Universidade de Aveiro
2008

Departamento de Electrónica, Telecomunicações e Informática

Vítor Manuel Costa Ribeiro

Controlo de sistemas dinâmicos com redes neuronais artificiais

Universidade de Aveiro
2008

Departamento de Electrónica, Telecomunicações e Informática

Vítor Manuel Costa Ribeiro

Controlo de sistemas dinâmicos com redes neuronais artificiais

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Electrónica e de Telecomunicações (Mestrado Integrado), realizada sob a orientação científica da Dra Petia Georgieva, Professora Auxiliar Convidada do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro.

o júri

presidente	Prof. Dr. Tomás António Mendes Oliveira e Silva Professor Associado da Universidade de Aveiro
orientador	Prof^{a}. Dra. Petia Georgieva Professora auxiliar convidada da Universidade de Aveiro
arguente	Prof. Dr. Fernando Gomes Martins Professor auxiliar do Departamento de Engenharia Química da Faculdade de Engenharia da Universidade do Porto

agradecimentos

Este trabalho assinala o fim de mais uma etapa da minha vida, mas que não seria possível sem a contribuição de algumas pessoas especiais, como os meus pais e amigos.
Queria agradecer à minha orientadora Prof. Dra. Petia Georgieva que me ajudou, com os seus conhecimentos científicos, com a sua disponibilidade e empenho exemplar. Em alguns momentos foi determinante pelo incentivo.
A algumas pessoas da biblioteca, como a Sra. Anabela, a Sra. Paula e a Sra. Sandra, que me ajudaram, sempre com a sua disponibilidade.
Mais uma vez à minha mãe, pelo incansável incentivo. Todas as palavras do Mundo não chegariam para lhe agradecer.
Ao meu Pai, pelo sacrifício hercúleo. És um exemplo.
Ao Nuno Magalhães, pela amizade de muitos anos.
Ao Nelson Cardoso, pela equipa afinada e que complementou, as fragilidades de parte a parte em alguns trabalhos práticos. Obrigado pela amizade.
Aos muitos amigos que fiz no decorrer do curso, como o Gaspar, entre outros.
A todos o meu mais sincero obrigado.
Queria agradecer a Deus, pela coragem que tive, ao optar por seguir em frente, há 3 anos atrás, após prolongada doença. Por isso e tudo o mais, muito obrigado. Nada disto teria sido possível sem Ele. Mesmo que nenhum deste esforço viesse no futuro a dar proveitos, não me arrependerei, nem me arrependo de nada.
Quero também dedicar este trabalho,sem mágoa, antes pelo contrário, com carinho, caridade e amizade, a todos aqueles familiares, fora do meu seio familiar mais directo, que nessa altura, não acreditaram, que eu chegasse até aqui. O meu afecto sincero para todos vós.

palavras-chave

Redes Neuronais Artificiais,Sistemas dinâmicos, Controlo,Identificação, Pêndulo Invertido

resumo

Este trabalho enquadra-se na área de algoritmos de controlo baseados em redes neuronais artificiais (RNA), designados também como controladores neuronais.

Foram estudadas três arquitecturas principais de RNAs (redes unidireccionais, redes recorrentes do tipo NNARX e NNARMAX e rede recorrente de Elman), quatro algoritmos de treino (gradiente descendente, método de Newton, método de Gauss-Newton e método de Levenberg-Marquardt) e diferentes estruturas de controlo neuronal.

Dos vários esquemas possíveis foram mais detalhadamente projectados três controladores designados como: "modelo inverso genérico", "modelo inverso especializado" e "modelo óptimo".

O processo de desenho e as características dos controladores neuronais foram ilustrados através de um caso de estudo não-linear e instável, pêndulo invertido. Foram testadas quatro estruturas de controlo: controlo inverso, controlo interno, controlo feedforward e controlo com linearização por realimentação. Foi efectuado um estudo comparativo entre esses estruturas neuronais e um compensador clássico do tipo PID.

keywords

Artificial Neural Networks, Control, Dynamical Systems, Identification, Inverted Pendulum

abstract

The present work is focused in the area of control algorithms based on Artificial Neural Networks (ANNs), termed also neural control.
Tree main ANN architectures are studied (feedforward, recurrent NN of NNARX and NNARMAX type and Elman NN) , four learning algorithms (gradient descent method, Newton method, Gauss-Newton method and Levenberg-Marquardt method) and a number of different neural control structures.
Among various possible solutions the following controllers were designed generic inverse model , specialized inverse model and optimal model.
The design process and the controller's characteristics were illustrated with the inverted pendulum nonlinear unstable case study.
Four neural control structures were tested: inverse control, internal control, feedforward control and feedback linearization control.
A comparative study between the above mentioned neural control structures and a classical PID controller was also accomplished.

Índice

Índice de Figuras

Índice de tabelas

Acrónimos

MSE	Mean Square Error
NNARX	Neural Network AutoRegressive with eXogenous input
NNARMAX	Neural Network AutoRegressive Moving Average with eXogenous input
NNOE	Neural Network Output Error
RNA	Artificial Neural Networks
RNU	Feedforward Neural Networks
RNR	Recurrent Neural Networks
PID	Proportional, Integral Derivative
SOM	Self Organizing Maps
ART	Adaptative Ressonance Theory
LB	Bandwidth
BPA	Back-Propagation Algorithm

1. Introdução

O controlo de sistemas não lineares é um assunto que actualmente tem atraído grande interesse nas sociedades académicas e industriais [University of Strathclyde]. Técnicas de controlo clássico (controladores proporcionais, PD, PID) são baseados em modelos linearizados dos sistemas físicos, o que representa perda de informações, que muitas vezes são importantes para o funcionamento da planta com altos níveis de exigência. A utilização do denominado controlo inteligente tem aberto uma nova perspectiva no tratamento de sistemas não lineares e no projecto dos seus controladores. Na tentativa de desenvolver um controlo mais versátil e robusto, muitos trabalhos na área do controlo inteligente foram desenvolvidos ao longo da ultima década. Neste sentido as Redes Neuronais Artificiais (RNA) pertencem as técnicas mais utilizadas para a implementação de um controlador inteligente.

RNAs são ferramentas computacionais, com um grande número de aplicações em varias áreas de investigação tais como reconhecimento de padrões, identificação, classificação, sistemas de fala, visão e controlo. As vantagem das RNA residem nas suas características principais: capacidade de aproximar funções não lineares, capacidade de generalização, robustez na presença de ruído e distúrbios de carga, robustez quando existem incertezas nos parâmetros da planta ou quando existem dinâmicas não modeladas. Os modelos obtidos através de redes neuronais permitem levar em consideração as não linearidades do processo.

Neste sentido a motivação principal deste trabalho é familiarizar com as técnicas e o formalismo da modelação e controlo baseado em RNA, ilustrar essas técnicas para casos de estudo desafiadores (como é o pêndulo invertido) e analisar o desempenho das várias estruturas de controlo baseado em RNA.

2. Redes Neuronais Artificiais

2.1. A inspiração biológica

Em 1894, um histologista e médico espanhol, Santiago Ramon y Cajal, o pioneiro da ciência que estuda actualmente o cérebro humano, chegou à conclusão que o cérebro humano era constituído por pequenas unidades, às quais ele chamou neurónios. Segundo os seus estudos, neurónios são células polarizadas, que recebem sinais através de extensões altamente ramificadas, designadas dendrites e enviam informação, através de extensões não ramificadas, denominadas axónios [Andrés Uribe, 1999]. O fluxo de informação é unidireccional das dendrites para os axónios(Fig. 2.1).

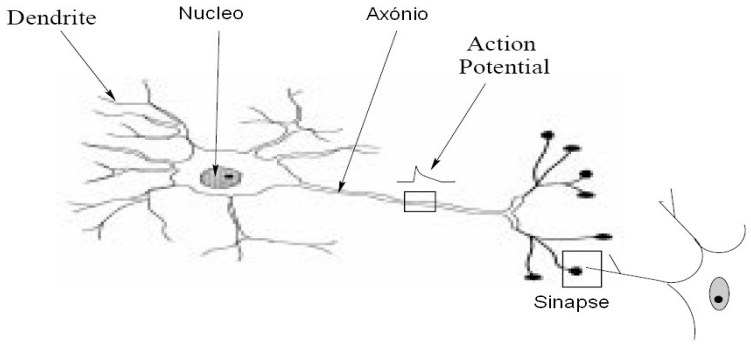


Figura 2.1: Elementos básicos de um neurónio biológico

Os axónios transmitem informação, para outras células neuronais, na forma de impulsos eléctricos, denominados potenciais de acção("action potentials" do inglês). Quando o potencial de acção chega ao terminal do axónio, o neurónio anterior no fluxo de informação, liberta neurotransmissores químicos, a partir da vesícula sináptica(Figura 2.2). Estes neurotransmissores químicos, fazem a mediação da comunicação inter-neuronal nas sinapses (estas fazem as ligações entre o neurónio anterior e posterior) e ligam-se aos receptores na membrana do neurónio posterior, para o inibir ou excitar. Devido ao elevado número de ligações entre neurónios e às várias inibições e excitações, cria-se uma redução da diferença de potencial eléctrico na membrana externa de cerca de 70mV(a superfície interior torna-se negativa relativamente à superfície exterior). Por este motivo e devido ao aumento da permeabilidade do sódio, há um movimento de sódio carregado positivamente(Na+), do fluido extracelular para o interior da célula, o citoplasma. Este sequência de acontecimentos, pode gerar um potencial de acção no neurónio posterior. Se isso acontecer, implica que o neurónio posterior é activado [Andrés Uribe, 1999].

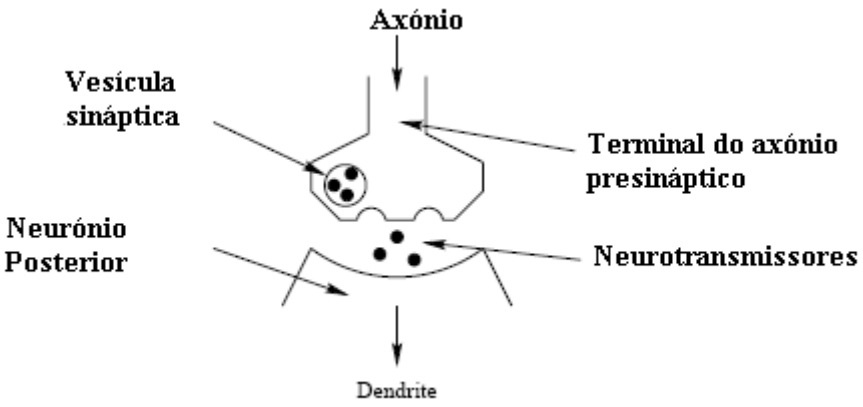


Figura 2.2: Comunicação inter-neuronal

Esta descrição fisiológica do cérebro humano, serviu de inspiração a muitos engenheiros e cientistas para desenvolver sistemas adaptativos, com capacidade para aprender, a partir da experiência. Desta forma surgiram as RNAs.

2.2. Breve descrição histórica

Uma RNA é constituída por unidades de processamento, denominadas neurónios, com capacidade para o processamento paralelo de informação e armazenamento de conhecimento empírico [Paulo Cortez et al, 2000] .

Entre os cientistas que se dedicaram ao estudo das RNAs estão McCulloch e Pitts em 1943. Foram os criadores das RNAs, aproximando a descrição biológica de um neurónio, com a matemática lógica [Paulo Cortez et al, 2000].

Em 1949, Hebb, propôs a regra de aprendizagem, que viria a ser conhecida como regra de Hebb, a partir de estudos das redes neuronais biológicas(RNB) e dos seus fenómenos de adaptação [F. Dias, 2005] .

Em 1958 o psicólogo americano Frank Rosenblatt, propôs um modelo computacional de neurónios ao qual ele designou de perceptrão. Relativamente ao modelo proposto por McCulloch e Pitts, o modelo de Frank Rosenblatt diferenciava-se pela introdução de pesos numéricos entre as ligações. Desta forma, substituiu-se as simples ligações inibitórias/excitatórias, dos criadores das RNAs [Andrés Uribe, 1999]. O modelo proposto por Rosenblatt, usava como única função de activação a função de Heaviside (tabela 2.1) [F. Dias, 2005]. Em 1969 Minsky e Papert, publicaram o livro Perceptrons. A publicação deste livro teve repercussões, pois levou ao abrandamento do ritmo da investigação das RNAs . Os autores alegavam a incapacidade de um perceptrão isolado desempenhar uma qualquer função em exclusivo [F. Dias, 2005].

Embora o estudo das RNAs nunca tenha parado, só em 1980 com o aparecimento dos computadores pessoais, é que as RNAs ressurgiram com grande vitalidade, com significativas contribuições nesta área, como por exemplo o estudo de Grossberg, que estabeleceu um novo tipo de SOM (Self Organizing Maps), conhecido como ART(Adaptative Ressonance Theory) [F. Dias, 2005]. Em 1982, Hopfield desenvolve um novo tipo de RNAs, baseado em redes recorrentes. No mesmo ano surgem as redes SOM do tipo Kohonen [Paulo Cortez et al, 2000].

Em 1986 um trabalho de Rumelhart, Hinton e Williams, dão um grande impulso no desenvolvimento de RNAs, com a criação do back-propagation algorithm (BPA), sendo actualmente o algoritmo de treino mais frequentemente usado. É usado em cerca de 70% das aplicações envolvendo RNAs [F. Dias, 2005] .

Em 1989 Cybenko, Hornick e outros, provam com os seus artigos, dois aspectos a sublinhar:

- Dado o número de neurónios necessários, ou seja a estrutura da RNA, usando funções de activação contínuas e diferenciáveis e com apenas uma camada escondida, as RNAs são capazes de aproximar qualquer função contínua [F. Dias, 2005].

- Dado o número de neurónios necessários, ou seja a estrutura da RNA, usando funções de activação contínuas e diferenciáveis e com duas camadas escondidas, as RNAs são capazes de aproximar qualquer função [F. Dias, 2005].

No início dos anos 90, Vapnik e seus colaboradores, realizaram um estudo, do qual resultou uma classe de redes supervisionadas, designadas Support Vector Machines, para a regressão e reconhecimento de padrões [Paulo Cortez et al, 2000] .

2.3. O componente básico: o neurónio

Dentro de uma arquitectura neuronal o neurónio, é o seu componente mais básico. É constituído por cinco partes essenciais:

- Um conjunto de entradas $\{ x_i \}$, i=1,2,3,4,....
- Um conjunto de conexões w_i , correlacionadas com um valor real ou binário, denominado de peso. É possível a existência de uma outra conexão, designada bias, que funciona como uma constante, ou um peso adicional. A bias tem uma entrada igual a +1, que serve , para que se estabeleçam as correctas condições operacionais para o neurónio [Paulo Cortez et al, 2000].

  Uma função que reduz os n argumentos de entrada a um único valor. Normalmente esta função é um somatório.

- Uma função de activação(*f*), que pode ser linear ou não linear.
- Saída(s) representada(s) na Fig. 2.2 pela variável y.

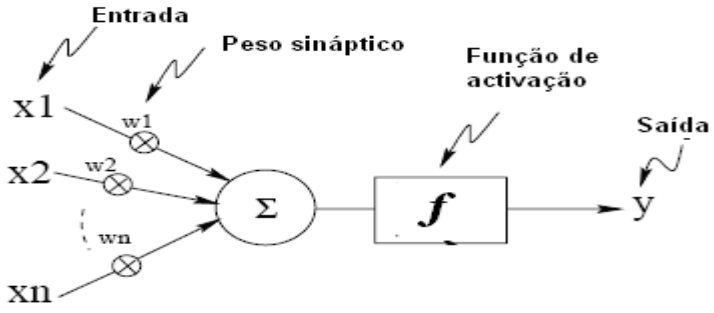


Figura 2.3: Modelo de um neurónio artificial

A saída representada na Fig. 2.3 é dada pela expressão da Eq. 2.1:

$$y = f(x) \tag{2.1}$$

As funções de activação mais usadas são mostradas na tabela 2.1. Na coluna "Desenho da função", x representa os valores no eixo horizontal, enquanto y, representa os valores no eixo vertical.

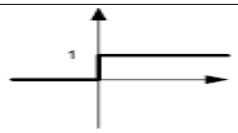
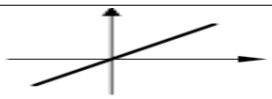
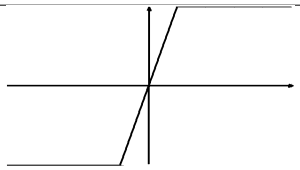
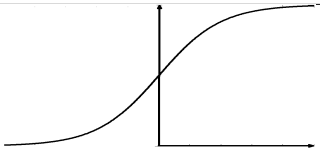
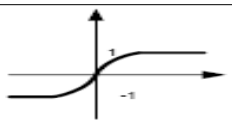
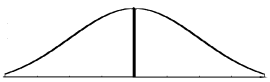
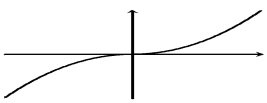
$$x = \sum w_i x_i, i = 1,2,3,4,\ldots. \tag{2.2}$$

Parte das funções representadas na tabela 2.1, têm contradomínio limitado. Exemplos disso são a função sigmoidal (contradomínio [0,1]),a função tangente hiperbólica (contradomínio [-1, 1]) e a função gaussiana (contradomínio [0,1]). Estas são três significativas funções de activação. A função sigmoidal é usada mais frequentemente.

2.4. Estruturas de RNAs

Existem vários tipos de topologias/estruturas de RNAs. Em geral estas dividem-se em duas categorias: redes neuronais unidireccionais e redes neuronais recorrentes.

Tabela 2.1: Tabela com funções de activação frequentemente usadas em RNAs

Nome	Função	Desenho da função
Heaviside	$\begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases}$	
Heaviside simétrico	$\begin{cases} 1 & x \geq 0 \\ -1 & x < 0 \end{cases}$	
Linear	$y = x$	
Linear com saturação	$\begin{cases} 0 & x < 0 \\ x & 0 \leq x \leq 1 \\ 1 & x > 1 \end{cases}$	
Linear simétrico com saturação	$\begin{cases} -1 & x < -1 \\ x & -1 \leq x \leq 1 \\ 1 & x > 1 \end{cases}$	
Logística sigmoidal	$y = \frac{(e^{x} - e^{-x})}{(e^{x} + e^{-x})}$	
Tangente hiperbólica	$y = (1 + e^{-x})^{-1}$	
Gaussiana	$y = e^{\left(-\frac{x^2}{2k^2}\right)}$	
Quadrada	$y = -\operatorname{sign}(x)\, x^2$	

1. Redes neuronais unidireccionais(RNU, na literatura inglesa , "feedforward neural networks"). Caracterizam-se pela unidireccionalidade das suas ligações, sempre no sentido da entrada para a saída, sem ligações laterais entre neurónios da mesma camada, nem realimentações[F. Dias,2005]. Podem ser redes neuronais de uma só camada escondida e uma só camada de saída, ou então podem ser redes neuronais de múltiplas camadas, sem limite para o número de camadas escondidas ou intermédias [Paulo Cortez et al, 2000].
2. Redes neuronais recorrentes(RNR). Estas redes têm pelo menos um "loop" de realimentação, i.e., tem pelo menos uma ligação, que liga a saída de um neurónio, à sua própria entrada, ou então à entrada de um neurónio constituinte, de uma camada mais longínqua da saída da estrutura. Em alternativa, ou existindo em conjunto, pode haver uma ligação de um neurónio a outro neurónio na mesma camada. As RNR tem características muito vantajosas, para a modelação de sistemas dinâmicos, pois tem a capacidade de guardar informação sobre o seu próprio estado interno [F. Dias, 2005]. Exigem algoritmos de aprendizagem mais complexos, em termos computacionais, comparativamente com as redes neuronais unidireccionais.

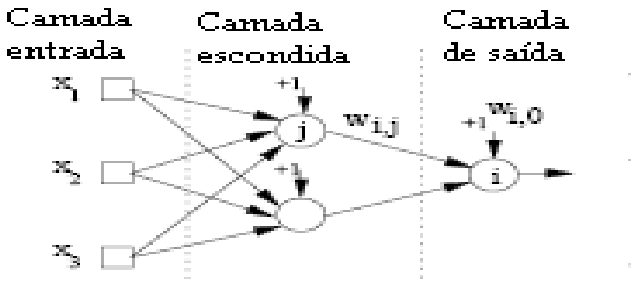


Figura 2.4: Estrutura geral de uma rede neuronal organizada em camadas

Na Fig. 2.5 podem ser vistos exemplos de estruturas de RNAs. Exemplos de redes do tipo unidireccional são a), b) e g) e de redes neuronais recorrentes são c), d), e) e f).

Dois problemas que se colocam, relativamente à escolha de uma estrutura de RNA para resolver um determinado problema que envolva RNAs, são os problemas de oversmoothing e overfitting. Se uma estrutura de uma RNA tem poucas camadas, tem poucos neurónios por camada, ou o esquema de ligações é inadequado, a rede não tem a capacidade de aproximar com precisão os valores da saída da rede neuronal, aos valores dos exemplos de treino(os exemplos de treino são um conjunto de valores, agrupados dois a dois, num par [entrada saída], que contêm os valores desejados de obter à saída da RNA). Este problema é conhecido como oversmoothing. Por outro lado, se uma rede é demasiado complexa, ou seja, tem muitas camadas de neurónios, ou muitos neurónios por camada, pode ocorrer o risco de o algoritmo de treino, precisar de muito tempo para treinar a RNA e de a mesma não generalizar correctamente. A RNA pode por este

motivo não ser capaz de tratar adequadamente as entradas, que não façam parte dos exemplos de treino. Este problema é conhecido por overfitting.

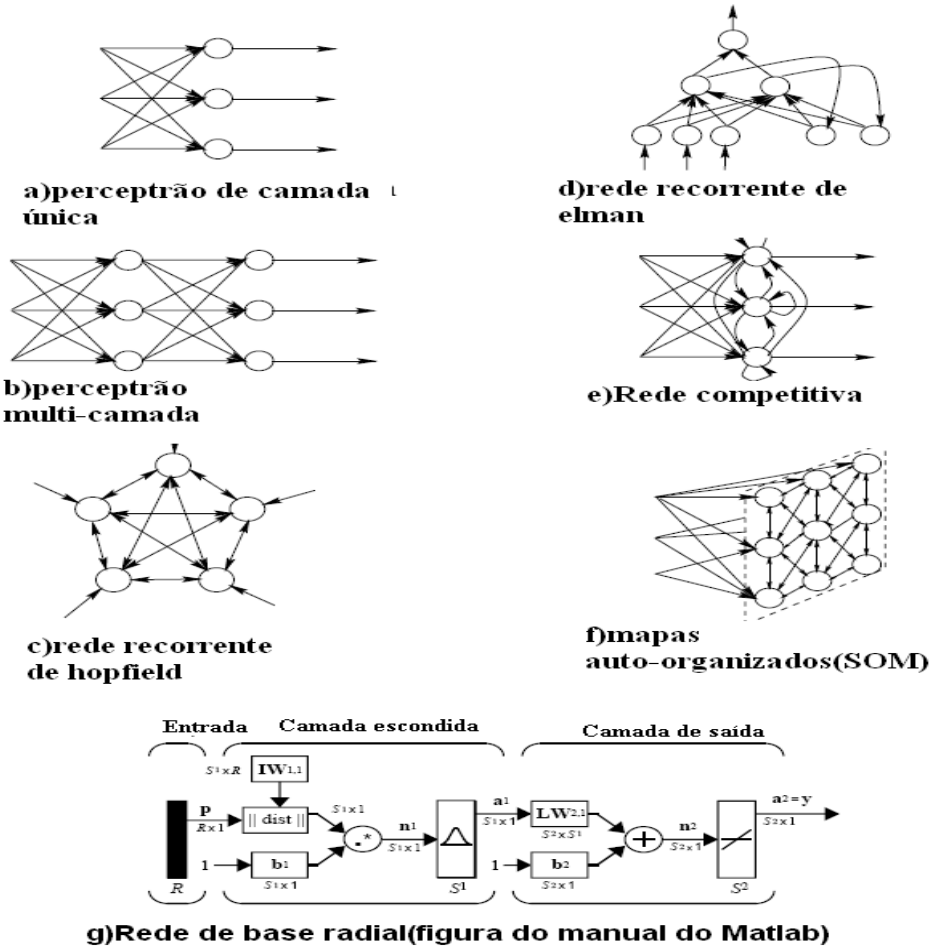


Figura 2.5: Estruturas de RNAs

2.5.Aprendizagem

A aprendizagem está directamente ligada com a melhoria do desempenho de um dado sistema. Numa RNA este objectivo é atingido, alterando os valores dos pesos das sinapses e/ou alterando a estrutura da RNA [Andrés Uribe, 1999]. A forma como é atingido esse objectivo está relacionada com o algoritmo de treino usado, que pode pertencer a um dos seguintes paradigmas de aprendizagem.

1. Supervisionada. O paradigma de aprendizagem supervisionado, é constituído por um "professor", i.e. para determinados valores de treino na entrada, são fornecidos valores correctos na saída. Estes valores, designados exemplos de

treino, são comparados com os valores obtidos na saída da rede, de modo a gerarem um erro, que serve para ajustar os valores dos pesos nas sinapses [Paulo Cortez et al, 2000].

2. De reforço. Neste caso também existe um "professor". Este é mais limitado, pois só fornece à rede uma indicação, sobre se a resposta da rede é correcta ou errada, tendo a mesma que usar esta informação, para melhorar a sua eficiência [Paulo Cortez et al, 2000]. O conceito de reforço é definido em [R. Sutton,1998]. Geralmente neste tipo de paradigmas, existe um valor escalar, que recompensa as respostas correctas com um prémio(valor escalar elevado) e as respostas incorrectas com um castigo(valor escalar baixo). Existe também neste paradigma uma função, que calcula a partir de um determinado estado, qual o máximo prémio acumulado, que pode ser obtido no futuro, partindo desse mesmo estado. Estas predições são feitas a partir do conhecimento da experiência passada. Isto quer dizer que os algoritmos não só se deixam conduzir, pelo valor instantâneo do prémio que recebem, mas também pelo máximo valor de prémio acumulado, que podem receber no futuro, partindo de um determinado estado.
3. Não supervisionada. Neste caso não existe "professor", nem exemplos de treino. A rede tem mecanismos de aprendizagem, que lhe permitem, tendo unicamente a informação das entradas, catalogá-las, como pertencendo a uma determinada categoria [F. Dias, 2005]. O paradigma de aprendizagem é usado com sucesso em redes com topologia ART, com um mecanismo de aprendizagem competitivo [Andrés Uribe, 1999]. Este mecanismo consiste na competição entre os neurónios da camada de saída(a camada competitiva), pelo estado activo. Como só um neurónio pode estar activo, estes competem, entre eles, por esse estado [Andrés Uribe, 1999].

2.6. Algoritmos de treino

O ajuste dos parâmetros numa RNA é feito por um algoritmo de treino. Estes parâmetros são os pesos de cada ligação nas sinapses. Ao ajuste destes pesos, designa-se treino. O treino é um processo iterativo, em que após cada iteração, os pesos são ajustados, de forma a minimizarem a função de custo, que usualmente é a expressão, para o erro quadrático médio(Mean Squared Error do inglês) da Eq. 2.3:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(o_i - y_i)^2 \tag{2.3}$$

"o" é o valor desejado e y é o valor efectivamente obtido na saída da RNA, após cada iteração de treino.

Antes de partir para o treino de uma rede, é necessário dispor de informação na forma apropriada, na forma de um vector do tipo [entradas saídas], do sistema a ser modelado, saídas estas que são os valores desejados, da Eq. 2.3 .

São discutidos nesta secção quatro algoritmos de treino, entre os vários, que se podem encontrar dispersos pela literatura. Todos estes quatro algoritmos de treino têm em comum o facto de utilizarem o paradigma supervisionado e apresentam uma estrutura de actualização dos pesos da rede neuronal, expressa pelas Eqs. 2.4 e 2.5.

$$w_{k+1} = w_k + \alpha_k . p_k \tag{2.4}$$

$$\Delta w_k = \left(w_{k+1} - w_k \right) = \alpha_k . p_k \tag{2.5}$$

p_k representa a direcção de procura, α_k é a taxa de aprendizagem. A escolha de p_k é o que distingue os algoritmos, que são apresentados nesta secção.

2.6.1. O método do gradiente descendente

Quando se actualiza o valor dos parâmetros, o objectivo é que o valor da função de custo na sua iteração k+1, seja menor que o valor da função de custo na iteração k:

$$F\left(w_{k+1} \right) < F\left(w_k \right) \tag{2.6}$$

A forma como este resultado é conseguido, depende da maneira como for escolhido p_k . A expansão em série de Taylor de primeira ordem em torno de w_k , é dada por :

$$F\left(w_{k+1} \right) = F\left(w_k + \Delta_k \right) = F\left(w_k \right) + G(w) . \Delta w_k \tag{2.7}$$

Onde G(w_k)= $\nabla F(w_k)|_{w=w_k}$. Para que $F\left(w_{k+1} \right)$ seja menor que $F\left(w_k \right)$, o produto $G(w,k) . \Delta w_k$, tem que ser menor que zero. Como $G\left(w_k \right) . \Delta w_k = G\left(w_k \right) . \alpha_k . p_k$ e α_k é sempre positivo, implica que $G\left(w_k \right) . p_k < 0$. Pressupondo que o módulo de p_k não varia, visto ser uma direcção, então o valor de p_k que faz tornar máximo negativo, o produto interno $p_k . G\left(w_k \right)$ é $p_k = -G\left(w_k \right)$.

A equação que faz a actualização dos pesos da rede neuronal é dada pela Eq. 2.8:

$$w_{k+1} = w_k - \alpha_k . G\left(w_k \right) \tag{2.8}$$

A escolha de α_k é crítica, na medida em que determina a velocidade de convergência do algoritmo, assim como pode ou não garantir, que o valor de $F(w_{k+1}) < F(w_k)$. Um valor demasiado baixo, pode levar a uma convergência demasiado lenta. Um valor demasiado elevado, pode levar à ocorrência da situação descrita na Fig. 2.6. Em [Sorensen, 1994] é apontado um valor entre 0.001 e 0.1 para o valor de α_k, ficando este valor dependente da escala da função F(w_k).

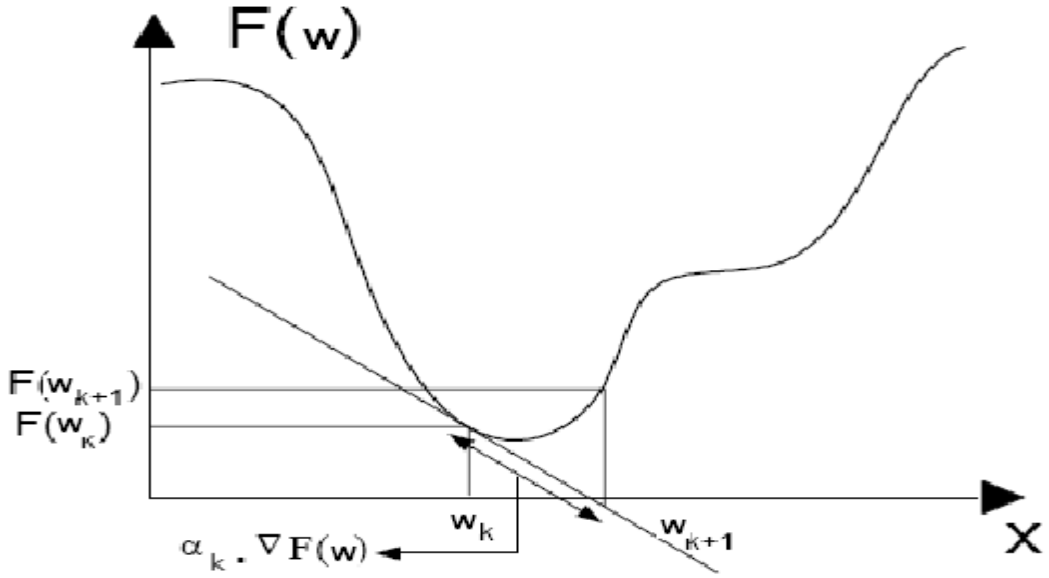


Figura 2.6: Uma má escolha de α_k

2.6.2. O algoritmo de Newton

O algoritmo do gradiente descendente utiliza a expansão em série de Taylor de 1ª ordem. O algoritmo de Newton utiliza a expansão em série de Taylor de 2ª ordem.

$$F(w_{k+1}) = F(w_k + \Delta w_k) = F(w_k) + G(w_k).\Delta w_k + \frac{1}{2}\Delta w_k^2 . H(w_k) \tag{2.9}$$

Onde $H(w_k) = \nabla^2 F(w_k)|_{w=w_k}$. Para localizar um extremo da Eq. 2.9 de forma a encontrar um mínimo, deriva-se a Eq. 2.9 e iguala-se a zero, conforme é demonstrado na Eq. 2.10.

$$G(w_k) + H(w_k).\Delta w_k = 0 \tag{2.10}$$

Pode obter-se o valor de Δw_k , pela Eq. 2.11.

$$\Delta w_k = -H(w_k)^{-1}.G(w_k) \tag{2.11}$$

A regra de iteração do método de Newton é dada pela Eq. 2.12.

$$w_{k+1} = w_k - H(w_k)^{-1}.G(w_k) \tag{2.12}$$

2.6.3. O algoritmo de Gauss-Newton

O algoritmo de Gauss-Newton é uma variação do algoritmo de Newton. Se considerarmos que $v_i = o_i - y_i$ e procedermos à substituição desta expressão na Eq. 2.3, tem-se que:

$$MSE = \frac{1}{N} * \sum_{i=1}^{N} v_i^2(w) \tag{2.13}$$

sendo o_i o valor i desejado na saída e y_i o valor i efectivamente obtido na saída. Traduzindo a Eq. 2.13 para a notação que tem vindo a ser utilizada até aqui:

$$F(w) = \frac{1}{N} * \sum_{i=1}^{N} v_i^2(w) \tag{2.14}$$

O j-ésimo elemento do gradiente da Eq. 2.14 é:

$$G(w)_j = \frac{2}{N} \sum_{i=1}^{N} v_i(w) \frac{dv_i(w)}{dw_j} \tag{2.15}$$

O gradiente pode ser descrito na forma matricial, conforme demonstrado na Eq. 2.16:

$$G(w) = \frac{2}{N}.J^T(w).v(w) \tag{2.16}$$

em que J(w) é a matriz Jacobiana.

A matriz hessiana H(w) é também usada, no método de Newton. O seu k,j-ésimo elemento(correspondente à linha k, coluna j) para funções quadráticas é expresso na forma da Eq. 2.17:

$$H(w)_{k,j}=\frac{2}{N}.\sum_{i=1}^{N}\frac{dv_i(w)}{dw_k}.\frac{dv_i(w)}{dw_j}+v_i(w)\frac{d^2v_i(w)}{dw_k.dw_j} \tag{2.17}$$

Na forma matricial a matriz Hessiana toma a forma da Eq. 2.18:

$$H(w)=\frac{2}{N}.\left[J^T(w).J(w)+S(w)\right] \tag{2.18}$$

Em que $S_{(w)}$ é dado pela Eq. 2.19:

$$S(w)_{kj}=\sum_{i=1}^{N}v_i(w).\frac{d^2v_i(w)}{dw_k\,dw_j} \tag{2.19}$$

É possível simplificar a matriz Hessiana H(w), utilizando a aproximação de que S(w) é muito pequeno, quando comparado com o primeiro termo. A matriz Hessiana H(w) toma a forma da Eq. 2.20:

$$H(w)\approx\frac{2}{N}J^T(w).J(w) \tag{2.20}$$

Substituindo a Eq. 2.20 e a Eq. 2.16, na Eq. 2.11, obtém-se a Eq. 2.21:

$$\Delta w_k=-\left[\frac{2}{N}.J^T(w_k).J(w_k)\right]^{-1}.\frac{2}{N}.J^T(w_k).v(w_k) \tag{2.21}$$

O ajuste dos pesos da rede, no algoritmo de Gauss-Newton, é dado por:

$$w_{k+1}=w_k-\left[\frac{2}{N}.J^T(w_k).J(w_k)\right]^{-1}.\frac{2}{N}.J^T(w_k).v(w_k) \tag{2.22}$$

2.6.4.O algoritmo de Levenberg-Marquardt

O algoritmo de Levenberg-Marquardt, resulta dos trabalhos independentes de [Levenberg, 1944] e [Marquardt, 1963] e é também uma variação do método de Newton. Uma dificuldade existente no algoritmo de Gauss-Newton é a possibilidade de a aproximação da matriz hessiana, não ser invertível. Por este motivo é possível utilizar a seguinte solução:

$$Hm(w) = H(w) + \mu I \tag{2.23}$$

I é a matriz identidade e μ é um valor tal , que torna a matriz Hessiana invertível. Hm(x) é a matriz modificada. A Eq. 2.24 é idêntica à Eq. 2.21. O que as diferencia é o facto de na Eq. 2.24 a matriz hessiana, presente na Eq. 2.21, ser substituída pela matriz Hessiana modificada da Eq. 2.23.

$$\Delta w_k = \left[-2\mathrm{J}^T w_k * J\left(w_k\right) + \mu_k I \right]^{-1} . 2 . J^T w_k . v\left(w_k\right) \tag{2.24}$$

Esta equação determina a alteração de parâmetros no algoritmo Levenberg-Marquardt. O valor de μ_k , além de tornar a matriz Hessiana invertível, determina a velocidade de convergência.

