UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Análise de um Controlador Baseado no Jacobiano Estimado da Planta Através de uma Rede Neural

Pedro Berretta de Lucena

Orientador: Prof. Dr. Fábio Meneghetti Ugulino de Araújo

Co-orientador: Prof. Dr. Andrés Ortiz Salazar

Natal, RN, dezembro de 2005

UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Análise de um Controlador Baseado no Jacobiano Estimado da Planta Através de uma Rede Neural

Pedro Berretta de Lucena

Orientador: Prof. Dr. Fábio Meneghetti Ugulino de Araújo

Co-orientador: Prof. Dr. Andrés Ortiz Salazar

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da UFRN (área de concentração: Automação e Sistemas) como parte dos requisitos para obtenção do título de Mestre em Ciências.

Natal, RN, dezembro de 2005

Análise de um Controlador Baseado no Jacobiano Estimado da Planta Através de uma Rede Neural

Pedro Berretta de Lucena

Dissertação de mestrado aprovada em 16 de dezembro de 2005 pela banca examinadora composta pelos seguintes membros:

_____

Prof. Dr. Fábio Meneghetti Ugulino de Araújo (orientador)

_____

Prof. Dr. Andrés Ortiz Salazar (Co-Orientador)

_____

Prof. Dr. André Laurindo Maitelli (Examinador Interno)

_____

Prof. Dr. Oscar Gabriel Filho (Examinador Externo)

Agradecimentos

Agradeço aos meus pais pela motivação que me deram para realizar este trabalho. Agradeço ao meu orientador, Prof. Dr. Fábio Meneghetti Ugulino de Araújo, que me acompanhou desde o início me orientando, ajudando e incentivando bastante. Da mesma forma, agradeço ao Prof. Dr. Andrés Ortiz Salazar e ao Prof. Dr. André Laurindo Maitelli, que também estiveram presentes desde o início do trabalho. Agradeço ao Prof. Dr. Francisco das Chagas Mota e ao Prof. Dr. Adrião Duarte Dória Neto por nossas discussões sobre este trabalho. Tenho um agradecimento especial ao Prof. Dr. Oscar Gabriel Filho, que, mesmo sendo de outra instituição, colaborou com o trabalho. Agradeço também ao Eng. José Soares da Costa Neto pelo tempo que dedicou para nossos estudos em sua casa. Meus agradecimentos também vão para o Eng. Wagner Augusto Frantz e para o Eng. Miracy Teixeira de Araújo Júnior por me ajudarem na diagramação desta dissertação e correção de erros. Agradeço ao Prof. Dr. Allan de Medeiros Martins por seu incentivo e ajuda. Também são merecedores do meu agradecimento o Prof. Dr. Felipe Mirapalheta e seu grupo de pesquisa que me ajudaram bastante com seus conhecimentos.

Resumo

Este trabalho apresenta uma análise da lei de controle baseada em um esquema híbrido indireto usando rede neural, proposto inicialmente por O. Adetona, S. Sathanathan e L. H. Keel. Implementações dessa lei de controle, para uma planta de nível de segunda ordem, resultaram em um comportamento oscilatório, mesmo com a convergência do identificador neural. Tais resultados motivaram a investigação da aplicabilidade dessa lei. A partir disso, foram feitas análises matemáticas de estabilidade e diversas implementações, com plantas simuladas e com plantas reais, com a finalidade de se analisar o problema. A análise mostrou que a lei foi desenvolvida desprezando-se certos componentes da dinâmica da planta a ser controlada. Sendo assim, para plantas onde esses componentes têm uma influência significativa em sua dinâmica, a lei tende a falhar.

Palavras-chave: Controle Inteligente, Controle Adaptativo, Redes Neurais, Inteligência Artificial.

Abstract

This work presents an analysis of the control law based on an indirect hybrid scheme using neural network, initially proposed for O. Adetona, S. Sathanathan and L. H. Keel. Implementations of this control law, for a level plant of second order, was resulted an oscillatory behavior, even if the neural identifier has converged. Such results had motivated the investigation of the applicability of that law. Starting from that, had been made stability mathematical analysis and several implementations, with simulated plants and with real plants, for analyze the problem. The analysis has been showed the law was designed being despised some components of dynamic of the plant to be controlled. Thus, for plants that these components have a significant influence in its dynamic, the law tends to fail.

Keywords: Intelligent Control, Adaptive Control, Neural Networks, Artificial Intelligence.

Sumário

Lista de Figuras

Capítulo 1

Introdução

Há alguns anos, a inteligência artificial (IA) vem sendo uma área bastante promissora para contribuir com a área de controle. Um bom motivo para isso, é que há uma certa dificuldade em se controlar sistemas não-lineares ou variantes no tempo com as técnicas convencionais de controle, e a inteligência artificial pode suprir essa falta. As redes neurais artificiais (RNA), ou simplesmente redes neurais, são um bom exemplo de ferramenta da inteligência artificial que desempenha esse importante papel de complementar ou substituir o controle convencional. Isso se dá devido à sua adaptabilidade através do aprendizado, generalização do mesmo e não-linearidade.

O primeiro sistema de controle neural foi provavelmente um desenvolvido por Widrow e Smith [1] em 1963. Eles usaram uma rede neural *Adaline* para estabilizar e controlar o ato de equilibrar um pêndulo invertido em cima de um carro. O carro se movia ao longo de um trilho unidimensional finito e reto. O controlador foi exigido para equilibrar o pêndulo invertido na posição vertical, mantendo a posição do carro dentro de certos limites. Outras demonstrações de controle neural foram apresentadas posteriormente por outros autores como, por exemplo, Waltz e Fu [2] em 1965; Michie e Cahmbers [3] em 1968; e Barto et al. [4] em 1983.

Um grande interesse em usar redes neurais para controle somente começou em torno de 1987. A partir da primeira Conferência do IEEE (*Institute of Electrical and Electronics Engineers*), tem-se um significativo aumento de conferências e artigos de jornais publicados sobre controle. Esses artigos têm demonstrado que as redes neurais podem ser aplicadas com

sucesso para controle de sistemas não-lineares com dinâmica desconhecida [5]. Várias novas estruturas de controle neural foram propostas. Por exemplo, "*feedback error learning*" por Kawato et al. [6] em 1987, "*neural internal model control*" por Hunt e Sbarbaro [7] em 1991, "*neural predictive control*" por Willis et al. [8] em 1992, "*forward and inverse modeling*" por Jordan e Jacobs [9] em 1990, "*neurofuzzy*" por Harris et al. [10] em 1993, "*generalized and specialized learnings*" por Psaltis et al. [11] em 1988.

A partir de 1992, ocorreu um aumento no uso de redes neurais com outros controladores. Esses tipos de estruturas híbridas tentam tirar vantagem dos pontos fortes de diferentes controladores e evitar as falhas dos esquemas de controle neural puros [5].

Conforme Narendra [12], em 1992, o controle adaptativo de sistemas não-lineares usando redes neurais iria evoluir, em termos gerais, na mesma direção que o controle adaptativo de sistemas lineares seguiu nas últimas duas décadas.

Em 2001, O. Adetona, S. Sathananthan e L. H. Keel participaram de um projeto patrocinado pela NASA (*National Aeronautics and Space Administration*) e publicaram um artigo [13] na "*Proceedings of the American Control Conference*", em Arlington, propondo uma lei de controle baseada no jacobiano da planta, ou seja, na derivada parcial da saída da planta com relação à entrada. Eles utilizaram uma rede neural RBF (de função de base radial), com treinamento em tempo de execução, para identificar a planta, possibilitando o cálculo do jacobiano a partir do modelo identificado da planta. Assim, o sistema de controle pode se auto-ajustar para atender a novos parâmetros de projeto ou à variação da planta. A estrutura de modelo do identificador neural utilizado foi o NNARX (*Neural Network AutoRegressive, eXternal input*) [14].

Já em 2003, A. L. Maitelli e O. Gabriel Filho [15] também chegaram a essa mesma lei de controle, porém, eles utilizaram uma rede neural MLP (*multilayer perceptron*) para identificar a planta. O algoritmo de treinamento da rede neural utilizado foi o de retropropagação [16].

Em ambas as publicações, o funcionamento do sistema de controle foi validado para uma planta não-linear de segunda ordem simulada, seguindo uma referência que variava no tempo.

Em 2005, implementamos esse esquema de controle para controlar uma planta simulada de nível de segunda ordem da *Quanser* [17], configuração #2, representada na Figura 1.1. Os resultados foram insatisfatórios, apesar de mostrar certa eficácia, devido à saída da planta ficar oscilando em torno da referência e não convergir para ela.

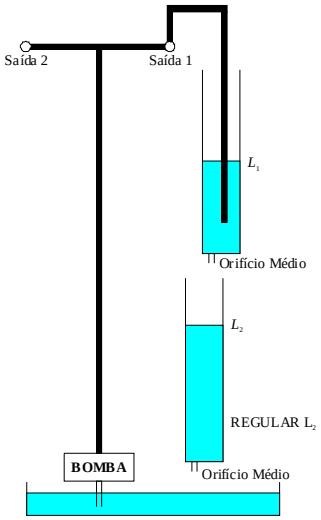

Figura 1.1 – Planta de nível de segunda ordem da *Quanser*, configuração #2.

Devido a esse comportamento, resolveu-se investigar o que estava acontecendo e testar o sistema de controle com a planta real da *Quanser* [18]. Resultados semelhantes foram obtidos. Algumas vezes, o sistema ficava oscilando muito devido ao sinal de controle ter valores absolutos muito grandes sendo saturados para não danificar o equipamento. Isso fazia com que o sinal de controle ficasse com uma forma de onda retangular. Então, resolveu-se testar com outras plantas da *Quanser* que estavam disponíveis no Laboratório de Engenharia de Computação e Automação (LECA), da Universidade Federal do Rio Grande do Norte (UFRN). O esquema de controle foi testado em um servo-motor [19] e as mesmas características foram observadas. Mais testes foram feitos com a planta *ball and beam* [19] e o controlador não conseguiu controlar adequadamente.

Para se entender melhor o que estava acontecendo, foi feita uma análise com plantas lineares de primeira ordem simuladas (em um computador digital) e reais (em um computador analógico [20]). Foi usado um modelo genérico de plantas lineares de primeira ordem com atraso unitário para se analisar o comportamento resultante em várias plantas atribuindo-se valores para os parâmetros do modelo genérico.

Uma breve introdução às redes neurais é apresentada no Capítulo 2, onde também, o algoritmo de retropropagação é comentado. O Capítulo 3 trata do esquema de controle

[13],[15]; da estrutura de modelo utilizada como identificador neural; e da lei de controle. No Capítulo 4, é feita uma análise matemática de estabilidade para a lei de controle e a determinação das condições de estabilidade para plantas lineares de primeira ordem com atraso unitário, sendo as principais contribuições deste trabalho. Outra contribuição que foi feita é a validação da lei de controle através de análises gráficas por meio de simulação computacional e de implementação prática de plantas feitas em um computador analógico. Alguns resultados previamente obtidos são mostrados e comentados no Capítulo 5. O Capítulo 6 traz a conclusão e alguns comentários sobre suas perspectivas.

Capítulo 2

Redes Neurais

Para se entender o funcionamento de uma rede neural, é importante entender antes o funcionamento de um neurônio artificial. Um neurônio artificial é um modelo aproximado de um neurônio biológico. Warren McCulloch, neurofisiologista, e Walter Pitts, matemático, propuseram um modelo de neurônio (Figura 2.1) que é base para grande parte das redes neurais hoje em dia.

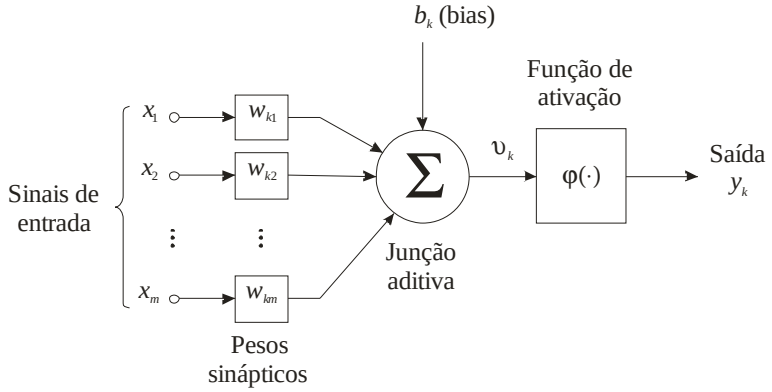

Figura 2.1 – Modelo de McCulloch e Pitts de um neurônio.

Um modelo de neurônio bastante utilizado que é uma pequena variação do modelo de McCulloch e Pitts é o mostrado na Figura 2.2. Matematicamente não há grande diferença, porém, ela existe na interpretação do *bias* (b_k), que passa a ser considerado como um peso sináptico de uma entrada fixa com valor um, e na função de ativação φ, que deixa de ser exclusivamente do tipo limiar e passa a ser uma função matemática qualquer.

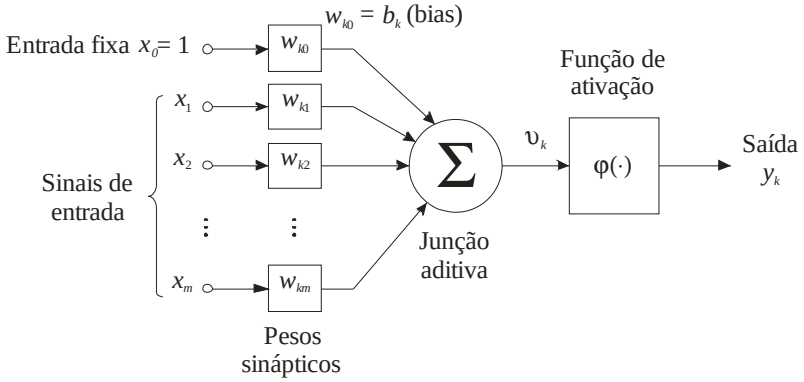

Figura 2.2 – Modelo de um neurônio baseado no modelo de McCulloch e Pitts.

Com esse modelo mostrado da Figura 2.2, podemos definir alguns parâmetros:

$$\upsilon_k = \sum_{j=0}^{m} w_{kj} x_j \tag{2.1}$$

$$y_k = \varphi(\upsilon_k) \tag{2.2}$$

onde υ_k é o campo local induzido do neurônio k; y_k é a saída do neurônio k; $\{w_{kj}\}$ é o conjunto de pesos que ligam nós j à entrada do neurônio k; o peso w_{k0} é o *bias* (b_k) do neurônio k; $\varphi(\upsilon_k)$ é a função de ativação, que, para os algoritmos de treinamento que utilizam sua derivada, como o de retropropagação discutido neste capítulo, deve ser uma função continuamente derivável. Nesse caso, as funções mais utilizadas são definidas nas expressões (2.3), (2.4) e (2.5):

$$\varphi(\upsilon) = \sigma\upsilon, \quad \sigma > 0 \tag{2.3}$$

$$\varphi(\upsilon) = \frac{1}{1+e^{-\sigma\upsilon}}, \quad \sigma > 0 \tag{2.4}$$

$$\varphi(\upsilon) = \rho\tanh(\sigma\upsilon), \quad (\rho,\sigma) > 0 \tag{2.5}$$

onde ρ e σ são constantes.

