

*Controlo Inteligente*

Realizado por:

Daniel Sousa, nº52727

João Carvalho, nº 49341

2º Trabalho

Redes neuronais na identificação e controlo de um processo térmico

# Índice

[Índice 3](#_Toc56351571)

[Introdução Teórica 5](#_Toc56351572)

[Aquisição dos dados 9](#_Toc56351573)

[Treino da rede do sistema 9](#_Toc56351574)

[Comparação com o modelo ARX obtido no laboratório passado 11](#_Toc56351575)

[Simulação 13](#_Toc56351576)

[Controlo do Processo 16](#_Toc56351577)

[Outra Rede Neuronal para o controlador 18](#_Toc56351578)

[Conclusão 21](#_Toc56351579)

[Anexos 23](#_Toc56351580)

[Bibliografia utilizada 33](#_Toc56351581)

# Introdução Teórica

As redes neuronais têm algumas parecenças com os neurónios que existem no sistema nervoso humano. O neurónio “artificial” é composto por *i* entradas, (ou potenciais de ação), a que estão associados pesos (ou coesão das ligações sinápticas) A função ativação, ,do neurónio devolve a sua resposta, tendo como parâmetro de entrada a soma ponderada das entradas e a polaridade, *b*. Alterando estes valores, esta arquitetura permite-nos modelar sistemas não lineares e têm a capacidade de processar informação em paralelo. A figura abaixo ilustra a estrutura de um neurónio.

Uma imagem com objeto, relógio

Descrição gerada automaticamente

Figura - Neurónio artificial

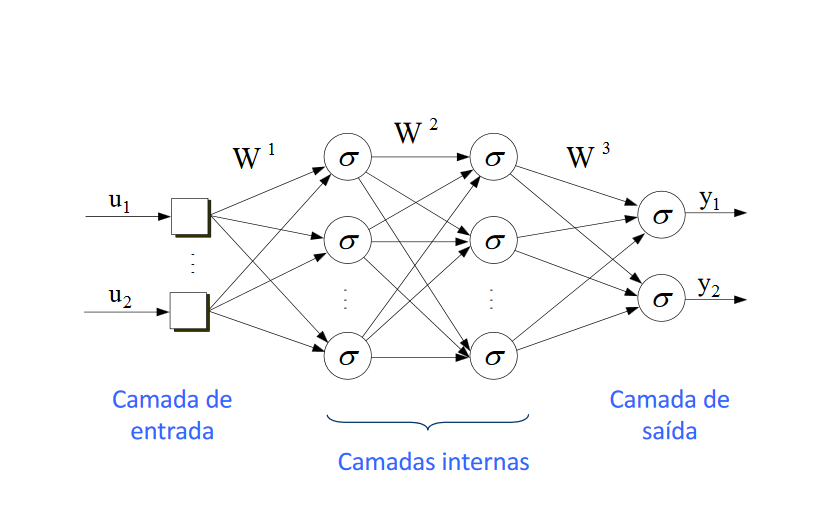
 Uma rede neuronal é formada por vários neurónios, tendo uma camada de entrada e outra de saída:

Figura - Rede Neuronal

 De forma a incorporar informação temporal na rede neuronal, podemos utilizar o vetor de regressão associado ao modelo NARX (*Nonlinear AutoRegressive with eXogenous input*). Desta forma a saída da rede neuronal vai ser uma função que tem como argumento o vetor de regressão.

Os parâmetros da rede neuronal são treinados de forma a minimizar o erro entre a saída da rede e a saída do processo a modelar.

As arquiteturas de controlo neuronal utilizadas neste trabalho foram:

* Controlo Neuronal Inverso Direto;
* Controlo por modelo interno (*IMC*);

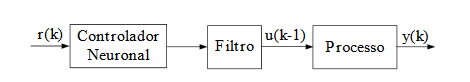


Figura - Modelo do controlador neuronal Inverso Direto

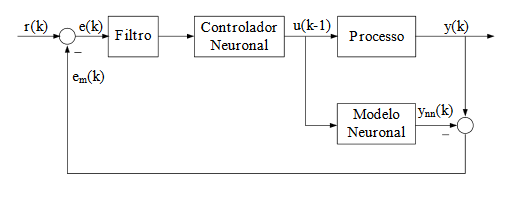


Figura - Modelo do controlador neuronal IMC

O filtro presente nos dois modelos é um filtro passa-baixo com ganho estático unitário. É utilizado de forma a evitar variações bruscas ou do controlador neuronal, no caso da figura 3, ou do erro e(k) do controlador IMC.

Durante a resolução deste trabalho, utilizámos, no modelo *IMC*, um filtro depois do controlador neuronal, e não antes deste.

A função transferência de um filtro de primeira ordem é dado por:

Sendo 1/a a localização pretendida do pólo do filtro.

