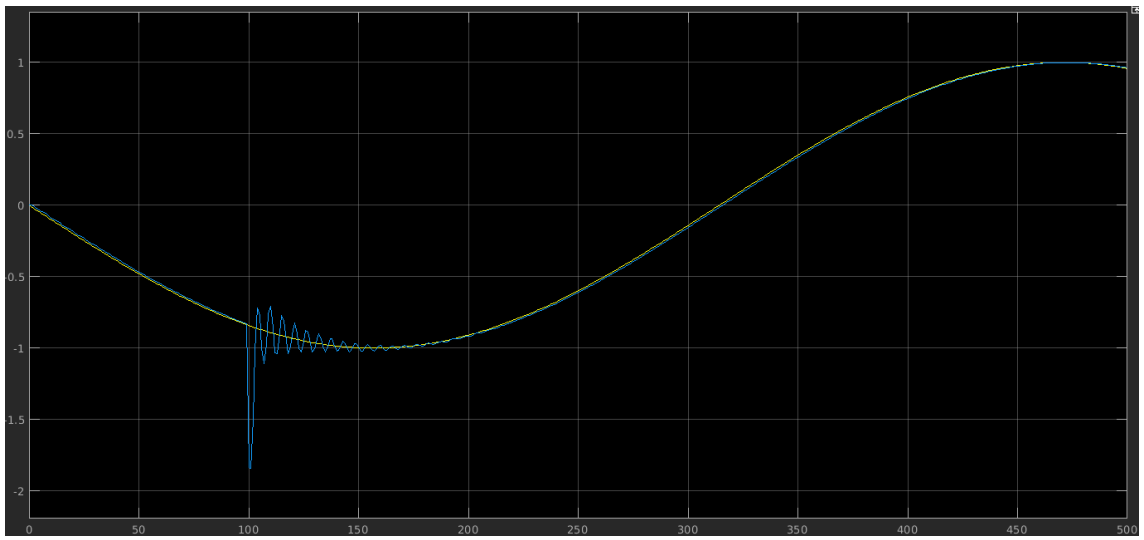


Figura 11: sine_9sugeno_wtaver_carga

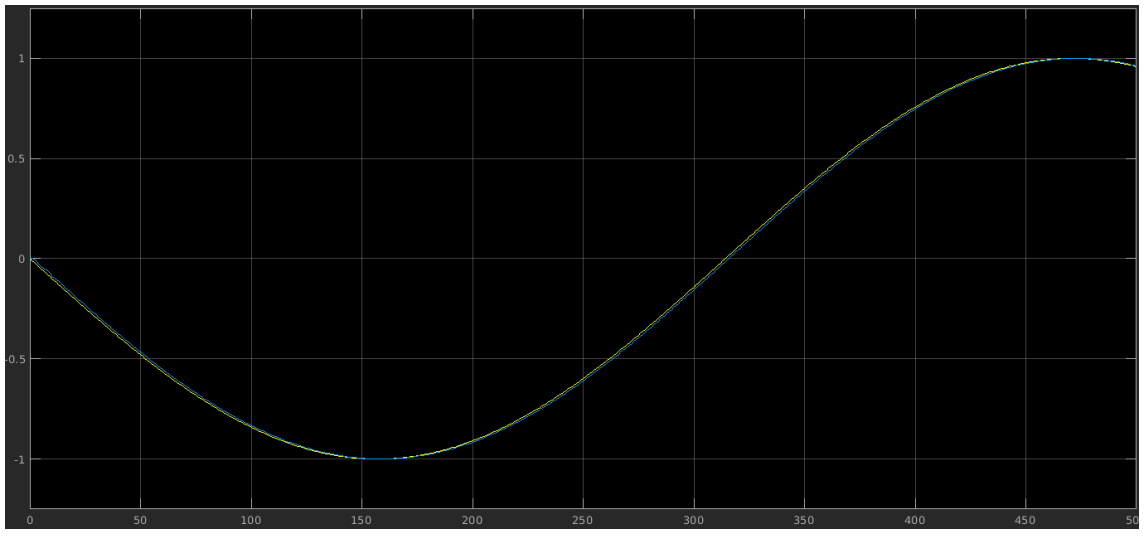


Figura 12: sine_9sugeno_wtaver_nothing

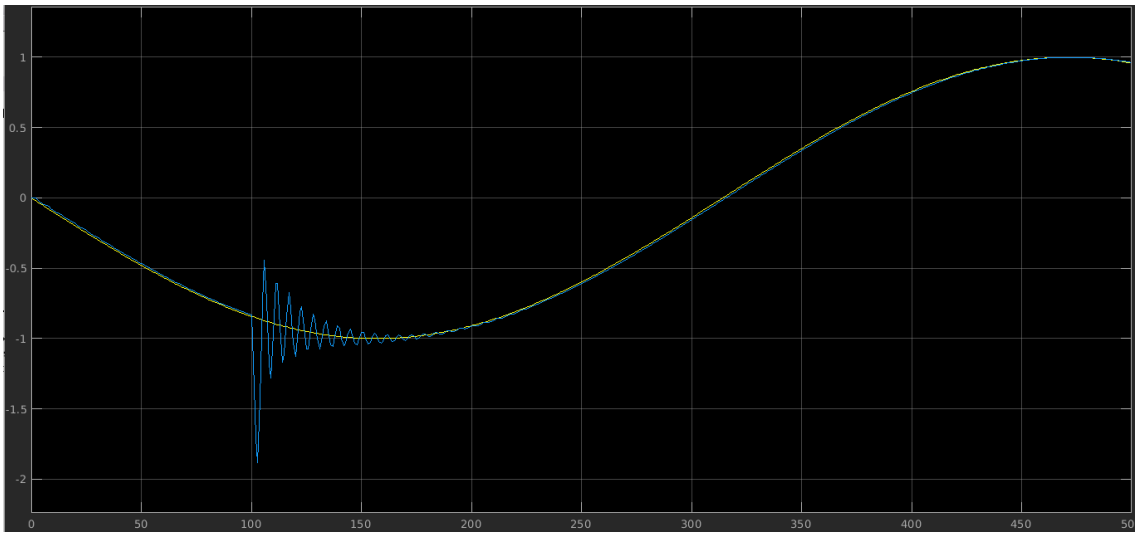


Figura 13: sine_9sugeno_wtsum_atuador

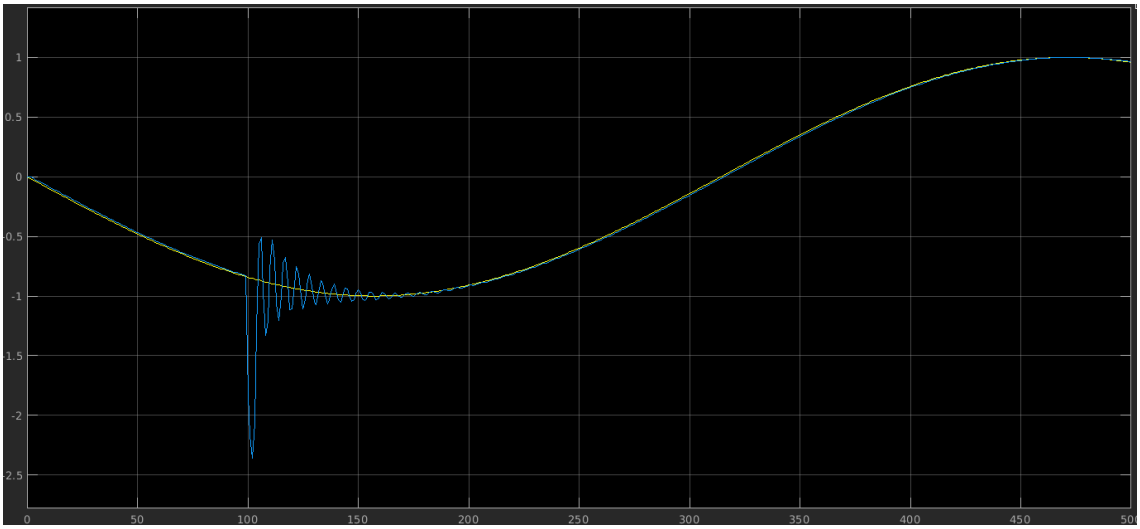


Figura 14: sine_9sugeno_wtsum_both

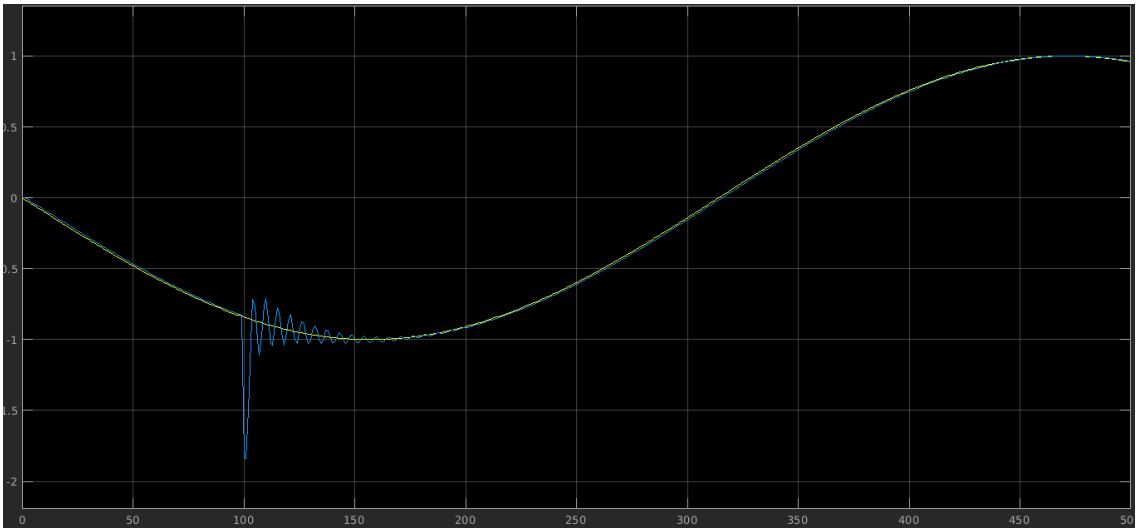


Figura 15: sine_9sugeno_wtsum_carga

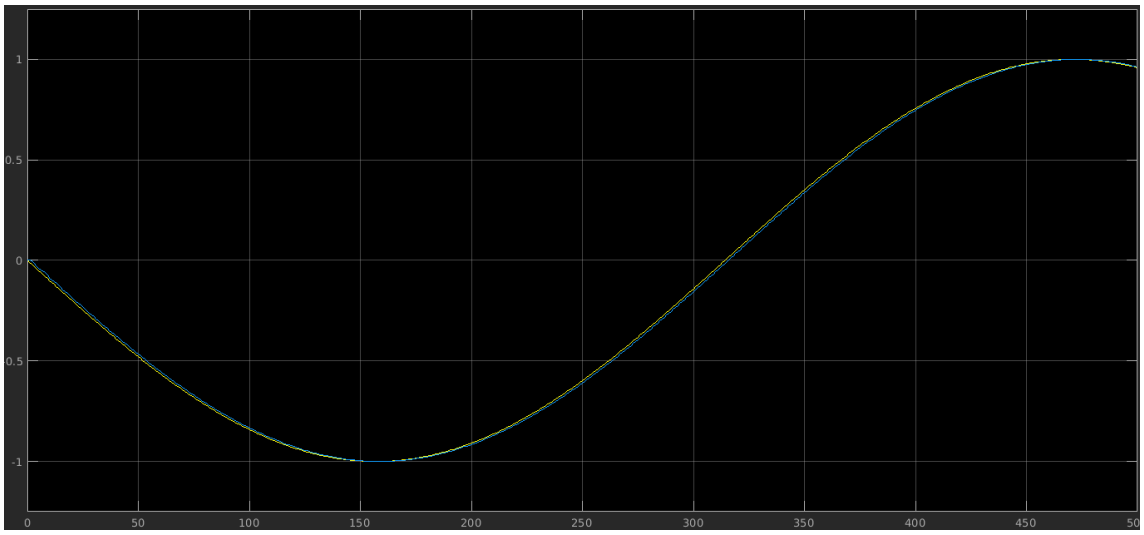


Figura 16: sine_9sugeno_wtsum_nothing

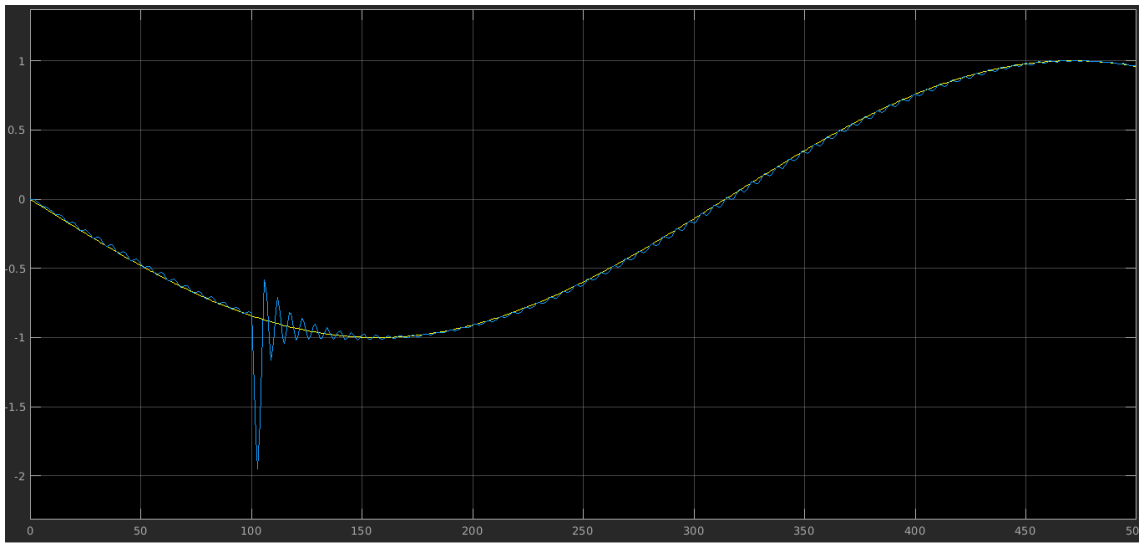


Figura 17: sine_25mamdani_centroid_atuador