A expressão (2.3) define uma função linear; a expressão (2.4) define a função logística, uma função não-linear do tipo sigmóide unipolar com amplitude dentro do intervalo $[0;1]$; e a expressão (2.5) define a função tangente hiperbólica, função não-linear do tipo sigmóide bipolar com amplitude dentro do intervalo $[-\rho;\rho]$. O termo sigmóide é devido ao

gráfico da função possuir um formato semelhante a um “s”. Na Figura 2.3 são mostrados gráficos das três funções.

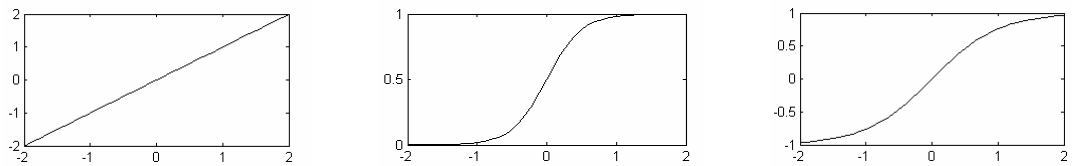

Figura 2.3 – (a) Gráfico da função linear com $\sigma = 1$. (b) Gráfico da função logística com $\sigma = 4$. (c) Gráfico da função tangente hiperbólica com $\rho = \rho = 1$.

Uma rede neural consiste em uma rede de neurônios com entradas conectadas a nós de fonte (valores externos) ou computacionais (saídas de neurônios). Existem várias arquiteturas de redes neurais, porém aqui só será abordada a arquitetura das redes MLP (*MultiLayer Perceptron*). Um MLP consiste em uma rede neural organizada em camadas. A camada de entrada é uma camada de nós de fonte, ou seja, não possui neurônios, já a camada de saída e as camadas ocultas (escondidas ou intermediárias), são constituídas de neurônios (nós computacionais). A Figura 2.4 ilustra um exemplo de MLP de configuração 5-2-3 (cinco nós na camada de entrada, dois nós na camada oculta, três nós na camada de saída).

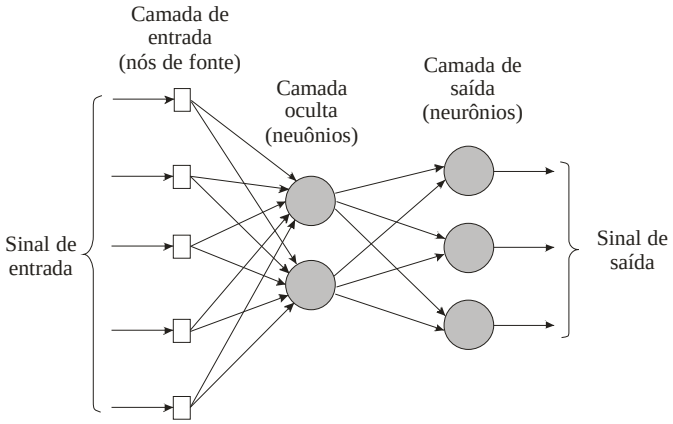

Figura 2.4 – Exemplo de uma rede MLP de configuração 5-2-3.

Uma observação importante a se fazer é que cada neurônio da rede neural MLP tem um *bias*, que é um peso correspondente a uma entrada fixa, de valor igual a um, que está implícita na representação da rede neural, e não está sendo considerada uma entrada externa, mas um parâmetro interno da rede. Uma forma prática de se implementar o *bias* em um MLP

é adicionando um nó de valor fixo igual a um em cada camada da rede, exceto na camada de saída.

2.1. Algoritmo de Retropropagação

Para treinarmos um MLP, uma das alternativas é utilizar o algoritmo de retropropagação (*back-propagation*) [16]. Considere o erro na saída do neurônio *j*, na iteração *n* (i.e. a apresentação do *n*-ésimo exemplo de treinamento), como sendo:

$$e_j(n) = d_j(n) - y_j(n), \quad \text{o neurônio } j \text{ é um nó de saída} \tag{2.6}$$

onde $d_j(n)$ é o valor desejado para a saída $y_j(n)$ do neurônio j na iteração n. O valor instantâneo da energia do erro para o neurônio *j* é definido como sendo $\frac{1}{2}e_j^2(n)$.

O valor instantâneo $\xi(n)$ da energia total do erro é obtido somando-se os termos $\frac{1}{2}e_j^2(n)$ de todos os neurônios da camada de saída. Podemos então escrever

$$\xi(n) = \frac{1}{2}\sum_{j \in C} e_j^2(n) \tag{2.7}$$

onde o conjunto *C* inclui todos os neurônios da camada de saída da rede.

Consideremos que N represente o número total de padrões (exemplos) contidos no conjunto de treinamento. Para se obter a energia média do erro quadrático, soma-se os $\xi(n)$ para todos os n e depois divide-se pelo tamanho do conjunto *N*, como segue:

$$\xi_{\text{med}} = \frac{1}{N}\sum_{n=1}^{N} \xi(n) \tag{2.8}$$

O objetivo do processo de aprendizagem é ajustar os parâmetros livres da rede (pesos e *bias*) para minimizar a função de custo ξ_{med} [16]. Para isso, é utilizada a correção $\Delta w_{ji}(n)$ aplicada ao peso $w_{ji}(n)$ na iteração *n*; a correção é definida por

$$\Delta w_{ji}(n) = \eta\, \delta_j(n)\, y_i(n) \tag{2.9}$$

onde o gradiente local da camada *l* é definido por

$$\delta_j^{(l)}(n) = \begin{cases} e_j^{(l)}(n)\varphi_j'\left(v_j^{(l)}(n)\right) & \text{, neurônio } j \text{ da camada de saída } l \\ \varphi_j'\left(v_j^{(l)}(n)\right)\sum_k \delta_k^{(l+1)}(n) w_{kj}^{(l+1)}(n) & \text{, neurônio } j \text{ da camada oculta } l \end{cases} \tag{2.10}$$

em que *k* são os neurônios da camada de saída e o apóstrofo em $\varphi'_j(\cdot)$ representa a diferenciação em relação ao argumento. As funções definidas pelas equações (2.11), (2.12) e (2.13), a seguir, expressam a derivada das funções de ativação representadas nas equações (2.3), (2.4) e (2.5) respectivamente:

$$\varphi'(v) = \sigma \tag{2.11}$$

$$\varphi'(v) = \sigma\, y_j(n)\left[1 - y_j(n)\right] \tag{2.12}$$

$$\varphi'(v) = \frac{\sigma}{\rho}\left[\rho - y_j(n)\right]\left[\rho + y_j(n)\right] \tag{2.13}$$

O ajuste dos pesos sinápticos da rede na camada *l* é feito de acordo com a seguinte equação:

$$w_{ji}^{(l)}(n+1) = w_{ji}^{(l)}(n) + \Delta w_{ji}^{(l)}(n) \tag{2.14}$$

Devido a um problema de convergência para um mínimo local, aconselha-se utilizar a correção como sendo [16]:

$$\Delta w_{ji}^{(l)}(n) = \eta\, \delta_j^{(l)}(n)\, y_i^{(l-1)}(n) + \alpha\, \Delta w_{ji}^{(l)}(n-1) \tag{2.15}$$

onde η é o parâmetro da taxa de aprendizagem e α é o parâmetro de momento [16]. O valor para α de 0,95, por exemplo, é sugerido por Demuth e Beale [21].

2.2. Treinamento Seqüencial e por Lote

Há dois modos de se treinar uma rede neural MLP utilizando o algoritmo de retropropagação: seqüencial ou por lote. O treinamento seqüencial é feito com a apresentação de um exemplo de cada vez para ajustar todos os pesos, repetindo esse processo até atingir uma tolerância desejada para a energia média do erro quadrático ξ_{med} ou um número determinado de iterações ou uma tolerância desejada para o erro máximo absoluto. Já no treinamento por lote, todos os exemplos são apresentados e então os pesos sinápticos são ajustados camada por camada, a partir da camada de saída até a camada de entrada; esse procedimento se repete até que uma tolerância desejada para a energia média do erro quadrático ξ_{med} seja atingida ou que se atinja um número máximo de iterações determinado ou uma tolerância para o erro máximo absoluto. O treinamento seqüencial, de uma forma geral, é mais rápido, que o treinamento por lote. Porém, o treinamento por lote faz o erro oscilar em intervalos pequenos, comparados com os intervalos de oscilação no treinamento seqüencial, na busca do mínimo global. Levando em consideração essas características, foi escolhido o método de treinamento seqüencial para a implementação do sistema.

Segundo Haykin [16], é uma boa prática tornar aleatória a ordem de apresentação dos exemplos de treinamento no treinamento seqüencial. É bom também tornar aleatórios os valores iniciais para os pesos da RNA em um intervalo pequeno e de simétrico, por exemplo, $[-1;1]$.

2.3. Acelerando a convergência

Uma heurística para acelerar a convergência é sugerida por Demuth e Beale [21]. Ela sugere a utilização de um parâmetro da taxa de aprendizagem η variável. Esse η é ajustado da seguinte forma:

- Se a variação, de uma época para outra, da função de custo ξ_{med} estiver de 0 a 4%, o parâmetro da taxa de aprendizagem η não sofre alteração; se for inferior a 0%, será aumentado em 5%; se for maior que 4%, será reduzido em 30%.

Haykin [16] já apresenta quatro heurísticas diferentes para acelerar a convergência da aprendizagem por retropropagação de um MLP. São elas:

- Cada parâmetro ajustável deve ter seu próprio parâmetro da taxa de aprendizagem (η).
- Cada η deve poder variar de uma iteração para outra.
- Quando a derivada da função de custo ξ'_{med} em relação ao peso sináptico *w* tem o mesmo sinal algébrico para iterações consecutivas do algoritmo, o parâmetro η para aquele peso particular deve ser aumentado.
- Quando o sinal algébrico da derivada da função de custo ξ'_{med} em relação a um peso sináptico particular *w* alterna-se para várias iterações consecutivas do algoritmo, o parâmetro η para aquele peso deve ser reduzido.

Essas heurísticas que sugerem a variação do parâmetro da taxa de aprendizagem, além de acelerar a convergência, ajudam a evitar que os pesos sinápticos, quando ajustados, façam a função de custo oscilar passando pelo mínimo a cada iteração do algoritmo de treinamento sem convergir para ele ou sem atingir a tolerância estipulada, pois, quando isso ocorre, o parâmetro da taxa de aprendizagem é reduzido para diminuir o passo.

Um algorítimo de treinamento que implementa essas últimas quatro heurísticas é o *delta-bar-delta* [22]. Ele utiliza um treinamento por lote em que, a cada iteração, os parâmetros da taxa de aprendizagem sejam alterados de acordo com a variação do sinal da derivada da função de custo ξ'_{med} com relação aos pesos w.

Para complementar essas quatro últimas heurísticas, pode-se acrescentar outra:

- Somente o parâmetro da taxa de aprendizagem e o sinal da derivada da função de custo ξ'_{med} com relação ao peso sináptico *w* devem ser usados para ajustá-lo, e não o valor de ξ'_{med}.

Essa última heurística evita que os pesos das camadas ocultas sejam ajustados de forma mais lenta que na camada de saída. Quando se usa o valor da derivada da função de custo ξ'_{med} com relação ao peso w para ajustá-lo, quanto mais próximo w estiver da camada de entrada, menor será o valor de ξ'_{med}, fazendo com que o ajuste de w seja menor, tornando mais lenta a convergência da rede. Além disso, o valor de ξ'_{med} não tem relação alguma com o valor ótimo para o ajuste do peso Δw, podendo ter valores altos e baixos durante o treinamento dependendo do seu andamento. Isso até atrapalha no ajuste do valor ótimo para o parâmetro da taxa de aprendizagem de w.

O algoritmo Rprop (*Resilient propagation*) [23] engloba essa última heurística às outras quatro anteriormente citadas, trabalhando no modo de treinamento por lote. Com uma pequena modificação do algoritmo RPROP, Igel e Hüsken [24] conseguiram uma melhora notável no desempenho do treinamento de uma rede neural, esse algoritmo foi chamado de iRprop (*improved Rprop*).

Capítulo 3

Esquema de Controle

Diversas estratégias de controle neural já foram propostas, sendo assim, não é viável avaliar profundamente todas elas. É importante se fazer uma pré-avaliação para se ter noção das mais promissoras antes da escolha de uma específica para se aprofundar e descobrir as falhas e como corrigi-las. Neste capítulo, serão mostradas as características mais relevantes de alguns esquemas de controle neural adaptativo a fim de se fazer uma escolha de um para se aprofundar. Os esquemas de controle neurais adaptativos são aqueles que utilizam o treinamento da rede neural (ou das redes neurais) em tempo real (*on-line*) para se adaptar à planta a ser controlada. Essa opção de se escolher um controle adaptativo foi feita na busca de um controlador mais genérico possível, onde possa se adaptar às diversas plantas que for controlar.

Existem duas categorias de controle adaptativo [25]: indireto e direto. No *controle adaptativo indireto*, os parâmetros da planta são estimados em tempo real e usados para calcular os parâmetros do controlador. Esse esquema também é denominado de *controle adaptativo explícito*, porque o seu projeto é baseado em um modelo explícito da planta. Já no *controle adaptativo direto*, o modelo da planta é parametrizado em termos dos parâmetros do controlador que são estimados diretamente sem cálculos intermediários envolvendo parâmetros estimados da planta. Esse esquema também é denominado de *controle adaptativo implícito* porque seu projeto é baseado na estimação de um modelo implícito da planta.

Os esquemas de controle usando rede neural, também podem ser divididos em dois grupos [5]: não-híbridos e híbridos. Os esquemas de controle neural não-híbridos são

controlados somente pelas redes neurais. Já os esquemas de controle neural híbridos, utilizam redes neurais trabalhando com outros controladores.

Controle não-híbrido direto

Nos esquemas não-híbridos diretos, como os da Figura 3.1 e Figura 3.2 [5], o treinamento da rede neural (controlador neural) tem como objetivo zerar a diferença entre o valor desejado para a saída da planta (referência) e a saída da planta, e não a diferença entre um valor desejado para a saída da rede neural e a saída da rede neural. Isso implica na necessidade de retro-propagar o erro através da planta, ou seja, é necessário obter a saída da planta para cada iteração do algoritmo de treinamento. Assim, só é possível se fazer um ajuste nos pesos da rede neural uma vez a cada período de amostragem, pois a saída da planta só é lida a cada período de amostragem.

