Proceedings of the IV Brazilian Conference on Neural Networks - IV Congresso Brasileiro de Redes Neurais
pp. 209-214, July 20-22, 1999 - ITA, São José dos Campos - SP - Brazil

Um Estudo Comparativo Entre Diferentes Técnicas de Otimização do Treinamento de Neurocontroladores

José Augusto Dantas de Rezende[1]
André Laurindo Maitelli[2]
[1]Universidade Federal do Rio Grande do Norte
[2]Universidade Federal do Rio Grande do Norte
e-mails: jaugusto@leca.ufrn.br, maitelli@leca.ufrn.br

Abstract

This paper presents a method of on-line neurocontrol and compares three different optimization methods applied to insuring convergence velocity. The question about the convergence time is critical in neurocontrollers with real time training, which the controller training must be during the sampling period, usually small. We present the results presented by three differents optimization methods in a linear and a non-linear plant. We will see that different methods conduce to different results in the controller performance.

1. Introdução

Na última década, a tecnologia de Redes Neurais Artificiais têm conseguido conquistar cada vez mais espaço, atuando de forma eficiente em diversas áreas da engenharia. Suas aplicações vão desde a classificação de padrões, passando pelo mapeamento de funções e chegando até a filtragem de sinais [5].

Mais recentemente, com o cada vez maior desenvolvimento da tecnologia da informática, tem-se permitido um melhor aproveitamento dessa tecnologia. O grande problema que interfere na aplicação das RNA's a problemas de dinâmica mais rápida é o geralmente lento processo de treinamento das redes.

O treinamento é o processo através do qual a RNA absorve conhecimento. É a partir do treinamento que a rede absorve as características do ambiente ao qual está inserida e aprende a realizar as mais diferentes tarefas. Nos casos mais comuns, o treinamento da rede se dá através da apresentação de pares entrada-saída da rede. Dizemos que a rede aprendeu quando a mesma é capaz de repetir o relacionamento entrada-saída da tarefa em questão. O treinamento da rede nada mais é que um processo de otimização, no qual os parâmetros livres são os pesos da rede a serem ajustados [5].

O método de treinamento mais comumente utilizado é o *Back-Propagation*, que é baseado no método do gradiente. Mais recentemente, uma série de trabalhos têm proposto uma série de melhorias visando o aumento da velocidade de processamento. As classes mais simples de algoritmos, os chamados algoritmos adaptativos, fazem uso das informações disponíveis para promover um auto-ajuste dos parâmetros livres da rede, como o coeficiente de aprendizado.

Os algoritmos podem ser de dois tipos, de acordo com o tipo de informações utilizadas para o ajuste dos parâmetros: globais ou locais. Os globais são os que utilizam informações de toda a rede para o ajuste dos parâmetros, também válidos para toda a rede. Nos algoritmos locais, o ajuste dos parâmetros é feito com o auxílio de informações locais da rede e os parâmetros ajustados também terão efeito em apenas algumas regiões da rede.

O método de Neurocontrolaador utilizado foi proposto em [1]. O Neurocontrolador original propõe o algoritmo Eta-Adaptativo, que é um algoritmo adaptativo geral, para o aumento da velocidade de treinamento da rede. Neste trabalho, propomos uma comparação entre os resultados obtidos pelo treinamento com este algoritmo, e algoritmos adaptativos de característica específica, que é o caso do *Delta-Bar-Delta*, [2], e *SuperSAB*, [3].

Apresentamos uma série de simulações com diferentes plantas, de diferentes características e diferentes sinais de referência a serem seguidos. Procedemos, então, uma comparação entre a qualidade de controle permitida por cada um dos métodos.

2. Problemas Associados ao Algoritmo Back-Propagation

Em virtude de sua simplicidade e bons resultados obtidos, o algoritmo *Back-Propagation* se tornou o mais conhecido e mais extensamente usado nos trabalhos relacionados a RNA's, seja qual for a aplicação à qual se destine a RNA.

