Universidade Estadual de Campinas
Faculdade de Engenharia Elétrica e de Computação

Modelagem de sistemas dinâmicos não lineares utilizando Sistemas *Fuzzy*, Algoritmos Genéticos e Funções de Base Ortonormal

Autor: Anderson Vinícius de Medeiros
Orientador: Prof. Dr. Wagner Caradori do Amaral
Co-orientador: Prof. Dr. Ricardo José Gabrielli Barreto Campello

Dissertação de Mestrado apresentada à Faculdade de Engenharia Elétrica e de Computação como parte dos requisitos para obtenção do título de Mestre em Engenharia Elétrica. Área de concentração: **Automação**.

Banca Examinadora

Akebo Yamakami, Dr. ..... DT/FEEC/Unicamp
Myriam Regattieri De Biase da Silva Delgado, Dr. ..... DAINF/CEFET-PR

Campinas, SP
Janeiro/2006

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

M468m

Medeiros, Anderson Vinícius de
Modelagem de sistemas dinâmicos não lineares utilizando sistemas *Fuzzy*, algoritmos genéticos e funções de base ortonormal / Anderson Vinícius de Medeiros. --Campinas, SP: [s.n.], 2006.

Orientadores: Wagner Caradori do Amaral, Ricardo José Gabrielli Barreto Campello
Dissertação (Mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Teoria dos sistemas dinâmicos. 2. Sistemas difusos. 3. Algoritmos genéticos. 4. Otimização matemática. 5. Identificação de sistemas. 6. Sistemas não-lineares. I. Amaral, Wagner Caradori do. II. Campello, Ricardo José Gabrielli Barreto. III. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. IV. Título.

Titulo em Inglês: Modeling of nonlinear dynamics systems using fuzzy systems, genetic algorithms and orthonormal basis functions
Palavras-chave em Inglês: Dynamic systems modeling, Takagi-Sugeno fuzzy model, Orthonormal basis functions, Optimization of models, genetic algorithms, Akaike information criterion, Similarity measures
Área de concentração: Automação
Titulação: Mestre em Engenharia Elétrica
Banca examinadora: Akebo Yamakami e Myriam Regattieri de Biase da Silva Delgado.
Data da defesa: 23/01/2006

Resumo

Esta dissertação apresenta uma metodologia para a geração e otimização de modelos *fuzzy* Takagi-Sugeno (TS) com Funções de Base Ortonormal (FBO) para sistemas dinâmicos não lineares utilizando um algoritmo genético. Funções de base ortonormal têm sido utilizadas por proporcionarem aos modelos propriedades como ausência de recursão da saída e possibilidade de se alcançar uma razoável capacidade de representação com poucos parâmetros. Modelos *fuzzy* TS agregam a essas propriedades as características de interpretabilidade e facilidade de representação do conhecimento. Enfim, os algoritmos genéticos se apresentam como um método bem estabelecido na literatura na tarefa de sintonia de parâmetros de modelos *fuzzy* TS.

Diante disso, desenvolveu-se um algoritmo genético para a otimização de duas arquiteturas, o modelo *fuzzy* TS FBO e sua extensão, o modelo *fuzzy* TS FBO Generalizado. Foram analisados modelos locais lineares e não lineares nos conseqüentes das regras *fuzzy*, assim como a diferença entre a estimação local e a global (utilizando o estimador de mínimos quadrados) dos parâmetros desses modelos locais. No algoritmo genético, cada arquitetura contou com uma representação cromossômica específica. Elaborou-se para ambas uma função de *fitness* baseada no critério de Akaike. Em relação aos operadores de reprodução, no operador de *crossover* aritmético foi introduzida uma alteração para a manutenção da diversidade da população e no operador de mutação gaussiana adotou-se uma distribuição variável ao longo das gerações e diferenciada para cada gene. Introduziu-se ainda um método de simplificação de soluções através de medidas de similaridade para a primeira arquitetura citada. A metodologia foi avaliada na tarefa de modelagem de dois sistemas dinâmicos não lineares: um processo de polimerização e um levitador magnético.

Palavras-chave: Modelagem de Sistemas Dinâmicos, Modelo *Fuzzy* Takagi-Sugeno, Funções de Base Ortonormal, Otimização de Modelos, Algoritmos Genéticos, Critério de Akaike, Medidas de Similaridade.

Abstract

This work introduces a methodology for the generation and optimization of Takagi-Sugeno (TS) fuzzy models with Orthonormal Basis Functions (OBF) for nonlinear dynamic systems based on a genetic algorithm. Orthonormal basis functions have been used because they provide models with properties like absence of output feedback and the possibility to reach a reasonable approximation capability with just a few parameters. TS fuzzy models aggregate to these properties the characteristics of interpretability and easiness to knowledge representation in a linguistic manner. Genetic algorithms appear as a well-established method for tuning parameters of TS fuzzy models.

In this context, it was developed a genetic algorithm for the optimization of two architectures, the OBF TS fuzzy model and its extension, the Generalized OBF TS fuzzy model. Local linear and nonlinear models in the consequent of the fuzzy rules were analyzed, as well as the difference between local and global estimation (using least squares estimation) of the parameters of these local models. Each architecture had a specific chromosome representation in the genetic algorithm. It was developed a fitness function based on the Akaike information criterion. With respect to the genetic operators, the arithmetic crossover was modified in order to maintain the population diversity and the Gaussian mutation had its distribution varied along the generations and differentiated for each gene. Besides, it was used, in the first architecture presented, a method for simplifying the solutions by using similarity measures. The whole methodology was evaluated in modeling two nonlinear dynamic systems, a polymerization process and a magnetic levitator.

Keywords: Dynamic Systems Modeling, Takagi-Sugeno Fuzzy Model, Orthonormal Basis Functions, Optimization of Models, Genetic Algorithms, Akaike Information Criterion, Similarity Measures.

À minha família e à minha namorada.

Agradecimentos

Acima de tudo, a Deus.

Ao meu orientador Prof. Wagner Caradori do Amaral pela orientação e compreensão com o ritmo do trabalho.

Ao meu co-orientador Prof. Ricardo J. G. B. Campello pela grande contribuição para o aprimoramento de toda a pesquisa.

À minha namorada Cynthia pela ajuda e companheirismo nas muitas noites, feriados e fins de semana investidos nesse mestrado.

A meus pais, Beto e Lúcia, pelos ensinamentos que formaram a base de toda minha educação.

Ao CNPq pelo apoio financeiro de parte dos trabalhos.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Siglas e Abreviações

ADN	Ácido Desoxirribonucléico
AG	Algoritmos Genéticos
AIC	Akaike Information Criterion
ARIMAX	Autoregressive Integrated Moving Average with Exogenous Inputs
ARMAX	Autoregressive Moving Average with Exogenous Inputs
ARX	Autoregressive with Exogenous Inputs
CSTR	Continuous Stirred Tank Reactor
EQM	Erro Quadrático Médio
FBO	Funções de Base Ortonormal
FIR	Finite Impulse Response
FP	Função de Pertinência
GFS	Genetic Fuzzy Systems
MDL	Minimum Description Length
MML	Minimum Message Length
MQ	Mínimos Quadrados
NARMAX	Nonlinear Autoregressive Moving Average with Exogenous Inputs
NARX	Nonlinear Autoregressive with Exogenous Inputs
NEFCON	Neural Fuzzy Controller

NFIR	Nonlinear Finite Impulse Response
PD	Proporcional Derivativo
PI	Proporcional Integral
RNA	Rede Neural Artificial
TS	Takagi-Sugeno

Capítulo 1

Introdução

Algoritmos genéticos (AG) têm sido aplicados na tarefa de otimização de sistemas *fuzzy* desde o início da década de 90. Sistemas *fuzzy* do tipo Takagi-Sugeno (TS), Mamdani ou relacional (além de redes neuro-*fuzzy*) foram projetados com fins de modelagem ou controle utilizando variações do algoritmo genético originalmente proposto por Holland no início dos anos 60. As pesquisas seguiram a tendência de otimizar cada vez mais parâmetros do sistema *fuzzy* com cada vez menos interação humana. Aos poucos foram analisados também outros critérios qualitativos para os sistemas obtidos, tais como interpretabilidade e simplicidade. Assim, não bastava a um modelo representar adequadamente determinado sistema real - era preciso desempenhar essa tarefa de maneira compreensível e simples. À medida que foram abordados sistemas mais complexos surgiram novas arquiteturas de otimização, envolvendo níveis hierárquicos e o conceito de co-evolução.

Em se tratando da obtenção de modelos *fuzzy*, um aspecto de especial interesse nas pesquisas recentes refere-se ao compromisso entre capacidade de representação do modelo *fuzzy* e sua complexidade. Neste sentido, foram definidas métricas para avaliar quão interpretável seria um modelo (em termos de simplicidade da base de regras ou disposição das funções de pertinência) face sua capacidade de representação de um determinado sistema real.

O presente trabalho aborda exatamente este compromisso. Incorpora-se aos modelos *fuzzy* TS com Funções de Base Ortonormal (FBO), propostos originalmente por Oliveira *et al* em 1999 [89], um método automático de otimização baseado em algoritmos genéticos. Este método permite a obtenção de modelos ao mesmo tempo simples (com um pequeno número de parâmetros, além de interpretável) e com satisfatória capacidade de representação. Mais especificamente, o algoritmo genético desenvolvido possibilita a otimização de duas arquiteturas, o modelo *fuzzy* TS FBO e sua extensão, o modelo *fuzzy* TS FBO Generalizado, proposto originalmente em [14, 15]. Para a primeira, composta por uma única base de funções ortonormais e referenciada no texto como arquitetura de

"pólo único"[1], otimiza-se um pólo real ou um par de pólos complexos conjugados (dependendo se as funções ortonormais utilizadas são de Laguerre ou de Kautz), o número de variáveis de entrada do modelo, a quantidade de funções de pertinência por variável de entrada e o número de funções na base ortonormal. Para a segunda arquitetura, otimiza-se um pólo complexo para cada modelo local, além do número de funções em cada base ortonormal. Em ambas, a disposição e configuração das funções de pertinência do modelo *fuzzy* também foram automaticamente determinadas. Note-se que o número de funções na base ortonormal (ou bases ortonormais) corresponde ao número de estados nos modelos locais do sistema *fuzzy*, influenciando diretamente a capacidade de representação do modelo.

As principais contribuições desta pesquisa residem na proposta de duas representações cromossômicas originais, na elaboração de uma função de *fitness* baseada no critério de informação de Akaike, na alteração do operador de *crossover* aritmético, no uso de um operador de mutação gaussiana com distribuições variáveis e distintas para cada gene e no uso de medidas de similaridade para a simplificação de funções de pertinência redundantes (e conseqüente redução da base de regras de inferência).

Esta dissertação está organizada da seguinte forma. Após esta introdução, o capítulo 2 apresenta o conceito de modelagem de sistemas dinâmicos e uma série de modelos lineares e não lineares. Alguns são apresentados apenas para contextualizar o trabalho (como as funções de transferência e as redes neurais) enquanto outros serão efetivamente utilizados nos capítulos posteriores (como os modelos em espaço de estados e os FBO lineares). Em particular, enfoca-se o modelo *fuzzy* TS, que constitui o pilar das arquiteturas sujeitas à otimização pelo AG.

Reserva-se o capítulo 3 para a apresentação detalhada dos modelos *fuzzy* TS FBO e *fuzzy* TS FBO Generalizado, obtidos com a utilização de funções de base ortonormal como filtros para a geração das variáveis lingüísticas (estados) do modelo *fuzzy* TS. As funções de base ortonormal possuem uma série de vantagens para a aplicação em modelagem de sistemas dinâmicos não lineares, como necessidade de um número menor de parâmetros para se atingir uma dada capacidade de representação e ausência de realimentação da saída do modelo. Neste capítulo discriminam-se quais parâmetros dos modelos serão otimizados autonomamente pelo AG. Além disso, descrevem-se duas variações do método de estimação de mínimos quadrados: estimação local e estimação global. A aplicação dessas duas variações na estimação dos coeficientes dos conseqüentes das regras de inferência do modelo *fuzzy* TS FBO Generalizado gera modelos com comportamentos bem distintos, os quais são analisados no capítulo 5. Como na arquitetura com pólo único a mesma base ortonormal está presente em todas as regras (dinâmica comum, diferenciando-se apenas as relações estáticas entre os estados das diversas regras), aplica-se apenas a estimação global dos parâmetros.

[1] Essa terminologia (pólo único) deverá ser interpretada como uma base única de funções ortonormais, parametrizada por um pólo real ou por um par de pólos complexos conjugados, em contraste com a extensão Generalizada, que será parametrizada por um número qualquer de pólos reais ou complexos conjugados.

O capítulo 4, o principal da dissertação, é dedicado à apresentação dos algoritmos genéticos, seus princípios, aplicações e fusão com sistemas *fuzzy*, formando os sistemas genético-*fuzzy*. Após uma revisão da evolução dos sistemas genético-*fuzzy*, o texto detalha cada componente ou etapa do AG, primeiramente de forma geral e então a implementação específica desta dissertação, seguindo a ordem: representação cromossômica, avaliação dos indivíduos que codificam os modelos, operador de seleção e operadores de reprodução (*crossover* e mutação). Por último, descreve-se um método para a eliminação de redundância de funções de pertinência no modelo *fuzzy* TS FBO com pólo único através do uso de medidas de similaridade. Tal método não é aplicado na arquitetura Generalizada pois a representação cromossômica elaborada para esta é menos flexível que aquela elaborada para a arquitetura com pólo único, conforme será justificado no capítulo 4. No decorrer de todo o capítulo apresentam-se as motivações e justificativas para o modo como foi projetado cada componente ou etapa do algoritmo genético.

Os resultados de modelagem são apresentados e analisados no capítulo 5. Aplica-se o método desenvolvido no capítulo 4 na tarefa de modelagem de dois sistemas dinâmicos não lineares com diferentes graus de complexidade: um processo de polimerização e um levitador magnético. Os resultados obtidos indicam a eficiência da metodologia proposta, possibilitando o levantamento de conjecturas sobre a aplicabilidade de cada arquitetura na modelagem de sistemas dinâmicos não lineares.

Finalmente, o capítulo 6 traça as conclusões sobre a pesquisa realizada e apresenta algumas perspectivas para sua continuação.

Capítulo 2

Modelagem de Sistemas Dinâmicos

O objetivo desse capítulo é descrever os principais conceitos relacionados ao contexto no qual se insere a presente dissertação. São apresentadas as bases teóricas que serão utilizadas em capítulos posteriores e os principais conceitos envolvidos. Dessa forma, parte-se de definições simples (como o conceito de sistema e modelo) e prossegue-se até a contextualização de todo o trabalho. Delega-se aos capítulos subseqüentes a tarefa de aprofundamento dos principais temas abordados.

2.1 O que é um Modelo?

Para se trabalhar de forma sistemática e eficiente com problemas que envolvem dependência temporal, é necessário que exista uma descrição das entidades ou processos envolvidos. Em uma definição simplista, tal descrição consiste em um modelo. A característica de dependência temporal relaciona-se ao fato de o sistema sendo analisado, definido primariamente como uma combinação de elementos que juntos desempenham alguma tarefa, ter seu comportamento dependente de variáveis externas que atuaram nele em algum momento no passado. No contexto deste trabalho, um modelo será definido como uma representação de um sistema dinâmico. Como será visto na seqüência, um modelo pode ser definido ou classificado de diversas outras formas.

Alguns dos objetivos para se construir um modelo são aumentar o conhecimento sobre um sistema, projetar um controlador, simular situações hipotéticas ou realizar predições sobre o comportamento futuro do sistema. Aplicações tão diversas vão requerer, conseqüentemente, modelos em formatos diferenciados. Modelos físicos, como uma maquete, são úteis na visualização de como os componentes físicos do sistema interagem entre si. Gráficos são outras formas de modelos. Eles podem apresentar relações de causa e efeito de maneira concisa e intuitiva. Porém, o tipo de modelo mais comum é o modelo matemático. Tais modelos podem ser obtidos através de métodos que vão desde a análise das leis físicas atuantes no sistema até uma abordagem puramente numérica, baseada

em um conjunto de dados experimentais.

A escolha do tipo do modelo depende do propósito definido para o mesmo e do tipo e quantidade de informação disponível sobre o sistema. No presente trabalho, como o objetivo é a obtenção de modelos que possam ser usados em sistemas de controle, buscar-se-á obter modelos matemáticos para os sistemas abordados. A próxima seção apresenta mais detalhes sobre as características de um modelo matemático.

2.1.1 Modelos Matemáticos

Muitas são as classificações para modelos matemáticos, não existindo um padrão unanimemente aceito. Descrevem-se na seqüência apenas as classificações mais relevantes para este trabalho.

Primeiramente, é possível distinguir *modelos lineares* de *modelos não lineares*. Seja o seguinte modelo matemático:

$$y = f(u) \tag{2.1}$$

Considera-se u a entrada do modelo, y sua saída e $f(.)$ a função que descreve o comportamento do sistema. Típicos sinais de entrada em sistemas seriam a força aplicada a uma massa, a voltagem alimentando um circuito elétrico ou a substância catalisadora de uma reação química. Em suma, são sinais de excitação. Geralmente, as entradas de um sistema são denotadas por $u_1(t)$, $u_2(t)$, ..., $u_n(t)$. Já as saídas, denotadas comumente por $y_1(t)$, $y_2(t)$, ..., $y_n(t)$, são as variáveis calculadas ou medidas, correspondendo ao efeito no sistema da entrada u. São exemplos a velocidade da massa após a aplicação da força, a carga em um capacitor ou a concentração do substrato da reação química. As notações exibidas referem-se ao tempo contínuo. No domínio discreto, utiliza-se normalmente o índice k (como $u(k)$ e $y(k)$).

Voltando à questão da linearidade, o modelo é dito linear se, para uma entrada $au_1(t) + bu_2(t)$ a saída é:

$$y = f(au_1(t) + bu_2(t)) = af(u_1(t)) + bf(u_2(t)) = ay_1 + by_2 \tag{2.2}$$

sendo a e b constantes, $u_1(t)$ e $u_2(t)$ entradas arbitrárias e $y_1 = f(u_1(t))$, $y_2 = f(u_2(t))$.

Qualquer relação ou modelo que não satisfaça (2.2) é não linear. Por exemplo, funções logarítmicas, trigonométricas ou potências tornam um modelo matemático não linear. A importância da linearidade em um modelo reside nas simplificações obtidas durante sua análise. Além disso, é possível em um modelo linear isolar os efeitos das variáveis de entrada e estudá-las separadamente.

No entanto, a maioria dos sistemas reais é não linear, caso não sejam feitas restrições quanto aos valores de suas entradas. A análise em determinadas faixas de operação pode vir a fornecer certo grau de linearidade ao modelo. Porém, dependendo da aplicação, a consideração da natureza não linear do sistema pode ser essencial nas tarefas sendo realizadas, não podendo ser descartada do modelo.

Todos os modelos obtidos nesse trabalho são não lineares.