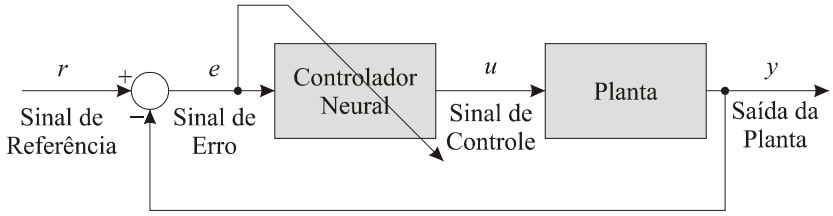

Figura 3.1 – Controle não-híbrido direto em malha fechada.

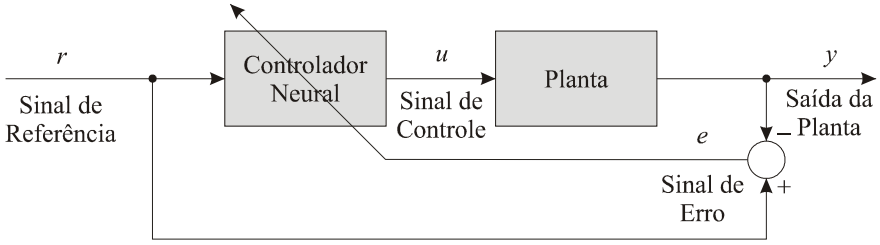

Figura 3.2 – Controle não-híbrido direto em malha aberta.

Controle não-híbrido indireto

Nos esquemas não-híbridos diretos, devido ao algoritmo de treinamento do controlador necessitar das saídas da planta para se calcular o erro e ajustar os pesos, e, como a saída só é lida a cada período de amostragem, só é possível se fazer uma correção de pesos por período de amostragem.

Já no esquema não-híbrido indireto da Figura 3.3, é possível se fazer vários ajustes de pesos durante um período de amostragem, pois existe um modelo da planta identificado por uma rede neural que pode ser usado para se ter uma saída estimada da planta sem a necessidade de se esperar o próximo período de amostragem. O treinamento do controlador pode ser feito juntando o identificador já treinado com o controlador e fixando-se os pesos do identificador. Outra forma de se fazer o treinamento é calculando o jacobiano do identificador já treinado (jacobiano estimado) e utilizando como se fosse o jacobiano da planta e assim faz-se o treinamento utilizando o jacobiano estimado como se estivesse retro-propagando o erro através da planta.

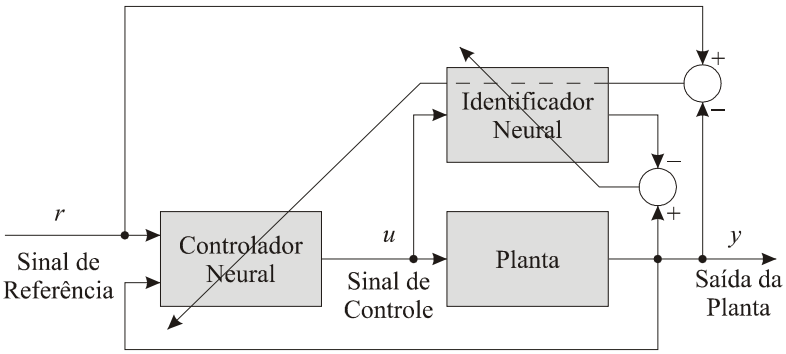

Figura 3.3 – Controle não-híbrido indireto.

Controle híbrido indireto

Para evitar a necessidade de treinamento de duas redes neurais utilizadas no esquema de controle não-híbrido indireto, pode-se utilizar um esquema de controle híbrido indireto substituindo-se o controlador neural por uma lei de controle como mostrado na Figura 3.4, eliminando, assim, o tempo gasto no treinamento do controlador neural, e, conseqüentemente, reduzindo o tempo utilizado durante um período de amostragem possibilitando sua redução.

Esse foi o esquema de controle escolhido para ser avaliado neste trabalho, com uma lei de controle que utiliza o jacobiano do modelo identificado pela rede neural [13],[15].

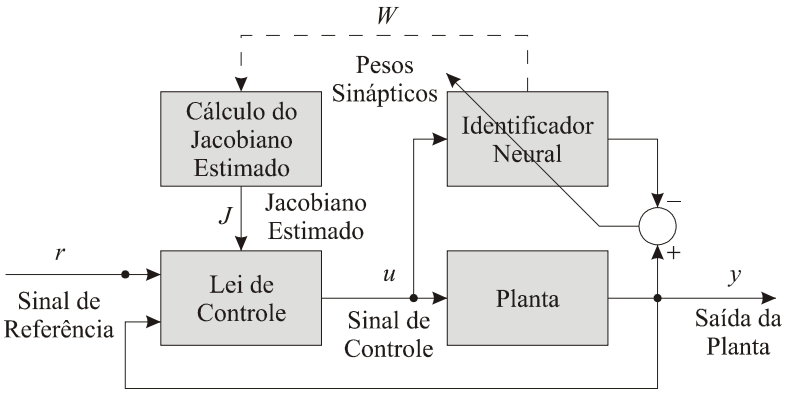

Figura 3.4 – Controle híbrido indireto.

3.1. O Identificador Neural

Identificação de um sistema é a tarefa de inferir uma descrição matemática, um modelo, de um sistema dinâmico a partir de uma série de medidas do sistema [14]. Existem vários motivos para se estabelecer descrições matemáticas de sistemas dinâmicos. Aplicações típicas abrangem simulação, predição, detecção de falha e desenvolvimento de sistema de controle.

As estruturas de modelo baseadas em rede neural, apropriadas para identificação de sistemas não-lineares, são generalizações das estruturas de modelo linear. Elas são caracterizadas por seu vetor de regressão, ou seja, por um vetor que contém as variáveis usadas para se estimar a saída do sistema. Algumas estruturas de modelo linear são: FIR (*Finite Impulse Response*), ARX (*AutoRegressive, eXternal input*), ARMAX (*AutoRegressive, Moving Average, eXternal input*), OE (*Output Error*) e SSIF (*State Space Innovations Form*).

Dependendo da escolha do vetor de regressão, diferentes estruturas de modelo neural emergem. Se o vetor de regressão for selecionado como para modelos ARX, a estrutura de modelo é chamada NNARX (*Neural Network* ARX). Do mesmo modo, NNFIR, NNARMAX, NNOE e NNSSIF.

As estruturas de modelo NNFIR e NNARX são BIBO (*Bounded Input, Bounded Output* – entrada limitada, saída limitada) estáveis por não possuírem realimentação da saída estimada. Porém, o vetor de regressão da estrutura NNFIR não possui valores atrasados da

saída da planta, exigindo um número maior de valores a ser usado no vetor de regressão composto somente de valores atrasados da entrada da planta. Isso faz da estrutura de modelo NNARX a estrutura preferida quando o sistema a ser modelado é determinístico ou o nível de ruído é insignificante [14].

Devido a esses motivos, a estrutura de modelo escolhida para identificar a planta foi NNARX. A expressão matemática da estrutura de modelo não-linear estendida NNARX é descrita como

$$\hat{y}[k+d] = f\left(y[k+d-1],...,y[k+d-n],u[k],u[k-1],...,u[k+d-n]\right), \qquad (3.1)$$

onde $\hat{y}$ é a saída estimada da planta, k é o instante atual, $d = n_y - n_u$ é o atraso da planta, n_y é a ordem da saída da planta, n_u é a ordem da entrada da planta, $n = n_y$ é a ordem da planta, f é uma função, que pode ser não-linear, mapeada pela rede neural, y é a saída da planta e u é a entrada da planta. A Figura 3.5 ilustra a estrutura de modelo NNARX.

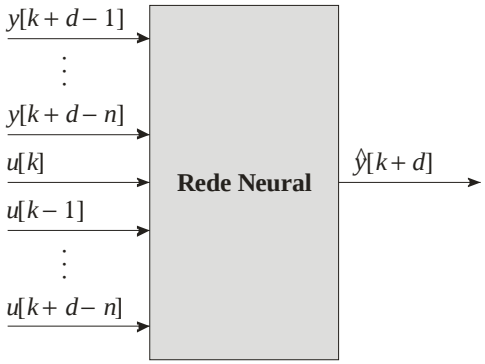

Figura 3.5 – Estrutura de modelo NNARX.

A RNA utilizada para mapear a função f foi um MLP que possui três camadas: a camada de entrada com nós de fonte; a camada oculta, com neurônios com função de ativação tangente hiperbólica; e a camada de saída com um neurônio com função de ativação linear.

Aqui, vale lembrar que cada neurônio da rede neural MLP tem um *bias*, que é um peso correspondente a uma entrada fixa, de valor igual a um, implícita na representação da rede neural. Essa entrada está sendo considerada um parâmetro interno da rede, por isso não aparece como uma entrada externa na representação da estrutura de modelo NNARX.

3.2. O Controlador

A idéia dos trabalhos [13] e [15] para se controlar uma planta, está em se obter uma variação do sinal de controle que resulte em uma variação na saída da planta, fazendo com que sua saída convirja para a referência. O jacobiano $J[k+d]$ da planta discretizada é a derivada, ou seja, taxa de variação, da saída $y[k+d]$ com relação à entrada $u[k]$. Pode-se representar o jacobiano no instante $k+d-1$ como segue [15]:

$$J[k+d-1]=\frac{\partial y[k+d-1]}{\partial u[k-1]} \cong \frac{\Delta y[k+d-1]}{\Delta u[k-1]}$$

utilizando o método de Euler [26] para aproximar a derivada, temos

$$J[k+d-1]=\frac{y[k+d]-y[k+d-1]}{u[k]-u[k-1]} \tag{3.2}$$

Isolando-se $u[k]$ e substituindo $y[k+d]$ pelo seu valor desejado (de referência) $r[k+d]$, temos a expressão da lei para o cálculo do sinal de controle u no instante k:

$$u[k]=u[k-1]+\frac{r[k+d]-y[k+d-1]}{J[k+d-1]} \tag{3.3}$$

A lei de controle dada pela expressão (3.3) também pode ser deduzida a partir da série de Taylor [26]. Sendo $f(x)$ uma função com $m+1$ derivadas contínuas, a série de Taylor para um ponto x próximo de x^* é representada pela seguinte expressão:

$$\begin{aligned} f(x) = {} & f(x^*)+f'(x^*)(x-x^*)+f''(x^*)\frac{(x-x^*)^2}{2!}+f'''(x^*)\frac{(x-x^*)^3}{3!}+ \\ & \cdots+f^{(m)}(x^*)\frac{(x-x^*)^m}{m!}+R_{m+1}(x) \end{aligned} \tag{3.4}$$

onde $R_{m+1}(x)$ tem qualquer uma das seguintes formas (ε é um ponto entre x e x^*):

$$R_{m+1}(x) = f^{(m+1)}(\varepsilon)\frac{\left(x-x^*\right)^{m+1}}{(m+1)!}$$

$$R_{m+1}(x) = \frac{1}{m!}\int_{x^*}^{x}(x-t)^m\ f^{(n+1)}(t)\,dt$$

Se for considerado que $y[k+d]$ só depende de $u[k]$, como também foi feito anteriormente para se obter a lei de controle, pode-se fazer [13]:

$$x \triangleq u[k] \Leftrightarrow f(x) = y[k+d]$$
$$x^* \triangleq u[k-1] \Leftrightarrow f\left(x^*\right) = y[k+d-1]$$

e assim, substituindo esses valores na série de Taylor truncada no termo de primeira ordem, temos

$$y[k+d] = y[k+d-1] + J[k+d-1]\left(u[k]-u[k-1]\right) \tag{3.5}$$

Isolando-se $u[k]$ e substituindo $y[k+d]$ pelo seu valor desejado $r[k+d]$, chega-se à expressão da lei de controle (3.3).

Levando em consideração que a planta a ser controlada é desconhecida ou varia no tempo, pelo princípio da equivalência à certeza, pode-se utilizar no lugar dos parâmetros da planta, os parâmetros do modelo identificado pela rede neural. Assim, a lei de controle fica de acordo com a expressão a seguir:

$$u[k] = u[k-1] + \frac{r[k+d]-\hat{y}[k+d-1]}{\hat{J}[k+d-1]} \tag{3.6}$$

onde $\hat{y}$ é a saída da planta estimada pela rede neural e $\hat{J}$ é o jacobiano estimado da planta, que, para uma estrutura de modelo NNARX com um MLP de três camadas de nós (duas de neurônios), é dado por

$$\hat{J}[k+d-1] = \frac{\partial \hat{y}[k+d-1]}{\partial u[k-1]} = \frac{\partial y_1^{(3)}}{\partial y_{n+1}^{(1)}}$$

$$= \frac{\partial y_1^{(3)}}{\partial \upsilon_1^{(3)}} \frac{\partial \upsilon_1^{(3)}}{\partial y_{n+1}^{(1)}}$$

$$= \frac{\partial y_1^{(3)}}{\partial \upsilon_1^{(3)}} \sum_{j=0}^{P} \left[\frac{\partial \varphi_j^{(2)}}{\partial \upsilon_j^{(2)}} \frac{\partial \upsilon_j^{(2)}}{\partial y_{n+1}^{(1)}} w_{1,j}^{(3)} \right]$$

$$\hat{J}[k+d-1] = \frac{\partial y_1^{(3)}}{\partial \upsilon_1^{(3)}} \sum_{j=0}^{P} \left[\frac{\partial \varphi_j^{(2)}}{\partial \upsilon_j^{(2)}} w_{j,n+1}^{(2)} w_{1,j}^{(3)} \right] \tag{3.7}$$

onde p é o número de neurônios da camada oculta, $\frac{\partial y_1^{(3)}}{\partial \upsilon_1^{(3)}}$ é a derivada da função de ativação do neurônio da camada de saída (terceira camada de nós) com relação ao seu campo local induzido, $\frac{\partial \varphi_j^{(2)}}{\partial \upsilon_j^{(2)}}$ são as derivadas das funções de ativação dos neurônios da camada oculta (segunda camada de nós) com relação aos seus respectivos campos locais induzidos.

3.3. Algoritmo do Esquema de Controle Híbrido Indireto

O algoritmo do esquema de controle, apresentado a seguir, é uma descrição resumida mostrando apenas os procedimentos mais relevantes para a implementação. Detalhes do treinamento da rede neural e da identificação foram omitidos no algoritmo por questão de simplificação. É importante lembrar que na identificação, existe um número de exemplos máximo que deve ser definido, e, quando esse limite é atingido, elimina-se o exemplo mais antigo para manter o número de exemplos dentro do limite. Se a inicialização dos pesos da rede neural NNARX for aleatória, o sinal de controle gerado inicialmente também será aleatório e servirá para excitar a planta, gerando uma saída, e assim, obtém-se o primeiro par de exemplos (entrada/saída) para o treinamento.

