Universidade Estadual de Campinas
Faculdade de Engenharia Elétrica
Departamento de Sistemas e Controle de Energia
Laboratório de Sistemas Modulares Robóticos

Implementação de Controladores Neurais de Kim–Lewis–Dawson com Parâmetros Otimizados por Algoritmos Genéticos

Mario Jungbeck

Dissertação submetida à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas, como parte dos requisitos exigidos para obtenção do título de Mestre em Engenharia Elétrica

Banca Examinadora:
Prof. Dr. Marconi Kolm Madrid (Orientador) - DSCE/FEEC/UNICAMP
Prof. Dr. Márcio Luiz de Andrade Netto - DCA/FEEC/UNICAMP
Prof. Dr. Fernando José Von Zuben - DCA/FEEC/UNICAMP
Prof. Dr. Felipe Martins Müller - DELC/CT/UFSM

Campinas, agosto de 2001

Para minha esposa Tatiane,
aos meus pais Bruno e Lori
e às minhas irmãs Mirian e Marcia.
Agradeço o amor a mim dedicado.

Agradecimentos

Agradeço a todas as pessoas que de alguma forma contribuíram para a realização deste trabalho. Em especial, agradeço:

- Ao professor Madrid pela orientação, amizade, companheirismo e pela crença na minha capacidade intelectual.
- Ao meu orientador do período de graduação, Prof. Ricardo Nederson do Prado, que me deu a oportunidade de fazer parte da equipe do GEDRE-NUPEDE-UFSM, sem dúvida o melhor laboratório de estudo e desenvolvimento de reatores eletrônicos do país. Agradeço ao Prof. Ricardo pela minha iniciação no maravilhoso mundo da ciência.
- Ao amigo e Prof. Felipe Martins Müller ("*Felipão*"), que teve papel fundamental no meu ingresso à UNICAMP.
- Ao meu amigo Marcio de Oliveira Buss, por todos os momentos ébrios vividos em Campinas.
- Aos amigos Fabrício e Bonani pelas proveitosas discussões nas áreas de controle, visão computacional, mpb e assuntos afins.
- A todos os amigos da comunidade dos Pampas, em particular: Jeferson, Emilene, Luciana, Marcelo, McPinto, Vinícius e Gustavo.
- Ao Convênio Capes pelo apoio financeiro.

Resumo

Esta dissertação propõe uma técnica que emprega algoritmos genéticos e teoria de redes neurais integrados, visando o desenvolvimento automático de controladores de alta performance para servomecanismos tipo elo-acionado, ou módulo de junta robótica.

Nesta abordagem, a teoria de redes neurais é utilizada no desenvolvimento de um controlador e um observador de estados não lineares. Devido às características apresentadas, estes controladores possuem potencial para resolver uma enorme variedade de problemas, inclusive problemas nos quais os métodos convencionais não são aplicáveis.

Os algoritmos genéticos são métodos de busca inspirados no processo evolutivo natural que apresentam-se como uma alternativa eficiente para o ajuste automático de controladores não lineares. O algoritmo genético proposto neste trabalho é utilizado para o ajuste paramétrico de controladores neurais.

Os resultados experimentais mostraram que tal técnica é muito eficiente para o controle de juntas robóticas e para uma boa parte de outros sistemas de engenharia que possuam dinâmica semelhante, podendo-se realizar sua aplicação prática com êxito, conseguindo-se uma excelente relação de custo/benefício.

Abstract

This thesis proposes a technique that integrated genetic algorithms and neural networks theory, seeking for the automatic development of high performance controller for servomechanisms like driven-links, or robotic joints modules.

The neural networks theory is used to develop nonlinear controllers. Due to the presented characteristics, these controllers possess potential to solve an enormous variety of problems, including those which the conventional methods are not suitable.

Genetic algorithms are search methods inspired by natural evolutionary process that come as an efficient alternative for automatic tuning of nonlinear controllers. The genetic algorithm proposed here is used for the parametric adjustment of neural net controllers.

The experimental results showed that such technique is very efficient for the control of robotic joints and for most of other engineering system that possesses similar dynamics. It can be assured its practical application with success and an excellent cost/benefit relation.

Sumário

Lista de Figuras

Capítulo 1

Motivação, Objetivos e Organização

1.1 Introdução

A crescente utilização da automação industrial tem provocado um elevado emprego de robôs nas mais diversas áreas. Juntamente com esta difusão, cresce a necessidade de controlar estas máquinas que, a cada dia, são mais exigidas no seu ambiente de trabalho. Os robôs atualmente em uso estão muito próximos de um manipulador, que consiste de um braço mecânico acionado por sistemas elétricos, hidráulicos ou pneumáticos, podendo possuir até seis graus de liberdade.

O controle de um manipulador pode ser feito de forma automática ou por um controlador humano. Quando este controle é feito de forma automática, uma gama de técnicas podem ser adotadas para solução do problema. As técnicas convencionais de controle (PD, PID), embora bastante comuns e largamente utilizadas na indústria, não apresentam um desempenho satisfatório frente a sistemas complexos, Lewis, Abdallah e Dawson (1993). Os métodos convencionais de controle de manipuladores, baseados em equações dinâmicas do modelo, requerem cálculo de forças e torque em tempo real. Para tal, são necessários modelos precisos e algoritmos eficientes.

Quando o problema em questão envolve incertezas, então tais controladores certamente terão um baixo desempenho e até mesmo podem tornar-se inviáveis do ponto de vista prático. Estes problemas têm motivado o desenvolvimento de pesquisas na área de identificação e controle adaptativo, que é o caso deste estudo.

Atualmente a literatura oferece duas abordagens principais para a utilização de Redes Neurais Artificiais em controle e identificação de sistemas dinâmicos: na primeira, o aprendizado de alguns parâmetros é feita *"off-line"* pela rede, utilizando medidas de entrada e saída da planta em algumas situações fundamentais. Na segunda, o aprendizado é feito *"on-line"*, com a Rede Neural adaptando seus pesos em função das variações do modelo dinâmico do sistema, o que se aplica perfeitamente ao controle de manipuladores, uma vez que estes possuem parâmetros que variam com o tempo.

Quando o problema envolve incertezas e há muito pouco conhecimento *a priori* dos objetos relacionados com a sua formulação, então buscas por soluções otimizadas certamente poderão ser muito custosas ou inviáveis, devido à ineficácia do processo iterativo de busca. Para tais casos pode-se adotar algum tipo de suavização ou relaxação da otimalidade pretendida, que conduza a soluções quase-ótimas que sejam satisfatórias dentro do contexto.

Esta abordagem vem sendo muito empregada em sistemas que resolvem problemas com o uso de técnicas computacionais de inteligência artificial, devido ao bom desempenho do processo de busca associado e ao fato de viabilizar a aplicação em sistemas reais. Os algoritmos genéticos, a lógica nebulosa e as redes neurais são exemplos de abordagens que, quando aplicadas com tais considerações, podem produzir excelentes resultados, principalmente no desenvolvimento de controladores para sistemas de engenharia, Souza (2000), Thapa, Jones e Zhu (2000).

Esta tese propõe uma técnica que integra algoritmos genéticos e teoria de redes neurais, visando o desenvolvimento automático de controladores de alto desempenho para servomecanismos tipo elo-acionado, ou módulo de junta robótica. Nesta abordagem, a teoria de redes neurais é utilizada no desenvolvimento de controladores não-lineares. Devido às características apresentadas, estes controladores possuem potencial para resolver uma enorme variedade de problemas, inclusive problemas para os quais os métodos convencionais não são eficientes ou aplicáveis. O preço pago por tal potencial está na complexidade envolvida no ajuste dos parâmetros destes controladores, Souza (2000), McCullagh, Choi e Bluff (1994).

Vários outros tipos de controladores não-lineares conhecidos e com características similares poderiam ser utilizados neste trabalho. Entretanto, os controladores neurais possuem o atrativo ou vantagem de poderem englobar em suas estruturas o conhecimento extraído pela experimentação do sistema (processo de aprendizagem), Cabrera e Narendra (1999), Narendra e Parthasarathy (1990).

Os métodos analíticos de projeto de controladores não-lineares geralmente são muito limitados e específicos, além de exigirem um bom conhecimento da dinâmica e dos parâmetros físicos do sistema a ser controlado. Portanto, é praticamente inviável produzir ferramentas computacionais baseadas em métodos analíticos de projeto para a geração automática de controladores não-lineares, Lewis, Jagannathan e Yesildirek (1999).

Os algoritmos genéticos são métodos de busca inspirados no processo evolutivo natural, e apresentam-se como uma alternativa eficiente para o ajuste automático de controladores não-lineares, Souza (2000). O algoritmo genético proposto neste trabalho é utilizado para o ajuste paramétrico de controladores neurais.

Tal abordagem mostrou-se muito eficiente, podendo-se assegurar sua aplicação com êxito, e com excelente relação de custo/benefício para o controle de juntas robóticas, havendo inclusive a possibilidade de extensão para uma infinidade de outros sistemas de engenharia que possuem dinâmica semelhante.

1.2 Redes Neurais

Os últimos anos têm mostrado um crescente interesse de pesquisadores no estudo da estrutura e mecanismos do cérebro. Este interesse levou, entre outras coisas, ao desenvolvimento de novos modelos computacionais, baseados no conhecimento biológico, para solução de problemas como reconhecimento de padrões, processamento de informação e predição de séries temporais.

Em meados de 1940, iniciaram-se os estudos da potencialidade da interconexão de vários componentes baseados no modelo do neurônio biológico. A partir disto vários trabalhos importantes foram desenvolvidos, como McCulloch e Pitt (1943), Hebb(1949), Grossberg(1976), Hopfield(1982), e uma quantidade substancial de pesquisadores aplicou-se no estudo e desenvolvimento do campo das redes neurais.

Hunt, Sbarbaro, Zbikowski e Gawthrop (1992) descrevem as seguintes características e propriedades de redes neurais como as mais importantes:

* **Sistemas Não-Lineares:** A possibilidade de aplicação de redes neurais no domínio de sistemas não-lineares, a qual provém da capacidade de aproximação de mapeamentos não-lineares.
* **Processamento Paralelo e Distribuído:** Redes neurais possuem arquitetura altamente paralela, assim, da implementação de uma rede neural pode-se esperar um sistema altamente tolerante a falhas, quando comparado com outros sistemas. Esta característica só se apresenta quando o sistema é implementado em ambientes que permitam a imlementação deste paralelismo.
* **Implementação em Hardware:** A possibilidade de implementação em hardware e o desenvolvimento de circuitos integrados (VLSI) conferem maior velocidade ao processamento das redes neurais.
* **Aprendizado e Adaptação:** Redes neurais são treinadas utilizando um conjunto de dados provenientes do sistema sob análise. Uma rede neural treinada apropriadamente possui habilidade de generalização quando dados ausentes no conjunto de treinamento são apresentados.
* **Fusão de Dados:** Redes neurais podem operar simultaneamente com dados quantitativos e qualitativos. Neste contexto representam um meio termo entre sistemas convencionais (quantitativos) e sistemas baseados em inteligência artificial (qualitativo).
* **Sistemas Multivariáveis:** Redes Neurais possuem habilidade natural para processar múltiplas entradas e apresentar múltiplas saídas, e são prontamente aplicadas a sistemas multivariáveis.

Do ponto de vista da teoria de controle, a habilidade das redes neurais em tratar com sistemas não-lineares é a sua característica mais importante. A grande diversidade de sistemas não-lineares é a principal razão pela qual uma teoria geral e sistemática para controle de sistemas não-lineares ainda não foi desenvolvida, Lewis et al. (1999).

1.3 Algoritmos Genéticos

O princípio da evolução é o conceito primordial da biologia que relaciona todos os organismos existentes numa cadeia histórica de eventos. Cada elemento da cadeia é produto de uma série de "acidentes" sucedidos de forma aleatória, sendo que a pressão seletiva do ambiente se encarrega de escolher "cuidadosamente" os indivíduos cujos genes irão perdurar, dando uma direção para a evolução ao decorrer de sucessivas gerações. Neste ponto é importante mencionar que cada organismo possui seu próprio genoma, e só no nível genético que um organismo consegue transmitir suas características à proxima geração.

Aqueles indivíduos que possuem boas qualidades relacionadas ao meio a que pertencem, têm grandes chances de continuarem existindo como tal, ou passarem suas características adiante através de reprodução. A evolução pode ser vista, resumidamente, como um processo iterativo de dois passos, consistindo de reprodução com variação aleatória, seguida de seleção "natural". O processo natural avalia os indivíduos através de certos critérios, normalmente vinculados ao nível de adaptação, que ditam se haverá sobrevivência individual e/ou manutenção da espécie em questão.

Os algoritmos de otimização baseados em processos evolutivos são chamados de Algoritmos Evolutivos, Holland (1975), e possuem como principal instância os algoritmos genéticos. Nos algoritmos genéticos, as soluções dos problemas são codificadas em cadeias de dados denominadas cromossomos, normalmente formadas por números binários. Cada indivíduo da população do algoritmo genético possui um cromossomo e um valor de adaptação (*fitness*). O algoritmo genético é iniciado a partir de uma população de indivíduos cujos cromossomos são gerados aleatoriamente ou a partir de informações *a priori* sobre as soluções do problema abordado. Os indivíduos da população inicial geram descendentes através do processo de reprodução, que envolve operadores de recombinação e mutação. Os indivíduos descendentes são avaliados através de uma função objetivo que mede seu grau de adaptação. A seleção consiste num processo de escolha realizado com base na medida de adaptação de cada indivíduo. Em média, os indivíduos mais adaptados tendem a passar para a próxima geração e os indivíduos menos adaptados tendem a ser eliminados.

O algoritmo genético, proposto neste trabalho para o ajuste paramétrico dos controladores neurais, utiliza codificação em ponto flutuante e operadores de reprodução especiais. Os parâmetros do controlador neural selecionados para ajuste são codificados numa cadeia cromossômica. Assim cada indivíduo tratado no algoritmo possui um cromossomo e representa um controlador neural.

O grau de adaptação de cada indivíduo é avaliado com base no desempenho do controlador neural associado, o qual é testado através da execução de uma tarefa de controle padrão que depende das especificações exigidas para o controlador. No cálculo da função objetivo podem ser utilizados parâmetros de medida de desempenho como sobresinal, erro de regime, tempo de estabilização, integral do erro, entre outros.

1.4 Algoritmos Genéticos e Controle Neural

A área de pesquisa de controladores neurais para sistemas não-lineares teve um progresso bastante significativo nos últimos anos, havendo muito avanço a partir das extensões permitidas pelo uso da computação evolutiva, Souza (2000). Mas apesar desses avanços, a criação e a aplicação prática destes tipos de controladores ainda está bastante defasada da grande parte do conhecimento teórico já produzido para emprego no controle de sistemas reais, tanto de grande, média, ou pequena complexidade, sendo que na atualidade há uma grande corrida tecnológica neste sentido, principalmente por parte da comunidade científica e dos setores industriais que empregam alta tecnologia no desenvolvimento de novos equipamentos e produtos.

Na atualidade, o projeto prático destes tipos de controladores possui fortes características restritivas, como grande complexidade computacional, operações com quantidades observáveis e não observáveis, e implementabilidade em tempo real, que tendem a reduzir a velocidade do avanço tecnológico pretendido.

Um dos primeiros trabalhos empregando algoritmos genéticos para o ajuste/geração de redes neurais foi desenvolvidos por Belew, McInerney e Schraudloph (1990). De um modo geral, os trabalhos nesta área propõem um algoritmo genético para o ajuste dos parâmetros de redes neurais, como taxa de aprendizado, termo *momentum*, número de neurônios em cada camada e número de camadas da rede, e ajuste dos pesos da rede. Na maioria dos casos, os resultados apresentados limitam-se a simulações, sem tratar da viabilidade de sua implementação prática.

1.5 Organização da Tese

Esta tese propõe o emprego do controlador neural de *Kim-Lewis-Dawson*, que, devido às suas características, possui potencial para resolver o problema de rastreamento de trajetória no controle de um módulo de junta robótica. Problema para o qual os métodos convencionais não são eficientes ou aplicáveis.

Nesta abordagem a teoria de redes neurais é utilizada no ajuste dos parâmetros livres do controlador, com o objetivo de gerar automaticamente um controlador neural otimizado para a tarefa de controle em questão.

O controlador otimizado é então aplicado em conjunto com um observador neural de estados, visando eliminar a presença de tacômetros e contornar os problemas da diferenciação numérica, mas mantendo a precisão do sistema de controle, além da redução de peso e volume do sistema como um todo.

O trabalho proposto possui a seguinte estrutura:

O **Capítulo 2** apresenta, inicialmente, uma introdução sobre os principais conceitos da teoria de redes neurais, tratando resumidamente sua formulação matemática, e uma breve

introdução sobre a utilização de redes neurais no controle de sistemas dinâmicos. Em seguida é apresentada a formulação básica do algoritmo genético clássico, é apresentada uma explicação detalhada sobre o algoritmo genético com codificação em ponto flutuante, proposto para o problema de otimização de controladores neurais. Finalmente, é apresentada a função objetivo utilizada para avaliação dos controladores neurais.

O **Capítulo 3** apresenta o controlador neural de *Kim-Lewis-Dawson*, tratando de modo detalhado sua formulação matemática. São apresentadas suas principais características e os resultados práticos obtidos na otimização destes controladores neurais quando aplicados ao controle de um pêndulo acionado.

O **Capítulo 4** estuda a aplicação de um observador neural em conjunto com o controlador de *Kim-Lewis-Dawson*. A formulação matemática para tal controlador e os principais resultados práticos, obtidos quando aplicado ao controle de um pêndulo acionado, são apresentados.

O **Capítulo 5** argumenta as contribuições do trabalho e apresenta uma lista de sugestões e perspectivas para trabalhos futuros.

O **Apêndice A** trata dos detalhes técnicos de implementação do hardware e software de controle em tempo real, apresentando também os diagramas de bloco do circuito eletrônico desenvolvido para os experimentos.

Capítulo 2

Conceitos Básicos

2.1 Redes Neurais

2.1.1 Introdução

O final da década de 80 marcou o ressurgimento da área de Redes Neurais Artificiais (RNAs). Esta forma de computação não algorítmica é caracterizada por sistemas que, em algum nível, lembram a estrutura do cérebro humano. Por não ser baseada em regras ou programas, a computação neural se constitui em uma alternativa à computação algorítmica convencional.

RNAs são sistemas paralelos e distribuídos compostos por unidades de processamento simples que executam determinadas funções matemáticas. Tais unidades são dispostas em uma ou mais camadas e interligadas por um grande número de conexões, geralmente unidirecionais. Na maioria dos modelos estas conexões estão associadas a pesos, os quais armazenam o conhecimento representado no modelo e servem para ponderar a entrada recebida pelo neurônio da rede.

Em RNAs, o procedimento usual na solução de problemas passa inicialmente por uma fase de aprendizagem, em que um conjunto de exemplos é apresentado para a rede, a qual adquire automaticamente as características necessárias para representar a informação fornecida. Essas características são utilizadas posteriormente para gerar respostas para o problema.

A capacidade de aprender através de exemplos e de generalizar a informação aprendida são, sem dúvida, os atrativos principais da solução de problemas através de RNAs. A generalização, que está associada à capacidade da rede aprender através de um conjunto reduzido de exemplos e posteriormente fornecer respostas coerentes para dados não apresentados na fase de treinamento, é uma demonstração de que a capacidade das RNAs vai muito além de simplesmente mapear relações entre entradas e saídas.

Outra característica importante é a capacidade de auto-organização e de processamento temporal que, aliada à capacidade de atuar como mapeadores universais, fazem das RNAs uma ferramenta computacional extremamente eficiente e atrativa para a solução de problemas complexos.

2.1.2 Modelo Matemático do Neurônio

O neurônio é a unidade fundamental de processamento de informação em uma rede neural. A Fig.2.1 mostra o modelo de um neurônio, onde podemos identificar três partes principais:

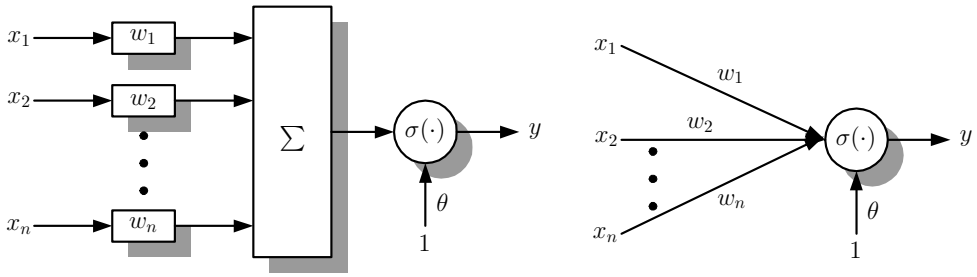

Figura 2.1: Modelo Não Linear do Neurônio: (a)Modelo Completo. (b)Modelo Simplifidado.

- **Conjunto de pesos:** Determinam as intensidades que o neurônio deve atribuir aos sinais de entrada.
- **Somador:** Efetua a soma ponderada dos valores $w_j x_j$ recebidos pelo neurônio.
- **Função de Ativação:** Limita a amplitude do sinal de saída do neurônio.

Em termos matemáticos a saída y pode ser descrita como

$$y = \sigma\left(\sum_{j=1}^{n} w_j x_j + \theta\right) \tag{2.1}$$

onde $x_1, x_2, \ldots, x_n$ são os sinais de entrada, $w_1, w_2, \ldots, w_n$ são os pesos do neurônio, θ corresponde ao limiar do neurônio e $\sigma(\cdot)$ é a função de ativação.

A função de ativação, $\sigma(\cdot)$, define o valor do sinal de saída do neurônio, e é selecionada de acordo com a aplicação dada para a rede neural. Existem várias funções que podem ser usadas como função de ativação de um neurônio. Entre elas, encontram-se a função degrau, a função linear, as funções seno e cosseno, e as funções sigmoidais, Haikin (1994).

A expressão matemática da saída y pode ser escrita na forma vetorial, definindo os sinais de entrada como um vetor $\mathbf{x} \in \mathbb{R}^n$, e os pesos como um vetor $\mathbf{w} \in \mathbb{R}^n$, assim

$$\mathbf{x} = \begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix}^T, \qquad \mathbf{w} = \begin{bmatrix} w_1 & w_2 & \dots & w_n \end{bmatrix}^T. \tag{2.2}$$

Então podemos equacionar a saída y como

$$y = \sigma\left(\mathbf{w}^T\mathbf{x} + \theta\right) \tag{2.3}$$

Na Fig.2.2 é apresentada uma rede neural que consiste de m neurônios, todos alimentados pelo mesmo vetor $\mathbf{x} \in \mathbb{R}^n$, e cada neurônio produzindo uma saída y_k, esta estrutura é chamada de rede neural de uma camada.

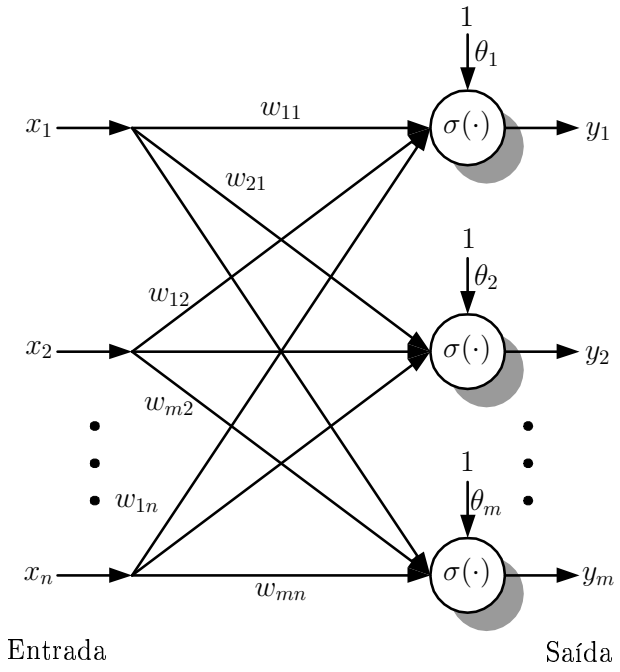

Figura 2.2: Rede Neural de uma Camada.

A equação para esta rede é dada por

$$y_k = \sigma\left(\sum_{j=1}^{n} w_{kj}x_j + \theta_k\right); \qquad k = 1, 2, ..., m. \tag{2.4}$$

Definindo o conjunto dos valores dos pesos e limiares como

$$\mathbf{W}^T = \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1n} \\ w_{21} & w_{22} & \dots & w_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m1} & w_{m2} & \dots & w_{mn} \end{bmatrix}, \qquad b = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_m \end{bmatrix} \tag{2.5}$$

podemos expressar a saída da rede por

$$\mathbf{y} = \sigma\left(\mathbf{W}^T\mathbf{x} + b\right). \tag{2.6}$$

Incluindo no vetor de entrada $\mathbf{x}$, mais um elemento de valor 1, e incluindo o vetor de limiares na matriz de pesos temos

$$\mathbf{x} = \begin{bmatrix} 1 & x_1 & x_2 & \dots & x_n \end{bmatrix}^T, \tag{2.7}$$

$$W^T = \begin{bmatrix} b_1 & w_{11} & w_{12} & \dots & w_{1n} \\ b_2 & w_{21} & w_{22} & \dots & w_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ b_m & w_{m1} & w_{m2} & \dots & w_{mn} \end{bmatrix}, \tag{2.8}$$

e podemos escrever a saída da rede como

$$\mathbf{y} = \sigma\left(\mathbf{W}^T\mathbf{x}\right). \tag{2.9}$$

Esta notação será adotada nos capítulo subseqüentes desta dissertação.