Os algoritmos de treino, podem ser divididos em algoritmos do tipo batch, ou do tipo recursivo. Existem versões dos algoritmos de treino anteriormente discutidos, do tipo batch e do tipo recursivo. Os algoritmos do tipo batch requerem, que a cada iteração todos os elementos do conjunto de treino sejam avaliados. Estes são geralmente úteis para identificar sistemas off-line, embora possam também ser aplicados on-line. Pelo contrário os algoritmos do tipo recursivo, ajustam os pesos para cada par entrada-saída do conjunto de treino. São mais usados para o treino on-line.

Um treino on-line, implica que o sistema a ser modelado e a rede estejam inseridas numa estrutura de treino, em que a rede é treinada em tempo real e os seus pesos ajustados amostra a amostra. Pelo contrário o treino off-line implica que sejam retirados do sistema a ser modelado, uma sequência de dados, que numa fase posterior e independente serão inseridos todos de uma só vez, na rede neuronal, em cada iteração,.

3. Identificação e controlo com redes neuronais artificiais

A identificação de sistemas é a área científica relativa à construção ou selecção de modelos de sistemas dinâmicos, para servirem determinados propósitos. A identificação de sistemas implica quatro fases genéricas [F. Dias,2005]:

-aquisição de dados

-selecção da estrutura do modelo

-treino do modelo

-validação do modelo

O processo de identificação é sequencial (ver Fig. 3.1). Isto não implica, que caso o modelo não seja validado, se tenha que repetir todo o processo. Se os passos iniciais tiverem sido bem executados, não necessitam de ser repetidos.

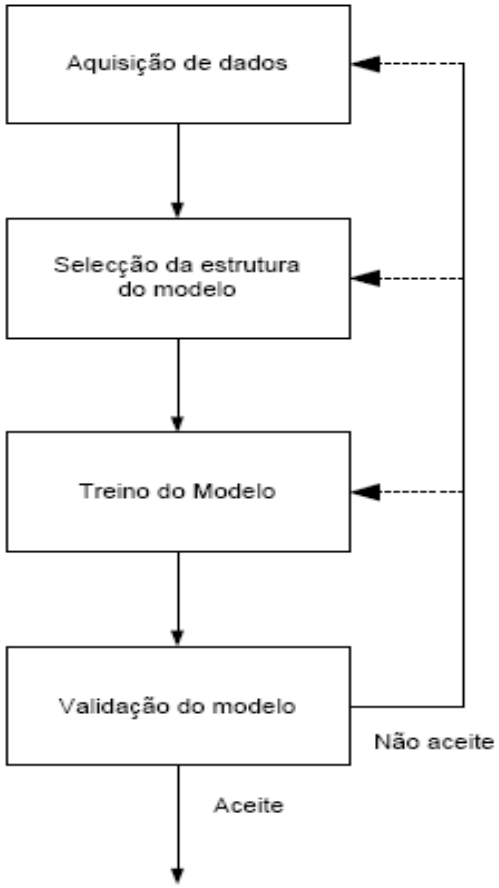


Figura 3.1: Fases da identificação de um sistema

3.1.Aquisição de dados

Para um boa aquisição de dados é necessário garantir condições experimentais favoráveis, como dados pouco afectados por ruído e ausência de perturbações significativas [F. Dias, 2005].

Um dos primeiros passos a executar nesta fase é a determinação do período de amostragem. Em [Karl Åström, 1997] é sugerido, que o período de amostragem seja escolhido de acordo com o tempo de subida do sistema em malha fechada, na forma mostrada pela Eq. 3.1:

$$Nr = \frac{tr}{h} \approx 4 \ a \ 10 \tag{3.1}$$

Tr é o tempo de subida, h é o período de amostragem e Nr é o número de amostras que o tempo de subida deve conter. Em termos da relação da frequência de amostragem com a largura de banda do sistema em malha fechada, a Eq. 3.1 sugere que deverá ser escolhido uma frequência de amostragem, 10 a 30 vezes superior à largura de banda do sistema em rad/s.

A escolha do período de amostragem, sugerida em 3.1 foi testada (50 Hz), mas não se revelou a mais apropriada nesta dissertação. Existiu dificuldade em captar a dinâmica do sistema a controlar em estudo, com a escolha do período de amostragem referida em 3.1. Se a dinâmica do sistema não for captada convenientemente os resultados de controlo não são satisfatórios . Foi usada a abordagem sugerida em [University of Michigan]. O período de amostragem mínimo é escolhido para uma largura de banda (em rad/s) 30 vezes inferior à frequência de amostragem, conforme sugere a Eq. 3.2:

$$h \leq \frac{1}{(30 * LB)} \tag{3.2}$$

LB é a largura de banda em rad/s do sistema em malha fechada.

A aquisição de dados deve ser feita, ou com o sistema em malha fechada ou com o sistema em malha aberta. Se possível deverá ser escolhido um sistema em malha aberta, mas quando o sistema é instável, por exemplo, ou quando se pretende retirar valores dentro da gama, onde o sistema é suposto operar, deverá ser escolhido um sistema em malha fechada. A escolha de um sinal de entrada também é crítico. Deverá ser escolhido um sinal de entrada com um espectro de frequência alargado, assim como uma gama de amplitudes abrangente [F. Dias, 2005]. Um sinal de ruído branco tem uma gama de frequências alargada.

Depois da aquisição de dados, estes deverão ser divididos em dados de treino e dados de validação. É sugerido na literatura, uma preparação dos dados, que inclui fazer uma mudança de escala, remoção de dados redundantes, remoção dos outliers e filtragem. A mudança de escala revela-se vantajosa, pelas seguintes justificações apresentadas em [F. Dias, 2005]:

- é provável que os sinais sejam medidos em unidades físicas diferentes e haverá uma tendência para que o sinal de maior magnitude domine o modelo .
- o algoritmo de treino torna-se mais robusto e leva a uma convergência mais rápida
- as funções de activação têm na sua maioria um comportamento constante a partir de determinados valores da entrada, fazendo com que a resposta da função de activação possa ser idêntica para valores muito distintos da entrada.
- a experiência mostrou que foram obtidos modelos mais precisos.

A remoção dos outliers (são erros de leitura, em que um determinado dado de entrada tem um valor correspondente errado na saída) não é fácil de ser feito, pois é necessário um conhecimento prévio do sistema. A remoção de dados redundantes, visa diminuir principalmente o tempo de treino. Em sistemas dinâmicos, não é desejável a remoção de dados redundantes devido à dependência, que a saída tem das amostras anteriores.

A filtragem torna-se necessária, quando se recolhe dados de treino e de validação de um sistema real, que vem afectado de ruído. Num sistema simulado sobre o Simulink não existe esse problema. Muito provavelmente também não existe o problema dos outliers em sistemas simulados sobre o Simulink.

3.2.Estrutura do modelo

3.2.1. Rede unidireccional de camada única

A Fig. 3.2 representa a estrutura de uma rede unidireccional de uma única camada escondida.

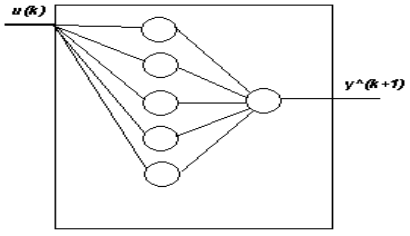


Figura 3.2: Rede unidireccional de camada única

Esta rede tem uma só entrada u(k), embora uma rede unidireccional de camada única, possa ter mais que uma entrada.

3.2.2. Rede recorrente de Elman

A Fig. 3.3 representa a estrutura de uma rede de Elman. Esta estrutura apresenta realimentações internas, o que lhe permite, ter memória. D é um atraso introduzido da saída da camada intermédia, para a entrada da mesma camada.

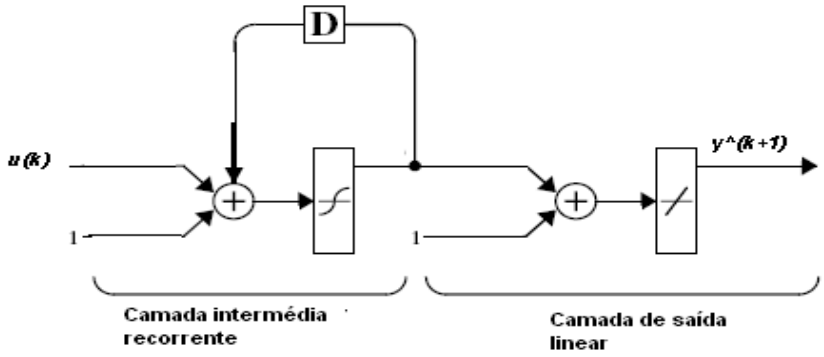


Figura 3.3: Rede de Elman

3.2.3. Outras estruturas de redes recorrentes

Às estruturas que se seguem, a literatura designa de classes de modelos não lineares baseados em RNAs [F. Dias, 2005].

3.2.3.1.NNARX(Neural Network AutoRegressive with eXogenous input)

Uma classe de modelos NNARX tem a estrutura da Fig.3.4. O vector de regressores é definido pela Eq. 3.3:

$$\varphi(k,\xi)=\left[y(k)\ldots y(k-n),u(k-td)\ldots u(k-td-m)\right] \tag{3.3}$$

A saída da rede neuronal é definido pela Eq. 3.4:

$$y\hat{}(k+1)=g\left(y(k)\ldots y(k-n),u(k-td)\ldots u(k-td-m)\right) \tag{3.4}$$

ξ é o vector de parâmetros(os pesos das sinapses da rede neuronal), y(k) é a saída da planta no instante k e y^(k+1) é a saída da rede neuronal no instante k+1.

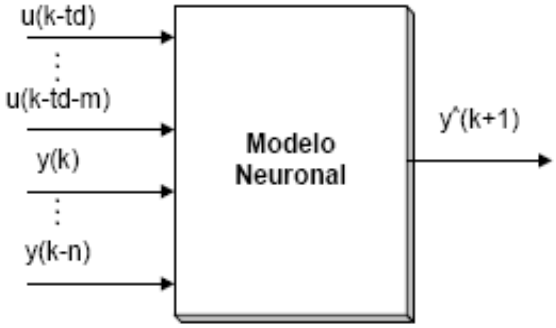


Figura 3.4: Classe de modelos NNARX

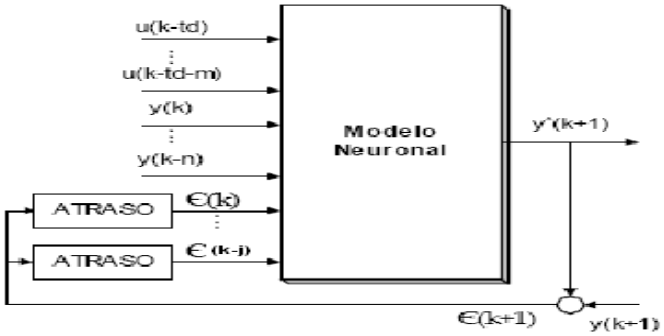


Figura 3.5: Classe de modelos NNARMAX

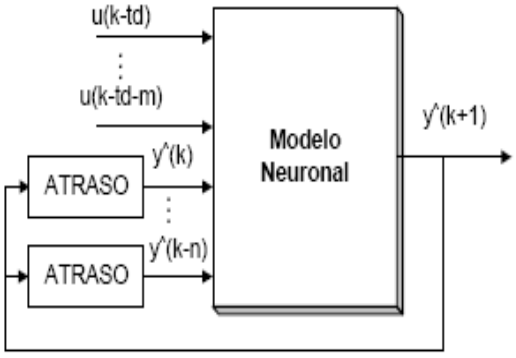


Figura 3.6: Classe de modelos NNOE

3.2.3.2. NNARMAX(Neural Network AutoRegressive Moving Average with eXogenous input)

A classe de modelos NNARMAX tem a estrutura da Fig. 3.5. O vector de regressores é definido pela Eq. 3.5:

$$\varphi(k,\xi)=[y(k)\dots y(k-n),u(k-td)\dots u(k-td-m),\varepsilon(k,\xi)..\varepsilon(k-j,\xi)] \tag{3.5}$$

A função de saída da RNA pertencente à classe de modelos NNARMAX é definido pela Eq. 3.6:

$$y\hat{}(k+1)=g(y(k)\dots y(k-n),u(k-td)\dots u(k-td-m),\varepsilon(k,\xi)\dots\varepsilon(k-j,\xi)) \tag{3.6}$$

3.2.3.3. NNOE(Neural Network Output Error)

Na classe de modelos NNOE a estrutura é a representada na Fig. 3.6. O vector de regressores é definido da seguinte forma:

$$\varphi(k,\xi)=[y\hat{}(k)\dots y\hat{}(k-n),u(k-td)\dots u(k-td-m)] \tag{3.7}$$

A função de saída da rede neuronal define-se como:

$$y\hat{}(k+1)=g(y\hat{}(k)\dots y\hat{}(k-n),u(k-td)\dots u(k-td-m),\varepsilon(k,\xi)\dots\varepsilon(k-j,\xi)) \tag{3.8}$$

Algumas considerações devem ser feitas, para uma melhor escolha da classe de modelos neuronal. A classe de modelos NNARX tem a vantagem de ser sempre estável, dado que existe uma pura relação algébrica entre a predição da saída y(k+1), os regressores e as entradas [M. Nørgaard et al,2000]. As classes NNARMAX e NNOE por terem realimentações do próprio modelo, têm implicações negativas na sua estabilidade.

3.3. Treino

Na secção 2.6 foram discutidos alguns algoritmos de treino. Existem diversas formas de utilizar esses algoritmos de treino, para treinar uma rede neuronal. A essas formas a literatura chama de estruturas de treino, que serão discutidas nesta secção. Na generalidade o treino visa minimizar uma determinada função de custo do tipo:

$$V_N(\xi, Z^N) = \frac{1}{N}\sum_{k=1}^{N}(o(k) - \hat{o}(k \tag{3.9}$$

ξ é o vector dos parâmetros da rede, Z^N é uma sequência de treino e N é o número de pontos,contidos na sequência.

3.3.1. O modelo directo(treino off-line)

O modelo directo da planta, tem como entradas e saídas as mesmas grandezas físicas do sistema a controlar ou planta. Pretende ser uma "cópia" desta, embora lhe esteja subjacente um determinado erro. A estrutura de treino usada geralmente para treinar um modelo directo é a representada pela Fig. 3.7.

O valor resultante da avaliação da Eq. 3.9, é usado para corrigir os pesos da RNA. Normalmente esta estrutura de treino é aplicada off-line (os algoritmos de treino são do tipo batch).

3.3.2. O modelo inverso

O modelo neuronal inverso tem como entrada a grandeza física que está à saída da planta e como saída a grandeza física, que está à entrada da planta. Existem dispersos pela literatura várias estruturas de treino, para treinar modelo inversos. Os modelos inversos depois de treinados e validados, são inseridos nas estruturas de controlo, funcionando como controladores.

Nesta secção discute-se duas estruturas de treino:o treino genérico e o treino especializado. Obtêm-se dois modelos inversos: o modelo inverso genérico e o modelo inverso especializado. Acresce-se o modelo óptimo, que resulta do treino, numa estrutura de treino especializado.

3.3.2.1.O modelo inverso genérico(treino off-line)

A estrutura de treino de um modelo inverso genérico é descrito na Fig. 3.8. Da Fig.3.8 pode concluir-se que a função de custo é do tipo:

$$V_N(\xi, Z^N) = \frac{1}{N}\sum_{k=1}^{N}\left(u(k) - u\hat{}(k)\right)^2 \tag{3.10}$$

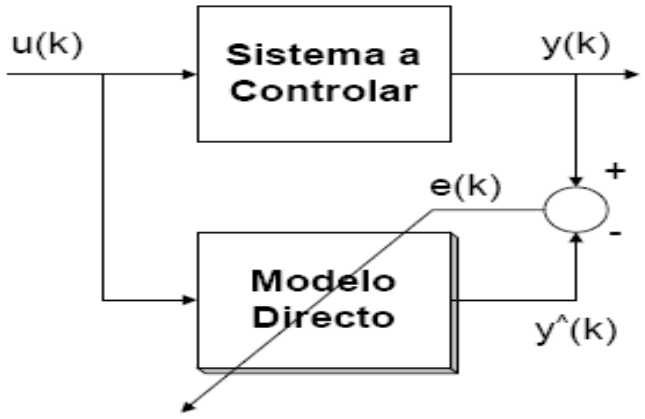


Figura 3.7: Estrutura de treino de um modelo directo

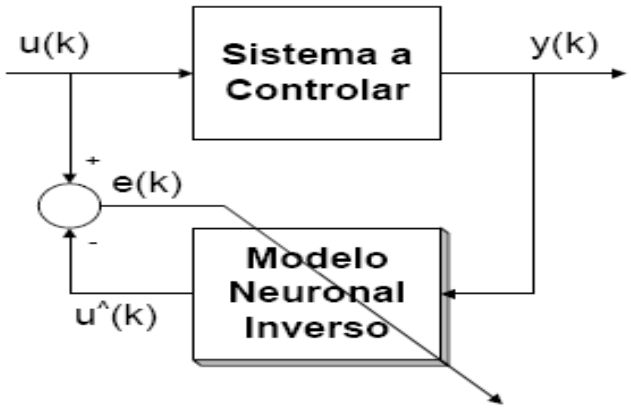


Figura 3.8: Estrutura de treino de um modelo inverso genérico

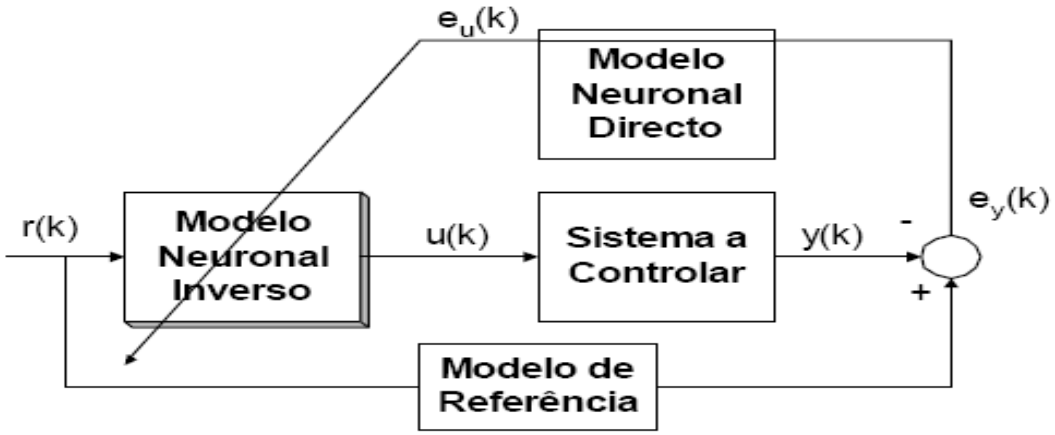


Figura 3.9: Estrutura de treino de um modelo inverso especializado

3.3.2.2. O modelo inverso especializado(treino on-line)

Quando um modelo inverso é inserido numa malha de controlo, como controlador, pretende-se minimizar uma função de custo do tipo:

$$V_N(\xi, Z^N) = \frac{1}{N}\sum_{k=1}^{N}\left(r(k) - y(k)\right)^2 \tag{3.11}$$

r representa a referência e y é a saída da planta. Este tipo de estrutura de treino designa-se na literatura inglesa como "goal directed" ou direccionada ao objectivo . Numa estrutura de controlo, o que é pretendido é que o valor inserido na referência, seja igual ao valor que se tem à saída da malha de controlo. Desta forma uma estrutura de treino que tenha a mesma filosofia, é mais desejável que uma estrutura que apenas minimiza o erro do sinal de controlo *u*.

A estrutura de treino de um modelo inverso especializado, utilizada durante esta dissertação foi a representada na Fig. 3.9. Esta figura demonstra que o erro a minimizar é o da Eq. 3.11. Nesta estrutura de treino o modelo neuronal inverso inicial, poderá ser um modelo inverso genérico que depois é ajustado, derivando num modelo inverso especializado. O modelo de referência tem o objectivo de filtrar a referência de modo a ter uma saída y(k) suavizada. Se a referência é uma onda quadrada, não convém que a saída tenha overshoot, então aplica-se um filtro, que faz com que a saída y(k) vá "atrás" de uma referência filtrada e suavizada.

3.3.2.3.O modelo óptimo(treino on-line)

Com a estrutura de treino especializado da Fig.3.9 existem outras funções de custo que podem ser aplicadas. Uma destas funções de custo que deriva do controlo óptimo e que foi utilizada durante nesta dissertação, tem a seguinte equação:

$$V_N(\xi, Z^N) = \frac{1}{N}\left(\sum_{k=1}^{N}\left(r(k) - y(k)\right)^2 + \rho * u^2(k)\right) \tag{3.12}$$

Ao controlador que resulta da aplicação desta função de custo chamou-se de "modelo óptimo".Neste caso o factor ρ serve para penalizar sinais quadrados do sinal de controlo, assim como amplitudes elevadas do mesmo. À medida que ρ é aumentado progressivamente, o sinal de controlo, vai-se tornando cada vez mais suave e atinge menores valores [M. Norgaard et al, 2000].

Para finalizar esta parte dedicada à identificação, são necessárias algumas considerações sobre a qualidade/validação do modelo. Para determinar se um determinado modelo pode ser validado, é necessário determinar se este gera um baixo

erro de treino(aqui baixo é relativo e depende da magnitude dos sinais envolvidos) e se o erro dos dados de validação é da mesma ordem do erro dos dados de treino. Se isto não acontecer, é porque a rede aprendeu características especificas, ou do sinal usado na fase de treino, ou do ruído que afecta o sistema. Neste caso o modelo não poderá ser validado.

3.4. Estruturas de controlo

Serão estudadas nesta secção 4 estruturas de controlo, baseadas em redes neuronais: controlo inverso(direct-inverse control), controlo interno(internal model control), controlo feedforward e controlo através de linearização por realimentação(feedback linearisation).

A primeira estrutura de controlo supra citada, é a mais simples de todas e consiste em colocar em série com a planta um modelo inverso da mesma, como é apresentado na Fig. 3.10.

Para um sistema que possa ser descrito pela seguinte equação:

$$y(k+1)=g\left(y(k)\ldots y(k-n),u(k)\ldots u(k-m+1)\right) \tag{3.13}$$

onde n é o número de regressores da saída y e m é o número de regressores da entrada *u.* A saída do modelo inverso da Fig. 3.10 pode ser descrito da seguinte forma:

$$u(k)=g^{-1}\left(y(k+1),\ldots,y(k-n+1),u(k-1),\ldots,u(k-m+1)\right) \tag{3.14}$$

A estrutura de controlo representada na Fig. 3.11, é denominada controlo interno. O controlo interno é uma estrutura de controlo, que permite que o sinal de realimentação reflicta os efeitos das perturbações, que afectam o sistema e as diferenças entre o modelo e o sistema real.

É possível chegar a algumas considerações, representando matematicamente o diagrama de blocos da Fig. 3.11, no domínio de Laplace. Supondo que S descreve o comportamento da planta ou do sistema a controlar, M^{-1} descreve o comportamento do modelo inverso e M o comportamento do modelo directo, usando a regra de Mason é possível descrever o sistema através da Eq. 3.15:

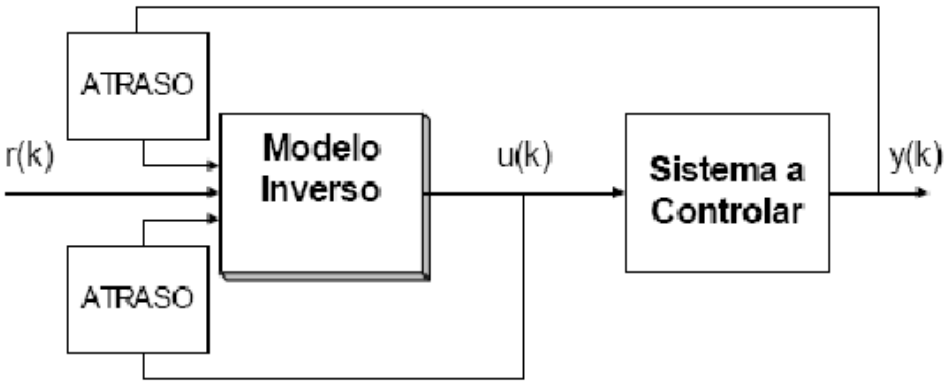


Figura 3.10: Estrutura de controlo inverso

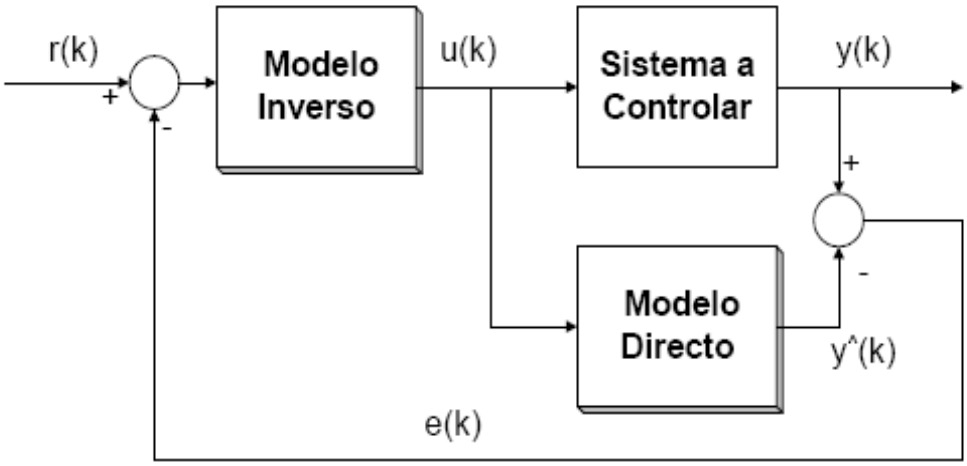


Figura 3.11: Estrutura de controlo interno

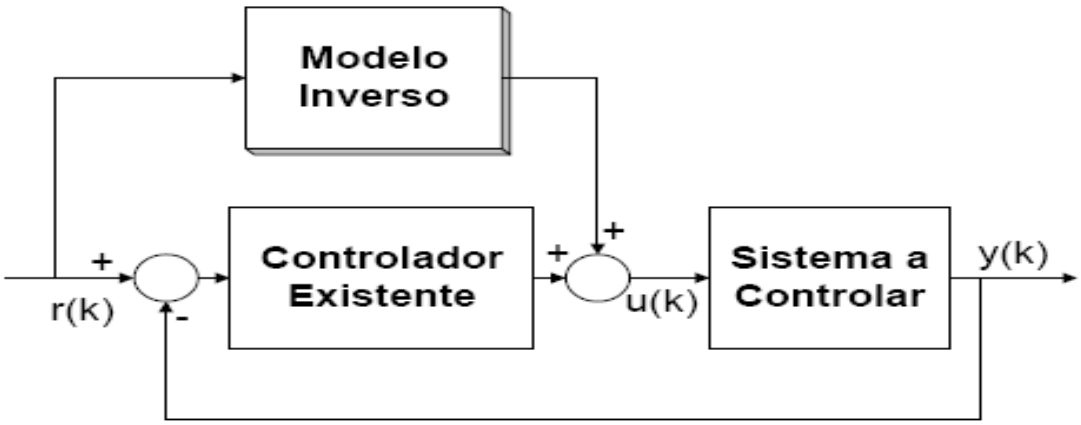


Figura 3.12: Estrutura de controlo feedforward

$$H(s)=\frac{S(s).M^{-1}(s)}{1-M^{-1}(s)*M(s)+S(s).M^{-1}(s)} \tag{3.15}$$

Caso o modelo directo e o modelo inverso, estejam bem conjugados então $M^{-1}(s)*M(s)=1$. Como Y(s)=H(s).R(s), tem-se que Y(s)=R(s), pois H(s)=1. A conclusão a retirar é que o correcta conjugação do modelo inverso e do modelo directo resulta no controlo perfeito e assegura a qualidade do controlo, reduzindo o efeito de perturbações da planta. Na prática o conjugação correcto do modelo inverso e do modelo directo é difícil de obter [F. Dias, 2005].

A estrutura de controlo feedforward da Fig. 3.12, resulta da adição em paralelo de um modelo inverso, com um controlador já existente. Este controlador já existente deve ter a capacidade de estabilizar a planta. O modelo inverso tem a função de seguir a referência com precisão, enquanto o controlador existente tem a função de estabilizar o sistema e eliminar perturbações [M. Nørgaard et al, 2000]. O controlador existente, pode ser por exemplo um PID. A forma como as ligações são efectuadas está representado na Fig. 3.13:

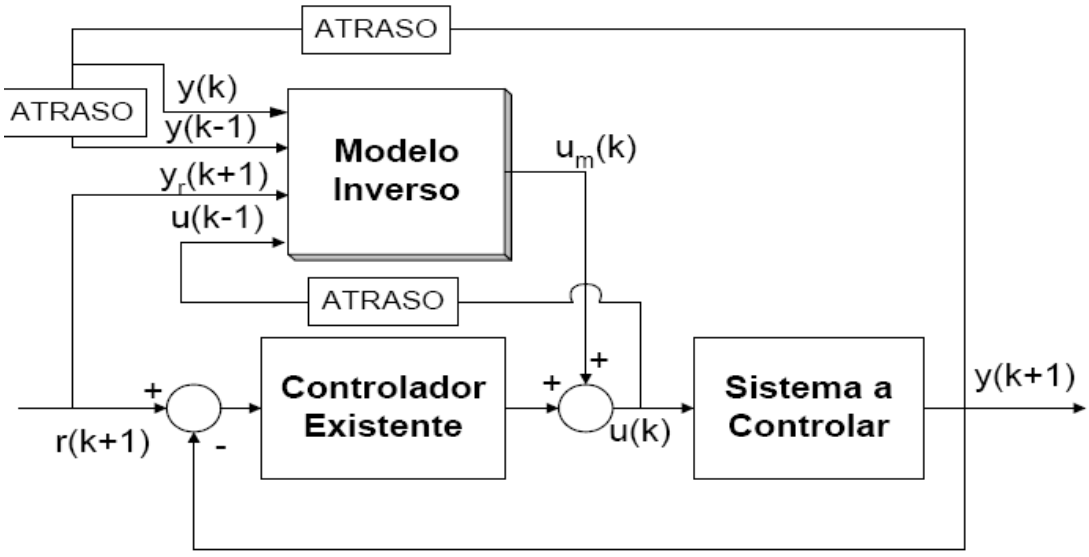


Figura 3.13: Esquema de ligações de uma estrutura de controlo feedforward

A estrutura de controlo da Fig. 3.14, denomina-se linearização por realimentação (feedback linearisation) e só faz sentido, se aplicada a sistemas não lineares, pois esta

estrutura de controlo, supõe a linearização da planta não linear, através de um sinal de controlo apropriado.

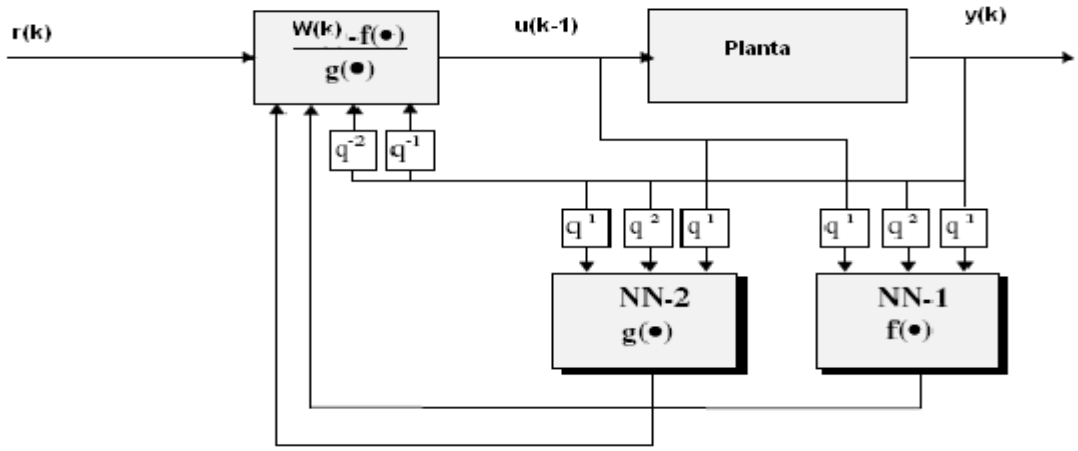


Figura 3.14: Esquema de uma estrutura de controlo de linearização por realimentação

Sabendo que a planta pode ser modelada por duas redes neuronais f e g através da Eq. 3.16:

$$y(k)=f\left[y(k-1),\ldots(y(k-n)),u(k-2),\ldots,u(k-m)\right]+u(k-1)*g\left[y(k-1),\ldots,y(k-1),u(k-2),\ldots,u(k-m)\right] \quad (3.16)$$

Se for escolhido um sinal w(k) que seja a combinação linear de saídas atrasadas, mais a referência, o sinal de controlo pode ser calculado pela Eq. 3.17 [M. Nørgaard, 2000b]:

$$u(k)=\frac{w(k)-f\left[y(k),\ldots(y(k-n-1)),u(k-1),\ldots,u(k-m+1)\right]}{g\left[y(k),\ldots(y(k-n-1)),u(k-1),\ldots,u(k-m+1)\right]} \quad (3.17)$$

4. Caso de estudo -pêndulo invertido

O pêndulo invertido é no meio científico, dedicado ao estudo do controlo de sistemas, um clássico, utilizado como "benchmark" no teste do desempenho dos vários algoritmos de controlo conhecidos da comunidade científica [Chandrasekara et al, 2004; J. Nelson, 2004]. O pêndulo invertido é constituído sobre uma base móvel sobre a qual oscila uma haste, normalmente designada por pêndulo invertido. Através do deslocamento da base ao longo de uma correia pretende-se manter a haste na posição vertical. O comportamento oscilante da haste tenta reproduzir o problema de estabilização frequente em situações típicas, tais como o controle da trajectória de um projéctil, ou do movimento de um satélite [R. Marques, 2001]. A Fig. 4.1 representa a implementação esquemática de um pêndulo invertido.

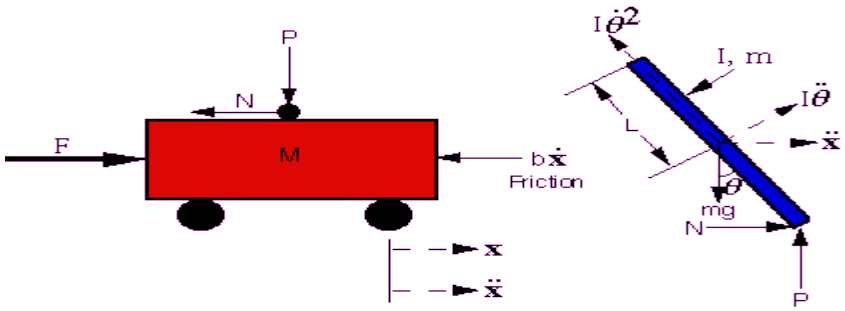


Figura 4.1: Pêndulo invertido

Tabela 4.1: Variáveis físicas do pêndulo invertido

M	Massa do carro(Kg)	1
m	Massa do pêndulo(Kg)	0.3
b	Coeficiente de fricção do carro (Kg/s)	19
L	Comprimento do pêndulo em relação ao seu centro de massa (m)	0.4
I	Momento de Inércia do pêndulo	0.006
F	Força aplicada no carro	
x	Deslocamento do carro	
θ	Ângulo do Pêndulo com a vertical	
g	Aceleração da gravidade(m/s^2)	9.8

A pertinência dos valores atribuídos às constantes físicas da tabela 4.1 pode ser conferida em [Sek Cheang, 2000].

