Proceedings of the IV Brazilian Conference on Neural Networks - IV Congresso Brasileiro de Redes Neurais
pp. 350-355, July 20-22, 1999 - ITA, São José dos Campos - SP - Brazil

IMPLEMENTAÇÃO EM HARDWARE DE UM CONTROLADOR NEURAL PARA ROBÔ

GHILSON R. CORRÊA[1], EDILBERTO P. TEIXEIRA[2], ELMO B. FARIA [3]
[1] Universidade Federal de Uberlândia - D.E.E. - 38.400-902 - Uberlândia-MG - Brasil
[2] Universidade Federal de Uberlândia - D.E.E. - 38.400-902 - Uberlândia-MG - Brasil
[3] Centro Universitário do Triângulo - D.C.C. - 38.402-310 - Uberlândia-MG - Brasil
Emails: ghilson@ufu.br, edilbert@ufu.br, ebfaria@ufu.br

Abstract

This paper presents the implementation of a neural controller applied for controlling a six joint robot. The Neural network is used as a nonlinear compensator for the dinamic system. A new control scheme was developed using the neural network to estimate the computed torque functions. The system was implemented using three PIC microcontrollers and a pentium computer. The details of the hardware and software design are presented including the laboratory results.

1. Introdução.

Muitos processos práticos envolvem comportamentos não-lineares. Todavia, os robôs, sistemas altamente não lineares e acoplados, são exemplos particularmente adequados para teste de estratégias de controle baseadas em redes neurais artificiais. Dentre as diversas tentativas de controle através de redes neurais, destacam-se aquelas onde se procura compensar as não linearidades inerentes à dinâmica dos robôs [10],[11],[12],[13] e [14]. Neste trabalho, usa-se o principio do controle através do torque computado efetuando-se a compensação através de uma rede neural de alimentação direta.

Ao contrário de alguns trabalhos onde a compensação é parcial, efetua-se a compensação completa da equação dinâmica. Isto foi possível através do treinamento *off-line* de uma rede neural com 20 neurônios na sua camada intermediária. Os detalhes do treinamento da rede, o projeto de hardware e os resultados obtidos em laboratório com um robô de seis juntas rotacionais são apresentados neste artigo.

O hardware implementado em laboratório foi desenvolvido atendendo as exigências do projeto. Entretanto, alguns fatores importantes foram levados em consideração uma vez que, optou-se por usar microcontroladores com arquitetura RISC, de modo a facilitar sua aplicação tanto em hardware como em software. Dentre estes, pode-se citar: facilidade de aquisição no mercado, custo reduzido, de forma a não inviabilizar o projeto, e ferramentas de suporte para utilização do dispositivo.

Procurou-se detalhar o treinamento da rede neural quando aplicada para processos MIMO não-lineares. Sendo assim, na seção 2, estão ilustrados o modelo da rede neural, a expressão matemática representativa e a forma de treinamento empregada. Na seção 3, detalham-se a estrutura da planta não-linear e as equações diferenciais que a regem. O hardware implementado em laboratório é mostrado na seção 4. Os resultados obtidos pela rede neural bem como as conclusões e os comentários finais estão dispostos na seção 5 e 6.

2. Modelo para a rede neural.

Uma rede neural feedforward multicamadas consiste de um determinado número de neurônios interconectados e organizados em sucessivas camadas. Cada neurônio da rede tem habilidade independente de processamento e não tem qualquer interconexão com outros neurônios da mesma camada. A descrição matemática genérica da rede neural com *q* camadas é:

$$\hat{y}(t+1) = \sum_{i=1}^{H_q} w_i^q s\left[\sum_{k=1}^{H_{q-1}} w_{ik}^{q-1} s\left[\ldots s\left[\sum_{i=1}^{n} w_{sj}^1 y(t-j+1) + \sum_{j=1}^{m} w_{s,n+j}^1 u(t-j+1)\right]\ldots\right]\right], \quad (1)$$

onde $\hat{y}(t+1)$ é a saída estimada pela rede neural, $y(t)$ e $u(t)$ são, respectivamente, as saídas e as entradas do processo, W_{sj}^k são os pesos sinápticos na camada *k*. A função de ativação é: $s(.) = \{1 + \exp[-(.)]\}^{-1}$.

A rede neural utilizada no projeto tem a estrutura de 3 camadas: uma de entrada, uma intermediária e uma de saída [8, 11, 12], conforme ilustração da figura 1.