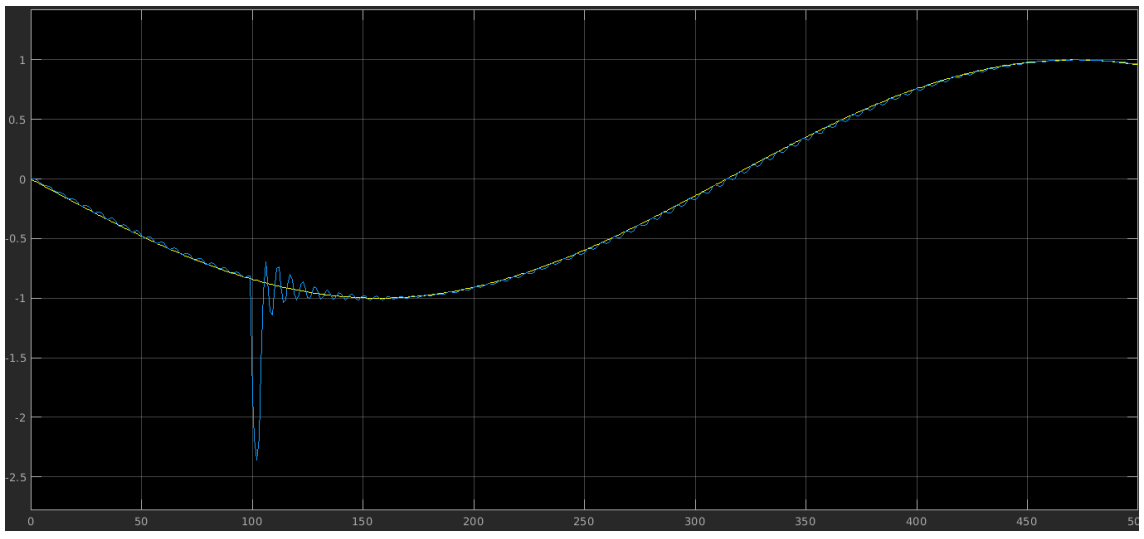


Figura 18: sine_25mamdani_centroid_atuador

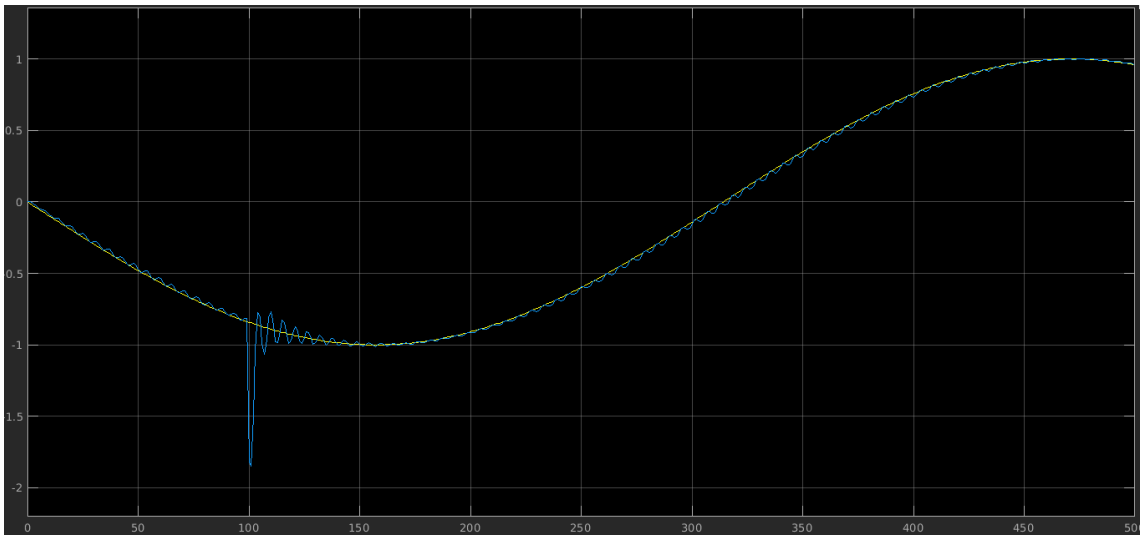


Figura 19: sine_25mamdani_centroid_carga

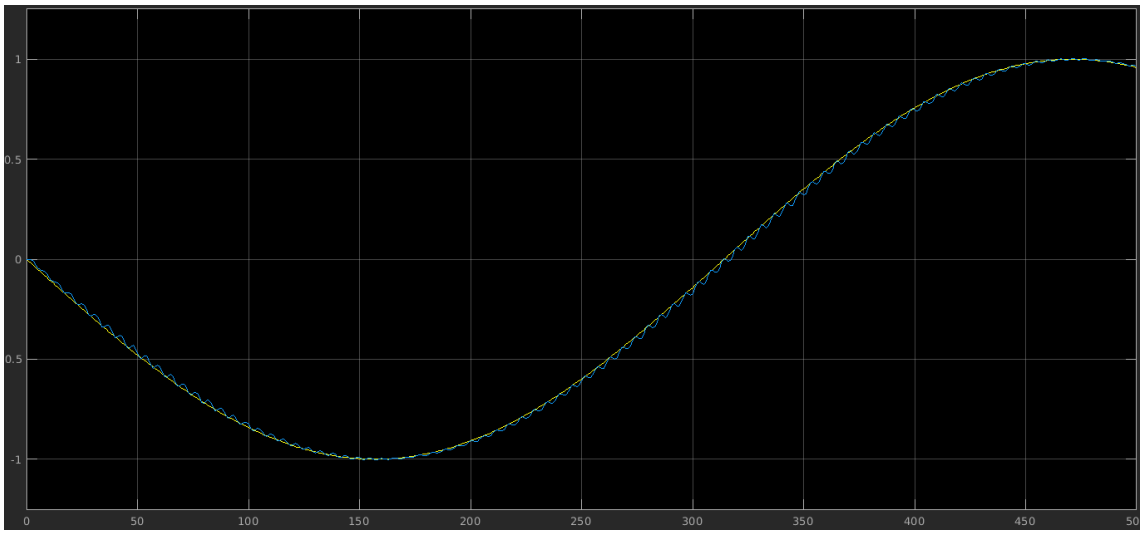


Figura 20: sine_25mamdani_centroid_nothing

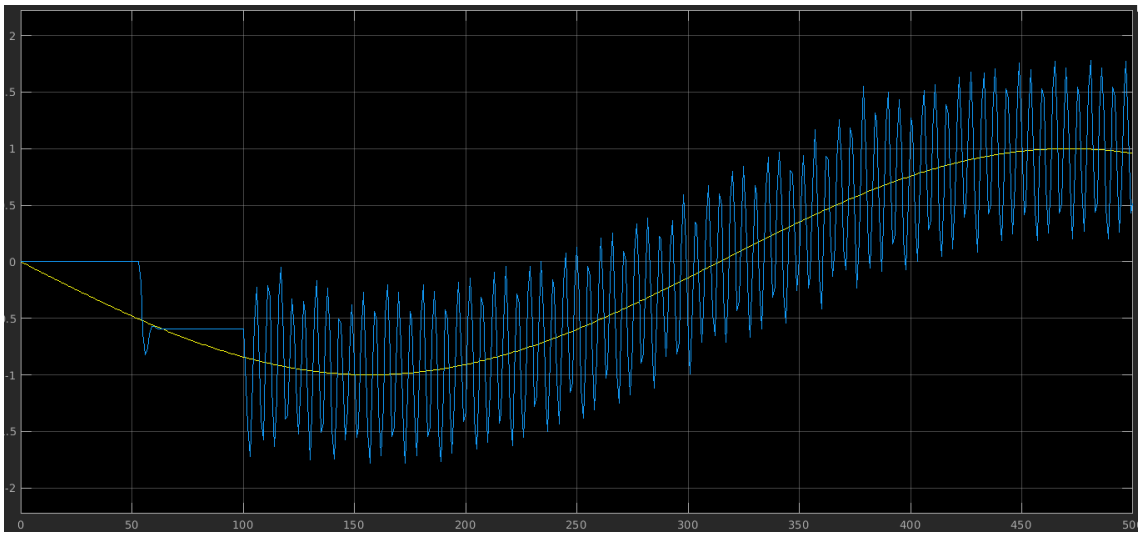


Figura 21: sine_25mamdani_mom_atuador

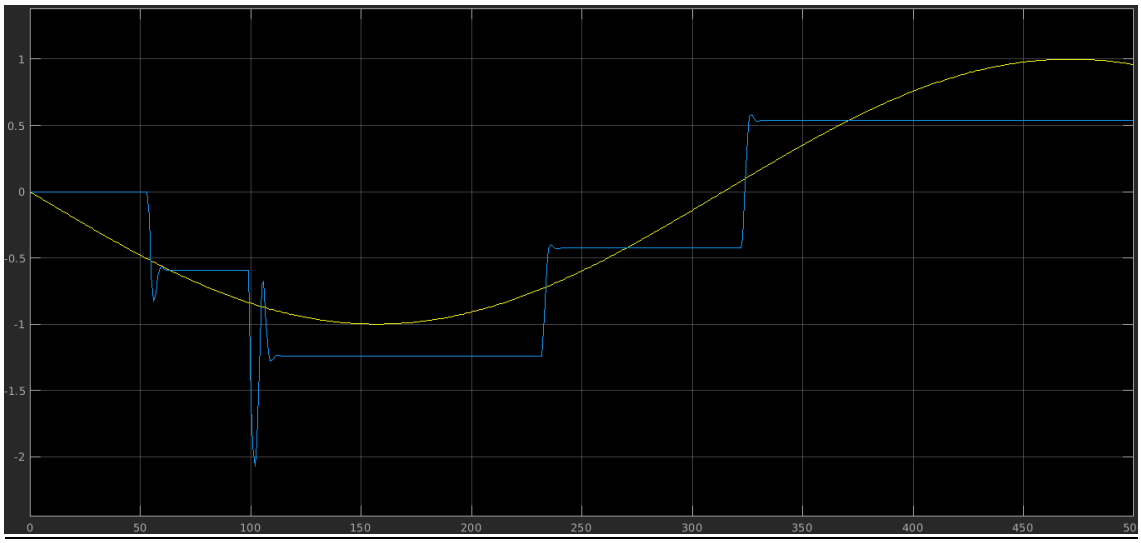


Figura 22: sine_25mamdani_mom_both

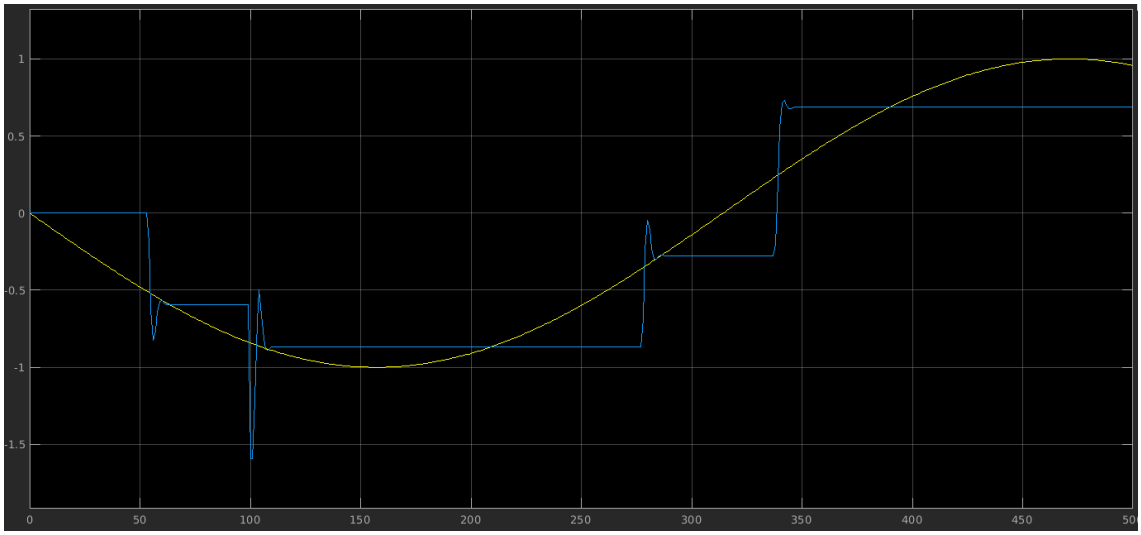


Figura 23: sine_25mamdani_mom_carga

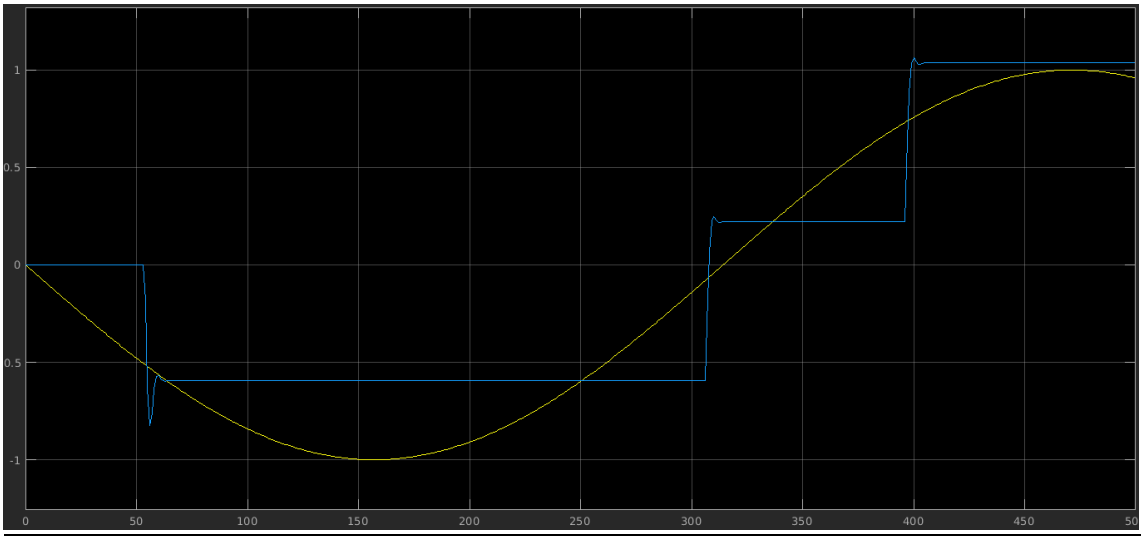


