UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE CIÊNCIAS E TECNOLOGIA
COORDENAÇÃO DOS CURSOS DE
PÓS-GRADUAÇÃO EM ENG. ELÉTRICA

PABLO JAVIER ALSINA
UFPB-CCT-DEE

CONTROLE NEUROADAPTATIVO MODULAR DE MANIPULADORES ROBÓTICOS

Orientador:

NARPAT SINGH GEHLOT, Dr.-Eng. (Japão)

Campina Grande, Paraíba
Novembro - 1996

UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE CIÊNCIAS E TECNOLOGIA
COORDENAÇÃO DOS CURSOS DE
PÓS-GRADUAÇÃO EM ENG. ELÉTRICA

PABLO JAVIER ALSINA

CONTROLE NEUROADAPTATIVO MODULAR DE MANIPULADORES ROBÓTICOS

Tese apresentada à Coordenação de Pós-Graduação em Engenharia Elétrica - COPELE - da Universidade Federal da Paraíba - UFPB - em cumprimento às exigências para a obtenção do grau de Doutor em Ciências, no domínio da Engenharia Elétrica.

ÁREA DE CONCENTRAÇÃO: **Processamento de Informação**

ORIENTADOR: **NARPAT SINGH GEHLOT, Dr.-Eng. (Japão)**

Campina Grande, Paraíba
Novembro - 1996

A461c Alsina, Pablo Javier.
Controle neuroadaptativo modular de manipuladores roboticos / Pablo Javier Alsina. - Campina Grande, 1996.
154 f.

Tese (Doutorado em Engenharia Eletrica) - Universidade Federal da Paraiba, Centro de Ciencias e Tecnologia.

1. Robotica Inteligente - Engenharia Eletrica. 2. Tese - Engenharia Eletrica. I. Gehlot, Narpat Singh. II. Universidade Federal da Paraiba - Campina Grande (PB).

CDU 621.3:004.896(043)

"CONTROLE NEUROADAPTATIVO MODULAR DE MANIPULADORES ROBÓTICOS

PABLO JAVIER ALSINA

Tese Aprovada em 14.11.1996

NARPAT SINGH GEHLOT, Dr.Engg., UFPB
Orientador

JOSÉ HOMERO FEITOSA CAVALCANTI, D.Sc., UFPB
Componente da Banca

ELMAR UWE KURT MELCHER, Dr., UFPB
Componente da Banca

EDILBERTO PEREIRA TEIXEIRA, Dr., UFU
Componente da Banca

ZSOLT LÁSZLO KOVÁCS, Ph.D., USP
Componente da Banca

FERNANDO ANTONIO CAMPOS GOMIDE, Ph.D., UNICAMP
Componente da Banca

CAMPINA GRANDE - PB
Novembro - 1996

AGRADECIMENTOS

Ao professor orientador: Dr. Narpat Singh Gehlot, pela paciência ao longo dos anos de trabalho conjunto, bem como pelo apoio e incentivo dados durante a elaboração desta Tese.

Ao CNPq - Conselho Nacional de Desenvolvimento Científico e Tecnológico, pela bolsa de doutorado que possibilitou este trabalho.

Ao CCT e ao DEE, pelo apoio institucional.

À empresa MEGABIT Informática, que gentilmente emprestou as suas instalações e equipamentos para a realização de grande parte deste trabalho.

Ao professor José Homero Feitosa Cavalcanti, pelas proveitosas discussões, que em muito enriqueceram este trabalho de Tese.

Ao professor Benedito Antonio Luciano, pelas dicas para a confecção deste documento em sua forma final.

Ao "Doutor" Ravi, pelas incontáveis vezes em que nos socorreu na impressão dos trabalhos científicos.

AGRADECIMENTOS ESPECIAIS

A meus pais, Juan Javier Alsina (in memoriam) e Odelsia Leonor Sanchez de Alsina, sem cujos gametas esta Tese não teria sido realizada.

RESUMO

Neste trabalho são desenvolvidas técnicas do controle de manipuladores robóticos baseadas em redes neurais artificiais e controle adaptativo. Um resumo sobre a teoria de redes neurais e as técnicas de neurocontrole mais comuns é apresentado. O cálculo da equação dinâmica de manipuladores robóticos é abordado. Uma representação para a equação dinâmica, em termos de diagrama de blocos, é desenvolvida. Com base neste diagrama, dois novos métodos recursivos para cálculo eficiente da matriz de inércia e da equação dinâmica de manipuladores são propostos: o Método dos Caminhos Inversos e o Método dos Caminhos Centrais. Uma forma fatorada para a matriz de inércia é derivada. A equação dinâmica é desenvolvida numa forma Linear-Em-Parâmetros e um método recursivo para cálculo da matriz de regressores é proposto. Uma estrutura modular para controle e adaptação de parâmetros de manipuladores robóticos é desenvolvida, resultando num Controlador Adaptativo Modular, onde para cada junta do manipulador é associado um módulo de controle e adaptação correspondente. É derivada uma forma Linear em Termos Cinemáticos para a equação dinâmica. Usando esta formulação, é proposto um esquema de controle de manipuladores robóticos baseado em redes neurais padronizadas: o Neurocontrolador Modular. Neste esquema, um módulo neural de controle é associado a cada junta do manipulador. Um método de treinamento inverso direto para o neurocontrolador modular é proposto e, baseado em técnicas de controle adaptativo, um esquema de treinamento adaptativo direto é desenvolvido. Para testar as estratégias de controle propostas, foi desenvolvida uma ferramenta para simulação dinâmica de manipuladores, denominada ROBOTLAB, a qual inclui apresentação gráfica de resultados em três dimensões.

ABSTRACT

In this work, robot manipulator control techniques based on artificial neural networks are developed. A theoretical review of Neural Networks and neurocontrollers is presented. A Block Diagram representation of the robot dynamics equation is developed. Based on this representation, two recursive algorithms for efficient computation of the inertia matrix and dynamics equation are proposed: the Inverse Path Method and the Central Path Method. A factored form of the inertia matrix is derived. A Linear-In-Parameter Robot Model is developed and a recursive method for the computation of regressor matrix is proposed. The concept of modular adaptive control of robotic manipulators is presented. In the modular scheme, control and adaptation modules are assigned to each link, allowing the control and adaptation of each link dynamic parameters independent of the other links. A Linear-In-Kinematic-Terms robot model is developed. Based on this model, the concept of Modular Neurocontroller for robotic manipulators is presented. In the proposed scheme, Neural Modules (Standardized Neural Networks) are assigned to each link. The modules are joined in a global control system that can be applied to any serial manipulator. A direct inverse training method for the modular neurocontroller is proposed and, based on adaptive control techniques, a direct adaptive training scheme is developed. In order to test the proposed control schemes, the ROBOTLAB software tool was developed. This tool allows the robot dynamics simulation, including 3D graphics output.

ÍNDICE

LISTA DE FIGURAS E TABELAS

LISTA DE SÍMBOLOS

SÍMBOLOS LATINOS

a_i	Comprimento do elo i.
A^{i}_{i+1}	Matriz de rotação ortogonal entre os referenciais dos elos i e i+1.
$C(q,\dot{q})$	Matriz NxN de termos em velocidade.
c_{ac}	Coeficiente de atrito "Coulomb".
c_{av}	Coeficiente de atrito viscoso.
d_i	Comprimento da junta i.
$f(.)$	Função (mapeamento) de dinâmica direta de um sistema.
$\hat{f}(.)$	Mapeamento aprendido por rede neural, aproximação de f(.).
$f_a(.)$	Função de ativação de um neurônio.
$f_e(.)$	Função de erro a ser otimizada por treinamento de neurocontrolador.
f_i	Vetor 3x1 de força aplicada no elo i pelo elo i-1.
$f_{d(i)}(.)$	Mapeamento direto (entrada-saída) do módulo neural i.
$f_{r(i)}(.)$	Algoritmo de propagação retroativa de erro através do módulo neural i.
f_R	Vetor 3x1 de força resultante atuando num corpo rígido.
F_i	Vetor 6x1 de esforço espacial do elo i.
F_{ci}	Vetor 6x1 de esforço espacial coriolis centrífugo do elo i.
F_m	Vetor global 6Nx1 de esforços espaciais inerciais.
F_{mi}	Vetor 6x1 de esforço espacial inercial do elo i.
h_c	Tensor de inércia de um corpo rígido em referencial centro de massa..
h_i	Tensor de inércia do elo i expresso no próprio sistema de referência.
$\vec{h}_i$	Vetor 6x1 de inércia do elo i equivalente ao tensor h_i.
H	Matriz global 6Nx6N de parâmetros dinâmicos do manipulador.
$\vec{H}$	Vetor 10Nx1 de parâmetros dinâmicos do manipulador.
$\hat{\vec{H}}$	Vetor 10Nx1 de parâmetros estimados.
$\tilde{\vec{H}}$	Vetor 10Nx1 de erro de estimação de parâmetros.
H_i	Matriz 6x6 de parâmetros dinâmicos do elo i.

$\vec{H}_i$	Vetor 10x1 de parâmetros dinâmicos do elo i.
$[H]_i$	Matriz 6x12 de parâmetros dinâmicos do elo i.
J	Matriz global 6NxN de máscara de juntas.
$J_i^{(p)}$	Vetor 3x1 de máscara de junta i prismática.
$J_i^{(r)}$	Vetor 3x1 de máscara de junta i rotacional.
J_i	Vetor 6x1 de máscara de junta i.
J_q	Índice de desempenho quadrático para rastreamento de trajetória.
K	Matriz global 6Nx10N de regressores de elos.
K_d	Matriz NxN de ganhos de realimentação derivativa.
K_{d_i}	Ganho derivativo para a junta i.
K_D	Matriz diagonal NxN de ganhos derivativos.
K_P	Matriz diagonal NxN de ganhos proporcionais.
K_i	Matriz 6x10 de regressores do elo i.
$\vec{K}_i$	Vetor 12x1 de termos cinemáticos do elo i.
K_{ri}	Matriz 6x10 de regressores de referência do elo i.
$\vec{K}_{ri}$	Vetor 12x1 de termos cinemáticos de referência do elo i.
L	Função de Lyapunov.
m_c	Massa de um corpo rígido.
$M(q)$	Matriz de inércia NxN.
M_i	Coluna i da matriz de inércia
m_i	Massa total do elo i.
N	Número de juntas de um manipulador.
No	Número de neurônios na camada oculta.
n_i	Vetor 3x1 de conjugado aplicado no elo i pelo elo i-1.
n_R	Vetor 3x1 de momento resultante atuando num corpo rígido.
p_{ci}	Vetor de posição do centro de massa do elo i.
${}^i p_{i+1}$	Vetor de Posição do sistema de referência do elo i+1 em relação ao elo i.
q	Vetor Nx1 de variáveis de juntas (ângulos ou deslocamentos).
q^*	Vetor Nx1 de variáveis de juntas desejadas.
q_c	Vetor Nx1 de trajetória de referência de juntas.
q_i	Variável de posição (deslocamento ou ângulo) da junta i.

q_i^*	Variável de posição desejada da junta i.
q_r	Vetor Nx1 de posições de referência de junta.
R	Matriz global 6Nx6N de rotações espaciais.
R_{i+1}^{i}	Matriz 6x6 de rotação espacial do elo i para o elo i+1.
s	Vetor Nx1 de superfície de erro do manipulador.
t	Tempo.
T	Período de amostragem.
tanh(.)	Função tangente hiperbólica.
U	Entrada de sistema dinâmico.
$\dot{v}_c$	Vetor 3x1 de aceleração linear do centro de massa de um corpo rígido.
$\dot{V}_{ci}$	Vetor 6x1 de aceleração espacial coriolis/centrífuga do elo i.
$\dot{V}_{cr_i}$	Vetor 6x1 de aceleração espacial coriolis/centrif. de referência do elo i.
v_i	Vetor 3x1 de velocidade linear do elo i.
V_i	Vetor 6x1 de velocidade espacial do elo i.
$\dot{V}_m$	Vetor global 6Nx1 de acelerações espaciais inerciais.
$\dot{V}_{mi}$	Vetor 6x1 de aceleração espacial inercial do elo i.
v_{ri}	Vetor 3x1 de velocidade linear de referência do elo i.
V_{ri}	Vetor 6x1 de velocidade espacial de referência do elo i.
W	Vetor de pesos de um neurônio.
$W^{(k)}$	Matriz de pesos da camada (k).
w_0	Limiar de ativação de um neurônio.
$w_{0(i)}^{(k)}$	Limiar de ativação do neurônio (i) na camada (k).
$W_0^{(k)}$	Vetor de limiares de ativação da camada (k).
w_i	Peso associado à entrada i de um neurônio.
$W_{(i)}^{(k)}$	Vetor de pesos do neurônio (i) na camada (k).
X_e	Vetor de entradas de um neurônio.
x_{ei}	Entrada i de um neurônio.
x_L	Combinação linear das entradas de um neurônio.
$X^{(k)}$	Vetor de saídas da camada (k).
$X_L^{(k)}$	Vetor de combinações lineares do vetor de entrada da camada (k).
x_s	Saída de um neurônio.

Y	Saída de sistema dinâmico.
$\hat{Y}$	Saída de sistema dinâmico identificada por rede neural emuladora.
Y^*	Saída desejada para um sistema dinâmico.
Y_k	Entrada de treinamento de neurocontrolador inverso direto de corrente.
Y_k^*	Entrada de referência de neurocontrolador inverso direto de corrente.

SÍMBOLOS GREGOS

α_i	Ângulo de torção do elo i.
Γ	Matriz 10Nx10N de ganhos de adaptação.
Γ_i	Matriz 10x10 de ganhos de adaptação do elo i.
$\delta^{(k)}$	Vetor de derivadas do erro quadrático para a camada (k).
ΔF_i	Vetor 6x1 de erro de saída de módulo neural i.
ε^2	Erro quadrático de saída de um neurônio ou uma rede neural.
$\boldsymbol{\varepsilon}^{(k)}$	Vetor de erros de saída da camada (k) de uma rede neural.
θ_i	Ângulo da junta i.
Λ	Matriz NxN de ganhos de realimentação positiva definida.
μ	Coeficiente de aprendizado.
ζ	Coeficiente de amortecimento.
$\sigma(.)$	Função sigmóide escalar ou vetorial.
τ	Vetor Nx1 de esforços de acionamento (forças ou conjugados).
$\tau_0(q,\dot{q})$	Vetor Nx1 de reações coriólis, centrífugas, gravitacionais e perturbações.
τ_a	Vetor Nx1 de dinâmica não modelada (atrito, perturbações, etc.).
$\tau_c(q,\dot{q})$	Vetor Nx1 de reações coriólis e centrífugas.
$\tau_g(q)$	Vetor Nx1 de reações gravitacionais.
τ_i	Esforço de acionamento (força ou conjugado) da junta i.
$\hat{\tau}_i$	Saída do módulo neural i correspondente à entrada $\bar{K}_i$.
τ_m	Vetor Nx1 de reações inerciais.
τ_{pd}	Vetor Nx1 de realimentação proporcional-derivativa.
$\tau^{(0)}$	Vetor Nx1 solução de Newton-Euler para acelerações de junta = 0.

$\tau^{(i)}$	Vetor Nx1 solução de Newton-Euler para aceleração = 1 na junta i .
$\Phi(q,\dot{q},\dot{q},\ddot{q})$	Matriz Nx10N de regressores do manipulador.
Φ_r	Matriz Nx10N de regressores de referência do manipulador.
Φ	Matriz de regressores.
ω_i	Vetor 3x1 de velocidade angular do elo i.
Ω_i	Vetor 6x1 de velocidade angular espacial do elo i.
ω_n	Frequência natural.
ω_{r_i}	Vetor 3x1 de velocidade angular de referência do elo i.
Ω_{r_i}	Vetor 6x1 de velocidade angular espacial de referência do elo i.
$(\omega\times)$	Operador matriz 3x3 produto vetorial sobre o vetor ω 3x1.
$(\bullet\omega)$	Operador matriz 3x6 produto escalar sobre o vetor ω 3x1.
$(\Omega\times)$	Operador matriz 6x6 produto vetorial sobre o vetor Ω 6x1.
$\vec{\omega}_i^2$	Vetor 6x1 de produtos de velocidades.
$(\times h\bullet)$	Operador matriz 3x6 produto vetorial-escalar sobre a matriz h 3x3.
$[\tau,F]_{ij}$	Matriz de transferência 1x6 associada ao caminho de F_j até τ_i.
$[\tau,\vec{H}]_{ij}$	Matriz de transferência 1x10 associada ao caminho de $\vec{H}_j$ até τ_i.
$[\tau_m,F_m]_{ij}$	Matriz de transferência 1x6 associada ao caminho de F_{m_j} até τ_{m_i}.
$[\tau_m,\dot{V}_m]_{ij}$	Matriz de transferência 1x6 associada ao caminho de $\dot{V}_{m_j}$ até τ_{m_i}.
$[F_m,\dot{V}_m]_{ij}$	Matriz de transferência 6x6 associada ao caminho de $\dot{V}_{m_j}$ até F_{m_i}.
$[F_m,\ddot{q}]_{ij}$	Matriz de transferência 6x1 associada ao caminho de $\ddot{q}_j$ até τ_{m_i}.
$[\Phi]_{[i,j]}$	Submatriz 1x10, partição i, j da matriz de regressores Φ.

CAPÍTULO 1

INTRODUÇÃO GERAL

1.1 Objetivo

Desde a década passada, um vigoroso esforço de pesquisa tem sido direcionado ao estudo das redes neurais artificiais. As redes neurais são constituídas por unidades computacionais denominadas neurônios artificiais, os quais, ao serem interligados massivamente em paralelo, apresentam propriedades coletivas tais como a capacidade de aprendizado. Esta interessante propriedade despertou o interesse dos pesquisadores no sentido de desenvolver as técnicas necessárias para aplicar as redes neurais artificiais no controle de sistemas dinâmicos não lineares, nos quais as técnicas de controle clássico, geralmente, não são suficientes para garantir satisfatoriamente as especificações de desempenho [Narendra & Parthasarathy, 1990]. Os manipuladores robóticos, sistemas altamente não lineares e acoplados, são exemplos particularmente adequados para o teste de estratégias de controle baseadas em redes neurais. Neste sentido, diversos neurocontroladores foram desenvolvidos, a maioria dos quais para casos particulares de manipuladores simples, com pequeno número de juntas. Para manipuladores com maior número de juntas, o projeto dos neurocontroladores esbarra no grande tamanho das redes neurais necessárias para a implementação dos mesmos, além de um período de aprendizado crescente. Assim, no panorama atual de pesquisas na área, pode-se verificar a ausência de uma abordagem genérica e sistemática no projeto de neurocontroladores para manipuladores robóticos. Dentro deste contexto, o objetivo do presente trabalho é apresentar uma contribuição ao desenvolvimento de técnicas sistemáticas e padronizadas de projeto de neurocontroladores para manipuladores robóticos, independentemente do tipo e número de graus de liberdade dos mesmos, fornecendo subsídios para uma escolha mais criteriosa da estrutura e dos parâmetros das redes neurais utilizadas.

1.2 Metodologia

Esta tese está baseada no conceito de controle modular de manipuladores robóticos, que propõe a utilização de módulos de controle independentes para acionamento das diferentes juntas de um dado manipulador. Neste sentido, controladores adaptativos e neuroadaptativos foram desenvolvidos com base nesta filosofia. Para verificar o desempenho dos controladores propostos e, devido aos altos custos de montagens experimentais nesta área, optou-se por testar as técnicas de neurocontrole desenvolvidas através de simulações computacionais.

A simulação dinâmica de um manipulador robótico requer o cômputo da equação dinâmica do mesmo que, dependendo do número N de juntas, pode resultar numa carga computacional excessiva. Neste sentido, foram desenvolvidos algoritmos, para cálculo da equação dinâmica de manipuladores seriais, caracterizados por um esforço computacional de ordem N^2. Visando simular a dinâmica de manipuladores robóticos seriais de maneira bem realista, incluindo a maior parte dos efeitos dinâmicos encontrados na prática, foi desenvolvida uma ferramenta denominada ROBOTLAB. Esta ferramenta é um software para simulação dinâmica de manipuladores robóticos em ambiente Windows, incluindo saída gráfica com animação em três dimensões.

Visando o projeto sistemático de controladores neuroadaptativos para manipuladores robóticos, optou-se por dividir os mesmos em módulos padronizados, de maneira que, para cada junta do manipulador, é associado um módulo de controle correspondente, com entradas e saídas bem definidas. Esta abordagem modular é, em particular, bastante útil no desenvolvimento do neurocontrolador, já que a tarefa de projeto é, desta maneira, dividida no projeto de redes neurais de menores dimensões: os módulos neurais de controle. Estes módulos são redes neurais bastante pequenas cujos parâmetros e estrutura podem ser determinados com mais facilidade, já que as entradas e saídas são padronizadas. Os módulos podem ser interligados de maneira a implementar um neurocontrolador para qualquer manipulador serial de N juntas.

1.3 Revisão Bibliográfica

1.3.1 Introdução

A seguir apresenta-se uma revisão bibliográfica descrevendo o atual panorama das pesquisas na área de controle baseado em redes neurais, dinâmica, simulação, controle adaptativo e neurocontrole de manipuladores, de maneira a situar adequadamente o presente trabalho em relação ao cenário de pesquisa nacional e internacional.

1.3.2 Redes Neurais e Neurocontroladores

A partir da década passada, houve um grande aumento de interesse na pesquisa de sistemas dotados de capacidade de aprendizado, usando modelos inspirados em neurônios biológicos, as chamadas Redes Neurais Artificiais [Hecht-Nielsen, 1990], [Hertz, et alii, 1991], [Kovács, 1996]. As redes neurais são constituídas de elementos adaptativos não lineares, usualmente chamados de Neurônios Artificiais, os quais possuem um elevado grau de interconexão mútua. Apesar da limitada capacidade de processamento de informação de um neurônio individual, a interconexão de um grande número deles resulta num sistema massivamente paralelo, com capacidade de aprender complicadas relações não lineares. Diferentes tipos de redes neurais foram modeladas nas últimas décadas. Algumas das mais conhecidas são o Perceptron [Rosenblatt, 1962], Adaline [Widrow, 1962] e as redes recorrentes de Hopfield [Hopfield, 1982]. Em particular, redes neurais baseadas em neurônios Adaline tipo sigmóide [Rummelhart, et alii, 1986] tornaram-se bastante utilizadas em aplicações em controle automático, após o desenvolvimento do algoritmo de propagação retroativa de erro [Rummelhart, et alii, 1986], o qual tornou-se o método mais difundido para treinamento de redes neurais multi-camadas.

A capacidade de aprender complicados mapeamentos não lineares torna as redes neurais artificiais candidatas promissoras para aplicações em controle de sistemas não lineares. Neste sentido, desde os trabalhos pioneiros de Werbos [Werbos, 1989], Narendra e Parthasarathy [Narendra & Parthasarathy, 1990], consideráveis esforços de pesquisa foram direcionados visando o desenvolvimento de técnicas de controle dinâmico baseadas em redes neurais. Desta maneira, surgiu um novo e amplo campo de pesquisas denominado Neurocontrole. Foram propostas diversas maneiras de incluir redes neurais no controle de sistemas dinâmicos, seja para identificação, seja associadas a controladores convencionais, ou mesmo desempenhando a função de controladores. Neste último caso, as redes neurais são chamadas mais apropriadamente de Neurocontroladores.

A questão da aplicação da capacidade de auto-aprendizado das redes neurais em sistemas de controle é abordada de maneira clara e concisa por Nguyen [Nguyen & Widrow, 1991]. Os problemas associados à implementação de neurocontroladores do tipo adaptativo e do tipo "por aprendizado" são discutidos por Tanomaru [Tanomaru & Omatu, 1991], [Tanomaru & Omatu, 1992] e Yamada [Yamada & Yabuta, 1991], [Yamada & Yabuta, 1993]. Narendra discute em detalhes o uso de métodos baseados em gradiente para otimizar o desempenho de sistemas dinâmicos que incluem redes neurais [Narendra & Parthasarathy, 1991]. O problema de treinamento de redes neurais em aplicações de servocontrole é abordado por Low [Low, et alii, 1993], que propõe um método eficiente para treinar neurocontroladores de sistemas não lineares. Um método, baseado em lógica nebulosa, para determinação on-line dos coeficiente de aprendizado e adaptação de neurocontroladores é apresentado por Cavalcanti [Cavalcanti, 1995].

Acompanhando o aprimoramento das técnicas de otimização de sistemas dinâmicos contendo redes neurais, foram publicados inúmeros trabalhos relativos ao uso de redes neurais nas mais diversas áreas de controle automático: identificação de sistemas [Babu & Eswaran, 1992], [Mistry & Nair, 1994], [Sira-Ramirez & Zak, 1991], [Chen, et alii, 1990]; controle robusto [Qing, 1992]; controle

ótimo [Zhang, et alii, 1992]; rejeição de perturbações [Mukhopadhyay & Narendra, 1993]; controle baseado em lógica nebulosa [Werbos, 1993], [Kim, et alii, 1993]; etc.

Acompanhando os desenvolvimentos teóricos em neurocontrole, diversas aplicações foram propostas, particularmente na área de engenharia elétrica. Diversos exemplos de neurocontroladores de corrente em sistemas eletrônicos industriais são encontrados na literatura. Dois neurocontroladores, (um Inverso Direto e outro Adaptativo Indireto), para controle de corrente de retificadores, são descritos em [Pinheiro, et alii, 1995]. Tai, [Tai, et alii, 1990], apresenta um neurocontrolador inverso direto para regulagem de corrente, tensão e velocidade de um motor de corrente contínua. Em [Patton, 1995] descreve-se um neurocontrolador de corrente para motor de corrente contínua sem escovas, treinado para emular a dinâmica de um controlador Proporcional-Integral. Em comparação com o controlador Proporcional-Integral emulado, após o treinamento, este neurocontrolador apresenta desempenho dinâmico semelhante, com as vantagens de adaptabilidade, maior robustez a variações paramétricas e baixa sensibilidade a ruídos. A estabilidade de um neurocontrolador de corrente para inversor PWM é analisada em [Itoh, et alii, 1990]. Em [Song, et alii, 1991] apresenta-se um neurocontrolador adaptativo de corrente estatórica para motores de indução, que apresenta distorção harmônica consideravelmente menor que controladores por histerese convencionais. Neurocontroladores para a corrente estatórica de motores de indução, baseados na configuração de controle inverso direto, são apresentados em [Alsina & Gehlot, 1992-A], [Alsina & Gehlot, 1993-C]. Um neurocontrolador adaptativo direto para a corrente estatórica de motores de indução é descrito em detalhes em [Alsina & Gehlot, 1995-E] e [Alsina & Gehlot, 1995-B]. Sistemas de controle neuroadaptativos para acionamento de motores de indução baseados em campo orientado foram sugeridos[Ba-Razzouk, et alii, 1995], [Alsina & Gehlot, 1993-D].

1.3.3 Dinâmica e Simulação de Manipuladores

Os manipuladores robóticos são sistemas mecânicos constituídos por corpos rígidos interligados por juntas (rotacionais ou prismáticas) acionadas por atuadores. Matematicamente, a dinâmica destes sistemas é descrita por uma equação diferencial vetorial não linear de segunda ordem. Assim, a dinâmica inversa de um manipulador genérico de N juntas é dada por:

$$\tau = M(q).\ddot{q} + \tau_0(q,\dot{q}) \qquad (1.1)$$

onde,

τ = Vetor Nx1 de esforços de acionamento (forças ou conjugados).
q = Vetor Nx1 de variáveis de juntas (ângulos ou deslocamentos).
$M(q)$ = Matriz de Inércia NxN.
$\tau_0(q,\dot{q})$ = Vetor Nx1 de reações coriólis, centrífugas, gravitacionais e perturbações.

A dinâmica direta é obtida invertendo a Equação (1.1):

$$\ddot{q} = M(q)^{-1}(\tau - \tau_0(q,\dot{q})) \qquad (1.2)$$

O custo de montagens experimentais na área de robótica é elevado, tornando difícil a sua adquisição por parte de instituições de ensino e pesquisa no Brasil. A utilização de simulação computacional de sistemas robotizados é uma solução alternativa de baixo custo para aplicações de ensino e pesquisa na área de robótica. A simulação dinâmica de manipuladores envolve a integração numérica da equação (1.2) de maneira a obter a evolução temporal do vetor de posições de junta q produzido por um vetor de esforços de acionamento correspondente. A obtenção da equação de dinâmica direta ou inversa em fórmula fechada é um procedimento tedioso, sendo praticável para manipuladores de duas ou três juntas. Para manipuladores de mais graus de liberdade a obtenção manual da fórmula fechada torna-se impraticável. É claro que métodos de computação simbólica podem ser utilizados com sucesso, mas a

equação resultante normalmente ocupa algumas páginas de símbolos, de maneira que métodos numéricos são mais adequados.

Historicamente, os dois métodos clássicos para obtenção da equação de dinâmica inversa de manipuladores robóticos são o Método Lagrange [Uicker, 1967] e o método Newton-Euler [Luh, et alii, 1980]. O primeiro, permite obter separadamente os termos da equação dinâmica: matriz de inércia, reações coriolis e centrífugas e reação gravitacional. Uma desvantagem deste método é a elevada carga computacional, que aumenta numa proporção N^4. Assim, para manipuladores com muitos graus de liberdade, a carga computacional torna-se enorme. Por outro lado, o Método Newton-Euler, que tornou-se popular a partir dos anos oitenta, é um método recursivo para o calculo dos conjugados de acionamento (dinâmica inversa), apresentando uma carga computacional proporcional ao número de juntas, sendo assim mais apropriado para aplicações em tempo real. Este método apenas calcula numericamente o vetor de esforços de acionamento, apresentando a desvantagem de não manter a estrutura da equação dinâmica.

Modelos dinâmicos simplificados, apresentando carga computacional menor, foram propostos. Por exemplo, Li [Li, 1989], apresenta um método de linearização da dinâmica de manipuladores baseado no método Lagrange. É claro que o preço a pagar pela simplificação é uma menor precisão na modelagem, de maneira que foram realizados esforços de pesquisa visando obter modelos acurados que apresentassem redução na carga computacional. Uma boa análise comparativa de vários métodos de cálculo da equação dinâmica é encontrada em [Zomaya, 1991]. Hollerbach [Hollerbach, 1980] apresentou uma forma recursiva para o método Lagrange com carga computacional de ordem N, mas perdendo informação relativa à estrutura da equação dinâmica. Baharin [Baharin & Green, 1991] e Li [Li, 1988] apresentaram métodos de cálculo da equação dinâmica baseados no método Lagrange com carga computacional de ordem N^3, mantendo a estrutura do modelo. Métodos mais eficientes [Lilly & Orin, 1991] , [Featherstone, 1987] , apresentando esforços computacionais de ordem N^2 foram desenvolvidos.

Modelos mais eficientes para aplicações específicas foram desenvolvidos. Um método de computação eficiente para a dinâmica inversa e a matriz de inércia, reorganizando as equações visando um conjunto mínimo de parâmetros dinâmicos, foi proposto por Kawasaki [Kawasaki, et alii, 1992]. Este método, orientado para identificação paramétrica, apresenta um esforço computacional de ordem N^2. Um método eficiente para o cálculo da equação dinâmica de manipuladores que incluem cadeias cinemáticas fechadas e atuadores redundantes foi apresentado por Nakamura [Nakamura & Ghoudoussi, 1989].

Formas mais elaboradas para a equação dinâmica foram propostas, visando uma notação mais simples, compacta e elegante, de maneira a possibilitar o uso de novas ferramentas matemáticas. Fang [Fang, et alii, 1994] apresentou um método recursivo para o cálculo da equação dinâmica baseado no conceito de corpos aumentados e sistemas compostos. Park [Park & Borrow, 1994], apresentou um método recursivo, baseado na geometria de Riemann, para o cálculo da equação dinâmica. Utilizando uma transformação de variáveis apropriada, Jain [Jain & Rodriguez, 1994] desenvolveu um método recursivo, baseado nas equações de Lagrange, que resulta em um modelo diagonalizado para a dinâmica de manipuladores.

Na simulação dinâmica de manipuladores é necessário integrar numericamente a Equação (1.2), a qual envolve a inversa da matriz de inércia M(q). Assim, para realizar a simulação de manipuladores é preciso obter explicitamente esta matriz. Como o método Newton-Euler apenas computa numericamente o vetor de esforços de acionamento, em princípio não pode ser utilizado para o cálculo da matriz de inércia, mas Walker [Walker & Orin, 1982] desenvolveu um artifício para possibilitar o cálculo da matriz de inércia a partir deste método recursivo. Balafoutis [Balafoutis & Patel, 1989] apresentou um método eficiente, baseado no conceito de elos generalizados e aumentados, para o cálculo da matriz de inércia e da dinâmica direta, apresentando carga computacional de ordem N^2. Alsina e Gehlot [Alsina & Gehlot, 1996-C], [Alsina & Gehlot, 1994-A], [Alsina & Gehlot, 1994-D] desenvolveram dois métodos para cálculo recursivo da matriz de inércia, com carga computacional de ordem N^2, baseados numa representação em diagrama de blocos da equação dinâmica.

Ao lado dos métodos acima, foram apresentados métodos para obtenção da dinâmica direta (Equação 1.2) que não requerem o cálculo da matriz de inércia. A partir do método Newton-Euler, Gougoussis [Gougoussis & Donat, 1990] desenvolveu um método para solução em tempo real da dinâmica direta, baseado em laços algébricos resolvidos através de hardware analógico. Ko [Ko, 1992] apresentou um método recursivo para solução da dinâmica direta, a qual é formulada de maneira idêntica a um Filtro de Kalman.

Por outro lado, em oposição à filosofia essencialmente algorítmica dos métodos acima, novas arquiteturas de computadores, dedicadas à solução da equação dinâmica de manipuladores, foram investigadas. Javaheri [Javaheri & Orin, 1988] apresentou algumas arquiteturas sistólicas para obtenção da matriz de inércia baseadas em 1, N e N(N+1)/2 processadores VLSI dedicados, conseguindo uma carga computacional de ordem N. Lee [Lee & Chang, 1988] propôs vários algoritmos paralelos para computação em tempo real da dinâmica direta de manipuladores robóticos baseados em Arquitetura SIMS (single-instruction multiple-data-stream), utilizando um número de processadores igual ao número de graus de liberdade do manipulador. O número de processadores ótimo para o cálculo da dinâmica inversa em sistemas de multiprocessadores foi investigado por Lee [Lee & Chen, 1990]. Fijany [Fijani, 1994], [Fijany, et alii, 1994] desenvolveu algoritmos paralelos para o cálculo da matriz de inércia e da dinâmica direta com carga computacional de ordem log(N). Um método paralelo baseado no método Newton-Euler para a obtenção em separado dos diversos termos da equação de dinâmica inversa foi proposto por Pu [Pu & Eldin, 1994].

A simulação computacional pode ser implementada para emular realisticamente os efeitos dinâmicos envolvidos no funcionamento de manipuladores. Simuladores dinâmicos de manipuladores são ferramentas poderosas com aplicações em várias áreas. No âmbito da pesquisa, a simulação dinâmica permite o estudo de estratégias de controle de robôs sem a necessidade de um protótipo real [Iñigo & Morton, 1991]. A simulação pode ser utilizada ainda para programação off-line de

manipuladores [Schroer & Tech, 1986], [Callan, 1986] e testes de funcionamento [Chevasin & Premier, 1984]. A simulação gráfica, incluindo animação em 3D é uma poderosa ferramenta para o ensino introdutório de robótica; neste sentido diversos simuladores foram desenvolvidos [Raz, 1989], [White, et alii, 1989]. Na manufatura flexível, a simulação gráfica permite a rápida programação de tarefas off-line e testes de funcionamento da célula de trabalho. Por outro lado, o uso de interfaces gráficas em telerobótica está se tornando uma ferramenta promissora para controle amigável de robôs à distância [Agba, 1995].

Dois pacotes de software para simulação dinâmica e gráfica de manipuladores robóticos, destinados a aplicações de ensino e pesquisa, merecem destaque: *Robotica* [Nethery & Spong, 1989] e *Robotics Toolbox* [Corke, 1996]. *Robotica*, um pacote de software para análise, simulação dinâmica e animação de robôs baseado na linguagem *Mathematica*, faz uso extensivo das capacidades de processamento simbólico da mesma. *Robotics Toolbox* é uma biblioteca de funções desenvolvidas como uma extensão da linguagem *Matlab*, voltada essencialmente para a área de robótica, com propósitos semelhantes ao pacote *Robotica*. O software ROBOTLAB [Alsina & Gehlot, 1995-C], [Alsina & Gehlot, 1996-A], [Alsina & Gehlot, 1996-B], apresentado nesta tese, inclui, além de um pacote para simulação dinâmica e gráfica de manipuladores robóticos, várias bibliotecas de funções relativas a diversas áreas da robótica, tais como: cinemática, dinâmica, geração de trajetória, controle, etc. sendo uma ferramenta útil, tanto na área de pesquisa como na área de ensino. ROBOTLAB e *Robotics Toolbox*, ao contrário de *Robotica* (baseado em soluções simbólicas), são baseados em soluções numéricas, mais eficientes do ponto de vista computacional. Apesar de apresentar um conjunto de funções básicas ligeiramente inferior a *Robotics Toolbox*, a capacidade de animação gráfica de ROBOTLAB é bastante superior, tanto a *Robotics Toolbox* como a *Robotica*. Ao contrário de *Robotics Toolbox*, ROBOTLAB e *Robotica* apresentam interfaces amigáveis, baseadas em menus e janelas auto-explicativos, que simplificam a interação com o usuário.

1.3.4 Controladores Adaptativos para Manipuladores

Devido alto grau de não linearidade, o controle acurado de manipuladores robóticos é um problema bastante complexo. Para a sua solução, diversas estratégias de controle têm sido propostas, desde o uso de simples controladores PID (Proporcional-Integral-Derivativo) até sofisticadas técnicas de controle adaptativo não linear. Para a maioria dos manipuladores industriais usados na atualidade, dotados de engrenagens de redução (que normalmente aumentam o grau de desacoplamento da dinâmica), controladores independentes para acionamento das juntas [Hsia, et alii, 1991] são suficientes para o controle de posição, embora a precisão não seja muito boa em aplicações de rastreamento de trajetória. Para requisitos mais severos de rastreamento de trajetória, estratégias de controle baseadas em modelo dinâmico, tal como Torque Computado [Alici & Daniel, 1993], apresentam melhor desempenho que os controladores PID independentes, embora que pagando o preço de aumento na carga computacional [Yoo, et alii, 1991].

Para manipuladores com acionamento direto (sem reduções), quando os requisitos de rastreamento de trajetória são severos, a precisão do modelo dinâmico utilizado no controlador afeta diretamente o desempenho do mesmo. Por este motivo técnicas de estimação de parâmetros e controle adaptativo são, geralmente, requeridas. As técnicas de controle adaptativo de manipuladores podem ser divididas em dois grandes grupos: técnicas que são baseadas em modelo dinâmico e técnicas que não são baseadas em modelo dinâmico. As técnicas que não se baseiam em modelo dinâmico requerem menor esforço computacional [Song, 1994] e, geralmente, o controle é decentralizado (controladores independentes para as juntas) [Seraji, 1989]. A pesar do menor custo computacional, estes métodos são menos precisos do que aqueles baseados em modelo. Alguns controladores adaptativos propostos são baseados em modelos linearizados [Liu & Lin, 1988], [Guo & Angeles, 1989], o que permite o uso de técnicas de controle clássico, mas por outro lado, a precisão só é boa para desvios pequenos em relação à trajetória nominal.

A formulação da equação dinâmica de manipuladores em forma Linear-Em-Parâmetros [Khosla, 1989] permitiu um grande avanço no controle adaptativo de manipuladores robóticos. Desde então, vários controladores adaptativos baseados no modelo não linear completo foram propostos para o controle de manipuladores [Kim & Hori, 1995]. O primeiro controlador deste tipo foi apresentado por Craig [Craig, et alii, 1986]. Este controlador era basicamente uma versão adaptativa, globalmente estável, do método Torque Computado. A principal desvantagem deste método era a necessidade de medição das acelerações de junta. Outra versão do método Torque Computado Adaptativo, utilizando uma lei Proporcional-Integral para identificação de parâmetros, é apresentada em [Kelly, et alii, 1989]. Slotine [Slotine & Li, 1988] apresentou um controlador adaptativo baseado em modelo, globalmente estável, que dispensa a medição das acelerações de junta. Este controlador adaptativo serviu de inspiração para inúmeros outros e mostrou excelentes resultados nos testes experimentais [Dessaint, et alii, 1990], [Dessaint, et alii, 1992]. Bayard [Bayard & Wen, 1988] apresentou uma série de novos controladores adaptativos baseados em modelo dinâmico, globalmente estáveis segundo Lyapunov. Stepanenko [Stepanenko & Su, 1994] apresentou um controlador adaptativo baseado em modelo, incluindo a dinâmica dos atuadores (motores elétricos). Técnicas de controle robusto foram utilizadas por Reed [Reed & Iannou, 1989] para melhorar o desempenho de controladores adaptativos baseados em modelo na presença de perturbações e melhorar a sua estabilidade. Colbaugh [Colbaugh & Seraji, 1994] descreve um controlador adaptativo baseado em modelo para controle em coordenadas cartesianas, globalmente estável segundo Lyapunov.

Nos últimos tempos, tem sido pesquisada a associação dos controladores adaptativos baseados no modelo dinâmico de manipuladores com sofisticadas técnicas de controle não linear, tais como Sistemas de Estrutura Variável, Lógica Nebulosa e Redes Neurais. Controladores adaptativos associados a sistemas de estrutura variável, projetados para minimizar o efeito de batimento("chattering") foram propostos por Vega [Vega, et alii, 1994] e Yao [Yao & Tomizuka, 1994]. Hsu [Hsu & Fu, 1994], utilizando lógica nebulosa, projetou um controlador adaptativo, robusto, baseado em modelo. Alsina e Gehlot [Alsina & Gehlot, 1995-D] apresentaram um

controlador adaptativo para manipuladores robóticos baseado em redes neurais modulares.

Apesar do bom desempenho, os controladores adaptativos baseados em modelo pagam o custo da elevada carga computacional envolvida no cálculo do modelo dinâmico. Neste sentido, esforços de pesquisa foram direcionados visando diminuir o esforço computacional envolvido na implementação destes controladores. Temeltas [Temeltas & Asher, 1992] propôs a utilização de regressores independentes para cada elo visando a identificação dos parâmetros dinâmicos de manipuladores através de arquiteturas paralelas, o que permite uma redução considerável do tempo de computação. Alsina e Gehlot [Alsina & Gehlot, 1994-C], [Alsina & Gehlot, 1995-F], apresentaram uma formulação modular para o controlador adaptativo de Slotine [Slotine & Li, 1988], permitindo o cálculo do esforço de controle de maneira recursiva e eficiente.

1.3.5 Neurocontroladores para Manipuladores

O desenvolvimento de arquiteturas VLSI de redes neurais [Kung & Hwang, 1989], voltadas para aplicações em robótica, torna viável o uso das mesmas na solução de uma amplia gama de problemas na área, desde o planejamento de movimentos [Gambarella & Haex], cinemática inversa [Alsina & Gehlot, 1994-E], [Alsina & Gehlot, 1995-A], controle de trajetória [Chin, et alii, 1992], controle híbrido de força e posição [Connolly & Pfeiffer, 1993], até sofisticadas técnicas de controle de movimento baseadas em realimentação de visão [Martinetz, et alii, 1990], [Walter & Schulten, 1993]. Nestes problemas, inerentemente não lineares, tenta-se aproveitar ao máximo a capacidade de aprender complexos mapeamentos não lineares das redes neurais, bem como o grande poder computacional decorrente do paralelismo massivo associado às mesmas.

Um grande número de neurocontroladores para controle de trajetória de manipuladores pode ser encontrado na literatura, embora que a maioria dos

estudos se restrinja a braços de robô simples, de um ou dois elos [Kuperstein & Wang, 1990], [Yuan, et alii, 1994], [Alsina & Gehlot, 1992-B], [Zomaya, 1993]. Uma análise comparativa de neurocontroladores para robôs de dois elos pode ser encontrada em [Alsina & Gehlot, 1994-B]. Chen [Chen, et alii, 1993] apresenta um esquema de controle para manipulador de duas juntas, no qual um controlador torque computado é associado a uma rede neural para compensação de erros de modelagem. Também para o controle de um robô de duas juntas, Ozaki [Ozaki, et alii, 1991] apresenta um esquema que utiliza redes neurais para identificar as matrizes do modelo dinâmico do manipulador, de maneira a implementar um compensador não linear semelhante ao controlador torque computado. O erro de realimentação é utilizado para treinar as redes neurais. Saad [Saad, et alii, 1994] implementou um neurocontrolador inverso direto para manipulador de duas juntas, associado a um controlador Proporcional Derivativo convencional. Liang [Liang & ElMaraghy, 1994] apresenta um neurocontrolador adaptativo direto, que dispensa medição ou estimação das acelerações de junta, no qual várias redes neurais são utilizadas para compensar os termos não lineares da equação dinâmica. Zomaya [Zomaya, 1993] propõe o uso de um esquema de controle para manipulador de duas juntas, no qual uma rede neural, usando aprendizado por reforço, é treinada para compensar as incertezas de modelagem de um controlador convencional baseado em modelo. Alsina [Alsina & Gehlot, 1993-A], [Alsina & Gehlot, 1993-B] apresenta um neurocontrolador inverso direto modular, baseado no método de formulação dinâmica Newton-Euler, para manipuladores de N graus de liberdade. Zeng [Zeng, 1994] descreve os resultados de simulação de um controlador para robô industrial, no qual uma rede neural, associada a um controlador Proporcional-Integral-Derivativo, é utilizada para compensar não linearidades. Sharifi [Sharifi, et alii, 1994] apresenta um neurocontrolador baseado na rede neural CMAC (Cerebellear Model Articulation Controller), funcionando em paralelo com um controlador de ganhos fixos. O neurocontrolador aprende a dinâmica inversa do manipulador por meio do erro de realimentação e é utilizado para compensação de não linearidades. Karakasoglu [Karakasoglu, et alii, 1993] propõe o uso de um conjunto de redes neurais dinâmicas recorrentes para o controle adaptativo decentralizado de manipuladores. Cada rede neural é treinada para aprender o modelo discreto de dinâmica inversa de cada elo independente.

Melhoras de desempenho consideráveis podem ser obtidas utilizando os neurocontroladores junto com outras técnicas avançadas de controle. A associação de neurocontroladores de robôs com técnicas de controle por sistemas de estrutura variável (VSS) permite melhorar o desempenho dos mesmos em termos de estabilidade e robustez [Pei, et alii, 1992], [Zihong, et alii, 1994]. Nos últimos anos, vários trabalhos foram apresentados associando neurocontroladores a lógica nebulosa [Chen & Gill, 1994], [Foo, et alii, 1994], [Pletl, 1995] e a algoritmos genéticos [Mester, 1995], apontando na direção de verdadeiros controladores inteligentes para manipuladores robóticos.

Apesar de um grande número de neurocontroladores eficientes para manipuladores robóticos terem sido propostos, a grande maioria foi desenvolvida para manipuladores de duas juntas. Quando o número de juntas aumenta, verifica-se um correspondente aumento na carga computacional, no tempo de treinamento e no tamanho das redes neurais necessárias para implementar os neurocontroladores. No sentido de diminuir estes problemas, Alsina e Gehlot [Alsina & Gehlot, 1995-D] desenvolveram o conceito de módulos neurais para controle de manipuladores. Nesta técnica, módulos neurais (redes neurais padronizadas de pequeno tamanho), são associados a cada elo do manipulador e interligados num esquema global de controle, emulando a dinâmica inversa de maneira seqüencial análoga ao método Newton-Euler.

1.4 Organização do Trabalho

A seguir, apresenta-se a organização geral desta Tese, de maneira a dar uma visão global do trabalho realizado e facilitar a leitura dos Capítulos seguintes.

No Capítulo 2 é apresentada uma revisão sucinta sobre redes neurais artificiais e as técnicas de controle baseadas nas mesmas. A Seção 2.2 apresenta uma revisão dos principais conceitos envolvidos nas redes neurais: descreve-se a sua unidade básica, o neurônio artificial, apresenta-se a estrutura das mesmas e sumariza-se

o algoritmo de propagação retroativa de erro, utilizado no treinamento da maioria das redes neurais multicamadas. A Seção 2.3 descreve o estado-da-arte das técnicas de neurocontrole. O uso de redes neurais em sistemas de controle é revisado. Apresenta-se o neurocontrolador inverso e as diferentes configurações de treinamento associadas ao mesmo. As conclusões do Capítulo são apresentadas na Seção 2.4.