A entrada para a rede neural é um vetor, contendo as posições, velocidades e acelerações de comando das juntas do robô. Os valores de entrada e saída da rede neural são

normalizados entre -1 e 1. A saída é composta de três vetores contendo respectivamente os torques no tempo (*t*) desenvolvidos em cada uma das juntas do robô. A camada de entrada é formada por 18 elementos, os quais estão conectados a um conjunto de 20 neurônios na camada intermediária, que por sua vez estão conectados a outros seis na unidade de saída. A camada intermediária e a camada de saída usam função de ativação do tipo sigmoidal.

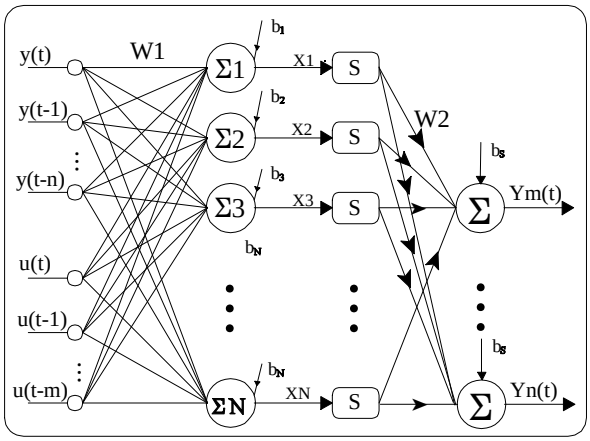

Figura 1. Estrutura da Rede Neural com uma camada intermediária.

O treinamento da rede neural é realizado após a coleta de 3600 dados provenientes do acionamento aleatório das seis juntas do robô. Geraram-se, então, doze vetores contendo as informações de entrada e saída da planta. O algoritmo utilizado para treinamento da rede neural foi o de propagação retroativa de erro [7]. Os dados relativos às entradas e saídas foram medidos em tempos fixos de amostragem. A entrada para a rede neural é um vetor, contendo as posições, velocidades e acelerações de comando das juntas do robô. Este esquema, permite que a rede neural seja treinada off-line.

A seguir, são apresentados os seguintes passos essenciais para treinamento da rede neural.

a) Apresentam-se os vetores de posição, velocidade e aceleração à entrada da rede neural.

b) Calcula-se a saída da camada intermediária utilizando-se a seguinte equação:

$$h_k = \frac{1}{1 + e^{-\sum_i V_{ik} S_i}} \qquad (2)$$

Onde:

V_{ik} = Elementos da matriz de pesos situada entre a camada de entrada e a intermediária.

S_i = Vetor de entrada.

c) Calcula-se a saída da rede com a seguinte equação:

$$u_j = \frac{1}{1 + e^{-\sum_k W_{kj} h_k}} \qquad (3)$$

Onde:

W_{kj} = Elemento da matriz de pesos situada entre a camada intermediária e a saída.

h_k = Saída da camada intermediária.

d) Calcula-se o valor de ΔW_{kj} a ser adicionado à matriz de pesos W_{kj}.

$$\Delta W_{kj} = \eta \delta_j h_k + \alpha \Delta W_{kj}(n-1) \qquad (4)$$

Onde:

η = Taxa de aprendizagem.

δ_j = Erro entre o valor desejado e o obtido na saída da rede.

α = Fator de amortecimento.

$\Delta W_{kj}(n\text{-}1)$ = último valor adicionado à matriz de pesos W_{kj}.

No trabalho, o fator de amortecimento α (expresso na eq. 4), que determina o efeito da variação do peso anterior para o cálculo do próximo, na direção do mapeamento desejado, usado no treinamento da rede neural, foi de 0,001. No sentido de reduzir o tempo de aprendizagem, variou-se a taxa de aprendizagem η que determina o ganho, a cada iteração do algoritmo, na faixa compreendida entre (0,65 a 0,018).

e) Atualizam-se as matrizes de pesos ΔW_{kj} e ΔV_{ik} , respectivamente.

f) Repetem-se os passos de *a* a *e* até que o erro quadrático mínimo estabelecido, seja atingido.

Este procedimento faz com que, durante o processo de aprendizagem, as conexões dos pesos sejam ajustadas para minimizar a soma quadrática de erro entre as saídas desejadas e as saídas da rede neural. Os sinais de erro são então propagados retroativamente para ajustar os pesos das conexões.

