TESE DEFENDIDA POR MARCO ANTONIO ARCOS CAMARGO E APROVADA PELA COMISSÃO JULGADORA EM 20 / 08 / 2002

ORIENTADOR

UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA MECÂNICA

Planejamento de Trajetórias de um Manipulador Robótico Usando Redes Neurais Artificiais

Autor : **Marco Antonio Arcos Camargo**
Orientador: **Eurípedes Guilherme de Oliveira Nóbrega**

08/2002

UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA MECÂNICA
DEPARTAMENTO DE MECÂNICA COMPUTACIONAL

Planejamento de Trajetórias de um Manipulador Robótico Usando Redes Neurais Artificiais

Autor : **Marco Antonio Arcos Camargo**
Orientador: **Eurípedes Guilherme de oliveira Nóbrega**

Curso: Engenharia Mecânica.
Área de concentração: Mecânica dos Sólidos e Projeto Mecânico

Dissertação de mestrado apresentada à comissão de Pós Graduação da Faculdade de Engenharia Mecânica, como requisito para obtenção do título de Mestre em Engenharia Mecânica.

Campinas, 2002
S.P. - Brasil

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

Ar27p

Arcos Camargo,Marco Antonio
Planejamento de trajetórias de um manipulador robótico usando redes neurais artificiais / Marco Antonio Arcos Camargo.--Campinas, SP: [s.n.], 2002.

Orientador: Eurípedes Guilherme de Oliveira Nóbrega.
Dissertação (mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Mecânica.

1. Robótica. 2. Redes neurais. 3. Trajetórias espaciais. 4. Trabalho - Planejamento. 5. Manipuladores (Mecanismo). I. Nóbrega, Eurípedes Guilherme de Olivera. II. Universidade Estadual de Campinas. Faculdade de Engenharia Mecânica. III. Título.

UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA MECÂNICA
DEPARTAMENTO DE MECÂNICA COMPUTACIONAL

DISSERTAÇÃO DE MESTRADO

Planejamento de Trajetórias de um Manipulador Robótico Usando Redes Neurais Artificiais

Autor : **Marco Antonio Arcos Camargo**
Orientador: **Eurípedes Guilherme de oliveira Nóbrega**

Prof. Dr. Eurípedes G. O. Nóbrega, Presidente
FEM/UNICAMP

Prof. Dr. Pablo Siqueira Meirelles
FEM/UNICAMP

Prof. Dr. Márcio Luiz Andrade Netto
FEEC/Unicamp

Campinas, 20 de Agosto de 2002

Dedicatória:

Dedico este trabalho aos meus pais, Andres e Flora, aos meus irmãos Mario, Fredy Carlos, Roxana e Nancy, pelo amor, pela ajuda, compreensão e incentivo emocional que sempre me brindaram, em forma muito especial ao meu irmão Mario e família por todo o apoio incondicional prestado o qual foi importante e determinante na realização deste trabalho.

Agradecimentos

Agradeço infinitamente a Deus, pela inteligência, pela saúde, pela paz, e principalmente por jamais ter deixado que eu perdesse a fé em buscar e realizar meus sonhos.

Aos meus pais, Andres e Flora, pelo amor, carinho, cuidado, educação, dedicação e apoio em todas as etapas da minha vida, sempre confiando, acreditando e sempre sonhando comigo.

Aos meus irmaos Mario e familia, Fredy e familia, Carlos, Roxana e Nancy, um agradecimento especial ao meu irmão Carlos sobretudo nos primeiros dias aqui em Campinas, pela ajuda, pelos conselhos, pelos cuidados e apoio brindado.

Ao meu orientador prof. Eurípedes pela orientação e paciência que teve para comigo ao longo destes anos, na realização deste trabalho.

Aos meus amigos, Blanca, Jorge Dias e família, Yurilev , Justo, Mari, Rossana, Abdon, John, Carlos Salinas, Roberto e Enith, pela grande amizade que me brindaram e pelos gratos e inesquecíveis momentos que tivemos juntos, os quais me ajudaram a superar as saudades de casa.

Ao meu amigo e colega de trabalho, André Bezerra pela grande ajuda e contribuição para a realização deste trabalho.

Aos meus amigos de casa, Jean, Kleber, Juliano e Jarbas pela agradável convivência.

Aos professores, funcionários, amigos e colegas do DMC e a todos que contribuíram para este trabalho, meus agradecimentos.

E finalmente ao povo brasileiro por ter me dado a oportunidade de realizar este trabalho.

"Quem quiser alcançar um objetivo distante tem de dar muitos passos curtos."

Helmut Schmidt

Resumo

Arcos Camargo, Marco Antonio: *Planejamento de trajetórias de um manipulador robótico usando redes neurais artificiais*. Campinas,: Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas,2002. Nro p.106. Dissertação (Mestrado).

O objetivo desta dissertação é apresentar o controle da trajetória de um robô manipulador tipo SCARA dentro de um volume de trabalho que é definido pelas equações cinemáticas. A determinação da trajetória dentro do volume de trabalho é obtida através do uso de redes neurais artificiais do tipo *perceptron* de múltiplas camadas que simulam os movimentos do manipulador entre dois pontos quaisquer dentro deste volume. As posições angulares dos segmentos são responsáveis pela geração dos movimentos do manipulador. Algumas simulações dos movimentos são apresentadas mostrando o comportamento da rede neural artificial na fronteira do volume e em configurações onde os modelos geométricos tradicionais, geralmente, apresentam singularidades. As redes neurais artificiais utilizadas neste trabalho permitem mapear e controlar as trajetórias do manipulador. Para a validação experimental utilizou-se o manipulador robótico não-planar Robix RCS-6 (Rotacional - Rotacional) de dois graus de liberdade.

Palavras chave

Robótica, Geração de Trajetórias, Redes Neurais, Volume de Trabalho.

Abstract

Arcos Camargo, Marco Antonio, *Trajectory planning for a robotic manipulator using artificial neural networks*, Campinas,: Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas,2002. Nro p.106. Dissertação (Mestrado).

The objective of this dissertation is to present the trajectory control of a SCARA type manipulator within a working volume defined by the cinematic equations. The determination of the trajectory inside the working volume is obtained using an artificial neural network of the type multi-layer *perceptron*. The initial simulate, the manipulator movements between two arbitrary points in the working volume. The angular positions of each link are the input for the generation of the manipulator movements. Some of the movement simulations presented in this work show the behavior of the artificial neural network in the border of the working volume, in configurations where the traditional geometric models generally present singularities. The artificial neural network used in this work also allows mapping and control of the trajectories. For the experimental validation, a non-planar robotic manipulator of two degrees of freedom of the type Robix RCS-6 (Rotational-Rotational) was used.

Key words

Robotics, Generation of Trajectories, Neural networks, Working volume.

Sumário

Lista de Figuras

Lista de Tabelas

Nomenclatura

Letras Latinas

b - *bias*

d - deslocamento do corpo 1

e(n) - sinal de erro para um neurônio qualquer

f(.,.), **h**(.,.) - funções vetoriais não-lineares

g - aceleração da gravidade

f(.) - função de ativação

h - altura de elevação

I_{zz1}, I_{zz2} – momento de inércia do corpo 1 e 2

J - função de custo

k – neurônio *k*

$k(\theta,\dot{\theta})$ - energia cinética total

k, j - índices que representam neurônios da rede neural artificial

k_1, k_2, k_3 e k_4 - pólos do sistema

L – Lagrangeano

m – massa

m - número de discretização em m partes

n – tempo discreto

n - número de iterações (épocas)

N - número total de padrões de treinamento

p – número de entradas

$\dot{\boldsymbol{p}}$ - vetor de velocidade do efetuador final

$\dot{q}$ - vetor de velocidade nas juntas

$\mathbf{s}(ti)$, $\mathbf{p}i$ - Vetor componente da seqüência S, significando uma representação condensada do sinal na vizinhança de ti, $i = 1,..., n$.

T – matriz de transformação de coordenadas

T – período

t - tempo continuo

u – controle não linear (lei de controle)

v – nova lei de controle

u_p – sinal de entrada da planta

u_k – combinação linear da saída do neurônio k

$\mathbf{u(t)}$ - vetor de entrada

$U(\theta)$ - energia potencial total

Va – velocidade angular

V_L - velocidade linear

w_p – pesos sinápticos

$\mathbf{w_{k0}, w_{k1}, \ldots, w_{kp}}$ - pesos sinápticos de um neurônio qualquer

$\mathbf{x}_p$ – vetor de entrada

$\mathbf{x_0, x_1, \ldots, x_p}$ - sinais de entrada de um neurônio qualquer

$\mathbf{X}^T$ – transposto do vetor **x**, representado pelo índice T

y_k – sinal de saída da rede neural

$\hat{\mathbf{y}}$ - vetor de estado estimado pela rede neural

$\mathbf{y(n)}$ - saída de um neurônio qualquer

$\mathbf{y(t)}$ - vetor de saída

$\mathbf{y(t)}$ - vetor de estado

y_d – sinal de saída desejado

z – novas coordenadas

z^{-1} - operador atraso unitário

Letras Gregas

Δt - intervalo de amostragem

$\Delta\theta$ - incremento angular

Δw - pequena variação aplicada ao peso w

$\varepsilon(\mathbf{n})$ - soma dos erros quadráticos instantâneos da **RNA**

ε_m - erro quadrático médio

$\eta(\mathbf{n})$ - passo ou taxa de aprendizagem

$\beta(\mathbf{n})$ - coeficiente de momento

$\delta(\mathbf{n})$ - gradiente local de um neurônio qualquer

$\nabla\varepsilon$ - vetor gradiente

ξ, τ - limiares

Φ, Ψ - funções não lineares

$\varphi_K(\cdot)$ - função de ativação não-linear do neurônio k

φ'_j - derivada da função de ativação.

$\varphi'_j(\cdot)$ - derivada da função de ativação associada

ψ - transformação das coordenadas não lineares

Superescritos

T - transposto

Abreviações

AD - analógico digital

DC - corrente direta ou continua

DOF – *Degree Of Fredom*

DLT – *Direct Linear Transformation*

INITFF – valores iniciais aos pesos e bias de uma rede *feed-forward*

MGD – modelo geométrico direto

MGI - modelo geométrico inverso

MLP - *Multi Layer Perceptron*

PIC - *Peripheral Interface Controller*

PURELIN – função de ativação linear

RNA's - Redes Neurais Artificiais

SCARA – *Selectively Compliant Arm for Robotic Assembly*

SIMUFF – simulação da rede *feed-forward*

TANSIG – função de ativação não linear tangente hiperbólica

tanh – tangente hiperbólica

TRAINLM – treinamento da rede *feed-forward* utilizando o método Levenberg-Marquardt

Siglas

DMC – Departamento de Mecânica Computacional

Capítulo 1

Introdução

O desenvolvimento inicial da robótica baseou-se no esforço de automatizar as operações industriais. Este esforço começou no século XVIII, na industria têxtil, com o aparecimento dos primeiros teares mecânicos. Com o contínuo progresso da revolução industrial, as fábricas procuraram equipar-se com máquinas capazes de realizar e produzir automaticamente determinadas tarefas. No entanto, a criação de verdadeiros robôs só foi possível após a invenção do computador na década de 1940. Em conseqüência disso, tornaram-se possíveis os aperfeiçoamentos das partes que constituem os robôs.

A robótica há muito tempo desperta o interesse de pesquisadores. O estudo da robótica requer diferentes tipos de conhecimento, pois é por natureza multidisciplinar, envolvendo diversas áreas tais como: controle, programação matemática, inteligência artificial, cinemática, dinâmica, computação gráfica entre outras. Estas diversas áreas se interligam de tal forma que fazem da robótica uma disciplina científica específica.

Sob o ponto de vista de controle, robôs e manipuladores representam sistemas de controle automáticos essencialmente não lineares. Estes sistemas são multivariáveis e com acoplamentos dinâmicos cuja tarefa de controle é por si mesma dinâmica.

Assim, surge o método de linearização por realimentação ("*feedback linearization*") como uma alternativa para melhorar o desempenho destes sistemas, no anexo apresenta-se o método em forma de talhada. Trata-se de um método de bastante sucesso, utilizado para a solução de uma classe de problemas não lineares, nas equações que descrevem a dinâmica de robôs.

Outras áreas de pesquisas recentes que tem provocado um relativo entusiasmo no estudo da robótica são as redes neurais artificiais (**RNA's**), inspiradas no neurônio biológico. E esta nova área de pesquisa tem como objetivo o desenvolvimento de sistemas artificiais equivalentes aos sistemas cerebrais que controlam e gerenciam o corpo e suas atividades. Tais sistemas neurais artificiais possuem a capacidade de "aprender" através de leis de aprendizado e assim realizar tarefas complexas. As **RNA's** têm recentemente atraído uma grande atenção da comunidade científica devido a sua versatilidade junto com a habilidade de processamento paralelo e coletivo. Portanto crescem as expectativas da aplicação das **RNA's** para resolver problemas relacionados à sistemas dinâmicos robóticos. Estas aplicações incluem a identificação e o controle de sistemas dinâmicos não lineares.

Neste trabalho, o principal objetivo é a utilização das redes neurais artificiais para o mapeamento e controle de trajetórias em um manipulador robótico. Para isso implementou-se um algoritmo que gera o volume de trabalho do manipulador e divide este volume em setores compostos por uma densidade de pontos, Kumar and Waldron (1980),em seguida aproxima uma trajetória específica dentro deste volume de trabalho, partindo de uma posição inicial, até atingir a posição final,. A metodologia consiste em treinar as redes neurais artificiais a partir das equações do modelo para em seguida possibilitar ao manipulador acompanhar todas as possíveis trajetórias.

Com o algoritmo implementado foram realizadas simulações no ambiente do MATLAB®, que foram validadas experimentalmente usando o manipulador robótico Robix RCS-6.

1.1 Objetivos do trabalho

Os principais objetivos deste trabalho são:

i. Mostrar a aplicação de redes neurais artificiais num sistema robótico.

ii. Realizar um algoritmo para o mapeamento do volume de trabalho e o controle da trajetória de um manipulador robótico através de redes neurais artificiais.

iii. Realizar as simulações e a validação experimental do algoritmo.

1.2 Organização do trabalho

A dissertação está organizada de acordo com os seguintes capítulos:

Capítulo 1

Neste capítulo, apresenta se a introdução, as motivações e objetivos a serem alcançados nesta dissertação, assim como a estrutura da elaboração deste trabalho.

Capítulo 2

O principal objetivo deste capítulo é definir alguns conceitos que foram utilizados para direcionar o trabalho. Tais conceitos estão relacionados a robôs manipuladores, redes neurais artificiais, identificação de sistemas e controle.

Capítulo 3

Neste capítulo foi feita a modelagem teórica do manipulador. O estudo do comportamento da trajetória do manipulador através das variáveis cinemáticas de posição e velocidade pelo método *Newton-Euler*, as equações de movimento utilizam o método *Lagrange*.

Capítulo 4

Neste capítulo são mostrados e comparados os resultados das simulações computacionais com os resultados experimentais do manipulador robótico Robix RCS-6.

Capítulo 5

Aqui são apresentadas as conclusões dos resultados obtidos e também são dadas as sugestões para trabalhos futuros.

Capítulo 2

Revisão Bibliográfica

Este capítulo tem como principal objetivo definir alguns conceitos que serão utilizados no decorrer deste trabalho. Tais conceitos estão relacionados a robôs manipuladores e redes neurais artificiais.

2.1 Sistemas Robóticos

Sistemas robóticos ou manipuladores robóticos são dispositivos mecânicos versáteis equipados com atuadores e sensores sob o controle de computadores [HALPERIN et al., 1998]. Além disso são robôs programáveis, o que significa que eles podem realizar uma variedade de tarefas simplesmente mudando o *software* que os comanda. Os robôs executam tarefas que envolvem movimento em um espaço físico que pode estar sendo ocupado por vários objetos, sujeito às leis da natureza.

A movimentação de um manipulador robótico é definida pelo histórico temporal das posições espaciais, desde o instante inicial até o instante final. Este histórico de posições, constituído pelas configurações dos ângulos das juntas do manipulador e torques associados, recebe o nome de trajetória [CRAIG, 1989]. Cada ponto de uma trajetória define o estado do manipulador naquele instante específico. Um manipulador robótico é constituído de um sistema mecânico composto por elos ("*links*") conectados através de juntas em uma cadeia aberta ou fechada, controlada por um sistema hierárquico programável em vários níveis. Esta cadeia de elementos mecânicos tem por

objetivo final o posicionamento e orientação da extremidade do último elo em um ponto do espaço, com ou sem controle de trajetórias pré- estabelecidas.

As juntas são tipicamente classificadas em rotacionais, que permitem movimentos de rotação entre os elos, e prismáticas, que permitem movimentos de translação. Define-se como *variáveis de juntas*, à representação do deslocamento relativo entre dois elos adjacentes, denotadas por θ_i, *deslocamento angular* para as juntas rotacionais e por d_i , *deslocamento linear* para as juntas prismáticas. A posição da extremidade do último elo é influenciada por todas as variáveis de juntas.

As juntas são acionadas através da aplicação de torques ou forças externas por meio de atuadores, que podem ser hidráulicos, pneumáticos e elétricos. Os atuadores hidráulicos permitem torques ou forças elevadas; os atuadores pneumáticos, permitem movimentos precisos com baixa potência; os atuadores elétricos, apesar de não possuírem muito torque, possuem facilidade de controle e interface que fazem deles a solução perfeita para manipuladores de pequeno e médio porte.

As atividades dos manipuladores são monitoradas através de sensores. A maioria deles utilizados no controle de juntas atuam medindo variáveis de juntas e algumas variáveis dos atuadores como: corrente e tensão elétrica. Porém, algumas plantas utilizam outros tipos tais como: sensores de torque, de posicionamento espacial, de visão, de proximidade, de deslizamento, de força, etc...

Em geral o número de juntas determina o número de graus de liberdade do mecanismo (*Degree-Of-Fredom*, **DOF**). Tipicamente, um robô deve possuir no mínimo seis DOF independentes, três para posicionamento e três para a orientação. Um robô com mais de seis DOF é dito *cinematicamente redundante*.

O estudo ou análise de manipuladores robóticos pode ser dividido em duas partes, a *análise cinemática e a análise dinâmica.* Na análise cinemática é feito o estudo das variáveis de movimento do sistema (posições, velocidades e acelerações) sem preocupação com suas causas. Já

a análise dinâmica estuda os movimentos preocupando-se com as suas causas, os torques ou forças aplicados às juntas.

Isaac Asimov em 1950 estabeleceu as "Três Leis Da Robótica" a seguir.

Primeira Lei: Um robô não deve ferir um ser humano, ou por negligência em suas ações, permitir que um ser humano venha a ser ferido;

Segunda Lei: Um robô deve obedecer às ordens dadas por seres humanos, exceto quando essas ordens forem conflitantes com a Primeira Lei.

Terceira Lei: Um robô deve sempre garantir sua própria existência, somente enquanto tal proteção não contrariar a primeira ou a segunda lei.

2.2 Cinemática Inversa

É um método fundamental no uso prático de manipuladores. Com este método é possível determinar todos os ângulos das juntas que poderiam ser usados para determinar uma dada posição e orientação do elemento final do manipulador.

O problema da cinemática inversa não é tão simples quanto parece sugerir a definição anterior. A solução das equações cinemáticas nem sempre é fácil ou mesmo possível de se determinar em uma forma fechada. Surgem aqui também as questões de existência de uma solução e de soluções múltiplas. Determinar uma solução entre outras soluções é um requisito mínimo para maioria dos sistemas de controle de robôs.

2.3 Dinâmica Inversa

As forças ou torques exigidos para causar um movimento podem ser determinadas pela dinâmica inversa do robô. O ato de acelerar um manipulador robótico a partir do repouso, deslizar a uma velocidade constante do efetuador e finalmente desacelerar até parar, exige que um complexo conjunto de funções de torque sejam aplicadas pelos atuadores das juntas. A dinâmica inversa determina os torques e/ou forças dadas as posições e velocidades espaciais do manipulador.

Para mover um manipulador de um lugar a outro seguindo um caminho previamente especificado, cada junta deve ser movida de acordo com informações fornecidas pela cinemática e pela dinâmica inversa. A determinação dos ângulos e torques das juntas correspondentes a cada posição espacial é definida como *produção de trajetórias.*

O problema de execução de trajetórias será abordado nesta dissertação usando redes neurais artificiais.

2.4 Sistemas de Controle para Robôs

Um sistema de controle para robôs (manipuladores ou móveis) deve possuir no mínimo três etapas [BUGMANN et al., 1998]. Na primeira, chamada de sensoramento, a posição do robô deve ser determinada. Isto pode ser feito usando combinação de sensores (câmeras de vídeo, ultra-som, etc.). Num segundo estágio, denominado de programação ou planejamento, usa-se a informação sobre a configuração (estado) atual do robô obtida a partir da leitura dos sensores para determinar o próximo estado a ser alcançado. Num terceiro e último estágio, chamado de execução, procedimentos clássicos de controle são usados para guiar o robô para o estado fornecido pelo segundo estágio. Ao atingir a posição especificada pelo planejamento, uma nova leitura sensorial é feita e o processo se repete até que o robô atinja a posição desejada.

O objetivo do trabalho é a programação de robôs usando redes neurais artificiais, como procedimento do segundo estágio descrito no parágrafo anterior.

2.5 Seqüências Temporais, Trajetórias de Robô e Redes Neurais Artificiais

Em muitos domínios de aplicação, a variável tempo é uma dimensão essencial. Este é o caso da robótica, na qual trajetórias de robôs podem ser interpretadas como seqüências temporais cuja ordem de ocorrência de suas componentes precisa ser considerada. Nesta dissertação, desenvolve-se um modelo de rede neural artificial para aprendizagem e reprodução de trajetórias do manipulador Robix RCS-6.

A tarefa de percepção consiste em "perceber" o mundo através de diferentes canais. O ser humano usa para isso os cinco sentidos: visão, audição, tato, olfato e paladar. Robôs, também são capazes de processar informações visuais, tácteis e auditivas, usando para isso vários sensores diferentes. A tarefa de cognição está ligada à capacidade de o ser humano entender, raciocinar, aprender e decidir usando as informações que foram coletadas do mundo real. Por fim, a escolha da ação está relacionada com a capacidade de atuar no e/ou de modificar o mundo. Em qualquer uma dessas tarefas, a informação temporal se faz presente.

2.6 Classes de Problemas em Seqüências Temporais

Em muitas aplicações científicas e de engenharia é necessário modelar processos dinâmicos que lidam com seqüências temporais. O tipo de informação que se deseja extrair da seqüência vai depender da aplicação. Normalmente, quando se processa algum tipo de padrão temporal, se está interessado em [HERTZ et al., 1991]:

Reconhecimento de Seqüências:

Neste caso, deseja-se gerar um padrão de saída particular quando uma seqüência de entrada específica é apresentada. A seqüência de entrada deve ser apenas identificada. Uma aplicação típica é o reconhecimento de voz, em que a saída indicaria a palavra que foi falada.

Reprodução de Seqüências Temporais:

Aqui, o sistema deve ser capaz de gerar a seqüência de entrada quando parte dela (um ou mais estados) lhe é apresentada. Este seria o caso apropriado quando se deseja que a rede aprenda uma seqüência melódica, ou seja capaz de prever o curso futuro de uma série temporal a partir de partes desta melodia.

Associação Temporal de Seqüências:

Para esta situação, uma seqüência de saída particular deve ser gerada em resposta a uma seqüência específica de entrada. A seqüência de entrada e a de saída podem ser bastante diferentes. Este caso inclui as duas classes anteriores como casos especiais.

Geração de Seqüências Temporais:

Neste caso, o sistema gera uma sucessão de estados entre dois pontos quaisquer não consecutivos dados: o ponto inicial e o ponto final. Este conceito está estreitamente ligado ao de interpolação de estados. Neste ponto é interessante comentar que na literatura os termos reprodução e produção de seqüências temporais são utilizados indistintamente. Este também será o caso para a presente dissertação.

Até o presente momento tem-se falado exaustivamente no termo seqüência temporal. Entretanto, não se comentou ainda que a variável tempo pode ser considerada explícita ou implicitamente em tal seqüência temporal. Desta forma, é interessante perceber que a variável *tempo* pode vir embutida em uma seqüência temporal de duas maneiras básicas [WANG, 1995]:

(i) **Ordem temporal:**

Se as componentes de um padrão temporal são retiradas de um alfabeto específico, a ordem temporal se refere à posição relativa destas componentes dentro da seqüência. Por exemplo, a seqüência **a-b-c** é considerada diferente da **c-b-a** por causa do ordenamento diferente. Ordem temporal também pode se referir a uma estrutura sintática, tal como sujeito-verbo-objeto, na qual cada componente é escolhida entre um número de símbolos possíveis.

(ii) **Duração do Tempo:**

Assumindo uma taxa de amostragem uniforme, a duração do tempo é inversamente proporcional à taxa de apresentação (ou de observação) da seqüência. A duração desempenha um papel crítico em algumas tarefas de processamento temporal, tanto no reconhecimento quanto na reprodução de padrões temporais.

No que se refere à duração do tempo, WANG & ARBIB (1993) chamam de sistemas invariantes à taxa de apresentação aqueles que não são afetados pela velocidade de observação da seqüência, mas são sensíveis à duração relativa da componente. Sistemas invariantes ao intervalo de duração não são afetados ao se variar as durações das apresentações para as componentes individuais de uma seqüência.

Resumindo, sistemas invariantes à taxa de apresentação são sensíveis tanto à ordem dos eventos quanto à duração relativa dos eventos, enquanto que sistemas invariantes ao intervalo de duração são sensíveis apenas à ordem dos eventos.

2.7 Planejamento de Trajetória

A problemática que trata este item é a especificação, representação e geração dos elementos que descrevem a trajetória desejada para o manipulador, na medida em que este precisa efetuar movimentos no espaço de trabalho. Informações desta natureza são requisitadas pelo sistema de controle de movimento do manipulador: o histórico do movimento desejado é traduzido em valores de referência (*set-points*) para o sistema de controle dos atuadores das juntas.

Para o usuário do sistema robótico é mais prático descrever uma trajetória desejada no espaço cartesiano (embora o controle de movimento possa estar sendo realizado no espaço das juntas). Há também, algumas vantagens práticas (como modularidade e independência das juntas) ao se fazer isso através da especificação de movimentos do sistema de coordenadas da ferramenta {F}, relativos ao sistema de coordenadas da tarefa {T}. O problema básico consiste, então, em se mover o manipulador de uma condição (posição e orientação) inicial para uma condição final desejada. A trajetória que o manipulador deveria efetuar ao se movimentar da condição inicial para a final pode ser determinada segundo vários critérios, como, por exemplo, um trajetória obtida

como solução de um problema de otimização para movimento em tempo mínimo. Contudo, mesmo que não haja necessidade do órgão terminal rastrear um trajeto preciso (que poderia, inclusive, ser definido por uma função analítica), a especificação de $\{T\}_{inicial}$ e $\{T\}_{final}$ somente, por exemplo, geralmente é suficiente para garantir um movimento adequado, isto é, um movimento suficientemente suave. Portanto, um maior detalhamento na especificação da trajetória é geralmente requisitado, usualmente na forma de seqüência de pontos-via (conjunto de posições e orientações intermediárias), temporização correspondente e restrições. Um esquema de planejamento de trajetória poderia, então, interpolar os pontos especificados da trajetória e fornecer seqüênciada e temporizadamente os "*set-points*" para o controlador do movimento.

