FÁBIO AUGUSTO PIRES BORGES

UM ESTUDO DE CONTROLADORES ROBÓTICOS UTILIZANDO REDES NEURAIS

FLORIANÓPOLIS
1999

UNIVERSIDADE FEDERAL DE SANTA CATARINA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

UM ESTUDO DE CONTROLADORES ROBÓTICOS UTILIZANDO REDES NEURAIS

Dissertação submetida à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Mestre em Engenharia Elétrica

FÁBIO AUGUSTO PIRES BORGES

Florianópolis, Junho de 1999

Um Estudo de Controladores Robóticos Utilizando Redes Neurais

Fábio Augusto Pires Borges

Esta dissertação foi julgada adequada para a obtenção do título **de Mestre em Engenharia** na especialidade **Engenharia Elétrica**, área de concentração **Controle, Automação e Informática Industrial**, e aprovada em sua forma final pelo curso de Pós-Graduação.

Florianópolis, 25 de Junho de 1999.

Prof. Marcelo Ricardo Stemmer, Dr.
Orientador

Prof. Edson Roberto de Pieri, Dr.
Co-orientador

Prof. Ildemar Cassana Decker, D. Sc.
Coordenador do curso de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Santa Catarina

Banca Examinadora

Prof. Marcelo Ricardo Stemmer, Dr.
Orientador

Prof. Edson Roberto de Pieri, Dr.
Co-orientador

Prof Werner Kraus Junior, Dr.

Prof. Jorge Muniz Barreto, Dr.

Prof. Raul Guenther, Dr.

Dedico este trabalho em memória de meu Pai, Dr Oscar Francisco Valério Borges.

Resumo da Dissertação apresentada à UFSC como parte dos requisitos necessários para a obtenção do grau de Mestre em Engenharia Elétrica

UM ESTUDO DE CONTROLADORES ROBÓTICOS UTILIZANDO REDES NEURAIS

FÁBIO AUGUSTO PIRES BORGES

Junho/99

Orientador: Prof. Marcelo Ricardo Stemmer, Dr.

Área de Concentração: Controle, Automação e Informática Industrial.

Palavras chave: seguimento de trajetória, espaço de trabalho, espaço de juntas, manipulador robótico, manipulador planar, tempo real.

Número de páginas: 95

Este trabalho trata da utilização das Redes Neurais Artificiais (RNA) no projeto de controladores voltados ao controle de manipuladores robóticos. As técnicas clássicas de controle de manipuladores robóticos mais utilizadas, como os chamados controladores PID junta por junta, apresentam limitado desempenho quando usadas para controlar o manipulador dentro de um seguimento de trajetória. Melhores resultados são obtidos quando utiliza-se técnicas baseadas na dinâmica inversa ou na passividade. Ambas as técnicas necessitam entretanto do modelo matemático da dinâmica inversa do manipulador. As redes neurais tem a capacidade de representar modelos matemáticos através de um aprendizado baseado em exemplos e também capacidade de adaptação em tempo real. Uma rede neural do tipo feedforward multicamadas, utilizando como algoritmo de treinamento o algoritmo Quickpropagation, uma versão mais rápida de Backpropagation, é utilizada em conjunto com as técnicas da dinâmica inversa e da Passividade na confecção de controladores voltados ao controle de seguimento de trajetória. São realizadas simulações de todos os casos de controle estudados.

Abstract of Dissertation presented to UFSC as a partial fulfillment of the requirements for the degree of Master in Eletrical Engineering.

A STUDY OF ROBOTIC CONTROLLERS USING NEURAL NETWORKS

FÁBIO AUGUSTO PIRES BORGES

June/99

Advisor: Prof. Marcelo Ricardo Stemmer, Dr.

Area of Concentration: Control, Automation and Industrial computing.

Keywords: tracking trajectory, joint space, work space, robotic manipulator, planar manipulator, real time.

Number of Pages: 95

This work addresses the usage of the Artificial Neural Networks in the design of controller's for robot manipulators. The most frequently used classical techniques for control of robot manipulators, like local independent joint PID controllers, have a limited performace when controlling the manipulador in trajectory tracking tasks. Better results are obtained when techniques based on inverse dynamic or passivity approaches are used. However, both inverse dynamic and passivity need the mathematical model of the robotic manipulator. Neural Networks have the capability to represent mathematical models through learning based on examples and the ability to real time adaptation. A multilayer feedforward Neural Net using the Quickpropagation learning algorithm, a faster version of Backpropagation, is used in conjunction with the inverse dynamic and passivity techniques for controller's design applied to the trajectory tracking problem. Simulations for all the control cases are performed.

ÍNDICE ANALÍTICO

ÍNDICE DE FIGURAS

1- INTRODUÇÃO

1.1 - ROBÓTICA

A palavra robótica nos sugere à primeira vista o tratamento de robôs sofisticados, popularmente conhecidos pelos filmes e artigos de ficção científica, que dão um aspecto romântico e ao mesmo tempo visionário ao assunto. Na verdade a robótica, em síntese, realmente trata do projeto, análise e comportamento de robôs, porém a maioria dos robôs estudados até então tratam-se de robôs industriais, que, de uma maneira mais simples, constituem valiosas ferramentas de produção industrial. A palavra *robot* , em português robô, teve sua origem de uma obra literária dos anos 20 escrita pelo taumaturgo tcheco Karel Capek chamada *Rossum's Universal Robots*. A palavra *robota* em tcheco significa apenas trabalho [33], [20] . Desde então o termo foi aplicado à uma grande variedade de dispositivos mecânicos tais como teleoperadores, veículos aquáticos, etc. Virtualmente, qualquer coisa que opere com algum grau de autonomia, usualmente sob controle computacional, em algum ponto foi classificado como um robô [33] . Neste trabalho o enfoque é dirigido aos chamados robôs industriais, que segundo o *Robot Institute of America* são definidos da seguinte maneira: " um robô industrial é um manipulador reprogramável e multifuncional, projetado para mover materiais, peças, ferramentas ou dispositivos em movimentos variáveis programados para a realização de uma infinidade de tarefas" [33], [10] . Dentro desta definição, os robôs industriais são usualmente chamados de robôs manipuladores ou simplesmente manipuladores. Tratam-se portanto de máquinas automáticas capazes de realizar movimentos, seja transportando objetos de interesse industrial, aplicando forças ou torques em pontos determinados ou mesmo executando tarefas minuciosas de posicionamento, sob uma determinada trajetória dependente da tarefa a ser cumprida.

1.2 - O PROBLEMA DO CONTROLE DE ROBÔS

Controlar um robô manipulador tem dois significados distintos, de acordo com a natureza da tarefa que deseja-se que o mesmo desempenhe.

Chama-se controle ponto a ponto ou **controle de posição** quando a tarefa de interesse a ser executada pelo manipulador é simplesmente levar o efetuador de um

ponto de partida até um ponto de chegada no espaço de trabalho, com o mínimo de erro possível em relação a este ponto de chegada.

Chama-se seguimento de trajetória ou **controle de trajetória** quando a tarefa de interesse a ser executada pelo manipulador é fazer com que o efetuador siga uma trajetória previamente definida no espaço de trabalho, com o mínimo de erro possível em relação a cada ponto desta trajetória.

O controle de posição constitui um processo mais simples, típico dos casos onde o interesse é que o robô simplesmente pegue uma peça que encontra-se em uma esteira, por exemplo, e largue esta peça em algum lugar, como um tanque contendo algum líquido de limpeza ou um forno, etc. Portanto não importa o transitório do movimento, mas sim a posição final que deve ser a mais precisa possível de acordo com a tarefa. Para este caso o controle não é tão crítico, pois basta evitar o sobressinal e o erro em regime e a tarefa terá execução satisfatória.

O controle de trajetória é mais elaborado e em alguns casos bastante complexo. Exemplos deste procedimento são muito encontrados na industria em tarefas como pintura ou usinagem de peças, soldagem, etc. Não só o erro em regime mas também o transitório é fundamental. Os controladores mais simples podem não apresentar comportamento satisfatório, sendo por vezes necessário recorrer a técnicas mais complexas que visem contornar os erros de seguimento de trajetória, que aumentam em proporção à velocidade de execução da tarefa.

Neste trabalho o problema do controle de trajetória será abordado e algumas soluções típicas serão apresentadas, bem como alternativas utilizando Redes Neurais Artificiais.

1.3 - REDES NEURAIS ARTIFICIAIS

As redes neurais artificiais, também chamadas de redes neuronais, constituem modelos de processamento distribuído paralelo. Elas consistem de um conjunto de processadores simples, chamados de neurônios, conectados entre si que operam em paralelo e possuem a capacidade de mudar o comportamento através de treinamento dinâmico [39].

A aplicação das redes neurais neste trabalho deve-se ao fato das mesmas serem capazes de, através de uma aprendizagem baseada em exemplos de padrões conhecidos, emular o comportamento de processos, neste caso particular, da dinâmica inversa de um manipulador robótico.

Neste trabalho opta-se pela utilização das redes neurais *feedforward* multicamadas devido a sua larga utilização e comprovada eficiência em aplicações voltadas para sistemas de controle. O algoritmo utilizado para treinamento é o algoritmo Quickpropagation e as razões de sua escolha serão descritas posteriormente no capítulo 4.

1.4 - OBJETIVOS DO TRABALHO

Este trabalho tem como objetivo demonstrar a utilização das redes neurais artificiais no controle de manipuladores robóticos através do projeto de um controlador neural adaptativo que controla um manipulador simples de dois graus de liberdade no seguimento de uma trajetória. Serão feitas comparações de desempenho entre as técnicas utilizadas e a partir daí estabelecidas algumas conclusões. Simulações computacionais ilustram os resultados obtidos.

O trabalho esta dividido basicamente em seis capítulos. No capítulo 1 tem-se uma introdução geral do assunto, abordando os principais tópicos a serem estudados e o objetivo do trabalho.

No capítulo 2 aborda-se uma introdução à robótica e a modelagem de robôs manipuladores. Apresenta-se o modelo dinâmico de um manipulador de dois graus de liberdade.

O capítulo 3 apresenta uma revisão bibliográfica de alguns dos controladores mais utilizados em robótica.

O capítulo 4 aborda as redes neurais artificiais, principalmente o algoritmo backpropagation, seus problemas e as soluções para os mesmos adotadas neste trabalho, entre elas o algoritmo quickpropagation, uma versão modificada do algoritmo backpropagation.

No capítulo 5 são apresentados os controladores neurais utilizados neste trabalho e suas formas de operação.

No capítulo 6 são apresentadas simulações que ilustram o desempenho das técnicas de controle utilizadas.

O capítulo 7 encerra o trabalho apresentando as principais conclusões obtidas com a realização do mesmo.

2 - ASPECTOS GERAIS DA ROBÓTICA

2.1 - INTRODUÇÃO

2.1.1 - Aspectos Gerais dos Manipuladores Robóticos

A figura 2.1 ilustra um manipulador robótico industrial típico. Tal mecanismo é composto de **elos** (*links*) interligados por **juntas**, formando uma cadeia cinemática aberta. As juntas são tipicamente rotatórias (de revolução) ou lineares (prismáticas). Uma junta de revolução é como uma dobradiça e permite relativa rotação entre dois elos. Uma junta prismática permite um movimento linear entre dois elos. As variáveis de junta são simbolizadas por $\mathbf{q}_i$ para uma junta de revolução e $\mathbf{d}_i$ para uma junta prismática, representando o afastamento relativo entre dois elos. Após o último elo de um manipulador existe um elemento importante chamado **efetuador final**.. Os efetuadores mais simples possuem o movimento de abre e fecha e tem o formato de uma garra. O tipo de efetuador final depende da tarefa a ser desempenhada pelo manipulador, podendo ser também uma pistola de pintura, um maçarico de solda, etc.

O **espaço de trabalho** de um manipulador robótico é o volume total capaz de ser alcançado pelo seu efetuador final. Este espaço de trabalho é tipicamente referenciado por coordenadas cartesianas à partir de um eixo situado na base do robô. O espaço de trabalho de um manipulador é limitado pela geometria do mesmo e também por restrições mecânicas de suas juntas.

O número de juntas de um manipulador robótico determina os **graus de liberdade (DOF)** do mesmo. Um manipulador robótico deve ter ao menos seis graus de liberdade para que seja capaz de alcançar todas as posições e orientações possíveis em um espaço de trabalho [33].

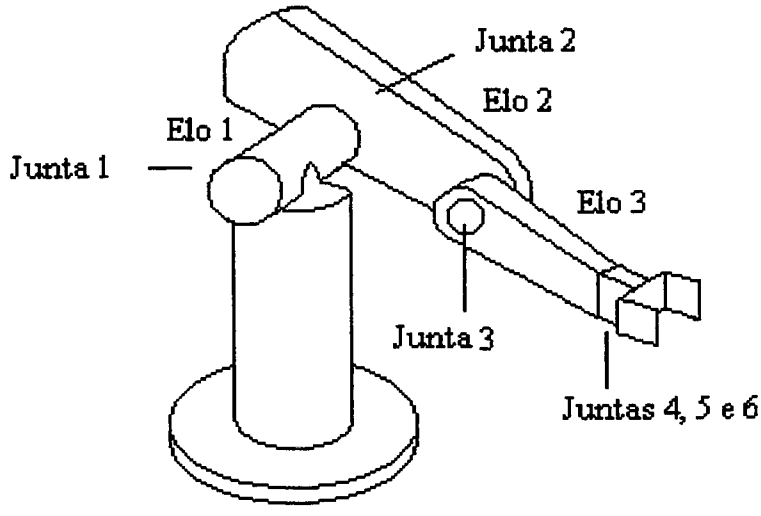


FIGURA 2.1 - Manipulador Robótico Típico.

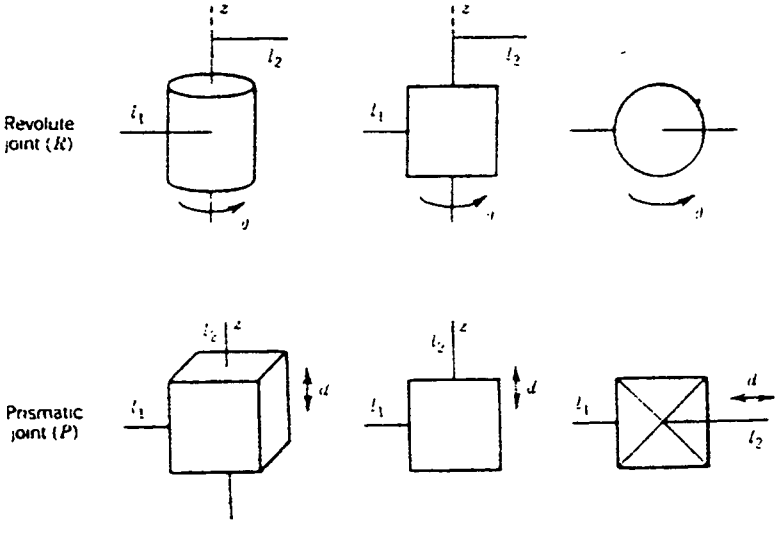


FIGURA 2.2 - Juntas Prismáticas e de Revolução.

Além da parte mecânica, um manipulador possui também outras partes fundamentais que juntas constituem o sistema robótico. Macroscopicamente os manipuladores podem ser divididos em três componentes : mecanismo, acionamento e sistema de controle [10] .

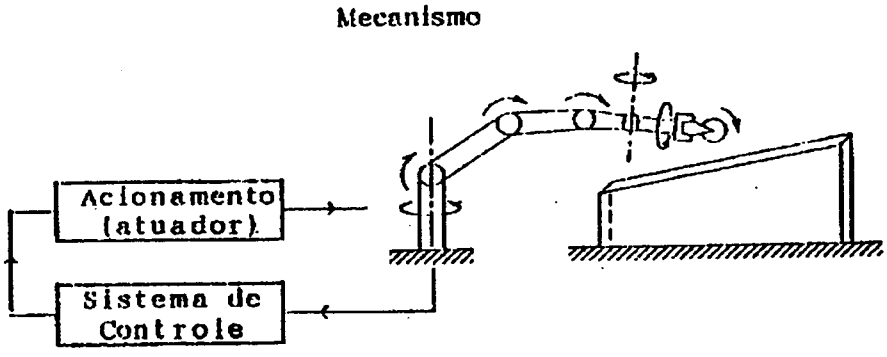


FIGURA 2.3 - Componentes Macroscópicos de um Manipulador.

O mecanismo, parte mecânica que executa os movimentos, pode ser dividido em outras três partes: braço, composto por elos e juntas, punho e efetuador.

O acionamento ou atuador é responsável pela movimentação física do mecanismo. Através dele aplica-se a força ou torque em cada junta. Existem diferentes tipos de acionadores utilizados em robótica, dentre eles cita-se os acionadores hidráulicos, os acionadores pneumáticos e os acionadores elétricos.

O sistema de controle de um manipulador robótico é responsável por fazer que o mesmo execute um movimento desejado. Este sistema é composto por sensores que fornecem informação acerca da tarefa que esta sendo executada, e pelo algoritmo ou lei de controle, que gera o sinal de comando em função das informações dos sensores e da tarefa desejada.

Os controladores usuais usados em robótica serão descritos no capítulo 3. De uma forma geral, o sistema de controle de um robô manipulador é o tema central deste trabalho.

2.1.2 - Cinemática

Considere o manipulador planar de dois elos da figura 2.4 e suponha que deseja-se posicionar algum objeto em seu efetuador. Como trata-se de duas juntas de revolução, as variáveis de junta são consequentemente os ângulos q_1 e q_2 . A equação (2.1) é chamada de **cinemática direta** do manipulador robótico, pois fornece as coordenadas (x, y) do efetuador no espaço de trabalho baseado nos valores conhecidos das variáveis de junta do mesmo [33, 2, 20, 5].

$$x(q_1,q_2) = l_1 \cos q_1 + l_2 \cos(q_1 + q_2)$$
$$y(q_1,q_2) = l_1 \operatorname{sen} q_1 + l_2 \operatorname{sen}(q_1 + q_2) \qquad (2.1)$$

Chama-se **cinemática inversa** do robô manipulador o contrário, ou seja, deseja-se obter os valores das variáveis de junta que posicionam o efetuador em determinada posição definida pelas coordenadas (x,y).

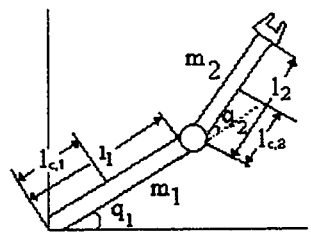


FIGURA 2.4 - Um manipulador Planar de Dois graus de Liberdade.

Define-se à partir dai o chamado **espaço das juntas**, formado pelas variáveis de juntas, que é o domínio da função cinemática direta cuja imagem é o próprio espaço de trabalho do manipulador.

A maneira usual de obter-se a matriz de transformação T da cinemática direta é através das transformações homogêneas [33, 2, 20, 5], que nada mais são que sucessivas mudanças de base utilizando a teoria de álgebra linear. Neste trabalho admite-se a simplificação de utilizar-se diretamente o espaço de juntas, ou seja, supõe-se conhecidas as juntas que resultariam nas coordenadas desejadas para o efetuador.

2.1.3 - Dinâmica

A chamada dinâmica do robô consiste em definir através de uma equação matemática as propriedades fisicas referentes ao movimento do manipulador tais como inércias, forças gravitacionais, etc.

Chama-se de modelo do robô o conjunto de equações diferenciais, chamadas equações de movimento, que governam a resposta dinâmica do manipulador às forças e torques aplicadas pelos atuadores. São basicamente uma descrição da relação existente entre os torques de entrada nas juntas e o movimento de saída, isto é o movimento da cadeia cinemática formada por elos e juntas.

Formuladas as hipóteses básicas que definem um modelo, as equações do movimento são estabelecidas por métodos usuais da mecânica, como o de Newton-Euler, baseado na interpretação direta da segunda lei de Newton, que descreve o sistema em termos das suas forças e momentos, e o de Lagrange, que descreve o sistema em termos de trabalho e energia utilizando coordenadas generalizadas para representação do mesmo.

Exemplos da obtenção do modelo de manipuladores robóticos são fartamente encontrados na bibliografia [2, 33 e 20, por exemplo]. Em resumo, a equação final obtida é a seguinte:

$$\tau = D(q)\ddot{q} + C(q,\dot{q})\dot{q} + F\dot{q} + G(q) \qquad (2.2)$$

Pode-se descrever os termos da equação como segue:

$D(q)$: **Matriz de inércia do manipulador** - Trata-se de uma matriz simétrica positiva definida nxn, sendo n o número de juntas do manipulador.

$C(q,\dot{q})$: **Matriz dos torques centrífugos e de Coriólis** - Matriz nxn.

$G(q)$: **Vetor dos torques gravitacionais** - Vetor nx1.

F : **Matriz dos parâmetros de atrito** - Matriz diagonal nxn.

τ : **Vetor dos torques aplicados nas juntas** - Vetor nx1.

2.2 - MODELAGEM DE UM MANIPULADOR PLANAR

Os Manipuladores robóticos mais utilizados na Indústria geralmente possuem movimentos tridimensionais e mais de três graus de liberdade. Tais atributos tornam os seus modelos matemáticos complexos, necessitando de pesada carga computacional para proceder a uma simulação de controle. Levando em conta este fator, procurou-se neste trabalho utilizar um modelo mais simples, mas que pode igualmente ilustrar os aspectos do controle de robôs, tendo como principal vantagem a facilidade de simulação computacional. Procurou-se trabalhar portanto com os chamados manipuladores planares.

Um manipulador planar de **n** graus de liberdade move-se somente no plano vertical **x-y**, como mostrado na figura 2.5, sendo que a gravidade atua na direção **y**. As coordenadas **x** da tarefa são as posições do efetuador no plano **x-y**.

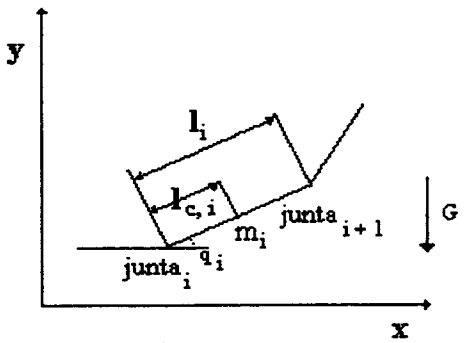


FIGURA 2.5 - Estrutura de um Manipulador Planar de n Graus de Liberdade.

Com respeito as **n** coordenadas de juntas **q** e as **m** coordenadas da tarefa **x** a cinemática do manipulador pode ser descrita com as seguintes equações [41]:

$$\begin{aligned} x &= p(q) \\ \dot{x} &= J(q)\dot{q} \\ \ddot{x} &= J(q)\ddot{q} + \dot{J}(q,\dot{q})\dot{q} \end{aligned} \tag{2.3}$$

Onde **p** é uma função vetorial de dimensão **m** representando a cinemática direta, **J** é a matriz jacobiana [33, 2, 20] e $\dot{J}$ é a derivada da Matriz Jacobiana em relação as coordenadas de junta. A Matriz Jacobiana tem dimensão **mxn**. Para manipuladores planares (**m**=2) com juntas de revolução, a posição **x** do efetuador $x = [x_1, y_1]^T$ pode ser expressada pelas seguintes equações recursivas [41]:

$$\varphi_i = \varphi_{i-1} + q_i, \qquad i = 1, \ldots, n \tag{2.4}$$

$$\begin{cases} x_i = x_{i+1} + l_i \cos(\varphi_i), \\ y_i = y_{i+1} + l_i \operatorname{sen}(\varphi_i) \end{cases} \qquad i = 1, \ldots, n \tag{2.5}$$

$$\varphi_0 = 0 \qquad x_{n+1} = y_{n+1} = 0$$

Onde l_i é o comprimento do i-ésimo elo. De (2.5) tem-se que:

$$\frac{\partial x_i}{\partial q_j} = -y_k, \quad \frac{\partial y_i}{\partial q_j} = x_k, \qquad k = \max(i, j) \tag{2.6}$$

No caso de manipuladores planares, o Jacobiano J é uma matriz 2 x **n** :

$$J = \begin{bmatrix} \frac{\partial x_1}{\partial q_1}, \ldots, \frac{\partial x_1}{\partial q_n} \\ \frac{\partial y_1}{\partial q_1}, \ldots, \frac{\partial y_1}{\partial q_n} \end{bmatrix} \tag{2.7}$$

Portanto, os componentes de J podem ser especificados sem qualquer cálculo adicional :

$$J = \begin{bmatrix} -y_1, \ldots, -y_n \\ x_1, \ldots, x_n \end{bmatrix} \tag{2.8}$$

Para o modelo completo é necessário obter-se $\dot{J}$ através da equação (2.9) :

$$\dot{J} = \sum_{k=1}^{n} \left(\frac{\partial J}{\partial q_k} \dot{q}_k \right) \tag{2.9}$$

Diferenciando (2.5) com respeito a **q** tem-se:

$$\begin{aligned} \frac{\partial^2 x_i}{\partial q_j \partial q_k} &= \begin{cases} -x_r, & j \le i \text{ e } k \le i \\ 0, & j > i \text{ ou } k > i \end{cases} \\ \frac{\partial^2 y_i}{\partial q_j \partial q_k} &= \begin{cases} -y_r, & j \le i \text{ e } k \le i \\ 0, & j > i \text{ ou } k > i \end{cases} \end{aligned} \tag{2.10}$$

Onde r = max (j, k). Substituindo (2.10) em (2.9) tem-se a expressão em (2.11).

$$\dot{J} = \begin{bmatrix} \dot{q}^T \begin{bmatrix} -x_1 & -x_2 & \ldots & -x_n \\ -x_2 & -x_2 & \ldots & -x_n \\ \ldots & \ldots & \ldots & \ldots \\ -x_n & -x_n & -x_n & -x_n \end{bmatrix} \\ \dot{q}^T \begin{bmatrix} -y_1 & -y_2 & \ldots & -y_n \\ -y_2 & -y_2 & \ldots & -y_n \\ \ldots & \ldots & \ldots & \ldots \\ -y_n & -y_n & -y_n & -y_n \end{bmatrix} \end{bmatrix} \tag{2.11}$$

Portanto, os únicos elementos de um modelo cinemático de um manipulador planar a serem calculados são x_i e y_i. Todas as outras partes podem ser expressadas em função de x_i e y_i.