O Capítulo 3 aborda o problema de formulação dinâmica de manipuladores. Um resumo sucinto sobre cinemática de manipuladores robóticos é apresentado na Seção 3.2. As equações de Dinâmica Inversa e de Dinâmica Direta são apresentadas na Seção 3.3. Na Seção 3.4, o método Newton-Euler para cálculo da dinâmica inversa é descrito e uma forma equivalente baseada em vetores 6x1 é apresentada. A relação das Equações de dinâmica direta e inversa com os problemas de simulação e controle de robôs é destacada na Seção 3.5. Na Seção 3.6, uma representação em forma de diagrama de blocos para a matriz de inércia é desenvolvida e dois métodos para cálculo recursivo da matriz de inércia são propostos [Alsina & Gehlot, 1996-C], [Alsina & Gehlot, 1994-A], [Alsina & Gehlot, 1994-D]. Uma forma fatorada da matriz de inércia é desenvolvida na Seção 3.7. As conclusões relativas ao Capítulo 3 são discutidas na Seção 3.8.

No Capítulo 4 apresenta-se uma estratégia de controle adaptativo modular para manipuladores robóticos. O controlador clássico de manipuladores robóticos Torque Computado, é descrito sucintamente na Seção 4.2. Na Seção 4.3, a dinâmica do manipulador é expressa em forma Linear-Em-Parâmetros e uma forma recursiva para o cálculo da matriz de regressores é apresentada na Seção 4.4. O controlador adaptativo baseado em modelo proposto por Slotine [Slotine & Li, 1988] é sumariamente descrito na Seção 4.5. A prova de estabilidade deste controlador é apresentada na Seção 4.6. Uma estrutura modular para o cálculo em tempo real deste controlador é proposta na Seção 4.7. Resultados de simulação computacional para um manipulador de três juntas são apresentados na Seção 4.8, mostrando o desempenho do controlador modular proposto. Na Seção 4.9 as conclusões relativas ao Capítulo 4 são apresentadas.

No Capítulo 5 são propostas técnicas baseadas em redes neurais modulares para controle de trajetória de Manipuladores Robóticos. Na Seção 5.2, o método Newton-Euler para cálculo recursivo da dinâmica inversa é reformulado em forma Linear em Termos Cinemáticos. Esta formulação é utilizada como base para desenvolver o esquema de neurocontrole modular, o qual é apresentado na Seção 5.3. O método de treinamento dos módulos neurais é discutido na Seção 5.4. Um esquema de treinamento adaptativo direto para o neurocontrolador modular é proposto na Seção 5.5. Resultados de simulação para um braço de três juntas são apresentados na Seção 5.6, verificando o bom desempenho dinâmico do esquema de neurocontrole modular proposto. As conclusões do Capítulo 5 são apresentadas na Seção 5.7.

No Capítulo 6 descreve-se o simulador de robôs ROBOTLAB [Alsina & Gehlot, 1995-C], [Alsina & Gehlot, 1996-A], [Alsina & Gehlot, 1996-B], que é uma ferramenta para simulação dinâmica de manipuladores incluindo animação gráfica em 3D. Na Seção 6.2 o software ROBOTLAB é descrito, apresentando-se as características gerais do mesmo. A interface com o usuário e o formato dos arquivos de entrada e das variáveis de estado são descritos na Seção 6.3. As várias opções de saída de dados em forma gráfica são apresentadas na Seção 6.4. Na Seção 6.5, as diversas funções ROBOTLAB são listadas, dando-se uma descrição sumária das mesmas. Na Seção 6.6 são apresentadas as conclusões relativas ao Capítulo 6.

No Capítulo 7 é feita uma avaliação do trabalho realizado e as conclusões finais são apresentadas. Na Seção 7.2 são destacadas as principais contribuições científicas do trabalho realizado e as principais publicações nacionais e internacionais decorrentes do mesmo. As perspectivas futuras para continuação do trabalho de tese são apresentadas na Seção 7.3. As considerações finais são discutidas na Seção 7.4.

CAPÍTULO 2

REDES NEURAIS E NEUROCONTROLADORES

2.1 Introdução

Neste Capítulo apresenta-se uma revisão sobre os principais conceitos envolvidos no estudo de redes neurais artificiais e o seu uso no controle de sistemas dinâmicos. Na Seção 2.2 são estudadas as redes neurais baseadas em elementos adaptativos lineares tipo sigmóide, "Adaline Sigmóide" [Rumelhart, et alii, 1986]. Na Seção 2.2.1 é apresentado o modelo do neurônio artificial. As características das redes neurais, constituídas por camadas de neurônios interligados, são discutidas na Seção 2.2.2 O algoritmo padrão de aprendizado para redes neurais multicamadas baseado na propagação retroativa de erro [Rumelhart, et alii, 1986] é apresentado na Seção 2.2.3. Na Seção 2.3 discute-se o uso de redes neurais como neurocontroladores de sistemas dinâmicos. Na Seção 2.3.1 é apresentado um resumo das diversas possibilidades de uso de redes neurais em sistemas de controles. Na Seção 2.3.2 é descrito o neurocontrolador inverso. Os esquemas de treinamento mais comuns para o neurocontrolador inverso são apresentados na Seção 2.3.3.

2.2 Redes Neurais Artificiais

2.2.1 O Neurônio Artificial

O Neurônio Artificial é a unidade básica da estrutura de uma rede neural. A estrutura típica de um neurônio artificial é mostrada na Figura 2.1. O vetor de entradas $X_e = [x_{e1} \quad x_{e2} \ldots\ldots \ x_{en}]^T$ é combinado linearmente através dos pesos w_i's, produzindo a saída linear x_L , que por sua vez, passando pela função de ativação $f_a(.)$, resulta na saída do neurônio x_s:

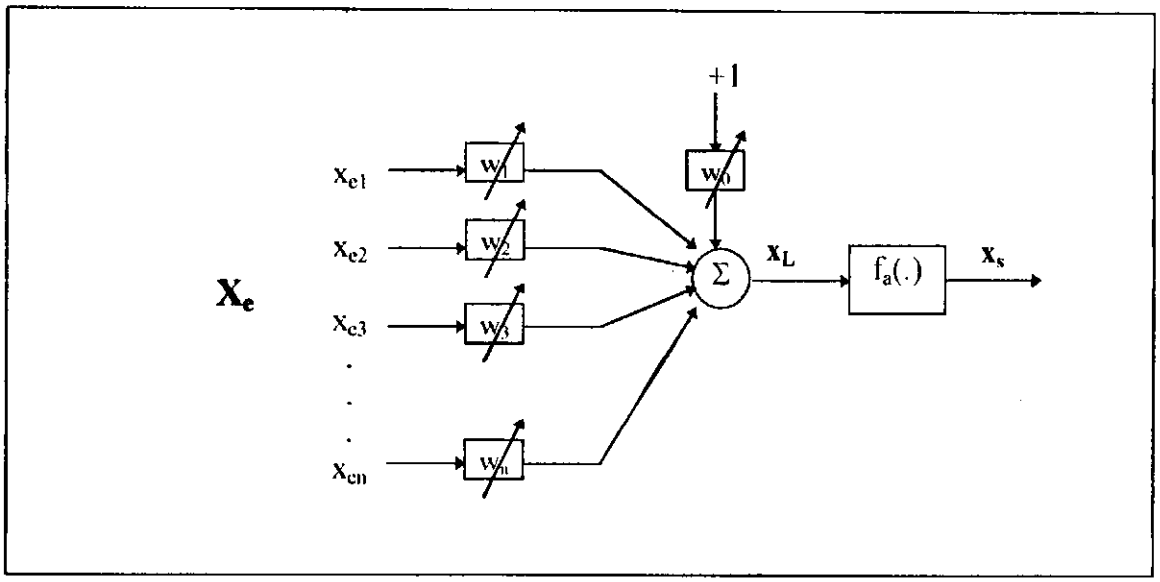

Fig. 2.1 - O Neurônio Artificial.

$$x_L = w_0 + w_1.x_{e1} + w_2.x_{e2} + ... + w_n.x_{en} = W^T.[1 \quad X_e^T]^T \tag{2.1}$$

$$x_s = f_a(x_L) \tag{2.2}$$

onde $W = [w_0 \quad w_1 \quad w_2 \quad .... \quad w_n]^T$ é o vetor de pesos do neurônio e w_0 é chamado de limiar de ativação do mesmo.

A capacidade de processamento de informação de um neurônio artificial é pequena. Um neurônio individual funciona como um classificador linear: dados dois conjuntos diferentes (classes), linearmente separáveis, de valores do vetor X_e, o neurônio, através da sua função de ativação (também chamada de Função Critério) pode determinar se um dado vetor de entradas pertence a uma ou outra classe. Para poder realizar esta tarefa, o neurônio deve "aprender", isto é, os seus pesos devem ser ajustados através de um algoritmo de aprendizado adequado, de maneira a classificar corretamente os diferentes vetores de entrada. A classificação é definida pela saída x_s, assim, diferentes faixas de valores de valores de x_s são associados às diferentes classes.

Os algoritmos de aprendizado podem ser classificados em dois grandes grupos: Aprendizado Supervisionado e Aprendizado Não Supervisionado. No aprendizado supervisionado utiliza-se um conjunto de pares entrada-saída, ou seja, valores do vetor de entrada associados aos valores correspondentes de saída desejados. A saída do neurônio correspondente a uma dada entrada é comparada à saída desejada e a diferença entre as duas é utilizada pelo algoritmo de aprendizado para corrigir os pesos de maneira a reduzir o erro entre as mesmas. No aprendizado não supervisionado não é necessária a apresentação das saídas desejadas para realizar o treinamento.

A função de ativação $f_a(.)$ fornece ao neurônio a capacidade de classificar o vetor de entrada. Diversas funções não lineares foram propostas como função critério: função sinal, função rampa, função quadrática, função sigmóide [Barnard & Casasent, 1989], função logaritmóide simétrica [Goh & Tan, 1994], etc. Quando a função de ativação utilizada é uma sigmóide, o neurônio é denominado Elemento Linear Adaptativo tipo Sigmóide, "Adaline" sigmóide [Rumelhart, et alii, 1986], [Widrow & Lehr, 1990]. A função sigmóide $\sigma(x_L)$, do tipo tangente hiperbólica [Nguyen & Widrow, 1991], que é mostrada na Figura 2.2, é uma função contínua, diferenciável e limitada entre -1 e 1. Matematicamente é expressa por:

$$\sigma(x_L) = \tanh(x_L) = \left(\frac{1 - e^{-2(x_L)}}{1 + e^{-2(x_L)}}\right) \tag{2.3}$$

A sua derivada é dada por:

$$\sigma'(x_L) = \frac{\partial(\tanh(x_L))}{\partial x_L} = 1 - (\tanh(x_L))^2 = 1 - x_s^2 \tag{2.4}$$

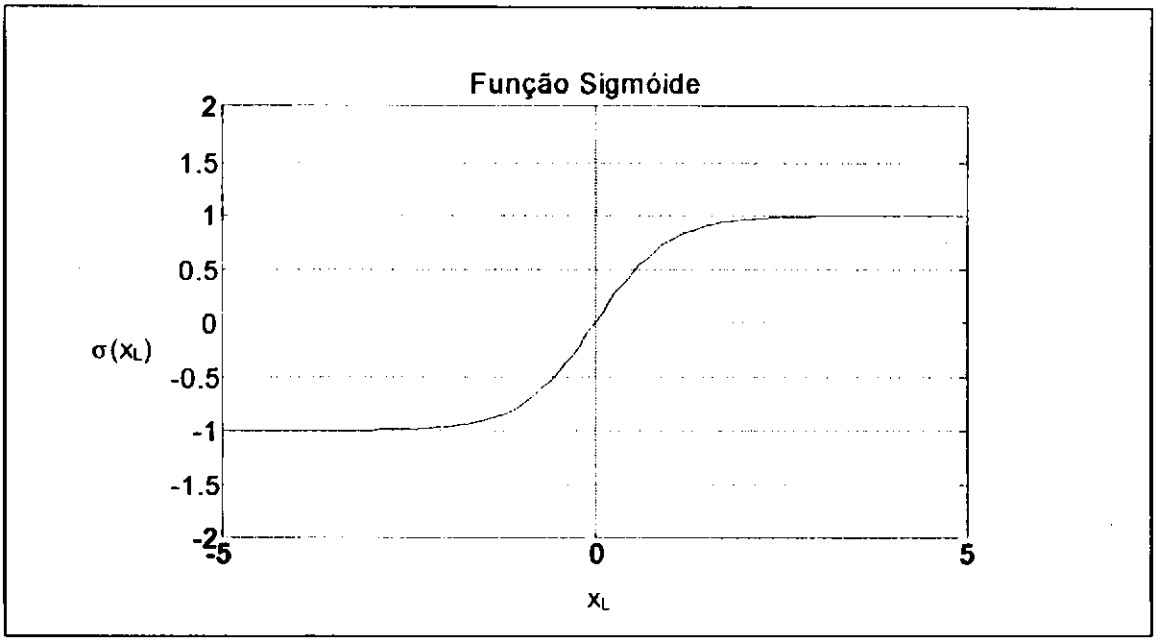

Fig. 2.2 - Função Sigmóide tipo Tangente Hiperbólica.

2.2.2 A Rede Neural Multicamadas

O neurônio artificial, tomado individualmente, possui pequena capacidade computacional. A interligação de um grande número de neurônios permite obter sistemas que apresentam um elevado poder computacional global, capazes de aprender complexos mapeamentos não lineares. Estes sistemas, que tiram vantagem do trabalho em paralelo de neurônios interligados massivamente, são denominados Redes Neurais Artificiais. Em princípio, os neurônios podem ser conectados em inumeráveis topologias. Em particular, uma topologia simples e amplamente utilizada é a rede neural multicamadas, na qual os neurônios são agrupados em camadas sucessivas e a saída de cada um dos neurônios de uma dada camada é ligada apenas a todos os neurônios da camada subseqüente. A Figura 2.3 mostra a estrutura típica de uma rede neural com quatro camadas: uma camada de entrada, duas camadas ocultas e uma camada de saída.

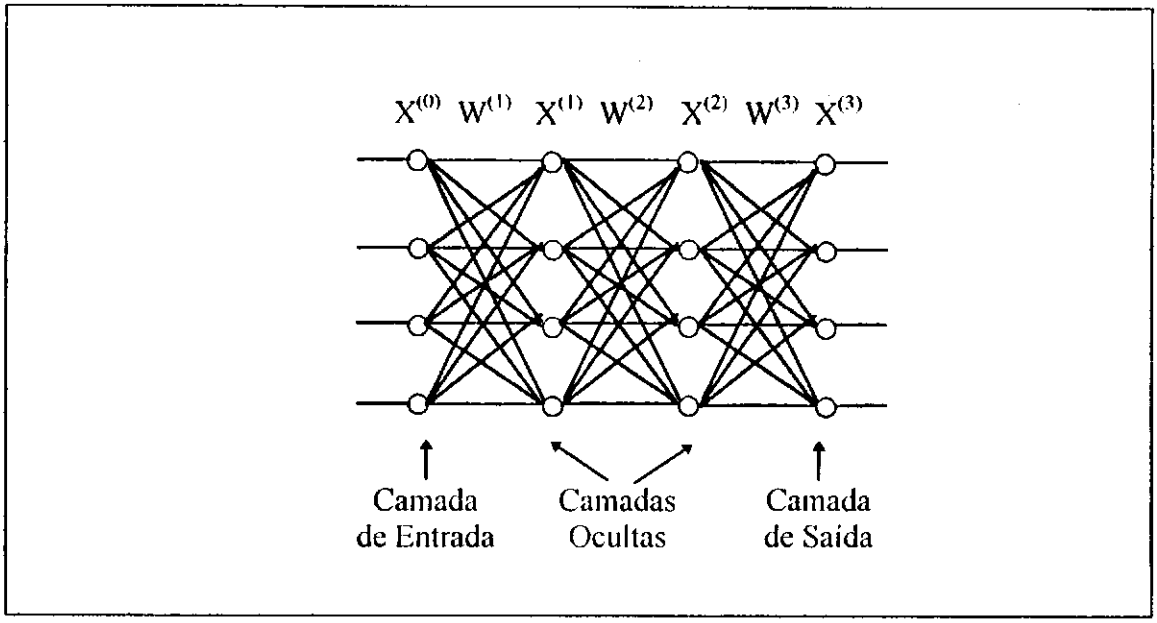

Fig. 2.3 - Rede Neural Multicamadas.

Numerando as camadas em ordem crescente a partir da entrada, uma dada camada (k) transforma o vetor de entradas $X^{(k-1)}$, (que é a saída da camada (k-1) anterior), no vetor de saídas $X^{(k)}$. Assim, numa rede de m camadas, o vetor de entradas da rede $X^{(0)}$ é transformado sucessivamente através das diversas camadas, resultando no vetor de saídas da rede $X^{(m)}$. Podemos dizer, então, que a rede neural *mapeia* o vetor de entradas $X^{(0)}$ no vetor de saídas $X^{(m)}$. Para a rede neural multicamadas de neurônios Adaline sigmóide, esta relação entrada-saída, que denominaremos "Mapeamento Direto" da rede neural, é dada matematicamente por:

$$X_{I.}^{(k)} = [W^{(k)}]^T.[1 \quad (X^{(k-1)})^T]^T \tag{2.5}$$

$$X^{(k)} = \sigma(X_{I.}^{(k)}) \tag{2.6}$$

onde, considerando que a camada (k) é constituída por (n) neurônios, utilizou-se a seguinte nomenclatura: $W^{(k)}$ é a matriz de pesos da camada (k), constituída pelos vetores coluna $W_{(i)}^{(k)}$ de pesos do neurônio (i) da mesma camada,

$$W^{(k)} = [W_0^{(k)} \quad W_{(1)}^{(k)} \quad W_{(2)}^{(k)} \quad W_{(3)}^{(k)} \ldots\ldots\ldots W_{(n)}^{(k)}] \tag{2.7}$$

$W_0^{(k)}$ é o vetor de limiares de ativação da camada (k) constituído pelos limiares $w_{0(i)}^{(k)}$ (onde o subscrito (i) indica o neurônio (i) da mesma camada):

$$W_0^{(k)} = [w_{0(1)}^{(k)} \quad w_{0(2)}^{(k)} \quad w_{0(3)}^{(k)} \quad \ldots\ldots\ldots w_{0(n)}^{(k)}]^T \tag{2.8}$$

Por comodidade de notação, adotamos $\sigma(X_{I,}^{(k)})$ como sendo a função sigmóide aplicada sobre cada elemento do vetor $X_{I,}^{(k)}$:

$$\sigma(X_{I,}^{(k)}) = [\tanh(X_{I,(1)}^{(k)}) \quad \tanh(X_{I,(2)}^{(k)}) \ldots\ldots\ldots \quad \tanh(X_{I,(n)}^{(k)})]^T \tag{2.9}$$

O ajuste dos pesos através de um algoritmo de aprendizado adequado permite à rede neural aprender complexos mapeamentos não lineares. De fato, de acordo com o teorema de Kolmogorov-Nielsen [Hecht-Nielsen, 1990], decorrente do trabalho sobre representação de funções contínuas [Kolmogorov, 1957], do matemático russo Kolmogorov, uma rede neural de três camadas pode aproximar qualquer função não linear contínua de várias variáveis:

Teorema de Kolmogorov-Nielsen: dada uma função arbitrária $f : [0 \quad 1]^n \rightarrow \Re^m$, $f(x) = y$, esta função sempre pode ser implementada de maneira exata por meio de uma rede neural de três camadas com a seguinte estrutura: camada de entradas constituída por um vetor n-dimensional (correspondendo às componentes do vetor x), camada oculta com (2.n+1) neurônios, e camada de saída com m neurônios (correspondendo às componentes do vetor y.

Vários trabalhos publicados posteriormente confirmaram, através de diferentes abordagens, que as redes neurais com uma camada oculta podem ser utilizadas como aproximadores universais de funções [Irie & Miyake, 1988], [Cybenko, 1989], [Chen & Chen, 1993].

A capacidade de armazenamento de informação de uma rede neural depende do número de neurônios da mesma. De maneira geral, na rede neural o armazenamento de informação é distribuído nas conexões da mesma. Isto lhe concede um elevado grau de redundância e robustez, de modo a que a computação na rede neural não é substancialmente afetada por danos em alguns neurônios [Venkatesh, 1992].

O número de elementos de entrada e saída de uma rede neural de três camadas é determinado pelas dimensões dos vetores de entrada e saída da mesma. Assim, o número de neurônios da camada oculta é quem define, em última instância, a capacidade de armazenamento da rede. Guez [Guez, et alii, 1988], Sartori [Sartori & Antsaklis, 1991] e Huang [Huang & Huang, 1991] demostraram por métodos diversos que pelo menos N neurônios ocultos são necessários para que uma rede neural de três camadas consiga classificar efetivamente (N+1) padrões de treinamento. Este número de neurônios apenas possibilita a implementação exata do conjunto de treinamento, não garantindo nada a respeito da capacidade de generalização da rede neural. Resultados experimentais indicam que uma rede neural com duas camadas ocultas, possuindo o mesmo número de pesos ajustáveis que uma rede neural com apenas uma camada oculta, é mais susceptível a cair num mínimo local indesejado durante o aprendizado [Villiers & Barnard, 1993].

Diversos métodos foram propostos para a determinação do número de neurônios na camada oculta necessários para que uma rede neural de três camadas desempenhe adequadamente a tarefa para a qual foi projetada. Murata [Murata, et alii, 1994] introduziu o *Critério de Informação da Rede*, derivado diretamente do *Critério de Informação de Akaike* [Akaike, 1974], para determinar se a adição de novos neurônios na camada oculta é vantajosa ou não. Choi [Choi & Choi, 1994] desenvolveu um método de síntese de redes neurais para aproximação de funções com capacidade de interpolação por partes. Este método, baseado na divisão do domínio de entrada num reticulado apropriado, permite o cálculo dos pesos da rede em forma fechada, o que diminui o tempo de treinamento. Embora requerendo um número

elevado de neurônios, uma boa capacidade de generalização é garantida. Sadjadi [Sadjadi, et alii, 1993] desenvolveu um algoritmo recursivo para adaptação de pesos que opera simultaneamente com um algoritmo recursivo para criação dinâmica de novos neurônios na rede neural. O método de treinamento proposto resulta na otimização on-line do número de neurônios da rede.

2.2.3 O Algoritmo de Aprendizado

Nesta Seção apresenta-se o algoritmo de Propagação Retroativa de Erro [Rumelhart, et alii, 1986], [Widrow & Lehr, 1990], que é um método padrão, amplamente utilizado para o treinamento de redes neurais multicamadas. Neste algoritmo, a correção dos pesos de um neurônio é realizada de maneira a minimizar o erro quadrático entre a saída x_s do mesmo e a saída desejada x_s^*, que é definido como:

$$\varepsilon^2 = (x_s^* - x_s)^2 = (x_s^* - \sigma(x_L))^2 \qquad (2.10)$$

A minimização de ε^2 é alcançada por meio do método do negativo do gradiente [Rumelhart, et alii, 1986], de tal modo que a correção dos pesos do neurônio é dada por:

$$\Delta W = -\mu \frac{\partial \varepsilon^2}{\partial W} \qquad (2.11)$$

onde $0 < \mu < 1$ é o coeficiente de aprendizado, que governa a velocidade de convergência do erro. O gradiente é dado por:

$$\frac{\partial \varepsilon^2}{\partial W} = -2.\varepsilon.\sigma'(x_L).X_e = -2.\varepsilon.(1 - x_s^2).X_e \qquad (2.12)$$

Assim, o algoritmo de aprendizado para um neurônio Adaline sigmóide é:

$$W = W + \Delta W = W + 2.\varepsilon.(1-x_s^2)X_e \qquad (2.13)$$

No treinamento de uma rede neural multicamadas aparece um problema: só é conhecida a saída desejada para os neurônios da última camada. O algoritmo de propagação retroativa de erro permite contornar este empecilho. Neste algoritmo, partindo dos neurônios da última camada , cujos erros de saída são conhecidos, os erros são propagados retroativamente através das sucessivas camadas, até a camada de entrada, de maneira a possibilitar a correção de todos os pesos. Seja o vetor de saída desejado $X^{(m)*}$ de uma rede de (m) camadas, o vetor de erros da última camada é definido como:

$$\varepsilon^{(m)} = (X^{(m)*} - X^{(m)}) \qquad (2.14)$$

O objetivo do algoritmo de propagação retroativa de erro é minimizar o erro quadrático de saída da rede, que é dado por:

$$\varepsilon^2 = [\varepsilon^{(m)}]^T . [\varepsilon^{(m)}] \qquad (2.15)$$

O vetor de derivadas do erro quadrático, associado a uma dada camada (k), é definido como:

$$\delta_i^{(k)} = -\frac{1}{2}\frac{\partial \varepsilon^2}{\partial X_{I,i}^{(k)}} = \varepsilon_i^{(k)}.\sigma'(X_{Ii}^{(k)}) \qquad (2.16)$$

onde $\varepsilon^{(k)}$ é o erro de saída da camada (k) e i = 1, ... , Nº de neurônios da camada (k). O algoritmo de propagação retroativa de erro é inicializado com o vetor $\varepsilon^{(m)}$ e, por iterações sucessivas permite calcular os vetores de erros da última camada até a primeira, de maneira a possibilitar a correção dos pesos das camadas ocultas.

Algoritmo de Propagação Retroativa de Erros:

- Inicializar erro de saída: $\varepsilon^{(m)} = (X^{(m)*} - X^{(m)})$

- Faça k (número da camada) variar de m até 1

  - Vetor de derivadas da camada (k):

    - Faça i variar de 1 até Nº de neurônios da camada (k).

$$\delta_i^{(k)} = \varepsilon_i^{(k)} . \sigma\,'(X_{Li}^{(k)}) \qquad (2.17)$$

    - Fim de i

  - Propagação Retroativa de Erros:

$$\varepsilon^{(k-1)} = W^{(k)} \delta^{(k)} \qquad (2.18)$$

  - Incremento na Matriz de Pesos da camada (k):

$$\Delta W^{(k)} = 2.\mu\,[(X^{(k-1)}).\,(\delta^{(k)})^T] \qquad (2.19)$$

  - Correção da Matriz de Pesos:

$$W^{(k)} = W^{(k)} + \Delta W^{(k)} \qquad (2.20)$$

- Fim de k

Este algoritmo deve ser realizado para todo o conjunto de treinamento, que é constituído de todos os pares de entradas e saídas desejadas, de maneira a que o erro de aprendizado convirja para zero. No algoritmo acima, a correção dos pesos é realizada sucessivamente, logo após a apresentação de cada padrão de treinamento, o que o caracteriza como do tipo "on-line". Quando os pesos da rede são corrigidos somente após a apresentação de todo o conjunto de padrões de treinamento, o algoritmo é caracterizado como do tipo "batch". Neste ultimo caso, porém, uma capacidade adicional de armazenamento de dados é requerida em cada conexão da rede. Os valores iniciais para os pesos da rede normalmente são pequenos e randômicos. Um método para inicializar os pesos com valores mais próximos do seu valor final, conseqüentemente acelerando o aprendizado, é apresentado em [Lee, 1993]. O coeficiente de aprendizado, que controla a velocidade de aprendizado da rede neural, deve ser escolhido cuidadosamente entre 0 e 1. Valores muito elevados podem levar a rede a convergir para valores indesejados, ou mesmo divergir.

Para melhorar o desempenho no aprendizado inumeráveis alternativas foram desenvolvidas. Por exemplo, em [Kwon, et alii, 1992] é apresentado um método modificado para o cálculo das derivadas que permite evitar a paralisia da rede e acelerar o aprendizado. Em [Wang, 1994] é apresentado um método que acelera o aprendizado, além de permitir aprender não somente um mapeamento não linear, mas também as suas derivadas. Um algoritmo que utiliza o gradiente junto com o conjugado do gradiente para acelerar o aprendizado é apresentado em [Guozhong & Yaming, 1991]. O uso de um coeficiente de aprendizado variável associado à um termo de momento permite acelerar bastante o aprendizado, como é mostrado em [Tanomaru & Omatu, 1991]. A adaptação de, não apenas os pesos e limiares, mas também da amplitude e declividade da função de ativação em neurônios Adaline sigmóide [Rangwala & Dornfeld, 1989], permite otimizar ainda mais o processo de aprendizado das redes neurais.

2.3 Neurocontroladores

2.3.1 Redes Neurais em Sistemas de Controle

Considerando que as redes neurais são sistemas distribuídos, altamente não lineares e robustos, dotados da capacidade de aprender complicados mapeamentos não lineares, a sua aplicação em controle de sistemas dinâmicos é uma conseqüência natural das suas propriedades, adequadas para a solução de problemas tais como identificação, controle, adaptação e sintonia de controladores. Uma boa revisão sobre aplicações de redes neurais em sistemas de controle pode ser encontrada em artigos como [Fukuda & Shibata, 1992] e [Werbos, 1993].

Uma aplicação imediata de redes neurais em sistemas de controle é na Identificação de Sistemas Dinâmicos [Chen, et alii, 1990], [Willis, et alii, 1991], [Babu & Eswaran, 1992]. A idéia é simples (ver Figura 2.4): dado um sistema dinâmico com vetor de entradas U e vetor de saídas Y, utilizar uma rede neural para aprender o mapeamento entrada-saída do sistema (U → Y), de maneira a emular o comportamento dinâmico do mesmo. Neste caso, a rede neural é apropriadamente chamada de rede neural "Emuladora".

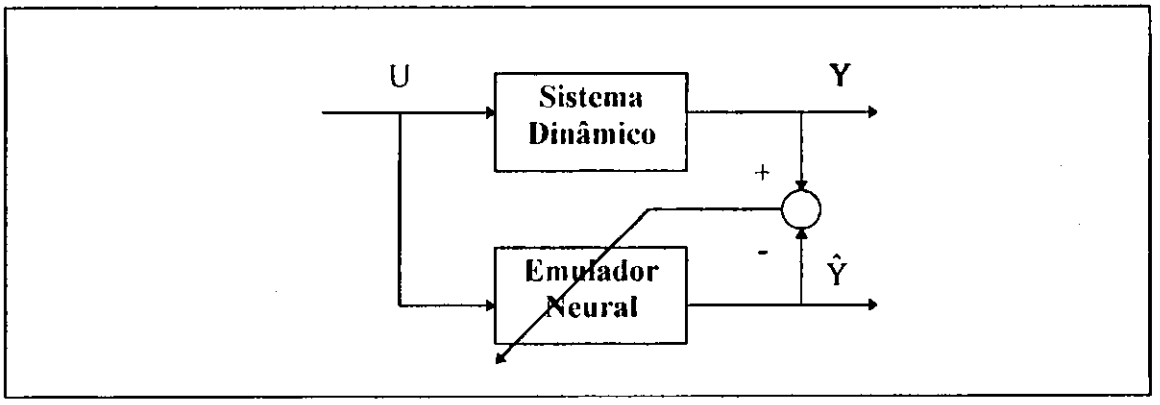

Fig. 2.4 - Identificação de Sistema usando Rede Neural.

A entrada do sistema U é aplicada à entrada da rede neural, que produz um valor estimado $\hat{Y}$, que é uma aproximação da saída Y do sistema. O erro entre a saída do sistema e a saída da rede neural, $Y - \hat{Y}$, é utilizado para treinar a rede por meio do algoritmo de propagação retroativa de erro. Quanto melhor a rede aprender o mapeamento $U \rightarrow Y$, menor será este erro. O aprendizado é realizado até o erro atingir valores aceitáveis. Para obter uma excitação persistente, de maneira a identificar adequadamente o sistema, é comum adicionar um sinal de ruído de baixa amplitude e faixa ampla à entrada do sistema.

Uma segunda área de aplicação de redes neurais em controle é em Sistemas Híbridos [Tanomaru & Omatu, 1991], [Khan, 1991], [Yamada & Yabuta, 1993], [Mukhopadhyay & Narendra, 1993], [Qing, 1992]. Nestes sistemas, a rede neural é associada a controladores convencionais, de maneira a compensar não linearidades ou perturbações. A idéia é aproveitar as vantagens dos controladores lineares convencionais aliadas às capacidades de aprendizado e mapeamento não linear das redes neurais. A Figura 2.5 mostra um exemplo de sistema híbrido.

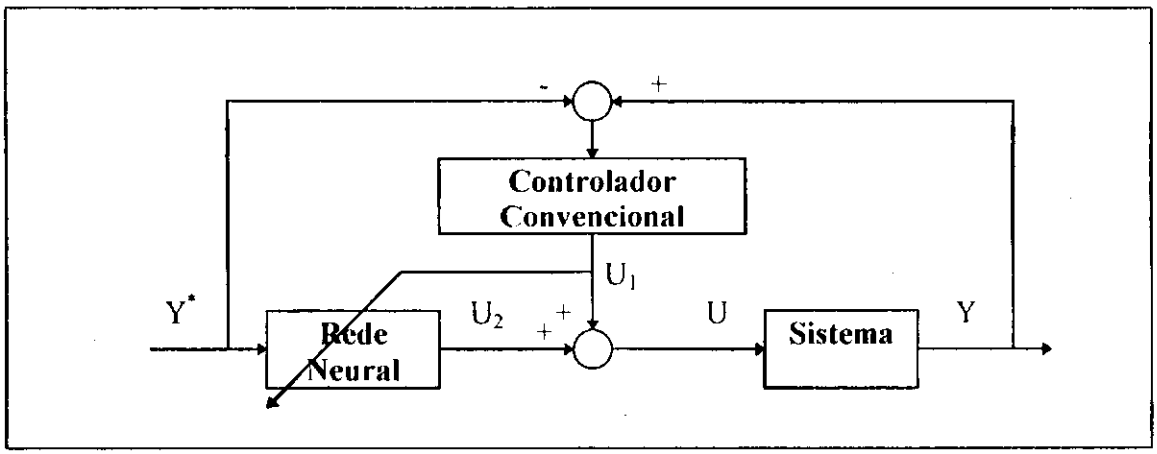

Fig. 2.5 - Sistema Híbrido.

A Sintonia Automática de Controladores Convencionais é outra área de aplicação das redes neurais [Swiniarski, 1990-A], [Yamada & Yabuta, 1991]. Neste caso, os ganhos de um controlador convencional são obtidos por meio de uma rede

neural que é treinada de maneira a minimizar o erro de rastreamento de trajetória do sistema. A Figura 2.6 mostra um exemplo de sintonia automática de controlador usando rede neural.

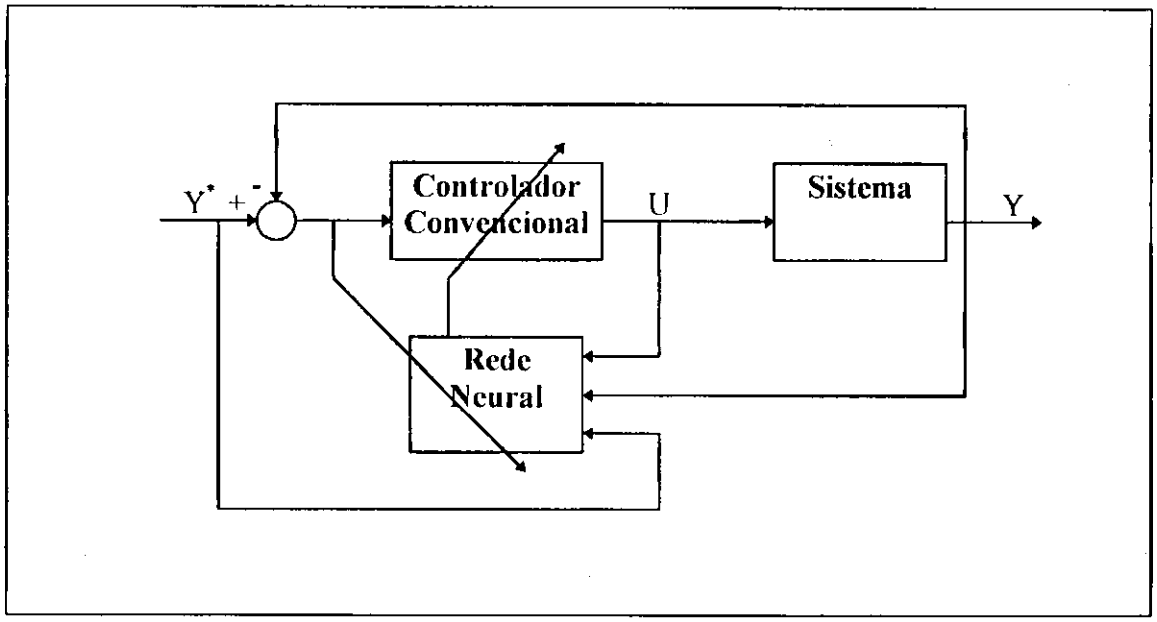

Fig. 2.6 - Sintonia Automática de Controlador usando Rede Neural.

Outra aplicação de redes neurais em sistemas de controle é desempenhando a função do próprio controlador. A idéia é aproveitar a capacidade de aprendizado das mesmas para implementar controladores não lineares adaptativos. Assim, a rede neural é treinada para, a partir da saída do sistema desejada Y^*, produzir a entrada U capaz de impor a saída $Y = Y^*$. Neste caso, a rede neural recebe o nome de Neurocontrolador, assunto abordado a seguir.

2.3.2 Neurocontrolador Inverso

Quando uma rede neural desempenha a função de controlador, é chamada apropriadamente de Neurocontrolador. O neurocontrolador se encarrega de produzir a entrada do sistema U necessária para impor a saída desejada Y^*, conforme é mostrado no diagrama simplificado da Figura 2.7.

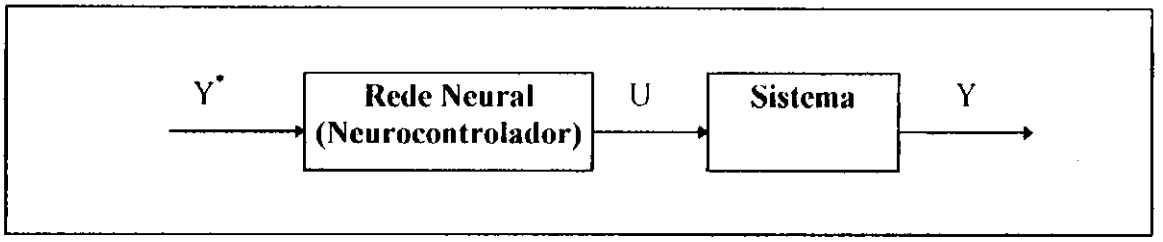

Fig. 2.7 - Neurocontrolador.

Pode-se treinar o neurocontrolador de maneira a aprender qualquer relação $Y^* \rightarrow U$, desde as ações de um operador humano (técnica aplicada com sucesso em vários casos práticos [Werbos, 1993]), até a emulação de complexos reguladores ótimos [Zhang, et alii, 1992]. Em particular, o neurocontrolador pode ser treinado de maneira a aprender a dinâmica inversa do sistema (desde que a dinâmica da planta seja inversível). Neste caso, sejam as dinâmicas direta e inversa dadas respectivamente pelos mapeamentos:

$$Y = f(U) \tag{2.21}$$

$$U = f^{-1}(Y) \tag{2.22}$$

então, o neurocontrolador é treinado para aprender o mapeamento inverso (2.22):

$$U = \hat{f}^{-1}(Y^*) \tag{2.23}$$

onde, desde que o aprendizado seja realizado com sucesso, teremos $\hat{f}^{-1}(.) \cong f^{-1}(.)$, de maneira que o sistema em conjunto com o controlador apresentam função de transferência aproximadamente unitária e a saída do sistema Y acompanha o valor desejado Y^*:

$$Y = f(U) = f(\hat{f}^{-1}(Y^*)) \cong Y^* \tag{2.24}$$

Obviamente que, para funcionar efetivamente, o neurocontrolador deve aprender o mapeamento (2.23) de maneira a aproximar acuradamente a Equação (2.22). Devido à estrutura distribuída e altamente não linear de uma rede neural, a análise de estabilidade dos neurocontroladores torna-se complicada, sendo um problema ainda em aberto, embora diversas provas foram obtidas para casos particulares [Chen & Liu, 1994], [Sadegh, 1993], [Demin, et alii, 1992]. Para melhorar o desempenho dos neurocontroladores, diversos autores propuseram a sua associação com outras técnicas de controle, tais como: controle adaptativo, sistemas de estrutura variável, lógica nebulosa, etc. [Swiniarski, 1990B], [Strefezza & Dote, 1991], [Kim, et alii, 1993].

2.3.3 Configurações de Treinamento

2.3.3.1 Treinamento do Neurocontrolador

Para que o neurocontrolador desempenhe adequadamente a sua função, deve ser submetido a um treinamento de maneira a aprender o mapeamento de dinâmica inversa com uma precisão satisfatória. O treinamento é realizado de maneira a minimizar uma dada função de erro $f_e(.)$, a qual pode ser função do erro de saída do neurocontrolador ou do erro de saída do sistema. Para poder utilizar métodos baseados em gradiente na otimização de sistemas dinâmicos que incluem neurocontroladores, (tal como o método de propagação retroativa de erro), é necessário computar a derivada da função de erro em relação à saída do neurocontrolador [Narendra & Parthasarathy, 1991]:

$$\delta = -\frac{\partial f_e}{\partial U} \tag{2.25}$$

A derivada δ contém informação suficiente para realizar o treinamento usando o método de propagação retroativa de erro.

O treinamento pode ser realizado off-line ou on-line. De acordo com a entrada U aplicada ao sistema, o treinamento off-line pode ser realizado em Malha Aberta ou em Malha Fechada [Low, et alii, 1993]. No treinamento em malha aberta, a entrada U é variada randomicamente, dentro dos seus limites de operação, de maneira a excitar persistentemente o sistema. Como o objetivo final do neurocontrolador é controlar a saída Y, o treinamento em malha aberta pode vir a não cobrir a faixa de variação projetada para Y, resultando numa capacidade de generalização pobre. Para contornar este problema, pode-se utilizar o treinamento em malha fechada. Neste caso, a entrada aplicada é obtida por meio de um controlador convencional simples (Proporcional, Proporcional-Integral, etc.), variando a referência do mesmo, Y^*, dentro da faixa de operação desejada. Um sinal randômico de baixa amplitude é adicionado à entrada do sistema de maneira a prover uma excitação persistente. De acordo com a trajetória de teste Y^* utilizada no treinamento, o neurocontrolador é classificado em "Por Aprendizado" ou "Adaptativo" [Yamada & Yabuta, 1990], Quando a trajetória de testes é repetida periodicamente, o treinamento é realizado com os dados coletados a cada período e o neurocontrolador é denominado "Por Aprendizado". Quando a trajetória de teste não se repete periodicamente e o treinamento é realizado on-line, a cada intervalo de amostragem, o neurocontrolador é chamado de Adaptativo.

De acordo com a maneira como é obtida a derivada δ da função de erro, três configurações de treinamento foram propostas inicialmente por Narendra [Narendra & Parthasarathy, 1990]: Controle Inverso Direto, Controle Adaptativo Direto e Controle Adaptativo Indireto.

2.3.3.2 Controle Inverso Direto

Nesta configuração, mostrada na Figura 2.8, o neurocontrolador é treinado para aprender a dinâmica inversa da planta, usando como padrões de treinamento os pares entrada-saída (Y, U), obtidos diretamente através da medição da saída e da entrada do sistema.

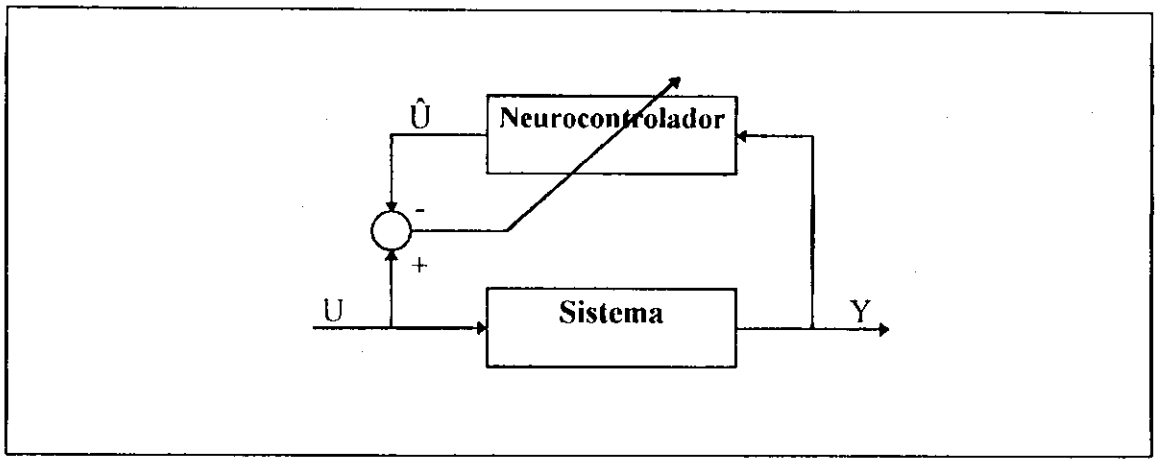

Fig. 2.8 - Configuração de Controle Inverso Direto.

O treinamento pode ser realizado off-line ou on-line através do método de propagação retroativa de erro. Nesta configuração, a função de erro a ser minimizada pelo processo de aprendizado é simplesmente:

$$f_e = \frac{(\Delta U^T . \Delta U)}{2} \tag{2.26}$$

onde ΔU é o erro de saída do neurocontrolador:

$$\Delta U = U - \hat{U} \tag{2.27}$$

então, a derivada da função de erro a ser utilizada no algoritmo de propagação retroativa é dada simplesmente por:

$$\delta = -\frac{\partial f_e}{\partial \hat{U}} = \Delta U \tag{2.28}$$

Esta configuração possui a vantagem de os dados para o treinamento serem disponíveis diretamente através da medição da entrada e saída do sistema, possibilitando aprendizado supervisionado para o neurocontrolador. Por outro lado, por ser dependente da capacidade de generalização do neurocontrolador, esta

configuração apresenta melhor desempenho se utilizada em conjunto com algum outro método. Além disso, esta configuração requer algum treinamento off-line [Tanomaru & Omatu, 1992].

2.3.3.3 Controle Adaptativo Direto

Nesta configuração, mostrada na Figura 2.9, a função de erro é definida sobre o erro de saída do sistema, permitindo assim otimizar o desempenho do controlador diretamente em relação ao seu objetivo, que é o controle da saída Y. O aprendizado é realizado essencialmente on-line e a função de erro é definida como:

$$f_e = \frac{(\Delta Y^T . \Delta Y)}{2} \tag{2.29}$$

onde o erro de trajetória de saída do sistema ΔY é dado por:

$$\Delta Y = Y^* - Y \tag{2.30}$$

A derivada da função de erro, necessária para computar o algoritmo de propagação retroativa, é então:

$$\delta = -\frac{\partial f_e}{\partial U} = \frac{\partial Y}{\partial U}\Delta Y \tag{2.31}$$

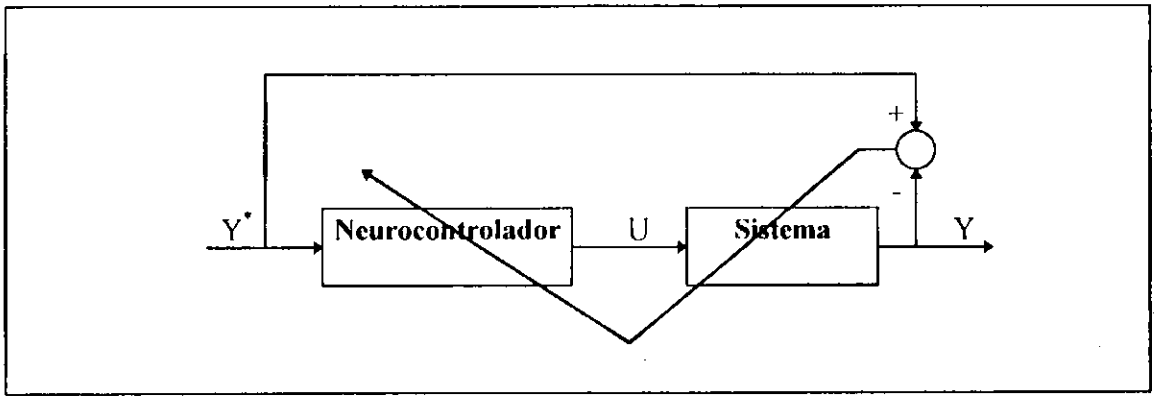

Fig. 2.9 - Configuração de Controle Adaptativo Direto.