2.1.3 Redes Neurais Multicamadas

A Fig.2.3 mostra uma rede neural com duas camadas de neurônios, onde as saídas da primeira camada alimentam a entrada dos neurônios da segunda camada. A primeira camada é chamada de camada escondida ou intermediária, e a segunda camada é chamada camada de saída. Redes com múltiplas camadas são conhecidas como redes MLP *"Multi Layer Perceptron"*. O poder de processamento das redes MLP é muito superior ao das redes de uma única camada.

O sinal de saída da rede apresentada na Fig.2.3 é definido da seguinte forma

$$\mathbf{y} = \sigma\left(\mathbf{W}^T\sigma\left(\mathbf{V}^T\mathbf{x}\right)\right); \tag{2.10}$$

onde $V \in \mathbb{R}^{L\times n}$ representa os pesos dos neurônios da primeira camada e $W \in \mathbb{R}^{L\times n}$ representa os pesos dos neurônios da segunda camada. Se na camada de saída a função de ativação for uma

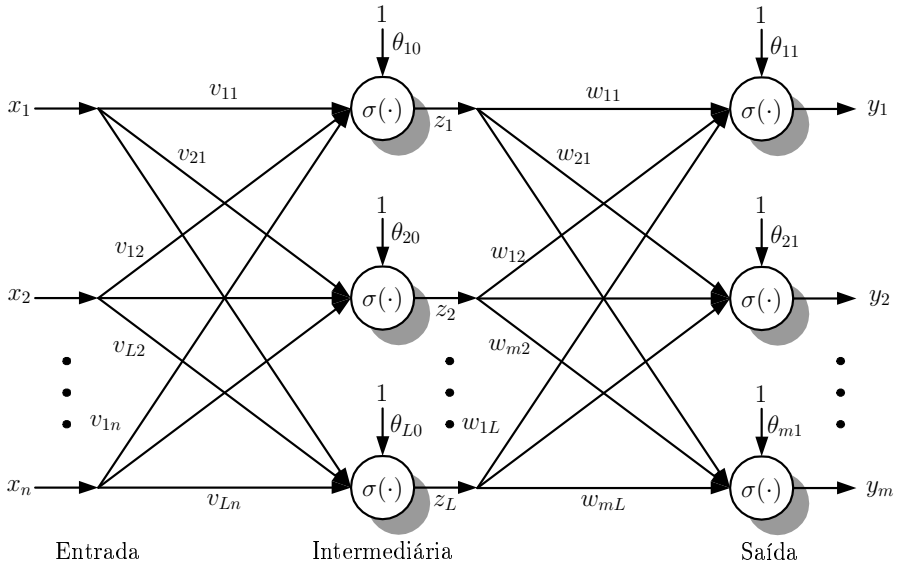

Figura 2.3: Rede Neural de Duas Camadas.

função linear do tipo $\sigma(x) = x$, então a equação de saída da rede torna-se

$$\mathbf{y} = \mathbf{W}^T \sigma\big(\mathbf{V}^T \mathbf{x}\big). \tag{2.11}$$

2.1.4 Treinamento de Redes Neurais Multicamadas

Existe uma infinidade de algoritmos capazes de treinar redes neurais, sendo que estes podem ser agrupados em dois grandes grupos: Aprendizado Supervisionado e Aprendizado Não Supervisionado.

Aprendizado Supervisionado: É o aprendizado mais comum no treinamento de RNAs. Dado um conjunto de entradas e um conjunto de saídas desejadas, o objetivo é ajustar os pesos da rede de tal forma que a saída da rede seja mais próximo possível do conjunto de saídas desejadas.

Aprendizado Não Supervisionado: Neste tipo de aprendizado, somente os dados de entrada estão disponíveis, caracterizando a ausência do par entrada/saída. Nesta forma de aprendizado busca-se encontrar uma "harmonia estatística" contida nos dados de entrada, desenvolvendo-se uma habilidade para criar classes de dados ou grupos automaticamente.

Como a rede neural que compõe a arquitetura do controlador de *Kim-Lewis-Dawson* possui uma arquitetura que exige treinamento supervisionado, o trabalho concentrou-se neste tipo de treinamento.

Existem vários métodos ou algoritmos de sintonia de parâmetros em redes neurais. Usualmente, estes algoritmos são baseados em algum método de otimização. Existem duas linhas básicas

para o desenvolvimento de tais algoritmos: uma baseada no uso do vetor gradiente, e outra no uso da matriz hessiana. Os métodos baseados na matriz hessiana normalmente requerem elevado esforço computacional por iteração, enquanto métodos baseados nos vetores gradiente requerem esforço computacional relativamente menor.

O algoritmo *Back-propagation*, Rumelhart, Hinton e Williams (1986), é baseado no método de otimização pelo vetor gradiente, sendo um dos algoritmos mais conhecidos para o treinamento de redes neurais.

2.1.5 Algoritmo *Back-Propagation*

Dada uma rede neural de duas camadas, como a rede neural apresentada na Fig.2.3, tem-se o seguinte modelo

$$y_i = \sigma\left(\sum_{\ell=1}^{L} w_{i\ell}\sigma\left(\sum_{j=1}^{n} v_{\ell j}x_j + \theta_{\ell 0}\right) + \theta_{i1}\right); \qquad i = 1, 2, \ldots, m. \tag{2.12}$$

Na Fig.2.3, chamamos os pesos da primeira camada de $v_{\ell j}$ e os pesos da segunda camada de $w_{i\ell}$. A entrada da primeira camada é x_j. Definindo como entrada da segunda camada

$$z_\ell = \sigma\left(\sum_{j=1}^{n} v_{\ell j}x_j + \theta_{\ell 0}\right); \qquad \ell = 1, 2, \ldots, L. \tag{2.13}$$

e adicionando os limiares à matriz de pesos, podemos escrever

$$y_i = \sigma\left(\sum_{\ell=0}^{L} w_{i\ell}z_\ell\right) \tag{2.14}$$

$$z_\ell = \sigma\left(\sum_{j=0}^{n} v_{\ell j}x_j\right). \tag{2.15}$$

Definindo a saída das camadas da rede como

$$u_i^2 = \sum_{\ell=0}^{L} w_{i\ell}z_\ell \tag{2.16}$$

$$u_\ell^1 = \sum_{j=0}^{n} v_{\ell j}x_j \tag{2.17}$$

podemos escrever

$$y_i = \sigma\left(u_i^2\right) \tag{2.18}$$

$$z_\ell = \sigma\left(u_\ell^1\right) \tag{2.19}$$

O algoritmo *Back-Propagation* é um algoritmo de gradiente descendente, Sjöberg e Ljung (1995) , e os pesos deste tipo de rede podem ser ajustados pelas seguintes equações, Lewis et al. (1999)

$$w_{i\ell} = w_{i\ell} - \eta \frac{\partial E}{\partial w_{i\ell}} \tag{2.20}$$

$$v_{\ell j} = v_{\ell j} - \eta \frac{\partial E}{\partial v_{\ell j}} \tag{2.21}$$

onde E é a função custo, definida por

$$E = \frac{1}{2} e^T e = \frac{1}{2} \sum_{i=1}^{m} e_i^2. \tag{2.22}$$

Dado um determinado vetor X, como entrada da rede, e um vetor Y, como a saída desejada para o vetor de entrada X, definimos

$$e_i = Y_i - y_i \tag{2.23}$$

onde y_i é calculado usando a Eq.2.12. O gradiente da função custo E em relação aos pesos da camada de saída é dado por

$$\frac{\partial E}{\partial w_{i\ell}} = \frac{\partial E}{\partial u_i^2} \frac{\partial u_i^2}{\partial w_{i\ell}} = \left[\frac{\partial E}{\partial e_i} \frac{\partial e_i}{\partial y_i} \frac{\partial y_i}{\partial u_i^2} \right] \frac{\partial u_i^2}{\partial w_{i\ell}} \tag{2.24}$$

e como

$$\frac{\partial E}{\partial e_i} = e_i \tag{2.25}$$

$$\frac{\partial e_i}{\partial y_i} = -1 \tag{2.26}$$

$$\frac{\partial y_i}{\partial u_i^2} = \sigma'(u_i^2) \tag{2.27}$$

podemos deduzir que

$$\frac{\partial E}{\partial u_i^2} = -\sigma' \left(u_i^2 \right) e_i \tag{2.28}$$

$$\frac{\partial E}{\partial w_{i\ell}} = -z_\ell \left[\sigma' \left(u_i^2 \right) e_i \right]. \tag{2.29}$$

De modo equivalente, para a primeira camada temos

$$\frac{\partial E}{\partial v_{\ell j}} = \frac{\partial E}{\partial u_\ell^1} \frac{\partial u_\ell^1}{\partial v_{\ell j}} = \left[\frac{\partial z_\ell}{\partial u_\ell^1} \sum_{i=1}^{m} \frac{\partial E}{\partial u_i^2} \frac{\partial u_i^2}{\partial z_\ell} \right] \frac{\partial u_\ell^1}{\partial v_{\ell j}} \tag{2.30}$$

como

$$\frac{\partial u_i^2}{\partial z_\ell} = w_{i\ell} \tag{2.31}$$

$$\frac{\partial z_\ell}{\partial u_\ell^1} = \sigma'(u_i^1) \tag{2.32}$$

$$\frac{\partial u_\ell^1}{\partial v_{\ell j}} = x_j \tag{2.33}$$

podemos deduzir que

$$\frac{\partial E}{\partial u_\ell^1} = -\sigma'\left(u_\ell^1\right) \sum_{i=1}^{m} w_{i\ell} \left[\sigma'\left(u_i^2\right) e_i\right] \tag{2.34}$$

$$\frac{\partial E}{\partial v_{\ell j}} = -x_j \left[\sigma'\left(u_\ell^1\right) \sum_{i=1}^{m} w_{i\ell} \left[\sigma'\left(u_i^2\right) e_i\right]\right]. \tag{2.35}$$

Assim podemos definir as equações de retro propagação do erro como

$$\delta_i^2 = -\frac{\partial E}{\partial u_i^2} = \sigma'\left(u_i^2\right) e_i \tag{2.36}$$

$$\delta_\ell^1 = -\frac{\partial E}{\partial u_\ell^1} = \sigma'\left(u_\ell^1\right) \sum_{i=1}^{m} w_{i\ell} \left[\sigma'\left(u_i^2\right) e_i\right]. \tag{2.37}$$

e a equação de ajuste dos pesos fica

$$w_{i\ell} = w_{i\ell} + \eta z_\ell \delta_i^2 \tag{2.38}$$

$$v_{\ell j} = v_{\ell j} + \eta x_j \delta_\ell^1 \tag{2.39}$$

Assumindo que as funções de ativação sejam funções sigmoidais, os erros de retro propagação podem ser equacionados como

$$\delta_i^2 = y_i \left(1 - y_i\right) e_i \tag{2.40}$$

$$\delta_\ell^1 = z_\ell \left(1 - z_\ell\right) \sum_{i=1}^{m} w_{i\ell} \delta_i^2 \tag{2.41}$$

devido ao fato de

$$\sigma(s) = \frac{1}{1 + e^{-s}} \tag{2.42}$$

$$\sigma'(s) = \sigma(s)\left(1 - \sigma(s)\right) \tag{2.43}$$

Em termos matriciais, podemos escrever que

$$\mathbf{z} = \sigma\left(\mathbf{V}^T \mathbf{X}\right) \tag{2.44}$$

$$\mathbf{y} = \sigma\left(\mathbf{W}^T \mathbf{z}\right) \tag{2.45}$$

$$\mathbf{e} = \mathbf{Y} - \mathbf{y} \tag{2.46}$$

$$\delta^2 = diag[\mathbf{y}]\,(I - diag[\mathbf{y}])\,\mathbf{e} \tag{2.47}$$

$$\delta^1 = diag[\mathbf{z}]\,(I - diag[\mathbf{z}])\,\mathbf{W}\delta^2 \tag{2.48}$$

e as equações de ajustes dos pesos são dadas por

$$\mathbf{W} = \mathbf{W} + \eta\mathbf{z}(\delta^2)^T \tag{2.49}$$

$$\mathbf{V} = \mathbf{V} + \eta\mathbf{X}(\delta^1)^T. \tag{2.50}$$

2.1.6 Redes Neurais Aplicadas ao Controle de Sistemas Dinâmicos

Duas classes distintas de problemas aparecem na indústria com grande freqüência, e merecem especial atenção no momento do projeto dos sistemas de controle. A primeira consiste de sistemas que foram projetados para operar de modo satisfatório apenas na vizinhança de um certo ponto de operação. Neste domínio a modelagem linear do problema e os controladores lineares podem conduzir a desempenhos plenamente satisfatórios. Porém com a crescente demanda de tecnologia, os sistemas estão sendo forçados a trabalhar em regiões cada vez maiores nos seus espaços de estado, onde acentuam-se as características não-lineares. Neste caso tanto a modelagem linear quanto o controle linear não conseguem atingir desempenhos satisfatórios.

Na segunda classe de problemas, modelos matemáticos não-lineares devem ser desenvolvidos, o que pode ser uma tarefa bastante árdua. Existe também, a possibilidade de uma certa quantidade de dados de entrada-saída do sistema não controlado estar disponível, de modo que um modelo e um controlador não-linear podem ser aproximados de forma adequada, segundo um determinado índice de desempenho, Cabrera e Narendra (1999).

As características não-lineares inerentes ao mapeamento entre as camadas das RNAs caracterizam-nas como ferramentas de modelagem bastante apropriadas para sistemas não lineares. Ferramentas para análise de sistemas não-lineares normalmente utilizam técnicas de linearização, transformando a tarefa de controle não linear em pequenas tarefas de controle linear, ou seja, controle linear por partes. Apesar de muitas vezes eficiente, essa abordagem não representa a realidade dos sistemas físicos, podendo resultar em soluções sub-ótimas.

Do ponto de vista da modelagem de sistemas, as técnicas neurais têm o seu resultado comparado com métodos clássicos. Porém, ainda não é possível saber, de modo genérico, em que situações a abordagem por RNAs é mais atrativa que os métodos tradicionais. Estas situações apresentam, de um modo geral, o problema de modelagem de sistemas complexos e estimativa de parâmetros com grande quantidade de variáveis.

A abordagem mais usual para controladores neurais é a utilização de redes neurais com treinamento *"off-line"*, isto é, primeiro o controlador neural aprende as características da planta e do controlador, para, somente depois de treinado, passar a controlar a planta real.

A Fig.2.4 mostra de um modo resumido três passos que devem ser seguidos na tarefa de identificação e controle de plantas não-lineares usando redes neurais, Cabrera e Narendra (1999).

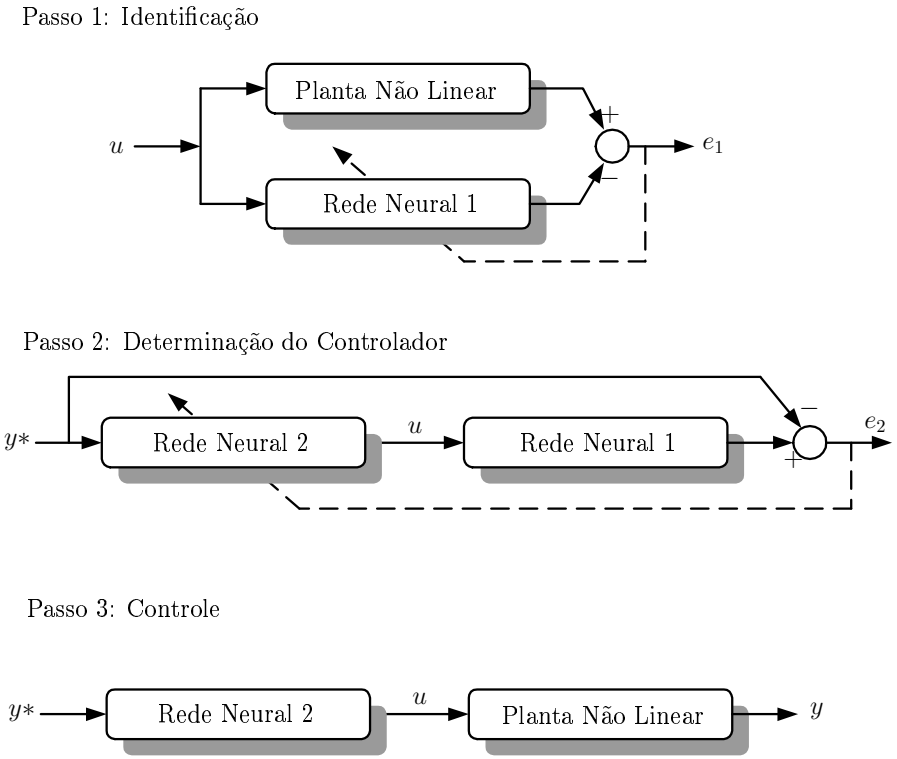

Figura 2.4: Uma Possível Abordagem Para Identificação e Controle de Planta Usando Rede Neural.

O primeiro passo consiste de uma rede colocada em paralelo com a planta e treinada para aproximar seu modelo, a Fig.2.5 mostra o sistema com mais detalhes. O Bloco chamado de Rede Neural 1 tem a função de representar o mapa

$$\begin{aligned} \widehat{y}(k+1) &= RN_1[y(k), y(k-1), \ldots, y(k-\ell_0+1), \\ &\quad u(k), u(k-1), \ldots, u(k-\ell_1+1)]. \end{aligned} \tag{2.51}$$

RN_1 representa uma rede neural com $\ell_0+\ell_1$ entradas e uma saída, treinada para aproximar o mapa que descreve a planta não linear. Quando o erro $e_1(k)$, entre $y(k)$ e $\widehat{y}(k)$, torna-se suficientemente pequeno, encerra-se o primeiro passo.

O segundo passo consiste no emprego de uma segunda rede neural, chamada de Rede Neural 2, que possui sua saída alimentando RN1, obtida no passo anterior

$$\begin{aligned} u(k+1) &= RN_2[\hat{y}(k), \hat{y}(k-1), \ldots, \hat{y}(k-\ell_2+1), \\ &\quad u(k-1), u(k-2), \ldots, u(k-\ell_3+1), y^*(k+d)]. \end{aligned} \tag{2.52}$$

RN_2 representa uma rede neural com $\ell_2+\ell_3$ entradas e uma saída, treinada para aproximar a lei de controle, d representa o atraso da planta e y^* é a trajetória de referência. RN_2 é treinada de modo que o erro $e_2(k)$, entre $y(k)$ e $\widehat{y}(k)$, torne-se suficientemente pequeno. O objetivo é treinar RN_2 para um determinado conjunto de trajetórias. A Fig.2.6 mostra o segundo passo com maiores detalhes.

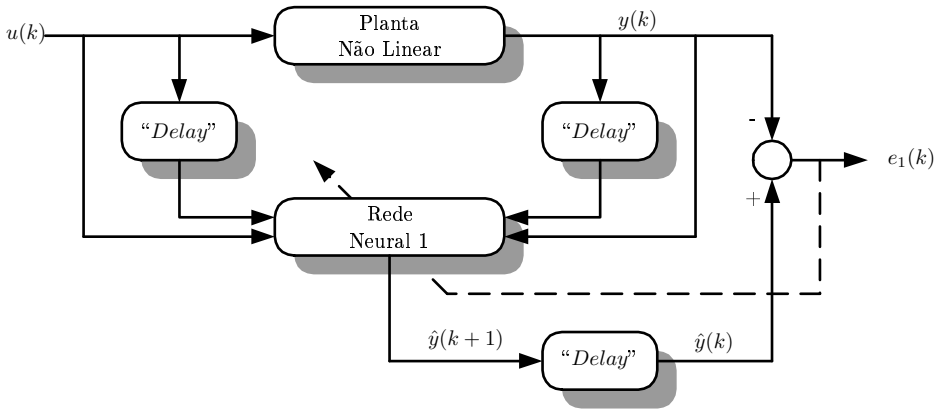

Figura 2.5: Identificação Usando Rede Neural.

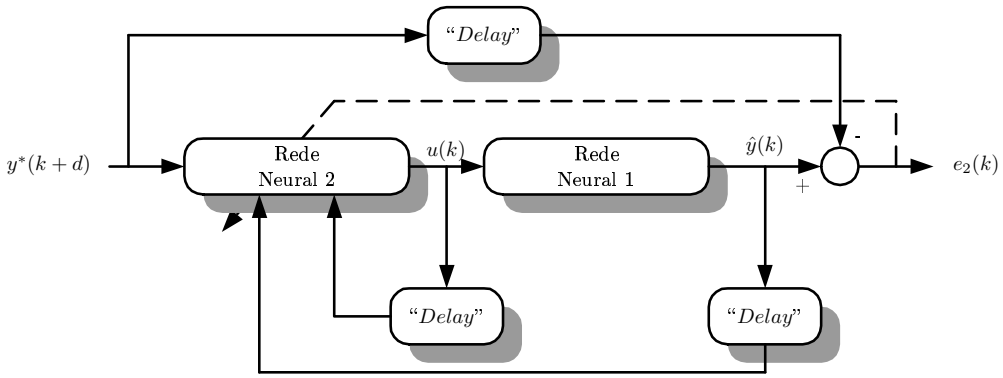

Figura 2.6: Determinação do Neurocontrolador.

Depois destes dois passos o controlador fica projetado, e a rede RN_2 é usada em conjunto com a planta, como mostrado na Fig.2.4, caracterizando o terceiro passo.

Da topologia básica mostrada acima, foram derivados dezenas de controladores neurais para aplicação em controle de sistemas não-lineares. Maiores informações podem ser obtidas em Hunt et al. (1992), Cabrera e Narendra (1999), Levin e Narendra (1993), Levin e Narendra (1996), Narendra e Parthasarathy (1990), Narendra (1997).

2.2 Algoritmos Genéticos

2.2.1 Introdução

Uma tarefa de busca e otimização possui vários componentes que devem ser levados em consideração, entre eles o espaço de busca, onde são consideradas todas as possibilidades de solução

de um determinado problema, e a função de avaliação (ou função de custo), uma maneira de avaliar os elementos do espaço de busca.

As técnicas de busca e otimização tradicionais iniciam-se com uma única solução que, iterativamente, é manipulada utilizando algum método de busca diretamente associado ao problema a ser solucionado. Por outro lado, as técnicas de computação evolutiva operam sobre uma população de soluções candidatas. Assim, elas podem fazer a busca em diferentes áreas do espaço de soluções, alocando um número de membros apropriado para a busca nas várias regiões envolvidas.

Os Algoritmos Genéticos (AGs) diferem dos métodos tradicionais de busca e otimização, principalmente em quatro aspectos:

- AGs trabalham com uma codificação do conjunto de parâmetros e não com os próprios parâmetros.
- AGs trabalham com uma população e não com um único ponto.
- AGs utilizam informações de custo, ou recompensa, e não derivadas ou outro conhecimento de maior qualidade.
- AGs utilizam regras de transição probabilísticas.

Os algoritmos genéticos são muito eficientes na busca por soluções ótimas, ou aproximadamente ótimas para uma grande variedade de problemas, pois não impõem limitações encontradas nos métodos de busca tradicionais. Além de ser uma estratégia de gerar-e-testar, por ser baseada na evolução biológica, ela permite identificar e explorar fatores ambientais e convergir para soluções ótimas, ou aproximadamente ótimas, em níveis globais. "Quanto melhor um indivíduo se adaptar ao seu meio ambiente, maior será sua chance de sobreviver e gerar descendentes", este é o conceito básico da evolução genética biológica.

Um algoritmo genético possui, basicamente, uma dinâmica em que inicialmente é gerada uma população, formada por um conjunto aleatório com distribuição uniforme de indivíduos, que podem ser vistos como possíveis soluções do problema. Durante o processo evolutivo, esta população é avaliada e para cada indivíduo é dada uma nota, ou índice, refletindo sua habilidade de adaptação a determinado ambiente. Uma porcentagem dos indivíduos mais adaptados é mantida, enquanto os indivíduos menos adaptados são descartados (Darwinismo). Os membros mantidos pela seleção podem sofrer modificações em suas características fundamentais através de mutações e ou recombinação genética (*crossover*), gerando descendentes para a próxima geração. Este processo, chamado de reprodução, é repetido até que uma solução satisfatória seja encontrada. Neste tipo de processo de evolução não existem restrições de convexidade ou exigências de continuidade da superfície de adaptação no espaço de busca, a qual deve ser maximizada.

2.2.2 O Algoritmo Genético Clássico

Algoritmos Genéticos são algoritmos de otimização global, baseados nos mecanismos de seleção natural e da genética. Eles empregam uma estratégia de busca paralela e estruturada, mas

aleatórias, que é voltada em direção ao reforço da busca de pontos de "alta aptidão", ou seja, pontos nos quais a função a ser minimizada (ou maximizada) tem valores relativamente baixos (ou altos).

Apesar de etapas aleatórias, elas não são necessariamente caminhadas não direcionadas (busca cega), pois exploram informações históricas para encontrar novos pontos de busca onde são esperados melhores desempenhos. Isto é feito através de processos iterativos, onde cada iteração é chamada de geração. Durante cada iteração, os princípios de seleção e reprodução são aplicados a uma população de candidatos.

Através da seleção, determina-se quais indivíduos conseguirão se reproduzir, gerando um número determinado de descendentes para a próxima geração, com uma probabilidade determinada pelo seu índice de aptidão. Em outras palavras, os indivíduos com maior adaptação relativa têm maiores chances de se reproduzir.

O ponto de partida para a utilização de Algoritmos Genéticos, como ferramenta para solução de problemas, é a representação destes de maneira que os *AGs* possam trabalhar adequadamente sobre eles. A maioria das representações são genotípicas e utilizam vetores de tamanho finito em um alfabeto finito.