Antes de determinar os componentes do modelo dinâmico, é necessário obter expressões para a posição do centro de massa para todos os segmentos e suas

correspondentes matrizes Jacobianas. Utilizando os resultados anteriores tem-se que a posição do centro de massa do i-ésimo elo é definida pela equação (2.12).

$$x_{c,i} = \left[x_1 - x_i + l_{c,i}\cos(\varphi_i), y_1 - y_i + l_{c,i}\,\mathrm{sen}(\varphi_i)\right]^T \tag{2.12}$$

Diferenciando (2.12) com respeito a **q** tem-se a equação (2.13).

$$\begin{aligned} \frac{\partial x_{c,i}}{\partial q_j} &= \begin{cases} -y_j + y_i - l_{c,i}\,\mathrm{sen}(\varphi_i) & j \le i \\ 0 & j > i \end{cases} \\ \frac{\partial y_{c,i}}{\partial q_j} &= \begin{cases} -x_j + x_i - l_{c,i}\cos(\varphi_i) & j \le i \\ 0 & j > i \end{cases} \end{aligned} \tag{2.13}$$

As Matrizes Jacobianas são divididas em duas partes, conforme (2.14), onde J_L e J_A são partes integrantes da Matriz Jacobiana J ambas associadas respectivamente as velocidades linear e angular dos centros de massa dos elos.

$$J = \begin{bmatrix} J_L \\ J_A \end{bmatrix} \tag{2.14}$$

A Matriz Jacobiana $J_L^{(i)}$, associada ao centro de massa do i-ésimo elo é definida por (2.15), sendo obtida de maneira simples pela substituição de (2.13) em (2.15). À seguir, as derivadas de $J_L^{(i)}$ com respeito a q são calculadas em (2.16).

$$J_L^{(i)} = \begin{bmatrix} \frac{\partial x_{c,i}}{\partial q_1} & \cdots & \frac{\partial x_{c,i}}{\partial q_n} \\ \frac{\partial y_{c,i}}{\partial q_1} & \cdots & \frac{\partial y_{c,i}}{\partial q_n} \end{bmatrix} \tag{2.15}$$

$$\frac{\partial J_L^{(i)}}{\partial q_k} = \begin{bmatrix} \frac{\partial^2 x_{c,i}}{\partial q_1 \partial q_k} & \cdots & \frac{\partial^2 x_{c,i}}{\partial q_n \partial q_k} \\ \frac{\partial^2 y_{c,i}}{\partial q_1 \partial q_k} & \cdots & \frac{\partial^2 y_{c,i}}{\partial q_1 \partial q_k} \end{bmatrix} \tag{2.16}$$

Utilizando a relação (2.17), introduz-se em (2.18) uma nova notação.

$$\begin{aligned} \frac{\partial^2 x_i}{\partial q_j \partial q_k} &= \begin{cases} -x_r + x_i - l_{c,i}\cos(\varphi_i), & j \le i \text{ e } k \le i \\ 0 & j > i \text{ e } k > i \end{cases} \\ \frac{\partial^2 y_i}{\partial q_j \partial q_k} &= \begin{cases} -y_r + y_i - l_{c,i}\,\mathrm{sen}(\varphi_i), & j \le i \text{ e } k \le i \\ 0 & j > i \text{ e } k > i \end{cases} \end{aligned} \tag{2.17}$$

$r = \max(j, k)$.

$$\psi_k^{(i)} = \left(\frac{\partial J_L^{(i)}}{\partial q_k}\right)^T J_L^{(i)} + \left(J_L^{(i)}\right)^T \frac{\partial J_L^{(i)}}{\partial q_k} \tag{2.18}$$

Para a obtenção do modelo dinâmico utiliza-se a formulação de Lagrange [2, 33 e 20, por exemplo], onde a equação em (2.19) representa e equação do movimento.

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}}\right) - \frac{\partial T}{\partial q} = \tau \tag{2.19}$$

A equação final obtida é a equação (2.2) e sua obtenção detalhada pode ser encontrada em várias fontes bibliográficas [2, 33 e 20, por exemplo] .

A Matriz H, chamada matriz das inércias do robô manipulador, que faz parte de (2.2), pode ser obtida através da expressão (2.20).

$$H = \sum_{i=1}^{n}\left(m_i J_L^{(i)T} J_L^{(i)} + J_A^{(i)T} I_i J_A^{(i)}\right) \tag{2.20}$$

Como o manipulador é planar, para fins de simplificação, o termo $J_A^{(i)T} I_i J_A^{(i)}$ pode ser representado conforme a expressão (2.21).

$$J_A^{(i)T} I_i J_A^{(i)} = I_i \begin{bmatrix} 1_{ixi} & 0 \\ 0 & 0 \end{bmatrix}_{nxn} \tag{2.21}$$

Onde I_i é o momento de inércia do elo i.

As forças Centrífugas e de Coriólis em (2.2) podem se expressadas por (2.22), ou, de uma maneira mais específica, por (2.23).

$$h_i = \sum_{j=1}^{n}\sum_{k=1}^{n} h_{ijk}\dot{q}_j\dot{q}_k, \qquad h_{ijk} = \frac{\partial H_{ij}}{\partial q_k} - \frac{1}{2}\frac{\partial H_{jk}}{\partial q_i} \tag{2.22}$$

$$h = \sum_{i=1}^{n} m_i \left(\left(\sum_{j=1}^{n} \psi_j^{(i)} \dot{q}_j\right)\dot{q} - \frac{1}{2}\begin{bmatrix} \dot{q}^T\psi_1^{(i)} \\ \dot{q}^T\psi_2^{(i)} \\ \cdots \\ \dot{q}^T\psi_n^{(i)} \end{bmatrix}\dot{q} \right) \tag{2.23}$$

As componentes do vetor gravitacional **g** são em geral calculadas pela expressão (2.24), mas no caso de manipuladores planares **g** pode ser calculada

recursivamente, isto é, a força da gravidade no i-ésimo elo i é igual a força da gravidade no elo i+1 mais a contribuição do elo i, conforme a expressão (2.25).

$$g_i = \sum_{j=1}^{n} m_j \vec{g}^T J_{Li}^{(j)} \tag{2.24}$$

$$\begin{aligned} g_n &= 9.81 m_n l_{c,n} \cos(\varphi_n) \qquad i = n-1,\ldots,1 \\ g_i &= g_{i+1} + 9.81\left(m_i l_{c,i} \cos(\varphi_i) + \sum_{k=i}^{n} m_k l_i \cos(\varphi_i) \right) \end{aligned} \tag{2.25}$$

2.3 - MODELO DO MANIPULADOR PLANAR DE DOIS GRAUS DE LIBERDADE

Considerando o Manipulador da figura 2.6, deseja-se obter a equação do movimento para tal mecanismo.

Considera-se m_1 como sendo a massa do elo 1, m_2 a massa do elo 2 e m_3 a massa do conjunto efetuador + carga, na extremidade do elo 2. l_{c1} e l_{c2} são as distâncias do eixo até os respectivos centros de massa dos elos 1 e 2.

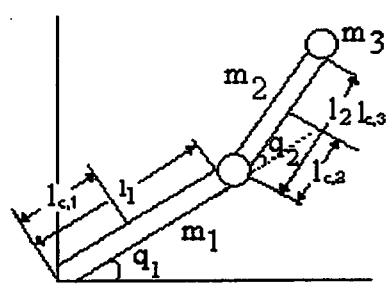


FIGURA 2.6 - Manipulador Planar de Dois graus de Liberdade.

Considera-se o efetuador + carga como sendo uma terceira junta, cujo comprimento do elo associado a mesma é zero. Portanto essa junta não acrescenta nenhum grau de liberdade adicional ao modelo, sendo **n** = 2, porém o centro de massa deste elo fictício possui uma Matriz Jacobiana associada a si que é importante ao modelo.

De (2.12) obtêm-se as coordenadas **x** dos centros de massa 1, 2 e 3.

$$x_{c,1} = \left[l_{c,1}\cos(q_1) \quad l_{c,1}\,\mathrm{sen}(q_1) \right]^T$$

$$x_{c,2} = \left[l_1\cos(q_1) + l_{c,2}\cos(q_1+q_2) \quad l_1\,\mathrm{sen}(q_1) + l_{c,2}\,\mathrm{sen}(q_1+q_2) \right]^T$$

$$x_{c,3} = \left[l_1\cos(q_1) + l_2\cos(q_1+q_2) \quad l_1\,\mathrm{sen}(q_1) + l_2\,\mathrm{sen}(q_1+q_2) \right]^T$$

Aplicando-se à (2.13) e substituindo-se em (2.15) obtêm-se as Matrizes Jacobianas das velocidades lineares associadas aos centros de massa 1, 2 e 3.

$$J_L^{(1)} = \begin{bmatrix} -l_{c,1}\,\mathrm{sen}(q_1) & 0 \\ l_{c,1}\cos(q_2) & 0 \end{bmatrix}$$

$$J_L^{(2)} = \begin{bmatrix} -l_1\,\mathrm{sen}(q_1) - l_{c,2}\,\mathrm{sen}(q_1+q_2) & -l_{c,2}\,\mathrm{sen}(q_1+q_2) \\ l_1\cos(q_1) + l_{c,2}\cos(q_1+q_2) & l_{c,2}\cos(q_1+q_2) \end{bmatrix}$$

$$J_L^{(3)} = \begin{bmatrix} -l_1\,\mathrm{sen}(q_1) - l_2\,\mathrm{sen}(q_1+q_2) & -l_2\,\mathrm{sen}(q_1+q_2) \\ l_1\cos(q_1) + l_2\cos(q_1+q_2) & l_2\cos(q_1+q_2) \end{bmatrix}$$

Aplicando-se os resultados à (2.20) obtêm-se a Matriz de Inércia H do manipulador.

$$H(q) = \begin{bmatrix} H_{11} & H_{12} \\ H_{12} & H_{22} \end{bmatrix}$$

$$H_{11} = m_1 l_{c,1}^{\,2} + m_2(l_1^{\,2} + l_{c,2}^{\,2}) + I_1 + I_2 + I_3 + 2l_1 l_{c,2} m_2\cos(q_2) + \left(l_1^{\,2} + l_2^{\,2} + 2l_1 l_2\cos(q_2)\right)m_3$$

$$H_{12} = I_2 + m_2 l_{c,2}^{\,2} + I_3 + l_1 l_{c,2} m_2\cos(q_2) + \left(l_2^{\,2} + l_1 l_2\cos(q_2)\right)m_3$$

$$H_{22} = m_2 l_{c,2}^{\,2} + I_2 + I_3 + l_2^{\,2} m_3$$

Aplicando-se os resultados obtidos à (2.22) ou (2.23) obtêm-se a Matriz das Forças Centrífugas e de Coriólis.

$$C(q,\dot{q}) = \mathrm{sen}(q_2)\left(l_1 l_{c,2} m_2 + l_1 l_2 m_3\right)\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$$

$$C_{11} = -\dot{q}_2 \;\; ; \;\; C_{12} = -(\dot{q}_1 + \dot{q}_2) \;\; ; \;\; C_{21} = \dot{q}_1 \;\; ; \;\; C_{22} = 0$$

Através de (2.24) ou (2.25) obtêm-se o vetor dos torques gravitacionais.

$$G = \begin{bmatrix} G_1 \\ G_2 \end{bmatrix}$$

$$G_1 = m_1 g l_{c,1} \cos(q_1) + m_2 g\left(l_1 \cos(q_1) + l_{c,2} \cos(q_1 + q_2)\right) + m_3 g\left(l_1 \cos(q_1) + l_2 \cos(q_1 + q_2)\right)$$

$$G_2 = m_2 g l_{c,2} \cos(q_1 + q_2) + m_3 g l_2 \cos(q_1 + q_2)$$

As forças de atrito são obtidas através de medidas experimentais e portanto não são apresentadas neste cálculo. No caso das simulações apresentadas neste trabalho elas serão desconsideradas

A Expressão (2.26) apresenta a equação final do movimento para o manipulador planar, semelhante a equação (2.2) desconsiderando porém as forças de atrito.

$$\tau = H(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) \qquad (2.26)$$

Geralmente quando se realizam simulações computacionais é utilizado o modelo representado em função de suas variáveis de estado. No caso do manipulador planar de 2 graus de liberdade tem-se um sistema de segunda ordem representado pela expressão (2.26). É interessante notar a necessidade da inversão da matriz de inércia H, conforme a equação (2.27). No caso aqui abordado trata-se de uma operação simples, porém a medida que tem-se mais juntas e consequentemente mais graus de liberdade e movimentos tridimensionais a inversão de H torna-se uma pesada tarefa computacional que deve ser realizada a cada instante de tempo. Explica-se portanto porque a maioria dos autores desenvolve seus experimentos com modelos planares, geralmente de 2 graus de liberdade. Em [5] é feito um estudo dos simuladores matemáticos mais utilizados e o autor conclui que a maioria deles é inadequado para modelos com mais de 3 graus de liberdade e com isso conclui à necessidade de desenvolver um novo simulador que melhor se adapte a essa tarefa.

Variáveis de estado: $x_1 = q \quad ; \quad x_2 = \dot{q}$

Equação de estados:

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = H(q)^{-1}\left(\tau - C(q,\dot{q})x_2 + g(q)\right) \end{cases} \qquad (2.27)$$

2.4 - CLASSIFICAÇÃO DOS MODELOS SEGUNDO OS PARÂMETROS DINÂMICOS A SEREM CONSIDERADOS

2.4.1. Introdução

Quando projeta-se controladores para manipuladores industriais é necessário ter em mente que, além da equação (2.2), existem dinâmicas não modeladas por esta que, dependendo do caso, podem influenciar decisivamente no comportamento do controle em malha fechada. A importância de considerar-se ou não tais parâmetros dinâmicos dá origem a diferentes modelagens além daquela apresentada na equação (2.2). Nesta seção aborda-se de maneira simplificada as três modelagens mais usuais em controle de robôs: o modelo rígido, o modelo com atuadores elétricos e o modelo com transmissões flexíveis.

2.4.2. O modelo com acionadores elétricos.

O acionamento elétrico é a forma mais simples e usual de acionamento encontrada em manipuladores industriais. Considera-se aqui o acionamento realizado por motores de corrente contínua controlados pela tensão de armadura.

A equação (2.28) é a equação da tensão V_i de armadura do motor acionador da i-ésima junta do manipulador, uma vez que tem-se um motor acionador para cada junta.

$$R_i i_i + L_i \frac{di_i}{dt} + Ke_i \frac{d\phi_i}{dt} = V_i \qquad (2.28)$$

onde:

- ϕ : posição angular do rotor.

- R : resistência da armadura.

- L: indutância da armadura.

- Ke : constante da tensão induzida.

- V : tensão aplicada a armadura.

Usualmente é necessário reduzir a velocidade na saída do motor. Utiliza-se então um redutor que é uma transmissão composta, como por exemplo um conjunto de

engrenagens, montada em série com a saída do atuador. Três hipóteses são consideradas a título de simplificação:

H2.1 - Os elos e as transmissões são rígidos.

H2.2 - O conjunto motor-transmissão é modelado por um rotor, o que significa que a energia potencial do conjunto motor-transmissão e a velocidade do seu centro de massa são independentes da posição do rotor.

H2.3- O conjunto motor-transmissão é montado diretamente sobre os elos.

Existe uma relação de transmissão entre a posição angular no rotor e a posição angular das juntas conforme (2.29). Esta relação de transmissão é usualmente incorporada ao valor das matrizes. Na equação (2.30) tem-se o torque nas juntas em função do torque gerado pelo motor na saída do redutor. A matriz J representa a inércia do rotor conforme H2.2. Na equação (2.31) tem-se a equação (2.28) reescrita em função das variáveis de junta.

$$q_i = N_i \phi_i \tag{2.29}$$

$$\tau = K_m I - J\ddot{q} \tag{2.30}$$

$$L\dot{I} + RI + K_e\dot{q} = V \tag{2.31}$$

Onde:

- $L = \text{diag}\,[\,L_i\,].\ i = 1,\ldots, n$
- $K_e = \text{diag}\,[\,Ke_i / N_i\,].\ i = 1,\ldots, n$
- $J = \text{diag}\,[\,J_i / N_i^2\,].\ i = 1,\ldots, n$
- $R = \text{diag}\,[\,R_i\,].\ i = 1,\ldots, n$
- $K_m = \text{diag}\,[\,Km_i / N_i\,].\ i = 1,\ldots, n$

Substituindo-se (2.26) em (2.30) tem-se o conjunto de equações (2.32) que representam as equações do modelo rígido acionado eletricamente.

$$\begin{cases} [H(q) + J]\ddot{q} + C(q,\dot{q}) + g(q) = K_m I \\ L\dot{I} + RI + K_e\dot{q} = V \end{cases} \tag{2.32}$$

Uma característica importante associada a este modelo é que ele é um sistema parcialmente acionado [10], ou seja, graus de liberdade adicionais são introduzidos com a inclusão da dinâmica elétrica e não são diretamente controlados pela tensão de armadura dos motores.

2.4.3. O modelo rígido.

Freqüentemente as constantes de tempo dos circuitos das armaduras dos motores L_i / R_i (i=1, ..., n) são muito menores que as constantes de tempo mecânicas do conjunto de elos e rotores definidas em (2.32). Por isso muitas vezes o modelo para manipuladores é desenvolvido desprezando a dinâmica elétrica dos atuadores. Isto significa que adicionalmente as hipóteses H2.1, H2.2 e H2.3, considera-se:

H2.4 - $L_i / R_i \approx 0 \quad (i=1, ..., n)$

H2.5 - $K_m R^{-1} K_e \approx 0$

Portanto o modelo em (2.32) passa a ser representado por (2.33).

$$u = [H(q) + J]\ddot{q} + C(q,\dot{q})\dot{q} + g(q) \qquad (2.33)$$

Observa-se que a equação (2.33) assemelha-se com a equação (2.26) a não ser pela inclusão da inércia do rotor que modela o conjunto motor-transmissão, o que incorpora um aspecto mais real a (2.26). Conclui-se portanto que toda vez que se trabalha com um modelo obtido unicamente das equações de Lagrange ou qualquer outro método como o modelo apresentado em (2.26) trabalha-se na verdade com o modelo rígido, assim chamado devido a hipótese H2.1, deprezando a dinâmica associada aos atuadores.

Grande parte dos resultados em robótica são baseados no modelo rígido. A inclusão da influencia de outras dinâmicas faz-se necessária somente quando as mesmas alteram significativamente o sistema de controle. No modelo rígido tem-se todos os graus de liberdade diretamente controlados, por isso diz-se que o sistema é totalmente acionado [10].

2.4.4. O modelo com transmissões flexíveis.

Um aspecto que por vezes tem influência significativa no modelo de um manipulador robótico, principalmente quando este é acionado por *harmonic drivers* [33] é a flexibilidade nas transmissões. A flexibilidade é uma característica de todos os materiais mas nos manipuladores a parcela significativa de flexibilidade esta associada as transmissões [13].

Supondo um manipulador com juntas de revolução acionado por motores DC, a flexibilidade na i-ésima junta pode ser modelada por uma mola torcional linear com uma determinada rigidez k_i ligando o rotor ao elo [32] conforme a figura 2.7 .

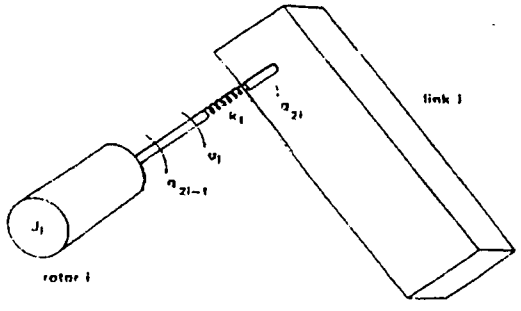


FIGURA 2.7 - Flexibilidade nas Transmissões.

O acoplamento elástico entre o rotor e o eixo de saída introduz graus de liberdade adicionais em relação ao manipulador rígido. Modela-se portanto o rotor como um elo fictício [32]. Conforme a figura 2.7, chama-se q_{1i} o ângulo que determina a posição do elo i e q_{2i} o ângulo que determina a posição do rotor i, sendo $q_{2i} = N_i\phi_i$, ϕ_i a posição angular do rotor que aciona o elo i e N_i a relação de transmissão no redutor i. Um conjunto de coordenadas generalizadas q é definido portanto, sendo $q^T = [\ q_1 \quad q_2\ ]^T$, $q_1 = [\ q_{11}\ ,\ \dots\ ,\ q_{1n}\ ]^T$ e $q_2 = [\ q_{21}\ ,\ \dots\ ,\ q_{2n}\ ]^T$ [32].

O modelo adotado neste trabalho é o modelo simplificado de Spong [32]. Neste modelo são consideradas as seguintes hipóteses:

H2.6 - Os elos são rígidos.

H2.7 - O conjunto motor-transmissão é modelado por um rotor, o que significa que a energia potencial do conjunto motor-transmissão e a velocidade do seu centro de massa são independentes da posição do rotor.

H2.8 - As transmissões são flexíveis e modeladas por uma mola torcional linear de rigidez k.

H2.9 - O conjunto motor-transmissão é montado diretamente sobre os elos.

H2.10 - $L_i / R_i \approx 0$ (i=1, ..., n).

H2.11 - $K_m R^{-1} K_e \approx 0$

H2.12 - A energia cinética ocasionada pela rotação do rotor em torno do seu próprio eixo é predominante quando comparada com a energia cinética ocasionada pela rotação do rotor em relação ao sistema inercial.

São introduzidas desta forma simplificações que conduzem as equações (2.34) e (2.35) que descrevem o modelo com transmissões flexíveis.

$$H(q)\ddot{q}_1 + C(q_1,\dot{q}_1)\dot{q}_1 + g(q_1) + K[q_1 - q_2] = 0 \qquad (2.34)$$

$$J\ddot{q}_2 - K[q_1 - q_2] = u \qquad (2.35)$$

O modelo flexível também é parcialmente acionado devido aos graus de liberdade adicionados ao considerar-se a dinâmica com transmissões flexíveis que não são diretamente controlados [32].

2.5 - CONCLUSÃO

Utilizando métodos tradicionais da mecânica como a formulação de Lagrange ou de Newton-Euler chega-se sempre à equação (2.2), chamada equação do movimento e que representa analiticamente o comportamento dinâmico de um manipulador robótico. Devido a complexidade da equação (2.2) para modelos com mais de 3 graus de liberdade, julgou-se melhor para este trabalho utilizar um modelo simplificado de 2 graus de liberdade composto de duas juntas rotacionais em um movimento planar. O modelo apesar de ser mais simples oferece uma boa noção do que aconteceria em um modelo mais completo em termos de controle, com a vantagem de

ser implementado de maneira relativamente simples pela maioria dos simuladores dinâmicos existentes.

As dinâmicas dos atuadores elétricos e da flexibilidade nas transmissões introduzem modificações significativas no modelo do manipulador quando consideradas. Entre estas modificações esta o fato de adicionarem graus de liberdade a mais ao modelo, tornando o sistema parcialmente acionado. É preciso portanto analisar com cuidado se é ou não significativa a consideração destas dinâmicas no projeto de sistemas de controle para manipuladores robóticos, pois os resultados obtidos para um modelo rígido podem não se aplicar aos casos onde elas atuam significativamente.

3 - PRINCIPAIS CONTROLADORES DE ROBÔS MANIPULADORES

3.1 - INTRODUÇÃO

Controlar um servo-mecanismo significa basicamente fazer com que o mesmo execute determinada tarefa com o mínimo de erro possível.. Neste capítulo são apresentadas algumas estratégias de controle utilizadas em robótica, suas vantagens e desvantagens de acordo com o ponto de vista dos autores pesquisados e a formulação matemática básica destes controladores.

3.2 - CONTROLADORES PD E PID

Controladores PD - Proporcional Derivativo e PID - Proporcional Integral derivativo são largamente aplicados dentro da teoria de controle clássico. Sua utilização abrange principalmente sistemas lineares SISO, ou seja, com uma variável de entrada e uma variável de saída à ser controlada.

Os manipuladores robóticos são sistemas não lineares altamente acoplados, ou seja, além de seu comportamento depender do estado das variáveis em um dado instante de tempo, a posição, velocidade e aceleração de determinada junta depende também dos valores de outra junta e assim por diante. Pode-se perguntar então como utilizar estes controladores em tal caso.

Em (3.1) tem-se a equação de um controlador PD, onde K_v e K_p são matrizes diagonais positivas definidas, q é o vetor da posição atual das juntas e q_d é o vetor da posição desejada para as juntas. K_p e K_v sendo diagonais implica que para cada junta tem-se um determinado valor de K_{p_i} e K_{v_i}, ou seja, existe um controlador PD independente para cada junta. Por isso que, em robótica, esse esquema de controle é chamado de controle PD descentralizado.

$$u = -K_p\tilde{q} - K_d\dot{\tilde{q}}, \qquad \text{sendo} \quad \tilde{q} = q - q_d \tag{3.1}$$

Em [33, 2] prova-se que este esquema de controle aplicado a equação (2.26) pela substituição de τ por u de (3.1) resulta assintóticamente estável quando compensada a parcela gravitacional. Se a parcela gravitacional é incluída aparecem erros

em regime permanente, que, segundo os autores, diminuem a medida que se aumenta o valor de K_p.

Em (3.2) tem-se a equação do controlador PID, que é semelhante a equação (3.1) porém com a inclusão de um termo integral multiplicado por um ganho K_i. Com a inclusão do termo integral obtêm-se erro em regime permanente nulo quando $t \to \infty$, porém a estabilidade do controle deixa de ser global e torna-se local.

$$u = -K_p\tilde{q} - K_d\dot{\tilde{q}} - K_i\int \tilde{q}\, dt, \qquad \text{sendo} \quad \tilde{q} = q - q_d \tag{3.2}$$

A grande vantagem destes controladores é sem dúvida a simplicidade quanto a implementação, além de exigir menos recursos computacionais. Pode-se obter também controladores bastante robustos com este esquema.

A desvantagem destes controladores é o seu alto erro transitório, o que os torna impraticáveis para algumas aplicações de seguimento de trajetória. O erro transitório deve-se principalmente ao fato destes controladores desconsiderarem as não-linearidades do modelo do manipulador, considerando o sistema como se fosse constituído de n duplos integradores independentes, o que na prática não ocorre.

Portanto, os controladores PD e PID são utilizados principalmente em tarefas do tipo ponto a ponto ou em seguimento de trajetórias onde o desempenho exigido se enquadre dentro das suas limitações.

3.3 - CONTROLADOR BASEADO NA DINÂMICA INVERSA

O Controlador baseado na dinâmica inversa é uma alternativa onde não se pode utilizar controladores do tipo PD ou PID devido as limitações dos mesmos. É um controlador utilizado em tarefas que exijam precisão e erro transitório pequeno.