Nota-se que para o cálculo da derivada da função de erro δ é necessário conhecer o Jacobiano da planta $\partial Y / \partial U$, uma grande desvantagem desta configuração. Isto é diretamente contrário à filosofia do neurocontrole, que procura a solução do problema de controle com o menor conhecimento possível do sistema. Esta desvantagem pode ser contornada em parte usando cálculo aproximado do Jacobiano. Em alguns casos, apenas o sinal do Jacobiano é suficiente para calcular a derivada δ [Tanomaru & Omatu, 1991]. Por outro lado, o uso de parâmetros de aprendizado variáveis permite melhorar consideravelmente o desempenho do neurocontrolador adaptativo direto [Cavalcanti, 1995].

2.3.3.4 Controle Adaptativo Indireto

Como foi discutido na Seção anterior, a configuração de controle adaptativo direto necessita de um conhecimento prévio do Jacobiano do sistema, além de assumir que a dinâmica da planta é inversível. A configuração de Controle Adaptativo Indireto, mostrada na Figura 2.10, não precisa do Jacobiano da planta.

Nesta configuração, uma rede neural adicional, o Emulador, treinada para emular a dinâmica direta da planta, é utilizada para, por meio de propagação retroativa, transformar o erro de trajetória ΔY no erro de saída do neurocontrolador ΔU, o qual contem informação suficiente para treinar o neurocontrolador usando o algoritmo de aprendizado. O emulador deve ser treinado off-line com um conjunto de dados suficientemente rico de maneira a possibilitar a identificação da dinâmica direta. Posteriormente, o emulador é utilizado para calcular as derivadas do erro de trajetória em relação à saída do neurocontrolador, permitindo assim, o treinamento on-line do mesmo. O emulador também pode ser treinado on-line, em conjunto com o neurocontrolador. Neste caso, por razões de robustez, é razoável corrigir os pesos da rede neural emuladora numa taxa mais rápida do que os pesos do neurocontrolador. Embora possua uma complexidade maior (duas redes neurais são necessárias), esta configuração pode ser utilizada na solução de complexos problemas de controle não linear [Nguyen & Widrow, 1991].

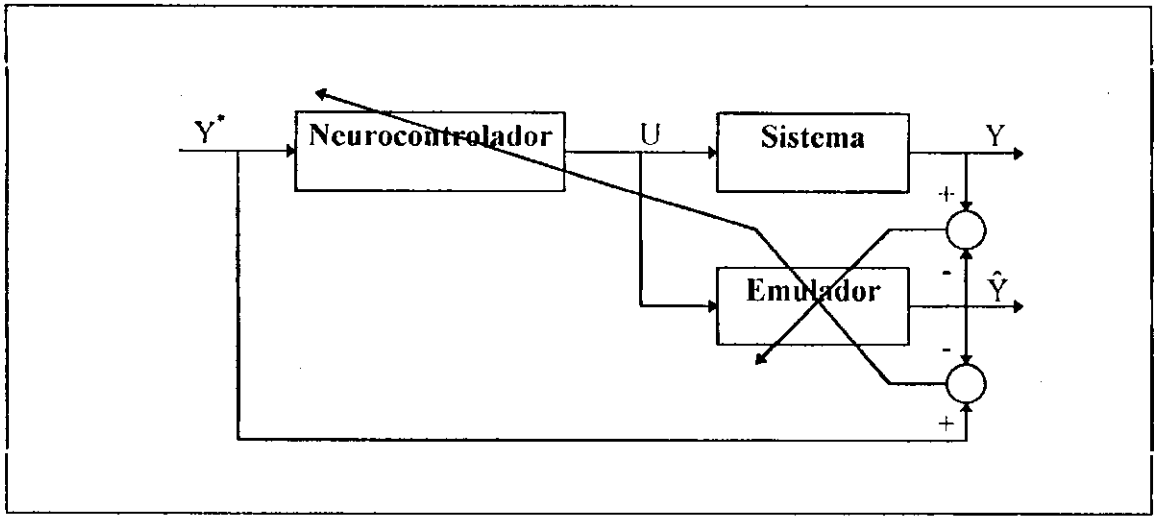

Fig. 2.10 - Configuração de Controle Adaptativo Indireto.

2.4 Conclusão

Neste capítulo, foram apresentados os principais conceitos associados às redes neurais multicamadas e às suas unidades básicas, os neurônios artificiais. O algoritmo de aprendizado padrão de Propagação Retroativa de Erros foi sumarizado. A capacidade das redes neurais multicamadas de aprender complexos mapeamentos não lineares foi discutida. O uso de redes neurais em sistemas de controle foi revisado. O conceito de Neurocontrolador foi apresentado. Em particular, estudou-se o neurocontrolador Inverso, dando-se ênfase às configurações de treinamento: Controle Inverso Direto, Controle Adaptativo Direto e Controle Adaptativo Indireto. As vantagens e desvantagens destas configurações foram discutidas. Visando o aprendizado e domínio das técnicas de neurocontrole, antes de estender o estudo ao campo de controle de manipuladores robóticos, investigaram-se várias técnicas para neurocontrole da corrente estatórica de motores de indução, (que caracteriza-se por ser um sistema não linear de múltiplas entradas e saídas relativamente mais simples do que um manipulador robótico). Neste sentido, vários trabalhos sobre controle de motores de indução foram publicados no decorrer da elaboração desta Tese: [Alsina & Gehlot, 1992-A], [Alsina & Gehlot, 1993-C], [Alsina & Gehlot, 1993-D], [Gehlot & Alsina, 1993], [Alsina & Gehlot, 1995-B], [Alsina & Gehlot, 1995-E].

CAPÍTULO 3

DINÂMICA DE MANIPULADORES

3.1 Introdução

Neste capítulo, descrevem-se métodos para obtenção da equação dinâmica de manipuladores robóticos. Na Seção 3.2 é feita uma revisão sumária sobre cinemática de manipuladores, abordando os problemas de cinemática direta e inversa. Na Seção 3.3 as equações de dinâmica inversa e direta de manipuladores robóticos são sumariamente apresentadas. Na Seção 3.4 descreve-se o Método Newton-Euler para cálculo da dinâmica inversa e uma forma equivalente em vetores 6x1 é desenvolvida. Os problemas de controle e simulação de robôs são abordados na Seção 3.5. Na Seção 3.6, a partir de uma representação por meio de diagrama de blocos, desenvolvem-se dois métodos recursivos para cálculo da matriz de inércia e da equação dinâmica de manipuladores. Uma forma fatorada da matriz de inércia é apresentada na Seção 3.7.

3.2 Cinemática de Manipuladores

3.2.1 Parâmetros Cinemáticos

Um manipulador robótico é constituído por uma série de corpos rígidos (elos), interligados por juntas prismáticas ou rotacionais, as quais são acionadas por atuadores (elétricos, pneumáticos, etc). A análise cinemática procura descrever as relações espaciais entre as diferentes partes do manipulador sem levar em conta os esforços aplicados pelos atuadores.

Quando uma ou mais juntas são acionadas, as posições relativas entre os diversos elos de um manipulador se modificam. Denavit e Hartenberg [Denavit & Hartenberg, 1955] desenvolveram um método genérico para a análise cinemática de

manipuladores, de maneira a descrever sistematicamente as relações espaciais entre as diferentes partes do mesmo. De acordo com este método, cada elo do manipulador é numerado seqüencialmente, partindo da base (cujo número é zero) em direção à ferramenta. A cada elo i (i = 1, ... ,N) do manipulador é associado um sistema referencial (x_i, y_i, z_i), fixo ao mesmo. O referencial de um elo i é atribuído da seguinte maneira: o eixo z_i é alinhado com o eixo da junta i; o eixo x_i é estabelecido sobre a reta normal aos eixos z_i e z_{i+1}, e apontando da junta i para a junta i+1; o eixo y_i é obtido pela "regra da mão direita". Estabelecidos os referenciais de elos, podem ser definidas quatro grandezas que caracterizam completamente cada elo do ponto de vista cinemático:

- Comprimento de elo, a_i = Distância entre z_i e z_{i+1} medida ao longo de x_i.

- Ângulo de torção de elo, α_i = Ângulo entre z_i e z_{i+1} medido em torno de x_i.

- Comprimento de junta, d_i = Distância entre x_{i-1} e x_i medida ao longo de z_i.

- Ângulo de Junta, θ_i = Ângulo entre x_{i-1} e x_i medido em torno de z_i.

Para uma junta i rotacional, o ângulo θ_i é modificado por meio do atuador, sendo então denominado de *variável da junta i*. Para uma junta i prismática, a variável de junta é o comprimento d_i. A variável de junta mais as três grandezas restantes (denominadas de parâmetros cinemáticos de elo) permitem descrever cinematicamente um determinado elo, constituindo a chamada "notação Denavit-Hartenberg". Os parâmetros Denavit-Hartenberg são mostrados esquematicamente na Figura 3.1.

Os sistemas de referência na base e na ferramenta do manipulador são atribuídos de maneira a que os eixos z_0 e z_{N+1} coincidam com os eixos z_1 e z_N respectivamente. Os eixos restantes são atribuídos de maneira a anular a maior quantidade possível de parâmetros cinemáticos.

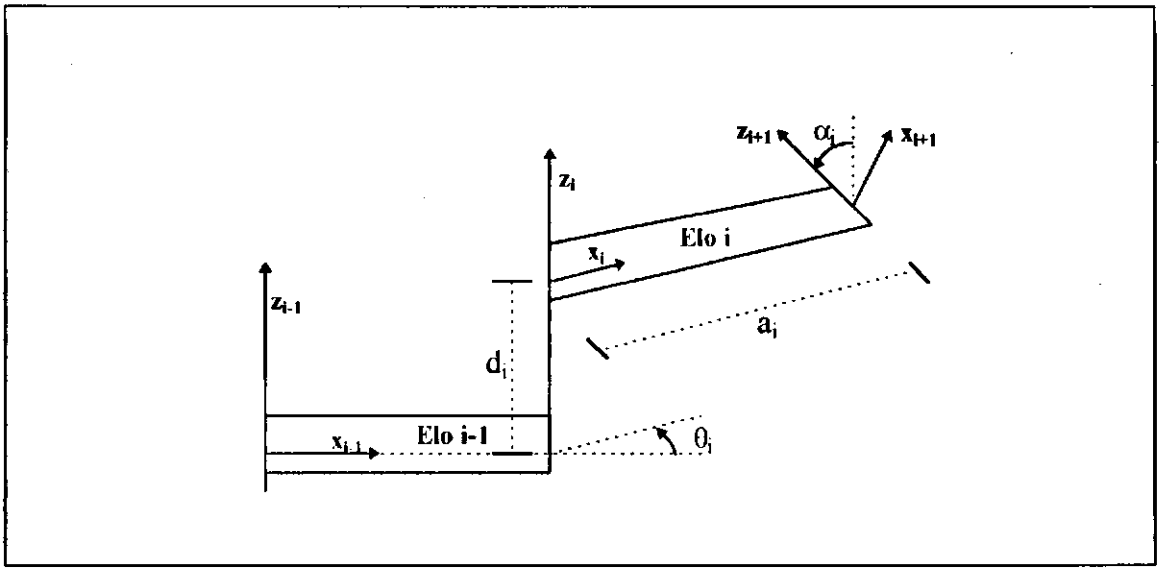

Fig. 3.1 - Parâmetros Denavit-Hartenberg.

3.2.2 Cinemática Direta

De acordo com a definição dos parâmetros Denavit-Harenberg, pode-se expressar a posição e orientação do referencial do elo i em relação ao referencial do elo i-1 através da seguinte seqüência de translações e rotações:

- Translação de uma distância a_{i-1} ao longo de x_{i-1}.

- Rotação de um ângulo α_{i-1} em torno de x_{i-1}.

- Translação de uma distância d_i ao longo de z_i.

- Rotação de um ângulo θ_i em torno de z_i.

Matematicamente, esta seqüência de translações e rotações pode ser computada utilizando Transformações Homogêneas ([Craig, 1986]):

$$ {}^{i-1}T_i = \begin{bmatrix} 1 & 0 & 0 & a_{i-1} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha_{i-1} & -\text{sen}\alpha_{i-1} & 0 \\ 0 & \text{sen}\alpha_{i-1} & \cos\alpha_{i-1} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & d_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta_i & -\text{sen}\theta_i & 0 \\ 0 & \text{sen}\theta_i & \cos\theta_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.1) $$

ou seja,

$$ {}^{i-1}T_i = \left[\begin{array}{c|c} A_i^{i-1} & {}^{i-1}p_i \\ \hline 0\,0\,0 & 1 \end{array}\right] = \begin{bmatrix} \cos\theta_i & -\text{sen}\theta_i & 0 & a_{i-1} \\ \cos\alpha_{i-1}\,\text{sen}\theta_i & \cos\alpha_{i-1}\cos\theta_i & -\text{sen}\alpha_{i-1} & -\text{sen}\alpha_{i-1}d_i \\ \text{sen}\alpha_{i-1}\,\text{sen}\theta_i & \text{sen}\alpha_{i-1}\cos\theta_i & \cos\alpha_{i-1} & \cos\alpha_{i-1}d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.2) $$

onde, ${}^{i-1}T_i$ é a matriz 4x4 de transformação homogênea do elo i em relação ao elo i-1. Como mostra a Equação (3.2), A matriz ${}^{i-1}T_i$ pode ser particionada na matriz 3x3 A_i^{i-1} de rotação ortogonal (que representa a orientação do referencial de elo i em relação ao referencial de elo i-1) e no vetor 3x1 ${}^{i-1}p_i$ (que representa a posição do referencial de elo i em relação ao referencial de elo i-1). A linha inferior, com três zeros e um 1 é adicionada para possibilitar inversão e produto de transformações homogêneas.

O problema de Cinemática Direta pode ser estabelecido da seguinte maneira: dado o vetor Nx1 de variáveis de junta $q = [q_1 \;\; q_2 \;\; \ldots\ldots \;\; q_N]^T$ (onde q_i pode ser θ_i ou d_i, de acordo com o tipo da junta i) determinar a orientação A_N^0 e a posição ${}^0p_N$ da ferramenta em relação ao referencial da base do manipulador. Este problema é facilmente solucionado utilizando as transformações homogêneas entre os elos:

$$ {}^0T_N = \left[\begin{array}{c|c} A_N^0 & {}^0p_N \\ \hline 0\,0\,0 & 1 \end{array}\right] = {}^0T_1 . {}^1T_2 . {}^2T_3 \ldots\ldots {}^{N-2}T_{N-1} . {}^{N-1}T_N \quad (3.3) $$

onde ${}^0T_N$ é a transformação homogênea que representa a orientação e posição do referencial N (ferramenta) em relação ao referencial 0 (base do manipulador).

3.2.3 Cinemática Inversa

O problema de Cinemática Inversa, como o próprio nome indica, é inverso ao problema de cinemática direta, podendo ser estabelecido da seguinte maneira: dadas a orientação e posição da ferramenta ${}^{0}T_{N}$, determinar o vetor de variáveis de junta, q, correspondente. Este problema é, basicamente, um problema de controle, ou seja, determinar os ângulos em que devem ser girados os motores de maneira a posicionar a ferramenta num local desejado.

Ao contrário do problema de cinemática direta, a solução do problema de cinemática inversa não é trivial. A matriz ${}^{0}T_{N}$ possui doze elementos que são função das variáveis de junta, resultando em doze equações (das quais apenas seis são independentes). Estas equações são não lineares e transcendentes. A solução em formula fechada para o problema de cinemática inversa é obtida somente para alguns casos particulares de manipuladores com estrutura simples, poucos graus de liberdade e possuindo geralmente interseção de dois ou mais eixos. Grande parte dos manipuladores industriais é construído de maneira a possuir solução em formula fechada, de maneira a diminuir a carga computacional dos controladores. Nestes casos particulares, a solução pode ser obtida por meios algébricos ou geométricos.

Outra possível maneira de resolver o problema de cinemática inversa é através de métodos numéricos. Neste caso, a solução numérica é obtida através de algumas aproximações sucessivas, por meio de um algoritmo adequado. Embora mais genéricos, os métodos numéricos possuem a desvantagem de elevada carga computacional, o que dificulta a sua aplicação em tempo real. Dentro desta linha, e em paralelo com o trabalho desta Tese, foi desenvolvido um método para obtenção da cinemática inversa de manipuladores robóticos baseado em redes neurais modulares [Alsina & Gehlot, 1994-E], [Alsina & Gehlot, 1995-A].

Qualquer que seja o método empregado no cálculo da cinemática inversa, deve levar em conta os problemas de existência e multiplicidade de soluções. Existem posições e orientações da ferramenta que não podem ser atingidas pelo manipulador, seja por estarem muito distantes ou por limitações da estrutura do mesmo. O algoritmo de cinemática inversa deve ser capaz de lidar com estas situações. Para manipuladores de até seis graus de liberdade, uma dada localização da ferramenta pode ser atingida por várias configurações de variáveis de junta, de maneira que o algoritmo de cinemática inversa deve incluir algum critério para a seleção de uma dentre as soluções possíveis. Para manipuladores com mais de seis graus de liberdade (chamados de manipuladores redundantes), o problema de cinemática inversa admite infinitas soluções, de maneira que algum critério de otimização deve ser adotado para que o algoritmo de cinemática inversa funcione adequadamente.

3.3 A Equação Dinâmica

Devido ao elevado grau de acoplamento entre as suas partes, os manipuladores robóticos são sistemas altamente não lineares, cuja dinâmica é descrita por uma equação diferencial vetorial não linear de segunda ordem. A equação dinâmica de um manipulador envolve termos inerciais (proporcionais às acelerações das juntas), termos coriolis (produtos de velocidades), centrífugos (velocidades ao quadrado), termos gravitacionais e termos adicionais de dinâmica não modelada (tais como atrito ou perturbações). A dinâmica inversa de um manipulador genérico de N juntas, relacionando a posição das juntas aos esforços de acionamento aplicados pelos atuadores, é dada por:

$$\tau = M(q).\ddot{q} + \tau_c(q,\dot{q}) + \tau_g(q) + \tau_a \qquad (3.4)$$

onde,

τ = Vetor Nx1 de esforços de acionamento (forças ou conjugados).

q = Vetor Nx1 de variáveis de juntas (ângulos ou deslocamentos).

$M(q)$ = Matriz de Inércia NxN.

$\tau_c(q,\dot{q})$ = Vetor Nx1 de reações coriólis e centrífugas.

$\tau_g(q)$ = Vetor Nx1 de reações gravitacionais.

τ_a = Vetor Nx1 de dinâmica não modelada (atrito, perturbações, etc.).

A equação de dinâmica direta, relacionando os esforços de acionamento às acelerações das juntas, é obtida invertendo a Equação (3.4):

$$\ddot{q} = M(q)^{-1}(\tau - \tau_c(q,\dot{q}) - \tau_g(q) - \tau_a) \tag{3.5}$$

Definindo o vetor de reações não inerciais (coriolis, centrífugas e dinâmica não modelada) como:

$$\tau_0 = \tau_0(q,\dot{q}) = \tau_c(q,\dot{q}) + \tau_g(q) + \tau_a \tag{3.6}$$

então, as equações de dinâmica inversa (3.4) e dinâmica direta (3.5) podem ser reescritas de maneira mais compacta:

$$\tau = M(q).\ddot{q} + \tau_0 \tag{3.7}$$

$$\ddot{q} = M(q)^{-1}(\tau - \tau_0) \tag{3.8}$$

Através de integração numérica da Equação (3.8) de dinâmica direta, as velocidades e posições de junta podem ser obtidas. Este procedimento é utilizado na simulação dinâmica de manipuladores robóticos. Na Seção a seguir é apresentado o método recursivo Newton-Euler para computação da Equação de dinâmica inversa.

3.4 O Método Newton-Euler

3.4.1 Forma em Vetores 3x1.

Para um manipulador robótico em movimento, um elo i do mesmo, é dotado de uma velocidade angular e de uma velocidade linear, as quais são representadas respectivamente pelos vetores ω_i e v_i 3x1, expressos no referencial do próprio elo, conforme mostra a Figura 3.2. Assim, a velocidade linear de um elo i+1, dotado de uma junta rotacional, é igual à velocidade linear do elo i mais uma nova componente causada pela velocidade angular do elo i. Pelo mesmo raciocínio, a velocidade angular do elo i+1 é igual à velocidade angular do elo i mais uma nova componente causada pela rotação da junta i+1:

$$v_{i+1} = A_{i+1}^{iT}(\omega_i \times {}^{i}p_{i+1} + v_i) \tag{3.9}$$

$$\omega_{i+1} = A_{i+1}^{iT}\omega_i + \begin{bmatrix} 0 & 0 & \dot{\theta}_{i+1} \end{bmatrix}^T \tag{3.10}$$

Nas Equações acima, a multiplicação pela matriz de rotação A_{i+1}^{iT} foi utilizada para expressar os vetores ω_i e v_i no referencial de elo i+1, (no qual estão expressos os vetores ω_{i+1} e v_{i+1}). Desta maneira, com todos os vetores expressos no mesmo referencial, a soma vetorial torna-se possível. Por um raciocínio análogo, para um elo i+1 dotado de uma junta prismática, os vetores de velocidade linear e angular são dados por:

$$v_{i+1} = A_{i+1}^{iT}(\omega_i \times {}^{i}p_{i+1} + v_i) + \begin{bmatrix} 0 & 0 & \dot{d}_{i+1} \end{bmatrix}^T \tag{3.11}$$

$$\omega_{i+1} = A_{i+1}^{iT}\omega_i \tag{3.12}$$

As Equações acima permitem calcular recursivamente, a partir das velocidades de junta, os vetores de velocidade linear e angular dos referenciais de elo de qualquer manipulador robótico serial.

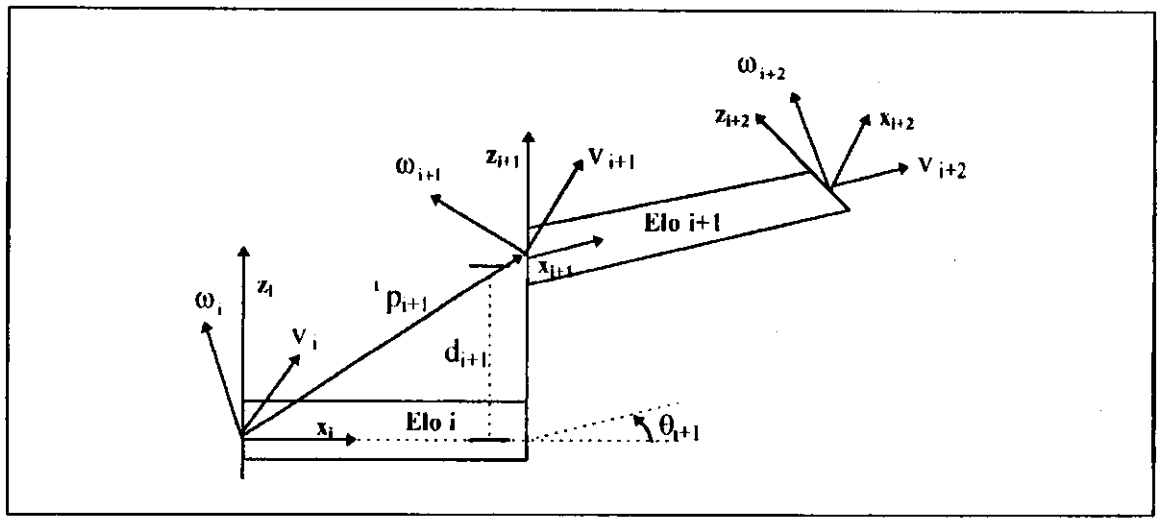

Fig. 3.2 - Propagação de Velocidades através dos Elos.

Um corpo rigido, cujo centro de massa é submetido a uma aceleração linear $\dot{v}_c$ e a uma aceleração angular $\dot{\omega}_c$, sofre a ação de uma força resultante f_R e de um momento resultante n_R, atuando no centro de massa, que são dados, respectivamente, pelas equações de movimento de Newton e de Euler:

$$f_R = m_c . \dot{v}_c \tag{3.13}$$

$$n_R = h_c . \dot{\omega}_c + \omega_c \times h_c . \omega_c \tag{3.14}$$

onde m_c é a massa do corpo rigido e h_c é o tensor de inércia do mesmo, expresso em referencial de centro de massa. Num manipulador robótico, a força resultante f_{Ri} no centro de massa p_{ci} de um elo i é resultado do balanço de forças atuando sobre o mesmo: força f_i, exercida sobre o elo i pelo elo i-1, e força de reação $-f_{i+1}$ devida à ação do elo i no elo i+1. De maneira análoga, o momento resultante n_{Ri} num elo i é resultado do balanço de momentos atuando no mesmo: conjugado n_i, exercido sobre o elo i pelo elo i-1, pelo conjugado de reação $-n_{i+1}$ devido à ação do elo i no elo i+1 e pelos momentos produzidos pelas forças f_i e $-f_{i+1}$ atuando sobre o elo. A Figura 3.3 mostra o balanço de forças e momentos num elo i de um manipulador robótico, que pode ser expresso como:

$$f_i = A^i_{i+1}.f_{i+1} + f_{Ri} \tag{3.15}$$

$$n_i = n_{Ri} + A^i_{i+1}.n_{i+1} + p_{c_i} \times f_{Ri} + {}^i p_{i+1} \times A^i_{i+1}.f_{i+1} \tag{3.16}$$

Nas Equações acima, a multiplicação pela matriz de rotação A^{iT}_{i+1} foi utilizada para possibilitar a soma de vetores, expressando os mesmos no referencial de elo i.

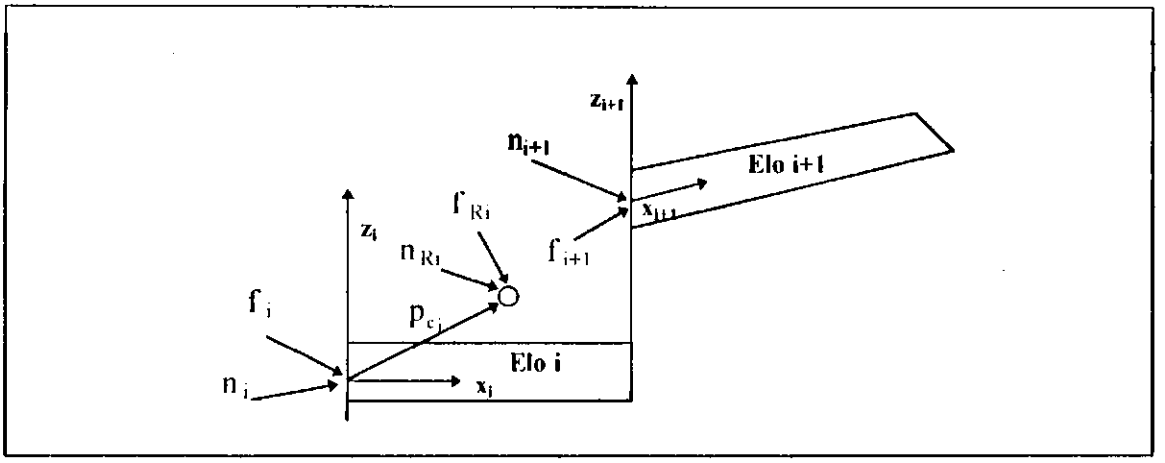

Fig. 3.3 - Balanço de Forças e Momentos no Elo i.

Desenvolvendo sucessivamente o balanço de forças e conjugados para cada elo de um manipulador, chega-se ao algoritmo de Newton-Euler, que permite calcular a dinâmica inversa de manipuladores robóticos de uma maneira recursiva eficiente [Luh, et alii, 1980]. A carga computacional deste método é de ordem N, ou seja proporcional ao número de juntas do manipulador, o que o torna ideal para aplicações em tempo real. Este método de formulação é realizado em dois estágios: Recursão para Frente, (obtida a partir das Equações (3.9) a (3.12) e suas derivadas) e Recursão para Trás, (obtida, após algumas manipulações algébricas, a partir das Equações de Newton e Euler (3.13) e (3.14), e do balanço de forças e momentos (3.15) e (3.16)). Na Recursão para Frente as variáveis cinemáticas são transformadas de um referencial de elo para outro, no sentido base-ferramenta, resultando nas velocidades e acelerações de cada elo. A Recursão para Trás transforma forças e conjugados partindo da ferramenta em direção à base do manipulador, obtendo os conjugados de acionamento de cada junta. A seguir, apresenta-se uma versão compacta do algoritmo

Newton-Euler, na qual todas as forças e conjugados atuando num elo estão expressos no seu próprio referencial.

- Recursão para Frente:

Inicialização: $v_0 = \omega_0 = \dot{\omega}_o = [0 \quad 0 \quad 0]^T$, $\dot{v}_0$ = aceleração da gravidade.

Faça i variar de 0 até N-1

$$v_{i+1} = A_{i+1}^{iT}(\omega_i \times {}^i p_{i+1} + v_i) + J_{i+1}^{(p)}\dot{q}_{i+1} \tag{3.17}$$

$$\omega_{i+1} = A_{i+1}^{iT}\omega_i + J_{i+1}^{(r)}\dot{q}_{i+1} \tag{3.18}$$

$$\dot{v}_{i+1} = A_{i+1}^{iT}(\dot{\omega}_i \times {}^i p_{i+1} + \dot{v}_i + \omega_i \times (\omega_i \times {}^i p_{i+1})) + 2(A_{i+1}^{iT}\omega_i) \times J_{i+1}^{(p)}\dot{q}_{i+1} + J_{i+1}^{(p)}\ddot{q}_{i+1} \tag{3.19}$$

$$\dot{\omega}_{i+1} = A_{i+1}^{iT}\dot{\omega}_i + (A_{i+1}^{iT}\omega_i) \times J_{i+1}^{(r)}\dot{q}_{i+1} + J_{i+1}^{(r)}\ddot{q}_{i+1} \tag{3.20}$$

Fim de i.

- Recursão para Trás

Inicialização: f_{N+1} , n_{N+1} = força , conjugado aplicado na garra.

Faça i variar de N até 1

$$f_i = A_{i+1}^{i} f_{i+1} + \dot{\omega}_i \times m_i p_{c_i} + \omega_i \times (\omega_i \times m_i p_{c_i}) + m_i \dot{v}_i \tag{3.21}$$

$$n_i = A_{i+1}^{i} n_{i+1} + {}^i p_{i+1} \times (A_{i+1}^{i} f_{i+1}) + h_i \dot{\omega}_i + \omega_i \times h_i \omega_i + m_i p_{c_i} \times \dot{v}_i \tag{3.22}$$

$$\tau_i = J_i^{(p)} f_i + J_i^{(r)} n_i \tag{3.23}$$

Fim de i.

onde as grandezas acima, expressas no próprio referencial de elo, são definidas como:

q_i	Variável de posição (deslocamento ou ângulo) da junta i.
v_i	Vetor 3x1 de velocidade linear do elo i.
ω_i	Vetor 3x1 de velocidade angular do elo i.
f_i	Vetor 3x1 de força aplicada no elo i pelo elo i-1.
n_i	Vetor 3x1 de conjugado aplicado no elo i pelo elo i-1.
A^i_{i+1}	Matriz de rotação ortogonal entre os referenciais dos elos i e i+1.
${}^i p_{i+1}$	Vetor de Posição do sistema de referência do elo i+1 em relação ao elo i.
p_{ci}	Vetor de posição do centro de massa do elo i.
m_i	Massa total do elo i.
h_i	Tensor de inércia do elo i expresso no próprio sistema de referência.
τ_i	Esforço de acionamento (força ou conjugado) da junta i.

Máscara de Junta Prismática: $J_i^{(p)} = [0\ 0\ 1]^T$ para junta i prismática.

$J_i^{(p)} = [0\ 0\ 0]^T$ para junta i rotacional.

Máscara de Junta Rotacional: $J_i^{(r)} = [0\ 0\ 0]^T$ para junta i prismática.

$J_i^{(r)} = [0\ 0\ 1]^T$ para junta i rotacional.

3.4.2 Forma em Vetores 6x1

A seguir é detalhado o Método Newton-Euler numa forma equivalente baseada em vetores 6x1, parecida à descrita em [Khosla, 1989]. Para isto as seguintes definições são feitas:

Operador Matriz 3x3 Produto Vetorial:

$$(p\times) = \begin{bmatrix} 0 & -p_z & p_y \\ p_z & 0 & -p_x \\ -p_y & p_x & 0 \end{bmatrix} \qquad (3.24)$$

Operador Matriz 6x6 Produto Vetorial:

$$\left(\begin{bmatrix} v \\ \omega \end{bmatrix}\times\right)=\begin{bmatrix} (\omega\times) & 2(v\times) \\ 2(v\times) & (\omega\times) \end{bmatrix} \tag{3.25}$$

Matriz 6x6 de Rotação Espacial do elo i para o elo i+1:

$$R_{i+1}^{i} = \begin{bmatrix} A_{i+1}^{i} & \mathbf{0} \\ ({}^{i}p_{i+1}\times)A_{i+1}^{i} & A_{i+1}^{i} \end{bmatrix} \tag{3.26}$$

Vetor 6x1 de Velocidade Espacial do elo i:

$$V_i = \begin{bmatrix} v_i^T & \omega_i^T \end{bmatrix}^T \tag{3.27}$$

Vetor 6x1 de Esforço Espacial do elo i:

$$F_i = \begin{bmatrix} f_i^T & n_i^T \end{bmatrix}^T \tag{3.28}$$

Vetor 6x1 de Velocidade Angular Espacial do elo i:

$$\Omega_i = \begin{bmatrix} 0^T & \omega_i^T \end{bmatrix}^T \tag{3.29}$$

Vetor 6x1 de Máscara da Junta i:

$$J_i = \begin{bmatrix} J_i^{(p)} \\ J_i^{(r)} \end{bmatrix} \tag{3.30}$$

Matriz simétrica 6x6 de parâmetros dinâmicos do elo i:

$$H_i = \begin{bmatrix} m_i I & m_i(p_{c_i}\times)^T \\ m_i(p_{c_i}\times) & h_i \end{bmatrix} \tag{3.31}$$

Assim, o Método Newton-Euler em termos de vetores 6x1 é dado por:

- Recursão para Frente:

Inicialização: $V_0 = 0$ $\dot{V}_0$ = aceleração da gravidade.

Faça i variar de 0 até N-1

$$V_{i+1} = R_{i+1}^{iT}.V_i + J_{i+1}\dot{q}_{i+1} \tag{3.32}$$

$$\dot{V}_{i+1} = R_{i+1}^{iT}.\dot{V}_i + \dot{V}_{c_i} + J_{i+1}\ddot{q}_{i+1} \tag{3.33}$$

Fim de i.

onde o Vetor 6x1 de Aceleração Espacial Coriolis/Centrífuga do elo i é dado por:

$$\dot{V}_{c_i} = -[(\Omega_i\times)(R_{i+1}^{iT})^{-1} + \dot{q}_{i+1}(J_{i+1}\times)](R_{i+1}^{i})^{-1}\Omega_i \tag{3.34}$$

- Recursão para Trás:

Inicialização: F_{N+1} = Esforço espacial aplicado na garra.

Faça i variar de N até 1

$$F_i = R_{i+1}^{i}F_{i+1} + H_i\dot{V}_i + F_{c_i} \tag{3.35}$$

$$\tau_i = J_i^T.F_i \tag{3.36}$$

Fim de i.

onde o Vetor 6x1 de Esforço Espacial Coriolis/Centrífugo do elo i é dado por:

$$F_{c_i} = (\Omega_i\times)H_i\Omega_i \tag{3.37}$$

3.5 Simulação e Controle

Os problemas de simulação e controle de robôs estão estreitamente relacionados à resolução das equações de dinâmica direta e inversa. O problema de controle pode ser estabelecido da seguinte maneira: dada a trajetória desejada $q^*, \dot{q}^*, \ddot{q}^*$ para as variáveis de junta, determinar os esforços de acionamento τ necessários para realizar esta trajetória. O cálculo da equação de dinâmica inversa é freqüentemente utilizado para resolver este problema, sendo a base de grande número de estratégias de controle baseadas em modelo, tal como o popular método Torque Computado. Neste contexto, a eficiência computacional do Método Newton-Euler é indispensável para implementações em tempo real.

O problema de simulação dinâmica consiste em determinar a evolução no tempo das trajetória das juntas $q, \dot{q}, \ddot{q}$ dado o vetor τ de esforços de acionamentos aplicados pelos atuadores. A solução deste problema é alcançada através de integração numérica da equação de dinâmica direta, conforme é esquematizado na Figura 3.4.

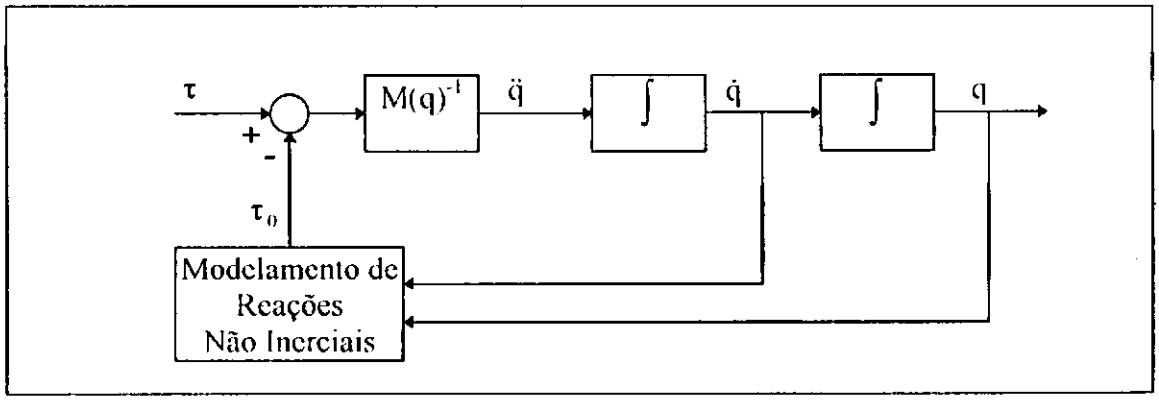

Fig. 3.4 - Simulação Dinâmica de Manipuladores.

O problema de simulação da dinâmica de manipuladores envolve a integração da Equação (3.5) de dinâmica direta através de métodos numéricos. Pode-se notar que a inversa da matriz de inércia $M(q)^{-1}$ é necessária para obter as acelerações de junta. Assim, o método de formulação Newton-Euler acima não pode ser utilizado diretamente para obter a dinâmica direta, já que este apenas computa numericamente o vetor de forças/conjugados de acionamento. Apesar disto, um artifício pode ser utilizado para obter a matriz de inércia através deste método [Walker & Orin, 1982]. Primeiro, o vetor de reações não inerciais τ_0 é computado a partir de,

$$\tau_0 = \tau^{(0)} \tag{3.38}$$

onde o vetor $\tau^{(0)}$ é obtido através do algoritmo Newton-Euler assumindo as acelerações de junta iguais a zero, $\ddot{q}_i = 0$ para $i = 1, \ldots\ldots, N$. Particionando a matriz de inércia em N colunas M_i:

$$M(q) = [M_1 \quad M_2 \ldots\ldots M_i \ldots\ldots M_N] \tag{3.39}$$

Da Equação (3.7) de dinâmica inversa, verifica-se que uma coluna M_i da matriz de inércia é numericamente igual ao vetor de esforços de acionamento obtido para aceleração da junta i igual a 1 e as das juntas restantes nulas, menos o vetor τ_0:

$$M_i = \begin{bmatrix} M_1 & M_2 & \cdots & M_i & \cdots & M_N \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} = \tau^{(i)} - \tau_0 \tag{3.40}$$

onde o vetor $\tau^{(i)}$ é obtido através do algoritmo Newton-Euler impondo $\ddot{q}_j = 1$ para $j = i$ e $\ddot{q}_j = 0$ para $j \neq i$.

3.6 Cálculo Recursivo da Matriz de Inércia

3.6.1 Diagrama de Blocos

Nesta Seção, são desenvolvidos dois métodos para cálculo recursivo da matriz de inércia de manipuladores robóticos [Alsina & Gehlot, 1996-C], [Alsina & Gehlot, 1994-A], [Alsina & Gehlot, 1994-D]. Estes dois métodos, baseados num diagrama de blocos da equação dinâmica desenvolvido a partir do Método Newton-Euler em vetores 6x1, permitem obter explicitamente a matriz de inércia de maneira recursiva e computacionalmente eficiente. O vetor de esforços não inerciais τ_0 é obtido convencionalmente pelo método Newton-Euler, como descrito na Equação (3.38), assumindo nulas as acelerações de junta. O esforço computacional envolvido no cômputo do vetor de esforços não inerciais τ_0 é sumarizado na Tabela 3.1.

Tabela 3.1 - Esforço computacional para o vetor τ_0 .

Nº de Produtos	Nº de Somas
156.N-80	108.N-51

Definem-se, para o elo i os vetores 6x1 de Aceleração Espacial Inercial $\dot{V}_{mi}$ e Esforço Espacial Inercial F_{mi} , respectivamente, como os vetores de aceleração espacial e esforço espacial obtidos pelo algoritmo Newton-Euler 6x1, impondo $V_0 = \dot{V}_0 = \dot{V}_{ci} = F_{N+1} = F_{ci} = 0$. Define-se também o vetor de reações inerciais τ_m como o vetor solução do algoritmo Newton-Euler para as condições acima, que numericamente é igual a $M(q)\ddot{q}$. Assim, de acordo com as restrições acima, o vetor τ_m pode ser obtido diretamente a partir da seguinte versão modificada da formulação Newton-Euler:

- Recursão para Frente para as Reações Inerciais:

Inicialização: $\dot{V}_{m0} = 0$

Faça i variar de 0 até N-1

$$\dot{V}_{mi+1} = R_{i+1}^{iT} \cdot \dot{V}_{mi} + J_{i+1}\ddot{q}_{i+1} \tag{3.41}$$

Fim de i.

- Recursão para Trás para as Reações Inerciais:

Inicialização: $F_{mN+1} = 0$

Faça i variar de N até 1

$$F_{mi} = R_{i+1}^{i} F_{mi+1} + H_i \dot{V}_{mi} \tag{3.42}$$

$$\tau_{mi} = J_i^T \cdot F_{mi} \tag{3.43}$$

Fim de i

Expressando por meio de diagrama de blocos as Equações (3.41), (3.42) e (3.43), junto com as reações não inerciais τ_0, sucessivamente para todas as juntas, uma representação em diagrama de blocos da equação de dinâmica inversa é obtida, conforme mostrado na Figura 3.5.

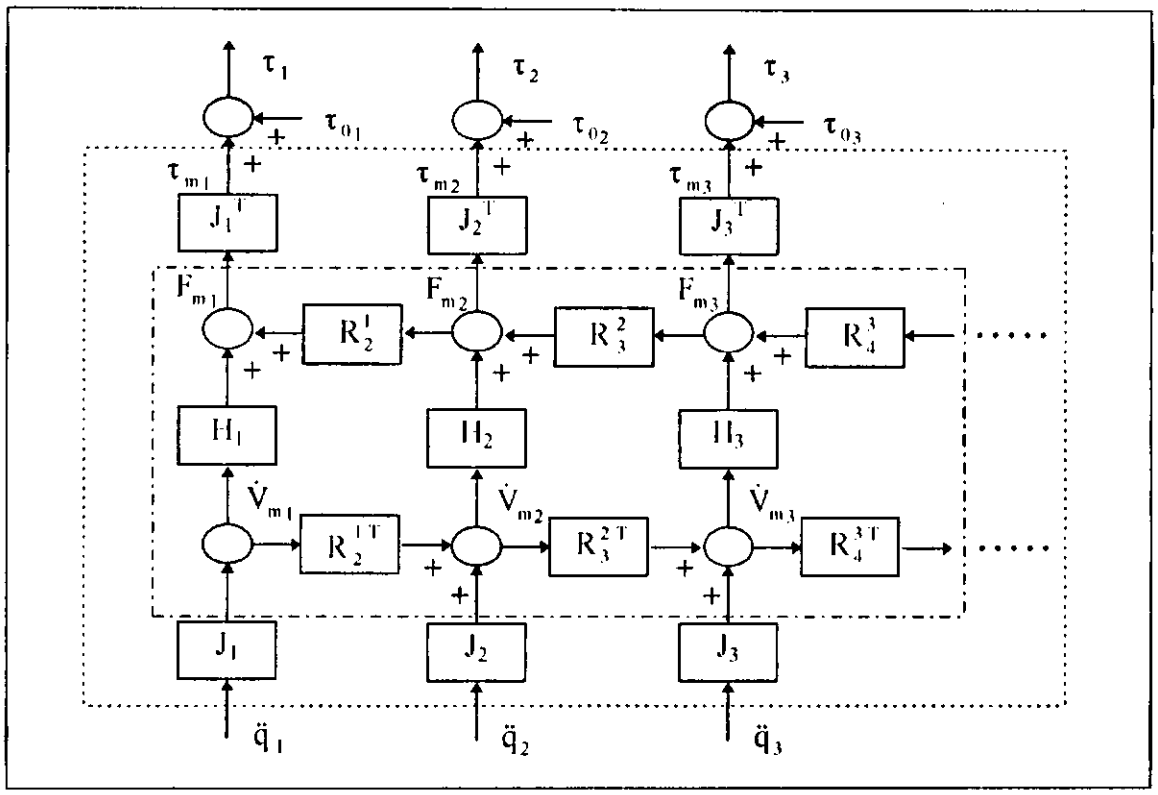

Fig. 3.5 - Diagrama de Blocos: Matriz de Inércia e Dinâmica Inversa.

3.6.2 Método dos Caminhos Inversos

Nota-se que, na Figura 3.5, o diagrama de blocos delimitado pela linha tracejada externa representa a matriz de transferência entre acelerações de junta e reações inerciais, ou seja, a matriz de inércia. A seguinte notação é adotada:

$[\tau_m, F_m]_{ij}$ = Matriz de transferência 1x6 associada ao caminho de F_{m_j} até τ_{m_i}.

$[\tau_m, \dot{V}_m]_{ij}$ = Matriz de transferência 1x6 associada ao caminho de $\dot{V}_{m_j}$ até τ_{m_i}.

$[F_m, \dot{V}_m]_{ij}$ = Matriz de transferência 6x6 associada ao caminho de $\dot{V}_{m_j}$ até F_{m_i}.

$[F_m, \ddot{q}]_{ij}$ = Matriz de transferência 6x1 associada ao caminho de $\ddot{q}_j$ até τ_{m_i}.

Assim, a partir do diagrama de blocos da Figura 3.5, as seguintes relações recursivas são obtidas:

$$[\tau_m, F_m]_{i,j+1} = [\tau_m, F_m]_{ij} . R_{j+1}^{j} \tag{3.44}$$

$$[\tau_m, \dot{V}_m]_{ij} = [\tau_m, F_m]_{ij} . H_j + [\tau_m, \dot{V}_m]_{i,j+1} . R_{j+1}^{jT} \tag{3.45}$$

$$M_{ij} = [\tau_m, \dot{V}_m]_{ij} . J_j \tag{3.46}$$

Visto que a matriz de inércia é simétrica, a mesma pode ser obtida através das equações acima, para $j \geq i$. Devido a que as matrizes de transferência são calculadas recursivamente, no sentido contrário aos caminhos do diagrama (das reações inerciais para as acelerações de junta), este método é denominado: "Método dos Caminhos Inversos". Assim, a partir das recursões (3.44) a (3.46), o seguinte algoritmo recursivo para cálculo da matriz de inércia é proposto:

Algoritmo para cálculo da Matriz de Inércia - Método dos Caminhos Inversos:

Faça i variar de 1 até N

Inicializar: $[\tau_m, F_m]_{ii} = J_i^T$

Se (i < N) Faça $[\tau_m, F_m]_{i,i+1} = [\tau_m, F_m]_{ii} R_{i+1}^{i}$

Se (i < N-1) Faça j variar de i+2 até N

$$[\tau_m, F_m]_{ij} = [\tau_m, F_m]_{i,j-1} . R_j^{j-1}$$

Fim de j.