Tradicionalmente os indivíduos são representados genotipicamente por vetores binários, onde cada elemento de um vetor denota a presença (1) ou ausência (0) de uma determinada característica: o seu genótipo. Os elementos podem ser combinados formando as características reais do indivíduo, ou o seu fenótipo. Teoricamente, esta representação é independente do problema, pois uma vez encontrada a representação em vetores binários, as operações-padrão podem ser utilizadas, facilitando o seu emprego em diferentes classes de problemas.

O algoritmo genético clássico utiliza cadeias binárias para representar as variáveis de busca. Estas cadeias binárias são denominadas de cromossomos, Michalewicz (1999). O *AG* trabalha com uma população de cromossomos (ou indivíduos), cada um representando uma possível solução para o problema em questão, e possuindo um valor de adaptação, determinado pela função de aptidão, que indica a qualidade da solução. Em um problema de otimização, o AG objetiva maximizar a função de *fitness*.

Suponha que se deseja maximizar uma função de k variáveis, $f(x_1, \cdots, x_k)$:$\mathbb{R}^k \to \mathbb{R}$, e que cada variável x_i possa assumir valores no intervalo $D_i = [a_i, b_i] \subseteq \mathbb{R}$ e $f(x_1, \cdots, x_k) > 0$ para todo $x_i \in D_i$. Para obter-se uma precisão de 6 casas decimais, cada intervalo D_i deve ser dividido em $(b_i - a_i) \cdot 10^6$ partes iguais. Seja m_i o menor inteiro tal que $(b_i - a_i) \cdot 10^6 \leq 2^{m_i} - 1$. Então a representação de cada variável x_i por uma cadeia binária de comprimento m_i satisfaz a precisão requerida. A seguinte fórmula converte o código binário de uma variável para o seu valor real:

$$x_i = a_i + decimal(100 \cdots 0101_2) \cdot \frac{b_i - a_i}{2^{m_i} - 1} \tag{2.53}$$

onde a função $decimal(\cdot)$ retorna o valor decimal da cadeia binária, Michalewicz (1999).

Com isso, cada cromossomo (solução possível) é representado por uma cadeia binária de comprimento $m = \Sigma_{i=1}^{k} m_i$; os primeiros m_1 bits representam um valor x_1 definido em $[a_1, b_1]$, o

próximo grupo de m_2 bits representam um valor x_2 definido em $[a_2, b_2]$, e assim por diante, Fig.2.7.

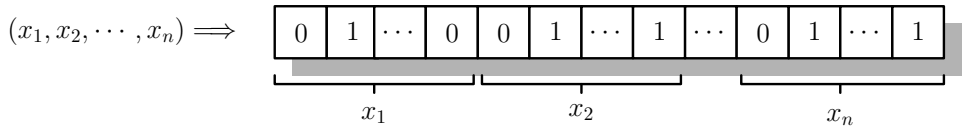

Figura 2.7: Codificação das Variáveis de Busca.

Em geral, os algoritmos genéticos tratam com populações contendo um número fixo de cromossomos. A população inicial é constituída de cromossomos cujos bits são gerados aleatoriamente. Entretanto, se existe alguma informação *a priori* sobre a região que contém as possíveis soluções ótimas, esta informação pode ser utilizada na sua geração.

As buscas por melhores soluções são realizadas por operadores genéticos, tipicamente recombinação (*crossover*) e mutação. O processo de aplicação destes operadores é chamado de reprodução. Após a reprodução, os cromossomos da população resultante devem ser avaliados através da função de *fitness*. Os cromossomos que passarão para a próxima geração são selecionados aleatoriamente de acordo com os seus valores de *fitness*, ou seja, o valor de *fitness* de um dado cromossomo indica a probabilidade deste passar para a próxima geração.

Após satisfeito algum critério de parada, o processo de evolução é finalizado, e a melhor solução encontrada é apresentada. A Fig.2.8 ilustra o fluxograma básico de um *AG*.

A evolução é um processo que opera sobre os cromossomos e não sobre os organismos. O processo evolutivo se dá durante a etapa de reprodução. Existem dois parâmetros relacionados ao processo de reprodução: a probabilidade de *crossover* p_c e a probabilidade de mutação p_m.

O *crossover* é o operador responsável pela recombinação de características dos pais durante a reprodução, permitindo que as próximas gerações herdem essas características. Ele é considerado o operador genético predominante, por isso é aplicado com probabilidade dada pela taxa de *crossover* p_c, que por conveniência deve ser maior que a taxa de mutação. Este operador pode ser utilizado de várias maneiras, sendo que as mais utilizadas são:

- Um-ponto: um ponto de cruzamento é escolhido e a partir deste ponto as informações genéticas dos pais serão trocadas. As informações anteriores a este ponto em um dos pais são ligadas às informações posteriores a este ponto no outro pai, como é mostrado no exemplo da Fig.2.9.
- Multi-pontos: é uma generalização desta idéia de troca de material genético através de pontos, onde muitos pontos de cruzamento podem ser utilizados.
- Uniforme: não utiliza pontos de cruzamento, mas determina, através de um parâmetro global, qual a probabilidade de cada variável ser trocada entre os pais.

Para realizar a operação de *crossover* simples nos indivíduos de uma população procede-se da seguinte maneira: para cada par de cromossomos na população executa-se os seguintes passos, Michalewicz (1999):

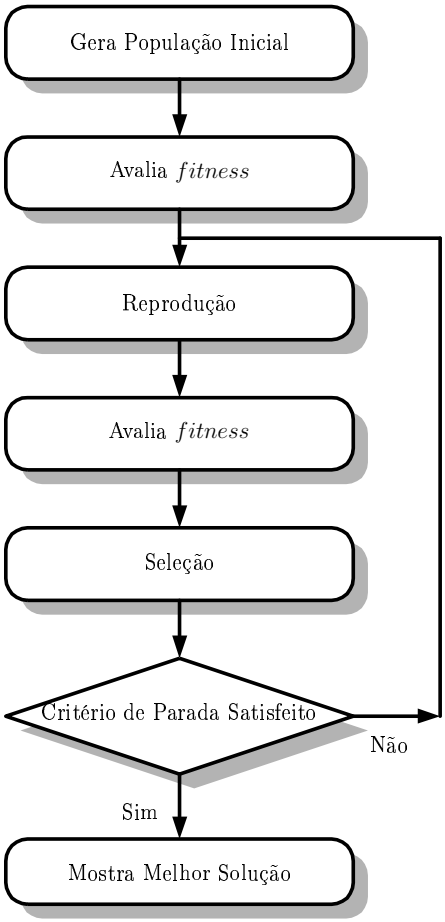

Figura 2.8: Fluxograma Básico de Um AG.

1. Gera-se um número aleatório real, r, com distribuição uniforme no intervalo $[0, 1]$.
2. Se $r < p_c$ então execute a recombinação da seguinte forma:
   - Gera-se um número aleatório (pos)com distribuição uniforme no intervalo $[1, m-1]$ que indica a posição de corte no cromossomo, onde m é o número de bits do cromossomo.
   - Os dois cromossomos
     $(a_1 a_2 \cdots a_{pos} a_{pos+1} \cdots a_m)$
     $(b_1 b_2 \cdots b_{pos} b_{pos+1} \cdots b_m)$
     são substituidos por:
     $(a_1 a_2 \cdots a_{pos} b_{pos+1} \cdots b_m)$
     $(b_1 b_2 \cdots b_{pos} a_{pos+1} \cdots a_m)$

Na reprodução, o processo de cópia do material genético dos pais está sujeito a erros, e a troca aleatória de um gene por outro neste processo é chamada de mutação.

O operador de mutação é necessário para a introdução e manutenção da diversidade

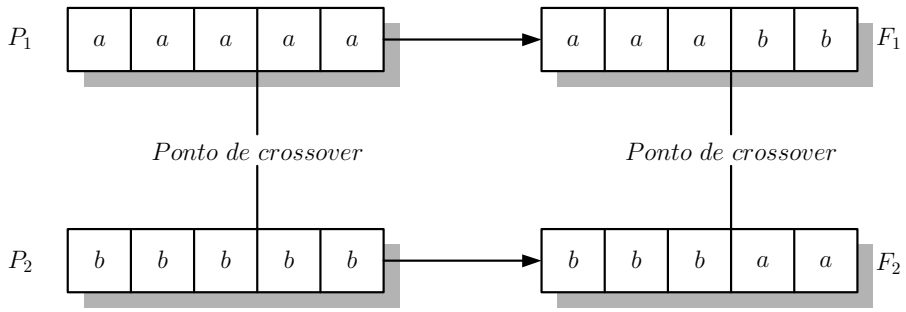

Figura 2.9: Crossover de Um-Ponto.

genética da população, ele altera arbitrariamente um ou mais componentes de uma estrutura escolhida, como é ilustrado na Fig.2.10, fornecendo meios para introdução de novos elementos na população. Desta forma, a mutação assegura que a probabilidade de se chegar a qualquer ponto do espaço de busca nunca seja zero, além de contornar o problema de mínimos locais, pois com este mecanismo altera-se "levemente" a posição no espaço de busca. O operador de mutação é aplicado aos indivíduos com uma probabilidade dada pela taxa de mutação p_m; geralmente utiliza-se uma taxa de mutação pequena, pois é um operador genético normalmente considerado secundário.

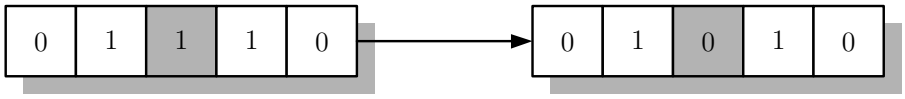

Figura 2.10: Mutação.

Nos *AGs*, a operação de mutação é realizada bit a bit nos cromossomos. Para cada cromossomo da população, e para cada bit deste cromossomo, realiza-se as seguintes operações, Michalewicz (1999):

1. Gera-se um número aleatório real r com distribuição uniforme no intervalo $[0, 1]$.
2. Se $r < p_m$ então execute a mutação neste bit, ou seja, mude o seu valor de "0" para "1" ou vice-versa.

Um dos princípios básicos do funcionamento dos AGs é um critério de seleção que vai fazer com que, depois de muitas gerações, o conjunto inicial de indivíduos gere indivíduos mais aptos. A maioria dos métodos de seleção são concebidos para escolher preferencialmente indivíduos com maiores graus de aptidão. Um método de seleção muito utilizado é o *Método da Roleta*, onde indivíduos de uma geração são escolhidos para fazer parte da próxima geração, através de um sorteio tipo roleta. O método da roleta é construído da seguinte maneira:

1. Calcula-se o valor de *fitness* F_i para cada cromossomo $\mathbf{v}_i$ $(i = 1, \cdots, tampop)$. Neste caso assume-se que os valores de *fitness* são positivos.

2. Calcula-se o *fitness* total da população

$$F = \sum_{i=1}^{tampop} F_i. \tag{2.54}$$

3. Calcula-se a probabilidade de seleção p_i para cada cromossomo $\mathbf{v}_i$:

$$p_i = F_i/F. \tag{2.55}$$

4. Calcula-se a probabilidade cumulativa q_i para cada cromossomo $\mathbf{v}_i$:

$$q_i = \sum_{i=1}^{j} p_j. \tag{2.56}$$

A seleção consiste em executar o algoritmo do método da roleta várias vezes até completar uma nova população. Em cada execução, um cromossomo é selecionado para formar a nova população através dos seguintes passos:

1. Gere um número aleatório real, r, com distribuição uniforme no intervalo $[0, 1]$.
2. Se $r < q_1$ então selecione o cromossomo $\mathbf{v}_1$; senão, selecione o i-ésimo cromossomo $\mathbf{v}_i$ $(2 \leq i \leq tampop)$ tal que $q_{i-1} \leq r \leq q_i$.

Neste esquema alguns cromossomos poderão ser selecionados mais de uma vez e outros poderão não ser selecionados. Em média, os melhores indivíduos geram várias réplicas, os indivíduos médios conseguem passar para a próxima geração, e os piores não são selecionados. Porém isto não implica que o pior indivíduo de uma geração não possa ser selecionado. Um exemplo de construção da roleta é apresentado na Fig.2.11.

	Indivíduo $\mathbf{v}_i$	*Fitness* $f(\mathbf{v}_i)$	*Fitness* Relativo
$\mathbf{v}_1$	10110	2,73	0,14
$\mathbf{v}_2$	11000	7,27	0,47
$\mathbf{v}_3$	10101	1,05	0,07
$\mathbf{v}_4$	00110	3,35	0,21
$\mathbf{v}_5$	10001	1,69	0,11

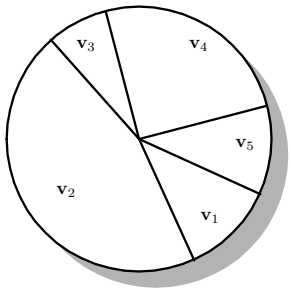

Figura 2.11: Método da Roleta.

2.2.3 Codificação em Ponto Flutuante

A representação binária, tradicionalmente utilizada nos algoritmos genéticos, apresenta algumas desvantagens quando aplicada em problemas multi-dimensionais que exijam alta precisão

numérica. Imagine um problema no qual seja necessário ajustar 200 parâmetros. Utilizando um algoritmo genético com codificação binária e reservando 10 bits para cada parâmetro, cada cromossomo que representa uma solução teria 2000 genes. Efetuar todas as operações de reprodução de um *AG* neste tipo de problema, significa elevar o custo em termos de recursos computacionais. Este problema poderia ser resolvido pela representação de números em ponto flutuante através da associação adequada de números em ponto fixo, criando uma representação que pode ser extremamente eficiente em termos computacionais. O leitor deve notar que existe grande dificuldade para estabelecer de forma genérica, qual o melhor tipo de codificação a utilizar, normalmente para cada aplicação deve-se fazer um estudo prévio sobre qual codificação produzirá melhores resultados em termos computacionais.

A codificação binária facilita a análise teórica dos algoritmos genéticos e permite a utilização de operadores genéticos simples. Porém, o paralelismo implícito dos *AGs* não depende da codificação binária. Conseqüentemente, pode ser interessante utilizar outros alfabetos e operadores genéticos. Neste sentido, por experiência histórica os alfabetos menores tendem a ser mais eficientes.

Como o problema de otimização tratado neste estudo consiste no ajuste paramétrico de variáveis definidas em domínios contínuos, optou-se pela codificação em ponto flutuante com operadores genéticos especiais.

As principais vantagens da codificação em ponto flutuante são: redução da carga computacional na execução do algoritmo (há uma equivalência entre o espaço de representação do *AG* e o espaço de busca do problema de otimização), e posibilidade de uso de operadores genéticos específicos para cada problema abordado.

2.2.4 Algoritmo Genético Proposto

O algoritmo genético desenvolvido baseia-se no mesmo princípio de reprodução utilizado no *AG* clássico. Para a codificação em ponto flutuante, foram definidas duas operações básicas de reprodução: o *crossover* e a mutação. A codificação utilizada permite que estes operadores controlem o modo como o espaço de busca é explorado.

Antes de descrever os operadores genéticos, alguns parâmetros relacionados aos genes dos cromossomos precisam ser definidos. Seja j um gene específico que representa um parâmetro a ser otimizado, então, associado a ele existem três parâmetros a considerar:

min_j: valor mínimo que o gene j pode assumir.

max_j: valor máximo que o gene j pode assumir.

$range_j = max_j - min_j$: largura do intervalo utilizado pelo gene j.

2.2.5 Operadores Genéticos

Os operadores genéticos são muito importantes para o sucesso em uma aplicação específica de um algoritmo genético. Como foi comentado anteriormente, são utilizados dois operadores genéticos no algoritmo genético proposto: o operador *crossover* e o operador mutação.

O Operador *Crossover*

O operador *crossover* recombina os genes dos cromossomos dos pais para formar os cromossomos dos filhos. A taxa de *crossover* p_c determina a probabilidade de um par de cromossomos gerar descendentes. Para o AG com codificação em ponto flutuante foram propostos três tipos diferentes de *crossover*: o *crossover* simples (*crossover* de um-ponto), o *crossover* aritmético e o *crossover* heurístico, Michalewicz (1999).

A operação de *crossover* simples pode ser resumida nos seguintes passos:

1. Seleciona-se dois cromossomos pelo método da roleta

$$P_1 = (0,10; 0,45; 0,32; 0,87; 0,65; \cdots ; 0,23)$$
$$P_2 = (0,14; 0,11; 0,33; 0,56; 0,99; \cdots ; 0,77)$$

2. Gera-se um número aleatório inteiro, *pos*, com distribuição uniforme no intervalo $[1, m-1]$ que indica a posição de corte no cromossomo.

$$pos = 3$$

3. Efetua-se a troca dos genes dos cromossomos.

$$F_1 = (0,10; 0,45; 0,32; 0,56; 0,99; \cdots ; 0,77)$$
$$F_2 = (0,14; 0,11; 0,33; 0,87; 0,65; \cdots ; 0,23)$$

Para operação de *crossover* aritmético procede-se da seguinte maneira:

1. Seleciona-se dois cromossomos pelo método da roleta

$$P_1 = (0,10; 0,45; 0,32; 0,87; 0,65; \cdots ; 0,23)$$
$$P_2 = (0,14; 0,11; 0,33; 0,56; 0,99; \cdots ; 0,77)$$

2. Gera-se um número aleatório real, r, com distribuição uniforme no intervalo $[0, 1]$.

$$r = 0,3$$

3. Efetua-se a soma dos genes dos cromossomos, seguindo-se a seguinte regra:

$$F_1 = rP_1 + (1-r)P_2$$
$$F_2 = rP_2 + (1-r)P_1$$
$$F_1 = (0,128; 0,212; 0,327; 0,653; 0,888; \cdots ; 0,608)$$
$$F_2 = (0,112; 0,348; 0,323; 0,777; 0,752; \cdots ; 0,392)$$

Para operação de *crossover* heurístico procede-se da seguinte maneira:

1. Seleciona-se dois cromossomos pelo método da roleta

$$P_1 = (0,10; 0,45; 0,32; 0,87; 0,65; \cdots ; 0,23)$$
$$P_2 = (0,14; 0,11; 0,33; 0,56; 0,99; \cdots ; 0,77)$$

2. Gera-se um número aleatório inteiro, r, com distribuição uniforme no intervalo $[0,1]$.

$$r = 0,25$$

3. Efetua-se a soma dos genes dos cromossomos, seguindo-se a seguinte regra:

$$F_1 = r(P_1 - P_2) + P_2$$
$$F_1 = (0,130; 0,195; 0,327; 0,637; 0,905; \cdots ; 0.635)$$

O Operador Mutação

O operador mutação atua como uma perturbação em torno do ponto $n-$dimensional representado pelo cromossomo. A taxa de mutação p_m é a probabilidade de um cromossomo ser alterado, ou sofrer mutação genética. A operação de mutação realizada é a mutação indutiva, e para tal procede-se da seguinte forma:

1. Gera-se um número aleatório real q com distribuição uniforme no intervalo $[0,1]$.
2. Se $q < p_m$, então para cada gene escolhido para mutação executa-se o seguinte procedimento:
   - Gera-se um número aleatório $l \in \{0,1\}$

$$gene_j(k+1) = \begin{cases} gene_j(k) + range_j \cdot w \cdot \Delta(k+1) & \text{se } l = 0 \\ gene_j(k) - range_j \cdot w \cdot \Delta(k+1) & \text{se } l = 1 \end{cases}$$
$$\text{Se } gene_j(k+1) > max_j \text{ então } gene_j(k+1) = max_j$$
$$\text{Se } gene_j(k+1) < min_j \text{ então } gene_j(k+1) = min_j$$

   onde $w \in [0;1]$ indica o valor máximo em relação a $range_j$ que pode ser adicionado ao gene j, e $\Delta(n)$ é uma função que retorna um valor aleatório com distribuição uniforme no intervalo $[0;1]$, de forma que a probabilidade de $\Delta(n)$ fique próxima de zero à medida que n cresce.

Esta propriedade implica na realização de buscas uniformes no início do processo de otimização (quando n é pequeno), e em buscas locais no final do processo. A função utilizada para $\Delta(n)$ é dada por:

$$\Delta(n) = (1 - r^{(1-\frac{n}{N_g})^b}) \tag{2.66}$$

onde r é um número aleatório com distribuição uniforme definido em $[0;1]$, N_g é o número máximo de gerações, e b é um parâmetro que determina o grau de dependência da função $\Delta(n)$ em relação ao número de gerações n. Nos experimentos realizados, adotou-se $b = 2,0$. A Fig.2.12 ilustra o comportamento da função $\Delta(n)$ em função de r para diversos valores da razão n/N_g.

Exemplo:

Os genes sublinhados (alelos) são os genes que sofrerão mutação.

cromossomo:$(0,10;\underline{0,45};0,32;\underline{0,87};0,65;\cdots;\underline{0,23})$

Após a mutação:

cromossomo:$(0,10;\underline{0,47};0,32;\underline{0,88};0,65;\cdots;\underline{0,20})$

O parâmetro w determina o tamanho da região na qual a perturbação ocorre. Nos experimentos realizados utilizou-se $w = 0,25$, significando uma alteração máxima de $\pm 25\%$ em relação ao seu valor atual.

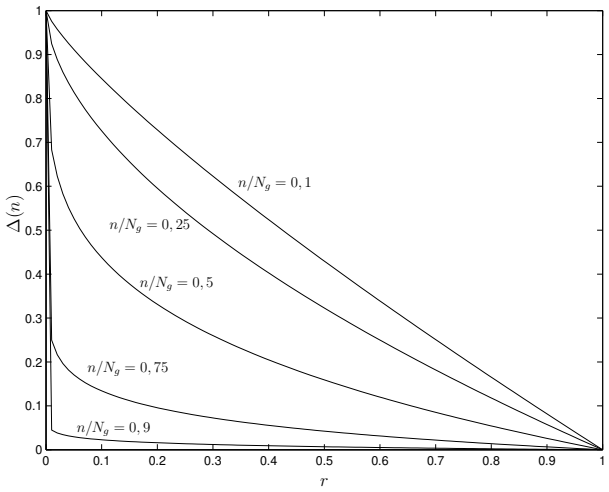

Figura 2.12: Função $\Delta(n)$ Empregada Pelo Operador de Mutação.

2.2.6 Reprodução e Seleção

O processo de reprodução explora o espaço de soluções através de operadores genéticos. No *AG* proposto, os processos de reprodução e seleção utilizam 3 sub-populações (sp) geradas a partir da população principal (Pop), Fig.2.13.

Estas sub-populações são formadas da seguinte maneira:

- a sub-população sp1 possui 50% do tamanho da população principal, contendo seus melhores

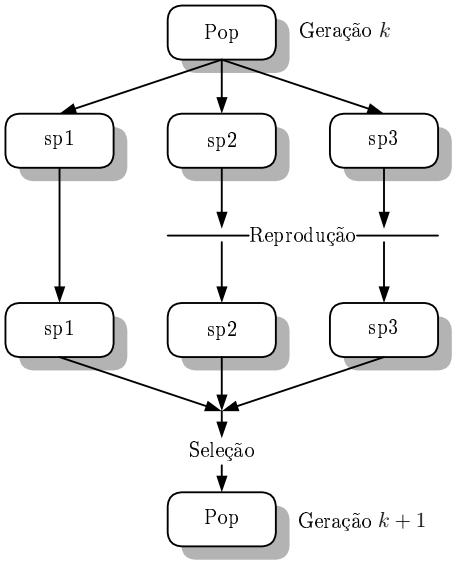

Figura 2.13: Processos de Reprodução e Seleção.

indivíduos.

- a sub-população sp2 possui 80% do tamanho da população principal, e é formada aplicando-se o algoritmo do método da roleta na população principal;
- a sub-população sp3 possui 20% do tamanho da população principal, e é formada por mutantes do melhor indivíduo da população principal;

No processo de reprodução participam apenas as subpopulações sp2 e sp3.

Após a reprodução, o *fitness* de cada cromossomo é calculado através de uma avaliação do controlador equivalente, quando aplicado na tarefa de controle em questão. A avaliação do controlador pode ser feita através de simulação, ou pela execução da tarefa num sistema de controle em tempo real.

O processo de seleção consiste na escolha dos N_m melhores cromossomos das sub-populações (seleção elitista). O processo completo de busca utilizado no algoritmo genético proposto está mostrado no fluxograma da Fig.2.14.

2.2.7 Função de *Fitness*

Geralmente, a aptidão do indivíduo é determinada através do cálculo da função-objetivo, que depende de especificações de projeto. A função de *fitness* é uma expressão numérica que mede quantitativamente o nível de adaptação de um indivíduo da população. Quanto maior o valor do *fitness*, mais adaptado é considerado o indivíduo. Neste contexto, em geral, um indivíduo

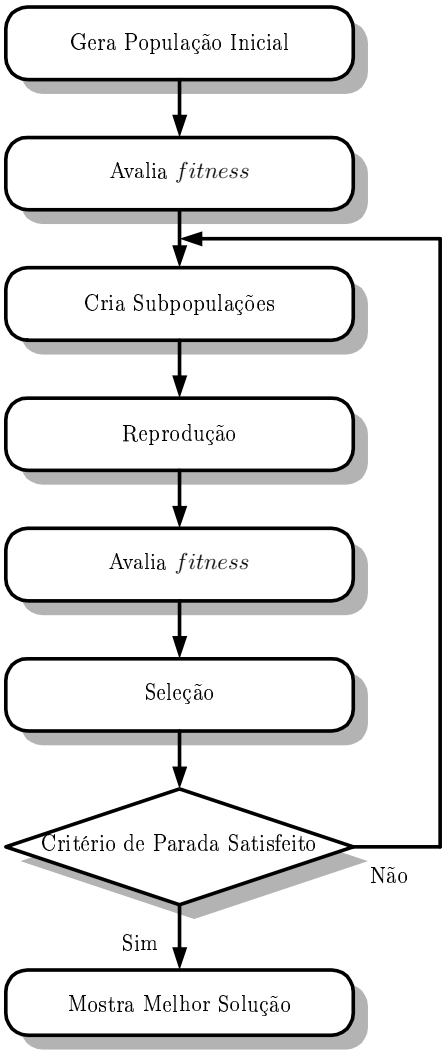

Figura 2.14: Fluxograma do AG proposto.

representa uma solução candidata do problema, portanto, a função de *fitness* pode ser entendida como a função-objetivo de um problema de maximização.