Algoritmo do esquema de controle

```
Programa Controle_Neural
Início
    NNARX.Inicializar;

    Repetir
    Início
        r[k + d] = Ler_Referência;
        y[k] = Ler_Saída_da_Planta;

        y_Estimado[k + d – 1] = NNARX.Estimar_Saída( k + d – 1 );
        J_Estimado[k + d – 1] = Calcular_Jacobiano_Estimado( k + d – 1 );
        u[k] = Calcular_Sinal_de_Controle;
        Aplicar_Sinal_de_Controle( u[k] );

        NNARX.Identificar_Planta;
        Próximo_Instante;
    Fim
Fim
```

No algoritmo, foi seguido um determinado padrão de representação: as palavras reservadas do algoritmo estão representadas em negrito; em itálico, são as variáveis; e as funções estão em formato normal.

Capítulo 4

Análise de Estabilidade para o Esquema de Controle

Foi mostrado [17], no Congresso Brasileiro de Redes Neurais, em 2005, que o esquema de controle em questão não atendeu às necessidades para a simulação do controle de nível em um sistema de tanques acoplados da *Quanser*. A saída do sistema ficava sempre oscilando perto da referência, mas não convergia. Por esse motivo, o esquema foi investigado na planta real e o mesmo problema foi verificado. Testes foram feitos em outras plantas reais da *Quanser* como *Ball and Beam* e *Servomotor*, e o esquema de controle também não teve desempenho satisfatório. Isso motivou a realização da investigação do problema através de análises matemáticas, mostradas neste capítulo, e através de análises gráficas, mostradas no Capítulo 5.

A Figura 4.1 mostra o esquema de controle com mais detalhes do que a Figura 3.4 para uma melhor avaliação.

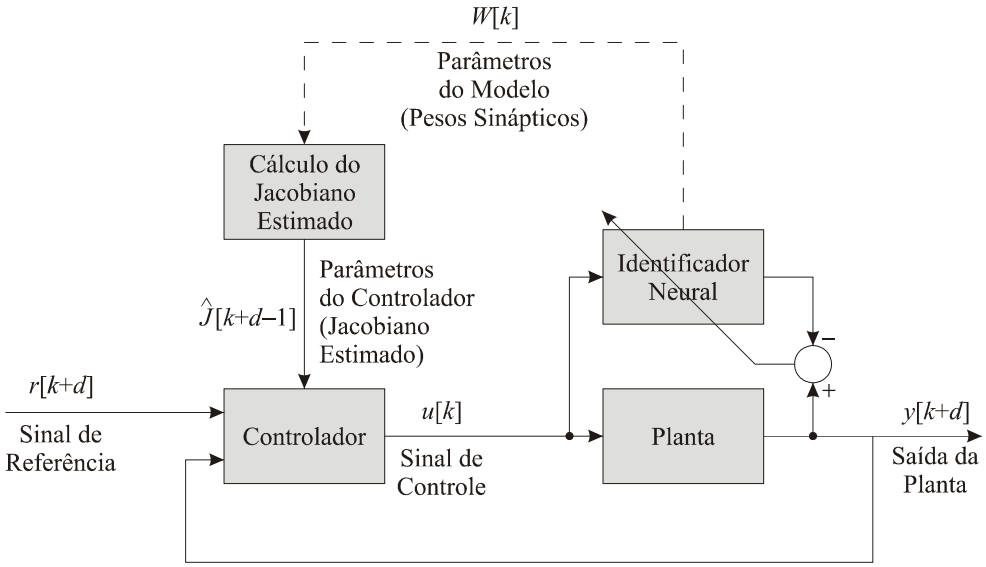

Figura 4.1 – Controle híbrido indireto detalhado.

Para se analisar a lei de controle sem a influência do identificador neural, é suposto que o modelo da planta é sempre identificado sem erro. Para isso, basta utilizar os parâmetros da própria planta no lugar dos parâmetros do identificador neural que são usados pela lei de controle. Assim, a lei de controle utilizada seria a da expressão (3.3) e não da expressão (3.4) seguindo o esquema da Figura 4.2.

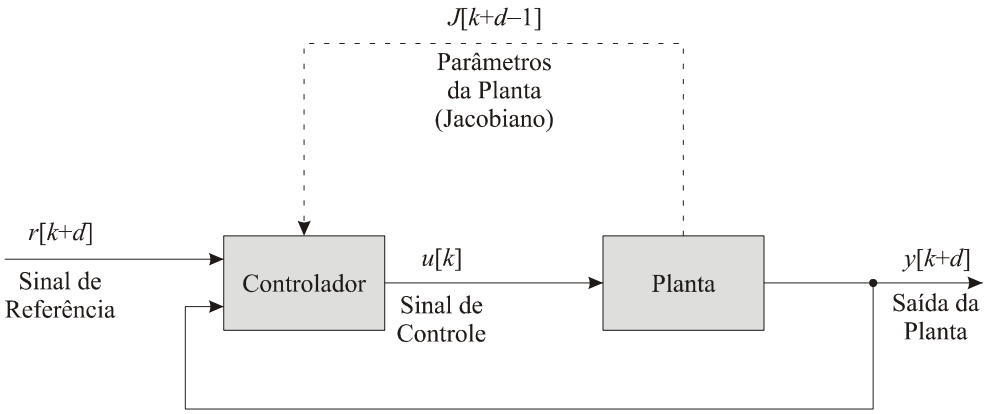

Figura 4.2 – Esquema de controle sem o identificador neural.

4.1. Análise Matemática para Plantas de Primeira Ordem

O modelo genérico, das plantas lineares discretas analisadas aqui, e que foram simuladas no Capítulo 5, é descrito pela expressão (4.1), considerando o atraso $d = 1$.

$$y[k+1] = a\,y[k] + b\,u[k] \tag{4.1}$$

Como a lei de controle foi deduzida partindo do princípio que $y[k+d]$ só depende de $u[k]$, ela seria perfeita para a planta com $a = 0$, fazendo sua saída ir para a referência no primeiro instante em que o sinal de controle atuasse e mantendo-a na referência nos instantes seguintes. Porém, quando $a \neq 0$, em determinados casos a lei falha porque não leva em consideração todos os parâmetros da dinâmica da planta.

Podemos mostrar que a estabilidade, neste caso, não depende de b, depende somente de a, fazendo uma análise de estabilidade no domínio de z. A expressão (4.3) mostra a função de transferência da planta no domínio de z.

$$\begin{aligned} y[k+1] &= a\,y[k] + b\,u[k] \\ Y(z)\,z &= aY(z) + bU(z) \end{aligned}$$

$$P(z) \triangleq \frac{Y(z)}{U(z)} = \frac{b}{z-a} \tag{4.3}$$

Para facilitar o entendimento, pode-se separar o cálculo do sinal de erro da lei de controle e calcular antes. A Figura 4.3 mostra como fica o esquema com essa nova interpretação. A Figura 4.4 mostra o diagrama de blocos do esquema de controle no domínio de z.

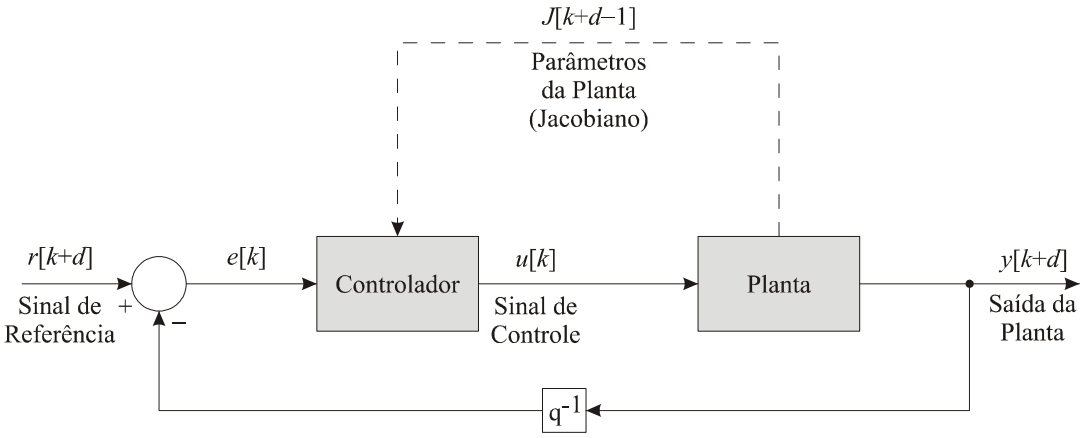

Figura 4.3 – Esquema de controle sem o identificador neural e com o cálculo do erro fora do bloco do controlador.

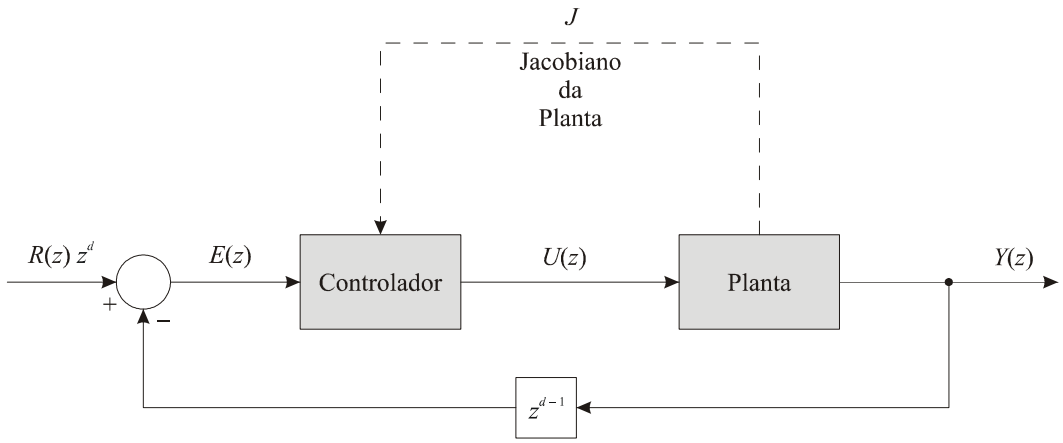

Figura 4.4 – Diagrama de blocos do esquema de controle no domínio de z .

A expressão (4.4) mostra a função de transferência do controlador no domínio de z, considerando o erro $e[k]=r[k+d]-y[k+d-1]$ e o jacobiano $J[k+d-1]=b$.

$$u[k]=u[k-1]+\frac{e[k]}{b}$$

$$U(z)=U(z)z^{-1}+\frac{E(z)}{b}$$

$$C(z)\triangleq\frac{U(z)}{E(z)}=\frac{z}{b(z-1)} \tag{4.4}$$

Juntando o controlador com a planta, a função de transferência fica de acordo com a expressão (4.5).

$$G(z)\triangleq\frac{Y(z)}{E(z)}=C(z)P(z)=\frac{z}{(z-a)(z-1)} \tag{4.5}$$

Considerando o diagrama de blocos do esquema de controle mostrado na Figura 4.4, é possível calcular uma função de transferência geral do esquema de controle em malha fechada [27] como mostra a expressão (4.6).

$$S(z) \triangleq \frac{Y(z)}{R(z)z} = \frac{G(z)}{1+G(z)} = \frac{z}{z^2 - a\,z + a} \tag{4.6}$$

Para um sistema discreto ser estável, basta que as raízes do polinômio característico discreto (denominador da função de transferência no domínio de z), quando igualado a zero, fiquem dentro do círculo de raio unitário com centro na origem do plano cartesiano complexo z, ou seja, basta que os pólos tenham módulo menor que um [29].

Para a equação $z^2 - a\,z + a = 0$ do nosso exemplo, onde $z = \frac{a \pm \sqrt{a^2 - 4\,a}}{2}$ (fórmula de Báscara), a condição $|z| < 1$ deve ser atendida para que o sistema seja estável, como mostra a Figura 5.3, Figura 5.6 e Figura 5.9. Com isso, nota-se que a estabilidade depende apenas do valor de a para a planta de modelo genérico que obedece à expressão (4.1). Caso $|z| = 1$, o sistema é oscilatório, como pode ser observado na Figura 5.2, Figura 5.5 e Figura 5.8. E, se $|z| > 1$, o sistema é instável, como mostra a Figura 5.1, Figura 5.4 e Figura 5.7.

Como não fica fácil de se definir as condições de estabilidade com relação aos parâmetros de um sistema de ordem maior que um a partir dos pólos, pode-se utilizar o critério de estabilidade de Jury [28] para se fazer isso.