O método consiste basicamente na seguinte equação:

$$\Delta w_{hj} = -\eta \frac{\partial E_h}{\partial w_{hj}} \qquad (1)$$

A seguir, propôs-se a adição de um termo chamado de momento normalizado e que acrescenta o

conhecimento da correção do erro feita no passo anterior na correção do peso atual. A equação final clássica do algoritmo *Back-Propagation* ficou da forma apresentada na equação (2), e é até hoje a equação de correção dos pesos mais utilizada:

$$\Delta w_{hj}(n) = -\eta \frac{\partial E(n)}{\partial w_{hj}(n)} + \alpha \Delta w_{hj}(n-1) \qquad (2)$$

Todavia, acharmos o coeficiente de aprendizado (η) ideal não é uma tarefa simples. O valor adequado depende do perfil da função de erro, que obviamente, muda com o treinamento da rede. Um pequeno coeficiente de aprendizado resultará em longo tempo de convergência. Já com um valor elevado de coeficiente de aprendizado, possivelmente teremos o sistema oscilando sobre um mínimo. Além do mais, o algoritmo, sendo baseado no método gradiente, não nos garante que o mínimo encontrado seja o mínimo global, e não um mínimo local indesejado, [3].

Temos uma série de outros problemas associados ao método, porém é este problema, o do ajuste do coeficiente de aprendizado, que os algoritmos aqui apresentados procuram solucionar.

3. Algoritmos Adaptativos

3.1. Algoritmos Adaptativos com Estratégias Globais – Eta-Adaptativo

Os algoritmos adaptativos são aqueles que fazem o ajuste de alguns dos parâmetros da rede. Entre as técnicas com estratégias globais, implementamos apenas uma, o Eta-Adaptativo. O uso do Eta-Adaptativo, como alternativa para melhoria da velocidade do treinamento em Neurocontroladores, foi proposto anteriormente em [1].

O método é bastante simples, consiste basicamente em aumentar o η (coeficiente de aprendizado) caso o erro global na saída da rede esteja diminuindo e diminuí-lo caso o erro esteja aumentando. Caso o erro global esteja aumentando, significa que o treinamento da rede está levando os pesos para uma direção errada, diminuímos então o η para que este treinamento errado não comprometa o valor dos pesos. Caso o erro esteja diminuindo, isto significa que o treinamento está realmente levando os pesos para uma região próxima a um mínimo, aumentamos então o η para que este mínimo seja atingido mais rapidamente.

A grande vantagem desse método é sua simplicidade, não faz uso de refinamentos como a derivada segunda ou um η para cada um dos pesos, como é comum em outros métodos baseados em estratégias locais. Ao contrário do que possa parecer, apesar da simplicidade do método, os resultados obtidos são bons. Abaixo apresentamos o algoritmo para a implementação do método.

```
Se (Erro ≤ 1,04*Erro_1)
    Se (Erro > Erro_1)
        η = η;
    Senão
        η = 1.05 * η;
Senão
    η = 0.7 * η;
FimSe
Erro_1= Erro;
```

Algoritmo 1: Algoritmo *Eta-Adaptativo*

A variável *Erro_1* serve para armazenar o erro global de uma iteração para que seja comparado com erro global da iteração do passo seguinte. A variável *Erro* é o próprio erro global na saída da rede [1].

3.2 Algoritmos Adaptativos com Estratégias Locais – Delta-Bar-Delta e SuperSAB

Os algoritmos que fazem uso de estratégias locais são aqueles em que são usadas informações locais da rede para se fazer os ajustes dos parâmetros. Além do mais, os ajustes realizados têm efeito local, ou seja, normalmente existe um parâmetro (como o η ou β) específico para cada neurônio ou camada da rede e os ajustes são calculados por neurônio (ou por camada) e afetam apenas o parâmetro específico do mesmo. Esta é a técnica utilizada pelo algoritmos *Delta-Bar-Delta e SuperSAB*.

3.2.1. Delta-Bar-Delta

Este algoritmo propõe um método para o auto-ajuste do coeficiente de aprendizado (η). Definimos um coeficiente para cada neurônio e promovemos o seu ajuste através da monitoração das derivadas de primeira e segunda ordem, definindo a partir deles o ajuste cabível para cada um dos coeficientes.