$[\tau_m, \dot{V}_m]_{iN} = [\tau_m, F_m]_{iN} . H_N$

$M_{iN} = [\tau_m, \dot{V}_m]_{iN} . J_N$

Se (i < N) Faça k variar de N-1 até i

$$[\tau_m, \dot{V}_m]_{ik} = [\tau_m, F_m]_{ik} . H_k + [\tau_m, \dot{V}_m]_{i,k+1} . R_{k+1}^{kT}$$

$$M_{ik} = [\tau_m, \dot{V}_m]_{ik} . J_k$$

Fim de k

Fim de i

O método proposto envolve alguns produtos e somas de vetores 6x1 e matrizes 6x6. Nota-se que as matrizes 6x6 de inércia espacial e rotação espacial incluem alguns elementos nulos. Assim, o produto entre uma matriz de inércia espacial e um vetor 6x1 envolve 24 produtos e 18 somas. Analogamente, o produto entre uma matriz de rotação espacial e um vetor 6x1 envolve 27 produtos e 21 somas. Desta maneira, o Método dos Caminhos Inversos para cálculo da matriz de inércia envolve um esforço computacional de ordem $O(N^2)$, que é sumarizado na Tabela 3.2. Para um manipulador de seis juntas, o método proposto envolve 1179 produtos e 993 somas. A multiplicação pelos vetores J_i e J_i^T não envolve nenhum esforço computacional adicional, desde que a pré-multiplicação pelo vetor de máscara J_i^T é, de fato, a escolha de uma linha (a 3ª ou a 6ª, caso a junta i seja prismática ou rotacional) de uma dada submatriz 6x6; de maneira similar, a pós-multiplicação pelo vetor J_i corresponde à escolha de uma coluna de uma dada submatriz 6x6.

Tabela 3.2 - Esforço Computacional: Método dos Caminhos Inversos.

Nº de Produtos	Nº de Somas
$39N^2 - 42N + 27$	$33N^2 - 36N + 21$

3.6.3 Método dos Caminhos Centrais

Na Figura 3.5, nota-se que, dentro do diagrama de blocos delimitado pela linha tracejada-pontilhada, um caminhos central $[F_m, \dot{V}_m]_{ii}$ pode ser dividido em dois caminhos paralelos: um direto, através da matriz H_i, e outro passando por $R_{i+1}^{i\,T}$, $[F_m, \dot{V}_m]_{i+1,i+1}$, R_{i+1}^{i}. Por outro lado, percebe-se que a matriz de transferência associada ao caminho $[F_m, \ddot{q}]_{ji}$ é composta da matriz R_{j+1}^{i} em série com a matriz de transferência do caminho $[F_m, \ddot{q}]_{j+1,i}$. Levando em conta estas considerações, as seguintes recursões são obtidas:

$$[F_m, \dot{V}_m]_{ii} = H_i + R_{i+1}^i . [F_m, \dot{V}_m]_{i+1,i+1} . R_{i+1}^{iT} \tag{3.47}$$

$$[F_m, \ddot{q}]_{ji} = R_{j+1}^j . [F_m, \ddot{q}]_{j+1,i} \tag{3.48}$$

A partir das recursões acima, é proposto um novo algoritmo recursivo para cômputo da matriz de inércia, denominado de Método dos Caminhos Centrais:

Algoritmo para cálculo da Matriz de Inércia - Método dos Caminhos Centrais:

Inicializar: $[F_m, \dot{V}_m]_{NN} = H_N$

$[F_m, \ddot{q}]_{NN} = [F_m, \dot{V}_m]_{NN} . J_N$

$M_{NN} = J_N^T . [F_m, \ddot{q}]_{NN}$

Faça i variar de N-1 até 1

$[F_m, \dot{V}_m]_{ii} = H_i + R_{i+1}^i . [F_m, \dot{V}_m]_{i+1,i+1} . R_{i+1}^{iT}$

$[F_m, \ddot{q}]_{ii} = [F_m, \dot{V}_m]_{ii} . J_i$

$M_{ii} = J_i^T . [F_m, \ddot{q}]_{ii}$

Faça j variar de i até 1

$[F_m, \ddot{q}]_{j,i+1} = R_{j+1}^j . [F_m, \ddot{q}]_{j+1,i+1}$

$M_{j,i+1} = J_j^T . [F_m, \ddot{q}]_{j,i+1}$

Fim de j

Fim de i

Nota-se que o algoritmo acima computa os elementos M_{ij} para $i \leq j$ (diagonal superior). Como a matriz de inércia M(q) é simétrica não é necessário calcular os elementos restantes abaixo da diagonal. A Tabela 3.3 mostra o esforço computacional envolvido no cálculo da matriz de inércia pelo Método dos Caminhos Centrais. Nota-se que o número de operações de máquina é de ordem $O(N^2)$. Para um

manipulador de seis juntas, o algoritmo acima envolve 2025 produtos e 1695 somas. A comparação entre os esforços computacionais para o cálculo da matriz de inércia através do Método dos Caminhos Centrais e do Método de Walker (baseado no algoritmo Newton-Euler), excluindo o cômputo do vetor τ_0 , é mostrada nos gráficos da Figura 3.6. Pode-se observar pela Figura que o Método dos Caminhos Centrais é mais eficiente do que o Método de Walker para N maior do que seis juntas (manipuladores redundantes).

Tabela 3.3 - Esforço Computacional: Método dos Caminhos Centrais.

Produtos	Somas
$(27N^2+621N-648)/2$	$(21N^2+531N-552)/2$

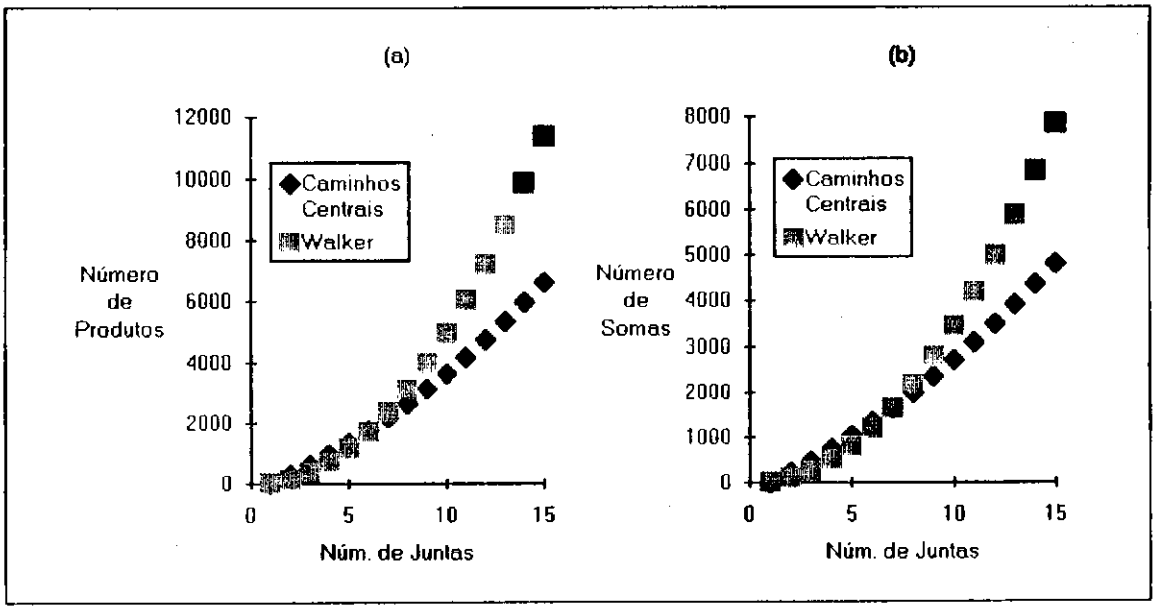

Fig. 3.6 - Esforço Computacional para a Matriz de Inércia.
a) Número de Produtos. b) Número de Somas.

3.7 Forma Fatorada da Matriz de Inércia

A seguir, uma forma fatorada para a matriz de inércia é derivada. Resolvendo sucessivamente para cada junta, as recursões "Para Frente" e "Para Trás" (Equações (3.41) a (3.43)), para cálculo das reações inerciais, os seguintes resultados são encontrados:

Acelerações Espaciais Inerciais:

$$\dot{V}_{m1} = J_1\ddot{q}_1 \tag{3.49.a}$$

$$\dot{V}_{m2} = R_2^{1T}\dot{V}_{m1} + J_2\ddot{q}_2 = R_2^{1T}J_1\ddot{q}_1 + J_2\ddot{q}_2 \tag{3.49.b}$$

$$\dot{V}_{m3} = R_3^{2T}\dot{V}_{m2} + J_3\ddot{q}_3 = R_3^{2T}(R_2^{1T}J_1\ddot{q}_1 + J_2\ddot{q}_2) + J_3\ddot{q}_3 = R_3^{1T}J_1\ddot{q}_1 + R_3^{1T}J_2\ddot{q}_2 + J_3\ddot{q}_3 \tag{3.49.c}$$

.....

Esforços Espaciais Inerciais:

$$F_{mN} = H_N\dot{V}_{mN} \tag{3.50.a}$$

$$F_{mN-1} = R_N^{N-1}F_{mN} + H_{N-1}\dot{V}_{mN-1} = R_N^{N-1}H_N\dot{V}_{mN} + H_{N-1}\dot{V}_{mN-1} \tag{3.50.b}$$

$$F_{mN-2} = R_{N-1}^{N-2}F_{mN-1} + H_{N-2}\dot{V}_{mN-2} = R_N^{N-2}H_N\dot{V}_{mN} + R_{N-1}^{N-2}H_{N-1}\dot{V}_{mN-1} + H_{N-2}\dot{V}_{mN-2} \tag{3.50.c}$$

.....

Reações Inerciais:

$$\tau_{m1} = J_1^T F_{m1} \tag{3.51.a}$$

$$\tau_{m2} = J_2^T F_{m2} \tag{3.51.b}$$

$$\tau_{m3} = J_3^T F_{m3} \tag{3.51.c}$$

.....

Nas Equações acima foi utilizada a notação:

$$R_j^i = R_{i+1}^i . R_{i+2}^{i+1} .... R_{j-1}^{j-2} . R_j^{j-1} \tag{3.52}$$

Definindo o vetor global 6Nx1 de acelerações espaciais inerciais, $\dot{V}_m$, e o vetor global 6Nx1 de esforços espaciais inerciais, F_m , respectivamente como:

$$\dot{V}_m = \left[\dot{V}_{m1}^T \quad \dot{V}_{m2}^T \quad \cdots \quad \dot{V}_{mN}^T \right]^T \tag{3.53}$$

$$F_m = \left[F_{m1}^T \quad F_{m2}^T \quad \cdots \quad F_{mN}^T \right]^T \tag{3.54}$$

Definindo respectivamente a matriz global 6Nx6N de rotação espacial, R, a matriz global 6Nx6N de parâmetros dinâmicos, H, e a matriz global 6NxN de máscara de juntas, J:

$$R = \begin{bmatrix} I & R_2^1 & R_3^1 & .... & R_N^1 \\ 0 & I & R_3^2 & .... & R_N^2 \\ 0 & 0 & .... & .... & .... \\ .... & .... & .... & I & R_N^{N-1} \\ 0 & 0 & .... & 0 & I \end{bmatrix} \tag{3.55}$$

$$H = \begin{bmatrix} H_1 & 0 & ... & 0 \\ 0 & H_2 & ... & ... \\ ... & ... & ... & 0 \\ 0 & ... & 0 & H_N \end{bmatrix} \tag{3.56}$$

$$J = \begin{bmatrix} J_1 & 0 & ... & 0 \\ 0 & J_2 & ... & ... \\ ... & ... & ... & 0 \\ 0 & ... & 0 & J_N \end{bmatrix} \tag{3.57}$$

Então, as Equações (3.49), (3.50) e (3.51) podem ser agrupadas em relações globais:

$$\dot{V}_m = R^T.J.\ddot{q} \tag{3.58}$$

$$F_m = R.H.\dot{V}_m = R.H.R^T.J.\ddot{q} \tag{3.59}$$

$$\tau_m = J^T.F_m = (J^T.R.H.R^T.J)\ddot{q} \tag{3.60}$$

Por outro lado, como $\tau_m = M(q)\ddot{q}$, então a seguinte importante forma fatorada para a matriz de inércia é obtida:

$$M(q) = (J^T.R.H.R^T.J) \tag{3.61}$$

e a equação de dinâmica inversa pode ser escrita da seguinte maneira:

$$\tau = M(q)\ddot{q} + \tau_0 \quad \Rightarrow \quad \tau = \tau_m + \tau_0 \quad \Rightarrow \quad \tau = (J^T.R.H.R^T.J)\ddot{q} + \tau_0 \tag{3.62}$$

Desenvolvendo, como exemplo, a Equação (3.61) para um manipulador de três juntas:

$$M(q) = \begin{bmatrix} J_1^T(H_1 + R_2^1H_2R_2^{1T} + R_3^1H_3R_3^{1T})J_1 & J_1^T(R_2^1H_2 + R_3^1H_3R_3^{2T})J_2 & J_1^T(R_3^1H_3)J_3 \\ J_2^T(H_2R_2^{1T} + R_3^2H_3R_3^{1T})J_1 & J_2^T(H_2 + R_3^2H_3R_3^{2T})J_2 & J_2^T(R_3^2H_3)J_3 \\ J_3^T(H_3R_3^{1T})J_1 & J_3^T(H_3R_3^{2T})J_2 & J_3^T(H_3)J_3 \end{bmatrix} \tag{3.63}$$

No diagrama de blocos da Figura 3.5, a parte delimitada pela linha tracejada externa corresponde à matriz de inércia $M(q) = [J^T.R.H.R^T.J]$ e a parte delimitada pela linha tracejada-pontilhada interna corresponde à matriz 6Nx6N $[R.H.R^T]$. Esta matriz pode ser particionada naturalmente em NxN submatrizes 6x6 (os termos entre parênteses na Equação (3.63)).

3.8 Conclusão

Neste Capítulo, foram discutidos métodos de formulação para as equações de dinâmica direta e inversa de manipuladores, destacando a sua estreita relação com os problemas de simulação e controle. O método recursivo Newton-Euler para cálculo da dinâmica inversa foi apresentado e uma versão compacta em termos de vetores 6x1 foi desenvolvida. O método de Walker, para obtenção da matriz de inércia através do algoritmo Newton-Euler, foi sumarizado. Uma representação em forma de diagrama de blocos para a matriz de inércia e a equação dinâmica de manipuladores robóticos foi derivada. A partir deste diagrama, dois algoritmos recursivos de ordem $O(N^2)$ para cômputo da matriz de inércia foram desenvolvidos: o Método dos Caminhos Inversos e o Método dos Caminhos Centrais. Em termos de esforço computacional, o Método dos Caminhos Centrais mostrou-se superior ao método de Walker para manipuladores com mais de seis juntas (manipuladores redundantes). Uma elegante forma fatorada para a matriz de inércia foi encontrada.

CAPÍTULO 4

CONTROLE ADAPTATIVO DE MANIPULADORES

4.1 Introdução

Neste capítulo, descreve-se um método de controle adaptativo modular para manipuladores robóticos. O clássico controlador Torque Computado, baseado em realimentação proporcional-derivativa e compensação de modelo dinâmico é apresentado sucintamente na Seção 4.2. Na Seção 4.3 a equação de dinâmica inversa é reescrita em forma Linear-Em-Parâmetros. Na Seção 4.4 descreve-se um método recursivo para o cálculo da matriz de regressores. O controlador adaptativo baseado em modelo é descrito na Seção 4.5 e a prova de estabilidade é sumarizada na Seção 4.6. Uma forma modular para o controlador adaptativo é proposta na Seção 4.7. Na Seção 4.8 são apresentados resultados de simulação para um manipulador de três juntas que atestam o bom desempenho do controlador adaptativo modular proposto.

4.2 Controlador Torque Computado

A maioria dos manipuladores industriais utilizados atualmente são dotados de engrenagens redução, o que permite atenuar em certo grau o acoplamento dinâmico entre as juntas. Assim, para aplicações de controle de posição em baixa velocidade, controladores de juntas Proporcionais-Derivativos independentes podem ser utilizados, obtendo-se um desempenho razoável. Os ganhos destes controladores são ajustados de maneira a garantir um controle robusto em toda a faixa de operação. Devido às não linearidades e acoplamentos entre juntas, o desempenho destes controladores varia à medida em que o manipulador muda de configuração. Para aplicações de rastreamento de trajetória, nas quais são exigidos uma maior precisão e um desempenho uniforme em toda a faixa de operação, deve-se aplicar algum tipo de compensação do modelo dinâmico de maneira a garantir as exigências de projeto. Nesta

Seção descreve-se o clássico controlador Torque Computado [Luh, et alii, 1980] para manipuladores robóticos. Este controlador é implementado em dois estágios: realimentação PD e Compensação de modelo dinâmico (ver Figura 4.1):

- Realimentação PD:

$$\ddot{q}_r = \ddot{q}^* + K_D(\dot{q}^* - \dot{q}) + K_P(q^* - q) \qquad (4.1)$$

- Compensação de Modelo:

$$\tau = \hat{M}(q).\ddot{q}_r + \hat{\tau}_0 \qquad (4.2)$$

onde q^* é o vetor Nx1 de posições de junta desejadas e $\ddot{q}_r$ é o vetor Nx1 de acelerações de referência. K_D e K_P são matrizes diagonais NxN de ganhos derivativos e proporcionais respectivamente. A Equação (4.2) representa o modelo dinâmico do manipulador e os termos denotados por ($\hat{}$) são obtidos com os valores aproximados disponíveis dos parâmetros dinâmicos. A compensação de modelo pode ser computada em tempo real por meio da formulação Newton-Euler.

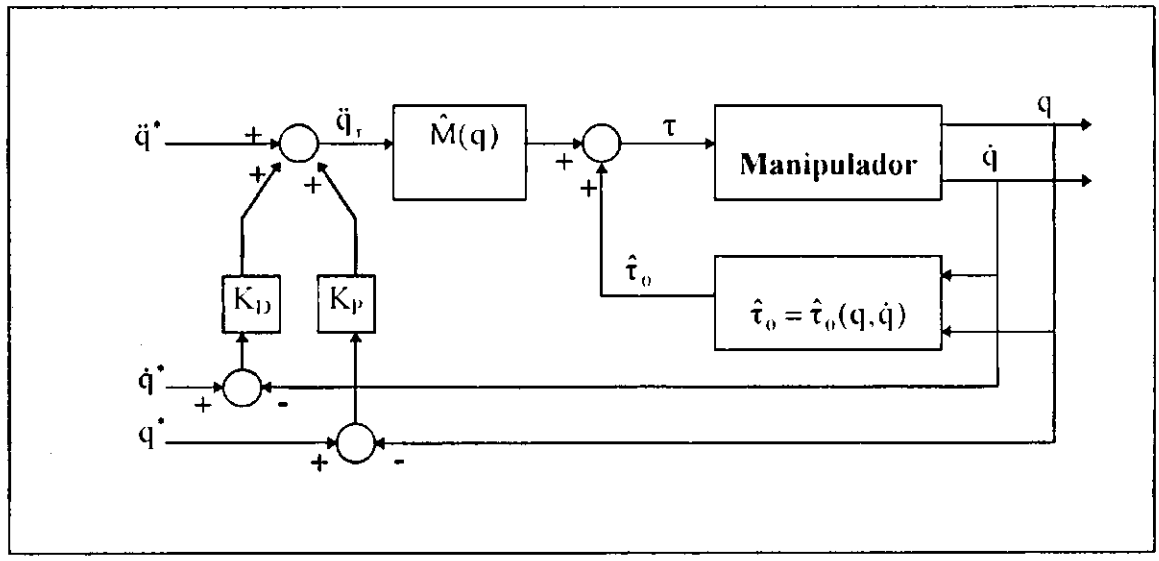

Fig. 4.1 - Controlador Torque Computado.

Substituindo as Equações (4.1) e (4.2) do controlador Torque Computado no modelo dinâmico do manipulador dado pela Equação (3.7), a dinâmica do sistema em malha fechada é obtida:

$$(\ddot{q}^* - \ddot{q}) + K_D(\dot{q}^* - \dot{q}) + K_P(q^* - q) = \hat{M}(q)^{-1}[(M(q) - \hat{M}(q))\ddot{q} + (\tau_0(q,\dot{q}) - \hat{\tau}_0(q,\dot{q}))] \quad (4.3)$$

Verifica-se que, quando os parâmetros do sistema não são conhecidos com precisão, a dinâmica de erro será governada pela Equação dinâmica (4.3), altamente não linear e acoplada, já que a matriz M(q) não é, geralmente, diagonal. Assim, uma perturbação numa única junta, introduzirá erros em todas as outras. Por outro lado, quando o modelo disponível é exatamente igual à dinâmica do manipulador real ($\hat{M}(q) = M(q)$ e $\hat{\tau}_0(q,\dot{q}) = \tau_0(q,\dot{q})$), o lado direito da Equação (4.3) é anulado e a dinâmica de erro torna-se desacoplada,

$$(\ddot{q}^* - \ddot{q}) + K_D(\dot{q}^* - \dot{q}) + K_P(q^* - q) = 0 \quad (4.4)$$

Escolhendo,

$$K_D = \text{diag}(2.\zeta.\omega_n) \quad (4.5.a)$$

$$K_P = \text{diag}(\omega_n^2) \quad (4.5.b)$$

então, o sistema em malha fechada pode ser considerado como um conjunto de N sistemas lineares independentes de segunda ordem, com coeficiente de amortecimento ζ e freqüência natural ω_n. Para manipuladores de acionamento direto (sem engrenagens de redução) e operação em alta velocidade, mesmo pequenos erros de modelagem podem afetar sensivelmente o desempenho, de maneira que técnicas de identificação de parâmetros ou controle adaptativo tornam-se necessárias.

4.3 Modelo Linear-Em-Parâmetros

A maioria dos controladores adaptativos baseados no modelo não linear de manipuladores são formulados assumindo uma forma Linear-Em-Parâmetros para a equação dinâmica. Nesta Seção, é obtida a forma Linear-Em-Parâmetros da equação dinâmica a partir do método recursivo Newton-Euler. Analisando o método Newton-Euler (Equações (3.32) a (3.37)), verifica-se que os parâmetros dinâmicos entram apenas na recursão "Para Trás", sendo que a recursão "Para Frente" envolve apenas variáveis cinemáticas. A seguir, uma forma Linear-Em-Parâmetros para a recursão "Para Trás" é desenvolvida. Definindo o Operador Matriz 3x6 Produto Escalar sobre um vetor ω 3x1 como:

$$(\bullet\omega) = \begin{bmatrix} \omega_x & 0 & 0 & 0 & \omega_z & \omega_y \\ 0 & \omega_y & 0 & \omega_z & 0 & \omega_x \\ 0 & 0 & \omega_z & \omega_y & \omega_x & 0 \end{bmatrix} \tag{4.6}$$

De acordo com esta definição, pode-se rescrever:

$$h_i.\omega_i = (\bullet\omega_i).\vec{h}_i \tag{4.7}$$

onde o vetor de inércia do elo i equivalente ao tensor h_i é dado por:

$$\vec{h}_i = \begin{bmatrix} h_{ixx} & h_{iyy} & h_{izz} & h_{iyz} & h_{izx} & h_{ixy} \end{bmatrix}^T \tag{4.8}$$

Utilizando esta nomenclatura, os termos da recursão "Para Trás" que dependem dos parâmetros dinâmicos podem ser reorganizados na seguinte forma Linear-Em-Parâmetros:

$$H_i\dot{V}_i + (\Omega_i\times)H_i.\Omega_i = K_i.\vec{H}_i \tag{4.9}$$

onde a matriz 6x10 de regressores do elo i, que depende apenas das variáveis cinemáticas é definida como:

$$K_i = \left[\begin{array}{c|c|c} \dot{v}_i & (\dot{\omega}_i\times)+(\omega_i\times)(\omega_i\times) & 0 \\ \hline 0 & (-\dot{v}_i\times) & (\bullet\dot{\omega}_i)+(\omega_i\times)(\bullet\omega_i) \end{array}\right] \tag{4.10}$$

e o vetor 10x1 de parâmetros dinâmicos do elo i é definido como:

$$\vec{H}_i = \left[m_i \quad (m_i.p_{c_i})^T \quad (\vec{h}_i)^T\right]^T \tag{4.11}$$

Então, a recursão Newton-Euler "Para Trás" em forma Linear-Em-Parâmetros é:

Recursão Newton-Euler "Para Trás" - Forma Linear-Em-Parâmetros:

Inicialização: F_{N+1} = Esforço espacial aplicado na garra.

Faça i variar de N até 1

$$F_i = R_{i+1}^i F_{i+1} + K_i.\vec{H}_i \tag{4.12}$$

$$\tau_i = J_i^T.F_i \tag{4.13}$$

Fim de i

resolvendo a recursão "Para Trás" Linear-Em-Parâmetros sucessivamente para cada junta, os seguintes resultados são encontrados:

Esforços Espaciais:

$$F_N = K_N\vec{H}_N \tag{4.14.a}$$

$$F_{N-1} = R_N^{N-1}F_N + K_{N-1}\vec{H}_{N-1} = R_N^{N-1}K_N\vec{H}_N + K_{N-1}\vec{H}_{N-1} \tag{4.14.b}$$

$$F_{N-2} = R_{N-1}^{N-2}F_{N-1} + K_{N-2}\vec{H}_{N-2} = R_N^{N-2}K_N\vec{H}_N + R_{N-1}^{N-2}K_{N-1}\vec{H}_{N-1} + K_{N-2}\vec{H}_{N-2} \tag{4.14.c}$$

.....

Esforços de Acionamento:

$$\tau_1 = J_1^T F_1 \qquad (4.15.a)$$

$$\tau_2 = J_2^T F_2 \qquad (4.15.b)$$

$$\tau_3 = J_3^T F_3 \qquad (4.15.c)$$

.....

Substituindo as Equações (4.14) nas Equações (4.15) e agrupando em forma matricial, a equação de dinâmica inversa do manipulador é expressa em forma Linear-Em-Parâmetros:

$$\tau = \Phi.\vec{H} \qquad (4.16)$$

onde,

τ = Vetor Nx1 de esforços de acionamento.

$\vec{H}$ = Vetor 10Nx1 de parâmetros dinâmicos do manipulador.

$\Phi = \Phi(q,\dot{q},\dot{q},\ddot{q})$ Matriz Nx10N de regressores do manipulador.

O vetor de parâmetros dinâmicos é dado por:

$$\vec{H} = \left[\vec{H}_1^T \quad \vec{H}_2^T \quad ..... \quad \vec{H}_N^T\right]^T \qquad (4.17)$$

A matriz de regressores do manipulador é dada por:

$$\Phi = \begin{bmatrix} J_1^T K_1 & J_1^T R_2^1 K_2 & .... & J_1^T R_N^1 K_N \\ 0 & J_2^T K_2 & .... & J_2^T R_N^2 K_N \\ .... & .... & .... & .... \\ 0 & 0 & .... & J_N^T K_N \end{bmatrix} \qquad (4.18)$$

É interessante notar que a Equação (4.18) pode ser fatorada da seguinte forma:

$$\Phi = J^T . R . K \tag{4.19}$$

onde a matriz 6Nx10N de regressores de elos global é definida como:

$$K = \begin{bmatrix} K_1 & 0 & \dots & 0 \\ 0 & K_2 & \dots & \dots \\ \dots & \dots & \dots & 0 \\ 0 & \dots & 0 & K_N \end{bmatrix} \tag{4.20}$$

Desta maneira, a Equação (4.16) de dinâmica inversa é expressa como:

$$\tau = J^T . R . K . \vec{H} \tag{4.21}$$

4.4 Cálculo da Matriz de Regressores

O esforço computacional envolvido no cálculo da matriz de regressores $\Phi(q,\dot{q},\dot{q},\ddot{q})$ dada na Equação (4.18) torna-se considerável para manipuladores com mais de três grau de liberdade, o que coloca dificuldade em implementações práticas, que dependem da sua obtenção a cada período de amostragem. Visando superar a complexidade computacional envolvida, é desenvolvido a seguir um método recursivo eficiente para o cálculo da matriz de regressores. A Figura 4.2 mostra o diagrama de blocos para o cálculo da dinâmica inversa em forma Linear-Em-Parâmetros correspondente à Equação (4.21). A linha tracejada delimita a parte do diagrama correspondente à matriz de regressores.

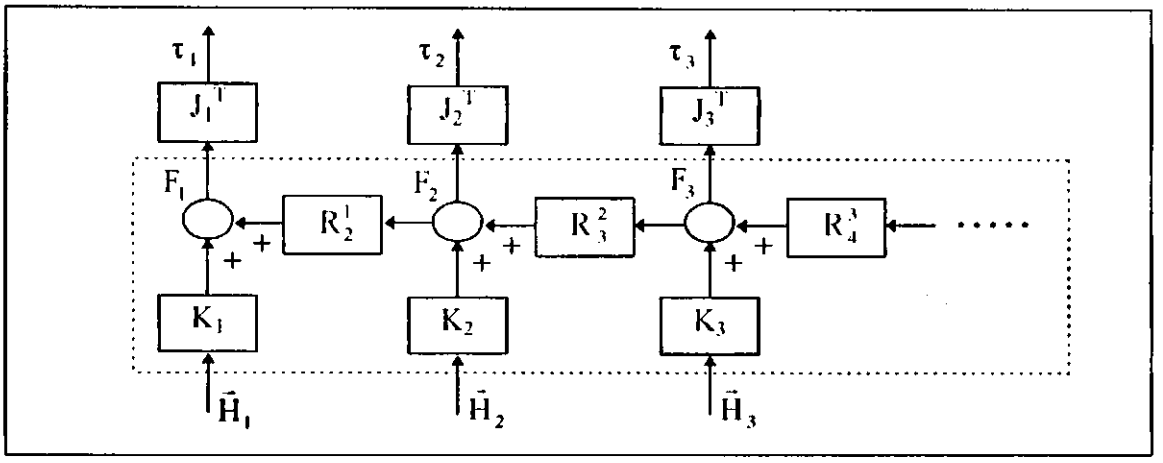

Fig. 4.2 - Diagrama de Blocos: Matriz de Regressores e Dinâmica Inversa.

Inicialmente, nota-se que matriz $[J^T R]$ pode ser particionada naturalmente em NxN submatrizes 1x6 do tipo $J_i^T R_j^i$:

$$J^T R = \begin{bmatrix} J_1^T & J_1^T R_2^1 & .... & J_1^T R_N^1 \\ 0 & J_2^T & .... & J_2^T R_N^2 \\ .... & .... & .... & .... \\ 0 & 0 & .... & J_N^T \end{bmatrix} \tag{4.22}$$

É interessante observar que, no diagrama de blocos da Figura 4.2, a submatriz 1x6 $J_i^T R_j^i$ é representada pela matriz de transferência $[\tau, F]_{ij}$, associada ao caminho $F_j \rightarrow \tau_i$. Pode-se notar no diagrama de blocos da Figura 4.2 que a matriz de transferência $[\tau, F]_{ij+1}$, associada ao caminho $F_{j+1} \rightarrow \tau_i$, é igual à matriz de transferencia $[\tau, F]_{ij}$, associada ao caminho $F_j \rightarrow \tau_i$, multiplicada pela matriz de rotação espacial R_{i+1}^i . Note-se ainda que, de acordo com a Equação (4.18), a matriz de regressores Φ é naturalmente particionada em submatrizes 1x10 do tipo $J_i^T R_j^i K_j$, que, no diagrama de blocos da Figura 4.2, correspondem às matrizes de transferência $[\tau, \dot{H}]_{ij}$ associadas aos caminhos $\dot{H}_j \rightarrow \tau_i$. Para denotar estas submatrizes, será adotada a notação especial $[\Phi]_{[i,j]}$, que correspondem à partição i, j da matriz de regressores Φ, e equivale a $[\tau, \dot{H}]_{ij}$. Nota-se que, no diagrama de blocos da Figura 4.2, a submatriz $[\Phi]_{[i,j]}$

é igual à matriz de transferência $[\tau,F]_{ij}$ vezes a matriz K_j. Levando em conta estas considerações, a seguinte recursão para o cálculo da matriz de regressores é proposta:

Recursão para a Matriz de Regressores:

Faça i variar de 1 ate N

Inicialização: $[\tau,F]_{ii} = J_i^T$, $[\Phi]_{[i,i]} = [\tau,F]_{ii}.K_i$

Se (i<N) Faça j variar de i até N-1

$$[\tau,F]_{ij+1} = [\tau,F]_{ij}.R_{j+1}^{j} \tag{4.23}$$

$$[\Phi]_{[i,j+1]} = [\tau,F]_{ij+1}.K_{j+1} \tag{4.24}$$

Fim de j

Fim de i

Observe-se que a recursão é inicializada a partir das submatrizes da diagonal e que as submatrizes abaixo da diagonal são nulas, sendo desnecessário o seu cálculo. As matrizes de regressores de elo, K_j, devem ser obtidas previamente, por meio da recursão Newton-Euler "Para Frente" e da Equação (4.10). A Tabela 4.1 mostra o esforço computacional envolvido no cálculo da matriz de regressores através do método acima. Para um manipulador de seis juntas, a computação da matriz de regressores por meio do método proposto envolve 1281 produtos e 1005 somas.

Tabela 4.1 - Esforço Computacional - Matriz de Regressores.

Termo	Produtos	Somas
K_i	86.N	75.N
$[\tau,F]_{ij}$	$(27.N^2 - 81.N + 54)/2$	$(21.N^2 - 63.N + 42)/2$
$\Phi_{[i,j]}$	$(33.N^2 - 33.N)/2$	$(23.N^2 - 23.N)/2$
Total	$30.N^2 + 29.N + 27$	$22.N^2 + 32.N + 21$

4.5 Controlador Adaptativo

Um grande número de novas estratégias de controle adaptativo de manipuladores foram desenvolvidas depois que a formulação Linear-Em-Parâmetros foi introduzida. Estas estratégias são baseadas em compensação de modelo dinâmico mais uma realimentação proporcional-derivativa τ_{pd}. De acordo com os sinais de compensação utilizados, as estratégias de controle adaptativo baseado em modelo para manipuladores robóticos podem ser classificadas em três tipos básicos [Kim & Hori, 1995]:

- Controlador Adaptativo Torque Computado [Craig, et alii, 1986]:

$$\tau = \Phi(q,\dot{q},\dot{q}^*,\ddot{q}^*)\hat{\bar{H}} + \tau_{pd} \tag{4.25}$$

- Controlador Adaptativo Entrada de Comando [Whitcomb, et alii, 1993]:

$$\tau = \Phi(q^*,\dot{q}^*,\dot{q}^*,\ddot{q}^*)\hat{\bar{H}} + \tau_{pd} \tag{4.26}$$

- Controlador Adaptativo Trajetória de Referência [Slotine & Li, 1988]:

$$\tau = \Phi(q,\dot{q},\dot{q}_r,\ddot{q}_r)\hat{\bar{H}} + \tau_{pd} \tag{4.27}$$

Nos controladores acima, $\hat{\bar{H}}$ denota o vetor de parâmetros estimados. O vetor q^* é a trajetória de junta desejada. A trajetória de referência q_r obedece às relações:

$$\dot{q}_r = \dot{q}^* + \Lambda(q^* - q) \tag{4.28.a}$$

$$\ddot{q}_r = \ddot{q}^* + \Lambda(\dot{q}^* - \dot{q}) \tag{4.28.b}$$

Onde Λ é uma matriz de ganhos positiva definida. A seguir, o Controlador Adaptativo Trajetória de Referência é detalhado. Este controlador adaptativo garante convergência global para o erro de rastreamento de posição e velocidade. O controlador adaptativo consiste de uma lei de controle, baseada em compensação de modelo dinâmico e realimentação proporcional-derivativa, associada a uma lei de adaptação de parâmetros, como é mostrado abaixo:

$$\tau = \Phi_r.\hat{H} + K_d.s \tag{4.29}$$

$$\dot{\hat{H}} = \Gamma.\Phi_r^T.s \tag{4.30}$$

onde Φ_r denota a matriz de regressores de referência $\Phi_r = \Phi(q, \dot{q}, \dot{q}_r, \ddot{q}_r)$. A matriz de ganhos de adaptação Γ 10Nx10N e a matriz de ganhos de realimentação K_d NxN são matrizes simétricas e positivas definidas. A superfície de erro do manipulador é dada pelo vetor Nx1:

$$s = (\dot{q}^* - \dot{q}) + \Lambda(q^* - q) = \dot{q}_r - \dot{q} \tag{4.31}$$

A Figura 4.3 mostra o Controlador Adaptativo Trajetória de Referência.

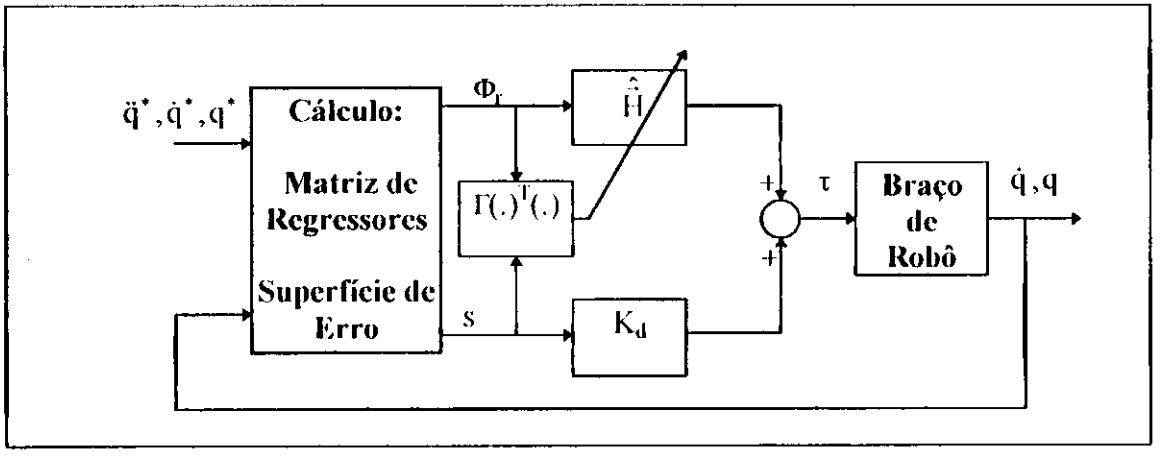

Fig. 4.3 - Controlador Adaptativo Trajetória de Referência.

4.6 Prova de Estabilidade

A seguir, é desenvolvida a prova de estabilidade, segundo Lyapunov, para o Controlador Adaptativo Trajetória de Referência. Inicialmente, a seguinte fatoração para o vetor de reações coriolis/centrífugas será introduzida:

$$\tau_c(q,\dot{q}) = C(q,\dot{q}).\dot{q} \tag{4.32}$$

onde a matriz NxN de termos em velocidade $C(q,\dot{q})$ se relaciona com a matriz de inércia através da seguinte propriedade [Slotine & Li, 1988]:

$$s^T.\dot{M}.s = 2.s^T.C.s \tag{4.33}$$

Usando a fatoração da Equação (4.32), a equação de dinâmica inversa pode ser escrita:

$$\tau = M(q).\ddot{q} + C(q,\dot{q}).\dot{q} + \tau_g(q) = \Phi.\vec{H} \tag{4.34}$$

Da mesma maneira, a Equação (4.29) do controlador pode ser rescrita:

$$\tau = \hat{M}(q).\ddot{q}_r + \hat{C}(q,\dot{q}).\dot{q}_r + \hat{\tau}_g(q) + K_d.s = \Phi_r.\hat{\vec{H}} + K_d.s \tag{4.35}$$

onde ($\hat{}$) denota termos estimados. Define-se o erro de estimação de parâmetros como:

$$\tilde{\vec{H}} = \hat{\vec{H}} - \vec{H} \tag{4.36}$$

De acordo com as definições acima, a prova de estabilidade é detalhada a seguir:

Prova de Estabilidade [Slotine & Li, 1988]*:*

Definindo a seguinte função candidata a função de Lyapunov:

$$L = (s^T.M.s + \tilde{\tilde{H}}^T.\Gamma^{-1}.\tilde{\tilde{H}})/2 \tag{4.37}$$

onde a matriz Γ é simétrica e positiva definida. Como a matriz de inércia M é simétrica e positiva definida, a função L é positiva definida. Derivando,

$$\dot{L} = s^T.M.\dot{s} + s^T.\dot{M}.s/2 + \tilde{\tilde{H}}^T.\Gamma^{-1}.\dot{\tilde{\tilde{H}}} \tag{4.38}$$

substituindo a Equação (4.33) na Equação (4.38),

$$\dot{L} = s^T.M.\dot{s} + s^T.C.s + \tilde{\tilde{H}}^T.\Gamma^{-1}.\dot{\tilde{\tilde{H}}} \tag{4.39}$$

Substituindo a derivada de superfície de erro (ver Equação (4.31)) e colocando s^T em evidência na equação acima, obtém-se:

$$\dot{L} = s^T(M.\ddot{q}_r - M.\ddot{q} + C.s) + \tilde{\tilde{H}}^T.\Gamma^{-1}.\dot{\tilde{\tilde{H}}} \tag{4.40}$$

Mas o termo de reações inerciais $M.\ddot{q}$ pode ser obtido diretamente da Equação (4.34) de dinâmica inversa: $M.\ddot{q} = \tau - C.\dot{q} - \tau_g$. Assim,

$$\dot{L} = s^T(M.\ddot{q}_r - \tau + C.\dot{q} + \tau_g + C.s) + \tilde{\tilde{H}}^T.\Gamma^{-1}.\dot{\tilde{\tilde{H}}} \tag{4.41}$$

Substituindo s na equação acima, de acordo com a Equação (4.31):

$$\dot{L} = s^T(M.\ddot{q}_r + C.\dot{q}_r + \tau_g - \tau) + \tilde{\tilde{H}}^T.\Gamma^{-1}.\dot{\tilde{\tilde{H}}} \tag{4.42}$$

Substituindo a Equação (4.35) do controlador de maneira a eliminar o vetor de esforços de acionamento τ da equação acima, obtém-se:

$$\dot{L} = s^T(M.\ddot{q}_r + C.\dot{q}_r + \tau_g - \hat{M}.\ddot{q}_r - \hat{C}\dot{q}_r - \hat{\tau}_g - K_d.s) + \tilde{\vec{H}}^T.\Gamma^{-1}.\dot{\tilde{\vec{H}}} \tag{4.43}$$

Reagrupando:

$$\dot{L} = -s^T.K_d.s + [\tilde{\vec{H}}^T.\Gamma^{-1}.\dot{\tilde{\vec{H}}} - s^T((\hat{M}-M)\ddot{q}_r + (\hat{C}-C)\dot{q}_r + (\hat{\tau}_g - \tau_g))] \tag{4.44}$$

Utilizando a propriedade de linearidade paramétrica,

$$\dot{L} = -s^T.K_d.s + [\tilde{\vec{H}}^T.\Gamma^{-1}.\dot{\tilde{\vec{H}}} - s^T.\Phi_r.\tilde{\vec{H}}] \tag{4.45}$$

Igualando a zero o termo entre colchetes na Equação acima, a lei de adaptação é obtida:

$$\tilde{\vec{H}}^T.\Gamma^{-1}.\dot{\tilde{\vec{H}}} = s^T.\Phi_r.\tilde{\vec{H}} = \tilde{\vec{H}}^T.\Phi_r^T.s \quad \Rightarrow \quad \dot{\tilde{\vec{H}}} = \Gamma.\Phi_r^T.s \tag{4.46}$$

O uso desta lei de adaptação paramétrica força a zero o termo entre colchetes na Equação (4.45). Assim,

$$\dot{L} = -s^T.K_d.s \leq 0 \tag{4.47}$$

Assim a função $\dot{L}$ é negativa definida. Como a função L é positiva definida, então L é uma função de Lyapunov e o sistema é assintoticamente estável.

4.7 Controlador Adaptativo Modular

Nesta Seção, o controlador adaptativo descrito na Seção 4.5 é equacionado em forma recursiva, permitindo o tratamento modular do problema de controle. Desde que o controlador adaptativo precisa do cálculo da dinâmica inversa para compensação do modelo dinâmico, o método Newton-Euler pode ser utilizado para computar os esforços de acionamento de maneira recursiva. Uma modificação deve ser feita, porém, no cálculo dos termos coriolis/centrífugos, visto que na matriz de regressores aparecem produtos de velocidade $\dot{q}_{r_i}.\dot{q}_j$. Inicialmente, define-se o vetor V_{r_i} 6x1 de velocidade espacial de referência do elo i, composto pelo vetor v_{r_i} 3x1 de velocidade linear de referência do elo i e pelo vetor ω_{r_i} 3x1 de velocidade angular de referência do elo i:

$$V_{r_i} = \begin{bmatrix} v_{r_i} \\ \omega_{r_i} \end{bmatrix} \tag{4.48}$$

Define-se também o vetor Ω_{r_i} 6x1 de velocidade angular espacial de referência do elo i:

$$\Omega_{r_i} = \begin{bmatrix} 0 \\ \omega_{r_i} \end{bmatrix} \tag{4.49}$$

Impõe-se que a matriz K_d seja diagonal, composta dos elementos diagonais K_{d_i} (K_d=diag(K_{d1}, K_{d2}, ... , K_{dN})). Desta maneira, garante-se uma realimentação PD independente para cada junta, o que possibilita a divisão da malha de realimentação em módulos de controle separados. Denotando ainda, K_{r_i}, a matriz de regressores de referência do elo i, a seguinte versão modificada para o método Newton-Euler é proposta, visando sua aplicação no controlador adaptativo:

- Recursão para Frente:

Inicialização: $V_0 = V_{r0} = 0$ $\quad \dot{V}_{r0}$ = aceleração da gravidade.

Faça i variar de 0 até N-1

$$V_{i+1} = R_{i+1}^{iT}.V_i + J_{i+1}\dot{q}_{i+1} \tag{4.50}$$

$$V_{r_{i+1}} = R_{i+1}^{iT}.V_{r_i} + J_{i+1}\dot{q}_{r_{i+1}} \tag{4.51}$$

$$\dot{V}_{r_{i+1}} = R_{i+1}^{iT}.\dot{V}_{r_i} + \dot{V}_{cr_i} + J_{i+1}\ddot{q}_{r_{i+1}} \tag{4.52}$$

onde o Vetor $\dot{V}_{cr_i}$ 6x1 de aceleração espacial coriolis/centrífuga de referência do elo i é:

$$\dot{V}_{cr_i} = -[(\Omega_i\times)(R_{i+1}^{iT})^{-1} + \dot{q}_{i+1}(J_{i+1}\times)](R_{i+1}^{i})^{-1}\Omega_{r_i} \tag{4.53}$$

$$K_{ri} = \left[\begin{array}{c|c} \dot{v}_{r_i} & (\dot{\omega}_{r_i}\times)+(\omega_{r_i}\times)(\omega_i\times) \\ \hline 0 & (-\dot{v}_{r_i}\times) \end{array} \middle| \begin{array}{c} \mathbf{0} \\ \hline (\bullet\dot{\omega}_{r_i})+(\omega_{r_i}\times)(\bullet\omega_i) \end{array}\right] \tag{4.54}$$

Fim de i

- Recursão para Trás:

Inicialização: F_{N+1} = Esforço espacial aplicado na garra.

Faça i variar de N até 1

$$F_i = R_{i+1}^{i}F_{i+1} + K_{r_i}.\tilde{\tilde{H}}_i \tag{4.55}$$

$$\tau_i = J_i^T.F_i + K_{d_i}s_i \tag{4.56}$$

Fim de i

As equações acima permitem a implementação do controlador em módulos de controle, como é mostrado na Figura 4.4. A cada junta é associado um módulo de controle e os módulos são concatenados seqüencialmente. A entrada de cada módulo é representada pelos termos K_{r_i} e s_i, dependentes da trajetória de referência. A saída de cada módulo é o esforço de acionamento τ_i da junta i. Note-se que os vetores de esforços espaciais são propagados sucessivamente através dos módulos.

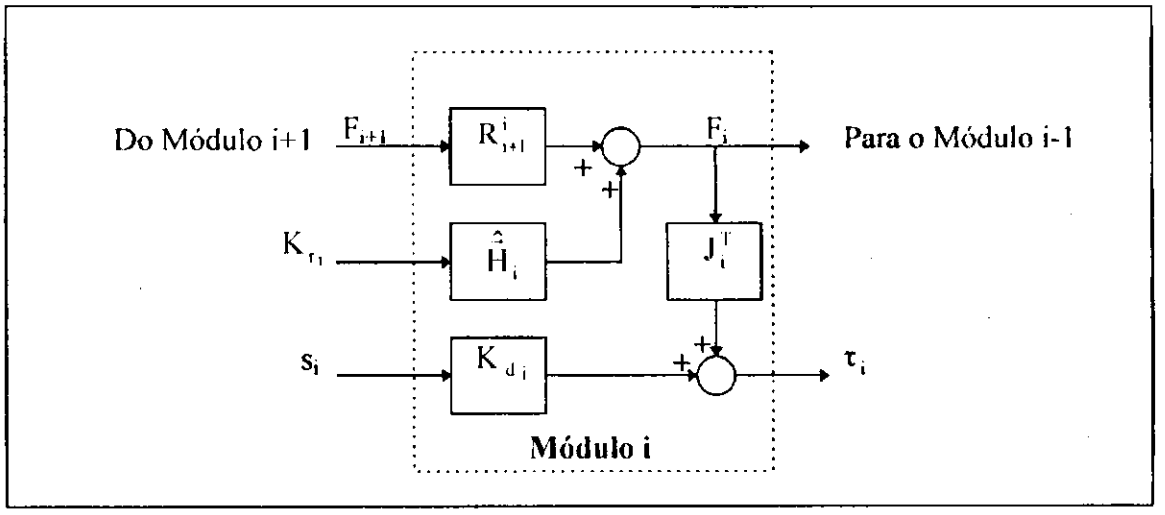

Fig. 4.4 - Módulo de Controle da Junta i.