# Aquisição dos dados

Através de um script, disponibilizado pelo professor, obtiveram-se dois conjuntos de dados. Um de estimação e outro de validação. Estes dados retratavam a resposta do sistema a um conjunto de valores de entrada, obtidos aleatoriamente, de 500 amostras com valores pertencentes ao intervalo: [2,0;5,0] V.

# Treino da rede do sistema

No treino da rede neuronal, após vários treinos com redes de 3 camadas com um número diferente de neurónios, optou-se por uma rede com 4 neurónios (que era o número mínimo pedido) na camada interna. Isto porque, para além do erro associado a cada rede, também tivemos em conta a especialização da rede, que aumenta com o número de neurónios. Quanto mais especializada for uma rede, menor a sua capacidade de representar o comportamento do sistema para conjuntos de dados diferentes àqueles de treino. O código utilizado neste capítulo está em Anexos sobre o nome de *treino\_modelo.mat*.

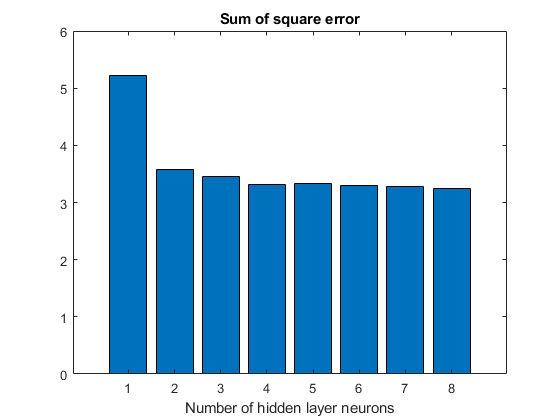


Figura 5- Erro quadrático da rede segundo o número de neurónios da camada interna

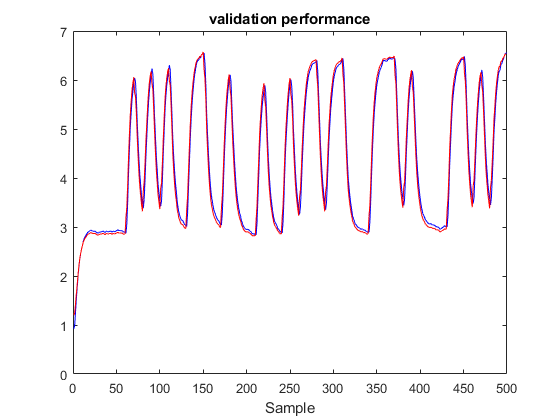


Figura 6- Resposta da rede com os dados de validação

Valor dos pesos e polaridade da rede obtida:

# Comparação com o modelo ARX obtido no laboratório passado

A partir do cálculo do erro quadrático da simulação com os dados de validação, pode se concluir que a rede neuronal é mais próxima ao comportamento do sistema do que o modelo ARX (3,1,2). Isto é, neste caso, a rede neuronal tem um melhor desempenho. O código utilizado neste capítulo está em Anexos sobre o nome de *ARXvsNN.mat*.