4.1.Modelo matemático do pêndulo invertido

Seguindo a Fig. 4.1 é possível proceder à modelação matemática do sistema. Partindo das equações Newtonianas do sistema da Fig. 4.1 é possível obter a equação dinâmica do carro, descrita da seguinte forma:

$$\frac{d^2x}{dt^2}=\frac{1}{M}\sum Fx=\frac{1}{M}(F-N-b\frac{dx}{dt}) \tag{4.1}$$

A equação dinâmica do pêndulo é dada por:

$$\frac{d^2\theta}{dt^2}=\frac{1}{I}\sum \tau=\frac{1}{I}(NL\cos(\theta)+PL\sin(\theta)) \tag{4.2}$$

Em que τ representa o torque do pêndulo e Fx as forças horizontais aplicadas no carro. As equações do movimento translaccional sobre os eixos x e y do pêndulo podem ser descritas pelas Eqs. 4.3 e 4.4:

$$m\frac{d^2x_p}{dt^2}=\sum_{pend} Fp_x=N \Leftrightarrow N=m\frac{d^2x_p}{dt^2} \tag{4.3}$$

$$m\frac{d^2y_p}{dt^2}=\sum Fp_y=P-mg \Leftrightarrow P=m(\frac{d^2y_p}{dt^2}+g) \tag{4.4}$$

Sendo xp e yp funções de θ :

$$x_p=x-L\sin(\theta) \tag{4.5}$$

$$y_p=L\cos(\theta) \tag{4.6}$$

Derivando em ordem ao tempo obtêm-se as Eqs. 4.7 e 4.8:

$$\frac{dx_p}{dt}=\frac{dx}{dt}-L\cos(\theta)\frac{d\theta}{dt} \tag{4.7}$$

$$\frac{d\,y_p}{d\,t} = -L\sin(\theta)\frac{d\,\theta}{d\,t} \tag{4.8}$$

As segundas derivadas do deslocamento horizontal e vertical do pêndulo, em ordem ao tempo, são dadas respectivamente, pelas Eqs. 4.9 e 4.10 :

$$\frac{d^2 x_p}{d\,t^2} = \frac{d^2 x}{d\,t^2} + L\sin(\theta)(\frac{d\,\theta}{d\,t})^2 - L\cos(\theta)\frac{d^2\theta}{d\,t^2} \tag{4.9}$$

$$\frac{d^2 y_p}{d\,t} = -L\cos(\theta)\left(\frac{d\,\theta}{d\,t}\right)^2 - L\sin(\theta)\frac{d^2\theta}{d\,t^2} \tag{4.10}$$

Procedendo a algumas manipulações algébricas obtêm-se os sistemas de Eqs. 4.11 e 4.12:

$$\left\{\begin{array}{l} \dfrac{d^2 x}{d\,t^2} = \dfrac{1}{M}\sum Fx = \dfrac{1}{M}\left(F - m\dfrac{d^2 x_p}{d\,t^2} - b\dfrac{d\,x}{d\,t}\right) \\ \dfrac{d^2\theta}{d\,t^2} = \dfrac{1}{I}\left(m\dfrac{d^2 x_p}{d\,t^2}L\cos(\theta) + m\dfrac{d^2 y_p}{d\,t^2}L\sin(\theta)\right) \end{array}\right\} \tag{4.11}$$

$$\left\{\begin{array}{l} \dfrac{\partial^2 x}{\partial\,t^2} = \dfrac{1}{M}\sum Fx = \dfrac{1}{M}\left(F - m\left(\dfrac{\partial^2 x}{\partial\,t^2} + Lsin(\theta)\left(\dfrac{\partial\,\theta}{\partial\,t}\right)^2 - L\cos(\theta)\dfrac{\partial^2\theta}{\partial\,t^2}\right) - b\dfrac{\partial\,x}{\partial\,t}\right) \\ \qquad \dfrac{\partial^2\theta}{\partial\,t^2} = \dfrac{1}{I}\left(m\left(\dfrac{\partial^2 x}{\partial\,t^2} + Lsin(\theta)\left(\dfrac{\partial\,\theta}{\partial\,t}\right)^2 - Lcos(\theta)\dfrac{\partial^2\theta}{\partial\,t^2}\right)L\cos(\theta)\right) + \\ \qquad\qquad + m\left(-Lcos(\theta)\left(\dfrac{\partial\,\theta}{\partial\,t}\right)^2 - L\sin(\theta)\dfrac{\partial^2\theta}{\partial\,t^2} + g\right) \end{array}\right. \tag{4.12}$$

Simplificando a equação do ramo superior e inferior do sistema de Eqs. 4.12 obtém-se o sistema de Eqs. 4.13:

$$\left\{\begin{array}{c} \dfrac{d^2x}{dt^2}=\dfrac{F-mL\sin\theta\left(\dfrac{d\theta}{dt}\right)^2+mL\cos\theta\dfrac{d^2\theta}{dt^2}-b\dfrac{dx}{dt}}{M+m} \\ \dfrac{d^2\theta}{dt^2}=\dfrac{m\dfrac{d^2x}{dt^2}L\cos\theta+gmL\sin\theta}{I+mL^2} \end{array}\right\} \tag{4.13}$$

Se $\sin\theta\approx\theta$ para um valor de θ pequeno e $\cos\theta\approx 1$, podemos eliminar todos os termos com não linearidades resultando na Eq. 4.14.

$$\left\{\begin{array}{c} \dfrac{d^2x}{dt^2}=\dfrac{F+mL\dfrac{d^2\theta}{dt^2}-b\dfrac{dx}{dt}}{M+m} \\ \dfrac{d^2\theta}{dt^2}=\dfrac{gmL\theta+m\dfrac{d^2x}{dt^2}L}{I+mL^2} \end{array}\right\} \tag{4.14}$$

Resolvendo o sistema de Eqs. 4.14 para o domínio de Laplace, partindo da equação do ramo inferior do sistema de Eqs. 4.14 obtém-se a Eq. 4.15 :

$$X(s)=\left[\frac{\left(I+mL^2\right)}{mL}-\frac{g}{s^2}\right]\theta(s) \tag{4.15}$$

Substituindo X(s) na equação do ramo superior do sistema de Eqs. 4.14, entretanto passado para o domínio de Laplace, obtém-se a Eq. 4.16, que nos dá a descrição matemática do pêndulo invertido linear, no domínio de Laplace:

$$\frac{\theta(s)}{F(s)}=\frac{s\dfrac{mL}{q}}{s^3+b\dfrac{\left(I+mL^2\right)}{q}s^2-\dfrac{(M+m)mgL}{q}s-\dfrac{bmgL}{q}s} \tag{4.16}$$

em que:

$$q=\left[(M+m)\left(I+mL^2\right)-(mL)^2\right] \tag{4.17}$$

4.2. Modelos implementados em Simulink

O Simulink é uma toolbox, que faz parte da plataforma computacional do Matlab e é muito utilizada na simulação de sistemas, lineares e não lineares. Por este motivo, foi aquela que se utilizou no decorrer desta tese. Nas Figs. 4.2 e 4.3 mostram-se os modelos lineares e não-lineares do pêndulo invertido, implementados em Simulink, em que L=l. O modelo linear do pêndulo invertido implementado em simulink é dado a partir da Eq. 4.14.

O modelo do pêndulo invertido não-linear implementado em simulink, está representado na Fig. 4.3 e é retirado directamente da Eq. 4.12.

As Figs. 4.2 e 4.3 mostram que os pêndulos são modelados com uma entrada, que representa a força aplicada no carro e quatro saídas, que são precisamente o deslocamento horizontal do carro, a velocidade do mesmo, o ângulo que o pêndulo faz com a horizontal e a velocidade angular. A saída que se pretende controlar, é a saída theta.

4.3. O pêndulo invertido linear

4.3.1. Identificação do pêndulo linear

4.3.1.1. Aquisição de dados

Um pêndulo real é um sistema instável, assim como não linear. Procedendo à sua linearização o pêndulo invertido linear, continua a ser um sistema instável, com dois pontos de equilíbrio, em $\theta=0$ e $\theta=\pi$ radianos. Para que seja possível treinar uma rede neuronal, de forma a identificar e controlar uma planta instável é necessário primeiro estabilizar a planta. Só assim é possível recolher os exemplos de treino, em torno do ponto de equilíbrio desejado. O ponto de equilíbrio desejado é $\theta=0$ rad. Nesta fase de identificação os primeiros passos são a escolha entre um sinal de malha aberta ou um sinal de malha fechada e a escolha da frequência de amostragem. Para o caso do pêndulo invertido linear, como este é um sistema instável, é forçoso escolher um sinal de malha fechada. A frequência de amostragem escolhida foi 200 Hz. Foi escolhida a abordagem em que a período de amostragem deve ser inferior a 1/(30*LB) (eq.3.2) [University of Michigan], em que LB é a largura de banda do sistema em malha fechada em rad/s. Esta condição é relevante porque garante que conseguimos amostrar o tempo de subida do sistema em malha fechada na resposta ao degrau, pelo menos 10 vezes [Karl Åström, 1997]. Uma frequência de amostragem elevada é desejável para o controlo com uma rede neuronal, na medida em que produz sinais de controlo suaves

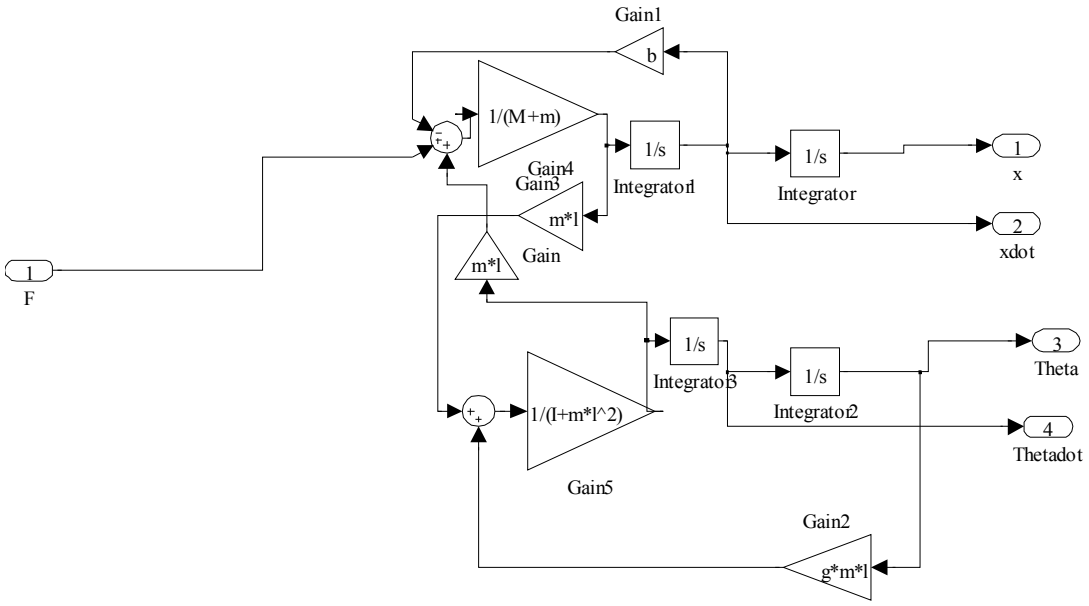


Figura 4.2: Modelo linear do pêndulo invertido implementado em Simulink

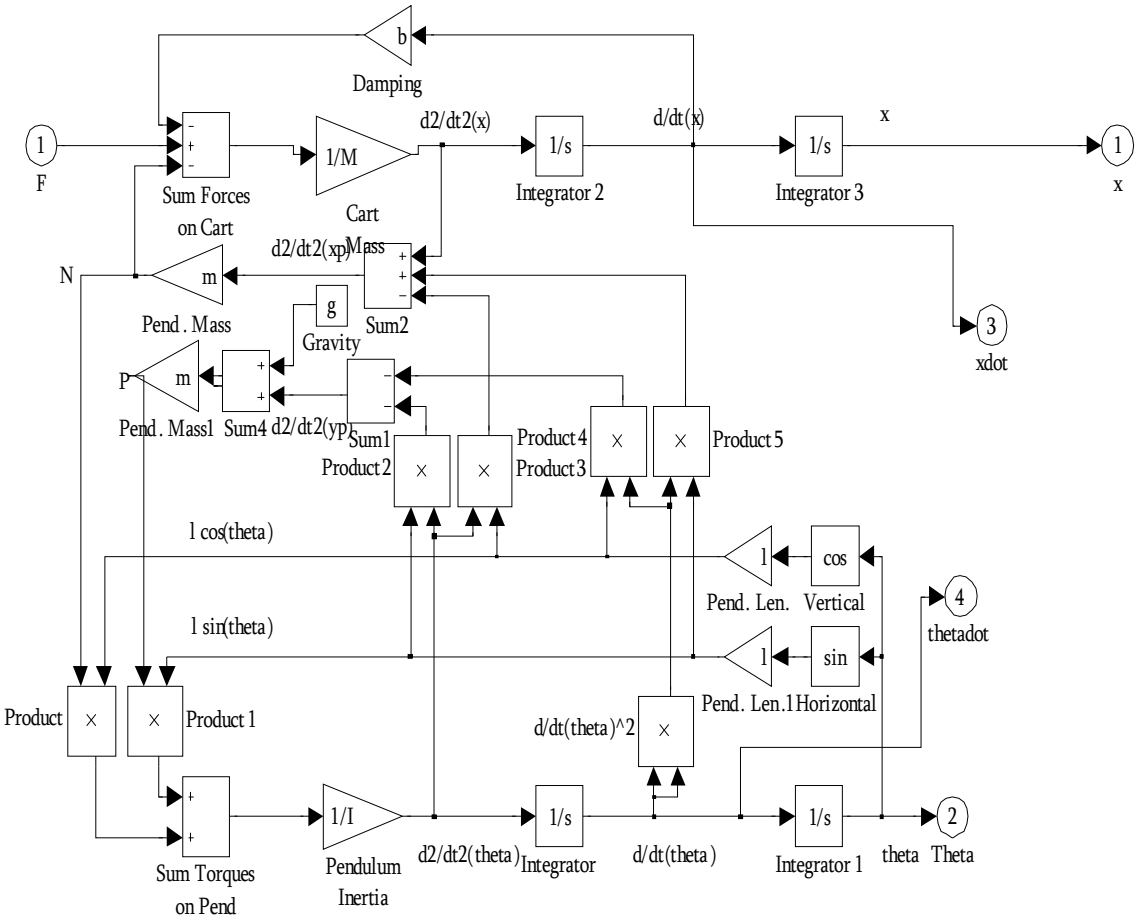


Figura 4.3: Modelo não linear do pêndulo invertido implementado em Simulink

e permite seguir a referência com rapidez. Uma frequência de amostragem demasiado elevada não é desejável, devido ao problema de mau condicionamento numérico[M. Nørgaard et al, 2000] .Na Fig. 4.4 está representado o esquema implementado em Simulink, para a recolha dos exemplos de treino.

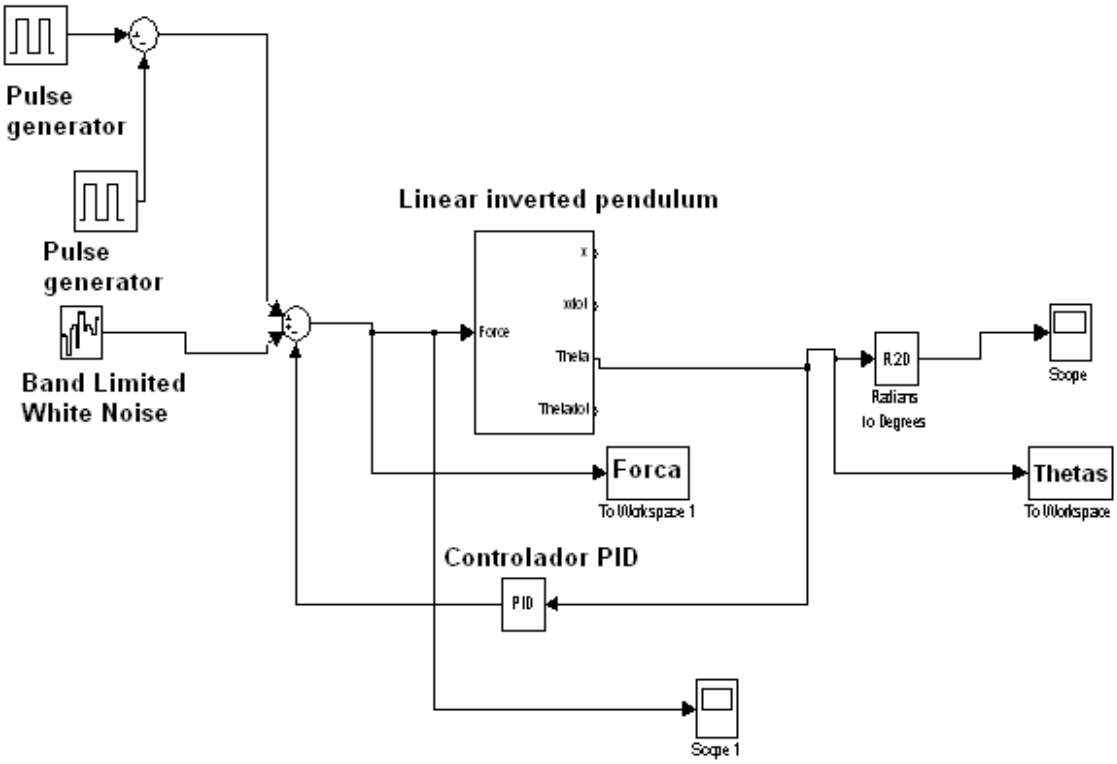


Figura 4.4: esquema em Simulink do sistema de aquisição de dados

É necessário sublinhar que os exemplos de treino, devem ser retirados da entrada e da saída da planta conforme é representado na Fig. 4.4, através dos blocos forca e thetas. Também é necessário sublinhar que a frequência de amostragem deve ser explicitamente declarada em cada bloco "thetas", "forca" e no gerador de ruído branco, caso contrário corre-se o risco de obter uma quantidade de valores obtidos pelo bloco thetas, diferente da quantidade de valores obtidos pelo bloco forca . Para o caso do pêndulo linearizado, foi utilizado uma potência de ruído de 0.1W/Hz(W/Hz é a unidade da grandeza física densidade espectral de potência). Este valor é definido na fonte de ruído branco.

Dado que o sistema é linear é possível estabilizar o sistema com um PID. O ajuste dos parâmetros do PID obedece a alguns preceitos. Para um pêndulo invertido linearizado é suficiente e desejável utilizar um PID não ajustado(ou detuned do inglês) [Tim Callinan,2003; M. Nørgaard et al,2000]. Utilizar um PID ajustado, implicaria obter dados para treinar a rede com demasiada influência da dinâmica do controlador. O

método para ajustar os parâmetros P,I,D, consiste em atribuir valores iniciais aos parâmetros P,I,D e ajustá-los com a ferramenta do matlab sisotool até que se consiga atingir o limite de estabilidade para o sistema. Os parâmetros Kp, Ki e Kd utilizados foram:

$$Kp = 16.73\,;\; Ki = 199.7\,;\; Kd = 0.2349$$

A Fig. 4.5 representa o diagrama de bode do sistema da Fig 4.4 em malha fechada.

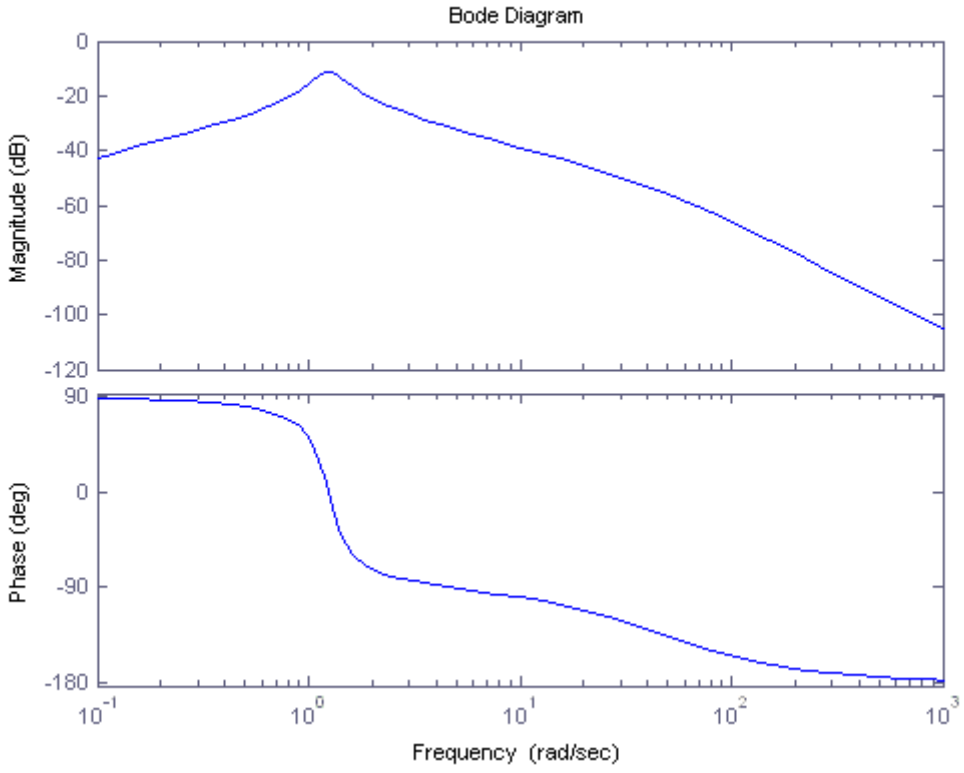


Figura 4.5: diagrama de bode do sistema em malha fechada

Analisando a Fig. 4.5, é possível chegar à conclusão, que a largura de banda útil em malha fechada do sistema é cerca de 2 rad/s. O período de amostragem inferior a 1/(30*LB) é cumprido. A Fig. 4.6 representa o lugar das raízes do sistema pêndulo invertido linear, em malha fechada. A partir desta figura é possível perceber a forma como foram ajustados os parâmetros do PID de forma a cumprir os preceitos anteriormente discutidos.

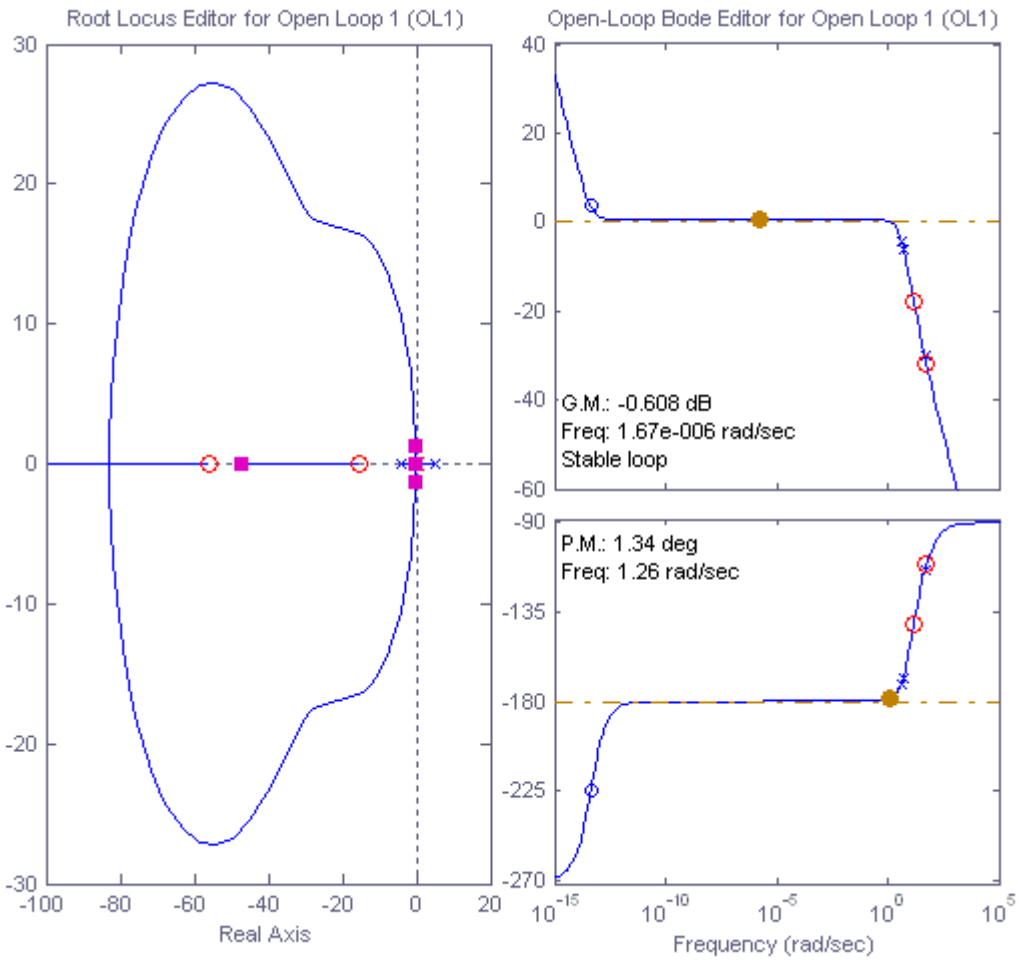


Figura 4.6: lugar das raízes dos sistema em malha fechada obtido com o sisotool

4.3.1.2. Selecção da estrutura do modelo

Nesta dissertação foi feito o estudo de três estruturas de RNAs: rede unidireccional de uma só camada escondida, rede de Elman e redes recorrentes, pertencentes às classes de modelos não lineares a NNARX e a NNARMAX.

O estudo começa por tentar avaliar, qual o número óptimo de neurónios na camada escondida, qual o algoritmo de treino mais eficiente e ao mesmo tempo tenta encontrar a estrutura mais apropriada de entre as 4 estruturas aqui estudadas. Para tentar perceber qual o número óptimo de neurónios na camada escondida e o algoritmo mais eficiente começou-se o estudo, treinando as duas redes mais simples, a rede unidireccional de camada única e a rede de Elman. Todos os algoritmos de treino em estudo foram testados nestas duas estruturas de RNAs.

Foram testados três algoritmos de treino : o algoritmo Gauss-Newton, o algoritmo Levenberg-Marquardt e o algoritmo Gradiente Descendente, para o treino das duas

primeiras estruturas. Para reproduzir as Figs. 4.7 a 4.12 é necessário utilizar a função *performance*, disponível em anexo, que chama as funções *treino1* e *treino2elm*, também disponíveis em anexo, para o treino de redes unidireccionais e de Elman, respectivamente. Os dados (5000 amostras) foram divididos em 75% , dados de treino e 25% dados de validação. O nº de neurónios da camada escondida foi progressivamente aumentado. A rede produz por isso diferentes valores de MSE (Eq. 2.3) .

As Figs. 4.7 a 4.9 representam a variação do MSE com o número de neurónios da camada escondida, quando a estrutura da RNA em estudo, é uma rede unidireccional de camada única. Os resultados demonstram que os algoritmos Levenberg-Marquardt e Gauss-Newton obtêm equivalente desempenho, enquanto o algoritmo Gradiente Descendente é o que obtém piores resultados. É de salientar a rápida convergência do algoritmo Levenberg-Marquardt. Este é mais rápido que os outros dois algoritmos em avaliação, que necessitam de várias iterações para atingir os MSEs especificados. Dez iterações foram suficientes, para o algoritmo Levenberg-Marquardt atingir o desempenho especificado.

As Figs. 4.10 a 4.12 representam a variação do MSE com o número de neurónios da camada escondida, quando a estrutura da RNA em estudo é uma rede de Elman. Foram testados os algoritmos Gauss-Newton, Levenberg-Marquardt e o Gradiente Descendente, para treinar a rede com estrutura de Elman. O treino da rede com esta estrutura, consome muito tempo e não produz resultados significativamente melhores que a rede unidireccional de camada única, como pode ser observado pelos gráficos. Verifica-se que aumentar o número de neurónios em qualquer estrutura, seja esta unidireccional de camada única ou Elman, não produz significativas melhorias.

Dado que não existe significativa diferença entre as performances obtidas, para os diferentes números de neurónios da camada escondida, no estudo da rede unidireccional de camada única (a estrutura interna de uma rede das classes de modelos NNARX e NNARMAX é unidireccional) foram escolhidos 5 neurónios, para a camada escondida das redes das classes de modelos NNARX e NNARMAX. Elege-se também o algoritmo Levenberg-Marquardt para o treino das redes com esta estrutura. O algoritmo Levenberg-Marquardt demonstrou ser de mais rápida convergência.

O primeiro passo consiste em determinar o número de regressores, ou seja o número de entradas e saídas atrasadas que devem ser colocados na entrada da rede neuronal de modo a proceder a uma identificação precisa do modelo. Um número de regressores demasiado pequeno implica que a dinâmica essencial da planta não irá ser modelada. Um número de regressores demasiado elevado também acarreta problemas, pois a rede produzirá informação redundante [M. Nørgaard et al, 2000] .

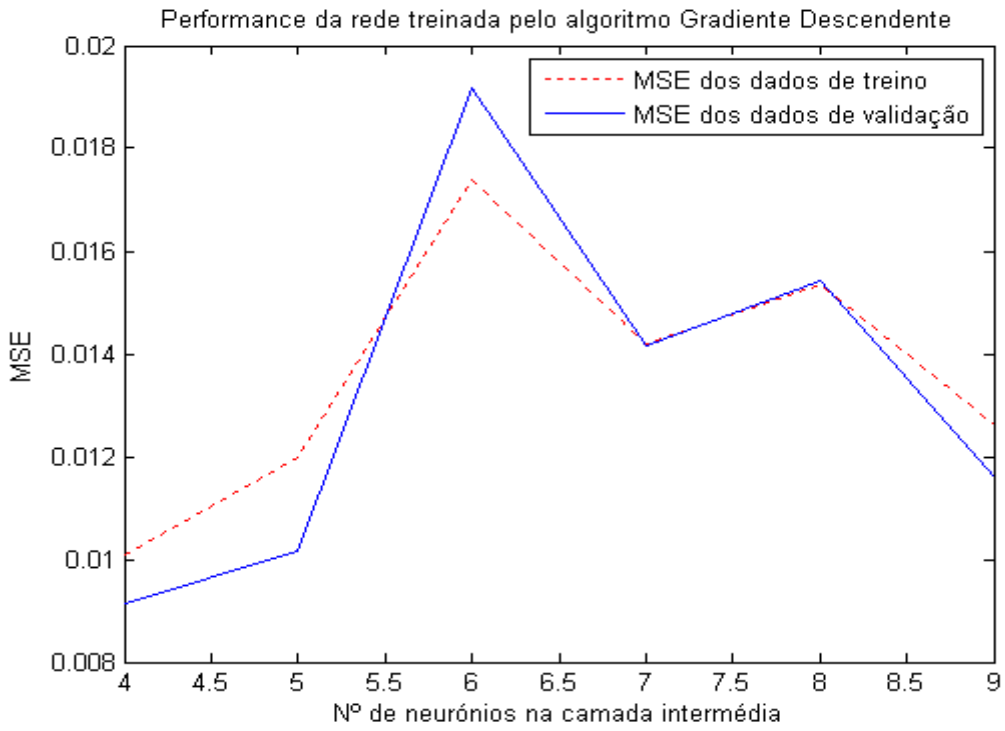


Figura 4.7: Desempenho da rede unidireccional treinada pelo algoritmo Gradiente Descendente

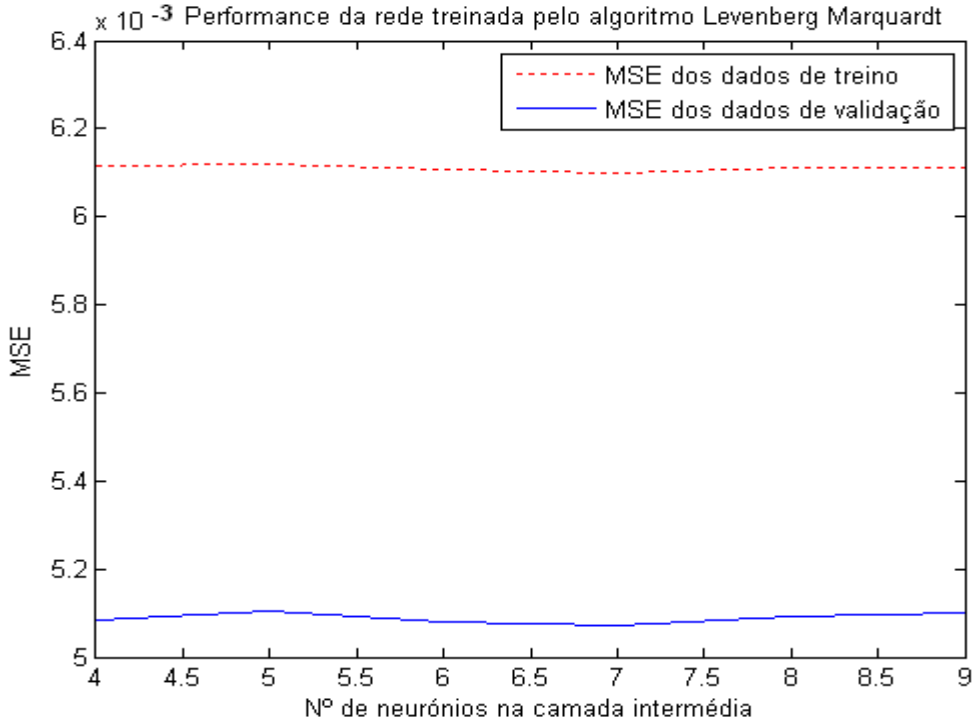


Figura 4.8: Desempenho da rede unidireccional treinada pelo algoritmo Levenberg-Marquardt

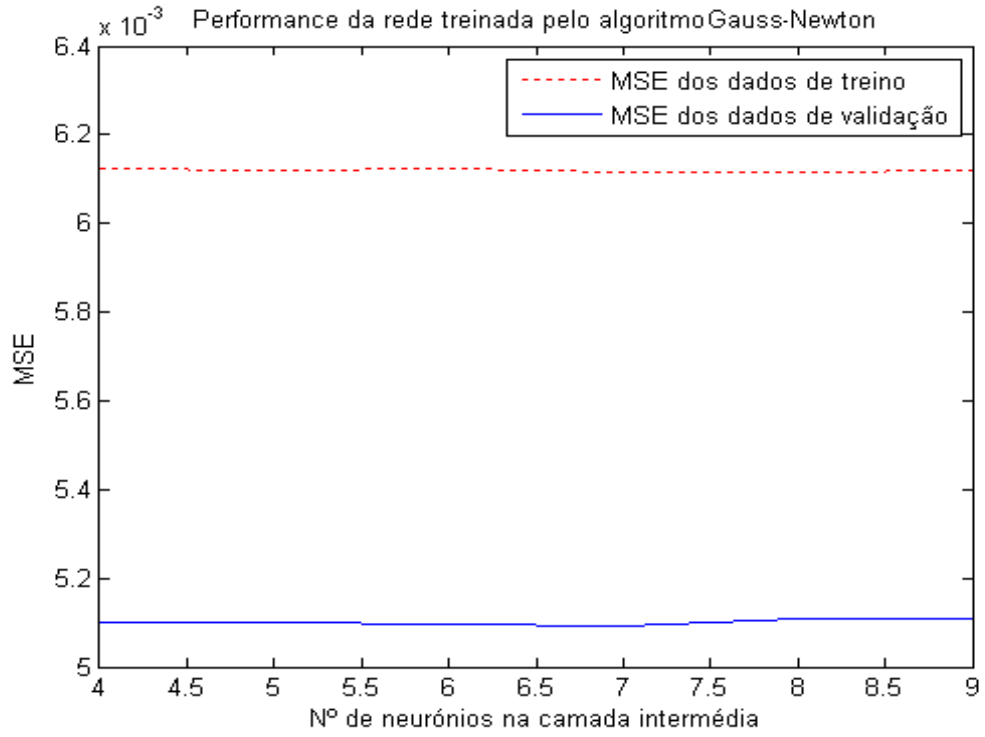


Figura 4.9: Desempenho da rede unidireccional treinada pelo algoritmo Gauss-Newton

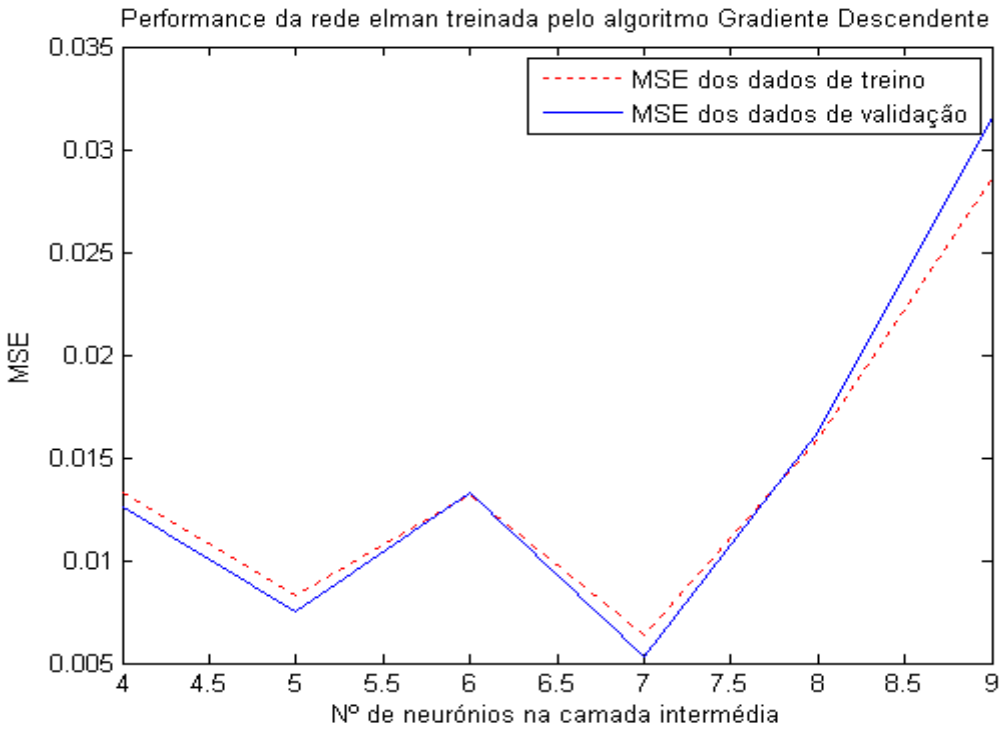


Figura 4.10: Desempenho da rede de Elman treinada pelo algoritmo gradiente descendente

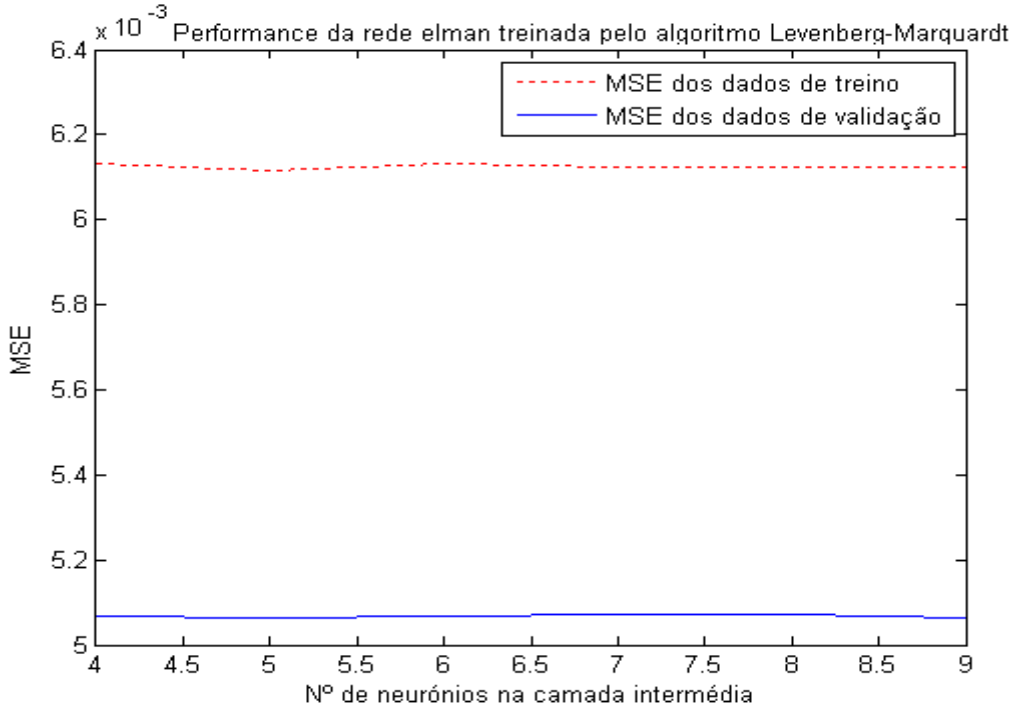


Figura 4.11: Desempenho da rede de Elman treinada pelo algoritmo Levenberg-Marquardt

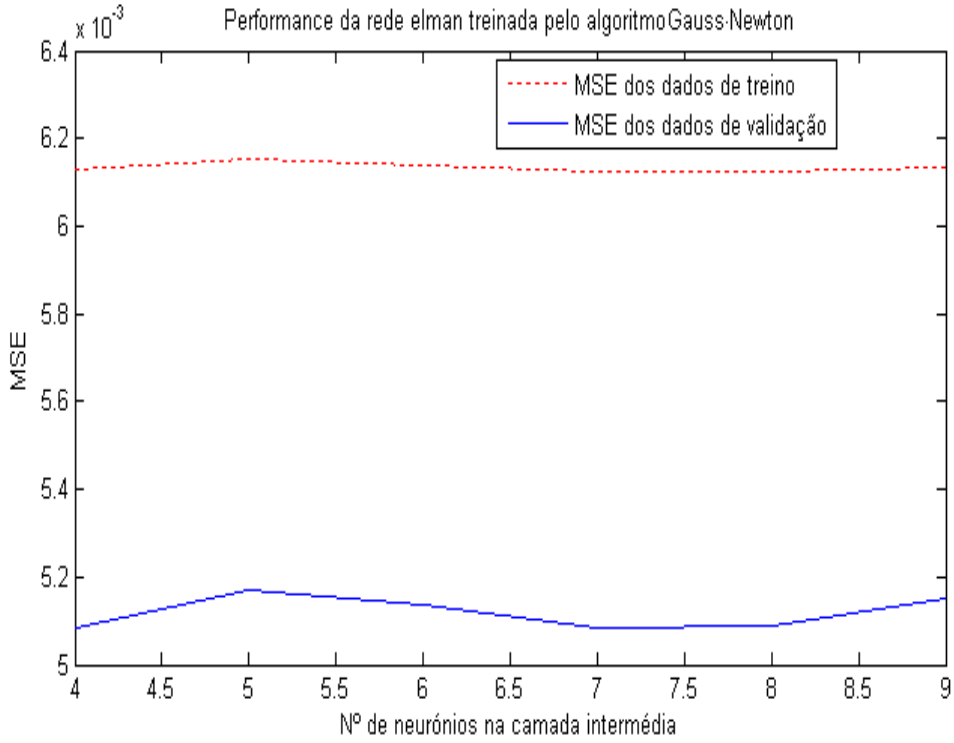


Figura 4.12: Desempenho da rede Elman treinada pelo algoritmo Gauss-Newton

Foi utilizado um pacote de software , que permite fazer a identificação e controlo de sistemas através de RNAs. Este pacote de software é gratuito e pode ser encontrado na internet. Este está subdividido em ferramentas para identificação(NNSYSID toolbox) e ferramentas para controlo(NNCTRL toolbox) [M. Nørgaard,2000a; M. Nørgaard,2000b].