Critério de estabilidade de Jury

Considerando o polinômio característico de uma planta linear discreta representado da seguinte forma:

$$Q(z) = a_n z^n + a_{n-1} z^{n-1} + \cdots + a_1 z + a_0, \quad a_n > 0\,, \tag{4.7}$$

podemos utilizar a seguinte tabela para se fazer o teste de Jury:

Tabela 4.1 – Tabela para o teste de estabilidade de Jury

z^0	z^1	z^2	$\cdots$	z^{n-k}	$\cdots$	z^{n-1}	z^n
a_0	a_1	a_2	$\cdots$	a_{n-k}	$\cdots$	a_{n-1}	a_n
a_n	a_{n-1}	a_{n-2}	$\cdots$	a_k	$\cdots$	a_1	a_0
b_0	b_1	b_2	$\cdots$	b_{n-k}	$\cdots$	b_{n-1}	
b_{n-1}	b_{n-2}	b_{n-3}	$\cdots$	b_{k-1}	$\cdots$	b_0	
c_0	c_1	c_2	$\cdots$	c_{n-k}	$\cdots$		
c_{n-2}	c_{n-3}	c_{n-4}	$\cdots$	c_{k-2}	$\cdots$		
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$			
l_0	l_1	l_2	l_3				
l_3	l_2	l_1	l_0				
m_0	m_1	m_2					

onde os elementos de cada linha par da tabela são os elementos da linha precedente em ordem reversa; e os elementos das linhas ímpares são definidos como

$$b_k = \begin{vmatrix} a_0 & a_{n-k} \\ a_n & a_k \end{vmatrix}, \qquad c_k = \begin{vmatrix} b_0 & b_{n-1-k} \\ b_{n-1} & b_k \end{vmatrix}, \qquad d_k = \begin{vmatrix} c_0 & c_{n-2-k} \\ c_{n-2} & c_k \end{vmatrix}, \cdots \tag{4.8}$$

As condições necessárias e suficientes para o polinômio $Q(z)$ não ter raízes fora do círculo de raio unitário, com $a_n > 0$, são as seguintes:

$$\begin{aligned} &Q(1) > 0 \\ &(-1)^n Q(-1) > 0 \\ &|a_0| < a_n \\ &|b_0| > |b_{n-1}| \\ &|c_0| > |c_{n-2}| \\ &|d_0| > |d_{n-3}| \\ &\quad\vdots \\ &|m_0| > |m_2| \end{aligned} \tag{4.9}$$

No caso do polinômio característico $Q(z) = z^2 - a\,z + a$ do sistema de controle completo com uma planta de primeira ordem e atraso unitário, a tabela Jury é

$$\begin{array}{ccc} z^0 & z^1 & z^2 \\ \hline a & -a & 1 \end{array}$$

Para a condição $Q(1) > 0$, temos

$$1 - a + a = 1 > 0$$

Para a condição $(-1)^2 Q(-1) > 0$, temos

$$1 + a + a = 1 + 2a > 0 \Rightarrow a > -\frac{1}{2}$$

Para a condição $|a_0| < a_2$, temos

$$|a| < 1 \Rightarrow -1 < a < 1$$

Juntando todas as condições, temos as seguintes condições de estabilidade para o sistema:

$$-\frac{1}{2} < a < 1 \tag{4.10}$$

4.2. Análise Matemática para Plantas de Ordem Maior que Um

Para plantas de maior ordem, funciona a mesma lógica: o único componente da dinâmica da planta que é levado em consideração no cálculo do sinal de controle é o jacobiano, portanto, plantas que poderiam ser controladas, caso esses parâmetros tivessem sido levados em consideração na elaboração da lei de controle, podem não ser controladas. Também é possível provar isso analisando o polinômio característico do sistema de controle com essa planta. Considerando uma planta discreta de ordem n e atraso d com a função de transferência da seguinte forma:

$$P(z) = \frac{Y(z)}{U(z)} = \frac{\beta_{n-d} z^{n-d} + \beta_{n-d-1} z^{n-d-1} + \cdots + \beta_1 z + \beta_0}{z^n + \alpha_{n-1} z^{n-1} + \cdots + \alpha_1 z + \alpha_0}, \tag{4.11}$$

com o polinômio do numerador

$$P_{NUM}(z) \triangleq \beta_{n-d} z^{n-d} + \beta_{n-d-1} z^{n-d-1} + \cdots + \beta_1 z + \beta_0 \tag{4.12}$$

e do denominador

$$P_{DEN}(z) \triangleq z^n + \alpha_{n-1} z^{n-1} + \cdots + \alpha_1 z + \alpha_0 , \tag{4.13}$$

onde β_{n-d} é o jacobiano da planta, temos

$$C(z) \triangleq \frac{U(z)}{E(z)} = \frac{z}{\beta_{n-d}(z-1)} \tag{4.14}$$

Para facilitar o entendimento com plantas de ordem maior que um, pode-se representar a função de transferência do controlador como segue:

$$C(z) = \frac{\left(\frac{z}{\beta_{n-d}}\right)}{(z-1)} \tag{4.15}$$

Assim, temos

$$G(z) = C(z)P(z) = \frac{\frac{P_{NUM}(z)}{\beta_{n-d}} z}{(z-1)P_{DEN}(z)}$$

$$S(z) \triangleq \frac{Y(z)}{R(z)z^d} = \frac{G(z)}{1+G(z)z^{d-1}} = \frac{\frac{\frac{P_{NUM}(z)}{\beta_{n-d}} z}{(z-1)P_{DEN}(z)}}{1+\frac{\frac{P_{NUM}(z)}{\beta_{n-d}} z}{(z-1)P_{DEN}(z)} z^{d-1}} = \frac{\frac{\frac{P_{NUM}(z)}{\beta_{n-d}} z}{(z-1)P_{DEN}(z)}}{\frac{(z-1)P_{DEN}(z)+\frac{P_{NUM}(z)}{\beta_{n-d}} z^d}{(z-1)P_{DEN}(z)}}$$

$$S(z) = \frac{\frac{P_{NUM}(z)}{\beta_{n-d}}}{(z-1)P_{DEN}(z)+\frac{P_{NUM}(z)}{\beta_{n-d}} z^d} \tag{4.16}$$

Considerando o polinômio característico do sistema representado pela função de transferência da expressão (4.16), temos

$$Q(z) = (z-1)P_{DEN}(z) + \frac{P_{NUM}(z)}{\beta_{n-d}} z^d$$

$$= (z-1)\left(z^n + \alpha_{n-1}z^{n-1} + \cdots + \alpha_1 z + \alpha_0\right) + \frac{\beta_{n-d}z^{n-d} + \beta_{n-d-1}z^{n-d-1} + \cdots + \beta_1 z + \beta_0}{\beta_{n-d}} z^d$$

$$\begin{aligned} Q(z) = {} & \left(z^{n+1} + (\alpha_{n-1} - 1)z^n + (\alpha_{n-2} - \alpha_{n-1})z^{n-1} + \cdots + (\alpha_0 - \alpha_1)z - \alpha_0\right) + \\ & \left(z^n + \frac{\beta_{n-d-1}}{\beta_{n-d}} z^{n-1} + \cdots + \frac{\beta_1}{\beta_{n-d}} z^{d+1} + \frac{\beta_0}{\beta_{n-d}} z^d\right) \end{aligned} \tag{4.17}$$

Pela expressão (4.17), percebe-se que os coeficientes do polinômio característico do sistema em malha fechada são função de uma relação dos coeficientes do sinal de entrada da planta (sinal de controle). Assim, a estabilidade do sistema de controle depende dos coeficientes do sinal de saída da planta e de uma razão dos coeficientes do sinal de entrada e o jacobiano, ou seja, depende dos seguintes termos:

$$\alpha_{n-1}, \cdots, \alpha_1, \alpha_0, \frac{\beta_{n-d-1}}{\beta_{n-d}}, \cdots, \frac{\beta_1}{\beta_{n-d}}, \frac{\beta_0}{\beta_{n-d}} \tag{4.18}$$

Com isso, pode-se concluir que devido à lei de controle não considerar os outros coeficientes do modelo da planta além do jacobiano β_{n-d}, não há como saber o quanto será a influência das parcelas que contém esses coeficientes para se gerar um sinal de controle que corrija o erro e, ao mesmo tempo, anule a influência dessas parcelas. Assim, quando a influência das parcelas que não contém o jacobiano tiver valor absoluto maior que o valor absoluto da influência da parcela que o contém, a lei tende a falhar. Isso já era esperado, pois, como foi dito no Capítulo 3, a lei de controle foi criada considerando que a saída da planta $y[k+d]$ só depende da entrada $u[k]$, e assim, desconsiderando os outros termos, saídas e entradas nos instantes anteriores.

Capítulo 5

Resultados

Para comprovar a análise realizada, foram feitos testes com plantas lineares baseadas na expressão (4.1). O identificador neural foi validado fazendo-se a previsão da saída da planta e comparando com a saída real obtida. Ele funcionou como esperado. Sendo assim, não havia erro de programação com ele, então, o problema em questão só poderia estar surgindo devido à lei de controle.

Toda a implementação de programa computacional foi feita através do *C++ Builder 5 Entertprise*, da *Borland*. Os primeiros testes foram feitos para verificar o funcionamento da lei de controle. Para tanto, o identificador neural foi eliminado para não influenciar nos resultados e no seu lugar de se utilizar seus parâmetros na lei de controle, foram utilizados os parâmetros da planta simulada. A seção 5.1 mostra os resultados desses testes. A seção 5.2 mostra os resultados das simulações com o identificador neural. Já a seção 5.3, mostra os resultados do esquema de controle sendo usado para plantas implementadas em um computador analógico.

O sinal da referência utilizado foi o mesmo utilizado por Adetona et al. [13] e Maitelli e Gabriel Filho [15] com uma pequena diferença: para $k = 0$, $r[k] = 0$, conforme mostrado na expressão (5.1).

$$r[k] = \begin{cases} 1 + \frac{1}{2}\left[\text{sen}\left(\frac{\pi k}{50}\right) + \text{sen}\left(\frac{\pi k}{100}\right) + \text{sen}\left(\frac{\pi k}{150}\right)\right], \text{ para } k = 1,2,3,\cdots \\ 0, \text{ para } k = 0,-1,-2,\cdots \end{cases} \quad (5.1)$$

5.1. Resultados do Controle Sem o Identificador Neural

Nesta seção, serão mostrados os resultados dos testes feitos com o objetivo de verificar a funcionalidade da lei de controle. A melhor forma para isso é utilizar os parâmetros da planta simulada na lei de controle ao invés de utilizar os parâmetros do modelo identificado pela rede neural. Isso faz com que os resultados obtidos reflitam a efetividade da lei de controle independente do identificador neural.

Os primeiros resultados com a utilização dos parâmetros do modelo da planta simulada na lei de controle foram obtidos para plantas com $b = 1$ e a com valores diferentes. A Figura 5.1 mostra o resultado para $a = 1{,}1$, a Figura 5.2 mostra o resultado para $a = 1$ e a Figura 5.3 mostra o resultado para $a = 0{,}9$.

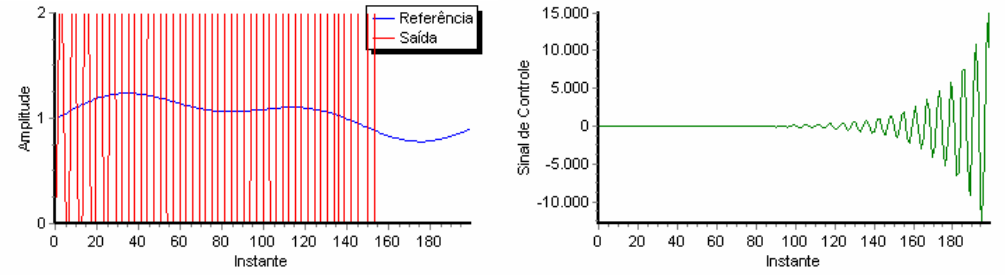

Figura 5.1 – Teste para $a = 1{,}1$ e $b = 1$.

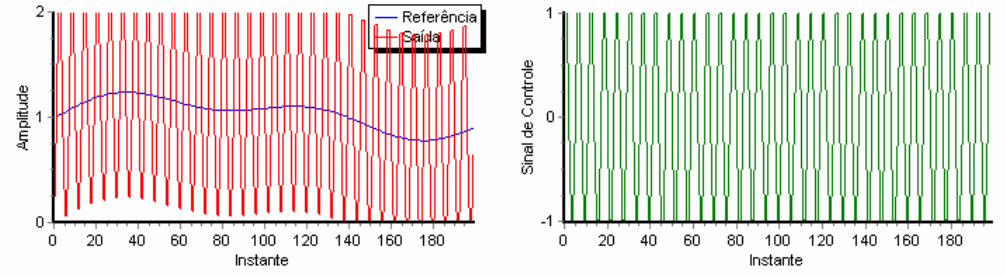

Figura 5.2 – Teste para $a = 1$ e $b = 1$.

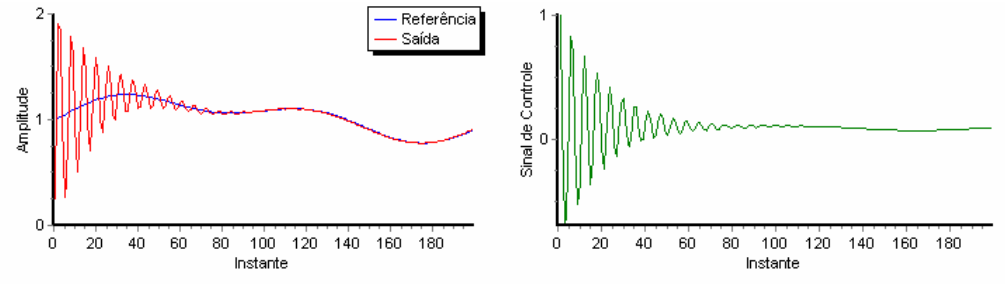

Figura 5.3 – Teste para $a = 0{,}9$ e $b = 1$.

Nos testes com $b = 1{,}5$, a Figura 5.4 mostra o resultado para $a = 1{,}1$, a Figura 5.5 mostra o resultado para $a = 1$ e a Figura 5.6 mostra o resultado para $a = 0{,}9$.

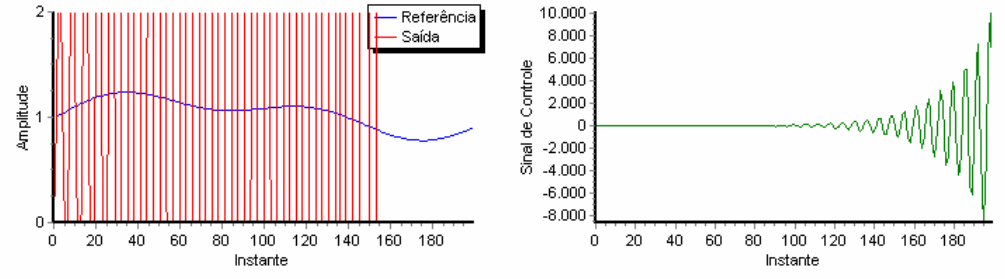

Figura 5.4 – Teste para $a = 1{,}1$ e $b = 1{,}5$.

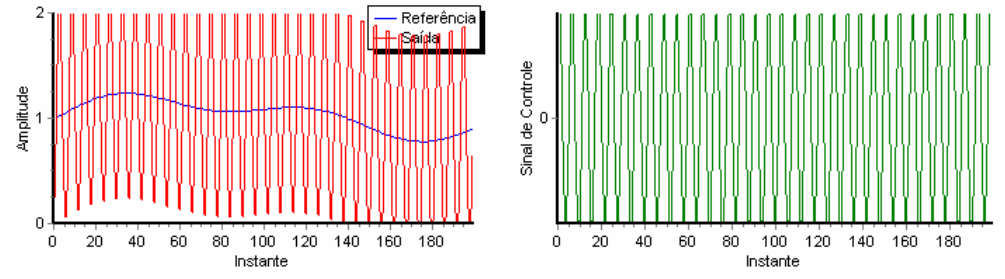

Figura 5.5 – Teste para $a = 1$ e $b = 1{,}5$.

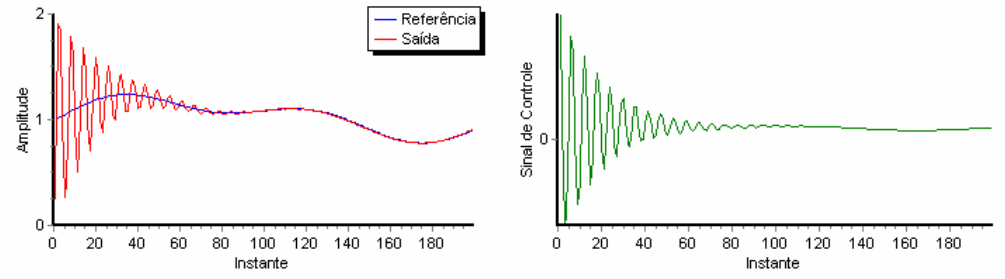

Figura 5.6 – Teste para $a = 0{,}9$ e $b = 1{,}5$.