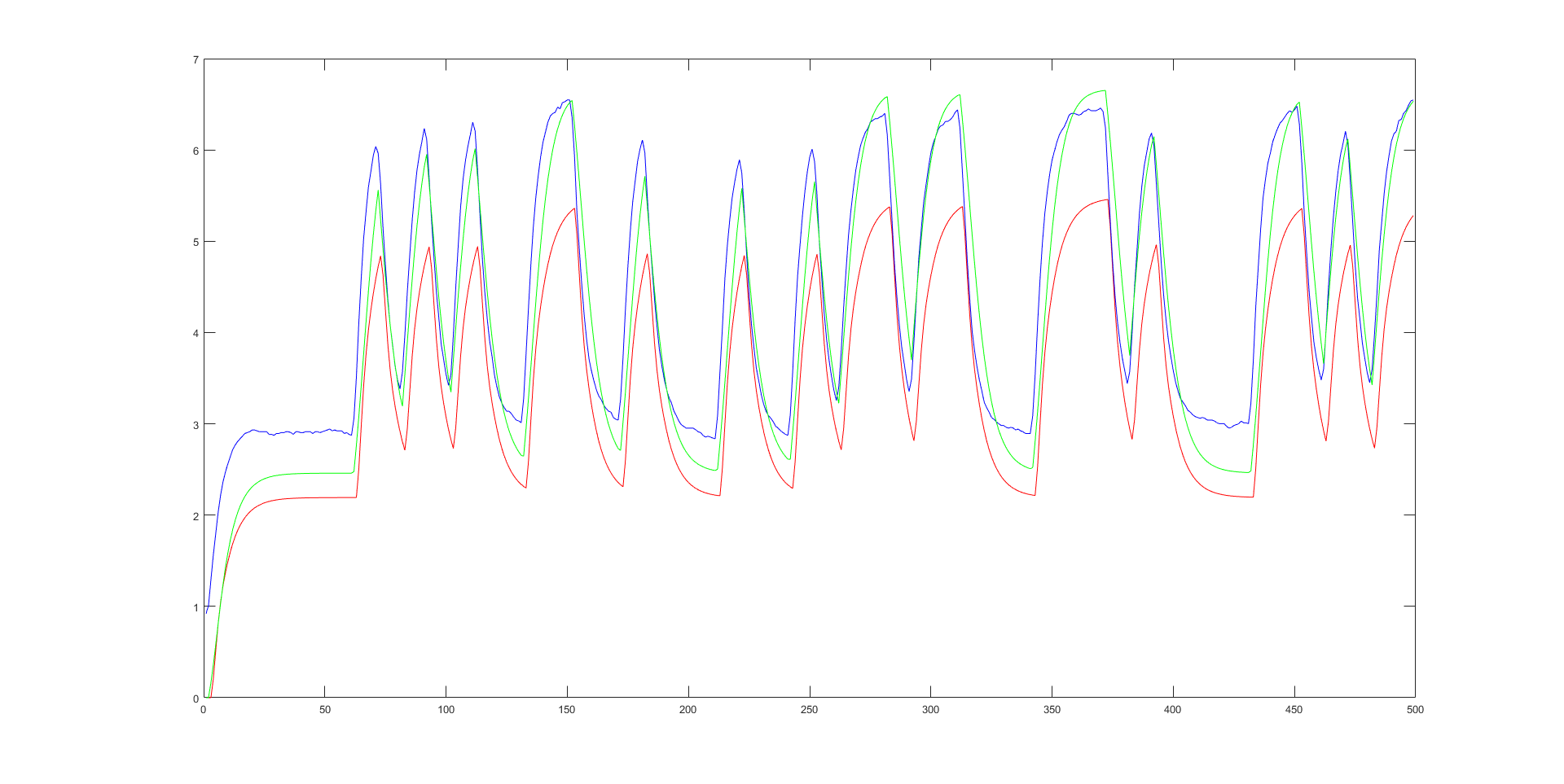
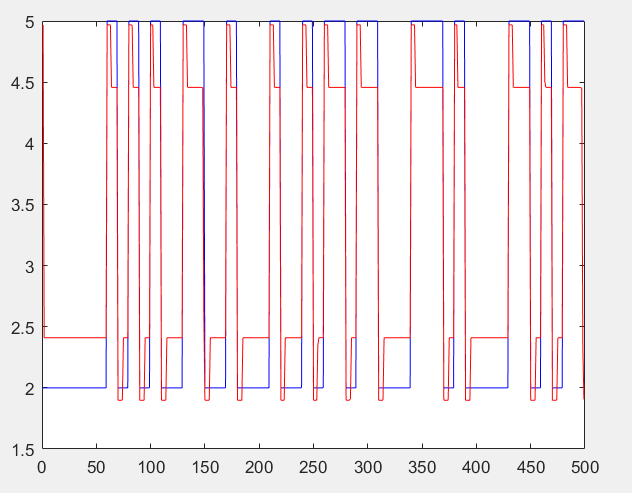


Figura 7- Resposta da rede neuronal e do modelo ARX (3,1,2) – a vermelho o modelo ARX, a verde o modelo neuronal e a azul a saída do processo

Implementação do controlador neuronal

Anteriormente, modelamos as dinâmicas do sistema, ou seja, a maneira como o sistema evolui ao longo do tempo quando este é excitado.

Para modular a ação de controlo foi feito o contrário, utilizou-se o inverso do processo, isto é, com base numa resposta do sistema tentamos chegar à excitação que a provocou. Para tal, usámos uma rede neuronal de 3 camadas internas e treinámo-la com o regressor, e posteriormente validou se o controlador, ficando então na forma . O código utilizado neste capítulo está em Anexos sobre o nome de *treino\_controlador.mat*.

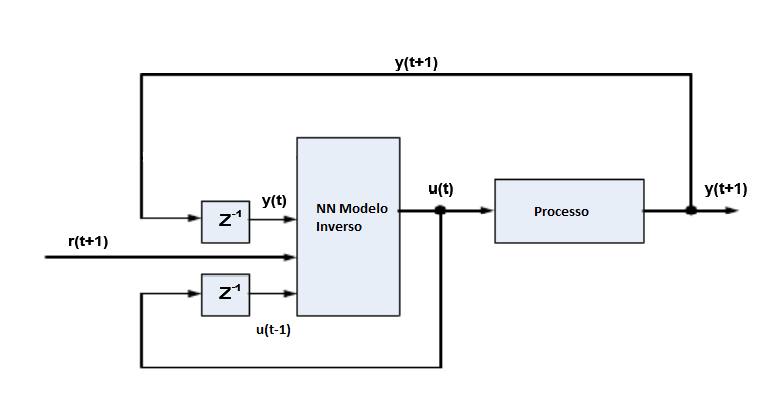


*Figura 8 - Resposta do treino do controlador - a azul a referência e a vermelho a ação de controlo.*

# Simulação

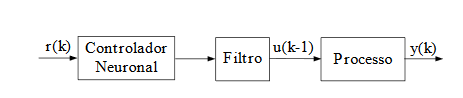
* Controlo inverso directo

Para o controlo inverso directo utilizamos o seguinte modelo O código utilizado neste capítulo está em Anexos sobre o nome de *sim\_InversoDireto.mat*.