Uma segunda forma de classificar modelos é baseada na característica da variável independente temporal. Um *modelo contínuo* é aquele no qual as variáveis são contínuas no tempo, ou seja, é possível acompanhar o sistema (e nele atuar) em qualquer instante de tempo. Esses modelos são geralmente descritos por equações diferenciais. Em contrapartida, os *modelos discretos* são aqueles nos quais as variáveis do sistema evoluem de uma forma "discretizada" em relação ao tempo agora discretizado. Os instantes de tempo nos quais as variáveis podem alterar seus valores podem corresponder ao momento no qual alguma amostragem é realizada ou a memória de um computador é lida. Modelos discretos são representados por equações a diferenças. Em engenharia, o fator mais sugestivo para o uso de modelos discretos é a disponibilidade de um computador digital para aquisição de dados ou implementação de um controlador. Os modelos abordados nessa dissertação são discretos.

Finalmente, algumas vezes há incerteza de natureza probabilística no valor de algum parâmetro do modelo. Se essa incerteza for significativa, um *modelo estocástico* deve ser usado. Nesse modelo, algumas variáveis são descritas em termos de uma função de distribuição de probabilidade, envolvendo, por exemplo, suas médias e variâncias. Os *modelos determinísticos*, por outro lado, não representam possíveis aleatoriedades presentes no sistema real. Assim, a saída em um dado instante para um modelo determinístico é completamente determinada a partir dos dados passados do modelo e de suas entradas. A característica de determinismo está presente nos modelos desenvolvidos no presente trabalho, os quais contemplam apenas incertezas de natureza possibilística (*fuzzy*).

Além destas características principais, pode-se ainda classificar um modelo como sendo a parâmetros distribuídos ou concentrados, variante ou invariante no tempo, autônomo ou não autônomo, não paramétrico ou paramétrico e multivariável ou monovariável [53]. Essa dissertação é caracterizada, em todos os pares anteriores, pelas segundas propriedades.

A partir da próxima seção serão descritos com mais detalhes uma série de modelos lineares e não lineares. Como afirmado no início deste capítulo, o objetivo aqui não será a construção de um material de referência sobre os inúmeros tipos de modelos existentes mas sim a descrição do contexto no qual está situada a presente pesquisa.

2.2 Modelos Lineares

Esta seção descreve alguns dos modelos lineares mais usuais. A seção 2.3 apresentará, por sua vez, modelos não lineares, quando serão explicitados aqueles que servirão como base para os capítulos posteriores.

2.2.1 Função de Transferência

A função de transferência de um sistema linear e invariante no tempo é definida como a razão entre as transformadas de Laplace da saída do sistema e de sua entrada, considerando-se todas as condições iniciais nulas [27]. A função de transferência de um sistema representa a relação entre as variáveis de entrada e saída do sistema, descrevendo a dinâmica do mesmo. Logo, não inclui nenhuma informação a respeito da estrutura e comportamento internos do sistema ou mesmo de sua estrutura física. De fato, as funções de transferência de sistemas físicos distintos podem ser idênticas.

Considere-se o sistema linear definido pela equação diferencial (2.3):

$$a_0 \overset{(n)}{y} + a_1 \overset{(n-1)}{y} + \ldots + a_{n-1}\dot{y} + a_n y = b_0 \overset{(m)}{u} + b_1 \overset{(m-1)}{u} + \ldots + b_{m-1}\dot{u} + b_m u, (n \geq m) \quad (2.3)$$

sendo y a saída do sistema e u sua entrada ($\overset{(n)}{y} \triangleq$ derivada de ordem n com relação ao tempo t). A função de transferência desse sistema é obtida tomando-se a transformada de Laplace de ambos os lados da equação (2.3), considerando-se todas as condições iniciais nulas, ou seja:

$$G(s) = \frac{\mathcal{L}\{y(t)\}}{\mathcal{L}\{u(t)\}} = \frac{Y(s)}{U(s)} = \frac{b_0 s^m + b_1 s^{m-1} + \ldots + b_{m-1}s + b_m}{a_0 s^n + a_1 s^{n-1} + \ldots + a_{n-1}s + a_n} \quad (2.4)$$

Assim, a função de transferência é uma representação algébrica em s de um sistema dinâmico [86].

Funções de transferência discretas são obtidas a partir das contínuas através de transformações baseadas em aproximações discretas para a derivada. Uma destas aproximações é denominada *método de Tustin*, definido como [3]:

$$s \leftarrow \frac{2(z-1)}{T_s(z+1)} \quad (2.5)$$

sendo T_s o período de amostragem. A notação $s \leftarrow z$ significa que s deve ser substituído por z. O mapeamento inverso, isto é, do domínio discreto para o contínuo, pode ser realizado diretamente ao se explicitar z na relação (2.5).

O conceito de função de transferência é importante na medida em que fornece um modelo matemático útil para utilização nas fases de análise e projeto de um controlador.

A função de transferência é uma propriedade do sistema em si, independente da magnitude e natureza dos sinais de entrada. Assim, caso a função de transferência de um sistema seja conhecida, sua saída pode ser estudada para vários sinais diferentes de entrada, obtendo-se dessa forma mais conhecimento sobre a natureza do sistema. Se este não for o caso, é possível obtê-la de forma experimental aplicando-se entradas conhecidas no sistema e analisando-se suas respostas. Um dos métodos aplicáveis nessa tarefa é o método de Sundaresan. Esse método permite obter os parâmetros de uma

função de transferência de segunda ordem com atraso puro de tempo desde que esteja disponível a saída do sistema a uma entrada degrau [3, 27].

2.2.2 Espaço de Estados

Uma alternativa às funções de transferência são os modelos em espaço de estados, os quais permitem explicitar as relações entre as variáveis internas do sistema. Para tal, é preciso identificar quais são as variáveis de estado do modelo, definidas como o conjunto de variáveis necessárias para descrever completamente o comportamento do sistema. Em outras palavras, conhecendo-se os valores das variáveis de estado é possível calcular a saída do sistema para qualquer instante futuro como uma função das variáveis de estado e dos valores atuais e futuros das entradas.

Considerando-se que as variáveis de estado do modelo são x_1, ..., x_n, a dinâmica de estados de um sistema linear invariante no tempo é dada pelo seguinte conjunto de equações diferenciais de primeira ordem:

$$\begin{aligned} \dot{x}_1 &= a_{11}x_1 + \ldots + a_{1n}x_n + b_{11}u_1 + \ldots + b_{1m}u_m \\ &\vdots \\ \dot{x}_n &= a_{n1}x_1 + \ldots + a_{nn}x_n + b_{n1}u_1 + \ldots + b_{nm}u_m \end{aligned} \tag{2.6}$$

e as saídas do sistema são fornecidas pelo sistema de equações (2.7):

$$\begin{aligned} y_1 &= c_{11}x_1 + \ldots + c_{1n}x_n + d_{11}u_1 + \ldots + d_{1m}u_m \\ &\vdots \\ y_p &= c_{p1}x_1 + \ldots + c_{pn}x_n + d_{p1}u_1 + \ldots + d_{pm}u_m \end{aligned} \tag{2.7}$$

sendo a_{ij}, b_{ij}, c_{ij} e d_{ij} coeficientes constantes.

Em notação matricial têm-se:

$$\begin{aligned} \dot{\mathbf{x}} &= A\mathbf{x} + B\mathbf{u} \\ \mathbf{y} &= C\mathbf{x} + D\mathbf{u} \end{aligned} \tag{2.8}$$

sendo $\mathbf{x} \in \Re^n$ o vetor de estados, $\mathbf{u} \in \Re^m$ o vetor de entrada e $\mathbf{y} \in \Re^p$ o vetor de saídas [27].

Para sistemas discretos é possível usar uma notação semelhante à da equação (2.8), bastando substituir o vetor de estado contínuo $\mathbf{x}$ por $\mathbf{x}(k)$, a saída $\mathbf{y}$ por $\mathbf{y}(k)$, a entrada $\mathbf{u}$ por $\mathbf{u}(k)$ e finalmente a derivada $\dot{\mathbf{x}}$ por $\mathbf{x}(k+1)$.

É preciso notar que a representação em espaço de estados de um sistema físico não é única, mas

depende da escolha das variáveis de estado. Algumas representações particularmente interessantes são a forma de Jordan e as formas canônicas controlável e observável. Existem procedimentos para a conversão de uma representação em outra, assim como para obter a função de transferência de um sistema partindo de seu modelo em espaço de estados [53].

A representação em espaço de estados é particularmente conveniente para sistemas multivariáveis. Além de suas vantagens em termos de cálculos computacionais, pode ser usada para adquirir maior percepção sobre o comportamento do sistema. Seu uso permite o projeto de sistemas de controle respeitando várias especificações de desempenho e ainda considerando as condições iniciais internas do sistema [87].

Como será visto no capítulo 3, os modelos *fuzzy* TS FBO Generalizados, que contituem um dos tópicos principais de interesse no presente trabalho, utilizam o conceito de representação em espaço de estados.

2.2.3 Representação Geral

A equação (2.9) fornece uma representação geral para toda uma classe de modelos lineares discretos [3, 72]. Alguns destes modelos são apresentados nas seções 2.2.4, 2.2.5 e 2.2.6

$$A(q)y(k) = q^{-n_k}\frac{B(q)}{F(q)}u(k) + \frac{C(q)}{D(q)}e(k) \tag{2.9}$$

O termo q^{-n_k} é introduzido para explicitar um possível atraso entre as dinâmicas de u e y (q^{-1} é o operador de atraso, ou seja, $q^{-1}x(k) = x(k-1)$). Os outros modelos são derivados deste a partir das diferentes atribuições dadas aos polinômios $A(q)$, $B(q)$, $C(q)$, $D(q)$ e $F(q)$. Embora o modelo (2.9) seja muito geral para aplicações práticas, o desenvolvimento de algoritmos e resultados teóricos para ele imediatamente engloba todos os casos especiais correspondentes a modelos mais realistas.

2.2.4 Resposta ao Impulso Finita

Sabe-se que um sistema linear, causal e invariante no tempo pode ser descrito por sua resposta ao impulso $g(\tau)$ da seguinte forma:

$$y(t) = \int_{\tau=0}^{\infty} g(\tau)u(t-\tau)d\tau \tag{2.10}$$

Conhecendo-se $g(\tau)|_{\tau=0}^{\infty}$ e $u(r)$ para $r \leq t$ é possível calcular a saída $y(r)$, $r \leq t$, para qualquer entrada. Assim, a resposta ao impulso caracteriza o sistema.

Ao se considerar o domínio discreto, a representação da resposta ao impulso é dada pelo somatório

de convolução:

$$y(k) = \sum_{i=0}^{\infty} g(i)u(k-i) \tag{2.11}$$

Caso o sistema seja assintoticamente estável é possível truncar o somatório infinito em um número máximo de termos L. Além disso, normalmente existem distúrbios atuando no sistema, como ruído de medição ou atuação de entradas não controladas. Para considerar de forma geral tais distúrbios o termo $e(k)$, representando ruído branco, é incluído na equação. Dessa forma, obtém-se o modelo de resposta ao impulso finita (FIR, do inglês *Finite Impulse Response*) [72].

$$y(k) = \sum_{i=0}^{L} g(i)u(k-i) + e(k) \tag{2.12}$$

2.2.5 ARX

Além da resposta ao impulso finita, outra forma simples de representar uma relação de entrada-saída em um sistema é através da seguinte equação a diferenças:

$$y(k) + a_1 y(k-1) + \ldots + a_{n_a} y(k-n_a) = b_1 u(k-1) + \ldots + b_{n_b} u(k-n_b) + e(k) \tag{2.13}$$

Como o termo de ruído $e(k)$ aparece na equação a diferenças, esse modelo é classificado como um modelo de erro na equação, ao contrário do modelo FIR, onde o erro é adicionado diretamente à saída $y(k)$.

Ao se introduzir a notação:

$$A(q) = 1 + a_1 q^{-1} + \ldots + a_{n_a} q^{-n_a}$$

e

$$B(q) = b_1 q^{-1} + \ldots + b_{n_b} q^{-n_b}$$

obtém-se o modelo autoregressivo com entradas externas (ou ARX, do inglês *AutoRegressive with eXogenous inputs*):

$$A(q)y(k) = B(q)u(k) + e(k) \tag{2.14}$$

O caso especial em que $n_a = 0$ resulta no modelo FIR. O modelo ARX é especialmente útil pois permite o uso de métodos simples porém poderosos de estimação dos parâmetros do modelo (regressão linear).

A principal desvantagem dos modelos ARX é a falta de liberdade na modelagem da dinâmica do

termo de ruído $e(k)$. Reescrevendo-se a equação (2.14) da seguinte forma:

$$y(k) = \frac{B(q)}{A(q)}u(k) + \frac{1}{A(q)}e(k) \tag{2.15}$$

observa-se que o ruído branco é filtrado por um filtro com a mesma dinâmica do sistema. Quando esta não for a representação mais fiel do sistema, o modelo ARMAX, descrito na seção seguinte, pode ser usado.

2.2.6 ARMAX

É possível adicionar flexibilidade ao modelo ARX ao se descrever a dinâmica do ruído como um processo de média móvel. A equação a diferenças desse novo modelo pode ser vista na equação (2.16):

$$\begin{aligned} y(k) + a_1 y(k-1) + \ldots + a_{n_a} y(k-n_a) &= b_1 u(k-1) + \ldots + b_{n_b} u(k-n_b) + e(k) \\ &+ c_1 e(k-1) + \ldots + c_{n_c} e(k-n_c) \end{aligned} \tag{2.16}$$

sendo $C(q) = 1 + c_1 q^{-1} + ... + c_{n_c} q^{-n_c}$. O modelo ARMAX, do inglês *AutoRegressive Moving Average with eXogenous inputs*, é escrito de forma compacta como:

$$A(q)y(k) = B(q)u(k) + C(q)e(k) \tag{2.17}$$

Assim como o modelo ARX, este é um modelo de erro na equação. O modelo ARMAX é considerado uma estrutura padrão usada em modelagem e controle de sistemas.

Uma variação com um integrador embutido no modelo é a representação ARIMAX, onde o I vem da operação de integração. O modelo ARIMAX é obtido substituindo-se $y(k)$ na equação (2.17) por $\Delta y(k) = y(k) - y(k-1)$. Tal variação é útil na descrição de sistemas com perturbações lentas [72].

2.2.7 Modelos FBO Lineares

Os modelos baseados em *Funções de Base Ortonormal*, FBO, serão mais detalhados no capítulo 3, em um contexto não linear. Por hora é suficiente afirmar que é possível desenvolver a resposta ao impulso do sistema $g(i)$, apresentada na equação (2.11), em uma base de funções ortonormais $\{\phi_j\}$, da seguinte forma [14]:

$$g(i) = \sum_{j=1}^{\infty} \alpha_j \phi_j(i) \tag{2.18}$$

sendo α_j escalares e $\phi_j(i)$ a i-ésima função ortonormal. Ao substituir $g(i)$ da equação (2.18) na equação (2.11), obtém-se:

$$\begin{aligned} y(k) &= \sum_{i=0}^{\infty}\sum_{j=1}^{\infty}\alpha_j\phi_j(i)u(k-i) \\ &= \sum_{j=1}^{\infty}\alpha_j\sum_{i=0}^{\infty}\phi_j(i)u(k-i) \\ &= \sum_{j=1}^{\infty}\alpha_j l_j(k) \end{aligned} \tag{2.19}$$

O sinal $l_j(k)$ é o resultado da filtragem linear da entrada $u(k-i)$ pelo filtro ortonormal caracterizado pela função ϕ_j. Pode-se truncar o último somatório da equação (2.19) em um número N de funções ortonormais, obtendo-se assim o modelo linear FBO [14]:

$$\hat{y}(k) = \sum_{j=1}^{N}\alpha_j l_j(k) \tag{2.20}$$

Uma das vantagens deste modelo é a possibilidade de se incorporar conhecimento sobre o sistema na base de funções ortonormais $\{\phi_j\}$. O efeito imediato é a sensível redução do número de parâmetros do modelo. A seção 3.1 apresentará outras características e vantagens dos modelos FBO, enquanto a seção 3.2.3 contextualizará sua aplicação no presente trabalho.

2.3 Modelos Não Lineares

Como afirmado na seção 2.1.1, muitas vezes faz-se necessário trabalhar com modelos não lineares. Nestas ocasiões é necessário considerar características intrínsecas importantes do sistema, que de outra forma sofreriam aproximações (linearização). Esta seção apresenta uma série de modelos não lineares comumente utilizados nas tarefas de modelagem e controle de sistemas dinâmicos.

2.3.1 NARX e NARMAX

Os modelos não lineares NARX e NARMAX (com o N inicial vindo do inglês *nonlinear*) são extensões de seus equivalentes lineares. O modelo NARX descreve a saída do sistema como uma função não linear dos termos passados da entrada e da saída, ou seja:

$$y(k) = \mathcal{F}[y(k-1),\ldots,y(k-n_a),u(k-1),\ldots,u(k-n_b)] \tag{2.21}$$

O modelo NARMAX inclui ainda os termos da dinâmica da perturbação atuando no sistema. Assim, tem-se:

$$y(k) = \mathcal{F}[y(k-1), \ldots, y(k-n_a), u(k-1), \ldots, u(k-n_b), e(k), \ldots, e(k-n_c)] \tag{2.22}$$

Nestas duas equações, n_a, n_b e n_c representam os maiores atrasos da saída, da entrada e do ruído, respectivamente, considerados no modelo, e $\mathcal{F}$ é uma função não linear.

Algumas realizações especiais da função $\mathcal{F}[.]$ permitem a obtenção de modelos lineares nos parâmetros, mesmo com regressores não lineares. Uma delas é a polinomial, que para o modelo NARX pode ser escrita como [3]:

$$y(k) = \sum_{m=0}^{l} \sum_{p=0}^{m} \sum_{n_1,n_m}^{n_a,n_b} c_{p,m-p}(n_1, \ldots, n_m) \prod_{i=1}^{p} y(k-n_i) \prod_{i=p+1}^{m} u(k-n_i) \tag{2.23}$$

sendo que nessa representação os produtórios são por definição unitários quando o índice final é menor que o inicial.

Nesta equação, l é o grau de não linearidade da função $\mathcal{F}[.]$, n_a e n_b são os máximos atrasos considerados da saída e da entrada, respectivamente, e $c_{p,m-p}$ são os parâmetros escalares que multiplicam os diferentes termos do modelo. Embora esse tipo de representação para a função $\mathcal{F}[.]$ possibilite o uso de métodos eficientes de estimação dos parâmetros do modelo, persiste um grande problema que é a definição da estrutura do modelo. Nessa tarefa, deverão ser estabelecidos os maiores atrasos n_a e n_b (e n_c, em um modelo NARMAX) e o grau de não linearidade l do modelo. A complexidade desta tarefa reside no fato de que para pequenos atrasos e graus de não linearidade o número de parâmetros no modelo cresce rapidamente. Este assunto será abordado na seção 3.1.

2.3.2 Modelo de Volterra

Por se tratar de uma generalização da resposta ao impulso, que é um modelo linear, a série de Volterra possibilita o uso dos conceitos e modos de pensar de sistemas lineares em sistemas não lineares. Isso é uma grande vantagem face os grandes avanços teóricos conquistados no tratamento de sistemas lineares tanto no domínio do tempo quanto no da freqüência [33].