Figura 24: sine_25mamdani_mom_nothing

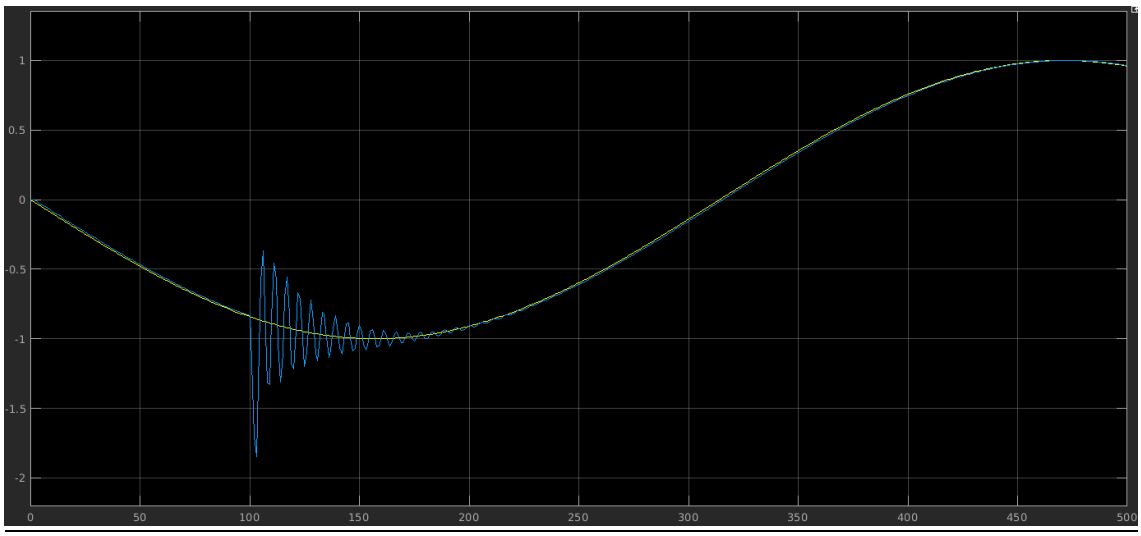


Figura 25: sine_25sugeno_wtaver_atuador

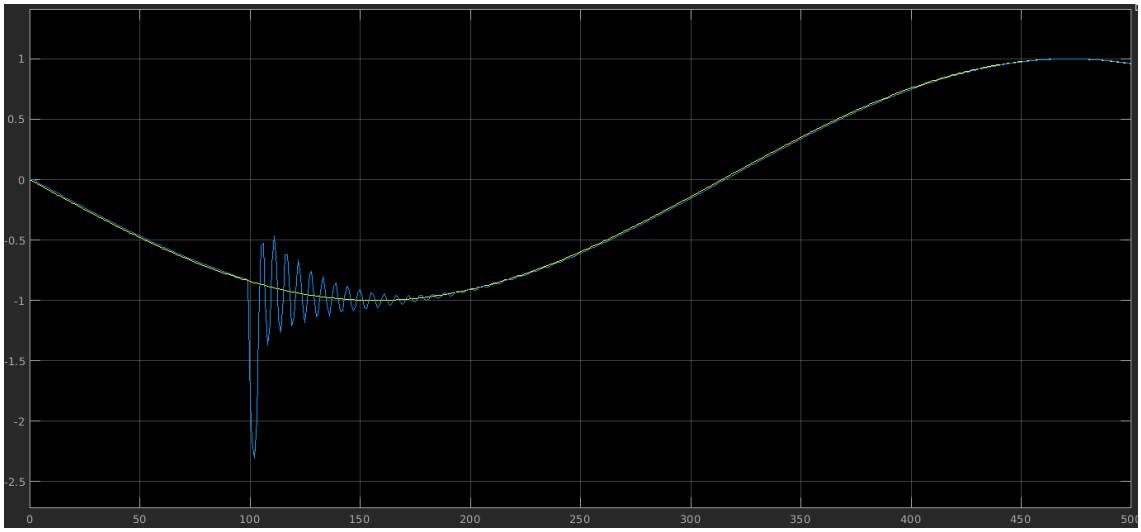


Figura 26: sine_25sugeno_wtaver_both

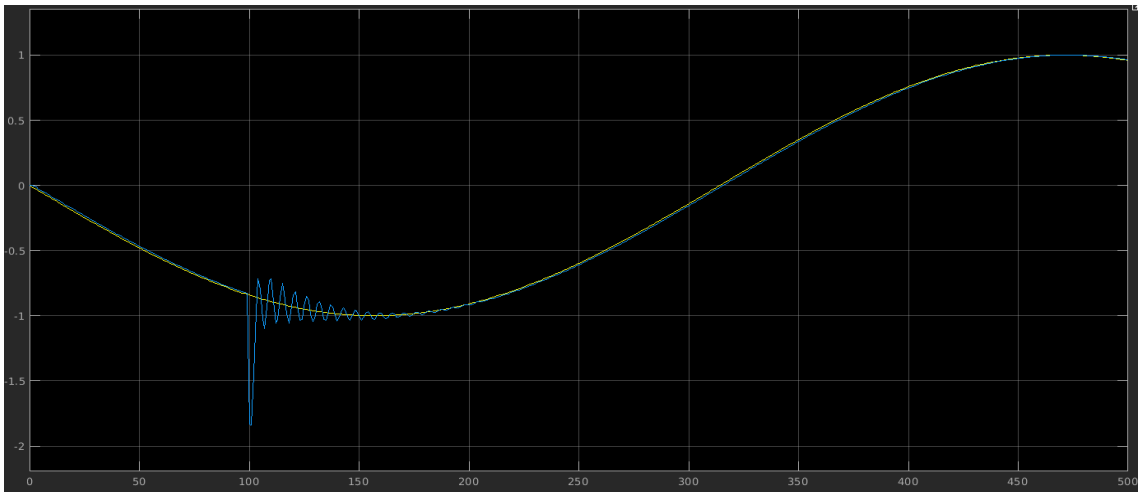


Figura 27: sine_25sugeno_wtaver_carga

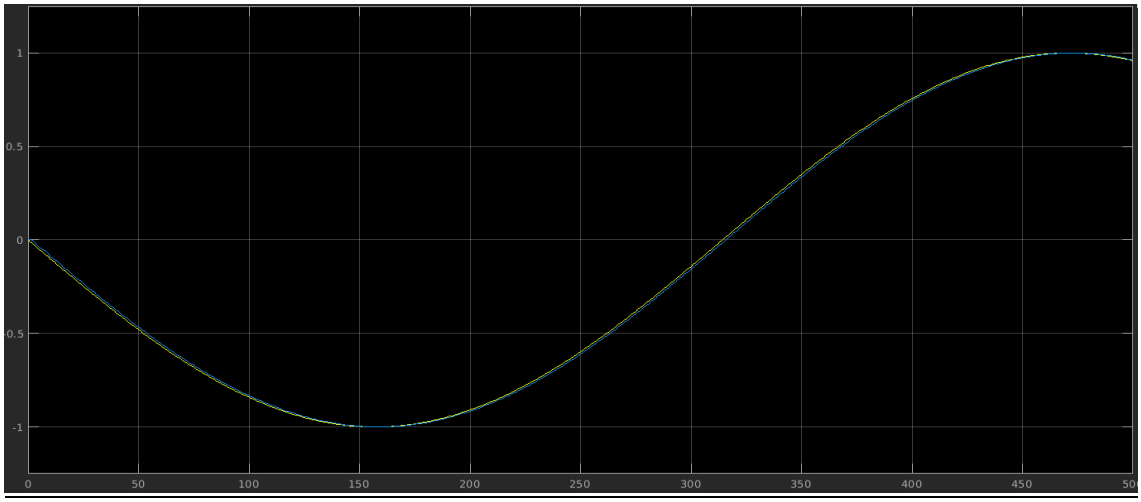


Figura28: sine_25sugeno_wtaver_nothing

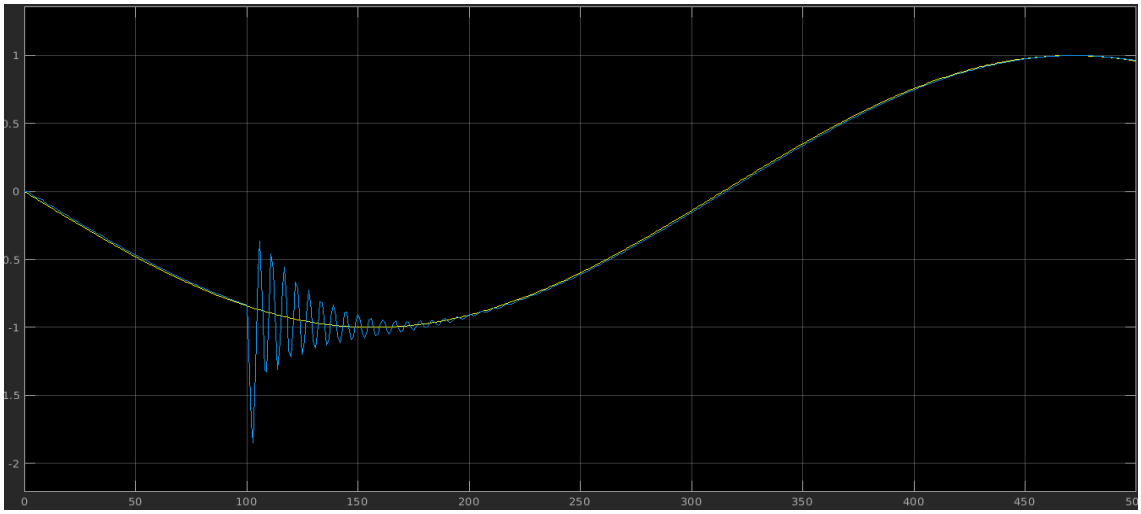


Figura 29: sine_25sugeno_wtsum_atuador

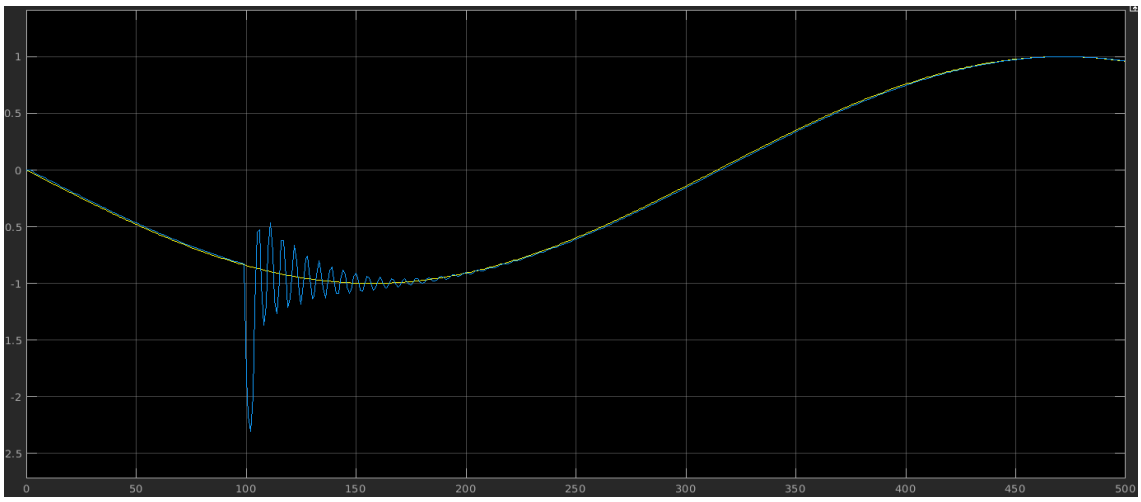


Figura 30: sine_25sugeno_wtsum_both

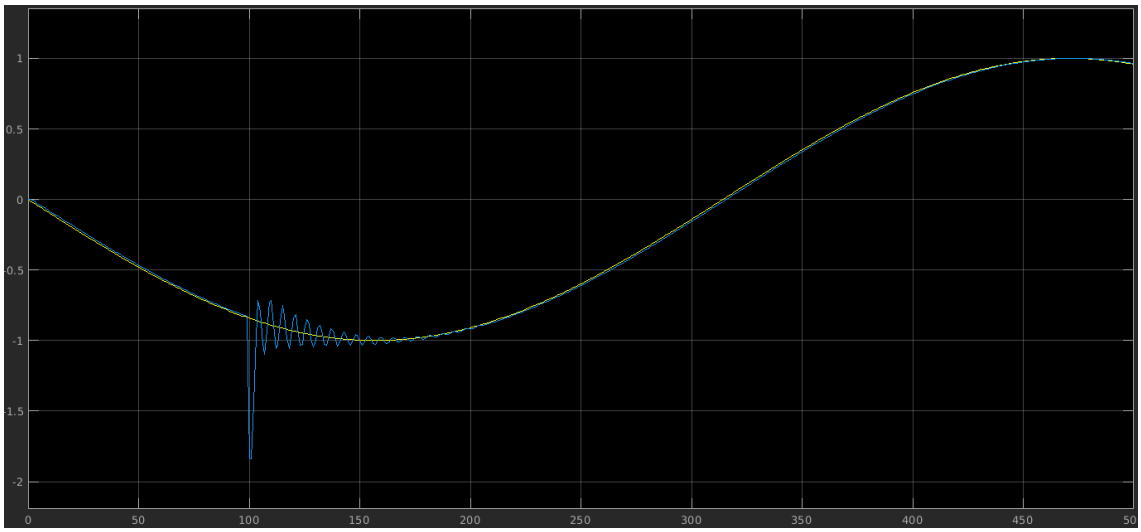