3. Estrutura da planta.

Como os robôs se caracterizam por terem comportamentos não lineares e efeitos de acoplamentos entre juntas, tais fatores dificultam o controle em velocidades altas, principalmente quando são empregados métodos de controle convencionais. Na tentativa de superar estas dificuldades, vários algoritmos de controle usando as características dinâmicas do robô manipulador têm sido propostos [1],[4],[5],[6],[9].

Como as redes neurais são apropriadas para se aproximar mapeamentos não lineares, elas se tornaram uma poderosa ferramenta de controle. Surgiu, então, um novo e

amplo campo de pesquisas denominado *neurocontrole*, que pode ser definido como: o uso de redes neurais como *controladores*. Diversas maneiras de incluir redes neurais no controle de sistemas dinâmicos foram propostas, seja para identificação, seja associadas a controladores convencionais, ou mesmo desempenhando a função de controladores. Por conseqüência, neste último caso, as redes neurais são também chamadas mais apropriadamente de *Neurocontroladores*.

A equação dinâmica de um robô manipulador descreve a relação entre a taxa de variação da configuração e os torques exercidos pelos atuadores em cada junta (motores elétricos, pneumáticos, etc.). Para um robô manipulador com *n* graus de liberdade a equação dinâmica baseada na formulação Lagrangeana pode ser descrita como:

$$\tau = M(q).\ddot{q} + H(q,\dot{q}) + G(q) + F(\dot{q}) \qquad (5)$$

Onde:
$\tau \rightarrow$ Vetor *nx1* de esforços de acionamento no tempo *t* (entrada generalizada forças/torques).
q, $\dot{q}$ e $\ddot{q}$ →Vetores *nx1* representando, respectivamente *posição, velocidade* e *aceleração* das variáveis de juntas.
M(q) →Matriz *nxn* simétrica e não-singular que representa a inércia.
H(q, $\dot{q}$) →Vetor *nx1* especificando termos de reações coriolis e forças centrífugas.
G(q) →Vetor *nx1* especificando os efeitos devido a gravidade e forças externas.
F($\dot{q}$) →Matriz diagonal *nxn* especificando coeficientes de atrito viscoso.
Cada elemento de M(q) e G(q) é uma função que depende da posição de cada junta do robô. Já o termo H(q, $\dot{q}$) que especifica reações coriolis e forças centrífugas é uma função tanto da posição (q) quanto da velocidade ($\dot{q}$) [2],[3] e [5].

O controlador neural usando o método torque computado, mostrado na figura 2, é implementado em dois estágios: realimentação de posição e velocidade e compensador não linear, proporcionado pela rede neural. Neste método, a rede neural é treinada para aprender a estrutura completa da equação dinâmica do robô. Desse modo, a aprendizagem passa a ser generalizada de maneira que todos os termos da equação dinâmica do robô sejam inseridos no processo de aprendizagem da rede neural. Contudo, para que este procedimento seja completado com sucesso, a rede neural precisa assimilar toda a dinâmica do robô.

Posteriormente, a rede neural é colocada no modo de operação, atuando como controlador. Neste caso, pode-se verificar pelo diagrama da figura 2 que a rede neural aprende o mapeamento dos torques a partir da posição, velocidade e aceleração desenvolvida em cada uma das juntas do robô. Então, fornecendo-se as posições, velocidades e acelerações desejadas pode-se gerar os torques desejados, capazes de levar o efetuador até a posição desejada.

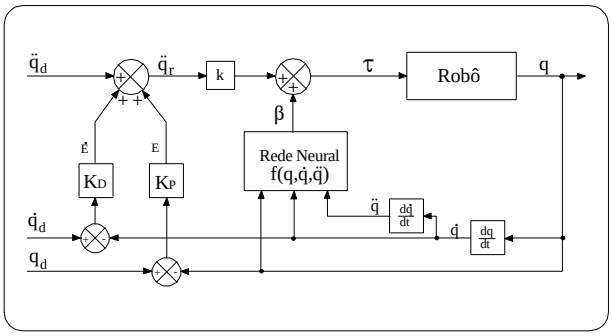

Figura 2- Diagrama de blocos do controlador neural usando o método torque computado.