No domínio do tempo, um modelo baseado na série de Volterra, ou simplesmente modelo de Volterra, é dado pela série infinita:

$$y(t) = y_1(t) + y_2(t) + \ldots + y_i(t) + \ldots \tag{2.24}$$

sendo o termo $y_1(t)$ definido exatamente como na equação (2.10) e os demais termos:

$$\begin{aligned} y_2 &= \int_0^\infty \int_0^\infty g_2(\tau_1, \tau_2) u(t-\tau_1) u(t-\tau_2) d\tau_1 d\tau_2 \\ &\vdots \\ y_i &= \int_0^\infty \dots \int_0^\infty g_i(\tau_1, \dots, \tau_i) u(t-\tau_1) \dots u(t-\tau_i) d\tau_1 \dots d\tau_i \\ &\vdots \end{aligned} \tag{2.25}$$

Como o termo $y_i(t)$ nessa expansão contém contribuições de ordem i da entrada $u(t)$, o modelo de Volterra representa uma generalização não linear de ordem i da integral de convolução linear que caracteriza completamente sistemas lineares invariantes no tempo. A equação (2.25) já supõe que o sistema é causal (limite inferior das integrais iguais a zero). A função $g_i(\tau_1, \dots, \tau_i)$ é denominada *kernel* de Volterra de ordem i, sendo i o grau de não linearidade do sistema. Assim, cada *kernel* g_i corresponde à generalização de dimensão i da resposta ao impulso de um sistema linear.

O modelo discreto de Volterra é obtido substituindo-se as integrais de convolução pelos somatórios de convolução. Em uma notação mais compacta, tem-se:

$$y(k) = \sum_{j=1}^{\infty} \sum_{\tau_1}^{\infty} \dots \sum_{\tau_j}^{\infty} g_j(\tau_1, \dots, \tau_j) \prod_{i=1}^{j} u(k-\tau_i) \tag{2.26}$$

Assim, embora tenha algumas limitações, o modelo de Volterra discreto representa uma extensão simples e lógica dos modelos lineares FIR. Além disso, sua estrutura é conveniente para o desenvolvimento de uma série de estratégias de controle. A principal limitação dos modelos de Volterra está relacionada ao elevado número de parâmetros que devem ser estimados, mesmo para modelos de baixa ordem e pequeno grau de não linearidade. Uma abordagem para o tratamento desse problema é o desenvolvimento dos coeficientes do modelo em termos de funções de base ortonormal, as quais serão detalhas na seção 3.1. O uso dessa técnica permite reduzir a dimensão do modelo mantendo a sua capacidade de representação [28].

2.3.3 Redes Neurais

As Redes Neurais Artificiais (RNA) são sistemas paralelos inspirados na estrutura física do cérebro humano. São compostos por unidades de processamento simples (nós) que computam determinadas funções matemáticas, normalmente não lineares, chamadas funções de ativação. Tais unidades são dispostas em uma ou mais camadas e interligadas por um grande número de conexões, simulando as ligações sinápticas no cérebro humano. Na maioria dos modelos estas conexões estão associadas a

pesos, os quais servem para ponderar a entrada recebida por cada nó da rede, codificando o conhecimento embutido no modelo [40].

O primeiro modelo de neurônio artificial foi proposto por McCulloch e Pitts em 1943 [75]. Trata-se de uma simplificação do que se sabia a respeito do neurônio biológico naquela época. A sua descrição matemática resultou em um modelo com n terminais de entrada, $x_1, x_2, \ldots, x_n$ e apenas um terminal de saída y. A saída y é uma função do somatório das entradas ponderadas pelos pesos correspondentes, tal qual indicado na Figura 2.1.

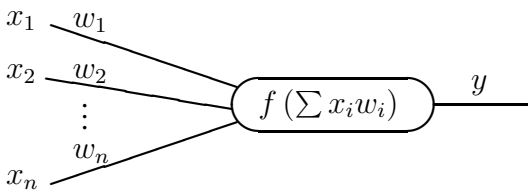

Figura 2.1: Modelo do neurônio de McCulloch e Pitts.

A função não linear $f(.)$ da Figura 2.1 dita se o neurônio foi ativado pelas entradas ou não. Normalmente é uma função degrau ou sigmoidal [4].

Frank Rosenblatt introduziu o conceito de "aprendizado" em redes neurais. O modelo proposto por Rosenblatt, conhecido como *perceptron*, era composto por uma estrutura de rede em camadas tendo como unidades básicas nós do tipo McCulloch e Pitts e uma regra de aprendizado [96].

A capacidade de representação das redes neurais está relacionada à sua arquitetura. As redes de uma camada resolvem apenas problemas linearmente separáveis. Para problemas não linearmente separáveis deve-se usar redes com uma ou mais camadas intermediárias (Figura 2.2). Com uma camada intermediária uma rede neural pode implementar qualquer função contínua em um domínio compacto [19, 20].

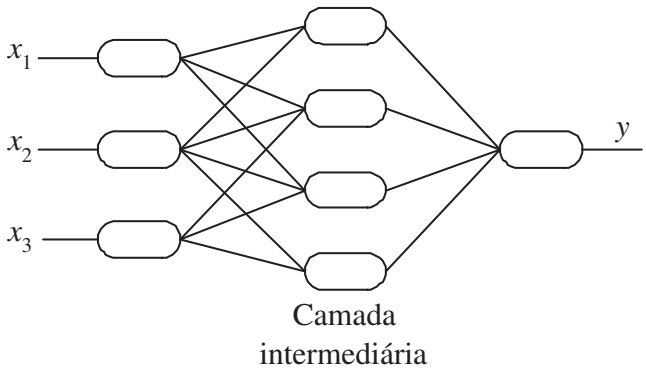

Figura 2.2: Exemplo de arquitetura de uma rede *perceptron* multicamadas.

Como uma rede neural não é linear em seus parâmetros, algoritmos especiais de treinamento devem ser usados. O algoritmo de aprendizado mais conhecido para treinamento de uma rede neural *perceptron* multicamadas é o *backpropagation* [10]. A maioria dos métodos de aprendizado para redes neurais desse tipo utiliza variações deste algoritmo.

O *backpropagation* é um algoritmo supervisionado que utiliza pares (entrada, saída desejada) para, através de um mecanismo de correção de erros, ajustar os pesos da rede. O treinamento ocorre em duas fases, *forward* e *backward*, onde cada fase percorre a rede em um sentido.

No contexto de modelagem de sistemas dinâmicos, foram já estudadas diversas arquiteturas para uma rede neural. Em particular, é possível estender os modelos FIR, ARX e ARMAX para os modelos NNFIR, NNARX e NNARMAX, sendo NN o acrônimo em inglês para rede neural (*neural network*). Esses modelos traduzem o fato de as entradas da rede neural serem compostas pelos termos regressores dos respectivos modelos lineares [85].

2.4 Modelos *Fuzzy*

Esta seção, ao contrário das anteriores, é mais longa e detalhada, uma vez que os modelos *fuzzy* são um dos pilares deste trabalho.

A seção 2.4.1 apresenta uma introdução a sistemas *fuzzy* e suas possíveis aplicações. A seção 2.4.2 detalha os conceitos envolvidos em um sistema *fuzzy*, como variáveis lingüísticas e base de regras de inferência. Finalmente, a seção 2.4.3 introduz as duas arquiteturas mais comuns de modelos *fuzzy*: *Fuzzy* Mamdani [73] e *Fuzzy* Takagi-Sugeno (*Fuzzy* TS) [104]. Além destas duas arquiteturas, existe ainda o modelo *fuzzy* relacional [29], não tratado neste trabalho.

2.4.1 Princípios e Aplicações

De concepção recente, a lógica *fuzzy* é uma das técnicas da grande área da Inteligência Computacional. Seus conceitos foram elaborados por Lofti Zadeh em meados de 1960 na Universidade de Berkeley, na Califórnia. Está inclusa na classe Cognitiva ou Simbólica. Existem ainda a Conexionista e a Evolucionista, cujas maiores representações são, respectivamente, as Redes Neurais e os Algoritmos Genéticos.

Não segue a lógica concebida por Aristóteles, filósofo grego (384 - 322 a.C.), que leva a uma linha de raciocínio lógico baseado em premissas e conclusões que devem ser necessariamente ou verdadeiras ou falsas. Além disso, dada uma declaração "*X*", então "*X e não X*" cobre todas as possibilidades. A lógica *fuzzy* viola estas suposições, admitindo que ao mesmo tempo uma dada afirmação seja parcialmente verdadeira e parcialmente falsa.

O conceito de dualidade, estabelecendo que algo pode e deve coexistir com o seu oposto, faz a lógica *fuzzy* parecer natural, até mesmo inevitável. Muitas das experiências humanas não podem ser classificadas simplesmente como verdadeiras ou falsas. Entre a certeza de ser e a certeza de não ser, existem infinitos graus de incerteza. Esta imperfeição intrínseca à informação representada numa linguagem natural foi tratada matematicamente no passado com o uso da teoria das probabilidades.

Pode-se definir a lógica *fuzzy* descrevendo a motivação para a sua criação. Necessitava-se de uma ferramenta capaz de capturar informações vagas, aproximadas, em geral descritas em uma linguagem natural e expressá-las de uma maneira sistemática. O passo seguinte seria convertê-las para um formato numérico, manipulável por um computador. Outra definição a colocaria como uma lógica que suporta os modos de raciocínio que são aproximados, ao invés de exatos, como estamos naturalmente acostumados a trabalhar. Para isso, utiliza-se uma abordagem multivalorada.

Um exemplo com essas características são as tarefas de tomadas de decisão. Muitas variáveis não são definidas em termos exatos. O uso da lógica *fuzzy* pode significar minimização de custos por facilitar a implementação dessas estratégias ou justificar as ações tomadas.

O controle de processos industriais foi a área pioneira de utilização da lógica *fuzzy*. As primeiras experiências datam de meados da década de 70 ([73] e referências inclusas). Hoje em dia, uma grande variedade de aplicações comerciais e industriais está disponível, destacando-se neste cenário o Japão e, mais recentemente, os EUA e a Alemanha. Dentre os exemplos típicos incluem-se produtos de consumo, tais como geladeiras (Sharp), ar condicionado (Mitsubishi), câmeras de vídeo (Canon, Panosonic), máquinas de lavar roupa (Sanyo), aspiradores de pó, etc. Na indústria automotiva destacam-se transmissões automáticas (Nissam, Lexus), injeção eletrônica, suspensão ativa e freios antibloqueantes. Sistemas industriais incluem controle de grupo de elevadores (Hitachi, Toshiba), veículos autoguiados e robôs móveis (Nasa, IBM), controle de motores (Hitachi), ventilação de túneis urbanos (Toshiba), controle de tráfego urbano e controle de partida e parada de metrôs (Tókio). Estas citações são ilustrativas, pois correntemente já foram anunciadas mais de 1000 patentes envolvendo lógica *fuzzy* [107].

Apesar do uso e da aplicação no Brasil ser incipiente, várias indústrias e empresas vêm desenvolvendo produtos e serviços utilizando de alguma forma os conceitos *fuzzy* (Villares, IBM, Klockner & Moeller, Robertshaw, Yokogawa, HI Tecnologia) [77].

Controladores baseados em lógica *fuzzy* possuem propriedades interessantes de confiabilidade e robustez. Sistemas convencionais processam equações complexas em seqüência. Caso ocorra algum erro em alguma delas, a discrepância final obtida pode ser significativa. Devido ao processamento independente de cada regra do controlador *fuzzy*, o efeito de um problema intermediário é amortecido. Isso implica que uma falha parcial do sistema não deteriora a performance do controlador como um todo.

A teoria dos conjuntos *fuzzy* pode ainda ser usada na modelagem de sistemas dinâmicos. Uma série de arquiteturas *fuzzy* foram já demonstradas como aproximadores universais [93]. Isto significa que são capazes de aproximar qualquer função contínua em um domínio compacto com qualquer nível de precisão desejado.

Além da propriedade interessante de aproximação universal dos modelos *fuzzy*, estes ainda adicionam uma nova dimensão à informação contida no modelo. Trata-se da dimensão lingüística, fornecendo descrições intuitivas sobre o comportamento do sistema modelado ([32]).

2.4.2 Elementos de um Sistema *Fuzzy*

A lógica *fuzzy* permite o tratamento de implicações lógicas seguindo regras naturais de raciocínio, analisando condições e estipulando conseqüências [100, 106]. É baseada na teoria dos conjuntos *fuzzy* [117].

Um conjunto *fuzzy* é definido como uma coleção de objetos cujo grau de pertinência ao conjunto varia entre zero e um. Formalmente, um conjunto *fuzzy* é caracterizado por uma função de pertinência (FP) que mapeia os elementos do domínio de discurso $\mathcal{U}$ no intervalo unitário $[0, 1]$, ou seja:

$$A : \mathcal{U} \rightarrow [0, 1] \tag{2.27}$$

Assim, o valor $A(x)$ indica com que grau o elemento x pertence ao conjunto A [93]. Um grau de pertinência 1 equivale ao clássico símbolo de pertinência $\in$ enquanto um grau de pertinência 0 equivale ao clássico símbolo $\notin$.

Um conceito relacionado com conjuntos *fuzzy* é o de variável lingüística. Entende-se por variável um identificador que pode assumir um dentre vários valores. Deste modo, uma variável lingüística pode assumir um valor lingüístico dentre vários outros em um conjunto de termos lingüísticos. Cada termo lingüístico é caracterizado por uma função de pertinência. A expressão lingüística das variáveis tratadas se dá através de predicados mnemônicos, como "erro", "temperatura" ou "variação da pressão". É permitido ainda o uso de modificadores qualificadores: "pequeno positivo", "muito grande negativo" ou "aproximadamente zero".

Em princípio, as funções de pertinência podem ser qualquer função que produza valores entre 0 e 1. Comumente estas são definidas como triangulares, trapezoidais, sigmóides ou gaussianas (como exibido na Figura 2.3). Funções de pertinência do tipo *singleton* também são usadas. As FP adotadas neste trabalho são gaussianas[1], descritas pela equação (2.28):

$$A(x) = \exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right) \tag{2.28}$$

[1] A justificativa para esta escolha será apresentada na seção 4.4.1.

com μ correspondendo ao centro da FP e σ à sua abertura.

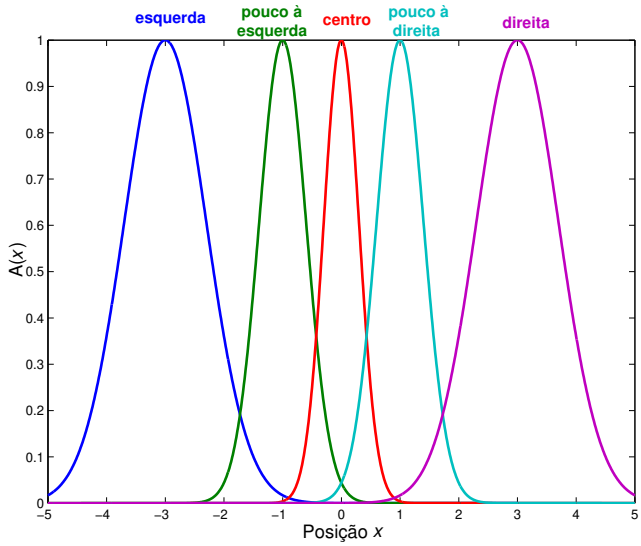

Figura 2.3: Exemplo de funções de pertinência gaussianas.

A definição satisfatória da quantidade e grau de sobreposição entre as funções de pertinência no universo de discurso é fundamental quando da aplicação de um sistema *fuzzy* [100]. Essa discussão será retomada nos capítulos 3 e 4.

Em um sistema *fuzzy* (seja um modelo *fuzzy* de um processo ou um controlador *fuzzy*) as funções de pertinência são responsáveis pela conversão das grandezas do domínio do mundo real, captadas por sensores, dispositivos computadorizados ou mesmo provenientes de outros segmentos do processo de controle, para números *fuzzy*. Essa conversão, conhecida como "*fuzzificação*", é essencial para a atuação da máquina de inferência *fuzzy*.

A etapa de inferência consiste em se avaliar um conjunto de regras (a base de conhecimento) do tipo "*se* ... *então* ..." que descrevem a dependência entre as variáveis lingüísticas de entrada e as de saída. Essas regras seguem o paradigma *modus ponens*, um mecanismo de inferência progressiva (ao contrário do *modus tollens*, regressivo, mais utilizado em sistemas especialistas), para fazer o mapeamento do conceito de implicação lógica.

A seção seguinte apresenta dois exemplos de bases de regras *fuzzy*, uma para o modelo *fuzzy* Mamdani e outra para o modelo *fuzzy* TS. Como será observado, os antecedentes das regras possuem a mesma estrutura geral, qual seja:

$$\begin{aligned} \text{se} \quad & (u_1 \text{ é } A_{1,1}) \text{ e } \ldots \text{ e } (u_n \text{ é } A_{1,n}) \\ & \vdots \end{aligned} \tag{2.29}$$

$$\text{se} \quad (u_1 \text{ é } A_{m,1}) \text{ e } \ldots \text{ e } (u_n \text{ é } A_{m,n})$$

sendo $u_1, \ldots, u_n$ as entradas do sistema e $A_{1,1}, \ldots, A_{m,n}$ os termos lingüísticos dessas variáveis, definidos como funções de pertinência que cobrem todo o universo de discurso.

A inferência consiste de dois passos: avaliação da premissa de cada regra (conjunção), através dos operadores t-norma, e em seguida a etapa de agregação, ponderando as diferentes conclusões das regras ativas sob o operador s-norma [100, 106]. Os operadores de t-norma e s-norma são normas e co-normas triangulares que fornecem métodos genéricos para as operações de intersecção e união em conjuntos *fuzzy*. Alguns exemplos de t-normas são o produto e o operador de mínimo. Para as s-normas têm-se como exemplos o operador de máximo ou a soma drástica [93].

Um sistema *fuzzy* contém um conjunto dessas regras, todas ativadas em paralelo. Assim, o sistema *fuzzy* raciocina com inferência associativa paralela. Quando uma entrada é fornecida, o controlador dispara as regras paralelamente, com diferentes graus de ativação, para inferir um resultado ou saída.

Em algumas arquiteturas *fuzzy*, após a inferência da ação a ser tomada, necessita-se de uma tradução do valor lingüístico para a variável numérica de saída, que pode representar funções como ajustar a posição de um botão ou acionar uma válvula. Este passo é conhecido como "*defuzzificação*". Como pode acontecer de saídas distintas serem acionadas num mesmo momento, com diferentes graus de ativação, deve-se encontrar o valor que melhor corresponda à distribuição de possibilidade da combinação dos conjuntos *fuzzy* de saída. A seção seguinte fornece um exemplo de um método de "*defuzzificação*".