O planejamento de trajetórias pode ser feito no espaço cartesiano ou no espaço das juntas. Como geralmente o controle de movimento é exercido sobre as variáveis das juntas, quando o planejamento é feito no espaço cartesiano, a cinemática inversa do mecanismo deve ser utilizada para a geração dos *set-points* correspondentes. Para o planejamento no espaço das juntas: converte-se os pontos-via e restrições, caso especificados no espaço cartesiano em coordenadas das juntas e faz-se a interpolação polinomial destas coordenadas (visando suavidade de movimento). A cada período de amostragem de controle o "gerador de trajetórias" determina os *set-points* (quais devem ser as posições das juntas no próximo instante de amostragem) para o controlador de movimento. A principal desvantagem nessa abordagem é que o movimento executado pelo manipulador no espaço cartesiano pode ser menos preciso e complexo, caso os pontos-via não estejam suficientemente próximos entre si. As vantagens, entretanto, são grandes: planeja-se diretamente em termos das variáveis controladas durante o movimento, maior facilidade de planejamento e não há problema com singularidades cinemáticas do mecanismo. Com relação à precisão de rastreamento, deve-se notar que geralmente os *set-points* são gerados sem se levar em consideração a dinâmica do manipulador, o que pode resultar em erros grandes no rastreamento do trajeto desejado.

A maioria dos robôs utilizados no meio industrial possuem juntas rotativas, motivo pelo qual este trabalho será dirigido a este tipo de robôs, mas poderá ser facilmente estendido a outros tipos de robôs. Para que a utilização de robôs na industria seja viável, é necessário que a sua programação seja simples e de fácil modificação, permitindo ao usuário rapidez e flexibilidade.

Os métodos de programação mais freqüentemente utilizados em robôs industriais são:

- Aprendizagem ponto-a-ponto: não apresenta modelagem cinemática.

  i) *Movimento angular*: Neste método são gravados os pontos de referência fornecidos pelos transdutores de posição localizados em cada junta e a partir desses pontos são geradas trajetórias(angulares) através da utilização de algoritmos de interpolação. A simplicidade deste método não exige o conhecimento do modelo geométrico do robô;

  ii) *Movimento cartesiano*: idêntico ao procedimento descrito anteriormente mas a obtenção dos pontos de referência é realizada utilizando o modelo geométrico do manipulador.

- Programação *off-line*: A diferencia com a aprendizagem ponto-a-aponto es que este usa modelo geométrico cinemático. A partir de um software para visualização gráfica do modelo geométrico de robôs, devem ser obtidos pontos de passagem correspondentes à trajetória do robô (expresso em coordenadas angulares). Esses pontos de passagem poderão ser obtidos a partir do movimento angular de cada junta, ou a partir do modelo geométrico. A partir de um conjunto de pontos correspondentes à trajetória a ser realizada pelo robô, torna-se possível a implementação de algoritmos *off-line* para a interpolação e filtragem, levando-se em consideração aspectos dinâmicos e testes de colisão.

- Programação *on-line* : A partir do conhecimento do modelo geométrico e das características da trajetória desejada (posição final, velocidade e forma do caminho), pode-se implementar algoritmos para modelagem cinemática inversa e controle de posição.

Os métodos descritos anteriormente estão associados com o tipo de aplicação requerida. Isto permitirá:

- A realização de tarefas mais precisas e complexas;
- A necessidade de utilização de posições determinadas analiticamente, ou informações provenientes de sensores de percepção externa;
- O uso de manipuladores em ambiente adversos à presença do homem;
- A necessidade do aumento de tempo útil de trabalho pois, durante a programação da tarefa pelo método de aprendizagem, o manipulador é utilizado, diminuindo o seu tempo útil.

Normalmente a programação de tarefas de robôs é realizada no espaço das juntas, não necessitando de um modelo geométrico, e a trajetória angular de mesma natureza dos sinais provenientes do transdutor de posição servirá como sinal de referência para o controlador de cada junta robótica (Figura 2.1) . Entretanto a realização de alguma tarefa em relação a um sistema de referência colocado na ferramenta (espaço cartesiano) exige o conhecimento completo do modelo geométrico, e torna necessária a utilização de uma transformação de coordenadas, tendo em vista que o sinal de referência correspondente à trajetória, necessária para o controle das juntas, deverá ser angular.

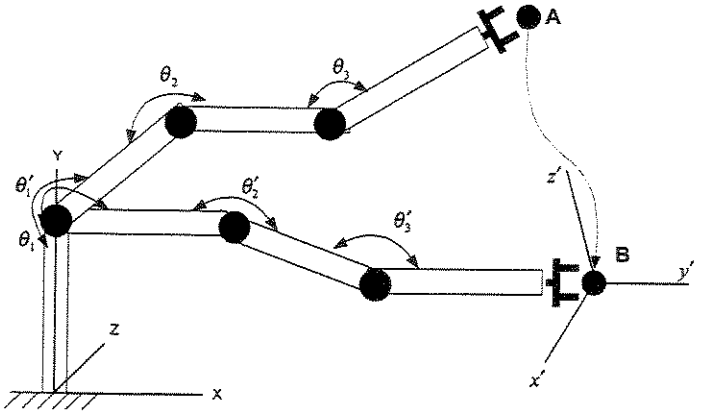


Figura 2.1- Espaço de coordenadas de um robô

Em um robô, os diferentes graus de liberdade podem ser associados a diversos sistemas de coordenadas, cada um associado a um grau de liberdade e servindo para descrever o movimento de cada grau de liberdade associado. Mais especificamente, o modelo geométrico é aquele que expressa a posição da garra em relação a um sistema de coordenadas solidário à base do robô, em função de suas coordenadas generalizadas. Esta relação se expressa matematicamente a partir de uma matriz e relaciona o sistema da base com o sistema de coordenadas do último elemento. Essa matriz é chamada de matriz de passagem homogênea do robô.

Embora a definição do modelo geométrico seja única, a maneira de obter a matriz de passagem homogênea do robô está associada ao sistema de referência utilizado. Existem diversas maneiras, e cada uma gera expressões diferentes que, embora equivalentes quantitativamente , podem diferir no número de operações aritméticas necessárias para fazer o cálculo numérico das mesmas. Tendo em vista a aplicação do modelo em sistemas de tempo real, é importante que se obtenha um modelo com o menor número possível de operações matemáticas.

Entretanto, a estratégia de controle nos robôs é exercida sobre seus atuadores, sendo que o sistema de controle só segue referências ou movimentos desejados definidos neste espaço(espaço das juntas). Contudo, o operador define as tarefas ou movimentos de referência no espaço operacional. Vemos assim que os movimentos desejados e as leis de controle estão em espaços de referência correspondentes às tarefas definidas no espaço operacional é denominada coordenação de movimentos, a qual é expressa matematicamente pela inversão do modelo geométrico. Existem dois métodos para a solução do problema da inversão do modelo geométrico:

- Métodos analíticos: Estes métodos não são gerais, isto é, a inversão analítica não é trivial e não há garantia de que seja possível fazê-la para um robô [Pieper, 1968], [Paul, 1981]; além disso, caso seja encontrada a solução, ela pode apresentar soluções múltiplas (problema de redundâncias).

- Métodos numéricos convergem para uma solução possível entre todas as existentes, [Goldenberg,1985], [Lee,1989] e com o atual desenvolvimento dos microprocessadores a utilização destes métodos em tempo real é viável. Os dois métodos mais utilizados são o

método da linearização das matriz de passagem do robô e o método recursivo que utiliza o cálculo do modelo geométrico e da matriz Jacobiana inversa.

2.8- Programação de Tarefas

2.8.1 Programação por Aprendizagem

Programar visa o estabelecimento de uma seqüência de operações a serem executadas pelo robô. A programação das tarefas pode ser realizada através de uma programação por aprendizagem ou a partir de uma linguagem de programação de computadores.

A programação por aprendizagem pode ser realizada pelos seguintes métodos:

- *Aprendizagem direta.* Neste método, Figura 2.2a, o operador guia fisicamente o robô por seu órgão terminal. Enquanto isto, os sensores de posição de cada junta do robô são utilizados para memorizar os pontos importantes da tarefa a ser executada.

- *Aprendizagem por simulação física.* Neste método, Figura 2.2b, o operador guia um simulador físico, que tem a geometria e os sensores idênticos ao robô original. Uma vez memorizada a tarefa, esta é transferida para o sistema de controle do robô. Este método é indicado para robôs de grande porte ou com estruturas mecânicas não reversíveis, bem como a programação em ambientes que exijam distância. Este método necessita de um modelo preciso e conhecido.

- *Aprendizagem por telecomando.* Neste método, Figura 2.2c, um dispositivo de telecomando (*teach pendant* ou *teach-in-box*) é utilizado para mover cada junta do robô isoladamente ou fornecer a posição e orientação da garra. Este método é adequado para qualquer tipo de robô e resulta mais barato que o método anterior, caso o telecomando seja feito a partir de um dispositivo de programação, que permita a atuação sobre cada junta independente.

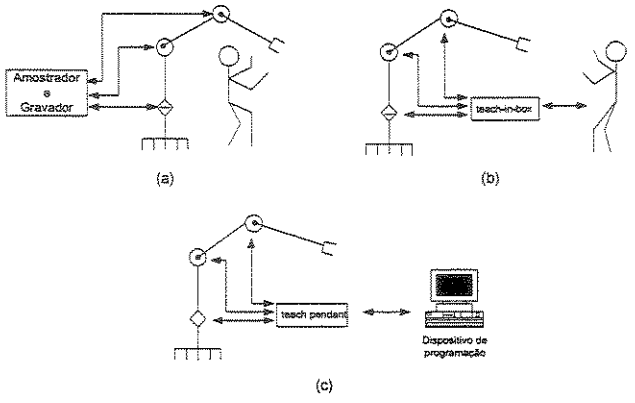


Figura 2.2 - programação por aprendizagem: (a)aprendizagem direta, (b) por simulação física e(c) por telecomando.

2.8.1.1- Memorização de Tarefas

A memorização de tarefas durante a fase de aprendizagem pode ser feita de duas maneiras:

- Programação ponto a ponto: São gravados apenas os pontos essenciais da trajetória do órgão, terminal, de modo que o movimento entre dois pontos consecutivos gravados fica determinado pelos algoritmos de controle utilizados nos servo-mecanismos das diversas ligações. Este método é adequado para aplicações que não requeiram um controle preciso da trajetória e da velocidade intermediários aos pontos essenciais da tarefa.

- Programação por caminhos contínuos: Os pontos da trajetória, na aprendizagem, são amostrados com uma taxa elevada e armazenados na memória. Estes pontos podem ser amostrados no controle segundo uma taxa programada, conduzindo o robô, continuamente por inércia ou por "arrasto", de maneira precisa e em uma velocidade que é função da taxa de amostragem.

2.8.2 Programação por Linguagens

A programação usando linguagens pode ser considerada como o processo pelo qual os programas são desenvolvidos sem a necessidade do uso do robô propriamente dito através da utilização de uma linguagem de computador. Com o atual avanço na tecnologia de desenvolvimento de hardware e software, a programação de robôs por linguagens, está cada vez mais próxima da realidade industrial. Estes avanços incluem uma maior sofisticação dos controladores, melhor precisão de posicionamento e incremento no número e tipos de sensores.

2.8.2.1 Níveis de Programação

As linguagens de programação de robôs podem ser classificadas nos seguintes níveis:

- Nível de junta. As linguagens classificadas neste nível requerem a programação individual de cada junta do robô para que uma dada posição seja alcançada.

- Nível de manipulador. Neste nível é necessário apenas fornecer a posição e orientação do órgão terminal e o sistema se encarrega de obter, através do modelo geométrico inverso do robô, as posições de cada junta.
- Nível de objeto: Neste nível é necessário apenas as especificações relativas ao posicionamento de objetos no interior do volume de trabalho do robô. Deste modo, é necessária a existência de um modelo matemático que representa o ambiente de trabalho no qual o robô se encontra.

- Nível de objetivo: Neste nível a tarefa não é realmente descrita, mas definida como, por exemplo: montar as peças A,B, e C. Neste caso é necessário, além do conhecimento do modelo, do ambiente, um conjunto de dados relativos a uma determinada tarefa.

2.8.2.2- Vantagens da Utilização de Programação por Linguagens

Entre as vantagens da utilização da programação por linguagens podemos citar :

- Redução de tempo em que o robô está fora da linha de produção, pois ele pode continuar operando enquanto a sua próxima tarefa está sendo programada;
- O programador não precisa entrar em contato com o meio ambiente de trabalho;
- Simplificação da programação de tarefas com alto grau de complexidade. As linguagens de programação facilitam a elaboração de tarefas complexas pois possuem formas analíticas de geração de trajetórias e análise de dados provenientes de sensores externos, além de várias estruturas de controle;
- Segurança na geração de trajetórias. As linguagens podem internamente gerar um modelo geométrico do ambiente de trabalho do robô e através de algoritmos de detecção de colisão, produzir trajetórias seguras e simulá-las por *software* antes da execução final.

2.9 Redes Neurais Artificiais

As redes neurais artificiais são modelos baseados no funcionamento do cérebro humano, e podem ser definidas como um modelo matemático com uma estrutura conveniente que tem analogia com o neurônio biológico [Kovács,1996], tal como mostra a Figura 2.3. O nome de rede neural foi dado a tais estruturas matemáticas por sua semelhança com a estrutura e funcionamento das células e dos tecidos nervosos. As redes neurais artificiais são utilizadas para obter resultados aproximados de problemas não lineares através de um mapeamento de entradas e saídas, com habilidade de aprendizagem e armazenamento de dados adaptados ao ambiente computacional. Haykin (1999), dá a seguinte definição para redes neurais.

Uma rede neural é um processador paralelamente distribuído, constituído de unidades de processamento simples, que tem a tendência natural para armazenamento de conhecimento adquirido através de experiências e torná-lo disponível para o uso. A rede neural se assemelha ao cérebro em dois aspectos:

1. *O conhecimento é adquirido pela rede a partir de seu ambiente através de um processo de aprendizagem.*

2. *Forças de conexão entre neurônios, conhecidas como pesos sinápticos, são utilizadas para armazenar o conhecimento adquirido.*

O processo de aprendizagem é chamado de *algoritmo de aprendizagem,* cuja função é modificar os pesos sinápticos da rede de uma forma ordenada para alcançar um objetivo de projeto.Redes neurais com arquitetura em camadas e fluxo de informações, com ou sem realimentação, possuem grande poder de adaptação e representação não linear. A presença de realimentação introduz memória no processamento, incluindo assim a capacidade de representação de sistemas dinâmicos.

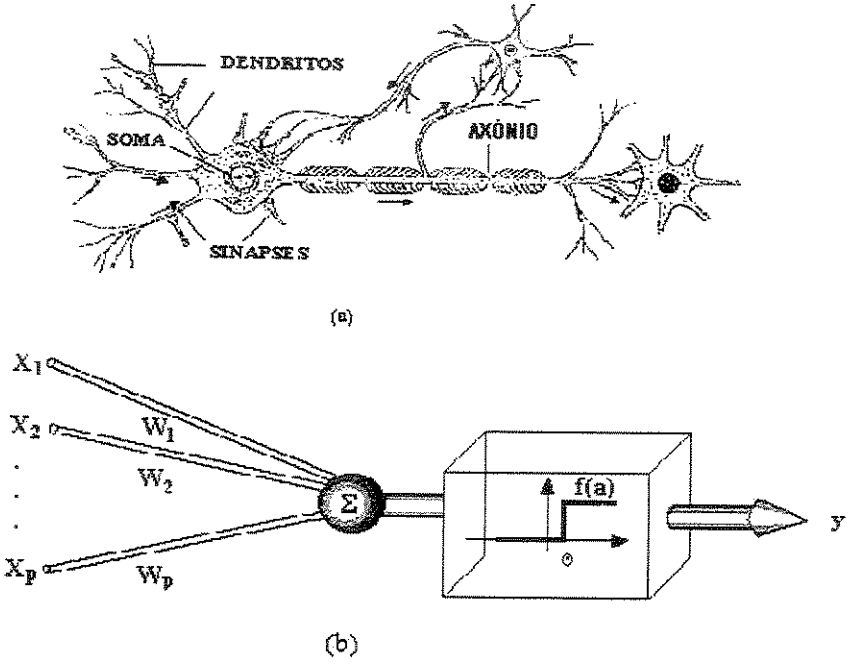


Figura 2.3 – Um neurônio: (a) conceito biológico; (b) mecanismo computacional

2.9.1 Modelo de Neurônio

Um neurônio é a unidade fundamental de processamento de informação de uma rede neural. [HAYKIN,1999]. MacCulloch-Pitts(1943) interpretou o funcionamento do neurônio biológico como sendo um circuito de entradas binárias combinadas por uma soma ponderada(com pesos) produzindo uma entrada efetiva. O modelo de um neurônio artificial tipo *perceptron* é apresentado na Figura 2.4

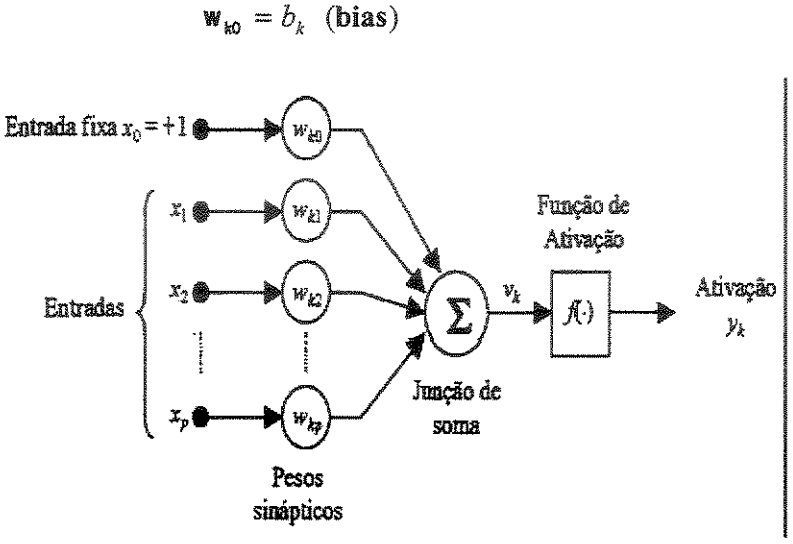


Figura2.4 - Modelo de um neurônio tipo *perceptron*

Identificamos três elementos básicos em um modelo neuronal:

1. Um conjunto de *sinapses* ou conexões de *entrada*, sendo cada entrada ponderada por um peso sináptico. Sendo assim, um sinal x_j na entrada da sinapse j conectada ao neurônio k é multiplicado pelo peso sináptico w_{kj}. Observe a ordem adotada paro os índices (subscritos) na notação aqui empregada: o primeiro índice se refere ao neurônio em questão e o segundo índice ao terminal de entrada da sinapse ao qual o peso se refere. Quando uma entrada fixa está presente (entrada x_o na Figura 2.4), o peso sináptico correspondente é denominado peso de polarização.

2. Uma junção de soma, responsável pela combinação aditiva dos sinais de entrada, ponderados pelos respectivos pesos das sinapses do neurônio.
3. Uma função de ativação geralmente não-linear e de formato sigmoidal, representando um efeito de saturação na ativação de saída y_k do neurônio. Tipicamente a excursão da ativação do neurônio é confinada ao intervalo (0,1) ou (-1,1).

Podemos descrever o modelo de neurônio ilustrado na Figura 2.4 pelo seguinte par de equações:

$$v_k = \sum_{j=0}^{p} w_{kj} x_j \tag{2.1}$$

$$y_k = f(v_k) \tag{2,2}$$

Onde $x_0, x_1, \ldots, x_p$ são os sinais de entrada, $w_{k0}, w_{k1}, \ldots, w_{kp}$ são os pesos sinápticos do neurônio k, v_k é o *nível de ativação interna* ou *potencial de ativação* do neurônio k, $f(\cdot)$é a função de ativação e y_k é a ativação da saída do neurônio k. O modelo neuronal da Figura 2.4 inclui também um parâmetro externo chamado bias representado por b_k aplicado ao neurônio artificial k. O bias tem o efeito de aumentar ou diminuir a entrada da função de ativação, dependendo se ele é positivo ou negativo.

Observe na Figura 2.4 a presença de uma entrada de polarização fixa $x_0 = -1$. Esta entrada, juntamente com o peso W_{k0} a ela associada tem o efeito de transladar a função de ativação em torno da origem (transformação afim), fazendo com que a ativação interna v_k do neurônio não seja nula quando todas as demais entradas $\{x_1, x_2, \ldots, x_p\}$ forem nulas. Podemos observar mais claramente que este efeito de polarização se descreve na equação (2,2)

$$y_k = f\left(\sum_{j=1}^{p} w_{kj} x_j - w_{k0} \right) \tag{2.3}$$

A entrada de polarização x_0 poderia assumir qualquer outro valor fixo diferente de zero. Alguns autores fazem uma distinção entre os termos polarização (quando a entrada fixa assume o

valor $x_0 = +1$) e limiar (quando $x_0 = -1$). Neste trabalho usaremos o termo polarização, independente do valor assumido pela entrada fixa x_0.

Podemos escrever a ativação do neurônio em notação vetorial, como segue:

- seja $\mathbf{x} = \{x_0 x_1 \ldots x_p\}^T$ o vetor de entradas do neurônio *k;*
- seja $\mathbf{w} = \{w_{k0}\ w_{k1}\ \ldots\ w_{kp}\}^T$ o vetor de pesos do neurônio *k;*

então a ativação y_k é dada por

$$y_k(\mathbf{w},\mathbf{x}) = f(\mathbf{w}^T \mathbf{x}) \tag{2.4}$$

2.9.2 Tipos de Função de Ativação

A função de ativação, representada por $\varphi(v)$ é responsável por definir a ativação de saída do neurônio em termos do seu nível de ativação interna ou *campo local induzido v.* Podemos identificar três classes principais de função de ativação:

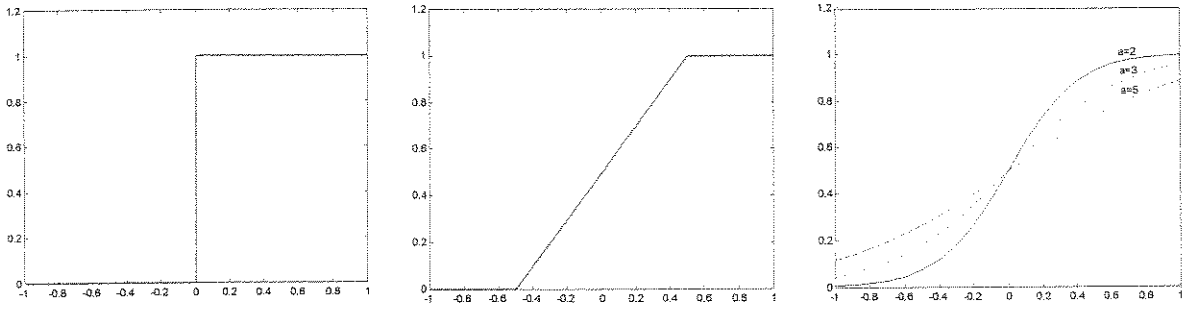


Figura 2.5 - Funções de ativação (a)função de limiar (b) Função Linear por Partes (c)Função Sigmóide

1. *função de limiar*. Para este tipo de função de ativação, descrita na Figura 2.5a, temos

$$\varphi(v)=\begin{cases}1 & se\ v\geq 0\\ 0 & se\ v<0\end{cases} \tag{2.5}$$

Esta forma de função de limiar é normalmente referida como *função de Heaviside*. De forma correspondente, a saída do neurônio k que emprega esta função de limiar é dada como

$$y_k=\begin{cases}1 & se\ v_k\geq 0\\ 0 & se\ v_k<0\end{cases} \tag{2.6}$$

onde o nível de atividade interna v_k do neurônio é:

$$v_k=\sum_{j=1}^{p} w_{kj}x_j+\mathrm{b}_k \tag{2.7}$$

Um neurônio com esta função de ativação é referido pela literatura como o modelo de *McCulloch-Pitts*, em reconhecimento ao trabalho pioneiro realizado por McCulloch e Pitts (1943).

2. *Função Linear por Partes*. Para a função linear por partes descrita na Figura 2.5b temos

$$\varphi(v)=\begin{cases}1 & se \quad v\geq+\frac{1}{2}\\ v & se\ +\frac{1}{2}>v>-\frac{1}{2}\\ 0 & se \quad v\leq-\frac{1}{2}\end{cases} \tag{2.8}$$

Onde se assume que o fator de amplificação dentro da região linear de operação é a unidade. Esta forma de função de ativação pode ser vista como uma aproximação de um amplificador não-linear .

2. *Função Sigmóide*. A função sigmóide é a função de ativação mais usada em redes neurais artificiais. Ela é definida como uma função estritamente crescente que tem um balanceamento adequado entre comportamento linear e não-linear. Um exemplo de função sigmóide é a função logística definida por

$$\varphi(v)=\frac{1}{1+\exp(-av)} \tag{2.9}$$

Onde a é o parâmetro de inclinação da função sigmóide. Variando-se o parâmetro a, obtém-se funções sigmóides com diferentes inclinações, como é mostrado na Figura 2.5c. No limite, quando o parâmetro de inclinação se aproxima do infinito, a função sigmóide se torna simplesmente uma função de limiar. Enquanto que a função de limiar assume o valor de 0 ou 1, uma função sigmóide assume um intervalo contínuo de valores entre 0 e 1.

Em muitas situações é desejável que a função de ativação varie entre -1 e $+1$, assumindo neste caso uma forma anti-simétrica em relação à origem. Uma função comumente empregada é a *tangente hiperbólica*, definida por

$$\varphi(v)=\tanh\left(\frac{v}{2}\right)=\frac{1-\exp(-v)}{1+\exp(-v)} \tag{2.10}$$

2.9.3 Redes Neurais de Múltiplas camadas

As redes neurais de múltiplas camadas são arquiteturas onde os neurônios são organizados em duas ou mais camadas de processamento, já que sempre existe pelo menos uma camada de entrada e uma camada de saída. Estas arquiteturas são as mais freqüentemente encontradas na literatura referente á redes neurais artificiais.

As redes neurais com apenas duas camadas são constituídas de uma camada de entrada que se conecta a uma camada de neurônios de saída. Os neurônios de camada de entrada são neurônios especiais, cujo papel é exclusivamente distribuir cada uma das entradas da rede (sem modificá-las) a todos os neurônios da camada seguinte. A forma mais simples deste tipo de rede neural, consiste de um único neurônio na camada de saída, sendo conhecido como *perceptron*. A Figura 2.4 é um exemplo de um *perceptron*, com p neurônios na camada de entrada e um único neurônio na camada de saída, de acordo com a notação adotada nesta seção. Um exemplo de rede neural com apenas duas camadas é apresentado na Figura 2.6(a) com 3 nós na camada de entrada e 4 neurônios na camada de saída. O *perceptron* foi objeto de intensa pesquisa durante os anos 50 e 60, mas em 1969 provaram matematicamente que este tipo de estrutura de processamento apresenta limitações importantes e podem ser aplicadas com sucesso a uma classe muito restrita de problemas [MINSKY & PAPERT, 1988]. Mais especificamente foi provado que o *perceptron* é capaz de resolver apenas problemas *linearmente separáveis*.

No entanto, com a utilização de redes de múltiplas camadas com pelo menos uma camada escondida (camada que não é nem entrada, nem saída), ou *perceptron de múltiplas camadas* (MLP-*multilayer perceptron*), muitas das limitações do *perceptron* deixam de existir.