O controlador consiste no cancelamento direto das não linearidades do modelo do robô manipulador, tornando o sistema linear e desacoplado. Tal sistema pode então ser controlado diretamente por algoritmos PD ou PID clássicos. O procedimento de linearização é um caso especial de aplicação da técnica de linearização por realimentação [31]. Em (3.3) apresenta-se a equação do controlador.

$$\tau = \hat{H}(q)(\ddot{q}_d - u) + \hat{C}(q,\dot{q})\dot{q} + \hat{G}(q) \tag{3.3}$$

Onde $\hat{H}(q)$, $\hat{C}(q,\dot{q})$ e $\hat{G}(q)$ são estimações para os valores reais das matrizes de (2.26).

Supondo que as matrizes estimadas sejam iguais as matrizes reais, substituindo (3.3) em (2.26) tem-se (3.4). O sistema torna-se portanto linear e formado por n duplos integradores desacoplados. Nesta situação pode-se definir u como uma equação de controle PD ou PID, conforme (3.5). Substituindo (3.5) em (3.4) tem-se (3.6). Pelas técnicas clássicas de controle, como o lugar das raízes por exemplo, escolhe-se os parâmetros PID de forma a garantir uma resposta criticamente amortecida para o sistema.

$$\ddot{\tilde{q}} = u \ , \quad \text{sendo } \tilde{q} = q - q_d \tag{3.4}$$

$$u = -K_p\tilde{q} - K_d\dot{\tilde{q}} - K_i\int \tilde{q}\, dt \tag{3.5}$$

$$\ddot{\tilde{q}} + K_v\dot{\tilde{q}} + K_p\tilde{q} + K_i\int \tilde{q}dt = 0 \tag{3.6}$$

A grande vantagem dos algoritmos baseados na dinâmica inversa esta no fato de tornar um sistema não-linear e acoplado um sistema linear desacoplado. Pode-se portanto obter erros de rastreamento pequenos quando se tem boas estimações em (3.3).

A principal dificuldade quando se trabalha com esta estratégia é que ela exige que as estimações em (3.3) sejam próximas dos valores reais, caso contrário o erro transitório aumenta e o sistema pode tornar-se instável. Obter boas estimações para as matrizes da equação do manipulador não é uma tarefa simples, pois a carga a ser transportada pelo efetuador varia e portanto proporciona alterações também nos valores reais das matrizes, que devem serem estimadas em tempo real. Sendo assim, mesmo em dinâmicas bastante conhecidas pode encontrar-se dificuldades em trabalhar com este algoritmo.

3.4- CONTROLADOR TIPO TORQUE COMPUTADO

O controlador tipo torque computado é uma alternativa de controle baseada no mesmo princípio de linearização do robô que o controlador baseado na dinâmica inversa. A diferença principal entre os mesmos é que o controlador do tipo torque computado utiliza-se das posições desejadas e de suas derivadas de ordem

superior para a estimação da dinâmica inversa que é aplicada na equação de controle. Este procedimento é descrito pela equação (3.7).

$$\tau = \hat{H}(q_d)(\ddot{q}_d - u) + \hat{C}(q_d, \dot{q}_d)\dot{q}_d + \hat{G}(q_d) \qquad (3.7)$$

A desvantagem deste controlador em relação ao anterior reside no fato da dinâmica inversa utilizada em sua lei de controle utilizar informações desejadas, que não são obtidas diretamente do manipulador que esta sendo controlado, o que não acontece no caso do controlador baseado na dinâmica inversa. Dependendo do algoritmo utilizado para a estimação da dinâmica inversa, este fato pode comprometer a precisão do controle, uma vez que as posições desejadas sempre estão afastadas das posições reais pela presença de um erro.

3.5 - CONTROLADOR BASEADO NA PASSIVIDADE

Os controladores baseados na passividade são chamados assim porque utilizam a propriedade da passividade para estabilização do manipulador robótico. Diferem dos controladores baseados na dinâmica inversa principalmente porque sua lei de controle não lineariza a equação do movimento e mesmo assim diminui significativamente o erro transitório e em regime permanente do sistema controlado.

Considerando-se um sistema que tem o mesmo numero de entradas e saídas (y(t) tem a mesma dimensão de u(t)), o sistema é dito passivo se:

$$\int_0^T y^T(t)u(t)\,dt \geq \gamma$$

Para $T > 0$ (finito) e algum $\gamma > -\infty$

Para uma descrição mais precisa, considere-se os sistemas dinâmicos que satisfazem equações da forma:

$$\frac{d}{dt}\begin{bmatrix} \text{Energia} \\ \text{Armazenada} \end{bmatrix} = \begin{bmatrix} \text{Pot. de} \\ \text{entrada} \end{bmatrix} + \begin{bmatrix} \text{Pot. gerada} \\ \text{internamente} \end{bmatrix} \qquad (3.8)$$

A potência de entrada pode ser representada pelo produto escalar da entrada "u" pela saída "z" , $z^T u$. Mais genericamente a equação (3.8) pode ser representada por (3.9).

$$\dot{V}(t) = z^T u - g(t) \tag{3.9}$$

Onde V(t) e g(t) são funções escalares do tempo.

Um sistema que verifica uma equação na forma (3.9) com V(t) limitada e $g(t) \geq 0$ é chamado passivo. Diz-se então que existe um mapeamento passivo entre "u" e "z".

Em sistemas passivos não se cria energia. O conceito de passividade portanto generaliza a idéia da conservação de energia em sistemas dinâmicos nos quais não é gerada potência internamente, ou seja, entre uma entrada e uma saída.

A existência de um mapeamento passivo entre a entrada e uma saída a ser controlada pode ser utilizada na estabilização de sistemas. Uma vez definida a entrada e a saída entre as quais há um mapeamento passivo, basta fechar a malha com um dissipador de energia (amortecimento) para que esta energia decresça ao longo do tempo.

Considerando agora a equação (2.26) de um manipulador robótico rígido totalmente acionado. Supõe-se que o robô representa um sistema cuja entrada é o torque τ e cuja saída é a velocidade nas juntas $\dot{q}$. Considerando a soma da energia cinética e da energia potencial do robô sendo representada pelo Hamiltoniano H e lembrando que:

$$\frac{dH}{dt} = \dot{q}^T \tau$$

Então:

$$\int_0^T q^T(t)\tau(t)\,dt = H(t) - H(0) \geq -H(0)$$

Que prova que de τ para $\dot{q}$, o robô rígido é passivo.

A expressão matricial desta propriedade estrutural da equação (2.26), ou seja, da passividade do mapeamento torque-velocidade do manipulador é dada pela anti-simetria da matriz $\left[\dot{H}(q) - 2C(q,\dot{q})\right]$. O controle baseado na passividade utiliza esta anti-simetria para estabilização do sistema e conta também com o fato deste ser totalmente acionado[10].

Em (3.10) tem-se a equação de um controlador baseado na passividade.

$$\tau = \hat{H}(q)\ddot{q}_r + \hat{C}(q,\dot{q})\dot{q}_r + \hat{g}(q) - K_D s \tag{3.10}$$

Onde $\hat{H}(q)$, $\hat{C}(q,\dot{q})$ e $\hat{G}(q)$ são estimações para os valores reais das matrizes de (2.26) e K_d é uma matriz diagonal positiva definida e $\dot{q}_r$ é a velocidade de referência

A velocidade de referência $\dot{q}_r$ é formada deslocando-se a velocidade desejada $\dot{q}_d$ de acordo com o erro de posição $\tilde{q}$ conforme (3.10). Tal operação representa simplesmente uma manipulação notacional que permite traduzir as propriedades relacionadas a energia (expressadas em termos do vetor da velocidade atual na junta $\dot{q}$) em propriedades de controle de trajetória (expressadas em termos do vetor do erro de velocidade virtual s) [31]. A definição de s em (3.12) pode ser vista como a equação de um filtro estável de primeira ordem em que s é a entrada e $\tilde{q}$ é a saída. Portanto, assumindo condições iniciais limitadas, se s é limitado, $\tilde{q}$ e $\dot{\tilde{q}}$ também são, e da mesma forma q e $\dot{q}$(para q_d e $\dot{q}_d$ limitadas). Pela mesma razão, se $s \to 0$ quando $t \to \infty$, então $\tilde{q} \to 0$ e $\dot{\tilde{q}} \to 0$ quando $t \to \infty$.

$$\dot{q}_r = \dot{q}_d - \Lambda\tilde{q}, \quad \text{sendo} \quad \tilde{q} = q - q_d \tag{3.11}$$

onde Λ é uma matriz diagonal positiva.

$$s = \dot{q} - \dot{q}_r = \dot{\tilde{q}} + \Lambda\tilde{q} \tag{3.12}$$

Substituindo-se (3.10) em (2.26) tem-se (3.13):

$$H(q)\dot{s} + C(q,\dot{q})s + K_d s = 0 \tag{3.13}$$

De maneira distinta ao que ocorre com o controle baseado na dinâmica inversa, a equação de malha fechada não conduz a uma dinâmica linear. Pode-se provar entretanto que a equação (3.13) é estável e que o erro $\tilde{q}$ tende a zero quando t tende ao infinito.

Supondo uma função não negativa (3.14).

$$V = \frac{1}{2}s^T H(q)s \tag{3.14}$$

A derivada de (3.14) em relação ao tempo resulta em (3.15).

$$\dot{V} = s^T\left(H(q)\dot{s} + \frac{1}{2}\dot{H}(q)s\right) \tag{3.15}$$

Substituindo-se (3.13) em (3.15) tem-se (3.16).

$$\dot{V} = s^T\left[\frac{1}{2}\dot{H}(q) - C(q,\dot{q})\right]s - s^T K_D s \tag{3.16}$$

Da anti-simetria de $\left[\dot{H}(q) - 2C(q,\dot{q})\right]$ tem-se que (3.17).

$$\dot{V} = -s^T K_D s \quad , \tag{3.17}$$

De (3.14) e (3.17) conclui-se que s é limitada. Portanto $\tilde{q}$ e $\dot{\tilde{q}}$ também são. Da dinâmica do sistema (3.13) $\dot{s}$ é então limitado e s e $\dot{V}$ são uniformemente contínuos. Portanto, do Lema de Barbalat [31], $s \to 0$ quando $t \to \infty$ e $\tilde{q} \to 0$ quando $t \to \infty$.

A grande vantagem do controlador baseado na passividade a ser explorada neste trabalho é o fato dele ser mais robusto do que o controlador baseado na dinâmica inversa e o controlador tipo torque computado, pois não necessita de estimações bem próximas da realidade para que a malha fechada seja estável. Um exemplo deste fato é o controlador adaptativo de Slotine e Li [10, 31, 20, 33]. Neste controlador as matrizes estimadas da equação (3.13) são parametrizadas em termos de uma matriz regressora $Y(q,\dot{q},q,\dot{q}_r)\hat{a}$, onde $\hat{a}$ é um vetor de parâmetros desconhecidos a serem estimados em tempo real. Prova-se que o algoritmo garante convergência de seguimento independente do valores iniciais do vetor $\hat{a}$.

A equação do algoritmo apresentada em (3.13) pode na verdade ser dividida em duas partes. A primeira parte constando das matrizes estimadas, ou seja uma parcela *feedforward* e a segunda parte uma parcela constituída de uma simples equação de um controlador PD. Portanto este controlador muitas vezes é chamado de controlador PD+*feedforward*.

3.6 - CONCLUSÃO

Os controladores PD e PID são estratégias simples, porém tem performance insatisfatória em muitas aplicações de seguimento de trajetória. Os controladores baseados na dinâmica inversa e controladores baseados na passividade apresentam boas soluções para o problema do erro de seguimento. O controlador baseado na dinâmica inversa pode ser usado em situações onde se tenha condições de realizar boas estimações em relação aos parâmetros da equação do movimento, uma vez que ele facilita a sintonia dos ganhos PD ou PID pelo fato de linearizar o sistema. O controlador baseado na passividade utiliza uma estratégia de controle mais robusta que os controladores baseado na dinâmica inversa e torque computado e em algumas situações pode apresentar melhor performance que os mesmos, apesar de não linearizar o sistema.

Os controladores aqui apresentados atuam sobre o modelo rígido, sem qualquer consideração acerca do que aconteceria caso atuassem sobre os modelos descritos no capítulo 2. Tais considerações podem ser encontradas na literatura pesquisada [10, 33, 32, 24] e em exemplos que serão apresentados posteriormente neste trabalho.

4 - FUNDAMENTOS TEÓRICOS DE REDES NEURAIS

4.1 - INTRODUÇÃO

4.1.1. Definição

As redes neurais artificiais, doravante chamadas aqui de RNA, são baseadas nas redes neurais biológicas do cérebro humano, embora estas ainda tenham comportamento em grande parte desconhecido pelos pesquisadores. Traçando-se um paralelo com sistemas de controle, pode-se dizer que hoje se possui um modelo de primeira ordem do neurônio. Em anos recentes, houve um crescente interesse em estudar os mecanismos e estrutura do cérebro. Este interesse conduziu ao desenvolvimento de novos modelos computacionais, baseados nestes conhecimentos biológicos, para resolver problemas complexos como reconhecimento de padrões, rápido processamento de informação e adaptação.

4.1.2 - O Neurônio Artificial

Como já foi dito, as RNA baseiam-se na estrutura do cérebro humano, embora se conheça pouco sobre a operação completa do mesmo. O sistema nervoso humano é constituído de células chamadas neurônios e é extremamente complexo. Para ter-se idéia é estimada a existência de cerca de 10^{11} neurônios participando de uma infinidade de interconexões. Cada neurônio compartilha muitas características com outras células do corpo, mas com uma capacidade única para receber, processar e transmitir sinais eletroquímicos sobre as vias neurais do sistema de comunicação do cérebro.

A Figura 4.1 mostra um neurônio biológico. Elementos chamados **Dentrites** prolongam-se do corpo da célula para outros neurônios onde eles recebem sinais num ponto de conexão chamado **synapse**. No lado da recepção da synapse, essas entradas são conduzidas para o corpo da célula, onde são somadas. Algumas entradas tendem a excitar a célula, outras tendem a inibir o seu disparo. Quando a excitação acumulada excede um limite, a célula dispara enviando um sinal através do **axônio** para outros neurônios.

O neurônio artificial foi projetado para imitar as características de 1ª ordem do neurônio biológico [28]. De fato, um conjunto de entradas é aplicado, cada uma produzindo uma saída para outro neurônio. Cada entrada é multiplicada por um **peso** (**weight**) correspondendo analogamente a força sináptica, e todas as entradas pesadas são então somadas para determinar o nível de ativação do neurônio. É adicionado ainda um valor limite θ conhecido por *threshold* ou off-set. A Figura 4.2 mostra o modelo que implementa esta idéia.

Como formalização, pode ser usada uma representação vetorial:

$$NET = X \cdot W + \theta$$

NET : Soma algébrica das entradas pesadas;

X : vetor de entrada;

W : vetor de pesos.

θ : Threshold ou off-set

O resultado da soma (NET) é processado por uma função de ativação, que tem como pré-requisito ser contínua e diferenciável, para produzir o sinal de saída do neurônio.

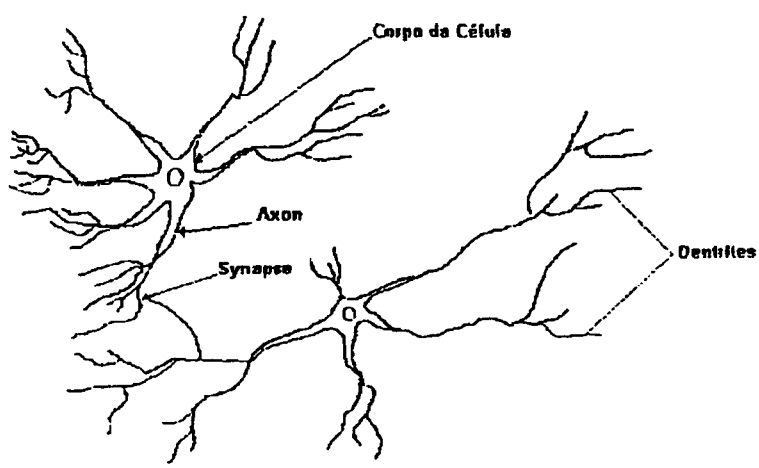


FIGURA 4.1 - Neurônio Biológico

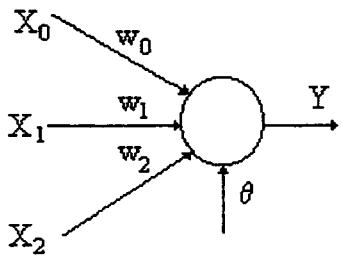


FIGURA 4.2 - Neurônio Artificial

4.1.3. Redes Neurais aplicadas a sistemas de controle

As RNA possuem propriedades e características importantes, que motivam a sua aplicação em sistemas de controle.:

- Constituem um sistema de processamento paralelo distribuído. O elemento de processamento básico de uma RNA tem uma estrutura muito simples que em conjunto com uma implementação em paralelo resulta em grande rapidez de processamento mesmo em processos complexos.

- Possuem aprendizado e adaptação. As RNA são treinadas através de exemplos colhidos do sistema em estudo. Uma RNA convenientemente treinada é capaz de generalizar os resultados para situações que não fazem parte do conjunto de treinamento. As RNA também podem apresentar adaptação on-line.

- São sistemas multivariaveis. As RNA podem processar muitas entradas e possuírem muitas saídas. Elas são prontamente aplicáveis à sistemas multivariáveis.

- Habilidade de trabalhar com sistemas não lineares. As RNA possuem propriedades que possibilitam o aprendizado de mapeamentos não lineares. Podem ser portanto boa alternativa aos complexos métodos convencionais de controle de sistemas não lineares.

A arquitetura mais utilizada em sistemas de controle são as redes *feedforward* multicamadas utilizando o algoritmo backpropagation associado a alguma modificação que vise superar as carências do mesmo. Dentro das aplicações de controle, dois esquemas de implementação tornaram-se populares: o controle direto e o controle indireto [39]. No **controle indireto** são utilizadas duas redes neurais, sendo a primeira treinada para representar o comportamento da planta a ser controlada, enquanto a

segunda é treinada para controlar a planta e utiliza o erro retropropagado através da primeira rede para ajustar os seus pesos. No **controle direto** somente uma RNA é utilizada, funcionando como controlador e o erro para o ajuste dos pesos é obtido diretamente através de uma função matemática que representa o comportamento da planta a ser controlada. Estas duas técnicas são bastante utilizadas, principalmente em substituição aos tradicionais controladores PD ou PID em situações onde a sintonia dos parâmetros destes é crítica.

As RNA podem também serem usadas somente como ferramentas de representação do modelo direto ou inverso de uma planta a ser controlada. Tal representação é geralmente utilizada em conjunto com alguma estratégia convencional de controle tal como o controle preditivo [26], esquemas IMC [3] ou controle tipo torque computado [30] por exemplo. Tais esquemas de controle exploram as RNA com este fim e são utilizados principalmente em situações de controle que exigem estratégias mais complexas que os tradicionais PD ou PID aplicados diretamente ao processo. O controle de manipuladores robóticos constitui um exemplo típico desta situação.

4.2 - REDE FEEDFORWARD MULTICAMADAS

A Arquitetura de RNA *feedforward* multicamadas é a arquitetura de RNA utilizada neste trabalho. Como exemplo de uma RNA *feedforward* multicamadas tem-se a figura 4.3 que trata de uma rede simples de três camadas.

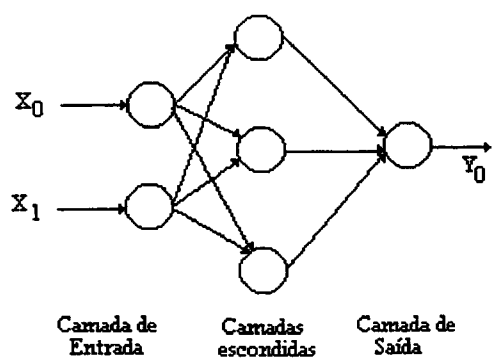


FIGURA 4.3 - Rede *Feedforward* Multicamadas

A rede divide-se em três camadas com funções distintas:

A **camada de entrada** tem a função de receber os sinais provenientes do mundo externo e distribui-los para os neurônios da camada escondida. Portanto os

neurônios da camada de entrada são considerados inativos pois não realizam processamento dos dados. O número de neurônios nesta camada é sempre igual ao número de entradas da rede (considerando um delay, caso exista, como entrada).

A **camada intermediária ou escondida**, que pode ser mais de uma, tem a função de processar os dados recebidos da camada de entrada. Os neurônios desta camada são aqueles apresentados na figura 4.2. Atribui-se as camadas internas o mapeamento não linear entre as entradas e as saídas da rede e a supressão dos padrões de ruído que porventura possam existir [17, 22].

A **camada de saída** possui neurônios idênticos aos da camada escondida e realiza o processamento final, oferecendo os resultados ao mundo externo. O número de neurônios desta camada é sempre igual ao número de saídas da rede.

As saídas dos nós de uma camada, ou seja, as saídas dos neurônios desta camada, são transmitidos aos nós de outras camadas através de uma conexão, que é amplificada, atenuada ou inibida por meio de elementos chamados **pesos**. A entrada de cada nó da rede, exceto na camada de entrada, recebe a soma das saídas dos nós da camada anterior ponderada pelos respectivos pesos. Faz parte desta soma também uma conexão que tem sua entrada mantida sempre em 1 (um) e ponderada, a exemplo dos pesos, por um elemento denominado ***threshold*** . O *threshold* confere uma característica de estabilidade à fase de treinamento e tem o efeito prático de provocar um deslocamento no eixo da função de ativação, por isso também é chamado de off-set.

Considerando uma representação matricial para o equacionamento de uma RNA *feedforward* multicamadas e tomando-se o exemplo da figura 4.3 tem-se que as equações (4.2) e (4.3) representam respectivamente as saídas da camada escondida e da camada de saída da rede. A notação W representa os pesos, y as saídas das camadas, θ os thresholds e in e out as entradas e saídas ao meio externo.

$$y_{in} = N(in) \tag{4.1}$$

$$y_E = F(W_{in} \cdot y_{in} + \theta_E) \tag{4.2}$$

$$y_{out} = F(W_E \cdot y_E + \theta_{out}) \tag{4.3}$$

$$out = D(y_{out}) \tag{4.4}$$

y_{in} = vetor n_{in} x 1; y_E = vetor n_E x 1; y_{out} = vetor n_{out} x 1; θ_E = vetor n_E x 1; θ_{out} = vetor n_{out} x 1; W_{in} = matriz n_E x n_{in} ; W_E = matriz n_{out} x n_E .

n_{in}, n_E e n_{out} são a quantidade de neurônios nas respectivas camadas.

De acordo com o exemplo da figura 4.3 tem-se que:

$$y_{in} = \begin{bmatrix} i_1 & i_2 \end{bmatrix}^T ; \; \theta_E = \begin{bmatrix} \theta_1^E & \theta_2^E & \theta_3^E \end{bmatrix}^T ; \; \theta_{out} = \begin{bmatrix} \theta_1^{out} \end{bmatrix}$$

$$W_{in} = \begin{bmatrix} w_{11}^{in} & w_{12}^{in} & w_{13}^{in} \\ w_{21}^{in} & w_{22}^{in} & w_{23}^{in} \end{bmatrix}^T \; ; \; W_E = \begin{bmatrix} w_{11}^{E} & w_{12}^{E} \\ w_{21}^{E} & w_{22}^{E} \\ w_{31}^{E} & w_{32}^{E} \end{bmatrix}^T .$$

A função F é chamada **função de ativação**. Em geral a maioria dos autores utilizam como função de ativação a função sigmoidal descrita pela equação (4.5) e pela figura 4.4, que é não linear, diferenciável e monótona. É necessário para o treinamento que esta função seja contínua e diferenciável e a função sigmoidal além de cumprir estes pré-requisitos possui uma derivada simples de ser obtida, descrita pela equação (4.6).

$$F(x) = \frac{1}{1 + e^{-\alpha x}} \tag{4.5}$$

$$F'(x) = F(x) \; (\; 1 - F(x) \;) \tag{4.6}$$

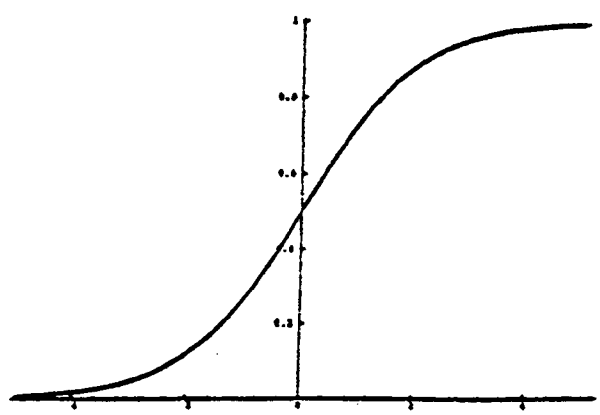


FIGURA 4.4 - Função Sigmoidal

O ganho α da função determina a inclinação da curva da figura 4.4. Afirma-se que este ganho, apesar de não ter influência decisiva sobre o desempenho da

rede, influi na velocidade de treinamento [22]. A maioria dos autores adota o valor padrão $\alpha = 1$, que é também adotado neste trabalho.

Algumas outras funções são utilizadas com intuito principalmente de aumentar a velocidade de treinamento da RNA, entre elas principalmente a função tangente hiperbólica [16] e a função arcotangente [22, 15]. Apesar das vantagens apresentadas por alguns autores em relação a estas funções [16] a maioria das abordagens mantém a função sigmoidal como função de ativação da rede.

Não necessariamente todos os neurônios da rede precisam ter a mesma função de ativação, embora esta seja a situação predominante na literatura. Em [22] por exemplo cita-se o caso de que as funções de ativação dos neurônios da camada de saída são lineares, ou seja, $f(x) = x$, e atribui-se algumas vantagens a esse procedimento. Neste trabalho entretanto adota-se a situação padrão onde todos os neurônios tem a mesma função de ativação.