O algoritmo *Delta-Bar-Delta*, [2], está baseado na regra *Delta-Delta*.. Esta regra propõe um ajuste dos coeficientes de aprendizado a partir do gradiente do erro da rede em função do coeficiente de aprendizado, como mostra a equação (3).

$$\Delta \eta_i(n) = \gamma \frac{\partial G(n)}{\partial \eta_i(n)} \qquad (3)$$

O parâmetro γ é uma pequena constante positiva, *G(n)* é a função erro na saída da planta. Não confundir com *E(n)*, que é o erro em função das sinapses. O

gradiente do erro da rede, em função do coeficiente de aprendizado pode ser simplificado conforme a equação (4) abaixo.

$$\Delta\eta_i(n) = \gamma \frac{\partial E(n)}{w_i(n)} . \frac{\partial E(n-1)}{w_i(n-1)} \quad (4)$$

Como forma de melhorar o desempenho e simplificar a equação do método *Delta-Delta*, foi proposto o algoritmo *Delta-Bar-Delta*, [5]. Este algoritmo também propõe coeficientes de aprendizado específicos para cada um dos pesos, baseado na idéia de que a função erro é unidimensional em relação a cada um dos pesos. Podemos então propor uma regra para encontrarmos os coeficientes específicos para cada um dos pesos que minimizem cada uma dessas funções.

A estimação é baseada na observação da vizinhança das derivadas parciais durante os últimos dois pesos sucessivos. Se as derivadas têm o mesmo sinal, o η é incrementado linearmente por uma pequena constante para acelerar o processo de convergência nas regiões de pequeno erro. Por outro lado, uma mudança no sinal entre duas derivadas sucessivas indica que o treinamento levou a uma ultrapassagem do mínimo local, ou seja o último incremento no peso foi muito grande. Como consequência, o η é decrementado exponencialmente, através da multiplicação do mesmo por uma constante de decremento menor que a unidade.

$$\Delta\eta_{hj}(n+1) = \begin{cases} \kappa & se \quad S_{hj}(n-1)D_{hj}(n) > 0 \\ -\Phi\eta_{hj}(n) & se \quad S_{hj}(n-1)D_{hj}(n) < 0 \\ 0 & caso \quad contrário \end{cases} \quad (5)$$

as variáveis S_{hj} *e* D_{hj} são dadas pelas equações (6) e (7), apresentadas abaixo:

$$S_{hj}(n) = (1-\xi)D_{hj}(n-1) + \xi S_{hj}(n-1) \quad (6)$$

$$D_{hj}(n) = \frac{\partial E(n)}{w_{hj}(n)} \quad (7)$$

As constantes κ, ξ e Φ estão entre 0 e 1.

O fato de o coeficiente η crescer linearmente e decrescer exponencialmente, permite ao algoritmo uma rápida reação a mudanças na curva da função erro e evita grandes oscilações e problemas de *overshoot* associados com o *Back-Propagation* convencional. Um termo de momento também foi utilizado por Jacobs em conjunção com o *Delta-Bar-Delta* para permitir um aumento a velocidade de convergência sem perda de generalidade. A equação final do método está apresentada na equação (8)

$$w_{hj}(n+1) = w_{hj}(n) - \eta_{hj}(n) . \frac{\partial E(n)}{\partial w_{hj}} + \alpha w_{hj}(n-1) \quad (8)$$

3.2.2 – SuperSAB

O *SuperSAB*, [3], é também baseado na técnica de monitoração das duas últimas derivadas parciais do erro. Caso não haja mudança de sinal, incrementa-se o η, caso contrário, decrementa-se. Dessa forma, o método se assemelha bastante ao Delta-Bar-Delta. Com uma diferença básica: tanto o incremento como o decremento do coeficiente de aprendizado (η) são feitos de maneira exponencial. Veja a equação básica do método abaixo:

$$\Delta\eta_{hij}(n+1) = \begin{cases} \eta_{hij}(n-1)u & se \quad \frac{\partial E_h}{\partial w_{hj}}(n) . \frac{\partial E_h}{\partial w_{hj}}(n-1) > 0 \\ \eta_{hij}(n-1)d & se \quad \frac{\partial E_h}{\partial w_{hj}}(n) . \frac{\partial E_h}{\partial w_{hj}}(n-1) < 0 \\ 0 & caso \quad contrário \end{cases} \quad (9)$$

O SuperSAB mostra-se um algoritmo consideravelmente mais rápido que o *Back-Propagation* convencional. Um dos possíveis problemas que certamente encontraremos é o grande número de parâmetros a serem ajustados a fim de se obter uma boa convergência. Entre esses parâmetros estão o η inicial, o fator momento (α) e os fatores de incremento (u) e decremento (d) do coeficiente de aprendizado. Para diminuir o número de parâmetros a serem ajustados podemos fazer: u = 1/d, com u entre 1.1 e 1.5.