Com a função *lipschit da NNSYSID toolbox* , podemos determinar quantas entradas e saídas atrasadas da planta devem ser colocadas na entrada da rede neuronal, de forma a identificar a planta de forma precisa. Há um ficheiro disponível em anexo, com o nome "*preprodata.m*" que escala os dados e determina o número de regressores. Os dados devem estar previamente no workspace do Matlab.

É necessário introduzir o conceito de quocientes de Lipschitz [M. Nørgaard et al,2000]. Dado um sistema descrito por um modelo NNARX (a determinação do número de regressores óptimo é válido também para outras classes de modelos), este é descrito, pelas Eqs. 4.18 e 4.19:

$$y(k)=g_0[\varphi(k),\theta] \tag{4.18}$$

$$\varphi^T(k)=[y(k-1)\dots y(k-n),u(k-d)\dots u(k-d-m)] \tag{4.19}$$

t=1,...,N em que N é a quantidade de exemplos de treino. Para todas as combinações de pares entrada-saída, os quocientes de Lipschitz são dados pela seguinte expressão:

$$q_{ij}=\left|\frac{y(k_i)-y(k_j)}{\varphi(k_i)-\varphi(k_j)}\right|,\quad i\neq j \tag{4.20}$$

Em que | | é a norma euclidiana. Por simplicidade escolhe-se um espaço de atrasos em que n=m. Nestas condições para um determinado espaço de atrasos calcula-se e escolhe-se os p maiores quocientes de Lipschitz. p é um valor entre 0.01.N e 0.02.N. Posteriormente avalia-se o critério da Eq. 4.21:

$$q=\left(\prod_{i=1}^{p}\sqrt{n}\,q_i^{\,n}\right)^{\frac{1}{p}} \tag{4.21}$$

Na Fig. 4.13 é possível observar o critério da Eq. 4.21 em função do número de atrasos da entrada e da saída da planta, ou seja do espaço de atrasos (lag space).

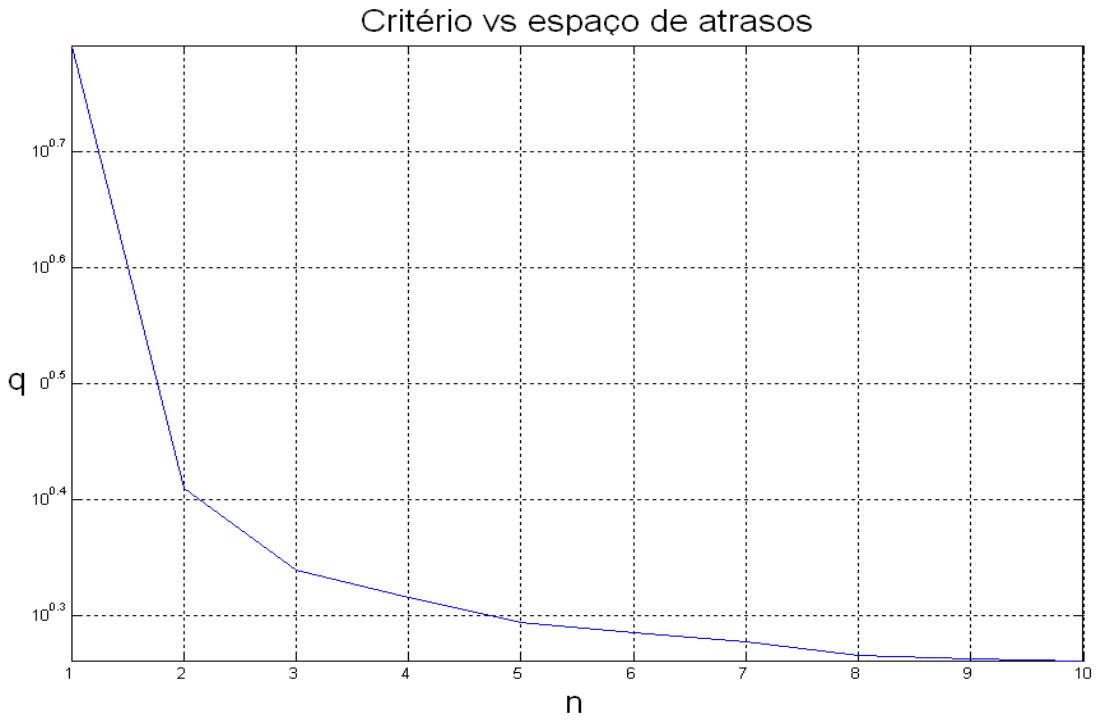


Figura 4.13: Critério vs espaço de atrasos

Para determinar o número de regressores óptimo é necessário encontrar na Fig. 4.13 o "joelho" do traçado. Este é o lugar onde aumentar o número de regressores não gera significativa diminuição do critério. O número de regressores está relacionado com a ordem da planta. A Fig. 4.13 indica que a planta é de 3ª ordem. A rede necessita de três entradas atrasadas e de três saídas atrasadas da planta , para que a rede neuronal, produza um modelo preciso. A Eq. 4.16 demonstra que a planta de facto é de 3ª ordem. Esta equivalência entre o número de regressores e a ordem da planta, só acontece se a saída da planta não vier afectada de ruído. O vector de regressores é descrito pela Eq. 4.22:

$$\wp(k)=[y(k-1),y(k-2),y(k-3),u(k-1),u(k-2),u(k-3)] \qquad (4.22)$$

Nas Figs. 4.14 e 4.15 estão representados os gráficos do ângulo que o pêndulo invertido faz com a vertical versus o tempo em amostras. Os gráficos apresentam 2 traçados. Um é a representação dos valores obtidos à saída da rede neuronal pertencente à classe de modelos NNARMAX. O outro é a a representação dos valores recolhidos directamente da planta. Utilizou-se a função *simulnarmax1*, disponível em anexo, que utiliza a função *narmax1* da NNSYSID toolbox. Foram usados 3 regressores do erro entre a saída da planta e a saída da rede neuronal e 3 regressores da entrada e da saída

da planta (Eq. 3.6). As amostras foram divididas em dados de treino e validação, numa proporção de 75% e 25%.

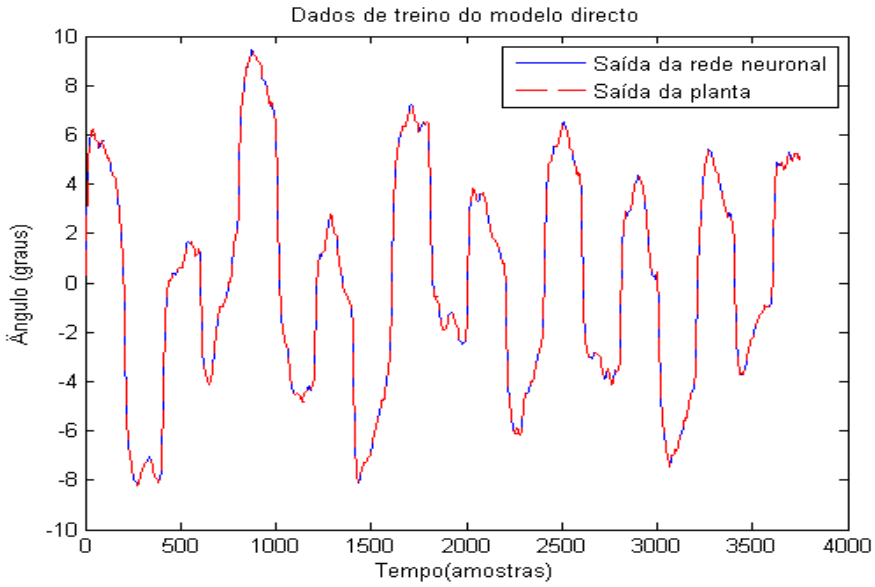


Figura 4.14: Dados de treino da classe de Modelos NNARMAX

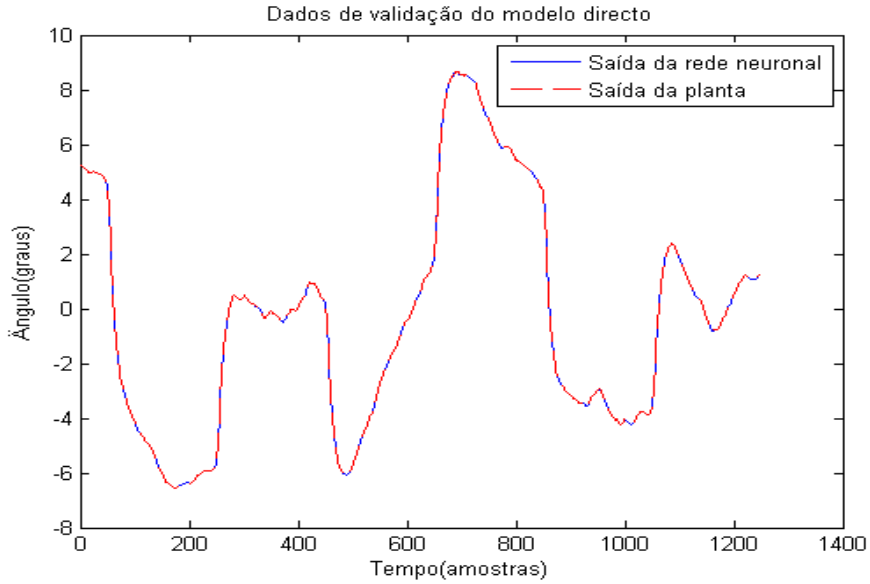


Figura 4.15: Dados de validação da classe de modelos NNARMAX

As Figs. 4.14 e 4.15 demonstram que a classe de modelos NNARMAX, gerou uma aproximação muito precisa, entre os dados da saída da RNA e os exemplos de treino.

O estudo efectuado nas Figs. 4.14. e 4.15, para a classe de modelos NNARMAX é repetido nas Figs. 416 e 4.17 para a classe de modelos NNARX.

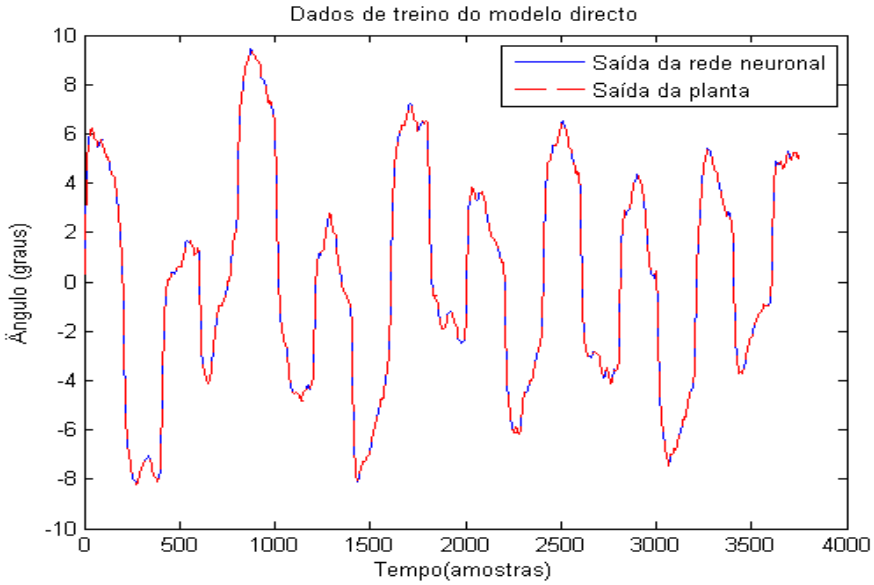


Figura 4.16: Dados de treino da classe de modelos NNARX

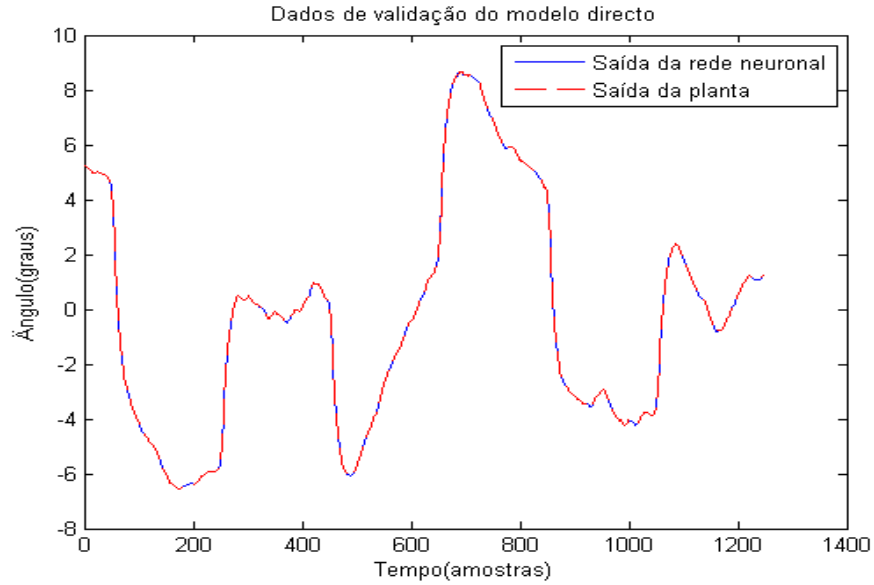


Figura 4.17: Dados de validação da classe de modelos NNARX

A tabela 4.2 mostra os valores do MSE, para os dados de validação e para os dados de treino, do modelo directo da planta assim como o melhor resultado de treino e de validação para a rede unidireccional de camada única e de Elman, entre os três

algoritmos de treino estudados. Pode observar-se a partir da tabela, que a estrutura de redes recorrentes pertencentes à classe de modelos NNARX ou NNARMAX, produz melhores resultados que a rede de Elman ou a rede unidireccional de camada única e são por isso as mais adequadas para a modelação de sistemas dinâmicos.

Tabela 4.2: MSE das estruturas de modelos

	Modelo directo da planta	
	Treino(75%)	Validação(25%)
NNARX	6.51e-012	8.277e-012
NNARMAX	1.632e-010	4.114e-009
Unidireccional	6.1e-003	5.07e-003
Elman	6.114e-003	5.062e-003

Para escolher a melhor classe de modelos, entre as classes de modelos NNARMAX e NNARX deve-se escolher aquela que gerar menor MSE [M. Nørgaard et al,2000]. A tabela 4.2 demonstra que a classe que melhor obedece a este critério é a NNARX. Será esta a usada para o caso do pêndulo invertido linearizado.

Para controlar o pêndulo invertido é necessário identificar o modelo inverso do mesmo, que não é mais do que construir uma RNA, que aceita valores de ângulo como entrada e responde com valores de força na saída. A classe de modelos do modelo inverso da planta, deverá ser a mesma que a escolhida para o modelo directo[M. Nørgaard et al, 2000]. Foi usada a função *general* pertencente à NNCTRL toolbox, que treina um modelo inverso da planta, numa estrutura de treino genérica. As Figs. 4.18 a 4.21, mostram os resultados do processo de identificação do modelo inverso da planta. A tabela 4.3 mostra os valores de MSE obtidos para o modelo inverso da planta. O MSE de treino e de validação são muito próximos, o que representa um critério de aprendizagem e generalização correcto da rede.

Tabela 4.3: MSE do modelo inverso da planta

	Modelo inverso da planta	
	Treino(75%)	Validação(25%)
NNARX	1.97e-005	4.04e-005

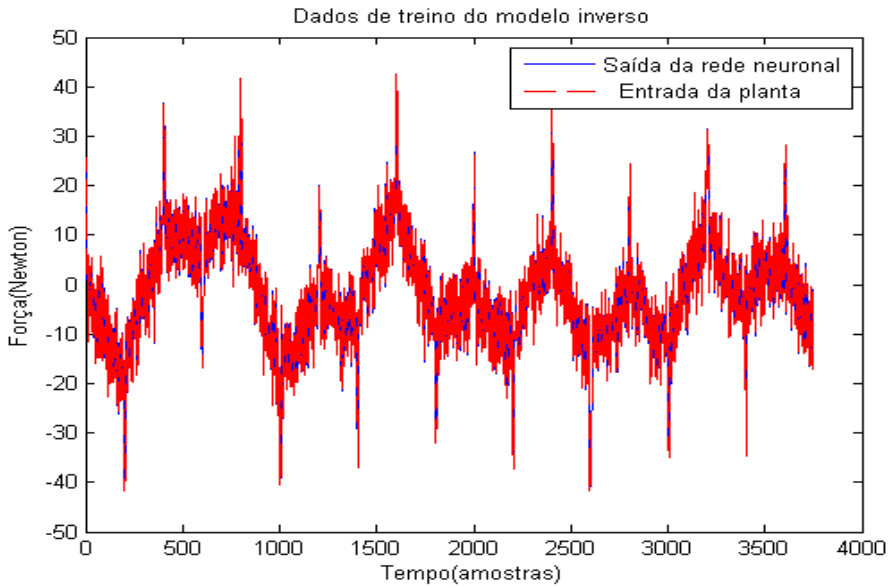


Figura 4.18: Dados de treino do modelo inverso da planta

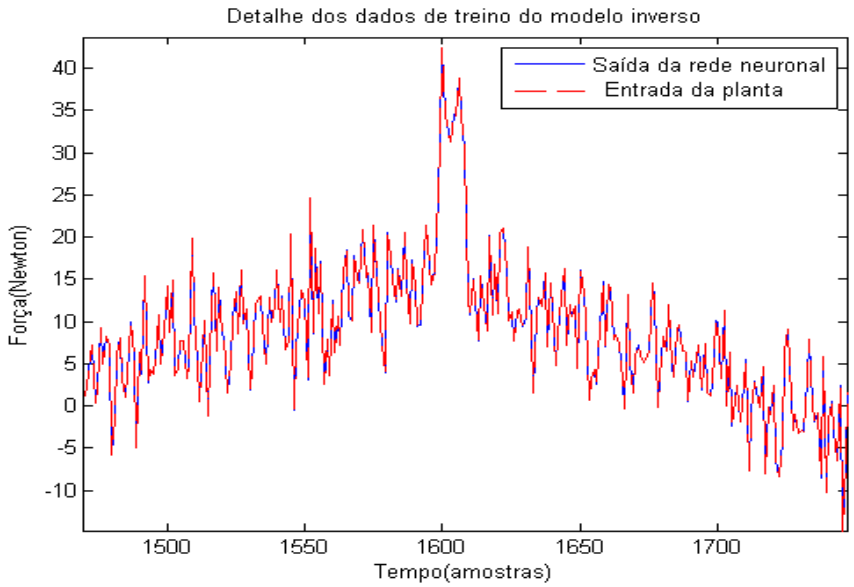


Figura 4.19: Detalhe dos dados de treino do modelo inverso

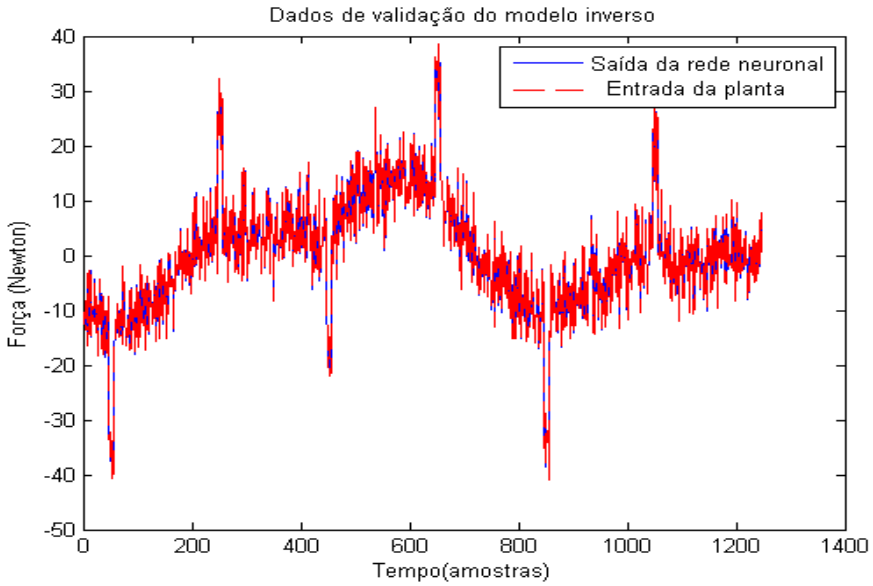


Figura 4.20: Dados da validação do modelo inverso da planta

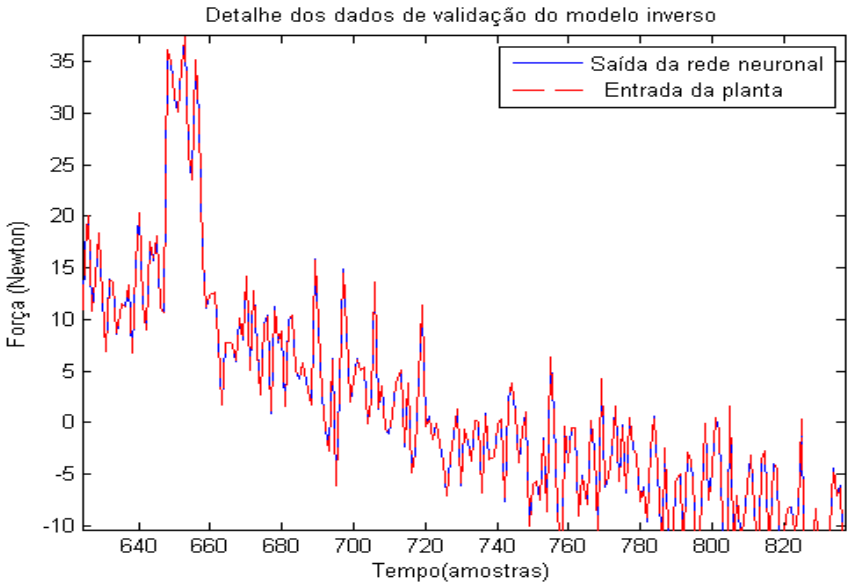


Figura 4.21: Detalhe dos dados da validação do modelo inverso da planta

As Figs. 4.18 a 4.21 demonstram, uma grande proximidade entre os traçados das curvas dos valores à saída da RNA e à entrada da planta, o que indica uma boa identificação do modelo inverso da planta. A tabela 4.3 reforça essa ideia, demonstrando baixos valores de MSE.

4.3.2. Controlo do pêndulo invertido linear

Foram treinados 3 modelos inversos da planta. A diferença entre estes está relacionada com a estrutura de treino e com a função de custo utilizadas. Os 3 modelos são o modelo inverso genérico(secção 3.3.2.1), modelo inverso especializado(secção 3.3.2.2) e modelo óptimo(secção 3.3.2.3). O modelo inverso genérico, foi treinado pela função "general" da NNCTRL toolbox. Este modelo inverso é treinado off-line. O "modelo inverso especializado", e o "modelo óptimo" são treinados on-line, pelas funções special2 e opttrain, respectivamente, do pacote de software NNCTRL toolbox.

Algumas considerações têm que ser feitas, sobre como conseguir treinar modelos inversos especializados e modelos óptimos. Existe um parâmetro nas funções *invinit2.m* e *optrinit.m*, denominado "trparms", que é um vector linha com duas colunas. Estas funções fazem a inicialização de parâmetros importantes, nas funções de treino especializado e óptimo, *special2* e *opttrain*, respectivamente. O manual da NNCTRL toolbox, não apresenta nenhuma regra empírica, para fazer a sintonia deste parâmetro trparms, mas pelas experiências realizadas, chegou-se à conclusão que um valor maior que 1 para o 1º elemento do vector, não traz grandes vantagens e é mais lento a convergir, para uma solução óptima. Um valor menor, mas próximo de 1 deverá ser usado. O 2º elemento do vector produz alterações mínimas, na maior parte das vezes, inobserváveis. A função *optrinit* tem outro parâmetro denominado *"rho"*. Este serve para penalizar sinais quadrados no sinal de controlo(u) da planta, fazendo com que o sinal de controlo seja mais suave e menor. As experiências demonstraram, que este parâmetro *"rho"* deve ser ajustado progressivamente, começando por um valor baixo, até encontrar uma resposta, que seja satisfatória.

A NNCTRL toolbox contém uma função própria para cada estrutura de controlo. É aconselhável, traduzir as redes obtidas com esta toolbox, para Simulink. Isto é feito essencialmente por duas razões:

1. A toolbox só oferece os resultados gráficos do controlo, como a saída da estrutura de controlo e o sinal de controlo. Não nos permite observar o sistema a funcionar em tempo real, nem aceder a outras variáveis que se pretendem medir.
2. O Simulink é uma ferramenta credível na simulação de processos reais. Isto permite verificar se os resultados obtidos com a NNCTRL toolbox são credíveis.

Relativamente ao ponto 2, pelas variadas experiências, que foram realizadas ao longo desta dissertação, sempre que se verificava, pela NNCTRL Toolbox, que o pêndulo invertido estava controlado, obtia-se idêntico resultado com o Simulink. As Figs. 4.22 a 4.24, representam as estruturas de controlo utilizadas para o estudo do pêndulo invertido linear.

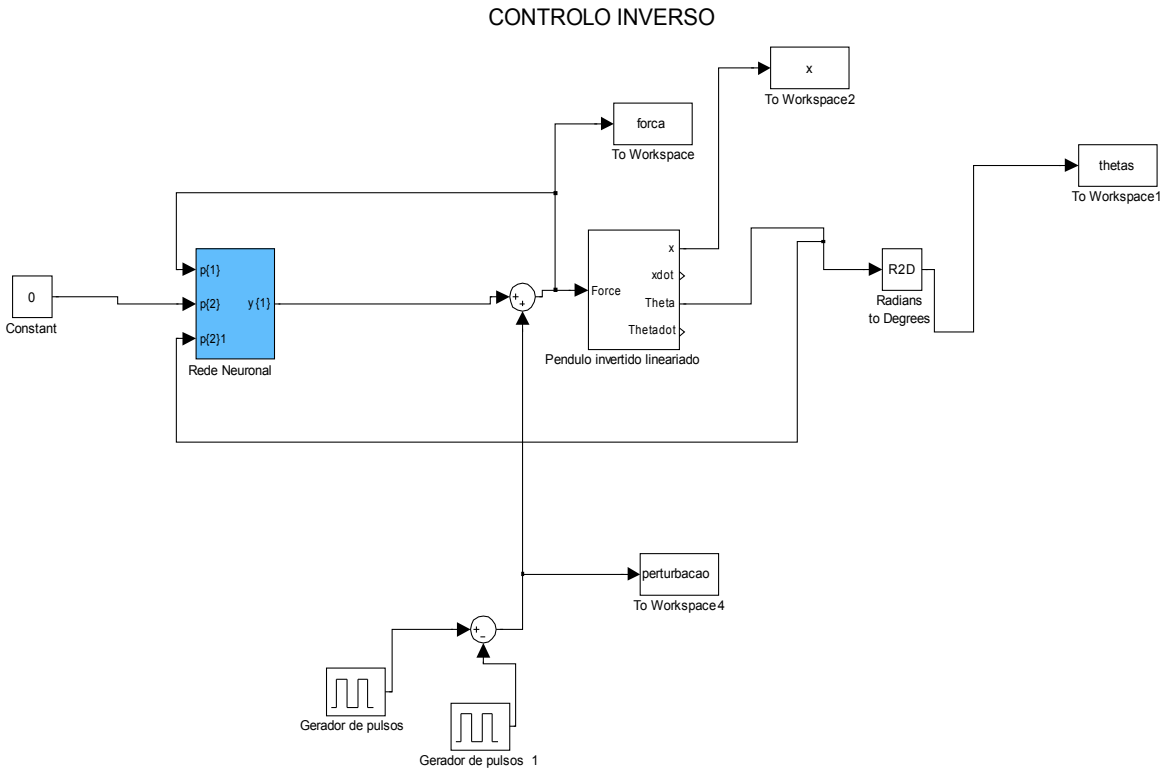


Figura 4.22: Esquema simulink estrutura de controlo inverso

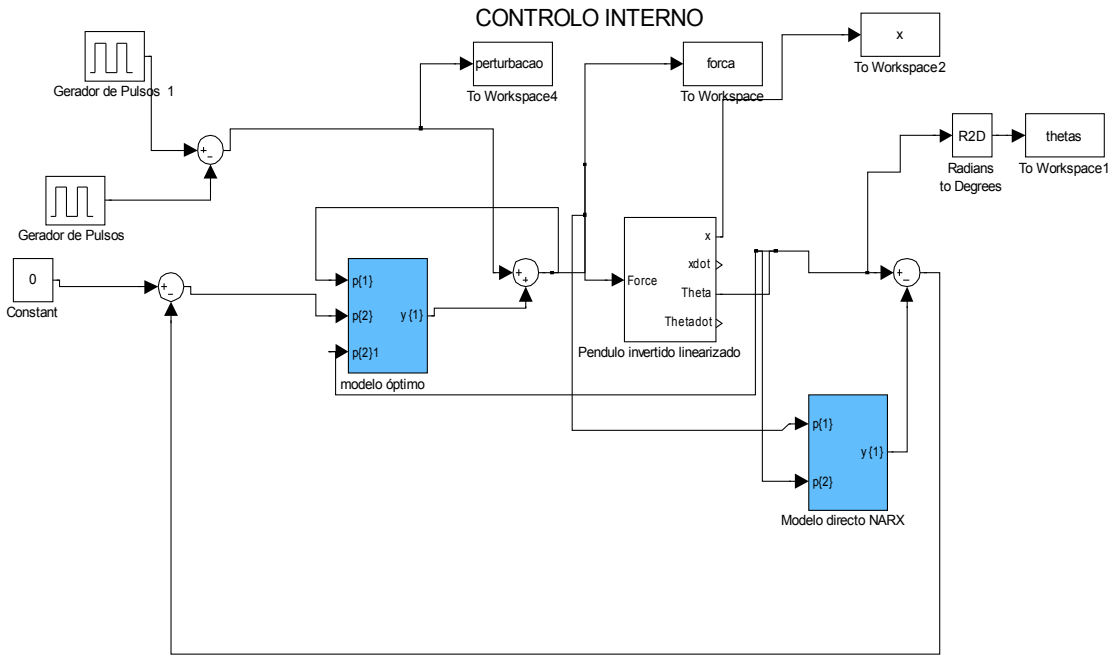


Figura 4.23: Esquema simulink da estrutura de controlo interno

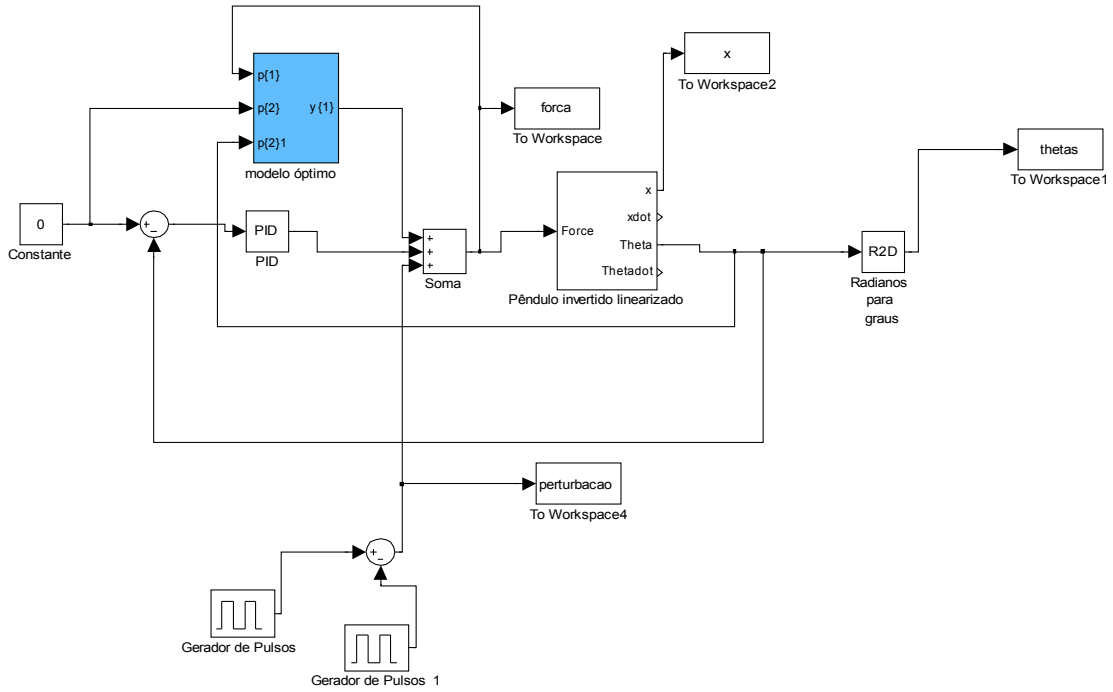


Figura 4.24: Esquema simulink da estrutura de controlo feedforward

Para fazer a tradução das redes obtidas com a NNCTRL toolbox para Simulink, foi criada a função *inverso_directo* disponível em anexo. Esta função tanto devolve um modelo directo, como um modelo inverso em Simulink. Antes de utilizar esta função, é necessário fazer algumas alterações no código do ficheiro matlab, que é especificamente alterar o nome do modelo Simulink, para aquele, com o qual se fez a aquisição de dados e alterar um dos parâmetros da função gensim da Neural network toolbox, que define a frequência de amostragem da rede neuronal, que vai ser criada. Este valor deve ser alterado para a frequência de amostragem escolhida na fase de aquisição de dados.

Depois de criada o modelo Simulink da RNA com a função *inverso_directo* é necessário fazer algumas adaptações. Estas estão relacionadas com a relação de entradas e saídas entre o modelo directo e o modelo inverso. Na Fig. 4.25 pode ser observado o modelo directo de uma planta, de 2ªordem. O modelo inverso correspondente está representado na Fig. 4.26. Na Fig. 4.27 como o y(k+1), é um valor indisponível, porque é um valor futuro, este é substituído pela referência.