2.4.3 Arquiteturas *Fuzzy* dos Tipos Mamdani e Takagi-Sugeno

A diferença entre os modelos *fuzzy* Mamdani e Takagi-Sugeno reside nos conseqüentes das regras da base de regras de inferência. Em um modelo Mamdani, as regras são do tipo:

$$\begin{aligned} &\text{se} \quad (u_1 \text{ é } A_{1,1}) \text{ e } \ldots \text{ e } (u_n \text{ é } A_{1,n}) \text{ então } y \text{ é } B_1 \\ &\vdots \\ &\text{se} \quad (u_1 \text{ é } A_{m,1}) \text{ e } \ldots \text{ e } (u_n \text{ é } A_{m,n}) \text{ então } y \text{ é } B_r \end{aligned} \tag{2.30}$$

sendo $B_1, \ldots, B_r$ os termos lingüísticos para a saída y (neste exemplo, única) definidos em um domínio $\mathcal{Y}$ por meio de funções de pertinência.

Cada regra acima define uma região *fuzzy* no espaço de entrada-saída. O conjunto de todas as regras particionam esse espaço em regiões *fuzzy* sobrepostas.

Esse modelo em particular requer, após o processo de inferência *fuzzy* da saída, a aplicação da etapa de "*defuzzificação*" citada na seção anterior. O método do *centro da área* é uma possibilidade

[100], porém este requer um considerável esforço computacional, pois surge em sua definição o cálculo numérico de integrais.

Uma segunda abordagem para a "*defuzzificação*" consiste no *centro dos máximos* [100]. Trata-se de calcular a média ponderada entre os valores de cada termo da variável lingüística de saída. Tal método é descrito pela equação (2.31), sendo $\hat{y}$ a saída final do sistema *fuzzy*, y_i a saída intermediária de cada uma das r regras e w_i o peso da i-ésima regra [93].

$$\hat{y} = \frac{\sum_{i=1}^{r} w_i y_i}{\sum_{i=1}^{r} w_i} \tag{2.31}$$

Como os antecedentes das regras apresentados em (2.31) utilizam o operador de conjunção "e", os pesos w_i de cada regra (seus níveis de ativação) são calculados através da operação de t-normas. Para isso, primeiramente computa-se o grau de pertinência $A_{s,j}(u_j)$ de cada entrada u_j à respectiva função de pertinência $A_{s,j}$. Considerando como exemplo o uso da t-norma do valor mínimo, então cada w_i é calculado segundo a equação (2.32) [38]:

$$w_i = \min\{A_{s,1}(u_1), \ldots, A_{s,n}(u_n)\} \tag{2.32}$$

No caso da t-norma produto, o nível de ativação w_i é calculado como [93]:

$$w_i = \prod_{j=1}^{n} A_{s,j}(u_j) \tag{2.33}$$

Além disso, o valor de y_i na equação (2.31) é escolhido como o valor para o qual $B_i(y_i)$ é máximo. Em funções de pertinência gaussianas ou triangulares, por exemplo, trata-se do valor modal.

Este método atende um requisito essencial em aplicações em controle, a continuidade [100]. Isso significa que uma mudança infinitesimal em uma variável de entrada não causa uma variação abrupta em nenhuma das variáveis de saída.

No modelo *fuzzy* TS, a base de regras é da forma:

$$\begin{aligned} &\text{se} \quad (u_1 \text{ é } A_{1,1}) \text{ e } \ldots \text{ e } (u_n \text{ é } A_{1,n}) \text{ então } y = f_1(u_1, \ldots, u_n) \\ &\vdots \\ &\text{se} \quad (u_1 \text{ é } A_{m,1}) \text{ e } \ldots \text{ e } (u_n \text{ é } A_{m,n}) \text{ então } y = f_r(u_1, \ldots, u_n) \end{aligned} \tag{2.34}$$

Assim, os conseqüentes das regras são agora compostos por uma função qualquer das variáveis de

entrada. Embora na equação (2.34) apareçam nas premissas das regras todas as variáveis de entrada, esta não é uma exigência da arquitetura *fuzzy* TS. Pode ser usado um número menor de variáveis nas premissas, por exemplo com o objetivo de diminuir o tamanho da base de regras. O modelo *fuzzy* TS combina uma descrição global baseada em regras com uma aproximação local que, no contexto de identificação de sistemas, é normalmente escolhida como um modelo de regressão linear. No caso de as funções f_i serem funções afins, por exemplo, o conseqüente da i-ésima regra seria da forma:

$$y = a_{i,1}u_1 + \ldots + a_{i,n}u_n + a_{i,n+1} \tag{2.35}$$

O modelo *fuzzy* TS resultante é fácil de se identificar, pois além de cada regra descrever uma região *fuzzy* na qual as saídas dependem das entradas de forma linear, os parâmetros dos conseqüentes $a_{i,1}, \ldots, a_{i,n+1}$ podem ser facilmente estimados, por exemplo, via mínimos quadrados[2].

Como as saídas das regras não estão definidas através de termos lingüísticos, a etapa de "*defuzzificação*" não é necessária após o processo de inferência *fuzzy*. A saída final $\hat{y}$ é calculada da mesma forma que na equação (2.31), com a diferença que o valor de cada y_i é o resultado direto do conseqüente da i-ésima regra.

Assim como apresentado para as redes neurais da seção anterior, modelos *fuzzy* dinâmicos podem ser construídos ao se adotar como variáveis de entrada nos antecedentes das regras as variáveis de regressão de entrada e/ou de saída do sistema real [32, 81].

Na seção 3.2.4 do capítulo 3 será apresentada uma arquitetura mais geral para sistemas *fuzzy*, os modelos *fuzzy* TS Generalizados, nos quais os conseqüentes das regras são modelos dinâmicos completos em espaço de estados.

2.5 Resumo

O objetivo deste capítulo foi descrever em linhas gerais o contexto de modelagem de sistemas dinâmicos, foco do presente trabalho. Inicialmente apresentaram-se as principais propriedades de modelos matemáticos, apontando-se quais destas propriedades estão presentes nos modelos desenvolvidos nesta dissertação. Em suma, tratar-se-á de modelos *não lineares*, *discretos*, *determinísticos*, *a parâmetros concentrados*, *invariantes no tempo*, *não autônomos*, *paramétricos* e *monovariáveis*.

Foram abordados alguns modelos específicos, tanto lineares quanto não lineares. Os modelos *fuzzy*, que constituem a base dos modelos analisados neste trabalho, foram estudados mais detalhadamente. O tema do próximo capítulo é o modelo *fuzzy* TS com *Funções de Base Ortonormal* (FBO).

[2]Mesmo no caso de modelos locais não lineares nas entradas o método dos mínimos quadrados pode ser usado pois a saída do modelo continua linear nos parâmetros.

Será descrita a arquitetura cujos parâmetros de projeto estarão sujeitos à otimização pelo algoritmo genético apresentado no capítulo 4.

Capítulo 3

Modelo *Fuzzy* TS FBO

O capítulo anterior discorreu sobre conceitos básicos de modelagem de sistemas dinâmicos, apresentando uma série de modelos lineares e não lineares, com maior ênfase nos modelos *fuzzy*. O presente capítulo analisa com maior profundidade o conceito de Funções de Base Ortonormal (FBO), introduzido na seção 2.2.7, e apresenta alguns exemplos de modelos baseados neste conceito. É dada maior atenção ao modelo *fuzzy* TS FBO, proposto originalmente em [89] e foco deste trabalho, resultado da incorporação de Funções de Base Ortonormal nos modelos *fuzzy* TS. A seção 3.3 explicita quais parâmetros desta arquitetura serão otimizados pelo algoritmo genético descrito no capítulo 4. Por fim, analisam-se dois modos de estimação dos coeficientes dos conseqüentes das regras dos modelos *fuzzy* TS FBO.

3.1 Funções de Base Ortonormal

A seção 2.2.7 introduziu os conceitos do modelo linear FBO. Trata-se do desenvolvimento da resposta ao impulso do sistema em uma base de funções ortonormais. A presente seção analisa os modelos não lineares FBO. A base para sua derivação será o modelo NARX, reproduzido aqui na equação (3.1) para facilitar referências.

$$y(k) = \mathcal{F}[y(k-1), \ldots, y(k-n_a), u(k-1), \ldots, u(k-n_b)] \tag{3.1}$$

Embora seja possível o uso de métodos eficientes de estimação para realizações particulares do operador $\mathcal{F}$, este modelo possui a desvantagem de apresentar recursão da saída, além de uma definição adequada da estrutura do modelo ser crítica. As funções de base ortonormal, como será explicado nesta seção, servem como ferramenta para resolver estes problemas, além de agregar outras vantagens na modelagem de sistemas dinâmicos.

É possível reescrever a equação (3.1) conforme a equação (3.2), na qual os termos da saída foram substituídos ao se utilizar a própria equação (3.1) de forma recursiva.

$$y(k) = \mathcal{G}[u(\lambda)]_{\lambda=-\infty}^{k-1} \tag{3.2}$$

Para sistemas com memória finita[1], é possível truncar a representação (3.2) em um número ν de termos, obtendo [89]:

$$\hat{y}(k) = \mathcal{G}[u(k-1), \ldots, u(k-\nu)] \tag{3.3}$$

A representação (3.3) é uma extensão não linear da resposta ao impulso finita descrita na seção 2.2.4. Trata-se do desenvolvimento do sinal de entrada $u(k)$ em uma base de funções de transferência no operador de deslocamento no tempo q^{-1}, base essa dada pelo conjunto de funções de transferência pulsadas definido pela equação (3.4):

$$\Phi_{\mathrm{NFIR},i}(q^{-1}) = q^{-i}, \quad i = 1, \ldots, \nu \tag{3.4}$$

Assim, a i-ésima entrada para o operador $\mathcal{G}$ na equação (3.3) é dada por:

$$\Phi_{\mathrm{NFIR},i}(q^{-1})u(k) = q^{-i}u(k) = u(k-i) \tag{3.5}$$

Note-se que as funções de transferência pulsadas da base (3.4) podem ser escritas de forma recursiva, ou seja, $\Phi_{\mathrm{NFIR},i+1}(q^{-1}) = q^{-1}\Phi_{\mathrm{NFIR},i}(q^{-1})$. O diagrama de blocos do modelo NFIR, considerando-se a recursão das funções de transferência de sua base, é exibido na Figura 3.1.

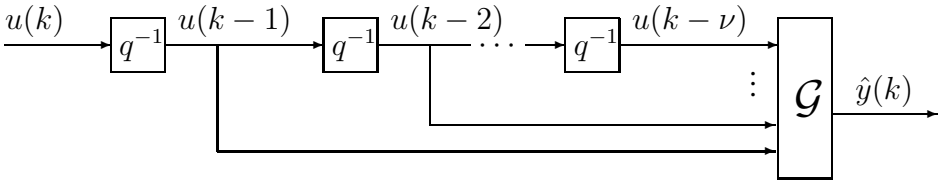

Figura 3.1: Diagrama de blocos do modelo NFIR.

Uma vantagem do modelo NFIR é não possuir recursão dos termos da saída. Dessa forma, não há realimentação de erros de predição no modelo, o que permite uma melhor performance nas tarefas de predição ao se considerar horizontes longos. Recursão da saída é ainda eventualmente associada a problemas de convergência nos algoritmos de identificação. Em contrapartida, a representação NFIR

[1]Um sistema possui memória finita se o valor de entrada $u(k-\nu)$, para ν suficientemente grande, não influencia significativamente a saída $y(k)$ [33].

em (3.3) requer um número muito maior de termos de regressão (ν) do que aquele requerido pela representação NARX original em (3.1) para uma mesma precisão, especialmente na representação de sistemas lentos.

No lugar de utilizar a base dada pelas funções da equação (3.4), é possível adotar as funções de base ortonormal, as quais tratam os problemas mencionados através da incorporação de conhecimento *a priori* sobre as dinâmicas do sistema na base de funções de transferência. O efeito imediato é a considerável diminuição do número de parâmetros no modelo para se alcançar uma dada capacidade de representação, além de ser uma arquitetura sem recursão da saída.

As bases de funções ortonormais são completas no espaço $L^2[0,\infty)$ das funções contínuas quadraticamente integráveis em $[0,\infty)$. Dessa forma, qualquer função neste espaço pode ser representada com precisão arbitrária por uma combinação linear das funções desta base. Considerando o caso discreto, isto significa que existe um inteiro positivo n tal que para qualquer $\epsilon > 0$ tem-se:

$$\sum_{k=0}^{\infty}\left(f(k)-\sum_{i=1}^{n}\alpha_i\Phi_i(k)\right)^2 < \epsilon \tag{3.6}$$

sendo $f(k)$ uma função quadraticamente somável, n o número de funções na base ortonormal, α_i escalares e Φ_i a i-ésima função ortonormal. A aproximação da função $f(k)$ será tão boa quanto maior for o número de funções ortonormais utilizado, sendo exata para um número infinito de funções. Embora em princípio sistemas com integradores não pertençam ao espaço $L^2[0,\infty)$, uma mudança de variáveis (análise da variação da saída do sistema, no lugar da saída instantânea) elimina o problema, permitindo também o tratamento desses sistemas.

Assim, da mesma forma que a base (3.4) é usada para gerar as entradas do operador não linear $\mathcal{G}$ no modelo NFIR (3.3), as funções de base ortonormal com funções de transferência $\Phi_{\mathrm{FBO},i}(q^{-1})$ geram os sinais [89]:

$$l_i(k) = \Phi_{\mathrm{FBO},i}(q^{-1})u(k) \tag{3.7}$$

os quais, através de um mapeamento não linear $\mathcal{H}$, fornecem a representação não linear FBO, *Funções de Base Ortonormal*, doravante denominada apenas modelo FBO:

$$\hat{y}(k) = \mathcal{H}[l_1(k),\ldots,l_n(k)] \tag{3.8}$$

Cada sinal l_i corresponde à convolução entre a entrada u e a i-ésima função da base ortonormal Φ_i. Visto de outra forma, é resultado da filtragem da entrada u pelo filtro ortonormal $\Phi_{\mathrm{FBO},i}$ [14]. A Figura 3.2 exibe o diagrama de blocos para o modelo FBO.

As funções de base ortonormal têm sido aplicadas com sucesso nas tarefas de modelagem e controle de sistemas dinâmicos ([14, 16, 83, 89, 108, 113] e referências inclusas). O modelo dinâmico

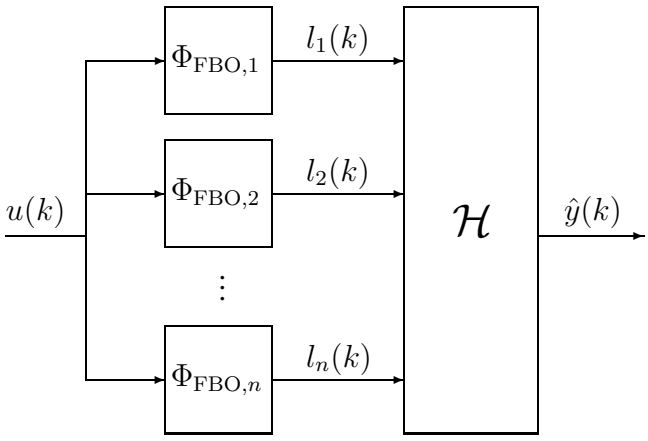

Figura 3.2: Diagrama de blocos do modelo FBO.

baseado em funções de base ortonormal, ou apenas modelo FBO, possui uma série de características positivas [14]:

- Como exposto no início desta seção, modelos FBO requerem um menor número de parâmetros para representar satisfatoriamente um sistema dinâmico, em comparação com modelos NFIR ou NARX;
- Não há necessidade de se conhecer previamente a estrutura exata do vetor de regressão (ordens e atrasos de transporte);
- Se necessário, é possível aumentar a capacidade de representação do modelo aumentando o número de funções ortonormais usadas;
- Assim como o modelo NFIR, o modelo FBO não possui recursão de termos da saída, evitando a realimentação no modelo de erros de predição. A ausência de realimentação da saída também implica em características estatísticas desejáveis para a estimação numérica dos parâmetros do modelo [13], a ser tratada na seção 3.4;
- Desacoplamento natural das saídas em sistemas multivariáveis;
- Tolerância a dinâmicas não modeladas;

- Capacidade para tratar atrasos de transporte.

As bases ortonormais mais tradicionais são a de Laguerre e a de Kautz [14, 55]. A diferença entre elas é que as funções ortonormais da base de Laguerre são parametrizadas por um único pólo real, enquanto as da base de Kautz por um par de pólos complexos conjugados. A base de Kautz é mais indicada para sistemas com dinâmicas dominantes pouco amortecidas, precisando de menos parâmetros que a base de Laguerre para alcançar determinado desempenho. Ambas as bases são realizações especiais das bases ortonormais generalizadas, nas quais as funções ortonormais podem ser parametrizadas por qualquer quantidade de pólos reais e complexos [55, 108]. Estas últimas, no entanto, necessitam de um maior conhecimento sobre as dinâmicas dominantes do sistema, e são aplicadas em sistemas complexos de alto grau de não linearidade [55]. Este trabalho concentra-se nas bases de Laguerre e de Kautz. Será apresentado na seção 3.2.4 o modelo *fuzzy* TS FBO Generalizado [14, 15]. Porém, como será visto, as bases utilizadas são também de Laguerre e de Kautz, sendo o termo "Generalizado" justificado pelo fato de ser possível o uso de diferentes pólos (reais ou complexos) em cada modelo local.

O conjunto das funções que formam a base de Laguerre pode ser escrito em função do pólo real estável p segundo a equação (3.9) [112]:

$$\Phi_{\mathrm{Lag},n}(q^{-1}) = \sqrt{1-p^2}\ \frac{q^{-1}\ (q^{-1}-p)^{n-1}}{(1-pq^{-1})^n}, \quad n = 1, \ldots, \infty \tag{3.9}$$

Já as funções da base de Kautz são parametrizadas pelo pólo complexo $\beta = \alpha \pm j\omega$ segundo a equação (3.10) [113]:

$$\begin{aligned}
\Phi_{\mathrm{Kau},n}(z) &= \frac{\sqrt{1-c^2}(z-b)}{z^2+b(c-1)z-c}\left[\frac{-cz^2+b(c-1)z+1}{z^2+b(c-1)z-c}\right]^{n-1}, \quad \text{para } n \text{ ímpar} \\
\Phi_{\mathrm{Kau},n}(z) &= \frac{\sqrt{(1-c^2)(1-b^2)}}{z^2+b(c-1)z-c}\left[\frac{-cz^2+b(c-1)z+1}{z^2+b(c-1)z-c}\right]^{n-1}, \quad \text{para } n \text{ par}
\end{aligned} \tag{3.10}$$

sendo $b = (\beta+\beta^*)/(1+\beta\beta^*)$ e $c = -\beta\beta^*$.