Neste trabalho, os *AGs* são utilizados para otimização de controladores neurais. Conseqüentemente, a função de *fitness* deve incluir medidas de desempenho destes controladores quando aplicados às tarefas de controle.

Em problemas de controle, várias medidas podem ser consideradas na avaliação de desempenho, por exemplo: consumo de energia, tempo de subida, tempo de estabilização, sobre-sinal máximo, erro de regime, entre outras. Para os problemas de controle abordados neste trabalho utiliza-se, uma medida baseada no espectro de potência, que vai permitir avaliar tal desempenho.

Seja $f(x)$ uma função contínua e periódica de uma variável x. A transformada de *Fourier* de $f(x)$, denotada por $\mathcal{F}\{f(x)\}$, é definida pela seguinte equação

$$\mathcal{F}\{f(x)\} = F(u) = \int_{-\infty}^{\infty} f(x) \exp(-j2\pi ux) dx \tag{2.67}$$

onde $j = \sqrt{-1}$.

A transformada de *Fourier* de uma função real geralmente é complexa, isto é,

$$F(u) = R(u) + jI(u) \tag{2.68}$$

onde $R(u)$ e $I(u)$ são os componentes real e imaginário de $F(u)$, respectivamente. Expressando a Eq.2.68 na forma exponencial temos

$$F(u) = \mid F(u) \mid e^{j\theta(u)} \tag{2.69}$$

onde

$$\mid F(u) \mid = [R^2(u) + I^2(u)]^{\frac{1}{2}} \tag{2.70}$$

e

$$\theta(u) = \tan^{-1}\left[\frac{I(u)}{R(u)}\right]. \tag{2.71}$$

A função $\mid F(u) \mid$ é chamada *Espectro de Fourier* de $f(x)$, e $\theta(u)$ é o *ângulo de fase*. O quadrado do espectro

$$\begin{aligned} P(u) &= \mid F(u) \mid^2 \\ &= R(u)^2 + I(u)^2 \end{aligned} \tag{2.72}$$

é comumente denominado *Espectro de Potência* de $f(x)$, Gonzalez e Woods (2000).

Assim, de um determinado sinal de referência, $f_{ref}(x)$, e de um sinal de resposta do sistema controlado por um controlador neural, $f_{resp}(x)$, podemos calcular o espectro de potência do sinal de erro $P(e(x))$, onde

$$e(x) = f_{ref}(x) - f_{resp}(x). \tag{2.73}$$

Deste modo, quanto menor o valor do espectro de potência do sinal de erro, temos um sinal de resposta mais próximo da referência, e usando como função de *fitness*

$$f(x) = \frac{1}{1 + \sum_{i=0}^{100} P_i(e(x))} \tag{2.74}$$

onde P_i representa a i-ésima harmônica, temos uma função que captura as características mais importantes da resposta do sistema frente à execução de uma determinada tarefa.

O método utilizado para a definição da função de *fitness* é muito eficiente do ponto de vista de sistemas de controle, pois, além de ser processado com relativa facilidade, tem o poder de analisar respostas aos mais variados estímulos podendo estes ser lineares ou não-lineares, contínuos ou não.

A função de *fitness*, baseada no espectro de potência, possui o grande atrativo de permitir fazer-se diferentes análises de uma determinada resposta, sem que para isso seja necessário o acréscimo de novas funções matemáticas, diferentes ponderações na soma dos espectros levam a diferentes funções de *fitness*. Esta função permite ainda a avaliação precisa de freqüências específicas, podendo ser útil para a avaliação de características particulares do sistema, permitindo a eliminação de freqüências espúrias e/ou freqüências ressonantes.

2.3 Sumário

Neste capítulo foram apresentados conceitos importantes para a compreensão dos capítulos seguintes. Na seção 2.1 foram apresentadas definições importantes concernentes à teoria de Redes Neurais Artificiais. Na seção 2.2 foram apresentadas as principais definições concernentes à teoria de Algoritmos Genéticos.

No próximo Capítulo é apresentada a topologia de controlador neural de *Kim-Lewis-Dawson*, que incorpora técnicas de controle ótimo linear associadas ao método de aprendizado da rede neural. O próximo Capítulo apresenta, ainda, a aplicação de Algoritmos Genéticos para a obtenção de parâmetros otimizados para o controlador neural.

Capítulo 3

Controlador Neural de

3.1 Introdução

Várias topologias de controladores foram desenvolvidos para o controle de movimentos de robôs. De um modo geral, estas propostas combinam linearização por realimentação de estados com técnicas de controle ótimo, Johansson (1990), Lin e Brandt (1996), Khoukhi (1999). Porém, em situações reais, a dinâmica dos robôs raramente é completamente conhecida, o que dificulta enormemente expressar suas dinâmicas precisas através de expressões matemáticas. Frente a este problema foram propostos controladores neurais para uso específico no controle da dinâmica não linear de sistemas manipuladores robóticos. Lewis, Liu e Yesildirek (1995), Narendra (1997), Mital e Chin (1998), Bogdanov e Timofeev (1999), Kim e Lewis (1999), Kim, Lewis e Dawson (2000), são alguns exemplos que podem ser citados.

Recentemente Kim et al. (2000) propuseram uma topologia de controlador neural que incorpora técnicas de controle ótimo linear associadas ao método de aprendizado da rede neural. As técnicas de controle ótimo linear conferem robustez ao controlador frente a uma certa variação nos parâmetros do modelo (incertezas), enquanto que a rede neural é usada para estimar adaptativamente estas variações. A descrição do controlador neural proposto por Kim et al. (2000) é apresentada a seguir.

3.2 Rede Neural para Aproximação de Funções Não-Lineares

A Fig.3.1 apresenta uma rede neural de duas camadas, conhecida como rede FLNN *"Functional Link Neural Network"*, onde apenas os pesos da segunda camada são ajustados via

treinamento, e os pesos da primeira são fixos, Lewis et al. (1999).

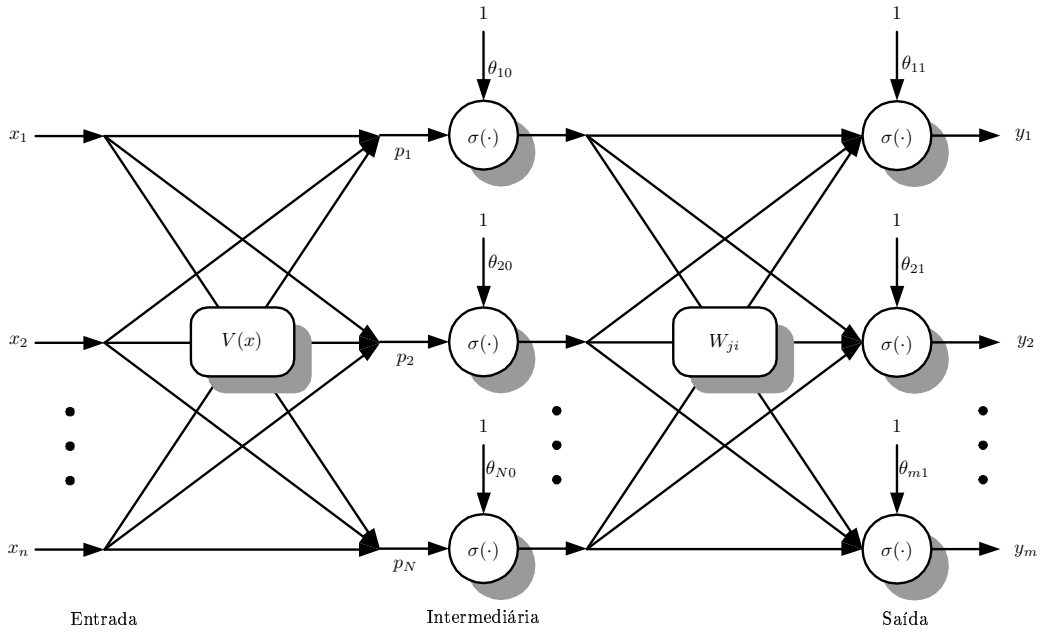

Figura 3.1: Rede Neural FLNN *"Functional Link Neural Network"*.

Uma rede FLNN pode ser usada para aproximar um mapa não linear $z(x) : X^n \to Y^m$, e o modelo matemático para esta rede é dado por

$$y_j(x) = \sum_{i=1}^{N} \left(w_{ji}\sigma_i\left(p_i + \theta_{i0}\right) + \theta_{j1}\right) + \varepsilon_j(x); \qquad j = 1, 2, \ldots, m. \tag{3.1}$$

onde $w_{ji} \in \mathbb{R}$ são os pesos da camada de saída, $\theta_{i0}, \theta_{j1} \in \mathbb{R}$ são os valores dos limiares, $p_i \in \mathbb{R}$ são as entradas dos neurônios da primeira camada. $\sigma(.) : \mathbb{R} \to \mathbb{R}$ é a função de ativação, N é o número dos neurônios e $\varepsilon_j(x)$ é o erro de aproximação.

Em geral, mesmo quando os melhores valores dos pesos são alcançados, a função não linear não é aproximada exatamente pela rede, restando um erro de aproximação $\varepsilon(x) \in \mathbb{R}^m$, Haikin (1994), Lewis et al. (1999).

Assumindo que esse erro possui um limitante superior

$$\|\varepsilon(x)\| \leq \varepsilon_M \qquad \forall x \in X^n. \tag{3.2}$$

e que os pesos, para a aproximação, existem para um dado valor de ε_M, o modelo da rede neural pode ser escrito, na forma matricial, como

$$y = W^T\sigma(p) + \varepsilon(x), \tag{3.3}$$

com $W \in \mathbb{R}^{N+1\times m}$, $\sigma(p) \in \mathbb{R}^{N+1}$, $y(x) \in \mathbb{R}^m$ e $\varepsilon(x) \in \mathbb{R}^m$. As entradas $p \in \mathbb{R}^N$ dos neurônios são definidas através de um pré-processamento $V(x)$ do vetor de entrada da rede $x \in \mathbb{R}^n$, Lewis et al. (1995).

Assim, uma estimativa $\hat{y}(x)$ de $y(x)$ pode ser escrita como

$$\hat{y} = \hat{W}^T \sigma(p) \qquad onde\ p = V(x) \qquad \forall x \in X^n, \tag{3.4}$$

onde $\hat{W}$ são estimativas dos valores consideradas ideais para os pesos W, ajustados através de algum algoritmo de treinamento eficiente, que se faz necessário devido à possibilidade de aproximação de funções não-lineares.

Embora existam várias outras arquiteturas de RNA's, Narendra e Parthasarathy (1990), Lewis et al. (1995), Brown, Ruchti e Feng (1993), Narendra (1997), Commuri e Lewis (1997), Kwan, Lewis e Dawson (1998), Bogdanov e Timofeev (1999), Cerqueira, Badan e Madrid (2000), Thapa et al. (2000), a rede FLNN apresentada acima é utilizada no restante deste estudo, por representar a rede utilizada no desenvolvimento do controlador proposto por Kim et al. (2000) que é o foco desta dissertação.

3.3 Dinâmica de Manipuladores Robóticos e suas Propriedades

O modelo dinâmico de um manipulador proporciona uma descrição matemática da relação entre os torques atuantes nas juntas e o movimento da estrutura. Pela formulação de *Lagrange*, as equações do movimento podem ser obtidas de um modo sistemático e independente do sistema de coordenadas.

O *Lagrangeano* de um sistema mecânico pode ser definido como:

$$L = T - U \tag{3.5}$$

onde T representa a energia cinética e U representa a energia potencial do sistema. As equações de *Lagrange* são expressas por:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i} = \xi_i \tag{3.6}$$

onde q_i representa a variável de coordenada e ξ_i representa a força associada a essa coordenada, Simon (1971).

Desse modo o modelo dinâmico de um manipulador robótico serial com n graus de liberdade, pode ser expresso na forma de *Lagrange* como, Lewis et al. (1999), Sciavicco e Siciliano (1996):

$$M(q)\ddot{q} + V_m(q,\dot{q})\dot{q} + F_v\dot{q} + f_c(\dot{q}) + g(q) + \tau_d(t) = \tau(t) \tag{3.7}$$

sendo $q(t) \in \mathbb{R}^n$ correspondente às variáveis generalizadas da junta, $M(q) \in \mathbb{R}^{n\times n}$ à matriz de inércia, $V_m(q,\dot{q}) \in \mathbb{R}^{n\times n}$ à matriz das forças de *Coriolis*/centrípeta, $g(q) \in \mathbb{R}^n$ às forcas gravitacionais, $F_v \in \mathbb{R}^{n\times n}$ matriz diagonal correspondente aos coeficientes de atrito viscoso, $f_c(\dot{q}) \in \mathbb{R}^n$ aos coeficientes de atrito seco, e $\tau_d(t) \in \mathbb{R}^n$ correspondente às perturbações externas. O torque aplicado às juntas é $\tau(t) \in \mathbb{R}^n$, Fu, Gonzalez e Lee (1987), Lewis et al. (1999).

Dada uma trajetória desejada $q_d(t) \in \mathbb{R}^n$, o erro de posição é definido como

$$e(t) = q_d(t) - q(t) \tag{3.8}$$

e a medida de desempenho instantâneo do sistema é definida por

$$r(t) = \dot{e}(t) + \Lambda e(t), \tag{3.9}$$

onde Λ é definida como a matriz de ganho crítico.

O modelo dinâmico do robô, Eq.3.7, pode então ser reescrito como

$$M(q)\dot{r}(t) = -V_m(q,\dot{q})r(t) - \tau(t) + h(x) \tag{3.10}$$

onde as funções não-lineares do modelo são agrupadas em

$$h(x) = M(q)(\ddot{q}_d + \Lambda\dot{e}) + V_m(q,\dot{q})(\dot{q}_d + \Lambda e) + g(q) + F_v\dot{q} + f_c(\dot{q}) + \tau_d(t) \tag{3.11}$$

e

$$x = [e^T, \dot{e}^T, q_d^T, \dot{q}_d^T, \ddot{q}_d^T]^T, \tag{3.12}$$

. Desse modo, temos em $h(x)$ uma função que captura toda a dinâmica não linear do modelo do manipulador robótico.

Define-se a lei de controle como

$$\tau(t) = h(x) - u(t), \tag{3.13}$$

com $u(x) \in \mathbb{R}^n$ sendo uma entrada de controle auxiliar a ser definida posteriormente. Temos então o sistema em malha fechada dado pela equação

$$M(q)\dot{r}(t) = -V_m(q,\dot{q})r(t) + u(t) \tag{3.14}$$

A dinâmica do sistema robótico ainda deve possuir as seguintes propriedades, Lewis et al. (1995):

* **Propriedade 1:** $M(q) = M(q)^T$ é uma matriz definida positiva limitada por

$$m_1 I \leq M(q) \leq m_2 I \qquad m1, m2 > 0; \qquad I \in \mathbb{R}^{n\times n}. \tag{3.15}$$

* **Propriedade 2:** A matriz $N(q,\dot{q}) = M(\dot{q}) - 2V_m(q,\dot{q})$ é tal que, $x^T N(q,\dot{q})x = 0$, para qualquer vetor x.

* **Propriedade 3:** As perturbações externas satisfazem $\|\tau_d\| < b_d$, onde b_d é uma constante conhecida e $b_d > 0$.

3.4 Controlador Ótimo de Torque Calculado (COTC)

3.4.1 Otimização Hamilton-Jacobi-Bellman (H-J-B)

Definindo a equação dinâmica do erro como

$$\dot{e}(t) = -\Lambda e(t) + r(t) \tag{3.16}$$

o seguinte sistema de equações de estado pode ser escrito

$$\dot{\tilde{z}} = \begin{bmatrix} \dot{e} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} -\Lambda & I \\ 0 & -M^{-1}V_m \end{bmatrix} \begin{bmatrix} e \\ r \end{bmatrix} + \begin{bmatrix} 0 \\ M^{-1} \end{bmatrix} u(t) \tag{3.17}$$

ou em notação abreviada

$$\dot{\tilde{z}} = A(q, \dot{q})\tilde{z} + B(q)u(t) \tag{3.18}$$

com $A(q, \dot{q}) \in \mathbb{R}^{2n \times 2n}$, $B(q) \in \mathbb{R}^{2n \times n}$ e $\tilde{z}(t) \in \mathbb{R}^{2n \times 1}$. Kim et al. (2000) definem como função de custo a seguinte função quadrática

$$J(u) = \int_{t_o}^{\infty} L(\tilde{z}, u)dt \tag{3.19}$$

com Lagrangeano

$$\begin{aligned} L(\tilde{z}, u) &= \frac{1}{2}\tilde{z}^T Q\tilde{z} + \frac{1}{2}u^T Ru \\ &= \frac{1}{2}\begin{bmatrix} e^T & r^T \end{bmatrix} \begin{bmatrix} Q_{11} & Q_{12} \\ Q_{12}^T & Q_{22} \end{bmatrix} \begin{bmatrix} e \\ r \end{bmatrix} + \frac{1}{2}u^T Ru \end{aligned} \tag{3.20}$$

Dada a função custo $J(u)$, 3.19 , o objetivo é encontrar a lei de controle $u(t)$, que minimiza $J(u)$ sujeita às restrições impostas pela equação diferencial Eq.3.17. A lei de controle que atinge este objetivo será denotada por $u^*(t)$. Note que apenas a porção $u(t)$ de Eq.3.13 é otimizada, pois os termos associados à gravidade, *Coriolis* e atrito não podem ser otimizados, por serem ditados pelas características físicas do sistema.

A condição necessária e suficiente para que $u^*(t)$ minimize $J(u)$ sujeito à Eq.3.17, segundo Lewis e Syrmos (1995), é que exista uma função $V(\tilde{z}, t)$ que satisfaça a equação H-J-B

$$\frac{\partial V(\tilde{z}, t)}{\partial t} + min_u \left[H\left(\tilde{z}, u, \frac{\partial V(\tilde{z}, t)}{\partial \tilde{z}}, t\right)\right] = 0 \tag{3.21}$$

com Hamiltoniano definido por

$$\left[H\left(\tilde{z}, u, \frac{\partial V(\tilde{z}, t)}{\partial \tilde{z}}, t\right)\right] = L(\tilde{z}, u) + \frac{\partial V(\tilde{z}, t)}{\partial \tilde{z}}\dot{\tilde{z}}. \tag{3.22}$$

A função $V(\tilde{z}, t)$, composta por $\tilde{z}$, $M(q)$, e port uma matriz simétrica e definida positiva $K = K^T \in \mathbb{R}^{n \times n}$, satisfaz a equação H-J-B

$$V(\tilde{z}, t) = \frac{1}{2}\tilde{z}^T P(q)\tilde{z} = \frac{1}{2}\tilde{z}^T \begin{bmatrix} K & 0 \\ 0 & M(q) \end{bmatrix} \tilde{z}, \tag{3.23}$$

onde Λ e K nas Eq.3.9 e 3.23 são obtidos através da solução da equação diferencial de *Riccati*, Fu et al. (1987)

$$PA + A^T P^T - PBR^{-1}B^T P + \dot{P} + Q = 0. \tag{3.24}$$

Assim

$$H\left(\tilde{z}, u, \frac{\partial V(\tilde{z}, t)}{\partial \tilde{z}}, t\right) = \frac{1}{2}\tilde{z}^T Q\tilde{z} + \frac{1}{2}u^T Ru + \tilde{z}^T P(q)\left[A(q, \dot{q})\tilde{z} + B(q)u(t)\right] \tag{3.25}$$

Derivando a função H em relação a u e igualando a zero obtém-se

$$\frac{\partial H}{\partial u} = Ru + P(q)B(q)^T P(q)\tilde{z} = 0. \tag{3.26}$$

Assim

$$\begin{aligned} u^*(t) &= -R^{-1}B^T(q)P(q)\tilde{z}. \\ &= -R^{-1}r(t). \end{aligned} \tag{3.27}$$

Como

$$\frac{\partial^2 H}{\partial u^2} = R > 0 \tag{3.28}$$

verifica-se que $u^*(t)$ é um ponto de mínimo de H.

A condição necessária e suficiente para a otimalidade é escolher uma função $V(\tilde{z},t)$ que satisfaça a Eq.3.21. Substituindo a Eq.3.22 na Eq.3.21 temos

$$\frac{\partial V(\tilde{z},t)}{\partial t} + \frac{\partial V(\tilde{z},t)}{\partial \tilde{z}}\dot{\tilde{z}} + L(\tilde{z},u^*) = 0 \tag{3.29}$$

como

$$\frac{dV(\tilde{z},t)}{dt} = \frac{\partial V(\tilde{z},t)}{\partial t} + \frac{\partial V(\tilde{z},t)}{\partial \tilde{z}}\dot{\tilde{z}} \tag{3.30}$$

então

$$\tilde{z}^T P(q)\dot{\tilde{z}} + \frac{1}{2}\tilde{z}^T\dot{P}(q)\tilde{z} + L(\tilde{z},u^*) = 0. \tag{3.31}$$

Inserindo Eq.3.17 e Eq.3.20 na Eq.3.31

$$\tilde{z}^T P(q)A(q)\tilde{z} + \frac{1}{2}\tilde{z}^T\left[\dot{P}(q) + Q - P(q)B(q)R^{-1}B(q)P(q)\right]\tilde{z} = 0 \tag{3.32}$$

e sendo

$$\tilde{z}^T P(q)A(q)\tilde{z} = \frac{1}{2}\tilde{z}^T\left[A^T(q)P(q) + P(q)A(q)\right]\tilde{z} \tag{3.33}$$

pode-se escrever

$$\frac{1}{2}\tilde{z}^T\left[\dot{P}(q) + A^T(q)P(q) + P(q)A(q) + Q - P(q)B(q)R^{-1}B(q)P(q)\right]\tilde{z} = 0 \tag{3.34}$$

provando que a Eq.3.23 satisfaz a equação H-J-B.

Substituindo as matrizes $A(q)$, $B(q)$ e $P(q)$ da Eq.3.34 por suas respectivas expressões matemáticas obtém-se

$$\begin{bmatrix} -K\Lambda & 0 \\ K & -V_m \end{bmatrix} + \begin{bmatrix} -\Lambda^T K & 0 \\ K & -V_m \end{bmatrix} - \begin{bmatrix} 0 & 0 \\ 0 & R^{-1} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & \dot{M} \end{bmatrix} = -\begin{bmatrix} Q_{11} & Q_{12} \\ Q_{12}^T & Q_{22} \end{bmatrix} \tag{3.35}$$

Pela a **Propriedade 2** dos manipuladores robóticos

$$\begin{bmatrix} -K\Lambda & 0 \\ K & 0 \end{bmatrix} + \begin{bmatrix} -\Lambda^T K & 0 \\ K & 0 \end{bmatrix} - \begin{bmatrix} 0 & 0 \\ 0 & R^{-1} \end{bmatrix} = -\begin{bmatrix} Q_{11} & Q_{12} \\ Q_{12}^T & Q_{22} \end{bmatrix} \tag{3.36}$$

de onde pode-se derivar as seguintes relações

$$Q = \begin{bmatrix} Q_{11} & Q_{12} \\ Q_{12}^T & Q_{22} \end{bmatrix} > 0, \qquad R^{-1} = Q_{22} \tag{3.37}$$

com $Q_{12} + Q_{12}^T < 0$.

$$K = K^T = -\frac{1}{2}\left(Q_{12} + Q_{12}^T\right) > 0, \tag{3.38}$$

e

$$\Lambda^T K + K\Lambda = Q_{11} \tag{3.39}$$

Com a lei de controle ótimo $u^*(t)$ obtida acima, os torques $\tau(t)$ aplicados às juntas do sistema robótico são calculados de acordo com a seguinte equação

$$\tau^*(t) = h(x) - u^*(t) \tag{3.40}$$

3.4.2 Análise de Estabilidade

A função $V(\tilde{z}, t)$ é uma função contínua, possui um único ponto de mínimo, e é crescente à medida que $\|\tilde{z}\|$ cresce. Devido a estas características, $V(\tilde{z}, t)$ pode ser usada como função candidata de *Lyapunov*.

E necessário mostrar que $dV/dt < 0$ para todo $\|\tilde{z}\| \neq 0$. Da solução da equação H-J-B, Eq.3.29, temos

$$\frac{dV(\tilde{z}, t)}{dt} = -L(\tilde{z}, u^*), \tag{3.41}$$

substituindo a Eq.3.27 na Eq.3.20 obtém-se

$$\begin{aligned} \frac{dV(\tilde{z}, t)}{dt} &= -\frac{1}{2}\left[\tilde{z}^T Q \tilde{z} + (B^T(q)P(q)\tilde{z})^T R^{-1}(B^T(q)P(q)\tilde{z})\right] < 0 \\ &\quad \forall t > 0, \quad \tilde{z} \neq 0 \end{aligned} \tag{3.42}$$

A derivada da função candidata de *Lyapunov*, $V(\tilde{z}, t)$, é definida negativa em relação ao tempo, logo, o sistema definido pela Eq.3.17 é estável no sentido de *Lyapunov*.

3.4.3 Controlador Neural

O diagrama de blocos da Fig.3.2 mostra os principais componentes do controlador neural de *Kim-Lewis-Dawson*.