4. Hardware implementado.

Descreve-se, nesta seção, o sistema de hardware implementado em laboratório e que foi utilizado no processo. O hardware digital implementado é baseado em três microcontroladores da família PIC da Microchip. Trata-se de uma família de microcontroladores com arquitetura RISC de 14-bits e custo bastante reduzido. Oferece uma ampla faixa de opções, desde manejamento de interrupções até geração de sinais PWM com processamentos independentes da CPU. Há um conjunto de instruções simples, mas extremamente poderoso que enfatiza as operações *byte*, *bit* e de *registradores* [15].

A figura 3 ilustra o diagrama de blocos da arquitetura geral do sistema desenvolvido para o robô MA2000. Basicamente, ele está dividido em dois níveis de hierarquia. O primeiro nível, o mais alto, é composto por um computador IBM-Pentium servindo como computador mestre de controle, passando os comandos para serem interpretados por três microcontroladores PIC16C73A que representam o segundo nível.

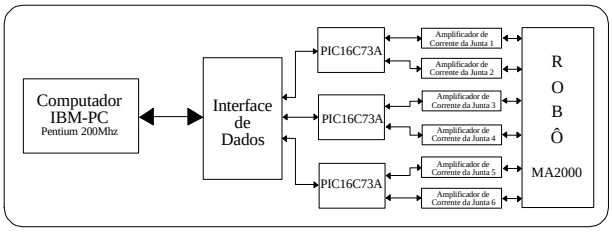

Figura 3- Diagrama de blocos simplificado da arquitetura geral do sistema.

Cada um destes microcontroladores controlam duas juntas individualmente. As juntas do robô MA2000 foram equipadas com sensores de posição e sensores de efeito hall, trabalhando em conjunto com circuitos auxiliares, para medição dos torques.

O bloco interface foi desenvolvido para permitir a integração do microcomputador com o meio ambiente, cuja função é: aquisição e monitoramento de sinais em tempo

real, supervisão e controle. A figura 4 ilustra o diagrama de blocos da interface de dados.

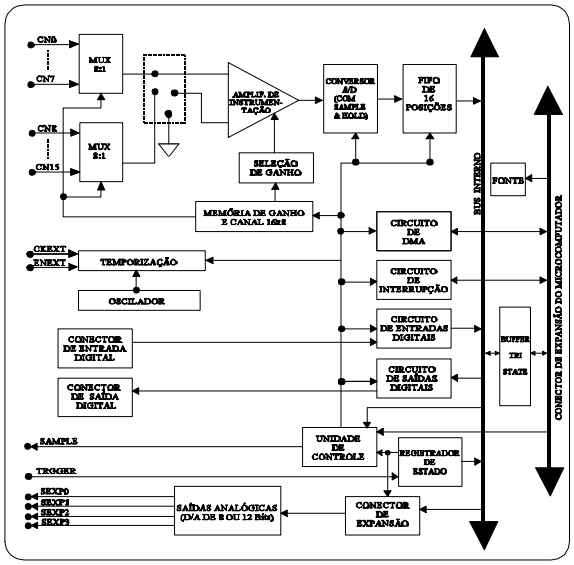

Figura 4- Diagrama de blocos da interface de dados.

O hardware foi desenvolvido de forma que três blocos idênticos pudessem controlar o sistema. Sendo assim, a figura 5 ilustra a arquitetura interna de um único chip. Observa-se que ela é baseada em registradores com o barramento de memória de dados separado do barramento de memória de programa (característica da arquitetura RISC).

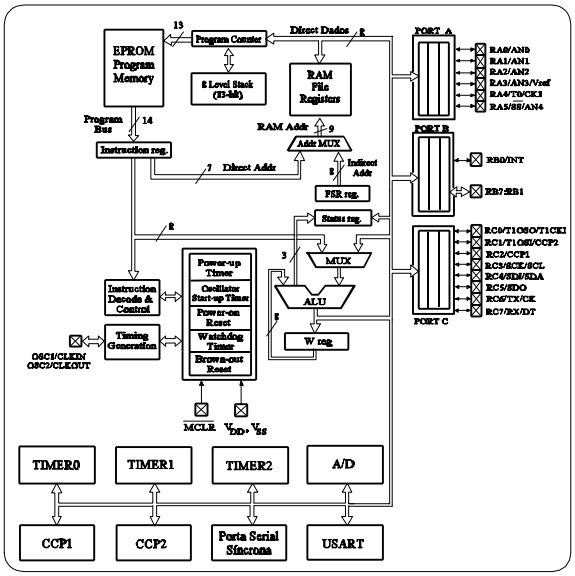

Figura 5- Diagrama de blocos do microcontrolador PIC16C73A.