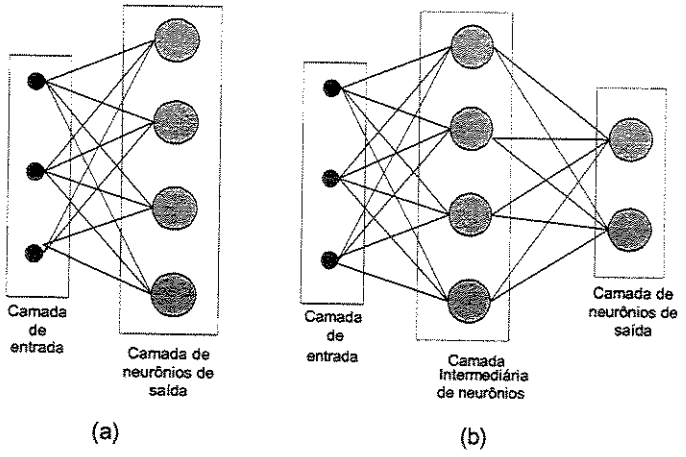


Figura 2.6 - Redes neurais em camadas:(a) rede com uma camada de neurônios,(b)rede com uma camada escondida de neurônios e uma camada de saída

Na Figura 2.6(b) apresentamos uma rede neural com uma camada escondida. Por simplicidade, esta arquitetura é referida como uma rede 3-4-2, isto é, 3 nós de entrada, 4 neurônios escondidos ou intermediários e 2 neurônios de saída. Como outro exemplo, uma rede neural com p nós na entradas, h_1 neurônios na primeira camada escondida, h_2 na segunda camada escondida e q neurônios na camada de saída é representada pela seguinte notação p-h_1-h_2-q

2.9.4 Aprendizado Supervisionado

A mais importante propriedade de uma rede neural artificial é sua capacidade de aprendizado. Uma rede neural aprende através de um processo iterativo de ajustes aplicados aos seus pesos sinápticos e limiares, o qual pode ser expresso na forma de um algoritmo computacional.

Não existe definição precisa, universalmente aceita, de "aprendizado". No contexto de redes neurais artificias, HAYKIN (1999) define aprendizado da seguinte forma: " Aprendizado é um processo pelo qual os parâmetros livres de uma rede são adaptados através de um processo de estímulo pelo ambiente no qual a rede está inserida". Assim, um processo de aprendizagem de uma rede neural implica na seguinte seqüência de eventos:

1. a rede é estimulada pelo ambiente de informação;
2. a estrutura interna da rede é alterada como resultado do estímulo;
3. devido às alterações que ocorreram em sua estrutura interna, a rede tem modificada sua resposta aos estímulos do ambiente.

Este tipo de aprendizado é caracterizado pela presença de um *"professor"* externo. A função do "professor" durante o processo de aprendizado é suprir a rede neural com uma *resposta desejada* a um determinado estímulo apresentado pelo ambiente. Definimos um *sinal de erro* como a diferença entre resposta desejada e a resposta observada na saída da rede neural. Os parâmetros

da rede são, então, ajustados de acordo com o sinal de erro. A Figura 2.7 apresenta um diagrama de blocos de um sistema com aprendizado supervisionado.

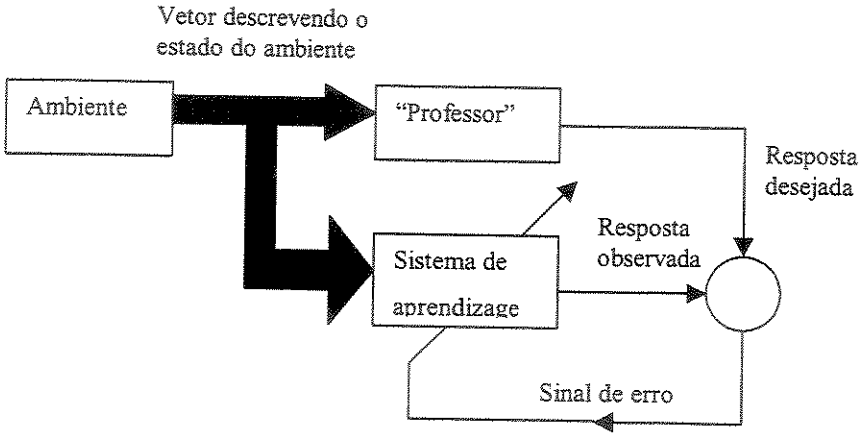


Figura 2.7 - Diagrama de blocos de um sistema com aprendizado supervisionado.

Uma forma de implementar uma estratégia de aprendizado supervisionado em redes neurais é através de procedimentos iterativos de correção do erro. Seja $s_j(n)$ a resposta desejada para um neurônio k no instante n e seja $y_j(n)$ a resposta observada para este neurônio. A resposta $y_j(n)$ é produzida por um estímulo, (vetor) x(n), aplicado à entrada da rede da qual o neurônio k faz parte. O sinal de erro para o neurônio k é definido como a diferença entre a resposta desejada e a resposta observada:

$$e_j(n) = s_j(n) - y_j(n) \tag{2.11}$$

O objetivo do procedimento de aprendizado por correção de erro é minimizar alguma *função de custo,* baseada no sinal de erro e_j(n), de modo que a resposta observada de cada neurônio da rede se aproxime da resposta desejada para aquele neurônio, em algum sentido estatístico. De fato,

uma vez definida uma função de custo, o problema de aprendizado torna-se um problema de otimização. Uma função de custo comumente empregada é a *soma dos erros quadráticos* :

$$J = \frac{1}{2}\sum_j e_j^2 \tag{2.12}$$

Onde o somatório é realizado sobre todos os neurônios da camada de saída da rede neural. A rede neural aprende através da minimização de J em relação aos pesos sinápticos da rede.
Note que J define uma superfície de erro sobre o espaço dos pesos. Se P é o número de pesos ajustáveis da rede neural então $J : \Re^P \rightarrow \Re$. A superfície de erro é caracterizada pela presença de mínimos locais e um, ou mais mínimos globais. Os métodos de otimização utilizados na minimização de J usualmente recorrem à informação de *gradiente do erro* para ajustar os parâmetros da rede. Teoricamente, estes métodos sempre atingem um ponto de mínimo da superfície de erro, mas observe que nada se pode afirmar sobre a natureza (local ou global) do ponto de mínimo obtido a partir de uma condição inicial arbitrária.

2.9.5 Retropropagação (Backpropagation)

A aprendizagem por retropropagação é um algoritmo padrão para o treinamento de *perceptrons* de múltiplas camadas, com o qual outros algoritmos de aprendizagem são comparados.

Este algoritmo realiza as derivadas parciais da função de custo (medida de desempenho) em relação aos parâmetros livres(pesos sinápticos e níveis de bias) da rede são determinados por retropropagação dos sinais de erro calculados pelos neurônios de saída através da rede, camada por camada.

O algoritmo de retropropagação é o mais utilizado no treinamento supervisionado para *Perceptrons* de múltiplas camadas, sendo basicamente uma técnica de gradiente e não uma técnica de otimização. Este algoritmo realiza o cálculo do gradiente no sentido inverso da direção da rede, baseando-se nos erros calculados em sua saída, como uma regra de cadeia, retornando o valor

desejado em sua camada de entrada, e assim, recalculando a rede. Pode ser usado para a atualização dos pesos processando seus dados após cada iteração para obter a convergência de um ponto estacionário desejado, onde a taxa de aprendizagem utilizada deve ser lentamente reduzida. Esta propriedade de convergência comporta-se como uma aproximação estocástica.

2.9.6 Algoritmo de Retropropagação

O sinal erro na saída do neurônio j na iteração n (isto é, apresentação do n-ésimo padrão de treinamento) é definido como a diferença da saída desejada d pela saída real ou observada y do sinal :

$$e_j(n) = d_j(n) - y_j(n) \tag{2.13}$$

Derivando ambos os lados da equação anterior em relação a $y_j(n)$, obtém-se:

$$\frac{\partial e_j(n)}{\partial y_j(n)} = -1 \tag{2.14}$$

Define-se o valor instantâneo do erro quadrático para o neurônio j como $\frac{1}{2}e_j^2(n)$. De forma correspondente, o valor instantâneo do erro quadrático $\varepsilon(n)$, é obtido somando-se os termos $\frac{1}{2}e_j^2(n)$ de todos os neurônios da camada de saída, para os quais os sinais de erro podem ser calculados diretamente. Podemos assim escrever que a soma instantânea dos erros ao quadrado é dada por:

$$\varepsilon(n) = \frac{1}{2}\sum_j e_j^2(n) \tag{2.15}$$

Derivando ambos os lados da equação anterior em relação à $e_j(n)$, obtemos

$$\frac{\partial \varepsilon(n)}{\partial e_j(n)} = e_j(n) \tag{2.16}$$

Considere que N represente o número total de padrões contidos no conjunto de treinamento. Então a *média do erro quadrático* é obtida somando-se os $\varepsilon(n)$ para todos os n, o qual representa uma *função de custo*

$$\varepsilon_m = \frac{1}{N} \sum_{n=1}^{N} \varepsilon(n) \tag{2.17}$$

O nível de atividade interna da rede v na entrada da não linearidade, associada ao neurônio j é portanto:

$$v_j(n) = \sum_{i=0}^{p} w_{ji}(n) y_i(n) \tag{2.18}$$

Derivando a equação anterior em relação à $w_{ji}(n)$:

$$\frac{\partial v_j(n)}{\partial w_{ji}(n)} = y_i(n) \tag{2.19}$$

Assim , o sinal funcional $y_j(n)$ que aparece na saída do neurônio j na iteração n é dada por:

$$y_j(n) = \varphi(v_j(n)) \tag{2.20}$$

e depois, derivando-se a equação anterior em relação à $v_j(n)$, obtém-se:

$$\frac{\partial y_j(n)}{\partial v_j(n)} = \varphi_j'(v_j(n)) \tag{2.21}$$

onde a diferenciação em relação ao argumento é representado por (.)'.

O algoritmo de retropropagação aplica uma correção $\Delta w_{ji}(n)$ ao peso sináptico $w_{ji}(n)$, que é proporcional à derivada parcial $\partial \varepsilon(n)/\partial w_{ji}(n)$. De acordo com a *regra da cadeia* do cálculo podemos expressar este gradiente como:

$$\frac{\partial \varepsilon(n)}{\partial w_{ji}(n)} = \frac{\partial \varepsilon(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \frac{\partial v_j(n)}{\partial w_{ji}(n)} \tag{2.22}$$

A derivada parcial $\partial \varepsilon(n) / \partial w_{ji}(n)$ representa um fator de sensibilidade, determinando a direção de busca no espaço de pesos, para o peso sináptico w_{ji}.

Substituindo as frações da equação anterior obtemos :

$$\frac{\partial \varepsilon(n)}{\partial w_{ji}(n)} = -e_j(n)\varphi_j'(v_j(n))y_i(n) \tag{2.23}$$

A correção $\Delta w_{ji}(n)$ aplicada a $w_{ji}(n)$ é definida pela *regra delta*:

$$\Delta w_{ji}(n) = -\eta \frac{\partial \varepsilon(n)}{\partial w_{ji}(n)} \tag{2.24}$$

onde η é o parâmetro da *taxa de aprendizagem* do algoritmo de retropropagação. Considerando a taxa de aprendizagem η do algoritmo de retropropagação, onde o sinal *negativo* indica um gradiente decrescente no espaço de pesos buscando uma direção para a mudança de peso que reduza o valor de $\varepsilon(n)$, define-se:

$$\Delta w_{ji}(n) = -\eta \frac{\partial \varepsilon(n)}{\partial w_{ji}(n)} \tag{2.25}$$

Substituindo a equação anterior, tem-se:

$$\Delta w_{ji}(n) = \eta \delta_j(n) y_i(n) \tag{2.26}$$

Onde o *gradiente local* $\delta_j(n)$ é definido por

$$\delta_j(n) = -\frac{\partial \varepsilon(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} = e_j(n)\varphi_j'(v_j(n)) \tag{2.27}$$

O gradiente local $\delta_j(n)$ para o neurônio de saída j é igual ao produto do sinal de erro $e_j(n)$ correspondente para aquele neurônio pela derivada $\varphi_j'(v_j(n))$ da função de ativação associada.

Quando o neurônio j é um neurônio escondido, não há nenhuma saída desejada pré-especificada para o neurônio. Assim, o sinal de erro de um neurônio escondido deve ser calculado em termos dos sinais de erro de todos os neurônios aos quais o neurônio escondido está diretamente conectado. Pode-se redefinir o gradiente local $\delta_j(n)$ para o neurônio oculto i como

$$\delta_i(n) = -\frac{\partial \varepsilon(n)}{\partial y_i(n)} \frac{\partial y_i(n)}{\partial v_i(n)} = -\frac{\partial \varepsilon(n)}{\partial y_i(n)} \varphi_i'(v(n)) \tag{2.28}$$

O neurônio k, que pertence à camada de saída

$$\varepsilon(n) = \frac{1}{2} \sum_k e_k^2(n) \tag{2.29}$$

Derivando-se a equação anterior em relação a $y_i(n)$, obtém-se:

$$\frac{\partial \varepsilon(n)}{\partial y_i(n)} = \sum_k e_k \frac{\partial e_k(n)}{\partial y_i(n)} \tag{2.30}$$

Utilizando a regra da cadeia para calcular a derivada parcial $\partial e_k(n)/\partial y_i(n)$ pode-se rescrever a equação anterior como:

$$\frac{\partial \varepsilon(n)}{\partial y_i(n)} = \sum_k e_k(n) \frac{\partial e_k(n)}{\partial v_k(n)} \frac{\partial v_k(n)}{\partial y_i(n)} \tag{2.31}$$

O erro na saída do neurônio k é dado pela diferença entre a saída desejada d_k e a saída observada y_k

$$e_k(n) = d_k(n) - y_k(n) = d_k(n) - \varphi_k(v_k(n)) \tag{2.32}$$

Derivando-se a equação anterior em relação à $v_k(n)$ temos

$$\frac{\partial e_k(n)}{\partial v_k(n)} = -\varphi_k'(v_k(n)) \tag{2.33}$$

Para o neurônio k o potencial de ativação é

$$v_k(n) = \sum_{i=0}^{p} w_{ki}(n) y_i(n) \tag{2.34}$$

Onde p é o número total de entradas (excluindo o bias) aplicadas ao neurônio k.

Derivando-se a equação anterior em relação a $y_i(n)$:

$$\frac{\partial v_k(n)}{\partial y_i(n)} = w_{ki}(n) \tag{2.35}$$

Assim, substituindo as equações, obtemos a derivada parcial desejada.

$$\frac{\partial \varepsilon(n)}{\partial y_i(n)} = -\sum_k e_k(n) \varphi_k'(v_k(n)) w_{ki}(n) = -\sum_k \delta_k(n) w_{ki}(n) \tag{2.36}$$

Finalmente, substituindo-se a equação anterior, obtemos a expressão para o gradiente local δ do neurônio intermediário i:

$$\delta_i(n) = \varphi_i'(v_i(n)) \sum_k \delta_k(n) w_{ki}(n) \tag{2.37}$$

Resumindo-se as relações do algoritmo de retropropagação, a correção $\Delta w_{ji}(n)$ aplicada ao peso sináptico que está conectando o neurônio intermediário i ao neurônio j é definido pela regra delta, representada simplesmente por

$$\Delta w_{ij}(n) = \eta \delta_i(n) y_j(n) \tag{2.38}$$

2.9.7 Redes Neurais em Identificação de Sistemas

Durante os últimos anos, numerosos estudos têm sido feitos na área de identificação e controle de sistemas físicos dinâmicos, devido principalmente, aos grandes avanços alcançados na micro-eletrônica e da popularização da informática. Basicamente, o procedimento envolve a modelagem do sistema, a identificação, o projeto dos controladores e os testes de verificação.

Podemos citar seis passos que devem ser seguidos na tarefa de identificação de um sistema dinâmico: desenvolver um modelo analítico do sistema; estabelecer os níveis de resposta dinâmica que provavelmente ocorrerão, usando o modelo analítico e as características das fontes de excitação; colocar a instrumentação requerida para o levantamento das medidas necessárias; realizar os experimentos e gravar os dados; aplicar as técnicas de identificação para o levantamento das características do sistema; e refinar o modelo analítico se necessário [Juang, 1994].

A identificação de sistemas é a abordagem experimental para modelar um processo ou uma planta de parâmetros desconhecidos. Envolve os seguintes passos: planejamento experimental, seleção de uma estrutura de modelo, estimação de parâmetros e a validação do modelo. O processo de identificação de sistemas, como o realizado na prática, é de natureza iterativa, até que seja construído um modelo satisfatório.

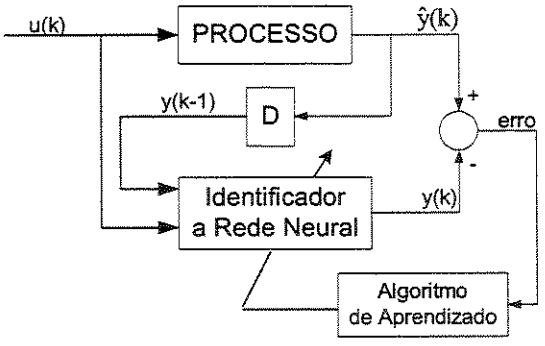


Figura2.8 - Modelo para identificação de sistemas com RNA's.

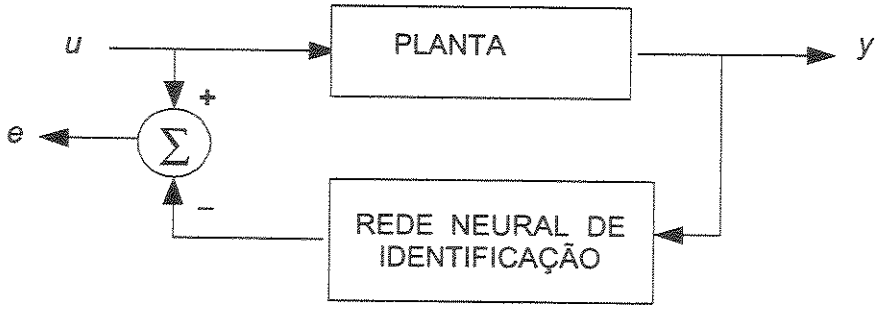


Figura 2.9 - Identificação de sistemas pela dinâmica inversa utilizando redes neurais

2.9.7.1 Identificação de Sistemas Utilizando o Modelo de Espaço de Estados

Considere-se que a planta seja descrita pelo modelo de espaço de estados, ela está representada pelas equações a seguir:

$$\mathbf{x}(n+1) = \mathbf{f}(\mathbf{x}(n), \mathbf{u}(n)) \tag{2.39}$$

$$\mathbf{y}(n) = \mathbf{h}(\mathbf{x}(n)) \tag{2.40}$$

Onde $\mathbf{f}(.,.)$ e $\mathbf{h}(.,.)$ são funções vetoriais não-lineares, que são assumidas como desconhecidas. Utilizam-se duas redes neurais para identificar o sistema: uma para lidar com a equação de processo (2.39) e a outra para lidar com a equação de medida (2.40), temos a Figura 2.10.

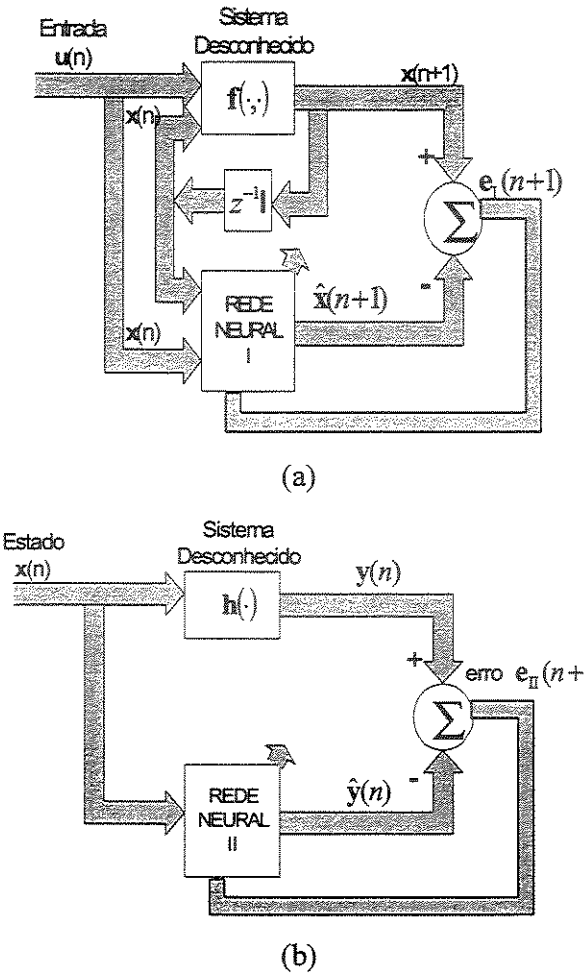


Figura 2.10 - Solução por espaço de estados para o problema de identificação de sistemas

Sendo o estado x(n), um atraso de x(n+1) considere que $\hat{\mathbf{x}}(n+1)$ representa a estimativa de x(n+1) produzida pela primeira rede neural, conforme Figura 2.10(a) pela rede I. Esta rede opera sobre uma entrada concatenada, consistindo da entrada externa u(n) e do estado x(n), para produzir $\hat{\mathbf{x}}(n+1)$. A estimativa $\hat{\mathbf{x}}(n+1)$ é subtraída do estado real x(n+1) para produzir o vetor de erro.

$$e_I(n+1) = \mathbf{x}(n+1) - \hat{\mathbf{x}}(n+1)$$

Onde x(n+1) é a resposta desejada. O vetor de erro $e_I(n+1)$ é usado para ajustar os pesos sinápticos da rede neural I, para minimizar a função de custo baseada no vetor de erro $e_I(n+1)$.

A segunda rede neural da Figura 2.10b opera sobre o estado real x(n) da planta desconhecida para produzir a saída estimada $\hat{\mathbf{y}}(n)$ da saída real y(n). A saída estimada é subtraída de y(n) para produzir o segundo vetor de erro

$$e_{II}(n) = \mathbf{y}(n) - \hat{\mathbf{y}}(n)$$

Onde y(n) é a resposta desejada. O vetor de erro $e_{II}(n)$ é usado para ajustar os pesos sinápticos da rede II para minimizar a norma euclidiana do vetor de erro $e_{II}(n)$.

As duas redes neurais mostradas na Figura 2.10 trabalham sincronizadas para fornecer uma solução por espaço de estados para o problema de identificação de sistemas mencionado por Narendra e Partasarathy(1990). Este modelo é conhecido como *modelo de identificação série-paralelo.*

2.9.7.2 Identificação de Sistemas Utilizando o Modelo de Entrada - Saída

Considere que o sistema tenha uma saída *y(n)* devido à entrada *u(n)* para a variável tempo discreto *n*. Escolhendo-se o modelo auto-regressivo não-linear, o modelo de identificação assume a forma seguinte:

$$\hat{y}(n+1) = \varphi(y(n),...,y(n-q+1),u(n),...,u(n-q+1))$$

Onde *q* é a ordem do sistema desconhecido, no tempo *n+1*. Os *q* valores passados da entrada e os *q* valores passados da saída estão disponíveis no modelo. A saída do modelo $\hat{\mathbf{y}}(n+1)$ representa uma estimativa da saída real *y(n+1)* para produzir o sinal de erro

$$e(n+1) = y(n+1) - \hat{y}(n+1)$$

Onde $y(n+1)$ é a resposta desejada. O erro é usado para ajustar os pesos sinápticos da rede neural de modo a minimizar o erro. O modelo de identificação da Figura 2.11 é da forma série-paralela [Haykin,1999],[Narendra,1990] porque a saída real do sistema (em vez da saída do modelo de identificação) é realimentada para a entrada do modelo.

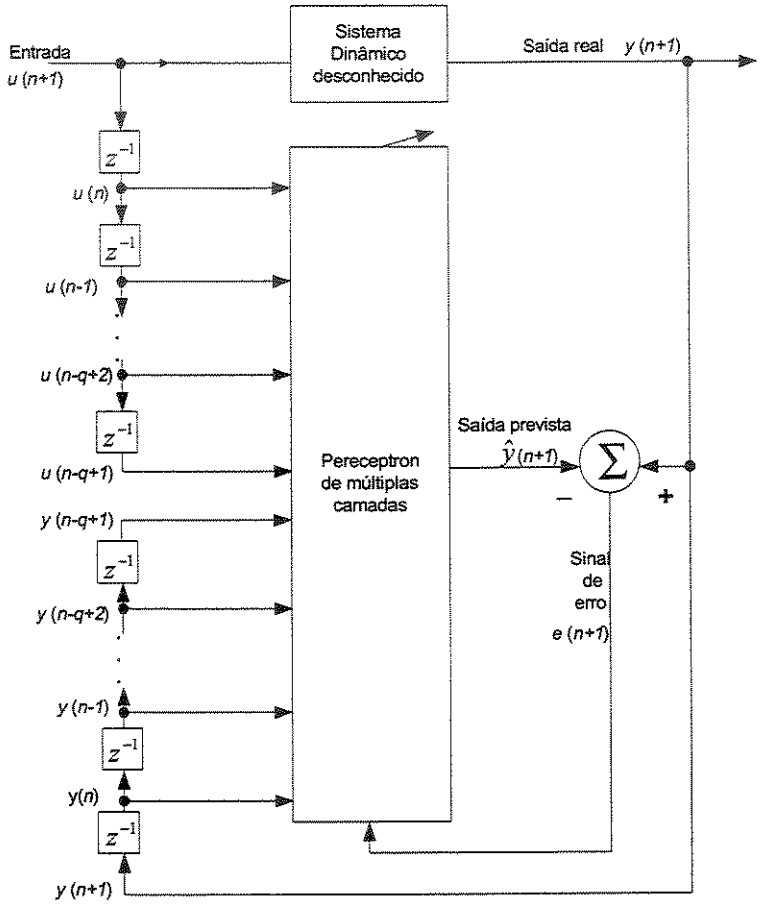


Figura 2.11 - Solução por entrada-saída para o problema de identificação de sistemas

2.9.8 Redes Neurais em Sistemas de Controle

Ultimamente têm sido desenvolvidas algumas pesquisas na aplicação de redes neurais em sistemas de controle e automação. Isso tem possibilitado novas técnicas que se inspiram na obtenção de melhores e mais eficientes ações de controle através do paradigma das redes neurais. Essas novas técnicas tentam aliar os benefícios das técnicas de controle ótimo e controle adaptativo.

De uma forma geral, o problema de controle de um processo dinâmico pode ser esboçado como na Figura 2.12, sendo este esquema chamado de sistema de controle com realimentação das variáveis de estado [Cerqueira, 1996].

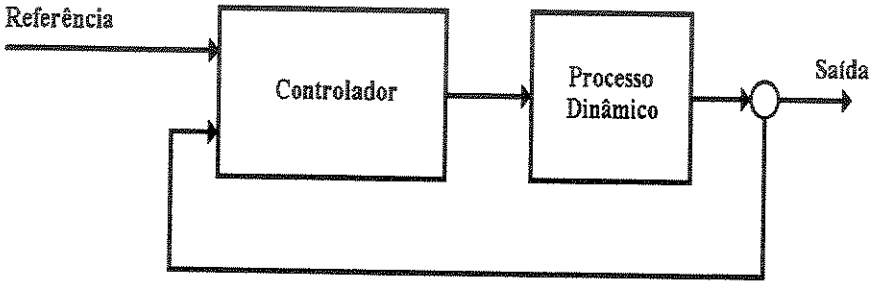


Figura 2.12 – Sistema de controle genérico

Um processo dinâmico ou sistema dinâmico é um conjunto de elementos coordenados entre si, e que formam uma estrutura organizada como se fosse um sistema único. Ele pode ser visto com mais detalhes na Figura 2.13. Ele é basicamente composto por:

Sistema de transformação – Responsável por fazer operações de transformações quaisquer. Possui como entradas os sinais de saída dos acionadores e variáveis externas de perturbação, que podem ser classificadas em mensuráveis e não mensuráveis. Os efeitos das transformações são refletidos nas chamadas variáveis de saída. Os sistemas de transformação apresentam

comportamento não linear e algumas vezes, apesar de terem os seus modelos matemáticos conhecidos teoricamente, apresentam parâmetros com valores desconhecidos ou pouco confiáveis e/ou variantes. Porém, muitas vezes, é possível fazer-se linearizações e utilizar o modelo em questão para o projeto das estratégias de controle do sistema.