A função inverso-directo já devolve o modelo Simulink com a relação de entradas e saídas correctas. Para o caso em que se tem um modelo directo com 2 regressores da entrada da planta e 2 regressores da saída da planta, a função inverso_directo devolve 1 regressor na entrada e 3 regressores na saída para o modelo inverso. No entanto o

modelo inverso só apresenta duas entradas disponíveis, como é possível ver na Fig. 4.28. Para que a RNA do modelo inverso da planta tenha disponível uma terceira entrada, para a referência é necessário seguir alguns passos. Estes são mostrados detalhadamente nas Figs. 4.29 a 4.33.

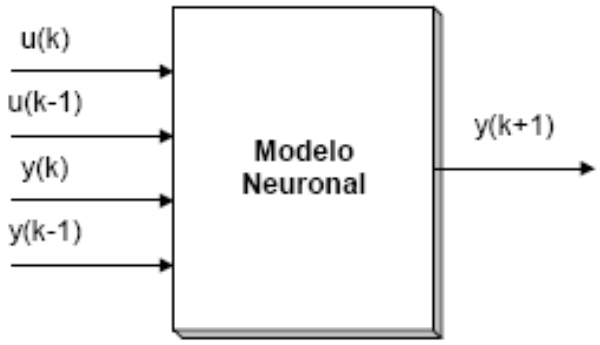


Figura 4.25: Modelo neuronal directo da planta

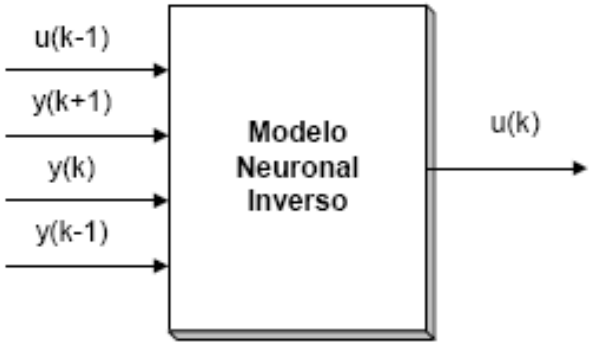


Figura 4.26: Modelo inverso correspondente

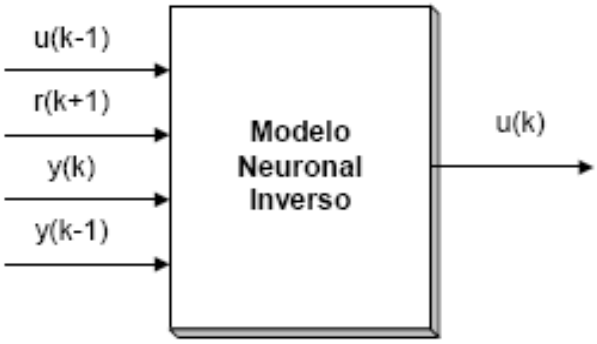


Figura 4.27: Modelo inverso correspondente com y(k+1) substituído por r(k+1)

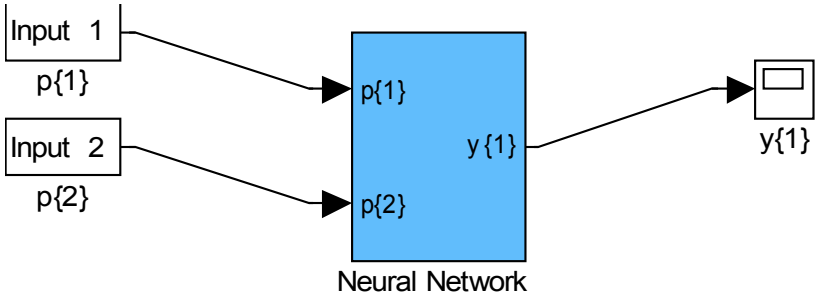


Figura 4.28: Modelo inverso da planta implementado em Simulink

Fazendo duplo-clique sobre a RNA da Fig. 4.28, aparece a estrutura interna da rede. A camada escondida, está representada na parte superior da Fig. 4.29 e a camada de saída na parte inferior.

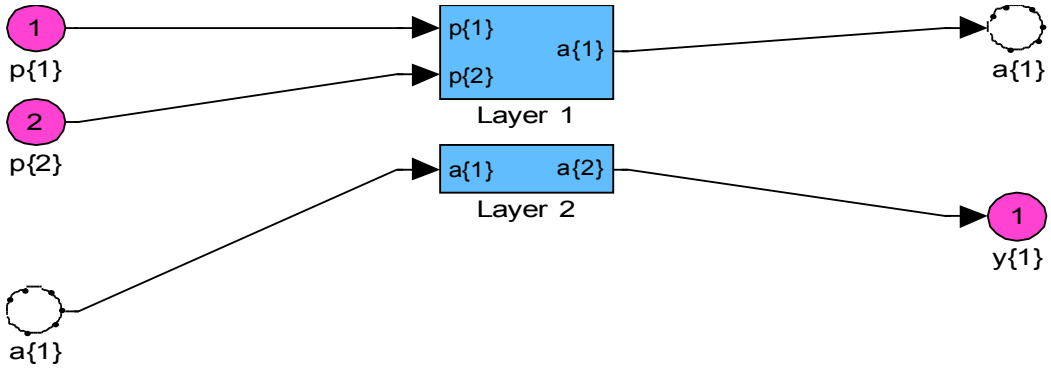


Figura 4.29: Estrutura interna de uma rede neuronal em Simulink

Fazendo duplo-clique sobre a camada escondida, aparece a estrutura da Fig. 4.30, com dois tap-delays(TDL), que são blocos que contêm os atrasos introduzidos, nas duas entradas da RNA. A entrada 2 da Fig. 4.30, é aquela que diz respeito aos valores de ângulo do pêndulo, que são realimentados da saída da planta e introduzidos na RNA.

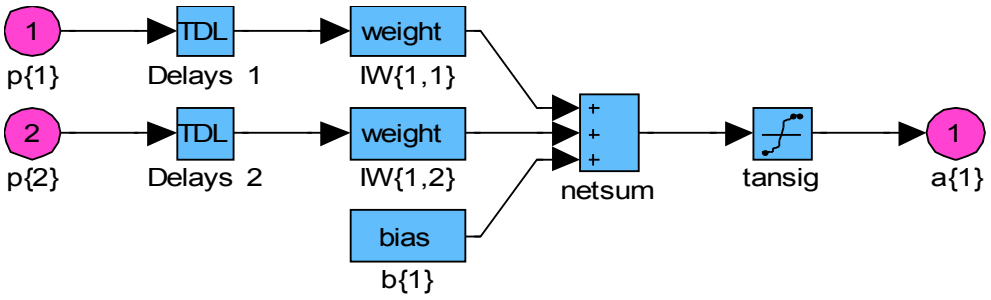


Figura 4.30: Esquema Simulink de uma rede neuronal

Fazendo duplo clique sobre o tap-delay da entrada 2 obtém-se a estrutura da Fig. 4.31:

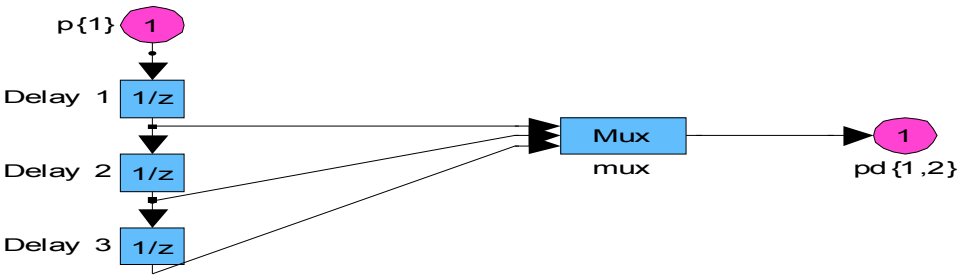


Figura 4.31: Esquema Simulink de uma rede neuronal

É necessário alterar a estrutura da Fig. 4.31 de modo a obter a estrutura da Fig. 4.32. A entrada 1 no diagrama da Fig 4.32, é aquela onde vai ser inserida a referência. A entrada 2 é aquela onde vão ser ligadas as saídas da planta. Após mais algumas alterações intuitivas, obtém-se a rede com a 3ª entrada disponível, na Fig. 4.33. A entrada do meio p{2} é onde será inserida a referência.

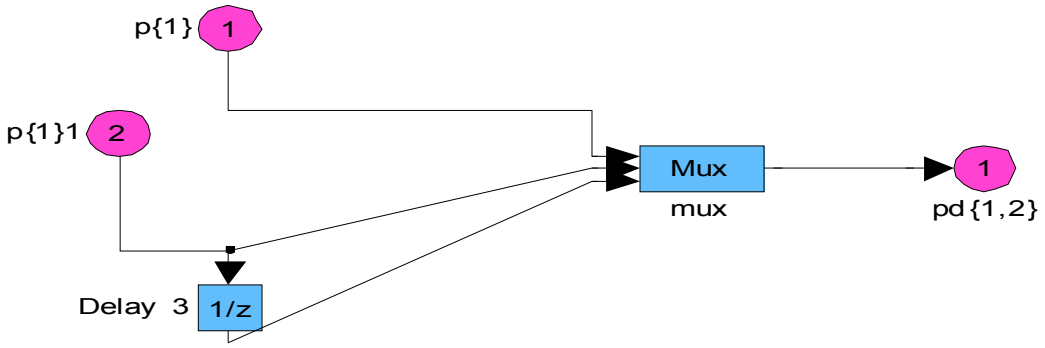


Figura 4.32: Esquema Simulink de uma rede neuronal

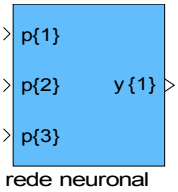


Figura 4.33: Esquema Simulink de uma rede neuronal

4.3.2.1.Estrutura de controlo inverso

Na Fig. 4.34 são amostrados os resultados do controlo de uma planta controlada por um modelo inverso genérico, inserido numa estrutura de controlo inverso. O modelo neuronal inverso é treinado off-line, através da estrutura de treino da Fig. 3.8.

Os 4 gráficos dizem respeito às variáveis mostradas nos diagramas Simulink, correspondentes às estruturas de controlo, das Figs. 4.22 a 4.24, na disposição mostrada a seguir:

$$\begin{bmatrix} thetas & forca \end{bmatrix}$$
$$\begin{bmatrix} perturbacaox \end{bmatrix}$$

A perturbação é um impulso com 1000 N de amplitude de muito curta duração, cerca de 8 ms.

A Fig. 4.35 diz respeito ao controlo de uma planta pelo modelo inverso especializado, numa estrutura de controlo inverso. O modelo neuronal inverso foi treinado on-line através da estrutura de treino da Fig. 3.9 e função de custo 3.11. Após o treino o modelo neuronal inverso treinado na estrutura de treino da fig 3.9, é inserido na estrutura de controlo da Fig. 4.22 .

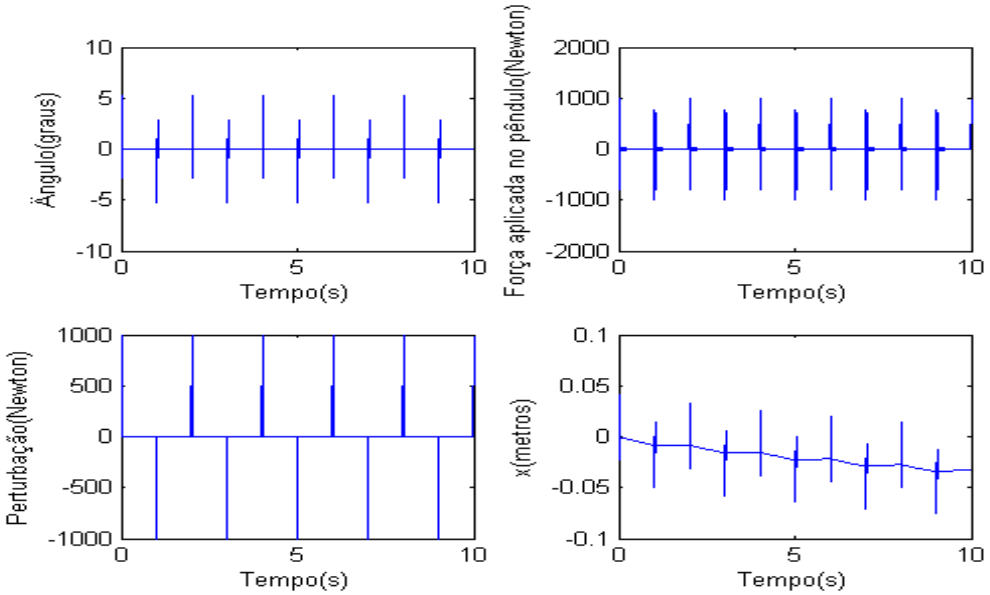


Figura 4.34: Resultados de controlo com o modelo inverso genérico inserido numa estrutura de controlo inverso

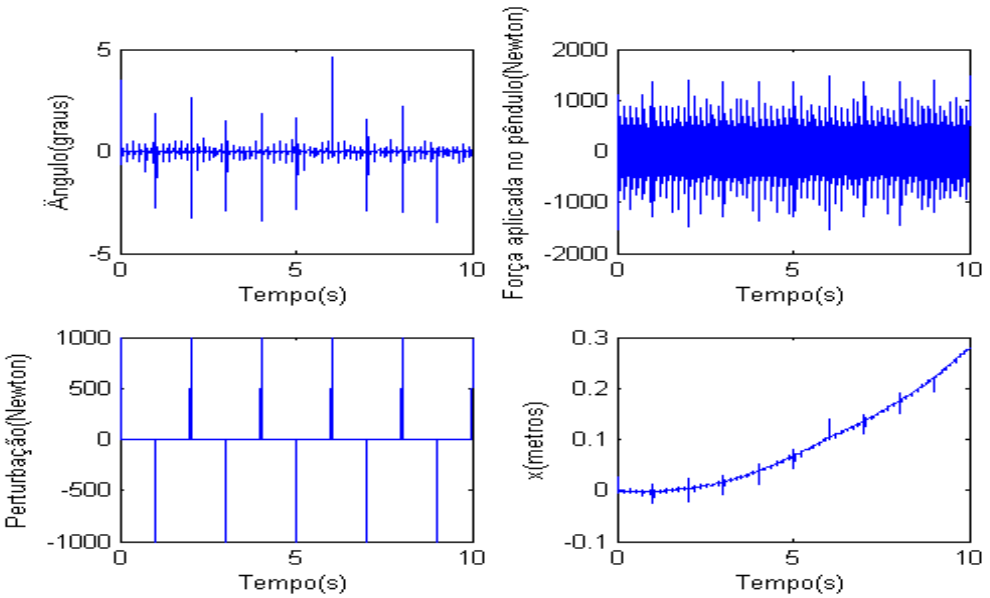


Figura 4.35: Resultados de controlo com o modelo inverso especializado inserido numa estrutura de controlo inverso

A Fig. 4.36 diz respeito ao controlo de uma planta pelo modelo óptimo, numa estrutura de controlo inverso. O modelo neuronal inverso foi treinado on-line através da estrutura de treino da Fig. 3.10 e função de custo 3.12. Após o treino o modelo neuronal inverso treinado na estrutura de treino da fig 3.9, é inserido na estrutura de controlo da Fig. 4.22 .

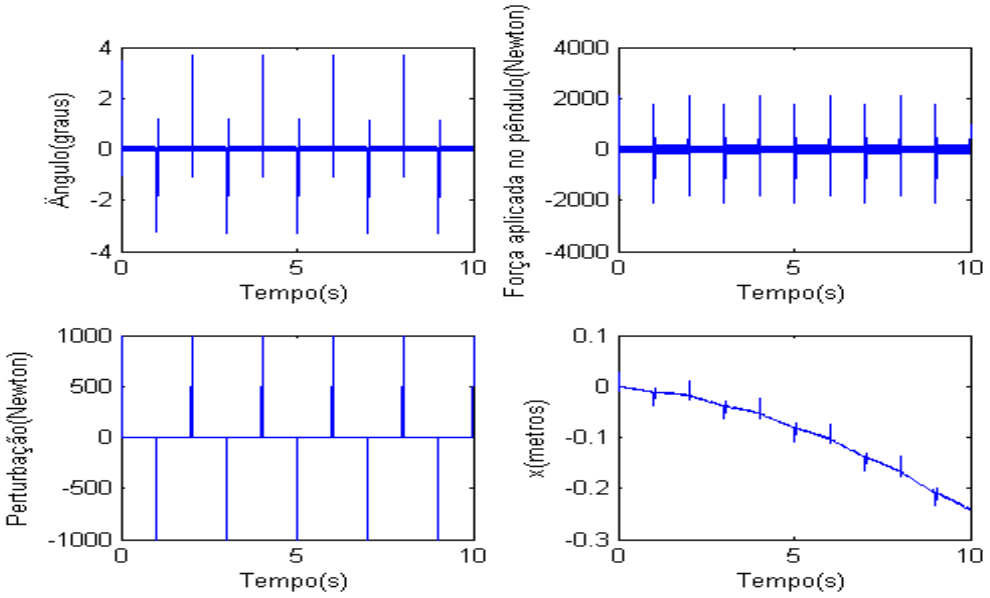


Figura 4.36: Resultados do modelo óptimo inserido numa estrutura de controlo inverso

O modelo óptimo e o modelo inverso genérico, parecem apresentar melhores resultados de controlo, ao nível da força aplicada no pêndulo, pois têm sinais mais suaves. A vantagem dos modelos óptimo e especializado, está, como se pode concluir pelas Figs. 4.34 a 4.36 e pelo valor do MSE apresentado na tabela 4.4, na maior rapidez com que o pêndulo retorna à posição de equilíbrio desejada em $\theta = 0^{\circ}$. O facto de o modelo óptimo, apresentar um sinal de força, mais suave que o modelo inverso especializado e um melhor resultado de controlo (avaliado pelo valor do MSE da tabela 4.4) que o modelo inverso genérico, faz com que o modelo óptimo, seja a melhor solução de compromisso.

4.3.2.2.Estrutura de controlo interno

As Figs. 4.37 a 4.39 apresentam os resultados de controlo para uma estrutura de controlo interno. Nesta estrutura de controlo é necessário simultaneamente duas RNAs-modelo directo e modelo inverso. O modelo directo foi treinado segundo a estrutura da Fig. 3.7. Para o treino do modelo inverso foram implementadas as estruturas das Figs. 3.9 e 3.10 com diferentes funções de custo(eq.3.10, 3.11 e 3.12)

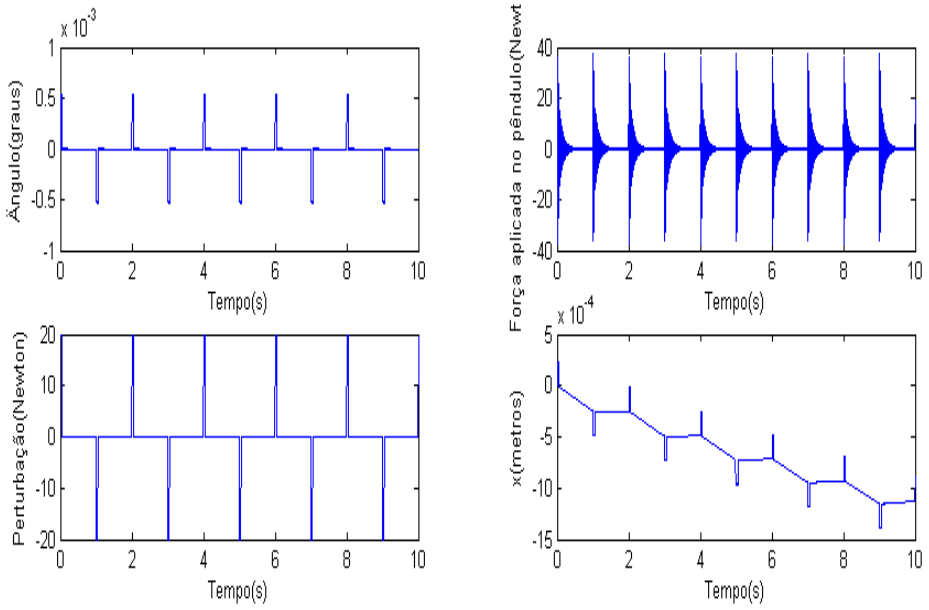


Figura 4.37: Resultados de controlo com o modelo inverso genérico inserido numa estrutura de controlo interno

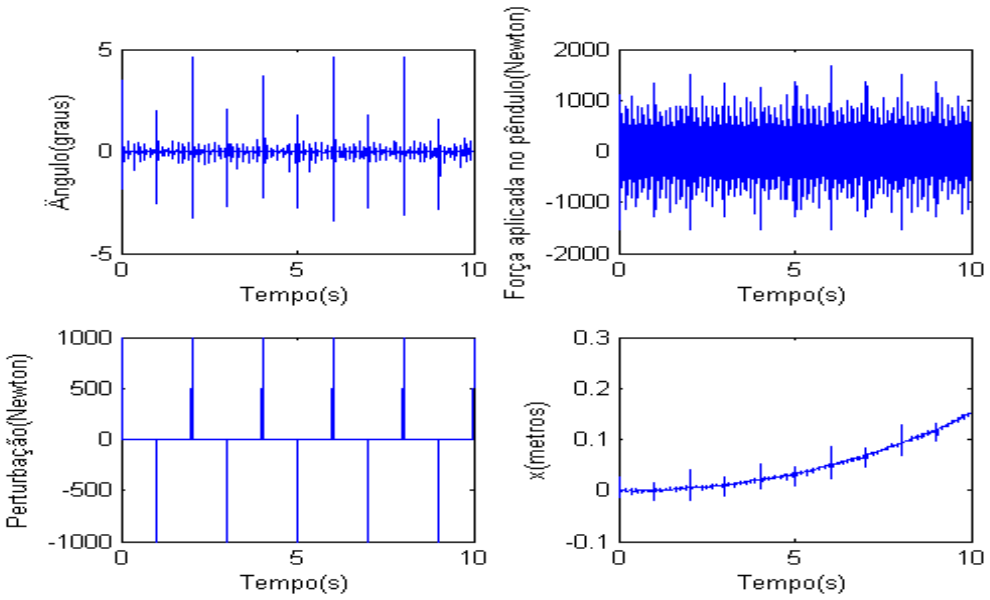


Figura 4.38: Resultados de controlo com o modelo inverso especializado inserido numa estrutura de controlo interno

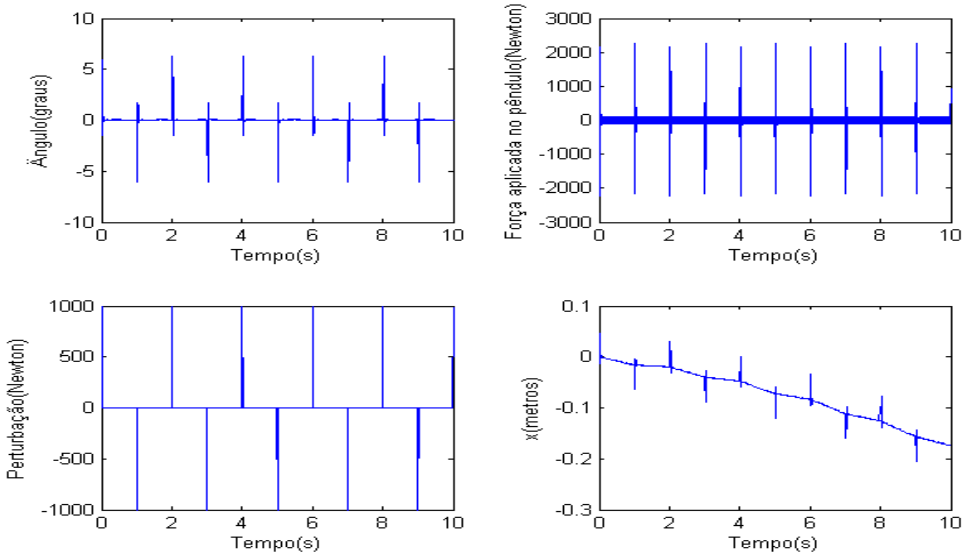


Figura 4.39: Resultados de controlo com o modelo óptimo inserido numa estrutura de controlo interno

Os resultados da estrutura de controlo interno, são muito similares aos obtidos, com a estrutura de controlo inverso não existindo significativas diferenças. O traçado do ângulo que o pêndulo invertido faz com a vertical é mais suave, com o modelo óptimo inserido numa estrutura de controlo interno(Fig. 4.39).

4.3.2.3.Estrutura de controlo feedforward

As Figs. 4.40 a 4.42 apresentam os resultados de controlo numa estrutura de controlo feedforward. O "controlador existente" da Fig. 3.12, é um PID com os seguintes parâmetros:

$$Kp = 475.5\,;\, Ki = 220.8\,;\, Kd = 25.63$$

Estes parâmetros foram obtidos utilizando parâmetros aleatórios de um PID, sendo posteriormente ajustados com a ferramenta sisotool, do Matlab.

Analisando as Figs. 4.40 a 4.42, pode verificar-se que a força aplicada no pêndulo pelo modelo óptimo apresenta amplitudes muito elevadas. Estas amplitudes mantém-se ao longo do tempo de uma forma continuada, o que faz com que este modelo com esta estrutura seja rejeitado. Isto acontece devido ao facto de a soma dos sinais de controlo do modelo óptimo e do PID levarem à saturação. Esta estrutura apresenta no entanto alguns resultados interessantes, quando os modelo inversos genéricos e especializados são inseridos nesta estrutura de controlo, como as amplitudes dos picos do deslocamento do ângulo θ, no geral menores que as obtidas com as estruturas anteriores.

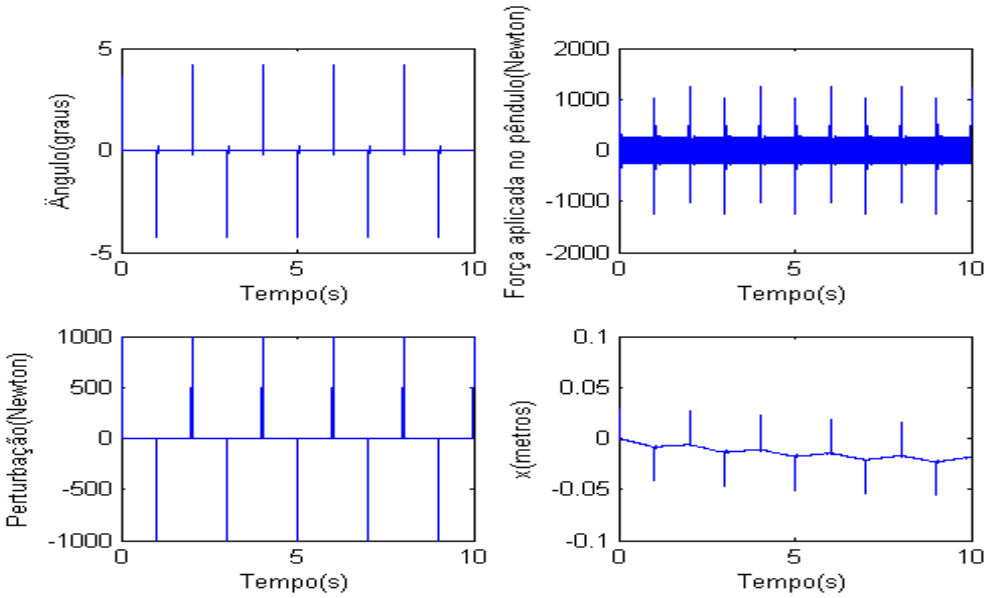


Figura 4.40: Resultados de controlo com o modelo inverso genérico inserido numa estrutura de controlo feedforward

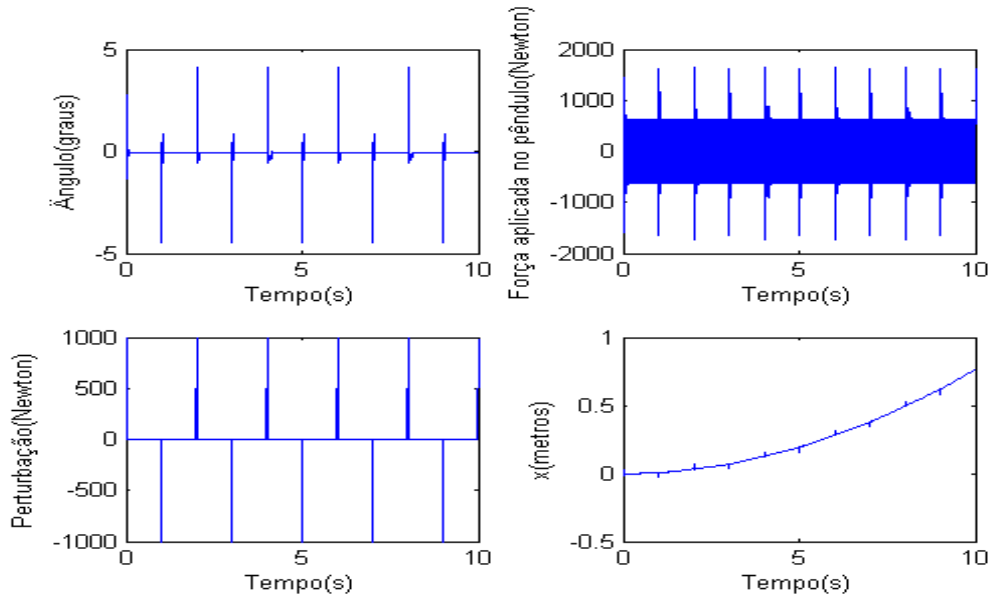


Figura 4.41: Resultados de controlo com o modelo inverso especializado inserido numa estrutura de controlo feeforward

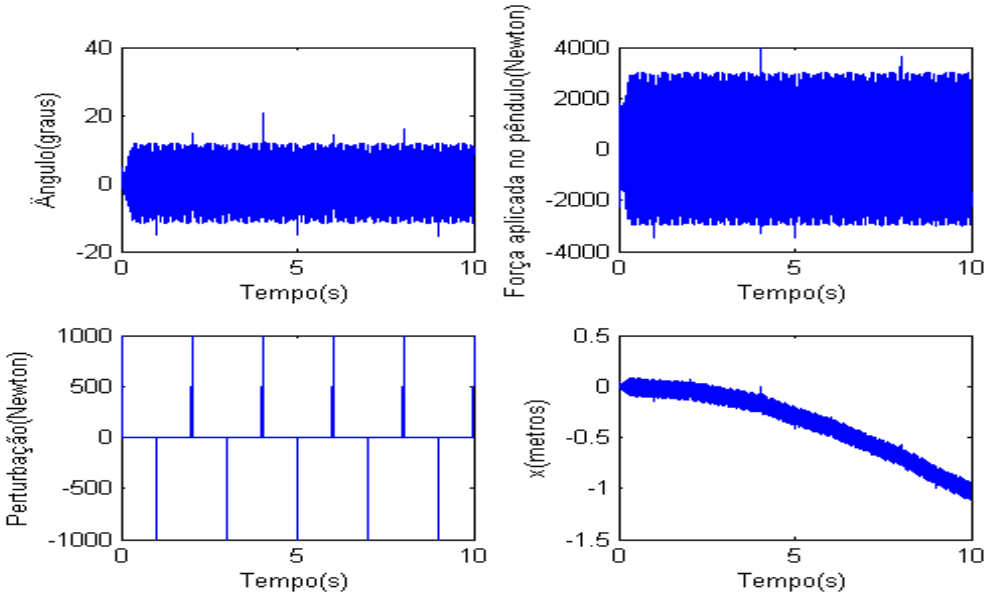


Figura 4.42: Resultados de controlo com o modelo óptimo inserido numa estrutura de controlo feedforward

4.3.2.4.Controlo com um PID

Retirando o controlador neuronal da estrutura feedforward, fica-se unicamente com o PID a controlar o pêndulo invertido. Os parâmetros do PID mantêm-se inalteráveis. Na Fig. 4.43 podemos ver o resultado do controlo, nesta situação.

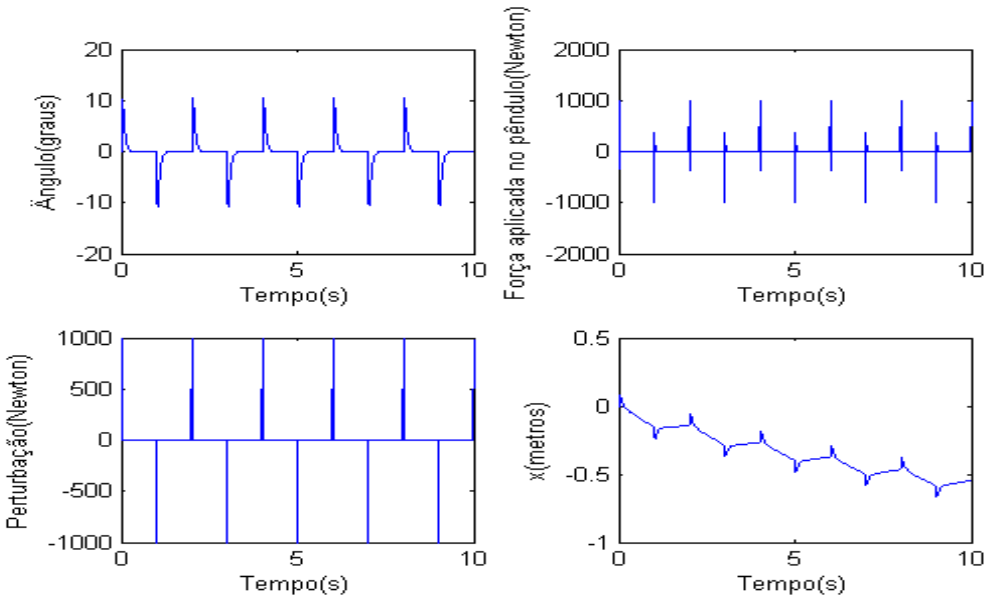


Figura 4.43: Controlo PID

O pêndulo invertido controlado por um controlador PID, apresenta quase o dobro da amplitude dos picos de deslocamento do ângulo θ, quando comparado com a generalidade dos restantes controladores neuronais. O tempo que o pêndulo invertido consome a voltar à posição de equilíbrio desejado $\theta = 0^{\circ}$, é maior com o controlador PID, do que com a generalidade das outras estruturas de controlo, com RNAs.

4.3.2.5. Resumo de resultados

Nas tabelas 4.4 e 4.5 estão resumidos os resultados de controlo recolhidos. As tabelas têm o objectivo de representar quantitativamente os resultados de controlo, tendo como medida de qualidade o MSE. Todos os valores das tabelas, são unidades de ângulo, em graus.

Tabela 4.4: Dados numéricos do controlo do pêndulo invertido

	Estrutura de controlo inverso			Estrutura de controlo interno			Estrutura de controlo feedforward		
	MSE	min [graus]	max [graus]	MSE	min [graus]	max [graus]	MSE	min [graus]	max [graus]
Modelo inverso genérico	0.3997	-5,287	5.261	0.4481	-5.287	5.260	0.2071	-4.22	4.197
Modelo inverso especializado	0.1199	-3.48	4.587	0.2079	-5.313	4.864	0.1721	-4.468	4.104
Modelo óptimo	0.1872	-3.271	3.696	0.4281	-6.088	6.26	58.64	-15.53	20.68

Tabela 4.5: Dados numéricos controlo PID

Controlo PID		
MSE	min [graus]	max [graus]
6.903	-10.65	10.62

O MSE traduz o erro quadrático médio, entre a saída da planta e o valor desejado à saída da planta que é precisamente 0 graus de ângulo. Desta forma tem-se uma indicação quantitativa da qualidade do controlo. Um valor alto de MSE, indica um controlo menos bom. que um MSE baixo. Por exemplo, um MSE elevado, dá uma indicação, de que existem picos de deslocamento do ângulo θ elevados e/ou que o tempo que o pêndulo invertido necessita para retornar á posição de equilíbrio desejado $\theta = 0^{o}$ é maior. Os valores min e max, demonstram a amplitude mínima e máxima, respectivamente, em graus, do ângulo obtido à saída da planta. De uma forma sumária pelo valor do MSE, podemos observar, que o modelo inverso especializado inserido numa estrutura de controlo inverso, apresenta melhores resultados de controlo, que o modelo óptimo e que o modelo inverso genérico. A estrutura de controlo interno, segue a mesma tendência, com resultados similares.

A estrutura de controlo feedforward, apresenta melhores resultados de controlo, que a estrutura de controlo, em que o PID é o único controlador do pêndulo invertido. Nota-se que a adição, em paralelo dos controladores neuronais modelo inverso genérico e modelo inverso especializado com o PID, melhoram significativamente os resultados de controlo, obtidos com o PID isolado.

4.4. O pêndulo invertido não linear

4.4.1. Identificação do pêndulo invertido não linear

4.4.1.1. Aquisição *de* dados

Para a identificação do pêndulo invertido não linear é necessário repetir a mesma sequência procedimental, utilizada com o pêndulo invertido linear. Neste caso existem dificuldades acrescidas, para fazer a aquisição de dados da planta, devido ao facto de além de termos um sistema instável, tem-se também um sistema não linear. Um PID está vocacionado, para estabilizar sistemas lineares, de modo que não pode ser usado no caso do pêndulo invertido não linear. A literatura ligada ao controlo com redes neuronais, sugere que se utilize uma lei de controlo, baseada em linearização por realimentação(feedback linearisation), para proceder à aquisição de dados [Tim Callinan, 2003; C. Anderson, 1989; J. K. Hedrick et al, 2005]. O controlador desenhado por esta técnica, produz um sinal, que torna o sistema planta+controlador num conjunto de r integradores em malha aberta. O número de integradores r, é o grau relativo do sistema planta+controlador. A técnica de controlo linearização por realimentação, faz com que os sistemas das Figs. 4.44 e 4.45 sejam formalmente equivalentes.