4. Neurocontrolador Implementado

O Neurocontrolador para sistemas SISO implementado foi proposto originalmente em [1]. O esquema baseia-se no método de otimização em malha fechada. Consiste basicamente do uso de duas estruturas neurais, conhecidas como Neuro-emulador e Neuro-Controlador. O Neuro-Emulador é responsável pelo aprendizado da dinâmica da planta. Toda a informação absorvida pelo Neuro-Emulador será usada posteriormente no processo de treinamento do neuro-Controlador que fará o controle da planta [6].

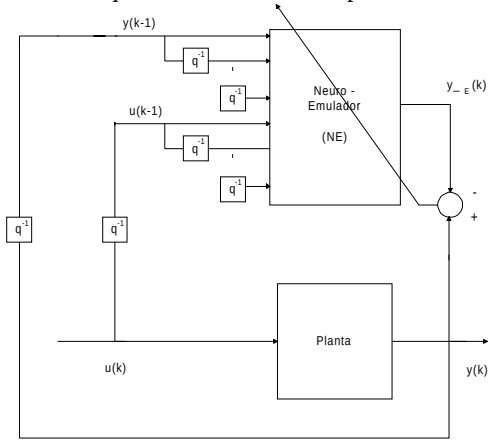

Figura 1: Treinamento do Neuro-emulador.

O esquema da figura (1) ilustra o treinamento do neuro-emulador.

Primeiramente efetuamos o treinamento do NE junto a planta. Apresentamos em sua entrada os valores passados de entrada e saída da planta e comparamos a saída da rede com a saída atual da planta. Ao final do processo de treinamento, a rede deve ser capaz de repetir o comportamento da planta. O Neuro-emulador é então utilizado para o treinamento do Neuro-Controlador, como veremos na figura abaixo que ilustra o treinamento do Neuro-controlador.

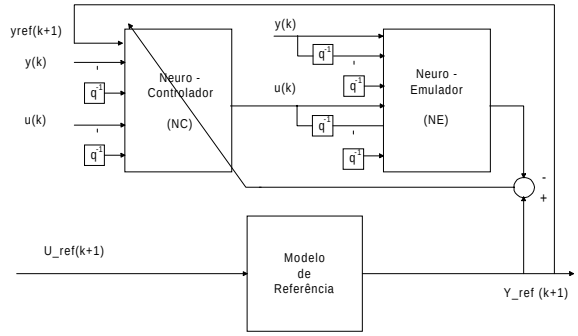

Figura 2: Treinamento do Neuro-controlador.

O Neuro-controlador aprende primeiramente a controlar o Neuro-emulador, fazendo-o acompanhar um sinal de referência. Após o treinamento, espera-se que o Neuro-Controlador esteja pronto para gerar os sinais de controle que serão colocados a entrada da planta para fazê-la seguir o modelo de referência.

Observa-se que o processo de treinamento do Neuro-Controlador só começa após concluído o processo de treinamento do Neuro-Emulador. O que ressalta ainda mais a necessidade de um algoritmo de treinamento rápido e eficiente, já que todo o treinamento da planta se dará durante o período de amostragem.

A arquitetura das Redes Neurais Artificiais utilizadas é do tipo Perceptron de Múltiplas Camadas com apenas uma camada intermediária e cinco neurônios na mesma.

5. Simulações

Apresentamos a seguir algumas simulações realizadas.

5.1. Sistema Linear de Segunda ordem do Tipo Zero

Nesta seção utilizamos como exemplo um sistema linear de Segunda ordem, apresentado em [1]. O sistema é do tipo um, com um pólo na origem e um pólo estável. A função de transferência do sistema é dado pela seguinte expressão:

$$H(s) = \frac{k_p}{s(s+p)} \qquad (10)$$

Discretizando a expressão acima e utilizando os valores k_p = 0,2, p = 0,5 e T_S = 1 s, teremos:

$$y(k+1)-1,6065.y(k)+0,6065.y(k-1) = 0,0852.u(k)+0,0722.u(k-1) \qquad (11)$$

Utilizamos como referência a ser seguida pelo sinal, um sinal senoidal de frequência angular w_n = 0,01π rad/s. Desta vez saturamos os sinais de controle da planta no intervalo [-1, 1].