Modelos baseados em funções ortonormais admitem a seguinte representação em espaço de estados [16, 89]:

$$\begin{aligned}
\mathbf{l}(k+1) &= \mathbf{A}\,\mathbf{l}(k) + \mathbf{b}\,u(k) \\
\hat{y}(k) &= \mathcal{H}(\mathbf{l}(k))
\end{aligned} \tag{3.11}$$

sendo $\mathbf{l}(k) = [l_1(k) \ldots l_n(k)]^T$ o vetor de estados ortonormais[2]. É possível incorporar ainda ao mo-

[2]Como afirmado anteriormente, os termos $l_i(k)$ são na verdade o resultado da filtragem da entrada u pelo filtro

delo FBO da equação (3.11) qualquer informação *a priori* sobre o atraso de transporte do sistema, substituindo nesta equação $u(k)$ por $u(k-\tau)$, sendo τ o atraso conhecido [14]. Assim, é possível reduzir ainda mais o número de parâmetros no modelo através da respectiva redução do número de funções ortonormais necessárias para determinada precisão.

A matriz **A** e o vetor **b** da equação (3.11) dependem unicamente da base de funções ortonormais adotada. Para o caso da base de Laguerre, utilizando-se (3.9) de forma recursiva obtém-se [14]:

$$\mathbf{A} = \begin{bmatrix} p & 0 & 0 & \dots & 0 \\ 1-p^2 & p & 0 & \dots & 0 \\ (-p)(1-p^2) & 1-p^2 & p & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ (-p)^{n-2}(1-p^2) & (-p)^{n-3}(1-p^2) & & \dots & p \end{bmatrix} \tag{3.12}$$

$$\mathbf{b} = \sqrt{1-p^2} \begin{bmatrix} 1 & -p & (-p)^2 & \dots & (-p)^{n-1} \end{bmatrix} \tag{3.13}$$

sendo p o pólo de Laguerre. Para a representação em espaço de estados da base de Kautz, consultar [113].

Escolhida a base de funções ortonormais, os tipos diferentes de modelos FBO são determinados pelo operador $\mathcal{H}$ da equação (3.11). A seção seguinte apresenta possíveis escolhas para este operador, sendo as duas últimas as mais exploradas no presente trabalho.

3.2 Arquiteturas dos Modelos FBO

São apresentados na seqüência os modelos FBO linear e de Volterra, os quais serão posteriormente utilizados nos modelos locais da arquitetura *fuzzy* TS FBO. Todos estes modelos possuem a desejável característica de serem lineares em seus parâmetros. Dessa forma, o operador genérico $\mathcal{H}$ pode ser escrito como [13]:

$$\mathcal{H}(\mathbf{l}(k)) = \boldsymbol{\lambda}(k)^T \boldsymbol{\theta} \tag{3.14}$$

sendo $\boldsymbol{\theta}$ o vetor de parâmetros livres a ser estimado e $\boldsymbol{\lambda}(k)$ um vetor de dados função dos estados ortonormais $\mathbf{l}(k)$, comumente denominado vetor de regressores [3]. Os parâmetros destes modelos podem ser estimados eficientemente através do método de mínimos quadrados, detalhado na seção 3.4.

ortonormal $\Phi_{\mathrm{FBO},i}$. Assim, esta é a interpretação correta para a expressão "estado ortonormal" usada no decorrer de todo o texto.

3.2.1 Modelo FBO Linear

Este modelo advém da implementação do operador $\mathcal{H}$ como uma combinação linear dos estados ortonormais, tal qual apresentado na seção 2.2.7. Na prática, adiciona-se à combinação linear um termo constante, obtendo-se assim um modelo afim, capaz de modelar um eventual nível constante na saída do sistema. Para esse modelo, o vetor de dados $\boldsymbol{\lambda}(k)$ da equação (3.14) consiste no próprio vetor de estados ortonormais concatenado com um elemento unitário, ou seja:

$$\boldsymbol{\lambda}(k)^T = \begin{bmatrix} 1 \\ \mathbf{l}(\mathrm{k}) \end{bmatrix} = \begin{bmatrix} 1 & l_1(k) & \dots & l_n(k) \end{bmatrix}^T \tag{3.15}$$

O vetor de parâmetros possui então dimensão igual à ordem do modelo mais um, $n + 1$.

3.2.2 Modelo FBO de Volterra

Conforme afirmado na seção 2.3.2, modelos de Volterra de ordem elevada apresentam o inconveniente de induzirem um grande número de parâmetros livres. Na prática, utilizam-se modelos de segunda ordem. Porém, mesmo neste caso, sistemas com dinâmicas dominantes lentas requerem um número elevado de parâmetros do modelo. As funções de base ortonormais são usadas para resolver este problema, além de acrescentar as vantagens descritas na seção 3.1.

O modelo FBO de Volterra é obtido ao se desenvolver cada um dos núcleos de ordem i da equação (2.26) em termos de funções ortonormais. Considera-se aqui apenas os núcleos de até segunda ordem, de modo a evitar o número elevado de parâmetros das ordens superiores. O desenvolvimento do núcleo de primeira ordem, que corresponde exatamente à resposta ao impulso do sistema, já foi apresentado na seção 2.2.7 ao se tratar dos modelos lineares FBO. O desenvolvimento do núcleo de segunda ordem é análogo ao apresentado naquela seção [14]. Dessa forma, o modelo FBO de Volterra de segunda ordem é dado por:

$$\hat{y}(k) = \alpha_0 + \sum_{j=1}^{n} \alpha_j l_j(k) + \sum_{j=1}^{n} \sum_{i=1}^{j} \phi_{j,i} l_j(k) l_i(k) \tag{3.16}$$

sendo α_j e $\phi_{i,j}$ os coeficientes de desenvolvimento de primeira e de segunda ordem, respectivamente, e α_0 um termo constante incluído pela razão apresentada na seção anterior. Considerando por simplicidade que os termos de primeira e segunda ordem possuem o mesmo número n de estados ortonormais, além de assumir a simetria existente entre os núcleos de Volterra [14] (o que permitiu limitar superiormente o último somatório da equação (3.16) em j), obtém-se o vetor de dados como:

$$\begin{aligned}\boldsymbol{\lambda}(k) \;=\; [\;\; 1 \;\; l_1(k) \;\; \ldots \;\; l_n(k) \;\; l_1(k)^2 \;\; l_2(k)l_1(k) \;\; l_2(k)^2 \;\; \ldots \\ l_n(k)l_1(k) \;\; l_n(k)l_2(k) \;\; l_n(k)^2 \;\;]^T\end{aligned} \tag{3.17}$$

O número de parâmetros livres para esta arquitetura é igual a $(n^2 + 3n + 2)/2$.

3.2.3 Modelo *Fuzzy* TS FBO

Os modelos apresentados nesta seção e na próxima são baseados no trabalho inicialmente proposto em [89].

O modelo *fuzzy* TS FBO é obtido ao se implementar o operador $\mathcal{H}$ segundo o sistema de inferência *fuzzy* apresentado na seção 2.4.3. Mais especificamente, adotam-se como variáveis de entrada do sistema *fuzzy* TS os próprios estados ortonormais $l_i(k)$. Dessa forma, a base de regras da equação (2.34) é reescrita como:

$$\begin{aligned}&\text{se} \quad (l_1 \text{ é } A_{1,1}) \text{ e } \ldots \text{ e } (l_n \text{ é } A_{1,n}) \text{ então } y = f_1(l_1, \ldots, l_n) \\ &\quad\vdots \\ &\text{se} \quad (l_1 \text{ é } A_{m,1}) \text{ e } \ldots \text{ e } (l_n \text{ é } A_{m,n}) \text{ então } y = f_r(l_1, \ldots, l_n)\end{aligned} \tag{3.18}$$

e a saída do modelo é calculada segundo a equação (3.19), sendo o peso w_i da i-ésima regra calculado agora em função dos estados $\mathbf{l}(k)$, e não mais diretamente a partir das entradas u_i, ou seja:

$$\hat{y}(k) = \frac{\sum_{i=1}^{r} w_i(\mathbf{l}(k))y_i}{\sum_{i=1}^{r} w_i(\mathbf{l}(k))} \tag{3.19}$$

O vetor de dados $\boldsymbol{\lambda}(k)$ para este modelo depende da escolha para as funções $f_i(.)$ de cada modelo local. Como exemplo, caso sejam adotadas funções afins do tipo:

$$y = a_{i,1}l_1 + \ldots + a_{i,n}l_n + a_{i,n+1} \tag{3.20}$$

então, a partir das equações (3.14), (3.19) e (3.20), tem-se:

$$\begin{aligned}\boldsymbol{\lambda}(k) \;=\; \gamma(k) \;[\; w_1(\mathbf{l}(k)) \;\; w_1(\mathbf{l}(k))l_1(k) \;\; \ldots \;\; w_1(\mathbf{l}(k))l_n(k) \;\; \ldots \\ w_r(\mathbf{l}(k)) \;\; w_r(\mathbf{l}(k))l_1(k) \;\; \ldots \;\; w_r(\mathbf{l}(k))l_n(k) \;\;]^T\end{aligned} \tag{3.21}$$

sendo $\gamma(k) = 1/\sum_{i=1}^{r} w_i(\mathbf{l}(k))$ o termo de normalização presente na equação (3.19). O vetor de

parâmetros $\boldsymbol{\theta}$ consiste nos coeficientes $a_{i,j}$ das funções afins nos conseqüentes das regras (3.20):

$$\boldsymbol{\theta} \;=\; [a_{1,1}\;\; a_{1,2}\;\; \ldots\;\; a_{1,n+1}\;\; \ldots\;\; a_{r,1}\;\; \ldots\;\; a_{r,n+1}\;\;]^T \tag{3.22}$$

Os modelos locais afins, a menos da constante $a_{i,n+1}$, são lineares. É possível utilizar também modelos locais não lineares, como os modelos de Volterra de segunda ordem. Neste caso, basta utilizar a expressão (3.16) nos modelos locais.

O número de parâmetros livres a serem estimados por mínimos quadrados para o modelo *fuzzy* TS FBO é calculado como a soma da quantidade de parâmetros de cada modelo local. A fórmula para modelos afins é a mesma da seção 3.2.1 e para modelos de Volterra de segunda ordem a da seção 3.2.2. O número total de regras r é calculado considerando uma base de regras completa, isto é, que contém tantas regras quantas são as combinações entre as funções de pertinência de todas as variáveis de entrada do modelo. Assim, caso a i-ésima variável de entrada possua m_i funções de pertinência associadas a ela, o número total de regras do modelo, considerando uma base de regras completa, é:

$$r = \prod_{i=1}^{n} m_i \tag{3.23}$$

Modelos *fuzzy* TS FBO foram aplicados com sucesso nas tarefas de modelagem [13] e controle [14] de sistemas dinâmicos, fornecendo resultados melhores (em termos de melhor aproximação com menor número de parâmetros) que modelos lineares FBO ou Volterra FBO. No entanto, nestes trabalhos a seleção da estrutura dos modelos e de alguns parâmetros livres foi realizada de modo empírico, sem a aplicação de métodos de otimização bem consolidados na literatura. A distribuição das funções de pertinência no universo de discurso, por exemplo, era feita de forma homogênea, com centros igualmente espaçados e larguras iguais às distâncias entre centros consecutivos [16, 39]. Além disso, determinava-se o pólo de Laguerre através da observação da resposta do sistema a uma entrada em geral do tipo degrau. Após se avaliar qual seria aproximadamente a dinâmica dominante do sistema, realizava-se uma busca iterativa discreta em torno desta região.

Tais metodologias, apesar de fornecerem soluções que demonstraram a superioridade das arquiteturas e métodos propostos, ainda eram passíveis de uma maior otimização, com o acréscimo de algum esforço computacional [13, 15, 16]. O presente trabalho foca essa otimização dos modelos *fuzzy* TS FBO. Os parâmetros do modelo que serão otimizados são apresentados na seção 3.3 e o método de otimização proposto é descrito no capítulo 4. No capítulo 5 realiza-se uma análise comparativa entre essas metodologias não automáticas e a abordagem aqui proposta.

3.2.4 Modelo *Fuzzy* TS FBO Generalizado

Conforme a seção 3.1, as bases de Laguerre e de Kautz são parametrizadas apenas por um pólo, respectivamente real e complexo. Embora tenha sido afirmado que os modelos com estas bases são capazes de aproximar sistemas dinâmicos com qualquer precisão desejada (aumentando-se o número de estados no modelo), quando o sistema sendo tratado apresenta mais de um modo dominante, o número de parâmetros no modelo pode crescer sensivelmente. Para tratar desses casos, foram propostas bases de funções ortonormais generalizadas [9, 84, 108]. Estas bases possuem a característica de poderem ser parametrizadas em qualquer número de pólos reais e/ou complexos. Em geral, são construídas de forma a englobar as bases de Laguerre e de Kautz como casos especiais.

A idéia do modelo *fuzzy* TS FBO Generalizado é possibilitar a incorporação de informações sobre diversas dinâmicas presentes no sistema utilizando ainda as bases de Laguerre e de Kautz. Para alcançar este objetivo, adota-se em cada modelo local das regras de inferência um modelo em espaço de estados da forma (3.11) parametrizado em um pólo distinto (real ou complexo). Dessa forma, os estados de cada modelo local podem refletir uma dinâmica dominante diferente do sistema, diferentemente do modelo *fuzzy* TS FBO apresentado na seção 3.2.3, no qual a diferença entre os modelos locais residia apenas nas relações estáticas entre os estados ortonormais e a saída. Assim, para o modelo *fuzzy* TS FBO Generalizado, os próprios estados presentes nas premissas das regras não são os mesmos de cada modelo local, mas sim obtidos através do processo de inferência *fuzzy* TS. Em suma, cada uma das r regras do sistema *fuzzy* passa agora a ter a forma [15, 16]:

$$R^i: \text{ se } (\ \zeta_1(k) \text{ é } A_{i,1}\ ) \text{ e } \ldots \text{ e } (\ \zeta_n(k) \text{ é } A_{i,n}\ ) \quad \text{então} \begin{cases} \mathbf{l}_i(k+1) &= \mathbf{A}_i\,\mathbf{l}_i(k) + \mathbf{b}_i\,u(k) \\ y_i(k) &= f_i(\mathbf{l}_i(k)) \end{cases} \tag{3.24}$$

sendo $\mathbf{A}_i$ e $\mathbf{b}_i$ parametrizados por um pólo p_i, fornecendo o vetor de estados $\mathbf{l}_i(k)$, $f_i(\mathbf{l}_i(k))$ o i-ésimo mapeamento estático que gera i-ésima saída $y_i(k)$ do respectivo modelo local e $\zeta_1(k), \ldots, \zeta_n(k)$ as variáveis das premissas das regras, obtidas segundo a equação (3.25).

$$\boldsymbol{\zeta}(k+1) = \frac{\sum_{i=1}^{r} w_i(\boldsymbol{\zeta}(k))\mathbf{l}_i(k+1)}{\sum_{i=1}^{r} w_i(\boldsymbol{\zeta}(k))} \tag{3.25}$$

sendo $\boldsymbol{\zeta}(k) = [\ \zeta_1(k)\ \ldots\ \zeta_n(k)\ ]^T$. Os pesos $w_i(\boldsymbol{\zeta}(k))$ correspondem ao nível de ativação da i-ésima regra, computados através da aplicação da t-norma adotada (ver seção 2.4.3). Seguindo o mesmo

mecanismo de inferência, a saída do modelo *fuzzy* TS FBO Generalizado é dada por:

$$\hat{y}(k) = \frac{\sum_{i=1}^{r} w_i(\boldsymbol{\zeta}(k)) y_i(k)}{\sum_{i=1}^{r} w_i(\boldsymbol{\zeta}(k))} \tag{3.26}$$

Assim como no modelo *fuzzy* TS FBO, caso os modelos locais $f_i(\mathbf{l}_i(k))$ de cada regra em (3.24) sejam lineares nos parâmetros, então o modelo Generalizado também o será, e novamente métodos eficientes de estimação poderão ser aplicados.

Para demonstrar que o modelo *fuzzy* TS FBO Generalizado é realmente uma generalização do modelo *fuzzy* TS FBO apresentado na seção 3.2.3, basta fazer com que os pólos p_i de cada modelo em espaço de estados em (3.24) sejam iguais. Dessa forma, a partir da equação (3.25), $\boldsymbol{\zeta}(k) = \mathbf{l}_i(k)$, para $i = 1, \ldots, m$. Assim, os modelos locais passam a compartilhar o mesmo vetor de estados ortonormais, e a saída final do modelo (3.26) recai no caso do modelo *fuzzy* TS FBO (3.19).

Foi demonstrado em [14, 15, 16] que os modelos *fuzzy* TS FBO são capazes de aproximar com precisão arbitrária sistemas dinâmicos não lineares discretos que sejam causais, invariantes no tempo, sem descontinuidades, com memória finita e entrada limitada em um intervalo fechado. A demonstração está baseada na capacidade de aproximação universal dos modelos *fuzzy* e de modelos de Volterra.

Esta dissertação está focada nos dois últimos modelos, *fuzzy* TS FBO e *fuzzy* TS FBO Generalizado. Trata-se da incorporação de um método de otimização na etapa de definição da estrutura destes modelos. A próxima seção define o escopo do método proposto.

3.3 Parâmetros de Projeto do Modelo

Os modelos apresentados nas seções 3.2.3 e 3.2.4 possuem uma série de parâmetros livres de projeto. Esta seção indica quais destes parâmetros serão definidos autonomamente pela metodologia proposta no presente trabalho, a qual é baseada nos Algoritmos Genéticos (AG), métodos de otimização não lineares, descritos em detalhes no capítulo 4. A seção 4.3 daquele capítulo apresentará uma revisão sobre a aplicação de algoritmos genéticos na otimização de sistemas *fuzzy*, quando então será possível visualizar melhor as contribuições dessa dissertação.

Para o modelo *fuzzy* TS FBO com pólo único, praticamente todos os parâmetros livres são definidos e otimizados pelo AG. Mais especificamente, os seguintes parâmetros são incorporados no método automático, não precisando então de nenhuma intervenção humana para sua definição:

- Pólo real ou complexo da base de funções ortonormais, o qual gera a matriz **A**, o vetor **b** e assim o vetor de estados ortonormais $\mathbf{l}(k)$;
- Número n de estados nos modelos locais. Como os estados são definidos por uma mesma base de funções ortonormais, este número é mantido fixo para todos os modelos locais;
- Número de estados nas premissas das regras (número de variáveis de entrada). Embora nos desenvolvimentos anteriores tenha ficado implícito que este número seria igual ao número de estados nos modelos locais, como afirmado na seção 2.4.3 pode ser usado um número menor de estados nas premissas das regras;
- Índice dos estados nas premissas das regras. Além de definir quantas variáveis de entrada existirão no sistema, o método escolhe quais seriam as mais adequadas dentro do conjunto de estados;
- Número de funções de pertinência por variável de entrada. Neste caso, é possível determinar números distintos de funções de pertinência para variáveis de entrada distintas;
- Configuração de cada função de pertinência. Como será justificado na seção 4.4.1 do capítulo 4, as funções de pertinência adotadas para os modelos *fuzzy* são gaussianas, parametrizadas em dois parâmetros: sua abertura e centro. Ambos são definidos automaticamente pelo AG.

Como está sendo suposto o uso de uma base de regras completa, o número de regras do modelo *fuzzy* será dado pela equação (3.23), considerando o número de estados nas premissas das regras e o número de funções de pertinência em cada estado definidos pelo AG.