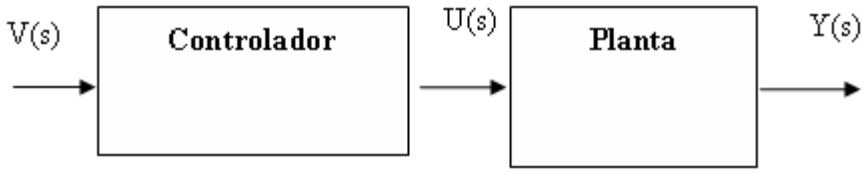


Figura 4.44: Controlador+ planta

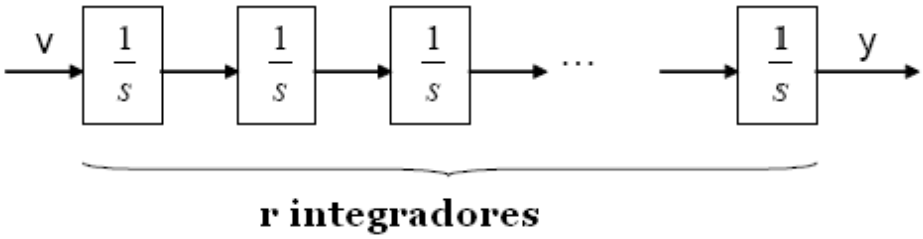


Figura 4.45: Sistema equivalente linearizado

Considere-se uma planta, com uma só entrada e uma só saída , cujas equações dinâmicas possam ser descritas pelo seguinte sistema de Eqs. 4.23:

$$\begin{cases} \dot{x} = f(x) + g(x)\,u \\ y = h(x) \end{cases} \tag{4.23}$$

x é o vector de estado e y é a saída da planta. y é derivado r vezes até que *u* apareça na equação e seja possível escrever u em função de v. Quando isto acontece: $\frac{d^r y}{dt^r} = v$. A partir desta equação pode-se obter o sinal de controlo *u,* em função de v. No domínio de Laplace tem-se que $Y(s) = \frac{1}{s^r} V(s)$. Mais detalhes sobre esta técnica de controlo não-linear, podem ser encontrados em [J. K. Hedrick et al, 2005]. Para aplicar esta técnica de controlo ao pêndulo invertido é necessário resolver o sistema de Eqs. 4.11, em função da segunda derivada do deslocamento linear ($\ddot{x}$) e do ângulo $\ddot{\theta}$. Para isso utiliza-se a função *solve* da forma mostrada no ficheiro *solvematrix.m* em anexo. A correspondência das variáveis utilizadas na função é mostrada no sistema de Eqs. 4.24:

$$\begin{cases} xdotdot = \ddot{x} \\ thetadotdot = \ddot{\theta} \\ x1 = x \\ x2 = \theta \\ x3 = \dot{x} \\ x4 = \dot{\theta} \end{cases} \tag{4.24}$$

$\dot{x}$ e $\dot{\theta}$ representam a primeira derivadas do deslocamento do carro e a primeira derivada do ângulo, em ordem ao tempo. No caso do pêndulo invertido o sinal de controlo *u*, é a força F aplicada no pêndulo invertido. O sistema de Eqs. 4.23 transforma-se nas Eqs. 4.25 e 4.26:

$$\dot{x} = f(x) + g(x).F \tag{4.25}$$

$$y = x_2 \tag{4.26}$$

onde f(x) é igual a:

$$\begin{bmatrix} x3 \\ x4 \\ \dfrac{-(m^2 l^2 \cos(x2)\, g \sin(x2) - \sin(x2) l^3 m^2 x4^2 - I\, m\, l\, x4^2 \sin(x2) - l^2 m\, b\, x3 - I\, b * x3)}{(-I\,M - I\,m - l^2 m\,M - l^2 m^2 + \cos(x2)^2 l^2 m^2)} \\ \dfrac{(-m\, l\, (m\, g \sin(x2) - \cos(x2) m\, x4^2 \sin(x2) l - \cos(x2)\, b\, x3 + g \sin(x2) M))}{(-I\,M - I\,m - l^2 m\,M - l^2 m^2 + \cos(x2)^2 l^2 m^2)} \end{bmatrix} \tag{4.27}$$

e g(x) é igual a:

$$\begin{bmatrix} 0 \\ 0 \\ \dfrac{(I * F + l^2 * m * F)}{(-I * M - I * m - l^2 * m * M - l^2 * m^2 + \cos(x2)^2 * l^2 * m^2)} \\ \dfrac{(-m * l * \cos(x2) * F)}{(-I * M - I * m - l^2 * m * M - l^2 * m^2 + \cos(x2)^2 * l^2 * m^2)} \end{bmatrix} \tag{4.28}$$

Determinar as Eqs. 4.27 e 4.28 manualmente, pode ser uma tarefa complicada e morosa, devido ao possível elevado número de derivadas de y a calcular. Por este motivo, utilizou-se uma toolbox, que corre sobre a plataforma Matlab e está disponível na internet, denominada NelinSys [M. Ondera]. Esta toolbox fornece várias ferramentas para a análise e síntese de sistemas não lineares. Para obter o sinal de controlo *u* que lineariza o pêndulo invertido, foi utilizada a função *Exakt* desta toolbox. Esta função devolve texto na linha de comandos do Matlab, onde devem ser especificados alguns parâmetros. Um exemplo do preenchimento destes parâmetros é mostrado em anexo.

O sistema utilizado para a aquisição de dados do pêndulo invertido não linear está representado na Fig.4.46. No bloco "Pole Placement SISO" deve ser definido os pólos do sistema em malha fechada, assim como o grau relativo r do sistema.

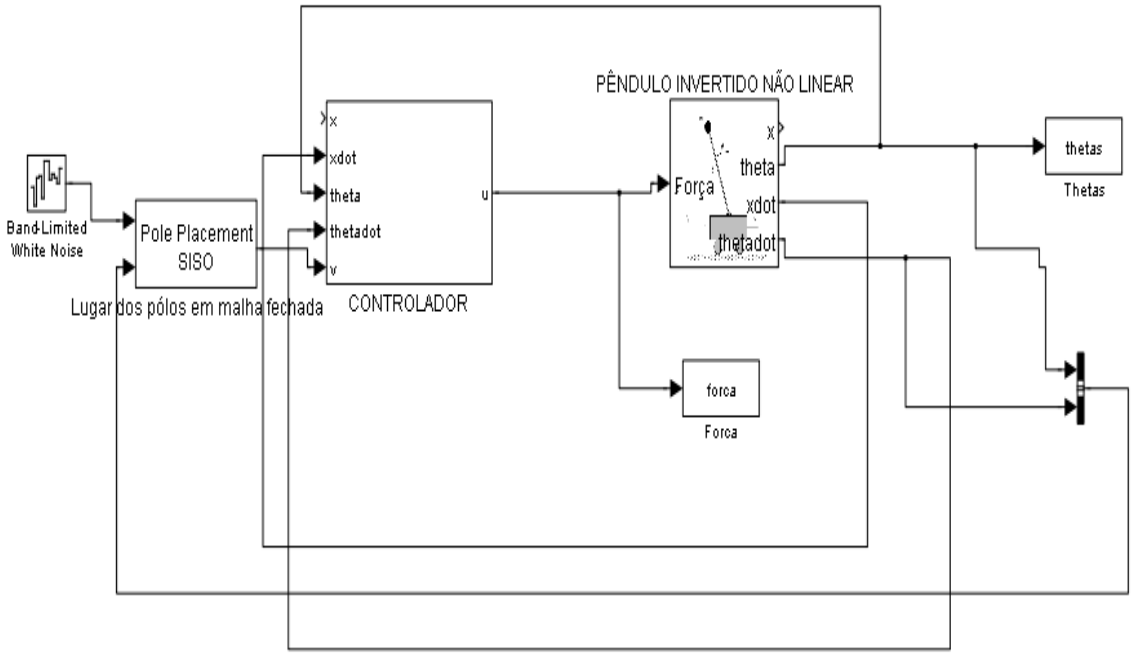


Figura 4.46: Sistema para aquisição de dados do pêndulo invertido não linear

A lei de controlo u que lineariza o pêndulo invertido é dado pela Eq. 4.29.

$$u = \frac{-(l.m.g.sin(x2) - l.m.cos(x2).x4.sin(x2) - l.m.cos(x2).b.x3)}{(l.m.cos(x2))} - \ldots$$
$$\ldots - \frac{(l.m.g.sin(x2).M - v.I.M - v.I.m - v.l.m.M - v.l.m + v.cos(x2).l.m)}{(l.m.cos(x2))} \quad (4.29)$$

O diagrama Simulink do controlador não linear está representado na Fig. 4.47.

Para ajustar o bloco Pole Placement SISO deve clicar-se duas vezes sobre este bloco, aparecendo a caixa de diálogo da Fig. 4.48. Em "Linear system order" deve ser especificado o grau relativo r. Em "Desired poles of a closed-loop system", deve ser definido o lugar dos pólos do sistema em malha fechada, que devem ser tantos, quanto o grau relativo do sistema r.

Em detalhe pode ser visto na Fig. 4.49, o interior do bloco "Pole placement SISO". Para observar este diagrama, clicar com o botão direito do rato sobre "Pole placement SISO" e depois clicar “Look under mask”.

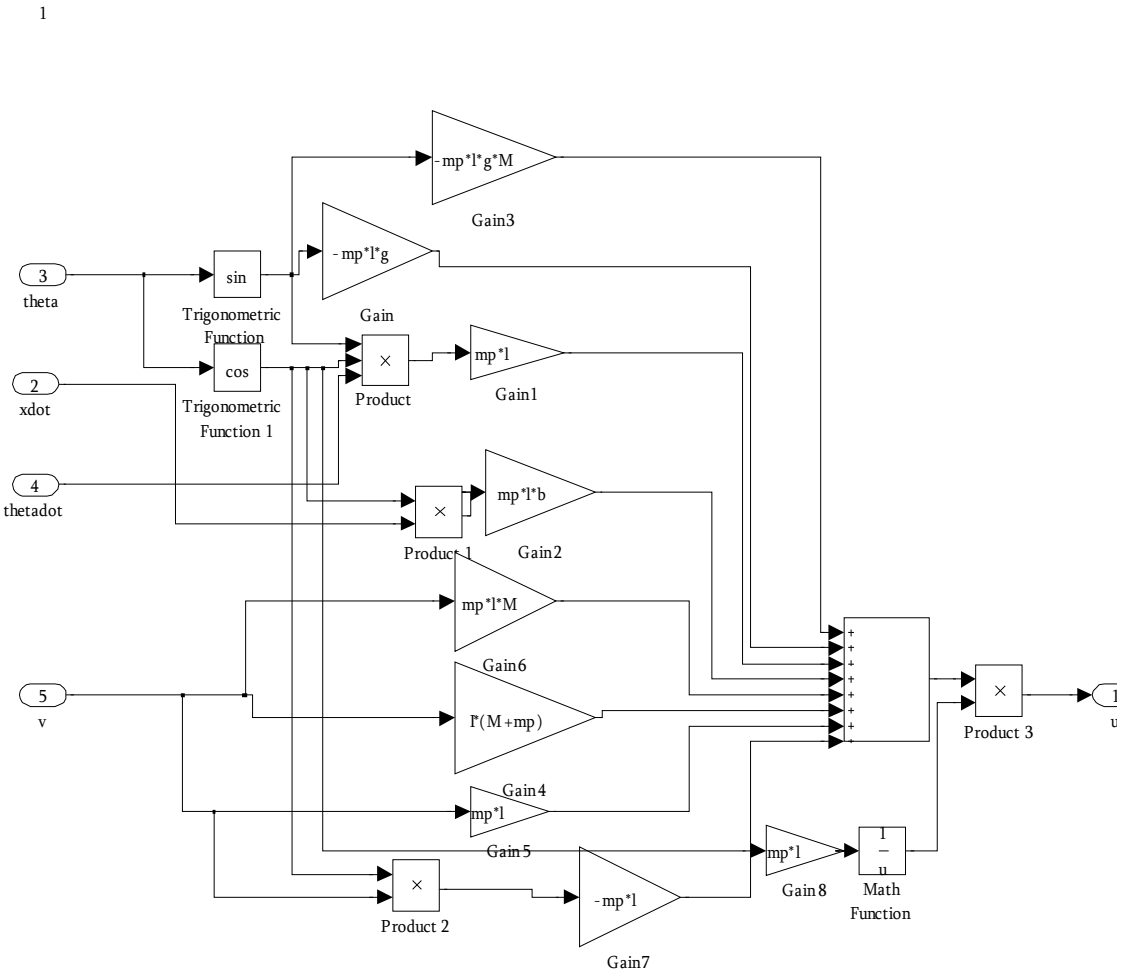


Figura 4.47: Diagrama Simulink do controlador de linearização por realimentação

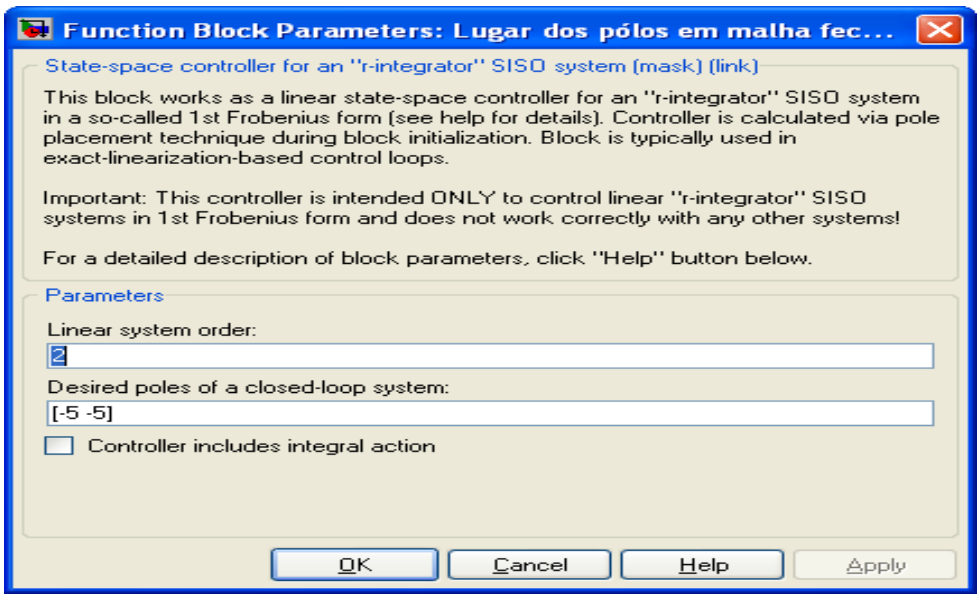


Figura 4.48: Parâmetros Pole Placement SISO

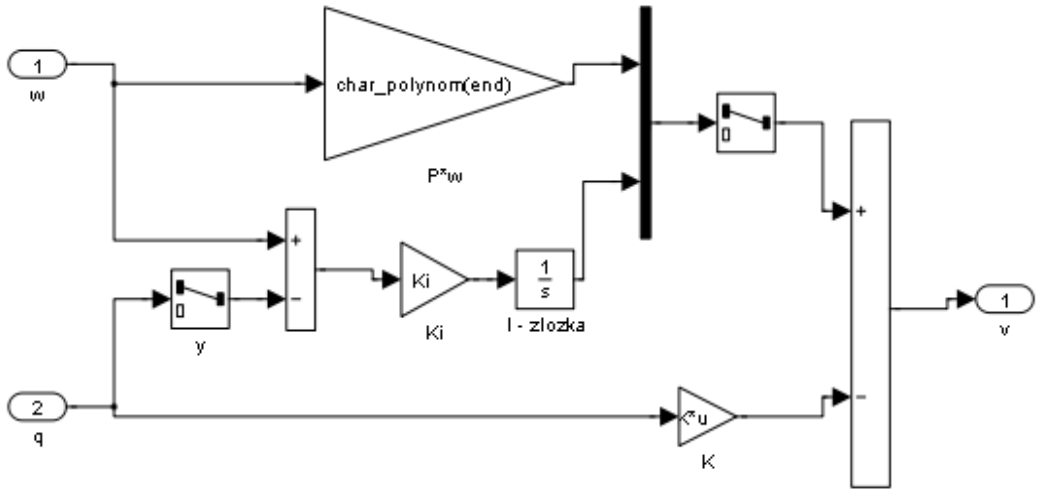


Figura 4.49: Detalhe do bloco Pole Placement SISO

Os ramos que culminam no integrador $\frac{1}{s}$ da Fig. 4.49, só interferem no sistema, se na Fig 4.48, for seleccionado "Controller includes integral action". K na parte inferior da Fig. 4.49, define a matriz de realimentação de estados, que é calculada com base nos pólos especificados na Fig. 4.48. A entrada w do sistema da Fig. 4.49 é onde é inserida a referência . O bloco amplificador P.w é uma amplificação do sinal da referência, que faz com o sistema em malha fechada tenha um ganho DC unitário. Como exemplo refere-se os parâmetros especificados da Fig. 4.48. Estes foram utilizados nesta dissertação, para a aquisição de dados do pêndulo invertido não linear, com dois pólos em [-5,-5]. A função de transferência do sistema em malha fechada é dada pela Eq. 4.30.

$$\frac{\theta(s)}{R(s)} = \frac{P.w}{(s+5)*(s+5)} = \frac{P.w}{s^2+10s+25} \tag{4.30}$$

R representa a referência. Para que o ganho DC seja unitário P.w=25. Os ganhos da matriz de realimentação de estados K, são dados pelos coeficientes de menor grau da polinomial característica do sistema em malha fechada [25,10].

Com a função bandwidth do Matlab calcula-se a largura de banda do sistema em malha fechada. O resultado obtido foi 3.2 rad/s. Pela condição da Eq. 3.2, foi escolhida a frequência de amostragem de 200 Hz.

4.4.1.2.Selecção da estrutura do modelo

Por analogia com o estudo do pêndulo linear devemos averiguar, qual o número óptimo de regressores que deve ter a RNA. Para isso apresenta-se na Fig. 4.50 o critério da Eq. 4.21, versus o número de entradas e saídas atrasadas da planta.

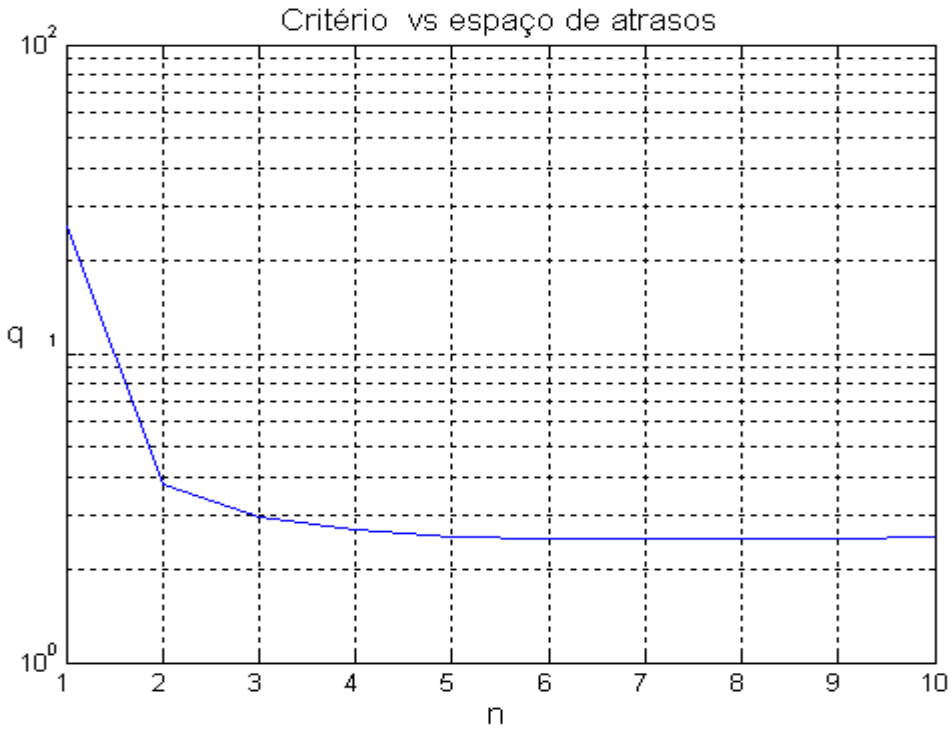


Figura 4.50: Critério vs espaço de atrasos

Constata-se que a partir de um número de entradas e saídas atrasadas igual a 3, não existe diminuição significativa do critério *q*.

Para averiguar qual das classes de modelos recorrentes NNARX ou NNARMAX apresenta melhor desempenho, apresenta-se as Figs. 4.53 a 4.56. As redes foram treinadas com 5 neurónios, na sua camada escondida. Os dados retirados do sistema de aquisição de dados estão divididos em 75% dados de treino e 25% dados de validação. Foram utilizadas as funções *simulnarmax1* e *simulnarx* (em anexo), para obter as Figs. 4.51 a 4.54. A função *simulnarmax1* treina a RNA com a estrutura NNARMAX e a função *simulnarx* treina a RNA com estrutura NNARX. As Figs. 4.51 a 4.54 demonstram, que a identificação do modelo directo da planta é muito preciso, dado que existe grande proximidade entre a curva, que representa os valores obtidos da saída da RNA e a curva que representa os valores obtidos da saída da planta.

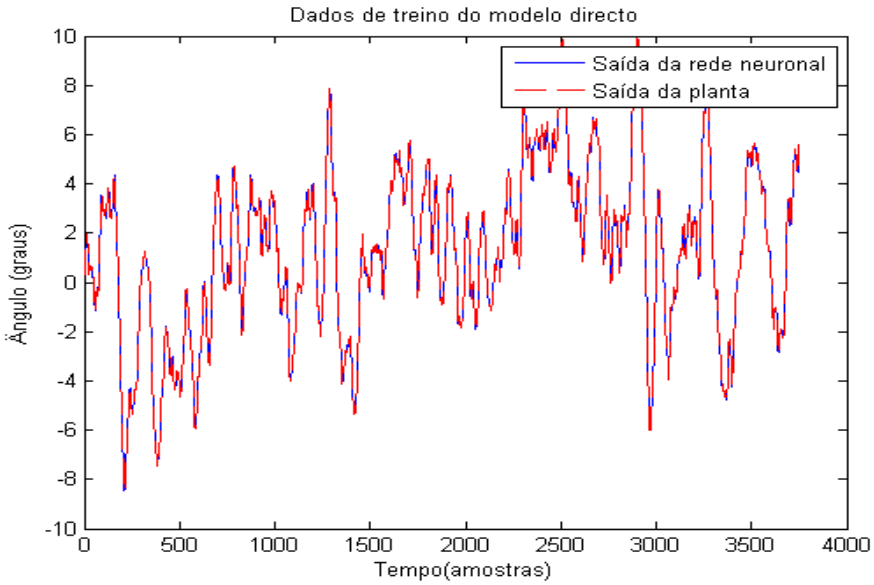


Figura 4.51: Dados de treino da classe de modelos NNARMAX

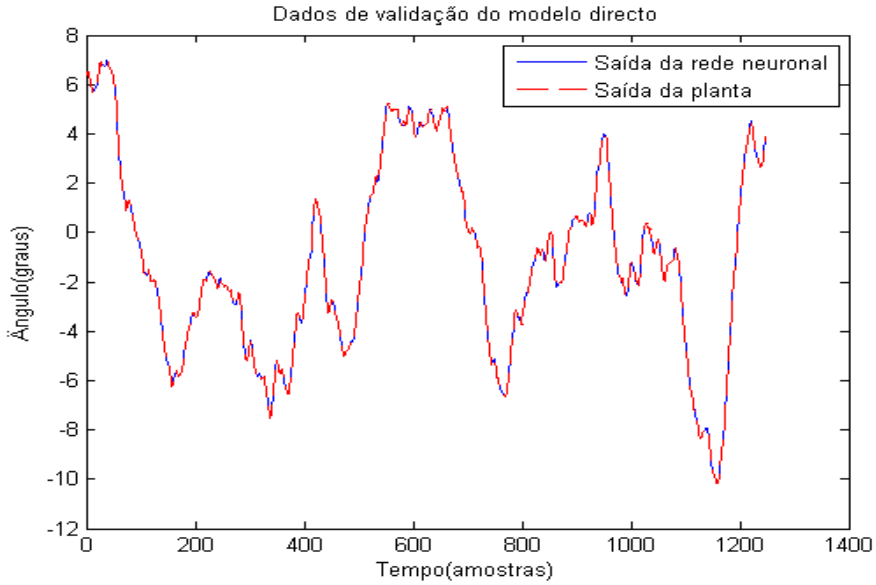


Figura 4.52: Dados de validação da classe de modelos NNARMAX

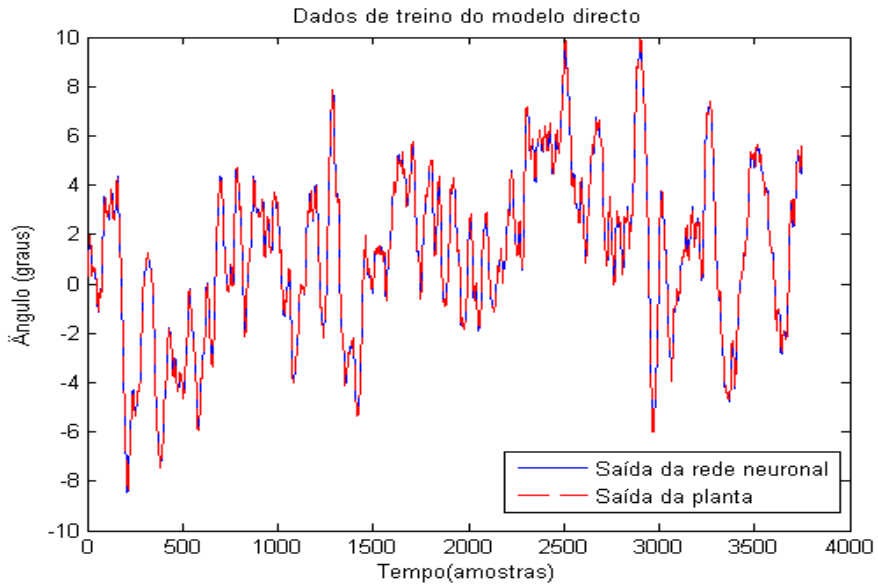


Figura 4.53: Dados de treino da classe de modelos NNARX

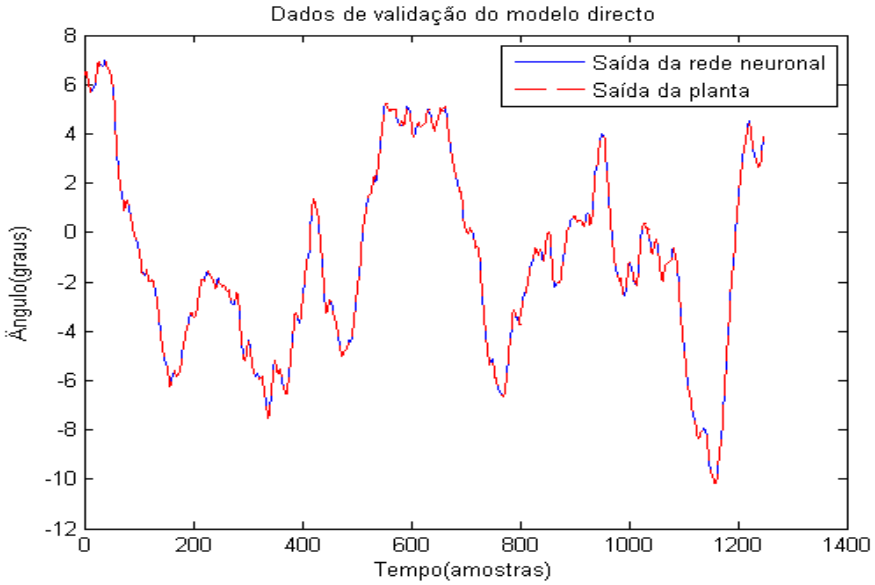


Figura 4.54: Dados de validação da classe de modelos NNARX

Na tabela 4.6 são apresentados os valores de MSE(eq.2.3), para as duas classes de modelos recorrentes. A tabela 4.6 afirma que a classe de modelos, que produz melhor desempenho é a NNARX.

Tabela 4.6: MSE das estruturas dos modelos NNARX e NNARMAX

	Modelo directo da planta	
	Treino	Validação
NNARX	9.47e-12	5,689e-11
NNARMAX	8.2769e-011	1.4652e-009

As Figs. 4.55 a 4.58 representam os dados de treino e validação do modelo inverso da planta obtidos da saída da RNA e da saída da planta. As figuras demonstram uma grande proximidade entre os traçados.

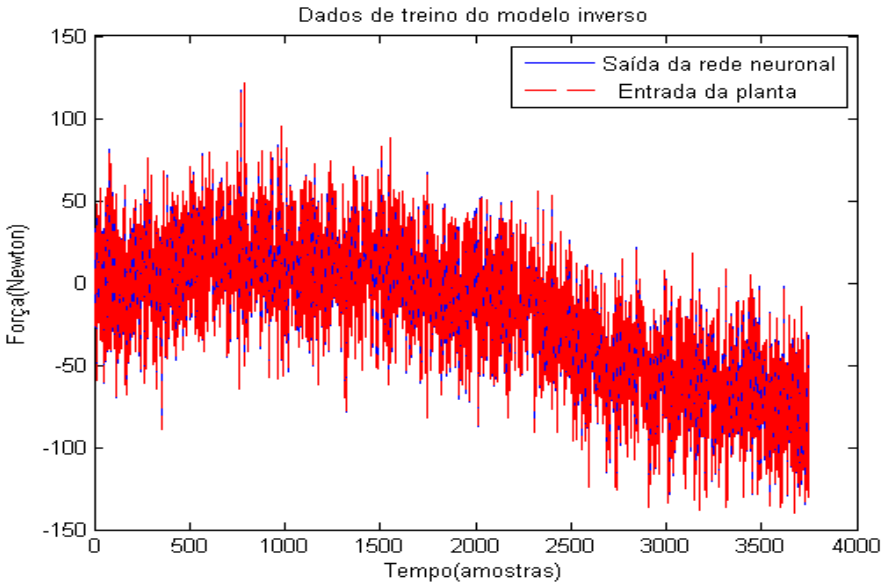


Figura 4.55: Dados de treino do modelo inverso da planta

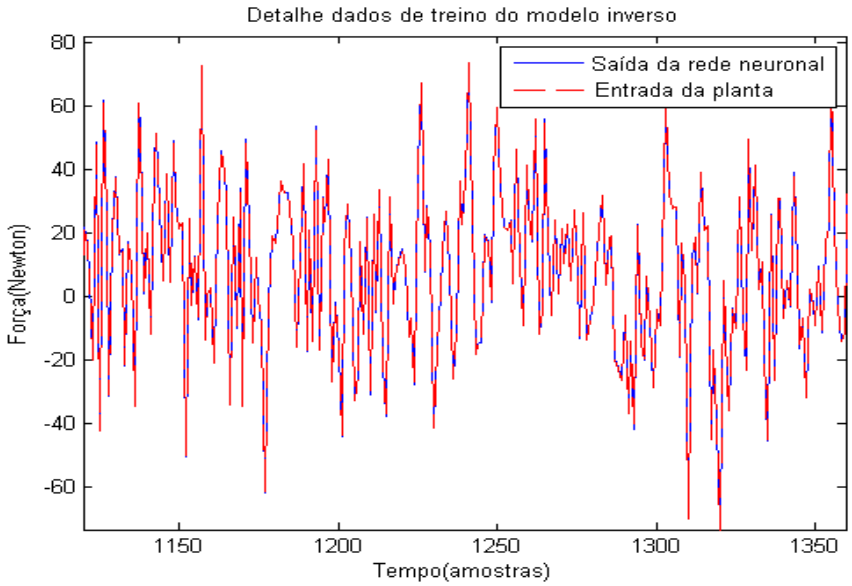


Figura 4.56: Detalhe dos resultados de treino do modelo inverso da planta

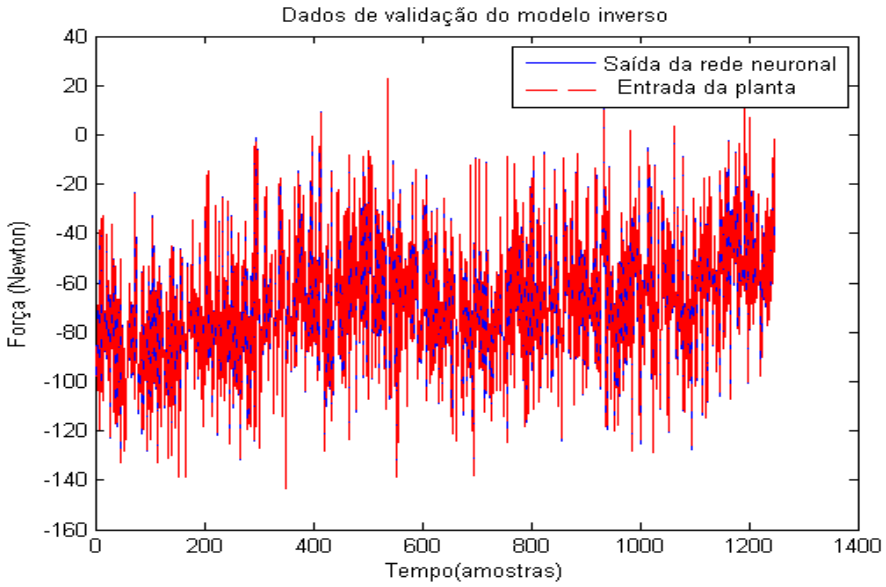


Figura 4.57: Resultados de validação do modelo inverso da planta

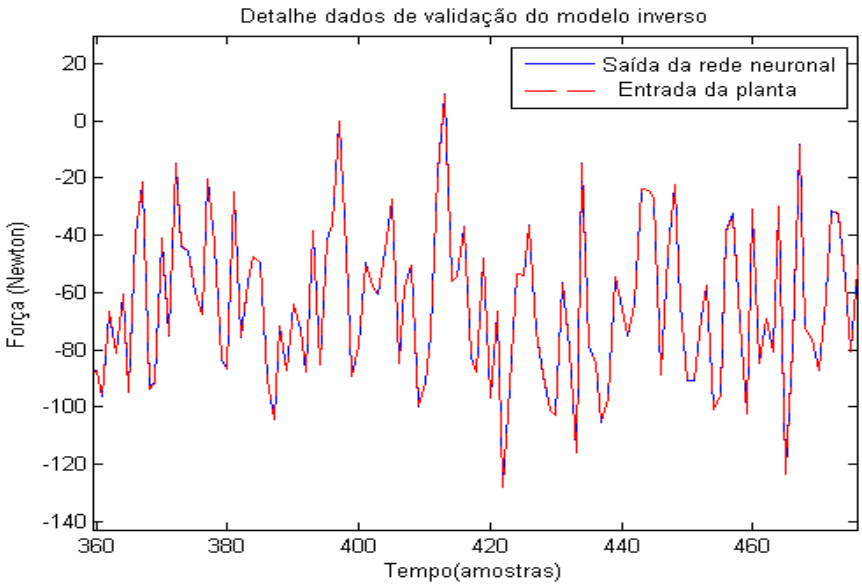


Figura 4.58: Detalhe dos resultados de validação do modelo inverso da planta

A tabela 4.7 mostra os valores de MSE obtidos para o modelo inverso da planta.

Tabela 4.7: MSE do modelo inverso da planta

	Modelo inverso da planta	
	Treino	Validação
NNARX	8.87e-003	0,03

O MSE de treino e de validação são muito próximos, o que representa um critério de aprendizagem e generalização correcto da rede.

4.4.2. Controlo do pêndulo invertido não linear

Para o estudo do controlo do pêndulo não linear, foram implementadas as estruturas de controlo apresentadas nas Figs. 4.22 a 4.24. Acrescenta-se a estas 3 estruturas de controlo uma quarta estrutura de controlo, denominada linearização por realimentação(feedback linearisation). Esta estrutura de controlo é uma versão com RNAs do sistema da Fig.4.46.

É necessário obter duas RNAs f e g, conforme sugere a Fig. 4.59. A função *simulnniol* (em anexo) treina estas duas redes f e g. A função *inverso_directo_nniol* (em anexo), obtém os respectivos modelos Simulink das RNAs f e g. O diagrama

Simulink da estrutura de controlo linearização por realimentação está representado na Fig. 4.59.