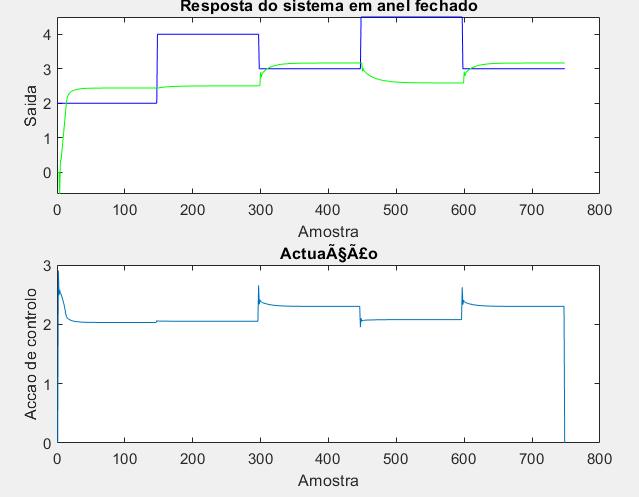


*Figura 9- Modelo de controlo inverso directo*

Após simularmos com vário valores o filtro de 1ª ordem utilizado foi : .



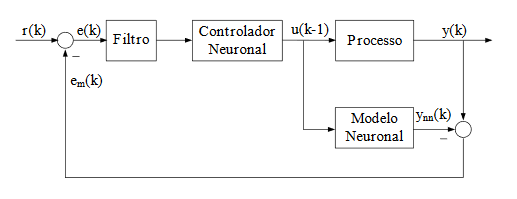
*Figura 10 - Modelo funcional do controlador inverso directo com filtro passa baixo*



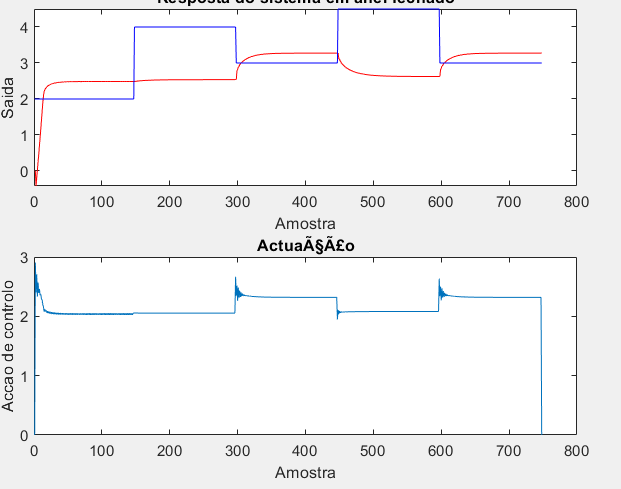
*Figura 11 - Resposta em simulação do sistema de controlo neuronal inverso directo- na parte superior a azul está representada a referência, a verde a saída e a azul no gráfico de baixo a ação de controlo.*

* Controlo neuronal por modelo interno- IMC

Para a implementação do bloco “Processo” recorremos à aproximação pelo modelo ARX, para este fosse diferente do bloco “Modelo Neuronal”, e, desta forma, aproximamo-nos melhor do cenário real em que, por norma, existe um erro entre o modelo neuronal e a resposta do processo. O código utilizado neste capítulo está em Anexos sobre o nome de *sim\_IMC.mat*.



*Figura 12- Modelo funcional de controlador neuronal IMC*



*Figura 13 - Resposta em simulação do sistema de controlo IMC- na parte superior a azul está representada a referência, a verde a saída e a azul no gráfico de baixo a ação de controlo.*

# Controlo do Processo

Depois de se ter simulado a resposta do sistema com os dois topos de controladores, passou-se, efetivamente, ao controlo do processo.