Sensores – Responsáveis pela medição das variáveis de saída assim também como das externas e informá-las ao controlador. Em sistemas físicos reais os sensores apresentam um ligeiro atraso na medição, não linearidades e sujeição a ruídos, mesmo nos casos de sensores projetados com base em tecnologias avançadas. No entanto, em geral é possível desprezar-se esses efeitos e considerar os sinais apresentados pelos sensores ao controlador em um dado instante de tempo, como sendo os valores reais das variáveis de saída.

Acionadores – Responsável pela tarefa de receber o sinal de saída do controlador e prover a potência necessária à operação do sistema de transformação. Esta tarefa também pode ser feita pela modificação ou atuação em variáveis de entrada do processo dinâmico. Assim como os sensores, os acionadores possuem problemas de atraso de tempo, não linearidades e sujeição a ruídos e distorções de sinais de baixa potência, gerados pelo controlador [Cerqueira, 1996].

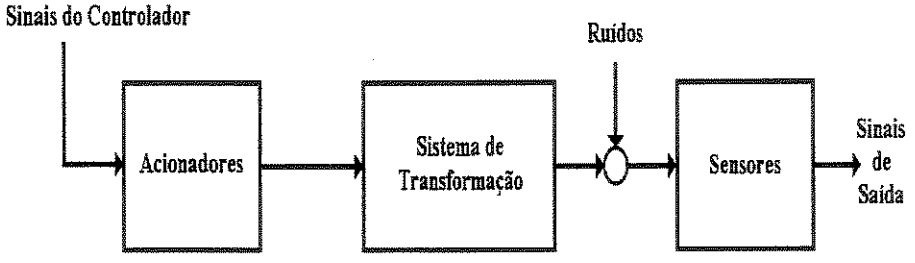


Figura 2.13 – Processo dinâmico

Para efeito de projeto de um sistema de controle, são considerados em conjunto os modelos matemáticos dos sistemas de transformação, dos sensores e dos acionadores. Assim, teremos um único modelo o que facilita o estudo de todo o processo dinâmico. O controlador é o elemento responsável em fazer com que as variáveis de saída do sistema tendam ou rastreem os valores desejados para os mesmos, atendendo também às especificações de desempenho que estejam preestabelecidas.

Usualmente o projeto de controladores tradicionais envolve análises matemáticas complexas e possuem muita dificuldade em controle de sistemas altamente não lineares. Para superar esta dificuldade, um número cada vez maior de trabalhos usando redes neurais para controle de sistemas tem sido desenvolvidos nos últimos anos. O uso da capacidade de aprendizado das redes neurais tem ajudado no projeto de controladores mais flexíveis, especialmente quando a dinâmica do sistema é complexa e altamente não linear.

Tomando diretamente o problema de controle não linear, vários estudos experimentais e simulações tem mostrado que o mapeamento entrada-saída e a capacidade de aproximação de funções não lineares das redes neurais podem ser usadas para projetar controladores que podem ser baseados na dinâmica inversa do sistema a controlar [Pham, 1999], ou que sejam capazes de identificar a dinâmica de um modelo de referência, [Park, 1996], ou ainda, controladores capazes de otimizar o desempenho do sistema. Estas habilidades (capacidade de aprendizado e de processamento paralelo [MacLoone, 1997]) são úteis para o projeto de controladores auto-ajustáveis, permitindo adaptações às mudanças de *hardware* ou das funções de custo.

Capítulo 3

Modelagem de robôs manipuladores

A modelagem de um sistema mecânico descreve as relações cinemática e dinâmicas em expressões matemáticas, levando em consideração parâmetros construtivos como massa, comprimento, inércia, graus de liberdade, entre outros. Estas relações são úteis para descrever o comportamento do sistema e para o projeto de técnicas de controle.

Neste trabalho foi utilizado um manipulador plano derivado da estrutura espacial RRP do manipulador SCARA, Figura3.1, onde a junta prismática foi substituída por um ponto p(x,y) localizado no centro geométrico da junta e não será analisada a orientação do elemento terminal.

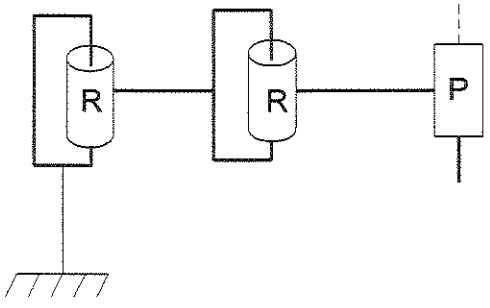


Figura 3.1- Representação do manipulador RRP-SCARA

De acordo com a Figura 3.2, observa-se que os eixos cartesianos originais do manipulador SCARA foram modificados na terceira junta, a junta prismática, que foi suprimida visando simplificar as equações matemáticas do modelo geométrico direto. Tal simplificação é possível, pois o movimento de translação ao longo do eixo coordenado Z, não influencia as coordenadas X e Y do ponto(x,y).

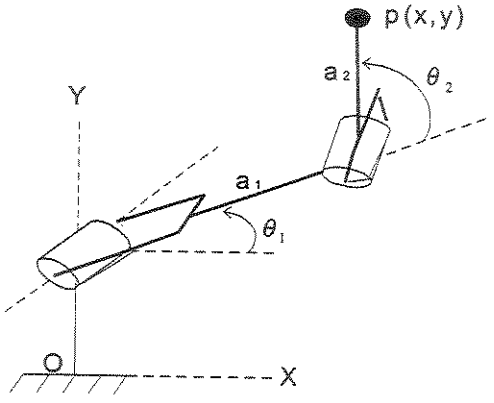


Figura 3.2- Manipulador do tipo RR planar

3.1 Modelagem Cinemática

A modelagem cinemática de robôs se preocupa em representar a posição e orientação do efetuador final do manipulador em um sistema de coordenadas cartesianas, normalmente o da base, descrevendo sua movimentação no tempo sem se preocupar com as forças que produzem este movimento. A análise cinemática de um robô manipulador é dividida em cinemática direta e cinemática inversa.

3.1.1 Cinemática Direta

O problema da cinemática direta parte das variáveis de posição, velocidade e aceleração referentes às juntas do robô para determinar as variáveis correspondentes de posição, velocidade e aceleração do efetuador final.

A estrutura mecânica do robô é formada por uma cadeia de ligações conectadas por juntas. Visando representar este encadeamento entre corpos é adotado um conjunto de sistemas de coordenadas (cartesiano tridimensional) localizados um em cada junta do manipulador. A posição relativa entre as ligações é determinada relacionando a posição e orientação entre cada sistema conforme a convenção de Denavit-Hartenberg [Asada,1986].

O método de Denavit-Hartenberg é baseado na representação da posição do corpo rígido por uma matriz [4x4], e utiliza um número mínimo de parâmetros para descrever completamente a relação cinemática do sistema. O par composto por um elo L_i e a respectiva junta são cinematicamente definidos por quatro parâmetros, sendo dois deles parâmetros de junta e os outro dois relativos ao elo:

- θ_i é o ângulo entre os eixos x_{i-1} e x_i, medido sobre o eixo z_{i-1}, usando a regra da mão direita.
- d_i distância entre a origem O_{i-1} e a interseção dos eixos z_{i-1} e x_i, medida sobre o eixo z_{i-1} ou a distância mais curta entre x_{i-1} e x_i;
- a_i é a mais curta distância entre o eixo z_{i-1} e z_i medida sobre o eixo x_i.
- α_i é o ângulo entre o eixo z_{i-1} e z_i, medido sobre o eixo x_i (utilizando a regra da mão direita).

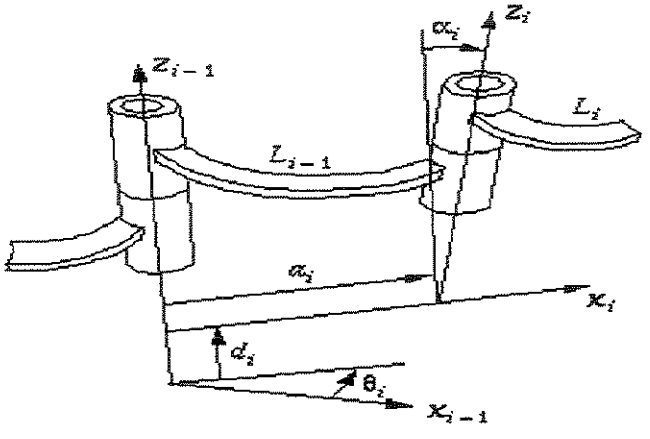


Figura3.3 - Denavit-Hartenberg

Para a junta de revolução, as quantidades d_i , a_i e α_i são constantes e θ_i é variável. Enquanto para a junta prismática θ_i, a_i e α_i são constantes e d_i é variável. Portanto, para qualquer caso a_i e α_i são constantes do manipulador.

A matriz [4x4] que representa a relação entre as juntas adjacentes O_i escrita em relação ao sistema de coordenadas O_{i-1}, pode ser representado pela matriz de transformação genérica, eq.3.1. Esta matriz para um sistema de vários graus de liberdade O_n^0, descreve a posição e orientação do efetuador final em relação ao sistema de coordenadas da base, e é obtida pela multiplicação das diversas transformações das juntas adjacentes.

$$O_i^{i-1} = \left[\begin{array}{ccc|c} \cos\theta_i & -\cos\alpha_i sin\theta_i & sin\alpha_i sin\theta_i & a_i\cos\theta_i \\ sin\theta_i & \cos\alpha_i\cos\theta_i & -sin\alpha_i\cos\theta_i & a_i sin\theta_i \\ 0 & sin\alpha_i & \cos\alpha_i & d_i \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{3.1}$$

A posição e orientação do efetuador final é representada em relação a base pela matiz O_n^0 que está escrita como uma função dos deslocamentos das juntas. A matriz [3x3] formada pelas três primeiras linhas e colunas de (3.1) representa a orientação e a última coluna de três elementos representa a posição do sistema de coordenadas da base. Para sistemas mais simples podemos escrever esta relação utilizando apenas relações trigonométricas.

O Jacobiano geométrico [Amaral,2000] de um robô representa a relação entre as velocidades do sistema. O vetor de velocidade do efetuador final $\dot{p} = [\dot{p}_1,...,\dot{p}_n]^T = [\vec{v}_e, \vec{w}_e]^T$ pode ser escrito como função das velocidades nas juntas $\dot{q} = [\dot{q}_1,...,\dot{q}_n]^T$ pela equação 3.2.

$$\dot{p} = J\,\dot{q} \tag{3.2}$$

Onde J é uma matriz [6xn] (n = número de graus de liberdade) e representa o Jacobiano geométrico do robô.

Para uma junta prismática temos:

$$J = \begin{bmatrix} \vec{b}_{i-1} \\ \vec{0} \end{bmatrix} \tag{3.3}$$

Para uma junta de revolução temos:

$$J = \begin{bmatrix} \vec{b}_{i-1} \times \vec{r}_{i-1,e} \\ \vec{b}_{i-1} \end{bmatrix} \tag{3.4}$$

onde

$\vec{b}_{i-1} = R_{i-1}^{0} \cdot \vec{b}$

$\vec{b} = [0,0,1]^T$

R_{i-1}^{0} é uma matriz [3x3] de transformação de coordenadas da junta em análise, com relação ao sistema de coordenadas da origem.

$X_{i-1,e} = O_n^0 \, \overline{X} - O_{i-1}^0 \, \overline{X}$

$\overline{X} = [0,0,0,1]^T$

$\vec{r}_{i-1,e}$ é a matriz [3x3] oriunda das três primeiras colunas e linhas da matriz $X_{i-1,e}$ que é [4x4]

A relação de aceleração pode ser obtido pela derivação da definição de velocidades, como está apresentado na equação a seguir.

$$\ddot{p} = J\,\ddot{q} + \dot{J}\dot{q} \tag{3.5}$$

3.1.2 Cinemática Inversa

A cinemática inversa determina as posições, velocidades e acelerações das juntas do robô a partir das correspondentes posições, velocidades e acelerações do efetuador final. Os problemas de cinemática direta sempre têm uma única solução, porém os problemas de cinemática inversa podem ou não apresentar soluções. Também, se a solução existir normalmente não é única.

A obtenção da cinemática inversa de um sistema tem como pré requisito a cinemática direta e a solução analítica para este problema depende diretamente da configuração do robô.

Técnicas de análise da cinemática inversa são apresentadas em [CRAIG,1989], [FU,1987] e [SLOTINE,1991]. Basicamente existem duas linhas de estudos que se destacam, os métodos numéricos e os analíticos. Porém, os métodos são muito específicos e são aplicados de forma particular a cada sistema.

A cinemática inversa de velocidade pode ser obtida analisando a relação (3.2) da cinemática direta, obtendo-se:

$$\dot{q} = J^{-1}\dot{p} \tag{3.6}$$

da mesma forma, a cinemática inversa da aceleração pode ser escrita como:

$$\ddot{q} = J^{-1}\left(\ddot{p} - \dot{J}\dot{q}\right) \tag{3.7}$$

3.2 Modelagem Dinâmica

A dinâmica reúne as relações entre a cinemática e a cinética. Para se mover o robô ao longo de uma trajetória, forças e/ou torques devem ser exercidos pelos motores. No sentido de que uma trajetória planejada seja executada é preciso se determinar as forças e torques agindo sobre a

dinâmica das juntas do manipulador. No caso o manipulador plano de 2 graus de liberdade, Figura 3.4, é composto de duas juntas de rotação RR e os parâmetros do manipulador estão baseados em [SCIAVICCO,1996]. As massas m_1 e m_2 correspondentes a cada elo são iguais, g é a aceleração da gravidade, θ_1 e θ_2 ângulos das juntas e τ_1 e τ_2 são torques aplicados ao sistema.

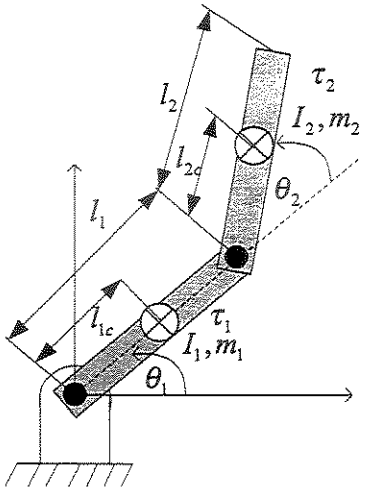


Figura 3.4. – Representação esquemática do sistema de 2 GDL

O manipulador robótico contém dois servomotores de acionamento sendo: um servomotor diretamente acoplado ao elo1 representando o torque τ_1 e um segundo servomotor ao elo2 que proporcionam os movimentos rotacionais ao manipulador .

3.2.1 Formulação de *Lagrange*

Nesta seção, será determinado o conjunto de equações dinâmicas que descreve o comportamento do sistema mecânico. Estas são as equações de movimento de *Lagrange*, que originam-se da diferença da energia cinética e potencial, conforme exposto por Craig(1989) e Spong(1989).

Energia cinética *(K)*

A energia cinética é o trabalho que deve ser executado sobre determinado corpo para levá-lo do repouso a uma certa velocidade.

$$k_1 = \frac{1}{2} m_1 l_{1c}^2 \dot{\theta}_1^2 + \frac{1}{2} I_1 \dot{\theta}_1^2$$

$$K_2 = \frac{1}{2} m_2 l_1^2 \dot{\theta}_1^2 + \frac{1}{2} m_2 l_{2c}^2 \left(\dot{\theta}_1^2 + 2\dot{\theta}_1\dot{\theta}_2 + \dot{\theta}_2^2\right) + m_2 l_1 l_{2c} \left(\dot{\theta}_1^2 + \dot{\theta}_1\dot{\theta}_2\right)\cos\theta_2 + \frac{1}{2} I_2 \left(\dot{\theta}_1^2 + 2\dot{\theta}_1\dot{\theta}_2 + \dot{\theta}_2^2\right)$$

Energia cinética total $K(\theta, \dot{\theta})$

$$K(\theta,\dot{\theta}) = \frac{1}{2} m_1 l_{1c}^2 \dot{\theta}_1^2 + \frac{1}{2} I_1 \dot{\theta}_1^2 + \frac{1}{2} m_2 l_1^2 \dot{\theta}_1^2 + \frac{1}{2} m_2 l_{2c}^2 \left(\dot{\theta}_1^2 + 2\dot{\theta}_1\dot{\theta}_2 + \dot{\theta}_2^2\right) + m_2 l_1 l_{2c} \left(\dot{\theta}_1^2 + \dot{\theta}_1\dot{\theta}_2\right)\cos\theta_2 + \frac{1}{2} I_2 \left(\dot{\theta}_1^2 + 2\dot{\theta}_1\dot{\theta}_2 + \dot{\theta}_2^2\right)$$

Energia potencial *(U)*

É a energia que um corpo tem em virtude de sua posição ou estado e é capaz de realizar trabalho. Podemos classificar a energia potencial em: energia potencial elástica e energia potencial gravitacional.

A energia potencial elástica é a energia armazenada em uma mola comprimida ou distendida, capaz de retorná-la a seu estado inicial.

A energia potencial gravitacional é aquela que está relacionada com a posição que um corpo ocupa no campo gravitacional terrestre e sua capacidade de vir a realizar trabalho mecânico, como elevar um determinado corpo de uma distância h acima de algum plano arbitrário de referência.

$$U_{elástica} = \frac{1}{2}K(X - X_0)^2$$

$$U_{gravitacional} = mg\Delta h$$

$$U_1 = m_1 g l_1 sin\theta_1$$

$$U_2 = m_2 g l_1 sin\theta_1 + m_2 g l_{2c} sin(\theta_1 + \theta_2)$$

Energia potencial total $U(\theta)$

$$U(\theta) = m_1 g l_1 sin\theta_1 + m_2 g l_1 sin\theta_1 + m_2 g l_{2c} sin(\theta_1 + \theta_2)$$

Define-se o Lagrangeano L do sistema por:

$$L(\theta, \dot{\theta}) = K - U$$

$$L = \frac{1}{2}m_1 l_{1c}^2 \dot{\theta}_1^2 + \frac{1}{2}I_1 \dot{\theta}_1^2 + \frac{1}{2}m_2 l_1^2 \dot{\theta}_1^2 + \frac{1}{2}m_2 l_{2c}^2 (\dot{\theta}_1^2 + 2\dot{\theta}_1\dot{\theta}_2 + \dot{\theta}_2^2) + m_2 l_1 l_{2c} (\dot{\theta}_1^2 + \dot{\theta}_1\dot{\theta}_2)\cos\theta_2$$
$$+ \frac{1}{2}I_2 (\dot{\theta}_1^2 + 2\dot{\theta}_1\dot{\theta}_2 + \dot{\theta}_2^2) - m_1 g l_1 sin\theta_1 - m_2 g l_1 sin\theta_1 - m_2 g l_{2c} sin(\theta_1 + \theta_2)$$

Derivadas parciais e equações de movimento (*Euler-Lagrange*):

Corpo 1

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\theta}_1} - \frac{\partial L}{\partial \theta_1} = \tau_1$$

$$H_{11}\ddot{\theta}_1 + H_{12}\ddot{\theta}_2 - h\dot{\theta}_1\dot{\theta}_2 - h(\dot{\theta}_1 + \dot{\theta}_2)\dot{\theta}_2 + g_1 = \tau_1$$

Corpo 2

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\theta}_2} - \frac{\partial L}{\partial \theta_2} = \tau_2$$

$$H_{21}\ddot{\theta}_1 + H_{22}\ddot{\theta}_2 - h\dot{\theta}_1 + g_2 = \tau_2$$

expressando em forma matricial temos

$$\begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix}\begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{bmatrix}+\begin{bmatrix} -h\dot{\theta}_1 & -h\dot{\theta}_1 - h\dot{\theta}_2 \\ h\dot{\theta}_1 & 0 \end{bmatrix}\begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix}+\begin{bmatrix} g_1 \\ g_2 \end{bmatrix}=\begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix}$$

onde:

$$H_{11} = m_1 l_{1c} + I_1 + m_2[l_1^2 + l_{2c}^2 + 2l_1 l_{2c}\cos\theta_2] + I_2$$

$$H_{22} = m_2 l_{2c}^2 + I_2$$

$$H_{12} = H_{21} = m_2 l_1 l_{2c}\cos\theta_2 + m_2 l_{2c}^2 + I_2 \qquad \text{e} \qquad h = m_2 l_1 l_{2c} sin\theta_2$$

As tabelas a seguir mostram o equacionamento da cinemática e dinâmica para o manipulador

Tabela 3.1 - Equacionamento Cinemático

	Cinemática direta	**Cinemática Inversa**
Posição	$\begin{cases} x = l_1 c_1 + l_2 c_{12} \\ y = l_1 s_1 + l_2 s_{12} \end{cases}$	$\begin{cases} \theta_2 = \tan^{-1}\left(\dfrac{\pm\sqrt{1-D^2}}{D}\right) \\ \theta_1 = \tan^{-1}\left(\dfrac{y}{x}\right) - \tan^{-1}\left(\dfrac{l_2 sin\theta_2}{l_1 + l_2\cos\theta_2}\right) \end{cases}$ $D = \dfrac{x^2 + y^2 - l_1^2 - l_2^2}{2l_1 l_2}$
velocidade	$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = J\begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix}$ $J = \begin{bmatrix} -l_1 s_1 - l_2 s_{12} & -l_2 s_{12} \\ l_1 c_1 + l_2 c_{12} & l_2 c_{12} \end{bmatrix}$	$\begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} = J^{-1}\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix}$ $J^{-1} = \dfrac{1}{l_1 l_2 S_2}\begin{bmatrix} l_2 c_{12} & l_2 s_{12} \\ -(l_1 c_1 + l_2 c_{12}) & -(l_1 c_1 + l_2 s_{12}) \end{bmatrix}$
Aceleração	$\begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} = J\begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{bmatrix} + \dot{J}\begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix}$ $\dot{J} = \begin{bmatrix} -\left(l_1 c_1\dot{\theta}_1 + l_2 c_{12}\dot{\theta}_{12}\right) & -l_2 c_{12}\dot{\theta}_{12} \\ -\left(l_1 s_1\dot{\theta}_1 + l_2 s_{12}\dot{\theta}_{12}\right) & -l_2 s_{12}\dot{\theta}_{12} \end{bmatrix}$	$\begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} = J^{-1}\left[\begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} - \dot{J}\begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix}\right]$

Tabela 3.2 – Equacionamento dinâmico

$\begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix}\begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{bmatrix}+\begin{bmatrix} -h\dot{\theta}_1 & -h\dot{\theta}_1 - h\dot{\theta}_2 \\ h\dot{\theta}_1 & 0 \end{bmatrix}\begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix}+\begin{bmatrix} g_1 \\ g_2 \end{bmatrix}=\begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix}$
$H_{11} = m_1 l_{1c} + I_1 + m_2[l_1^2 + l_{2c}^2 + 2l_1 l_{2c}\cos\theta_2] + I_2$ $H_{22} = m_2 l_{2c}^2 + I_2$ $H_{12} = H_{21} = m_2 l_1 l_{2c}\cos\theta_2 + m_2 l_{2c}^2 + I_2$ $h = m_2 l_1 l_{2c} sin\theta_2$

3.3 Geração de trajetória

Como foi mencionado anteriormente, em muitas aplicações a programação de tarefas de robôs é realizada no espaço das juntas e da trajetória angular.

Para que se possa lidar com o problema de movimentação de manipuladores, é necessário o desenvolvimento de um planejador de trajetórias, responsável por calcular o histórico temporal das posições, velocidades e acelerações desejadas, partindo de uma posição inicial, no tempo t_i, para uma posição final, no tempo t_f, podendo ou não possuir especificações intermediárias de posição.

Para que uma trajetória possa ser realizada fisicamente pelo manipulador, ela deve obedecer certas restrições impostas pelas características cinemáticas e dinâmicas do mesmo. Os valores assumidos pelas trajetórias de referência, além de finitos, devem ser suportados pela estrutura mecânica do manipulador.

As trajetórias de referência para robôs manipuladores podem ser geradas considerando que o movimento seja feito ponto-a-ponto, quando apenas as posições inicial e final desejadas em um

certo período de tempo, ou por caminho contínuo (*path motion*), quando é fornecida uma seqüência finita de pontos ao longo do caminho desejado.

3.3.1 Discretização do caminho

A discretização tem por objetivo fazer com que o elemento terminal siga um caminho pré-determinado definido entre a posição inicial e a posição final desejada no espaço, portanto o caminho é discretizado em m partes.

A Figura 3.5 mostra a discretização entre o ponto inicial da trajetória $\vec{x}_i$ e o ponto final $\vec{x}_f$ para que o elemento terminal do robô siga uma linha reta. Deste modo então, o algoritmo de geração de trajetória calcula a trajetória da posição inicial até a posição intermediária e, após ela ser atingida, inicia-se o cálculo da trajetória desta posição até a posição intermediária posterior. Isto é realizado até que a posição final seja alcançada.

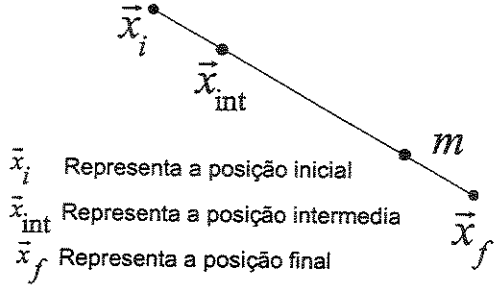


Figura 3.5 – Discretização do caminho em m partes

3.3.1.1 Discretização Linear

Discretiza o caminho desejado, em m partes, de forma linear, fazendo desta forma com que o elemento terminal do robô siga uma linha reta.

Seja (X_1, Y_1) o vetor posição inicial do elemento terminal do manipulador robótico e (X_2, Y_2) o seu vetor posição final.

Para discretizar a posição tem-se as seguintes equações:

$$x_i = x_{i-1} + \mathrm{Inc}_1$$
$$y_i = y_{i-1} + \mathrm{Inc}_2$$

onde

$$i = 1,2,\ldots, m;$$

x_i *e* y = pontos intermediários do vetor posição

$$\mathrm{Inc}_1 = (X_1 - X_2)/m;$$
$$\mathrm{Inc}_2 = (Y_1 - Y_2)/m;$$

e m é o número de partes em que o caminho será dividido. O vetor (x_i, y_i) é denominado vetor posição intermediário.

As equações acima podem ser estendidas a um sistema não planar a seguir

$$x_i = x_{i-1} + \mathrm{Inc}_1$$
$$y_i = y_{i-1} + \mathrm{Inc}_2$$
$$z_i = z_{i-1} + \mathrm{Inc}_3$$

e

$$Inc_1 = (X_1 - X_2)/m;$$
$$Inc_2 = (Y_1 - Y_2)/m;$$
$$Inc_3 = (Z_1 - Z_2)/m;$$

3.3.1.2 Discretização em semicírculo

Discretiza o caminho desejado em forma de um semicírculo. A equação de um círculo é dado por:

$$(x-h)^2 + (y-k)^2 = r^2$$

os tipos de discretizações possíveis em semicírculo são:

- no plano X-Y podendo variar o valor da posição Z e com Y=f(X).
- no plano X-Z podendo variar o valor da posição Y e com Z=f(X).
- no plano Y-Z podendo variar o valor da posição X e com Y=f(Z).

Seja (X_1,Y_1,Z_1) o vetor posição inicial do elemento terminal do robô e (X_2,Y_2,Z_2) o seu vetor posição final.