Figura 31: sine_25sugeno_wtsum_carga

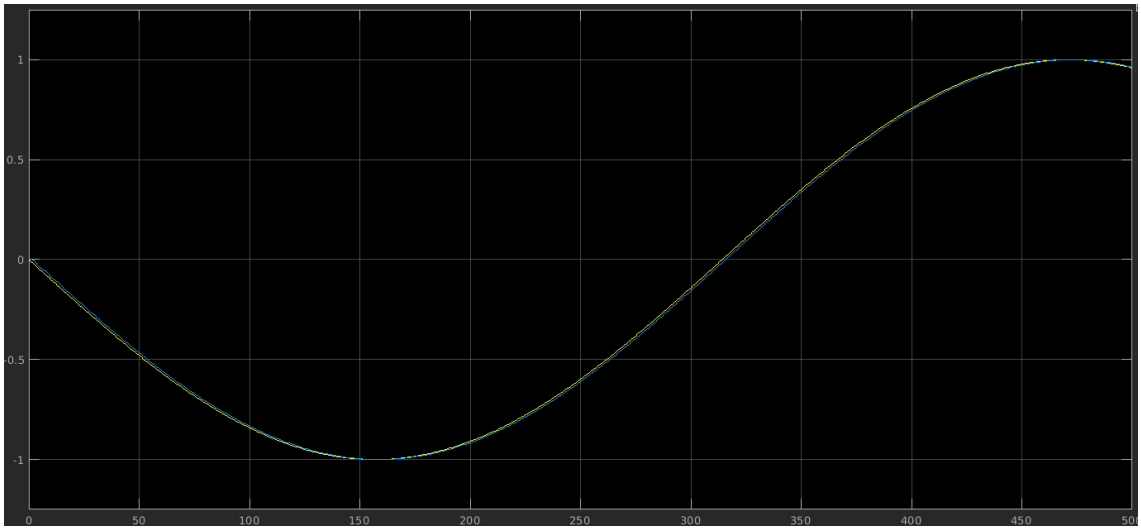


Figura 32: sine_25sugeno_wtsum_nothing

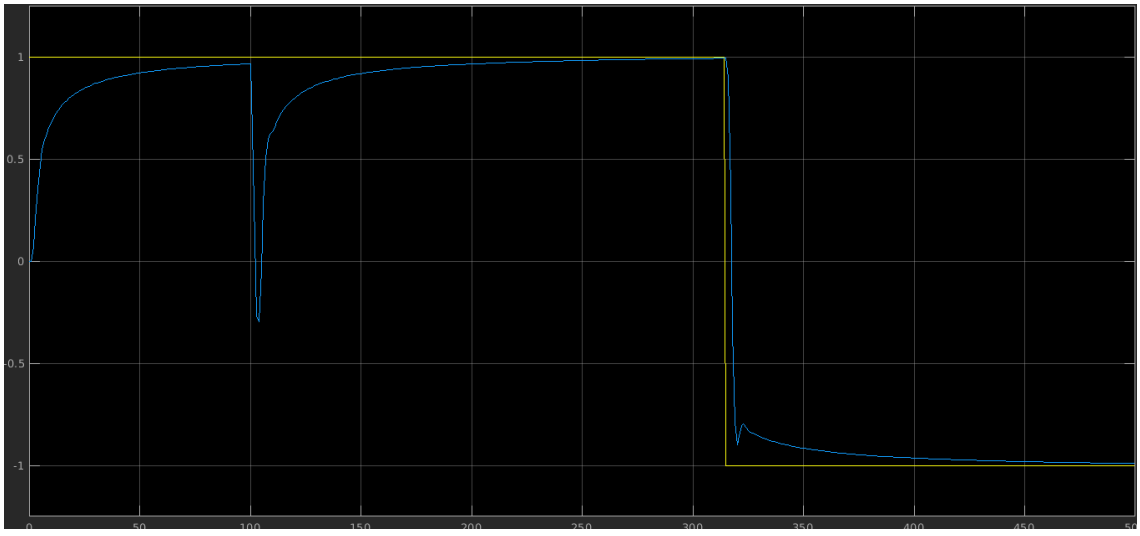


Figura 33: square_9mamdani_centroid_atuador

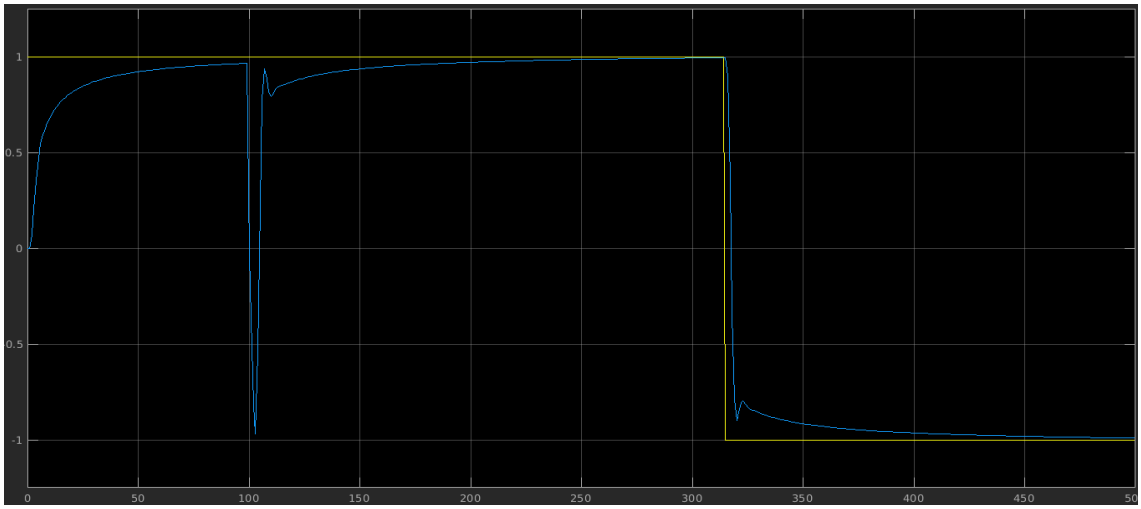


Figura 34: square_9mamdani_centroid_both

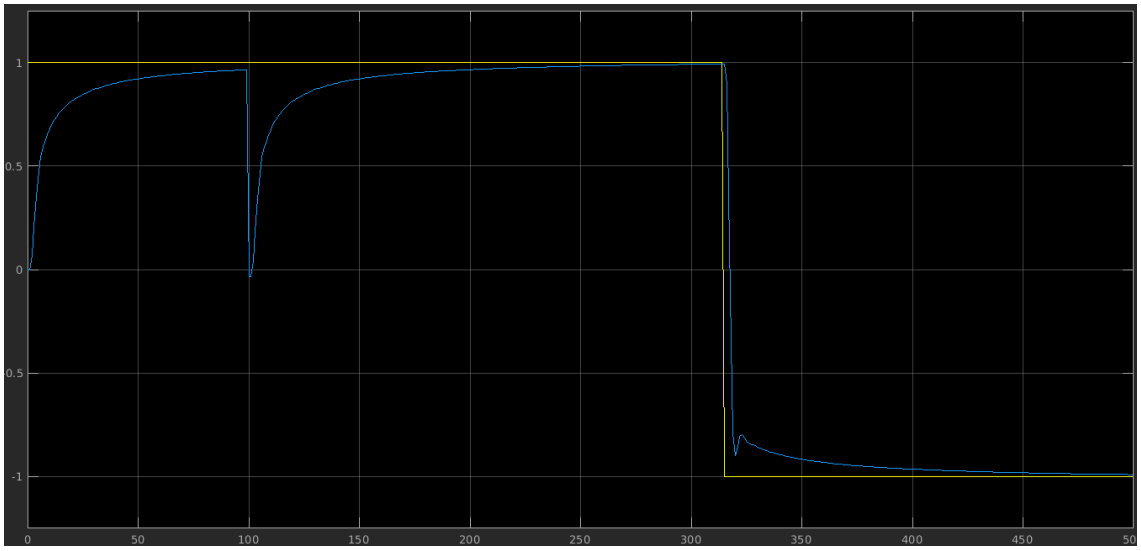


Figura 35: square_9mamdani_centroid_carga

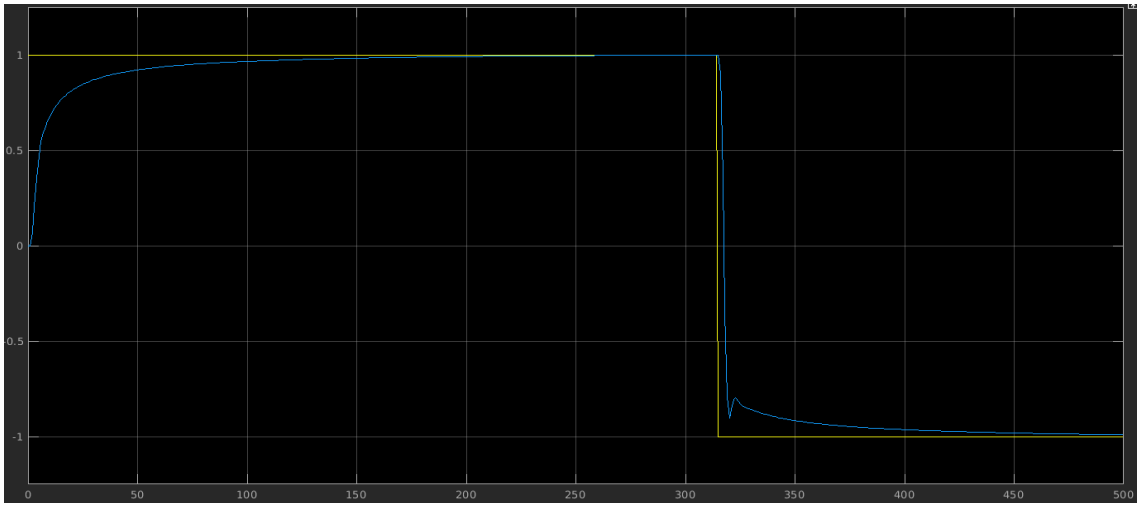


Figura 36: square_9mamdani_centroid_nothing

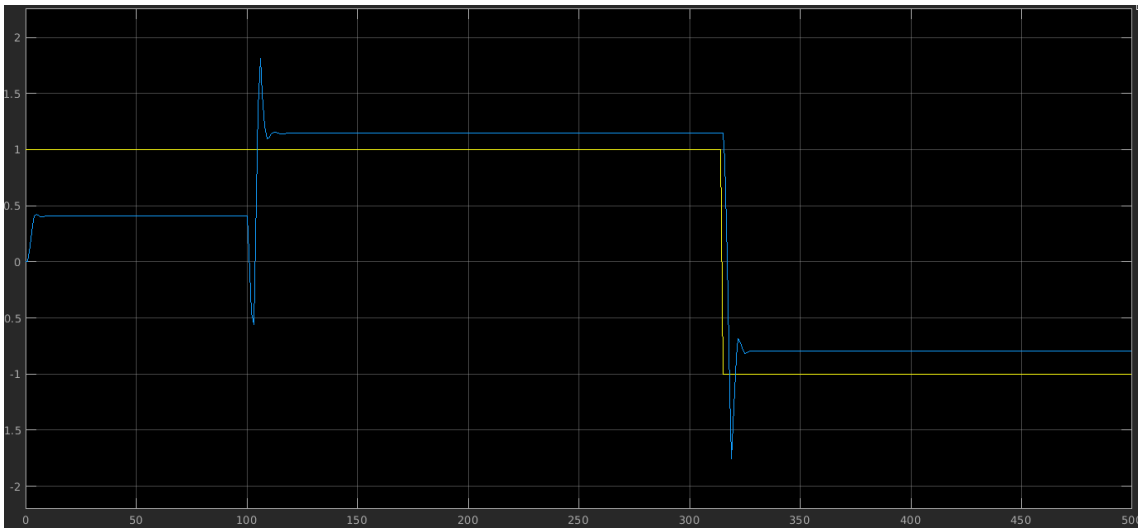


Figura 37: square_9mamdani_mom_atuador