Nos testes com $b = 0{,}5$, a Figura 5.7 mostra o resultado para $a = 1{,}1$, a Figura 5.8 mostra o resultado para $a = 1$ e a Figura 5.9 mostra o resultado para $a = 0{,}9$.

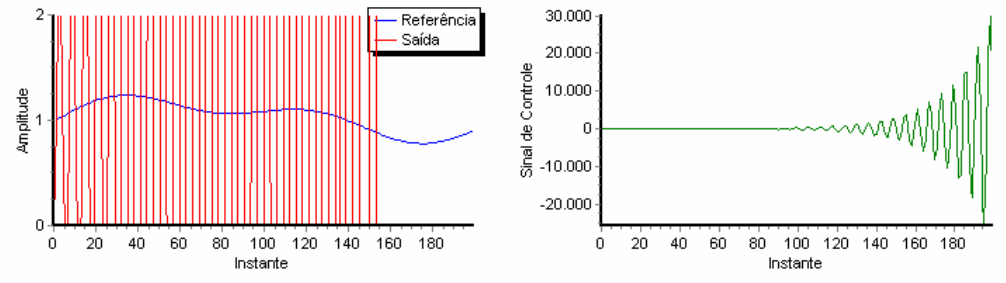

Figura 5.7 – Teste para $a = 1{,}1$ e $b = 0{,}5$.

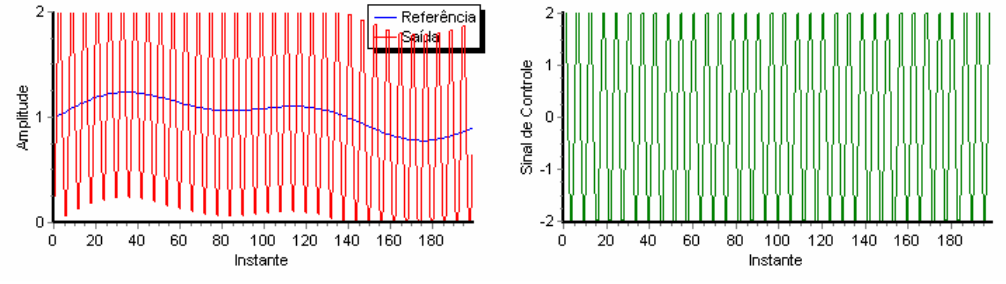

Figura 5.8 – Teste para $a = 1$ e $b = 0{,}5$.

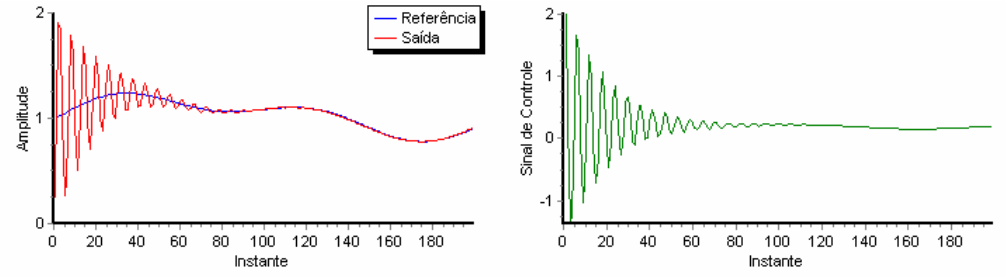

Figura 5.9 – Teste para $a = 0{,}9$ e $b = 0{,}5$.

Com as observações dos resultados anteriores, percebe-se que o esquema de controle funciona para certos casos e falha para outros. Uma observação pode ser feita a respeito dos parâmetros da planta em questão: como demonstrado na seção 4.1, levando em consideração valores maiores que $-\frac{1}{2}$, quando $a < 1$, o controlador cumpre sua função fazendo a saída da planta convergir para a referência; caso contrário, ele falha. Isso independe do valor de b, pois ele, neste caso, corresponde ao jacobiano da planta e é eliminado pelo sistema de

controle. No caso dessas plantas de primeira ordem com atraso unitário, somente o valor de a vai definir se o sistema por completo será ou não estável.

5.2. Resultados para Plantas Simuladas

Depois dos testes utilizando os próprios parâmetros da planta na lei de controle, foram feitos testes com os parâmetros do identificador neural. Assim, a lei de controle utilizada nos testes seguintes é descrita pela expressão (3.4).

Levando em consideração que o identificador neural é treinado para poucos instantes, não é necessário utilizar uma estrutura de modelo não-linear para se identificar uma planta não-linear. Pois, supõe-se que o sistema, para aquele pequeno intervalo de tempo em que o modelo será utilizado, é suficientemente linear. Sendo assim, foi utilizada uma rede neural linear no identificador neural para se fazer os testes seguintes. Como uma rede neural linear não necessita de mais de uma camada de neurônios [16], foi utilizada somente uma. E, como o sistema a ser identificado é de única entrada e única saída (SISO), é necessário apenas um neurônio nessa camada de saída.

O motivo de se utilizar uma rede neural linear, é que houve problema em alguns testes feitos com uma rede neural não-linear. O problema é que, algumas vezes, os neurônios da camada oculta ficavam saturados, valendo $-\rho$ ou ρ para a função de ativação tangente hiperbólica, ou valendo 0 ou 1 para função de ativação logística. Quando se calculava o jacobiano estimado, ele valia zero devido às derivadas das funções de ativação valerem zero quando saturadas. O jacobiano estimado não pode ser zero, pois no cálculo do sinal de controle ele é usado em um denominador de uma fração. Se o jacobiano for zero, significa que a saída da planta não varia com uma variação da entrada, portanto, a planta não pode ser controlada.

Outros dois motivos para se utilizar a rede neural linear é por seu tempo de treinamento ser menor e não há necessidade de se calcular o jacobiano estimado, pois seu valor já está explícito no peso ligado à entrada $u[k]$. Além disso, as plantas usadas nos testes seguintes foram lineares, não necessitando de um identificador não-linear.

O algoritmo de treinamento utilizado para o identificador neural foi o *back-propagation* seqüencial com parâmetro de momento $\alpha = 0{,}95$ e com o parâmetro da taxa de aprendizagem adaptativo iniciando em $\eta = 0{,}01$. O número máximo de exemplos foi três. Assim, a cada novo exemplo que surge, o mais antigo é eliminado do conjunto de exemplos de treinamento para manter no máximo três. O modelo geral das plantas usadas nos testes segue a estrutura da expressão (4.1).

Nos testes para plantas com $b = 1$, a Figura 5.10 mostra o resultado para $a = 1{,}1$, a Figura 5.11 mostra o resultado para $a = 1$ e a Figura 5.12 mostra o resultado para $a = 0{,}9$.

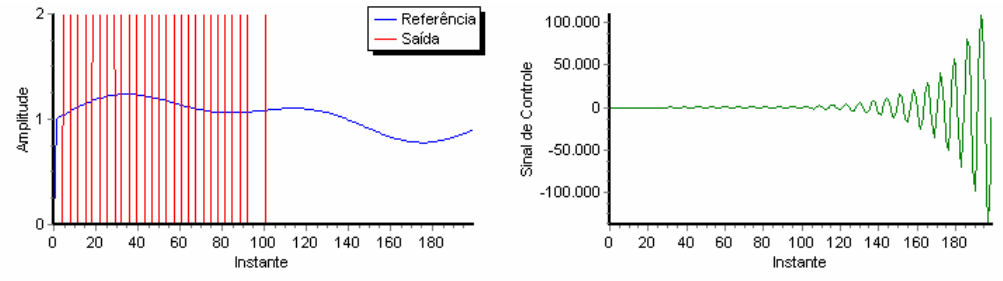

Figura 5.10 – Teste para $a = 1{,}1$ e $b = 1$.

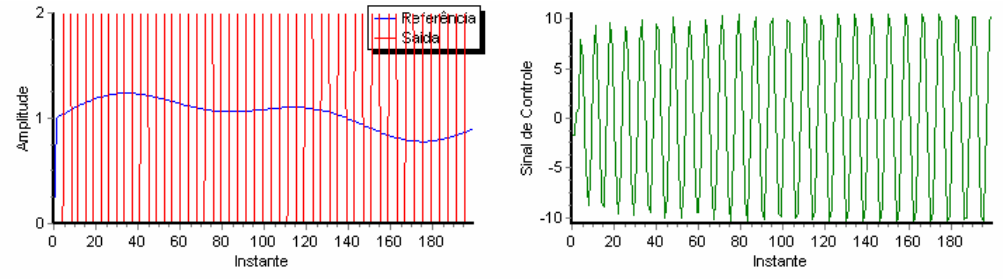

Figura 5.11 – Teste para $a = 1$ e $b = 1$.

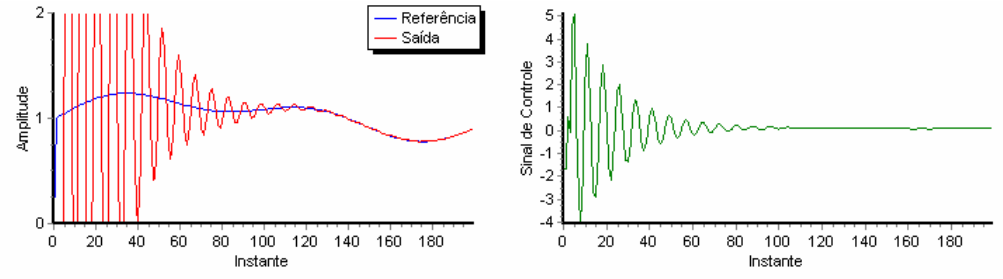

Figura 5.12 – Teste para $a = 0{,}9$ e $b = 1$.

Nos testes com $b = 1{,}5$, a Figura 5.13 mostra o resultado para $a = 1{,}1$, a Figura 5.14 mostra o resultado para $a = 1$ e a Figura 5.15 mostra o resultado para $a = 0{,}9$.

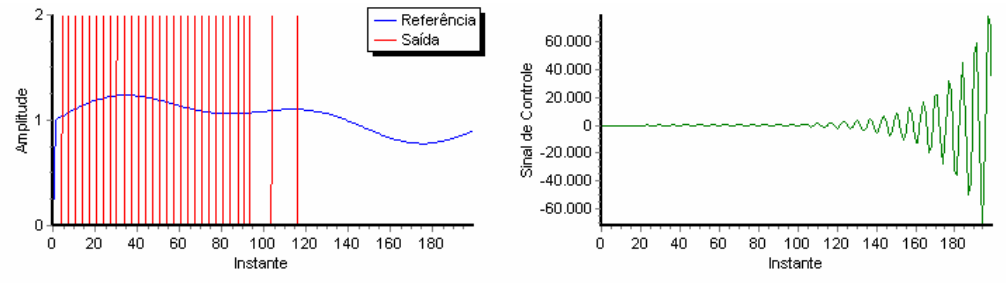

Figura 5.13 – Teste para $a = 1{,}1$ e $b = 1{,}5$.

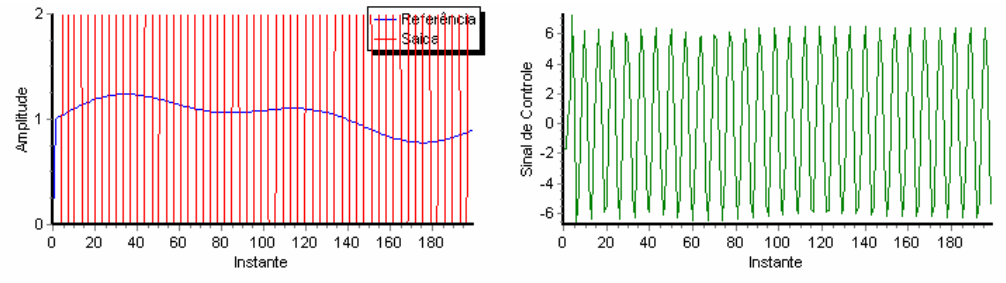

Figura 5.14 – Teste para $a = 1$ e $b = 1{,}5$.

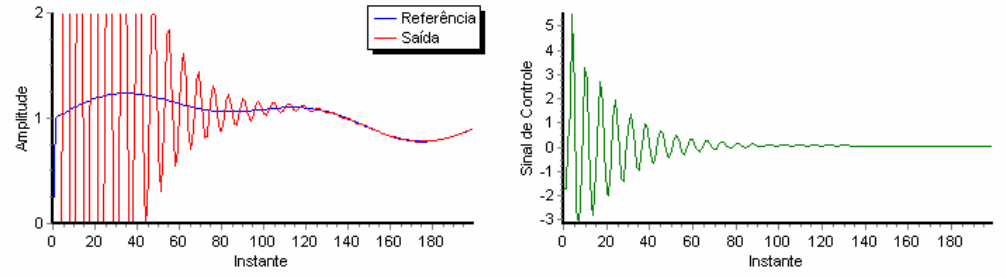

Figura 5.15 – Teste para $a = 0{,}9$ e $b = 1{,}5$.

Nos testes com $b = 0{,}5$, a Figura 5.16 mostra o resultado para $a = 1{,}1$, a Figura 5.17 mostra o resultado para $a = 1$ e a Figura 5.18 mostra o resultado para $a = 0{,}9$.

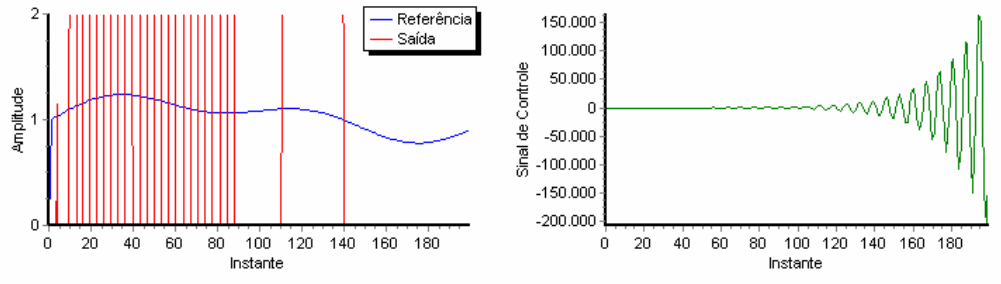

Figura 5.16 – Teste para $a = 1{,}1$ **e** $b = 0{,}5$.