Temos então para a discretização em semicírculo, entre a posição inicial e a posição final, as seguintes equações:

$y_i = \left(r_1^2 - (c_1 - h_1)^2\right)^{1/2} + k_1$ semicírculo no plano x-y

$z_i = \left(r_2^2 - (c_2 - h_2)^2\right)^{1/2} + k_2$ semicírculo no plano x-z

$z_i = \left(r_3^2 - (c_3 - h_3)^2\right)^{1/2} + k_3$ semicírculo no plano y-z

onde

$i = 1,2,..., m;$

$r_1 = abs(X_2 - X_1) / 2;$

$r_2 = abs(X_2 - X_1) / 2;$

$r_3 = abs(Y_2 - Y_1) / 2;$

$h_1 = X_1;$ $\quad k_1 = Y_1;$

$h_2 = X_1;$ $\quad k_2 = Z_1;$

$h_3 = Y_1;$ $\quad k_3 = Z_1;$

$c_1 = x_{1-1} + Inc_1$ onde $Inc_1 = (X_1 - X_2) / m;$

$c_2 = y_{1-1} + Inc_2$ onde $Inc_2 = (X_1 - X_2) / m;$

$c_3 = y_{1-1} + Inc_3$ onde $Inc_3 = (Y_1 - Y_2) / m;$

e m é o número de partes em que o caminho será dividido.

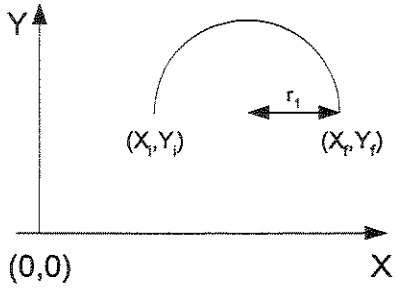


Figura 3.6- Discretização do caminho em semicírculo no plano x-y

Capítulo 4

Resultados Simulados e Experimentais

Este capítulo mostra os resultados obtidos referentes à análise do manipulador robótico. Para isso são executadas as simulações, no ambiente do MATLAB® devido à sua característica de possibilitar a confiabilidade e facilidade de uso. Também são apresentados os resultados experimentais.

4.1 Algoritmo para a Geração de Trajetória

De acordo com a Figura 4.1, o algoritmo para a geração de trajetórias do manipulador planar de 2GDL é iniciado com a introdução dos pontos inicial e final da trajetória. Além destes é necessário fornecer o conjunto de pontos gerados a partir dos incrementos dos ângulos de cada junta de rotação R, que serão utilizados no treinamento das redes neurais. Estas redes do tipo *perceptron*, determinarão as coordenadas generalizadas de cada uma das juntas de rotação e as coordenadas operacionais dos extremos de cada segmento do manipulador do tipo RR. Em um segundo passo estas mesmas redes permitirão obter a trajetória.

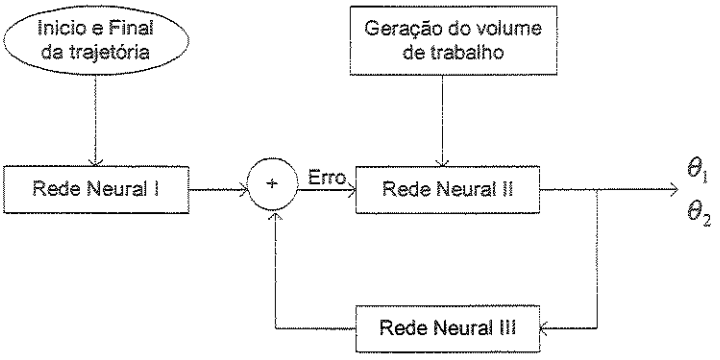


Figura 4.1 - Representação do algoritmo para geração da trajetória

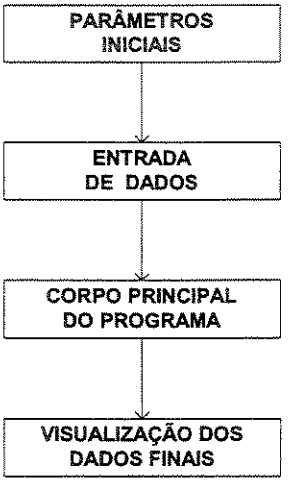


Figura 4.2 - Algoritmo simplificado para a geração de trajetórias

No algoritmo simplificado da Figura 4.2, no bloco denominado parâmetros iniciais é feita a escolha da discretização, escolhido o número de segmentos da tarefa a ser realizada, o número de divisões de cada segmento e determinados os erros máximos de posição.

No bloco denominado entrada de dados é dado o valor da posição final desejada para cada tarefa a ser realizada. A posição final de cada segmento torna-se a inicial para o outro segmento.

No bloco denominado corpo principal do programa é realizada a divisão do segmento em *m* pontos utilizando a discretização escolhida. Cada ponto passa a ser uma posição final desejada. E realizado o cálculo do erro de posição. Neste ponto o algoritmo pode armazenar o conjunto de ângulos para a posição atual, caso os erros sejam menores que os desejados.

No bloco denominado dados finais é obtido o conjunto de ângulos (trajetória desejada) em arquivos com extensão **dat**.

A rede neural 1 é formada por 2 neurônios, um para cada coordenada cartesiana, ambos com função de ativação linear. Define-se X_2 e Y_2 como os vetores que contém os pontos que formam o volume de trabalho do manipulador, ou seja, as coordenadas cartesianas que representam a posição do elemento terminal. A saída da rede 1 é a entrada da rede 2, e juntamente com a saída da rede 3, formará o erro (vide Figura 4.1).

O vetor de entrada para o treinamento da rede neural 1 é formado pelo ponto inicial (X_i, Y_i) e final da trajetória (X_f, Y_f), de acordo com a Figura 4.3, onde *b* é o bias, +/- $1/\Delta s$ são os pesos sinápticos, s é simplesmente uma variável qualquer e (X_1, Y_1) é a saída da rede. Os pontos inicial e final devem estar contidos no volume de trabalho.

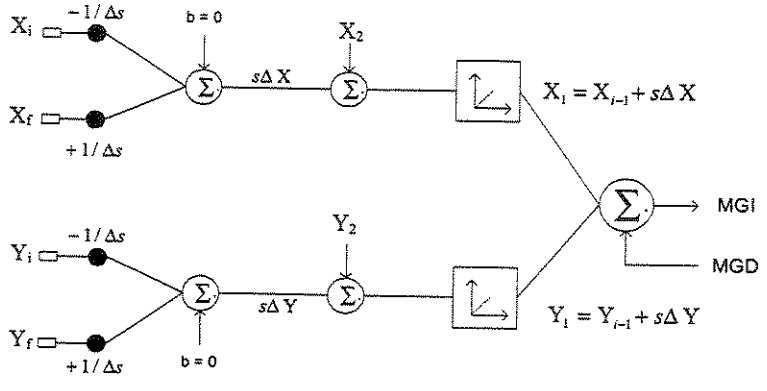


Figura 4.3 - Rede neural 1

Para o treinamento da rede 2, que representa o MGI, foi utilizado o vetor que contém os pares ordenados, $p(x_2,y_2)$, gerados a partir da equação cinemática do modelo geométrico direto, com incrementos de $\Delta\theta_1$ radianos para θ_1 e de $\Delta\theta_2$ radianos para θ_2. A saída desejada corresponde ao vetor $Ti[\theta_1,\theta_2]$, que é o vetor alvo a ser alcançado no treinamento. A dimensão do vetor X_2, que é igual a dimensão do vetor Y_2, é função dos incrementos de $\Delta\theta_1$ e de $\Delta\theta_2$.

Para o treinamento da rede neural 3, que representa o MGD, foi utilizado o vetor $Pd[\theta_1,\theta_2]$, gerado a partir dos incrementos de θ_1 e θ_2. A saída desejada corresponde ao vetor alvo $Td[X_2,Y_2]$, gerado do modelo geométrico direto.

Na Figura 4.4, apresenta as redes 2 e 3, do tipo MLP. compostas por duas camadas intermediárias, a primeira com 5 neurônios e a segunda com 10 neurônios e dois neurônios na camada de saída. O treinamento de ambas redes neurais foi realizado através do método de retropropagação do erro, ou *Backpropagation* [HAYKIN,1999]. A função de ativação adotada nas duas camadas intermediárias é do tipo sigmóide, e a função de ativação da camada de saída é do tipo linear. Por muitas tentativas realizadas experimentalmente adotou-se a rede 2-5-10-2 por ser relativamente rápida para o treinamento e sua resposta é precisa com uma acentuada convergência. Deve ser observado que o vetor de entrada tem dimensão Nx2, onde N é o número de linhas, sendo função dos incrementos dos ângulos das juntas.

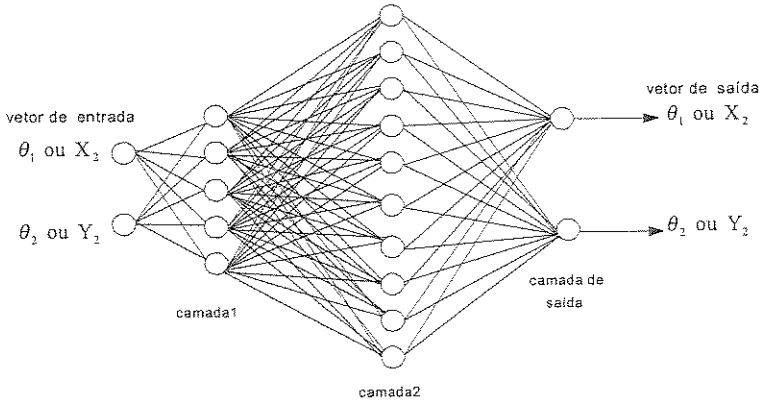


Figura 4.4 Rede neural do tipo MLP.

O plano bidimensional XY foi dividido em setores, de tal modo que o volume de trabalho, Figura 4.5, fosse distribuído entre eles. Cada setor corresponde a um incremento de $\Delta\theta_1$ e o número de pontos por setor é função do incremento $\Delta\theta_2$. Quanto menor for o incremento na primeira junta de rotação, maior será o número de setores e, quanto menor for o incremento na segunda junta de rotação, maior será o número de pontos $P(x_2,y_2)$ por setor. Ao dividir o volume de trabalho em setores com um número inteiro associado a cada um deles, nota-se que alguns destes setores terão uma concentração de pontos maior em relação aos demais, devido a superposição dos setores. Quanto maior a densidade de pontos em um setor, maior será a precisão ao alcançar este ponto.

Para a geração do volume de trabalho, optou-se por analisá-lo somente no primeiro quadrante, por simplicidade de visualização.

Para realizar a trajetória desejada dentro do volume de trabalho, no caso, uma reta com inicio e extremidade respectivamente em (X_i,Y_i) e $e(X_f,Y_f)$, o algoritmo gera o volume de trabalho e treina as redes 2 e 3 para determinar a qual setor um dos pontos pertence; caso o ponto não pertença ao volume de trabalho, a rede retorna o valor zero para o respectivo setor e o ponto não é armazenado.

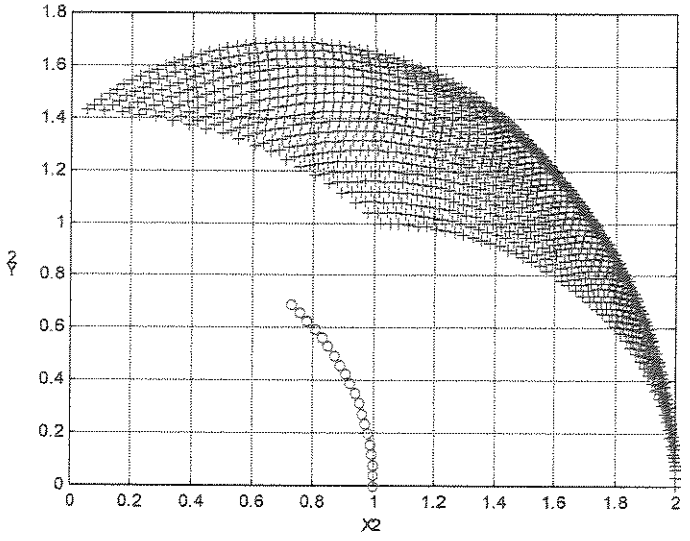
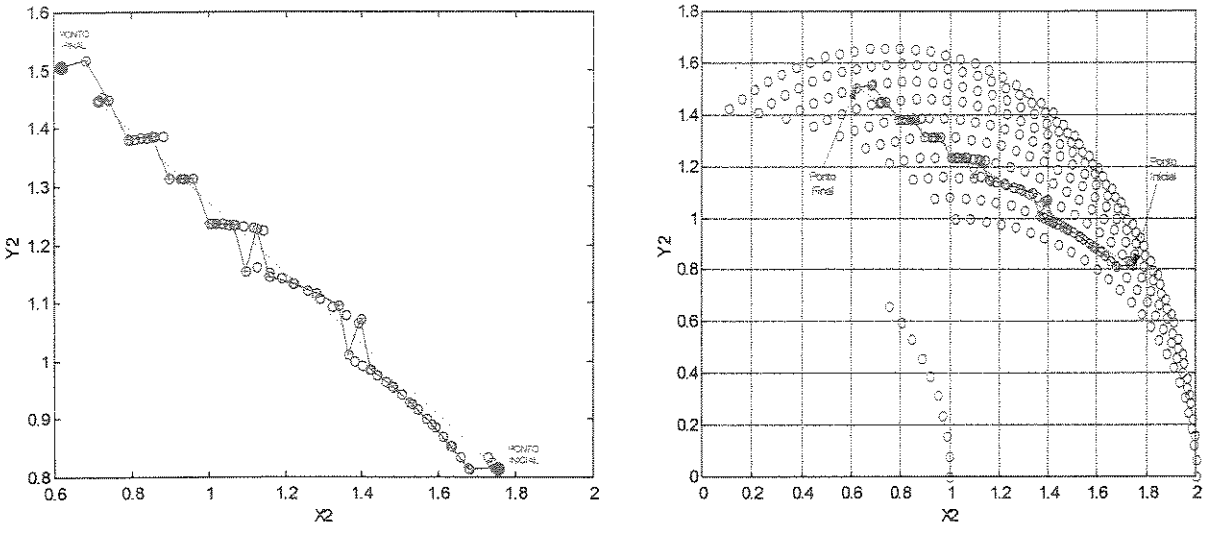


Figura 4.5 - Volume de trabalho do manipulador RR planar

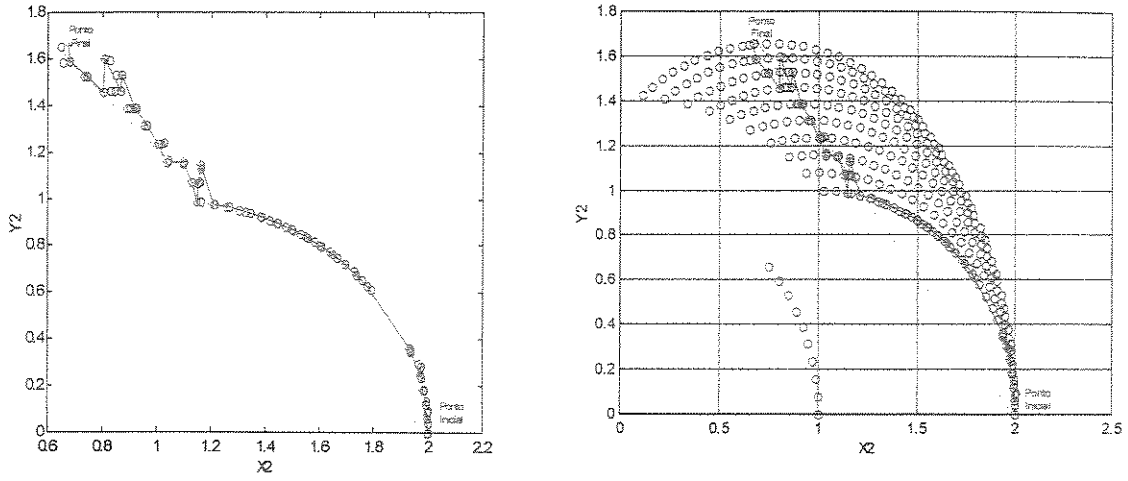
A rede1 tem por função determinar o conjunto de pontos(X_1,Y_1) que definem exatamente uma reta com início e final da trajetória. Este conjunto de pontos é formado através de incrementos $s\Delta x$ e $s\Delta y$ somados ao ponto inicial (X_i,Y_i) até atingir o ponto final (X_f,Y_f). Somente são armazenados os pontos (X_1,Y_1) dentro do volume de trabalho. Em seguida todos os pontos determinados pelas redes 2 e 3, (X_2,Y_2) são apresentados à rede1. Os pontos (X_2,Y_2) que apresentarem o menor erro em relação aos pontos (X_1,Y_1) são selecionados como pontos da trajetória do manipulador. Em seguida os pontos (X_2,Y_2) selecionados, são ligados entre si dentro do volume de trabalho, gerando uma aproximação da trajetória exata. O vetor de saída do modelo geométrico inverso $Ti[\theta_{r1},\theta_{r2}]$ também é obtido através da rede neural e é armazenado.



(a) (b)

Figura 4.6 - Trajetória executada dentro do volume de trabalho

Na Figura 4.6a e 4.6b, a trajetória está contida totalmente no volume de trabalho. Em ambos os casos a trajetória exata é representada por uma linha reta. O ângulo da primeira junta de rotação foi variado de $0 < \theta_1 < \pi/4$ e o ângulo da segunda junta foi variado de $0 < \theta_2 < \pi/2$. Foram adotados os seguintes valores para as constantes que representam as dimensões do braço e antebraço, l_1=1 e l_2=1, de acordo com a Figura 3.4.



(a) (b)

Figura 4.7 - Trajetória parcialmente dentro do volume de trabalho

A Figura 4.7a e 4.7b,mostra a trajetória para o caso em que está contida parcialmente no volume de trabalho. A trajetória exata é representada por uma linha reta. O ângulo da primeira junta de rotação foi variado de $0 < \theta_1 < \pi/4$ e o ângulo da segunda junta rotacional foi variado de $0 < \theta_2 < \pi/2$. Os valores para as constantes que representam as dimensões do braço e antebraço, são l_1=1 e l_2=1, como no caso anterior.

4.2 Validação Experimental

Para possibilitar a verificação e validação do algoritmo desenvolvido foi realizado o controle da trajetória no robô experimental não planar Robix™ RCS-6, Figura 4.8. Através dos experimentos realizados nele foi possível implementar e testar o software para o controle dos servomotores, necessários para a realização da trajetória. Os materiais utilizados na parte experimental foram o microcomputador Pentium III© 500MHz 128 Mb RAM, fonte simétrica estabilizada de corrente contínua Digilab 5Va2A, placa de controle dos servomotores, software de interface necessário para envio dos ângulos de referência gerados no Matlab©.

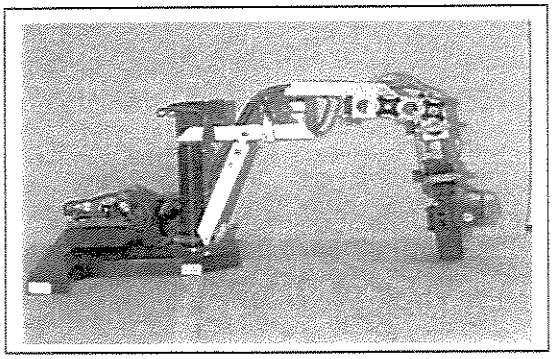


Figura 4.8 - Manipulador robótico Robix™ RCS-6

4.2.1 Descrição da Bancada Experimental

A bancada experimental mostrada na Figura 4.9 consiste de: manipulador robótico Robix™ RCS-6, microcomputador, fonte de alimentação e uma placa controladora de servomotores.

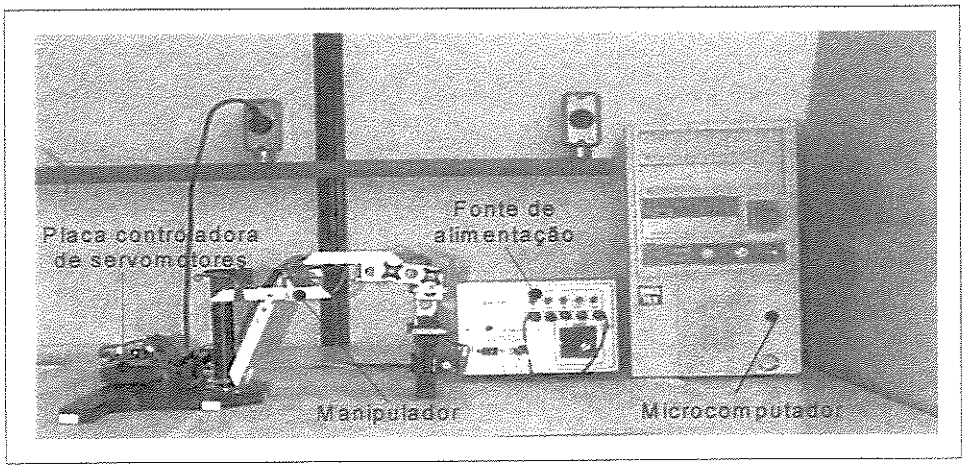


Figura 4.9 - Vista frontal da bancada experimental

4.2.1.1 Placa de Controle dos Servomotores

A placa de controle usada é o módulo *SbServoControl*© [Solbet,2002] que substituiu a interface eletrônica original do Robix™ RCS-6, esta placa é capaz de controlar até oito servomotores simultaneamente e é do tipo utilizado em aeromodelismo, Figura 4.10. A placa é conectada ao microcomputador do tipo PC através de um cabo serial com conector DB9 utilizando o protocolo de *hardware* RS232-C. a transmissão de dados é realizada a 4800 Kbps.8.N.1 (protocolo de transmissão de dados 8N1, 8 *databits, parity NONE* e 1 *stop bit*.).

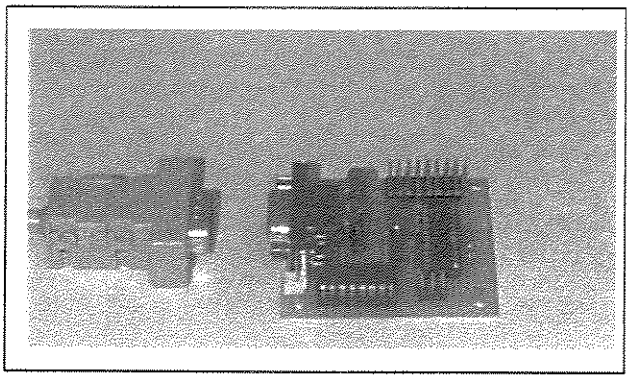


Figura 4.10 - Placa de controle *SbServoControl*©

A placa de controle *SbServoControl*© é baseada no microcontrolador programável PIC 16F84© [Microchip, 2002]. O PIC (*Peripheral Interface Controller*) é um circuito integrado provido de uma memória programável, porta de entrada e saída, timer, contadores, comunicação serial, PWM, conversores AD, entre outros, encapsulados em uma única pastilha de silício [Microchip, 2002]. A estrutura de máquina interna é do tipo *Harvard* que é caraterizada por dois barramentos internos, um para dados e outro para instruções. O barramento de dados é sempre de 8 bits e o de instruções de 14 bits. Além disso, o PIC 16F84 possui 18 pinos, 13 portas configuráveis como entrada ou saída, 4 interrupções disponíveis, memória de programação que

permite a regravação do programa diversas vezes (EEPROM Flash), memória não volátil interna (EEPROM) e 35 instruções de programação.

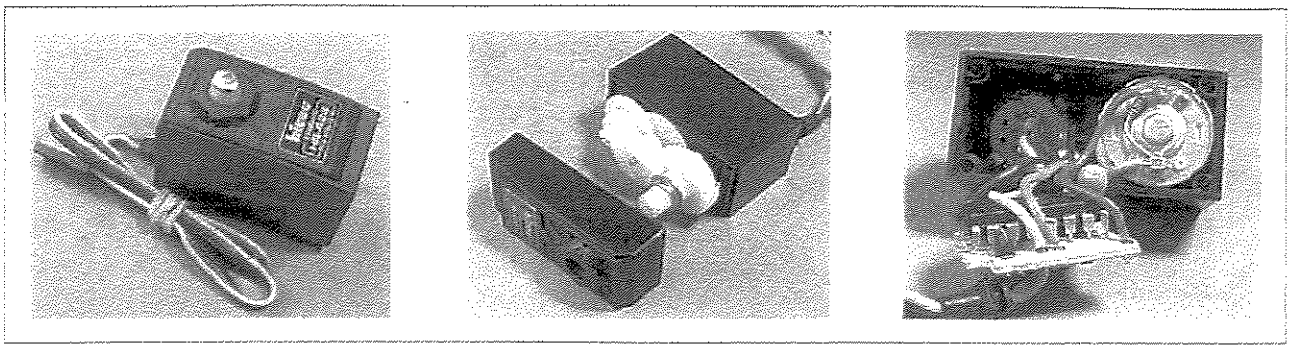
Foram utilizados os servomotores do tipo Hitec HS422 © Figura 4.11. Os servomotores utilizam um pequeno motor DC que é controlado por um circuito eletrônico e um dispositivo de realimentação constituído por um potenciômetro conectado mecanicamente ao eixo do motor DC e eletricamente ao circuito eletrônico. A variação da resistência do potenciômetro em função da rotação informa ao circuito eletrônico a rotação e direção do motor.

Ao girar em determinada direção o motor DC, move uma série de engrenagens que amplificam e transferem o torque do motor ao eixo externo onde são conectados os controles mecânicos. Se o motor fosse conectado diretamente ao eixo de controle, o tamanho físico do motor teria que ser 10 vezes maior que o apresentado pelos servomotores utilizados, além disso, não seria possível obter baixas rotações.

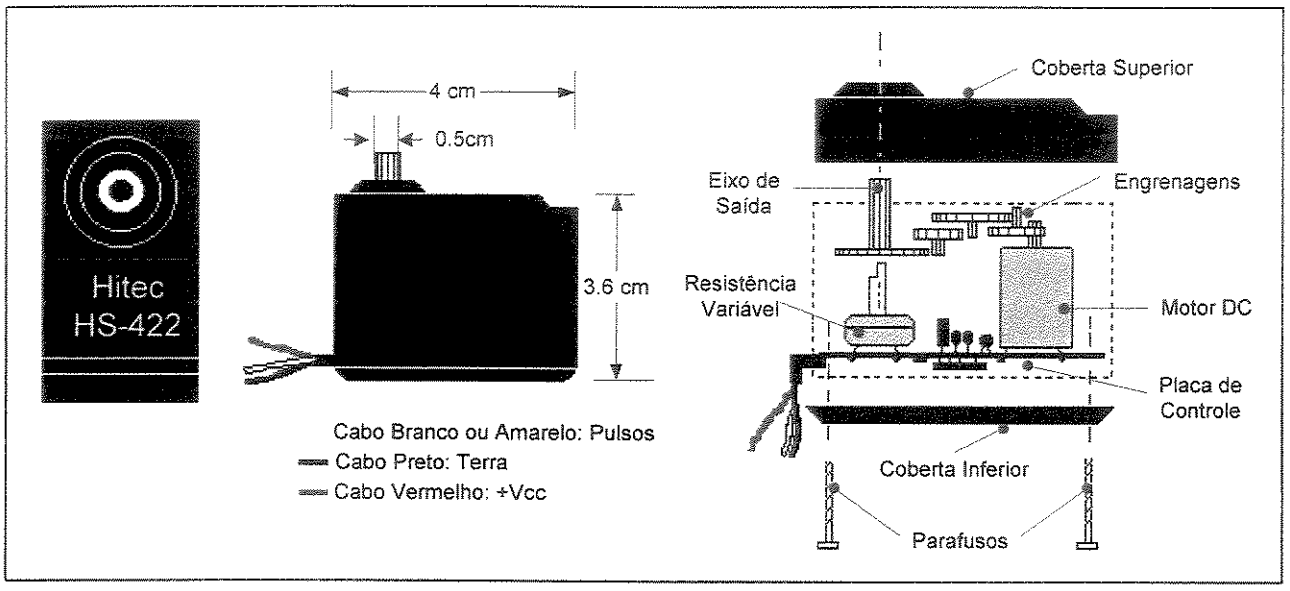
O torque dos servomotores que utilizamos neste trabalho é medido em onça por polegadas , para maior facilidade converteremos as unidades para g/cm (gramas/centímetro). O servomotor Hitec HS422 apresenta um torque de 290 g/cm . Isto significa que com uma polia, engrenagem ou braço de 4,2 mm de raio ligada ao eixo do servomotor , o servomotor estará apto a levantar até 1200 gramas.