Uma característica importante da função sigmoidal é mapear a saída de cada neurônio em valores compreendidos entre 0 e 1. Devido a esta característica cada neurônio deve receber em sua entrada valores também compreendidos entre 0 e 1. As entradas da rede recebem sinais provenientes do meio externo bem como as suas saídas fornecem sinais ao meio externo. É necessário portanto uma função de normalização que condicione uma variação máxima e mínima do meio externo a uma variação entre 0 e 1. A função descrita pela equação (4.7) seria capaz de realizar esta tarefa, entretanto deve-se observar que a função sigmoidal dada pela equação (4.5) atinge os limites 0 e 1 para valores de entrada infinitamente grandes, positivos ou negativos. Isto significa que os limites de normalização dificilmente serão atingidos. Tal situação pode ser contornada de duas maneiras. A primeira estaria em fixar limites bem acima e bem abaixo do esperado, o problema no entanto seria em determinar quanto acima ou quanto abaixo deveria se extrapolar. Para evitar esta dúvida, a segunda opção fornece uma saída direta para o problema. A equação (4.7) pode ser modificada à fim de restringir a sua faixa de atuação entre 0.1 e 0.9. Isto pode ser obtido através da equação (4.8). Nas equações (4.1) e (4.4) pode-se notar a aplicação de (4.8) e da respectiva equação de desnormalização (4.9) na tarefa de receber sinais do meio externo e enviá-los ao mesmo.

$$N(x) = \frac{x - X_{min}}{X_{max} - X_{min}} \tag{4.7}$$

$$N(x) = \frac{0.9 - 0.1}{X_{max} - X_{min}}\left(x - X_{min}\right) + 0.1 \tag{4.8}$$

$$D(y) = \frac{y - 0.1}{0.9 - 0.1}\left(X_{max} - X_{min}\right) + X_{min} \tag{4.9}$$

A notação utilizada nas equações (4.1) e (4.4) foi adotada com fins de simplificar o equacionamento, mas torna-se claro que para cada entrada e cada saída tem-se diferentes valores de X_{max} e X_{min} de acordo com as grandezas envolvidas.

4.3 - ASPECTOS DE TREINAMENTO

Seja o sistema mostrado na figura 4.5 tendo como entradas $u = [\, u_1 , u_2 , \ldots , u_n \,]^T$ e as correspondentes saídas $y = [\, y_1 , y_2 , \ldots , y_n \,]^T$; então é conveniente representar este sistema pela relação R : U→Y que mapeia o espaço de entrada U: { u ∈ U / u é a entrada do sistema } no espaço de saída Y: { y (u) ∈ Y / y é a saída do sistema com entrada u }. O objetivo do treinamento é que a rede possa aprender a relação R. A saída da RNA é denotada como $\hat{y}(u,w)$ e pertence ao espaço de saída Y, parametrizado com pesos w ∈ W dado a entrada u.

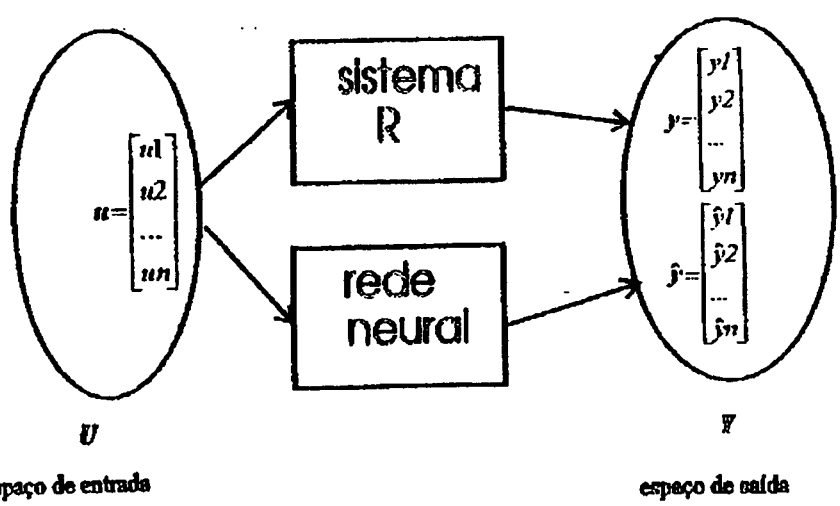


FIGURA 4.5 - Diagrama Esquemático da Relação Entrada - Saída de Um Sistema e Uma Rede Neural

O processo de treinamento da RNA consiste no ajuste dos parâmetros internos da rede (pesos) w, que são usados para representar a relação R dentro de uma determinada tolerância. A equação (4.10) representa matematicamente o processo de treinamento.

$$\min_{w \in W} \left\| \hat{y}(u,w) - y(u) \right\| \text{ para todo } u \in U. \tag{4.10}$$

A restrição imposta ao treinamento é que a diferença entre a saída da rede $\hat{y}(u,w)$ e a saída do sistema y(u) para a mesma entrada u seja menor que uma tolerância preestabelecida, isto é, $\left\| \hat{y}(u,w) - y(u) \right\| < \varepsilon_a$,. Se ε_a, é suficientemente pequeno então $\hat{y}$ representa y.

A Relação contínua exata do sistema R(u) não necessita ser conhecida. A rede é treinada baseado em um conjunto de medidas discretas das entradas e saídas { (u_1 , y_1); (u_2 , y_2); ... ; (u_p , y_p) } tomado do sistema aqui denominado de ***pares de treinamento***. O processo de treinamento consiste em fazer com que a rede aprenda a relação R baseada nos *pares de treinamento*.

Supondo que existam P padrões de treinamento disponíveis para treinar a RNA, sendo u_p o p-ésimo padrão de entrada e y_p o p-ésimo padrão de saída , portanto (u_p , y_p) é o p-ésimo padrão de treinamento. A equação (4.11) representa o processo de treinamento como um todo.

$$\min_{w \in W} \left\| \hat{y}(u_P,w) - y(u_P) \right\|_2 = \min_{w \in W} \sqrt{\frac{1}{P} \sum_{p=1}^{P} \left(\hat{y}(u_p,w) - y(u_p) \right)^2} \text{ com } p = 1, 2, \ldots, P \tag{4.11}$$

O erro E_p corresponde ao erro do padrão de treinamento p e é definido pela equação (4.12). Geralmente se quer minimizar o erro total E_t compreendido pela soma de todos os erros E_p conforme a equação (4.13).

$$E_p = \left\| \hat{y}(u_p,w) - y(u_p) \right\|_2 \tag{4.12}$$

$$E_t = \frac{1}{P} \sum_{p=1}^{P} E_p \tag{4.13}$$

Além de minimizar o erro E_t de acordo com o objetivo ε_a o treinamento precisa passar por outra fase importante chamada **validação**. A validação consiste em

verificar se o treinamento mapeia R sobre determinadas condições de interesse do usuário. O processo mais comum de validação é escolher dois conjuntos distintos de *pares de treinamento* no espaço U, um de treinamento e outro de teste. O treinamento é realizado sobre o primeiro conjunto e quando concluído a rede é testada mediante a apresentação do segundo. Desta forma avalia-se a **capacidade de generalização da rede**, que é a capacidade da RNA representar satisfatoriamente um mapeamento R dentro de qualquer vetor no espaço U e portanto a validade do treinamento. As principais causas do fracasso de um treinamento que atinge E_t mínimo são atribuídos a escolha errada do conjunto de *pares de treinamento* utilizado no treinamento, ou seja, escolheu-se um conjunto pouco representativo dentro do espaço U e também ao fenômeno do **sobretreinamento** [22 , 17], que ocorre quando a rede que aproxima o conjunto de pontos discretos apresenta uma resposta errônea devido ao excesso de treinamento. A figura 4.6a e 4.6b mostra os gráficos de aproximação típicos de uma rede bem sucedida e de uma rede sobretreinada.

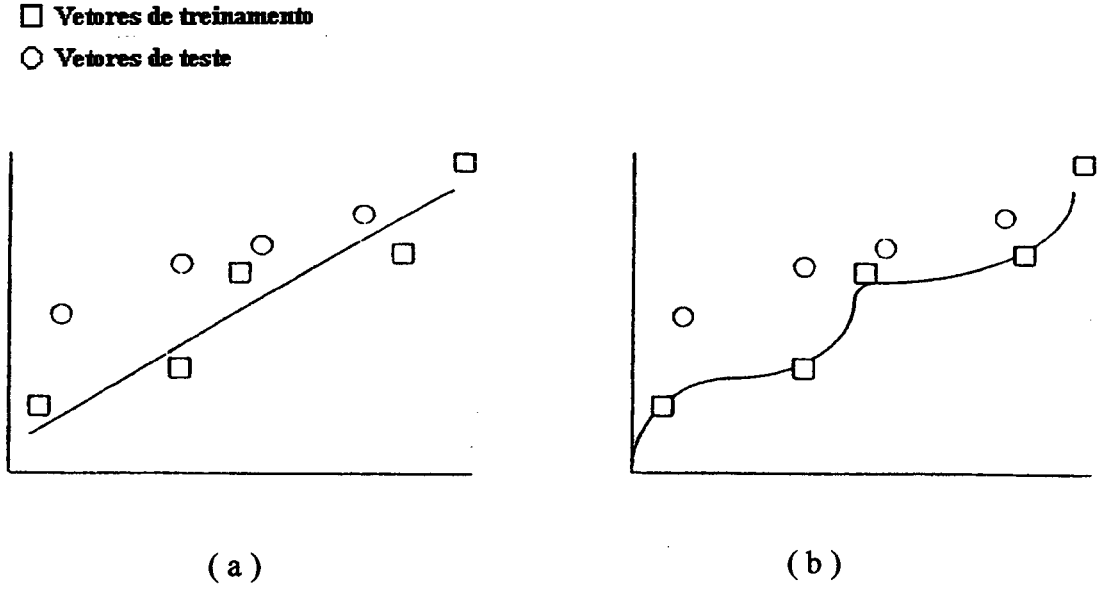


FIGURA 4.6 - Rede Bem Sucedida e Rede Sobretreinada

Vários algoritmos de treinamento para redes *feedforward* multicamadas tem surgido ao longo dos anos, principalmente depois da publicação dos trabalhos do *PDP research group* [27] que introduziu o algoritmo Backpropagation. Este trabalho detém-se mais detalhadamente neste algoritmo e em algumas modificações do mesmo propostas por [9] visando melhorar seu tempo de treinamento.

4.4 - O ALGORITMO BACKPROPAGATION

4.4.1. Introdução

O algoritmo Backpropagation surgiu com o trabalho do *PDP research group* [27] e desde então se tornou o mais popular e difundido método de treinamento em RNA *feedforward* multicamadas. O algoritmo é baseado no critério dos mínimos quadrados, que consiste em minimizar o somatório do quadrado da distância em relação a curva objetivo de cada ponto amostrado. Para tanto utiliza o método do gradiente descendente que consiste de a cada iteração alterar os elementos de ponderação à fim de dar um passo no sentido contrário ao gradiente da função, minimizando-a assim em um total de passos K. A fim de entender melhor a descrição matemática do algoritmo uma notação padrão é adotada.

$\frac{\partial E_p}{\partial w_{ij}}$: derivada parcial do erro E_p com respeito a w_{ij}.

$\frac{\partial E_t}{\partial w_{ij}}$: derivada parcial do erro E_t com respeito a w_{ij}.

Portanto : $\frac{\partial E_t}{\partial w_{ij}} = \sum_p \frac{\partial E_p}{\partial w_{ij}}$.

∇E_t : Gradiente com respeito ao conjunto completo de treinamento.

$\nabla E_t = \left(\frac{\partial E_t}{\partial w_1}, \frac{\partial E_t}{\partial w_2}, \ldots, \frac{\partial E_t}{\partial w_n} \right)$, sendo w_1 , w_2 , ... , w_n os pesos da rede.

∇E_p : Gradiente com respeito ao padrão p.

$\nabla E_p = \left(\frac{\partial E_p}{\partial w_1}, \frac{\partial E_p}{\partial w_2}, \ldots, \frac{\partial E_p}{\partial w_n} \right)$

Do ponto de vista da RNA, o algoritmo visa a minimização do somatório da diferença entre o padrão de saída da rede e o padrão de saída desejado ao longo de todo o conjunto de padrões amostrado, ou seja, minimiza E_t em (4.13).

4.4.2. Descrição Matemática do Algoritmo

O método do gradiente descendente é descrito pela equação (4.15), que aponta a direção Ξ (W) onde a função tende ao mínimo, considerando (4.14) como a função a ser minimizada.

$$J(W) = \frac{1}{P}\sum_{P} E_p{}^T E_p \tag{4.14}$$

$$\Xi(W) = -\nabla J(W) = -\frac{1}{P}\sum_{P}\frac{\partial E_p(W)}{\partial W} \tag{4.15}$$

Aproveitando as propriedades de paralelismo das RNA e também a possibilidade de propagar reversamente o erro é possível obter uma forma recursiva de cálculo da equação (4.15) com facilidade de implementação computacional, o que tornou o backpropagation tão popular no treinamento de RNA.

Considera-se Δw_{ji} a direção de minimização para um determinado peso w_{ji} conforme a figura 4.7. Pretende-se obter esta direção calculando recursivamente a equação (4.15), portanto calculando a derivada do erro na saída y_j em relação ao peso w_{ji} conforme a equação (4.16). O parâmetro η é conhecido como **taxa de aprendizagem** e determina o tamanho do passo a ser dado na direção de minimização.

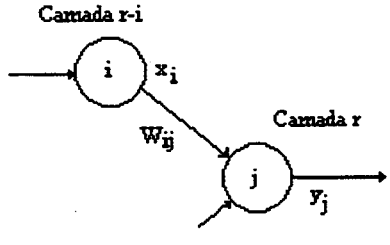


FIGURA 4.7 - Conexão entre Dois Neurônios

$$\Delta w_{ji} = -\eta\frac{\partial E_p}{\partial w_{ji}} \tag{4.16}$$

A derivada da equação (4.16) é calculada segundo o principio da regra da cadeia.

$$f'(net_j) \tag{4.17}$$

onde: $\frac{\partial net_j}{\partial w_{ji}} = y_i$

A outra derivada da equação (4.17) será representada pelo parâmetro δ_j, resultando na equação (4.18).

$$\delta_j = -(d_j - y_j) \cdot f'(net_j) \tag{4.18}$$

O valor de δ_j é obtido à partir de (4.19). A derivada $f'(net_j)$ é obtida utilizando a função de ativação adotada, neste caso a função sigmoidal (4.5), resultando na equação (4.20).

$$\Xi(W) = -\nabla J(W) = -\frac{1}{P}\sum_P \frac{\partial E_p(W)}{\partial W} \tag{4.19}$$

$$\delta_j = -(d_j - y_j) \cdot y_j \cdot (1 - y_j) \tag{4.20}$$

Mas, o resultado obtido em (4.20) serve apenas para o cálculo de variações dos pesos conectados aos neurônios da camada de saída. Para os pesos das camadas internas utiliza-se a retropropagação do erro da camada de saída, uma vez que não é possível obte-lo diretamente como em (4.20). O parâmetro δ_j para as camadas internas é representado pela equação (4.21).

$$\delta_j = -\frac{\partial E}{\partial y_j} \cdot y_j \cdot (1 - y_j) \tag{4.21}$$

$$-\frac{\partial E}{\partial y_i} = -\sum_k \frac{\partial E}{\partial net_k} \cdot \frac{\partial net_k}{\partial y_j}$$

$$= -\sum_k \frac{\partial E}{\partial net_k} \cdot \frac{\partial \sum w_{km} \cdot y_m}{\partial y_j}$$

$$= -\sum_k \frac{\partial E}{\partial net_k} \cdot w_{km}$$

$$-\frac{\partial E}{\partial y_i} = -\sum_k \delta_k \cdot w_{km} \qquad (4.22)$$

Substituindo-se (4.22) em (4.21) tem-se a equação (4.23), que é utilizada no cálculo do parâmetro δ_j das camadas internas, onde k representa o somatório sob os neurônios da camada posterior à camada do neurônio j.

$$\delta_j = y_j(1-y_j)\cdot \sum_k \delta_k \cdot w_k \qquad (4.23)$$

Os pesos são atualizados após a apresentação completa do conjunto de parâmetros através da regra delta generalizada [27] conforme a equação (4.24). O índice k refere-se ao instante de amostragem, logo, k-1 refere-se ao instante anterior.

$$w_{ji}(k) = w_{ji}(k-1) + \Delta w_{ji}(k-1) \qquad (4.24)$$

Para aumentar a velocidade de treinamento sem induzir oscilações, Rumelhart [27] introduziu uma modificação em (4.24) adicionando um termo α chamado **momento** que nada mais faz do que tornar (4.24) um filtro passa baixas que pretende atenuar as oscilações dos pesos em torno da direção mínima típicas no Backpropagation.

$$w_{ji}(k) = w_{ji}(k-1) + \Delta w_{ji}(k-1) + \alpha \Delta w_{ji}(k-2) \qquad (4.25)$$

É importante salientar que os thresholds são tratados como pesos quaisquer do ponto de vista do algoritmo, com a diferença que sua entrada é sempre unitária.

Os seguintes passos portanto constituem um treinamento utilizando o backpropagation:

1) Selecionar o próximo par do conjunto de treinamento, aplicando o vetor de entrada para a entrada da rede.

2) Calcular a saída da rede.

3) Calcular o erro entre a saída da rede e a saída desejada (erro E_p).

4) Calcular ΔW_p , acumulando os resultados para cada vetor de treinamento à fim de obter-se ao final da apresentação do conjunto completo de treinamento o valor de ΔW_t.

5) Repetir do passo 1 até a apresentação completa do conjunto de treinamento.

6) Atualizar os pesos da rede.

7) Verificar se o Erro total desejado foi alcançado, se não, retomar novamente desde o início.

Os passos acima referem-se a um treinamento visando minimizar E_t . Pode-se no entanto minimizar E_p , o que implicaria na atualização dos pesos após o passo 4.

4.4.3. Limitações do algoritmo Backpropagation

Apesar de sua facilidade de implementação e seu baixo custo computacional, o algoritmo de treinamento Backpropagation em sua forma original possui limitações que por vezes desencorajam a sua utilização e obrigam os pesquisadores a buscar alternativas em alguma técnica auxiliar. As principais limitações do algoritmo são as seguintes:

a) **Tempo de treinamento** - O tempo de treinamento no algoritmo Backpropagation é considerado muito longo para muitas aplicações práticas. Este tempo de treinamento longo deve-se principalmente ao método do gradiente descendente. O gradiente é um ponto extremamente local em relação a mudança ótima na função. Mesmo considerando pequenas distâncias o gradiente pode apontar para direções completamente diferentes. Este comportamento indesejado demanda muito mais tempo de treinamento do que o caso de uma rota direta de convergência sob uma direção ótima. Em [22] mostra-se que o comportamento típico do método gradiente descendente

é uma trajetória no espaço dos pesos de zigue zague em torno da direção ótima de minimização. A forma que o Backpropagation tem de contornar este problema é através do ajuste da taxa de aprendizado e do momento, que mesmo corretamente ajustadas muitas vezes não são capazes de acelerar suficientemente o treinamento.

b) **Ajuste crítico dos parâmetros η e α** - Quando se ajusta a taxa de aprendizado η deve-se prevenir dois aspectos, primeiro não deixa-la muito alta, pois tal fato acarreta em um comportamento oscilatório em relação a variação dos pesos a cada iteração [22, 17], e segundo também não subdimensionar o seu valor, o que implicaria em uma velocidade menor de convergência ao treinamento. Portanto o ajuste de η não é simples e em muitos casos uma taxa de aprendizado ótima é inviável. Ajusta-se um η que responde bem a uma etapa do treinamento mas começa a provocar oscilações acentuadas em uma etapa posterior ou então tem-se um η que não provoca oscilações mas que torna muito lento o aprendizado. Diretamente ligado ao ajuste de η está o ajuste do parâmetro α. Tal parâmetro quando utilizado tende a contribuir para a diminuição da oscilação dos pesos durante o treinamento, mas em muitos casos não se consegue um ajuste de α que atenue suficientemente as oscilações em todas as etapas do mesmo. Alguns pesquisadores, como em [39, 9], definem valores típicos de η e α, mas o ajuste destes parâmetros, como em [19], é na maioria das vezes feito por tentativa e erro, um processo por vezes longo e difícil caso se busque valores ótimos para os parâmetros.

c) **Mínimos locais** - Os algoritmos do tipo gradiente descendente são sensíveis ao problema dos mínimos locais, que pode ser melhor visualizado considerando a função de erro apresentada na figura 4.8, onde o ponto A representa um mínimo local e o ponto B representa o mínimo global da função de erro. O problema dos mínimos locais pode ou não ser crítico para o treinamento de uma RNA, pois o objetivo geral do treinamento não é alcançar o mínimo global e sim satisfazer um erro mínimo pré definido como suficiente para a tarefa desejada. Portanto soluções que levam à mínimos locais muitas vezes são aceitas sem a necessidade de contornar o problema. Uma observação importante é que muitas vezes um mínimo local pode ser confundido com uma região de planície da função de erro, conforme mostrado na figura 4.9. As regiões de planície tendem muitas vezes a tornar lento o treinamento, pois sendo o gradiente descendente local em relação a direção ótima de minimização ele causa pouca progressão no

treinamento quando nestas regiões. O treinamento pode ser inclusive interrompido devido a derivada das funções de ativação tornar-se muito próxima de zero, não retropropagando portanto o erro da camada de saída para as camadas escondidas, problema que particularmente foi tratado em [9] com a proposta de adicionar um off-set de 0.1 à derivada da função de ativação conforme a equação (4.26), solução esta que também é adotada neste trabalho (Equação 4.26).

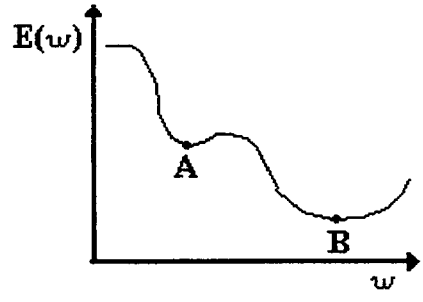


FIGURA 4.8 - Mínimos Locais e Mínimos Globais

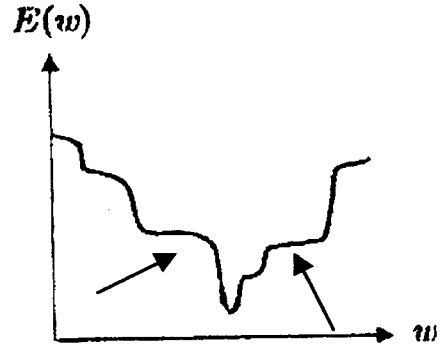


FIGURA 4.9 - Regiões de Planície

$$f'(net) = y_j(1 - y_j) + 0.1 \qquad (4.26)$$

Os problemas do algoritmo Backpropagation tem levado muitos autores a buscar alternativas ou mesmo algoritmos de treinamento que busquem outros princípios de minimização do erro em uma RNA feedforward multicamadas que não o método do gradiente descendente. Em [15] Jondarr faz um excelente trabalho ao catalogar todos os algoritmos mais conhecidos da literatura que propõem mudar ou acrescentar algum

aspecto novo ao Backpropagation que venha a melhorar a sua velocidade de treinamento. O autor chama o conjunto de algoritmos de família Backpropagation, pois todos eles baseiam-se no algoritmo original. Para se ter uma idéia foram catalogados neste trabalho 61 algoritmos, subdivididos de acordo com a estratégia de mudança proposta por seus autores. Em [22] são apresentados dois algoritmos que visam resolver o problema dos mínimos locais, o algoritmo de otimização genética e o algoritmo simulated annealing, ambos não baseados no método do gradiente descendente. Enfim, existem um razoável número de propostas e pode se dizer que diariamente surge alguma nova em relação ao algoritmo Backpropagation. Alguns autores, como em [25, 29], se aventuram a comparar diferentes propostas, afirmando que uma ou outra é melhor. Como em [9], afirma-se que este tipo de comparação é difícil de ser feita, pois o desempenho do algoritmo de treinamento esta muito ligado ao processo que esta sendo aprendido pela RNA. Portanto, ao escolher uma solução deve-se ter em mente o que melhor resolve o problema em questão, talvez considerando aspectos como a relação custo / benefício que cada estratégia oferece e evidentemente a eficiência da solução.

4.5 - O ALGORITMO QUICKPROPAGATION

O algoritmo Quickpropagation foi inicialmente proposto por Fahlman em [9] e trata-se de uma série de modificações heurísticas feitas no Backpropagation original. Em [29] e em [25] este algoritmo é citado como um dos mais eficientes quando se trata de aumentar a velocidade de treinamento do Backpropagation e com este objetivo explora-se as suas potencialidades neste trabalho.

O ponto de partida para o Quickpropagation é o método de Newton. Este método utiliza os três primeiros termos da série de Taylor no ponto corrente para descrever o contorno do espaço de pesos e prever a direção do mínimo.

A série de Taylor em um ponto corrente do espaço de pesos W, dado um modelo m_t para o erro E em qualquer nova direção W + S é dada pela equação (4.27).

$$E(W+S) \approx m_t(W+S) = E(W) + \nabla E(W)^T S + \frac{1}{2} S^T \nabla^2 E(W) S \qquad (4.27)$$

Resolve-se (4.27) para achar um gradiente zero para algum passo final N, onde S^N é a direção de Newton :

$$\nabla m_t(W + S^N) = 0$$

O sistema linear a ser resolvido também pode ser expresso conforme (4.28).

$$\nabla^2 E(W) S^N = -\nabla E(W) \tag{4.28}$$

A matriz $\nabla^2 E(W)$ é também chamada de Hessiana. Sendo a Hessiana positiva definida, uma iteração portanto será necessária para alcançar o mínimo.

O método de Newton não é usado em aplicações práticas devido a dificuldade do cálculo da Hessiana, no entanto a sua idéia básica é ponto de partida para vários outros métodos que buscam aproximar o valor desta Hessiana.

No Quickpropagation, a Hessiana é aproximada recursivamente mediante duas hipóteses simplificadoras:

H4.1. A curva Erro x Peso para cada peso pode ser representada por uma parábola de inclinação ascendente.

H4.2. As mudanças na derivada de um peso não são dependentes das mudanças dos demais pesos.

O resultado da aproximação é a equação (4.29) que é bastante simples de ser implementada e usa somente informações locais sobre os pesos que estão sendo atualizados.

$$\Delta w_{ij} = \frac{\dfrac{\partial E}{\partial w_{ij}}}{\dfrac{\partial E}{\partial w_{ij}}(t-1) - \dfrac{\partial E}{\partial w_{ij}}} \cdot \Delta w_{ij}(t-1) \tag{4.29}$$

A aproximação da Hessiana em (4.29) é considerada bastante grosseira do ponto de vista matemático, mas junto com outras Heurísticas inseridas por Fahlman ela tem um comportamento bastante eficiente do ponto de vista de aumentar a velocidade de treinamento. Estas Heurísticas partem da análise do comportamento de (4.29) em três diferentes situações:

S4.1. **A derivada corrente é menor que a derivada anterior, mas na mesma direção**. A mudança de pesos irá seguir a mesma direção anterior. O passo pode

ser grande ou pequeno e dependerá de quanto a derivada foi reduzida pelo passo anterior.

S4.2. **A derivada corrente está em direção oposta a derivada anterior**. Significa que cruzou-se um mínimo e se está agora no lado oposto do vale. Neste caso o próximo passo conduzirá para algum lugar entre a posição anterior e a atual.