Alguns processos considerados atrativos na escolha do microcontrolador PIC16C73A foram: sua habilidade de gerar dois sinais PWM, independentes do processamento da CPU, gerar interrupções pelo hardware interno de modo que um dos registradores de 16 bits possa ser usado como registrador de período programável, número de temporizadores e ou /contadores. Além do mais, enquanto uma instrução estiver sendo executada e valores estiverem sendo lidos ou escritos na memória ou em I/O pela via apropriada, outra instrução já estará sendo carregada pela via de memória de programa.

Os circuitos amplificadores, usados para acionamento das juntas, convertem os sinais PWM provenientes dos microcontroladores em tensões de saída proporcionais às respectivas razões cíclicas dos sinais PWM. Estes circuitos são implementados usando uma ponte em "H" com transistores darlington.

5. Resultados experimentais.

Usando dados reais obtidos através de medições de posição, velocidade, aceleração e de torque do robô, após o término de cada seção de treinamento da rede neural, verificou-se o tempo gasto e a taxa de caimento do erro quadrático. Em seguida, alterou-se a taxa de aprendizagem e realizou-se novo treinamento, sempre verificando o tempo gasto e a taxa de erro. Posteriormente, o mesmo procedimento foi realizado com o número de neurônios na camada intermediária e por último, combinando os dois procedimentos. Observou-se que durante o treinamento da rede neural, o ajuste do fator de amortecimento exerceu pouca influência na convergência da curva de aprendizagem.

Os resultados obtidos ao final do processo de treinamento de uma única rede neural capaz de aprender o comportamento dinâmico do robô manipulador com seis graus de liberdade, são mostrados nas figuras 6, 7, 8, 9,10 e 11.

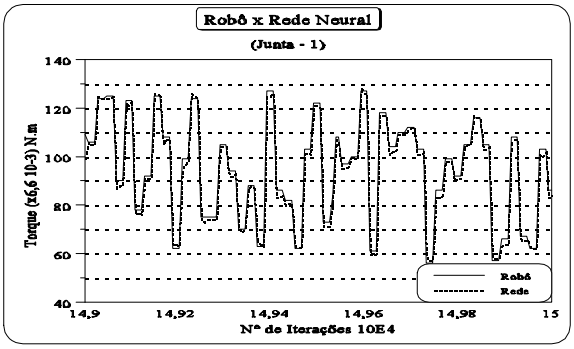

Figura 6- Resposta da Rede neural - Junta 1.

As figuras 12(a), (b) e (c) ilustram os resultados provenientes do rastreamento de trajetória obtidos da juntas 1, 2 e 3 pelo controlador neural implementado. O tempo para completar a trajetória foi estabelecido em 8,0s. A trajetória de referência foi gerada dentro do espaço de trabalho do robô MA2000, entre (0 - 270º).

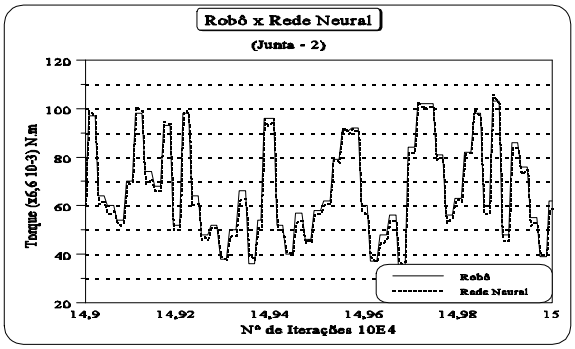

Figura 7- Resposta da rede neural - Junta 2.

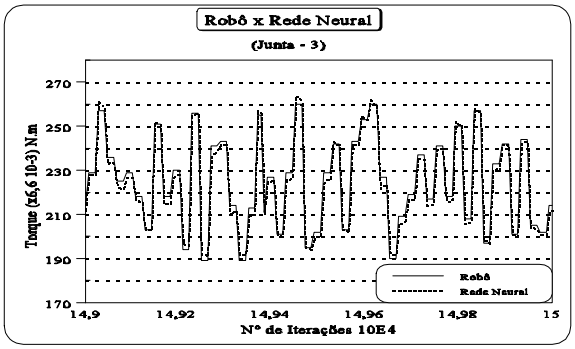

Figura 8- Resposta da Rede neural - Junta 3.