A seguir o esquema de adaptação de parâmetros dado na Equação (4.35) é desenvolvido em forma recursiva, de maneira a permitir a sua implementação em módulos de adaptação padronizados para os parâmetros de cada elo. Sem perda de generalidade, a matriz Γ é assumida diagonal, e particionada em submatrizes 10x10, das quais, aquelas associadas à diagonal, denotadas por Γ_1 , ... , Γ_N, serão denominadas matrizes de ganho de adaptação dos elos correspondentes:

$$\Gamma = \begin{bmatrix} \Gamma_1 & 0 & \cdots & 0 \\ 0 & \Gamma_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \Gamma_N \end{bmatrix} \tag{4.57}$$

Então, de acordo com a Equação (4.18), a Equação (4.30) de adaptação paramétrica pode ser expandida nas seguintes equações individuais para cada elo:

$$\dot{\tilde{H}}_1 = \Gamma_1 K_{r_1}^T (J_1 s_1) \tag{4.58.a}$$

$$\dot{\tilde{H}}_2 = \Gamma_2 K_{r_2}^T (R_2^{1T} J_1 s_1 + J_2 s_2) \tag{4.58.b}$$

$$\dot{\tilde{H}}_3 = \Gamma_3 K_{r_3}^T (R_3^{1T} J_1 s_1 + R_3^{2T} J_2 s_2 + J_3 s_3) \tag{4.58.c}$$

.....

Verifica-se que os termos entre parêntese nas equações acima podem ser obtidos recursivamente. Denominando cada um destes termos como erro de velocidade espacial ΔV_i do elo i , a seguinte recursão é proposta:

Recursão para Adaptação de Parâmetros:

Inicialização: $\Delta V_0 = 0$

Faça i variar de 1 até N

$$\Delta V_i = R_{i-1}^i . \Delta V_{i-1} + J_i s_i \tag{4.59}$$

$$\dot{\tilde{H}}_i = \Gamma_i . K_{r_i}^T . \Delta V_i \tag{4.60}$$

Fim de i

Os parâmetros dinâmicos do elo i podem ser estimados integrando a Equação (4.60). As equações acima permitem a implementação de módulos para a adaptação dos parâmetros dinâmicos dos elos, como é mostrado na Figura 4.5. A cada elo é associado um módulo de adaptação e os módulos são concatenados seqüencialmente. A entrada de cada módulo é representada pelos termos K_{r_i} e s_i, dependentes da trajetória de referência. A saída de cada módulo é o vetor de

parâmetros estimados do elo i. Nota-se que os vetores de erro de velocidade espacial são propagados sucessivamente através dos módulos.

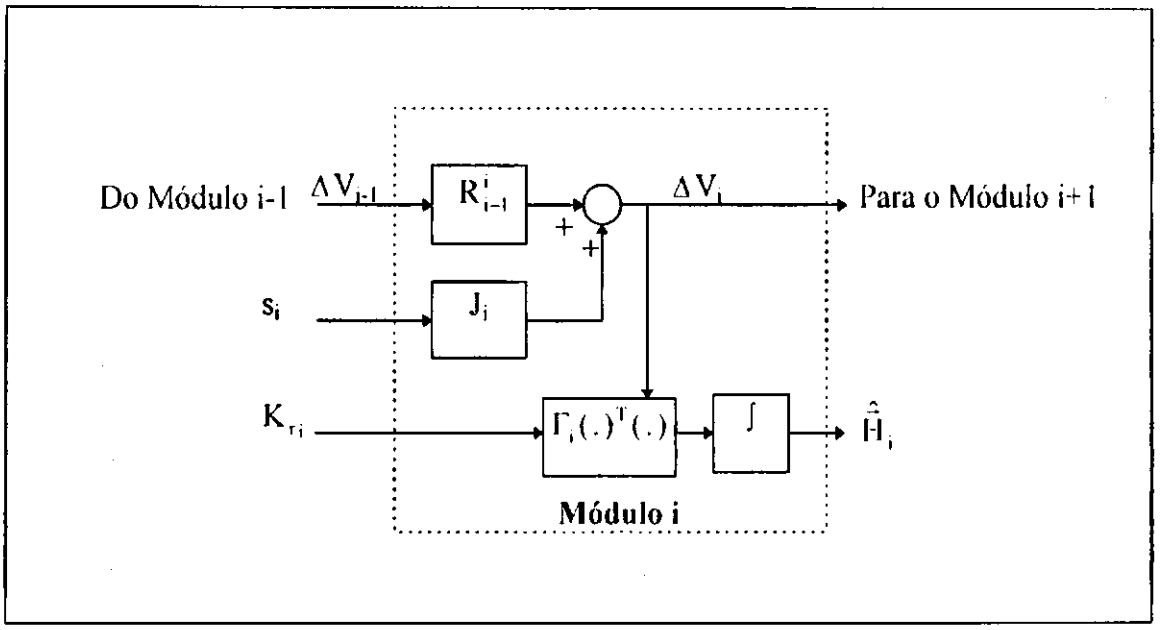

Fig. 4.5 - Módulo de Adaptação de Parâmetros do Elo i.

O tratamento modular do controle adaptativo de manipuladores permite subdividir o problema em tarefas menores de controle e adaptação das juntas e elos individuais por meio de módulos com entradas e saídas padronizadas. Esta padronização de entradas e saídas sugere a sua implementação em sistema de multiprocessadores, onde os módulos de controle e adaptação de cada elo seriam realizados por meio de um processador dedicado associado ao elo correspondente.

4.8 Resultados de Simulação

Nesta Seção, são apresentados resultados de simulação computacional de controle de um manipulador de três juntas utilizando o esquema de controle adaptativo proposto. Os parâmetros cinemáticos (na convenção Denavit-Hartenberg) do manipulador de três juntas simulado são descritos na Tabela 4.2. Os parâmetros dinâmicos do mesmo manipulador são:

$$\vec{H}_1 = [3.000 \quad 0.000 \quad 0.000 \quad -0.750 \quad 0.125 \quad 0.125 \quad 0.000 \quad 0.000 \quad 0.000 \quad 0.000]^T \quad (4.61.a)$$

$$\vec{H}_2 = [1.800 \quad 0.360 \quad 0.000 \quad 0.000 \quad 0.000 \quad 0.096 \quad 0.096 \quad 0.000 \quad 0.000 \quad 0.000]^T \quad (4.61.b)$$

$$\vec{H}_3 = [1.500 \quad 0.225 \quad 0.000 \quad 0.000 \quad 0.000 \quad 0.045 \quad 0.045 \quad 0.000 \quad 0.000 \quad 0.000]^T \quad (4.61.c)$$

Tabela 4.2 - Parâmetros Cinemáticos - Manipulador de Três Juntas.

θ_i	d_i (m)	a_i (m)	α_i (rad)
q_1	0.50	0	0
q_2	**0**	**0**	$\pi/2$
q_3	0	0.40	0

Os ganhos do controlador, foram escolhidos por tentativa e erro, obtendo-se os seguintes valores para todas as juntas: $K_d = 4$, $\Lambda = 4$. As matrizes diagonais de ganhos de adaptação de parâmetros foram escolhidas para todos os elos iguais à matriz identidade vezes o valor: $\Gamma = 0.5$.

Inicialmente, simulou-se o rastreamento de trajetória de junta, onde a trajetória para todas as juntas foi gerada através de um polinômio interpolador de 5ª ordem, variando de zero a $\pi/4$ rad, com velocidades iniciais e finais nulas. O valor inicial para todos os parâmetros dinâmicos estimados foi zero, ou seja, assumiu-se que nenhum conhecimento a respeito dos mesmos era disponível. A Figura 4.6 mostra o rastreamento de trajetória de junta para as condições mencionadas. A convergência dos parâmetros estimados para o mesmo caso é mostrada na Figura 4.7. Pode-se observar que no começo da trajetória o erro de rastreamento é considerável, devido à pobre estimativa inicial assumida para os parâmetros dinâmicos. Após o trecho inicial, os parâmetros convergem para valores constantes e o erro de rastreamento converge para valores cada vez menores.

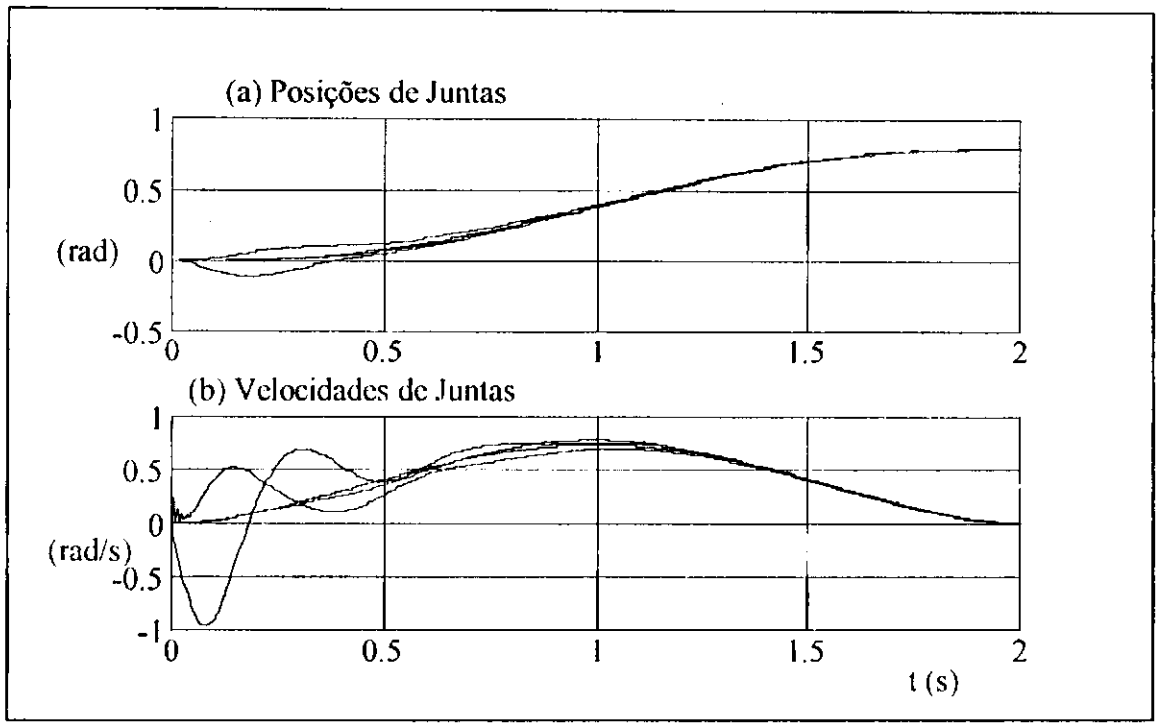

Fig. 4.6 - Rastreamento de Trajetória de Junta - Estimativa Inicial Nula.

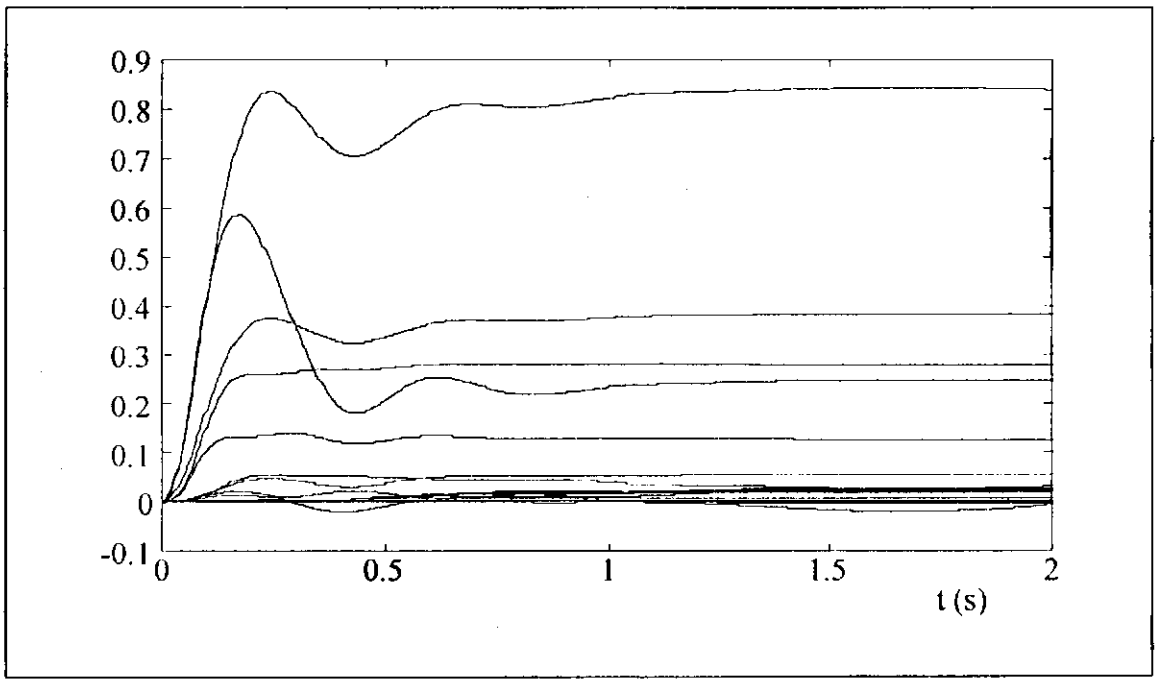

Fig. 4.7 - Convergência dos Parâmetros Adaptados.

A seguir, realizou-se a mesma simulação utilizando porém, uma estimativa inicial de parâmetros dinâmicos contaminada com ruído de amplitude igual a 50 % do valor nominal dos mesmos. A Figura 4.8 mostra os resultados de rastreamento de trajetória para esta situação. Verifica-se que o erro de rastreamento no trecho inicial da trajetória é bem menor do que no caso anterior.

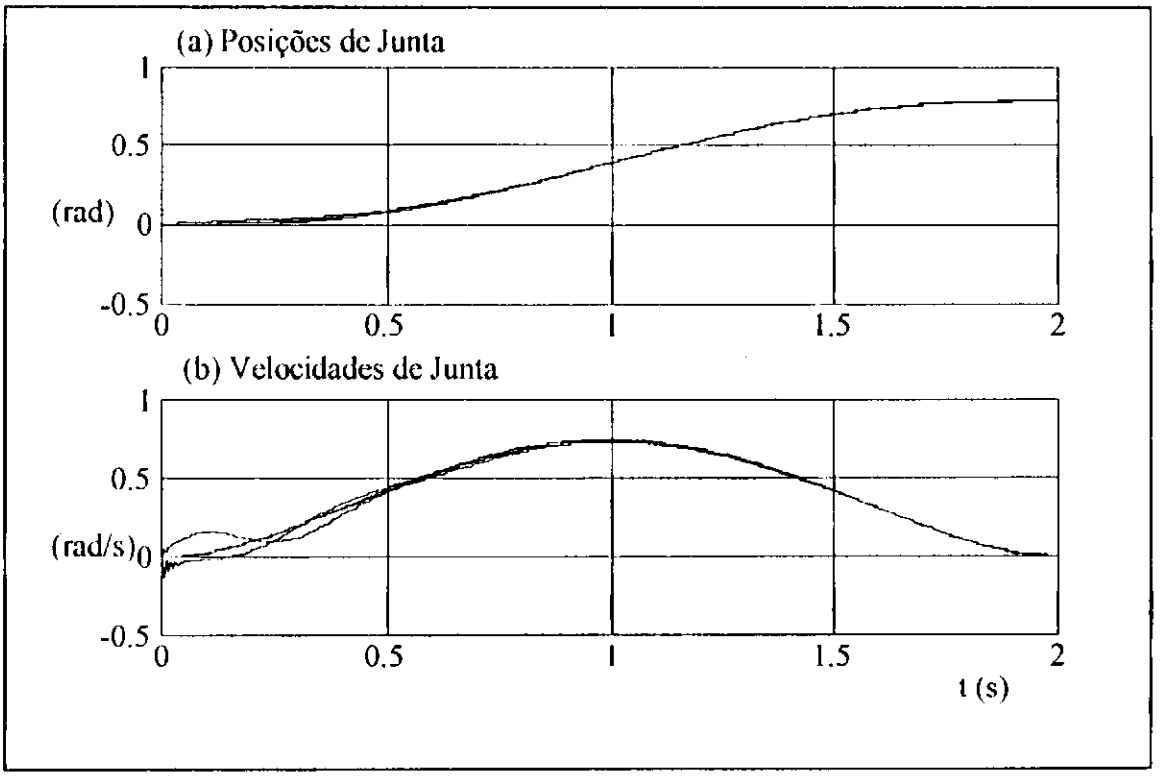

Fig. 4.8 - Rastreamento de Trajetória de Junta - 50 % de Erro Inicial.

Simulou-se o rastreamento de uma trajetória circular em espaço cartesiano, utilizando uma estimativa de parâmetros inicial com 20 % de erro. A Figura 4.9 mostra o rastreamento de trajetória de juntas para esta simulação. O rastreamento da trajetória circular no espaço cartesiano é mostrado na Figura 4.10, onde o manipulador de três juntas é representado por diagrama unifilar. Verifica-se que a garra segue acuradamente a trajetória estabelecida. O erro inicial é bastante pequeno, devido à pequena incerteza paramétrica inicial.

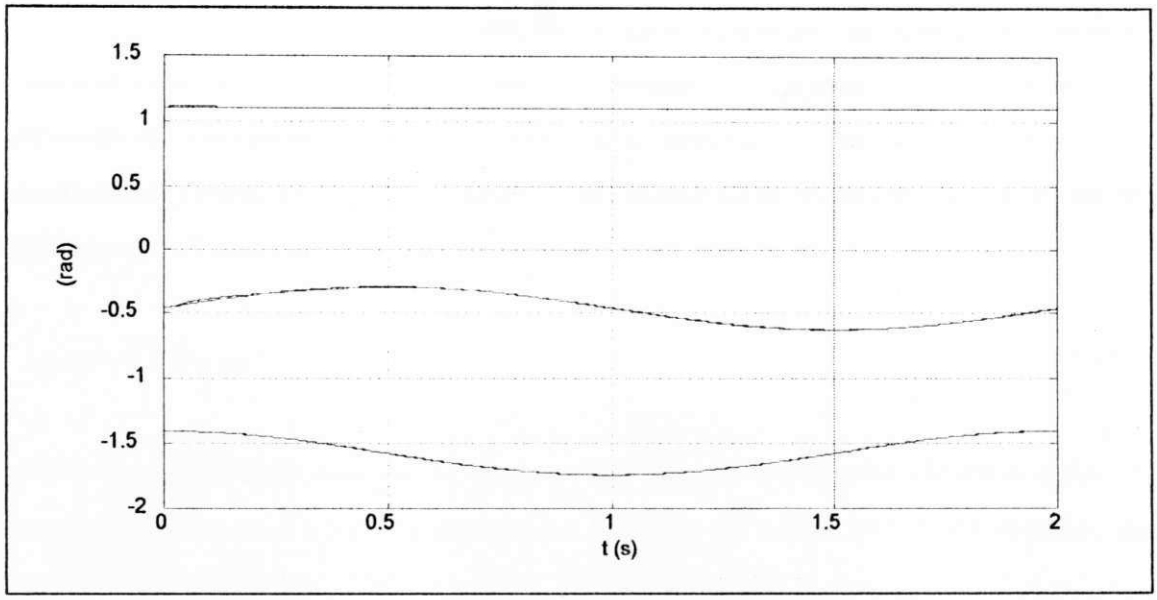

Fig. 4.9 - Rastreamento de Círculo no Espaço 3D - Posições de Juntas.

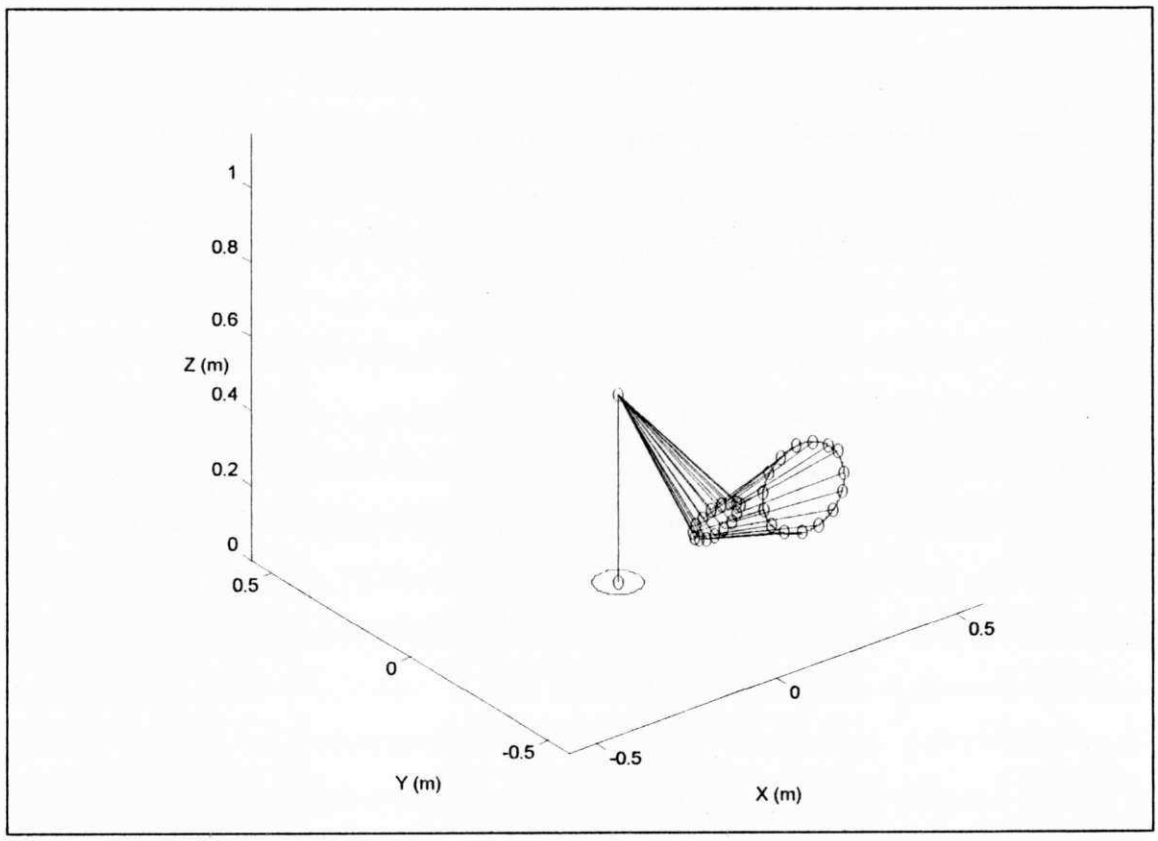

Fig. 4.10 - Rastreamento de Círculo no Espaço 3D - Diagrama Unifilar.

A seguir, por meio de simulação dinâmica do mesmo manipulador de três juntas descrito anteriormente, realizou-se uma análise de desempenho comparativa, entre o Controlador Adaptativo Modular proposto e um controlador Torque Computado padrão. Dado o erro de rastreamento de trajetórias de junta,

$$\Delta q(t) = q^*(t) - q(t) \tag{4.62}$$

Adota-se, como uma medida do desempenho de um controlador de robô, para o rastreamento de uma trajetória de teste no intervalo de tempo ($0 \rightarrow t$), o índice de desempenho quadrático:

$$J_q(t) = \int_0^t \left(\sqrt{[\Delta q(t)]^T . [\Delta q(t)]} \right) dt \tag{4.63}$$

Como trajetória de teste adotou-se, para todas as juntas, uma trajetória senoidal, de amplitude $\pi/4$ rad e freqüência 0.5 Hz. Para análise comparativa, adotou-se um controlador Torque Computado com 50 % de erro nos parâmetros do modelo dinâmico, ganhos proporcionais e integrais iguais a $K_P = 2500$ e $K_D = 100$ para todas as juntas. Para a trajetória de teste descrita, a Figura 4.11 mostra a evolução do índice de desempenho do controlador Torque Computado e do Controlador Adaptativo Modular Proposto. Para o controlador adaptativo, duas condições foram estudadas: a) parâmetros do modelo dinâmico com erro inicial de 50 % (igual ao controlador Torque Computado); b) parâmetros dinâmicos iniciais nulos (ou seja: nenhum conhecimento prévio a respeito dos mesmos). Na Figura 4.11, pode-se verificar que, para o controlador Torque Computado, o índice de desempenho descreve aproximadamente uma reta em função do tempo. O fato da declividade permanecer aproximadamente constante indica que, para o controlador Torque Computado, o erro de rastreamento de trajetória se mantém relativamente no mesmo nível médio à medida que a trajetória se repete. Por outro lado, para o Controlador Adaptativo modular, o índice de desempenho apresenta uma declividade que diminui gradativamente, o que significa que o erro de rastreamento de trajetória converge para valores cada vez menores.

A partir da Figura 4.11, verifica-se que, mesmo para o caso de uma grande incerteza nos parâmetros dinâmicos, o erro de rastreamento de trajetória rapidamente converge para valores pequenos, embora que o erro inicial seja grande. Assim, embora que o Controlador Adaptativo apresente um erro maior durante o início da adaptação, após um certo tempo supera o desempenho do controlador Torque Computado, pois este não melhora com o tempo.

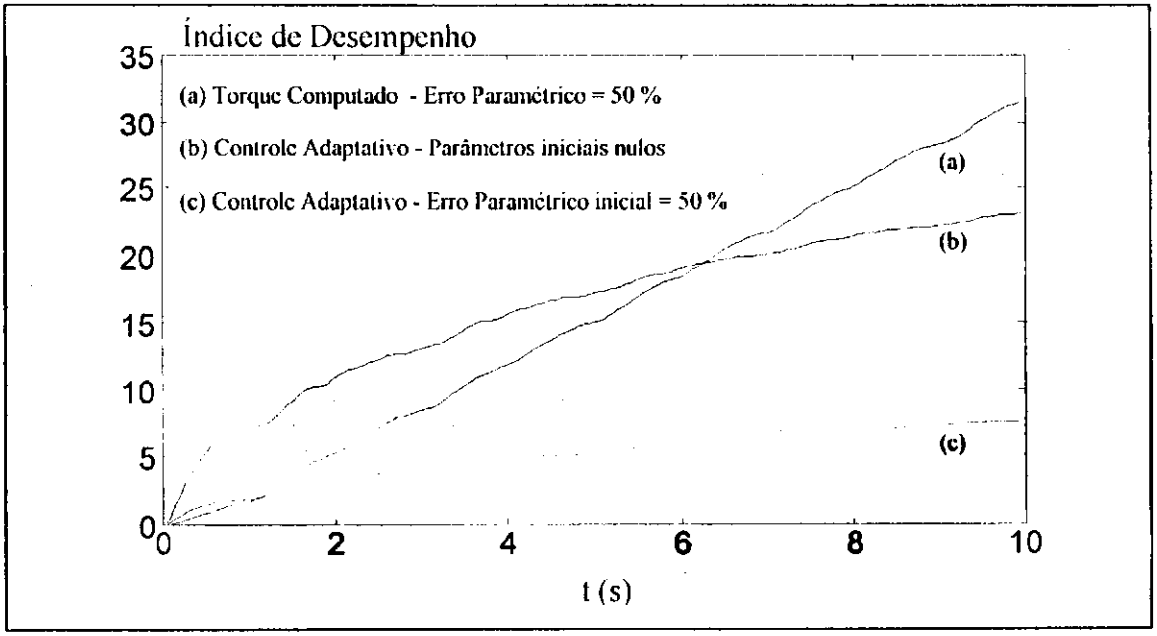

Fig. 4.11 - Análise de Desempenho - Controlador Adaptativo Modular.

4.9 Conclusão

Neste Capítulo foi desenvolvida uma forma Linear-Em-Parâmetros para a equação dinâmica de manipuladores e um método recursivo para cálculo da matriz de regressores. Um esquema modular para controle adaptativo de manipuladores foi proposto. Módulos padronizados para controle das juntas e adaptação dos parâmetros dos elos foram sugeridos. Resultados de simulação atestaram o bom desempenho do controlador adaptativo proposto, o qual mostrou-se superior ao controlador Torque Computado quando utiliza-se, em ambos, o mesmo nível de conhecimento a respeito do modelo dinâmico do manipulador.

CAPÍTULO 5

NEUROCONTROLADOR MODULAR

5.1 Introdução

Neste Capítulo, o Neurocontrolador Modular para manipuladores robóticos é apresentado. Na Seção 5.2 uma forma Linear em Parâmetros Cinemáticos é desenvolvida para o algoritmo Newton-Euler. Com base nesta formulação específica, Módulos Neurais padronizados para controle de manipuladores são desenvolvidos na Seção 5.3. Na Seção 5.4, é proposto um esquema de treinamento inverso direto para os módulos neurais. A partir das técnicas de controle adaptativo descritas no Capítulo 4, um esquema de treinamento adaptativo direto para o neurocontrolador modular é desenvolvido na Seção 5.5. Resultados de simulação do neurocontrolador proposto para um manipulador de três juntas são apresentados na Seção 5.6.

5.2 Modelo Linear em Termos Cinemáticos

Nesta Seção, o método Newton-Euler é reformulado numa forma Linear em Termos Cinemáticos. Definindo o operador matriz 3x6 produto vetorial-escalar sobre a matriz h 3x3:

$$(\times h\bullet)=\begin{bmatrix} 0 & h_{23} & -h_{32} & (h_{33}-h_{22}) & -h_{12} & h_{13} \\ -h_{13} & 0 & h_{31} & h_{21} & (h_{11}-h_{33}) & -h_{23} \\ h_{12} & -h_{21} & 0 & -h_{31} & h_{32} & (h_{22}-h_{11}) \end{bmatrix} \qquad (5.1)$$

Utilizando esta nomenclatura, o vetor 6x1 de esforço espacial coriolis/centrífugo do elo i pode ser expresso por:

$$F_{c_i}=(\Omega_i\times)H_i\Omega_i=\begin{bmatrix} m_i(\times[p_{c_i}\times]\bullet) \\ (\times h_i\bullet) \end{bmatrix}\vec{\omega}_i^2 \qquad (5.2)$$

onde o vetor 6x1 de produtos de velocidades é dado por:

$$\vec{\omega}_i^2 = \left[\omega_{x_i}^2 \quad \omega_{y_i}^2 \quad \omega_{z_i}^2 \quad \omega_{y_i}.\omega_{z_i} \quad \omega_{z_i}.\omega_{x_i} \quad \omega_{x_i}.\omega_{y_i}\right]^T \tag{5.3}$$

Desta maneira, os termos da Recursão Newton-Euler para Trás dependentes dos parâmetros dinâmicos podem ser expressos como:

$$H_i \dot{V}_i + (\Omega_i \times) H_i \Omega_i = [H]_i . \vec{K}_i \tag{5.4}$$

onde a matriz 6x12 de parâmetros dinâmicos do elo i e o vetor 12x1 de termos cinemáticos do elo i são dados respectivamente por:

$$[H]_i = \begin{bmatrix} m_i.I & m_i.(p_{c_i}\times)^T & m_i(\times[p_{c_i}]\bullet) \\ m_i.(p_{c_i}\times) & h_i & (\times h_i \bullet) \end{bmatrix} \tag{5.5}$$

$$\vec{K}_i = \begin{bmatrix} \dot{V}_i \\ \vec{\omega}_i^2 \end{bmatrix} \tag{5.6}$$

Assim, de acordo com a Equação (5.4), o método Newton-Euler pode ser expresso em forma Linear em Termos Cinemáticos:

- Recursão para Frente - Forma Linear em Termos Cinemáticos:

Inicialização: $V_0 = 0$ $\dot{V}_0$ = aceleração da gravidade.

Faça i variar de 0 até N-1

$$V_{i+1} = R_{i+1}^{iT}.V_i + J_{i+1}\dot{q}_{i+1} \tag{5.7}$$

$$\dot{V}_{i+1} = R_{i+1}^{iT}.\dot{V}_i - [(\Omega_i \times)(R_{i+1}^{iT})^{-1} + \dot{q}_{i+1}(J_{i+1}\times)](R_{i+1}^{i})^{-1}\Omega_i + J_{i+1}\ddot{q}_{i+1} \tag{5.8}$$

Fim de i

- Recursão para Trás:

Inicialização: F_{N+1} = Esforço espacial aplicado na garra.

Faça i variar de N até 1

$$F_i = R_{i+1}^i F_{i+1} + [H]_i . \dot{K}_i \quad (5.9)$$

$$\tau_i = J_i^T . F_i \quad (5.10)$$

Fim de i

5.3 Neurocontrolador Modular

Nesta Seção, o conceito de Módulo Neural é introduzido. A interligação dos módulos resulta num neurocontrolador modular que consiste de realimentação de Proporcional-Derivativa mais a compensação do modelo dinâmico não linear por meio das redes neurais. O Neurocontrolador Modular foi inspirado no controlador Torque Computado descrito no Capítulo 4, que também consiste de realimentação Proporcional-Derivativa associada a compensação da dinâmica não linear do manipulador, a qual é implementada, de maneira diversa, por meio de um modelo dinâmico adequado, cujos parâmetros devem ser conhecidos com precisão. O vetor Nx1 de acelerações de referência de junta é definido como:

$$\ddot{q}_r = \ddot{q}^* + K_D(\dot{q}^* - \dot{q}) + K_P(q^* - q) \quad (5.11)$$

onde q^* é o vetor Nx1 de posições de junta desejadas. K_D e K_P são matrizes diagonais NxN de ganhos derivativos e proporcionais respectivamente. No controlador Torque Computado os esforços de acionamento podem ser calculados a partir da aceleração de referência $\ddot{q}_r$, por meio do Método Newton-Euler. A seguir, mostra-se como os esforços de acionamento podem ser calculados para implementar o controlador Torque

Computado, através da versão Linear em Termos Cinemáticos do algoritmo Newton-Euler. Esta forma particular de expressar o método Torque Computado é obtida a partir dos vetores 12x1 de termos cinemáticos de referência $\vec{K}_{r_i}$ dos elos i = 1, ... , N, definidos como os vetores de termos cinemáticos computados através da Recursão Newton-Euler para Frente, substituindo as acelerações $\ddot{q}_i$ pelas acelerações de referência $\ddot{q}_{r_i}$,

- Recursão para Frente - Termos Cinemáticos de Referência:

Inicialização: $V_0 = 0$ $\dot{V}_{r0}$ = aceleração da gravidade.

Faça i variar de 0 até N-1

$$V_{i+1} = R_{i+1}^{iT}.V_i + J_{i+1}\dot{q}_{i+1} \tag{5.12}$$

$$\dot{V}_{r_{i+1}} = R_{i+1}^{iT}.\dot{V}_{r_i} - [(\Omega_i\times)(R_{i+1}^{iT})^{-1} + \dot{q}_{i+1}(J_{i+1}\times)](R_{i+1}^{i})^{-1}\Omega_i + J_{i+1}\ddot{q}_{r_{i+1}} \tag{5.13}$$

$$\vec{K}_{r_{i+1}} = \begin{bmatrix} \dot{V}_{r_{i+1}} \\ \dot{\omega}_{i+1}^2 \end{bmatrix} \tag{5.14}$$

Fim de i

- Recursão para Trás:

Inicialização: F_{N+1} = Esforço espacial aplicado na garra.

Faça i variar de N até 1

$$F_i = R_{i+1}^{i}F_{i+1} + [H]_i.\dot{K}_{r_i} \tag{5.15}$$

$$\tau_i = J_i^T.F_i \tag{5.16}$$

Fim de i

Caso os parâmetros dinâmicos disponíveis sejam iguais aos parâmetros do manipulador, o controlador Torque Computado consegue desacoplar a dinâmica do robô e as acelerações de junta do manipulador acompanham as acelerações de referência. Quando os parâmetros dinâmicos não são conhecidos com precisão, o desempenho do controlador Torque Computado se deteriora. Em lugar da recursão Newton-Euler para Trás, que depende dos parâmetros dinâmicos, propõe-se usar redes neurais padronizadas, os módulos neurais, para o cálculo dos esforços de acionamento de uma maneira seqüencial, análoga á recursão Para Trás. A cada elo i do manipulador associa-se um módulo neural i correspondente, com entradas F_{i+1} e $\vec{K}_{r_i}$, e com saídas F_i e τ_i . Os módulos devem ser concatenados seqüencialmente, partindo do último elo até a base do manipulador. Assim, a saída do módulo i+1 (o vetor F_{i+1}) serve de entrada para o módulo i, cuja saída é o vetor F_i , que por sua vez serve de entrada para o módulo i-1. Denominando $f_{d(i)}(.)$ o mapeamento direto (entrada-saída) do módulo neural i, então, o mesmo pode ser descrito pelas seguintes equações:

$$F_i = f_{d(i)}([(R^i_{i+1}.F_{i+1})^T \ (\vec{K}_{r_i})^T]^T) \tag{5.17}$$

$$\tau_i = J_i^T.F_i \tag{5.18}$$

Nota-se que estas relações são análogas a uma etapa da recursão Newton-Euler para Trás, mas dispensando o conhecimento dos parâmetros dinâmicos. O módulo neural associado ao elo i é representado esquematicamente na Figura 5.1. Também nesta Figura, o esquema da recursão Newton-Euler Para Trás é apresentado, ilustrando claramente as semelhanças entre as duas abordagens. Os módulos interligados compõem o Neurocontrolador Modular. Espera-se que, se submetido a um treinamento adequado, o Neurocontrolador Modular aprenda as relações $(F_{i+1}, \vec{K}_{r_i}) \rightarrow F_i$ e emule corretamente a recursão Newton-Euler para Trás. A Figura 5.2 mostra o diagrama de blocos do neurocontrolador modular proposto.

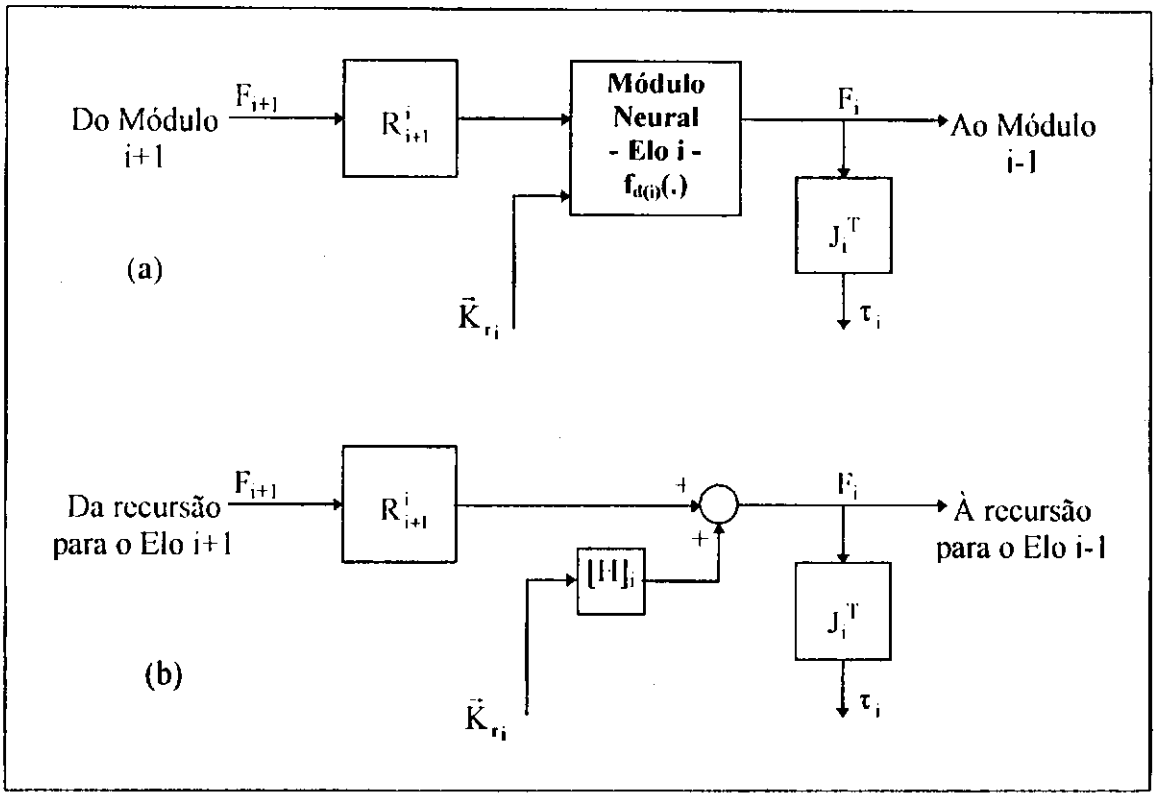

Fig. 5.1 - Módulo Neural do Elo i.

a) Módulo Neural. b) Recursão Newton-Euler Para Trás.

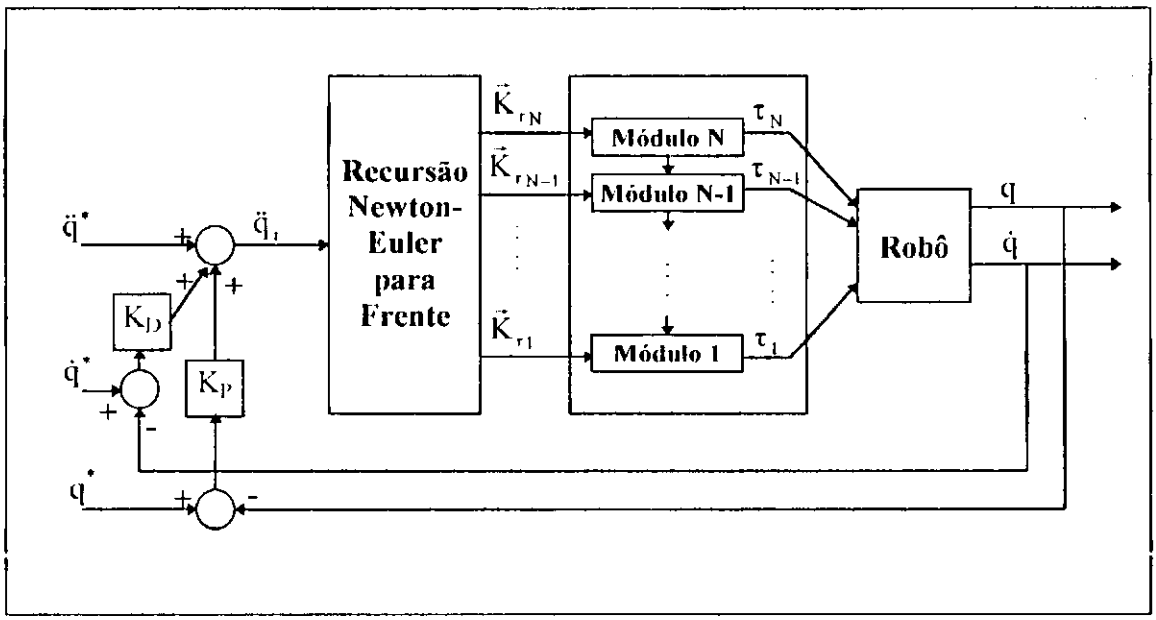

Fig. 5.2 - Diagrama de Blocos - Neurocontrolador Modular.

O conceito de módulo neural permite projetar o neurocontrolador com redes neurais padronizadas de três camadas, de pequenas dimensões, com algumas características fixas. Por exemplo, cada módulo possui um número de entradas fixas. São dezoito entradas; seis correspondentes ao vetor F_{i+1} e doze correspondentes ao vetor $\vec{K}_{ri}$. O número de saídas também está padronizado: são seis saídas correspondentes ao vetor F_i. A escolha do número de neurônios da camada oculta será discutida posteriormente. Como a cada elo é associado um módulo neural, a princípio, o esquema pode ser estendido a manipuladores com número de juntas arbitrário.

O tratamento modular do neurocontrolador para manipuladores permite subdividir o problema em tarefas menores de neurocontrole das juntas e elos individuais por meio de módulos com entradas e saídas padronizadas. Esta padronização de entradas e saídas sugere a sua implementação em sistema de multiprocessadores, onde cada módulo neural seria realizado por meio de um processador dedicado associado ao elo correspondente.

5.4 Esquema de Treinamento Inverso Direto

Cada módulo neural i é treinado por meio do algoritmo de propagação retroativa de erro a partir do vetor de erro de saída do mesmo, denotado por ΔF_i. Propõe-se obter este vetor de erro a partir da soma de duas parcelas: uma parcela de erro local $\Delta\tau_i$, correspondente a τ_i, e uma parcela obtida por propagação retroativa do erro de saída do módulo neural do elo i-1. O erro local consiste da diferença entre o esforço de acionamento τ_i, aplicado à junta i, e a saída $\hat{\tau}_i$ do módulo neural i correspondente à entrada $\vec{K}_i$ do módulo i, ou seja:

$$F_i = f_{d(\cdot)}([(R_{i+1}^i . F_{i+1})^T \ (\vec{K}_i)^T]^T) \tag{5.19}$$

$$\hat{\tau}_i = J_i^T . F_i \tag{5.20}$$

Assim,

$$\Delta\tau_i = \tau_i - \hat{\tau}_i \tag{5.21}$$

Como o erro $\Delta\tau_i$ é formulado diretamente a partir da saída $\hat{\tau}_i$ do módulo neural, o método de treinamento é caracterizado como inverso direto. Denominando $f_{r(i)}(.)$ o algoritmo de propagação retroativa do erro ΔF_i através do módulo neural i, então, a seguinte recursão é proposta para cálculo do erro de saída do módulo neural i:

$$\Delta F_i = J_i.\Delta\tau_i + (R_i^{i-1})^{-1}.f_{r(i-1)}(\Delta F_{i-1}) \tag{5.22}$$

onde a primeira parcela da soma corresponde a um erro local do módulo i e a segunda parcela é proveniente do módulo i-1. Assim, o treinamento é iniciado no módulo neural do elo 1, prosseguindo seqüencialmente, por meio de propagação retroativa de erro, até o módulo do elo N. Obviamente, o erro de saída do módulo neural 1 deve ser inicializado como:

$$\Delta F_1 = J_1\Delta\tau_1 \tag{5.23}$$

Isto porque o vetor de erro ΔF_0 não é definido, já que não existe o módulo neural 0. A Figura 5.3 mostra o procedimento de propagação retroativa de erro através do esquema modular.

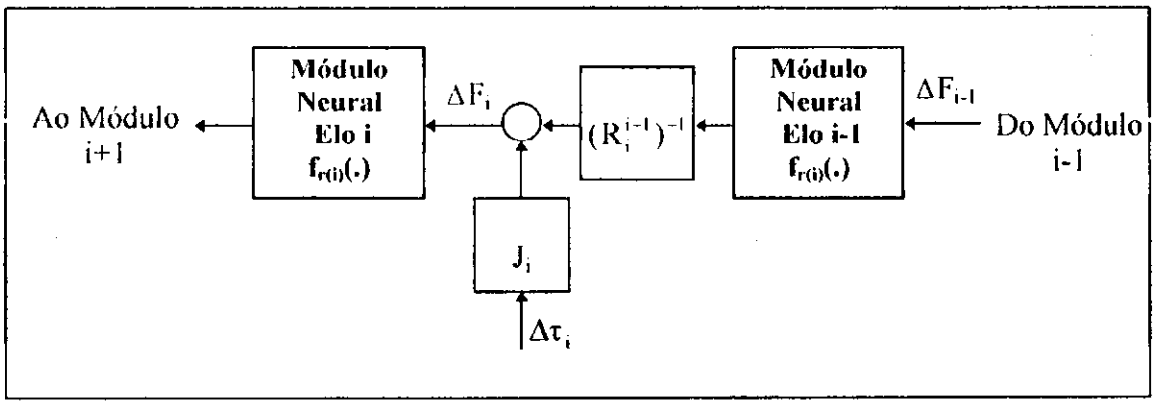

Fig. 5.3 - Propagação de Erro através de Esquema Modular.