Neste caso, o neuro-emulador e o neuro-controlador têm cada um 4 neurônios na entrada e 1 na camada de saída.

Os resultados obtidos por cada um dos métodos estão apresentados abaixo:

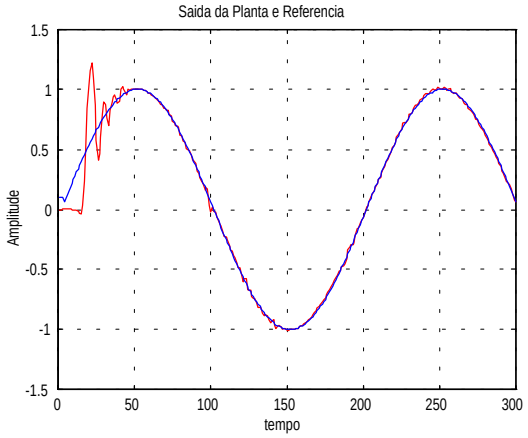

Figura 3: Saída da Planta com o Neuro-Controlador treinado pelo Eta-Adaptativo

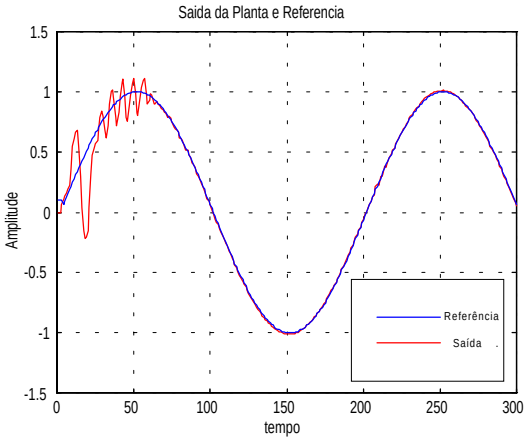

Figura 4: Saída da Planta com o Neuro-Controlador treinado pelo Delta-Bar-Delta

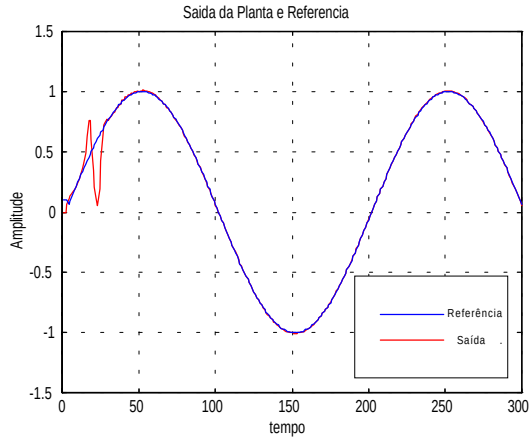

Figura 5: Saída da Planta com o Neuro-Controlador treinado pelo SuperSAB

Algumas comparações podem ser feitas, a tabela 1 apresenta alguns índices que utilizamos para efetuarmos a comparação entre os controladores.

O primeiro deles foi o Erro Total na Saída da Planta (ETSP), ou seja, a soma (em módulo) de todas as diferenças entre a saída da planta e a referência. Outro item de análise é o máximo *overshoot* (MO) apresentado pelos diferentes métodos. O terceiro item foi o Erro Total Médio (ETM), que é dado pelo Erro Total na Saída da Planta dividido pelo número de períodos de amostragem na qual a planta foi observada. O quarto item é o Erro Total em Regime (ETR), dado pela soma de todos os erros após os cem primeiros períodos de amostragem, o que dá uma idéia do desempenho em regime proporcionado pelo controlador. Neste primeiro exemplo, utilizamos para cálculo do Erro Total em Regime o erro acumulado após os primeiros 75 períodos de amostragem, já que o sinal de referência tem período igual a 75.