A função não linear da dinâmica do manipulador robótico, Eq.3.11, é representada por

$$h(x) = W^T \sigma(p) + \varepsilon(x), \qquad \|\varepsilon(x)\| \leq \varepsilon_M \tag{3.43}$$

com vetor x dado pela Eq.3.12.

Uma estimativa $\hat{y}(x)$ de $y(x)$ pode ser escrita como

$$\hat{y} = \hat{W}^T \sigma(p), \tag{3.44}$$

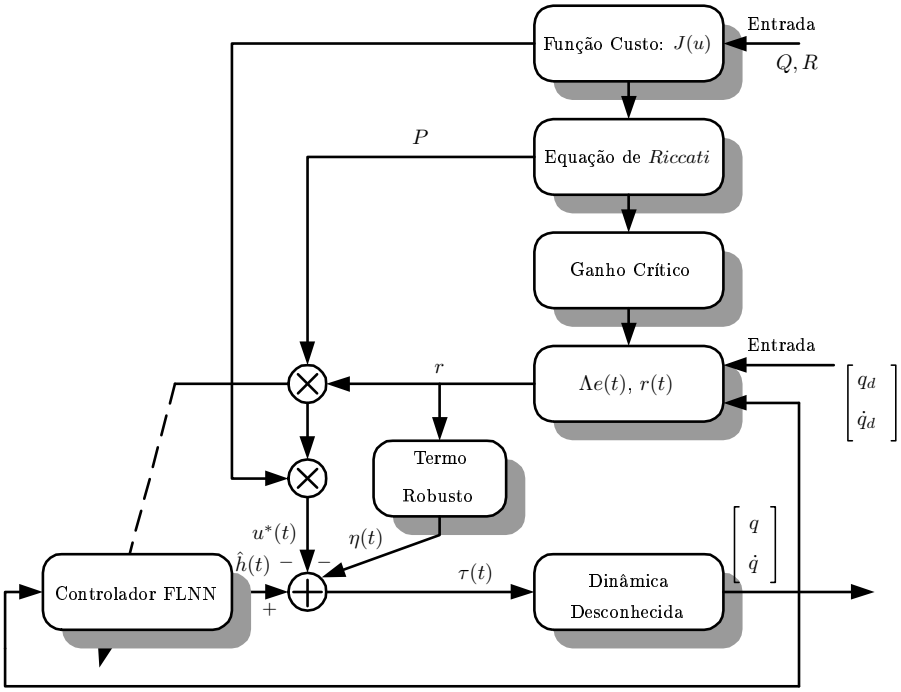

Figura 3.2: Diagrama de Blocos do Controlador Neural Ótimo.

e o vetor de torques externos de equilíbrio dinâmico do manipulador passa a ser descrito por

$$\tau(t) = \hat{W}^T \sigma(p) - u^*(t) - \eta(t), \tag{3.45}$$

onde $\eta(t)$ é um vetor que confere robustez ao sistema.

Então, a Eq.3.10 pode ser escrita como,

$$M(q)\dot{r}(t) = -V_m(q,\dot{q})r(t) + \tilde{W}^T\sigma(p) + \varepsilon(x) + u^*(t) + \eta(t) + \tau_d(t), \tag{3.46}$$

onde $\tilde{W}$ é o erro de estimativa dos pesos da rede, $\tilde{W} = W - \hat{W}$.

A equação de estados, definida pela Eq.3.17, pode então ser escrita da seguinte forma:

$$\dot{\tilde{z}} = A\tilde{z} + B[\tilde{W}^T\sigma(p) + \varepsilon(x) + u^*(t) + \eta(t) + \tau_d(t)]. \tag{3.47}$$

Inserindo a lei de controle ótimo, Eq.3.27, obtém-se

$$\dot{\tilde{z}} = (A - BR^{-1}B^TP)\tilde{z} + B[\tilde{W}^T\sigma(p) + \varepsilon(x) + \eta(t) + \tau_d(t)] \tag{3.48}$$

Segundo Kim et al. (2000), utilizando-se como termo robusto

$$\eta(t) = -k_z \frac{r(t)}{\|r(t)\|}, \tag{3.49}$$

com $k_z \geq b_d$, e uma função de sintonia dos pesos da rede dada por

$$\dot{\hat{W}} = F\sigma(p)r^T - \kappa\|\tilde{z}\|\hat{W}, \tag{3.50}$$

com $F = F^T > 0$ e $\kappa > 0$, fica garantido que $e(t)$, $r(t)$ e $\tilde{W}$ sejam limitados. Assim o sistema dado pela Eq.3.48 também é estável no sentido de *Lyapunov*.

3.5 Resultados Experimentais

Esta seção apresenta os principais resultados obtidos na simulação e implementação do controlador estudado. O problema abordado consiste no controle de trajetória de uma junta robótica simples, que pode ser vista como um pêndulo simples acionado. Os detalhes técnicos sobre o *hardware* de controle e o sistema mecânico utilizado encontram-se no **Apêndice A**.

Os algoritmos desenvolvidos para as simulações foram escritos em linguagem *MatLab*, e os algoritmos desenvolvidos para as implementações foram escritos em linguagem *C*. O período de amostragem utilizado, tanto na simulação como na implementação, foi fixado em 5 milisegundos. Para a implementação, este tempo foi estabelecido através de uma interrupção de *hardware*.

O sistema foi implementado com as seguintes características principais:

- As matrizes Q e R que compõem a função custo foram definidas com os seguintes valores:

$$Q = \begin{bmatrix} 3 & -1 \\ -1 & 4 \end{bmatrix}, \qquad R^{-1} = 4. \tag{3.51}$$

- O controlador é composto por oito neurônios na camada interna, com oito entradas:

$$p = \begin{bmatrix} 1 & q^T & \dot{q}^T & e^T & \dot{e}^T & r^T & q_d^T & \dot{q}_d^T \end{bmatrix}^T \tag{3.52}$$

- A função de ativação utilizada na camada intermediária foi a função sigmoidal:

$$\sigma(x) = \frac{1}{1 + \exp(-x)} \tag{3.53}$$

- A função de ativação utilizada na camada de saída foi a função linear:

$$\sigma(x) = x \tag{3.54}$$

- Tempo de execução: $20s$
- O sinal de referência usado foi

$$q_d = \pi sen(2\pi f t) \tag{3.55}$$

  com freqüência $f = 0.05Hz$.
- Caso 1: $F = aI$, $a = 50$ e $\kappa = 0.001$

- Caso 2: $F = aI$, $a = 10$ e $\kappa = 0.0001$, onde I representa a matriz identidade com dimensão $m \times m$, sendo m o número de neurônios da camada intermediária da rede.

Podemos observar, prontamente, que as respostas do sistema simulado possuem excelente qualidade. A resposta à simulação nos dois casos propostos possui resultados similares, tanto quantitativamente, os sinais de erro possuem valores relativamente próximos, como qualitativamente, as respostas possuem forma de onda muito semelhantes. Podemos dizer, devido à qualidade das resposta atingidas via simulação, que o controlador de *Kim-Lewis-Dawson* é um controlador que alcança bons resultados no controle do sistema dinâmico em estudo.

A simulação é parte fundamental para a análise e projeto de sistemas de controle. O simulador *Matlab* é um dos mais conhecidos e empregados para este fim, sendo que o conhecimento de seus parâmetros torna-se uma necessidade. Na simulação de sistemas dinâmicos, o parâmetro mais importante a ser considerado é o passo de integração, pois este interfere diretamente na qualidade da solução. Nos estudos realizados, as simulações com utilização de passo variável apresentaram as melhores respostas. No entanto, para aproximar o sistema simulado do sistema implementado, as simulações apresentadas nas Fig.3.3 e Fig.3.5 foram realizadas com passo de integração de $2,5ms$.

Outra característica do sistema simulado, que merece especial atenção, é o fato do sistema de equações que representa a dinâmica do sistema, não englobar o modelo dinâmico do motor utilizado para fornecer o torque de equilíbrio para a junta. Ver modelo do sistema no **Apêndice A**. Isso influencia muito na qualidade das respostas obtidas.

Nas Fig.3.4 e Fig.3.6 são apresentadas as respostas do sistema implementado. Podemos notar que a diferença entre estes sinais e os sinais apresentados pela simulação são bastante grandes. Estas diferenças podem ser explicadas por alguns fatores que compõe o sistema físico real, e não o sistema simulado, como: a presença do motor elétrico, que fornece o torque ao eixo do pêndulo, os atritos presentes neste motor (escovas, rolamentos, etc.), e a presença de atrito seco (*Coulomb*) no sistema mecânico (rolamentos, caixas de redução, correias de transmissão, etc.).

Outro ponto importante a destacar é a grande diferença entre a resposta dos dois sistemas implementados. Nos dois casos apresentados, podemos observar uma grande variação entre os sinais de erro, tanto quantitativa como qualitativa. Os sinais de erro apresentam valores e forma de onda diferentes. Este comportamento deve-se ao fato de ter-se utilizado de diferentes parâmetros de controle para a rede neural de cada controlador.

3.6 Otimização de Parâmetros

Uma rápida análise nas respostas apresentadas na seção anterior demonstra uma característica peculiar das redes neurais e, por conseqüência, dos controladores neurais. Redes neurais apresentam uma grande capacidade de aproximar funções ou classificar padrões, no entanto, são sensíveis aos seus parâmetros de controle, como taxa de aprendizado e termo *momentum*[1].

[1] Haikin (1994) define o termo *momentum* como uma constante, usualmente positiva, que possui o objetivo de aumentar a taxa de aprendizado e, ao mesmo tempo, evitar e instabilidade do algoritmo de aprendizagem.

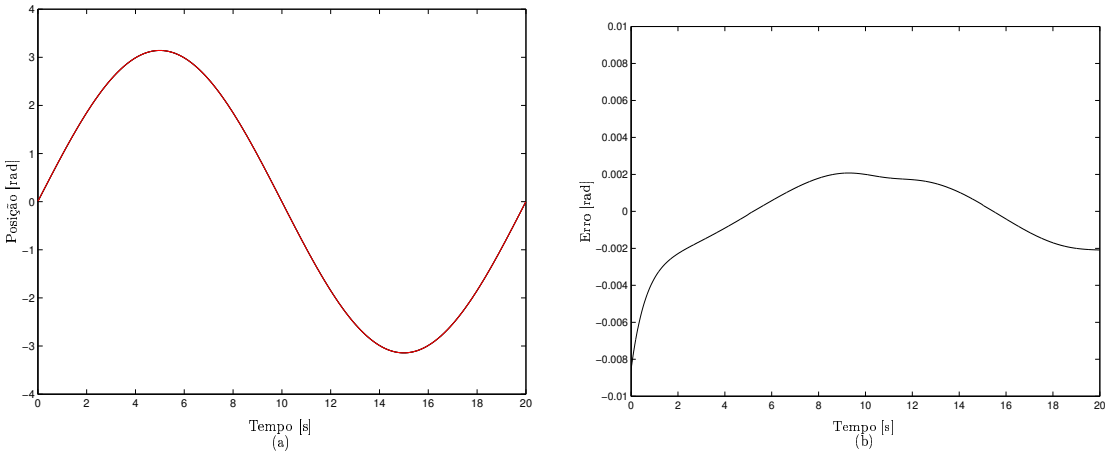

Figura 3.3: Resposta do Sistema Controlado - Simulação Caso 1 - (a) (−)Trajetória de Referência, (−)Trajetória do Sistema. (b) Erro de Posição.

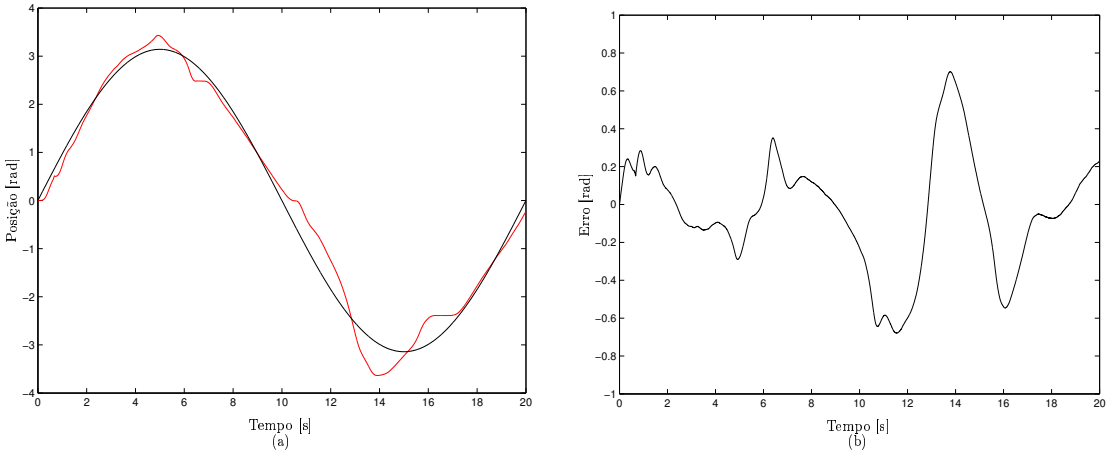

Figura 3.4: Resposta do Sistema Controlado - Implementação Caso 1 - (a) (−)Trajetória de Referência, (−)Trajetória do Sistema. (b) Erro de Posição.

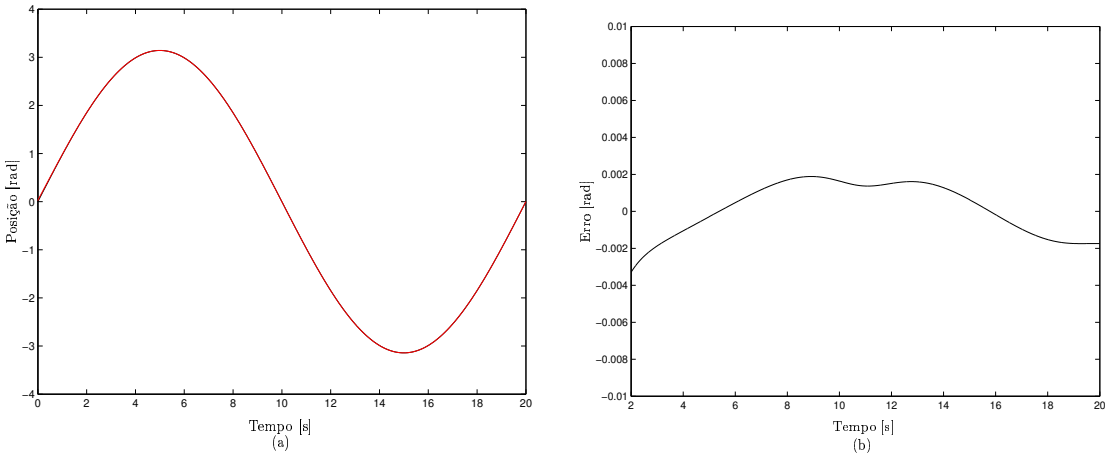

Figura 3.5: Resposta do Sistema Controlado - Simulação Caso 2 - (a) (−)Trajetória de Referência, (−)Trajetória do Sistema. (b) Erro de Posição.

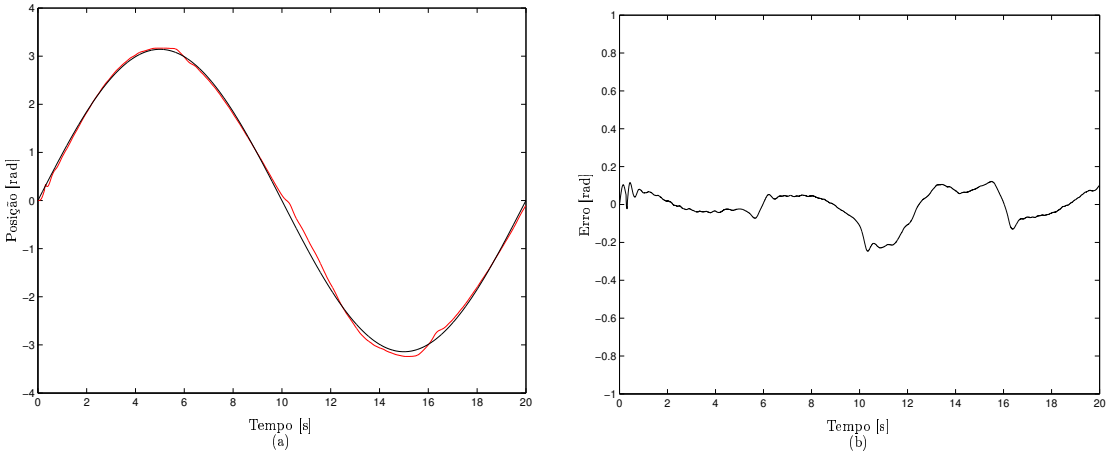

Figura 3.6: Resposta do Sistema Controlado - Implementação Caso 2 - (a) (−)Trajetória de Referência, (−)Trajetória do Sistema. (b) Erro de Posição.

A escolha adequada da taxa de aprendizado e do termo *momentum* são de extrema importância para que se alcance o melhor valor de sintonia para os pesos da rede neural. Geralmente, os valores de parâmetros são escolhidos por tentativa e erro ou pela experiência do projetista. McCullagh et al. (1994) usaram a abordagem de otimização genética para determinar os parâmetros ótimos de controle para uma rede neural MLP, e a ordem em que os padrões de treinamento são apresentados à rede neural. Belew et al. (1990) usaram AGs para a otimização dos valores da taxa de aprendizado e termo *momentum*, apresentando resultados muito satisfatórios.

Como o controlador neural de *Kim-Lewis-Dawson*, descrito anteriormente, apresenta grande sensibilidade às variações da taxa de aprendizado e termo *momentum*, e como, do ponto de vista de controle, procura-se sempre por controladores que levam o sistema dinâmico em questão ao menor erro possível em relação a uma determinada referência, propomos uma estratégia de otimização genética para a escolha destes parâmetros, tendo por base os trabalhos realizados por Belew et al. (1990), McCullagh et al. (1994), Choi e Bluff (1995).

O algoritmo genético proposto no **Capítulo 2** foi utilizado para a otimização dos parâmetros de controle da rede neural. Como cada indivíduo da população representa um controlador neural, para a avaliação de um indivíduo da população é necessária uma execução da tarefa de controle no sistema do elo acionado descrito no **Apêndice A**.

Os parâmetros do AG utilizados nos experimentos encontram-se descritos na Tabela 3.1, onde I representa a matriz identidade. A matriz F possui dimensão $m \times m$, sendo m o número de neurônios da camada intermediária da rede. No caso $m = 8$.

Parâmetros	**Valor**
Tamanho da População	30
Número de Gerações	50
Taxa de Crossover	53%
Taxa de Mutação	5%.
Taxa Aprendizado $F = aI$	$a \subset [0, 50]$
Termo $Momentum$	$\kappa \subset [0, 1]$.
b	2,0
w	0,25

Tabela 3.1: Parâmetros do Algoritmo Genético.

A Fig.3.7 mostra as respostas dos melhores controladores durante o processo de sintonia, utilizando-se a referência dada na Eq.3.55. Na primeira geração(−), observa-se que a resposta de posição possui um alto valor de erro. Após 5 gerações(−), o erro diminuiu. Finalmente, após 50 gerações(−), o processo de sintonia já obteve uma boa solução. O melhor controlador apresenta uma resposta que pode ser considerada rápida, com erro praticamente nulo, Fig.3.8.

No gráfico que ilustra o comportamento do termo $\sum V^2(k)$, nota-se que o gasto energético do melhor controlador (−) é bastante inferior ao gasto do melhor controlador da primeira geração (−). A Fig.3.9 ilustra a evolução do *fitness* do melhor indivíduo através das gerações, e os parâmetros otimizados do melhor controlador obtido encontram-se descritos na Tabela 3.2.

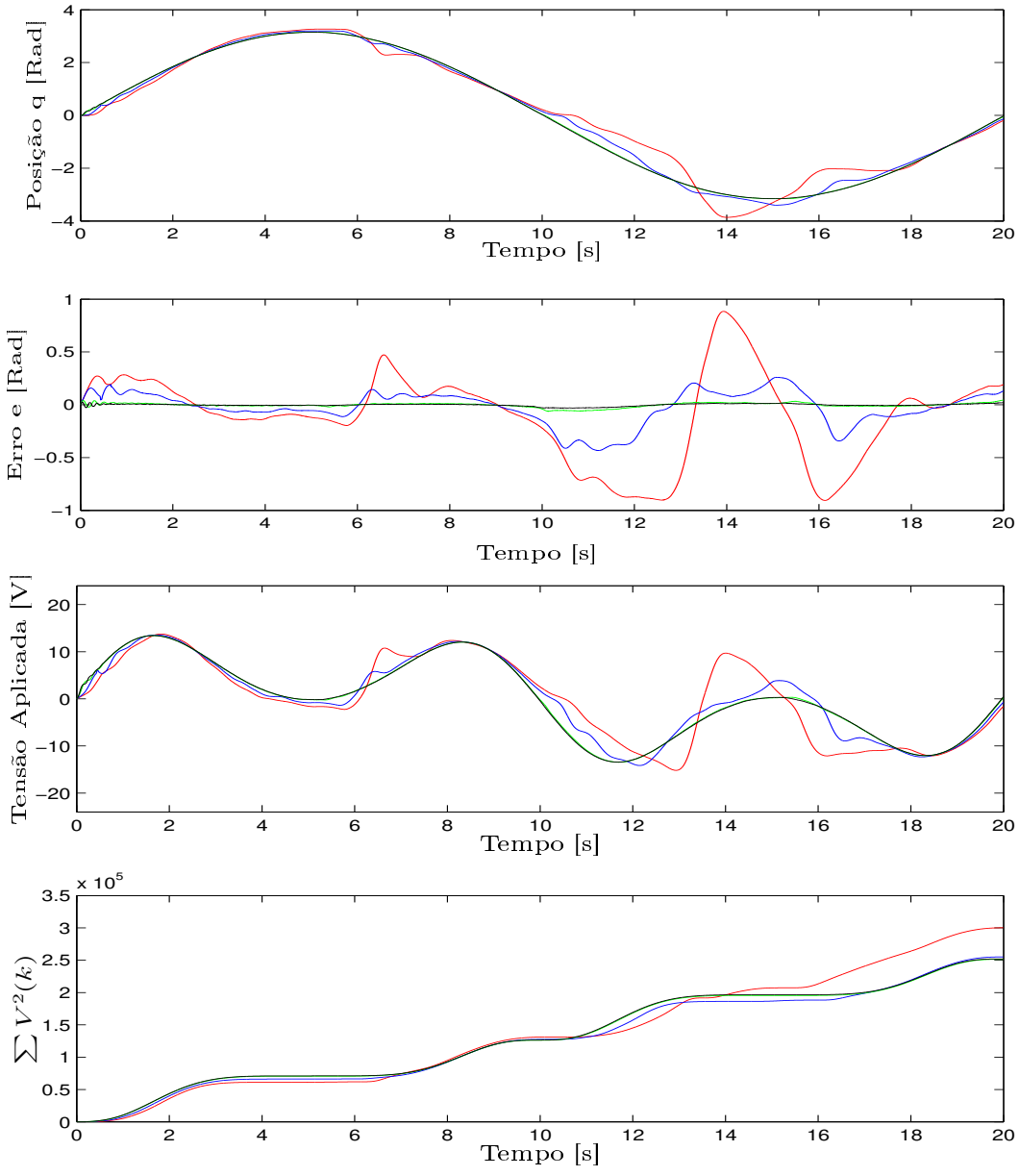

Figura 3.7: Evolução da Resposta do Neurocontrolador: (–) 1a. geração, (–) 5a. geração, (–) 20a. geração e (–) 50a. geração.

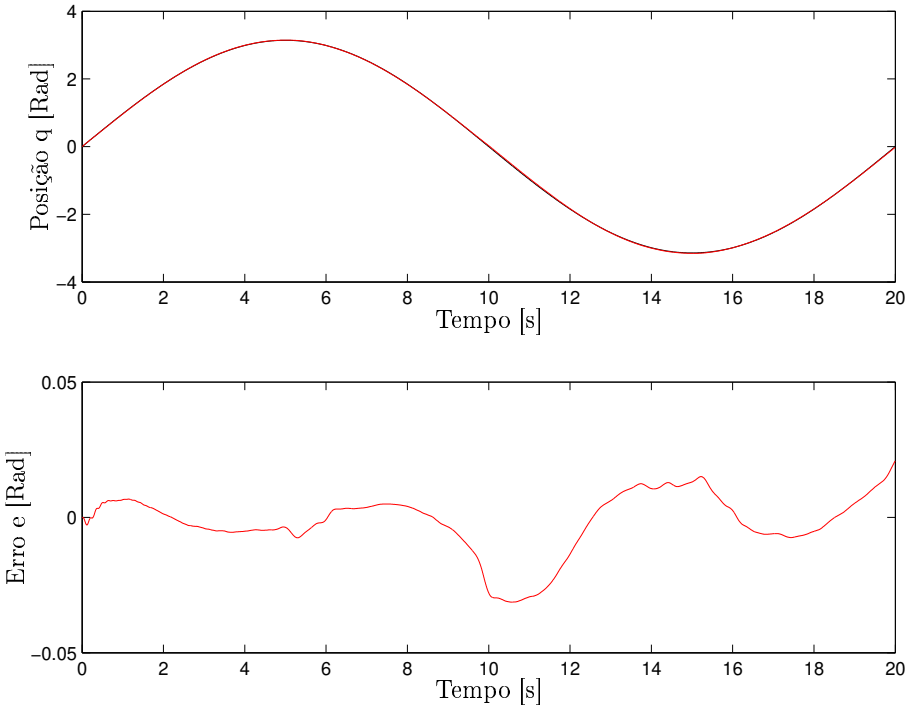

Figura 3.8: Resposta do Melhor Neurocontrolador Obtido no Processo de Evolução.