Verifica-se que, no treinamento dos módulos neurais, é necessário calcular os vetores $\vec{K}_i$, os quais dependem das posições, velocidades e acelerações de juntas, e representam a resposta dinâmica do elo i do manipulador aos conjugados τ_i (i=1,...,N) aplicados. Para evitar a medição das acelerações de junta propõe-se utilizar, em lugar de $\vec{K}_i$, os vetores equivalentes $\vec{K}_i^*$, obtidos através da recursão Newton-Euler para Frente substituindo as acelerações de junta $\ddot{q}_i$ pelas acelerações desejadas $\ddot{q}_i^*$. Espera-se que, devido à realimentação Proporcional-Derivativa, a trajetória de junta se mantenha nas vizinhanças da trajetória desejada de maneira que esta aproximação seja razoável, visto que apenas os termos de $\vec{K}_i$ dependentes das acelerações irão apresentar alguma diferença em relação a $\vec{K}_i^*$. Assim, a Equação (5.19) é modificada para a seguinte forma alternativa, que dispensa a medição das acelerações de junta:

$$F_i = f_{d(i)}([(R_{i+1}^i . F_{i+1})^T \ (\vec{K}_i^*)^T]^T) \tag{5.24}$$

5.5 Esquema de Treinamento Adaptativo Direto

Nesta Seção, um esquema de treinamento adaptativo direto para o neurocontrolador modular é desenvolvido a partir do controlador adaptativo descrito no Capítulo 4. Pela propriedade de linearidade em parâmetros (Equação (4.16)), a relação entre pequenas variações paramétricas $\Delta\vec{H}$ e pequenas variações $\Delta\tau$ nos esforços de acionamento pode ser obtida:

$$\tau + \Delta\tau = \Phi.(\vec{H} + \Delta\vec{H}) \qquad \Rightarrow \qquad \Delta\tau = \Phi.\Delta\vec{H} \tag{5.25}$$

ou, para o modelo dinâmico do controlador adaptativo (Equação (4.29)):

$$\Delta\tau = \Phi_r.\Delta\hat{\vec{H}} \tag{5.26}$$

Por outro lado, desde que o período de amostragem T seja suficientemente pequeno, a seguinte aproximação é aceitável:

$$\dot{\hat{H}} \cong \frac{\Delta \hat{H}}{T} \tag{5.27}$$

onde $\Delta \hat{H}$ é a variação no vetor de parâmetros estimados durante o período T. Assim, substituindo a aproximação acima na lei de adaptação de parâmetros descrita pela Equação (4.30), a seguinte aproximação é obtida:

$$\Delta \hat{H} \cong \Gamma . \Phi_r^T . s \tag{5.28}$$

Substituindo a aproximação (5.28) na Equação (5.26), obtém-se:

$$\Delta \tau \cong [\Phi_r . \Gamma . \Phi_r^T] . s \tag{5.29}$$

A expressão acima transforma erros de trajetória (superfície de erro s) em erros nos esforços de acionamento $\Delta \tau$. Assim, a matriz $[\Phi_r . \Gamma . \Phi_r^T]$ pode ser utilizada como uma aproximação do Jacobiano inverso da planta, possibilitando a implementação de um esquema adaptativo direto para treinamento do neurocontrolador modular. A partir da superfície de erro s, os erros nos esforços de acionamento $\Delta \tau$ são computados usando a expressão (5.29). Os elementos $\Delta \tau_i$ do vetor $\Delta \tau$ podem ser utilizados então, para o cômputo do erro de saída dos módulos neurais, de acordo com a Equação (5.22). Desta maneira, os módulos neurais podem ser treinados, por meio do algoritmo de propagação retroativa, da mesma maneira descrita na Seção anterior. A única diferença é que os erros $\Delta \tau_i$ são obtidos diretamente a partir de erros de rastreamento de trajetória, resultando num Neurocontrolador Adaptativo Direto.

De acordo com a Equação (4.19), a matriz $[\Phi_r . \Gamma . \Phi_r^T]$ pode ser fatorada da seguinte maneira:

$$[\Phi_r.\Gamma.\Phi_r^T] = [J^T.R.(K.\Gamma.K^T).R^T.J] \tag{5.30}$$

onde, a matriz $[K.\Gamma.K^T]$ é particionada em submatrizes 6x6:

$$[K.\Gamma.K^T] = \begin{bmatrix} (K_1\Gamma_1K_1^T) & 0 & \cdots & 0 \\ 0 & (K_2\Gamma_2K_2^T) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & (K_N\Gamma_NK_N^T) \end{bmatrix} \tag{5.31}$$

Nota-se que a estrutura da forma fatorada (5.30) é muito semelhante à forma fatorada da matriz de inércia dada na Equação (3.61), diferindo apenas em que, no lugar da matriz H, aparece a matriz $[K.\Gamma.K^T]$. Assim, os erros nos esforços de acionamento $\Delta\tau_i$ podem ser computados a partir das superfícies de erros de trajetória s_i, através de um algoritmo recursivo, análogo ao utilizado para cálculo das reações inerciais τ_{m_i} a partir das acelerações $\ddot{q}_i$ (Equações (3.41) a (3.43)). Para isto, basta simplesmente substituir: τ_{m_i} por $\Delta\tau_i$, $\ddot{q}_i$ por s_i, $\dot{V}_{m_i}$ por ΔV_i , F_{m_i} por ΔF_i e H_i por $[K_i.\Gamma_i.K_i^T]$; assim:

Recursão para Frente - Treinamento Adaptativo Direto:

Inicialização: $\Delta V_0 = 0$

Faça i variar de 0 até N-1

$$\Delta V_{i+1} = R_{i+1}^{iT}.\Delta V_i + J_i.s_i \tag{5.32}$$

Fim de i

Recursão para Trás - Treinamento Adaptativo Direto

Inicialização: $\Delta F_{N+1} = 0$

Faça i variar de N até 1

$$\Delta F_i = R_{i+1}^i . \Delta F_{i+1} + (K_i . \Gamma_i . K_i^T) . \Delta V_i \qquad (5.33)$$

$$\Delta \tau_i = J_i^T . \Delta F_i \qquad (5.34)$$

Fim de i

A partir do algoritmo recursivo acima, um esquema modular de treinamento adaptativo direto é desenvolvido para o neurocontrolador proposto. Neste esquema, módulos de treinamento adaptativo direto, interligados seqüencialmente, são associados aos elos correspondentes do manipulador, conforme mostrado na Figura 5.4. Associando estes módulos de treinamento aos módulos neurais de controle (Figura 5.1), um neurocontrolador modular adaptativo direto para manipuladores robóticos é obtido.

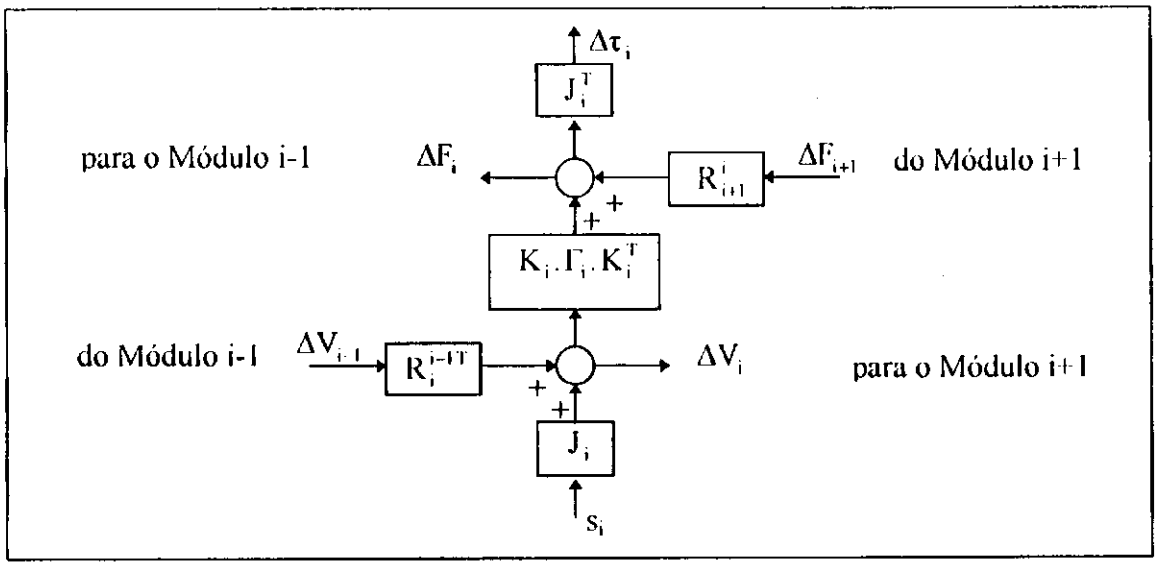

Fig. 5.4 - Módulo para Treinamento Adaptativo Direto.

5.6 Resultados de Simulação

Nesta Seção são apresentados resultados de simulação computacional do neurocontrolador modular proposto para um manipulador de três juntas. Os parâmetros cinemáticos (na convenção Denavit-Hartenberg) do manipulador de três juntas simulado são descritos na Tabela 5.1. Os parâmetros dinâmicos do mesmo manipulador são dados pelos seguintes vetores:

$$\vec{H}_1 = [3.000 \quad 0.000 \quad 0.000 \quad -0.750 \quad 0.125 \quad 0.125 \quad 0.000 \quad 0.000 \quad 0.000 \quad 0.000]^T \quad (5.35.a)$$

$$\vec{H}_2 = [1.800 \quad 0.360 \quad 0.000 \quad 0.000 \quad 0.000 \quad 0.096 \quad 0.096 \quad 0.000 \quad 0.000 \quad 0.000]^T \quad (5.35.b)$$

$$\vec{H}_3 = [1.500 \quad 0.225 \quad 0.000 \quad 0.000 \quad 0.000 \quad 0.045 \quad 0.045 \quad 0.000 \quad 0.000 \quad 0.000]^T \quad (5.35.c)$$

Tabela 5.1 - Parâmetros Cinemáticos - Manipulador de Três Juntas.

θ_i	d_i (m)	a_i (m)	α_i (rad)
q_1	0.50	0	0
q_2	0	0	$\pi/2$
q_3	0	0.40	0

Os ganhos de realimentação do neurocontrolador modular foram escolhidos iguais aos ganhos do controlador Torque Computado estudado no Capítulo 4, o que facilita o estudo comparativo do desempenho dos mesmos. Assim, para todas as juntas escolheram-se $K_D = 100$ e $K_P = 2500$, com um período de amostragem igual a $T = 5$ ms. Cada Módulo Neural foi implementado em uma rede de neurônios Adaline tipo sigmóide, possuindo dezoito entradas e seis neurônios na camada de saída. Na camada oculta, o número No de neurônios ocultos foi variado numa ampla faixa, de maneira a verificar experimentalmente a sua influência na resposta dinâmica do sistema. O coeficiente de aprendizado foi estabelecido como $\mu = 0.2$. Os pesos dos

módulos neurais foram inicializados com valores pequenos e randômicos ($|W_{ij}| < 0.1$). O treinamento inicial foi realizado on-line, utilizando o esquema inverso direto, através do rastreamento de uma trajetória de junta senoidal de amplitude $\pi/4$ rad e freqüência igual a 0.5 Hz. A Figura 5.5 mostra, sob estas condições de teste, a evolução do índice de desempenho quadrático J_q (Equação (4.63)), para diferentes valores do número No de neurônios da camada oculta dos módulos neurais do neurocontrolador proposto.

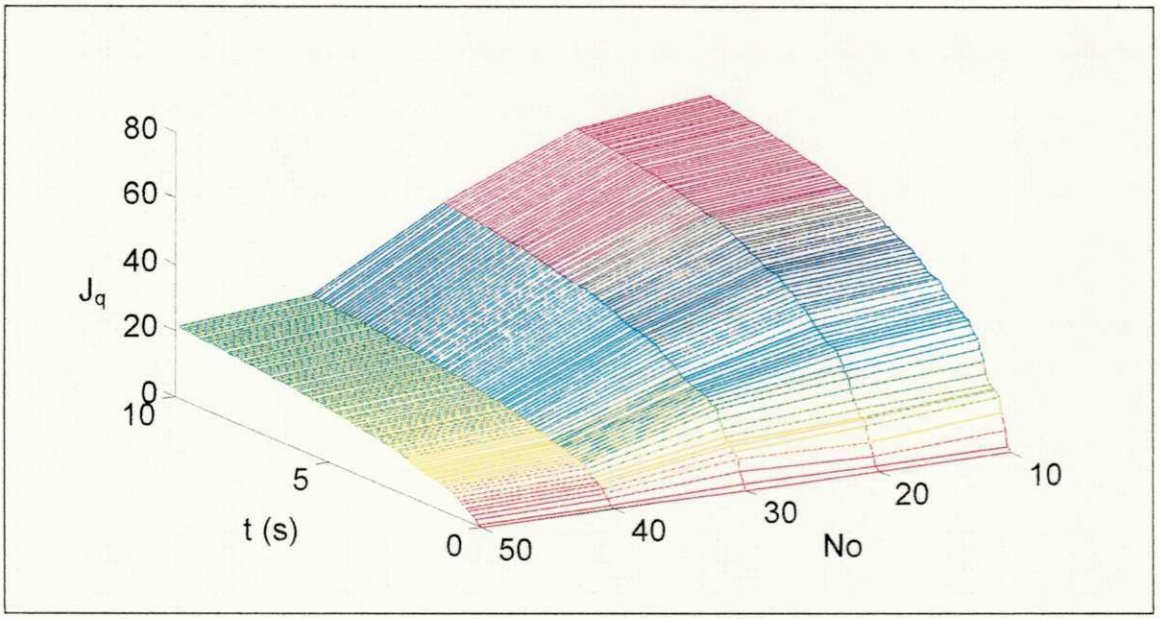

Fig. 5.5 - Índice de Desempenho em função do Número de Neurônios Ocultos.

Para um número No menor do que dez, observou-se que o neurocontrolador modular não consegue, de modo algum, rastrear a trajetória de referência. Isto pode ser atribuído a que o número de neurônios é insuficiente para aprender alguma informação sobre a dinâmica do sistema. Para valores maiores, pode-se observar na Figura 5.5 que, aumentando gradualmente No, o índice de desempenho melhora progressivamente. Não entanto, para valores de No maiores do que sessenta, o sistema torna-se instável, perdendo-se o controle da trajetória. Por este motivo, o valor de No deve ser escolhido entre dez e cinqüenta neurônios, estabelecendo um compromisso entre margem de estabilidade e desempenho.

A Figura 5.6 mostra a convergência do erro de aprendizado do neurocontrolador modular para a trajetória senoidal descrita acima, adotando No = 10. Na mesma figura mostra-se o erro de aprendizado médio para cada período da trajetória desejada, verificando-se uma convergência aproximadamente exponencial. A Figura 5.7 mostra o rastreamento de posições e velocidades de junta para o estágio inicial do treinamento. A Figura 5.8 mostra a evolução do erro de rastreamento de trajetória, bem como os esforços de acionamento de junta para a mesma simulação. Verifica-se que, à medida que a trajetória desejada é repetida, os módulos neurais vão aprendendo a compensar as não linearidades do sistema e o erro de rastreamento converge para valores cada vez menores.

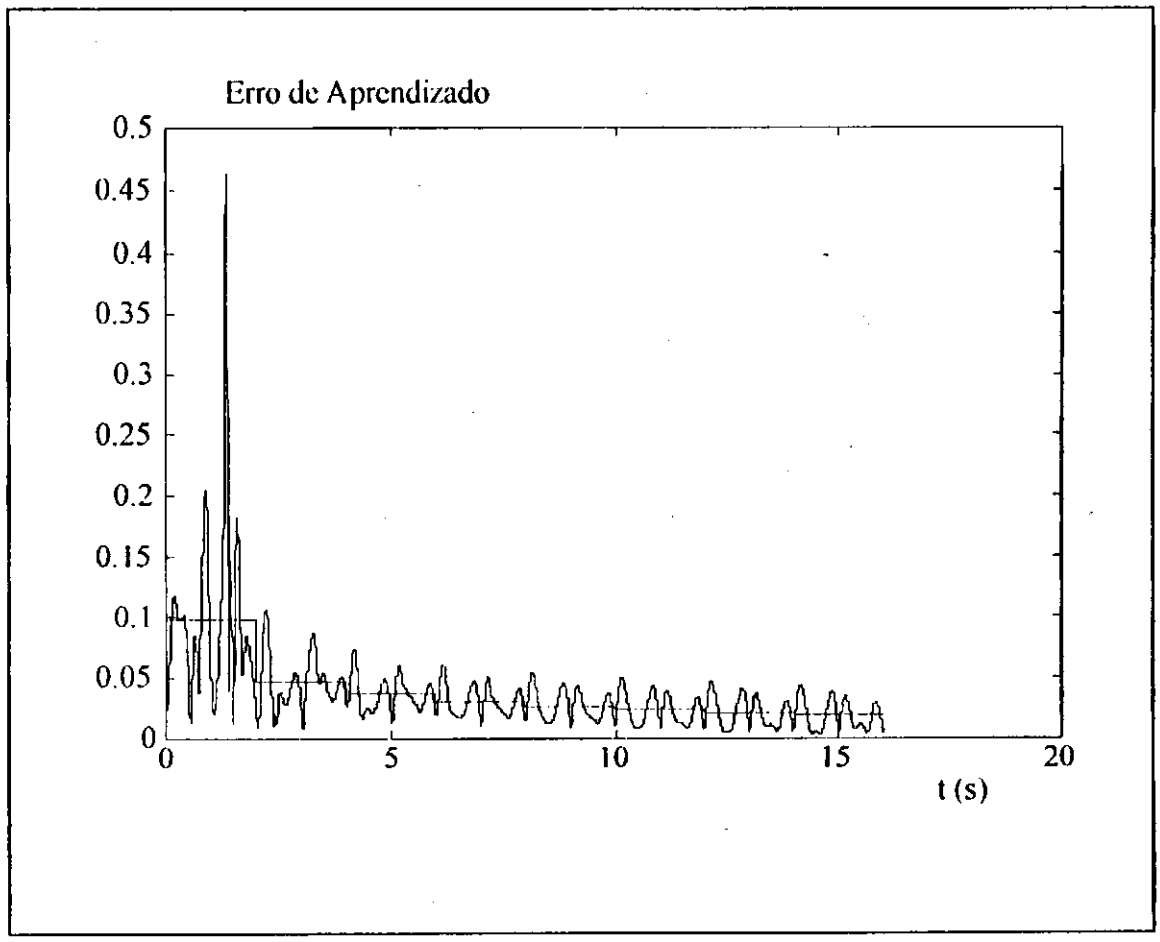

Fig. 5.6 - Neurocontrolador Modular - Erro de Aprendizado.

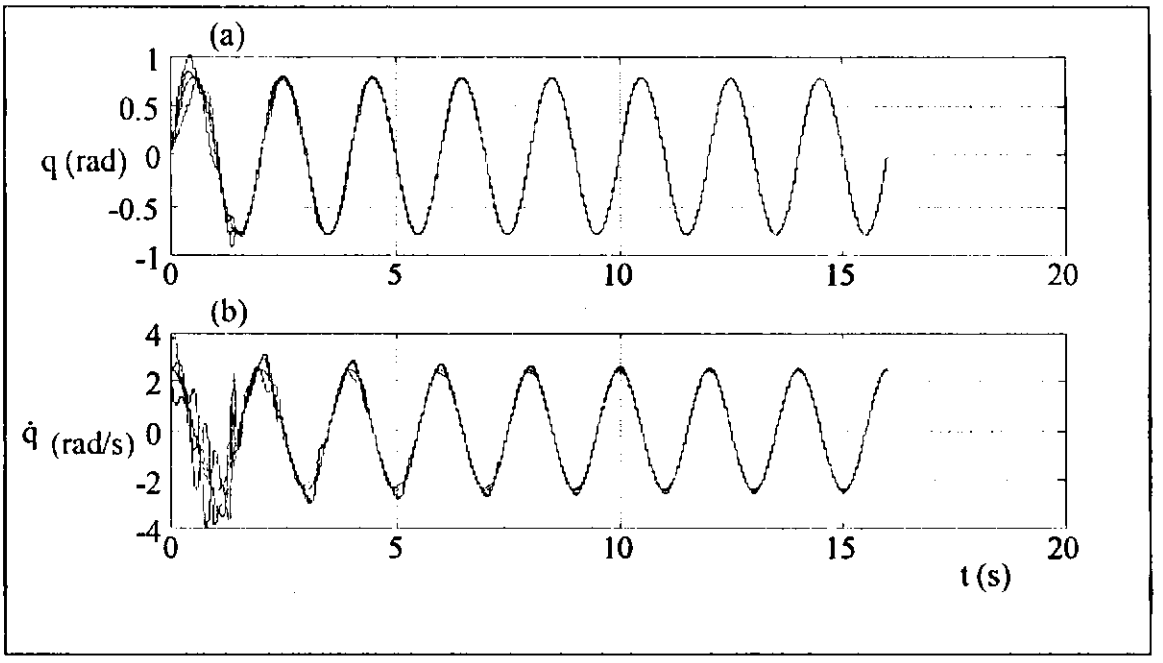

Fig. 5.7 - Rastreamento de Trajetória de Junta Senoidal.
a) Posições de Junta. b) Velocidades de Junta.

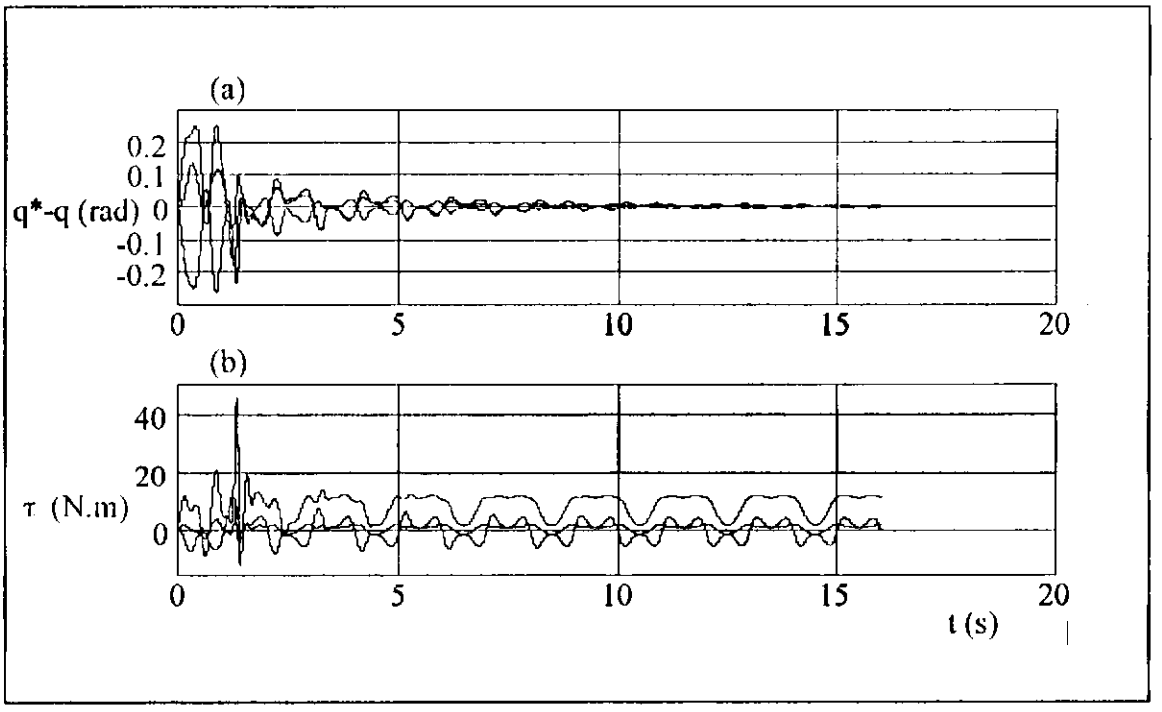

Fig. 5.8 - Convergência de Erro para Trajetória Senoidal.
a) Erro de Rastreamento. b) Esforços de Acionamento.

Após o treinamento inicial, estudou-se a resposta do neurocontrolador a variações de amplitude na trajetória desejada. A Figura 5.9 mostra a resposta do neurocontrolador para uma trajetória de junta senoidal de freqüência 0.5 Hz, com um degrau na amplitude, variando de $\pi/4$ rad a $\pi/8$ rad, aplicado no instante t = 2.0 s. Verifica-se que, apesar de submeter o sistema a uma variação de amplitude de 50 %, o rastreamento de trajetória permanece praticamente inalterado.

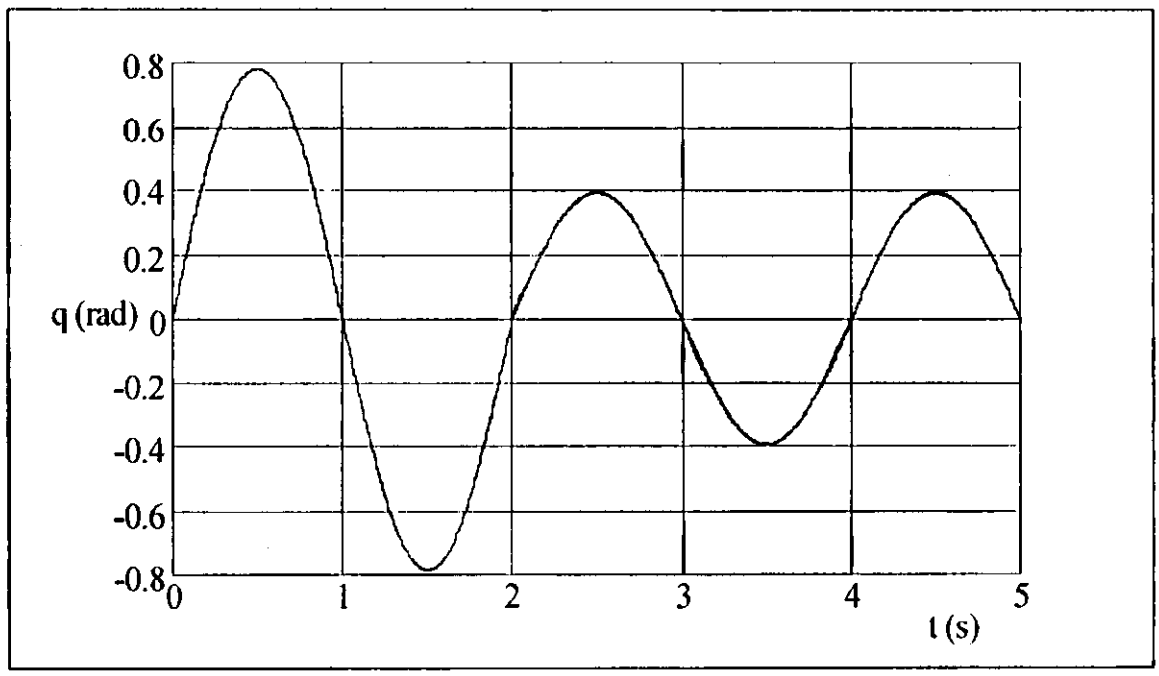

Fig. 5.9 - Neurocontrolador Modular - Variação em Amplitude.

Para estudar a resposta do neurocontrolador proposta a variações de freqüência, simulou-se o rastreamento de uma trajetória senoidal de amplitude $\pi/4$ rad, com um degrau em freqüência, variando de 0.5 Hz a 0.75 Hz, aplicado no instante t = 2.0 s. O rastreamento de trajetória de junta para estas condições é mostrado na Figura 5.10. Verifica-se que, apesar de um pequeno erro logo após a aplicação do degrau, o neurocontrolador consegue retomar o rastreamento de trajetória em poucos períodos.

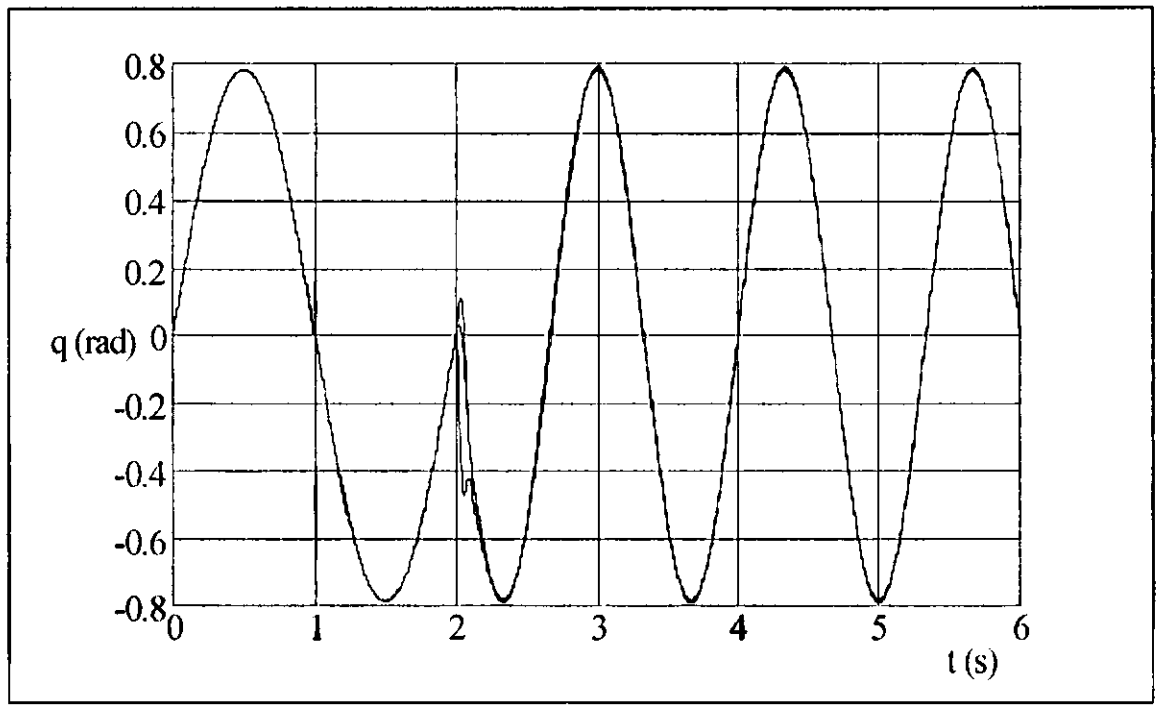

Fig. 5.10 - Neurocontrolador Modular - Variação em Freqüência.

Para verificar a capacidade de generalização do neurocontrolador modular proposto, ou seja a capacidade de rastrear trajetórias não aprendidas previamente, simulou-se o rastreamento de uma trajetória circular no espaço cartesiano. Esta trajetória é definida por uma circunferência com 0.10 metros de diâmetro situada num plano perpendicular ao braço estendido. O tempo para completar a circunferência foi estabelecido em 2.0 segundos. A Figura 5.11 mostra a trajetória de juntas para a primeira tentativa de rastreamento da circunferência no espaço cartesiano. A Figura 5.12 mostra, para a mesma situação, o rastreamento da circunferência no espaço cartesiano. O manipulador de três juntas é representado simplificadamente por meio de um diagrama unifilar, em diversas poses ao longo da trajetória proposta. Pode-se verificar que, apesar de o neurocontrolador não ter sido treinado previamente com esta trajetória, o manipulador consegue rastrear acuradamente a circunferência, mesmo na primeira tentativa.

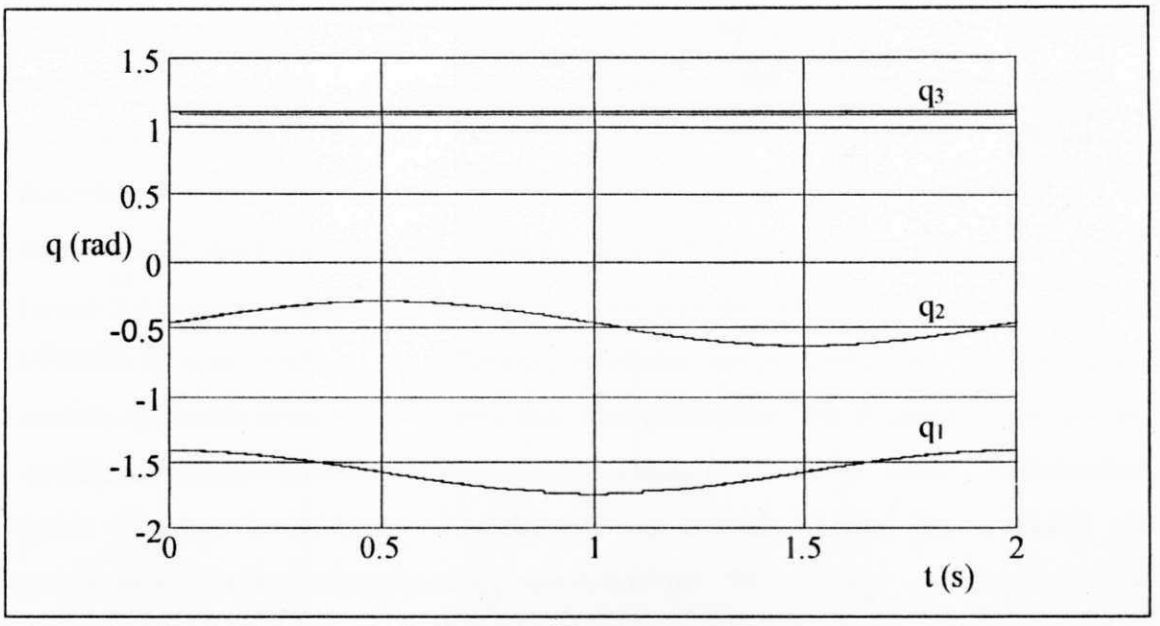

Fig. 5.11 - Rastreamento de Trajetória Circular - Espaço de Juntas.

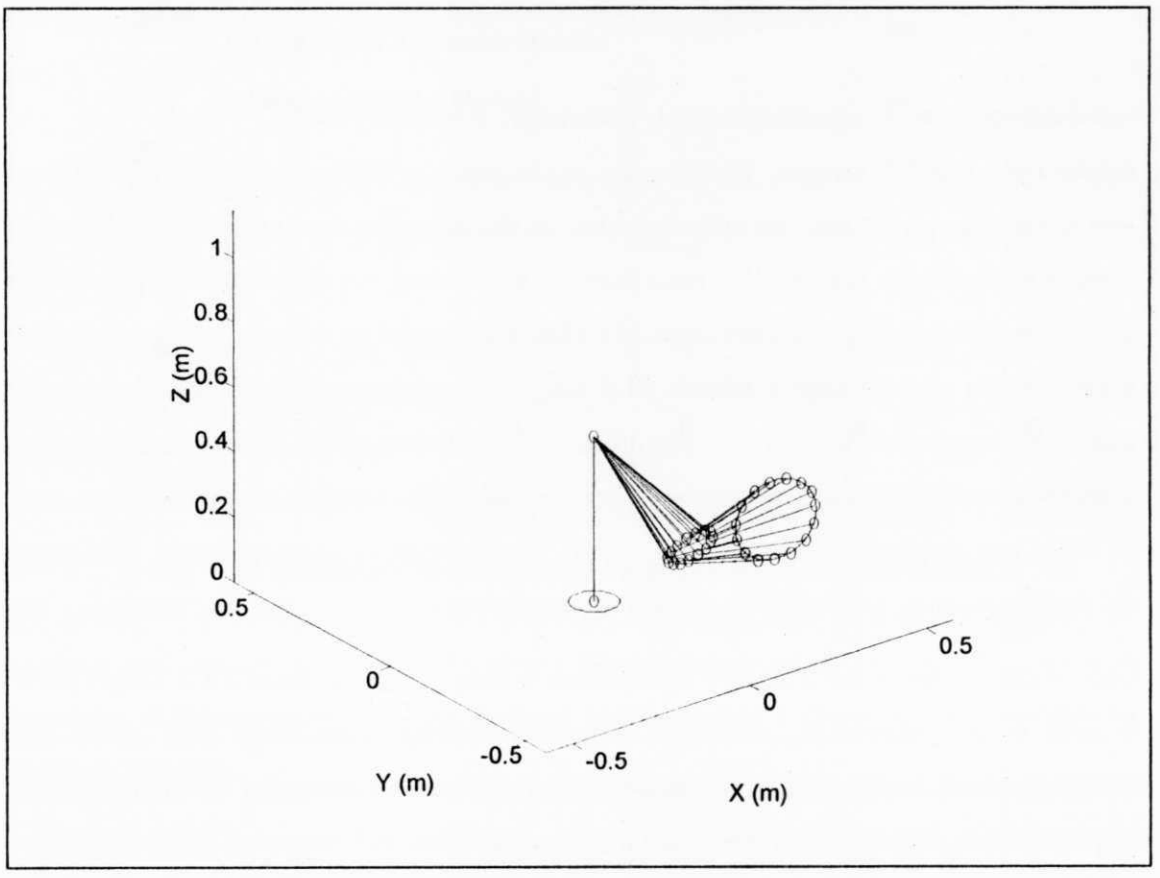

Fig. 5.12 - Rastreamento de Trajetória Circular - Diagrama Unifilar.

Visando uma análise comparativa, estudou-se a evolução do índice de desempenho J_q (Equação (4.63)) para o neurocontrolador modular, para o controlador adaptativo modular e para o controlador Torque Computado (estes últimos descritos no Capítulo 4). Para os três controladores, simulou-se o rastreamento de trajetórias de junta senoidais, com amplitude $\pi/4$ rad e freqüência igual a 0.5 Hz. A Figura 5.13 mostra a evolução do índice de desempenho dos três controladores para a trajetória de teste descrita acima. Para o controlador Torque Computado utilizou-se um modelo dinâmico com 50 % de erro nos seus parâmetros. Os parâmetros iniciais do modelo dinâmico utilizado no controlador adaptativo modular foram estabelecidos iguais a zero. O número de neurônios na camada oculta dos módulos do neurocontrolador modular foi estabelecido como No = 50.

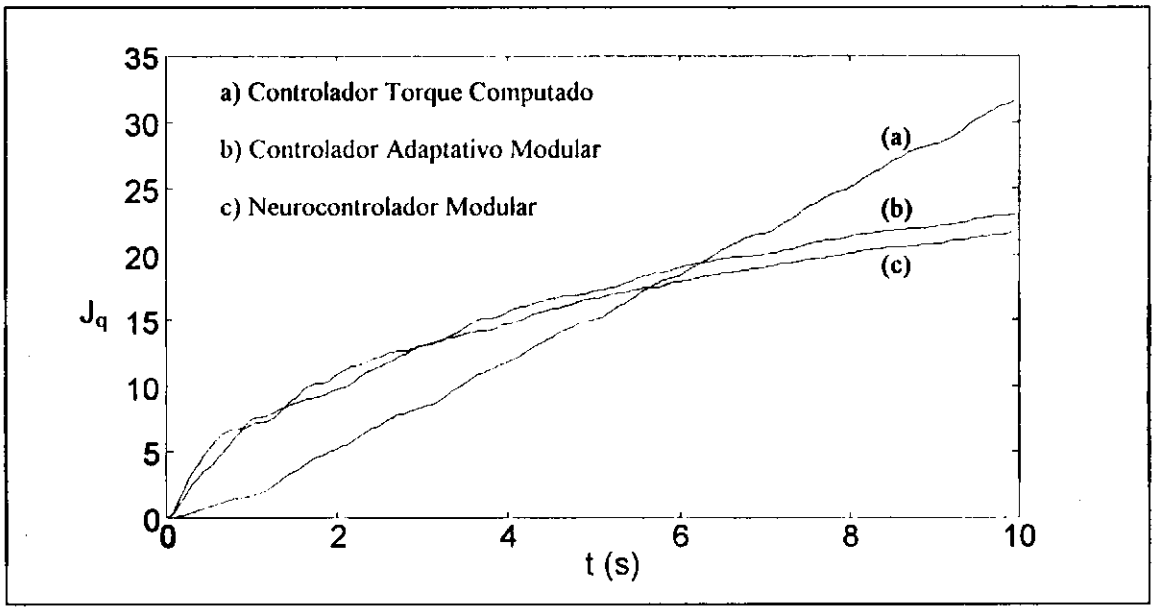

Fig. 5.13 - Análise Comparativa de Desempenho dos Controladores.

Pela Figura 5.13, verifica-se que, para o controlador Torque Computado, a declividade da curva $J_q(t)$ permanece aproximadamente constante, indicando que o erro de rastreamento de trajetória se mantém relativamente no mesmo nível médio à medida que a trajetória se repete. Por outro lado, para o Controlador Adaptativo modular e para o Neurocontrolador Modular, o índice de desempenho se comporta aproximadamente igual, apresentando uma declividade que diminui gradativamente. Isto significa que o

erro de rastreamento de trajetória converge para valores cada vez menores. Assim, embora que o controlador adaptativo e o neurocontrolador apresentem um erro maior durante o início da adaptação, após um certo tempo superam o desempenho do controlador Torque Computado, pois este não melhora com o tempo.

A seguir, conforme mostrado na Figura 5.14, simulou-se o rastreamento da trajetória de teste senoidal, utilizando o neurocontrolador modular, junto com o esquema de treinamento adaptativo direto proposto na Seção 5.5. Nesta simulação, estabeleceu-se um número de neurônios ocultos No = 10 para todos os módulos do neurocontrolador. Na Figura 5.15, compara-se a evolução do índice de desempenho J_q para o mesmo neurocontrolador modular, usando os esquemas de treinamento inverso direto e adaptativo direto. Verifica-se que o neurocontrolador adaptativo direto apresenta uma resposta mais rápida que a do neurocontrolador inverso direto, embora que, após algum tempo, o erro de rastreamento converge, de maneira semelhante em ambos os casos, para níveis cada vez menores.

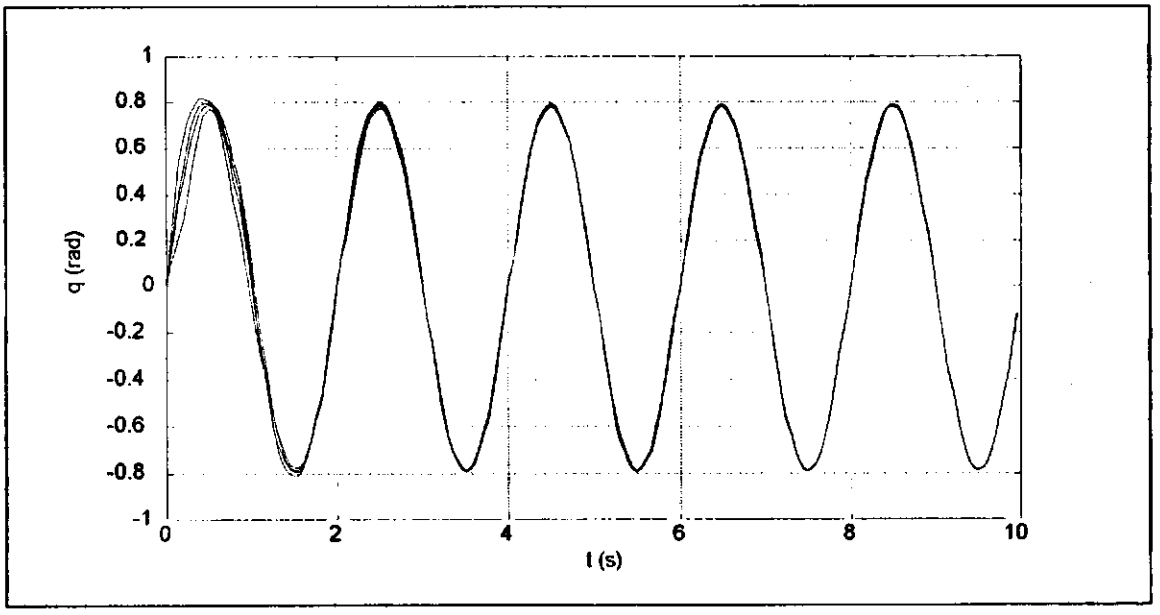

Fig. 5.14 - Rastreamento de Trajetória com Treinamento Adaptativo Direto.

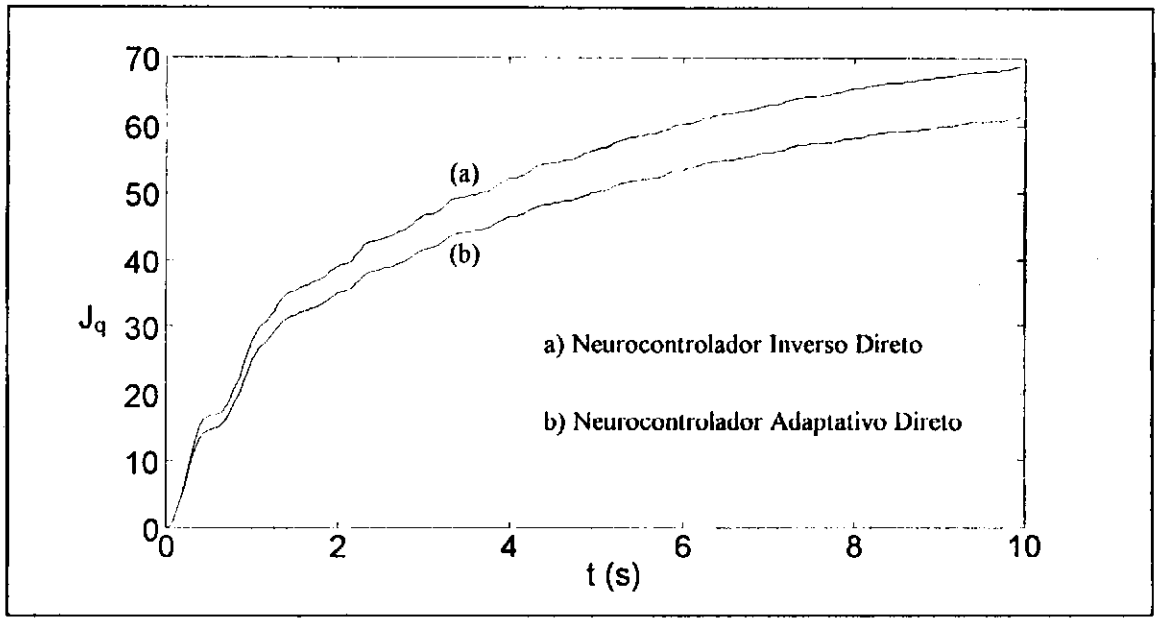

Fig. 5.15 - Índice de Desempenho - Esquemas de Treinamento.

5.7 Conclusão

Neste Capítulo, apresentou-se o conceito de Neurocontrole Modular. Baseados numa forma Linear em Termos Cinemáticos do método Newton-Euler, foram desenvolvidos Módulos Neurais padronizados para controle de manipuladores. Verificou-se experimentalmente que o número de neurônios na camada oculta de um módulo neural deve ser limitado entre dez e cinqüenta unidades. Os módulos neurais foram concatenados seqüencialmente, resultando num esquema global para controle de manipuladores de número de graus de liberdade arbitrário. Foi proposto um esquema de treinamento inverso direto para o neurocontrolador modular e, a partir do controlador adaptativo descrito no Capítulo 4, um esquema de treinamento adaptativo direto foi desenvolvido. Resultados de simulação computacional para um manipulador de três juntas permitiram verificar o bom desempenho do neurocontrolador proposto, mesmo em trajetórias não aprendidas previamente. A análise comparativa mostrou que o neurocontrolador modular inverso direto apresenta desempenho semelhante ao controlador adaptativo modular. Verificou-se que o treinamento adaptativo direto garante uma resposta mais rápida do que o esquema de treinamento inverso direto para o neurocontrolador modular.

CAPÍTULO 6

MANUAL DO ROBOTLAB

6.1 Introdução

Neste capítulo, descreve-se o ambiente ROBOTLAB para simulação dinâmica e animação gráfica de manipuladores robóticos. Na Seção 6.2 são apresentadas as características gerais do programa ROBOTLAB. Na Seção 6.3 descrevem-se as variáveis utilizadas para representar o estado do robô, bem como o formato dos arquivos do usuário que especificam as características do robô a ser simulado. A Seção 6.4 descrevem as diversas possibilidades de apresentação gráfica dos resultados de simulação. Um sumário das funções ROBOTLAB disponíveis é apresentado na Seção 6.5.

6.2 Características Gerais

O Simulador Gráfico de Robôs ROBOTLAB foi desenvolvido em linguagem MATLAB 4.0 [Moler, et alii, 1987], para rodar em computadores IBM PC. ROBOTLAB é um ambiente para simulação dinâmica de manipuladores robóticos com interface gráfica incluindo animação em 3D. ROBOTLAB é baseado numa biblioteca de funções desenvolvidas para a solução de problemas de cinemática, dinâmica, geração de trajetória e controle de robôs, sendo assim uma ferramenta adequada tanto para ensino, como para pesquisa na área de Robótica. ROBOTLAB visa suprir a falta de um protótipo real, simulando manipuladores robóticos de maneira bem realista. Devido aos altos custos de montagens experimentais na área de robótica, ROBOTLAB torna-se uma alternativa viável de baixo custo para ensino de um curso introdutório de robótica, permitindo o enriquecimento do conteúdo programático através da visualização dos conceitos teóricos na tela do computador. Na área de pesquisa, ROBOTLAB permite,

de maneira segura, testar exaustivamente novas estratégias de controle, antes de sua aplicação em protótipos reais.