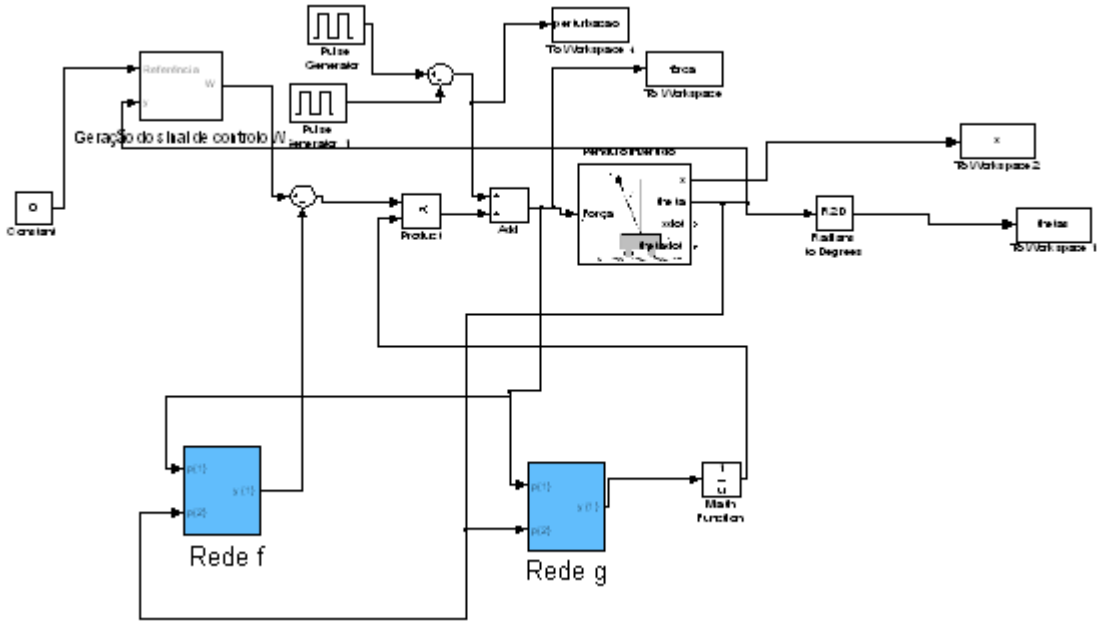


Figura 4.59: Estrutura de controlo linearização por realimentação implementada em Simulink

O diagrama Simulink da Fig. 4.59 implementa a estrutura de controlo da Fig. 3.14. Como é sugerido pela Eq. 3.17, o sinal da RNA f é subtraído ao sinal W e dividido pelo sinal da RNA g. As RNAs f e g pertencem à classe de modelos NNARX. Têm três entradas e saídas atrasadas e 5 neurónios na sua camada escondida. O sinal W é o resultado da combinação linear de entradas atrasadas com a referência. Isto permite definir os pólos em malha fechada, através de um polinomial característica, no domínio da transformada z[M. Norgaard et al, 2000]. A Fig. 4.60 mostra o detalhe do subsistema Simulink "Geração de sinal de controlo W". Este subsistema Simulink, foi implementado através da averiguação do ficheiro fblcon.m da NNCTRL toolbox. Este ficheiro simula o controlo com a estrutura de controlo linearização por realimentação. O código que permite chegar ao subsistema da Fig. 4.60 é o seguinte:

```
w  = sum(Am)*ref - sum(Am(2:nam)*[y;y_old(1:nam-2)]);
```

Am é vector dos coeficientes da polinomial característica definida no ficheiro *fblinit.m*, ref é a referência e nam é o comprimento do vector Am. O diagrama Simulink do bloco "Polinomial característica" representado na Fig. 4.61, pode ser visto na Fig. 4. 62.

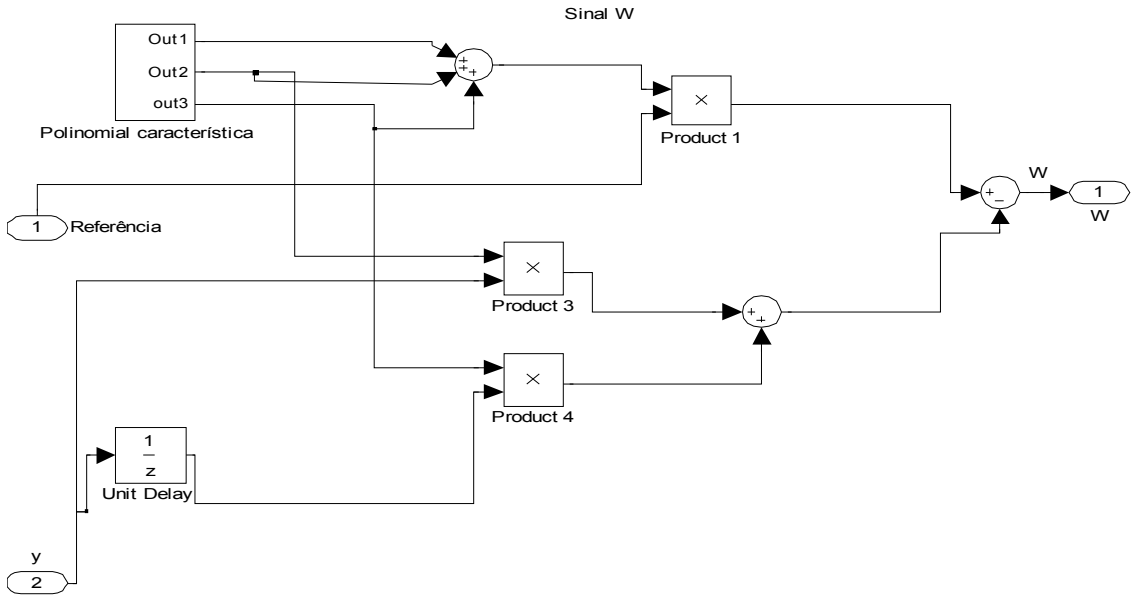


Figura 4.60: Detalhe do bloco "Geração do sinal de controlo W"

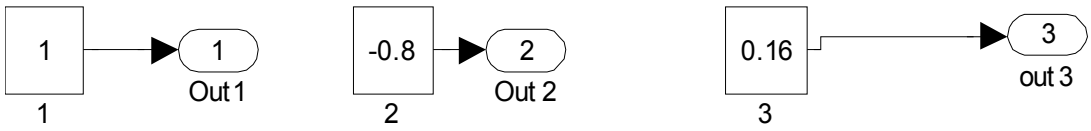


Figura 4.61: Detalhe do bloco "Polinomial Característica"

Para o caso concreto da Fig. 4.61:

$$Am = z^2 - 0.8.z + 0.16$$

nam=3

Isto implica que o sistema tem dois pólos em z=0.4.

4.4.2.1. Estrutura de controlo inverso

As Figs. 4.62 a 4.64 representam os resultados do controlo do pêndulo invertido não linar inserido numa estrutura de controlo inverso. O modelo inverso genérico e o modelo óptimo apresentam melhores resultados de controlo, ao nível da força aplicada no pêndulo invertido. Nota-se que o sinal de controlo(força aplicada no pêndulo) é mais suave com estes 2 controladores. O tempo que cada um dos 3 controladores necessita para voltar ao ponto de equilíbrio desejado $\theta = 0^{\circ}$, é para todos eles bastante reduzido. Os modelos especializado e óptimo apresentem no entanto, melhores resultados, como se pode deduzir pelo MSE tabelado na tabela 4.8 . Os valores da tabela 4.8 e os traçados da força aplicada no pêndulo invertido sugerem, que o modelo óptimo é um melhor compromisso, entre a qualidade de controlo da variável θ e a força aplicada no

pêndulo invertido. O pêndulo invertido linear demonstrou igual comportamento, quando inserido na mesma estrutura de controlo .

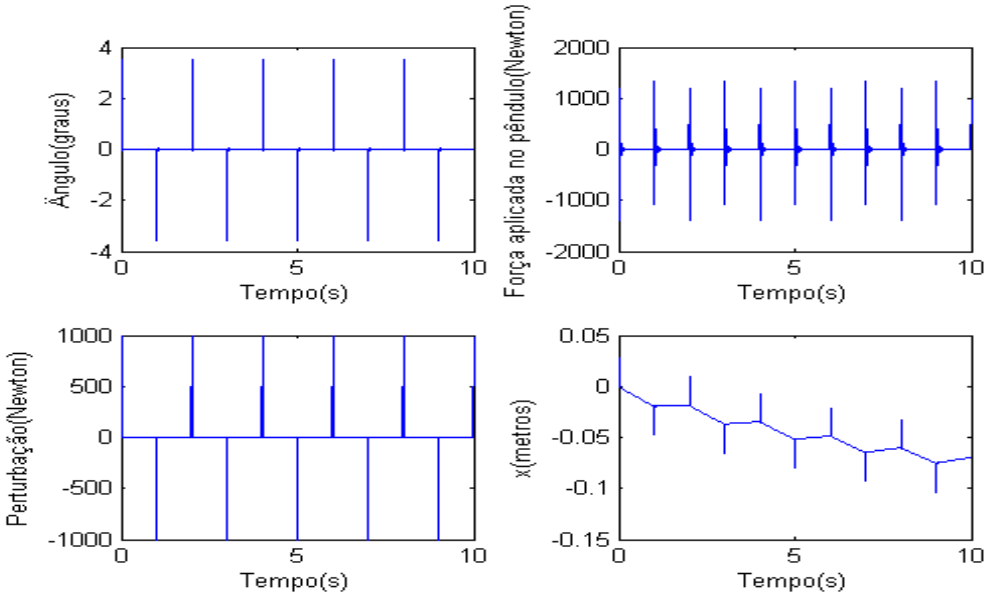


Figura 4.62: Resultados de controlo com o modelo inverso genérico inserido numa estrutura de controlo inverso

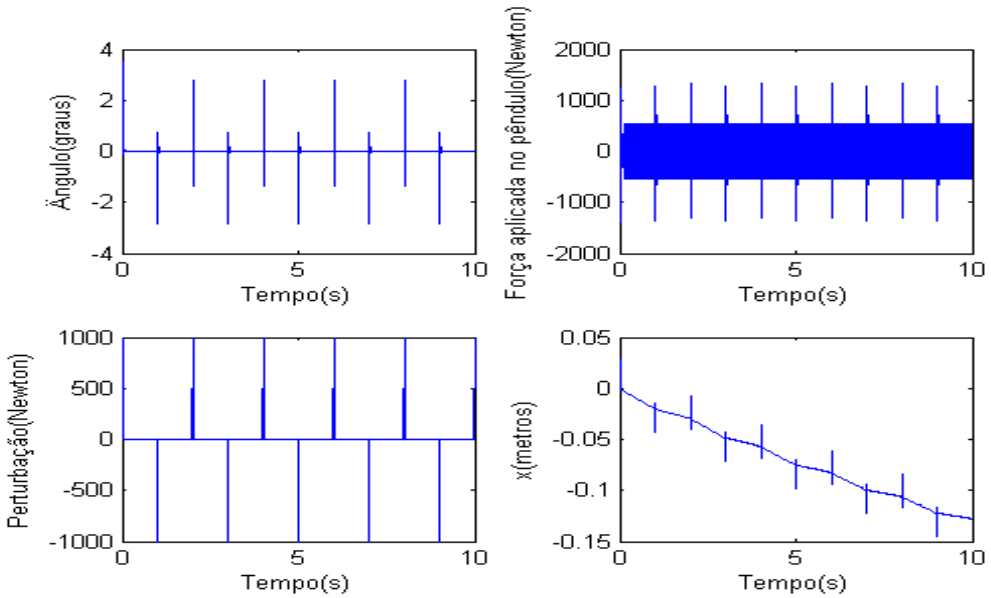


Figura 4.63: Resultados de controlo com o modelo inverso especializado inserido numa estrutura de controlo inverso

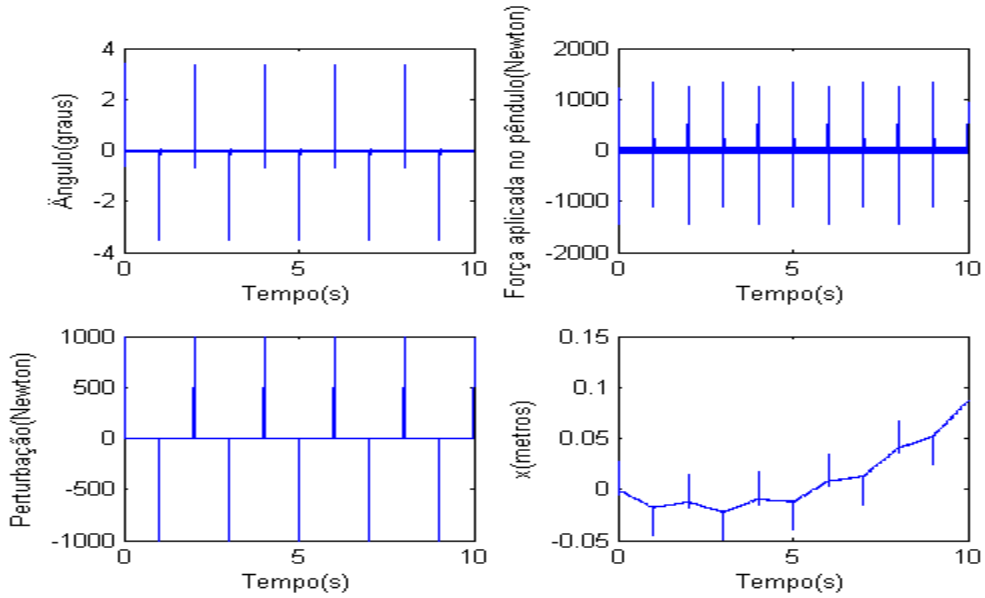


Figura 4.64: Resultados de controlo com o modelo óptimo inserido numa estrutura de controlo inverso

4.4.2.2. Estrutura de controlo interno

As Figs. 4.65 a 4.67 apresentam os resultados de controlo para uma estrutura de controlo interno.

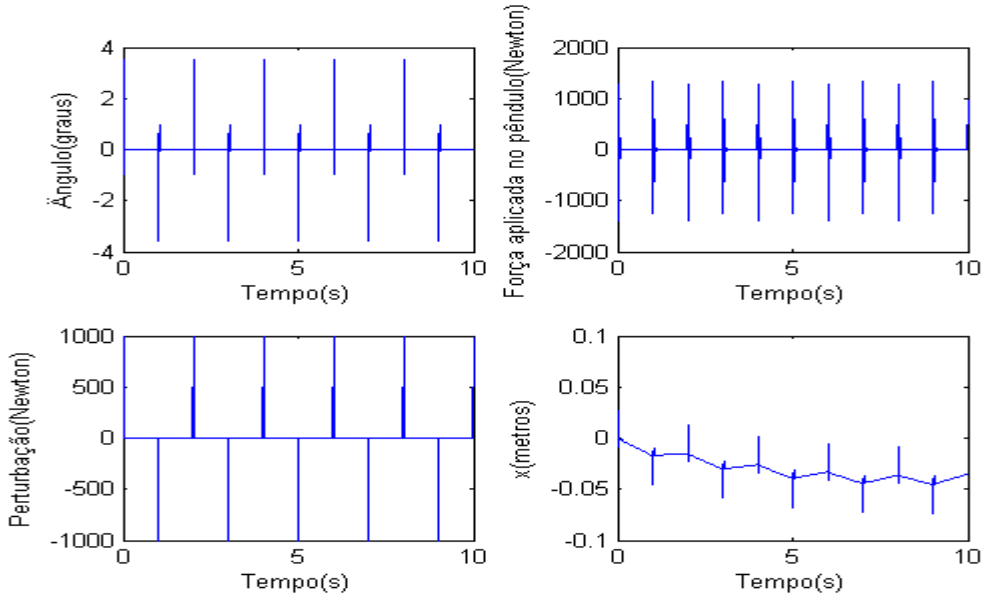


Figura 4.65: Resultados de controlo com o modelo inverso genérico inserido numa estrutura de controlo interno

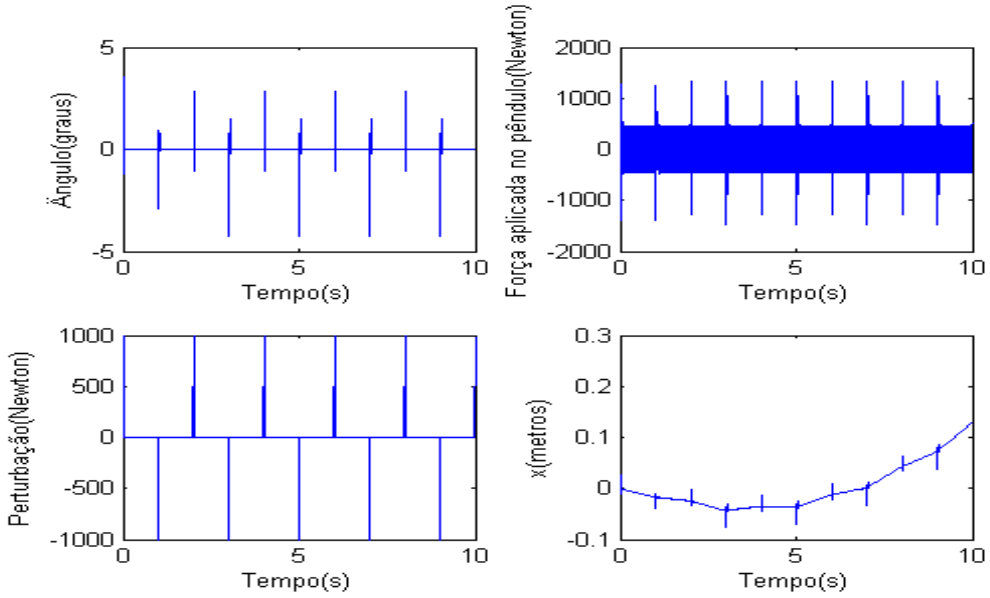


Figura 4.66: Resultados de controlo com o modelo inverso especializado inserido numa estrutura de controlo interno

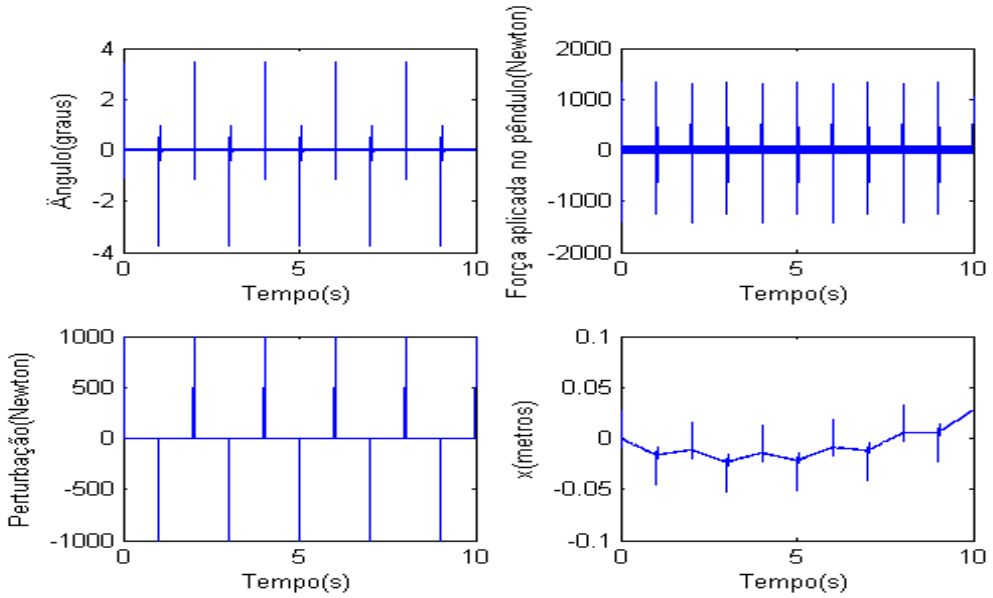


Figura 4.67: Resultados de controlo com o modelo óptimo inserido numa estrutura de controlo interno

Os resultados com a estrutura de controlo interno são muito similares aos obtidos com a estrutura de controlo inverso, não existindo significativas diferenças entre as duas estruturas de controlo.

4.4.2.3. Estrutura de controlo feedforward

As Figs. 4.68 a 4.70 apresentam os resultados de controlo numa estrutura de controlo feedforward. O controlador existente utilizado é um PID com os seguintes parâmetros, que já foram utilizados para o pêndulo invertido linear:

$$Kp = 475.5 ; Ki = 220.8 ; Kd = 25.63$$

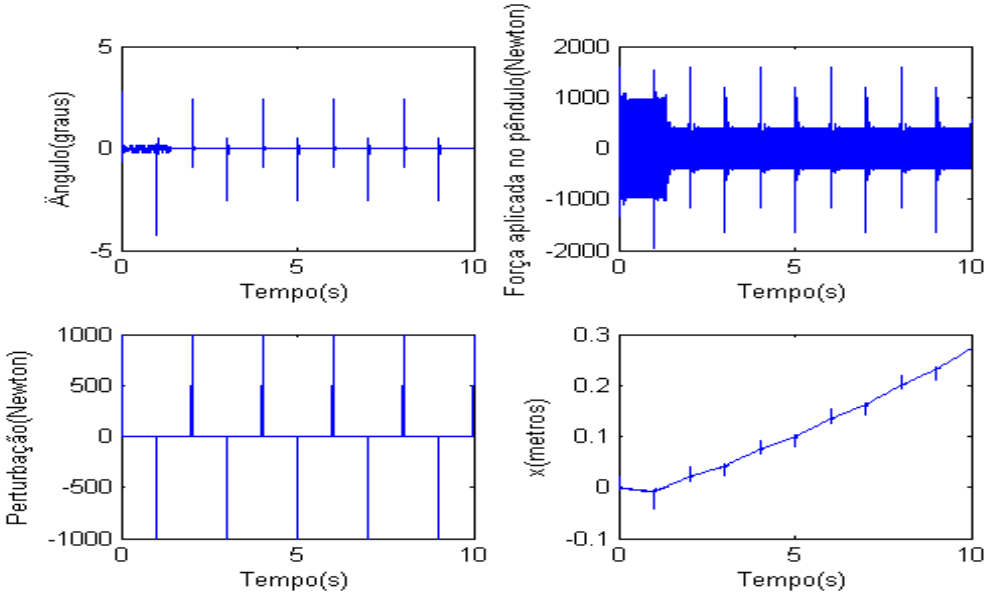


Figura 4.68: Resultados de controlo com o modelo inverso genérico inserido numa estrutura de controlo feedforward

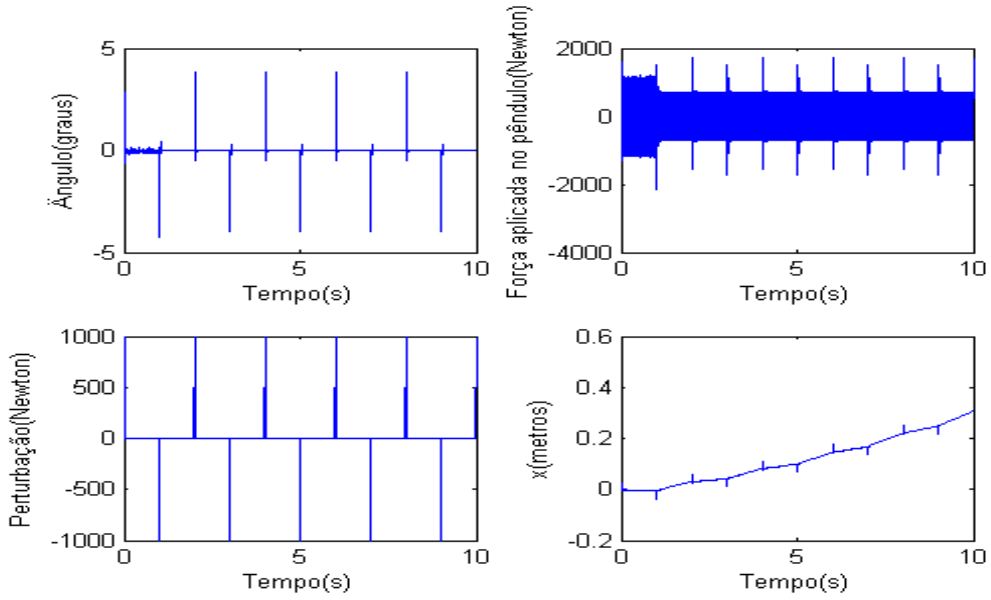


Figura 4.69: Resultados de controlo com o modelo inverso especializado inserido numa estrutura de controlo feedforward

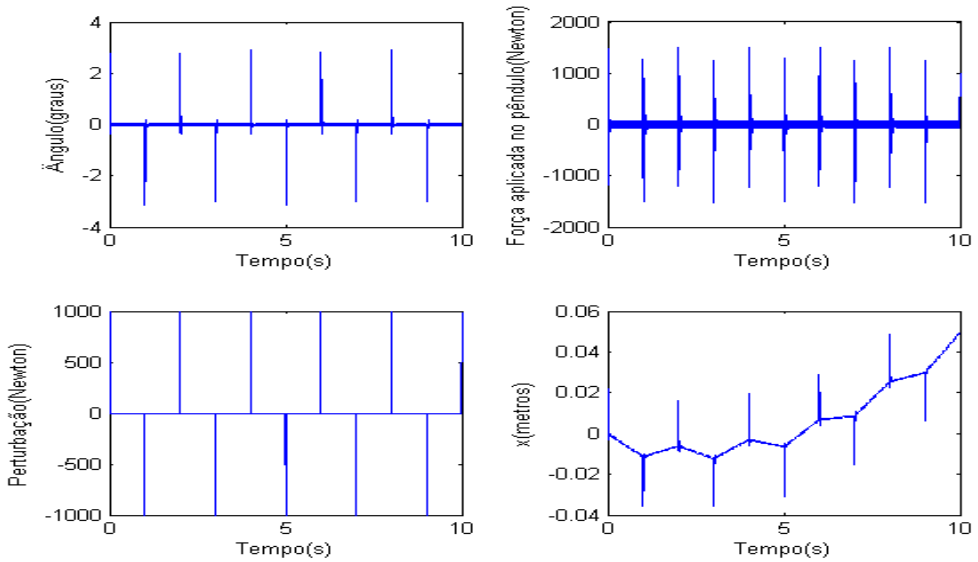


Figura 4.70: Resultados de controlo com o modelo óptimo inserido numa estrutura de controlo feedforward

A estrutura de controlo feedforward apresenta, para o caso do modelo inverso genérico, picos do deslocamento do ângulo θ , menores que os restantes controladores baseados em redes neuronais, embora apresente um pico inicial de maior valor. O modelo óptimo apresenta um sinal de controlo mais suave.

4.4.2.4. Estrutura de controlo linearização por realimentação(com redes neuronais)

A Fig. 4.73 mostra o resultado de controlo, numa estrutura de controlo linearização por realimentação. A polinomial característica utilizada foi :

$$Am = z^2 - 0.4.z + 0.04$$

O que implica 2 pólos em z=0.2 . Esta estrutura de controlo apresenta resultados da variável controlada θ , melhores ao nível dos picos de deslocamento do ângulo θ . Estes são de menor amplitude, que nas outras estruturas de controlo baseadas em RNAs. Ajustando os pólos em malha fechada para o lado direito em direcção a s=0, no domínio de Laplace, i.e. colocando os pólos mais próximos de z=1, no domínio da transformada z, é possível obter sinais da força aplicada no pêndulo de menor amplitude. No entanto, o tempo que o pêndulo invertido demora a responder à perturbação é mais longo, ou seja demora mais tempo a voltar ao ponto de equilíbrio desejado $\theta = 0^{\circ}$. Esta estrutura de controlo com RNAs é bastante flexível, pois permite adequar o controlo às necessidades do utilizador, bastando para isso ajustar a polinomial característica da Fig. 4.62. Um exemplo do que aqui foi dito pode ser visto na Fig. 4.74 em que :

$$Am = z^2 - 1.6Z + 0.64$$

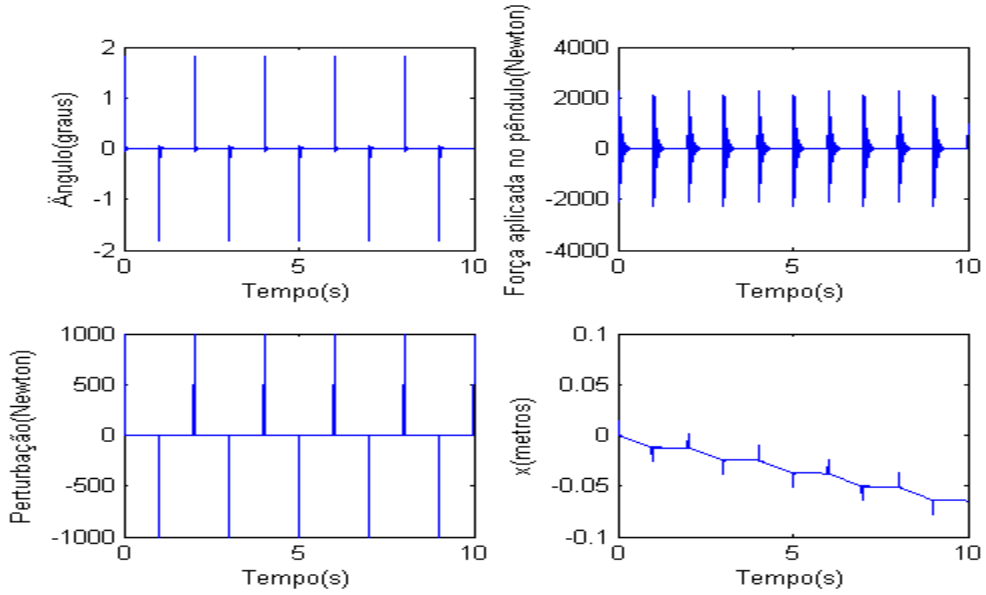


Figura 4.71: Controlo numa estrutura de linearização por realimentação

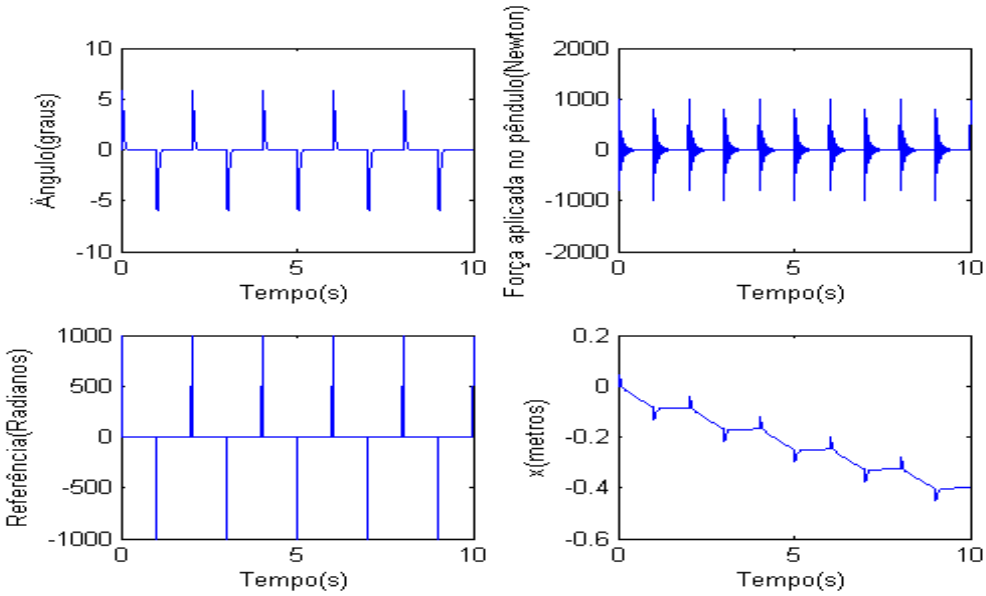


Figura 4.72: Exemplo de controlo numa estrutura de controlo com linearização por realimentação

Ou seja o sistema tem pólos em z=0.8. Vê-se claramente que o sistema é mais lento a voltar para o ponto de equilíbrio desejado (θ=0) e os picos de deslocamento do ângulo θ , tem maior amplitude, devido ao facto de os pólos se aproximarem de z=1, o que torna a resposta do sistema mais lenta. No entanto os valores da força aplicada são mais baixos.

4.4.2.5.Resultados do controlo com um PID

Por analogia com o pêndulo linear foi testado o comportamento do pêndulo não linear, numa estrutura de controlo feedforward, retirando a parte feedforward, ou seja retirando o controlador realizado pela RNA (ver Fig. 3.12). Neste caso o controlo reduz-se a um sistema com realimentação e controlador tipo PID. Os parâmetros do PID mantêm-se inalteráveis, relativamente aos usados no caso do pêndulo linear (capítulo 4.3.2.3). Na Fig. 4.75 podemos ver o controlo da planta por um PID. O PID apresenta piores resultados, relativamente às estruturas de controlo com RNAs, quer ao nível da amplitude dos picos de deslocamento do ângulo relativamente ao ponto de equilíbrio desejado , θ=0º , quer também ao nível do tempo que o pêndulo, demora a voltar para a posição de equilíbrio desejado θ=0º , que é notoriamente mais elevado.

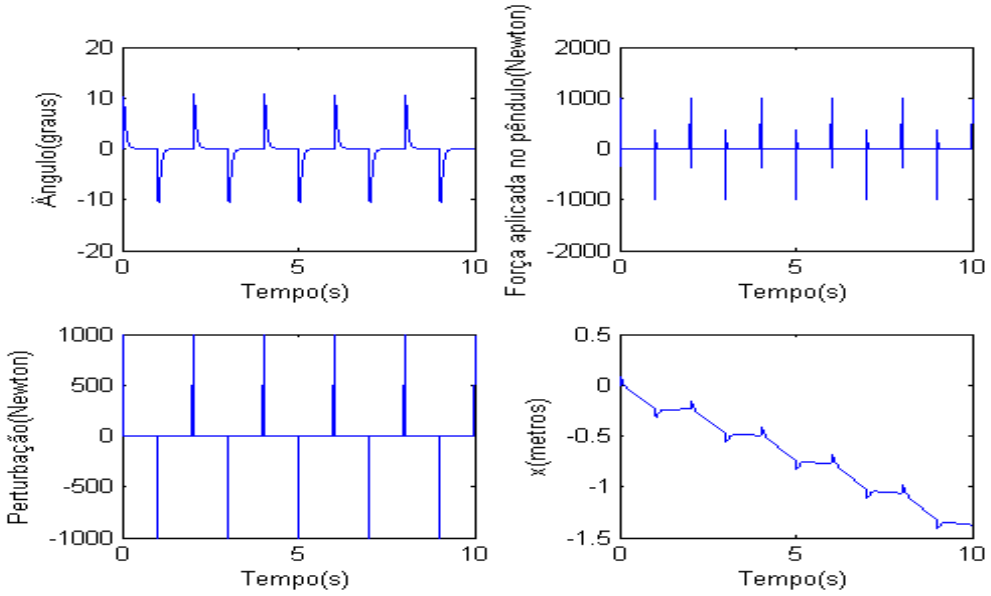


Figura 4.73: Controlo com um PID

4.4.2.6.Resumo de resultados

As tabelas 4.8 e 4.9 apresentam o resumo dos resultados de controlo para o pêndulo invertido não linear. O MSE traduz a qualidade do controlo. Um MSE alto indica um controlo de pior qualidade que um MSE baixo. Pelas tabelas 4.8 e 4.9 podemos ver que a estrutura de controlo que gera menores amplitudes dos picos do deslocamento do ângulo θ relativamente ao ponto de equilíbrio desejado θ=0º é a estrutura de controlo linearização por realimentação, assim como gera o terceiro valor de MSE mais baixo entre todas as estruturas de controlo. O modelo óptimo e o modelo inverso especializado inseridos numa estrutura de controlo inverso obtêm melhores resultados de controlo que o modelo inverso genérico, inserido na mesma estrutura de controlo.

Tabela 4.8: Resumo dos resultados obtidos

	Estrutura de controlo inverso			Estrutura de controlo interno			Estrutura de controlo feedforward		
	MSE	min [graus]	max [graus]	MSE	min [graus]	max [graus]	MSE	min [graus]	max [graus]
Mod. inverso genérico	0.094	-3.597	3.494	0.1016	-3.597	3.492	0.0581	-4.258	2.794
Mod. inverso especializado	0.0623	-2.853	3.515	0.1172	-4.243	3.514	0.1191	-4.274	3.855
Modelo óptimo	0.0892	-3.544	3.426	0.1088	-3.735	3.454	0.0616	-3.139	2.923

Tabela 4.9: Resumo dos resultados obtidos para o controlo PID e controlo linearização por realimentação

Controlo PID			Estrutura de controlo linearização por realimentação		
MSE	min [graus]	max [graus]	MSE	min [graus]	max [graus]
6.954	-10.59	10.64	0.0684	-2.412	2.413

A qualidade do controlo da estrutura de controlo interno é dependente da qualidade do conjugação do modelo inverso da planta e do modelo directo da planta .Por este motivo é compreensível que o modelo óptimo e inverso especializado obtenham pior desempenho que o modelo inverso genérico, pois este último, pode ter uma melhor conjugação com o modelo directo da planta. O modelo óptimo inserido numa estrutura de controlo feedforward é o que obtém melhor resultado de controlo, entre todas a estruturas de controlo estudadas , quer sejam elas baseadas em RNAs ou não. O controlo PID é aquele que apresenta piores resultados de controlo, tendo por exemplo um MSE cerca de 100 vezes superior ao MSE de um modelo óptimo, inserido numa estrutura de controlo feedforward. Podemos ver pelos valores tabelados que a RNA com o PID inseridos numa estrutura de controlo feedforward, têm melhor desempenho que o PID isolado a controlar a planta. Isto sugere que a RNA, melhora o desempenho do controlador PID e que pela adição simples de um controlador neuronal podemos ter um melhor controlo sobre a planta.

5. Conclusões

Os estudos efectuados neste trabalho permitem fazer as seguintes conclusões:

1. Relativamente à identificação dos sistemas dinâmicos (lineares ou não lineares) através de Redes Neuronais Artificiais (RNA).

Foram analisadas ***três arquitecturas*** principais – redes unidireccional (feedforward neural network- FFNN), redes recorrentes do tipo NNARX e NNARMAX (com realimentação de camada de saída para a camada de entrada) e rede de Elman (também recorrente mas com realimentação de camada escondida para a camada de entrada). Concluiu-se que as redes NNARX e NNARMAX são as mais adequadas para modelação de processos dinâmicos. O vector das entradas de NNARX (NNARMAX) consta entradas e saídas atrasadas do processo a ser identificado e nessa forma introduz memoria da dinâmica do processo. Por essa razão nos estudos seguidos são implementados redes recorrentes de classe NNARX ou NNARMAX.

Confirmou-se a regra empírica que o ***número de entradas e saídas*** da planta a colocar como entradas da rede é igual à ordem da planta.