Devido ao tipo de aplicação a que se destinam, estes tipos de servos possuem um controle muito simples. O controlador utilizado é analógico, tipo proporcional discreto no tempo, um resistor variável ligado ao eixo de saída serve como *encoder*.

O servomotor utilizado possui internamente um circuito eletrônico responsável pelo posicionamento, em um determinado ângulo dos respectivos sinais recebidos, de tal forma que o tipo de controle interno no servo é de malha fechada sendo que não há qualquer *feedback* externo.



(a)



(b)

Figura 4.11 - Servomotor Hitec- HS422

O servomotor recebe sinais de comando PWM com nível de tensão entre 0 e +5V, cujo período em estado alto, variando de 1 a 2 milisegundos, determina a posição comandada.

Seu controlador não possui memória, logo a ação de controle somente ocorre imediatamente após o envio de pulsos comandados sendo preciso repetir periodicamente este sinal para manter o servo na posição desejada. Os pulsos devem se repetir a cada 10 - 20 ms com

uma duração entre 0.3 - 2.1ms tal como mostra a Figura 4.12, porém o intervalo de tempo entre os pulsos mantêm-se constante, e a variação da largura dos mesmos indica ao servo a posição desejada.

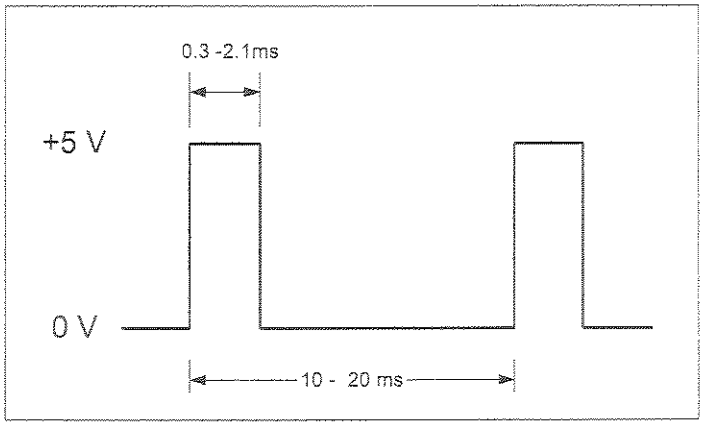


Figura 4.12 - Sinal de pulsos do servomotor

4.2.1.2 Software de Interface

Para que o manipulador consiga realizar a trajetória é necessário gerar o vetor com os ângulos de referência a ser seguidos por cada um dos servomotores. Os ângulos necessários para realizar a trajetória foram gerados no Matlab®, estes ângulos são salvos num arquivo com extensão *.dat, contendo os ângulos em graus, denominados de padrões de movimento. Em seguida estes valores são enviados ao manipulador através da porta serial de saída RS232-C do computador. Para tanto foi o *software* rb2.exe desenvolvido para o ambiente Windows® .

O *software* de interface rb2.exe [Benzatti-Zampieri,2001] foi realizado em linguagem computacional *C Builder 3*®. Através dele pode-se ler e enviar os arquivos dos ângulos. O software converte em graus para o protocolo de entrada da placa de controle *SbServoControl*® a qual converte os valores recebidos em largura de pulsos correspondentes a cada ângulo.

4.3 Modelo Matemático

O modelo geométrico correspondente para o manipulador, Robix™ RCS-6, RR não planar é dado pela equação (4.1) correspondente à Figura 4.13

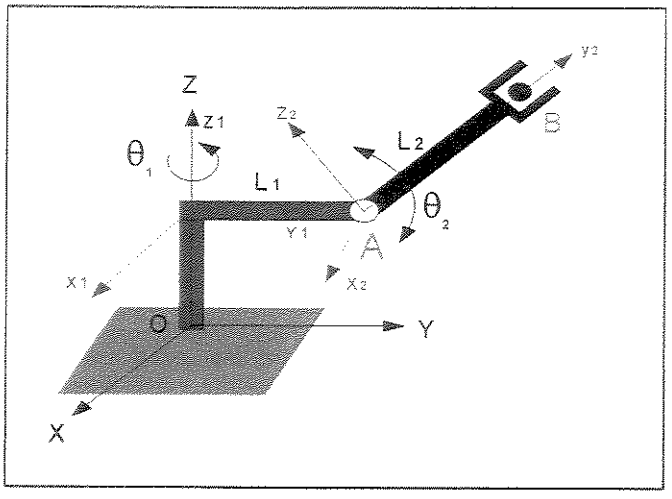


Figura 4.13 - Modelo geométrico do manipulador não planar

$$
\begin{aligned}
X_B &= \cos\theta_1(L_2\cos\theta_2 + L_1) \\
Y_B &= \sin\theta_1(L_2\cos\theta_2 + L_1) \\
Z_B &= L_2 sin\theta_2 + H
\end{aligned}
\qquad (4.1)
$$

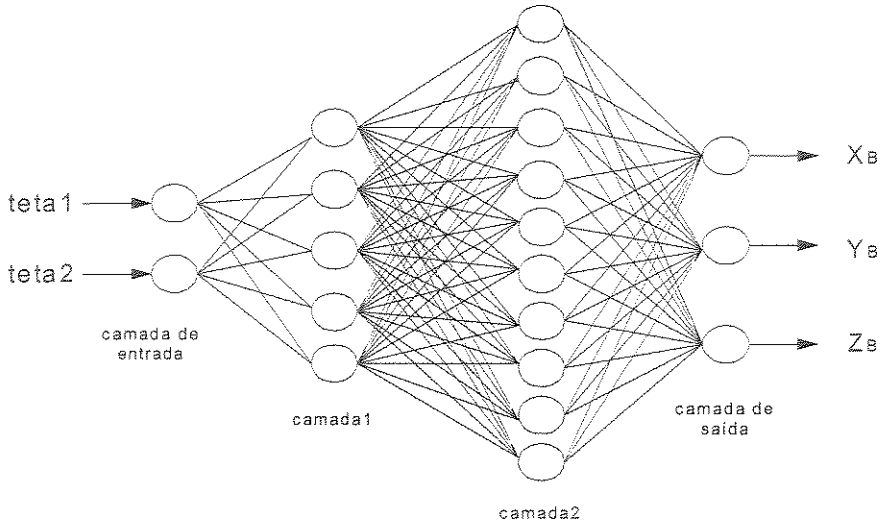


Figura 4.14 - Rede Neural de múltiplas camadas para o manipulador não planar

Para o mapeamento do volume de trabalho e a geração da trajetória está sendo aplicada uma rede neural de múltiplas camadas, Figura 4.14, composta por duas camadas intermediárias, a primeira com 5 neurônios e a segunda com 10 neurônios e pela camada de saída. Ao realizar o mapeamento e geração da trajetória do manipulador robótico optou-se pela utilização de algumas funções do "Toolbox" de redes neurais do ambiente Matlab[], ou parte delas como: INITFF - cria valores iniciais para os pesos e bias de uma rede *feed-forward*; TRAINLM – trainamento da rede *feed-forward* utilizando o método Levenberg – Marquardt, buscando-se uma somatória de erro quadrático na saída de 0.001 em amplitude; SIMUFF - simula a rede *feed-forward*; TANSIG - função de ativação não linear tangente hiperbólica na camada intermediária; PURELIN - função de ativação linear na camada de saída.

O gráfico abaixo representa o mapeamento simulado do volume de trabalho do manipulador

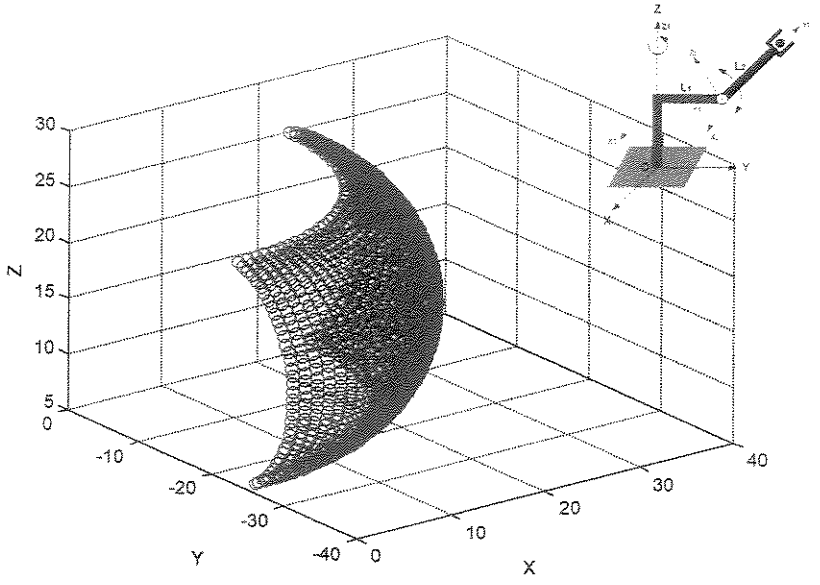


Figura 4.15 - Volume de trabalho do manipulador não planar Robix™ RCS-6

4.4 Trajetória Simulada

Esta seção mostra o resultado da trajetória simulada a qual está contida totalmente dentro do volume de trabalho do manipulador robótico. O ângulo da primeira junta de rotação foi variado de $0 < \theta_1 < \pi/2$ e o ângulo da segunda junta foi variado de $-\pi/3 < \theta_2 < 5\pi/12$. Foram adotados os seguintes valores para as constantes que representam as dimensões do braço e antebraço, l_1 =20.4cm e l_2 =10.5cm com uma altura h=15cm., a trajetória simulada será uma tarefa simples que consiste em pegar um objeto de una posição inicial e levar até uma posição final, está tarefa pode ser realizada o número de vezes que o usuário veja por conveniente.

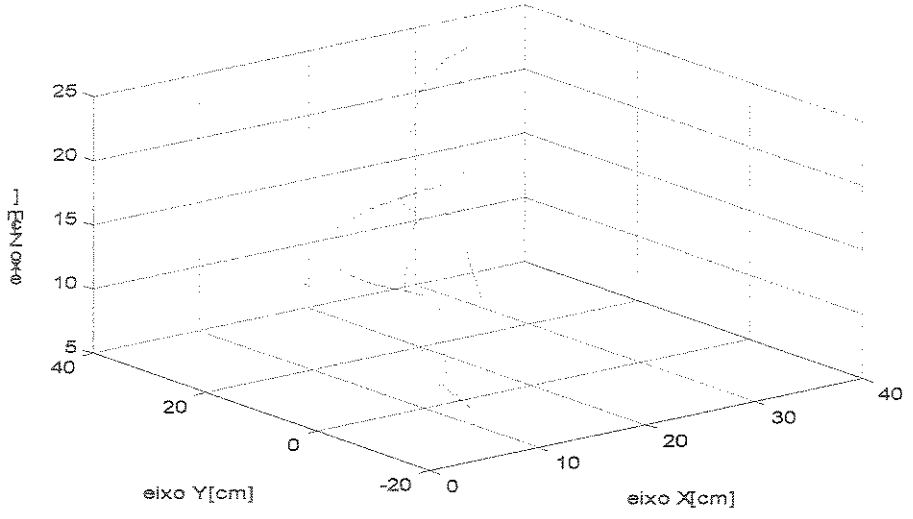


Figura 4.16 - Trajetória realizada dentro do volume de trabalho

4.5 Trajetória Experimental

Esta seção trata da obtenção da trajetória experimental do manipulador robótico, devido ao tipo de servomotores utilizado neste trabalho, e por eles possuirem um controle interno, sendo que não há qualquer tipo de controle externo para determinar se o manipulador realizou ou não a trajetória desejada. Uma das formas para realizar a verificação seria instrumentar o manipulador robótico com *encoders* para adquirir as posições reais dos ângulos de rotação correspondente a cada junta de rotação, mas este método foi inviável devido às dimensões do manipulador, então optou-se pela filmagem e o posterior processamento da imagem, para a reconstrução da respectiva trajetória em 3D, para testar a veracidade do programa desenvolvido e apresentado. A seguir serão abordados algum conceitos relacionados a este tema.

4.5.1 Calibração da Câmera : Conceito

Calibração da câmera segundo [Lang,1971] é um processo preliminar para calcular os parâmetros físicos da câmera (parâmetros extrínsecos), como centro da imagem, posição e orientação, etc, (calibração explícita). Alguns parâmetros ópticos e geométricos internos da câmera (parâmetros intrínsecos) podem ser calibrados para uso em medições 3D. Esta é a calibração implícita. Aplicações: medidas 3D, monitoramento de segurança e movimentação de objetos em trajetórias. As pesquisas atuais sobre calibração de câmeras consistem basicamente de um objeto de referência com forma geométrica simples ou padrão colocado numa certa posição e então estabelecer uma relação entre os parâmetros extrínsecos e as coordenadas dos objetos de calibração de acordo com os princípios de projeção e perspectiva.

A calibração das câmeras é necessária para a reconstrução tridimensional das coordenadas no espaço. Esta calibração é obtida pela medição manual dos pontos de referência na imagem, os pontos de referência e seus respectivos valores de coordenadas no espaço devem ser conhecidos (distância real com relação a algum ponto).

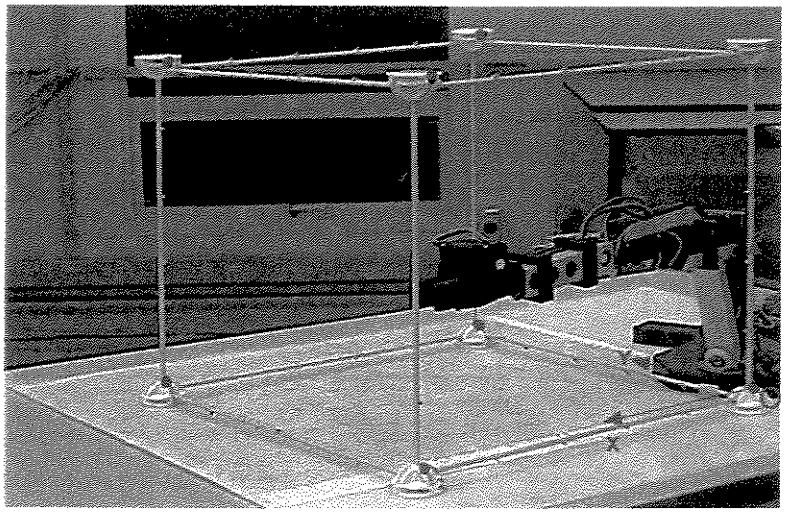


Figura 4.17 - Cubo de referência utilizado para calibração

Normalmente, para esta finalidade podem ser utilizados cubos de referência tal como mostra a Figura 4.17. A reconstrução tridimensional é considerada a partir das calibrações e medições de no mínimo duas câmeras [Barros,1997]

A principal diferença entre equipamentos para registro de movimentos e a utilização de câmeras de infravermelho ou de vídeo, é que nos sistemas onde câmeras de infravermelho são utilizadas, marcadores ativos (LEDs), emitem luz infravermelha que é registrada pelas câmeras. Uma variação deste tipo de sistema adota marcadores passivos que refletem a luz vinda de emissores infravermelho. Estes sistemas têm a vantagem de operar com maior resolução espacial e temporal nos registros, além do que a identificação automática (por *software*) dos marcadores é facilitada, uma vez que esses marcadores destacam-se fortemente dos demais objetos presentes nas imagens. Por esse motivo optou-se por utilizar marcadores passivos no manipulador robótico, para o caso em estudo só foi preciso utilizar um marcador.

Os equipamentos necessários para filmagem da trajetória do manipulador são: a) duas câmeras digitais de vídeo JVC DVL-9500, resolução de 720X480 e freqüência de 30 quadros/s b) um conversor analógico-digital para sinal de vídeo (*Capture Board*), capaz de adquirir o sinal de vídeo em tempo real (60 Fields/s), c) computador PC 486 ou superior, monitor capaz de operar com paletas de mais de 256 cores simultaneamente, mínimo de 8 Mbytes de memória RAM e sistema operacional Windows for 32bits (Windows 95 ou NT) e d) tripés para as câmeras de vídeo.

4.5.2. Reconstrução de Coordenadas

A fim de descrevermos o movimento de uma partícula é necessário que conheçamos sua posição no espaço, em relação a um dado referencial, em função do tempo. A posição da partícula no espaço, relativa ao referencial, é definida com a ajuda de três coordenadas independentes, por exemplo, as coordenadas cartesianas. Em relação ao parâmetro tempo (t), temos:

$$X=X(t), \quad Y=Y(t), \quad Z=Z(t) \tag{4.2}$$

A obtenção de coordenadas espaciais de pontos a partir do registro estereoscópico de suas projeções em imagens é uma metodologia bastante difundida na Biomecânica, denominada reconstrução tridimensional de coordenadas. Os procedimentos de calibração de câmeras e reconstrução tridimensional que utilizamos foram inicialmente propostos por Abdel-Aziz e Karara (1971) e são conhecidos como DLT (Direct Linear Transformation). As equações básicas do método de reconstrução tridimensional de coordenadas (DLT) são mostradas a seguir.

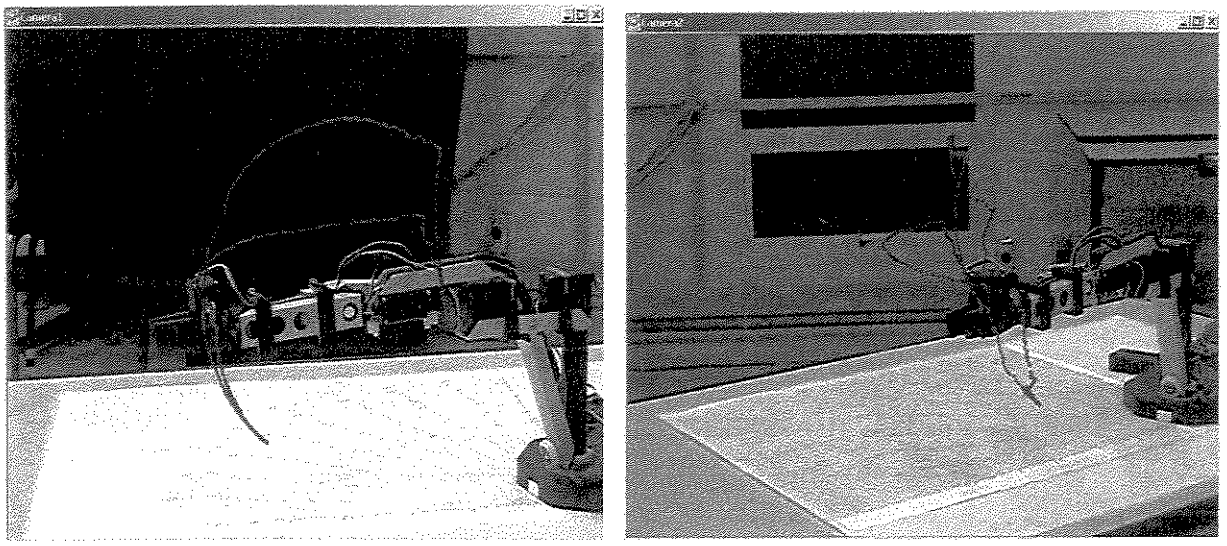
$$\begin{aligned}
\left(n_1^k - n_3^k x_i^k\right)X_i + \left(n_4^k - n_6^k x_i^k\right)Y_i + \left(n_7^k - n_9^k x_i^k\right)Z_i + n_{10}^k - x_i^k = 0 \\
\left(n_2^k - n_3^k x_i^k\right)X_i + \left(n_5^k - n_6^k x_i^k\right)Y_i + \left(n_8^k - n_9^k x_i^k\right)Z_i + n_{11}^k - y_i^k = 0
\end{aligned} \tag{4.3}$$

Este sistema de equações (4.3) é aplicado duas vezes, a primeira para quantificar os parâmetros da transformação (calibração) e a segunda para efetuar a reconstrução propriamente dita. Para a calibração das câmeras temos que: x_i e y_i são as coordenadas de tela do i-ésimo ponto de um sistema de referências conhecido, para cada câmera *k*, *Xi*, *Yi* e *Zi* são as coordenadas espaciais do i-ésimo ponto de referência e *nkh* (h=1,...,11) são os 11 parâmetros da transformação para a k-ésima câmera, a serem determinados. O número mínimo de pontos de referência com coordenadas conhecidas são seis.

Para a reconstrução tridimensional temos que: xk_i e yk_i são as coordenadas de tela do ponto de interesse na i-ésima imagem, da k-ésima câmera. *nkh* (h=1,...,11) são os parâmetros de calibração para a k-ésima câmera e X_i, Y_i e Z_i são as coordenadas espaciais do ponto de interesse na i-ésima imagem, a serem determinadas. O número mínimo de câmeras a serem usadas são duas.

Para a obtenção automática das coordenadas de tela dos pontos de interesse utilizou-se o sistema DVIDEOW [Figueroa,1998] que permite a perseguição automática do marcador que é utilizada na análise tridimensional do movimento do manipulador robótico. Esta análise consiste em determinar as coordenadas do marcador no espaço, utilizando técnicas de reconstrução, a partir de medições em imagens bidimensionais. As medições são obtidas aplicando o "tracking" automático para duas seqüências de imagens filmadas com duas câmeras estáticas, com diferentes ângulos de visão. Os parâmetros da construção são obtidos a partir da calibração das câmeras

empregando a DLT (*Direct Linear Transformation*) e alguns pontos de referência dos quais é possível obter a informação das coordenada reais e coordenadas na imagem. A Figura 4.18a e 4.18b correspondentes às câmeras1 e 2 mostram a trajetória, em dois ângulos diferentes, realizada pelo manipulador dentro do seu volume de trabalho.

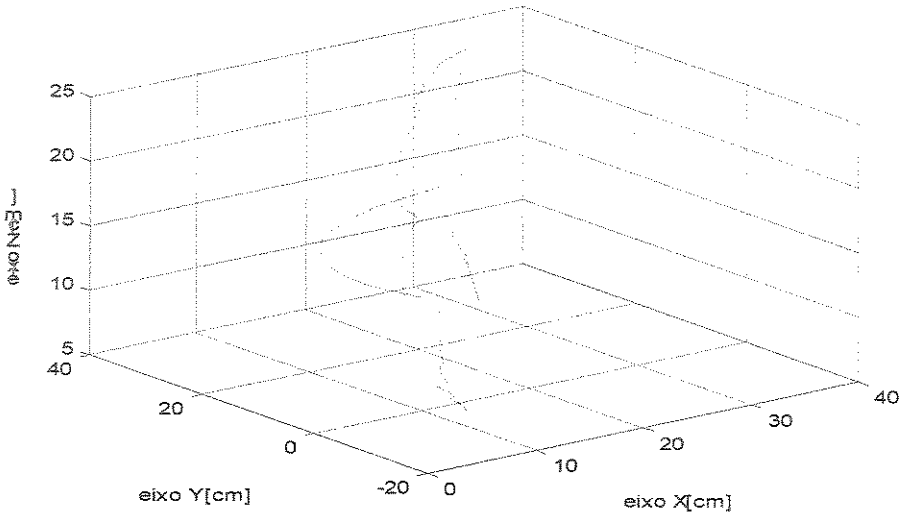


(a) (b)

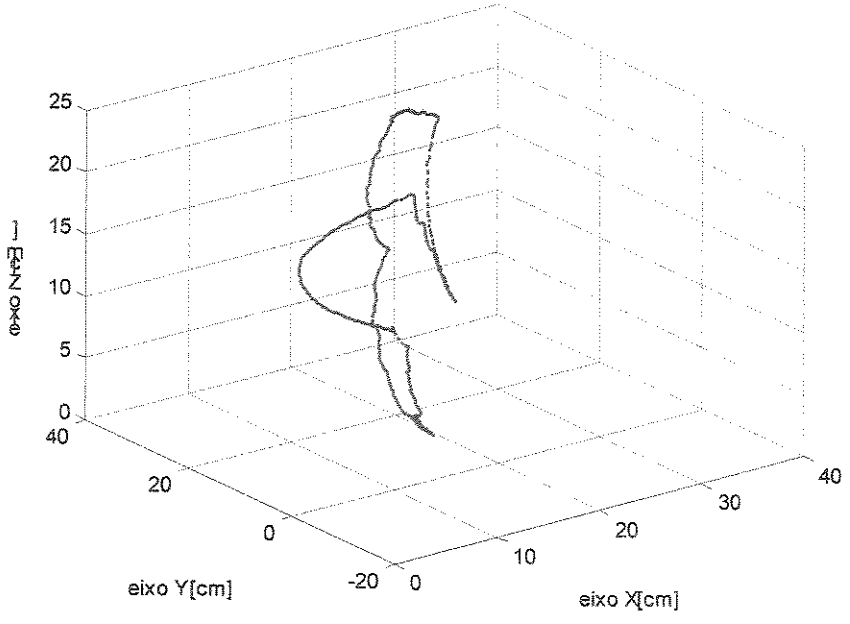
Figura 4.18 – Trajetória experimental do manipulador vista em dois ângulos diferentes

Os resultados obtidos a partir dos testes realizados, com diferentes seqüências, foram bastantes satisfatórios, tanto o algoritmo de mapeamento e controle de trajetória através de redes neurais artificiais como a verificação experimental da trajetória para o manipulador robótico, Robix™ RCS-6, não planar RR.

A seguir o gráfico 4.19 apresenta a comparação entre a trajetória simulada e experimental do manipulador. Note que as variações na trajetória experimental são devidas as folgas existentes no manipulador robótico.



(a)



(b)

Figura4.19 a) Trajetória simulada b) Trajetória experimental do manipulador robótico

Os gráficos 4.20 a 4.23 mostram as coordenadas em X, Y, e Z da trajetória simulada e experimental do manipulador, as trajetórias experimentais foram digitalizadas e medidas permitindo assim a reconstrução dos pontos em duas dimensões assim como o cálculo das distâncias respectivas com as coordenadas de referência.