ROBOTLAB é um simulador genérico, não estando limitado a um manipulador específico. Qualquer manipulador serial pode ser simulado, desde que o usuário especifique as suas características cinemáticas, dinâmicas e geométricas através de um arquivo de dados padronizado. As opções disponíveis de simulação dinâmica e saída gráfica incluindo animação em 3D tornam ROBOTLAB um verdadeiro laboratório virtual de robótica.

O formato de dados de entrada e saída segue o formato MATLAB padrão. A interface com o usuário dá-se, de maneira amigável, através de menus em janelas auto-explicativos acessíveis por mouse. A Figura 6.1 mostra como exemplo o menu principal ROBOTLAB.

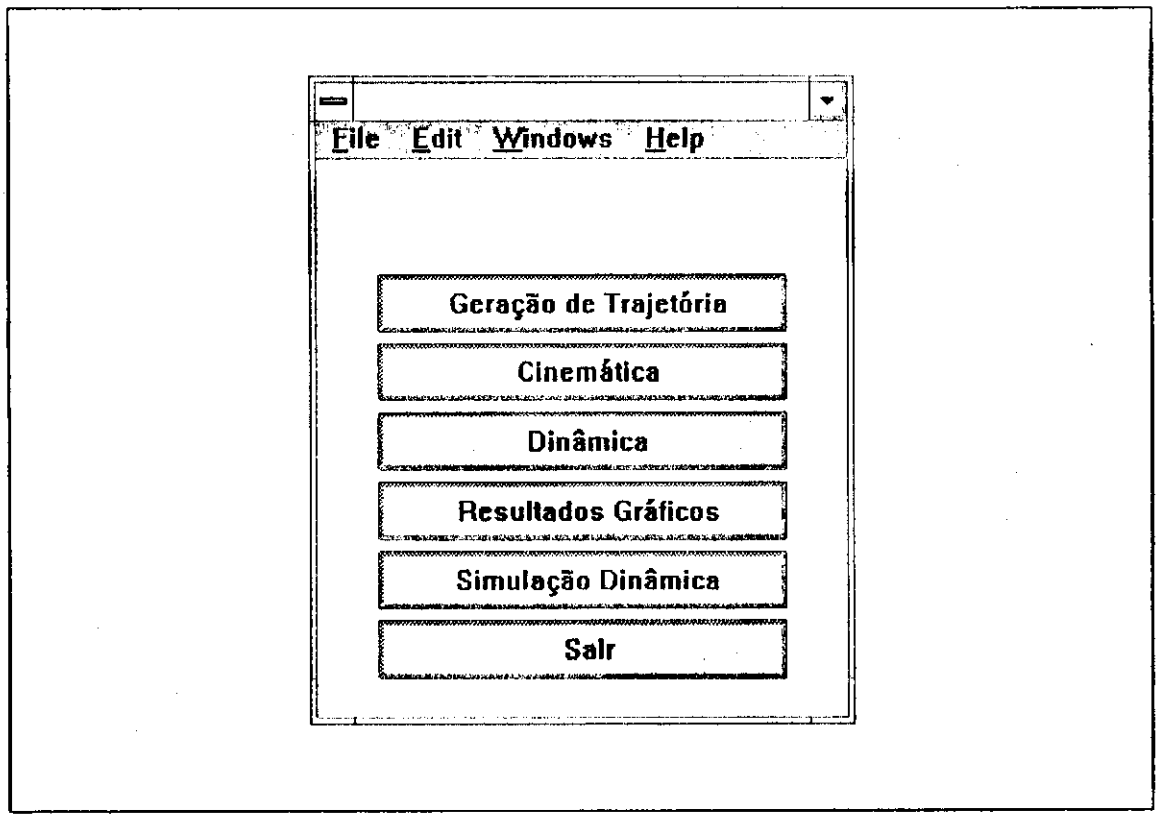

Fig. 6.1 - Menu principal do ROBOTLAB.

O programa ROBOTLAB foi construído de maneira modular e aberta de maneira a poder ser expandido pelo usuário para atender suas próprias necessidades. Assim, ROBOTLAB é dividido em cinco módulos principais: Geração de trajetória, Cinemática, Dinâmica, Resultados Gráficos e Simulação Dinâmica. Cada módulo dispõe de um conjunto de funções básicas, desenvolvidas para cada problema específico, que, devido à estrutura aberta, podem ser utilizadas pelo usuário em seus próprios programas.

Geração de Trajetória: Este módulo de ROBOTLAB contém um conjunto de funções básicas para gerar trajetórias (posição, velocidade e aceleração) em espaço de junta, utilizando polinômio de interpolação de quinta ordem entre os pontos inicial e final do percurso. Podem ser obtidos os coeficientes do polinômio interpolador, e vetor de estados do manipulador em qualquer ponto da trajetória. A trajetória gerada é gravada no arquivo **trajdata.mat**.

Cinemática: Neste módulo, estão incluídas várias funções relativas ao cálculo da cinemática de manipuladores: "Home Position", Transformadas de Elos, Cinemática Direta, Cinemática Inversa, Ângulos de Euler, etc. Devido à complexidade do problema de cinemática inversa, o usuário deve fornecer um arquivo em formato MATLAB "Scriptfile" para cálculo da cinemática inversa do seu próprio manipulador. Este arquivo deve calcular o vetor de variáveis de juntas dadas a posição e orientação da ferramenta. O usuário dispõe ainda da opção de visualização gráfica em 3D do manipulador na configuração espacial escolhida.

Dinâmica: Este módulo inclui uma série de funções, baseadas na formulação Newton-Euler, relativas ao cálculo dos componentes da equação dinâmica do manipulador numa dada situação de funcionamento. Dispõe-se assim, de funções para o cálculo da Dinâmica Direta, Dinâmica Inversa, Matriz de Inércia, reações Coriolis/Centrífugas, reações gravitacionais, Atrito Coulomb, Atrito Viscoso, etc.

Simulação Dinâmica: Este módulo consiste basicamente de um simulador dinâmico de robôs utilizando integração numérica por método de Runge-Kutta de quarta ordem. O usuário deve fornecer um arquivo de controlador do seu próprio robô de maneira a calcular, através de qualquer lei de controle da sua escolha, o vetor de conjugados de acionamento. O usuário tem acesso aos valores de posição, velocidade e aceleração das juntas, armazenados em variáveis adequadas. O usuário deve fornecer ainda um arquivo para inicialização de qualquer parâmetro necessário ao seu controlador. O estado inicial e o tempo de simulação são requeridos interativamente pelo programa. Os resultados de simulação são armazenados no arquivo **simudata.mat**.

Resultados Gráficos: ROBOTLAB apresenta um módulo de saída para resultados gráficos, incluindo animação 3D, que o tornam um verdadeiro laboratório virtual de Robótica. Este módulo faz uso de uma série de primitivas gráficas, disponíveis para o usuário construir seu arquivo de parâmetros geométricos do robô. Utilizando os dados de simulação dinâmica ou geração de trajetória, este módulo permite apresentar os resultados graficamente de diversas formas. O usuário pode visualizar os resultados na forma de Curvas de Resposta no Tempo, Animação 3D, Trajetória da Ferramenta no Espaço, Diagrama Unifilar, etc.

6.3 Variáveis de Estado e Arquivos do Usuário

Os dados de entrada ou saída no ambiente ROBOTLAB devem obedecer ao formato MATLAB, no qual uma matriz é representada entre colchetes, os elementos de uma linha separados por espaços em branco e as linhas separadas por ponto e vírgula. Por exemplo, para criar um vetor coluna 3x1 q com os ângulos de junta $q_1 = 90^0$, $q_2 = 45^0$ e $q_3 = -45^0$ para um manipulador de três juntas, a notação correta é:

```
q = [pi/2;pi/4;-pi/4]                                    (6.1)
```

Particularmente, entradas de dados solicitadas pelo programa, seja através de arquivos do usuário ou pelo teclado devem ser feitas neste formato.

ROBOTLAB utiliza um conjunto de variáveis globais que definem a cada instante o estado atual do manipulador. Estas variáveis podem ser utilizadas pelo usuário para elaborar seus próprios programas, tais como controladores ou rotinas para cálculo de cinemática inversa. As variáveis de estado são:

Q = Vetor Nx1 de variáveis de junta.
DQ = Vetor Nx1 de velocidades de junta.
D2Q = Vetor Nx1 de acelerações de junta.
TSIM = Tempo atual de simulação.

O arquivo de controlador do usuário deve especificar as seguintes variáveis, associadas à trajetória de junta desejada e ao esforço de acionamento correspondente:

Qr = Vetor Nx1 de variáveis de junta de referência.
DQr = Vetor Nx1 de velocidades de junta de referência.
D2Qr = Vetor Nx1 de acelerações de junta de referência.
TORQ = Vetor Nx1 de esforços de acionamento das juntas.

Os resultados de simulação dinâmica e geração de trajetória são armazenados temporariamente nos arquivos **simudata.mat** (resultados de simulação dinâmica) ou **trajdata.mat** (resultados de geração de trajetória). Estes arquivos armazenam as seguintes matrizes:

THIST = Tempo de simulação.
QHIST= Variáveis de junta.
DQHIST = Velocidades de junta.
D2QHIST = Acelerações de junta.
TORQHIST = Esforços de acionamento.
QRHIST = Variáveis de junta de referência.
DQRHIST = Velocidades de junta de referência.
D2QRHIST = Acelerações de junta de referência.

ROBOTLAB permite a simulação de qualquer manipulador robótico serial da escolha do usuário. A especificação do robô a ser simulado deve ser feita através de um arquivo de parâmetros fornecido pelo usuário. Este arquivo, no formato "script-file" da linguagem MATLAB, deve fornecer os parâmetros cinemáticos, dinâmicos e geométricos do manipulador a ser simulado. ROBOTLAB inclui o arquivo **campina.m** com os parâmetros de um manipulador robótico de três juntas. Este arquivo está incluído no Apêndice I. Os parâmetros cinemáticos devem ser fornecidos na forma de uma matriz Nx5 denominada KINPAR, na qual cada linha consiste de cinco elementos que descrevem completamente a cinemática do elo: o tipo de junta (rotacional ou prismática) e os quatro parâmetros Denavit-Hartenberg na convenção proposta por Craig [Craig, 1986]. Assim, a linha da matriz KINPAR correspondente a uma junta i deve seguir o seguinte formato:

$$KINPAR(i,:) = [jf_i \quad a_i \quad \alpha_i \quad d_i \quad \theta_i] \tag{6.2}$$

onde o "flag" de junta jf_i é igual às strings 'rot' (junta i rotacional) ou 'pri' (junta i prismática). Os parâmetros Denavit-Hartenberg são:

$\mathbf{a_i}$ = Comprimento do elo i.
$\boldsymbol{\alpha_i}$ = Ângulo de torção do elo i.
$\mathbf{d_i}$ = Comprimento da junta i.
$\boldsymbol{\theta_i}$ = Ângulo da junta i.

A variável de junta (d_i ou θ_i , conforme a junta i seja prismática ou rotacional) deve assumir o valor correspondente ao seu valor na posição "Home". Os parâmetros cinemáticos da ferramenta devem ser fornecidos no vetor 1x5 TOOLKINPAR, seguindo o mesmo formato, mas seu primeiro elemento deve ser igual à string 'tool'.

De maneira análoga, os parâmetros dinâmicos são fornecidos numa matriz Nx12 denominada DYNPAR, na qual cada linha, associada ao elo correspondente, contem a massa, vetor de posição do centro de massa vezes a massa do elo, tensor de inércia (na forma de um vetor 1x6), coeficiente de atrito viscoso (c_{av}) e

coeficiente de atrito Coulomb (c_{ac}). Assim para o elo i, a linha da matriz DYNPAR correspondente deve obedecer ao seguinte formato:

$$DYNPAR(i,:) = \left[m_i \quad m_i p_{c_{ix}} \quad m_i p_{c_{iy}} \quad m_i p_{c_{iz}} \quad h_{ixx} \quad h_{iyy} \quad h_{izz} \quad h_{iyz} \quad h_{izx} \quad h_{ixy} \quad c_{av} \quad c_{ac} \right] \quad (6.3)$$

As características geométricas de cada elo são fornecidas através de variáveis apropriadas, associadas a partes do elo. Assim, os elos são numerados a partir da base. Cada elo é construído em partes básicas numeradas, criadas a partir de funções primitivas gráficas especiais (esferas, hemisférios, cilindros, prismas e cones). Cada parte é definida por duas variáveis. A primeira destas variáveis é denominada da seguinte forma: P<xx>L<yy>, onde <xx> é um número de dois algarismos que indica o número correspondente a esta parte do elo e <yy> é um número de dois algarismos correspondente ao número do elo. Esta variável deve ser inicializada, utilizando as primitivas gráficas disponíveis, de maneira a armazenar a forma e tamanho da parte correspondente. Assim os diferentes tipos de partes devem ser especificados da seguinte maneira:

- Esfera: PxxLyy = [1 raio]

onde o Nº 1 foi utilizado para classificar a parte como uma esfera e "raio" é o valor do raio da mesma.

- Hemisfério: PxxLyy = [2 raio]

onde o Nº 2 foi utilizado para classificar a parte como um hemisfério e "raio" é o valor do raio da mesmo.

- Cilindro: PxxLyy = [3 h_z raiovec]

onde o Nº 3 foi utilizado para classificar a parte como um cilindro, h_z é a altura do mesmo e "raiovec" é um vetor linha contendo os valores do raio do cilindro a intervalos igualmente espaçados da altura h_z. O número de elementos deste vetor é opcional.

- Prisma: PxxLyy = [4 h_z lxvec lyvec]

onde o Nº 4 foi utilizado para classificar a parte como um prisma, h_z é a altura do mesmo e "lxvec" e "lyvec" são vetores linha contendo os valores dos comprimentos dos lados x e y do prisma a intervalos igualmente espaçados da altura h_z. O número de elementos destes dois vetores deve ser igual e é opcional.

- Cone: PxxLyy = [5 h_z raio]

onde o Nº 5 foi utilizado para classificar a parte como um cone, h_z é a altura do mesmo e "raio" é o valor do raio da base.

A segunda variável, associada à parte, deve ser inicializada com uma matriz de transformação homogênea indicando a posição e orientação da parte em relação ao referencial do elo ao qual está associada. O formato geral desta variável é TP<xx>L<yy>, onde <xx> e <yy> foram definidos acima. Assim, por exemplo a parte 3 do elo 4 poderia ser gerada da seguinte forma:

```
P03L04 = [3  2  [1  1  0.5]];

TP03L04 = [1  0  0  5
           0  1  0  0
           0  0  1  0
           0  0  0  1];
```

A variável P03L04 indica que se trata de um cilindro (Nº 3) de altura 2 com raio = 1 na base, raio = 1 no centro e raio 0.5 na extremidade superior do mesmo. A variável TP03L04 indica que o cilindro está alinhado com o sistema de referências do elo (matriz de orientação = identidade) e localizado a 5 unidades na direção positiva do eixo x. Desta maneira, o usuário pode especificar o seu próprio robô de maneira

acurada, incluindo tantos detalhes quantos quiser. A Figura 6.2 mostra o manipulador CAMPINA, cujo arquivo de parâmetros está incluído em ROBOTLAB (ver Apêndice I).

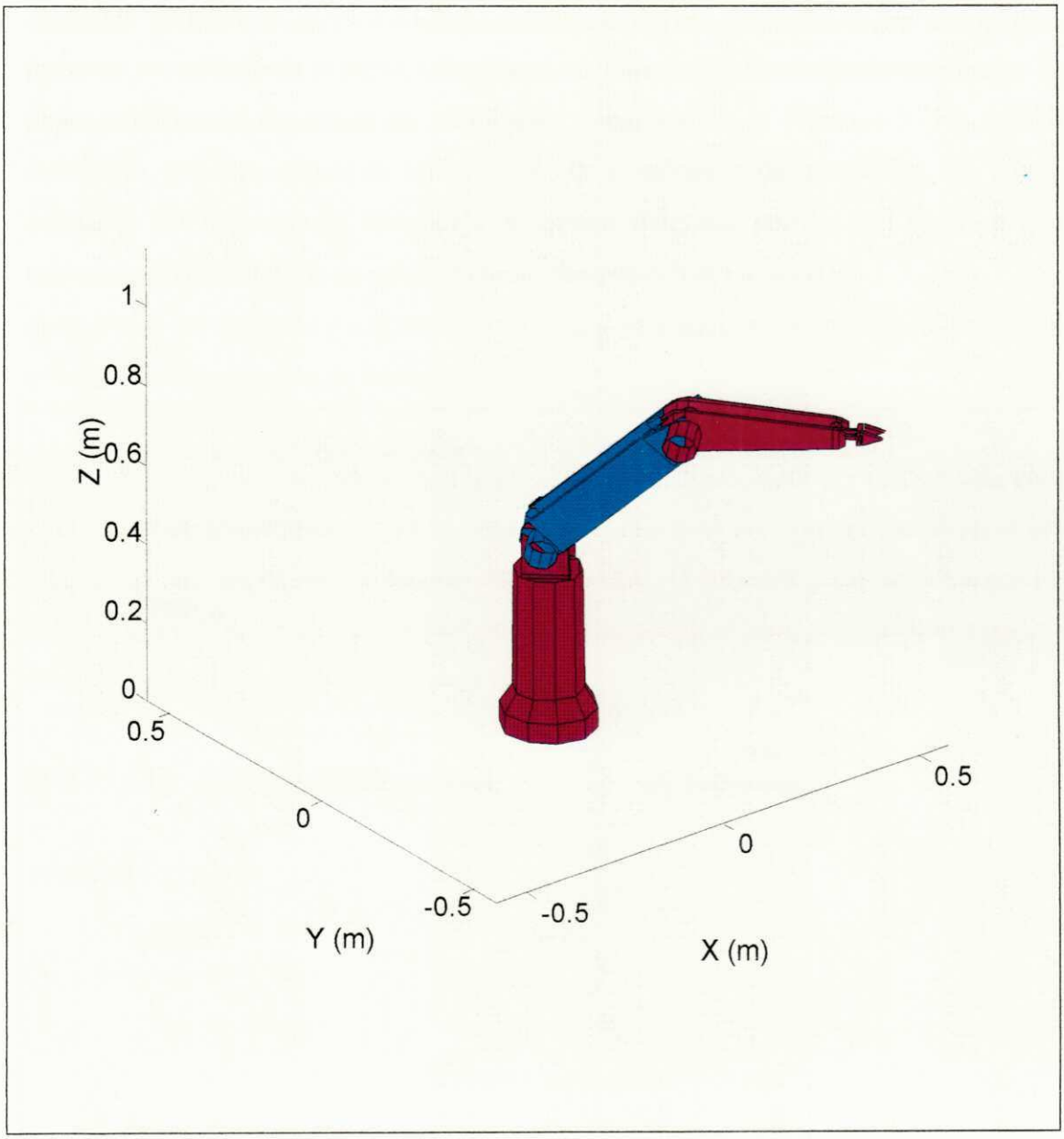

Fig. 6.2 - Exemplo de modelamento geométrico: o robô CAMPINA.

6.4 Apresentação de Resultados

O usuário de ROBOTLAB dispõe de uma interface gráfica para apresentação de resultados incluindo animação em 3D. Os resultados de simulação dinâmica podem ser apresentados na forma de curvas de resposta no tempo, que mostram a evolução da posição, velocidade, aceleração e esforços de acionamentos de juntas em função do tempo de simulação, como mostra a Figura 6.3. No espaço cartesiano, o usuário dispõe de varias opções de visualização dos resultados, tais como, animação 3D, trajetória da ferramenta no espaço, diagrama unifilar (ver Figura 6.4) e rastreamento de trajetória da garra em múltiplas poses (ver Figura 6.5).

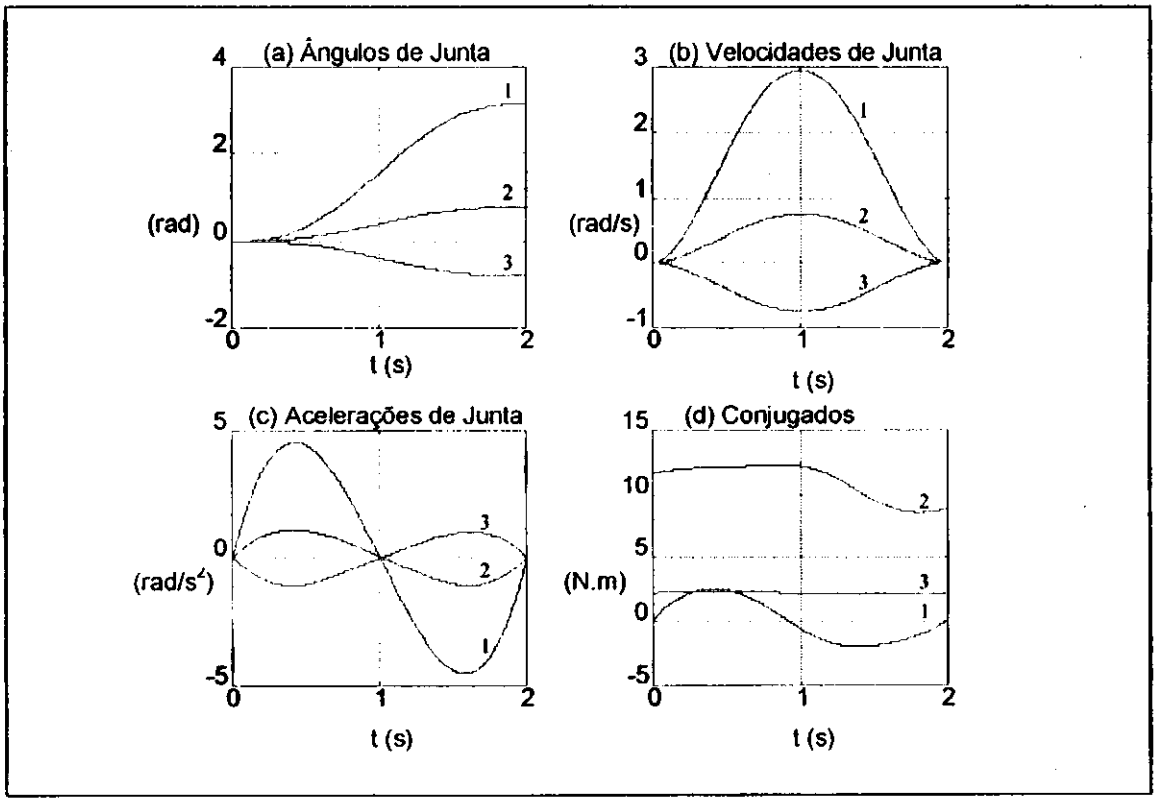

Fig. 6.3 - Curvas de Resposta no Tempo.

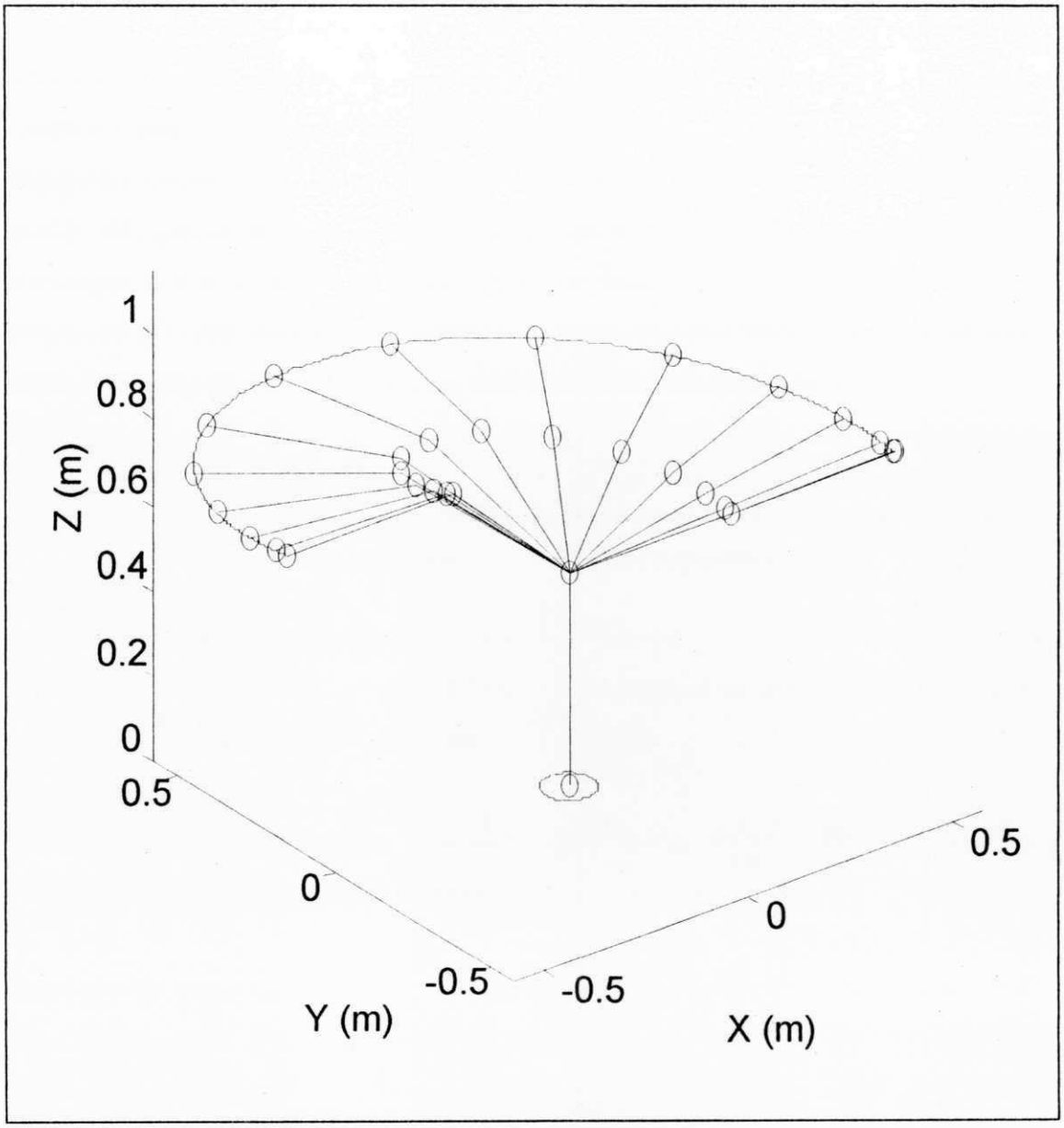

Fig. 6.4 - Diagrama Unifilar.

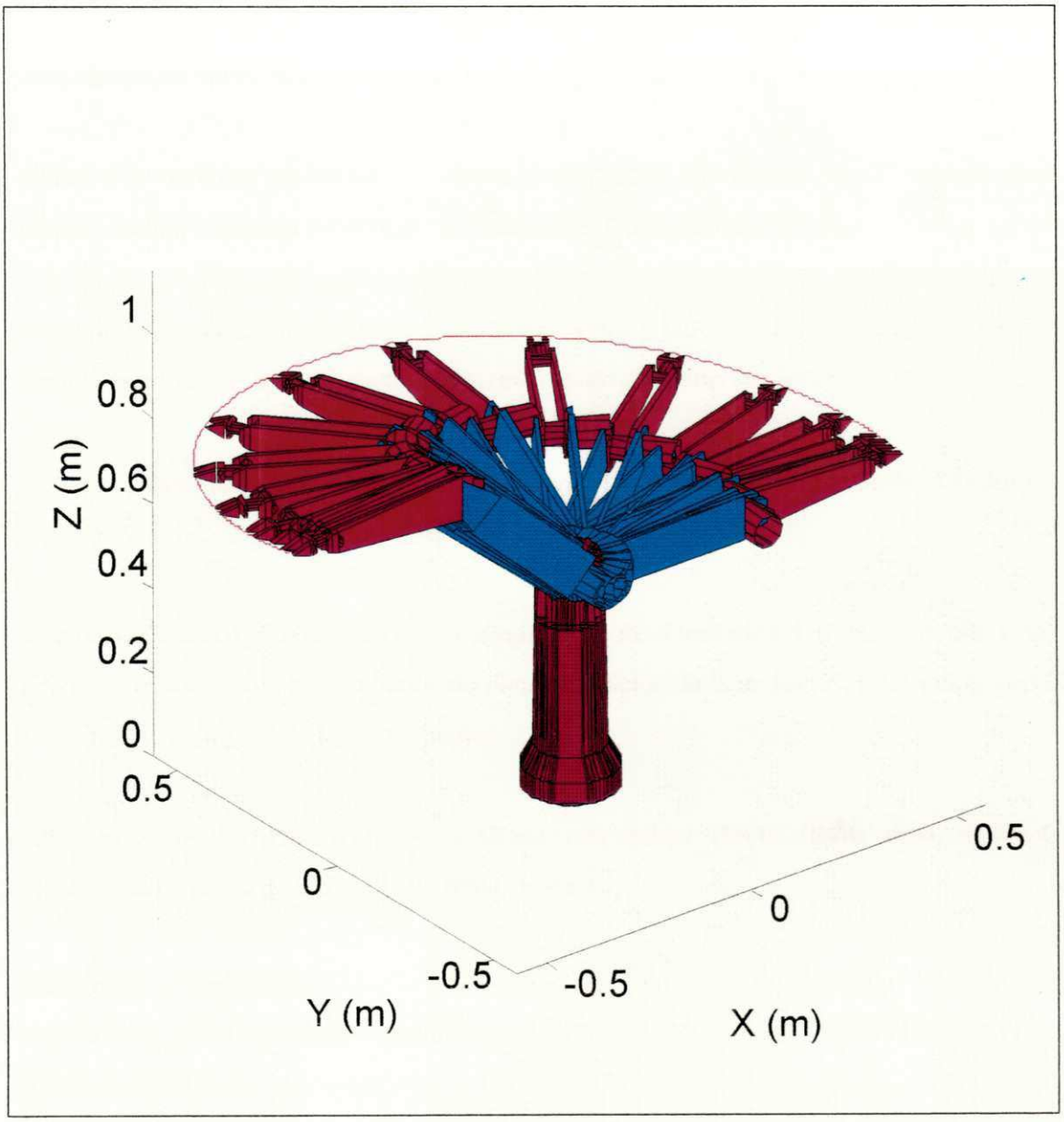

Fig. 6.5 - Rastreamento de Trajetória. Animação 3D em múltiplas poses.

6.5 Funções ROBOTLAB

A seguir o conjunto das funções ROBOTLAB é listado, incluindo uma descrição sumária das mesmas.

function a=**coeftraj**(trajlim,tf) Gera coeficientes polinômio de 5ª ordem para 0<t<tf, dada a trajetória de q0 a qf: trajlim=[q0;dq0;d2q0;qf;dqf;d2qf].

function tauc=**coritorq**(q,dq) Calcula vetor tauc de torques de reações coriolis/centrífugas a partir de posição, velocidade de juntas (Vetores q, dq).

function tauco=**coultorq**(dq) Calcula vetor tauco de torques de atrito "Coulomb" a partir de velocidade de juntas (Vetor dq).

function d2q=**dirdyn**(q,dq,tau) Computação de Dinâmica Direta. Calcula vetor d2q de acelerações de junta a partir de posição e velocidade de juntas (Vetores q, dq) e vetor tau de conjugados de acionamento.

function basetool=**dirkin**(jvar) Calcula cinemática direta (transformação base-ferramenta) a partir de variáveis de junta (jvar).

function Vm=**escpm**(V) Transforma vetor V em matriz Vm (3x6) equivalente para produto escalar vezes matriz M (3x3). M(3x3)*V(3x1) = Vm(3x6)*Mv(6x1).

function ang=**eulerang**(R) Calcula ângulos de Euler a partir de matriz de rotação.

function taug=**gravtorq**(q) Calcula vetor taug de torques de reação gravitacional a partir de posição de juntas (Vetor q).

function homejvar=**homepos** Fornece vetor de variáveis de junta em "home-position".

function M=**inermat(q)** Calcula Matriz de Inércia M a partir de vetor de posições de junta q.

function taui=**inertorq**(q, d2q) Calcula vetor taui de torques inerciais a partir de posição, e velocidade de juntas (Vetores q, d2q).

function tau=**invdyn**(q,dq,d2q) Computação de Dinâmica Inversa. Formulação Newton-Euler. Calcula vetor de conjugados tau a partir de posição, velocidade e aceleração de juntas (Vetores q, dq, d2q).

function QVEC=**invkin**(ROTMAT,POSVEC) Função cinemática inversa. Calcula o vetor de variáveis de junta QVEC a partir da orientação ROTMAT (3x3) e posição POSVEC (3x1) da ferramenta. O usuário deve fornecer o arquivo <INVKINNAME>.m (script-file) que calcula QVEC em função das variáveis ROTMAT e POSVEC.

function [x,y,z]=**pcone**(h,radius,TRANSF) Gera coordenadas x,y,z de cone de altura h e raio da base "radius". (Coordenadas em relação à localização TRANSF).

function [x,y,z]=**pcylind**(h,radiusvec,TRANSF) Gera coordenadas x,y,z de cilindro de altura h e raios igualmente espaçados definidos no vetor "radiusvec" . (Coordenadas em relação a localização TRANSF).

function [x,y,z]=**phemis**(radius,TRANSF) Gera coordenadas x,y,z de hemisfério de raio "radius" em localização TRANSF.

function [x,y,z]=**pprism**(hzvec,sidexvec,sideyvec,TRANSF) Gera coordenadas x,y,z de prisma de dimensões x, y dadas nos vetores sidexvec e sideyvec correspondentes às alturas dadas no vetor hzvec. (Coordenadas em relação à localização TRANSF).

function [x,y,z]=**pspher**(radius,TRANSF) Gera coordenadas x,y,z de esfera de raio "radius" em localização TRANSF.

function taur=**reactorq**(q,dq) Calcula vetor de torques de reação taur (vetor de conjugados - torques inerciais) a partir de posição, velocidade juntas (Vetores q, dq).

function [STATEF,DSTATEF] = **rgkrobot**(STATEI,TAU,h) Rgkrobot resolve sistema de equações diferenciais do robô através de método de Runge-Kutta de quarta ordem. STATE é o vetor de estados, DSTATE a sua derivada (I e F indicam estado inicial e final), TAU é o vetor de esforços de acionamento e h o intervalo de integração.

function DX=**robotdyn**(X,TAU) Calcula derivada DX do vetor de estado do manipulador X=[q';dq'] para o vetor de esforços de acionamento TAU.

function R=**rotmatx**(ang) Gera matriz R (3x3) correspondente a uma rotação "ang" (rad) em torno do eixo x.

function R=**rotmaty**(ang) Gera matriz R (3x3) correspondente a uma rotação "ang" (rad) em torno do eixo y.

function R=**rotmatz**(ang) Gera matriz R (3x3) correspondente a uma rotação "ang" (rad) em torno do eixo z.

function **showlink**(linknumb,Tlinkbase) Apresentação em 3-D do elo número "linknumber" na localização "Tlinkbase".

function **showpart**(partnumb,linknumb,Tlinkbase) Apresentação 3-D da parte número "partnumb" do elo número "linknumb" na localização "Tlinkbase".

function **showrob**(jvar) Apresenta imagem em 3-D do robô na tela na configuração de variáveis de junta jvar.

function **showunif**(jvar) Apresenta diagrama unifilar 3-D do robô na tela na configuração de variáveis de junta jvar.

function [q,dq,d2q,ti]=**traject**(trajlim,tf,np) Gera np pontos de trajetória [q,dq,d2q] utilizando polinômio de 5ª ordem. Variáveis de junta variando de q0<q< qf e tempo variando de 0<t<tf. trajlim=[q0;dq0;d2q0;qf;dqf;d2qf].

function [q,dq,d2q]=**trajpnt**(t,a) Calcula valor de trajetória q, dq, d2q no instante t de acordo com os coeficientes "a" do polinômio interpolador de 5ª ordem.

function linktb=**transfb**(linkt) Calcula transformações de elo em relação à base a partir de transformações de elo.

function linkt=**transfor**(jvar) Calcula transformações de elo em função de variáveis de junta.

function [xt,yt,zt]=**transfpa**(x,y,z,TRANSF) Função TRANSFPA move parte (superfície x,y,z) para localização determinada por matriz de transformação homogênea TRANSF, de maneira a obter novas coordenadas de superfície xt,yt,zt.

function y = **vetpm**(x) Retorna matriz equivalente a produto vetorial x = vetor 3x1 , y = matriz 3x3.

function tauvi=**visctorq**(dq) Calcula vetor tauvi de torques de atrito viscoso a partir de velocidade de juntas (Vetor dq).

6.6 Conclusão

Neste Capítulo, foi dada uma descrição do ambiente ROBOTLAB para simulação dinâmica de manipuladores robóticos. As características do mesmo foram apresentadas dando-se destaque para a interface amigável e saída de dados gráfica incluindo animação em 3D. Os diferentes módulos do programa foram descritos sumariamente. O formato dos dados e arquivos do usuário foi apresentado. As possibilidades de saída gráfica, incluindo simulação gráfica em 3D, foram mostradas. Um sumário das funções ROBOTLAB foi incluído.

CAPÍTULO 7

CONCLUSÃO GERAL

7.1 Introdução

Neste Capítulo, é feita uma análise conclusiva do trabalho realizado. Na Seção 7.2, são destacadas as principais contribuições científicas desta tese, junto com as publicações decorrentes da mesma. Na Seção 7.3, são discutidas algumas possíveis linhas de pesquisa a serem seguidas futuramente para dar continuidade ao trabalho realizado. As considerações finais são discutidas na Seção 7.4.

7.2 Contribuições Científicas

No entendimento do autor, a principal contribuição do presente trabalho é a própria idéia do neurocontrolador modular. A partir desta idéia, tornou-se possível estabelecer limites mais claros para o projeto das redes neurais controladoras. A possibilidade de utilizar redes neurais padronizadas de pequenas dimensões, decorrente desta filosofia, permite também diminuir consideravelmente o tempo de aprendizado, o que é importante para aplicações em tempo real. Dentro desta contribuição maior, varias contribuições menores são destacadas a seguir.

No Capítulo 2, foi realizada uma revisão sumária dos principais conceitos teóricos envolvidos no estudo de redes neurais e neurocontroladores. Visando uma familiarização com estes conceitos, no início do trabalho de tese , foram realizados estudos de controle de corrente estatórica de motor de indução, por ser este um sistema não linear relativamente mais simples do que um manipulador robótico. Embora não incluídos nesta tese, já que a mesma está centrada na área de robótica, diversos trabalhos sobre controle de motores de indução foram realizados, resultando em várias publicações nacionais e internacionais: [Alsina & Gehlot, 1992-A], [Gehlot & Alsina,

1993], [Alsina & Gehlot, 1993-C], [Alsina & Gehlot, 1993-D], [Alsina & Gehlot, 1995-B], [Alsina & Gehlot, 1995-E]. Estes trabalhos iniciais permitiram dominar as técnicas de neurocontrole, servindo de base para sua posterior aplicação no controle de manipuladores robóticos.

No Capítulo 3, foram discutidos métodos de cálculo para as equações de dinâmica direta e inversa de manipuladores robóticos. O clássico método Newton-Euler foi apresentado e uma versão equivalente, em termos de vetores 6x1, foi desenvolvida. A relação entre os problemas de simulação e controle de robôs e o cálculo das equações de dinâmica direta e inversa foi destacada. O método de Walker para cálculo da matriz de inércia usando o método Newton-Euler foi descrito. A partir da formulação Newton-Euler, uma representação equivalente, em forma de diagrama de blocos, foi desenvolvida para a matriz de inércia. Com base neste diagrama de blocos, foram desenvolvidos dois métodos recursivos, de ordem N^2, para cálculo da matriz de inércia de manipuladores robóticos: o Método dos Caminhos Inversos e o Método dos Caminhos Centrais. Este último apresentou complexidade computacional menor do que o método de Walker para o cálculo da matriz de inércia de manipuladores redundantes. A partir de manipulações algébricas, uma forma fatorada compacta foi desenvolvida para a matriz de inércia. Assim, as principais contribuições apresentadas no Capítulo 3 são:

a) Representação da Matriz de Inércia por meio de um Diagrama de Blocos.

b) Novo método para cálculo da equação dinâmica e matriz de inércia de manipuladores robóticos: o Método dos Caminhos Inversos.

c) Novo método para cálculo da equação dinâmica e matriz de inércia de manipuladores robóticos: o Método dos Caminhos Centrais.

d) Forma Fatorada da Matriz de Inércia.

Algumas publicações nacionais e internacionais foram decorrentes destas contribuições: [Alsina & Gehlot, 1994-A], [Alsina & Gehlot, 1994-D], [Alsina & Gehlot, 1996-C]. Embora não incluídos nesta tese, foram também realizados estudos sobre cálculo de cinemática inversa de manipuladores robóticos usando redes neurais modulares. Estes estudos paralelos resultaram nas publicações: [Alsina & Gehlot, 1994-E] e [Alsina & Gehlot, 1995-A].

No Capítulo 4, o problema de controle adaptativo baseado em modelo dinâmico para manipuladores robóticos foi abordado. Visando a sua aplicação em controle adaptativo, uma forma Linear-Em-Parâmetros para a equação dinâmica de manipuladores foi desenvolvida. Um método recursivo eficiente, apresentando carga computacional de ordem N^2, foi desenvolvido para o cálculo da matriz de regressores do modelo Linear-Em-Parâmetros de manipuladores robóticos. Um esquema modular para controle adaptativo de manipuladores foi proposto. As equações de controle e adaptação de parâmetros foram reorganizadas em forma recursiva e módulos padronizados de controle e adaptação de parâmetros para cada elo foram propostos. Sugere-se a implementação de cada módulo em um microprocessador dedicado, concatenado com outros módulos por meio de entradas e saídas padronizadas em um sistema de multiprocessadores global, visando o cálculo eficiente do algoritmo de controle em tempo real. As principais contribuições apresentadas no Capítulo 4 são:

a) Método Recursivo para Cálculo da Matriz de Regressores.

b) Esquema Modular para Controle de Manipuladores.

c) Esquema Modular para Adaptação de Parâmetros de Controlador.

Duas publicações resultaram destas contribuições: [Alsina & Gehlot, 1994-C] e [Alsina & Gehlot, 1995-F].

No Capítulo 5 foram desenvolvidas novas técnicas baseadas em redes neurais para controle de trajetória de manipuladores robóticos. A formulação

Newton-Euler foi modificada para uma forma Linear em Termos Cinemáticos. Com base nesta formulação, foi desenvolvido um esquema modular de controle baseado em redes neurais para manipuladores robóticos. Módulos Neurais padronizados foram associados a cada elo. A implementação dos módulos por meio de redes neurais de três camadas foi discutida. Foram estabelecidos limites para o número de entradas, saídas e neurônios da camada oculta dos módulos neurais. A concatenação dos módulos resultou num Neurocontrolador Modular para manipuladores robóticos. Um esquema de treinamento inverso direto para o neurocontrolador modular foi proposto. A partir do controlador adaptativo descrito no Capítulo 4, foi desenvolvido um método de treinamento adaptativo direto para o neurocontrolador modular. Resultados de simulação para um manipulador de três juntas comprovaram o bom desempenho do Neurocontrolador Modular proposto, mesmo em trajetórias não aprendidas previamente. O neurocontrolador modular apresentou desempenho semelhante ao controlador adaptativo modular. Devido aos mecanismos de adaptação, estes dois controladores mostraram desempenho superior do que o controlador Torque Computado padrão. As principais contribuições apresentadas no Capítulo 5 são:

a) Expressão do algoritmo Newton-Euler em forma Linear em Termos Cinemáticos.

b) Neurocontrolador Modular para Manipuladores Robóticos.

c) Esquema de Treinamento Inverso Direto para o Neurocontrolador Modular.

d) Esquema de Treinamento Adaptativo Direto para o Neurocontrolador Modular.

Algumas publicações nacionais e internacionais resultaram destas contribuições: [Alsina & Gehlot, 1992-B], [Alsina & Gehlot, 1993-A], [Alsina & Gehlot, 1993-B], [Alsina & Gehlot, 1994-B], [Alsina & Gehlot, 1995-D].

No Capítulo 6, foi desenvolvido o software ROBOTLAB para simulação dinâmica de manipuladores robóticos. Este software é construído de maneira

modular e aberta, de maneira a que o usuário pode produzir suas próprias bibliotecas, expandindo e melhorando o mesmo. ROBOTLAB apresenta interface amigável através de menus em janelas e resultados gráficos incluindo animação em 3D, o que o tornam um verdadeiro laboratório virtual de robótica. A principal contribuição apresentada no Capítulo 6 é:

a) ROBOTLAB - "Toolbox" de Robótica para MATLAB. Ferramenta para simulação dinâmica de manipuladores com saída gráfica em três dimensões.

Decorrentes do trabalho apresentado no Capítulo 6, as seguintes publicações foram realizadas: [Alsina & Gehlot, 1995-C], [Alsina & Gehlot, 1996-A], [Alsina & Gehlot, 1996-B].

7.3 Perspectivas Futuras

Neste trabalho, foram pesquisados diversos assuntos relativos à robótica, envolvendo as áreas de dinâmica, simulação gráfica, controle adaptativo, e neurocontrole de manipuladores. A seguir são sugeridas algumas possíveis linhas de pesquisa a serem seguidas futuramente para dar continuidade ao trabalho realizado.

A verificação experimental dos controladores desenvolvidos, utilizando um manipulador real, é desejável para a validação das estratégias de controle propostas. Neste sentido, utilizando um manipulador robótico de cinco eixos, modelo RHINO-XR4, disponível atualmente no Laboratório de Robótica, Automação e Visão (DEE-UFPB), estão em andamento os estudos para a implementação em tempo real dos algoritmos de controle propostos.

Estima-se que o desempenho dos controladores propostos possa ser melhorado por meio da sua associação a outras técnicas, tais como lógica nebulosa, algoritmos genéticos, sistemas de estrutura variável, etc. Por outro lado, nesta tese, não foi obtido nenhum resultado teórico a respeito da estabilidade do neurocontrolador

modular proposto. Neste sentido, estudos sobre o uso conjunto do neurocontrolador modular e sistemas de estrutura variável estão em andamento, visando estabelecer teoricamente algumas condições de estabilidade e robustez.

No momento, estão sendo realizados esforços no sentido de melhorar o ambiente ROBOTLAB, de maneira a simular uma célula de trabalho completa. Espera-se no futuro, transformar o ambiente ROBOTLAB numa ferramenta, de interface gráfica amigável, para programação off-line de tarefas de células de trabalho.

A filosofia de controle modular, aqui descrita, contém implicitamente a idéia de implementação em hardware. Sendo assim, um estudo a respeito da implementação dos controladores propostos em sistemas de multiprocessadores seria desejável. Sugere-se, por exemplo, estudar a possibilidade de implementação do esquema modular por meio de N processadores interligados, implementando cada módulo de controle por meio de um processador correspondente. Também, a implementação, por meio de alguma arquitetura paralela, dos métodos propostos para cálculo da equação dinâmica, certamente resultará num menor tempo de processamento, o que é fundamental para aplicações de simulação em tempo real.

7.4 Considerações Finais

Apresentou-se a filosofia de controle neuroadaptativo modular para manipuladores robóticos. Esta filosofia constitui-se numa tentativa de avanço no sentido do desenvolvimento de técnicas sistemáticas de projeto de neurocontroladores para manipuladores robóticos, independentemente do tipo e número de graus de liberdade dos mesmos. Dentro do conhecimento do autor, não foi encontrado nenhum trabalho deste tipo na literatura, sendo que a maioria das publicações se limita ao projeto do neurocontrolador para algum protótipo específico, geralmente com poucos graus de liberdade.

A filosofia de controle modular foi fundamentada, em grande parte, na estrutura recursiva do algoritmo Newton-Euler. Neste contexto, tentou-se unificar a notação e as equações foram expressas de uma maneira compacta e elegante, obtendo-se as formas Linear-Em-Parâmetros e Linear em Termos Cinemáticos, além de um novo método de formulação dinâmica. A representação da equação dinâmica na forma de diagrama de blocos permite visualizar a estrutura da mesma de um modo mais direto e intuitivo, possibilitando novas inspirações para a concepção de formulações dinâmicas mais eficientes.

Discutiu-se a necessidade de técnicas mais sofisticadas para controle de manipuladores robóticos em situações em que as exigências de desempenho são severas e não realizáveis por controladores convencionais. Por outro lado, o aumento na complexidade dos controladores é acompanhado por um aumento na carga computacional, o que pode tornar-se numa limitação para implementações em tempo real. Dentro deste contexto, a filosofia de controle modular procura subsídios para implementações mais simples e eficientes, seja em formas algorítmicas ou em hardware. A implementação em hardware dos módulos neurais, numa estrutura de multiprocessadores, seria o alvo último a ser alcançado pela filosofia de controle neuroadaptativo modular.