Para o modelo *fuzzy* TS FBO Generalizado, os seguintes parâmetros são definidos pelo AG:

- Pólos reais ou complexos de cada modelo local;
- Número de estados nos modelos locais. Como agora os modelos locais são parametrizados em pólos distintos, o número de estados de cada modelo local é otimizado de forma independente;
- Índice dos estados nas premissas das regras. Embora o número de estados nas premissas das regras seja uma entrada do AG, o método fornece o conjunto de estados mais adequados;
- Configuração de cada função de pertinência, também gaussiana.

A seção 4.4.1 apresentará as justificativas para a otimização de menos parâmetros livres no modelo *fuzzy* TS FBO Generalizado. Em suma, os argumentos serão baseados na complexidade da tarefa de se otimizar toda a arquitetura deste modelo, além de ser possível o uso de alguma informação *a*

priori sobre as características do sistema sendo modelado (como o número de regiões de operação esperado).

Além destes parâmetros otimizados pelo AG, restam ainda todos os coeficientes $\boldsymbol{\theta}$ dos modelos locais. Como afirmado anteriormente, estes parâmetros podem ser estimados usando o método dos mínimos quadrados. A seção seguinte apresenta tal método e analisa duas opções possíveis de sua aplicação. A primeira, estimação global, será usada em ambas arquiteturas propostas. Já a segunda, estimação local, será aplicada apenas no modelo *fuzzy* TS FBO Generalizado.

3.4 Estimação Global × Estimação Local

Conforme afirmado nas seções anteriores, a adoção de modelos FBO lineares e FBO de Volterra nos conseqüentes das regras dos modelos *fuzzy* TS FBO mantém a característica de linearidade nos parâmetros, seguindo a estrutura da equação (3.14). Sendo assim, o estimador de mínimos quadrados pode desempenhar de forma ótima a tarefa de estimação dos parâmetros $\boldsymbol{\theta}$ [3].

O desenvolvimento do estimador de mínimos quadrados (MQ) é baseado na equação (3.27), obtida a partir das equações (3.11) e (3.14) ao se supor que é cometido um erro de aproximação ξ em relação à saída real $y(k)$ a partir do vetor estimado de parâmetros $\hat{\boldsymbol{\theta}}$:

$$y(k) = \boldsymbol{\lambda}(k)^T \hat{\boldsymbol{\theta}} + \xi \tag{3.27}$$

Para tal, supõe-se que seja feito um conjunto de N medidas da saída $y(k)$, fornecendo o vetor $\mathbf{y}$, correspondentes a N vetores $\boldsymbol{\lambda}(k)$, agrupados na matriz $\boldsymbol{\Lambda} = [\boldsymbol{\lambda}_1(k)\ \boldsymbol{\lambda}_2(k)\ \ldots\ \boldsymbol{\lambda}_N(k)]^T$. Assim:

$$\mathbf{y} = \boldsymbol{\Lambda}\hat{\boldsymbol{\theta}} + \boldsymbol{\xi} \tag{3.28}$$

Os parâmetros $\hat{\boldsymbol{\theta}}$ são obtidos a partir de (3.28) ao se minimizar uma função de custo baseada em uma norma quadrática do erro de aproximação $\boldsymbol{\xi}$, função essa da forma [3]:

$$J = \sum_{i=1}^{N} \xi(i)^2 = \|\boldsymbol{\xi}\|^2 \tag{3.29}$$

Substituindo $\boldsymbol{\xi}$ da equação (3.28) em (3.29) e minimizando esta equação, chega-se à seguinte equação, que caracteriza o estimador de mínimos quadrados [3]:

$$\hat{\boldsymbol{\theta}} = [\boldsymbol{\Lambda}^T\boldsymbol{\Lambda}]^{-1}\boldsymbol{\Lambda}^T\mathbf{y} \tag{3.30}$$

O estimador de mínimos quadrados apresenta as seguintes propriedades [13, 72][3]:

- Não polarizado, ou seja, o valor esperado (média) do erro de estimação é nulo;
- Melhor estimador linear não polarizado, implicando que ele minimiza a variância do erro de estimação;
- Consistente, o que significa que a variância do erro de estimação tende a zero na medida em que o número N de medições tende a infinito.

O estimador de MQ tal qual foi apresentado acima consiste na estimação global dos parâmetros dos modelos *fuzzy* TS FBO, pois cada vetor $\boldsymbol{\lambda}(k)$ compondo a matriz $\boldsymbol{\Lambda}$ contém informações sobre todas as regras e, conseqüentemente, sobre todos os modelos locais. Logicamente, o vetor de parâmetros $\hat{\boldsymbol{\theta}}$ também abrange os coeficientes dos conseqüentes de todas as regras, como ilustrado na equação (3.22). Assim, em apenas um passo (uma aplicação da equação (3.30)), todos os parâmetros livres dos modelos locais são estimados.

Dessa forma, os modelos locais não serão necessariamente aproximações locais do sistema não linear sendo modelado, o que acontece com a estimação local, explicada a seguir [2, 51, 81]. Esta característica de interpretação local é importante em algumas aplicações, como em controladores de ganho adaptativo, nos quais os modelos locais são usados para se projetar controladores locais, além de ser útil para a análise e validação do modelo [51]. No entanto, normalmente a estimação global dos parâmetros fornece uma melhor aproximação do sistema como um todo, sendo adequada para as tarefas de predição não linear ou controle baseado em modelos [2, 23, 32, 51].

Na estimação local, aqui aplicada apenas no modelo *fuzzy* TS FBO Generalizado, os parâmetros de cada modelo local são estimados de forma independente dos outros modelos locais. No lugar de apenas uma execução do algoritmo de mínimos quadrados, realizam-se r execuções, cada uma estimando os parâmetros livres dos conseqüentes de uma regra. Assim, agora tanto o vetor de parâmetros $\hat{\boldsymbol{\theta}}_i$ quanto a matriz $\boldsymbol{\Lambda}_i$ dizem respeito a uma regra i específica. O estimador local de mínimos quadrados será então dado por [52, 72]:

$$\hat{\boldsymbol{\theta}}_i = [\boldsymbol{\Lambda}_i^T \boldsymbol{\Psi}_i \boldsymbol{\Lambda}_i]^{-1} \boldsymbol{\Lambda}_i^T \boldsymbol{\Psi}_i \mathbf{y} \tag{3.31}$$

Os vetores $\boldsymbol{\lambda}_i(k)$ que formam a matriz $\boldsymbol{\Lambda}_i$ são definidos ligeiramente diferentes para a estimação local, já que agora cada modelo local possui um conjunto específico de estados ortonormais. Considerando modelos locais lineares para manter a simplicidade na análise, para a i-ésima regra têm-se:

$$\boldsymbol{\lambda}_i(k) = [1\ \mathbf{l}_i^T] = [1\ l_{i1}(k)\ l_{i2}(k)\ \ldots\ l_{in}(k)] \tag{3.32}$$

[3]Desde que o erro de aproximação ξ e a saída $y(k)$ sejam estatisticamente independentes.

sendo $\mathbf{l}_i$ o conjunto de estados ortonormais da respectiva regra.

A matriz de ponderação $\mathbf{\Psi}_i$ da equação (3.31) é definida como uma matriz diagonal dada por:

$$\mathbf{\Psi}_i = \text{diag}[\psi_1 \ \psi_2 \ \ldots \ \psi_N] \tag{3.33}$$

sendo ψ_j calculado como:

$$\psi_j = \frac{w_j}{\sum_{i=1}^{m} w_i} \tag{3.34}$$

e os termos w_j correspondendo ao peso da j-ésima regra, calculados de acordo com a t-norma adotada.

Além das diferenças entre estimação global e local discutidas anteriormente, outro aspecto relevante é a complexidade dos cálculos das matrizes inversas nas equações (3.30) e (3.31). No primeiro caso, têm-se um problema de complexidade $O([r(n+1)]^3)$ enquanto a estimação local representa uma complexidade menor, $O(r(n+1)^3)$ [11, 52, 36]. Para modelos *fuzzy* TS FBO Generalizados com um número de regras (modelos locais) r elevado, a estimação global pode demandar um elevado esforço computacional.

O capítulo 5 apresenta os resultados obtidos utilizando-se tanto a estimação local quanto a global na tarefa de modelagem de um sistema levitador magnético. Naquela ocasião será discutido o desempenho de cada um dos métodos dentro da arquitetura proposta neste trabalho.

3.5 Resumo

Este capítulo primeiramente detalhou o conceito de Funções de Base Ortonormal, explicitando o conjunto de vantagens ao se utilizar essa metodologia em modelagem de sistemas dinâmicos. Em particular, foram apresentadas as bases de Laguerre e de Kautz.

Em seguida discutiu-se brevemente os modelos FBO Linear e de Volterra, e mais apuradamente os modelos *fuzzy* TS FBO e *fuzzy* TS FBO Generalizado, os quais constituem o foco do presente trabalho. Para estes modelos, foram listados os parâmetros de projeto que serão definidos e otimizados de forma autônoma.

Por último, descreveram-se duas formas de estimação dos coeficientes dos conseqüentes das regras dos modelos *fuzzy*: estimação global, mais adequada nas tarefas que envolvem predição, por permitir em geral uma melhor aproximação da função desejada; e local, que permite interpretação local dos modelos locais, sendo útil também na análise e validação dos modelos.

O próximo capítulo detalha o método usado para a otimização dos parâmetros livres de projeto da arquitetura apresentada neste capítulo.

Capítulo 4

Projeto de Sistemas *Fuzzy* TS FBO utilizando Algoritmos Genéticos

A computação evolutiva é constituída pelos métodos computacionais inspirados na teoria da evolução natural das espécies. Tais métodos são baseados em uma população de indivíduos sujeitos a avaliação, modificações e seleção. As instâncias dos algoritmos fundamentados em princípios evolutivos são chamadas de algoritmos evolutivos. Historicamente, os algoritmos evolutivos incluem as técnicas de estratégias evolutivas, programação genética e algoritmos genéticos.

Os algoritmos genéticos foram desenvolvidos por Holland no início dos anos 60 [45]. Originalmente, foram projetados como um sistema formal para adaptação, e não otimização. Suas características básicas eram a forte ênfase em recombinação (*crossover*), uso de um operador de seleção probabilístico e a interpretação da mutação como um operador secundário. Embora em sua forma original os algoritmos genéticos representassem soluções através de cadeias binárias, um grande número de variantes foi desenvolvido para ampliar o âmbito de aplicações do algoritmo.

Este capítulo descreve a técnica dos algoritmos genéticos e a arquitetura elaborada para o projeto automático do sistema *Fuzzy* TS FBO. Em cada seção, a partir da seção 4.4, detalha-se uma característica ou componente dos algoritmos genéticos de um modo geral e então a sua configuração específica na metodologia ora proposta.

4.1 Princípios Biológicos dos Algoritmos Evolutivos

Baseados nas teorias da Evolução Darwiniana (Charles R. Darwin, 1809-1882) e da Seleção Natural (Gregor Mendel, 1822-1884), os algoritmos evolutivos têm por objetivo encontrar o indivíduo ótimo de uma população geneticamente refinada. Este modelo faz uma analogia computacional com os critérios probabilísticos de seleção e evolução naturais, por meio de simulações de gerações que se

sucedem, formando uma população final mais adequada, ou adaptada, ao ambiente em questão.

Até meados do século XIX, os naturalistas acreditavam que cada espécie havia sido criada separadamente por um ser supremo ou através de geração espontânea. O trabalho do naturalista Carolus Linnaeus sobre a classificação biológica de organismos despertou o interesse pela similaridade entre certas espécies, levando a acreditar na existência de uma certa relação entre elas. Outros trabalhos influenciaram os naturalistas a favor da teoria da seleção natural, tais como os de Jean Baptiste Lamark, que sugeriu uma teoria evolucionária baseada no "uso e desuso" de órgãos; e de Thomas Robert Malthus, que propôs que fatores ambientais tais como epidemias e carência de alimentos limitavam o crescimento de uma população.

Depois de mais de 20 anos de observações e experimentos, Charles Darwin apresentou em 1858 sua teoria de evolução através de seleção natural, simultaneamente ao naturalista inglês Alfred Russel Wallace. No ano seguinte, Darwin publicou o seu *On the Origin of Species by Means of Natural Selection* com a sua teoria completa [21].

Este trabalho influenciou não apenas as áreas de Biologia, Botânica e Zoologia, mas também teve grande impacto sobre o pensamento religioso, filosófico, político e econômico da época. A teoria da evolução e a computação nasceram praticamente na mesma época: Charles Babbage, um dos fundadores da computação moderna e amigo pessoal de Darwin desenvolveu sua máquina analítica em 1833.

Por volta de 1900, o trabalho de Gregor Mendel, desenvolvido em 1865, sobre os princípios básicos de herança genética, foi redescoberto pelos cientistas e também influenciou as pesquisas relacionadas à evolução. A moderna teoria da evolução combina os fundamentos da genética de Mendel com as idéias de Darwin e Wallace sobre a seleção natural, criando o princípio básico de Genética Populacional: a variabilidade entre indivíduos em uma população de organismos que se reproduzem sexualmente é produzida pela mutação e pela recombinação genética.

Os princípios de Darwin procuram explicar a biodiversidade a partir de uma proposta inicial para sobrevivência de genes - a formação de indivíduos ou fenótipos - e um processo de mudanças graduais que adaptam e transformam os indivíduos de acordo com as exigências ambientais. O modelo ressalta o fato de que os indivíduos mais aptos irão, provavelmente, sobreviver por um período de tempo mais longo e deixarão uma herança genética mais intensa na população. A evolução dita darwiniana é um modelo de aproximação gradual para evolução - não permite saltos evolucionários ou macromutações. No paradigma darwiniano o principal mecanismo operacional da transferência genética é a reprodução sexuada.

A reprodução sexuada é uma estratégia de constituição de novos indivíduos com, pelo menos, dois indivíduos compartilhando seus genes na formação da descendência. Esse tipo de reprodução ocorre em dois contextos:

- O contexto da interação ambiente $\times$ indivíduo, envolvendo a sobrevivência física do indivíduo até a idade reprodutiva e os mecanismos de seleção do(s) parceiro(s) de reprodução.

- O contexto das células reprodutivas, envolvendo, na maioria dos casos, uma competição entre as células masculinas de um ou mais indivíduos.

Enquanto a seleção do parceiro reprodutivo ocorre em um universo macroscópico, cruzamentos e mutações na cadeia ADN (em inglês, DNA, *DeoxyriboNucleic Acid*) ocorrem no contexto da célula embrião, ou seja, da microbiologia. A configuração do código ADN é metaforicamente associada, nos algoritmos evolutivos, a uma configuração do problema que se deseja resolver.

A busca pela solução ótima de um problema através de um algoritmo evolutivo é associada à busca do indivíduo mais adaptado ao ambiente. A abordagem computacional resume as diferentes instâncias de decisão do fenômeno biológico em uma só etapa e simula o compartilhamento de ADN através da manipulação direta do cromossomo. O processo, portanto, é modelado por uma abordagem pseudoevolutiva que permite uma implementação computacional eficiente [68].

Todas as instâncias dos algoritmos evolutivos compartilham esses fundamentos biológicos. A técnica dos algoritmos genéticos, explorada na presente pesquisa, é detalhada na seqüência. Bons pontos de partida para o aprofundamento nos demais métodos são os trabalhos de Bäck *et al.* [6, 7].

4.2 Algoritmo Genético Básico

O algoritmo genético básico apresentado nesta seção pode, com algumas modificações, descrever o funcionamento da maioria dos algoritmos evolutivos. As principais diferenças residirão na representação da população e na ordem e forma de atuação dos operadores de seleção e reprodução.

Primeiramente, uma população de indivíduos é gerada, geralmente de forma aleatória. Um requisito para essa etapa é a representatividade adequada do espaço de busca por parte dos indivíduos criados. O passo seguinte é a avaliação de cada indivíduo da população. Uma função de *fitness* é usada para aferir a aptidão de cada elemento para a solução do problema que está sendo tratado. Baseado nesse cálculo, um operador de seleção atua escolhendo de forma privilegiada os melhores indivíduos da população. Estes são então submetidos a operadores genéticos de recombinação e mutação para a composição da população que pertencerá à geração seguinte. Essa nova população é, como a original, submetida à avaliação. O processo se repete até que uma condição de parada especificada seja atendida. Algumas possibilidades de condições de parada são:

- A n-ésima geração corresponde à quantidade máxima de gerações estabelecida na inicialização do algoritmo genético, sem que se tenha conseguido descobrir pelo menos um indivíduo que satisfaça a solução do problema;

- A n-ésima geração possui pelo menos um indivíduo que seguramente satisfaça a solução do problema, sendo n um número menor que a quantidade máxima de gerações fixada para o algoritmo genético;
- O melhor indivíduo da população se repete por um número pré-estabelecido de vezes. Esta situação caracteriza o encerramento do algoritmo por estagnação do melhor indivíduo;
- A média da adaptabilidade da população não se altera por um determinado número de gerações. Este caso é chamado de encerramento por estagnação da evolução da população;
- A diversidade da população atinge um limite inferior indicando convergência.

A próxima subseção discute a importância da definição adequada dos diversos parâmetros envolvidos durante a aplicação de algoritmos genéticos. A seguinte cita alguns exemplos de aplicações promissoras ou já bem-sucedidas dessa técnica.

4.2.1 Configuração do Algoritmo Genético

Ao se utilizar algoritmos genéticos para solucionar um problema, deve-se analisar a influência dos parâmetros de ajuste dessa ferramenta. Uma determinada configuração pode ter um desempenho melhor que outra em um problema específico e não existe uma metodologia para a determinação ótima desse conjunto de parâmetros [10].

A primeira decisão relaciona-se à representação cromossômica dos indivíduos. Além da arquitetura escolhida, é preciso determinar qual será a precisão desejada. Pode-se, por exemplo, utilizar a representação binária para variáveis que na realidade são números reais. Assim, um conjunto de genes é necessário para o mapeamento de cada um desses valores. Uma precisão maior implica num maior número de genes, aumentando o tamanho do cromossomo, seu espaço ocupado na memória e acarretando perda de performance do algoritmo. Pelo contrário, uma representação com baixa precisão pode mesmo impossibilitar a determinação de uma solução adequada do problema. Para evitar esses problemas, adota-se neste trabalho uma representação utilizando números reais e inteiros, como será apresentado na seção 4.4.1.

Outro parâmetro relaciona-se ao tamanho da população. O desempenho global e a eficiência dos algoritmos genéticos são diretamente afetados por esse número. Com uma população pequena o desempenho pode ser ruim, pois neste caso a população fornece uma pequena cobertura do espaço de busca do problema. Uma população grande geralmente fornece uma cobertura representativa do domínio do problema, além de prevenir convergências prematuras para soluções locais ao invés de globais. No entanto, para se trabalhar com grandes populações, são necessários maiores recursos computacionais, sob pena de o algoritmo consumir um período de tempo muito maior que o desejável.