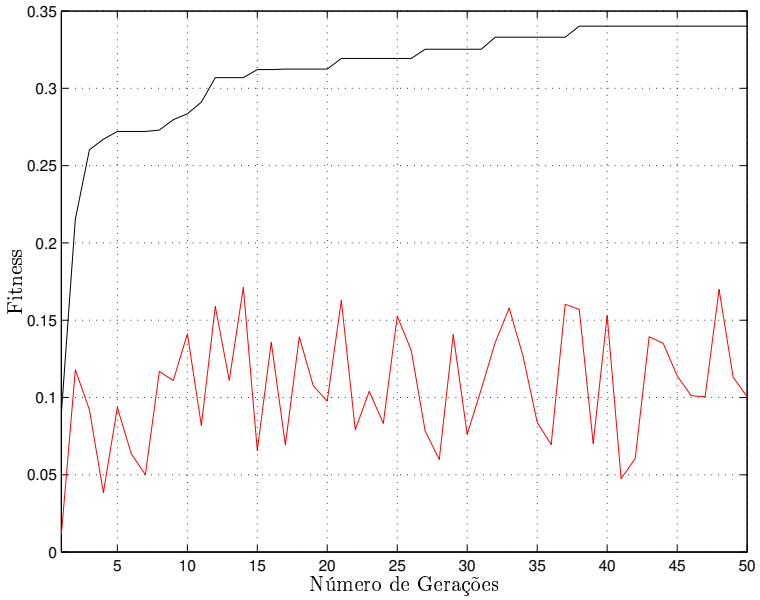

Figura 3.9: Evolução do *fitness*: (–) *Fitness* do Melhor Indivíduo, (–) *Fitness* Médio da População.

Parâmetros	Valor
Taxa Aprendizado $F = aI$	$a = 3.4494514465$
Termo $Momentum$	$\kappa = 0.0000166545.$

Tabela 3.2: Parâmetros do Melhor Controlador.

A Fig.3.10 apresenta o comportamento da variação dos pesos da rede do melhor controlador obtido na execução da tarefa de controle. Os pesos são inicializados em zero. E o processo de sintonia em tempo real pode ser observado.

Como o torque de equilíbrio dinâmico aplicado ao sistema, Eq.3.45, é função dos valores dos pesos, entre outras variáveis, podemos observar, de imediato, que a superfície de controle é variável no tempo, o que impossibilita o traçado de uma única figura que representa este dinamismo. Por outro lado, ao considerar-se o sistema em instantes específicos de tempo, pode-se elaborar tal figura.

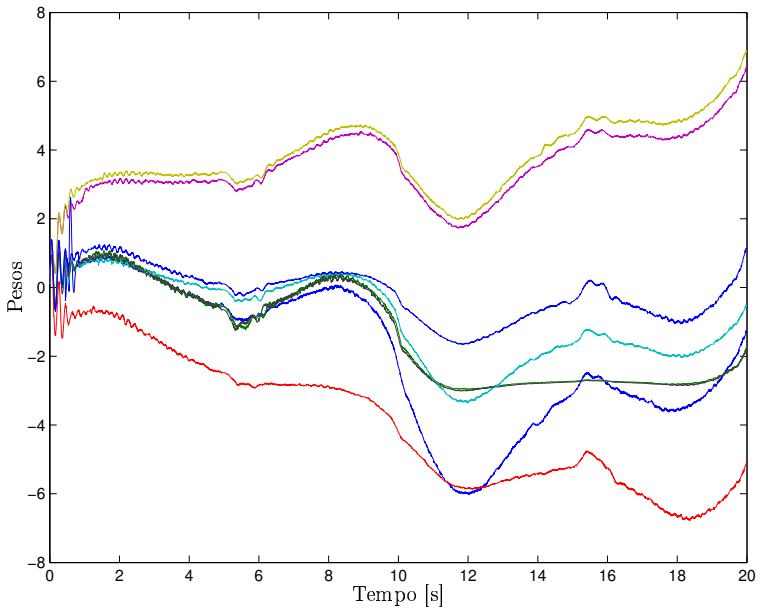

Figura 3.10: Variação dos Pesos da Rede do Melhor Controlador - 8 Neurônios.

A Fig.3.11 apresenta a superfície de controle do melhor controlador obtido no processo de ajuste, para os instantes $0s$, $10s$, $15s$ e $20s$. Estas figuras foram geradas numericamente, através do cálculo da saída do controlador para um conjunto de pontos definidos dentro do espaço das variáveis de entrada do controlador. Nota-se que a superfície de controle apresenta fortes características não-lineares, além de uma grande variação em função do tempo.

Com a intenção de mostrar a grande capacidade do controlador neural de *Kim-Lewis-Dawson* otimizado, em rastrear os mais diversos sinais de referência, mostramos a seguir, Fig.3.12 e Fig.3.13, a resposta do sistema controlado em relação a duas funções de referência: uma função não contínua (degrau) e uma função não linear. A linha preta representa a referência, e a linha

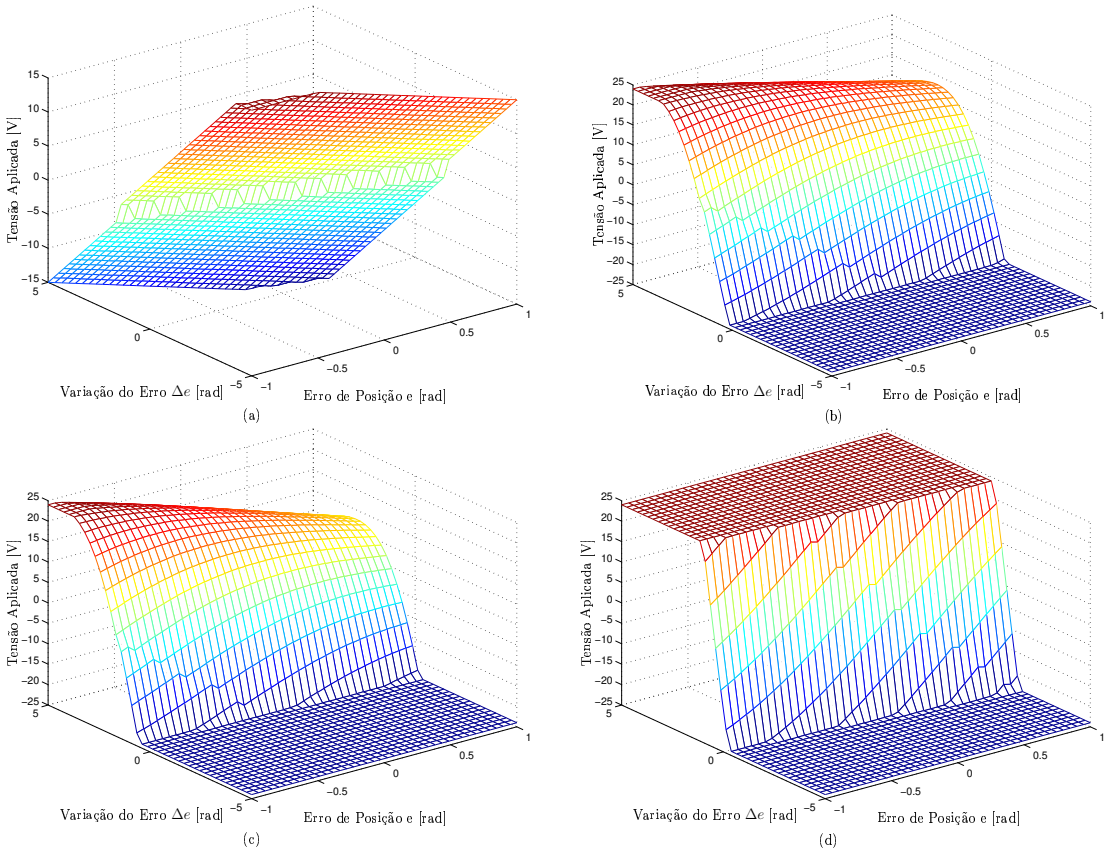

Figura 3.11: Superfícies de Controle do Melhor Controlador: (a) $0s$, (b) $10s$, (c) $15s$, (d) $20s$.

vermelha representa a posição rastreada pelo sistema.

Observa-se, pela análise da figuras, que o controlador de *Kim-Lewis-Dawson* possui uma excelente resposta para ambas as referências. Pode-se então concluir que este controlador poderia ser utilizado para quaisquer sinais de referência. É importante salientar que para ambos os casos os parâmetros de controle da rede foram mantidos nos valores apresentados pela Tabela 3.2.

A técnica de controle convencional (PID) não apresentou um desempenho satisfatório quando aplicado ao mesmo problema de rastreamento de trajetória. Nos experimentos realizados, o controlador PID, otimizado para executar uma função senoidal, levou o sistema à instabilidade quando aplicado às duas referências propostas a seguir.

As funções de referência utilizadas são descritas abaixo:

- Função não Contínua (Degrau):

$$q_d = \begin{cases} 0 & \text{se } t < 5s \\ \pi/2 & \text{se } t \geq 5s \text{ e } t < 12.5s \\ \pi & \text{se } t \geq 12.5s \end{cases}$$

- Função não Linear:

$$q_d = 0.5sen(2\pi t/20) + sen(6\pi t/20) + 1.5sen(10\pi t/20)$$

3.7 Sumário

Neste capítulo foi feita a descrição matemática do controlador neural de *Kim-Lewis-Dawson*, além da apresentação dos resultados obtidos na experimentação do sistema e otimização de parâmetros por Algoritmos Genéticos.

Na seção 3.2 foram introduzidos os conceitos pertinentes à Rede Neural para Aproximação de Funções Não-Lineares. Na seção 3.3 foi apresentado o modelo matemático da Dinâmica de Manipuladores Robóticos. Na seção 3.4 é apresentada a topologia do controlador neural de *Kim-Lewis-Dawson*, incorporando técnicas de controle ótimo linear associadas ao método de aprendizado da rede neural. Nas seções 3.5 e 3.6 são apresentados os resultados obtidos na experimentação do sistema e otimização de parâmetros por Algoritmos Genéticos.

No próximo Capítulo é apresentado o desenvolvimento teórico de um observador neural de estados, baseado na teoria sobre *Rede Neural para Aproximação de Funções Não-Lineares*, e na *Dinâmica de Manipuladores Robóticos e suas Propriedades*, proposto por Kim e Lewis (1999). Este observador tem função de eliminar a presença de tacômetros e contornar os problemas da diferenciação numérica, porém mantendo a precisão do sistema de controle.

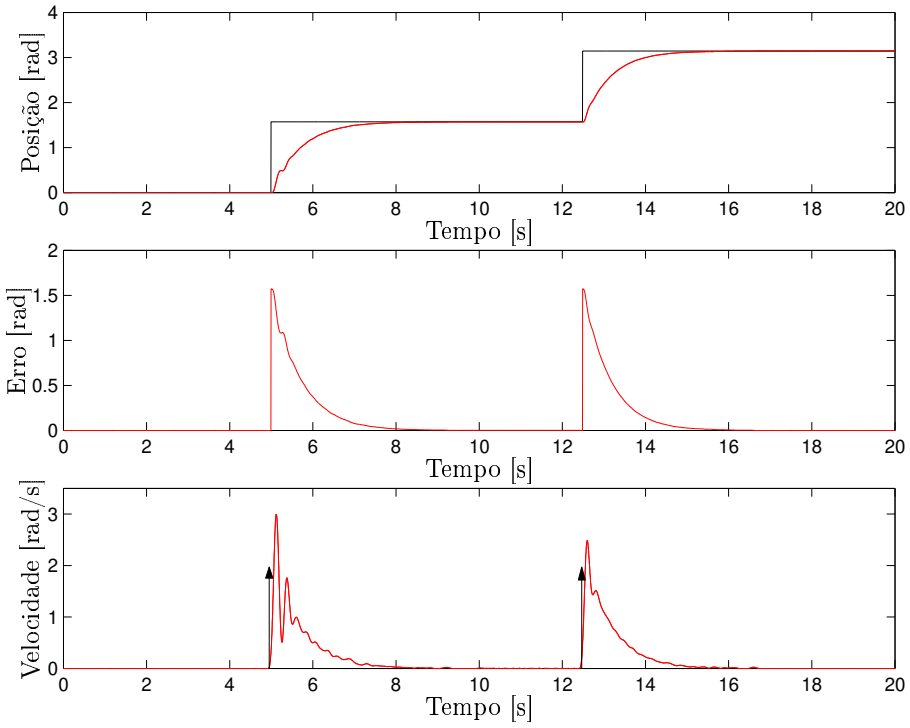

Figura 3.12: Referência Não Contínua; (−)Referência, (−)Resposta do Servomecanismo.

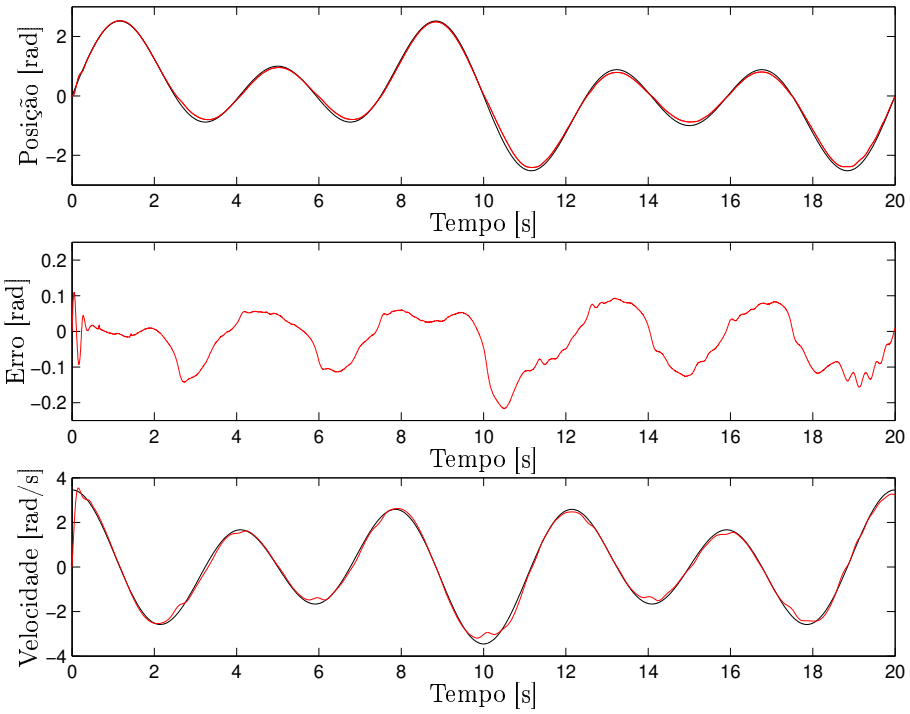

Figura 3.13: Referência Não Linear; (−)Referência, (−)Resposta do Servomecanismo.

Capítulo 4

Observador Neural de Estados

4.1 Introdução

A literatura apresenta grande número de trabalhos relacionados com a aplicação de técnicas de controle ótimo para o controle de manipuladores robóticos, Lin e Brandt (1996), Khoukhi (1999). Sendo que a maioria destas topologias usam informações sobre todos os componentes do vetor de estados do sistema para obtenção dos respectivos modelos dinâmicos, Lewis et al. (1995), Kim et al. (2000).

Estas informações podem ser obtidas mediante o uso de sensores apropriados para cada componente do vetor de estado. As medidas de posição são geralmente obtidas por meio de codificadores ópticos (*encoders*), que conferem grande precisão à aquisição de dados, além de possuírem baixo peso e custo relativo. Os sensores de velocidade mais comuns são os tacômetros, que comumente fornecem medidas contaminadas por grande quantidade de ruído, sendo ainda sensores que normalmente possuem volume e peso elevados.

A falta de qualidade na medida de velocidade, que pode reduzir consideravelmente o desempenho de um controlador, e fatores como espaço físico, custo e peso, justificam o fato de um grande número de manipuladores usar apenas sensores de posição para cada junta.

Informações sobre velocidade poderiam ser obtidas mediante derivação numérica. Porém, em alguns casos, devido a problemas de quantização, estas informações não podem ser obtidas mediante o uso deste método. O método de derivação numérica é eficiente apenas dentro de uma faixa mediana de velocidade do sistema.

Uma opção muito usada para eliminar a presença de tacômetros e contornar os problemas da diferenciação numérica, mas mantendo a precisão do sistema de controle, é o uso de observadores de estados. Neste contexto, um observador neural é apresentado em Kim e Lewis (1999). Neste tipo de controlador somente a matriz de inércia do modelo de manipuladores robóticos é considerada

conhecida. Os termos de *Coriolis*, gravidade e atritos são considerados como desconhecidos.

O desenvolvimento teórico do observador neural de estados está baseado na teoria sobre *Rede Neural para Aproximação de Funções Não-Lineares*, e na *Dinâmica de Manipuladores Robóticos e suas Propriedades*, apresentados no **Capítulo 3**.

4.2 Observador Neural

No desenvolvimento do observador e do controlador a ele associado, são usados os subíndices "o" e "c" referindo-se ao observador e ao controlador, respectivamente.

Definindo o erro de estimação da posição e da velocidade por

$$\tilde{q} = q - \hat{q}, \qquad \dot{\tilde{q}} = \dot{q} - \dot{\hat{q}}, \tag{4.1}$$

e selecionando como variáveis de estado $x_1 = q$ e $x_2 = \dot{q}$, a equação dinâmica do manipulador, Eq.3.7, pode ser reescrita como, Kim e Lewis (1999):

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= h_o(x) + M^{-1}(x_1)\tau(t) \end{aligned} \tag{4.2}$$

onde $h_o(x)$ é uma função não linear dada por

$$\begin{aligned} h_o(x) = -M^{-1}(x_1)[&V_m(x_1, x_2) + \\ &+ g(x_1) + F_v x_2 + f_c(x_2) + \tau_d(t)]. \end{aligned} \tag{4.3}$$

Assume-se que a matriz de inércia $M(q)$ seja conhecida, e que os termos de *Coriolis*, que possuem alto grau de incerteza e são difíceis de calcular analiticamente, e os termos referentes ao atrito, que possuem forma bastante complicada, sejam considerados como desconhecidos. De acordo com a propriedade de aproximação de funções das redes neurais, estes termos desconhecidos podem ser aproximados por uma rede neural. Neste sentido pode-se então equacionar $h_o(x)$ como

$$h_o(x) = W_o^T \sigma_o(x) + \varepsilon_o(x) \tag{4.4}$$

com $\|\varepsilon_o(x)\| \leq \varepsilon_{o,M}$ e $\sigma_o(x) \in \mathbb{R}^{N_o}$.

Uma estimativa de (4.4) em termos de $\hat{x}_1$ e $\hat{x}_2$ pode ser dada por

$$\hat{h}_o(\hat{x}) = \hat{W}_o^T \sigma_o(\hat{x}), \tag{4.5}$$

com os pesos da rede neural, $\hat{W}_o$, ajustados por algum algoritmo de treinamento eficiente.

Definindo para o observador a seguinte equação dinâmica:

$$\begin{aligned} \dot{\hat{z}}_1 &= \hat{x}_2 + k_D \tilde{x}_1 \\ \dot{\hat{z}}_2 &= \hat{W}_o^T \sigma_o(\hat{x}) + M^{-1}(x_1)\tau(t) + K\tilde{x}_1 \end{aligned} \tag{4.6}$$

com $k_D > 0$ e $K = K^T > 0$, as estimativas $\hat{x}_1$ e $\hat{x}_2$ para os estados x_1 e x_2 da Eq.4.2 são obtidas como, Nicosia e Tomei (1990):

$$\begin{aligned} \hat{x}_1 &= \hat{z}_1 \\ \hat{x}_2 &= \hat{z}_2 + k_P \tilde{x}_1 \end{aligned} \tag{4.7}$$

com $k_P > 0$. Desse modo, a equação dinâmica do observador pode ser reescrita em termos de $\hat{x}_1$ e $\hat{x}_2$,

$$\begin{aligned} \dot{\hat{x}}_1 &= \hat{x}_2 - k_D \tilde{x}_1 \\ \dot{\hat{x}}_2 &= \dot{\hat{z}}_2 + k_D \dot{\tilde{x}}_1 = \hat{W}_o^T \sigma_o(\hat{x}) + \\ &\quad + M^{-1}(x_1)\tau(t) + K\tilde{x}_1 + k_P \dot{\tilde{x}}_1 \end{aligned} \tag{4.8}$$

com $\tilde{x}_1 = x_1 - \hat{x}_1$ e $\tilde{x}_2 = x_2 - \hat{x}_2$.

A equação dinâmica do erro é obtida pela subtração da Eq.4.8 pela Eq.4.2, Kim e Lewis (1999),

$$\begin{aligned} \dot{\tilde{x}}_1 &= \tilde{x}_2 - k_D \tilde{x}_1 \\ \dot{\tilde{x}}_2 &= W_o^T \sigma_o(x) - \hat{W}_o^T \sigma_o(\hat{x}) + \\ &\quad - K\tilde{x}_1 - k_P \dot{\tilde{x}}_1 + \varepsilon_o(x) \end{aligned} \tag{4.9}$$

Adicionando e subtraindo $W_o^T \sigma_o(\hat{x})$ obtém-se

$$\begin{aligned} \dot{\tilde{x}}_1 &= \tilde{x}_2 - k_D \tilde{x}_1 \\ \dot{\tilde{x}}_2 &= W_o^T \sigma_o(\hat{x}) - K\tilde{x}_1 - k_P \dot{\tilde{x}}_1 + \varepsilon_o(\tilde{x}) \end{aligned} \tag{4.10}$$

com $\tilde{W} = W - \hat{W}$.

Considerando que $w_o(\tilde{x})$

$$w_o(\tilde{x}) = W_o^T [\sigma_o(x) - \sigma_o(\hat{x})]. \tag{4.11}$$

é limitado por

$$\|w_o(\tilde{x})\| \leq \varsigma_{o,M}, \tag{4.12}$$

com $\varsigma_{o,M} > 0$, então a dinâmica do erro do observador pode ser equacionada por:

$$\begin{aligned} \dot{\tilde{x}}_1 &= \tilde{x}_2 - k_D \tilde{x}_1 \\ \dot{\tilde{x}}_2 &= -k_p \tilde{x}_2 - (K - k_P k_d I)\tilde{x}_1 + \tilde{W}_o^T \sigma_o(\hat{x}) + \\ &\quad + \varepsilon_o(x) + w_o(\tilde{x}) \end{aligned} \tag{4.13}$$

onde $I \in \mathbb{R}^{n \times n}$ é a matriz identidade.

Este equacionamento da dinâmica do erro é de fundamental importância para a obtenção de uma regra para a condução do aprendizado baseada no erro de estimativa da posição.

Um algoritmo de ajuste de pesos da rede neural do observador deve ser planejado, e a estabilidade do sistema deve ser garantida. Suponha que os ganhos do observador são escolhidos de modo a satisfazer as seguintes condições:

$$k_P > k_D^2/2 - N_o/2 \tag{4.14}$$

$$\lambda_{min}(K) > (k_P^2 + N_o k_D^2)/2k_D \tag{4.15}$$

onde N_o é o número de neurônios da rede neural e, dada a seguinte função de ajuste de pesos

$$\dot{\hat{W}}_o = -k_D F_o \sigma_o(\hat{x}) \tilde{x}_1^T - \kappa_o F_o \|\tilde{x}_1\| \hat{W}_o - \kappa_o F_o \hat{W}_o \tag{4.16}$$

com $F_o = F_o^T > 0$ e $\kappa_o > 0$, então os erros de estimação $\tilde{x}_1$, $\tilde{x}_2$, e $\tilde{W}_o$ são limitados, e o sistema é estável no sentido de *Lyapunov*, Kim e Lewis (1999).

4.3 Controlador Neural

Como o observador desenvolvido acima deve estar associado a um controlador, foi proposto o uso do controlador neural de *Kim-Lewis-Dawson*, desenvolvido no **Capítulo** 3. Desse modo, o texto a seguir desenvolve a teoria necessária para a aplicação do observador descrito acima, quando aplicado ao controlador neural de *Kim-Lewis-Dawson*.

Dada uma trajetória desejada $q_d(t) \in \mathbb{R}^n$, os vetores de erro, $e(t)$ e $\dot{e}(t)$ são definidos como

$$\begin{aligned} e(t) &= q_d(t) - q(t) \\ \dot{e}(t) &= \dot{q}_d(t) - \dot{q}(t) \end{aligned} \tag{4.17}$$

e a medida de desempenho instantâneo do sistema é obtida por

$$\hat{r}(t) = \dot{\hat{e}}(t) + \Lambda e(t). \tag{4.18}$$

O modelo dinâmico do robô, Eq.3.7, pode então ser reescrito como

$$M(q)\dot{\hat{r}}(t) = -V_m(q,\dot{q})\hat{r}(t) - \tau(t) + h(x) \tag{4.19}$$

sendo que as funções não-lineares deste modelo estão agrupadas em $h(x)$:

$$\begin{aligned} h(x) = {} & M(q)(\ddot{q}_d + \ddot{\tilde{q}} + \Lambda\dot{e}) + \\ & + V_m(q,\dot{q})(\dot{q}_d + \dot{\tilde{q}} + \Lambda e) + \\ & + g(q) + F_v\dot{q} + f_c(\dot{q}) + \tau_d(t). \end{aligned} \tag{4.20}$$

Desse modo temos em $h(x)$ uma função que captura toda a dinâmica não linear do modelo do robô em questão.