S4.3. **A derivada atual tem a mesma direção da anterior, mas é do mesmo tamanho ou maior em magnitude**. Neste caso surge um problema. Se (4.29) for rigorosamente aplicada, o passo tenderá ao infinito ou a função de erro tenderá a um máximo local.

A solução encontrada para a situação S4.3 é a introdução de um fator μ, chamado de máximo fator de crescimento. Determina-se que nenhum passo pode ser maior em magnitude que μ vezes o passo anterior para o dado peso. Se isto ocorrer, substitui-se o passo atual por μ vezes o passo anterior. Apesar de criar-se mais um fator a ser sintonizado, afirma-se em [9] que a sintonia de μ, apesar de depender do processo à ser aprendido, não é crítica. O autor sugere o valor de μ de 1.75 e afirma que este valor funciona bem para a maioria dos processos.

O Quickpropagation é um algoritmo que muda seus pesos baseado no que ocorreu na mudança de pesos anterior. Necessita-se portanto alguma forma de dar início a esse processo, uma vez que em um primeiro instante $w_{ji}(t - 1) = 0$. Além disso, necessita-se também alguma maneira de reiniciar o processo mediante a ocorrência de um passo de tamanho zero.

A maneira mais eficiente de realizar esta "ignição" é somar a equação (4.29) com a derivada corrente multiplicada pela taxa de aprendizagem η, à semelhança do Backpropagation. Para prevenir o efeito indesejado na vizinhança de um mínimo, utiliza-se (4.29) sozinha mediante a situação S4.2.

Uma última Heurística ainda é necessária. Para prevenir que alguns processos ocasionem demasiado crescimento nos pesos, ocasionando erros de *overflow* durante o treinamento, um termo de decaimento d é adicionado à derivada calculada para cada peso, mantendo-os dentro de um valor aceitável, conforme a equação (4.30).

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial w_{ij}} + d \cdot w_{ij} \tag{4.30}$$

Neste trabalho particularmente não utilizou-se (4.30) porque o processo treinado não apresentou os problemas de crescimento excessivo de pesos citados em [9].

A versão do Quickpropagation utilizada neste trabalho foi extraída de [15]. Trata-se da versão que Veitch and Holmes utilizaram em seu trabalho e possui ligeiras modificações em relação a versão original de [9].

À seguir é apresentado o algoritmo Quickpropagation na forma na qual ele foi utilizado neste trabalho.

Algoritmo Quickpropagation

INICIO.

For each weight w_i

IF $\Delta w_{i-1} > 0$ THEN

IF $\frac{\partial E}{\partial w_i} > 0$ **THEN**

$$\Delta w_i = \eta \cdot \frac{\partial E}{\partial w_i}$$

IF $\frac{\partial E}{\partial w_i} > \frac{\mu}{1+\mu} \cdot \frac{\partial E}{\partial w_{i-1}}$ **THEN**

$$\Delta w_i = \Delta w_i + \mu \cdot \Delta w_{i-1}$$

ELSE

$$\Delta w_i = \Delta w_i + \frac{\frac{\partial E}{\partial w_i} \cdot \Delta w_{i-1}}{\frac{\partial E}{\partial w_{i-1}} - \frac{\partial E}{\partial w_i}}$$

ELSEIF $\Delta w_{i-1} < 0$ THEN

IF $\frac{\partial E}{\partial w_i} < 0$ **THEN**

$$\Delta w_i = \eta \cdot \frac{\partial E}{\partial w_i}$$

IF $\frac{\partial E}{\partial w_i} < \frac{\mu}{1+\mu} \cdot \frac{\partial E}{\partial w_{i-1}}$ **THEN**

$$\Delta w_i = \Delta w_i + \mu \cdot \Delta w_{i-1}$$

ELSE

$$\Delta w_i = \Delta w_i + \frac{\frac{\partial E}{\partial w_i} \cdot \Delta w_{i-1}}{\frac{\partial E}{\partial w_{i-1}} - \frac{\partial E}{\partial w_i}}$$

ELSE

$$\Delta w_i = \eta \cdot \frac{\partial E}{\partial w_i}$$

$$w_i = w_i + \Delta w_i$$

END

FIM.

4.6 - CONCLUSÃO

Redes neurais são ferramentas bastante poderosas para a utilização em sistemas de controle pois proporcionam o mapeamento de processos físicos utilizando somente exemplos de padrões de entrada e saída dos mesmos.

As redes neurais do tipo *feedforward* multicamadas possuem larga utilização e comprovada eficiência em sistemas de controle, por isso foram escolhidas para serem utilizadas neste trabalho. O algoritmo de treinamento Backpropagation é sempre o ponto de partida quando se fala em treinamento de RNA *feedforward* multicamadas, mas apesar de ser tão popular ele apresenta problemas como velocidade de treinamento, mínimos locais e dificuldade no ajuste dos seus parâmetros de treinamento. Dentro da literatura inúmeras soluções são propostas, mas mostra-se suficiente para os objetivos deste trabalho a adoção do algoritmo Quickpropagation. Tal algoritmo pretende aumentar a velocidade de treinamento do Backpropagation à um nível capaz de proporcionar a aplicação das RNA pretendida neste trabalho.

5 - CONTROLADORES ROBÓTICOS UTILIZANDO REDES NEURAIS

5.1 - INTRODUÇÃO

Conforme apresentado no capítulo 3, os controladores robóticos mais usuais possuem limitações de performance, principalmente em relação ao controle de seguimento de trajetória. As redes neurais são bons aproximadores de modelos matemáticos, conforme visto no capítulo 4. Neste capítulo utiliza-se estas potencialidades das RNA com o objetivo de melhorar a performance dos controladores usualmente utilizados na robótica e com isso propor novos controladores que podem servir como alternativa na resolução do problema do controle de robôs manipuladores.

Apesar de recente, a utilização de RNA no controle de robôs não chega a ser uma novidade. Vários trabalhos já foram publicados neste sentido [3, 4, 23, 30, por exemplo]. O que se pretende neste trabalho é aproveitar a experiência destes outros autores e apresentar os controladores do capítulo 3 acrescidos de RNA, analisando as vantagens e limitações deste processo.

5.2 - PROJETO DE UMA REDE NEURAL PARA REPRESENTAR A DINÂMICA INVERSA DE UM MANIPULADOR ROBÓTICO DE DOIS GRAUS DE LIBERDADE

5.2.1. Introdução

No capítulo 2 apresentou-se o modelo genérico de um manipulador com 2 graus de liberdade transportando uma carga desconhecida. Nesta seção projeta-se uma rede neural que aprenda a dinâmica inversa de um exemplo de robô manipulador contendo valores práticos aplicados a esse modelo. Mostra-se com isso o procedimento de projeto adotado neste trabalho e as dificuldades e aspectos teóricos necessários a se levar em conta dentro desse procedimento.

5.2.2. Um Exemplo de Robô Manipulador

O Robô Manipulador que será utilizado neste trabalho é o da Figura 2.6. A tabela 5.1 contém os parâmetros para o modelo. Tais parâmetros foram extraídos de

[10], que os obteve dos elos 2 e 3 do "exemplo padrão" apresentado em Trosh e Desoyer (1990) [40].

	Elo 1	Elo2	Efetuador + Carga
Comprimento do elo l_i (m)	0.45	0.20	
Massa m_i (kg)	100	25	m_3 (20kg nominal)
Momento de Inércia I (kgm^2)	6.25	0.61	$m_3 \times l_2^2$
Distância do centro de massa $l_{c,i}$ (m)	0.15	0.10	l_2
Inércia dos rotores J (kgm^2)	4.77	3.58	

TABELA 5.1 - Parâmetros do Robô Manipulador

A equação de movimento utilizada será a equação (2.33), que descreve o modelo rígido. Portanto são incluídas na equação as respectivas inércias dos rotores.

5.2.3. Obtenção do Conjunto de Pares de Treinamento.

Um dos passos mais importantes no treinamento de uma rede neural é definir o conjunto de treinamento, chamado aqui de *pares de treinamento,* e o conjunto de validação, que indicará se o treinamento realizado esta dentro do esperado. O primeiro fator considerado aqui foi o objetivo do treinamento, que no caso é fazer com que a RNA aprenda a dinâmica inversa do robô manipulador, ou seja, dado um conjunto de entrada U na rede, composto pelos parâmetros dinâmicos do robô que são as posições e suas derivadas no instante t, esta deverá apresentar na saída um conjunto Y, contendo os torques que seriam aplicados ao robô e resultariam no conjunto U. O treinamento é feito portanto em um processo inverso ao da equação do movimento (dinâmica direta) do Manipulador. Sabe-se que a equação do movimento é extremamente não linear, portanto a relação R: U→Y dependerá muito do tipo de movimento executado pelo manipulador, ou seja, a sua trajetória. Em [23] afirma-se que o domínio da relação R é a trajetória descrita pelo robô em questão. O autor utiliza um conjunto de trajetórias estocásticas que são controladas em malha fechada por um controlador PID de ganhos fixos. Em [3] um esquema semelhante é utilizado. Um conjunto de 4 trajetórias controladas em malha fechada por um dado controlador é utilizado para o

aprendizado da dinâmica direta de um robô manipulador de dois graus de liberdade. Para a escolha do conjunto de trajetórias utilizado, deve-se ter em mente a natureza dos movimentos que serão realizados pelo robô. Infelizmente, infinitas possibilidades não são possíveis de serem contempladas e portanto um conjunto finito e o mais genérico possível de trajetórias deve ser escolhido, sendo que estas devem estar a semelhança daquelas executadas pelo robô em seus trabalhos. Este aspecto pode figurar como uma grande limitação das RNA aplicadas ao controle de robôs, mas como se verá mais adiante, com um esquema suficiente robusto consegue-se bons resultados mesmo com trajetórias diferentes das do conjunto de treinamento, principalmente quando um treinamento on-line é utilizado.

As trajetórias utilizadas neste trabalho pretendem descrever a progressão dos ângulos q_1 e q_2. Portanto, estas trajetórias pertencem ao espaço de juntas e são descritas por dois polinômios de 5ª ordem da mesma forma que em [13] e em [24]. O objetivo de se usar polinômios de 5ª ordem é obter uma trajetória suave de aceleração limitada e portanto de torque limitado. O esquema de geração do conjunto de treinamento é descrito pela figura 5.1. Um controlador PD junta por junta controla o robô em malha fechada. A trajetória desejada é aplicada na entrada do controlador que faz o seu seguimento. As variáveis de entrada e saída da rede são colhidas na saída e na entrada do modelo respectivamente. Os ganhos do controlador devem proporcionar o menor erro possível para que as trajetórias geradas estejam próximo das trajetórias desejadas. Tal procedimento minimiza a influência dos ganhos do controlador no treinamento.

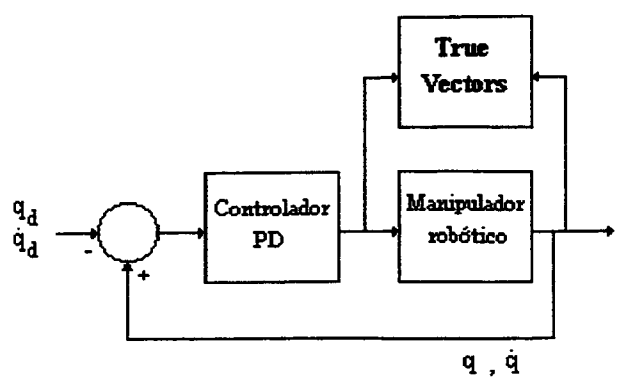


FIGURA 5.1 - Geração das Trajetórias de Treinamento

	q_1 (inicial)	q_2 (inicial)	q_1 (final)	q_2 (final)	tempo (s)
trajetória 1	-90^0	0^0	-67.5^0	22.5^0	1.0
trajetória 2	-90^0	0^0	-70^0	40^0	1.5
trajetória 3	-90^0	0^0	-50	20^0	1.5
trajetória 4	-90^0	0^0	-50^0	40^0	2.0

TABELA 5.2 - Trajetórias que compõem o Conjunto de Treinamento

q_1 (ini.)	q_2 (ini.)	q_1 (interm.)	q_2 (interm.)	q_1 (final)	q_2 (final)	tempo (s)
-90^0	0^0	-70^0	30^0	-90^0	0^0	2.5

TABELA 5.3 - Trajetória que compõe o Conjunto de Validação

As tabelas 5.2 e 5.3 mostram as trajetórias que compõem os conjuntos de treinamento e validação. Escolheu-se como conjunto de validação uma trajetória que tem um perfil diferente das demais pertencentes ao conjunto de treinamento. O manipulador parte de uma posição inicial até uma posição intermediária em uma trajetória cujo tempo é de 1 segundo, permanece 0.5 segundos em regulação e leva mais 1 segundo em uma trajetória de retorno a posição inicial.

A definição do tamanho do conjunto de treinamento é outro aspecto importante. O tamanho da RNA utilizada está diretamente relacionado ao tamanho do conjunto de treinamento [22 , 19]. O conjunto de treinamento deve ter uma quantidade de pares de treinamento suficientes à representação do processo, neste caso, da trajetória. Se muitos parâmetros forem escolhidos corre-se o risco de uma generalização pobre do comportamento do manipulador para uma trajetória não pertencente ao conjunto de treinamento, além da necessidade de uma RNA maior, capaz de aprender este conjunto. Em contrapartida, se o conjunto de treinamento é subdimensionado, a RNA não aprende corretamente todos os aspectos necessários do comportamento do manipulador nas dadas trajetórias.

Neste trabalho adotou-se definir inicialmente quantas amostras caracterizam o processo em cada trajetória individualmente. Para tanto ensaiou-se separadamente cada trajetória, treinando uma RNA que aprendesse o comportamento do manipulador ao longo da mesma e testando com algum conjunto de validação. Após estes ensaios definiu-se que 10 vetores igualmente espaçados são suficientemente representativos para cada trajetória do conjunto de treinamento. Adotou-se usar 30 vetores para representar cada trajetória descrita na tabela 5.2, totalizando um conjunto de treinamento composto de 120 vetores.

Como componentes do vetor de entrada, pertencente ao conjunto U, utilizou-se as posições, velocidades e acelerações instantâneas nas juntas do robô e como componentes do vetor de saída os torques nas juntas . Observa-se, baseando-se em outros trabalhos como [30] e [23], serem estas variáveis suficientes para treinar uma RNA que mapeie a dinâmica inversa do manipulador.

5.2.4. Algoritmo de treinamento e tamanho da rede.

O algoritmo de treinamento utilizado dependerá da necessidade do treinamento e do processo aprendido. Como o objetivo do trabalho é controlar o manipulador, e para tanto necessita-se de uma RNA capaz de ser treinada on-line, a velocidade de treinamento torna-se crucial. Optou-se portanto pela utilização do algoritmo Quickpropagation, apresentado na seção 4.5. A figura 5.2 mostra a comparação da velocidade de treinamento entre os algoritmos Backpropagation e Quickpropagation para uma RNA de três camadas com seis neurônios na entrada, seis neurônios na camada escondida e dois neurônios na saída no treinamento do processo em questão.

Outro aspecto analisado na definição do algoritmo de treinamento foi o problema dos mínimos locais. Após alguns ensaios verificou-se que, apesar do problema acontecer, ele não é crítico, não necessitando de uma estratégia que especificamente resolva o problema. O mínimo encontrado na maioria dos casos encontra-se dentro do satisfatório para o treinamento. Adotou-se como forma de encontrar a melhor solução treinar cada Rede testada no mínimo 10 vezes com diferentes inicializações para os pesos, sempre utilizando inicializações randômicas numa distribuição -0.01 à 0.01 para cada peso e off-set.

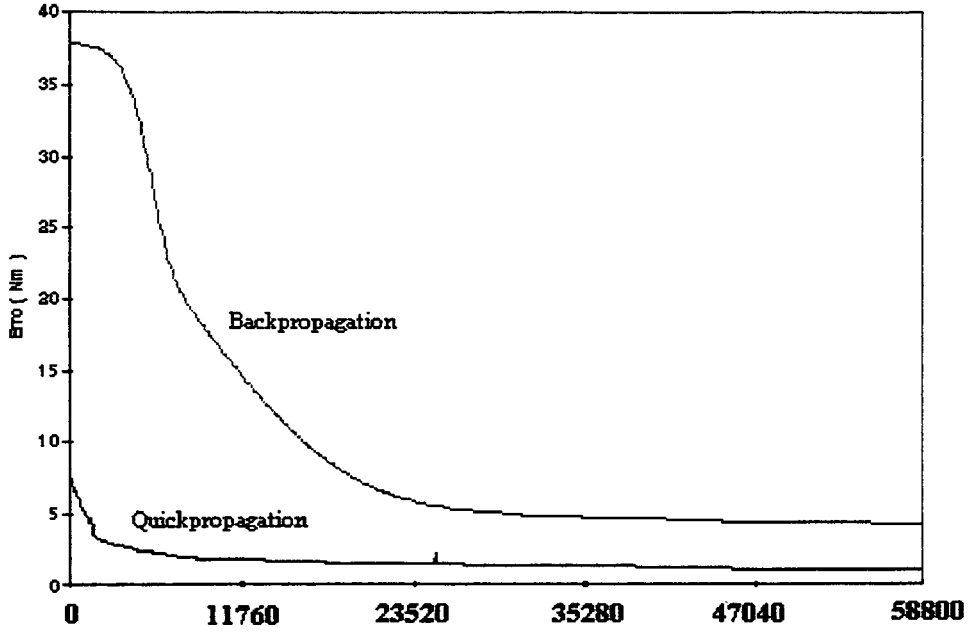


FIGURA 5.2 - Comparação Entre os Algoritmos Backpropagation e Quickpropagation

O tamanho da Rede, ou seja, o número de camadas escondidas e a quantidade de neurônios em cada camada é também um importante aspecto a ser definido dentro do projeto de uma RNA. Algumas regras básicas existem. Em [22], por exemplo, o autor propõe um método analítico, mais heurístico do que matemático, para a definição do tamanho de uma RNA. O método é chamado de regra da pirâmide geométrica e consiste basicamente em definir o tamanho da rede de acordo com uma progressão geométrica da entrada para a saída. A rede tem portanto poucos neurônios na entrada e uma quantidade maior na saída. O número de neurônios na camada intermediária, considerando uma RNA de três camadas, é definido pela equação (5.1). Segundo o autor o método funciona bem contando que a RNA não tenha poucas entradas e poucas saídas e o processo não seja complexo, caso contrário o método pode resultar em uma subestimação do número de neurônios na camada escondida.

$$n^{\circ}\text{ neurônios camada escondida} = \sqrt{m \cdot n}\text{, onde } m = \text{entradas}, n = \text{saídas} \qquad (5.1)$$

Uma afirmação que figura em muitos trabalhos [26, 17, 22, 19] é que para os processos encontrados no mundo real não existe necessidade de usar-se uma RNA que possua mais de quatro camadas e que a grande maioria deles pode ser treinada com uma rede de apenas três camadas. Esta constatação reduz o trabalho a escolha entre uma arquitetura de 3 ou 4 camadas e a definição do número de neurônios em cada uma delas.

Dois aspectos importantes estão estreitamente ligados a escolha do tamanho da rede. Um deles é a complexidade do processo e o outro é o tamanho do conjunto de treinamento. Em [19] e em [22] chama-se atenção que o número de parâmetros livres de uma RNA, no caso os pesos, não deve ser muito maior que o numero de pares de treinamento do conjunto de treinamento utilizado. A escolha de uma rede com grande número de parâmetros livres tende a fazer com que detalhes não importantes do conjunto de treinamento sejam aprendidos, ocasionando o sobretreinamento. Uma relação ótima entre estes aspectos deve ser buscada pelo projetista, ou seja, uma rede nem tão pequena que não consiga aprender o conjunto de treinamento e nem tão grande que ocasione o sobretreinamento. Uma relação de dois vetores de treinamento para cada peso é considerada razoável neste trabalho. O que aconselha-se aqui é iniciar o treinamento com uma rede pequena e ir aumentando-a gradativamente, até atingir um tamanho onde se tenha um erro de validação razoável sem sobretreinamento. Muito embora pareça não ser necessário adicionar uma segunda camada escondida, este procedimento pode em certos casos diminuir o erro de treinamento e melhorar a validação. Em [19] analisa-se mais detidamente este aspecto. Uma RNA previamente conhecida é usada como processo a ser aprendido. Mostra-se que redes de duas camadas com grande número de parâmetros livres apresentam melhor resultado do que uma rede com o mesmo tamanho que a original considerando um número fixo de iterações de treinamento. Pode-se dizer portanto, como em [22], que adicionar mais uma camada escondida por vezes melhora o desempenho da rede, aumentando a velocidade de treinamento e a capacidade de generalização da rede. Neste trabalho este procedimento é confirmado com sucesso, como será visto adiante. Outro aspecto associado ao tamanho da rede é o problema dos mínimos locais. Em [17] afirma-se que redes com grande número de parâmetros livres podem ser mais sensíveis ao problema dos mínimos locais que redes menores. No treinamento realizado neste trabalho notou-se que em redes maiores o desvio padrão dos erros oriundos de diferentes sessões de treinamento, sob um mesmo conjunto de pares de treinamento, era maior quanto maior a rede, o que pode indicar uma sensibilidade maior ao problema dos mínimos locais.

Acredita-se que por enquanto não exista uma maneira analítica de definir o tamanho de uma RNA. Na maioria das vezes a tentativa e erro é o único processo viável

de escolha. As regras heurísticas citadas anteriormente são baseadas na experimentação e fornecem algumas diretrizes básicas, mas o processo é que vai dizer qual a melhor rede a ser usada.

A tabela 5.4 resume o procedimento de escolha do tamanho da rede utilizada para treinar o robô manipulador apresentado neste trabalho. São comparados os erros em relação às trajetórias pertencentes ao conjunto de treinamento e também é analisado o erro em relação ao conjunto de validação. Cada um destes 5 conjuntos possui 1000 pontos de amostragem. Dez sessões de treinamento foram feitas para cada RNA, cada uma com diferentes conjuntos de pesos iniciais com distribuição randômica. O resultado apresentado corresponde ao menor erro de validação das dez sessões. Para a escolha da rede foram considerados os aspectos menor erro de validação, menor erro de treinamento, velocidade de treinamento e menor tamanho possível. A rede que melhor preencheu estes aspectos foi uma RNA de 4 camadas, com 2 camadas escondidas de 4 neurônios cada. Esta RNA além de apresentar um erro de validação baixo, em unidades de torque, possui velocidade de treinamento maior do que uma rede de 3 camadas. Observa-se que o aumento de neurônios à partir daí, apesar de melhorar a performance até certo ponto, acrescenta maior carga computacional e portanto não é vantajoso. Outros tamanhos de rede também foram testados, mas a tabela 5.4 resume os melhores resultados obtidos.

Se a rede é treinada somente off-line considera-se que qualquer uma das configurações acima poderia preencher satisfatoriamente as exigências de desempenho. Porém em um treinamento on-line a rede será constantemente retreinada sob diferentes conjuntos de treinamento. É fundamental portanto que o tamanho da RNA escolhido torne esta o menos sensível possível ao problema dos mínimos locais e do sobretreinamento. Uma maneira adotada neste trabalho de analisar este aspecto foi considerar o desvio padrão do erro de treinamento e do erro de validação em relação ao conjunto de 10 sessões de treinamento realizadas. Considerou-se que um desvio padrão baixo para o erro de treinamento indica que a rede é menos sensível ao problema dos mínimos locais, uma vez que o treinamento depende menos do conjunto de pesos inicial. Um desvio padrão baixo para o erro de validação associado a um erro de validação mínimo também baixo indica menor ocorrência do sobretreinamento durante as sessões. Tudo isso, apesar de heurístico, também ajuda a selecionar o melhor tamanho

para a RNA. A tabela 5.5 mostra os desvios padrão para cada tamanho de rede testado. Novamente conclui-se que a melhor opção seria a configuração com 4 camadas e 4 neurônios em cada camada escondida, pois ela apresenta os menores valores de desvio padrão.

Rede	Erro de treinamento	Erro de validação
6-6-2	1.29038	2.1155
6-8-2	0.74022	1.943
6-10-2	1.06772	1.56
6-20-2	1.1296	2.1775
6-4-4-2	0.6763461	1.55
6-5-3-2	0.7009214	1.09
6-6-3-2	0.6361004	1.2715
6-6-6-2	0.5274	1.361
6-8-8-2	0.82679	1.1685

TABELA 5.4 - Erros de Treinamento e Validação

Rede	desvio trein.	desvio valid.
6-6-2	0.1504	0.7166
6-8-2	0.1785	0.4796
6-10-2	0.2849	0.7623
6-20-2	0.3293	0.2138
6-4-4-2	0.1294	0.2058
6-5-3-2	0.2106	1.8200
6-6-3-2	0.3085	0.9565
6-6-6-2	0.2826	1.3798
6-8-8-2	0.2123	11.015

TABELA 5.5 - Desvios Padrão em Relação às Sessões de Treinamento

O exemplo de processo apresentado nesta seção trata de um modelo simples de manipulador de dois graus de liberdade. No entanto para modelos mais complexos, de seis graus de liberdade, naturalmente a carga computacional cresce, mas o procedimento de projeto pode ser conduzido de maneira semelhante. Algumas alternativas podem ser buscadas para contornar o problema do tamanho da rede, como em [23], onde o autor divide o problema em várias redes, uma para cada junta, considerando em cada rede a influência das demais juntas.

5.3 - CONTROLADOR BASEADO NA DINÂMICA INVERSA UTILIZANDO REDES NEURAIS

O controlador baseado na dinâmica inversa visto na seção 3.2 é uma estratégia de controle bastante eficiente, uma vez que pode tornar linear e desacoplado um sistema extremamente não linear como é o manipulador robótico. A dificuldade de obter-se o modelo matemático de um robô manipulador constitui sua principal limitação. Grandezas difíceis de mensurar como o atrito, posição do centro de massa, momento angular, etc, fazem com que as aproximações pesquisadas para determinados robôs sejam imperfeitas. Além da dificuldade de estimação do modelo do robô existe também dificuldade em calcular a dinâmica inversa em tempo real, o que deve ser feito numericamente através de algoritmos como o de Luh-Walker-Paul [2]. Em [6] é tratado um manipulador industrial muito popular chamado PUMA 560. Várias propostas de representação matemática do modelo são apresentadas e algumas diferem entre si em aspectos importantes, o que mostra a dificuldade da obtenção do modelo exato mesmo em manipuladores extremamente estudados. Tendo-se em mãos uma aproximação razoável e um algoritmo de cálculo da dinâmica inversa rápido o suficiente ainda pode-se utilizar com sucesso o controlador baseado na dinâmica inversa. Situações onde não se tem condições de obtenção do modelo do robô controlado entretanto inviabilizam o uso desta estratégia.