Arquiteturas paralelas e/ou distribuídas são necessárias para aumentar a eficiência computacional neste contexto.

Mais um fator relacionado à população é a parcela de indivíduos que será selecionada para formar a geração seguinte. Com um valor baixo, a maior parte da população será substituída, podendo ocorrer a perda de indivíduos de alta adaptabilidade. Com um valor alto, o algoritmo pode tornar-se muito lento.

Em relação à taxa de ocorrência da mutação, como será ressaltado na seção 4.7, normalmente um valor baixo é adotado. A definição adequada da taxa de recombinação também é imprescindível para o funcionamento adequado do algoritmo genético. Quanto maior for esta taxa, mais rapidamente novas estruturas serão introduzidas na população. Mas se for demasiadamente elevada, elementos com boas adaptabilidades poderão ser retirados mais rapidamente do que são gerados melhores indivíduos. Com um valor baixo, a evolução da população pode estagnar, convergindo para um mínimo local.

As metodologias adotadas para a definição desses parâmetros normalmente são empíricas, ou seja, baseadas na experiência prévia do projetista. Métodos automáticos de sintonização desses parâmetros, denominados *meta-algoritmos*, têm sido propostos na literatura. Um deles, por exemplo, explora a utilização de um sistema *fuzzy* para controlar dinamicamente a configuração de um algoritmo genético [64]. A base de conhecimento utilizada neste sistema *fuzzy* poderia ser obtida diretamente de um projetista de algoritmos genéticos experiente ou automaticamente. A seção 4.3 apresenta outras referências úteis na definição automática dos parâmetros dos algoritmos genéticos.

4.2.2 Aplicações

Soluções adaptativas são normalmente requeridas em sistemas que atuam em um ambiente dinâmico. Sistemas adaptativos, ou evolutivos, tentam resolver problemas acumulando conhecimento sobre eles e utilizando estas informações para gerar soluções razoáveis. Áreas comuns de aplicação são: configuração de sistemas complexos, alocação de tarefas, seleção de rotas e outros problemas de otimização.

Para a definição inequívoca do escopo de aplicação dos algoritmos evolutivos é necessária a introdução dos conceitos de *métodos fortes* e *métodos fracos* [79]. Os métodos fortes são aqueles concebidos para resolverem problemas genéricos mas desenvolvidos para atuarem em um ambiente específico, bem determinado, onde estão presentes características como linearidade, estacionariedade, diferenciabilidade e/ou continuidade. Uma classe ainda mais restrita são os *métodos específicos*, elaborados para resolverem problemas específicos em ambientes também específicos.

Por outro lado, os métodos fracos são concebidos para serem aplicados a problemas genéricos em ambientes também genéricos, ou seja, não requisitando as características existentes para os métodos fortes. Embora não garantam a obtenção da solução ótima do problema sendo abordado, geralmente

fornecem uma boa aproximação para essa solução, em tempo de execução computacional compatível com a necessidade do problema.

Os métodos fracos, que englobam as técnicas evolutivas, devem ser aplicados apenas quando não existem métodos fortes ou específicos efetivos para o problema em questão. A tarefa do projeto automático de um sistema *fuzzy*, que representa o enfoque central do presente trabalho, é uma área potencial de aplicação para os métodos fracos, em especial para os algoritmos evolutivos, o que é corroborado pelo volume de pesquisa nessa área (ver seção 4.3.1).

Segue uma listagem de algumas aplicações dos algoritmos genéticos:

- Controle de sistemas dinâmicos [62, 54, 118];
- Síntese de circuitos analógicos: para determinadas entradas e saídas, o algoritmo genético gera a topologia , o tipo e o valor dos componentes do circuito [60];
- Síntese de protocolos de comunicação: determinação de quais funções do protocolo devem ser implementadas em *hardware* e quais devem ser implementadas em *software* para que o melhor desempenho seja alcançado [69];
- Gerenciamento de redes: supervisão do tráfego nos *links* e das filas nos *buffers* de roteadores para descobrir rotas ótimas e para reconfigurar as rotas existentes no caso de falha de algum *link* [42];
- Otimização evolutiva multi-critério: otimização de funções com múltiplos objetivos que sejam conflitantes [25];
- Problemas de otimização complexos. Casos típicos são problemas de alocação, localização, roteamento de veículos e o problema do caixeiro viajante, todos com grande potencial de aplicação na área de logística [8, 63].
- Ciências biológicas: modelagem de processos biológicos para o entendimento do comportamento de estruturas genéticas [59];
- Definição da arquitetura de redes neurais [49, 101, 110].
- Determinação e otimização dos componentes de sistemas *fuzzy* (ver referências na seção 4.3.1);

Como foi afirmado na seção 3.3 do capítulo 3, o tema desta dissertação relaciona-se (porém não é restrito) ao último item da listagem anterior, considerando no entanto sistemas *fuzzy* TS baseados em funções de bases ortonormais. A seção seguinte é dedicada a uma descrição mais detalhada dessa área de pesquisa.

4.3 Projeto Automático de Modelos *Fuzzy*

As pesquisas recentes têm mostrado que sistemas inteligentes híbridos fornecem métodos eficientes para aplicações práticas. Ao se compensar as deficiências de uma técnica com os benefícios de outra, criam-se estruturas de enorme potencial [18, 48, 97].

Desde a última metade da década de 80 têm sido realizadas intensas pesquisas nos campos das redes neurais, sistemas *fuzzy* e algoritmos genéticos. O grande número de publicações nessa área indica que estudos combinando essas diferentes ferramentas têm aumentado significativamente. Em particular, combinações de redes neurais e sistemas *fuzzy* já foram incorporadas em diversos produtos [48, 106].

Cada metodologia tem suas vantagens. Por exemplo, os algoritmos genéticos e as redes neurais possuem a desejável característica de adaptabilidade. Sistemas *fuzzy* e redes neurais podem aproximar funções não lineares. Sistemas *fuzzy* podem ainda incorporar o conhecimento especializado de um ser humano de forma transparente. Já as redes neurais podem tratar dessas informações implicitamente, além da característica de aprendizagem. Os algoritmos genéticos caracterizam-se pelas propriedades de busca local e global.

Um exemplo de problema que motivaria a aplicação de um sistema inteligente híbrido relaciona-se à definição adequada dos parâmetros de um algoritmo genético. As taxas de ocorrência dos operadores de reprodução, o tamanho da população, a precisão na representação dos indivíduos, podem, por exemplo, ser automaticamente calculadas utilizando-se um sistema *fuzzy* [64, 43, 103].

Normalmente a configuração oposta é mais comumente pesquisada, qual seja, a utilização de algoritmos genéticos para a determinação dos parâmetros ótimos de um sistema *fuzzy*, ou mesmo de um sistema neuro-*fuzzy*.

Para este último caso, relata-se por exemplo o uso de algoritmos genéticos na definição adequada das funções de pertinência para um sistema *fuzzy* cujas regras de controle são aprendidas através do treinamento não supervisionado de uma rede neural, a qual também implementa as operações de inferência e *defuzzyficação* [102]. Uma peculiaridade desse estudo é o uso de uma taxa de ocorrência variante para o operador de mutação. Estipula-se que resultados melhores são encontrados ao decrescer essa taxa conforme o algoritmo aproxima-se da solução ótima [35].

Outro trabalho nesse sentido foca o uso de um modelo chamado NEFCON, *NEural Fuzzy CONtroller* [5]. O algoritmo de aprendizagem utilizado é baseado em reforço. Novamente, um algoritmo genético é empregado para otimizar os parâmetros do sistema *neurofuzzy*, aplicado ao controle de plantas não lineares e com retardo.

No caso do uso de algoritmos evolutivos em sistemas *fuzzy*, a meta geralmente estabelecida é a definição de um conjunto de funções de pertinência (de entrada e/ou de saída) além da geração automática da base de regras de inferência [65, 31, 24, 94, 78]. A próxima seção apresenta uma

revisão sucinta da evolução das pesquisas envolvendo os chamados *Genetic Fuzzy Systems* - GFS [17, 18].

4.3.1 Evolução dos Sistemas Genético-*Fuzzy*

No início da década de 90 são encontrados na literatura científica os primeiros artigos descrevendo o uso de algoritmos genéticos (AG) para o projeto de sistemas *fuzzy*. Desde então, têm sido propostas arquiteturas cada vez mais sofisticadas: é definido automaticamente um maior número de parâmetros, é viabilizada uma maior flexibilidade durante o projeto, são considerados múltiplos objetivos, é realizada a integração com outros métodos de otimização e finalmente foram realizadas aplicações complexas em áreas antes inexploradas. Segue uma breve revisão que ilustra a evolução da pesquisa nessa área.

Lee e Takagi propuseram em 1993 [65] um método para projetar automaticamente uma série de elementos de um sistema de controle *fuzzy*. Neste trabalho são referenciados outros desde o ano de 1989, que não consideravam todos os componentes do sistema *fuzzy* simultaneamente. A proposta de Lee e Takagi determinava, utilizando algoritmos genéticos, as funções de pertinência (FP) para um número pré-determinado de variáveis de entrada, o número de regras e os parâmetros dos conseqüentes das regras de um sistema do tipo Takagi-Sugeno-Kang. A aplicação consistiu no controle de um pêndulo invertido.

Em 1994, Liska e Melsheimer [71] propuseram um AG para o projeto de um sistema *fuzzy*, englobando:

- Número de regras e suas estruturas;
- Parâmetros das funções de pertinência;
- Uso do método do gradiente conjugado para melhorar o resultado final.

A aplicação inicial foi a modelagem de um sistema dinâmico não linear, utilizando um modelo *fuzzy* do tipo Mamdani.

Controladores *fuzzy* PI e PD foram abordados por Ng e Li em 1994 [82]. Um algoritmo genético foi aplicado para otimizar as funções de pertinência exponenciais parametrizadas pelas constantes α, β e σ, conforme a equação (4.1).

$$\mu(x) = exp\left(-\frac{|x \pm \alpha|^{\beta}}{\sigma}\right) \tag{4.1}$$

Além disso, a base de regras completa do sistema do tipo Mamdani e os ganhos proporcional e integral dos controladores foram também obtidos com o algoritmo genético. A aplicação foi um

sistema de controle de nível em um tanque.

Ainda em 1994, Park *et al.* [91] abordaram o modelo *fuzzy* relacional. Um AG obtém simultaneamente a matriz *fuzzy* relacional e os parâmetros das funções de pertinência triangulares, de quantidade fixa. Efetua-se uma aplicação no controle de um motor DC.

Homaifar e McCormick, em 1995 [47], projetaram um sistema de controle *fuzzy* do tipo Mamdani, com uma base de regras completa. Um AG foi usado para calcular simultaneamente as FP triangulares de 2 variáveis de entrada e a base de regras. Esta metodologia foi utilizada para o controle de posicionamento de um carro (*Cart Controller*) e de um caminhão (*Truck-backing*).

Um modelo hierárquico distribuído para um AG foi proposto em 1996 por Kim e Zeigler [56]. Na arquitetura proposta divide-se o problema em duas etapas. Na primeira, são analisados grandes *clusters* com pouco detalhamento. Na segunda, explora-se com maior precisão os *clusters* mais promissores, em níveis maiores de detalhamento. Aplica-se o método a um controlador *fuzzy* Mamdani em uma simulação da produção de oxigênio para a atmosfera marciana. A arquitetura otimiza o tipo e quantidade das FP. A base de regras é completa e simétrica (fixa, não otimizada). São implementadas restrições para impedir o crescimento exponencial da base de regras.

Linkens e Nyongesa compilaram em 1996 [70] trabalhos sobre o uso de técnicas inteligentes de controle. É dada ênfase em lógica fuzzy, redes neurais e algoritmos genéticos, bem como em suas combinações. Dentre as referências inclusas nesse trabalho, citam-se as seguintes arquiteturas: base de regras fixa, com otimização das FP; FP fixas com otimização da base de regras; projeto simultâneo dos dois itens anteriores; e aplicações em sistemas genético-*fuzzy* em tempo real (controle robótico e *schedule*).

Em 1997 Jikai *et al.* [50] elaboraram um sistema para otimização *on-line* de um controlador neuro-*fuzzy* com um AG. Este, em um primeiro momento, codificava os pesos sinápticos da rede neuro-*fuzzy* e procedia com a otimização. Em seguida as FP gaussianas também eram otimizadas. A aplicação concentrou-se no controle em tempo real de um forno elétrico.

Marco Russo, em 1998 [98], desenvolveu um sistema para aplicação em classificação e modelagem, que também englobava os três componentes maiores da inteligência computacional: sistemas *fuzzy*, algoritmos genéticos e redes neurais. Dentre as características do sistema projetado, cita-se uma codificação de comprimento proporcional (e não exponencial) ao número de parâmetros e um método de simplificação de base de regras (através da inclusão de um termo de penalização na função de *fitness*). O método de busca local *hill-climbing* é implementado para acelerar o processo de aprendizagem. Nesta implementação, o método converte um determinado indivíduo para uma rede neuro-*fuzzy*, treina-a e então retorna à codificação cromossômica no AG, com um valor de adaptação (*fitness*) maior.

Em 1999 Kumar e Wu [57] elaboraram um sistema para obtenção de modelos para séries tempo-

rais usando AG e teorias de sistemas *fuzzy*. Testa-se o sistema em um conjunto de dados simulados e na previsão do comportamento dos mercados de ações de Taiwan e da Malásia. Em ambos os casos, os resultados obtidos são melhores que aqueles fornecidos por um modelo ARIMA.

Semelhantemente ao trabalho de Homaifar e McCormick [47], Wu e Liu desenvolveram em 2000 [116] um AG que otimiza as FP triangulares de entrada e a base de regras completa de um controlador *fuzzy* do tipo Mamdani. As melhorias no método referem-se a uma maior flexibilidade no desenvolvimento do sistema *fuzzy*, através da inclusão de um número maior de parâmetros de projeto no algoritmo genético. Outras características são uma codificação com números reais, o uso do operador de *crossover* convexo, de mutação não-uniforme e seleção por *rank*. A aplicação também é em um sistema de controle de posicionamento.

Delgado *et al.* introduziram um sistema genético-*fuzzy* hierárquico em 2001 [22]. A arquitetura conta com o uso do método dos mínimos quadrados para a estimação (local ou global) dos conseqüentes não lineares de modelos TS, além de um procedimento de poda para a eliminação de redundância nas regras. Testa-se o sistema em problemas de classificação e aproximação de funções. Em 2002, Delgado *et al.* deram continuidade ao trabalho [25]. Um sistema genético-*fuzzy* gera autonomamente todo um sistema classificador, desde a forma e quantidade de funções de pertinência das variáveis de entrada até a base de regras. Uma arquitetura hierárquica é usada, formada por um conjunto de populações sujeitas a um processo de co-evolução. Implementa-se uma otimização multicritério (considerando acurácia, interpretabilidade, simplicidade e autonomia) baseada em algoritmos genéticos.

Em 2004 foi publicado um artigo por Cordón *et al.* [18] retratando o estado da arte da pesquisa em sistemas genético-*fuzzy*. São descritas as principais arquiteturas já propostas, as aplicações já realizadas e as novas tendências nessa área. Uma análise crítica da importância dos algoritmos genéticos durante o projeto de sistemas *fuzzy* é efetuada, levantando aspectos como as diferentes abordagens possíveis (com maior ou menor grau de autonomia no projeto), as poucas alternativas existentes e a necessidade de formalização de um método de avaliação de desempenho de um GFS. Em seguida são apresentadas algumas questões que ainda permanecem em aberto, incitando a pesquisa contínua para a melhoria do desempenho das arquiteturas atuais bem como para a criação de outras novas.

Algumas publicações recentes estão focadas na aplicação de sistemas genético-*fuzzy* dentro de um contexto maior, não sendo estes os destaques maiores da pesquisa científica. Dessa forma, relata-se a aplicação de diversas configurações dos GFS como parte de sistemas complexos, os quais abordam, por exemplo, pré-processamento de dados em sistemas de potência [58] ou controle multi-agente de manipuladores robóticos [30]. Neste último caso, um sistema de controle *fuzzy* do tipo Mamdani é operado por 3 agentes independentes. Trata-se do controle de um robô com a tarefa de interagir com um humano. São ponderados para a avaliação do desempenho do controlador o tempo de atuação, o

erro em termos da distância a uma trajetória pré-estabelecida e a energia de operação. Um algoritmo genético é aplicado na otimização das FP de entrada e saída dos três controladores. Não há critério de interpretabilidade do resultado, podendo ser obtidas, por exemplo, FP sobrepostas (o que de fato ocorre nos resultados registrados).

Leung *et al.* trabalharam em 2004 com um controlador *fuzzy* para sistemas não lineares sujeitos a incertezas em seus parâmetros [66]. O enfoque maior da pesquisa é o desenvolvimento de fundamentos teóricos para a garantia de estabilidade do sistema. Um AG é usado na identificação de um ganho de controle e na solução da equação para a estabilidade. Em seguida, novo AG é usado na otimização das FP do controlador. Aplica-se o método à estabilização de um manipulador robótico com dois braços.

A seção seguinte apresenta o resumo da proposta desse trabalho. Trata-se de uma combinação entre o uso de procedimentos bem estabelecidos, a extensão de metodologias proeminentes e a sugestão de uma nova arquitetura de representação.

4.3.2 Proposta para Otimização do Sistema *Fuzzy* TS FBO

A principal contribuição científica do trabalho descrito nessa dissertação é a proposta de uma arquitetura para o projeto automático de sistemas *fuzzy* TS FBO e sistemas *fuzzy* TS FBO Generalizado utilizando algoritmos genéticos.

A arquitetura proposta baseia-se em um nível único de representação, ou seja, um cromossomo codifica toda a solução do problema. Dada a complexidade da tarefa de se projetar automaticamente todo um sistema *fuzzy*, algumas abordagens envolvendo co-evolução adotam níveis hierárquicos de codificação [24, 94], com várias populações representando elementos distintos dos sistemas *fuzzy*. Segundo essa metodologia, indivíduos de uma população são formados pela combinação de indivíduos de outras populações. A avaliação do *fitness* de um elemento é função do *fitness* de seus subcomponentes. As abordagens co-evolutivas apresentam uma estrutura que facilita uma implementação paralela.

O presente trabalho se propõe a mostrar que é possível a obtenção de bons resultados utilizando apenas um nível de representação, desde que os operadores genéticos utilizados levem em conta a codificação elaborada, considerando as diferentes semânticas em diferentes seções do cromossomo. Dessa forma, por exemplo, o operador de mutação por inversão (ver seção 4.7) está imediatamente descartado, uma vez que pode alocar a um gene um alelo incompatível com a semântica daquela posição.

Além disso, são aplicados critérios especiais para a obtenção de sistemas parcimoniosos, porém com bom desempenho na tarefa de modelagem (critério de Akaike, ver seção 4.5.1), e para a manutenção da interpretabilidade característica de um sistema de aproximação do raciocínio humano

(medidas de similaridade, ver seção 4.9).