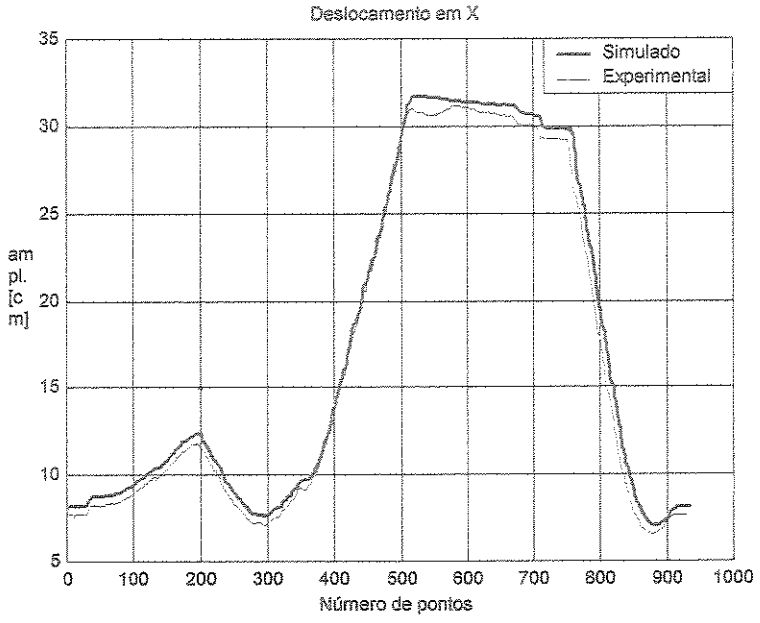


Figura 4.20 - Gráfico simulado e experimental do deslocamento no eixo X do manipulador

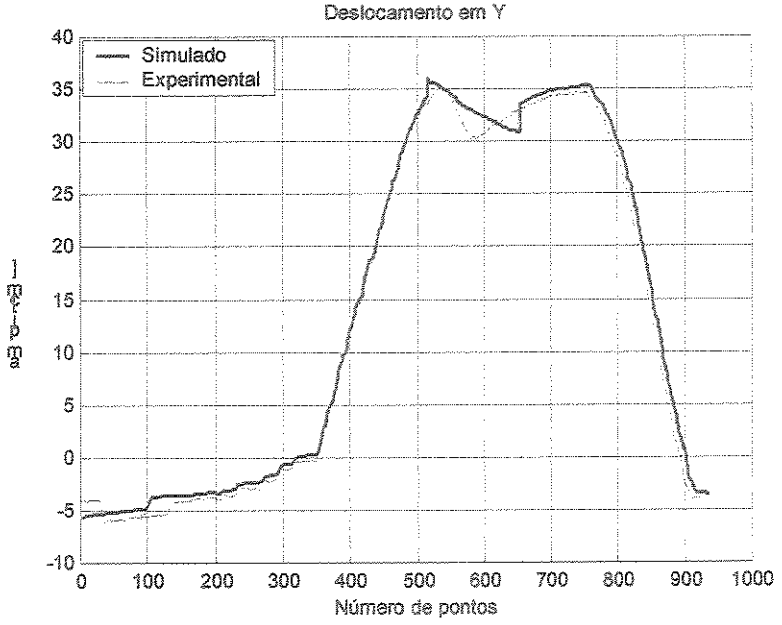


Figura 4.21 - Gráfico simulado e experimental do deslocamento no eixo Y do manipulador

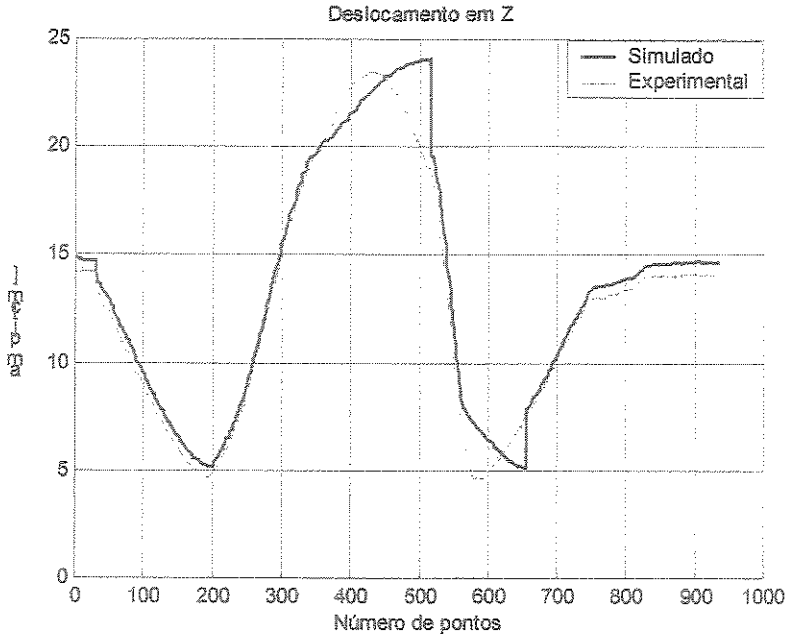


Figura 4.22 - Gráfico simulado e experimental do deslocamento no eixo Z do manipulador

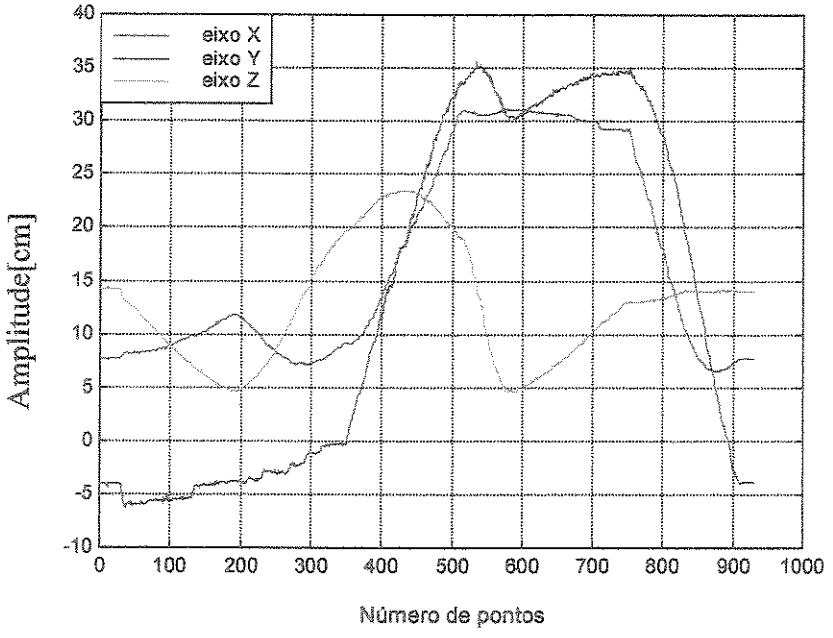


Figura 4.23 - Gráfico experimental das coordenadas de tela do manipulador robótico

Capítulo 5

Conclusões

Foi realizado satisfatoriamente o controle da trajetória através do uso de redes neurais artificiais do tipo *perceptron*, que foram treinadas com os pontos do volume de trabalho do manipulador robótico.

Os resultados das simulações e os resultados experimentais apresentados no capítulo 4 mostraram que a solução proposta para o problema de controle e geração de trajetórias é factível. A rede neural artificial mapeou o volume de trabalho e determinou a posição angular dos segmentos que são os responsáveis pela geração dos movimentos do manipulador robótico.

O algoritmo usado permitiu controlar a trajetória entre dois pontos qualquer dentro do volume de trabalho do manipulador robótico. Note-se que a trajetória experimental tem uma pequena diferença em relação a trajetória simulada cujo motivo é devido às folgas existentes no manipulador experimental.

A flexibilidade do programa permitiu que uma trajetória seja composta de vários segmentos. Isto tornou possível a obtenção de trajetórias com um certo grau de complexidade (cada segmento foi discretizado de uma forma diferente obtendo, desta forma, segmentos com

perfis diferentes). Pode-se simular trajetórias circulares e elípticas, restritas experimentalmente pela configuração do manipulador.

Os desenvolvimentos teóricos feitos no decorrer do texto com certeza fornecem os subsídios básicos necessários para a continuidade da pesquisa em sistemas por aprendizados e a RNA's, em robótica e em outras áreas. Foram tratados aspectos como modelagem, controle e geração de trajetórias.

5.1 Perspectivas Futuras

Como sugestões para trabalhos futuros podemos citar:

- Implementar o método mediante uma linguagem de programação mais rápida, mediante uma linguagem de programação pilada.
- Extensão da estrutura robótica o que significa aumentar o número de graus de liberdade do manipulador o que permitirá testar de forma mais ampla o algoritmo desenvolvido neste trabalho.
- Realizar o estudo de controle trajetórias num manipulador com desvio de obstáculos assim como a implementação de outras estratégias para o desvio de obstáculos utilizando redes neurais artificias.
- Aplicar a metodologia implementada num sistema robótico com dois manipuladores simultâneos, simulando dois braços mecânicos, assim como fazer uso de mecanismos de correção de erros para eliminar os erros provocados pelas folgas existentes na estrutura.

Bibliografia

Abdel-Aziz, Y. I., Karara, H. M.. *Direct linear transformation from comparator coordinates into object-space coordinates*. Proc. ASP/UI Symp. on Close-Range Photogrammetry. Urbana, Illinois, 1971.

Amaral, S., Pieri,E., Guenther, R., Controle a estrutura variável de roboôs manipuladores interagindo com ambientes cinemáticos. Revista Controle & Automação, v11 n.2, p.117-127, Mai., Jun., jul. e Agosto 2000.

Asada, H. e Slotine,J.J.E. *Robot Analysis and Control*. New York:John Wiley & Sons, Inc.,1986, 266p.

Barros, Ricardo M. Leite de. *Concepção e implementação de um sistema para análise cinemática de movimentos humanos*. Campinas: Faculdade de Educação Física, Universidade estadual de campinas, 1997. 180p. tese (PhD).

Benzatti, D. and Zampieri, D.(2001). Implementado em um microcontrolador real de um sistema de controle de robô bípede baseado em redes neurais. Relatório de bolsa de iniciação cientifica- Fapesp 99/10603-5.

Bugmann, G.;Koay, K. L.;Barlow, N.; Phillips, M.; Rodney, D. (1998) Stable encoding of robot trajectories using normalized radial basis Functions: Application to a Autonomous Wheelchair. *Proceeding of the 29th International Symposium On Robotics, Advanced robotics*: Beyond 2000, Girminghan, UK, p.27-30, April 1998.

Campos, R,. *Implementação de um mecanismo de Calibração e identificação de Parâmetros para robôs*. Campinas – SP: Faculdade de engenharia Mecânica , Universidade estadual de Campinas, 1993. 145p.Tese (Mestrado).

Cerqueira, Jés de Jesus. *Análise de Uma Classe de Neurônios Artificiais para Aplicações em Sistemas Robóticos*. Campinas – SP: Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas, 1996. 103p. Tese (Mestrado)

Craig, John J. *Introduction to Robotics: Mechanics and Control*. Addison-Wesley Publishing Company,Inc,1989, 450p.

Cruz, J. M. da *Projeto e Desenvolvimento de um Sistema de Geração Automática de Trajetória para Manipuladores*. Campinas-SP: Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas, 1993. 157p. Tese (Mestrado).

Ferreira, E. P. *Robotica Básica*. Versão preliminar Publicada para a V Escola Brasileiro-Argentina de Informática. Rio de janeiro, 1991.

Figueroa Rivero, P. J., *Perseguição de Marcadores Para Análise de Movimentos Humanos*. Campinas-SP: Instituto de Computação, Universidade Estadual de Campinas, 1998. 97p. Tese (Mestrado).

Fu, K S. et all. *Robotics, control, sensing, vision and intelligence*. Singapore: McGraw-Hill,1987, 580p

Goldenberg, A A., et al. A complete generalized Solution to the Inverse Kinematics of Robots. *IEEE journal of Robotics and Automation*, RA-1,n.1, p14-20, March 1985.

Halperin, D.; Kavraki, L.; Latombe, J.-C.. Robot algorithms. *CRC Handbook of Algorithms and Theory of Computation*, M. Atallah (ed.), Boca Raton, FL:CRC Press, 1998. Cap. 21.

Haykin, Simon. *Neural Networks A Comprehensive Foundation*. New Jersey: Prentice Hall, Second Edition, 1999, 900p

Hertz, J., Krogh, A.; Palmer, R.G. *Introduction to the theory of neural computation*: Redwood City, CA. Addison-Wesley Publishing Company, 1991

Juang, Jer-Nan. *Applied System Identification*. Reading: PTR Prentice-Hall, 1994.Cap.2: Time-Domain Models, p. 15-35.

Koivo, A, J. *Fundamentals for Control of Robotics Manipulators*. New York: John Wiley & sons, Inc., 1989, 450p

Kovács, Z. L., *Redes neurais Artificiais: Fundamentos e aplicações*. Edição Acadêmica, 1996

Lang, S., 1971, *Algebra Linear*, Editora Bücher Ltda.

Lee, C. S. G., and Ziegler, M. A Geometric Approach in Solving the Inverse Kinematics of Puma *Robots. IEEE Trans. Aerospace and Electronic Systems*, v.AES-20, n.6, p.795-706, June 1989.

MacLoone, S., Irwin, G. W. Fast parallel off-line training of multilayer perceptrons. *IEEE Transactions on Neural Networks*, v. 8, n. 3, p. 646-653, may 1997

Merian, James L. *Dinâmica*. Livros Técnicos e científicos S.A, 1994.

Minsky,M. L. and Papert, S. *Perceptrons*. MA: MIT Press,1988.

Microchip,2002. Microcontroladores pic. http://www.microchip.com

Naredra,K. S.and Levin U. Arsiel, Control of nonlinear Dynamical Systems neural networks-party: Observability, Identification, and control. *IEEE Transactions on Neural networks*, v.7, n.1,p.30-42, January, 1996

Narendra, K. S. and Parthasarathy, K. Identification and Control of dynamical Systems Using Neural Networks. *IEEE Transactions on Neural Networks*, v., n.1, p.4-27, March 1990

Nguyen, D. & Widrow, B. Neural Networks for self-Learning Control systems, *IEEE Control Magazine Systems*, 1990.

Park, Y., Choi, M., Lee, K. Y. An optimal tracking neuro-controller for nonlinear dynamic systems. *IEEE Transactions on Neural Networks*, v.7, n.5, p.1099-1110, September 1996

Paul, Richard P. *Robot manipulators: mathematics, programming, and control*. Massachusetts Institute of Technology: Cambridge, Massachusetts ,1981.

Pham, D. T., Oh, S. J. Identification of plant inverse dynamics using neural networks. *Artificial Intelligence in Engineering*, v. 13, p. 309-320, 1999

Pieper, D. L., The Kinematics of Manipulators Under Computer Control. Stanford University, 1968. Thesis (Ph. D.).

l

Ray, S.R.; Kargupta, H. *A temporal sequence processor based on the biological reaction-diffusion process. Complex Systems*, v.9, n.4. p.320-334, October 1996.

Sciavicco, Lorenzo e Siciliano, Bruno. *Modeling and control of robot manipulator*. McGraw-Hill Co., NY,458p.,1996

Santos, Ilmar, F. *Dinâmica dos Sistemas Mecânicos - Modelagem, Simulação, Visualização e Verificação*. Campinas: MAKRON BOOKS do Brasil Ltda, 1998, 256 p

Saramago, M P. *Projeto de Desenvolvimento de um Sistema de Calibração e Medida de Precisão de Robôs Industriais*. Campinas-SP: Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas, 1993. 114P. Tese (Mestrado)

Slotine,J.J. E Li, W. *Applied Nonlinear Control*. New Jersey: Prentice-Hall, 1991, 266p

Solbet (2002). Microcontroladores e Robótica. http://www.solbet.com.br

Spong, Mark W. & Vidyasagar, M. *Robot Dynamics and Control*. John Wiley & Sons, Inc, 1989.

Stone, Henrry W. *Kinematics Modeling Identification, and Control of Robotics manipulators*. Kluwer Academic Publishers, 1987.

Wang, D. L.; Arbib, M. A. Complex temporal sequence learning based on short-term memory. *Proceedings of IEEE*, v.78, p.1536-1543. 1990.

Wang, D. L.; Arbib, M. A. Timing and chunking in processing temporal order. *IEEE Transactions on Systems, Man, Cybernetics*, v.23, p.993-1009. 1993.

Wang, D. L.; Yuwono, B. Anticipation-based temporal pattern generation. *IEEE Transactions on Systems, Man, Cybernetics*, v.25, p.615-628. 1995.

W.McCulloch, and W.Pits. *A logical calculus of the ideas imminent in nervous activity*. Bulletin of Mathematical Biophysics, 5:115-133, 1943.

Anexo I

I.1 Linearização por Realimentação

A linearização por realimentação é um método que permite transformar a dinâmica de um sistema não-linear numa dinâmica linear, através de uma realimentação linear da saída. Para atingir este objetivo torna-se quase sempre necessário efetuar uma mudança de variável de estado e introduzir uma variável de entrada auxiliar. Depois de ter o sistema não-linear modificado para que se comporte como linear é possível utilizar técnicas lineares conhecidas. Para se efetuar o controle do sistema original, como a de realimentação por exemplo. O método para poder ser aplicado exige o conhecimento do modelo descritivo do sistema não-linear inicial. A idéia fundamental da linearização por realimentação de estado é construir uma lei de controle que linearize o sistema não linear após uma mudança de variáveis de estado. Este método diferencia-se dos métodos tradicionais, como a linearização Jacobiana (Slotine,1991) em que a linearização é *local*. Isto é válida apenas para uma região em torno de um determinado ponto, enquanto a linearização por realimentação *global* aplica-se a todo o domínio do espaço de estados com a eventual excepção de pontos isolados. Além disso a linearização pelo jacobiano é *aproximada* e a linearização por realimentação é *exata*.

Determinados tipos de sistemas são susceptíveis de serem linearizados exatamente por realimentação: é o caso em que as variáveis de entrada e de estado são separáveis no sistema de equação de estado, ou seja sistemas representáveis pelo modelo dado pela equação a seguir

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{f}(x) + \mathbf{g}(\mathbf{x})u \\ y &= \mathbf{h}(\mathbf{x}) \end{aligned} \tag{I.1}$$

Para os casos em que é possível efetuar a linearização exata, há normalmente que efetuar uma mudança de variável de estado

$$\mathbf{z} = \mathbf{z}(\mathbf{x}) \tag{I.2}$$

e introduzir uma lei de controle

$$u=u(\mathbf{x},v) \tag{I.3}$$

Ou seja uma mudança de variável de controle em que v é uma nova variável intermediária que conduz o sistema original a um sistema linear da mesma forma que a equação (I.1).

A linearização por realimentação tem sido utilizada com êxito numa grande gama de aplicações como: controle de robôs , manipuladores, peças de artilharia, helicópteros, aviões e satélites, além de ser usada em aparelhagem médica e nas indústrias química e farmacêutica.

I.1.1 Linearização Entrada - Estado

Na linearização entrada-estado objetivo é linearizar por realimentação um sistema descrito pela equação de estado

$$\dot{\mathbf{x}} = f(\mathbf{x},u) \tag{I.4}$$

em que o estado é completamente acessível. A linearização é feita com uma mudança da variável de estado $\mathbf{z} = \mathbf{z}(\mathbf{x})$ e com uma mudança na variável de entrada $u=u(\mathbf{x},v)$ de modo a se obter um sistema linear e invariante no tempo, com a seguinte equação

$$\dot{\mathbf{z}} = \mathbf{A}\mathbf{z} + \mathbf{b}v \tag{I.5}$$

da equação anterior, o sistema linear pode ser controlado com qualquer das técnicas usadas em sistemas lineares.

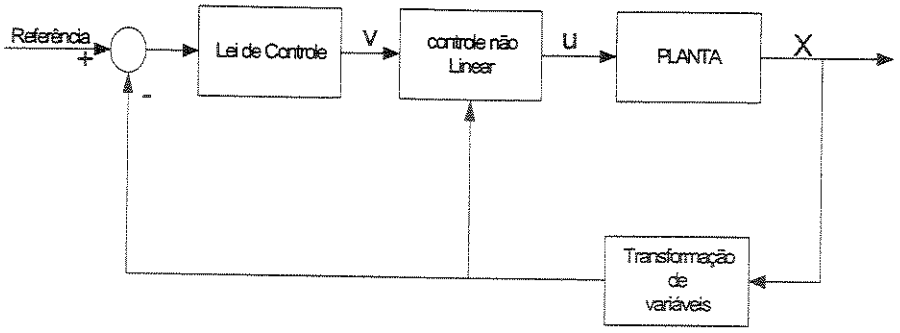


Figura I.1 - Linearização de entrada-estado

I.2 Modelo dinâmico do manipulador R-P

O manipulador plano de 2 graus de liberdade da Figura I.2, é composto de uma junta de rotação R, situada no corpo 1 e uma junta prismática ou de translação P, situada no corpo 2. Os parâmetros do manipulador são: a massa m_1 do corpo 1, a massa m_2 do corpo 2, onde ambas são diferentes, g é a aceleração da gravidade, l a posição do corpo 1, θ o ângulo do corpo 1, $\dot{\theta}$ a velocidade angular do corpo 1, d o deslocamento absoluto do corpo 2, $\dot{d}$ a velocidade linear do corpo 2, I_{zz1} a inércia do corpo 1, I_{zz2} a inércia do corpo 2, τ_1 o torque e τ_2 a força de translação.

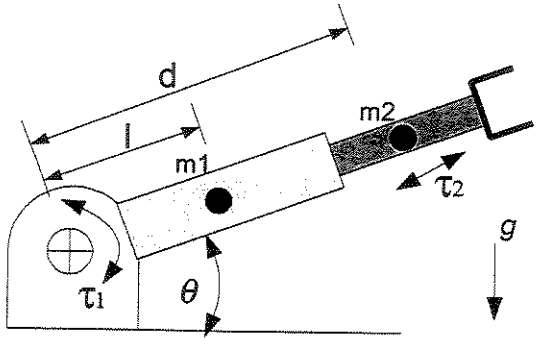


Figura -I.2. – Representação esquemática do sistema de 2 GDL

O manipulador robótico contém dois motores de acionamento sendo: um motor diretamente acoplado ao corpo 1, representando o torque τ_1 e uma rotação θ. Um motor fixo ao corpo 2 e acoplado ao corpo 1 através de uma rosca sem fim, que proporciona um movimento prismático do corpo 2 em relação ao corpo 1, fazendo o deslocamento d do corpo 2 variável. Possuindo estes, diferentes massas representadas por m_1 e m_2 respectivamente. A posição l permanece constante entre o centro de massa do corpo 1 e o eixo de rotação deste corpo.

I.2.1 Formulação de *Lagrange*

Determina o conjunto de equações dinâmicas que descreve o comportamento do sistema mecânico. Estas são as equações de movimento de *Lagrange*, que origina-se da diferença da energia cinética e potencial, conforme exposto por Craig(1989) e Spong(1989).

Energia cinética *(K)*

De forma similar ao equacionamento desenvolvido no capítulo 3 vem:

$$K_1 = \frac{1}{2} I \dot{\theta}^2$$

$$K_2 = \frac{1}{2} m \dot{d}^2$$

$$K_1 = \frac{1}{2} m_1 l^2 \dot{\theta}^2 + \frac{1}{2} I_{zz1} \dot{\theta}^2$$

$$K_2 = \frac{1}{2} m_2 \left(d^2 \dot{\theta}^2 + \dot{d}^2 \right) + \frac{1}{2} I_{zz2} \dot{\theta}^2$$

Energia cinética total $K(\theta, \dot{\theta})$

$$K(\theta,\dot{\theta}) = \frac{1}{2}(m_1 l^2 + I_{zz1} + I_{zz2} + m_2 d^2)\dot{\theta}^2 + \frac{1}{2} m_2 \dot{d}^2 \qquad \text{(I.6)}$$

Energia potencial *(U)*

$$U_{elástica} = \frac{1}{2}Kx^2$$

$$U_{gravitacional} = m.g.\Delta h$$

$$U_1 = m_1 gl \operatorname{sen}\theta + m_1 gl$$

$$U_2 = m_2 gd \operatorname{sen}\theta + m_2 gd_{máx}$$

Energia potencial total $U(\theta)$

$$U(\theta) = g(m_1 l + m_2 d)\operatorname{sen}\theta + m_1 gl + m_2 gd_{\max} \tag{I.7}$$

Equação dinâmica de Lagrange *(L)*

$$L = K - U$$

$$L = \frac{1}{2}(m_1 l^2 + I_{zz1} + I_{zz2} + m_2 d^2)\dot{\theta}^2 + \frac{1}{2}m_2 \dot{d}^2 - g(m_1 l + m_2 d)\operatorname{sen}\theta + m_1 gl + m_2 gd_{\max} \tag{I.8}$$

Derivadas parciais e equações de movimento (*Euler-Lagrange*):

Corpo 1

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\theta}} - \frac{\partial L}{\partial \theta} = \tau_1$$

$$\ddot{\theta}\left(m_1 l^2 + I_{zz1} + I_{zz2} + m_2 d^2\right) + 2m_2 d\dot{\theta}\,\dot{d} + (m_1 l + m_2 d)g\cos\theta = \tau_1 \tag{I.9}$$

Corpo 2

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{d}} - \frac{\partial L}{\partial d} = \tau_2$$

$$m_2\ddot{d} - m_2 d\dot{\theta}^2 + m_2 g \operatorname{sen}\theta = \tau_2 \tag{I.10}$$

De acordo com as equações de movimento do sistema não linear:

$$\ddot{\theta}\left(m_1 l^2 + I_{zz1} + I_{zz2} + m_2 d^2\right) + 2m_2 d\dot{\theta}\,\dot{d} + \left(m_1 l + m_2 d\right) g \cos\theta = \tau_1$$

$$m_2\ddot{d} - m_2 d\dot{\theta}^2 + m_2 g \operatorname{sen}\theta = \tau_2 \tag{I.11}$$

Das equações acima, pode-se escrever na forma matricial a seguir

$$M(\theta) = \begin{bmatrix} M_1 & 0 \\ 0 & M_2 \end{bmatrix} = \begin{bmatrix} m_1 l^2 + I_{zz1} + I_{zz2} + m_2 d & 0 \\ 0 & m_2 \end{bmatrix} \tag{I.12}$$

$$h(\theta,\dot{\theta}) = \begin{Bmatrix} h_1 \\ h_2 \end{Bmatrix} = \begin{Bmatrix} 2m_2 d\dot{\theta}\dot{d} \\ -m_2 d\dot{\theta}^2 \end{Bmatrix} \tag{I.13}$$

$$G(\theta) = \begin{Bmatrix} G_1 \\ G_2 \end{Bmatrix} = \begin{Bmatrix} (m\ l + m_2 d) g \cos\theta \\ m_2 g \operatorname{sen}\theta \end{Bmatrix} \tag{I.14}$$

I.3 Linearização do modelo dinâmico

A partir das equações do modelo dinâmico do manipulador R-P. equação (I.11), pode-se obter as equações de estado adotando a técnica da linearização por realimentação para sistemas de múltiplas entradas. Para um manipulador de n-segmentos seriais tem-se as seguintes variáveis de estado.

$$\dot{\xi}_1 = \xi_2$$

$$\dot{\xi}_2 = -M(\xi)^{-1} h(\xi_1, \xi_2) + M(\xi_1)^{-1} u \tag{I.15}$$

com uma lei de controle:

$$u = M(\xi_1) v + h(\xi_1, \xi_2) \tag{I.16}$$

para o manipulador da Figura I.1 tem-se que

$$\xi_1 = \begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix} = \begin{Bmatrix} d \\ \theta \end{Bmatrix} \tag{I.17}$$

$$\xi_2 = \dot{\xi}_1 = \begin{Bmatrix} x_3 \\ x_4 \end{Bmatrix} = \begin{Bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{Bmatrix} = \begin{Bmatrix} \dot{d} \\ \dot{\theta} \end{Bmatrix} \tag{I.18}$$

$$\xi_3 = \dot{\xi}_2 = \begin{Bmatrix} \dot{x}_3 \\ \dot{x}_4 \end{Bmatrix} = \begin{Bmatrix} \dfrac{\tau_1 - h_1 - G}{M_1} \\ \dfrac{\tau_2 + h_2 - G_2}{M_2} \end{Bmatrix} \tag{I.19}$$