* Controlo inverso directo

A resposta obtida era a esperada de acordo com a simulação. Ver figura abaixo. O código utilizado neste capítulo está em Anexos sobre o nome de *ControladorInversoDireto*.mat

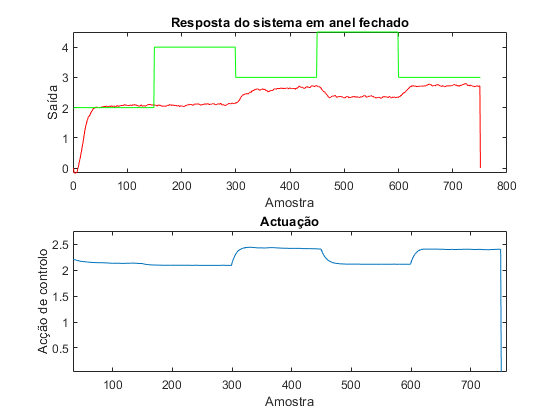


Figura - Resposta do processo em modelo inverso direto

* Controlo neuronal por modelo interno- IMC

A resposta obtida do controlador neuronal no modelo IMC era a esperada de acordo com a simulação. Ver figura abaixo. O código utilizado neste capítulo está em Anexos sobre o nome *de ControladorIMC.mat.*

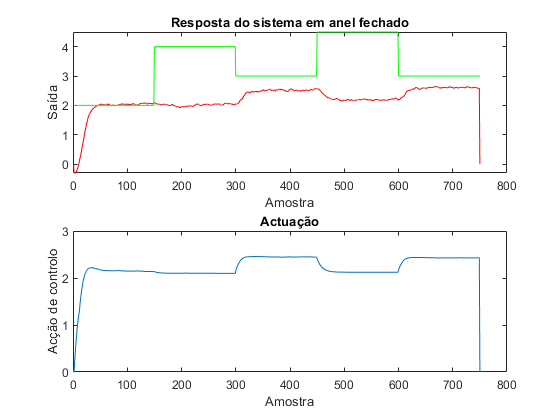


Figura - Resposta do processo em modelo interno

# Outra Rede Neuronal para o controlador

Ao longo da experiência percebemos que o *matlab* nem sempre treinava as redes da mesma forma, mesmo que com os mesmos dados de estimação. Este pormenor em nada (visível a olho nu) alterou o comportamento do modelo do sistema. Contudo alterou, de forma significativa, o comportamento do controlador. Ver as figuras abaixo.

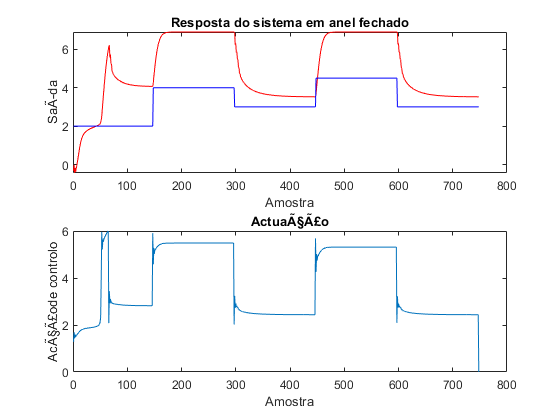


Figura - Simulação do modelo ID

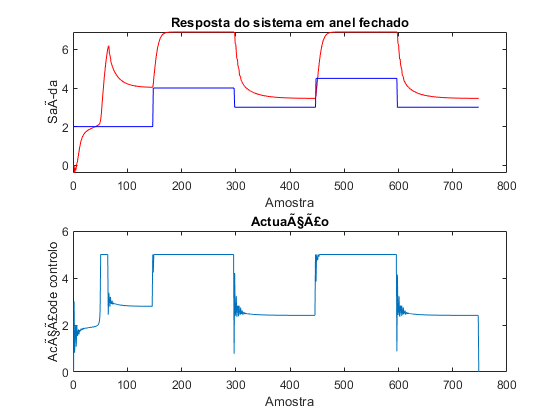


Figura - Simulação do modelo IMC

A resposta com este controlador foi pior à obtida com o controlador usado nos capítulos anteriores. Tal como se pode comprovar pela resposta do processo:

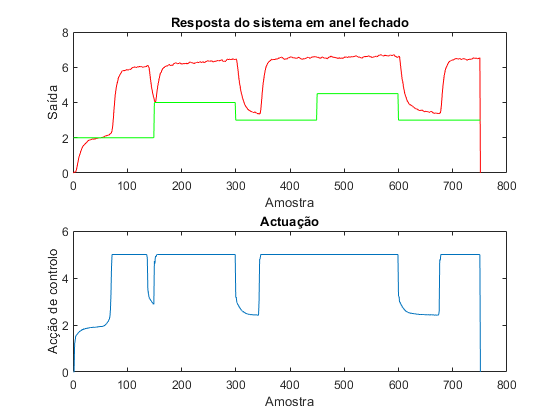


Figura - Resposta do sistema com o modelo ID

# Conclusão

A resposta do sistema ficou longe do ideal, isto deve-se ao facto das redes neuronais, apesar de representarem o sistema com grande precisão, requerem um grande volume e variedade de dados para a sua eficácia, o que não foi possível devido às circunstâncias. Por isso, obtivemos uma rede, que apesar de representar bem o sistema para dinâmicas semelhantes às de treino, não simulava bem o sistema quando a dinâmica de excitação não era idêntica.