REFERÊNCIAS BIBLIOGRÁFICAS

[Agba, 1995], Agba, E. I., "SeaMaster: An ROV-Manipulator System Simulator". IEEE Computer Graphics and Applications, January, 1995, pp. 24-31.

[Alici & Daniel, 1993], Alici, G. & Daniel, R. W., "Experimental Comparison of Model-Based Robot Position Control Strategies". Proceedings of the 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems IROS-93, Yokohama, Japan, July, 1993, pp. 76-83.

[Akaike, 1974], Akaike, H., "A New Look at the Statistical Model Identification". IEEE Transactions on Appl. Comp., Vol. AC-19, 1974, pp. 716-723.

[Alsina & Gehlot, 1992-A], Alsina, P. J. & Gehlot, N. S., "Induction Motor Current Control by Neural Servo Controller using Direct Inverse Control Strategy". Proceedings of the IEEE Emerging Technologies and Factory Automation Conference, ETFA-92, Melbourne, Australia, August, 1992.

[Alsina & Gehlot, 1992-B], Alsina, P. J. & Gehlot, N. S., "A Comparison of Control Strategies of Robotic Manipulators using Neural Networks". Proceedings of the Industrial Electronics Conference IECON-92, San Diego, USA, November, 1992. Vol. 2, pp. 688-693.

[Alsina & Gehlot, 1993-A], Alsina, P. J. & Gehlot, N. S., "A Modular Neurocontroller for N-Link Robot Manipulator based on Newton-Euler Formulation". Proceedings of the International Conference on Intelligent Autonomous Systems, IAS-93, Pittsburgh, USA, February, 1993.

[Alsina & Gehlot, 1993-B], Alsina, P. J. & Gehlot, N. S., "Trajectory Control of N-Link Robot Manipulator based on Modular Neurocontroller". Proceedings of the IEEE/RSJ International Conference on Intelligent Robots & Systems, IROS-93, Yokohama, Japan, July, 1993. Vol. 1, pp. 245-251.

[Alsina & Gehlot, 1993-C], Alsina, P. J. & Gehlot, N. S., "Current Control of Induction Motors based on Input Voltage - Output Current Model using Neural Networks". Proceedings of the IEEE International Conference on Industrial Electronics, Control & Instrumentation, IECON-93, Hawaii, USA, November, 1993.

[Alsina & Gehlot, 1993-D], Alsina, P. J. & Gehlot, N. S., "Torque and Flux Control of Field Oriented Induction Motors based on Neural Networks". Anais do Congresso Brasileiro de Eletrônica de Potência, COBEP-93, Uberlândia, Brasil, Novembro/Dezembro, 1993, pp. 402-406.

[Alsina & Gehlot, 1994-A], Alsina, P. J. & Gehlot, N. S., "An Alternate Formulation of Manipulator Dynamics with Recursive Computation of Inertia Matrix for Computer Simulation". Proceedings of the IEEE International Symposium on Industrial Electronics, ISIE-94, Chile, Vol. 1, May, 1994, pp. 335-338.

[Alsina & Gehlot, 1994-B], Alsina, P. J. & Gehlot, N. S., "Análise Comparativa de Neurocontroladores para Manipulador Robótico". Anais do 10º Congresso Brasileiro de Automática / 6º Congresso Latino Americano de Controle Automático. Rio de Janeiro, Brasil, Setembro, 1994. Vol. 2, pp. 1040-1045.

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

[Alsina & Gehlot, 1994-D], Alsina, P. J. & Gehlot, N. S., "Um Novo Método para Computação de Dinâmica de Manipuladores Robóticos usando Método Recursivo para Cálculo de Matriz de Inércia". Anais do 10º CBA, Rio de Janeiro, Brasil, Vol. 2, Setembro, 1994, pp. 909-914.

[Alsina & Gehlot, 1994-E], Alsina, P. J. & Gehlot, N. S., "Direct and Inverse Kinematics for Robotic Manipulators based on Modular Neural Networks". Proceedings of the International Conference on Automation, Robotics & Computer Vision, ICARCV-94, Singapore, November, 1994. Vol. 3, pp. 1743-1747.

[Alsina & Gehlot, 1995-A], Alsina, P. J. & Gehlot, N. S., "Robot Inverse Kinematics: A Modular Neural Network Approach". Proceedings of the 38th Midwest Symposium on Circuits and Systems, Rio de Janeiro, Brasil, August, 1995.

[Alsina & Gehlot, 1995-B], Alsina, P. J. & Gehlot, N. S., "Adaptive Neurocontroller for the Induction Motor Stator Current". Proceedings of the 38th Midwest Symposium on Circuits and Systems, Rio de Janeiro, Brazil, August, 1995.

[Alsina & Gehlot, 1995-C], Alsina, P. J. & Gehlot, N. S., "ROBOTLAB - Uma Ferramenta para Ensino e Pesquisa em Robótica". Anais da Primeira Amostra de Produção Científica da Universidade Estadual da Paraíba, Campina Grande, Brasil, Outubro, 1995, pp. 83.

[Alsina & Gehlot, 1995-D], Alsina, P. J. & Gehlot, N. S., "Modular Adaptive Neurocontrol of Robotic Manipulators". Proceedings of the IEEE International Conference on Industrial Electronics, Control & Instrumentation, IECON-95, Orlando, USA, November, 1995, pp. 1342-1347.

[Alsina & Gehlot, 1995-E], Alsina, P. J. & Gehlot, N. S., "Neuro-Adaptive Control of Induction Motor Stator Current". Proceedings of the IEEE International Conference on Industrial Electronics, Control & Instrumentation, IECON-95, Orlando, USA, November, 1995, pp. 1434-1439.

[Alsina & Gehlot, 1995-F], Alsina, P. J. & Gehlot, N. S., "Adaptive Controller of Robotic Manipulators: A Modular Approach". Proceedings of the IEEE Control Systems Society 34th Conference on Decision & Control CDC-95, New Orleans, USA, December, 1995, pp. 57-62.

[Alsina & Gehlot, 1996-A], Alsina, P. J. & Gehlot, N. S., "ROBOTLAB - Um Laboratório Virtual de Robótica". Anais do XI Congresso Brasileiro de Automática CBA-96, São Paulo, Brasil, Setembro, 1996, pp. 875-880.

[Alsina & Gehlot, 1996-B], Alsina, P. J. & Gehlot, N. S., "ROBOTLAB: A Graphic Robot Simulator". Paper accepted for publication on the Proceedings of the International Conference on Automation, Robotics & Computer Vision, ICARCV-96, Singapore, December, 1996.

[Alsina & Gehlot, 1996-C], Alsina, P. J. & Gehlot, N. S., "A Block Diagram-Based Computation Method for the Inertia Matrix of Robot Manipulators". Paper accepted for publication on the Proceedings of the International Conference on Automation, Robotics & Computer Vision, ICARCV-96, Singapore, December, 1996.

[Ba-Razzouk, et alii, 1995], Ba-Razzouk, A., Chériti, A., Olivier, G. & Sicard, P., "Field Oriented Control of Induction Motors using Neural Networks Decouplers". Proceedings of the IEEE International Conference on Industrial Electronics, Control & Instrumentation, IECON-95, Orlando, USA, November, 1995, pp. 1428-1433.

[Babu & Eswaran, 1992], Babu, M. & Eswaran, C., "On-Line System Identification using Recurrent Neural Networks". Proceedings of the International Conference in Automation, Robotics & Computer Vision - ICARCV-92, Vol. 1, Singapore, 1992, pp. NW-2.7.1-NW-2.7.5.

[Baharin & Green, 1991], Baharin, I. B. & Green, R. J. "Computationally-effective Recursive Lagrangian Formulation of Manipulation Dynamics". International Journal of Control, Vol. 54, Nº 1, 1991, pp. 195-214.

[Balafoutis & Patel, 1989], Balafoutis, C. A. & Patel, R. V., "Efficient Computation of Manipulator Inertia Matrices and the Direct Dynamics Problem". IEEE Transactions on Systems, Man and Cybernetics, Vol. SMC-19, Nº 5, September/October, 1989, pp. 1313-1321.

[Barnard & Casasent, 1989], Barnard, E. & Casasent, D., "A Comparison between Criterion Functions for Linear Classifiers with an Application to Neural Nets". IEEE Transactions on Systems, Man & Cybernetics, Vol. 19, No. 5, September/October, 1989, pp. 1030-1040.

[Bayard & Wen, 1988], Went, J. T. & Bayard, D. S., "New Class of Control Laws for Robotic Manipulators. Part 2. Adaptive Case". International Journal of Control, Vol. 47, Nº 5, May, 1988, pp. 1387-1406.

[Callan, 1986], Callan, J. F. "The Simulation and Programming of Multiple-Arm Robot System". Robotics Eng. Vol. 8, 1986, pp. 26-29.

[Cavalcanti, 1995], Cavalcanti, J. H. F., "Adaptation and Learning Factor for Neural Controllers". Proceedings of the IEEE International Conference on Industrial

Electronics, Control & Instrumentation, IECON-95, Orlando, USA, November, 1995, pp. 1406-1410.

[Chen & Chen, 1993], Chen, T. & Chen, H., "Aproximations of Continuous Functionals by Neural Networks with Applications to Dynamic Systems". IEEE Transactions on Neural Networks, Vol. 4, No. 6, November, 1993, pp. 910-918.

[Chen & Gill, 1994], Chen, Y. M. & Gill, K. F., "Fuzzy-Neuro Control of Robotic Manipulators". Proceedings of the Third International Conference on Automation, Robotics and Computer Vision, ICARCV-94, Singapore, November, 1994, pp. 1729-1733.

[Chen & Liu, 1994], Chen, F. C. & Liu, C. C., "Adaptively Controlling Nonlinear Continuous-Time Systems using Multilayer Neural Networks". IEEE Transactions on Automatic Control, Vol. 39, No. 6, June, 1994, pp. 1306-1310.

[Chen, et alii, 1990], Chen, S., Billings, S. A. & Grant, P. M., "Non-Linear System Identification using Neural Networks". International Journal of Control, Vol. 51, No. 6, June, 1990, pp. 1191-1214.

[Chen, et alii, 1993], Chen, P. C. Y., Mills, J. K. & Smith, K. C., "On the Dynamics of a Neural Network for Robot Trajectory Tracking". Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS-93, Yokohama, Japan, July, 1993, pp. 252-258.

[Chevasin & Premier, 1984], Chevasin, K. & Premier, R., "Robot Activity Simulation and Plant Testing". ISA Transactions, Vol. 23, No. 4, 1984, pp. 51-54.

[Chin, et alii, 1992], Chin, L., Mital, D. P., Tan, S. S. & Lay, A. S., "Robotic Control Simulations with a Neural Network Controller". Proceedings of the International Conference on Automation, Robotics and Computer Vision, ICARCV-92, Singapore, November, 1992, pp. RO-7.2.1-NW-7.2.9.

[Choi & Choi, 1994], Choi, C. H. & Choi, J. Y., "Construtive Neural Networks with Piecewise Interpolation Capabilities for Function Approximations". IEEE Transactions on Neural Networks, Vol. 5, No. 6, November, 1994, pp. 936-944.

[Colbaugh & Seraji, 1994], Colbaugh, R. & Seraji, H., "Adaptive Tracking Control of Manipulators: Theory and Experiments". Proceedings of the IEEE International Conference on Robotics & Automation ICRA-94, San Diego, USA, 1994, Vol. 4, pp. 2992-2999.

[Connolly & Pfeiffer, 1993], Connolly,T. H. & Pfeiffer, F., "Neural Network Hybrid Position/Force Control". Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS-93, Yokohama, Japan, July, 1993, pp. 240-244.

[Corke, 1996], Corke, P. I., "A Robotics Toolbox for Matlab". IEEE Robotics & Automation Magazine, March, 1996, pp. 24-32.

[Craig, 1986], Craig, J. J., "Introduction to Robotics - Mechanics & Control". Addison-Wesley, 1986.

[Craig, et alii, 1986], Craig., J. J., Hsu, P. & Sastry, S., "Adaptive Control of Mechanical Manipulators". Proceedings of the IEEE International Conference on Robotics & Automation ICRA-86, San Francisco, USA, April, 1986, pp. 190-195.

[Cybenko, 1989], Cybenko, G., "Approximation by Superpositions of a Sigmoidal Function". Math., Contr., Signal, Syst., Vol. 2, No. 4, pp. 303-314, 1989.

[Demin, et alii, 1992], Demin, X., Dehun, J. & Zhang, R., "Stability Analyses of Neural Net Control System". Proceedings of the International Conference in Automation, Robotics & Computer Vision - ICARCV-92, Vol. 2, Singapore, 1992, pp. CO-8.5.1-CO-8.5.4.

[Denavit & Hartenberg, 1955], Denavit, J. & Hartenberg, R. S., "A Kinematic Notation for Lower-Pair Mechanisms based on Matrices". ASME Journal of Applied Mechanics 77, 1955, pp. 215-221.

[Dessaint, et alii, 1990], Dessaint, L. A., Saad, M., Hébert, B. & Gargour, C., "An Adaptive Controller for a Direct-Drive SCARA Robot: Analysis and Simulation". Proceedings of the IEEE International Conference on Industrial Electronics, Control & Instrumentation, IECON-90, Pacific Grove, USA, November, 1995, pp. 414-420.

[Dessaint, et alii, 1992], Dessaint, L. A., Saad, M., Hébert, B. & Al-Haddad, K., "An Adaptive Controller for a Direct-Drive SCARA Robot". IEEE Transactions on Industrial Electronics, Vol. 39, No. 2, April, 1992, pp. 105-111.

[Fang, et alii, 1994], Fang, Y. J., Basu, A. & Fang, X. D., "A New Manipulator Dinamics Modelling Method". Transactions of the Third International Conference on Automation, Robotics & Computer Vision ICARCV-94, Singapore, November, 1994, pp. 967-971.

[Featherstone, 1987], Featherstone, R., "Robot Dynamics Algorithms". Boston Kluwer Academic Publishers, 1987.

[Fijany, 1994], Fijany, A., "Schur Complement Factorizations and Parallel O(Log N) Algorithms for Computation of Operational Space Mass Matrix and Its Inverse". Proceedings of the IEEE International Conference on Robotics & Automation ICRA-94, San Diego, USA, 1994, Vol. 3, pp. 2369-2376.

[Fijany, et alii, 1994], Fijany, A., Sharf, I. & D'Eleuterio, G. M. T., "Parallel O(Log N) Algorithms for the Computation of Manipulator Forward Dynamics". Proceedings of the IEEE International Conference on Robotics & Automation ICRA-94, San Diego, USA, 1994, Vol. 2, pp. 1547-1553.

[Foo, et alii, 1994], Foo, S. S., Chin, L., Ling, K. V. & Chen, L. H., "Fuzzy Neural Controller Design for a Robot Arm". Proceedings of the Third International Conference

on Automation, Robotics and Computer Vision, ICARCV-94, Singapore, November, 1994, pp. 1868-1872.

[Fukuda & Shibata, 1992], Fukuda, T & Shibata, T., "Theory and Applications of Neural Networks for Industrial Control Systems". IEEE Trans. on Industrial Electronics, Vol. 39, No. 6, December, 1992, pp. 472-489.

[Gambardella & Haex, 1993], Gambardella, L. & Haex, M., "Incorporating learning in Motion Planning Techniques". Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS-93, Yokohama, Japan, July, 1993, pp. 712-715.

[Gehlot & Alsina, 1993], Gehlot, N. S. & Alsina, P. J., "A Discrete Model of Induction Motors for Real-Time Control Applications". IEEE Transactions on Industrial Electronics, Vol. 40, No. 3, June, 1993, pp. 317-325.

[Gogoussis & Donat, 1990], Gogoussis, A. & Donath, M., "A Method for the Real Time Solution of the Forward Dynamics Problem for Robots Incorporating Friction". Transactions of the ASME, Vol. 112, December, 1990, pp. 630-639.

[Goh & Tan, 1994], Goh, Y. S. & Tan, E. C., "Experiments using Symmetric Logarithmoid as Activation Function in Multi-Layer Feed-Forward Neural Networks". Proceeding of the Third International Conference on Automation, Robotics & Computer Vision, ICARCV-94, Singapore, November, 1994, pp. 823-825.

[Guez, et alii, 1988], Guez, A., Protopopsecu, V. & Barhen, J., "On the Stability, Storage Capacity, and Design of Nonlinear Continuous Neural Networks". IEEE Transactions on Systems, Man and Cybernetics, Vol. 18, No. 1, January/February, 1988, pp. 80-87.

[Guo & Angeles, 1989], Guo, L. & Angeles, J., "Controller Estimation for the Adaptive Control of Robotic Manipulators". IEEE Transactions on Robotics & Automation, Vol. 5, No. 3, June, 1989, pp. 315-323.

[Guozhong & Yaming, 1991], Guozhong, Z. & Yaming, S., "A Combined Gradient Learning Algorithm for Multi-Layered Neural Networks". Proceedings of the IEEE International Conference on Industrial Electronics, Control & Instrumentation, IECON-91, Kobe, Japan, November, 1991, pp. 1492-1495.

[Hecht-Nielsen, 1990], Hecht-Nielsen, R., "Neurocomputing". Addison-Wesley, 1990.

[Hertz, et alii, 1991], Hertz, J., Krogh, A. & Palmer, R. G., "Introduction to the Theory of Neural Computation". Addison-Wesley, 1991.

[Hollerbach, 1980], Hollerbach, J. M., "A RecursiveLagrangian Formulation of Manipulator Dynamics and Comparative Study of Dynamics Formulation Complexity".IEEE Trans. on Syst., Man and Cybernetics, Vol. SMC-10, No. 11, November, 1980, pp. 730-736.

[Hopfield, 1982], Hopfield, J. J., "Neural Networks and Physical Systems with Emergent Collective Computational Abilities". Proc. Nat. Acad. Sci. USA, Vol. 79, April, 1982, pp. 2554-2558.

[Hsia, et alii, 1991], Hsia, T. C. S, Lasky, T. A. & Guo, Z., "Robust Independent Joint Contrller for Industrial Manipulators". IEEE Transactions on Industrial Electronics, Vol. 38, No. 1, February, 1991, pp. 21-25.

[Hsu & Fu, 1994], Hsu, F. Y. & Fu, L. C., "Adaptive Robust Fuzzy Control for Robotic Manipulators". Proceedings of the IEEE International Conference on Robotics & Automation ICRA-94, San Diego, USA, 1994, Vol. 1, pp. 649-654.

[Huang & Huang, 1991], Huang, S. C. & Huang, Y. F., "Bounds on the Number of Hidden Neurons in Multilayer Perceptrons". IEEE Transactions on Neural Networks, Vol. 2, No. 1, January, 1991, pp. 47-55.

[Iñigo & Morton, 1991], Iñigo, R. M. & Morton, J. S. "Simulation of the Dynamics of an Industrial Robot". IEEE Trans. on Education, Vol. 34, No. 1, February, 1991, pp. 89-99.

[Irie & Miyake, 1988], Irie, B. & Miyake, S., "Capacity of Three-Layered Perceptrons". Proceedings of IEEE ICNN 1, 1988, pp. 641-648.

[Itoh, et alii, 1990], Itoh, Y., Okuma, F. S. & Uchikawa, Y., "Stability Analysis of a Digital Current Controller for a PWM Inverter using Neural Network". Proceedings of the IEEE International Conference on Industrial Electronics, Control & Instrumentation, IECON-90, Pacific Grove, USA, November, 1990, pp. 1129-1134.

[Jain & Rodriguez, 1994], Jain, A. & Rodriguez, G., "Diagonalized Dynamics of Robot Manipulators". Proceedings of the IEEE International Conference on Robotics & Automation ICRA-94, San Diego, USA, 1994, Vol. 1, pp. 334-339.

[Javaheri & Orin, 1988], Javaheri, M. A. & Orin, D. E., "Systolic Architecture for the Manipulator Inertia Matrix". IEEE Transactions on Systems, Man and Cybernetics, Vol. SMC-18, N° 6, November/December, 1988, pp. 939-951.

[Karakasoglu, et alii, 1993], Karakasoglu, A., Sudharsanan & Sundareshan, K., "Identification and Decentralized Adaptive Control using Dynamical Neural Networks with Application to Robotics Manipulators". IEEE Transactions on Neural Networks, Vol. 4, No. 6, November, 1993, pp. 919-930.

[Kawasaki, et alii, 1992], Kawasaki, H., Murata, A. & Kanzaki, K., "An Efficient Method for the Computation of Inverse Dynamics and Inertia Matrix based on the Minimum Set of Dynamics Parameters of Manipulators". Proceedings of the International Conference on Automation, Robotics & Computer Vision, ICARCV-92, Singapore, 1992, Vol. 3, pp. RO-1.2.1 - RO-1.2.5.

[Kelly, et alii, 1989], Kelly, R., Carelli, R. & Ortega, R., "Adaptive Motion Control Design of Robot Manipulators: an Input-Output Approach". International Journal of Control, Vol. 50, Nº 6, December, 1989, pp. 2563-2581.

[Khan, 1991], Khan, E., "Neural Net Architectures to convert Existing Servo Controllers into Intelligent Adaptive Controllers". Proceedings of the IEEE International Conference on Industrial Electronics, Control & Instrumentation, IECON-91, Kobe, Japan, November, 1991, pp. 1459-1462.

[Khosla, 1989], Khosla, P. K. "Categorization of Parameters in the Dynamic Robot Model". IEEE Transactions on Robotics & Automation, Vol. 5, No. 3, June, 1989, pp. 261-268.

[Kim & Hori, 1995], Kim, K. & Hori, Y., "Experimental Evaluation of Adaptive and Robust Schemes for Robot Manipulator Control". Proceedings of the IEEE International Conference on Industrial Electronics, Control & Instrumentation, IECON-95, Orlando, USA, November, 1995, pp. 1248-1253.

[Kim, et alii, 1993], Kim, S. H., Kim, Y. H., Sim, K. B. & Jeon, H. T., "On Developing an Adaptive Neural-Fuzzy Control System". Proceedings of the 1993 IEEE/RSJ International Conference on Intelligent Robots & Systems, IROS-93, Yokohama, Japan, July, 1993, pp. 950-957.

[Ko, 1992], Ko, K., "Manipulator Dynamics based on Stochastic System". Proceedings of the International Conference on Automation, Robotics & Computer Vision, ICARCV-92, Singapore, 1992, Vol. 3, pp. RO-2.5.6 - RO-2.5.6.

[Kolmogorov, 1957], Kolmogorov, A. N., "On the Representation of Continuous Functions". Dokl.Akad. Nauk., USSR, 114, 1957.

[Kovács, 1996], Kovács, Z. L., "Redes Neurais Artificiais - Fundamentos e Aplicações". Edição Acadêmica, São Paulo, Brasil, 1996.

[Kung & Hwang, 1989], Kung, S. Y. & Hwang, J. N., "Neural Network Architectures for Robotic Applications". IEEE Transactions on Robotics and Automation, Vol. 5, No. 5, October, 1989, pp. 641-657.

[Kuperstein & Wang, 1990], Kuperstein, M. & Wang, J., "Neural Controller for Adaptive Movements with Unforessen Payloads". IEEE Transactions on Neural Networks, Vol.1, No. 1, March, 1990, pp. 137-142.

[Kwon, et alii, 1992], Kwon, H. Y., Kim, B. C., Kim, D. K. & Hwang, H. Y., "Backpropagation Learning Improvement and its Effects on Recall Capability". Proceedings of the International Conference on Automation, Robotics & Computer Vision, ICARCV-92, Singapore, November, 1992, Vol. 1, pp. NW-1.6.1-NW-1.6.5.

[Lee & Chang, 1988], Lee, C. S. G. & Chang, P. R., "Efficient Parallel Algorithms for Robot Forward Dynamics Computation". IEEE Transactions on Systems, Man and Cybernetics, Vol. SMC-18, Nº 2, March/April, 1988, pp. 238-251.

[Lee & Chen, 1990], Lee, C. S. G. & Chen, C. L., "Efficient Mapping Algorithms for Scheduling Robot Inverse Dynamics Computation on a Multiprocessor System". IEEE Transactions on Systems, Man and Cybernetics, Vol. SMC-20, N° 3, May/June, 1990, pp. 582-595.

[Lee, 1993], Lee, J., "An Initialization Method for Multilayer Neural Networks as Function Approximators". Proceedings of the IEEE/RSJ International Conference on Intelligent Robots & Systems, IROS-93, Yokohama, Japan, July, 1993, pp. 1725-1728.

[Li, 1988], Li, C. J. "A New Method of Dynamics for Robot Manipulators". IEEE Transactions on Systems, Man and Cybernetics, Vol. SMC-18, N° 1, January/February, 1988, pp. 105-114.

[Li, 1989], Li, C. J., "An Efficient Method for Linearization of Dynamic Models of Robot Manipulators". IEEE Transactions on Robotics and Automation, Vol. 5, No. 4, August, 1989, pp. 397-408.

[Liang & ElMaraghi, 1994], Liang, F. & ElMaraghi, H. A., "Direct adaptive Neurocontrol of Flexible Joint Robots using Localized Polynomial Networks". Proceedings of the IEEE International Conference on Robotics & Automation, ICRA-94, San Diego, USA, Vol. 4, 1994, pp. 3186-3191.

[Lilly & Orin, 1991], Lilly, K. W. & Orin, D. E., "Alternate Formulations for the Manipulator Inertia Matrix". International Journal of Robotic Research, Vol. 10, N° 1, 1991, pp. 64-74.

[Liu & Lin, 1988], Liu, M. H. & Lin, W., "Multivariable Self-Tuning Control with Decoupling for Robotic Manipulators". IEEE Proceeedings, Vol. 135, Pt. D, No. 1, January, 1988, pp. 43-48.

[Low, et alii, 1993], Low, T. S., Lee, T. H. & Lim, H. K., "A Methodology for Neural Network Training for Control of Drives with Nonlinearities". IEEE Transactions on Industrial Electronics, Vol. 39, No. 2, April, 1993, pp. 243-249.

[Luh, et alii, 1980], Luh, J. Y. S., Walker, M. W. & Paul, R. P. C., "On-Line Computational Scheme for Mechanical Manipulators". ASME Journal of Dynamic Systems, Measurement and Control, Vol. 102, June, 1980, pp. 69-76.

[Martinetz, et alii, 1990], Martinetz, T. M., Ritter, H. J. & Schulten, K. J., "Three-Dimensional Neural Net for Learning Visuomotor Coordination of a Robot Arm". IEEE Transactions on Neural Networks, Vol. 1, No. 1, March, 1990, pp. 131-136.

[Mester, 1995], Mester, G., "Neuro-Fuzzy Genetic Controller Design for Robot Manipulators". Proceedings of the IEEE International Conference on Industrial Electronics, Control & Instrumentation, IECON-95, Orlando, USA, November, 1995, pp. 87-92.

[Mistry & Nair, 1994], Mistry, S. I. & Nair, S. S., "Identification and Control Experiments using Neural Designs". IEEE Control Systems, June, 1994, pp. 48-77.

[Moler, et alii, 1987], Moler, C., Little, J. & Bangert, S., "PC-MATLAB for MS-DOS Personal Computers - User's Guide". The Math Works, Inc., 1987.

[Mukhopadhyay & Narendra, 1993], Mukhopadhyay, S. & Narendra, K. S., "Disturbance Rejection in Nonlinear Systems using Neural Networks". IEEE Transactions on Neural Networks, Vol. 4, No. 1, January, 1993, pp. 63-72.

[Murata, et alii, 1994], Murata, N.,Yoshizawa S. & Amarai, S., "Network Information Criterion - Determining the Number of Hidden Units for an Artificial Neural Network Model". IEEE Transactions on Neural Networks, Vol. 5, No. 6, November, 1994, pp. 865-872.

[Nakamura & Ghoudoussi, 1989], Nakamura, Y. & Ghoudoussi, M., "Dynamics Computation of Closed-Link Robot Mechanisms with Nonredundant and Redundant Actuators". IEEE Transactions on Robotics & Automation, Vol. 5, No. 3, June, 1989, pp. 294-302.

[Narendra & Parthasarathy, 1990], Narendra, K. S. & Parthasarathy, K., "Identification and Control of Dynamical Systems using Neural Networks". IEEE Transactions on Neural Networks, Vol. 1, No. 1, March, 1990, pp. 4-27.

[Narendra & Parthasarathy, 1991], Narendra, K. S. & Parthasarathy, K., "Gradient Methods for the Optimization of Dynamical Systems containing Neural Networks". IEEE Transactions on Neural Networks, Vol. 2, No. 2, March, 1991, pp. 252-262.

[Nethery & Spong, 1994], Nethery, J. F. & Spong, M. W., "Robotica: A Mathematica Package for Robot Analysis". IEEE Robotics & Automation Magazine, March, 1994, pp. 13-20.

[Nguyen & Widrow, 1991], Nguyen, D. H. & Widrow, B., "Neural Networks for Self-Learning Control Systems". International Journal of Control, Vol. 54, No. 6, 1991, pp. 1439-1451.

[Ozaki, et alii, 1991], Ozaki, T., Suzuki, T., Furuhashi, T., Okuma, S. & Uchikawa, Y., "Trajectory Control of Robotic Manipulators using Neural Networks". IEEE Transactions on Industrial Electronics, Vol. 38, No. 3, June, 1991, pp. 195-202.

[Park & Borow, 1994], Park, F. C. & Borrow, J. E., "A Recursive Algorithm for Robot Dynamics using Lie Groups". Proceedings of the IEEE International Conference on Robotics & Automation ICRA-94, San Diego, USA, 1994, Vol. 2, pp. 1535-1540.

[Patton, 1995], Patton, J. B., "Brushless DC Motor Control using a General Regression Neural Network". Proceedings of the IEEE International Conference on Industrial Electronics, Control & Instrumentation, IECON-95, Orlando, USA, November, 1995, pp. 1422-1427.

[Pei, et alii, 1992], Pei, H. L., Leung, T. P. & Zhou, Q. J., "Neural Network Robot Control Methods". Proceedings of the International Conference on Automation, Robotics and Computer Vision, ICARCV-92, Singapore, November, 1992, pp. NW-3.1.1-NW-3.1.5.

[Pinheiro, et alii, 1995], Pinheiro, H., Joós, G. & Khorasani, K., "Neural Network-Based Controllers for Voltage Source PWM Front End Rectifiers". Proceedings of the IEEE International Conference on Industrial Electronics, Control & Instrumentation, IECON-95, Orlando, USA, November, 1995, pp. 488-493.

[Pletl, 1995], Pletl, S., "Neuro-Fuzzy Control of Rigid and Flexible-Joint Robotic Manipulator". Proceedings of the IEEE International Conference on Industrial Electronics, Control & Instrumentation, IECON-95, Orlando, USA, November, 1995, pp. 93-97.

[Pu & Eldin, 1994], Pu, H. J. & Eldin, A. N., "A Novel Parallel Forward Newton-Euler Algorithm for Computing the Robot Inverse Dynamics". Transactions of the Third International Conference on Automation, Robotics & Computer Vision ICARCV-94, Singapore, November, 1994, pp. 977-981.

[Qing, 1992], Qing, S., "Neural Network Application to Discrete Time Robust Controllers". Proceedings of the International Conference in Automation, Robotics & Computer Vision - ICARCV-92, Vol. 2, Singapore, 1992, pp. CO-8.9.1-CO-8.9.5.

[Rangwala & Dornfeld, 1989], Rangwala, S. S. & Dornfeld, D. A., "Learning and Optimization of Machining Operation using Computing Abilities of Neural Networks". IEEE Transactions on Systems, Man & Cybernetics, Vol. 1, No. 2, March/April, 1989, pp. 299-314.

[Raz, 1989], Raz, T., "Graphics Robot Simulator for Teaching Introductory Robotics". IEEE Transactions on Education, Vol. 32, May, 1989, pp. 153-159.

[Reed & Iannou, 1989], Reed, J. S. & Iannou, P. A., "Instability Analysis and Robust Adaptive Control of Robotic Manipulators". IEEE Transactions on Robotics & Automation, Vol. 5, No. 3, June, 1989, pp. 381-386.

[Rosenblatt, 1962], Rossenblatt, F., "The Principles of Neurodynamics". Spartan, 1962.

[Rumelhart, et alii, 1986], Rumelhart, D. E., Hinton, G. E. & Williams, R. J., "Learning Internal Representations by Error Propagation". In *Parallel Distributed Processing*, Vol. 1, Ch. 8, Rumelhart, D. E. & McClelland, J. L., Eds., Cambridge, MA: M.I.T. Press, 1986.

[Saad, et alii, 1994], Saad, M., Bigras, P., Dessaint, L. A. & Al-Haddad, K., "Adaptive Robot Control using Neural Networks". IEEE Transactions on Industrial Electronics, Vol. 40, No. 2, April, 1994, pp. 173-181.

[Sadegh, 1993], Sadeh, N., "A Perceptron Network for Functional Identification and Control of Nonlinear Systems". IEEE Transactions on Neural Networks, Vol. 4, No. 6, November, 1993, pp. 982-988.

[Sadjadi, et alii, 1993], Sadjadi, M. R. A., Sheedvash, S. & Trujillo, S. O., "Recursive Dynamic Node Creation in Multilayer Neural Networks". IEEE Transactions on Neural Networks, Vol. 4, No. 2, March, 1993, pp. 242-256.

[Sartori & Antsaklis, 1991], Sartori, M. A. & Antsaklis, P. J., "A Simple Method to Derive Bounds on the Size and to Train Multilayer Neural Networks". IEEE Transactions on Neural Networks, Vol. 2, No. 4, July, 1991, pp. 467-471.

[Schroer & Tech, 1986], Schroer, B. J. & Tech, W., "A Graphic Simulation Tool with off-line Robot Programming". Simulation, Vol. 47, August, 1986, pp. 63-67.

[Seraji, 1989], Seraji, H., "Decentralized Adaptive Control of Manipulators: Theory, Simulation, and Experimentation". IEEE Transactions on Robotics & Automation, Vol. 5, No. 2, April, 1989, pp. 183-201.

[Sharifi, et alii, 1994], Sharifi, F. J., Fakhry, H. H. H. & Wilson, W. J., "Integration of a Robust Trajectory Planner with a Feedforward Neural Controller for Robotic Manipulators". Proceedings of the IEEE International Conference on Robotics & Automation, ICRA-94, San Diego, USA, Vol. 4, 1994, pp. 3192-3197.

[Sira-Ramirez & Zak, 1991], Sira-Ramirez, H. J. & Zak, S. H., "The Adaptation of Perceptron with Applications to Inverse Dynamics Identification of Unknown DynamicSystems". IEEEE Transactions on Systems, Man and Cybernetics, Vol. 21, No. 3, May/June, 1991, pp. 634-642.

[Slotine & Li, 1988], Slotine, J. J. E. & Li, W., "Adaptive Manipulator Control: A Case Study". IEEE Transactions on Automatic Control, Vol. 33, No. 11, November, 1988, pp.995-1003.

[Song, 1994], Song, Y. D., "Adaptive Motion Tracking Control of Robot Manipulators - Non-Regressor Based Approach". Proceedings of the IEEE International Conference on Robotics & Automation ICRA-94, San Diego, USA, 1994, Vol. 4, pp. 3008-3013.

[Song, et alii, 1991], Song, J. W., Lee, K. C., Cho, K. B. & Won, J. S., "An Adaptive Learning Current Controller for Field Oriented Controlled Induction Motor by Neural Network". Proceedings of the IEEE International Conference on Industrial Electronics, Control & Instrumentation, IECON-91, Kobe, Japan, November, 1991, pp. 469-474.

[Stepanenko & Su, 1994], Stepanenko, Y. & Su, C. Y., "Adaptive Motion Control of Rigid-Link Electrically Driven Robot manipulators". Proceedings of the IEEE International Conference on Robotics & Automation ICRA-94, San Diego, USA, 1994, Vol. 1, pp. 630-635.

[Strefezza & Dote, 1991], Strefezza, M. & Dote, Y. "Fuzzy and neural Networks Controller". Proceedings of the IEEE International Conference on Industrial

Electronics, Control & Instrumentation, IECON-91, Kobe, Japan, November, 1991, pp. 1438-1442.

[Swiniarski, 1990-A], Swiniarski, R. W., "Neural Network based Self-Tuning PID Controller with Fourier Transformation of Temporal Patterns". Proceedings of the IEEE International Conference on Industrial Electronics, Control & Instrumentation, IECON-90, Pacific Grove, USA, November, 1990, pp. 1227-1232.

[Swiniarski, 1990-B], Swiniarski, R. W., "Neuromorphic Fuzzy Variable Structure Controller". Proceedings of the IEEE International Conference on Industrial Electronics, Control & Instrumentation, IECON-90, Pacific Grove, USA, November, 1990, pp. 1221-1226.

[Tai, et alii, 1990], Tai, H. M., Wang, J. & Ashenayi, K., "Motor Speed Regulation using Neural Networks". Proceedings of the IEEE International Conference on Industrial Electronics, Control & Instrumentation, IECON-90, Pacific Grove, USA, November, 1990, pp. 1215-1220.

[Tanomaru & Omatu, 1991], Tanomaru, J. & Omatu, S., "Towards effective Neuromorphic Controllers". Proceedings of the IEEE International Conference on Industrial Electronics, Control & Instrumentation, IECON-91, Kobe, Japan, November, 1991, pp. 1395-1400.

[Tanomaru & Omatu, 1992], Tanomaru, J. & Omatu, S., "Process Control by On-Line Trained Neural Controllers". IEEE Trans. on Industrial Electronics, Vol. 39, No. 6, December, 1992, pp. 511-521.

[Temeltas & Asher, 1992], Temeltas, H. & Asher, G. M., "Implementation of Parallel Structures for Robot Dynamic Model Identification on a Transputer Network". Proceedings of the International Conference on Automation, Robotics & Computer Vision, ICARCV-92, Singapore, 1992, Vol. 3, pp. RO-2.4.1 - RO-2.4.5.

[Uicker, 1967], Uicker, J. J., "Dynamic Force Analysis of Spatial Linkage". ASME Journal of Applied Mechanics 34, 1967, pp. 418-424.

[Vega, et alii, 1994], Vega, V. P., Liu, Y. H. & Arimoto, S., "Variable Structure Robot Control Undergoing Chattering Attenuation: Adaptive and Nonadaptive Cases". Proceedings of the IEEE International Conference on Robotics & Automation ICRA-94, San Diego, USA, 1994, Vol. 3, pp. 1824-1829.

[Venkatesh, 1992], Venkatesh, S. S., "Robustness in Neural Computation: Random Graphs and Sparsity". IEEE Transactions on Information Theory, Vol. 38, no. 3, May, 1992, pp. 1114-1119.

[Villiers & Barnard, 1993], Villiers, J. de & Barnard, E., "Backpropagation Neural Nets with One and Two Hidden Layers". IEEE Transactions on Neural Networks, Vol. 4, No. 1, January, 1993, pp. 136-141.

[Walker & Orin, 1982], Walker, M. W. & Orin, D. E., "Efficient Dynamic Computer Simulation of Robot Mechanisms". ASME Journal of Dynamic Systems, Measurement and Control 104, 1982, pp. 205-211.

[Walter & Schulten, 1993], Walter, J. A. & Schulten, K. J., "Implementation of Self-Organizing Neural Networks for Visuo-Motor Control of an Industrial Robot". IEEE Transactions on Neural Networks, Vol. 4, No. 1, January, 1993, pp. 86-95.

[Wang, 1994], Wang, J., "The Propagation and Learning of Derivative Signals in Neural Networks: Single Variable Mapping Case". ". Proceedings of the International Conference on Automation, Robotics & Computer Vision, ICARCV-94, Singapore, November, 1994, pp. 818-822.

[Werbos, 1989], Werbos, P. J., "Neural Networks for Control and System Identification". Proceedings of the 28-th CDC, Tampa, USA, 1989.

[Werbos, 1993], Werbos, P. J., "Neurocontrol and Elastic Fuzzy Logic: Cappabilities, Concepts, and Applications". IEEE Transactions on Industrial Electronics, Vol. 40, No. 2, April, 1993, pp. 170-180.

[Whitcomb, et alii, 1993], Whitcomb, L. L., Rizzi, A. A. & Koditschek, D. E., "Comparative Experiments with a New Adaptive Controller for Robot Arms". IEEE Transactions on Robotics and Automation, Vol. 9, No. 3, February, 1993.

[White, et alii, 1989], White, R. B., Read, R. K., Koch, M. W. & Schilling, R. J. , "A Graphics Simulator for a Robotic Arm". IEEE Transactions on Education, Vol. 32, No. 4, November, 1989, pp. 417-429.

[Widrow & Lehr, 1990], Widrow, B. & Lehr, M. A., "30 Years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation". Proceedings. of the IEEE, Vol. 78, No. 9, September, 1990, pp.1415-1441.

[Widrow, 1962], Widrow, B., "Generalization and Information Storage in Networks of Adaline Neurons". Self-Organizing Systems, M. C. Yovits, Ed. Wash. DC: Spartan, 1962, pp. 435-461.

[Willis, et alii, 1991], Willis, M. J., Di Massino, C., Montague, G. A., Tham, M. T. & Morris, A. J., "Artificial Neural Networks in Process Engineering". IEEE Proceedings-D, Vol. 138, No. 3, May, 1991, pp. 256-266.

[Yamada & Yabuta, 1990], Yamada, T. & Yabuta, T., "Nonlinear Neural Network Controller for Dynamic System". Proceedings of the IEEE International Conference in Industrial Electronics, Control & Instrumentation, IECON-90, Pacific Grove, USA, November, 1990, pp1245-1249.

[Yamada & Yabuta, 1991], Yamada, T. & Yabuta, T., "Remarks on an Adaptive Type Self-Tuning Controller using Neural Networks". Proceedings of the IEEE International Conference on Industrial Electronics Control & Instrumentation, IECON-91, Kobe, Japan, November, 1991, pp. 1389-1394.

[Yamada & Yabuta, 1993], Yamada, T. & Yabuta, T., "Application of Learning Type Feedforward Feedback Neural Network Controller to Dynamic Systems". Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems - IROS-93, Yokohama, Japan, July, 1993, pp. 225-231.

[Yao & Tomizuka, 1994], Yao, B. & Tomizuka, M., "Robust Desired Compensation Adaptive Control of Robot Manipulators with Guaranteed Transient Performance". Proceedings of the IEEE International Conference on Robotics & Automation ICRA-94, San Diego, USA, 1994, Vol. 3, pp. 1830-1836.

[Yoo, et alii, 1991], Yoo, D. S., Chung, M. J. & Bien, Z., "Real-Time Implementation and Evaluation of Dynamic Control Algorithms for Industrial Manipulators." IEEE Transactions on Industrial Electronics, Vol. 38, No. 1, February, 1991, pp. 26-31.

[Yuan, et alii, 1994], Yuan, M., Hong, G. S., & Poo, A. N., "Neural Adaptive Controller: Application to Robot Manipulator". Proceedings of the Third Int. Conference on Automation, Robotics and Computer Vision, ICARCV-94, Singapore, November, 1994, pp. 1734-1737.

[Zeng, 1994], Zeng, G., "Robot Manipulator Robust-Tracking Control based on Artificial Neural Network Compensation". Proceedings of the Third International Conference on Automation, Robotics and Computer Vision, ICARCV-94, Singapore, November, 1994, pp. 1752-1755.

[Zhang, et alii, 1992], Zhang, R., Dehong, J. & Demin, X., "A New Method based on NLN for Designing the High Order Time-Optimal Control System". Proceedings of the International Conference in Automation, Robotics & Computer Vision - ICARCV-92, Vol. 1, Singapore, 1992, pp. NW-3.6.1-NW-3.6.4.

[Zhihong, et alii, 1994], Zhihong, M. Paplinski, A. P. & Wu, W. R., "A Slidding Mode Control Scheme using Neural Networks for Robotic Manipulators". Proceedings of the Third International Conference on Automation, Robotics and Computer Vision, ICARCV-94, Singapore, November, 1994, pp. 1738-1742.

[Zomaya, 1991], Zomaya, A. Y., "Efficient Robot Dynamics for High Sampling Rate Motions: Case Studies and Benchmarks". International Journal of Control, Vol. 54, N° 4, 1991. pp. 793-814.

[Zomaya, 1993], Zomaya, A. Y., "Trends in Neuro-Adaptive Control for Robot Manipulators". Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS-93, Yokohama, Japan, July, 1993, pp. 754-759.

APÊNDICE I

ARQUIVO DE PARÂMETROS - ROBÔ CAMPINA

```
% Braço de robô CAMPINA - 3 graus de liberdade

% Parâmetros Cinemáticos

d1=0.5;
a2=0.4;
a3=0.3;
alf1=pi/2;
Ltool=0.1;

KINPAR=[rot   0   0   d1   0
        rot   0   alf1 0   0
        rot   a2  0   0    0];

TOOLKINPAR=[tool  (a3+Ltool) 0  0  0];

% Parametros Dinâmicos

m1=3.0;
m2=1.8;
m3=1.5;

DYNPAR=[m1  0  0 -m1*d1/2  m3*d1*d1/3  m3*d1*d1/3 0  0  0  0  0.01  0.001
        m2  m2*a2/2  0  0  0  m2*a2*a2/3  m2*a2*a2/3 0  0  0  0.01  0.001
        m3  m3*a3/2  0  0  0  m3*a3*a3/3  m3*a3*a3/3 0  0  0  0.01  0.001];

% LINK 1

h11=0.1;
r11=[0.1 0.1 0.075];

P01L01=[3 h11 r11];
TP01L01=[eye(3) [0;0;-0.5]];

h21=0.3;
r21=[0.075 0.05*sqrt(2)];

P02L01=[3 h21 r21];
TP02L01=[eye(3) [0;0;-0.4]];

hz31=[0 0.01 0.125 0.15];
sx31=[0.1 0.1 0.09 0.05];
```

```
sy31=[0.01 0.01 0.01 0.01];

P03L01=[4 hz31 sx31 sy31];
TP03L01=[eye(3) [0;-0.045;-0.1]];

hz41=[0 0.01 0.125 0.15];
sx41=[0.1 0.1 0.09 0.05];
sy41=[0.01 0.01 0.01 0.01];

P04L01=[4 hz41 sx41 sy41];
TP04L01=[eye(3) [0;0.045;-0.1]];

% LINK 2

h12=0.07;
r12=[0.035 0.035];

P01L02=[3 h12 r12];
TP01L02=[eye(3) [0;0;0.05]];

hz22=[0 0.025 0.05 0.45 0.375 0.5];
sx22=[0.01 0.01 0.01 0.01 0.01 0.01];
sy22=[0.05 0.09 0.1 0.1 0.09 0.05];

P02L02=[4 hz22 sx22 sy22];
TP02L02=[rotmaty(pi/2) [-0.05;0;0.035]];

P03L02=P02L02;
TP03L02=[rotmaty(pi/2) [-0.05;0;-0.035]];

h42=0.08;
r42=[0.025 0.025];

P04L02=[3 h42 r42];
TP04L02=[eye(3) [0;0;-0.04]];

% LINK 3

P01L03=P01L02;
TP01L03=[eye(3) [0;0;0.04]];

hz23=[0 0.025 0.05 0.35 0.3625 0.375];
sx23=[0.01 0.01 0.01 0.01 0.01 0.01];
sy23=[0.05 0.09 0.1 0.05 0.045 0.025];

P02L03=[4 hz23 sx23 sy23];
TP02L03=[rotmaty(pi/2) [-0.05;0;0.025]];

P03L03=P02L03;
```

```
TP03L03=[rotmaty(pi/2) [-0.05;0;-0.025]];

h43=0.06;
r43=[0.025 0.025];

P04L03=[3 h43 r43];
TP04L03=[eye(3) [0;0;-0.03]];

h53=0.06;
r53=[0.0125 0.0125];

P05L03=[3 h53 r53];
TP05L03=[eye(3) [0.3;0;-0.03]];

h63=0.05;
r63=[0.0125 0.0125];

P06L03=[3 h63 r63];
TP06L03=[rotmaty(pi/2) [0.3;0;0]];

hz73=[-0.02 -0.02 0.02 0.02];
sx73=[0 0.01 0.01 0];
sy73=[0 0.03 0.03 0.0];

P07L03=[4 hz73 sx73 sy73];
TP07L03=[eye(3) [0.35;0;0]];

hz83=[0.0 0.04];
sx83=[0.01 0.01];
sy83=[0.03 0.0];

P08L03=[4 hz83 sx83 sy83];
TP08L03=[rotmaty(pi/2) [0.36;0;0.015]];

P09L03=P08L03;
TP09L03=[rotmaty(pi/2) [0.36;0;-0.015]];
```