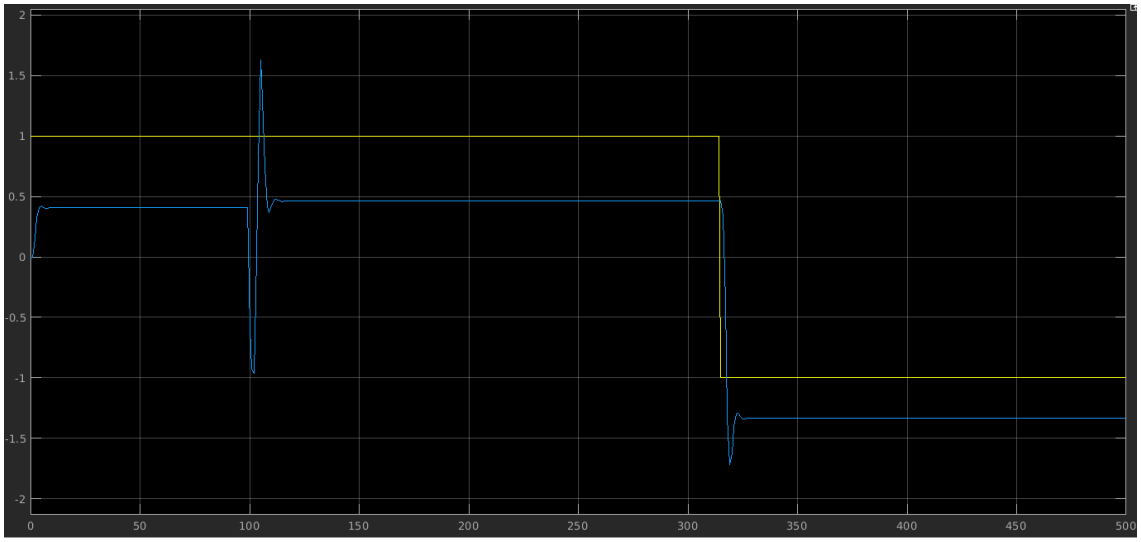


Figura 38: square_9mamdani_mom_both

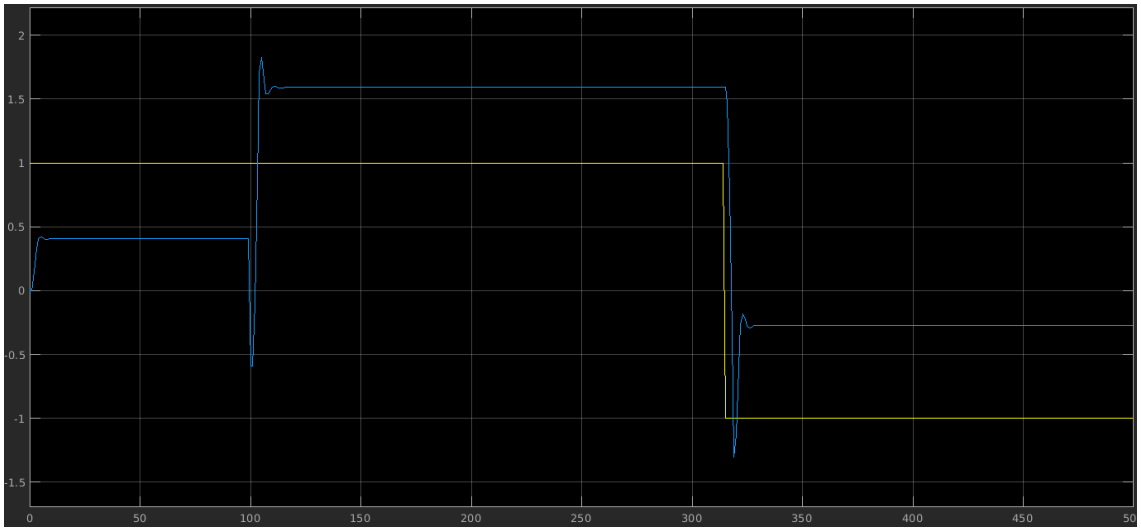


Figura 39: square_9mamdani_mom_carga

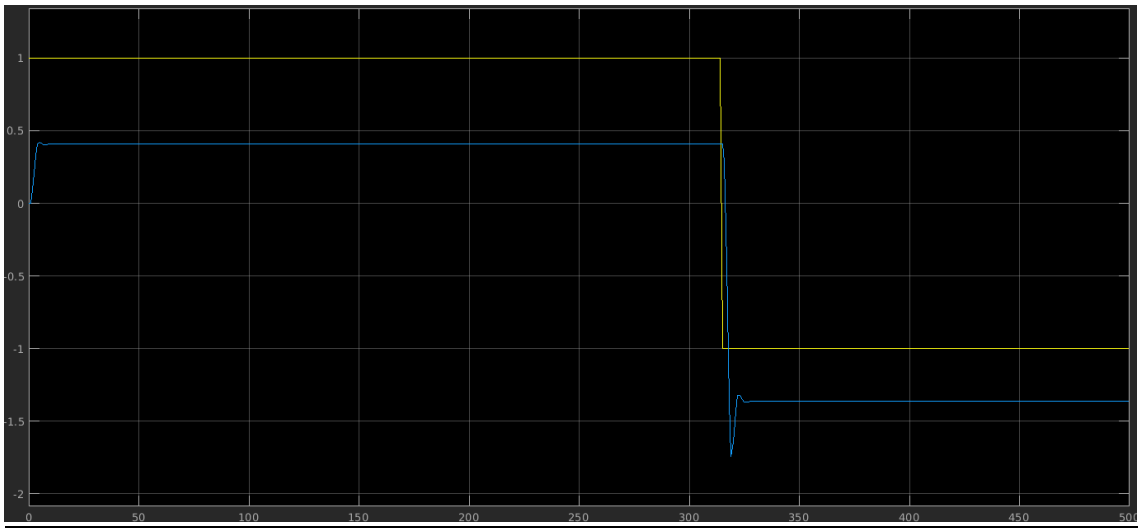


Figura 40: square_9mamdani_mom_nothing

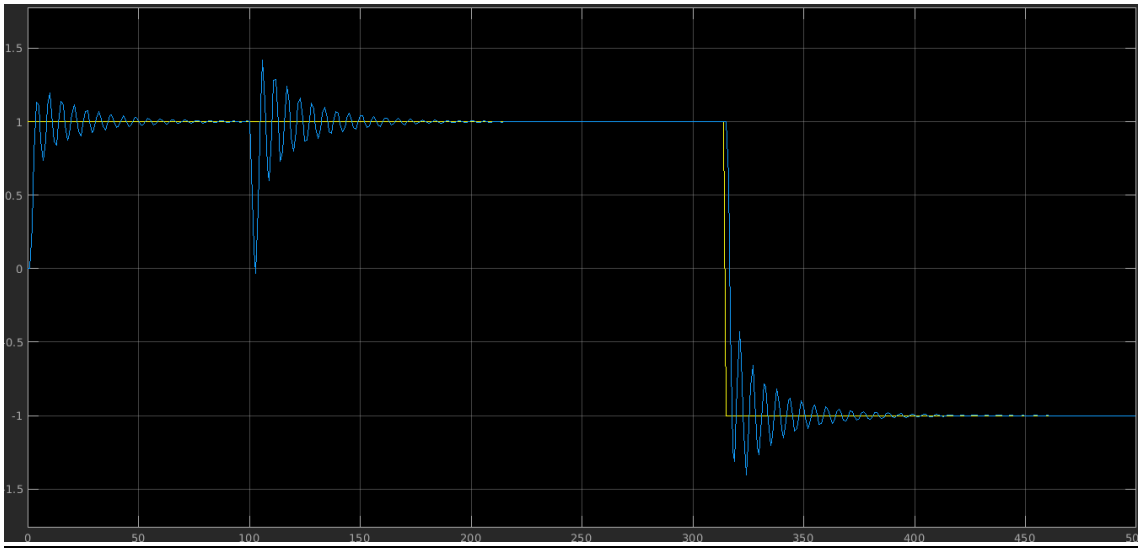


Figura 41: square_9sugeno_wtaver_atuador

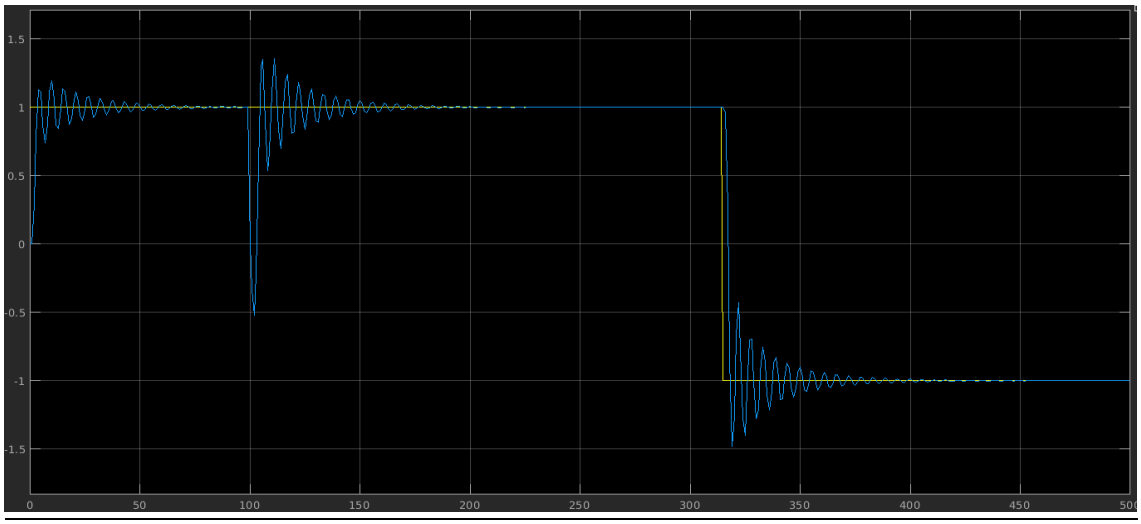


Figura 42: square_9sugeno_wtaver_both

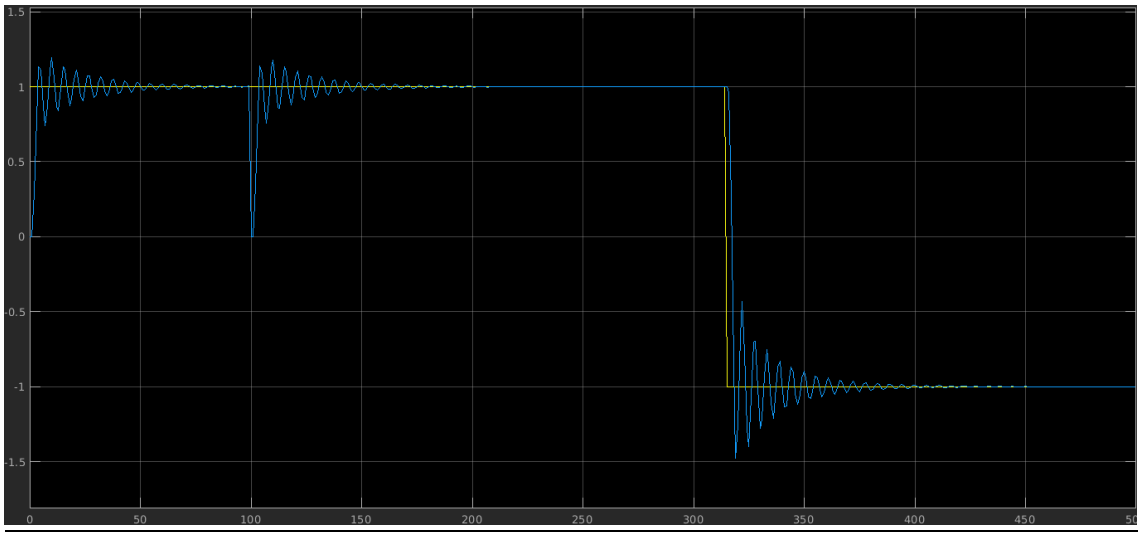


Figura 43: square_9sugeno_wtaver_carga

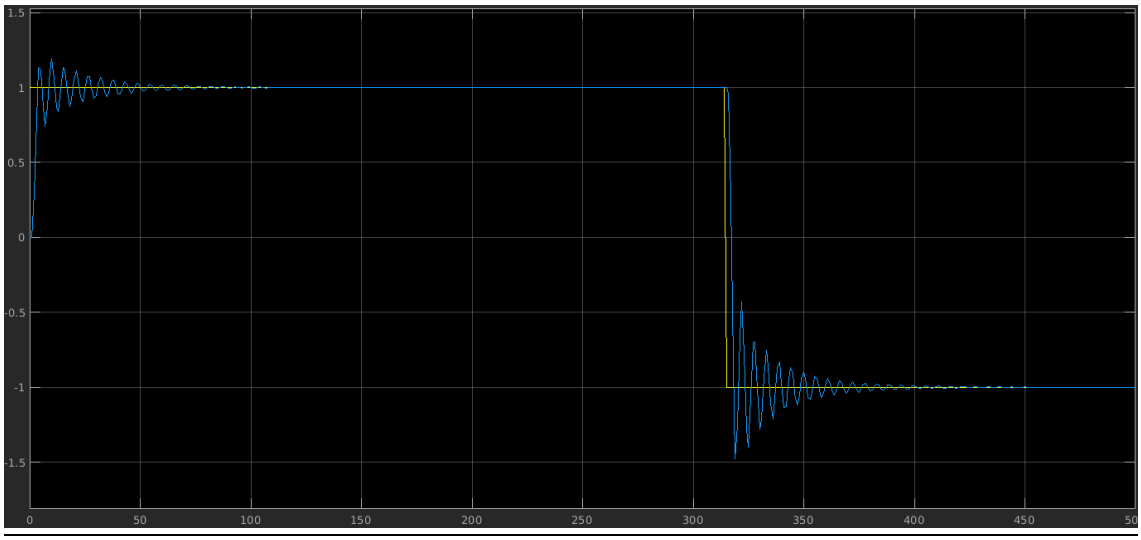


Figura 44: square_9sugeno_wtaver_nothing