Define-se a lei de controle como

$$\tau(t) = h(x) - u(t), \tag{4.21}$$

com $u(x) \in \mathbb{R}^n$ sendo uma entrada de controle auxiliar a ser definida posteriormente. Tem-se então o sistema em malha fechada dado pela equação

$$M(q)\dot{\hat{r}}(t) = -V_m(q,\dot{q})\hat{r}(t) + u(t) \tag{4.22}$$

Definindo a equação dinâmica do erro como

$$\dot{\tilde{e}}(t) = -\Lambda e(t) + \hat{r}(t) \tag{4.23}$$

o seguinte sistema de equações de estado pode ser escrito:

$$\dot{\tilde{z}} = \begin{bmatrix} \dot{\tilde{e}} \\ \dot{\hat{r}} \end{bmatrix} = \begin{bmatrix} -\Lambda & I \\ 0 & -M^{-1}V_m \end{bmatrix} \begin{bmatrix} e \\ \hat{r} \end{bmatrix} + \begin{bmatrix} 0 \\ M^{-1} \end{bmatrix} u(t) \tag{4.24}$$

ou em notação abreviada

$$\dot{\tilde{z}} = A(q,\dot{q})\tilde{z} + B(q)u(t) \tag{4.25}$$

com $A(q,\dot{q}) \in \mathbb{R}^{2n\times 2n}$, $B(q) \in \mathbb{R}^{2n\times n}$, $\tilde{z}(t) \in \mathbb{R}^{2n\times 1}$. Definindo-se como função de custo a seguinte função quadrática:

$$J(u) = \int_{t_o}^{\infty} L(\tilde{z},u)dt \tag{4.26}$$

com Lagrangeano

$$L(\tilde{z},u) = \frac{1}{2}\tilde{z}^T Q\tilde{z} + \frac{1}{2}u^T Ru. \tag{4.27}$$

O objetivo é encontrar a lei de controle $u(t)$, que minimiza a Eq.4.26 sujeita às restrições da equação diferencial imposta pela Eq.4.24. A lei de controle que atinge este objetivo é denotada por $u^*(t)$. A condição necessária e suficiente para que $u^*(t)$ minimize a Eq.4.26 sujeita à Eq.4.24, segundo Lewis e Syrmos (1995), é que exista uma função $V(\tilde{z},t)$ que satisfaça a equação H-J-B, Eq.3.21

A seguinte função $V(\tilde{z},t)$ composta por $M(q)$ e uma matriz simétrica e definida positiva $K = K^T \in \mathbb{R}^{n\times n}$ satisfazendo a equação H-J-B, é empregada:

$$V = \frac{1}{2}\tilde{z}^T P(q)\tilde{z} = \frac{1}{2}\tilde{z}^T \begin{bmatrix} K & 0 \\ 0 & M(q) \end{bmatrix} \tilde{z} \tag{4.28}$$

onde Λ e K em Eq.4.23 e Eq.4.28 são determinados pela solução da equação diferencial de *Riccati*, Eq.3.24.

Seguindo o raciocínio desenvolvido no **Capítulo** 3, a lei de controle ótimo $u^*(t)$ que minimiza a Eq.4.26 sujeita à Eq.3.17 é:

$$u^*(t) = -R^{-1}B^T P(q)\tilde{z} = -R^{-1}\hat{r} \tag{4.29}$$

A função não linear da dinâmica do robô, Eq.4.20, pode ser representada por

$$h_c(x) = W_c^T \sigma_c(p) + \varepsilon_c(x), \qquad \|\varepsilon_c(x)\| \leq \varepsilon_{c,M} \tag{4.30}$$

Sendo que o erro de aproximação $\|\varepsilon_c(x)\|$ da rede neural possui um limitante superior conhecido $\varepsilon_{c,M}$.

Uma estimativa $\hat{h}_c(x)$ de $h_c(x)$ pode ser calculada por

$$\hat{y}_c = \hat{W}_c^T \sigma_c(p). \tag{4.31}$$

Os torques externos $\tau(t)$, aplicados so sistema, passam a ser dados pela equação

$$\tau(t) = \hat{W}_c^T \sigma_c(p) - u^*(t) - \eta(t), \tag{4.32}$$

onde $\eta(t)$ é um vetor que confere robustez ao sistema. Então, a Eq.4.19 fica

$$\begin{aligned} M(q)\dot{\hat{r}}(t) &= -V_m(q,\dot{q})\hat{r}(t) + \tilde{W}_c^T \sigma_c(p) + \\ &\quad + \varepsilon_c(x) + u^*(t) + \eta(t) + \tau_d(t), \end{aligned} \tag{4.33}$$

com $\tilde{W}_c = W_c - \hat{W}_c$.

A descrição no espaço de estados da Eq.4.22 pode ser dada por

$$\dot{\tilde{z}} = A\tilde{z} + B[\tilde{W}_c^T \sigma_c(p) + \varepsilon_c(x) + u^*(t) + \eta(t) + \tau_d(t)] \tag{4.34}$$

com $\tilde{z}$, A e B fornecidos pela Eq.4.25. Inserindo a lei de controle ótimo da Eq.4.29 na Eq.4.34, temos

$$\begin{aligned} \dot{\tilde{z}} &= (A - BR^{-1}B^T P)\tilde{z} + \\ &\quad + B[\tilde{W}_c^T \sigma_c(p) + \varepsilon_c(x) + \eta(t) + \tau_d(t)]. \end{aligned} \tag{4.35}$$

Utilizando-se o termo robusto

$$\eta(t) = k_z \frac{\hat{r}(t)}{\|\hat{r}(t)\|}, \tag{4.36}$$

com $k_z \leq b_d$, e uma função de adaptação dos pesos da rede

$$\dot{\hat{W}}_c = F_c \sigma_c(p)(B^T P(q)\tilde{z})^T - \kappa_c \|\tilde{z}\| \hat{W}_c \tag{4.37}$$

com $F_c = F_c^T > 0$ e $\kappa_c > 0$, então os erros $e(t)$, $\hat{r}(t)$ e os pesos $\tilde{W}_c(t)$ são limitados, e portanto estáveis no sentido de *Lyapunov*, Kim et al. (2000).

O diagrama de blocos da Fig.4.1 mostra os principais componentes do controlador neural

4.4 Resultados de Implementação

O observador neural e o controlador neural discutidos acima foram implementados no sistema mecânico descrito no **Apêndice A**, e os principais resultados são apresentados a seguir.

O sistema foi implementado com as seguintes características principais:

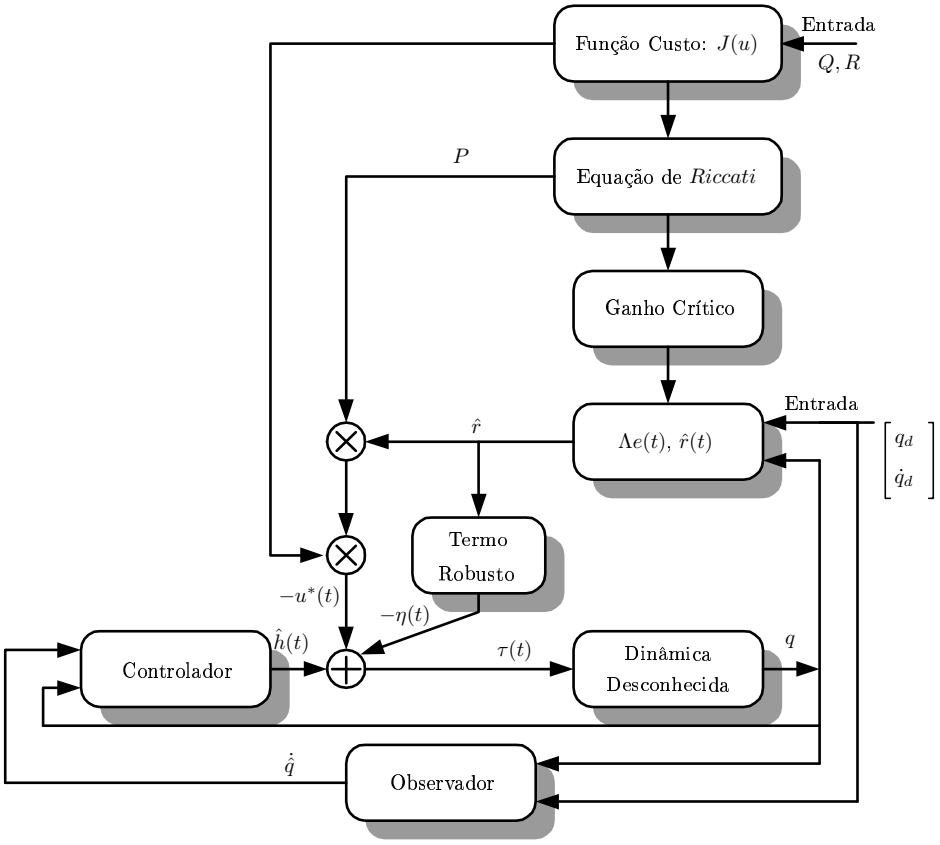

Figura 4.1: Controlador Neural Ótimo mais Observador de Estados

- Tanto o observador como o controlador foram discretizados com um período de amostragem de $5ms$. No processo de discretização, as equações diferenciais foram solucionadas utilizando-se o método do trapézio.

- As matrizes Q e R que compõe a função custo possuem os seguintes valores:

$$Q = \begin{bmatrix} 3 & -1 \\ -1 & 4 \end{bmatrix}, \qquad R^{-1} = 4. \tag{4.38}$$

- O controlador é composto por oito neurônios na camada interna, com oito entradas:

$$x_c = p_c = \begin{bmatrix} 1 & q^T & \dot{q}^T & e^T & \dot{e}^T & \hat{r}^T & q_d^T & \dot{q}_d^T \end{bmatrix} \tag{4.39}$$

- O observador é composto de sete neurônios na camada interna, com sete entradas:

$$x_o = p_o = \begin{bmatrix} 1 & \tilde{q}^T & \hat{q}^T & \dot{\hat{q}}^T & \|\tilde{q}\| & \|\hat{q}\| & \|\dot{\hat{q}}\| \end{bmatrix} \tag{4.40}$$

- A função de ativação utilizada na camada intermediária foi a função sigmoidal:

$$\sigma(x) = \frac{1}{1 + \exp(-x)} \tag{4.41}$$

- A função de ativação utilizada na camada de saída foi a função linear:

$$\sigma(x) = x \tag{4.42}$$

- Tempo de execução: $20s$
- O observador foi implementado com $F_o = diag[2, \cdots, 2]$, $\kappa_o = 0.0001$, $k_D = 20$, $k_P = 200$, $K = diag[400, \cdots, 400]$
- O controlador foi implementado com $F_c = diag[2, \cdots, 2]$, $\kappa_c = 0.0001$.

Com a intenção de testar o desempenho do controlador, foram utilizados neste experimento dois sinais diferentes de referência. Cada sinal de referência utilizado foi gerado pela seguinte equação:

$$q_d = \pi \sin(2\pi f t) \tag{4.43}$$

com freqüências $f = 0.05Hz$ para baixa velocidade, e $f = 0.25Hz$ para alta velocidade.

As evoluções do sistema controlado estão apresentadas nas Fig.4.2 e Fig.4.3. Observa-se que o aprendizado está ativo em todo o período de execução (treinamento *on-line*). Nota-se que a mudança no sinal de referência causa um crescimento momentâneo no sinal de erro, situação que força o controlador a uma readaptação dos pesos da rede, no entanto, após decorrido algum tempo, o controlador se adapta às novas condições e leva o sinal de erro a valores menores. A Fig.4.4 apresenta uma ilustração do erro de estimativa de velocidade do observador, demonstrando a validade da aplicação deste observador.

A Fig.4.4 apresenta uma comparação entre a velocidade real do servomecanismo e a velocidade estimada pelo observador. Pode-se observar que o erro $\tilde{x}_2$ entre as duas variáveis é pequeno, fato que confere comprovação prática à teoria utilizada no projeto deste observador.

A Fig.4.5 apresenta uma comparação entre o desempenho do controlador neural apresentado no **Capítulo 3**, que utiliza a medição da velocidade, e o desempenho do controlador neural apresentado neste **Capítulo**, utilizando estimativa de velocidade.

A comparação entre os sinais de erro dos dois controladores demonstra uma pequena melhora da resposta quando utiliza-se o controlador em conjunto com o observador de estados. Esta melhoria deve-se, principalmente, à presença de ruído nas medidas de velocidade utilizadas pelo primeiro controlador. Os resultados obtidos neste experimento justificam o projeto do controlador com observador de estados, que, apesar de possuir uma etapa de projeto mais difícil, elimina a necessidade do uso de tacômetros, mantendo a qualidade da resposta do servomecanismo frente a uma determinada tarefa de controle.

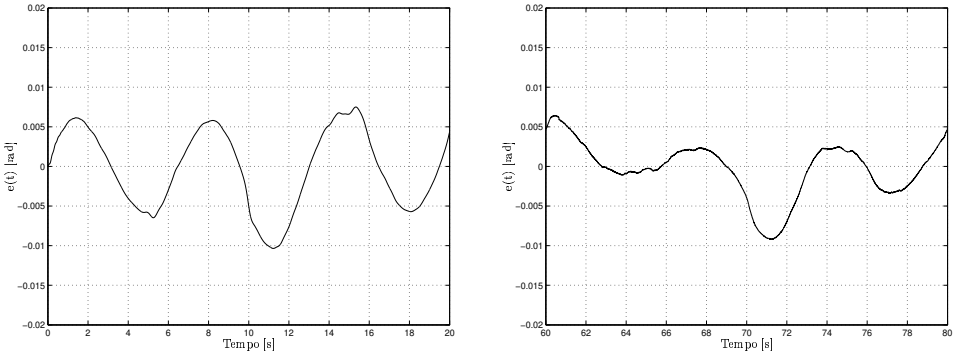

Figura 4.2: Performance do Controlador para Referência de Baixa Velocidade.

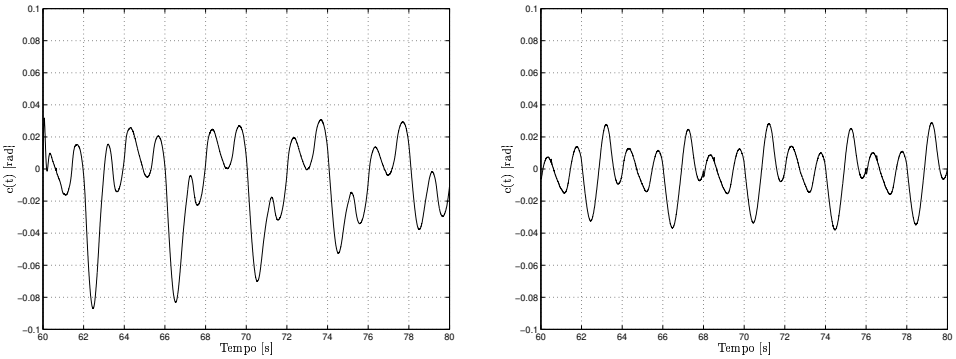

Figura 4.3: Performance do Controlador para Referência de Alta Velocidade.

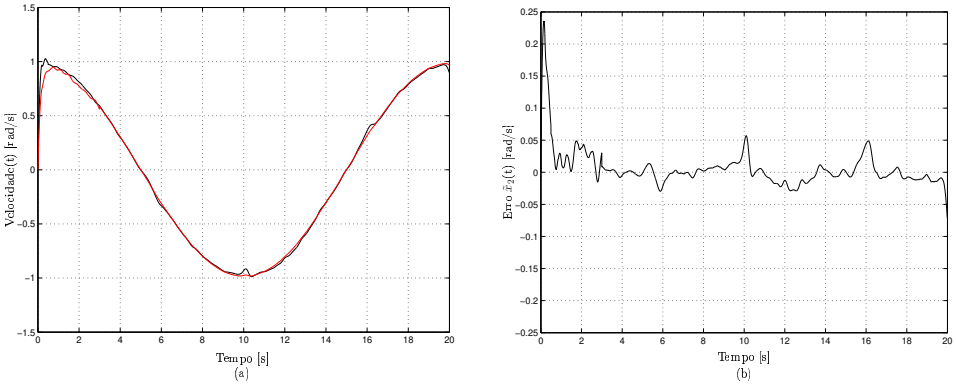

Figura 4.4: Performance do Observador para Referência de Baixa Velocidade. (a) (−)Velocidade Estimada (−)Velocidade Real; (b) (−)Erro de Estimação de Velocidade $\tilde{x}_2(t)$

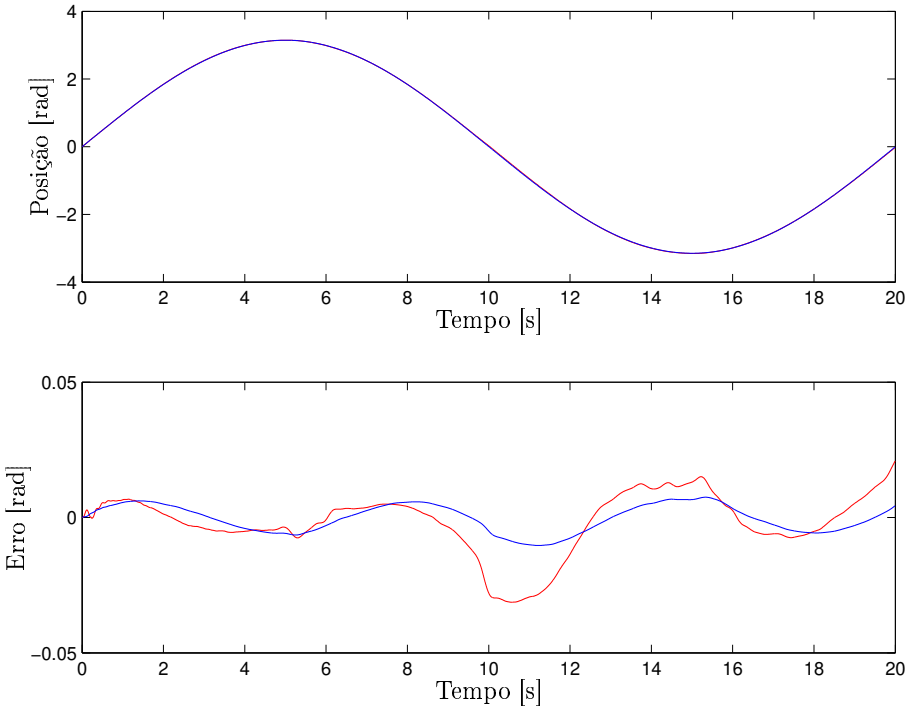

Figura 4.5: Comparação Entre a Performance dos Controladores Neurais, (−)Com Medição da Velocidade, (−)Com Estimativa da Velocidade.

Capítulo 5

Conclusões e Direções Futuras

Nesta tese, foi abordado o problema de implementação prática de controladores neurais de *Kim-Lewis-Dawson* com ajuste de parâmetros por algoritmos genéticos. Este estudo resultou em contribuições importantes para a síntese de controladores não-lineares, destacando-se três pontos principais: desenvolvimento de metodologia experimental baseada em algoritmos genéticos conjugada à teoria de redes neurais para busca numérica de controladores eficientes, no sentido de otimização de funções de desempenho "multi-objetivo"; implementação digital usando microcomputadores de baixo custo, ou seja, com boa relação custo/benefício; e construção de um exemplo que valida a aplicação prática da abordagem proposta para o controlador neural. Em particular, foi criado um procedimento computacional para o projeto de controladores de servomecanismos, e que é extensível a uma infinidade de outros sistemas não-lineares com características semelhantes ao sistema estudado.

A abordagem proposta, utilizando algoritmos genéticos para o ajuste paramétrico do controlador neural, possibilitou a obtenção de excelentes resultados experimentais, que representam um grande avanço na área de algoritmos genéticos aplicados ao controle, já que a grande maioria dos resultados publicados pertence ao campo da simulação.

Os controladores neurais de *Kim-Lewis-Dawson* obtidos do processo de otimização apresentaram em todos os casos experimentados uma resposta de posição rápida com erro de regime e sobresinal aproximadamente nulos, considerando-se as constantes de tempo do servomecanismo usado para prova experimental. Isto significa, que todos os objetivos incorporados na função de *fitness* foram atingidos, comprovando a adequação da função proposta.

O tempo total necessário para completar o processo de sintonia dos controladores depende do intervalo gasto para cada tarefa de controle, e do número de indivíduos da população do algoritmo genético. O tempo total decorrido na execução do algoritmo genético proposto durante processo de aprendizagem foi de aproximadamente 8 horas. Do ponto de vista prático, o processo de sintonia poderia ser interrompido a partir do momento em que um determinado objetivo fosse alcançado, por exemplo, quando a função de *fitness* atingisse um valor desejado. Este e vários

outros critérios de parada podem ser utilizados com o objetivo de reduzir o tempo de sintonia.

Não foi foco principal da pesquisa até aqui realizada comparar o desempenho do algoritmo genético proposto em função da variação de seus parâmetros, como taxa de mutação, taxa de *crossover*, etc. No entanto, estes parâmetros podem também influenciar na quantidade de gerações executadas para se obter resultados satisfatórios. Contudo, não é possível garantir que um conjunto de parâmetros seja mais adequado para todos os tipos de problemas, mesmo porque o procedimento de busca do algoritmo genético é um processo aleatório. Conseqüentemente, a estratégia adotada consistiu na escolha empírica de valores de parâmetros cujos resultados mostraram-se satisfatórios.

Devido à generalidade estrutural do algoritmo genético proposto, ele pode ser usado em muitos tipos de aplicações práticas de controle, mas é evidente que as funções de *fitness* devem ser adaptadas para representar adequadamente cada objetivo de controle desejado.

A seguir, destacam-se alguns pontos interessantes que foram fundamentais para obtenção dos resultados positivos apresentados.

- O algoritmo genético proposto possui características que permitem sua aplicação prática em problemas de controle em tempo real.
- A escolha de uma função de *fitness* adequada para problemas de controle.
- Desenvolvimento de rotinas em C para implementação de algoritmos genéticos e controladores neurais: Este ponto foi fundamental para viabilizar o desenvolvimento dos algoritmos de controle em tempo real.
- Construção de um sistema mecânico e circuitos eletrônicos para validação prática da técnica proposta.

Visando a continuação e a extensão da pesquisa realizada, foi feita a seguinte listagem contendo idéias e sugestões para trabalhos futuros:

- Desenvolver um procedimento que garanta que o espaço de busca de parâmetros do controlador encontre-se sempre em uma região estável do sistema.
- Desenvolver a abordagem de controle neural com compensação de gravidade para servomecanismos robóticos com um, dois ou mais graus de liberdade.
- Aplicar o algoritmo de otimização e controle para gerar controladores para outros tipos de sistemas dinâmicos.
- Realizar estudos na área de otimização multi-objetivo a fim de sintetizar funções de *fitness* que incluam outras medidas de desempenho dos controladores, complexidade numérica do algoritmo, desgaste do sistema mecânico, etc.
- Criar um algoritmo genético para configuração estrutural do controlador neural, como definição do número de neurônios da camada intermediária, tipo de função de ativação, etc.

Os resultados práticos obtidos demonstram o grande poder de adaptação do controlador neural de *Kim-Lewis-Dawson* frente aos mais diversos sinais de trajetória de referência. Este poder de adaptação dos referidos controladores é de extrema importância, no que diz respeito ao controle de trajetórias de manipuladores robóticos, uma vez que permite a inclusão de uma infinidade de sinais de referência no controle de movimento do manipulador, facilitando enormemente tarefas como planejamento/rastreamento de trajetórias e desvio de obstáculos.

Apêndice A

Descrição do Sistema Experimental

Os resultados mais relevantes da pesquisa realizada dependeram de vários fatores técnicos que são essenciais para a realização de um trabalho de qualidade com dados experimentais confiáveis. Este Anexo tem por finalidade explicitar alguns detalhes do *hardware* e do *software* relacionados aos experimentos práticos realizados.

A.1 O Sistema Experimental

O sistema prático utilizado para a obtenção dos resultados apresentados é composto de um elo robótico acionado, um motor de corrente contínua (CC) com imã permanente, um codificador óptico incremental (*encoder*), um circuito de interfaceamento, um contador $UP/DOWN$, um amplificador de potência, e um microcomputador do tipo $PC486$ com relógio de 100 MHz. O algoritmo de controle foi implementado via *software*, sendo que o sinal de controle calculado é enviado para a interface de E/S (Entrada/Saída). A saída do sinal de controle é gerada por uma palavras 12 bits, este sinal é amplificado, por um recortador PWM, para alimentar um motor CC que está conectado ao elo. A posição do pêndulo é lida através de um contador $UP/DOWN$ cuja contagem é função dos movimentos de um *encoder* óptico convenientemente acoplado ao elo. A Fig.A.1 ilustra como todas as partes do sistema prático foram interligadas.

A seguir, são descritas de forma mais detalhada as partes que compõem o sistema experimental.