ALGORITMO	ETSP	MO	ETM	ETR
Eta-Adaptativo	11,2485	0,6305	0,0375	2,5189
Delta-Bar-Delta	12,1261	0,7199	0,0404	1,9212
SuperSAB	4,7993	0,5534	0,0160	0,6010

Tabela 1: Tabela de análise dos resultados obtidos com diferentes métodos de otimização no sistema de segunda ordem

A análise dos gráficos e dos números das tabelas nos permite afirmar que o algoritmo *SuperSAB* permite um menor erro tanto em regime como durante todo o período de simulação. O período transitório é mais curto e o *overshoot* do sistema controlado pelo Neurocontrolador treinado com este método é também menor do que o *overshoot* obtido com outros métodos. Apenas o esforço de controle exigido pelo Neurocontrolador é um pouco maior (em torno de 5,9 %) do que o exigido no controlador treinado pelo *Delta-Bar-Delta*, o qual também permite um erro em regime menor do que o controlador treinado com o *Eta-Adaptativo*, embora possua um transitório mais longo e com maior *overshoot*.

5.2. Sistema Não-Linear

A planta utilizada como exemplo nessa seção é um sistema não-linear apresentado em [4], e é dado pela seguinte equação dinâmica discreta:

$$y(k+1) = \frac{0{,}9.y(k) + u(k)}{1 + y(k)^2} \qquad (12)$$

onde *y(k)* é a saída da planta no instante discreto *k* e *u(k)*, o sinal de controle da planta no instante *k*.

Os sinais de controle no sistema foram saturados dentro do intervalo [-1,1]. Nesse sistema utilizamos um modelo de referência de segunda ordem:

$$y\_Ref(k+1) - 1{,}3205.y\_Ref(k) + 0{,}4966.y\_Ref(k-1) = 0{,}0983.u\_Ref(k) + 0{,}0778.u\_Ref(k-1) \qquad (13)$$

o modelo de referência em questão é equivalente a um modelo discreto com $\zeta = 0{,}7$, $w_n = 0{,}5$ rad/s e $T_s = 1$s.

Neste caso, o neuro-emulador e o neuro-controlador têm 2 neurônios na entrada e 1 na saída.