Como foi visto no capítulo 4, uma RNA é capaz de calcular a dinâmica inversa de um robô manipulador somente com amostras das entradas e das saídas do mesmo ao longo do seguimento de um conjunto de trajetórias pré definidas.

O controlador baseado na dinâmica inversa utilizando redes neurais trabalha com o mesmo principio de funcionamento do controlador da seção 3.2, no

entanto a equação (3.3) é substituída pela equação (5.2), que representa uma aproximação da dinâmica inversa do manipulador obtida através do treinamento de uma RNA.

$$\tau = R(U, W) \tag{5.2}$$

$$U = [q, \dot{q}, (\ddot{q}_d - u)]^T$$

W = matriz de pesos.

O ganho u, a exemplo da seção 3.2 é um controlador PD ou PID como o da equação (3.5), ou seja :

$$u = -K_p\tilde{q} - K_d\dot{\tilde{q}} - K_i\int \tilde{q}\, dt$$

A figura 5.3 representa o controlador abordado nesta seção em uma malha fechada de controle. A operação é semelhante ao controlador da seção 3.2, no entanto a RNA assume o papel do cálculo da dinâmica inversa, que antes era matematicamente calculada com base na estimativa do modelo do robô.

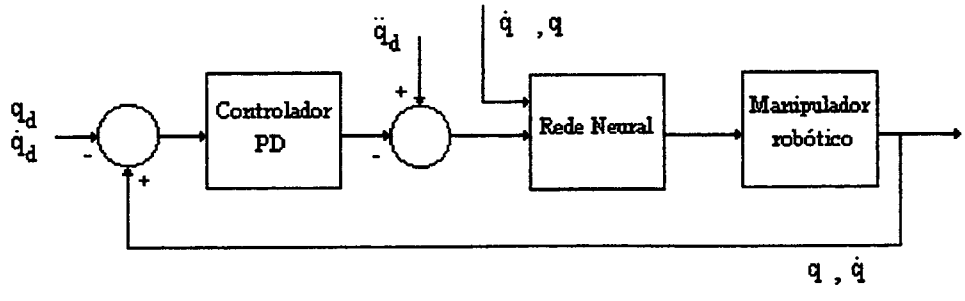


FIGURA 5.3 - Controlador Baseado na Dinâmica Inversa Utilizando RNA

Analisando o bloco neural em separado tem-se que entradas da RNA são o vetor de posições atuais q , que terá dimensão igual ao número de juntas do robô, e sua derivada primeira, que é o vetor das velocidades atuais $\dot{q}$. Um terceiro conjunto de entradas é obtido através do vetor das acelerações desejadas e da entrada u, que é oriunda do sinal de controle obtido na saída de um controlador PD ou PID. As entradas aplicadas a RNA produzem um vetor de torques de saída com a mesma dimensão n das juntas.

Até o presente momento não se tem uma compreensão clara de como trabalham os pesos da RNA. Tal dificuldade não permite estabelecer condições necessárias ou suficientes para provar que mediante um erro E_t limitado para o treinamento se garanta estabilidade em malha fechada para o sistema. Torna-se necessário avaliar o controle em situação de operação para definir o quanto ele é confiável em manter estável o sistema.

Neste trabalho não obteve-se bons resultados com o controlador baseado na dinâmica inversa utilizando RNA. Acredita-se que este fato deve-se a influência que os erros de seguimento de trajetória tem nas posições e velocidades atuais inseridas na entrada da RNA previamente treinada para representação da dinâmica inversa. Em alguns casos, principalmente naqueles em que a trajetória desejada era diferente do conjunto de treinamento, a RNA levou o sistema a instabilidade, certamente pelo fato de não ser capaz de generalizar de maneira satisfatória trajetórias tendo as posições e velocidades atuais como entradas, pois estas carregam a influência dos erros de seguimento de trajetória. Como o controlador baseado na dinâmica inversa exige boas estimações para esta dinâmica inversa, sua utilização ficou bastante limitada

5.4- CONTROLADOR TIPO TORQUE COMPUTADO UTILIZANDO REDES NEURAIS

O controlador tipo torque computado, conforme visto na seção 3.4 é bastante semelhante ao controlador baseado na dinâmica inversa, com a diferença fundamental de utilizar, ao invés das posições atuais medidas junto ao manipulador controlado, as posições desejadas previamente conhecidas. O controlador tipo torque computado utilizando RNA substitui as posições e velocidades atuais medidas junto ao processo presentes na estratégia de controle utilizada no controlador baseado na dinâmica inversa utilizando RNA pelas posições e velocidades desejadas. A estratégia de controle é descrita pela equação (5.3) e ilustrada pela figura 5.4.

$$\tau = R(U, W) \tag{5.3}$$

$$U = [q_d, \dot{q}_d, (\ddot{q}_d - u)]^T$$

W = matriz de pesos e threshold's.

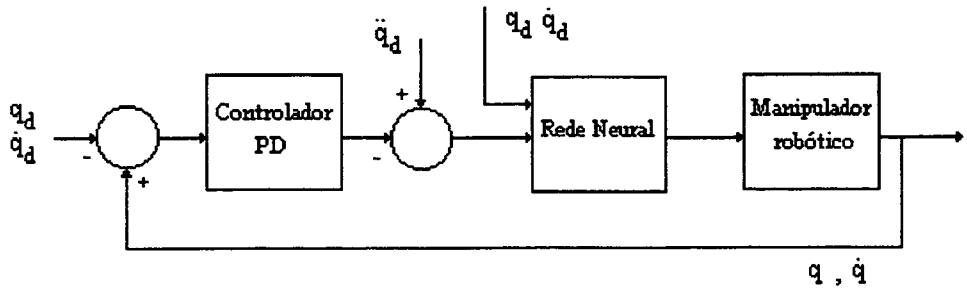


FIGURA 5.4 - Controlador Tipo Torque Computado Utilizando RNA

Obteve-se maior sucesso na utilização desta estratégia de controle do que na estratégia baseada na dinâmica inversa. Acredita-se que o fato das entradas da RNA não sofrerem a influência dos erros de seguimento de trajetória facilita à mesma oferecer um torque de controle melhor em suas saídas, que não leve o sistema à instabilidade. Resolveu-se portanto utilizar esta estratégia ao invés do controle baseado na dinâmica inversa.

Sabe-se que o controlador baseado na dinâmica inversa e o controlador tipo torque computado trabalham com a hipótese de cancelamento das não linearidades do sistema, o que exige boas estimações para a dinâmica inversa. Exige-se da RNA portanto uma boa performance, o que obriga o projetista a tomar cuidados maiores no procedimento de projeto da rede. O processo real de validação de uma RNA para estes controladores é portanto a sua própria utilização dentro da malha de controle, o que é uma dificuldade prática.

A figura 5.5 mostra a simulação de um seguimento de trajetória utilizando um controlador tipo torque computado usando RNA. A RNA neste exemplo é parcialmente treinada, ou seja, o treinamento não é considerado o ideal. O manipulador em questão é o mesmo da subseção 5.2.2. Pode observar-se pelos resultados que o sistema torna-se instável, exigindo um melhor treinamento da RNA a ser buscado pelo projetista. Ilustra-se portanto a afirmação anterior sobre a validação de uma RNA atuando neste tipo de controle.

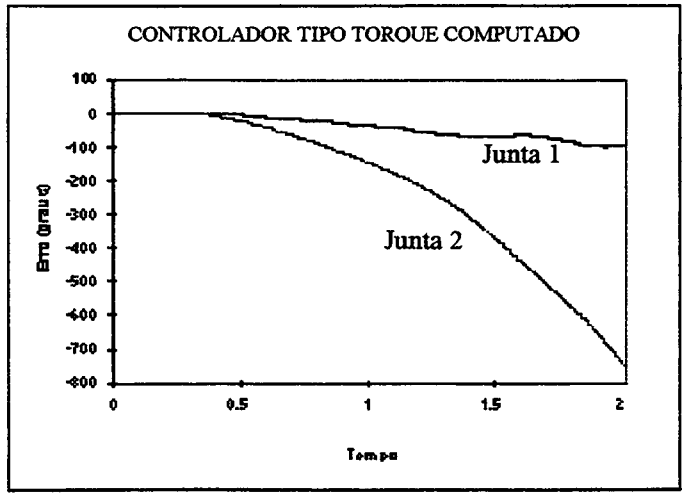


FIGURA 5.5 - Simulação de uma RNA Parcialmente Treinada Atuando em um Controlador Tipo Torque Computado

No capítulo 6 mostra-se simulações da performance de um controlador tipo torque computado utilizando RNA aplicado ao modelo da subseção 5.2.2.

5.5 - CONTROLADOR BASEADO NA PASSIVIDADE UTILIZANDO REDES NEURAIS

Na seção 3.4 apresentou-se o controlador baseado na passividade e foi provado que, devido ao mapeamento passivo existente entre as entradas e as saídas do modelo rígido de um manipulador robótico, é possível obter um sistema estável em malha fechada com um erro de seguimento assintóticamente estável através de um controlador constituído de uma parcela feedforward mais uma parcela PD, conforme a equação (3.9).

$$\tau = \hat{H}(q)\ddot{q}_r + \hat{C}(q,\dot{q})\dot{q}_r + \hat{g}(q) - K_D s$$

Nota-se que a parcela feedforward somada ao PD na verdade trata-se da dinâmica inversa do manipulador. O controlador baseado na passividade utilizando redes neurais substitui a parcela feedforward em (3.10) por uma estimação da dinâmica inversa do manipulador feita através de uma RNA, conforme a equação (5.3).

$$\tau = R(U, W) - K_D s \tag{5.3}$$

$$U = [\, q_d,\ \dot{q}_r,\ \ddot{q}_r \,]^T$$

W = matriz de pesos e threshold's.

$$\dot{q}_r = \dot{q}_d - \Lambda\tilde{q}$$

$$s = \dot{q} - \dot{q}_r = \dot{\tilde{q}} + \Lambda\tilde{q}$$

A figura 5.6 mostra a operação de controlador baseado na passividade utilizando RNA em uma malha fechada de controle. A substituição do cálculo da parcela feedforward por uma estimação por redes neurais introduz a modificação significativa no esquema.

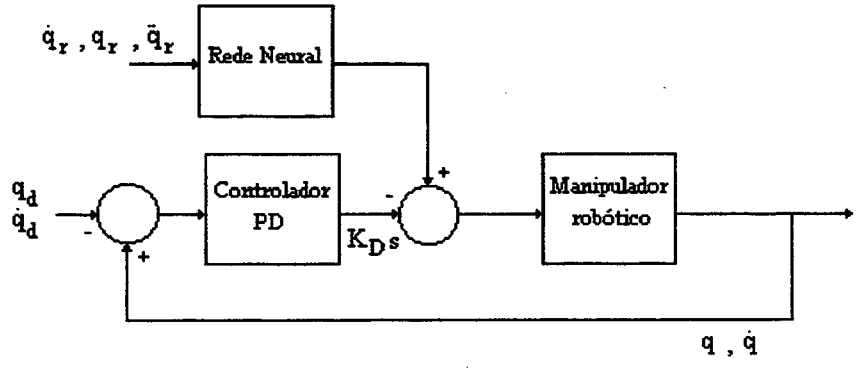


FIGURA 5.6 - Controlador Baseado na Passividade Utilizando Redes Neurais

Analisando o bloco neural em separado tem-se que entradas da RNA são o vetor de posições desejadas q_d , que terá dimensão igual ao número de juntas do robô, o vetor da velocidade de referência $\dot{q}_r$, calculada segundo a equação (3.10), e a sua derivada, ou seja a aceleração de referência $\ddot{q}_r$. As entradas aplicadas a RNA produzem um vetor com a mesma dimensão n das juntas que é somado ao sinal de controle produzido por um controlador PD, resultando no torque aplicado ao manipulador.

O controlador baseado na passividade possui uma importante vantagem em relação ao controlador baseado na dinâmica inversa e em relação ao controlador tipo torque computado que é o fato dele ser mais robusto que estes. Tal fato é bastante relevante quando se trata de substituir a dinâmica inversa presente nas duas propostas por uma RNA. A RNA no controlador baseado na passividade admite maiores erros de estimação sem que o processo torne-se instável. Tal afirmação pode ser ilustrada pela figura 5.6, que trata de uma simulação semelhante aquela representada pela figura 5.4 onde a mesma rede e o mesmo controlador PD foram utilizados. Percebe-se que o

sistema mantêm-se estável, ao contrário do caso anterior. O comportamento do sistema de controle torna-se bem menos crítico a medida que, com uma boa estimação inicial, ele tem boas chances de ser bem sucedido. Uma validação que aprove a rede dá maior segurança ao projetista do que no caso do controlador baseado na dinâmica inversa ou do controlador tipo torque computado, portanto. A impossibilidade de garantir estabilidade em malha fechada no entanto permanece pelas mesmas razões apresentadas na seção 5.3.

No capítulo 6 mostra-se simulações da performance de um controlador baseado na passividade utilizando RNA aplicado ao modelo da subseção 5.2.2.

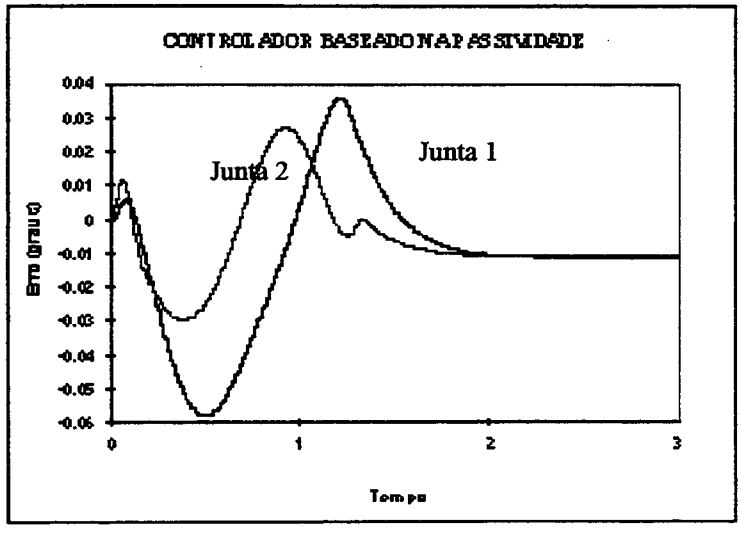


FIGURA 5.7 - Simulação de uma RNA Parcialmente Treinada Atuando em um Controlador Baseado na Passividade

5.6 - UM ESQUEMA DE ATUALIZAÇÃO ON-LINE PARA A REDE NEURAL

Um manipulador robótico movendo-se ao longo de uma determinada trajetória, como visto anteriormente, é um processo cujo comportamento é determinado por variáveis não lineares. Um mapeamento neural representa uma relação R que mapeia um conjunto U de situações de interesse em um conjunto Y de saída desejado. Dentro deste conjunto (U, Y) algumas amostras são retiradas para o treinamento e o restante é generalizado pela RNA. Qualquer que seja a arquitetura adotada para a RNA, no entanto, ela não é capaz de lidar com infinitas possibilidades de mapeamento entre U e Y. Mesmo dentro de um conjunto U pré-determinado de trajetórias às quais já se espera

um determinado comportamento existe a possibilidade do surgimento de uma nova situação para a qual a rede apresenta erros insatisfatórios em tempo real. Esta nova situação pode ser originada por fatores como erros de amostragem não previstos, variação brusca ocasionada por alguma perturbação e principalmente pela variação na carga transportada pelo manipulador. A carga transportada pelo efetuador não é fixa. A RNA é normalmente treinada para um valor nominal de carga. Durante a execução das tarefas do manipulador valores aleatórios são acrescidos ao valor nominal alterando aquela situação de treinamento. Seria desejável portanto que a rede de alguma maneira pudesse aprender em tempo real cada nova situação, minimizando assim os erros de seguimento apresentados em malha fechada. Tal esquema implica em uma atualização on-line da rede, tornando o controlador em questão, no caso aqueles apresentados nas seções 5.3, 5.4 e 5.5, adaptativo.

O esquema de adaptação on-line proposto neste trabalho consta de um retreinamento periódico da RNA. Os vetores utilizados no retreinamento são obtidos do próprio controle do processo e armazenados. Um período Tt é definido para o inicio de cada retreinamento. Ao longo deste período Tt os vetores são amostrados em um período Tv << Tt enquanto os torques de saída da rede são aplicados as entradas do manipulador em um período Tr que é o período de amostragem da RNA e funciona de forma semelhante a um período de amostragem de um controlador PID digital. A figura 5.8 representa o esquema de funcionamento de um controlador baseado na passividade utilizando uma RNA adaptativa.

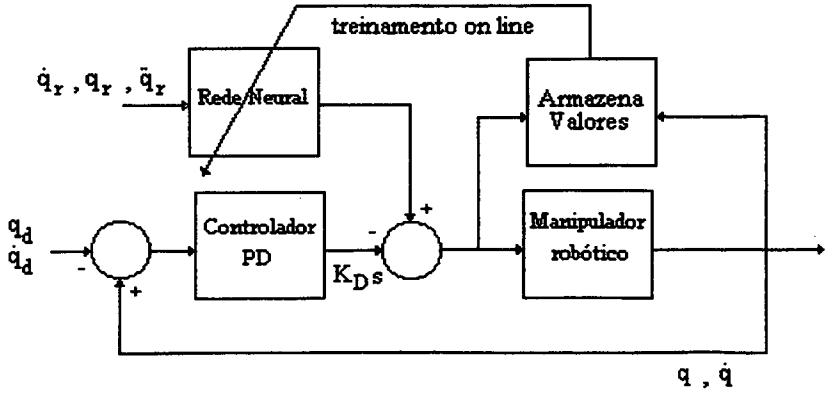


FIGURA 5.8 - Controlador Baseado na Passividade Utilizando uma RNA Adaptativa

Um aspecto importante em relação ao tamanho da RNA adotado neste esquema é que este não deve proporcionar uma rede sensível ao problema do sobretreinamento, pois uma rede com tais características fatalmente teria seu desempenho piorado com as atualizações on-line. Um número não muito grande de vetores deve ser escolhido para a atualização, procurando não incrementar demais a carga computacional. Um aspecto interessante abordado em [17] é o fato de que retreinamentos constantes fazem uma RNA muitas vezes "esquecer" o seu treinamento original, particularizando demais o comportamento e fazendo com que a rede perca sua boa capacidade de generalização. Faz-se necessário então que uma amostra do conjunto de treinamento original seja incorporada aos vetores amostrados em tempo real, prevenindo este aspecto. A quantidade de vetores amostrados em tampo real no entanto deve ser dominante em relação ao conjunto de treinamento on-line, senão a RNA pode adaptar-se muito pouco a uma nova situação.

Dentro do contexto de treinamento on-line descrito torna-se fundamental a velocidade de treinamento. O algoritmo de treinamento possui um *Deadline* que é determinado pelo período Tt e deve realizar sua tarefa dentro deste tempo. Portanto o número de iterações de treinamento ou épocas, que corresponde a apresentação completa do conjunto de treinamento, é limitado. Algoritmos que necessitam de centenas de iterações para um treinamento satisfatório tornam-se difíceis de serem utilizados devido a velocidade de processamento que exigiriam do *hardware* utilizado.

Neste trabalho utilizou-se com sucesso o algoritmo *Quickpropagation* descrito na seção 4.5 para o treinamento on-line da rede. Para o exemplo apresentado na subseção 5.2.2 julgou-se suficiente 15 iterações de treinamento para obter uma performance satisfatória.

O esquema de atualização on-line é descrito através da figura 5.9. Uma amostragem é feita em tempo real e armazenada em um vetor A de tamanho limitado. Quando o vetor A atinge seu tamanho máximo a amostra mais antiga é desconsiderada e substituída pela sua subsequente e assim por diante até que se reserve espaço para a amostra atual, em um sistema de pilha cujo "fundo" sempre cai. Em um primeiro momento espera-se que o conjunto A tenha um número mínimo de elementos para iniciar-se o processo de atualização, quando tal premissa é atingida inicia-se um retreinamento da RNA utilizando o conjunto A adicionado a um conjunto T, retirado do

conjunto original de Pares de Treinamento usado no treinamento original da rede. Para esta estratégia de treinamento considera-se que A deve ser maior que T, pois os parâmetros atuais devem ser dominantes em relação a condição inicial. É importante ressaltar que o conjunto T não altera-se ao longo do treinamento assim como o faz o conjunto A, permanecendo o mesmo fixo enquanto o conjunto A é constantemente atualizado. O retreinamento é realizado sempre em paralelo com a execução do controle durante um período Tt. Quanto atinge-se Tt os pesos da RNA são atualizados segundo o treinamento imediatamente anterior. Um novo retreinamento é então redisparado utilizando um novo conjunto A que foi amostrado durante o período Tt anterior. Este conjunto A pode ou não conter elementos do conjunto A anterior, dependendo do tamanho máximo definido para este e dos períodos de amostragem e treinamento definidos. Quando percebe-se que o erro de posição torna-se constante, ou seja, o treinamento não surte mais efeito, o mesmo é paralisado até que uma nova variação no erro de posição seja detectada.

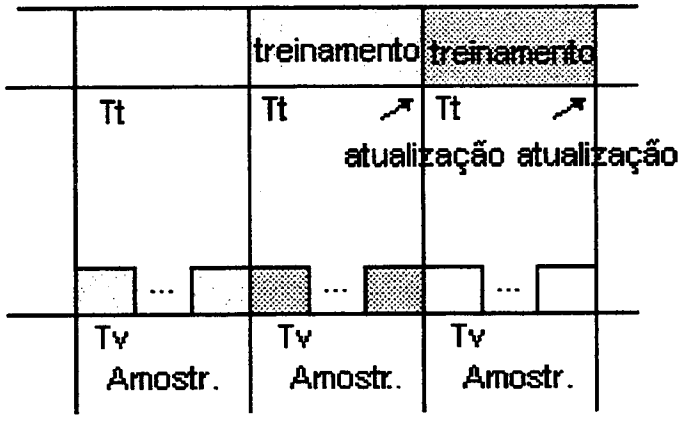


FIGURA 5.9 – Esquema de atualuzação on-line da RNA

A estratégia de atualização proposta corre dois riscos considerados importantes: o sobretreinamento e a aquisição de amostras defeituosas. Em relação ao sobretreinamento, que pode levar a rede a não generalizar novas situações, um projeto bem feito da rede associado a um bom treinamento inicial podem resolver este problema, uma vez que no retreinamento a rede sofrerá poucas iterações e portanto uma mudança brusca de comportamento é uma possibilidade que não tem grande chance de ocorrer dentro destas condições. O treinamento on-line serve somente como um ajuste fino em relação a variação do processo em tempo real, regulado pelo período de treinamento Tt.

Quanto a aquisição de amostras defeituosas, isto poderá ocorrer devido a uma má definição dos períodos Tr e Tt envolvidos no ajuste on-line e também a uma definição errada do tamanho do conjunto A, ocasionando uma atualização também errada nos pesos da rede. O tamanho de A deve ser representativo em relação ao processo e ao mesmo tempo não deve ser dimensionado de tal forma que permita que uma mesma seqüência seja exaustivamente retreinada. Se ocorrerem amostras espúrias decorrentes de alguma perturbação momentânea, julga-se que a constante atualização do conjunto A sob parâmetros adequadamente dimensionados poderá resolver este problema, proporcionando à rede manter-se dentro de um mapeamento aceitável.

Neste trabalho desconsidera-se um aspecto prático importante dentro de uma implementação real que é a possibilidade de sensoriamento em tempo real das variáveis de entrada envolvidas, principalmente da aceleração, à qual a obtenção segundo [10] constitui-se uma dificuldade prática. Tal consideração não é feita porque julga-se que estratégias de saneamento desta dificuldade passam por experimentações práticas com modelos do mundo real, onde a dificuldade poderá ser sentida e as possíveis formas de contornar o problema poderão ser testadas. Poderia por exemplo ter sido proposta a utilização de delays na entrada da rede em substituição a medição da velocidade e aceleração das juntas, técnica comum utilizada na teoria de redes neurais usada para sintetizar derivadas de ordem superior. Os resultados teóricos poderiam funcionar bem, mas nenhuma afirmação poderia ser feita antes de testar o processo na prática, pois o que na realidade estaria se tentando fazer é resolver um problema prático e não teórico. Por esta razão prefere-se considerar somente as dificuldades teóricas encontradas, projetando este e possíveis outros aspectos para trabalhos futuros que visem a sua análise.

No capítulo 6 apresenta-se simulações considerando a atualização on-line da rede, onde são descritos os detalhes estruturais da implementação desta técnica para o exemplo apresentado na subseção 5.2.2.

5.7 - CONCLUSÃO

Os controladores apresentados neste capítulo são extensões daqueles apresentados no capítulo 3 com a substituição da dinâmica inversa por uma estimação da mesma feita por uma RNA. O controlador baseado na dinâmica inversa e o controlador

tipo torque computado exigem uma RNA mais robusta em relação a variação das situações de operação em tempo real pois erros de estimação podem levar com facilidade o sistema a instabilidade, sendo portanto propostas mais delicadas e com menores garantias de funcionamento em processos do mundo real. O controlador baseado na passividade é mais robusto em relação a estimação da dinâmica inversa e portanto permite que a rede neural possa apresentar um mapeamento menos robusto em relação a variações do processo e projeta melhores chances de estabilizar o sistema em aplicações práticas.

A atualização da RNA em tempo real constitui uma melhoria no desempenho destes controladores, uma vez que adapta a rede às variações do processo em tempo real, manifestadas principalmente pela variação da carga transportada pelo efetuador.

6- SIMULAÇÕES

6.1 - INTRODUÇÃO

Com a intenção de ilustrar as estratégias de controle propostas neste trabalho são realizadas simulações em cima de um modelo exemplo. Dentro destas simulações são consideradas diferentes situações de controle. Ao final do capítulo pretende-se que se tenha uma visão mais clara das vantagens e desvantagens de cada estratégia e da sua eficiência em relação a diminuição dos erros de seguimento de trajetória.