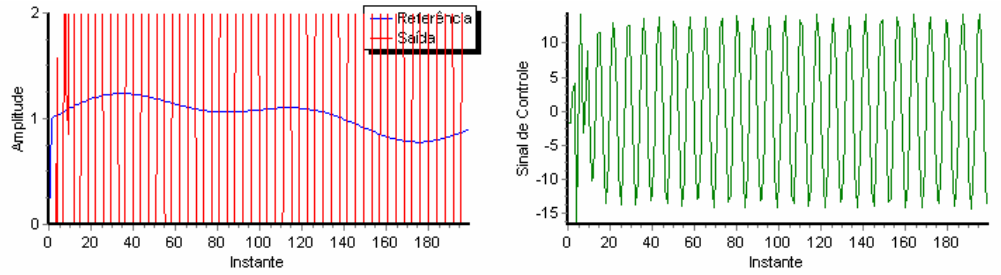

Figura 5.17 – Teste para $a = 1$ **e** $b = 0{,}5$.

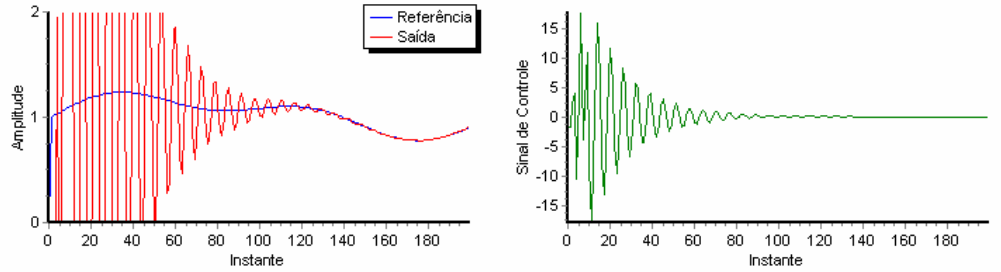

Figura 5.18 – Teste para $a = 0{,}9$ **e** $b = 0{,}5$.

Os resultados utilizando o identificador neural foram semelhantes aos resultados utilizando os próprios parâmetros da planta. Porém, observou-se que o comportamento difere nos primeiros instantes, pois, neles, a rede neural ainda não tem identificado perfeitamente a planta.

5.3. Resultados para Plantas Implementadas em um Computador Analógico

O motivo de se implementar as plantas em um computador analógico, é que a planta se torna um sistema físico real, sujeito a ruídos e apresentando não-linearidades quando seus

componentes são saturados. Para implementar as plantas analogicamente, foi utilizado um computador analógico *GP-6* (Figura 5.19) da *Comdyna, Inc.* [20]. A programação do sistema de controle foi feita em um computador digital (Processador *AMD Duron*™ 1200MHz, 256MB de memória RAM sendo que 8MB são dedicados para vídeo) e a comunicação com o computador analógico foi feita através de uma placa de aquisição de dados e controle, *MULTIQ*™ *3* da *Quanser* [30], com oito conversores A/D (conversor de sinal analógico para digital) e oito D/A (conversor de sinal digital para analógico).

Figura 5.19 – Computador analógico GP-6 da Comdyna.

As plantas implementadas no computador analógico foram plantas contínuas no tempo correspondentes às plantas discretas no tempo usadas nos testes das seções 5.1 e 5.2. As expressões (5.2) expressam o modelo genérico no tempo contínuo t, para as plantas implementadas no computador analógico, e o modelo genérico no tempo discreto (no domínio do instante k).

$$\begin{cases} \dot{y}(t) = a_c\, y(t) + b_c\, u(t) \\ y[k+1] = a_d\, y[k] + b_d\, u[k] \end{cases} \tag{5.2}$$

O sistema de controle utiliza um conversor D/A para enviar os sinais de controle calculados no computador para a planta. Assim, temos um segurador de ordem zero (ZOH) entre o computador e a planta. A expressão (5.3) calcula, a partir da função de transferência da planta no domínio de s, a função de transferência da planta discretizada com o segurador de ordem zero no domínio de z.

$$P_{ZOH}(z) = \left(1 - z^{-1}\right) \mathcal{Z}\left\{ \mathcal{L}^{-1}\left\{ \frac{P(s)}{s} \right\} \right\} \tag{5.3}$$

A expressão (5.4) mostra a função de transferência da planta no domínio de s e a expressão (5.5) mostra a função de transferência da planta discretizada com o ZOH no domínio de z.

$$\dot{y}(t) = a_c\, y(t) + b_c\, u(t)$$

$$P(s) \triangleq \frac{Y(s)}{U(s)} = \frac{b_c}{s - a_c} \tag{5.4}$$

$$P_{ZOH}(z) = \frac{Y(z)}{U(z)} = \left(1 - z^{-1}\right) \mathcal{Z}\left\{ \mathcal{L}^{-1}\left\{ \frac{b_c}{s(s - a_c)} \right\} \right\}$$

$$\frac{b_c}{s(s - a_c)} = -b_c \frac{1}{s(a_c - s)} = \frac{-b_c}{a_c} \frac{1}{s\left(1 + \dfrac{1}{-a_c} s\right)}$$

$$\frac{Y(z)}{U(z)} = \left(1 - z^{-1}\right) \frac{-b_c}{a_c} \mathcal{Z}\left\{ \mathcal{L}^{-1}\left\{ \frac{1}{s\left(1 + \dfrac{1}{-a_c} s\right)} \right\} \right\}$$

$$\mathcal{Z}\left\{ \mathcal{L}^{-1}\left\{ \frac{1}{s\left(1 + \dfrac{1}{-a_c} s\right)} \right\} \right\} = \frac{z\left(1 - e^{T_s a_c}\right)}{(z - 1)\left(z - e^{T_s a_c}\right)}$$

$$\frac{Y(z)}{U(z)} = \left(1 - z^{-1}\right) \frac{-b_c}{a_c} \frac{z\left(1 - e^{T_s a_c}\right)}{(z - 1)\left(z - e^{T_s a_c}\right)} = \frac{-b_c (z - 1)\left(1 - e^{T_s a_c}\right)}{a_c (z - 1)\left(z - e^{T_s a_c}\right)} = \frac{-b_c\left(1 - e^{T_s a_c}\right)}{a_c\left(z - e^{T_s a_c}\right)}$$

$$\frac{Y(z)}{U(z)} = \frac{b_c\left(e^{T_s a_c} - 1\right)}{a_c\, z - a_c\, e^{T_s a_c}} \tag{5.5}$$

onde, T_s é o período de amostragem utilizado. A partir da expressão (5.5), obtém-se a expressão (5.6), que representa o modelo da planta discreta no domínio do instante k.

$$a_c\, z\, Y(z) - a_c\, e^{T_s a_c}\, Y(z) = b_c\left(e^{T_s a_c} - 1\right) U(z)$$

$$a_c\, y[k+1] - a_c\, e^{T_s a_c}\, y[k] = b_c\left(e^{T_s a_c} - 1\right) u[k]$$

$$y[k+1] = e^{T_s a_c}\, y[k] + \frac{b_c}{a_c}\left(e^{T_s a_c} - 1\right) u[k] \tag{5.6}$$

Comparando a expressão (5.6) com a expressão (5.2), podemos concluir que:

$$\begin{cases} a_d = e^{T_s a_c} \\ b_d = \dfrac{b_c}{a_c}\left(e^{T_s a_c} - 1\right) \end{cases} \tag{5.7}$$

A partir das expressões (5.7), podemos obter as expressões (5.8) para calcular os parâmetros do modelo da planta no tempo contínuo (a_c e b_c) a partir dos parâmetros do modelo da planta no tempo discreto (a_d e b_d) e do período de amostragem T_s.

$$\begin{cases} a_c = \dfrac{\ln(a_d)}{T_s} \\ b_c = \dfrac{b_d \ln(a_d)}{(a_d - 1)T_s}, \quad a_d \neq 1 \end{cases} \tag{5.8}$$

Para $a_d = 1 \Rightarrow b_c = \dfrac{0}{0}$, ou seja, surge uma indeterminação matemática. Para se resolver esse problema, pode-se calcular b_c aplicando limite quando a_d tende a um, e utilizando a regra de L'Hôpital, como mostra a expressão (5.9).

$$b_c = \lim_{a_d \to 1} \frac{b_d \ln(a_d)}{(a_d - 1)T_s} = \lim_{a_d \to 1} \frac{\dfrac{d\left[b_d \ln(a_d)\right]}{d\, a_d}}{\dfrac{d\left[(a_d - 1)T_s\right]}{d\, a_d}} = \lim_{a_d \to 1} \frac{\dfrac{b_d}{a_d}}{T_s} = \frac{b_d}{T_s} \tag{5.9}$$

Assim, temos as expressões (5.10) que mostram todas as possibilidades para se calcular os parâmetros da planta analógica (contínua no tempo).

$$\begin{cases} a_c = \dfrac{\ln(a_d)}{T_s} \\ b_c = \begin{cases} \dfrac{b_d \ln(a_d)}{(a_d - 1)T_s}, & a_d \neq 1 \\ \dfrac{b_d}{T_s}, & a_d = 1 \end{cases} \end{cases} \tag{5.10}$$

Para os testes mostrados nesta seção, foi utilizado um período de amostragem $T_s = 0{,}1\text{s}$. As plantas contínuas foram implementadas com os parâmetros calculados a partir dos parâmetros das plantas discretas utilizadas nos testes anteriores através das expressões

(5.10). Os parâmetros das plantas contínuas correspondentes aos das plantas discretas são mostrados a seguir:

$$\begin{cases} a_d = 1{,}1 \\ b_d = 1 \end{cases} \Leftrightarrow \begin{cases} a_c = 0{,}9531 \\ b_c = 9{,}531 \end{cases}$$

$$\begin{cases} a_d = 1 \\ b_d = 1 \end{cases} \Leftrightarrow \begin{cases} a_c = 0 \\ b_c = 10 \end{cases}$$

$$\begin{cases} a_d = 0{,}9 \\ b_d = 1 \end{cases} \Leftrightarrow \begin{cases} a_c = -1{,}054 \\ b_c = 10{,}54 \end{cases}$$

$$\begin{cases} a_d = 1{,}1 \\ b_d = 1{,}5 \end{cases} \Leftrightarrow \begin{cases} a_c = 0{,}9531 \\ b_c = 14.3 \end{cases}$$

$$\begin{cases} a_d = 1 \\ b_d = 1{,}5 \end{cases} \Leftrightarrow \begin{cases} a_c = 0 \\ b_c = 15 \end{cases}$$

$$\begin{cases} a_d = 0{,}9 \\ b_d = 1{,}5 \end{cases} \Leftrightarrow \begin{cases} a_c = -1{,}054 \\ b_c = 15{,}8 \end{cases}$$

$$\begin{cases} a_d = 1{,}1 \\ b_d = 0{,}5 \end{cases} \Leftrightarrow \begin{cases} a_c = 0{,}9531 \\ b_c = 4{,}766 \end{cases}$$

$$\begin{cases} a_d = 1 \\ b_d = 0{,}5 \end{cases} \Leftrightarrow \begin{cases} a_c = 0 \\ b_c = 5 \end{cases}$$

$$\begin{cases} a_d = 0{,}9 \\ b_d = 0{,}5 \end{cases} \Leftrightarrow \begin{cases} a_c = -1{,}054 \\ b_c = 5{,}268 \end{cases}$$

Nos testes para plantas com $b_c = 10$, a Figura 5.20 mostra o resultado para $a_c = 1$, a Figura 5.21 mostra o resultado para $a_c = 0$ e a Figura 5.22 mostra o resultado para $a_c = -1$.

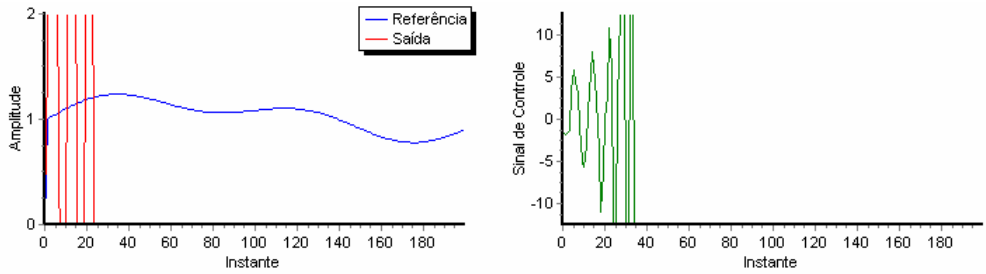

Figura 5.20 – Teste para $a_c = 0{,}9531$ e $b_c = 9{,}531$.

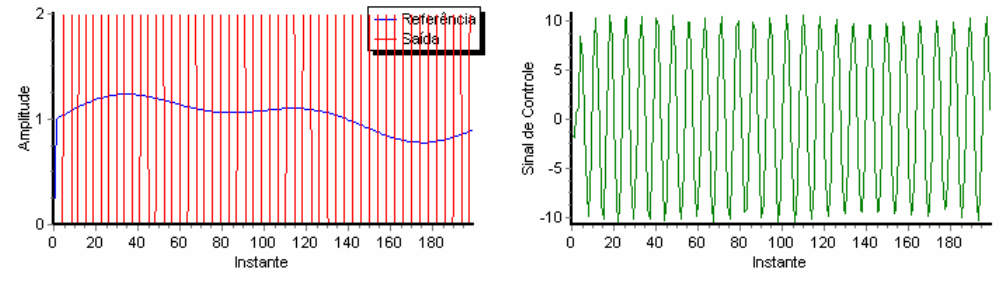

Figura 5.21 – Teste para $a_c = 0$ e $b_c = 10$.

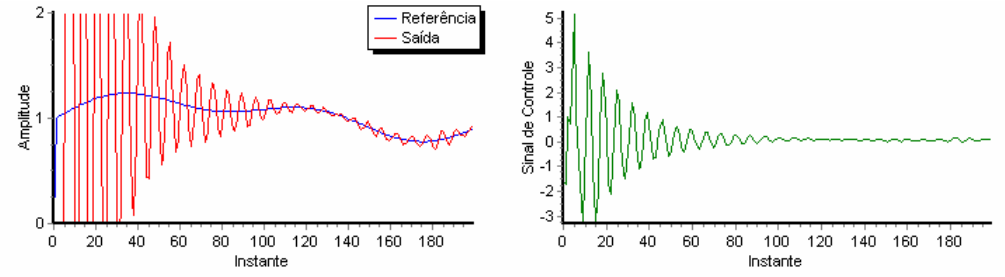

Figura 5.22 – Teste para $a_c = -1,054$ e $b_c = 10,54$.

Nos testes para plantas com $b_c = 15$, a Figura 5.23 mostra o resultado para $a_c = 1$, a Figura 5.24 mostra o resultado para $a_c = 0$ e a Figura 5.25 mostra o resultado para $a_c = -1$.