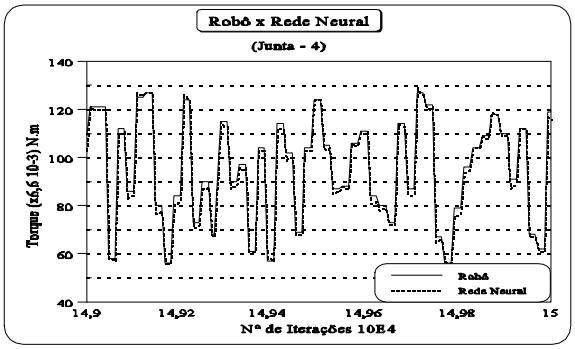

Figura 9- Resposta da rede neural - Junta 4.

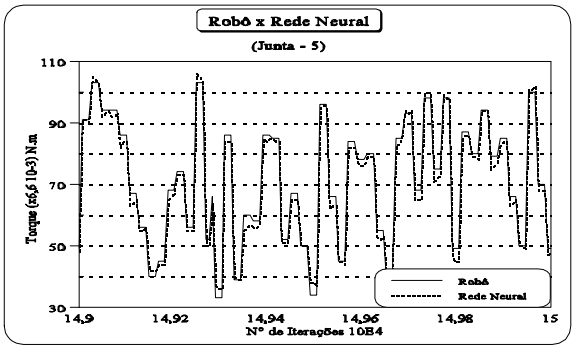

Figura 10 - Resposta da rede neural - Junta 5.

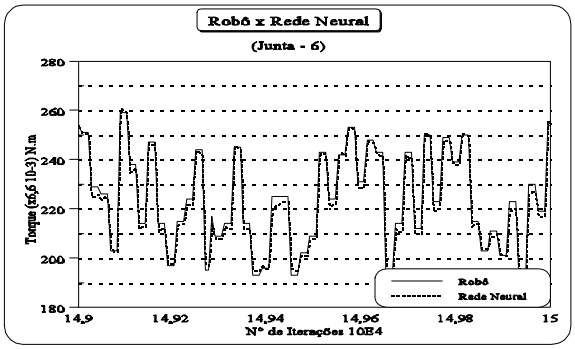

Figura 11- Resposta da rede neural - Junta 6.

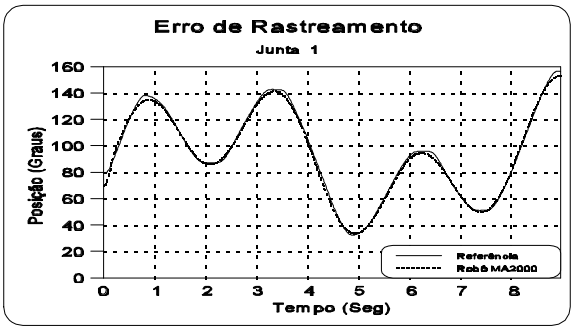

Figura 12(a)- Resposta do neurocontrolador no rastreamento de trajetória da junta 1.

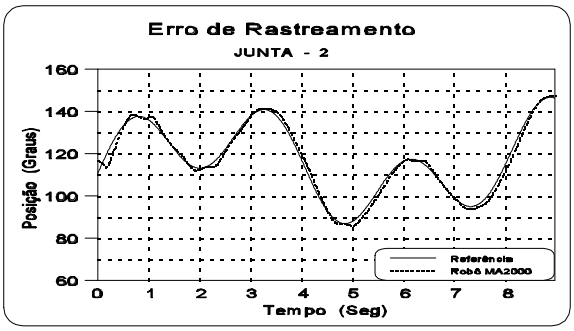

Figura 12(b)- Resposta do neurocontrolador no rastreamento de trajetória da junta 2.

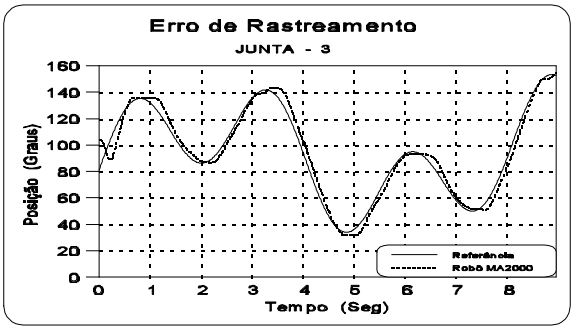

Figura 12(c)- Resposta do neurocontrolador no rastreamento de trajetória da junta 3.