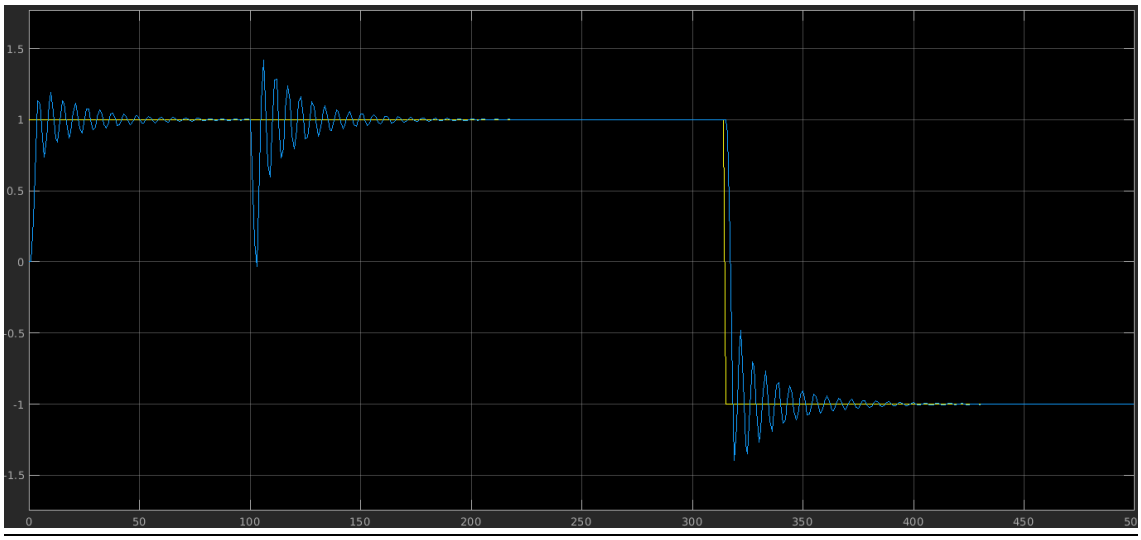


Figura 45: square_9sugeno_wtsum_atuador

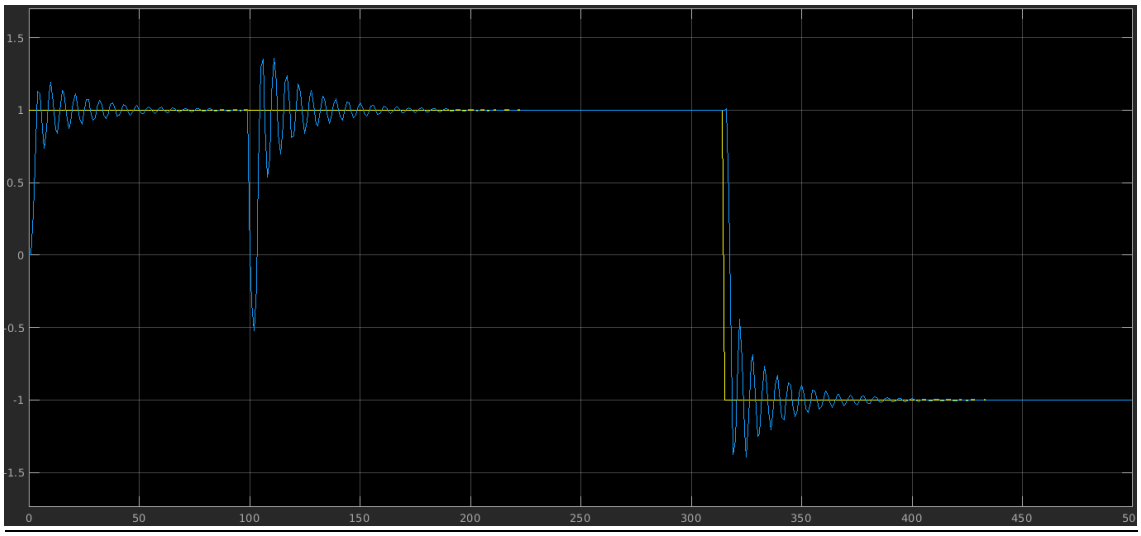


Figura 46: square_9sugeno_wtsum_both

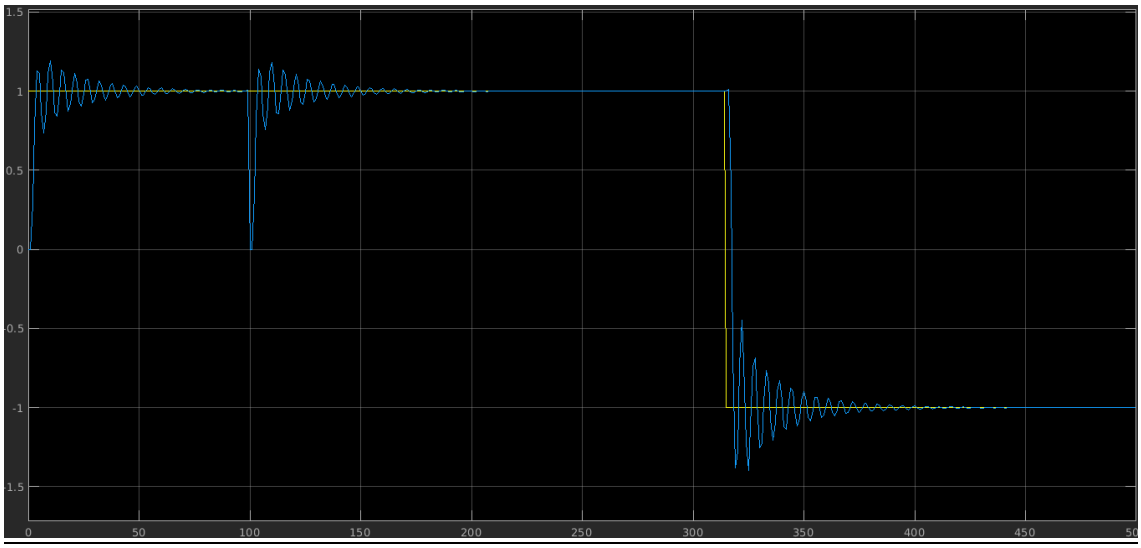


Figura 47: square_9sugeno_wtsum_carga

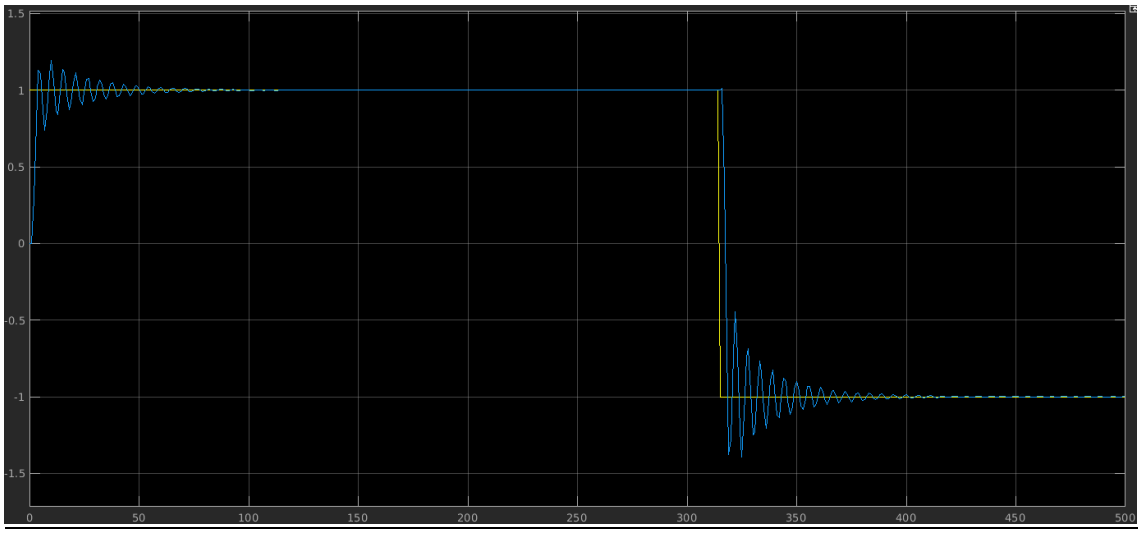


Figura 48: square_9sugeno_wtsum_nothing

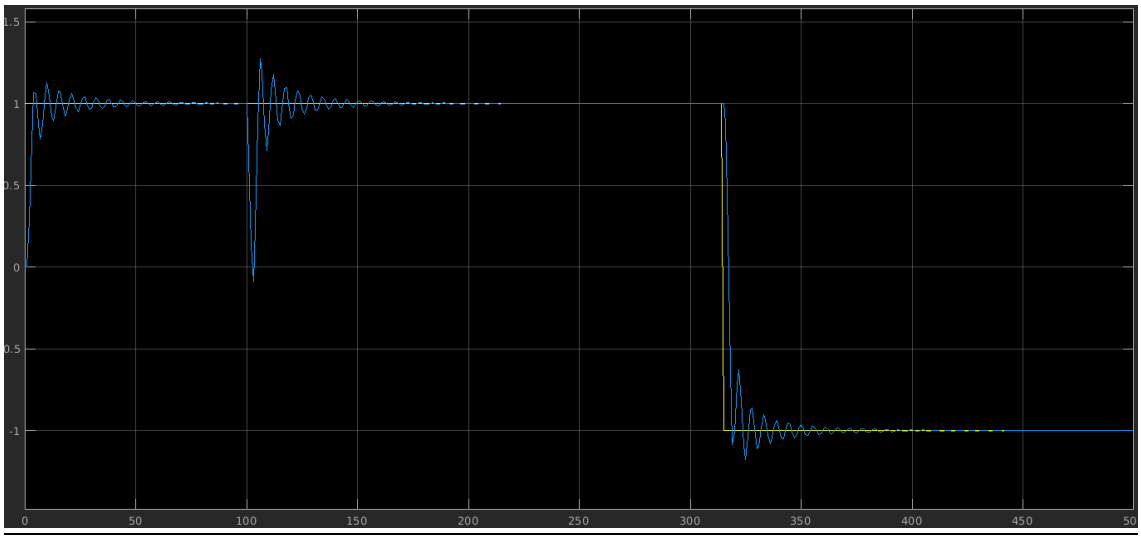


Figura 49: square_25mamdani_centroid_atuador

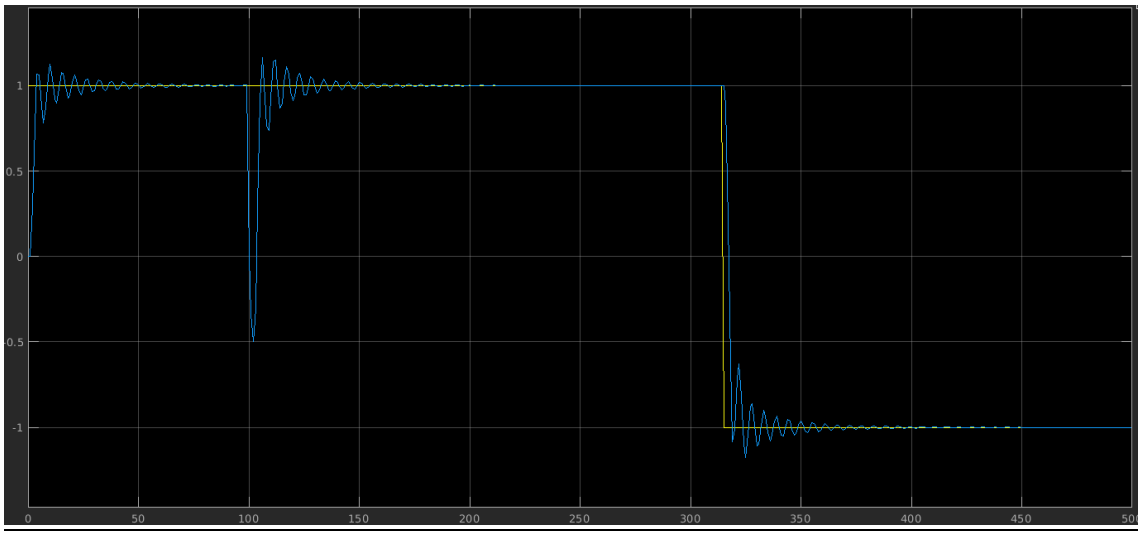


Figura 50: square_25mamdani_centroid_both

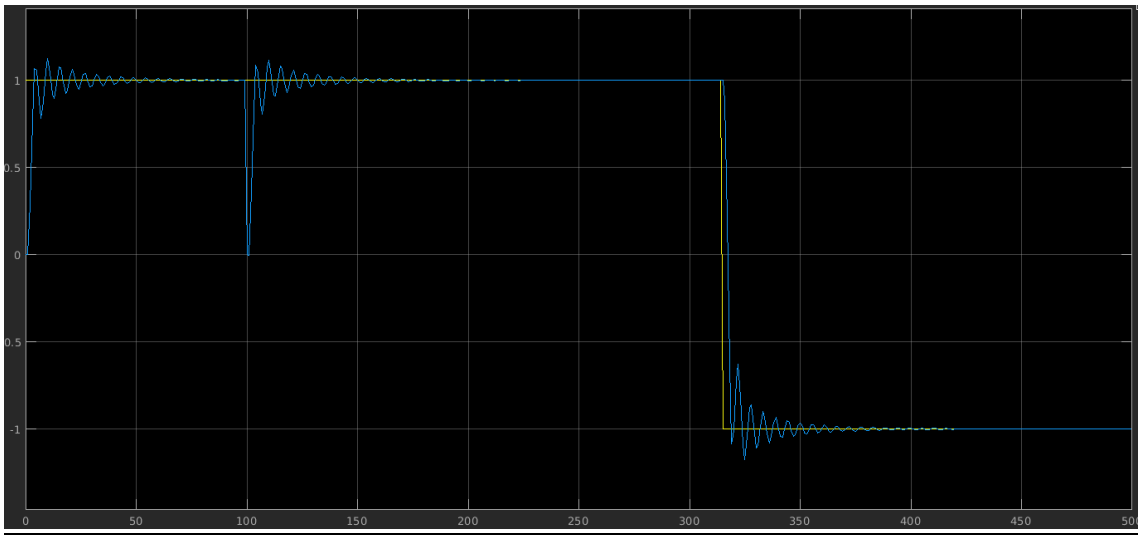