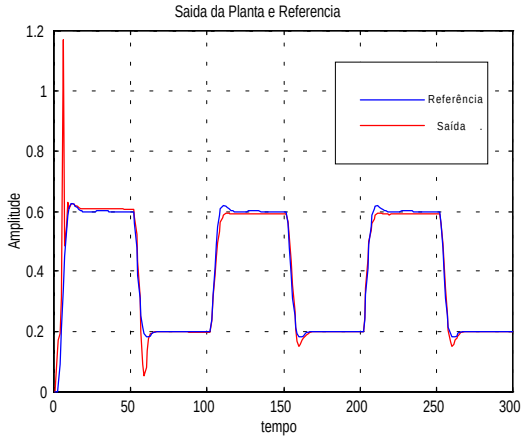

Figura 6: Saída da Planta com o Neuro-Controlador treinado pelo Eta-Adaptativo

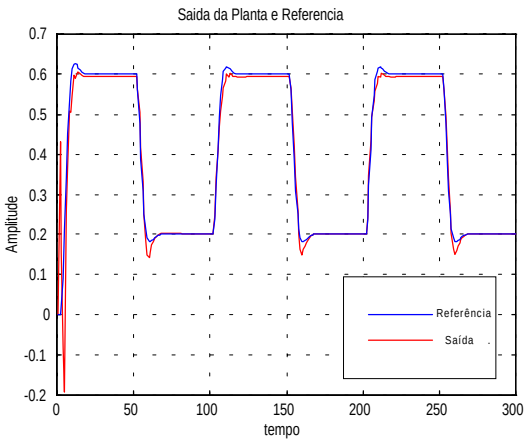

Figura 7: Saída da Planta com o Neuro-Controlador treinado pelo Delta-Bar-Delta

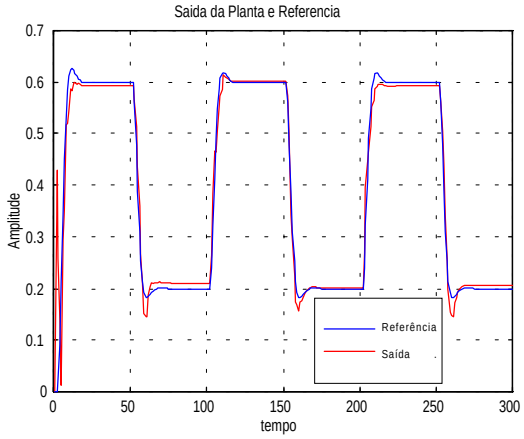

Figura 8: Saída da Planta com o Neuro-Controlador treinado pelo SuperSAB

A seguir apresentamos uma tabela com os mesmos parâmetros de comparação para que possamos analisar o desempenho dos métodos quando aplicados a um sistema não-linear.

ALGORITMO	ETSP	MO	ETM	ETR
Eta-Adaptativo	4,1375	0,8256	0,0138	1,7948
Delta-Bar-Delta	3,5925	0,4300	0,0120	1,4878
SuperSAB	4,0414	0,4300	0,0135	1,7776

Tabela 1: Tabela de análise dos resultados obtidos com diferentes métodos de otimização no sistema não-linear.

Pelos gráficos e índices apresentados na tabela acima, podemos concluir que o controlador treinado com o algoritmo *Delta-Bar-Delta* possui um Erro Total na Saída da Planta, bem como um Erro Total Médio inferior aos demais algoritmos, o que implica que este algoritmo permite à rede uma melhor qualidade de desempenho. O *overshoot* proporcionado pelo método também é o menor, embora seja o mesmo do observado pelo *SuperSAB, é*, porém, substancialmente menor do que o obtido com o algoritmo *Eta-Adaptativo*. O Erro Total em Regime, que é o erro acumulado excluídos os cem primeiros períodos de amostragem (onde geralmente ocorrem as maiores diferenças entre a saída da planta e a referência) também aponta um melhor desempenho do *Delta-Bar-Delta*. Se compararmos os índices do *SuperSAB* com o *Eta-Adaptativo*, veremos que o *SuperSAB* leva vantagem em todos eles, o que o coloca como o segundo melhor algoritmo com relação ao treinamento do controlador aplicado a esta planta não-linear.

5. Conclusões

A primeira dificuldade que surge, em qualquer trabalho de Neurocontrole, é a necessidade de se melhorar a velocidade do processo de treinamento da rede, visto que as mesmas possuem um lento processo de convergência. A revisão bibliográfica que resulte em métodos realmente úteis na melhoria de velocidade de convergência é, então, fundamental para o êxito da estratégia de controle. O problema surge porque, na maioria dos casos, os métodos são idealizados para serem implementados em problemas de classificação de padrões.

Dificilmente, ao final do trabalho, poderemos concluir qual o melhor algoritmo para o treinamento da rede. No máximo, pode-se estabelecer qual a classe de aplicação que cada um dos algoritmos melhor se enquadra. O importante é que se tenha uma ferramenta que possa disponibilizar ao usuário uma escolha entre vários tipos de algoritmos para que este, a partir de suas próprias simulações e ensaios, possa definir qual a melhor ferramenta para sua aplicação.

À luz dos exemplos apresentados, verificamos que o método SuperSAB apresentou o melhor desempenho.

Referências

[1] Maitelli, A.L. e Gabriel, O.F. (1996). Um Esquema de Controle Adaptativo Neural com Treinamento em Tempo Real. Dissertação de Mestrado. UFRN. Natal-RN

[2] Jacobs, R.A. (1988). Increased Rates of Convergence Through Learning Rate Adaptation. *Neural Networks*, 1:295-307.

[3] Tollenaere T. (1990). SuperSAB: Fast Adaptive Backpropagation with Good Scaling Properties. *Neural Networks*, 3:561-573.

[4] Maia, C. A. e Resende, P. Um Controlador Neural Gain Scheduling para Plantas Não-Lineares. Revista SBA Controle & Automação, 9 (3): 135-140.

[5] Haykin, S. (1994). *Neural Networks*. Macmillan College Publishing Company. Ontario. Canada.

[6] Hrycej, T. (1997). *Neurocontrol: Towards an Industrial Control Methodology*. John Wiley & Sons. Ulm. Alemanha.

Agradecimentos

A CAPES e Eletrobrás, pelo apoio financeiro.