6. Conclusão.

Quando não se necessita de precisão elevada no posicionamento, com uma boa sintonia, os controladores convencionais podem apresentar resultados excelentes. Este é o motivo pelo qual eles são usados na maioria do robôs comerciais existentes. Entretanto, quando precisão é um fator fundamental, mesmo em velocidades não tão altas os métodos convencionais se tornam inadequados.

O método mostrado neste trabalho apresentou resultados bastante promissores quando empregado no controle de um robô de seis juntas. A sintonia foi facilmente obtida, ou seja, os parâmetros K_D e K_P são escolhidos por tentativa e erro de forma que o erro de controle atinja assintóticamente o valor zero, bem como o seguimento de trajetórias dinâmicas. Vale lembrar que todos os termos dinâmicos, do sistema, foram considerados (à exceção da carga no efetuador). Para obtenção dos resultados aqui apresentados, o tempo necessário para treinamento da rede em uma máquina Pentium 200 Mhz foi de 298.26 segundos. Outro ponto interessante a se notar é que as respostas ao degrau das juntas, ficaram praticamente idênticas, o que leva à conclusão de que todas atingiram um grau de aprendizagem praticamente no mesmo instante. O controle pode ser implementado na prática com o mínimo de hardware e software. O estudo mostra também que é possível controlar com precisão um robô no rastreamento de trajetórias.

Referências.

[1] Craig, J. J., "*Adaptative Control of Mechanical Manipulators.*" Reading MA: Addison-Wesley, 1988.

[2] Craig, J. J., "*Introduction to Robotics: Mechanics and Control.*" Reading MA: Addison-Wesley, 1986

[3] Paul, R. P., "*Robot Manipulators: Mathematics, Programming and Control.*" Cambridge, MA: M.I.T. Pres, 1987.

[4] Liu, C. H. *"A comparison controller design and simulation for an industrial manipulator."* IEEE Trans. Ind. Electron.,vol. IE-33, nº 1, pp. 58-65, Feb, 1986.

[5] Luh, J. Y. S., Walker, M. W. & Paul, R. P. C. *"On-line computacional scheme for mechanical manipulators."* J. Dyn. Syst., Meas. Contr., vol. 102, pp. 69-76, june, 1980..

[6] Luh, J. Y. S. *"Conventional controller design for industrial robots - A tutorial."* IEEE Trans. Syst. Man. Cyber.,vol. SMC-13, nº 3, Mai, 1983.

[7] Rumelhart, D. McClelland, J., 1986. *"Parallel Distributed Processing."* Vol. 1 and 2, MIT Press, Cambridge, MA, 4th Edition.

[8] Narenda, K. S. & Parthasarathy, K., *"Identification and Control of Dynamical Systems Using Neural Networks."* IEEE Transactions on Neural Networks, Vol.1Nº1, pp.4-26, march, 1990.

[9] Psaltis, D., Sideris, A. and Yamamura, A.A., *"A Multilayered Neural Network Controller."* IEEE Control Systems Mag., vol. 8, pp. 17-21, April, 1988.

[10] Kawato, M., Furukawa, K. and Suzuki, R., *"A hierarchical neural network model for control and learning of voluntary movement."* Biol. Cybern., Vol. 57, pp. 169-185, 1987.

[11] Alsina, P. J. & Gehlot, N. S., Setembro 1994. *"Identificação Modular de Parâmetros Dinâmicos de Manipuladores Robóticos."* Anais do 10º CBA, Rio de Janeiro, Brasil, Vol. 2, pp. 1004-1009.

[12] Oliveira, S. R. J. & Teixeira, E. P. *"Identificação da Dinâmica Inversa de Sistemas Não-Lineares Através de Redes Neurais Artificiais."* I SBAI - UNESP, Rio Claro, setembro, 1993.

[13] Corrêa, G. R., Teixeira, E. P. e Oliveira S. R. J., "*implementacion de un control neural de multiples propositos.*" Revista internacional Informacion Tecnologica - CIT, Vol. 9, Nº 03, pp. 81-87, 1998.

[14] Tanomaru J. & Omatu, S., *"Process Control by On-Line Trained Neural Controllers."* IEEE Transactions on Industrial Eletronics, Vol. 39, Nº.6, Dec. 1992.

[15] Microchip Techonology Incorporated, *"PIC16C7X Data Sheet, DS30390E*", USA, 1997.