Apesar das redes terem conseguido uma boa aproximação da dinâmica do processo, a ação do controlador ficou muito aquém do esperado. Isto aconteceu devido ao erro das entradas passadas.

Observou-se ainda que o modelo ID teve um desempenho muito parecido com o por Modelo Interno, o que demonstra que a rede neuronal do processo caracteriza o mesmo com qualidade. E como tal o erro, e(k), é praticamente nulo, o que torna o modelo IMC igual ao Inverso Direto.

# Anexos

sim\_inversoDireto.mat

%%% Script para testar o controlador %%%

clear all, close, clc

load ('Nets\net.mat')

load ('NetsC\netc.mat')

load('dataset\_prof.mat') % PCT 37-100 (Processo TÃ©rmico)

load processmodel.mat;

Ref = zeros(1,750);

Ref(1:150) = 2;

Ref(150:300) = 4;

Ref(300:450) = 3;

Ref(450:600) = 4.5;

Ref(600:750) = 3;

a=-0.1;

Nmax=length(Ref);

u =zeros(Nmax,1);

uf=zeros(Nmax,1);

y = zeros(Nmax,1);

disp('Em modo controlo!')

for index= 3:Nmax-1

u(index,1) = sim(netc, [Ref(index+1) y(index,1) u(index-1,1)]'); % Função do controlador

%u(index,1)=2;

uf(index,1) = a \* uf(index-1,1) + (1-a) \* u(index,1);

uf(index,1) = max(min(uf(index,1),5),0); % SaturaÃ§Ã£o da excitaÃ§Ã£o

y(index+1,1) = sim(net,[y(index,1), uf(index,1), uf(index-1,1)]');

end

subplot(2,1,1), plot(y(3:end),'r'), hold on, plot(Ref(3:end),'b'),hold off,

title('Resposta do sistema em anel fechado')

ylabel('SaÃ­da'), xlabel('Amostra')

subplot(2,1,2), plot(u(3:end))

title('ActuaÃ§Ã£o')

ylabel('AcÃ§Ã£ode controlo'), xlabel('Amostra')

sim\_IMC.mat

%%% Script para testar o controlador %%%

clear all, close, clc

load ('Nets\net.mat')

load ('NetsC\netc.mat')

load('dataset\_prof.mat') % PCT 37-100 (Processo TÃ©rmico)

Ref = zeros(1,750);

Ref(1:150) = 2;

Ref(150:300) = 4;

Ref(300:450) = 3;

Ref(450:600) = 4.5;

Ref(600:750) = 3;

%ARX(3,1,2)

A = [ 1 -1.504 0.8344 -0.2358 ];

B = [ 0 0 0.1037];

erro = 0;

a=-0.4;

Nmax=length(Ref);

u =zeros(Nmax,1);

uf=zeros(Nmax,1);

y = zeros(Nmax,1);

y\_process = zeros(Nmax,1);

disp('Em modo controlo!')

for index= 4:Nmax-1

u(index,1) = sim(netc, [Ref(index+1)-erro y(index,1) u(index-1,1)]'); % Função do controlador

uf(index,1) = a \* uf(index-1,1) + (1-a) \* u(index,1);

uf(index,1) = max(min(uf(index,1),5),0); % SaturaÃ§Ã£o da excitaÃ§Ã£o

y\_process(index+1,1) = -A(2)\*y(index,1)-A(3)\*y(index-1,1) -A(4)\*y(index-2,1) + B(1)\*uf(index,1) + B(2)\*uf(index-1,1) + B(3)\*uf(index-2,1);

y(index+1,1) = sim(net,[y(index,1), uf(index,1), uf(index-1,1)]');

erro = y\_process(index+1,1) - y(index+1,1)

end

subplot(2,1,1), plot(y(3:end),'r'), hold on, plot(Ref(3:end),'b'), plot (y\_process(3:end),'g'),hold off,

title('Resposta do sistema em anel fechado')

ylabel('SaÃ­da'), xlabel('Amostra')

subplot(2,1,2), plot(uf(3:end))

title('ActuaÃ§Ã£o')

ylabel('AcÃ§Ã£ode controlo'), xlabel('Amostra')

treino\_modelo.mat

clc, clear all, close all

Nhmax = 4;

Nsimax = 3;

Ny = 1;

load dataset\_prof.mat

Le = length(Ye);

Lv = length(Yv);

Unet\_e = [Ye(1:Le-1) Ue(2:Le) Ue(1:Le-1)];