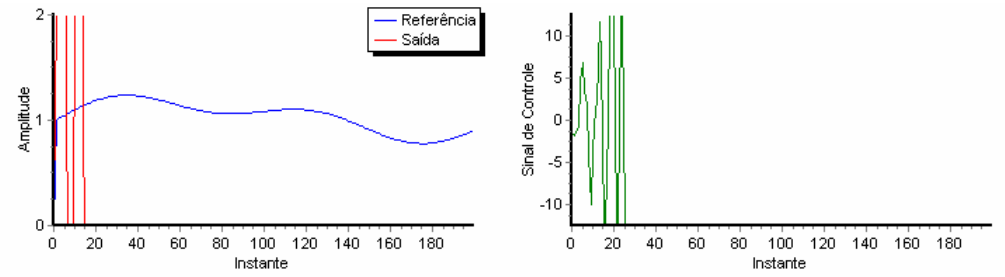

Figura 5.23 – Teste para $a_c = 0,9531$ e $b_c = 14,3$.

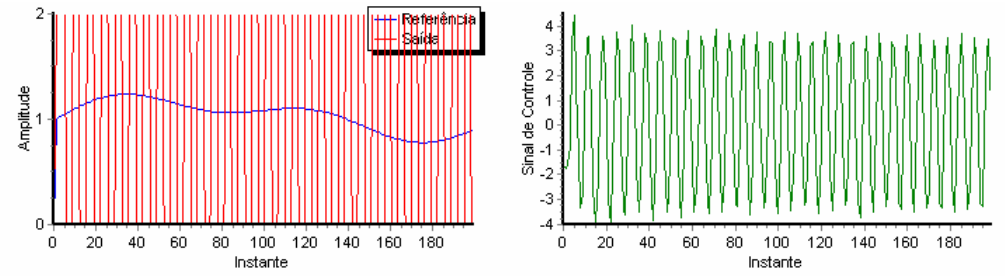

Figura 5.24 – Teste para $a_c = 0$ e $b_c = 15$.

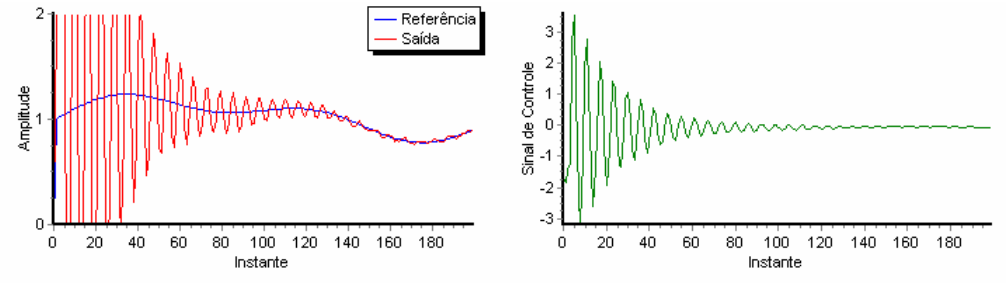

Figura 5.25 – Teste para $a_c = -1,054$ e $b_c = 15,8$.

Nos testes para plantas com $b_c = 5$, a Figura 5.26 mostra o resultado para $a_c = 1$, a Figura 5.27 mostra o resultado para $a_c = 0$ e a Figura 5.28 mostra o resultado para $a_c = -1$.

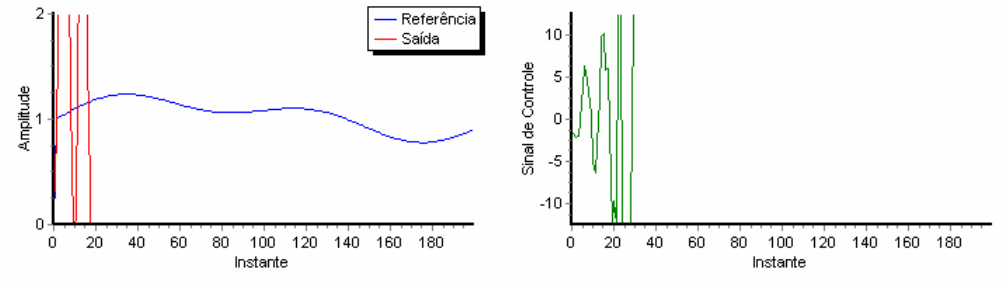

Figura 5.26 – Teste para $a_c = 0,9531$ e $b_c = 4,766$.

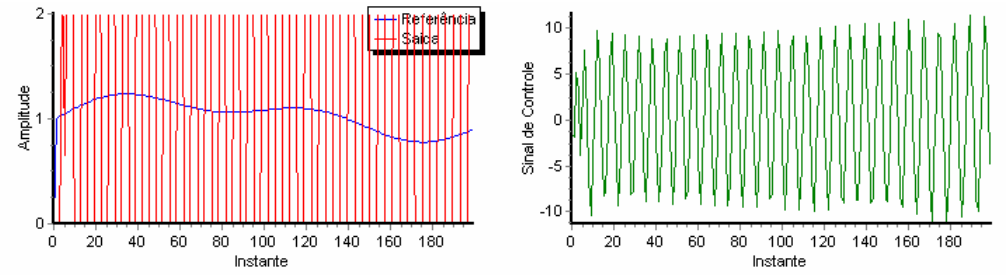

Figura 5.27 – Teste para $a_c = 0$ e $b_c = 5$.

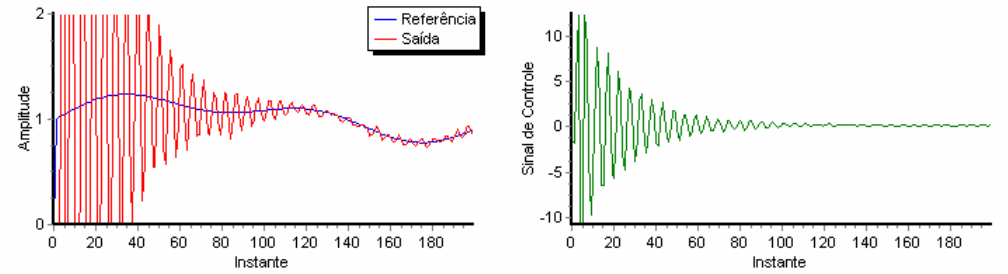

Figura 5.28 – Teste para $a_c = -1,054$ e $b_c = 5,268$.

Como se pode observar, o comportamento do esquema de controle para as plantas reais foi bastante semelhante ao seu comportamento em plantas discretas simuladas. Ficou bem claro o comportamento instável na Figura 5.20, Figura 5.23 e Figura 5.26; oscilatório na Figura 5.21, Figura 5.24 e Figura 5.27; e estável na Figura 5.22, Figura 5.25 e Figura 5.28. Uma diferença notável é que na planta analógica, os sinais são saturados em valores absolutos menores que 15V, e depois disso, seu comportamento não é mais o mesmo. Já nas plantas simuladas digitalmente, os valores podem variar numa faixa muito mais extensa, nos limites de representação dos valores em variáveis numéricas. A saturação normalmente ocorre quando o sistema é instável, pois seus estados ou saída crescem até atingir a saturação dos componentes. Mas, mesmo para um sistema estável, quando se exigem valores além dos valores possíveis para os estados ou saída, também ocorre saturação desses.

Capítulo 6

Conclusão

Com a análise feita para se descobrir o motivo que fazia com que o esquema de controle não funcionasse bem para todos os casos, conclui-se que o problema ocorre devido ao fato de $y[k+d]$ não depender somente de $u[k]$. Como a lei de controle é baseada na utilização do jacobiano (derivada parcial da saída $y[k+d]$ com relação à entrada $u[k]$), ficou esquecida a influência das entradas e saídas nos outros instantes, de acordo com a ordem e o atraso da planta. Pois, o modelo da planta pode ter como entradas valores anteriores da entrada da planta e da saída da planta.

Devido aos outros coeficientes do modelo da planta, além do jacobiano, terem sido desprezados na lei de controle, não há como saber o quanto será a influência das parcelas que contém esses coeficientes para se gerar um sinal de controle que corrija o erro e, ao mesmo tempo, anule a influência dessas parcelas.

Isso faz com que a lei de controle fique restrita aos casos onde o valor absoluto da influência do sinal de controle na variação da saída seja maior do que o valor absoluto da soma das influências das outras entradas e das saídas anteriores, de acordo com a ordem da saída e a ordem da entrada da planta.

Com essa conclusão, pode-se ter em vista o desenvolvimento de um método para analisar se uma planta discretizada poderá ser controlada pela lei de controle em questão. Através dessa análise, também poderá ser determinado se um período de amostragem usado na discretização da planta servirá ou não para a planta ser controlada pela lei de controle.

Outra perspectiva mais relevante é a elaboração de uma nova lei de controle que além de ser baseada no jacobiano, também leve em consideração as outras derivadas parciais da saída com relação às entradas e saídas anteriores de acordo com a ordem da entrada e da saída da planta. Essa nova lei de controle seria uma generalização da lei de controle estudada aqui, assim, abrangeria uma quantidade maior de plantas e faria um controle melhor nas plantas onde a lei já funcionava. Espera-se que essa nova lei de controle seja a contribuição de uma próxima etapa desta pesquisa, possivelmente durante um programa de doutorado.

Referências Bibliográficas

[1] WIDROW, B.; SMITH, F. W. **Pattern-recognizing control systems**. In 'Computer and Information Sciences Symposium Proceedings'. Spartan, Washington, DC, 1963.

[2] WALTZ, M. D.; FU, K. S. **A heuristic approach to reinforcement learning control systems**. IEEE Transactions on Automatic Control AC-10(4), p. 390-398. 1965.

[3] MICHIE, D.; CHAMBERS, R. A. **Boxes: An experiment in adaptive control**. In J. T. Tou and R. H. Wilcox (Eds.). 'Machine Intelligence'. Chapter 2, p. 137-152. Edingurgh: Oliver and Boyd, 1968.

[4] BARTO, A. G.; SUTTON, R. S.; ANDERSON, C. W. **Neuronlike adaptive elements that can solve difficult learning control problems**. IEEE Transactions on Systems Man and Cybernetics 13(5), p. 834-846. 1983.

[5] NG, G. W. **Application of neural networks to adaptive control of nonlinear systems**. London, England: Research Studies Press Ltd., 1997.

[6] KAWATO, M.; FURUKAWA, K.; SUZUKI, R. **A hierarchical neural network model for control and learning of voluntary movement**. Biological Cybernetics 57, p. 169-185. 1987.

[7] HUNT, K. J.; SBARBARO, D. **Neural networks for non-linear internal model control**. IEE Proceedings Control Theory and Applications 138, p. 431-438. 1991.

[8] WILLIS, M. J.; MONTAGUE, G. A.; DIMASSIMO, C; THAM, M. T.; MORRIS, A. J. **Artificial neural networks in process estimation and control**. Automatica 28(6), p. 1181-1187. 1992.

[9] JORDAN, M. I.; JACOBS, R. A. **Learning to control an unstable system with forward modeling**. In: R. P. Lippmann, S. E. Moody and D. S. Touretzky (Eds.). 'Advances in Neural Information Processing Systems'. San Mateo: Morgan Kaufmann, 1990.

[10] HARRIS, C. J.; MOORE, C. G.; BROWN, M. **Inteligent control**: aspects of fuzzy logic and neural nets. World Scientific. 1993.

[11] PSALTIS, D.; SIDERIS, A.; YAMAMURA, A. **A multilayered neural network controller**. In: IEEE Control Systems Magazine 8(2), p. 17-21. 1988.

[12] NARENDRA, K. S. **Adaptive control of dynamical systems using neural networks**. In D. White and D. Sofge (Eds.). In: Handbook of Intelligent Control, p. 141-183. New York: Van Nostrand Reinhold, 1992.

[13] ADETONA, O.; SATHANANTHAN, S.; KEEL, L. H. **Robust Nonlinear Adaptive Control Using Neural Networks**. In: Proc of the American Control Conference, p. 3884-3889. Arlington, VA. USA, 2001.

[14] NϕRGAARD, M.; RVN, O.; POULSE, N. K.; HANSEN, L. K. **Neural networks for modelling and control of dynamic systems**. London, England: Springer-Verlag London Limited, 2001.

[15] MAITELLI, A. L.; GABRIEL FILHO, O. **Controlador híbrido indireto baseado em redes neurais – parte I**: desenvolvimento e implementação. In: VI SIMPÓSIO BRASILEIRO DE AUTOMAÇÃO INTELIGENTE, p. 183-188. Bauru, 2003.

[16] HAYKIN, S. **Neural networks**: a comprehensive foundation. 2. ed. New Jersey: Prentice Hall, 1999.

[17] LUCENA, P. B.; ARAÚJO, F. M. U.; SALAZAR, A. O. **Controle de nível usando rede neural**. In: VII CONGRESSO BRASILEIRO DE REDES NEURAIS. Natal, 2005. 5 p.

[18] QUANSER Consulting Inc. **Coupled Water Tank Experiments**.

[19] APKARIRIAN, J. A comprehensive and modular laboratory for control systems design and implementation. Quanser Consulting, 1995.

[20] SPIESS, R. **Analog computers**. Comdyna, Inc., 1968. Disponível em: <http://www.comdyna.com/>. Acesso em: 18, nov., 2005.

[21] DEMUTH, H.; BEALE, M. **Neural Networks Toolbox User's Guide**. Massachusetts, USA: The MathWorks, 1992.

[22] JACOBS, R. A. **Increased rates of convergence through learning rate adaptation**. In: Neural Networks, V. 1, p. 295-307. USA: Pergamon Press plc, 1988.

[23] RIEDMILLER, M.; BRAUN, H. **A direct adaptive method for faster backpropagation learning**: The RPROP algorithm. In: Proceedings of the IEEE International Conference on Neural Networks, p. 586-591. San Francisco: IEEE Press, 1993.

[24] IGEL, C.; HÜSKEN, M. **Improving the Rprop learning algorithm**. In: Proceedings of the Second International ICSC Symposium on Neural Computation, p. 115-121. ICSC Academic Press, 2000.

[25] IOANNOU, P. A.; SUN, J. **Robust Adaptive Control**. Prentice Hall, 1996 (fora de impressão em 2003). Disponível em: <http://www-rcf.usc.edu/~ioannou/Robust_Adaptive_Control.htm>. Acesso em: 21, mar. 2005.

[26] RICE, J. R. **Numerical methods, software, and analysis**. 2. ed. San Diego: Academic Press, 1993.

[27] OGATA, K. **Discrete-time control systems**. New Jersey, USA: Prentice Hall, 1987.

[28] PHILLIPS, C. L.; NAGLE, H. T. **Digital control system analysis and design**. 3. ed. New Jersey, EUA: Prentice-Hall, 1995.

[29] ÅSTRÖM, K. J.; WITTENMARK, B. **Computer-controlled systems:** theory and design. 2. ed. New Jersey, USA: Prentice Hall, 1990.

[30] QUANSER CONSULTING, INC. **MULTIQ™ I/O board programming manual**.