Figura 51: square_25mamdani_centroid_carga

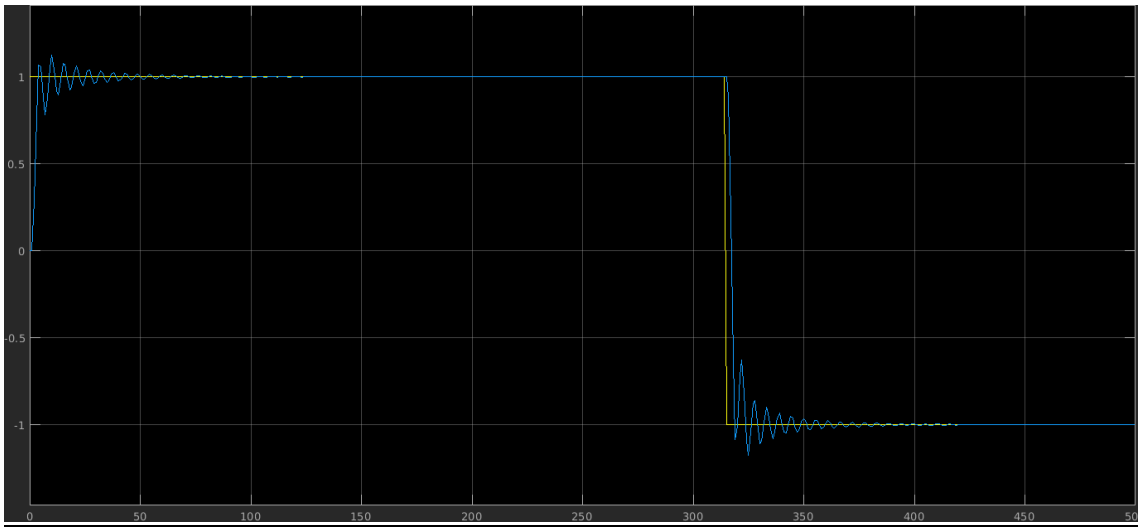


Figura 52: square_25mamdani_centroid_nothing

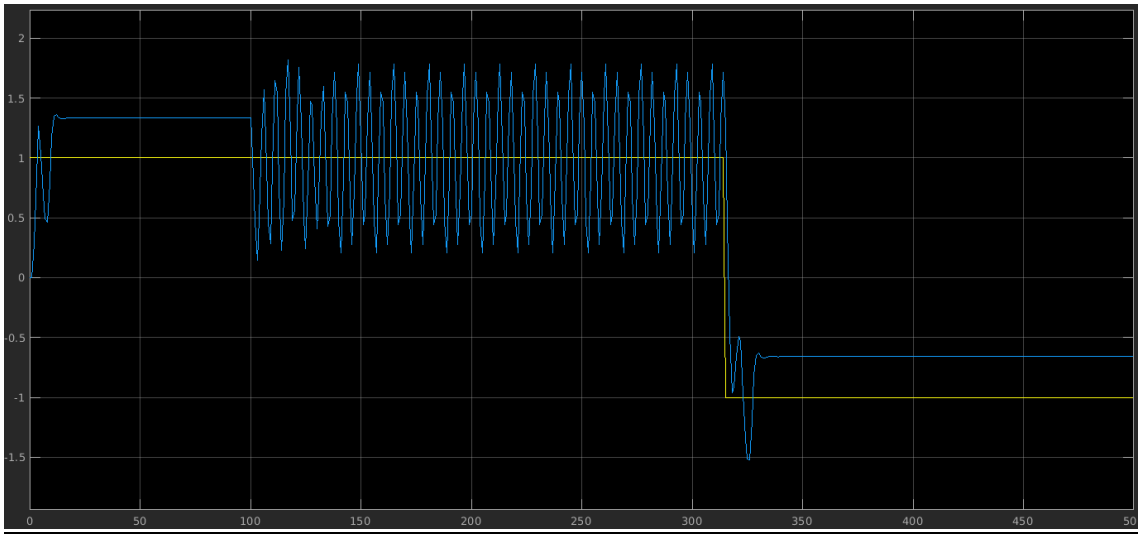


Figura 53: square_25mamdani_mom_atuador

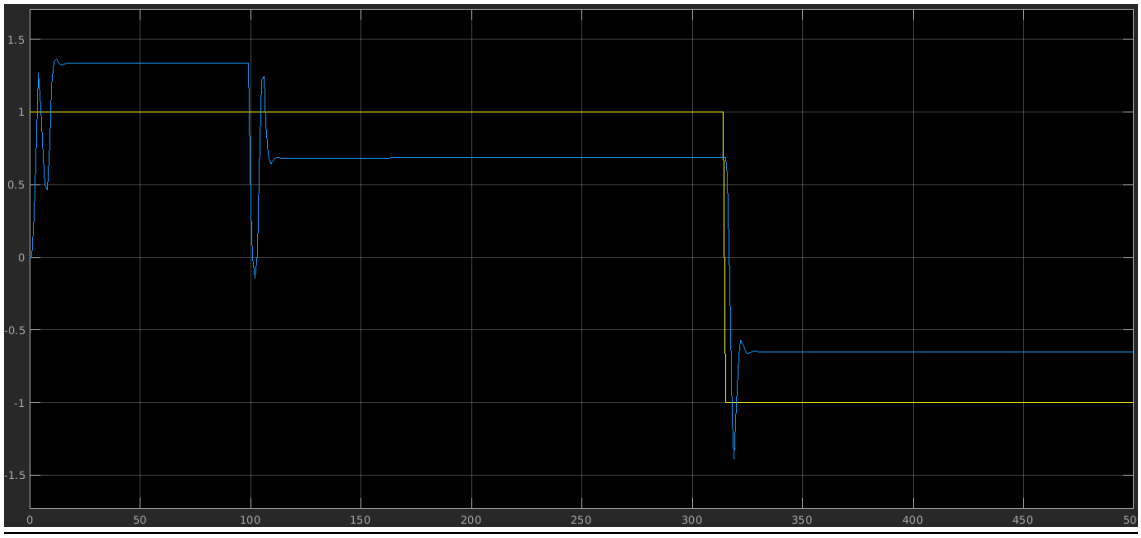


Figura 54: square_25mamdani_mom_both

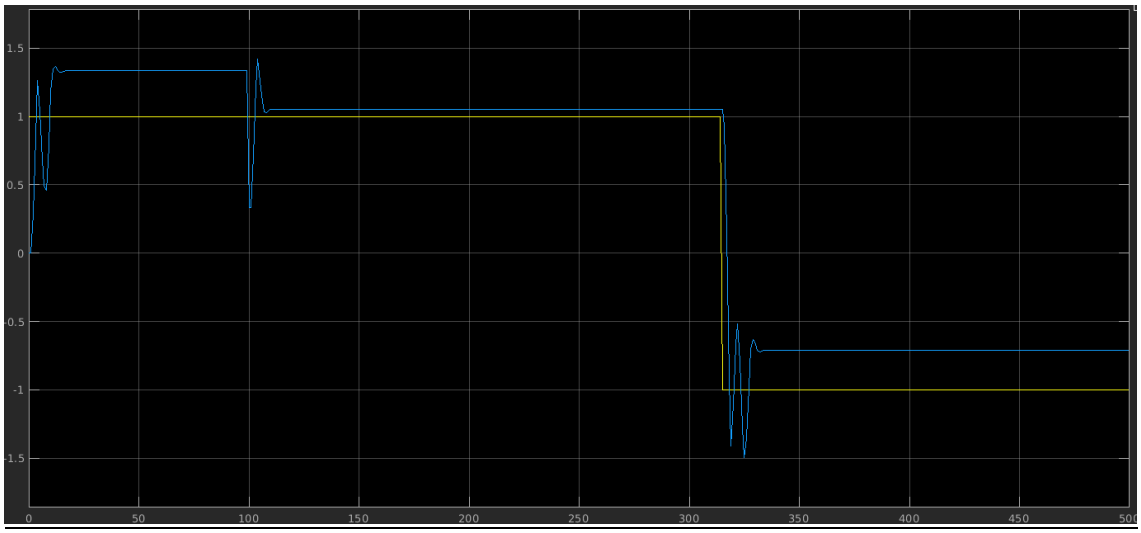


Figura 55: square_25mamdani_mom_carga

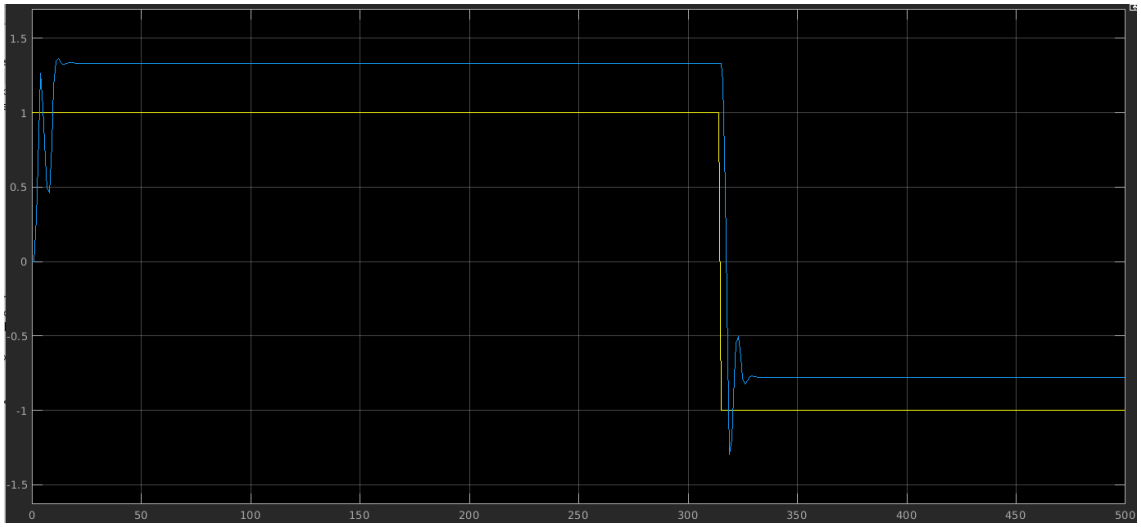


Figura 56: square_25mamdani_mom_nothing

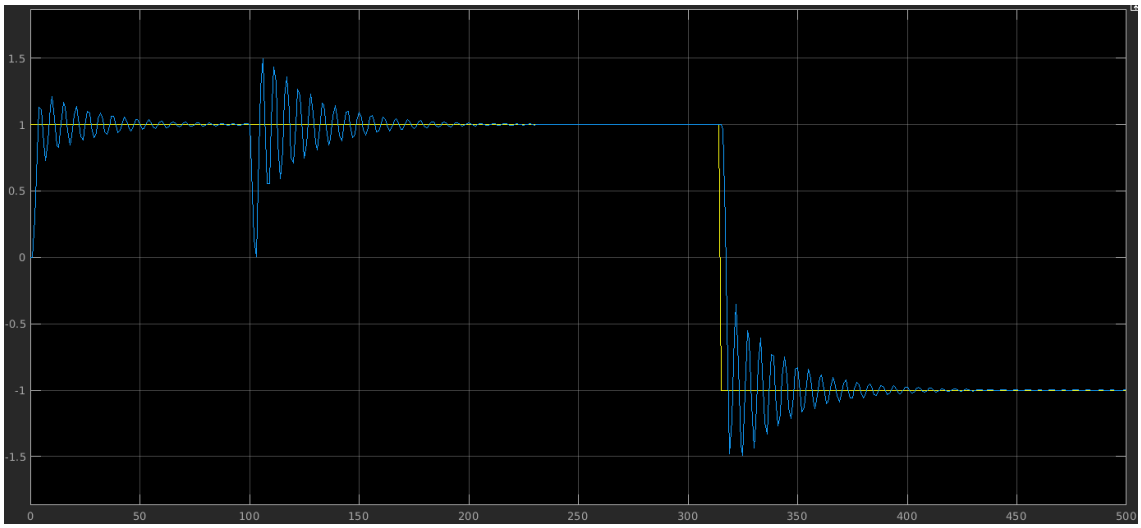


Figura 57: square_25sugeno_wtaver_atuador

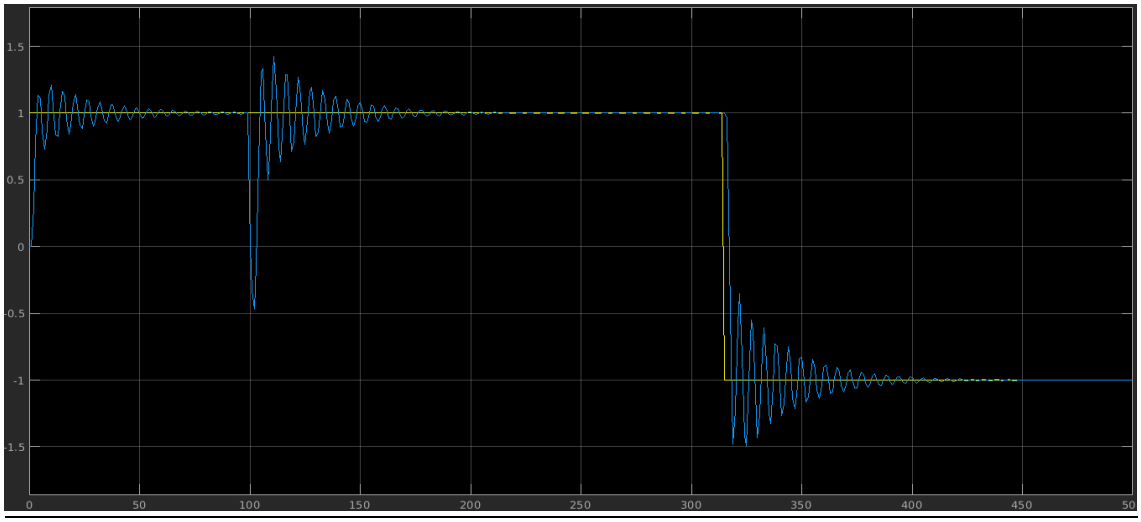