%Unet\_e = [Ye(2:Le) Ue(2:Le) Ue(1:Le-1)]; PQ NAO ISTO

Ynet\_e = Ye(2:Le);

Unet\_v = [Yv(1:Lv-1) Uv(2:Lv) Uv(1:Lv-1)];

Ynet\_v = Yv(2:Lv);

[trash, Nu] = size(Unet\_e);

minu = min(min(Unet\_e));

maxu = max(max(Unet\_e));

for Nh = 1:Nhmax

clc

Errosq = 10e10;

Netid = ['Net\_Nh' num2str(Nh)];

for Nsim = 1:Nsimax

net = newff(ones(Nu,1)\*[minu maxu], [Nh Ny] ,{'tansig','purelin'}, 'trainlm' );

net = init(net);

net.biasConnect = [ 1 0]';

net.performParam.ratio = 0.5; % RegularizaÃ§Ã£o

net.trainParam.epochs = 400;

net.trainParam.show = 100;

net.trainParam.goal = 1e-5;

net.performFcn = 'sse';

net.trainParam.mu\_max = 1e15;

[net,tr] = train(net,Unet\_e',Ynet\_e');

Ynn = sim(net,Unet\_v')';

erro = (Yv(2:end) - Ynn);

SEQ = erro' \* erro;

if SEQ <= Errosq

Errosq = SEQ;

eval([Netid ' = net']);

end

end

eval(['save Nets\' Netid '.mat net']);

Sum\_sq(Nh) = Errosq;

end

idoptinet = find(Sum\_sq <= min(Sum\_sq));

netopt = ['Net\_Nh' num2str(idoptinet)];

eval(['load Nets\' netopt '.mat']);

%Ynn = sim(net,Unet\_v')';

save('Nets\net.mat', 'net', '-mat', '-v7.3')

Ynn = sim(Net\_Nh3,Unet\_v')';

figure(1)

plot(Yv, 'b'), hold on, plot(Ynn, 'r'), hold off

title('validation performance'), xlabel('Sample')

figure(2)

bar(Sum\_sq), title('Sum of square error')

xlabel('Number of hidden layer neurons')

clear Net\_\*

delete Nets\Net\_\*

treino\_controlador.mat

clear all, close all, clc

load('dataset\_prof.mat') % PCT 37-100 (Processo TÃ©rmico)

Nsimax = 5;

M = length(Ue);

Unn = [Ye(2:M) Ye(1:M-1) Ue(1:M-1)]'; % 3 Entradas

%Unn = [Ye(1:M-1) Ue(2:M)]'; % 3 Entradas

Uvv = Uv(2:M);

Utrain = Ue(2:M); % Target

[Nu, trash] = size(Unn); % NÃºmero de entradas

Ny = 1; % NÃºmero de saÃ­das

%OOOOOOOOOOOOOOOOOOO Topologia da Rede Neuronal

Nh = 3;

minu = min(min(Unn));

maxu = max(max(Unn));

%OOOOOOOOOOOOOOOOOOO ConstruÃ§Ã£o da Rede Neuronal

disp('\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ConstruÃ§Ã£o da Rede...')

for Nsim = 1:Nsimax

netc = newff(ones(Nu,1)\*[minu maxu], [Nh Ny] ,{'tansig','purelin'}, 'trainlm' );

netc = init(netc);

netc.biasConnect = [ 1 0]';

netc.performParam.ratio = 0.5; % RegularizaÃ§Ã£o

netc.trainParam.epochs = 400;

netc.trainParam.show = 100;

netc.trainParam.goal = 1e-5;

netc.performFcn = 'sse';

netc.trainParam.mu\_max = 1e15;

%OOOOOOOOOOOOOOOOOOO Treino da Rede Neuronal

clc

disp('\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Treino da Rede...')

disp('Â ')

[netc,tr] = train(netc,Unn,Utrain');

end

W1=netc.IW{1,1};

W2=netc.LW{2,1};

B1=netc.b{1,1};

%OOOOOOOOOOOOOOOOOOO SimulaÃ§Ã£o da Rede Neuronal (Conunto de treino)

Unn = [Yv(2:M) Yv(1:M-1) Uv(2:M)]'; % 3 Entradas

Ynn = sim(netc,Unn);

Errs = (Utrain - Unn')' \* (Utrain - Unn')

plot(Uvv,'b');

hold on

plot(Ynn,'r');

%eval(['load NetsC\' netopt '.mat']);

save('NetsC\netc.mat', 'netc', '-mat', '-v7.3')

ControladorInversoDireto.mat

clear all, close, clc

load ('Nets\net.mat')

load ('NetsC\netc.mat')

W1c=netc.IW{1,1};

W2c=netc.LW{2,1};

B1c=netc.b{1,1};

Ref = zeros(750,1);

Ref(1:150) = 2;

Ref(150:300) = 4;

Ref(300:450) = 3;