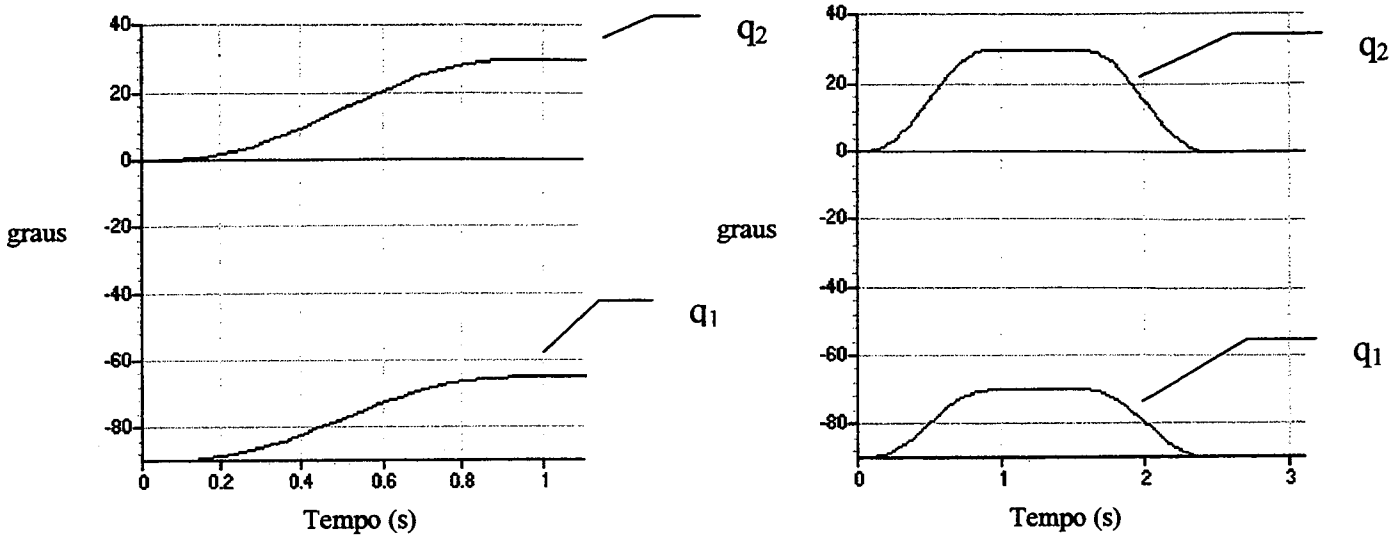
O modelo a ser adotado como exemplo em todas as situações deste capítulo é aquele descrito na subseção 5.2.2. Na seção 6.3 no entanto são acrescentados ao modelo alguns parâmetros referentes a consideração dos efeitos da dinâmica elétrica e da flexibilidade nas transmissões do manipulador. Todas as simulações utilizadas neste capítulo foram feitas no software Matlab, utilizando o aplicativo simulink, onde foram desenvolvidos blocos para o robô manipulador e para os controladores em questão.

6.2 - SIMULAÇÕES ENVOLVENDO UM MODELO RÍGIDO

O modelo rígido, cuja equação do movimento é descrita pela equação (2.33), será simulado de acordo com as trajetórias descritas na tabela 6.1 e na figura 6.1.

	q_1 (ini.)	q_2 (ini.)	q_1 (interm.)	q_2 (interm.)	q_1 (final)	q_2 (final)	tempo (s)
TRAJ1	-90^0	0^0	-	-	-65^0	30^0	1.0
TRAJ2	-90^0	0^0	-70^0	30^0	-90^0	0^0	2.5

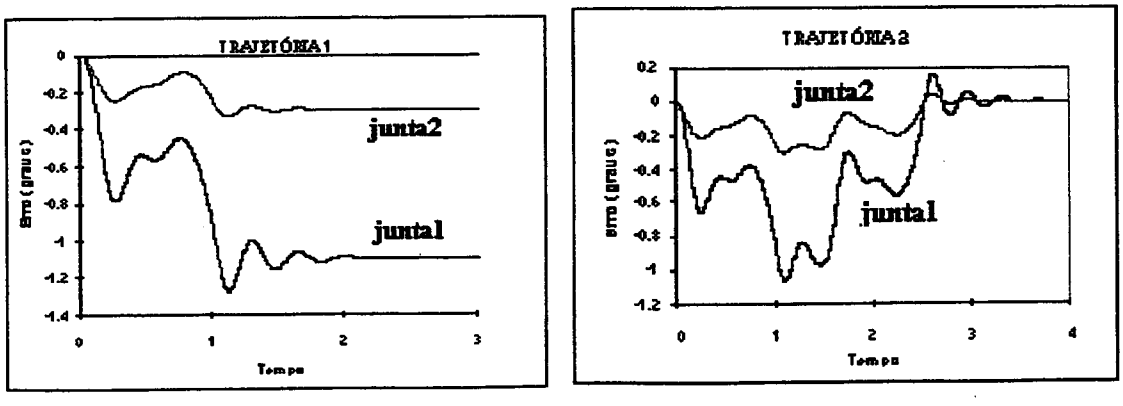
TABELA 6.1 - Trajetórias de Simulação



(a) Trajetória 1 (b) Trajetória 2

FIGURA 6.1 – Trajetórias de Simulação

Os parâmetros para a simulação a serem adotados para o modelo são os mesmos apresentados na tabela 5.1. Simula-se inicialmente um controlador clássico PD junta por junta. Os valores para os ganhos do controlador são $K_{p1} = K_{p2} = 10.000$ e $K_{v1} = K_{v2} = 200$. A figura 6.2 apresenta os resultados da simulação de ambas as trajetórias descritas na tabela 6.1. Considera-se o manipulador transportando sua carga nominal, ou seja, $m_3 = 20kg$.



(a) (b)

FIGURA 6.2 - Controlador PD de Ganhos Fixos

Os controladores apresentados no capítulo 5 pretendem proporcionar uma melhora em relação ao comportamento do controle em malha fechada descrito pela

figura 6.2. O primeiro passo para a implementação desses controladores é o treinamento da rede neural utilizada para a estimação da dinâmica inversa do modelo. O procedimento de treinamento adotado neste trabalho esta descrito na seção 5.2.

Inicialmente considera-se uma estratégia de controle onde a RNA não é atualizada em tempo real, ou seja, uma RNA com treinamento off-line. Adota-se para a rede um período de amostragem T_r = 5ms, semelhante ao período adotado para os controladores PD digitais que são utilizados neste esquema de controle.

Conforme já comentado na seção 5.3, não obteve-se boa performance neste trabalho para o controlador baseado na dinâmica inversa utilizando RNA. Por este motivo as simulações referentes a esse controlador não são apresentadas.

Procede-se então a simulação de um controlador tipo torque computado utilizando redes neurais. A titulo de comparação os ganhos para o controlador PD são mantidos iguais aos da simulação da figura 6.2. A figura 6.3 apresenta os resultados das simulações aplicadas ao controle de seguimento das trajetórias da tabela 6.1. Em 6.3.a e 6.3.b procede-se a simulação considerando o robô transportando carga nominal, ou seja, m_3 = 20kg. Em 6.3.c e 6.3.d considera-se m_3 = 40kg. Observa-se através da figura 6.3 que a performance é melhorada em pelo menos 100 vezes em relação a figura 6.2, o que ilustra a diferença de resultados entre as estratégias de controle. Quando adiciona-se uma carga diferente daquela para a qual a RNA foi treinada tem-se uma piora na performance do controlador, fato ilustrado em 6.3.c e 6.3.d. Esta piora pode ser atenuada com a adoção de uma estratégia de atualização on-line da RNA.

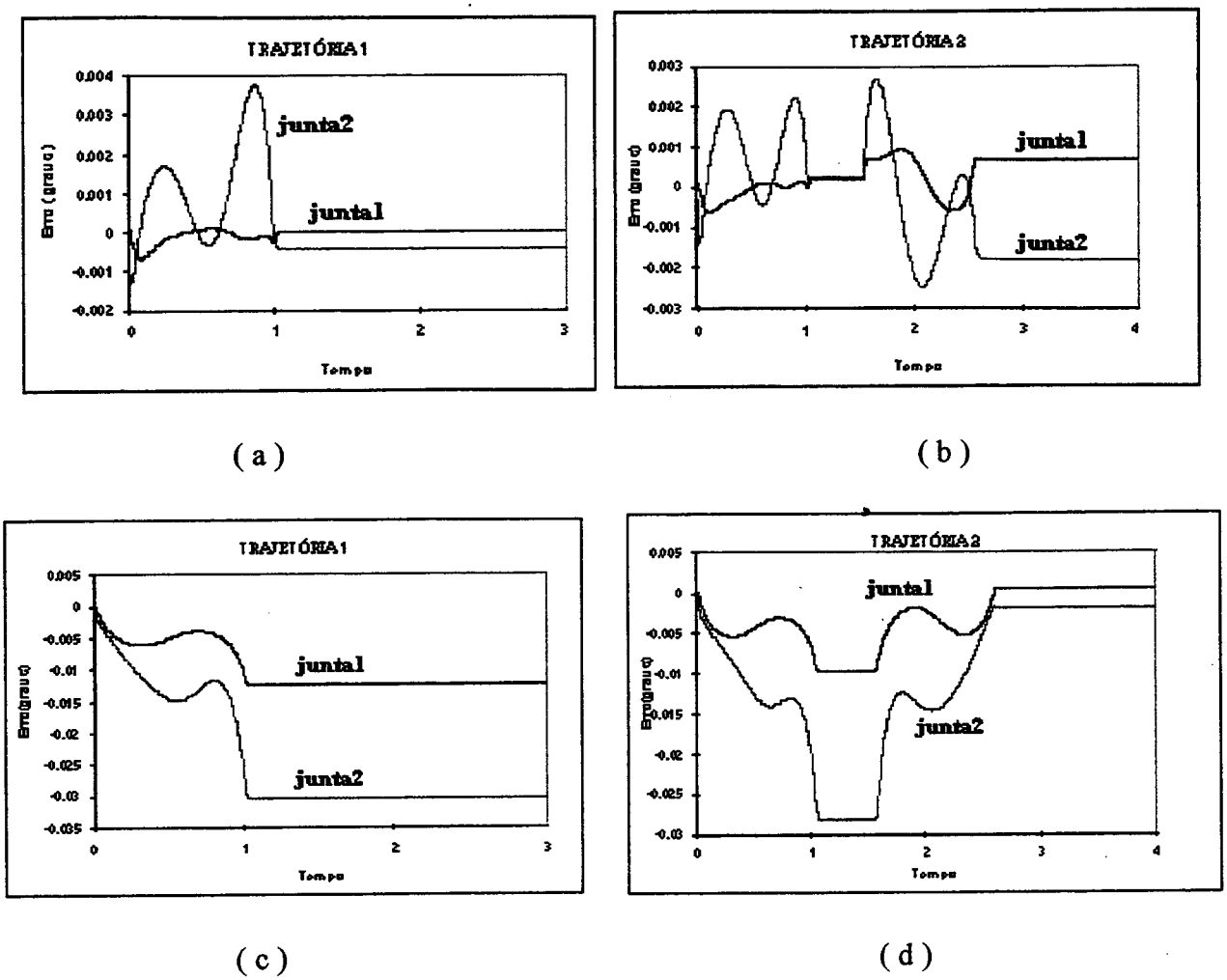
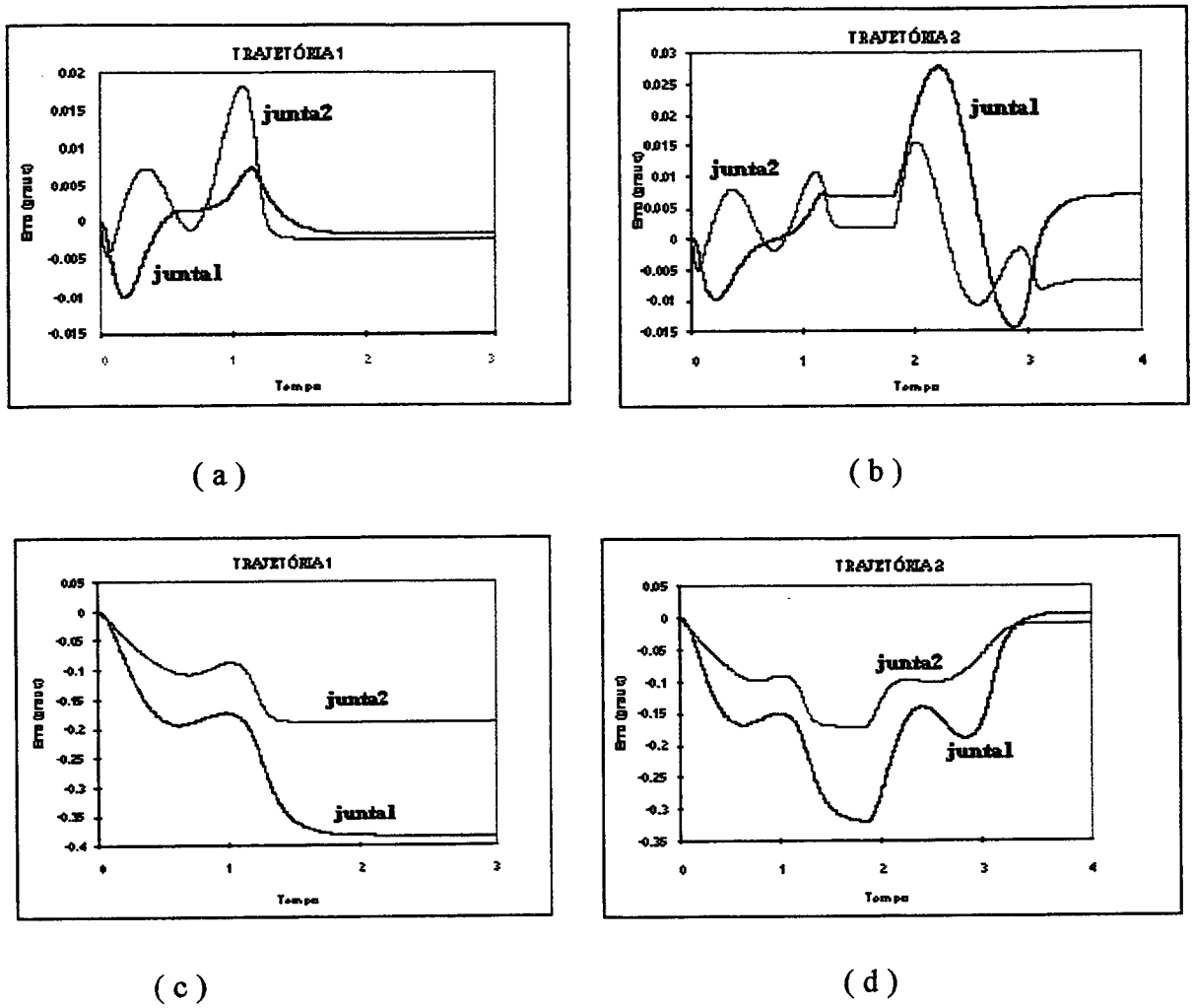


FIGURA 6.3 - Controlador Tipo Torque Computado utilizando RNA.

À seguir procede-se a simulação de um controlador baseado na passividade utilizando RNA. Mantém-se a mesma RNA utilizada nas simulações da figura 6.3. Escolhe-se como ganhos da parcela linear $K_{v1} = K_{v2} = 50$ e $\Lambda_1 = \Lambda_2 = 200$, ou seja, os mesmos ganhos PD utilizados na simulação anterior. Em 6.4.a e 6.4.b tem-se a simulação do controle de seguimento das trajetórias da tabela 6.1 considerando carga nominal. Em 6.4.c e 6.4.d tem-se as simulações considerando $m_3 = 40$kg. Observa-se novamente que a adição de carga no efetuador do manipulador piora a performance do controle.

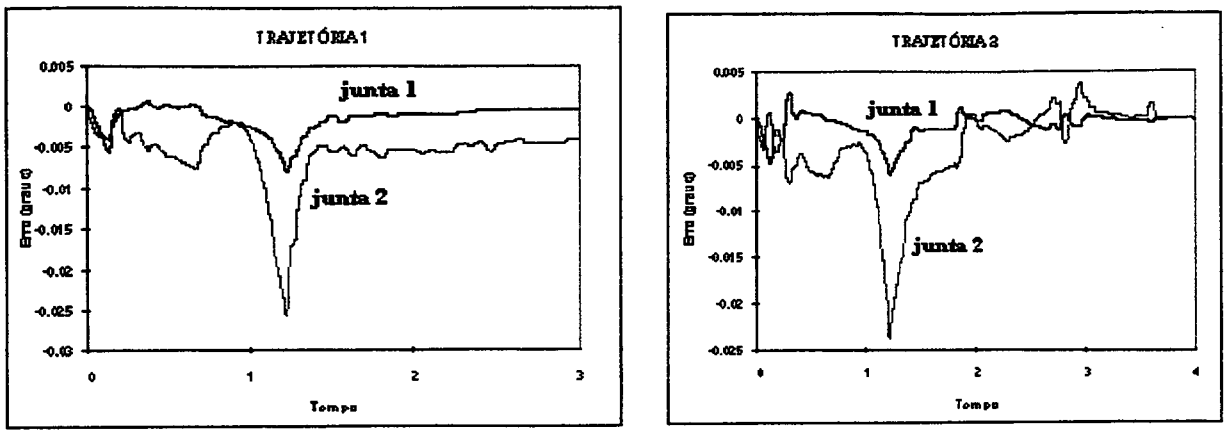


(a) (b)

(c) (d)

FIGURA 6.4 - Controlador baseado na Passividade utilizando RNA

Com o propósito de melhorar o desempenho do controle quando sob a situação descrita nas figuras 6.3.c, 6.3.d e 6.4.c e 6.4.d adota-se a estratégia de atualização on-line da RNA descrita na seção 5.3. Para tal estratégia adotou-se T_v = 2.5ms e Tt = 50ms. Portanto a RNA será atualizada a cada 50ms. A figura 6.5 mostra os resultados da simulação do controle de seguimento das trajetórias da tabela 6.1 considerando m_3 = 40kg. Utiliza-se o mesmo controlador tipo torque computado das simulações da figura 6.3 acrescido de uma estratégia de atualização on-line da RNA. Observa-se uma melhora de desempenho em relação a figura 6.3.c e 6.3.d que pode ser atribuída a adaptação da RNA em relação a nova situação que encontra-se o modelo, diminuindo tanto o erro transitório quanto o erro em regime permanente.



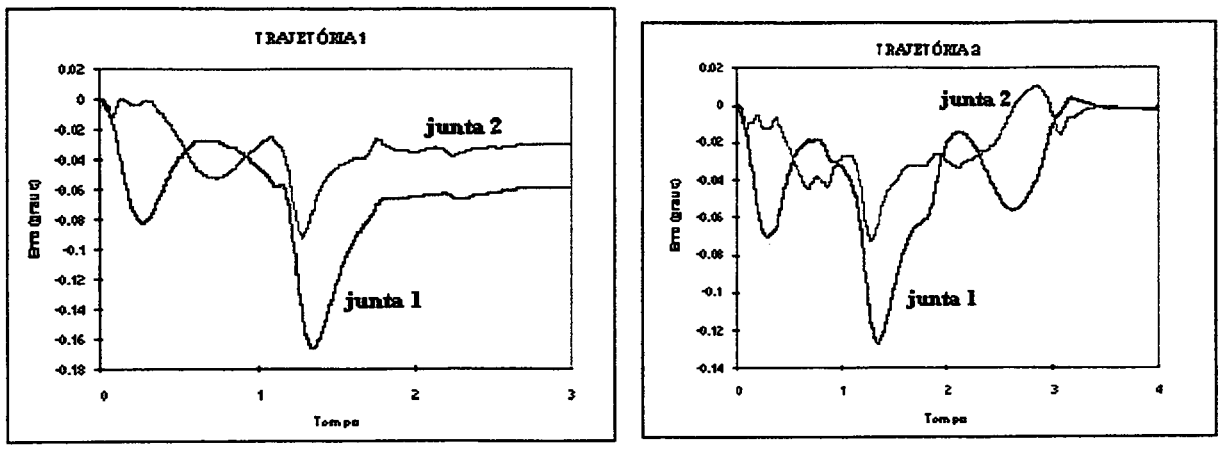
(a) (b)

FIGURA 6.5 - Controlador tipo torque computado utilizando uma RNA com atualização on-line

Da mesma maneira que na simulação da figura 6.5, tem-se na figura 6.6 a simulação do controle do seguimento das trajetórias da tabela 6.1 com m_3 = 40kg. Desta vez utiliza-se o controlador baseado na passividade com uma RNA atualizada em tempo real. Os períodos configurados para amostragem e treinamento on-line são mantidos os mesmos e o controlador é também o mesmo utilizado nas simulações da figura 6.4.

A utilização de um treinamento dinâmico implementa uma melhora significativa em relação a performance off-line, como ilustra a figura 6.6. Os erros de seguimento são diminuídos pois a RNA adapta-se as mudanças ocorridas no processo.

O modelo rígido de um robô planar de dois graus de liberdade foi o objeto de estudo desta seção. A dinâmica deste modelo não possui variáveis não controláveis, por isso ele é dito totalmente acionado [10]. Esta porém nem sempre é a situação encontrada na prática. Por vezes torna-se necessário considerar o efeito causado pelas demais dinâmicas que influenciam na equação do movimento. Na seção seguinte considera-se a hipótese que as dinâmicas não modeladas na equação do movimento no modelo rígido são importantes e portanto alteram o comportamento do processo. Tal consideração altera de maneira significativa a performance do controlador.



(a) (b)

FIGURA 6.6 - Controlador Baseado na Passividade utilizando uma RNA com Atualização on-line.

6.3 - SIMULAÇÕES CONSIDERANDO A DINÂMICA DOS ATUADORES ELÉTRICOS E A FLEXIBILIDADE NAS TRANSMISSÕES DOS MOTORES

6.3.1. Introdução

Na seção 6.2 ilustrou-se através das simulações que os controladores propostos no capítulo 5 realmente melhoram a performance em relação a estratégia clássica de controle utilizando um controlador PD junta por junta. O modelo de robô utilizado para as simulações no entanto trata-se de um modelo rígido, cuja equação do movimento, conforme visto no capítulo 2, desconsidera a dinâmica associada aos atuadores elétricos utilizados para a aplicação de torque nas juntas e além disso desconsidera também que as transmissões que acoplam os rotores dos motores de acionamento aos elos do manipulador possuam flexibilidade. Muitas vezes estas simplificações são verdadeiras em casos práticos, porém existem também manipuladores nos quais a desconsideração destas dinâmicas conduz a erros consideráveis ou mesmo inviabilizam o controle de seguimento de trajetória da forma aqui apresentada. Nesta seção pretende-se ilustrar através de simulações os efeitos da inclusão destas dinâmicas no modelo do manipulador a ser controlado pelos controladores propostos e com isso mostrar as limitações dos mesmos quando atuando sobre estes modelos.

6.3.2. Simulações considerando a dinâmica dos atuadores elétricos.

Conforme foi visto no capítulo 5, os controladores ali propostos são baseados naqueles vistos no capítulo 3. Portanto estes controladores possuem basicamente as mesmas características dos seus antecessores e também por conseqüência as mesmas limitações.

No caso do controle de um manipulador acionado eletricamente comenta-se na subseção 2.4.2 que esta dinâmica associa graus de liberdade adicionais ao modelo do manipulador, ou seja, o sistema possui mais variáveis a serem controladas do que entradas de controle. Tal fator influencia diretamente no desempenho dos controladores do capítulo 3. Em [10] mostra-se que o modelo do manipulador é dividido em dois subsistemas, um subsistema contendo o modelo dos atuadores, no qual são aplicados os sinais de controle e outro subsistema contendo o modelo rígido a ser controlado. Em [33] demonstra-se o efeito causado quando a constante de tempo associada a dinâmica elétrica é considerada. O autor mostra o caso de um controlador PD atuando em um manipulador de 1 grau de liberdade. O lugar das raízes da equação do controlador associada ao modelo resulta em dois pólos complexos estáveis tendendo ao infinito conforme aumenta-se o ganho do controlador. Dai conclui-se que o comportamento em malha fechada pode resultar em uma oscilação subamortecida cujo fator de amortecimento depende do ganho do controlador. A dinâmica de alta freqüência atribuída aos atuadores elétricos impõe portanto um limite aos ganhos do controlador PD utilizado.

Em relação ao controlador baseado na dinâmica inversa afirma-se em [10] não ser possível o cancelamento direto das não linearidades realizado pelo mesmo, o que, segundo o autor, requer a busca de outra técnica que torne viável este cancelamento. Esta colocação comprova-se ao buscar-se uma RNA que possa representar a dinâmica inversa no controlador tipo torque computado utilizando RNA. Nas simulações envolvendo o modelo rígido encontrou-se uma RNA que generaliza adequadamente o conjunto de trajetórias da tabela 4.2, porém ao buscar-se uma RNA que execute tarefa semelhante em relação ao modelo considerando os atuadores elétricos encontra-se uma dificuldade muito maior que no caso anterior. Acredita-se que os graus de liberdade adicionais devidos a dinâmica elétrica são os responsáveis por esta dificuldade, uma vez que a rede conta somente com as posições e suas derivadas para

estimar a tensão de entrada necessária ao cancelamento. No caso de considerar-se diferentes trajetórias, os erros inerentes a estimação de uma trajetória diferente daquelas utilizadas no conjunto de treinamento por parte da RNA são agravados pela presença da dinâmica elétrica e tornam o processo em malha fechada instável para este caso. Seria necessário portanto que nos casos onde a dinâmica elétrica não é completamente desprezível fosse dimensionada uma RNA que proporciona-se erros mínimos, o que na prática pode ser inviável.

Em relação ao controlador baseado na passividade afirma-se também em [10] que não existe uma relação de passividade entre as entradas de controle e a saída a ser controlada, no caso a posição angular das juntas, existindo esta relação somente quando se considera as entradas e as saídas do subsistema envolvendo os atuadores. Dependendo do quanto a dinâmica elétrica é excitada pela trajetória a ser controlada é possível utilizar o mesmo esquema do modelo rígido, mesmo que a dinâmica elétrica insira erros de estimação na RNA. O que ocorre portanto é uma limitação da trajetória de acordo com as características da dinâmica elétrica.

Na figura 6.7 tem-se a simulação do modelo apresentado na subseção 5.2.2 considerando a dinâmica dos atuadores elétricos, onde [10] $L_i = 8x10^{-5}$ V·s/A, $R_i = 1.5$ ohms, as constantes de tensão induzida são $K_1^e = 25.05$ e $K_2^e = 21.71$ e as constantes de torque são $K_1^m = 25.05$ V·s e $K_2^m = 21.71$ V·s. O dimensionamento da RNA dá-se de acordo com os procedimentos descritos na seção 5.2, desta vez porém o treinamento é feito considerando o manipulador acionado elétricamente. Os ganhos do controlador linear são $K_{vi} = 20$ e $\Lambda_i = 50$ e os períodos da RNA são os mesmos das simulações da seção 6.1.

Na figura 6.7.a e 6.7.b ilustra-se o comportamento do controlador baseado na passividade utilizando uma RNA com treinamento off-line. Neste caso m_3 tem seu valor nominal. Observa-se que o comportamento não é comprometido pela consideração da dinâmica elétrica, uma vez que esta não é excitada pela trajetória em questão. Na figura 6.7.c e 6.7.d considera-se $m_3 = 40$kg, uma situação onde utiliza-se o treinamento on-line da RNA. Observa-se que igualmente o comportamento do controle é satisfatório.