Figura 58: square_25sugeno_wtaver_both

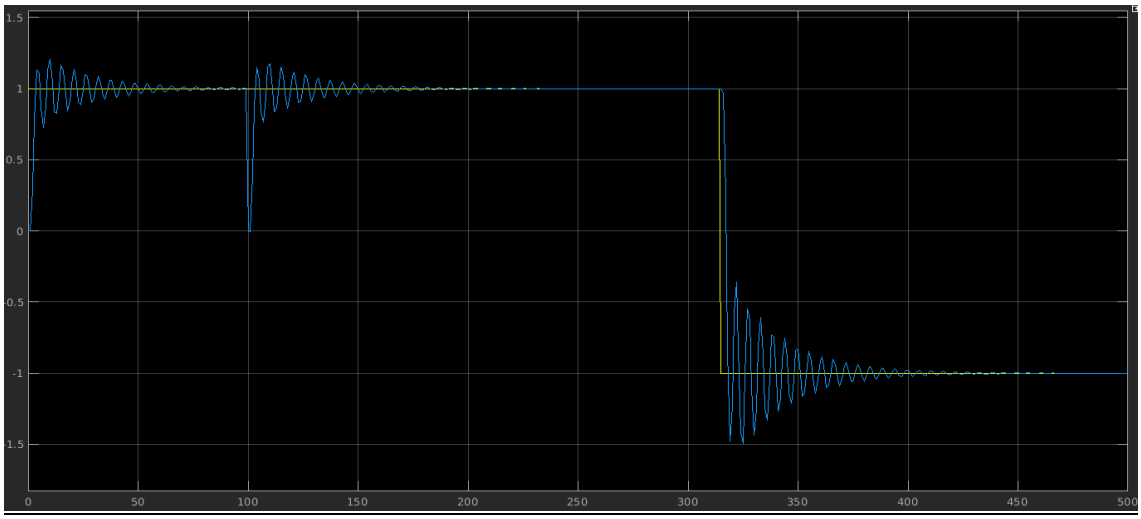


Figura 59: square_25sugeno_wtaver_carga

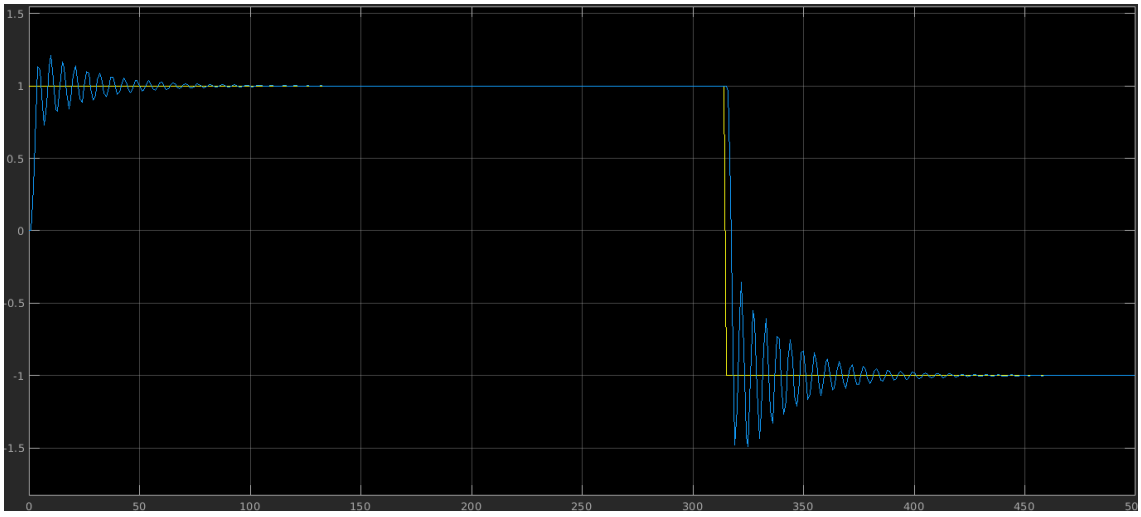


Figura 60: square_25sugeno_wtaver_nothing

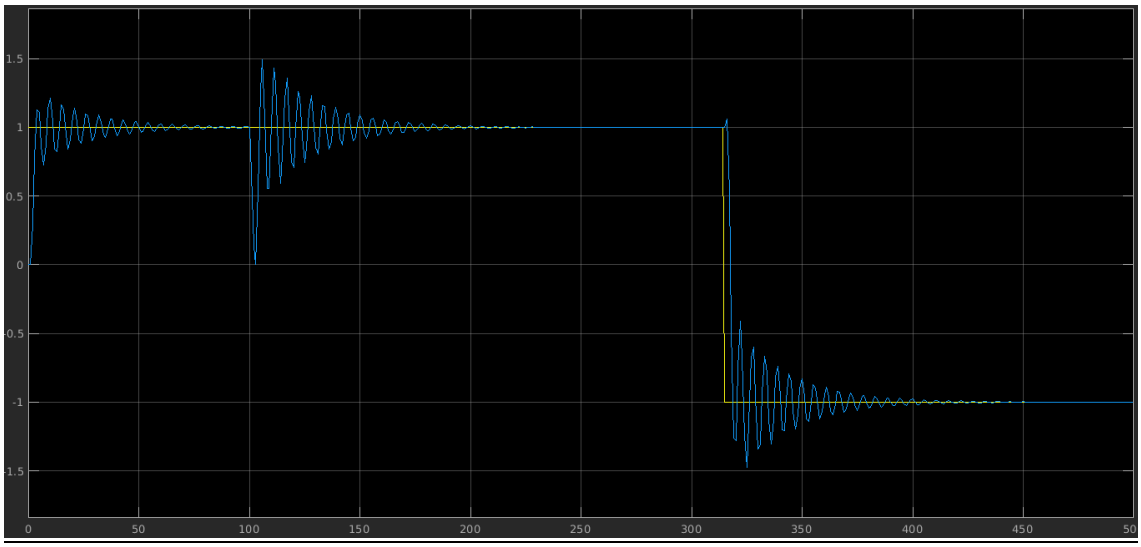


Figura 61: square_25sugeno_wtsum_atuador

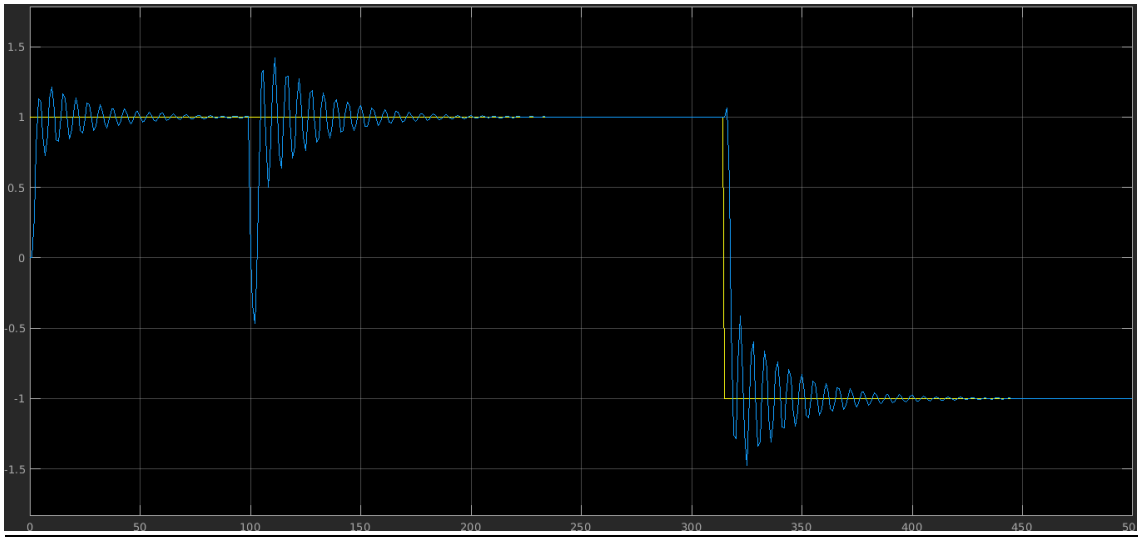


Figura 62: square_25sugeno_wtsum_both

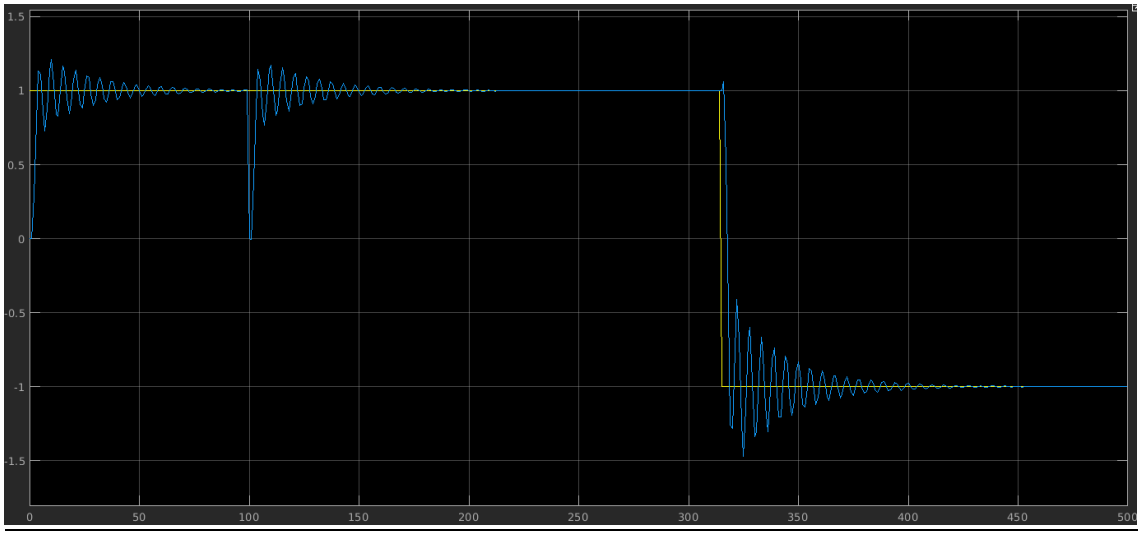


Figura 63: square_25sugeno_wtsum_both

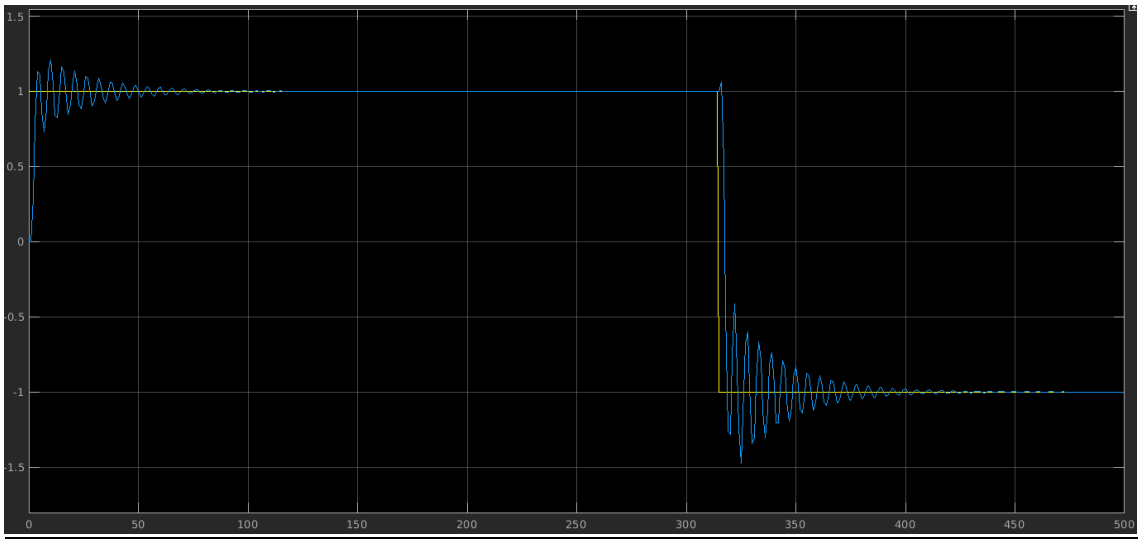


Figura 64: square_25sugeno_wtsum_nothing