A qualidade da rede como modelo dinâmico também depende dos dados de treino. Neste contexto o processo de aquisição dos dados que captam a dinâmica a ser identificada é fortemente influenciado pela frequência de amostragem (Fa). Foram testadas regras quantitativas para a escolha certa do valor de Fa. Por um lado valores de Fa demasiado baixos levam a perda de informação da dinâmica, mas por outro lado valores demasiado altos introduzem ruído no treino e problemas de mal condicionamento numérico.

2. Relativamente às estruturas de treino de RNA para identificar a dinâmica inversa da planta

Os controladores neuronais normalmente incluem ***uma componente – modelo neuronal inverso*** (por exemplo no esquema do controlo inverso, Fig. 3.10 e controlo feedforward, Fig. 3.13) e muitas vezes constam ***segunda componente – modelo neuronal directo*** (por exemplo no esquema do controlo interno, Fig. 3.11). Neste sentido as estruturas de treino modelo inverso genérico, modelo inverso especializado e modelo óptimo são alternativas para a modelação de dinâmica inversa do processo que controlamos.

Através destas estruturas projectou-se a primeira componente (modelo neuronal inverso) do controlador neuronal. A segunda componente é directamente obtida pela estrutura de treino - modelo directo (Fig. 3.7).

O modelo inverso genérico é o mais simples mas tem problemas numéricos devido a inversão directa do papel das entradas e saídas. Por essa razão foram desenvolvidas as estruturas alternativas do modelo inverso especializado e modelo óptimo.

Estes apresentam geralmente melhores resultados (especialmente quando introduzidos numa estrutura de controlo inverso, Fig. 3.10). No entanto o modelo óptimo tem a vantagem de considerar explicitamente na função de custo o esforço da entrada (no caso do pêndulo é a forca aplicada) e assim atingir um compromisso entre desempenho e esforço.

3. Relativamente às estruturas do controlo

As estruturas do controlo inverso e do controlo interno apresentam resultados similares, no entanto o controlo interno é mais imune as perturbações. O desempenho do controlo interno depende principalmente da conjugação entre as duas componentes - modelo neuronal inverso e modelo neuronal directo. Obviamente a estrutura de controlo inverso é mais simples de implementar, devido a existência de uma só componente - modelo neuronal inverso.

A estrutura de controlo feedforward deverá ser escolhida se for pretendido melhorar o desempenho de um sistema já compensado em malha fechada. Por exemplo reduzir o efeito de perturbações mensuráveis ou bem conhecidas. A aplicação do controlo feedforward no caso do pêndulo produz resultados razoáveis com deslocamentos angulares bem atenuados.

A estrutura de controlo linearização por realimentação, demonstra uma enorme flexibilidade, pois permite colocar os pólos do sistema em malha fechada (já um sistema linearizado), em qualquer lugar desejado. A aplicação deste controlo para o pêndulo não linear leva também a deslocamentos angulares bem atenuados.

O controlador PID demonstrou piores resultados nos dois casos - pêndulo invertido linear e não linear. É o mais lento, o tempo de estabelecimento é mais longo (se em principio consegue estabilizar o sistema) e os deslocamentos angulares atingem amplitudes mais altas (por exemplo 10 graus) .

Referências

[Paulo Cortez et al,2000] Paulo Cortez e José Neves, *Redes Neuronais Artificiais*, Unidade de ensino Departamento de Informática Escola de Engenharia Universidade do Minho, Braga, Portugal, 2000

[Pedro Gaspar, 2007]Pedro Miguel Coelho Gaspar,*Estudo comparativo sobre a utilização de diferentes estruturas de redes neuronais na modelação de sistemas,* Departamento de Electrónica, Telecomunicações e informática, Universidade de Aveiro,Portugal,2007

[M. Nørgaard et al,2000] M. Nørgaard, O. Ravn, N.K. Poulsen, L.K. Hansen (2000)*:Neural Networks for modelling and control of dynamic systems,* Springer-verlag, London,UK,2000

[M. Nørgaard,2000a] M. Nørgaard:*Neural Network Based System Identification Toolbox,*Tech. Report. 00-E-891, Department of Automation, Technical University of Denmark,2000a.

[M. Nørgaard,2000b] M. Nørgaard:*Neural Network Based Control System Design Toolkit ver. 2* ,Tech. Report. 00-E-892, Department of Automation, Technical University of Denmark,2000b.

*[Andrés Uribe,1999]*Andrés Pérez-Uribe*:Structure-Adaptable Digital Neural Networks,*Thesis,Département d' Informatique, Échole Polytechnique Féderale de Lausanne,1999

[Tim Callinan,2003] Tim Callinan*:Artificial Neural Network identification and control of the inverted pendulum",Agosto 2003*

*[E. Muhando et al,2006]E*ndusa Muhando, Hiroshi Kinjo,Eiho Uezato, Tomonobu Senjyu, Tetsuhiko Yamamoto, *"Evolutionary Neurocontroller Design for Nonlinear Dynamic Systems: Case of the Multi-trailer Back-up Control Problem"* Faculty of Engineering, University of the Ryukyus,Nishihara,Okinawa, Japan, 2006

*[F. Dias,2005]*Fernando Manuel Rosmaninho Morgado Ferrão Dias*:Técnicas de controlo não-linear baseadas em Redes Neuronais: do algoritmo à implementação,*Dissertação de doutoramento,Departamento de Electrónica e Telecomunicações, Universidade de Aveiro, Aveiro,Portugal, 2005

*[Howard Demuth]*Howard Demuth,Mark Beale,Martin Hagan, *Neural Network Toolbox Users Guide,*The Mathworks Inc.

[R. Sutton,1998] Richard Sutton and Andrew G.Barto, *Reinforcement Learning,*A Bradford Book, The MIT Press, Cambridge, Massachussets, Londres, Inglaterra, 1998.

*[C. Anderson.1989]*Charles Anderson*, Learning to Control an Inverted Pendulum Using Neural* Networks, American Control Conference, Atlanta, Georgia, 1989

[Haykin,1999] Simon Haykin,*Neural Networks,*Prentice-Hall,Inc, 1999

[H.Marquez,2003] Horacio J. Marquez, Nonlinear control systems,2003

[R.Marques,2001]Roberto Marques,Alan Borsato, Marcus Leal, Paulo Leite,*Tese final de fim de curso- Análise Matemática e Implementação de um Pêndulo Invertido, UCP,2001*

[Sorensen,1994] O.Sorensen, Neural Networks in Control Applications. PhD Thesis, Department of control Engineering, Institute of Electronic Systems, Aalborg University, Denmark, 1994

[Levenberg,1944] K. Levenberg, *"A method for the solution of certain problems in least squares*",*Quart. Appl. Math.*, 1944, Vol. 2, pp. 164–168.

[Marquardt, 1963] D. Marquardt, *"An algorithm for least-squares estimation of nonlinear parameters,"SIAM J. Appl. Math.*, 1963, Vol. 11, pp. 431–441.

[A.Melo, 2007]António Pereira de Melo, *"Sistemas de Controlo"*,Universidade de Aveiro,Aveiro,2007

[M. Nechyba et al.]Michael C. Nechyba and Yangsheng Xu Neural Network Approach to Control System Identification with Variable Activation Functions, The Robotics Institute Carnegie Mellon University Pittsburgh, PA 15213

[Karl Åström,1997] Karl Åström and Björn Wittenmark. Computer Controlled Systems. Prentice Hall, 1997.

[University of Michigan]The University of Michigan. Control tutorials for matlab - Digital Control Example: Inverted Pendulum using State-Space method disponível na internet em http://www.library.cmu.edu/ctms/ctms/examples/pend/diginvss.htm

[Sek Cheang,2000]Sek Un Cheang, Wei Ji Chen,Stabilizing Control of an Inverted Pendulum System Based on H- Loop Shaping Design Procedure,Faculty of Science and Technology University of Macau Macau, china,2000

[J. K. Hedrick et al,2005]J. K. Hedrick and A. Girard,Control of Nonlinear Dynamic Systems: Theory and Applications,2005

[M. Ondera] *M. Ondera,*Matlab-Based Tools for Nonlinear Systems,Department of Automation and Control, Faculty of Electrical Engineering and Information Technology Slovak University of Technology in Bratislava

[A. Drummond et al,1999] Adriana de C. Drummond, Kleyton C. de Oliveira e Adolfo Bauchspiess, Estudo do Controle de Pêndulo Inverso sobre Carro utilizando Rede Neural de Base Radial,Proceedings of the IV Brazilian Conference on Neural Networks - IV Congresso Brasileiro de Redes Neurais.1999

[University of Strathclyde] University of Strathclyde de Glasgow, Industrial Control Centre , http://www.icc.strath.ac.uk/research/wind_energy

[Chandrasekara et al, 2004] Chandrasekara, C.; Davari, A. Inverted pendulum: an experiment for control laboratory , 2004 ,Atlanta, Georgia

[J. Nelson, 1994]John Nelson and L. Gordon Kraft, Real-Time Control of an Inverted Pendulum System Using Complementary Neural Network and Optima Techniques, Junho de 1994, Department of Electrical and Computer Engineering, University of New Hampshire

Anexos

Anexos A -Ficheiros Matlab

```
%Prototipo
%PERFORMANCE.M
%treinox(função de treino, modelo Simulink)

treino1('trainlm','controllinear1');
treino1('traingd','controllinear1');
treino1('trainbfg','controllinear1');
treino2elm('trainlm','controllinear1');
treino2elm('traingd','controllinear1');
treino2elm('trainbfg','controllinear1');
```

```
%PROTOTIPO
%TREINO1
%treino1(função de treino, modelo Simulink)
function y=treino1(treino,model)


nonlinear;
sim(model,25);


P=forca';
T=thetas';
tam=length(P);
percent=0.75; %percentagem de dados de treino
P=P(1:round(percent*tam));
T=T(1:round(percent*tam));
VV.P=forca(round(0.75*tam)+1:end)';
VV.T=thetas(round(0.75*tam)+1:end)';

for i=4:9
    net=newff([-10 10],[i 1],{'tansig' 'purelin'},treino);
    net.trainParam.epochs=500;
    net.trainParam.lr=0.01;
    net.trainParam.mu=1;
    net.trainParam.max_fail=100;



    [net,tr,Y,E,Pf,Af] = train(net,P,T,[],[],VV);


    mse1(i-3)=tr.perf(end);
    mse2(i-3)=tr.vperf(end);
end
i=4:9;
figure;
plot(i,mse1,'r',i,mse2,'b');
legend('MSE dos dados de treino','MSE dos dados de validação');
```

```
xlabel('N° de neurónios na camada intermédia');
ylabel('MSE');

if strcmp(treino,'traingd')
    title('Performance da rede treinada pelo algoritmo Gradiente
Descendente');
else if strcmp(treino,'trainlm')
        title('Performance da rede treinada pelo algoritmo Levenberg-
Marquardt');
    else if strcmp(treino,'trainbfg')
            title('Performance da rede treinada pelo algoritmo Gauss-
Newton');
        end
    end
end
```

```
%PROTOTIPO
%TREINO2ELM
%treino2ELM(função de treino, modelo Simulink)
function y=treino2elm(treino,model)


    nonlinear;%contém parâmetros fisicos do pendulo invertido
    sim(model,25);


   P=forca';
T=thetas';
tam=length(P);
percent=0.75; %percentagem de dados de treino
P=P(1:round(percent*tam));
T=T(1:round(percent*tam));
VV.P=forca(round(0.75*tam)+1:end)';
VV.T=thetas(round(0.75*tam)+1:end)';
    for i=4:9
        net=newelm([-10 10],[i 1],{'tansig' 'purelin'},treino);
        net.trainParam.epochs=100;
        net.trainParam.lr=0.01;
        net.trainParam.mu=1;
        net.trainParam.max_fail=100;


        [net,tr,Y,E,Pf,Af]  = train(net,P,T,[],[],VV)

        mse1(i-1)=tr.perf(end);
        mse2(i-1)=tr.vperf(end);
    end
    i=4:9;
    figure;
    plot(i,mse1,'r',i,mse2,'b');
    legend('MSE dos dados de treino','MSE dos dados de validação');
    xlabel('N° de neurónios na camada intermédia');
    ylabel('MSE');

    if strcmp(treino,'traingd')
```

```
        title('Performance da rede Elman treinada pelo algoritmo
Gradiente Descendente');
    else if strcmp(treino,'trainlm')
        title('Performance da rede Elman treinada pelo algoritmo
Levenberg-Marquardt');
    else if strcmp(treino,'trainbfg')
       title('Performance da rede Elman treinada pelo algoritmo Gauss-
Newton');
        end
        end
    end
```

```
%PROTOTIPO
%SIMULNARX

clear all;
close all;
trparms=settrain;

nonlinear
%chama modelo Simulink do sistema com aquisição de dados
sim('pendulonaolinear2',25);

P=forca';
T=thetas';
tam=length(P);
percent=0.75; %percentagem de dados de treino
P=P(1:round(percent*tam));
T=T(1:round(percent*tam));
VP=forca(round(0.75*tam)+1:end)';
VT=thetas(round(0.75*tam)+1:end)';
NetDeff=['HHHHH';'L----'];

[n1,n2]=size(NetDeff);
na=3;
nb=3;

nmax=max(na,nb);
NN=[na,nb,1];
[W1f,W2f,critvec,iter,lambda]=nnarx(NetDeff,NN,[],[],trparms,T,P)

Y=nnsimul('nnarx',NetDeff,NN,W1f,W2f,T,P)


figure;
plot(Y*180/pi);
hold on;
plot(T(nmax+1:end)*180/pi,'r--')
xlabel('Tempo(amostras)');
ylabel('Ângulo (graus)');
legend('Saída da rede neuronal','Saída da planta');
title('Dados de treino do modelo directo');

save('directonarxnlinear.mat','NN', 'NetDeff', 'W1f','W2f');
%dados de validação
```

```
YsimV = nnsimul('nnarx',NetDeff,NN,W1f,W2f,VT,VP);
%erro de validação
erro=mse(VT(nmax+1:end)-YsimV);
figure;
plot(180*YsimV/pi);
hold on;
plot(VT(nmax+1:end)*180/pi,'r--')
xlabel('Tempo(amostras)');
ylabel('Ângulo(graus)');
legend('Saída da rede neuronal','Saída da planta');
title('Dados de validação do modelo directo');
%modelo inverso
NetDefi=['HHHHH';'L----'];
na=3;
nb=3;
NN=[na,nb,1];
trparms.lambda= 1.1;
trparms.maxiter=1500;
[W1i,W2i,PI_vector,iteration,lambda]=general(NetDefi,NN,[],
[],trparms,T,P);

save('inversofrnlinear.mat','NN', 'NetDefi', 'W1i','W2i');

[Y,PI]=invsim(NetDefi,NN,W1i,W2i,T,P);



figure;
plot(Y,'b');
hold on;
plot(P(nmax:end-1),'r--');
xlabel('Tempo(amostras)');
ylabel('Força(Newton)');
legend('Saída da rede neuronal','Saída da planta');
title('Dados de treino do modelo inverso');

VP=forca(round(0.75*tam)+1:end)';
VT=thetas(round(0.75*tam)+1:end)';

[Ysim,PI]=invsim(NetDefi,NN,W1i,W2i,VT,VP);
%erro de validação
erro2=mse(VP(nmax:end-1)-Ysim);
figure;
plot(Ysim,'b');
hold on;
plot(VP(nmax:end-1),'r--')
xlabel('Tempo(amostras)');
ylabel('Força (Newton)');
legend('Saída da rede neuronal','Saída da planta');
title('Dados de validação do modelo inverso');
```

```
%PROTOTIPO
%SIMULNARMAX1

clear all;
```

```
close all;
trparms=settrain;

nonlinear
sim('pendulonaolinear2',25);

P=forca';
T=thetas';
tam=length(P);
percent=0.75; %percentagem de dados de treino
P=P(1:round(percent*tam));
T=T(1:round(percent*tam));
VP=forca(round(0.75*tam)+1:end)';
VT=thetas(round(0.75*tam)+1:end)';
NetDeff=['HHHHH';'L----'];

[n1,n2]=size(NetDeff);
na=3;
nb=3;
nc=3;
nmax=max(na,nb);
NN=[na,nb,nc,1];
[W1f,W2f,Chat,critvec,iteration,lambda]=nnarmax1(NetDeff,NN,[],[],
[],trparms,T,P)

Y=nnsimul('nnarmax1',NetDeff,NN,W1f,W2f,T,P)


figure;
plot(Y*180/pi);
hold on;
plot(T(nmax+1:end)*180/pi,'r--')
xlabel('Tempo(amostras)');
ylabel('Ângulo (graus)');
legend('Saída da rede neuronal','Saída da planta');
title('Dados de treino do modelo directo');

save('directonarmaxnlinear.mat','NN', 'NetDeff', 'W1f','W2f');
%dados de validação



YsimV = nnsimul('nnarmax1',NetDeff,NN,W1f,W2f,VT,VP);
erro=mse(VT(nmax+1:end)-YsimV);
figure;
plot(180*YsimV/pi);
hold on;
plot(VT(nmax+1:end)*180/pi,'r--')
xlabel('Tempo(amostras)');
ylabel('Ângulo(graus)');
legend('Saída da rede neuronal','Saída da planta');
title('Dados de validação do modelo directo');

%%modelo inverso
NetDefi=['HHHHH';'L------'];
na=2;
nb=2;
NN=[na,nb,1];
trparms.lambda= 1;
```

```
[W1i,W2i,PI_vector,iteration,lambda]=general(NetDefi,NN,[],
[],trparms,T,P);

save('inversofr3.mat','NN', 'NetDefi', 'W1i','W2i');

[Y,PI]=invsim(NetDefi,NN,W1i,W2i,T,P);



figure;
plot(Y,'b');
hold on;
plot(P(nmax:end),'r--');
xlabel('Tempo(amostras)');
ylabel('Força(Newton)');
legend('Saída da rede neuronal','Saída da planta');
title('Dados de treino do modelo inverso');

VP=forca(round(0.75*tam)+1:end)';
VT=thetas(round(0.75*tam)+1:end)';

[YsimV,PI]=invsim(NetDefi,NN,W1i,W2i,VT,VP);
erro2=mse(VP(nmax:end-1)-Ysim);
figure;
plot(YsimV,'b');
hold on;
plot(VP(nmax:end),'r--')
xlabel('Tempo(amostras)');
ylabel('Força (Newton)');
legend('Saída da rede neuronal','Saída da planta');
title('Dados de validação do modelo inverso');
```

```
%PROTOTIPO
%SIMULNNIOL

clear all;
close all;
trparms=settrain;


sim('pendulonaolinear2',25);

P=forca';
T=thetas';
tam=length(P);
percent=0.75; %percentagem de dados de treino
P=P(1:percent*tam);
T=T(1:percent*tam);
VP=forca(round(0.75*tam)+1:end)';
VT=thetas(round(0.75*tam)+1:end)';
NetDeff=['HHHHH';'L----'];
NetDefg=['HHHHH';'L----'];

[n1,n2]=size(NetDeff);
na=2;
nb=2;
```

```
nmax=max(na,nb);
NN=[na,nb,1];
[W1f,W2f,W1g,W2g,critvec,iteration,lambda]=nniol(NetDeff,NetDefg,NN,[],
[],[],[],trparms,T,P)

[Y,NSSE]=ioleval(NetDeff,NetDefg,NN,W1f,W2f,W1g,W2g,T,P)


figure;
plot(Y*180/pi);
hold on;
plot(T(nmax+1:end)*180/pi,'r--')
xlabel('Tempo(amostras)');
ylabel('Ângulo (graus)');
legend('Saída da rede neuronal','Saída da planta');
title('Dados de treino do modelo directo');

save('iolinearizationlinear.mat','NN',
'NetDeff','NetDefg','W1g','W2g','W1f','W2f');
%dados de validação



[YsimV,NSSE]=ioleval(NetDeff,NetDefg,NN,W1f,W2f,W1g,W2g,VT,VP)
figure;
plot(180*YsimV/pi);
hold on;
plot(VT(nmax+1:end)*180/pi,'r--')
xlabel('Tempo(amostras)');
ylabel('Ângulo(graus)');
legend('Saída da rede neuronal','Saída da planta');
title('Dados de validação do modelo directo');
```

```
function Yhat=nnsimul(method,NetDef,NN,W1,W2,Y,U,obsidx)
%  NNSIMUL
%  TEM UM BUG que foi corrigido em relação ao original.Função colocava
como entrada para todas os modelos a saída realimentada quando por
exemplo no modelo narx tem que ser a saída da planta.
%        Simulate a neural network model of a dynamic system from a
sequence
%        of controls alone (not using observed outputs). The simulated
output
%        is compared to the observed output. For NNARMAX and state space
%        models the residuals are set to 0.
%
%
%  Call:
%  Network generated by NNARX (or NNRARX):
%           Ysim = nnsimul('nnarx',NetDef,NN,W1,W2,Y,U)
%  (Likewise for NNOE and NNARMAX1+2)
%
%  Network generated by NNSSIF:
%           Ysim = nnsimul('nnssif',NetDef,nx,W1,W2,Y,U,obsidx)
%
%  Inputs:
```

```
%        See NNVALID/IFVALID
%
%  Output:
%        Ysim: Vector containing the simulated outputs.
%
%  NB! Does not work for models generated by NNIOL.

%  Programmed by : Magnus Norgaard, IAU/IMM, Technical Univ. of Denmark
%  LastEditDate  : July 17, 1996

% >>>>>>>>>>>>>>>>>>>>>>>>>>>>      GET PARAMETERS
<<<<<<<<<<<<<<<<<<<<<<<<<<<<
skip = 1;
if strcmp(method,'nnarx') | strcmp(method,'nnrarx'),
  mflag=1;

elseif strcmp(method,'nnarmax1') | strcmp(method,'nnrarmx1'),
  mflag=2;

elseif strcmp(method,'nnarmax2') | strcmp(method,'nnrarmx2'),
  mflag=3;

elseif strcmp(method,'nnoe'),
  mflag=4;

elseif strcmp(method,'nnssif')
  mflag=5;

else
  disp('Unknown method!!!!!!!!');

end


% >>>>>>>>>>>>>>>>>>>>>>>>>>>>     INITIALIZATIONS
<<<<<<<<<<<<<<<<<<<<<<<<<<<<
[ny,Ndat] = size(Y);                    % # of outputs and # of data
[nu,Ndat] = size(U);                    % # of inputs
na        = NN(1);


% ---------- NNARX/NNOE model ----------
if mflag==1 | mflag==4,
  nb = NN(2:1+nu);
  nc = 0;
  nk = NN(2+nu:1+2*nu);

% --------- NNARMAX1 model --------
elseif mflag==2,
  nb = NN(2:1+nu);
  nc     = 0;
  nk     = NN(2+nu+1:2+2*nu);

% --------- NNARMAX2 model --------
elseif mflag==3,
  nb = NN(2:1+nu);
  nc     = NN(2+nu);
  nk     = NN(2+nu+1:2+2*nu);
end
```

```
nmax     = max([na,nb+nk-1]);            % 'Oldest' signal used as input
to the model
N        = Ndat - nmax;                  % Size of training set
nab      = na+sum(nb);                   % na+nb
nabc     = nab+nc;                       % na+nb+nc
outputs     = 1;                         % Only MISO models considered

% --------- NNSSIF model --------
if mflag==5,
  nx = NN;
  na = nx;
  nab = nx+nu;
  nabc = nab+ny;
  nk      = 1;
  nmax    = 1;                          % 'Oldest' signal used as input to
the model
  N       = Ndat - nmax;                % Size of training set
  outputs = ny;
  obsidx=obsidx(:)';                    % Find row indices
  rowidx=obsidx;
  for k=2:ny,
    rowidx(k)=obsidx(k)+rowidx(k-1);
  end
  nrowidx = 1:nx;                       % Not row indices
  nrowidx(rowidx)=[];
  Cidx=[1 rowidx(1:ny-1)+1];
  C = zeros(ny,nx);
  C(1:ny,Cidx)=eye(ny);
end


% --------- Common initializations --------
L_hidden = find(NetDef(1,:)=='L')';     % Location of linear hidden
neurons
H_hidden = find(NetDef(1,:)=='H')';     % Location of tanh hidden neurons
L_output = find(NetDef(2,:)=='L')';     % Location of linear output
neurons
H_output = find(NetDef(2,:)=='H')';     % Location of tanh output neurons
[hidden,inputs] = size(W1);
inputs          = inputs-1;
y1       = [zeros(hidden,N);ones(1,N)];
Yhat     = zeros(outputs,N);


% >>>>>>>>>>>>>>>>>>>>>>>>>>>   COMPUTE NETWORK OUTPUT
<<<<<<<<<<<<<<<<<<<<<<<<<<<
% ----------  NNARX/NNOE/NNARMAX1/NNARMAX2 model ----------
if mflag==1 | mflag==2 | mflag==3 | mflag==4,

% -----  CONSTRUCT THE REGRESSION MATRIX PHI   -----
  PHI_aug = [zeros(nab,N);ones(1,N)];
  jj  = nmax+1:Ndat;
  for k = 1:na, PHI_aug(k,:)    = Y(jj-k); end
  index = na;
  for kk = 1:nu,
    for k = 1:nb(kk), PHI_aug(k+index,:) = U(kk,jj-k-nk(kk)+1); end
    index = index + nb(kk);
  end
```

```
  % ----- DETERMINE SIMULATED OUTPUT -----
  for t=1:N,
    h1 = W1(:,[1:nab nabc+1])*PHI_aug(:,t);;
    y1(H_hidden,t) = pmntanh(h1(H_hidden));
    y1(L_hidden,t) = h1(L_hidden);

    h2 = W2*y1(:,t);
    Yhat(H_output,t) = pmntanh(h2(H_output,:));
    Yhat(L_output,t) = h2(L_output,:);
    if (mflag==4)
        for d=1:min(na,N-t),
            PHI_aug(d,t+d) = Yhat(:,t);
        end
    end
  end

% ---------- State space model ----------
elseif mflag==5,
  % -----  CONSTRUCT THE REGRESSION MATRIX PHI   -----
  PHI = zeros(inputs,N);
  PHI(nx+1:nx+nu,:) = U(:,1:N);
  PHI_aug = [PHI;ones(1,N)];              % Augment PHI with a row
containing ones

  % ----- DETERMINE SIMULATED OUTPUT -----
  for t=1:N,
    h1 = W1*PHI_aug(:,t);                 % Hidden neuron outputs
    y1(H_hidden,t) = pmntanh(h1(H_hidden));
    y1(L_hidden,t) = h1(L_hidden);

    h2 = W2*y1(:,t);                      % Predicted states
    y2(H_output,t) = pmntanh(h2(H_output,:));
    y2(L_output,t) = h2(L_output,:);
    y2(nrowidx,t)  = y2(nrowidx,t) + PHI_aug(nrowidx+1,t);
    Yhat(:,t)      = C*y2(:,t);

    for d=1:min(1,N-t),
      PHI_aug(1:nx,t+1) = y2(:,t);
    end
  end
end


% >>>>>>>>>>>>>>>>>>>>>>>>>>      PLOT THE RESULTS
<<<<<<<<<<<<<<<<<<<<<<<<<<
si = figure-1;
Y = Y(:,nmax+1:Ndat);
for k=1:outputs,
  if outputs>1,
    figure(si+k);
  end
  plot(Y(k,:),'b-'); hold on
  plot(Yhat(k,:),'r--');hold off
  xlabel('time (samples)')
  if outputs==1,
    title('Output (dashed) and simulated output (solid)')
  else
    title(['Output (dashed) and simulated output (solid)  #' int2str(k)])
  end
```

```
  grid
end
```

```
%PROTOTIPO
%INVERSO_DIRECTO
%função que gera o modelo Simulink para o modelo inverso ou directo e
% inverso_directo(metodo,modelo,W1i,W2i,na,nb,n2)
% Método é a classe de modelos, que tanto pode ser 'narx', para a classe
de modelos NARX, 'narm', para a classe de modelos NARMAX, ou 'nnoe', para
a classe de modelos NNOE. Esta última não foi usada nesta dissertação,
mas que também foi incluída nesta função..
% Modelo pode ser 'inverso', ou 'directo', conforme se queira o modelo
inverso ou directo da planta, respectivamente.
% W1i e W2i ,são os pesos das redes obtidas com a NNCTRL toolbox.
% na e nb são o número de saídas e entradas atrasadas, respectivamente,
utilizadas no modelo.
% n2 é o número de neurónios na camada intermédia.

function narx_net=inverso_directo(metodo,modelo,W1i,W2i,na,nb,n2)

nonlinear
%alterar aqui o nome do modelo Simulink
sim('pendulonaolinear2',25);

P=forca';
T=thetas';
tam=length(P);
percent=0.75;
P=P(1:percent*tam);
T=T(1:percent*tam);




[y]=thetas(1:round(percent*tam))';
[u]=forca(1:round(percent*tam))';

y=con2seq(y); u=con2seq(u);


%passagem da rede para a forma da nn toolbox do matlab
p=[u;y]; t=y;
if  strcmp(modelo,'inverso')



    narx_net2=newnarxsp(minmax(p),[1:nb-1],[1:na+1],[n2 1],
{'tansig','purelin'}, 'trainlm');
    narx_net2.IW{1}=W1i(n2:-1:1,na+2:na+nb);
    narx_net2.IW{3}=W1i(n2:-1:1,1:na+1);

    narx_net2.LW{2}=W2i(n2:-1:1);

    narx_net2.b{1}=W1i(n2:-1:1,na+nb+1);
    narx_net2.b{2}=W2i(n2+1);
```

```
    gensim(narx_net2,0.005);
end

if (strcmp(modelo,'directo'))
    if (strcmp(metodo,'narx'))
        narx_net=newnarxsp(minmax(p),[1:nb],[1:na],[n2 1],
{'tansig','purelin'}, 'trainlm');
        narx_net.IW{1}=W1i(n2:-1:1,na+1:na+nb);
        narx_net.IW{3}=W1i(n2:-1:1,1:na);
        na
        narx_net.LW{2}=W2i(n2:-1:1);

        narx_net.b{1}=W1i(n2:-1:1,na+nb+1);
        narx_net.b{2}=W2i(n2+1);
        gensim(narx_net,0.005);


    end

    if (strcmp(metodo,'nnoe'))
        p=[u];
        narx_net=newnarx(minmax(p),[1:nb],[1:na],[n2 1],
{'tansig','purelin'}, 'trainlm');
        narx_net.IW{1}=W1i(n2:-1:1,na+1:na+nb);
        narx_net.LW{3}=W1i(n2:-1:1,1:na);

        narx_net.LW{2}=W2i(n2:-1:1);

        narx_net.b{1}=W1i(n2:-1:1,na+nb+1);
        narx_net.b{2}=W2i(n2+1);
        gensim(narx_net,0.005);
    end

    if   strcmp(metodo,'narm')
        narx_net=newnarxsp(minmax(p),[1:nb],[1:na],[n2 1],
{'tansig','purelin'}, 'trainlm');
        narx_net.IW{1}=W1i(n2:-1:1,na+1:na+nb);
        narx_net.IW{3}=W1i(n2:-1:1,1:na);

        narx_net.LW{2}=W2i(n2:-1:1);

        narx_net.b{1}=W1i(n2:-1:1,na+nb+1);
        narx_net.b{2}=W2i(n2+1);
        gensim(narx_net,0.005);
    end
end
```

```
%PROTOTIPO
%INVERSO_DIRECTO_NNIOL
%função que gera o modelo Simulink para criar as redes f e g da estrutura
%de controlo linearização por realimentação
%metodo 'narx' 'nnoe' 'narm'
%modelo 'directo 'inverso'
%W1i e W2i pesos dos modelos
%p vector de entrada
%na nº de regressores na saída
```

```
%nº de regressores na entrada
%n2 nº de neurónios na camada intermédia
function narx_net=inverso_directo_nniol(metodo,modelo,W1i,W2i,na,nb,n2)
nonlinear
%alterar aqui o nome do modelo simlink
sim('pendulonaolinear2',50);

P=forca';
T=thetas';
tam=length(P);
percent=0.75;
P=P(1:percent*tam);
T=T(1:percent*tam);



[y]=thetas(1:round(percent*tam))';
[u]=forca(1:round(percent*tam))';

y=con2seq(y); u=con2seq(u);


%passagem da rede para a forma da nn toolbox do matlab
p=[u;y]; t=y;

if (strcmp(modelo,'directo'))
    if (strcmp(metodo,'narx'))
        narx_net=newnarxsp(minmax(p),[1:nb-1],[0:na-1],[n2 1],
{'tansig','purelin'}, 'trainlm');
        narx_net.IW{1}=W1i(n2:-1:1,na+1:na+nb-1);
        narx_net.IW{3}=W1i(n2:-1:1,1:na);

        narx_net.LW{2}=W2i(n2:-1:1);

        narx_net.b{1}=W1i(n2:-1:1,na+nb);
        narx_net.b{2}=W2i(n2+1);
        gensim(narx_net,0.005);


    end

    if (strcmp(metodo,'nnoe'))

        narx_net=newnarx(minmax(p),[1:nb-1],[0:na-1],[n2 1],
{'tansig','purelin'}, 'trainlm');
        narx_net.IW{1}=W1i(n2:-1:1,na+1:na+nb-1);
        narx_net.IW{3}=W1i(n2:-1:1,1:na);

        narx_net.LW{2}=W2i(n2:-1:1);

        narx_net.b{1}=W1i(n2:-1:1,na+nb);
        narx_net.b{2}=W2i(n2+1);
        gensim(narx_net,0.005);
    end

    if   strcmp(metodo,'narm')
        narx_net=newnarxsp(minmax(p),[1:nb-1],[0:na-1],[n2 1],
{'tansig','purelin'}, 'trainlm');
```

```
        narx_net.IW{1}=W1i(n2:-1:1,na+1:na+nb-1);
        narx_net.IW{3}=W1i(n2:-1:1,1:na);

        narx_net.LW{2}=W2i(n2:-1:1);

        narx_net.b{1}=W1i(n2:-1:1,na+nb);
        narx_net.b{2}=W2i(n2+1);
        gensim(narx_net,0.005);
    end
end
```

```
%PROTOTIPO
%solvematrix
%xdotdot=2ª derivada de x em ordem ao tempo
%thetadotdot=2ªderivada de theta em ordem ao tempo
%x1=x
%x2=theta
%x3=1ª deriva de x em ordem ao tempo
%x4=1ª derivada de theta em ordem ao tempo
sol=solve('xdotdot -1/M * ( F - m * ( xdotdot + l * sin (x2) * x4 ^ 2-l *
cos (x2) * thetadotdot) - b * x3 ) = 0 ' , 'thetadotdot - ( 1/I ) * ( m *
(xdotdot + l * sin(x2) * x4^2-l * cos(x2) * thetadotdot) * l* cos (x2) +
m * l * sin (x2) * ( -l * cos(x2) * x4^2 - l * sin (x2) *thetadotdot +
g )) = 0',' xdotdot ','thetadotdot')
```

```
%PROTOTIPO
%preprodata
%pré-processamento de dados
[forca2,xs]=dscale(forca');
[thetas2,xs2]=dscale(thetas');
lipschit(forca2,thetas2,[1:10],[1:10]);
```

Anexos B - Exemplo de uso do programa NELINSYS

Exact linearisation for SISO systems

State-space equations: x = f(x) + g(x) u

y = h(x)

System order: 4

Specify parameters' identifiers (separated by spaces), if there are any: M m l I b g

System matrix: f(x) = [x3;x4;-(m^2*l^2*cos(x2)*g*sin(x2)-sin(x2)*l^3*m^2*x4^2-I*m*l*x4^2*sin(x2)-l^2*m*b*x3-I*b*x3)/(-I*M-I*m-l^2*m*M-l^2*m^2+cos(x2)^2*l^2*m^2);-m*l*(m*g*sin(x2)-cos(x2)*m*x4^2*sin(x2)*l-cos(x2)*b*x3+g*sin(x2)*M)/(-I*M-I*m-l^2*m*M-l^2*m^2+cos(x2)^2*l^2*m^2)]

Input matrix: g(x) = [0;0;(I+l^2*m)/(-I*M-I*m-l^2*m*M-l^2*m^2+cos(x2)^2*l^2*m^2);-m*l*cos(x2)/(-I*M-I*m-l^2*m*M-l^2*m^2+cos(x2)^2*l^2*m^2)];

Output matrix: h(x) = x2

Em "System order" deve ser especificado o número de variáveis de estado do vector x, neste caso 4. Em "Specify parameters identifiers (separated by spaces), if there are any:" deve ser especificado os parâmetros físicos do pêndulo invertido. Em "System Matrix" deve ser especificado a Eq. 4.27. Em "Input Matrix" deve ser especificado a Eq. 4.28. Em "Output matrix" deve ser especificado a Eq. 4.26.

A função devolve a matriz de transformação[q1;q2], o grau relativo do sistema em "Relative degree of the system" e o sinal de controlo *u*. No caso do pêndulo invertido:

Transformation equations:

q1 = x2 ;q2 = x4

Relative degree of the system: 2

Nonlinear feedback:

u =- (l.m.g.sin(x2) – l.m.cos(x2).x4.sin(x2)–l.m.cos(x2).b.x3+ l.m.g.sin(x2).M–v.I.M-v.I.m-v.l.m.M-v.l.m+v.cos(x2). l.m)/(l.m.cos(x2))

A matriz de transformação, devolve a forma como devem ser afectadas as variáveis de estado(neste caso o ângulo e a sua derivada), de forma a obter o controlo do sistema linearizado, por realimentação das variáveis de estado. O tamanho desta matriz é igual a r. No caso do pêndulo invertido, as variáveis x2 e x4 são realimentadas directamente da saída do sistema, sem operações algébricas, tal como sugerido pela matriz de transformação.