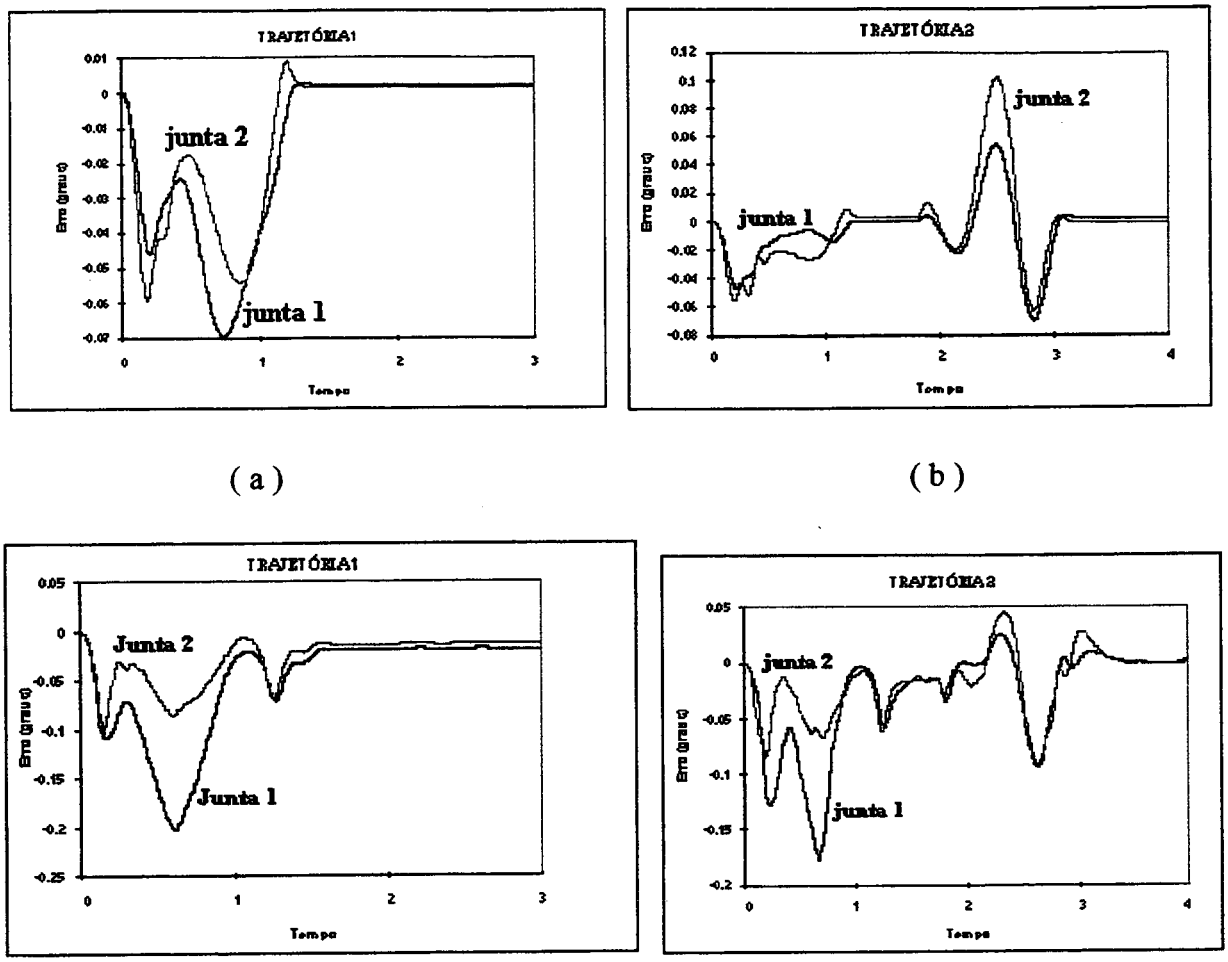


FIGURA 6.7 - Controlador Baseado na Passividade utilizando RNA atuando sobre um Modelo Rígido Acionado Eletricamente

6.3.3. Simulações considerando transmissões flexíveis.

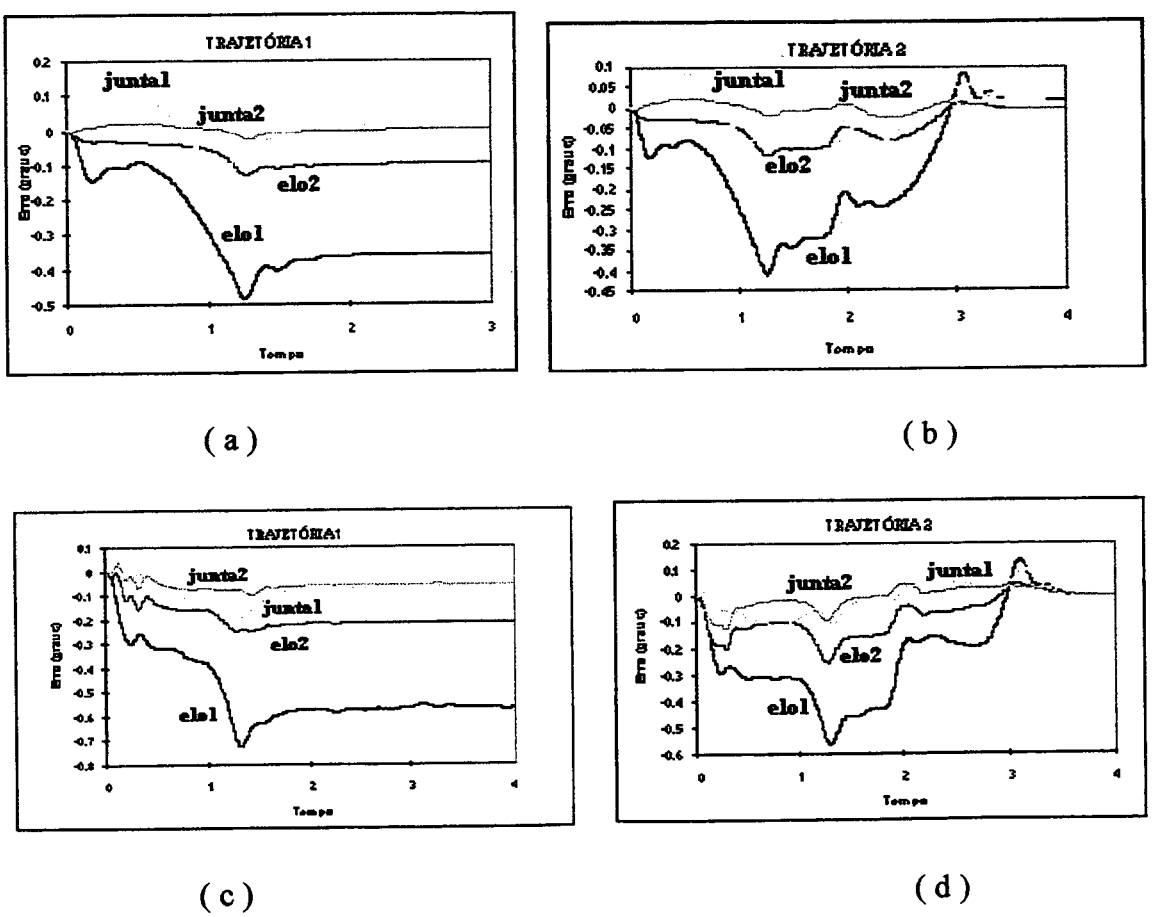
Conforme comentado na subseção 2.4.4, a parcela significativa de flexibilidade em manipuladores robóticos esta associada as transmissões que acoplam o rotor dos motores aos elos do robô. Estas transmissões flexíveis quando consideradas alteram significativamente a equação do movimento do manipulador conforme (2.33) e (2.34). Em [33] é analisado o controle em malha fechada de um manipulador de 01 junta com transmissão flexível sendo controlado através de um controlador PD. A exemplo de quando considera-se as dinâmicas dos atuadores elétricos, o sistema também pode ser dividido em dois subsistemas, um subsistema dos elos e outro dos rotores. Existirá uma

diferença entre a posição angular dos elos e dos rotores dos motores de acionamento, que será maior a medida que a constante de rigidez K associada as transmissões for menor. A posição dos elos é no caso o interesse do controle, porém em [33, 10] demonstra-se que ao fechar-se a malha de controle com o controlador PD utilizando as posições dos elos como realimentação o lugar das raízes do sistema mostra dois pólos complexos conjugados no semiplano direito, concluindo-se portanto que tal sistema de controle é instável. Demonstra-se também que se utilizadas as posições angulares dos rotores, ou seja, as posições das juntas, como realimentação, o lugar das raízes do sistema mostra dois pólos complexos conjugados sobre o eixo imaginário com dois zeros complexos conjugados próximos, sendo portanto o sistema estável. Por conseguinte o sistema de controle assim implementado controla somente a dinâmica associada aos rotores, ficando as posições dos elos, que é o objetivo do controle, dependentes da constante de rigidez K.

Em relação a aplicação dos controladores do capítulo 3, Em [10] Guenther afirma que, por ser o sistema dinâmico parcialmente acionado, não é possível cancelar diretamente as não linearidades, não sendo possível portanto a utilização do controlador baseado na dinâmica inversa na forma em que este é apresentado no capítulo 3. Esta afirmação confirma-se ao tentar-se utilizar o controlador tipo torque computado utilizando RNA. Consegue-se algum resultado somente para constantes de rigidez muito elevadas e neste ponto isto é semelhante a desconsiderar a flexibilidade nas transmissões. O controlador baseado na passividade no entanto pode ser utilizado, porém não existe mapeamento passivo entre as entradas e a posição angular dos elos existindo sim entre as entradas e a posição angular dos rotores. Levando em consideração este aspecto e também o fato de não ser possível utilizar diretamente as posições dos elos pelas razões já descritas, um controlador baseado na passividade utilizando RNA se utilizará da posição angular dos rotores ou juntas como realimentação, ficando a posição final dos elos sujeita ao erro inserido de acordo com a dimensão da constante de rigidez K e de acordo com a trajetória seguida.

A figura 6.8 ilustra a simulação do controle de seguimento das trajetórias da tabela 6.1. O modelo considerado é o mesmo das demais simulações deste capítulo porém considerando a flexibilidade nas transmissões, com uma constante de rigidez $K_1 = K_2 = 30.000$. Em 6..8.a e 6.8.b tem-se as simulações considerando $m_3 = 20$kg e um

treinamento apenas off-line para a RNA, que é realizado levando em conta as entradas e as posições e derivadas em relação aos elos. Observa-se nos gráficos que há bastante diferença entre as posições dos elos e das juntas devido a flexibilidade nas transmissões. O sinal oferecido pela RNA na verdade não consegue mapear a dinâmica flexível, sendo limitado somente ao efeito da dinâmica do manipulador. Em 6.8.c e 6.8.d tem-se simulações considerando m_3 = 40kg e um treinamento on-line para a RNA. Nestas simulações confirma-se a afirmação anterior de que a RNA não é capaz de considerar a dinâmica flexível.



(a) (b)

(c) (d)

FIGURA 6.8 - Controlador Baseado na Passividade aplicado a um Modelo com Transmissões Flexíveis

Para as simulação da figura 6.8 o treinamento da RNA é semelhante ao descrito na seção 5.2, porém o modelo utilizado considera a flexibilidade nas transmissões. O processo em malha fechada utilizado para a coleta dos pares de treinamento apresenta um controlador PD realimentado pelas posições e velocidades das

juntas e não dos elos. Os pares de treinamento são formados pelas variáveis de junta e respectivos torques de entrada no manipulador.

O controlador baseado na passividade utilizando RNA carece portanto de novas ferramentas que proporcionem de alguma forma compensar o efeito da flexibilidade nas transmissões. Em [10, 33, 24] são abordadas alternativas que venham a solucionar o problema do controle de robôs considerando a flexibilidade nas transmissões e várias soluções são analisadas. Acredita-se que trabalhos posteriores possam buscar alguma alternativa às limitações dos controladores aqui apresentados em relação ao problema da flexibilidade nas transmissões considerando alguma destas soluções.

Dos controladores aqui propostos utilizando RNA, o controlador baseado na passividade utilizando RNA é o único portanto que pode ser utilizado diante do problema da flexibilidade nas transmissões, porém tem seu comportamento limitado à dimensão da constante K de rigidez nas transmissões e depende também de uma trajetória suave, ou seja, uma trajetória que não venha a excitar os modos flexíveis das transmissões, pois caso isso aconteça o comportamento do processo em malha fechada pode tornar-se instável.

6.4 - CONCLUSÃO

Os Controladores propostos no capítulo 5 confirmam a sua eficiência através das simulações realizadas ao longo deste capítulo. No caso de considerar-se o modelo rígido descrito na subseção 2.4.3 como sendo o modelo do manipulador a ser controlado tem-se que o controlador tipo torque computado apresenta melhores resultados em relação aos erros de posição do que o controlador baseado na passividade, apesar da limitações já comentadas na seção 5.3. Quando simula-se uma situação de carga diferente da nominal tem-se no ajuste *on-line* da RNA um importante fator de diminuição dos erros de posicionamento para ambas as estratégias de controle. Quando considera-se a dinâmica dos atuadores elétricos conforme a subseção 2.4.2 tem-se dificuldades na utilização do controlador tipo torque computado devido ao sistema ser parcialmente acionado. O controlador baseado na passividade pode ser utilizado com sucesso, mas sua eficiência depende principalmente de uma trajetória que não venha a excitar as dinâmicas dos atuadores elétricos, que é o caso da simulação apresentada.

Quando considera-se a flexibilidade nas transmissões a utilização do controlador tipo torque computado conforme apresentado aqui neste trabalho torna-se praticamente inviável. O controlador baseado na passividade pode ser utilizado contanto que se tenha uma constante de rigidez K nas transmissões de dimensão adequada e também uma trajetória que não venha a excitar os modos flexíveis. Percebe-se no entanto que o controlador tem seus erros aumentados devido a existência desta dinâmica, carecendo de alguma modificação em sua estrutura no sentido de solucionar este problema.

7 - CONCLUSÃO

Os controladores clássicos descritos no capítulo 3 tem seu uso limitado as seguintes situações:

- No caso de controladores PD e PID de ganhos fixos à processos onde não se exija precisão no controle;

- No caso dos controladores baseados na passividade e na dinâmica inversa em processos onde se tenha possibilidade de obter matematicamente com alguma precisão a dinâmica inversa do manipulador e contando também com pouca variação na carga transportada pelo efetuador.

No caso de ter-se processos de difícil estimação ou com grande variação de parâmetros em tempo real necessita-se de alguma ferramenta de estimação e ajuste destes parâmetros.

As RNA se mostram grandes ferramentas de estimação. Neste trabalho foram utilizadas para estimar em tempo de treinamento e em tempo real a dinâmica inversa de manipuladores robóticos. Conclui-se que tal ferramenta obtém êxito nesta tarefa, destacando como principais vantagens:

- Possibilidade de aprendizado através de exemplos, sem a necessidade de conhecimento detalhado do modelo matemático do processo;

- Capacidade de generalização do comportamento do modelo para um conjunto de valores diferentes daqueles utilizados no conjunto de treinamento;

- Possibilidade de ajuste em tempo real.

A arquitetura de RNA utilizada na tarefa proposta neste trabalho foi a arquitetura *feedforward* multicamadas. Dentro desta arquitetura adotou-se configurações típicas encontradas ao longo da literatura para as funções de ativação e inicialização dos pesos da rede, mas considerou-se como fator decisivo no desempenho do treinamento, principalmente o treinamento *on-line*, a velocidade de convergência do algoritmo de treinamento. O algoritmo Backpropagation, em sua forma original proposta em [27] se

mostra muito lento para a velocidade de treinamento exigida. O algoritmo de treinamento Quickpropagation, de [9], mostra-se bem mais veloz na execução da tarefa, e por isso veio a ser escolhido como algoritmo de treinamento das RNA utilizadas neste trabalho.

Os controladores utilizando RNA apresentados neste trabalho mostram-se boas alternativas aos métodos clássicos, uma vez que além de reduzirem bastante o erro de seguimento de trajetória proporcionam uma estratégia de adaptação as mudanças ocorridas no modelo do manipulador robótico em tempo real. Dentre essas propostas destaca-se o controlador tipo torque computado utilizando RNA, que é uma alternativa à dificuldade de utilização do controlador baseado na dinâmica inversa utilizando RNA encontrada neste trabalho e o controlador baseado na passividade utilizando RNA.

Em relação ao comportamento apresentado nas simulações conclui-se que o controlador tipo torque computado utilizando RNA apresenta melhor performance que o controlador baseado na passividade utilizando RNA quando utilizado o modelo rígido, provavelmente por conseguir compensar de forma mais eficiente o efeito das não linearidades do modelo, porém sua utilização não é possível nos modelos que consideram as dinâmicas dos acionadores elétricos e a flexibilidade nas transmissões. O controlador baseado na passividade utilizando RNA possui boa performance tanto no modelo rígido quanto no modelo com acionamentos elétricos, porém no modelo com transmissões flexíveis, apesar de estabilizar o sistema em malha fechada, não consegue compensar os efeitos desta dinâmica em relação ao erro inserido na posição desejada para elos do manipulador. Sugere-se, para este caso, que esta estratégia possa ser utilizada em conjunto com alguma outra estratégia que vise melhorar o desempenho do controle, um trabalho futuro à ser desenvolvido.

Tem-se ainda como sugestão de futuro trabalho a utilização destes controladores em experimentos práticos, talvez utilizando manipuladores não planares. Novas dificuldades como medição das grandezas, dinâmica dos acionadores e flexibilidade reais serão encontradas, o que poderá resultar em um trabalho mais completo e abrangente.

BIBLIOGRAFIA

[1] ARMSTRONG, B. , O. KHATIB AND J. BURDICK, "The Explicit Dynamic Model and Inertial Parameters of the PUMA 560 Arm", presented at the IEEE Conf. Robot. Autom., San Francisco, USA, April 1986, pp 510-518.

[2] ASADA, HARUHIKO, SLOTINE, JEAN-JACQUES E., "Robot Analysis and Control", Wiley-Intercience, a division of John Wiley and Sons, Inc., USA, 1986.

[3] AZEVEDO, F. M., BARRETO, J. M., " IMC Scheme Using Neural Networks for Robot Arms"

[4] BOAVENTURA, C. S., LILLY, K., W., "A Constrained Motion Algorithm for The Shuttle Remote Manipulator System", IEEE Control Systems Magazine, pp 6-15, outubro de 1995.

[5] BUSH, FERNANDO, "Implementação de um Ambiente Simulador de Robôs Manipuladores Utilizando o Paradigma da Programação Orientada a Objeto", Dissertação de Mestrado, Programa de pós Graduação em Engenharia Elétrica - UFSC, Florianópolis, Novembro de 1994.

[6] CORKE P. I., B. ARMSTRONG-H\'ELOUVRY. "A meta-study of PUMA 560 dynamics: A critical appraisal of literature data". Robotica, 13(3):253-258, 1995.

[7] DEMUTH, H., BEABLE, M., "Neural Network Toolbox User Guide", The Math Works, Inc, Massachusetts, 1992.

[8] ERLIC, M. AND W. S. LU, "A Reduced-Order Adaptative Velocity Observer for Manipulator Control", presented at IEEE Conf. Robot. Autom. , Atlanta, Georgia, USA, May 1993, vol 2, pp. 328-332.

[9] FAHLMAN, S. E. , "An empirical study of learning speed in back-propagation networks", Technical report CMU-CS-88-162, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15313, May 1991.

[10] GUENTHER, R, "Controle Adaptativo e a Estrutura Variável de Robôs Manipuladores com Incertezas Dinâmicas no Acionamento Elétrico e nas Transmissões Flexíveis", Tese de Doutorado, COPPE/UFRJ, Rio de Janeiro, 1993.

[11] GUENTHER, R., "Um Modelo para a Simulação do Manipulador Puma 560 Incluindo a Dinâmica dos Motores Elétricos e a Flexibilidade nas Transmissões", 2º Simpósio Brasileiro de Automação Inteligente - CEFET / Curitiba PR- Brasil, Pag 201-206, Setembro de 1995.

[12] GUENTHER, R., LIZZARALDE, F., J.P.V.S, "Estudo Comparativo de Estratégias de Controle de Manipuladores", 9º CBA- UFES - Vitória/ ES - Brasil, Pag 201-206, 1992

[13] GUENTHER, R., MACHADO, M. A., "O Projeto de um Controlador Adaptativo para Robôs Manipuladores", VI Congresso Nacional de Inginieria Mecanica", Santiago, Chile, pp. 149-154, Novembro de 1994.

[14] HUNT, K. J., SBARBARO, D., ZBIKOWSKI, R., GAWTHROP, P. J., "Neural Networks for Control Systems-A Survey", Automática, Vol 28, no 6, pp 1083-1112, 1992.

[15] JONDARR, C. G. H. , "Back Propagation Family Album", Technical Report C/TR96-05 , Department of Computing, Macquarie University, NSW, Agosto de 1996.

[16] KALMAN, B. L., KWASNY, S. C., "Why Tanh? Choosing a Sigmoidal Function", International Joint Conference on Neural networks, Baltimore, MD, 1992.

[17] KLINGUELFUS, M. C., "Ambiente Integrado para Síntese de Controladores Neurais Adaptativos ", Dissertação de mestrado, Programa de pós Graduação em Engenharia Elétrica - UFSC, Florianópolis, Maio de 1996.

[18] KRÖHLING, R. A., "Algoritmos de Controle Não Convencionais: Estudo de Caso", Dissertação de Mestrado, Departamento de Engenharia Elétrica, UFES, Vitória-ES, 1994.

[19] LAWRENCE S., GILES, C. L., TSOI, A. C., 'What Size Neural Networks Gives Optimal Generalization? Convergence Properties of Backpropagation", Technical Report UMIACS-TR-96-22 and CS-TR-3617, Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742, Agosto de 1996.

[20] LEWIS, F. L., ABDALLAH, C. T., DAWSON, D. M., " Control of Robot Manipulators", Macmillan Publishing Company, a division of Macmillan, Inc., USA, 1993.

[21] LEWIS, F. L., LIU, K., AYDIN, Y., "Neural Net Controller with Guaranteed Tracking Performance", IEEE Transaction on Neural Networks, vol 6, no 3, pp 703-715, maio de 1995.

[22] MASTERS, T., "Pratical Neural Networks Recipies in C++", Academic Press, INC, San Diego CA, 1993.

[23] NEWTON, R., T., XU, Y., "Neural Network Control of a Space Manipulator", IEEE Control Systems Magazine, pp 14-22, dezembro de 1993.

[24] RAMIREZ, A. R. G., "Controle de Posição de Robôs Manipuladores com Transmissões Flexíveis", Dissertação de Mestrado, Programa de pós Graduação em Engenharia Elétrica - UFSC, Florianópolis, Março de 1998.

[25] RIEDMILLER, M., BRAUN, H., "A Direct Adaptative Method for Faster Backpropagation Learning: The Rprop Algorithm", In Proceeding of The IEEE International Conference on Neural Networks", University of Karlsruhe, W-76128 Karlsruhe FRG, Abril de 1994.

[26] RITA, D. J., "Controle de Processos Usando Redes Neurais Artificiais: Uma Aplicação Experimental", Dissertação de mestrado, Departamento de Engenharia Química, UFSC, Florianópolis-SC, agosto de 1995.

[27] RUMELHART, D., McCLELLAND, J. L., "Parallel Distribuited Processing", Vol I, : Foundations, The MIT Press, 1969. Third Printing, 1988.

[28] SEPEDA FILHO, I. H., STEMMER, M. R., "Redes Neurais", Notas do curso de Redes Neurais, Departamento de Engenharia Elétrica, LCMI, Florianópolis, Setembro de 1993.

[29] SHIFFMANN, W., JOOST M., WERNAER, R., "Optimization, of the Backpropagation Algorithm for Training Multilayer Perceptons", Technical Report, University of Koblenz, Institute of Physics, Rheinau 1, 56075 Koblenz, Germany, 1994.

[30] SHOHAM, M., LI, J., HACHAM, Y., KREINDLER, E., "Neural Network Control of Robot Arms", Annals of the CIRP Vol 4, Pag 407-410, Janeiro de 1992.

[31] SLOTINE, JEAN-JACQUES E., LI, WEIPING, "Applied Nonlinear Control", Prentice-Hall Innternational, Inc., New Jersey, USA, 1991.

[32] SPONG, M. W., "Modeling and Control of Elastic Joint Robots", Trans. of the ASME, Journal of Dynamics Systems, Measurement an Control, vol 109, pp 310-319, 1987.

[33] SPONG, M. W., VIDYASAGAR, M., "Robot Dynamics and Control", John Wiley & Sons, Inc, New York, 1989.

[34] SSPA SYSTEMS, " Simnom for Windows version 1.02", Outubro de 1993.

[35] STEMMER, M.R.; DE PIERI, E.R.; BORGES, F.A.P. , "An Adaptive Neural Controller for a Manipulator Robot". 5th IFAC Workshop on Intelligent Manufacturing Systems (IMS'98), Gramado, RS, Brazil, November 9-11, 1998.

[36] STEMMER, M.R.; DE PIERI, E.; BORGES, F.A.P. , "Comparação de Performance entre Controladores Clássicos e um Controlador Torque Computado Neural Aplicados ao Robô PUMA 560". III Congresso Brasileiro de Redes Neurais (III CBRN), Florianopolis, 21 a 24 de julho de 1997.

[37] STEMMER, M.R.; DE PIERI, E.R.; BORGES, F.A.P. , "Um Controlador Neural Adaptativo para o Robô Manipulador PUMA 560". XII Congresso Brasileiro de Automatica, Uberlandia, MG, 14 a 18 de Setembro de 1998.

[38] TARN, T. J., A. K. BEJCZY, G. T. MARTH AND A. K. RAMADORAI, "Performance Comparison of Four Manipulator Servo Schemes", IEEE Control System Magazine, vol. 13, no. 1, pp. 22-29.

[39] TORRES, GERMANO LAMBERT, " Introdução as Redes Neuronais ", Notas do Curso de Introdução as Redes Neuronais, Grupo de Inteligência Artificial, Escola Federal de Engenharia de Itajubá, Itajubá, dezembro de 1992.

[40] TROCH, I., DESOYER, K., "Benchmark Robotic System", IFAC Theory Committee Report, 45-49, 1990.

[41] ZLAJPAH, L. , "Dynamic Simulation of n-R planar Manipulators", In Proc. EUROSIM Congress 95, Vienna, Àustria, 1995.