A.1.1 Elo Robótico

O elo robótico utilizado nos experimentos consiste de um sistema mecânico relativamente simples composto de dois eixos e engrenagens do tipo polias e correias sincronizadas. O motor está

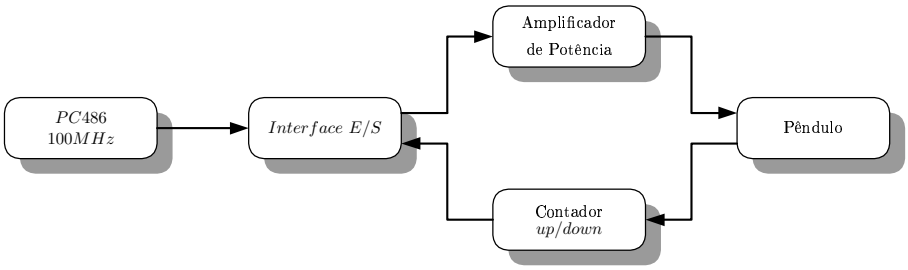

Figura A.1: Diagrama do Sistema Experimental.

acoplado a um dos eixos através de uma engrenagem que reduz a velocidade por um fator de $4:1$. Este primeiro eixo está acoplado ao segundo também através de uma engrenagem com fator de redução de $4:1$. Portanto, no primeiro eixo a velocidade do motor fica reduzida de $4:1$ e no segundo de $16:1$. O corpo do elo pode ser conectado a qualquer um dos eixos, dependendo da redução desejada. Nos experimentos realizados, ele foi acoplado ao primeiro eixo para obter-se efeitos mais intensos das não-linearidades causadas pela ação do torque gravitacional. Para conseguir-se medidas mais precisas da sua posição, o *encoder* foi conectado ao mesmo eixo de movimento deste elo. As etapas de projeto, usinagem e construção deste sistema foram realizadas pela própria equipe do Laboratório de Sistemas Modulares Robóticos ($LSMR - FEEC$). A Fig.A.2 mostra um vista da bancada de laboratório que foi utilizada nos testes experimentais cujos resultados estão apresentados no corpo deste trabalho, e também um *croquis* mostrando alguns detalhes da construção do elo acionado.

A.1.2 Motor

O motor utilizado para acionar o elo é um servomotor de corrente contínua modelo $E-530$, da *Electro-Craft*. As especificações técnicas deste motor estão descritas na Tabela A.1.

Modelo	E−530
Potência Máxima	28 [Watts]
Torque Máximo	192.5×10^{-3} [N.-m.]
Inércia da Armadura	2.29×10^{-4} [N.-m.-s]^2
Coeficiente de Amortecimento	6×10^{-4} [N.-m./KRPM]
Constante de Tempo Elétrica	2.06×10^{-3} [s]
Constante de Tempo Mecânica	8.30×10^{-3} [s]
Rotação Máxima (sem carga)	6000 [RPM]
Resistência Elétrica de Armadura	1.55 [Ohms]
Indutância da Armadura	3.19×10^{-3} [H]
Tensão	24 [V]

Tabela A.1: Especificações Técnicas do Motor.

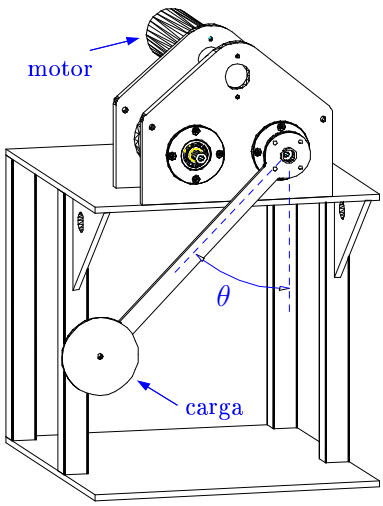

Figura A.2: Bancada Experimental Com a Montagem do Elo Acionado.

A.1.3 Circuito de Entrada/Saída (I/O)

Este circuito tem a função de decodificar os endereços que serão utilizados para a escrita e leitura de dados. A Fig.A.3 mostra o diagrama esquemático do circuito de Entrada/Saída.

São utilizados *buffers* em todas as linhas de comunicação externa com o microcomputador, pois segundo a descrição técnica dos *slots*, cada linha de sinal consegue acionar seguramente apenas um dispositivo TTL do tipo LS. O circuito integrado $U2$ ($74F245$) é um transmissor bidirecional para barramento (*tri-state*), que permite o tráfego de dados entre o computador e os circuitos a ele conectados. A direção do fluxo de dados é determinada pelo nível lógico aplicado ao pino DIR, neste caso, conectado ao sinal $-IOR$. Os circuitos integrados $U3$ e $U4$ são *buffers* ($74LS244$) de 8 bits cada, que permitem a passagem dos sinais de controle e do barramento de endereços para o circuito. O circuito integrado $U5$ ($74F138$) é um demultiplexador/decodificador 3×8 que estabelece a faixa de endereços de E/S através dos sinais $ADR2$, $ADR3$ e $ADR4$, e

é utilizado para gerar e/ou distinguir oito endereços disponibilizados para escrita/leitura ($PE1$, $PE2$, ...,$PE8$).

A.1.4 *Encoder* Óptico

O sensor utilizado para medir a posição angular do elo é um *encoder* óptico modelo $HEDS-6310$, Hewlett-Packard (1993), que possui uma resolução de 1024 pulsos por volta. Este *encoder* dispõe de três fases de saída, sendo duas fases (FA e FB) deslocadas de 90^o entre si, cujos sinais são usados para as medições do deslocamento angular, e uma terceira fase (FS) que emite um pulso de referência por volta, e que pode ser utilizado para estabelecer o referencial das medições. O circuito de decodificação dos sinais do *encoder* (ver figura A.4) é baseado no circuito integrado $HCTL2020$ fabricado pela própria HP. Este CI integra num só componente toda a lógica de detecção de fase e um circuito contador *UP/DOWN* de 16 bits.

A lógica de detecção de fase é responsável pela identificação do sentido de rotação do eixo de movimento do elo, ou do eixo do *encoder* que está solidário. Em cada pulso emitido pelo *encoder* é possível identificar quatro transições de estado nas fases FA e FB. Com isso, consegue-se uma resolução total de 4096 pulsos por volta, que equivale a uma precisão de $0,09$ graus na medida da posição angular.

A.1.5 Geração de Sinais Digitais PWM e Amplificador de Potência

O circuito amplificador de potência compõe o estágio responsável pela amplificação dos sinais provenientes do algoritmo de controle. O diagrama de blocos do circuito amplificador de potência encontra-se ilustrado na Fig.A.5. O circuito é baseado no amplificador de potência do tipo *full-bridge*, cujas principais características são:

- transistores de potência tipo $MOSFET$;
- alimentação de até 200 Volts;
- capacidade de corrente de saída (contínua) de até 33A;
- capacidade de corrente de saída (pulsada) de até 110A;
- freqüência de chaveamento de 20KHz

O circuito desenvolvido é composto de dois estágios. O primeiro estágio, Fig.A.5, é responsável pelo condicionamento da palavra digital de 12 bits. Neste estágio, a palavra digital é transformada em um sinal de tensão de valor médio igual ao valor numérico contido na palavra digital. Para atingir este objetivo o circuito foi projetado com um contador ascendente de 12 bits, que possui sua saída comparada com a palavra digital proveniente do algoritmo de controle. Se a palavra digital for maior que o valor do contador um sinal de tensão com valor zero é gerado. Se a palavra digital for menor ou igual ao valor do contador, um sinal de tensão com valor um é gerado.

Este sinal ainda é condicionado de modo a atingir níveis de tensão aceitáveis para acionamento dos transistores $MOSFET - IRF540N$ (International Rectifier). O segundo estágio consiste da amplificação de potência e tem a função de fornecer a corrente necessária para alimentação do motor. A Fig.A.6 apresenta o sinal de tensão aplicado ao motor, pelo amplificador de potência, em um determinado instante de tempo.

A.2 Alguns Detalhes do Software

Todas as rotinas de acesso a *hardware*, geração de interrupções e controle em tempo real foram escritas em linguagem C.

A.2.1 Interrupções

Aplicações de controle em tempo-real exigem precisão e confiabilidade quanto ao período de amostragem. É necessário garantir que o sinal de controle e os dados lidos sejam atualizados a cada período de amostragem, que no caso deve ser fixo.

O sistema de amostragem utilizado é baseado na geração de interrupções. Os microcomputadores do tipo $PC - AT$ possuem contadores internos que podem ser programados para gerar interrupções. Neste trabalho, utilizou-se o relógio de sistema do PC para gerar regularmente a interrupção $INT\,0{\times}1C$. Normalmente, este relógio está programado para gerar uma interrupção a cada $54,9$ milisegundos ($18,2$ interrupções por segundo), podendo ser reprogramado para obtenção de períodos de amostragem de até no mínimo $0,838$ microsegundos. Nos experimentos realizados utilizou-se um período de amostragem de 5 milisegundos, tempo suficiente para o cálculo do sinal de controle pelo algoritmo de controle. Geralmente, aplicações de robótica permitem intervalos de amostragem de até 10 milisegundos para que se obtenha bons desempenhos e grande precisão.

Uma vez inicializada a rotina de interrupção, em cada instante de amostragem executam-se os seguintes passos:

1. Incrementa a variável global **tickcount**, que indica quantos intervalos de amostragem se passaram desde a inicialização da rotina de interrupção. Esta variável global é utilizada para determinar o tempo decorrido na tarefa de controle.
2. Lê o estado do contador e armazena o resultado na variável global **position**.
3. Escreve-se, no endereço correspondente à saída, a palavra digital de controle armazenada na variável global **PWMsinal**, que foi calculada no intervalo de amostragem anterior.

Maiores detalhes sobre as rotinas, variáveis locais/globais, e exemplos de programas podem ser encontrados em Souza (2000).

A.3 Modelo Matemático do Sistema

Com a finalidade de permitir a simulação e a identificação de parâmetros o sistema dinâmico do elo acionado usado como paradigma experimental desta dissertação foi modelado matematicamente. Os valores numéricos necessários para obtenção do modelo completo do sistema foram encontrados mediante exaustivas experimentações práticas. O modelo matemático utilizado neste estudo é apresentado a seguir.

Da Fig.A.2 temos:

$$ml\ddot{\theta} + h(l\dot{\theta}) + mgsen(\theta) = \tau \tag{A.1}$$

onde m representa a massa do elo, l representa seu comprimento, g representa a constante gravitacional, θ representa a posição do elo e $h(l\dot{\theta})$ representa o coeficiente de atrito viscoso.

Considerando que a função não linear $h(l\dot{\theta})$, pode ser aproximada por $h(l\dot{\theta}) = kl\dot{\theta}$ temos:

$$ml\ddot{\theta} + kl\dot{\theta} + mgsen(\theta) = \tau \tag{A.2}$$

Dos experimentos realizados sobre o sistema físico real foram extraídos os seguintes dados:

$$\begin{aligned} m &= 0.210Kg \\ l &= 0.285m \\ g &= 9.820m/s^2 \\ k &= 0.300Nms/rad. \end{aligned}$$

Temos então

$$0.059\ddot{\theta} + 0.086\dot{\theta} + 2.06sen(\theta) = \tau, \tag{A.3}$$

comparando com o modelo de *Lagrange* podemos escrever:

$$\begin{aligned} M(q) &= 0.059, \quad V_m(q,\dot{q}) = 0 \\ F_v &= 0.085, \quad f_c(\dot{q}) = 0 \\ g(q) &= 2.06\sin(q). \end{aligned} \tag{A.4}$$

Este modelo respeita as Propriedades dos Manipuladores Robóticos apresentadas no *Capítulo 3*.

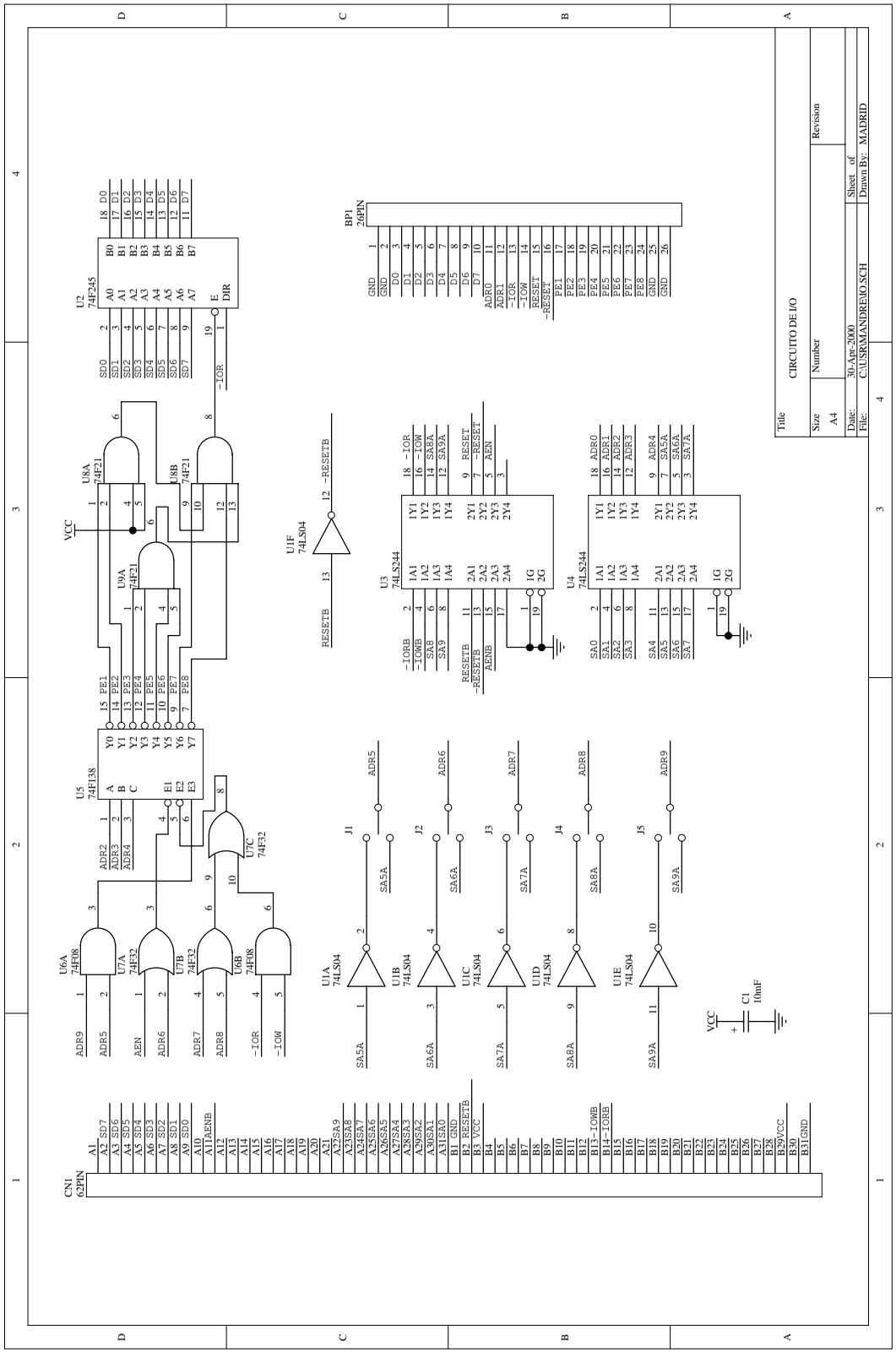

Figura A.3: Circuito de Interfaceamento Entrada/Saída.

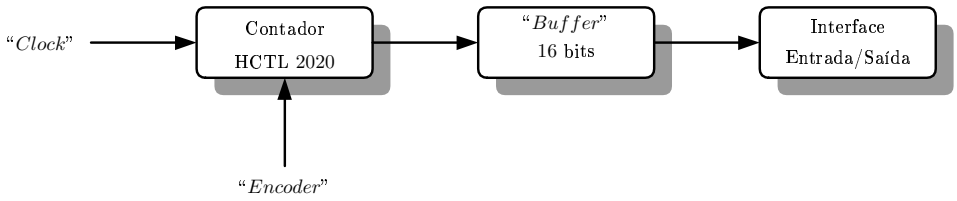

Figura A.4: Diagrama de Blocos do Circuito Contador *UP/DOWN*.

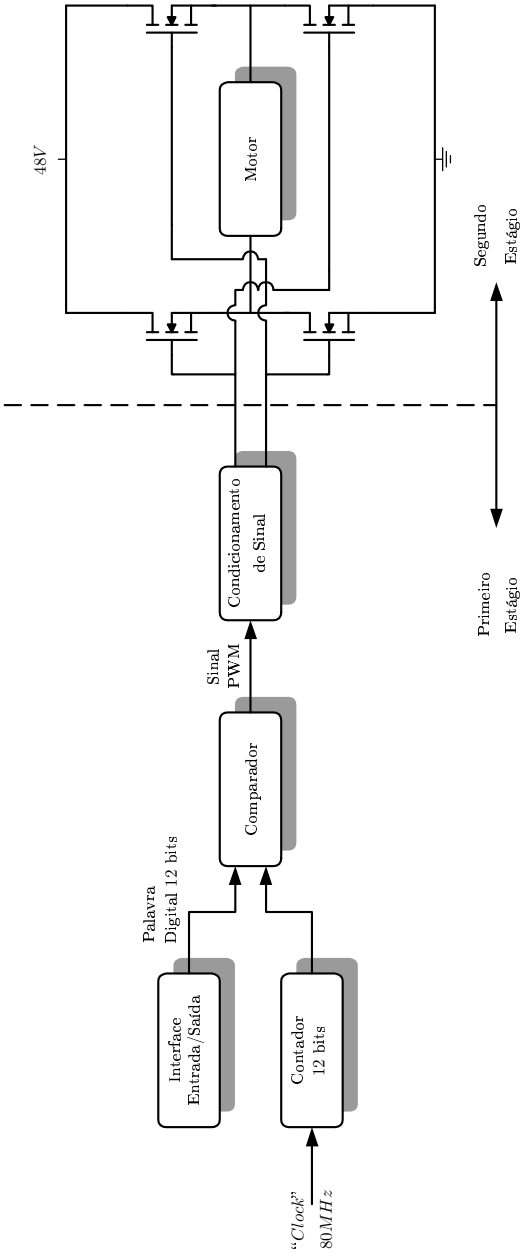

Figura A.5: Diagrama de Blocos do Circuito Amplificador de Potência.

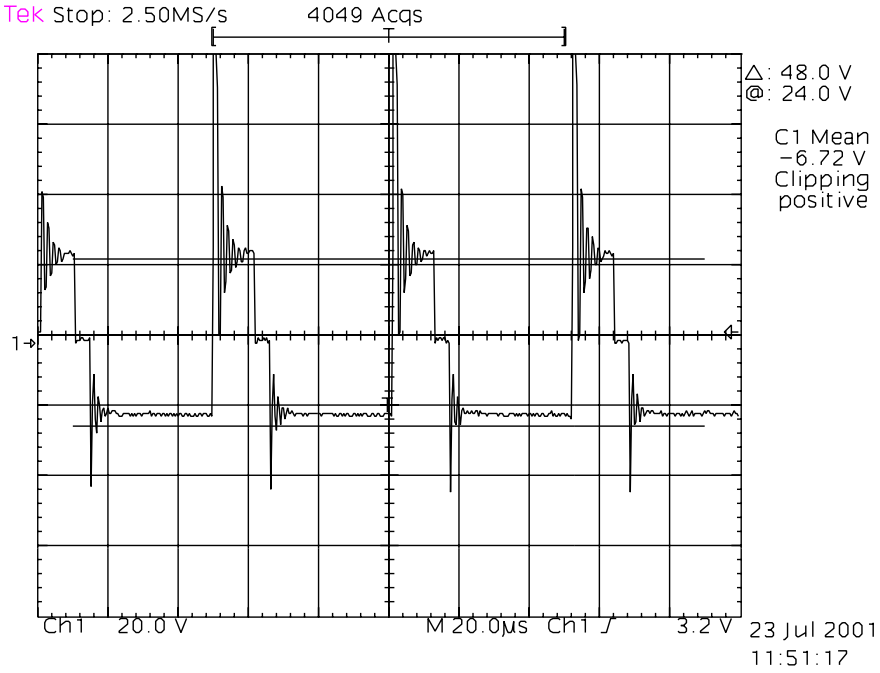

Figura A.6: Sinal de Tensão Aplicado ao Motor.

Apêndice B

Índice de Autores

Referências Bibliográficas

Belew, R., McInerney, J. e Schraudloph, N. (1990), 'Evolving networks: Using the genetic algorithm with connectionist learning', *CSE Technical Report CS90174, University of California* .

Bogdanov, A. e Timofeev, A. (1999), 'Robust optimal neural control of robots', *International Joint Conference on Neural Networks - IJCNN '99* **3**, 2026–2031.

Brown, R., Ruchti, T. e Feng, K. (1993), 'On identification of partially known dynamic nonlinear systems with neural network', *Proceedings of the 1993 IEEE International Symposium on Intelligent Control* pp. 499–504.

Cabrera, J. e Narendra, K. (1999), 'Issues in the application of neural networks for tracking based on inverse control', *IEEE Transactions on Automatic Control* **44**(11), 2007–2027.

Cerqueira, J., Badan, A. e Madrid, M. (2000), 'A nonlinear linear identifier based on artificial nerouns: Application for a robotic system', *Tutorial - Laboratório de Sistemas Modulares Robóticos - UNICAMP* .

Choi, B. e Bluff, K. (1995), 'Genetic optimization of control parameters of a neural network', *Proceedings Second New Zealand International Two-Stream Conference on Artificial Neural Networks and Expert Systems* pp. 174–177.

Commuri, S. e Lewis, F. (1997), 'Cmac neural network for control of nonlinear dynamical systems: Structure, stability and passivity', *Automatica* **33**(4), 635–641.

Fu, K., Gonzalez, R. e Lee, C. (1987), *Robotics: Control, Sensing, Vision, and Intelligence*, McGraw-Hill.

Gonzalez, R. e Woods, R. (2000), *Processamento de Imagens Digitais*, Edgard Blücher Ltda.

Haikin, S. (1994), *Neural Networks. A Coprehensive Foundation*, Macmillan Publihing Company.

Hewlett-Packard (1993), *Optoelectronics Designer's Catalog*, Hewlett-Packard Co.

Holland, J. (1975), *Adaption in Natural and Artificial Systems*, University of Michigan Press, An Arbor, MI.

Hunt, K., Sbarbaro, D., Zbikowski, R. e Gawthrop, P. (1992), 'Neural networks for control systems - a survey', *Automatica* **28**(6), 1083–1112.

Johansson, R. (1990), 'Quadratic optimization of motion coordination and control', *IEEE Transactions on Automatic Control* **35**(11), 1197–1208.

Khoukhi, A. (1999), 'An optimal design for robot dynamic control', *Proceedings of the IEEE International Symposium on Industrial Electronics* **2**, 861–866.

Kim, Y. e Lewis, F. (1999), 'Neural network output feedback control of robot manipulators', *IEEE Transactions on Robotics and Automation* **15**(2), 301–309.

Kim, Y., Lewis, F. e Dawson, D. (2000), 'Intelligent optimal control of robotic manipulators using neural networks', *Automatica* **36**(9), 1355–1364.

Kwan, C., Lewis, F. e Dawson, D. (1998), 'Robust neural network control of rigid link eletrically driven robots', *IEEE Transcations on Neural Networks* **9**(4), 581–588.

Levin, A. e Narendra, K. (1993), 'Control of nonlinear dynamical systems using neural networks: Controllability and stabilization', *IEEE Transactions on Neural Networks* **4**(2), 192–206.

Levin, A. e Narendra, K. (1996), 'Control of nonlinear dynamical systems using neural networks - part 2: Observability, identification, and control', *IEEE Transactions on Neural Networks* **7**(1), 30–42.

Lewis, F., Abdallah, C. e Dawson, D. (1993), *Control of Robot Manipulators*, Macmillan Publishing Company.

Lewis, F., Jagannathan, S. e Yesildirek, A. (1999), *Neural Network Control of Robot Manipulators and Nonlinear Systems*, Taylor and Francis Inc.

Lewis, F., Liu, K. e Yesildirek, A. (1995), 'Neural net robot controller with guaranteed tracking performance', *IEEE Transactions on Neural Networks* **6**(3), 703–715.

Lewis, F. e Syrmos, V. (1995), *Optimal Control*, 2nd edn, John Wiley and Sons, Inc.

Lin, F. e Brandt, R. (1996), 'An optimal control approach to robust control of robot manipulators', *Proceedings of the 1996 IEEE International Conference on Control Applications* pp. 31–36.

McCullagh, J., Choi, B. e Bluff, K. (1994), 'Genetic optimization of a neural network', *Proceedings of the Fifth Australian Conference on Neural Networks* pp. 109–112.

Michalewicz, Z. (1999), *Genetic Algorithms + Data Strutures = Evolution Programs*, Springer-Verlag.

Mital, D. e Chin, L. (1998), 'Application of neural networks in robotic control', *IEEE Region 10 International Conference on Global Connectivity in Energy, Computer, Communication and Control - TENCON '98.* **1**, 162–165.

Narendra, K. (1997), 'Neural networks for real-time control', *Proceedings of the 36th Conference on Decision and Control* pp. 1026–1031.

Narendra, K. e Parthasarathy, K. (1990), 'Identification and control of dynamical systems using neural networks', *IEEE Transactions on Neural Networks* **1**(1), 4–27.

Nicosia, S. e Tomei, P. (1990), 'Robot control by using only position measurements', *IEEE Transactions on Automatic Control* **35**, 703–715.

Rumelhart, D., Hinton, G. e Williams, R. (1986), *Learning Internal Representation by Error Propagating in Parallel Distributed Processing: Exploitation in the Microstructure of Cognition*, MIT Press, Cambridge, MA.

Sciavicco, L. e Siciliano, B. (1996), *Modeling and Control of Robot Manipulators*, McGraw-Hill International Editions.

Simon, A. (1971), *Mechanics*, McGraw Hill.

Sjöberg, J. e Ljung, L. (1995), 'Overtraining regularization and searching for a minimum, with application to neural networks', *International Journal of Control* **62**(3), 1391–1407.

Souza, M. (2000), Otimização de controladores nebulosos de takagi-sugeno utilizando algoritmos genéticos, Master's thesis, Universidade Federal de Campinas.

Thapa, B., Jones, B. e Zhu, Q. (2000), 'Nonlinear control with neural networks', *Fourth Internatinoal Conference on Knowledge-Based Intelligent Engineering Systems and Allied Technologies* pp. 868–873.