Então a Lei de controle será da forma:

$$u = M(\xi_1) V + h(\xi_1, \xi_2) = \begin{Bmatrix} \tau_1 \\ \tau_2 \end{Bmatrix} \tag{I.20}$$

substituindo-se as matrizes M e h na equação(I.19), temos

$$\ddot{\xi}_2 = \begin{bmatrix} \frac{1}{b + m_2 x_1^2} & 0 \\ 0 & \frac{1}{m_2} \end{bmatrix} \begin{Bmatrix} 2m_2 x_1 x_3 x_4 + (a + m_2 x_1) g \cos x_2 \\ (g \text{ sen } x_2 - x_1 x_4^2) m_2 \end{Bmatrix} + \begin{bmatrix} \frac{1}{b + m_2 x_1^2} & 0 \\ 0 & \frac{1}{m_2} \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \end{Bmatrix} \qquad \text{(I.21)}$$

onde : a= $m_1 l$

b= $m_1 l^2 + I_{zz1} + I_{zz2}$

comparando-se a equação (I.21) com a equação (I.16), obtêm-se a expressão para a lei de controle não linear

$$u = \begin{bmatrix} b + m_2 x_1^2 & 0 \\ 0 & m_2 \end{bmatrix} \begin{Bmatrix} v_1 \\ v_2 \end{Bmatrix} + \begin{Bmatrix} 2m_2 x_1 x_3 x_4 + (a + m_2 x_1) g \cos x_2 \\ (g \text{ sen } x_2 - x_2 x_4^2) m_2 \end{Bmatrix} \qquad \text{(I.22)}$$

utilizando-se o sistema de coordenadas ξ, pode-se escrever na forma matricial, o modelo do sistema não-linear para o manipulador R-P:

$$\dot{\xi} = f(x) + g(x)u \qquad \text{(I.23)}$$

$$\dot{\xi} = \begin{Bmatrix} \dot{\xi}_1 \\ \dot{\xi}_2 \end{Bmatrix} = \begin{Bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{Bmatrix} = \begin{Bmatrix} x_3 \\ x_4 \\ \frac{-2m_2 x_1 x_3 x_4 - (a + m_2 x_1) g \cos x_2}{b + m_2 x_1^2} \\ x_1 x_4^2 - g \text{ sen } x_2 \end{Bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{1}{b + m_2 x_1^2} & 0 \\ 0 & \frac{1}{m_2} \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \end{Bmatrix} \qquad \text{(I.24)}$$

com

$$f(x)=\begin{Bmatrix} x_3 \\ \dfrac{-2m_2x_2x_3x_4-(a+m_2x_3)g\cos x_1}{b+m_2x_3^2} \\ x_4 \\ x_3x_2^2-g\,\mathrm{sen}\,x_1 \end{Bmatrix} \tag{I.25}$$

$$g(x)=\begin{bmatrix} 0 & 0 \\ \dfrac{1}{b+m_2x_3^2} & 0 \\ 0 & 0 \\ 0 & \dfrac{1}{m_2} \end{bmatrix} \tag{I.26}$$

I.4 Controle do modelo linearizado

Realizando-se a transformação das variáveis não lineares obtemos o controle do modelo linearizado.

$$\psi_1 = T_1(\xi_1)=\xi_1=\begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix}=\begin{Bmatrix} y_1 \\ y_2 \end{Bmatrix} \tag{I.17}$$

$$\psi_2 = T_2(\xi_2)=\dot{\psi}_1=\dot{\xi}_1=\xi_2=\begin{Bmatrix} x_3 \\ x_4 \end{Bmatrix}=\begin{Bmatrix} y_3 \\ y_4 \end{Bmatrix} \tag{I.28}$$

Nas coordenadas ψ e com a lei de controle , o sistema torna-se:

$$\dot{\psi}_1=\dot{\xi}_1=\xi_2=\psi_2=\begin{Bmatrix} x_3 \\ x_4 \end{Bmatrix}=\begin{Bmatrix} y_3 \\ y_4 \end{Bmatrix} \tag{I.29}$$

$$\psi_2 = \dot{\xi}_2 = V = \begin{Bmatrix} v_1 \\ v_2 \end{Bmatrix} \tag{I.30}$$

e a nova lei de controle linearizada é obtida a partir da condição da equação(I.30), pode-se escrever as equações do erro da seguinte forma:

$$\begin{Bmatrix} erro_1 \\ erro_2 \end{Bmatrix} = \begin{Bmatrix} \ddot{e}_1 + k_2\dot{e}_1 + k_1 e_1 \\ \ddot{e}_2 + k_4\dot{e}_2 + k_3 e_2 \end{Bmatrix} \tag{I.31}$$

adotando os pólos da equação do erro no semi-plano esquerdo para obter um sistema estável; na posição -10, para garantir uma boa convergência do erro, tem-se:

$$(s+10)^2 = \ddot{e}_1 + k_2\dot{e}_1 + k_1 e_1 = s^2 e_1 k_2 s e_1 + k_1 e_1 = e_1\left(s^2 + k_2 s + k_1\right) \tag{I.32}$$

$$(s+10)^2 = \ddot{e}_2 + k_4\dot{e}_2 + k_3 e_2 = s^2 e_2 + k_4 s e_2 + k_3 e_2 = e_2\left(s^2 + k_4 s + k_3\right) \tag{I.33}$$

$$s^2 + 20s + 100s = s^2 + k_2 s + k_1 \tag{I.34}$$

$$s^2 + 20s + 100s = s^2 + k_4 s + k_3 \tag{I.35}$$

onde por comparação obtemos:

$k_1 = 100$ $\qquad$ $k_2 = 20$

$k_3 = 100$ $\qquad$ $k_4 = 20$

Assim, obtém-se a nova lei de controle V:

$$V = \begin{Bmatrix} V_1 \\ V_2 \end{Bmatrix} = \begin{Bmatrix} \dot{y}_3^d - k_1\left(y_1 - y_1^d\right) - k_2\left(y_3 - y_3^d\right) \\ \dot{y}_4^d - k_3\left(y_2 - y_2^d\right) - k_4\left(y_4 - y_4^d\right) \end{Bmatrix} \tag{I.36}$$

I.5 Identificação e Linearização do Sistema por Redes Neurais

I.5.1 Identificação

Para a identificação do sistema não linear estão sendo utilizadas redes neurais dinâmicas[Narendra,1996] de múltiplas camadas, neste caso está sendo aplicada duas camadas, uma intermediária com seis neurônios e uma última camada composta de quatro saídas. O estado inicial é obtido através de varias combinações dos dados de entrada de deslocamento, ângulo, velocidade, força e torque dos motores. No treinamento da rede é utilizado o algoritmo Levenberg-Marquardt buscando-se uma somatória do erro quadrático na saída desejada de 0.005 em amplitude.

Os gráficos abaixo representam a identificação do sistema por redes neurais do movimento do manipulador representado na Figura(I.2) com a aplicação de uma força de translação e um torque

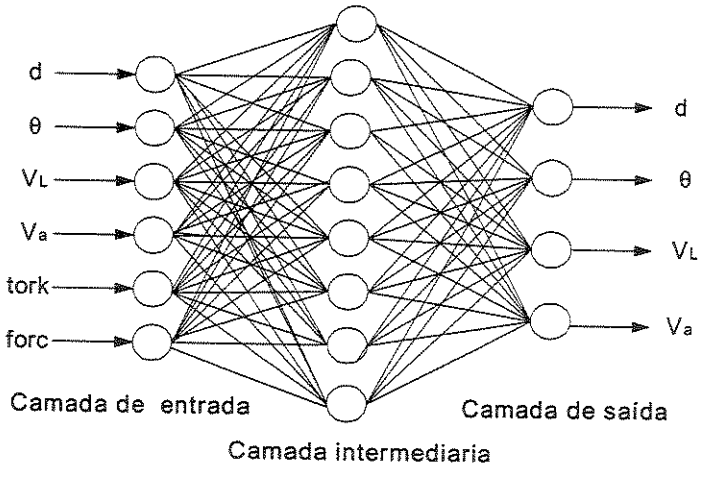


Figura I.3 - Rede neural de múltiplas camadas para identificação

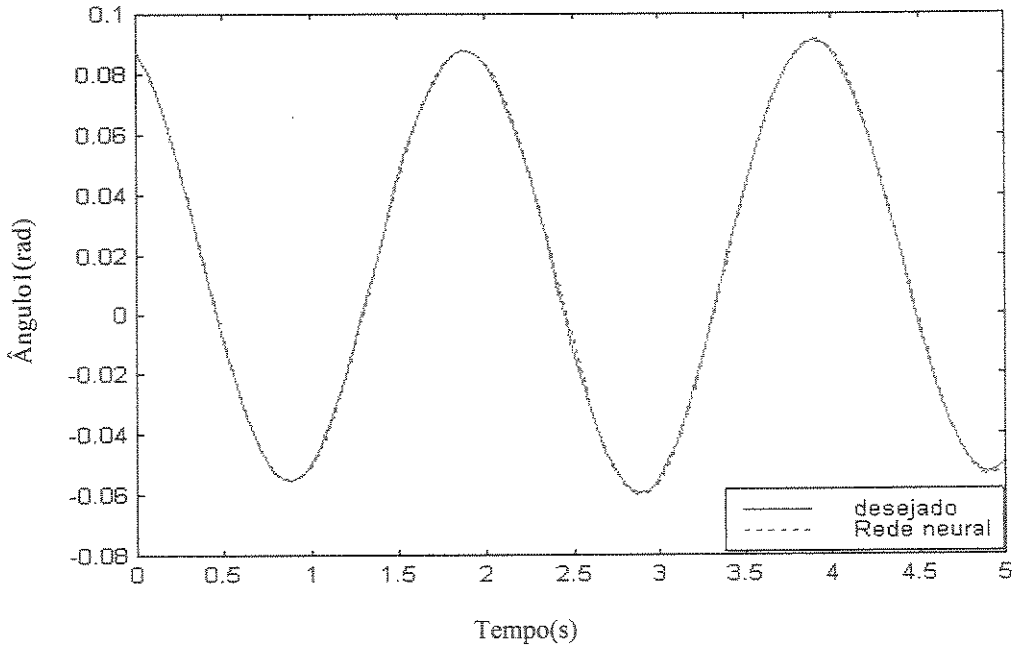
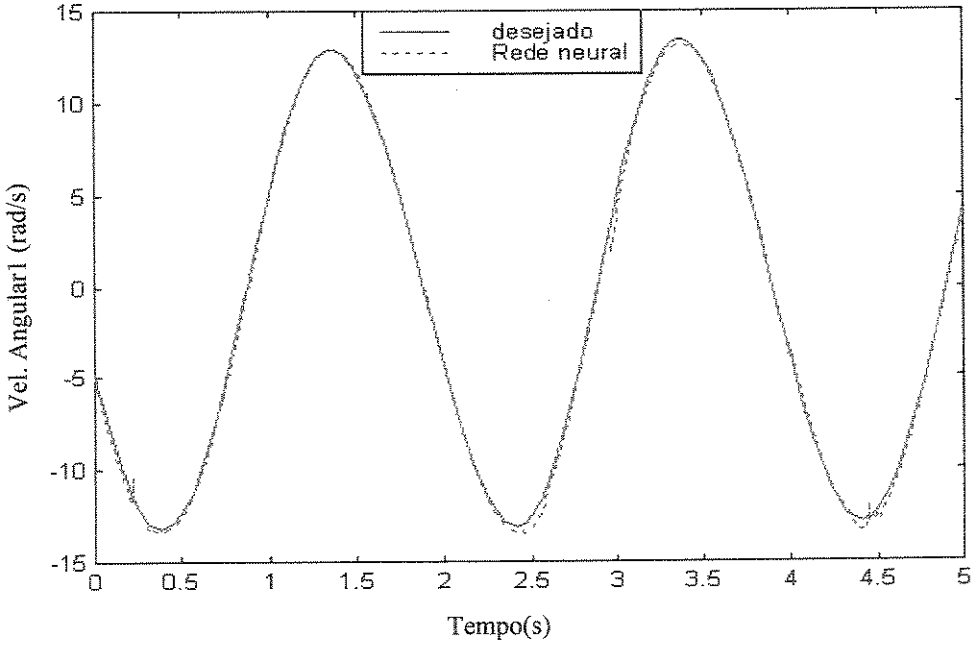


Figura I.4 Ângulo do corpo1



Figural.5- Velocidade angular do corpo1

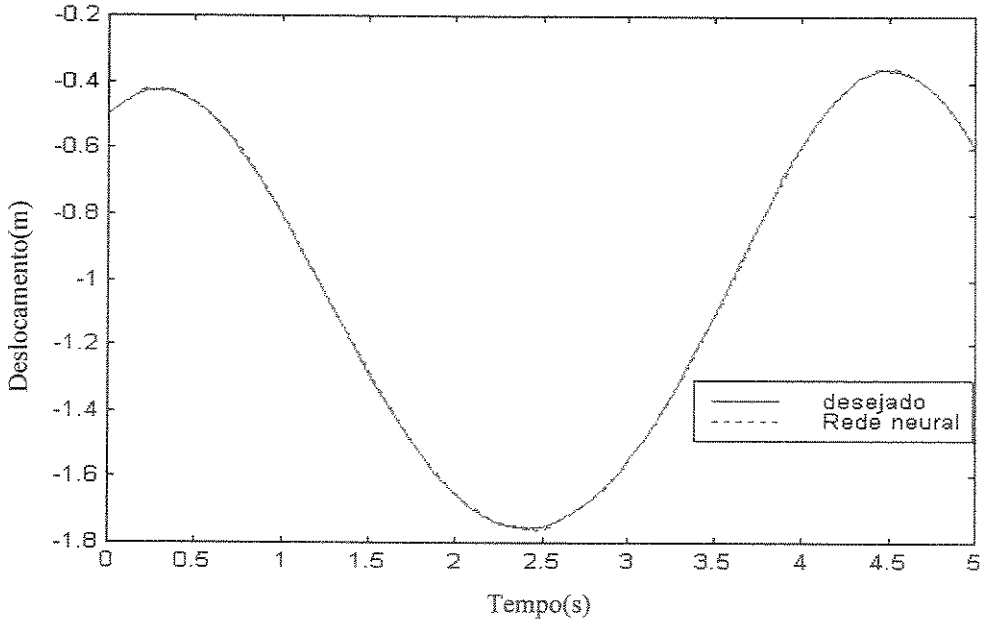


Figura I.6 – Deslocamento do corpo 2

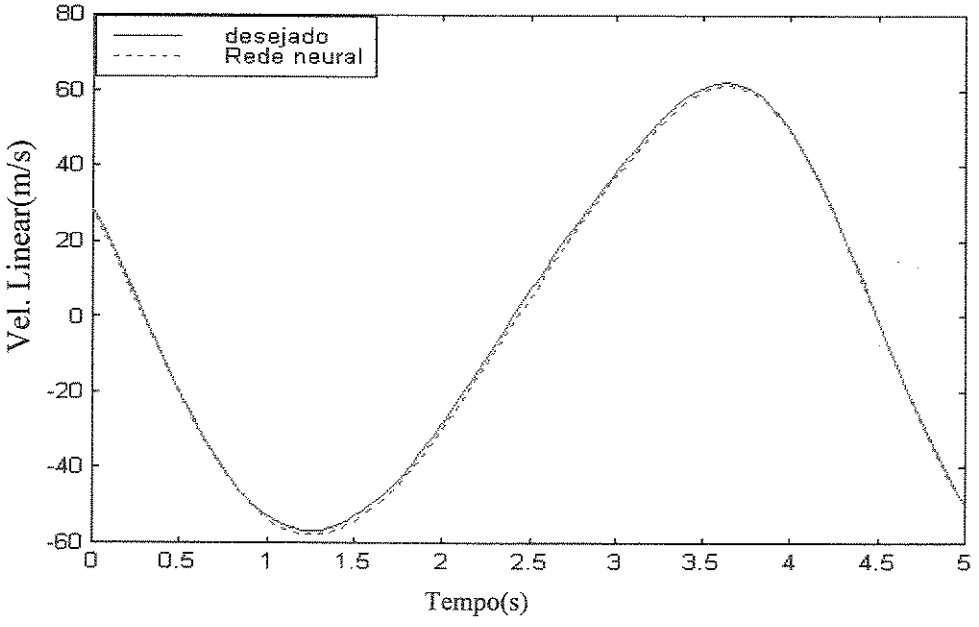


Figura I.7 Velocidade linear do corpo 2

Para gerar os gráficos anteriores foram utilizados os seguintes valores:

- $m_1 = 1$ Kg (massa do corpo1)
- $m_1 = 1$ Kg (massa do corpo2)
- $g = 9.8$ m/s^2 (aceleração da gravedade)
- $I_{zz1} = 1$ Kg m^2 (momento de inércia do corpo1)
- $I_{zz2} = 1$ Kg m^2 (momento de inércia do corpo2)
- $\tau_1 = 0.2$ Nm (Torque aplicado ao corpo1)
- $\tau_2 = 0.2$ N (Força aplicada ao corpo2)

I.5.2 Linearização

Na linearização do sistema utilizou-se uma rede neural do tipo MLP, composta por uma camada de entrada com seis neurônios, sendo quatro de realimentação do estado x e uma para cada sinal da lei de controle v_1 e v_2, e duas saídas tal como mostra a Figura I.8.

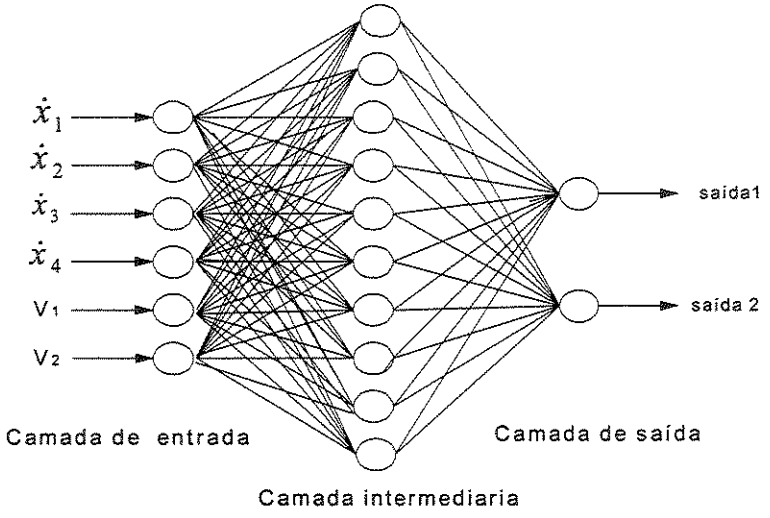


Figura I.8 Rede neural do tipo MLP de linearização

Foi feita a análise referente ao desempenho do sistema robótico através de gráficos correspondentes à linearização do sistema por redes neurais conforme a Figura (I.9) a qual representa o diagrama de blocos do sistema.

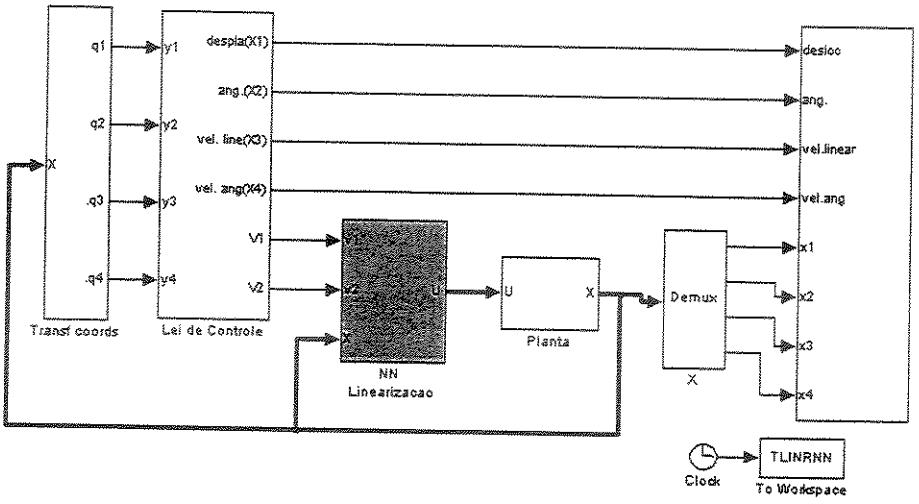


Figura I.9 – Diagrama de blocos do sistema com uma rede neural de linearização

O sinal de entrada da rede é formado pelos sinais de referência V_1 , V_2 e pela realimentação de seu próprio sinal de saída, tendo-se em vista que, quanto mais informações para a rede, melhor deveria ser seu desempenho. Uma pequena diferença apresentada nesta montagem, é a realimentação feita com o sinal se saída da planta, mostrando-se com isto a não linearidade dos subsistemas pela sua relação de entrada-saída, ou seja, rede e planta não lineares onde se houvesse um outro bloco linear e está relação não seria possível.

A seguir os seguintes gráficos I.10 e I.11 mostram os resultados da rede neural a qual acompanha um sinal de onda senoidal, onde representam o deslocamento e a velocidade angular do corpo1 no sistema robótico.

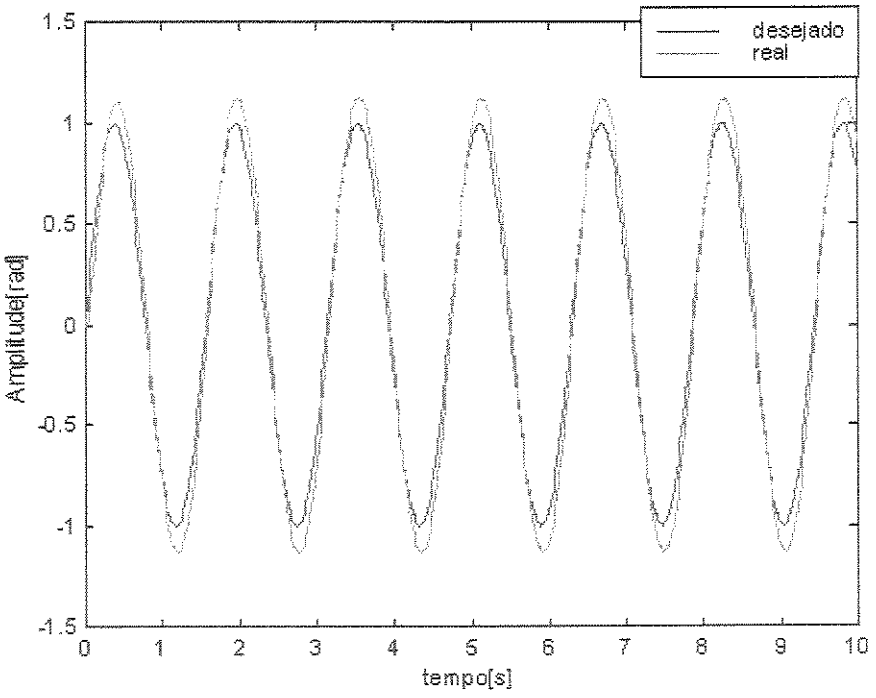


Figura I.10- Gráfico do deslocamento angular do corpo1

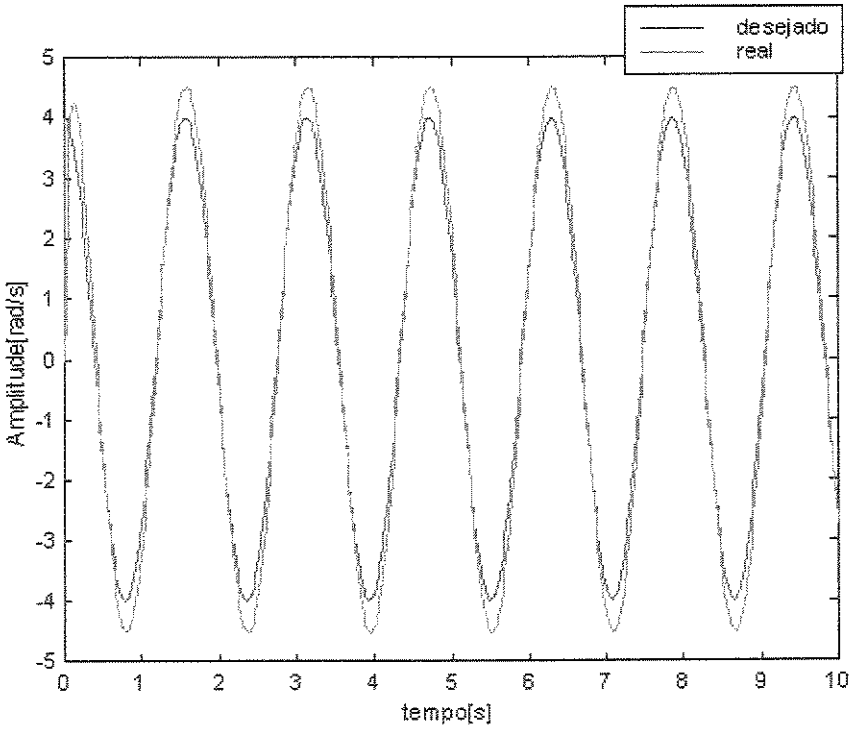


Figura I.11- Gráfico da velocidade angular do corpo1

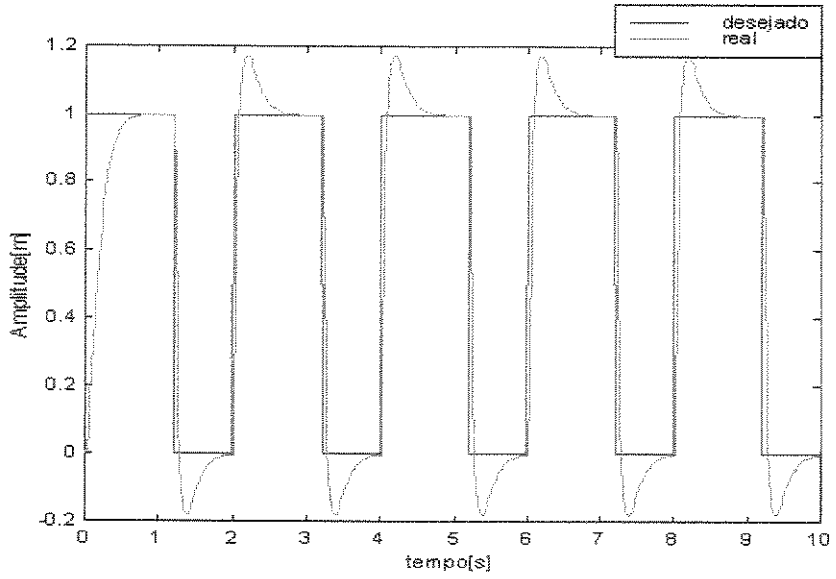


Figura I.12- Gráfico do deslocamento linear do corpo2

O gráfico da Figura I.12 torna –se crítico na análise de aproximação da rede ao sinal desejado, devido ao sobressinal acentuado em relação ao sinal real. No entanto, a rede mostra um bom desempenho na sua função de acompanhamento da trajetória desejada.

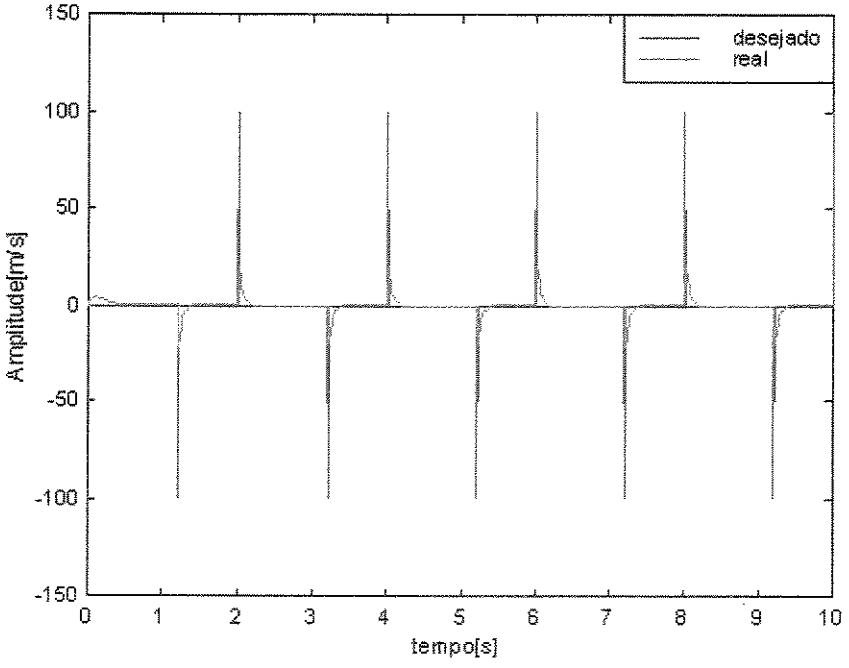


Figura I.13- Gráfico da velocidade linear do corpo2