Ref(450:600) = 4.5;

Ref(600:751) = 3;

Ts = 0.08; %confirmar

a = 0.9;

Nmax=length(Ref);

u =zeros(Nmax,1);

uf=zeros(Nmax,1);

y = zeros(Nmax,1);

erro = zeros(750,1);

usbinit();

disp('Em modo controlo!')

for index = 2:length(Ref)-1

y(index,1) = usbread(0);

erro(index,1) = y(index,1) - Ref(index,1);

tic

if index <=2

u(index,1) = Ref(index);

else

%u(index,1) = sim(netc, [Ref(index+1), y(index,1), u(index-1,1), u(index-2,1)]');

u(index,1) = W2c \* tanh (W1c \* [Ref(index+1), y(index,1), u(index-1,1)]' + B1c);

end

uf(index,1) = a \* uf(index-1,1) + (1-a) \* u(index-1,1);

uf(index,1) = max(min(uf(index,1),5),0); % SaturaÃ§Ã£o da excitaÃ§Ã£o

usbwrite(uf(index),0)

Dt = toc;

pause(Ts-Dt)

end

usbwrite(0,0)

erro = sumsqr(erro);

subplot (2,1,1), plot (y(1:749)), hold on, plot (Ref(1:749)), hold off

save expdataDI.mat Ref uf y -mat

ControladorIMC.mat

clear all, close, clc

load ('Nets\net.mat')

load ('NetsC\netc.mat')

W1c=netc.IW{1,1};

W2c=netc.LW{2,1};

B1c=netc.b{1,1};

W1=net.IW{1,1};

W2=net.LW{2,1};

B1=net.b{1,1};

Ref = zeros(750,1);

Ref(1:150) = 2;

Ref(150:300) = 4;

Ref(300:450) = 3;

Ref(450:600) = 4.5;

Ref(600:750) = 3;

Ts = 0.08;

a = 0.9;

Nmax=length(Ref);

u =zeros(Nmax,1);

uf=zeros(Nmax,1);

y = zeros(Nmax,1);

erro = zeros(750,1);

usbinit();

disp('Em modo controlo!')

for index = 2:length(Ref)-1

y(index,1) = usbread(0);

erro(index,1) = y(index,1) - Ref(index,1);

tic

if index <=2

u(index,1) = Ref(index);

else

%ym = sim(net,[y(index-1,1), uf(index-1,1), uf(index-2,1)]');

ym = W2 \* tanh (W1 \* [y(index,1), uf(index,1), uf(index-1,1)]' + B1);

epsilon = y(index,1) - ym;

%u(index,1) = sim(netc, [Ref(index+1)-epsilon, y(index,1), u(index-1,1), u(index-2,1)]');

u(index,1) = W2c \* tanh (W1c \* [Ref(index+1), y(index,1), u(index-1,1)]' + B1c);

end

uf(index,1) = a \* uf(index-1,1) + (1-a) \* u(index-1,1); %filtro passa baixo

uf(index,1) = max(min(uf(index,1),5),0); % SaturaÃ§Ã£o da excitaÃ§Ã£o

usbwrite(uf(index),0)

Dt = toc;

pause(Ts-Dt)

end

erro = sumsqr(erro);

usbwrite(0,0)

save expdataIMC.mat Ref uf y -mat

ARXvsNN.mat

%comparacao da rede neuronal utilizada com o modelo arx do primeiro

%trabalho

load ('Nets\net4.mat')

load('dataset\_prof.mat') % PCT 37-100 (Processo TÃ©rmico)

Nmax = length(Uv) - 1

%ARX(3,1,2)

A = [ 1 -1.504 0.8344 -0.2358 ];

B = [ 0 0 0.1037];

y = zeros(length(Uv),1);

yNN = zeros(Nmax,1);

erro = zeros(length(Uv),1);

erroNN = zeros(length(Uv),1);

for index = 3:Nmax

y(index+1,1) = -A(2)\*y(index,1)-A(3)\*y(index-1,1) -A(4)\*y(index-2,1) + B(1)\*Uv(index,1) + B(2)\*Uv(index-1,1) + B(3)\*Uv(index-2,1);

erro(index+1) = y(index+1,1)-Yv(index+1,1);

end

for index = 3:Nmax

yNN(index) = sim(net,[yNN(index-1,1), Uv(index-1,1), Uv(index-2,1)]');

erroNN(index+1) = yNN(index,1)-Yv(index,1);

end

errosqrARX=sumsqr(erro)

errosqrNN=sumsqr(erroNN)

plot ( 1,1), plot (y(1:Nmax),'r'), hold on, plot (Yv(1:Nmax),'b'), plot (yNN(1:end),'g'), hold off

# Bibliografia utilizada

slidesNN-2020.vrs2.pdf, Departamento de Eng. Electrotécnicae de Computadores (versão 2020)