Os parâmetros de projeto incluídos no algoritmo genético foram apresentados na seção 3.3 do capítulo 3. A partir da próxima seção são detalhados os vários componentes e processos que integram um algoritmo genético e apresentadas as propostas específicas dessa dissertação para o projeto automático dos modelos *fuzzy* TS FBO e *fuzzy* TS FBO Generalizado.

4.4 Representação Cromossômica

O primeiro passo para a utilização dos algoritmos genéticos é a definição da representação cromossômica dos indivíduos da população. Os cromossomos são compostos por genes, dígitos alfanuméricos, que serão, tal como na biologia, alterados quando da reprodução. Segundo a abordagem Pittsburgh, cada elemento é uma possível solução do problema (ao contrário da abordagem Michigan, na qual a população como um todo é a solução) [94]. O conjunto de todas as configurações que o cromossomo pode assumir forma o espaço de busca do algoritmo genético. Assim, caso o cromossomo represente n parâmetros de uma função, o espaço de busca resultante é um espaço de n dimensões.

Tradicionalmente, os indivíduos são representados por um vetor binário [46], como ilustrado na Figura 4.1, onde cada elemento do vetor (gene) 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 seja, o seu fenótipo. Teoricamente, esta representação é independente do problema. Uma vez encontrada a representação em vetores binários, as operações padrão de reprodução podem ser utilizadas.

1	1	0	1	0	0

Figura 4.1: Representação cromossômica binária.

Dependendo do problema a ser tratado, outras representações podem ser mais eficientes e, portanto, adotadas. Pode-se pensar em se utilizar números naturais, caracteres, ou mesmo uma combinação dos dois. Um exemplo seria a representação de uma solução do problema de roteamento de veículos. A Figura 4.2 exibe um cromossomo identificando três tipos de veículos (A, B e C) atendendo um conjunto de localidades, rotuladas como 4, 9 e 12 (veículo A), 5 e 2 (veículo B) e 7 e 1 (veículo C).

A	4	9	12	B	5	2	C	7	1

Figura 4.2: Representação cromossômica mista.

Relata-se ainda o uso de representações com valores reais, através de permutações ou por meio de árvores hierárquicas [6]. Mesmo a representação binária apresenta a variante de se adotar o código Gray [6]. A escolha pela representação a ser utilizada é dependente da natureza do problema em estudo. Além disso, os operadores genéticos, abordados na seção 4.7, também estão intrinsecamente conectados ao tipo de representação adotada.

A primeira população de indivíduos normalmente é criada de maneira aleatória. É recomendado apenas o cuidado para que seus indivíduos tenham uma ampla representatividade no domínio do problema em estudo. Em caso contrário, corre-se o risco do processo de busca paralisar em um ponto de máximo local, em um problema de maximização, ou mínimo local, em um problema de minimização, fornecendo assim um resultado indesejável para a solução global do problema.

Um importante fator a ser considerado durante a elaboração da representação cromossômica é a manutenção da validade dos indivíduos durante o processo evolutivo. Idealmente, após a criação da população, todos os operadores genéticos devem manter a validade de um indivíduo, no sentido de que este continue representando uma solução factível do problema. Caso tal factibilidade possa vir a ser violada, algum procedimento deve ser executado para corrigir, ou eliminar, o indivíduo problemático. Uma primeira opção é aplicar um método de verificação e correção em cada indivíduo após a atuação dos operadores de reprodução. Obviamente, a etapa corretiva pode demandar alto custo computacional, devendo ser postergada o máximo possível. A segunda opção é a implementação de uma função de penalização dos indivíduos que representam soluções infactíveis. Tal função deve ser cuidadosamente elaborada para que não faça com que o algoritmo genético forneça, ao final do processo evolutivo, soluções simplesmente factíveis, sem a otimização do critério de desempenho (ver seção 4.5).

4.4.1 Proposta de Representação Cromossômica

Duas representações cromossômicas foram elaboradas, uma para a arquitetura *fuzzy* TS FBO com pólo único[1] e outra para a *fuzzy* TS FBO Generalizado. Em ambas as representações, diferentes elementos do sistema são codificados no mesmo cromossomo, com os seguintes objetivos em mente:

1. A representação seja simples, transparente;

2. Os operadores genéticos sejam aplicados entre elementos com a mesma semântica. Por exemplo, durante um *crossover* não são combinados genes representando um pólo do modelo com genes representando o centro de uma função de pertinência;

[1] Quando o contexto não for suficiente para distinguir as duas arquiteturas, falar-se-á em "*fuzzy* TS FBO com pólo único" referindo-se à arquitetura descrita na seção 3.2.3 para diferenciá-la da arquitetura "*fuzzy* TS FBO Generalizado", descrita na seção 3.2.4.

3. Parâmetros que não tenham uma quantidade fixa pré-determinada (como o número de estados nas premissas das regras) possuem certo grau de liberdade para aumentar ou diminuir seu número;

4. Restrições de integridade sejam automaticamente preservadas, ou facilmente passíveis de correção, durante a aplicação dos operadores genéticos;

5. A representação não seja esparsa.

Em ambas as representações descritas a seguir, é preciso representar pólos complexos que definem a dinâmica de estados de um modelo. Adota-se uma representação em coordenadas cartesianas, para se atender ao primeiro requisito anteriormente definido e dispensar a conversão entre coordenadas (polar para cartesiana) em cada avaliação de *fitness*. No entanto, na população inicial os pólos são gerados em coordenadas polares e são garantidamente estáveis (contidos no interior do círculo unitário). Seus valores em coordenadas cartesianas são então trivialmente obtidos.

As funções de pertinência dos termos lingüísticos são escolhidas como gaussianas, parametrizadas por dois valores: seu centro e sua abertura. Essa escolha não é crítica uma vez que tem sido confirmado que a maior importância na definição das partições das variáveis de entrada de sistemas *fuzzy* reside na disposição e grau de sobreposição das funções de pertinência[2], e não propriamente em seu formato (se gaussiana, triangular ou trapezoidal) [93]. Espinosa *et al.* exibem um estudo no qual concluem que modelos *fuzzy* TS com funções de pertinência gaussianas geram interpolações não lineares das vizinhanças dos modelos locais [32]. Esses modelos apresentam melhores resultados em uma série de exemplos de aproximação de funções, comparados a modelos utilizando funções de pertinência triangulares ou polinomiais. De fato, a maioria dos resultados de aproximação universal em sistemas *fuzzy* assume o uso de funções de pertinência gaussianas [14, 39, 61, 115].

A arquitetura aqui proposta permite ainda duas interpretações para a codificação das funções de pertinência. Na primeira, os valores de seus centros são absolutos. Na segunda, as posições de seus centros são relativas. Apenas o valor do centro da primeira função de pertinência é absoluto. A partir da segunda, o valor real do centro é obtido somando-se seu valor de centro a todos os anteriores, ou seja, o parâmetro centro define a distância entre os centros de funções de pertinência vizinhas. Essa implementação possibilita o controle do grau mínimo ou máximo de sobreposição entre as funções de pertinência de forma trivial, bastando para isso impor esses limites diretamente aos respectivos genes do cromossomo. Claramente, o controle das aberturas (ou larguras) das funções de pertinência pode ser feito independentemente da interpretação dada a seus centros.

[2]Esta tarefa pode ser associada à definição de uma *moldura cognitiva*, a qual é composta por vários conjuntos *fuzzy* normais usados como pontos de referência para o processamento de informações *fuzzy* [93]. As propriedades mais importantes relativas às molduras cognitivas, como especificidade e foco de atenção, não fazem menção ao tipo de função de pertinência utilizada.

Esta segunda interpretação (posição relativa dos centros) é uma opção natural da arquitetura proposta, porém a técnica das medidas de similaridades, descrita na seção 4.9, é além de um método eficiente no controle da sobreposição entre funções de pertinência, um mecanismo de simplificação de modelos *fuzzy*. Sendo assim, a primeira interpretação apresentada será adotada neste trabalho.

A inicialização de todos os parâmetros codificados na população é feita segundo uma distribuição uniforme, já que em princípio não há informações sobre regiões do espaço de busca probabilisticamente mais promissoras, onde se poderia, por exemplo, centrar Gaussianas (ver seção 4.7.1).

Uma variável aleatória com distribuição uniforme em um intervalo $[a, b]$ é tal que todo número nesse intervalo possui a mesma probabilidade de ocorrência. Dessa forma espera-se que a inicialização da população seja representativa de todo o espaço de busca, desde que, para cada parâmetro sendo otimizado, sejam estipulados os limites a e b de seus universos. Esta etapa depende do problema sendo tratado e do conhecimento sobre as ordens de grandeza dos parâmetros codificados no algoritmo genético. Como exemplo, a decisão pela inicialização das partes real e imaginária dos pólos dos modelos *fuzzy* TS FBO estabelece o intervalo $[0, 1]$. Para a parte real do pólo, a justificativa para se trabalhar apenas no semiplano direito do círculo unitário[3] é que não existe sistema equivalente contínuo para sistemas discretos (como é o caso) com pólos no semiplano esquerdo do plano complexo z. Em relação à parte imaginária do pólo, como este é sempre complexo conjugado (base de Kautz) basta representar o semiplano superior, pois o semi plano inferior é um espelho [41].

Descreve-se a seguir a representação para o modelo *fuzzy* TS FBO com pólo único e em seguida para o modelo Generalizado.

Representação para o modelo *fuzzy* TS FBO com pólo único

Para essa arquitetura, os modelos locais são parametrizados no mesmo pólo complexo, ou seja, possuem a mesma dinâmica de estados. O que diferencia cada modelo local são as relações estáticas desses estados. A arquitetura proposta é mais flexível que no caso do sistema *fuzzy* TS FBO Generalizado, descrito adiante nesta seção, no sentido de que permite maior autonomia no projeto do modelo *fuzzy*. Primeiramente não é necessária a informação *a priori* da quantidade de modelos locais a ser obtida. Em adição, é permitido ao algoritmo variar o número de funções de pertinência por variável de entrada, além de variar o próprio número de variáveis de entrada (estados). O número de estados presentes nos conseqüentes das regras também é otimizado. A representação matricial elaborada é facilmente interpretável.

A Figura 4.3 ilustra o cromossomo elaborado para a representação em questão. Trata-se de uma matriz com valores reais e inteiros. As únicas informações que devem ser previamente fornecidas ao algoritmo genético são os números máximos de variáveis de entrada (estados) e de funções de

[3] Os pólos são inicializados dentro do círculo unitário para a garantia de sua estabilidade.

pertinência por variável, dados respectivamente por max_{vl} e max_{tl}.

α	σ_{11}	ξ_{11}	σ_{21}	ξ_{21}	$\cdots$	$\sigma_{max_{vl}1}$	$\xi_{max_{vl}1}$	k
ω	σ_{12}	ξ_{12}	σ_{22}	ξ_{22}	$\cdots$	$\sigma_{max_{vl}2}$	$\xi_{max_{vl}2}$	
	σ_{13}	ξ_{13}	σ_{23}	ξ_{23}	$\cdots$	$\sigma_{max_{vl}3}$	$\xi_{max_{vl}3}$	
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	
	$\sigma_{1max_{tl}}$	$\xi_{1max_{tl}}$	$\sigma_{2max_{tl}}$	$\xi_{2max_{tl}}$	$\cdots$	$\sigma_{max_{vl}max_{tl}}$	$\xi_{max_{vl}max_{tl}}$	

Figura 4.3: Representação cromossômica para o modelo *fuzzy* TS FBO com pólo único.

Na Figura 4.3, o pólo de Kautz é representado pelos parâmetros α e ω (coordenadas cartesianas, da forma $\alpha \pm j\omega$), na primeira coluna da matriz. Na seqüência das colunas, cada par representa o conjunto de funções de pertinência associadas à respectiva variável de entrada. Dessa forma, a segunda e terceira colunas codificam as funções de pertinência da primeira variável de entrada, sendo as aberturas determinadas pelos parâmetros σ_{ij} e os centros por ξ_{ij}, com os índices i e j referenciando a variável de entrada e o termo lingüístico, respectivamente. Os limites máximos para essas quantidades são, respectivamente, max_{vl} e max_{tl}. O número de estados nos conseqüentes das regras é determinado pelo parâmetro k (número comum de estados em todos os modelos locais).

Nesta representação cromossômica deve-se observar alguns aspectos. Primeiramente, uma variável deve possuir ao menos dois termos lingüísticos (funções de pertinência) para que sua existência seja justificada. Assim, caso em algum momento do processo de otimização reste apenas um termo lingüístico para determinada variável, este termo deve ser excluído, assim como a respectiva variável. Esta ação é tomada pois uma variável com um único termo lingüístico pode ser tida como o próprio universo de discurso, não representando uma partição significativa. Outra condição que deve ser satisfeita é que o número k de estados nos conseqüentes das regras não seja inferior à quantidade de variáveis (estados) nas premissas das regras. Embora sejam permitidas até max_{vl} variáveis de entrada, um cromossomo pode exibir um número menor (as duplas de colunas correspondentes às variáveis inexistentes são nulas). O parâmetro k, que também define a dimensão do vetor de estados da base ortonormal, deve então sempre ser maior ou igual ao número efetivo de variáveis nas premissas das regras. Finalmente, os operadores de recombinação (*crossover*) implementados devem respeitar a semântica dos elementos no cromossomo, bem como suas restrições de valoração.

Computacionalmente, a população de indivíduos com esta representação é uma matriz de dimensão $N(max_{tl}(2max_{vl} + 2))$, sendo N o número de indivíduos na população. Diferentemente da representação apresentada na próxima seção, alguns *loci* de um cromossomo podem vir a ter valores nulos. Esta condição ocorre quando uma variável lingüística não utiliza todos os termos lingüísticos possíveis, ou o número máximo de variáveis não é alcançado. Em termos de eficiência computacional, esta é uma desvantagem dessa representação. A aplicação das medidas de similaridade para

simplificação do sistema *fuzzy* (ver seção 4.9) tende a criar ainda mais lacunas. Uma alternativa seria o uso de uma estrutura de dados mais sofisticada, como uma lista duplamente encadeada. No entanto, a representação matricial é extremamente simples e transparente, agilizando consideravelmente a aplicação dos operadores genéticos, justificando assim sua adoção.

Representação para o modelo *fuzzy* TS FBO Generalizado

Nesta implementação, cada modelo local é parametrizado em um pólo complexo (base de Kautz). O número de modelos locais deve ser fornecido por um especialista humano no processo sendo modelado, ou seja, é uma entrada do algoritmo. Essa restrição é justificada devido ao alto grau de complexidade na elaboração de uma arquitetura algorítmica e computacional ao mesmo tempo flexível e eficiente para a otimização também do número de modelos locais do sistema *fuzzy*. Na arquitetura ora proposta o número de modelos locais corresponde ao número de regras do sistema *fuzzy*, número este dado pela combinação de todos os termos lingüísticos das variáveis lingüísticas de entrada (base de regras completa).

Assim, considerando o conhecimento do número r de modelos locais esperados para o modelo *fuzzy*, os parâmetros livres de projeto, otimizados pelo algoritmo genético, são: os r pólos, compostos pelo par abcissa e ordenada (valores reais); um conjunto de m funções de pertinência gaussianas para cada um dos n estados nas premissas das regras[4], compostas por um valor de centro e um de abertura (valores reais); r números indicando as quantidades de estados nos conseqüentes das regras (valores inteiros); e n inteiros especificando quais estados estarão presente nas premissas das regras. Adota-se uma representação para o cromossomo na forma de um vetor, conforme ilustrado na Figura 4.4.

α_1	ω_1	$\ldots$	α_r	ω_r	σ_{11}	ξ_{11}	$\ldots$	σ_{1m}	ξ_{1m}	$\ldots$	σ_{n1}	ξ_{n1}	$\ldots$

$\ldots$	σ_{nm}	ξ_{nm}	k_1	$\ldots$	k_r	L_1	$\ldots$	L_n

Figura 4.4: Representação cromossômica para o modelo *fuzzy* TS FBO Generalizado.

Nessa figura, os pólos dos r modelos locais são parametrizados em coordenadas cartesianas, na forma $\alpha_i \pm j\omega_i$; as aberturas das m funções de pertinência das n variáveis de entrada correspondem aos parâmetros σ_{ij} e os centros aos ξ_{ij}; os números de estados nos conseqüentes das r regras são codificados por k_i; e os índices dos n estados nas premissas por L_i. Assim, para uma base de regras completa, existem $r = m^n$ regras.

Nessa representação, o cromossomo possui tamanho fixo. Cada *locus* (posição) é ocupado por um alelo significativo (ao contrário da representação anterior, que permite *loci* vazios). A população é

[4]Por simplicidade o número m é fixo e igual para todos os estados.

formada por uma matriz na qual cada linha representa um indivíduo codificado na forma da Figura 4.4. A Figura 4.5 apresenta um exemplo numérico de um cromossomo e o respectivo modelo codificado. Trata-se de um modelo com duas regras (logo, dois pólos, uma variável de entrada e duas funções de pertinência).

0,27	0,44	0,13	0,56	1,12	-0,47	1,70	1,08	5	3	2

$$\Downarrow$$

$$R^1: \text{ se } (\ \zeta_2(k) \text{ é } A_{1,2}\ ) \text{ então } \begin{cases} \mathbf{l}_1(k+1) &= \mathbf{A}_1\,\mathbf{l}_1(k) + \mathbf{b}_1\,u(k) \\ y_1(k) &= f_1(\mathbf{l}_1(k)) \end{cases}$$

$$R^2: \text{ se } (\ \zeta_2(k) \text{ é } A_{2,2}\ ) \text{ então } \begin{cases} \mathbf{l}_2(k+1) &= \mathbf{A}_2\,\mathbf{l}_2(k) + \mathbf{b}_2\,u(k) \\ y_2(k) &= f_2(\mathbf{l}_2(k)) \end{cases}$$

Figura 4.5: Exemplo numérico da representação cromossômica para o modelo *fuzzy* TS FBO Generalizado.

Os parâmetros $\mathbf{A}_1$ e $\mathbf{b}_1$ do modelo local da regra R^1, bem como o vetor de estados $\mathbf{l}_1(k)$ com 5 estados, são obtidos a partir do primeiro pólo codificado no cromossomo, 0,27 $\pm j$ 0,44. Os respectivos parâmetros da segunda regra são obtidos a partir do segundo pólo, 0,13 $\pm j$ 0,56. O segundo modelo local possui 3 estados no vetor de estados $\mathbf{l}_2(k)$. As duas funções de pertinência presentes na premissa das regras ($A_{1,2}$ e $A_{2,2}$), cuja variável de entrada é o resultado da inferência *fuzzy* sobre o segundo estado ortonormal dos modelos locais, possuem aberturas iguais a 1,12 e 1,70 e centros iguais a -0,47 e 1,08, respectivamente.

4.5 Avaliação

Após a definição da representação cromossômica, é necessário elaborar uma estratégia de avaliação dos indivíduos da população. Esta função de avaliação depende do problema sendo tratado. Uma propriedade que ela deve possuir é explicitar quais cromossomos representam as melhores configurações para a solução do problema em questão, assim como apontar aqueles que geram elementos insatisfatórios. Convenciona-se que essa função, também chamada função de *fitness*, deve atribuir valores mais altos aos indivíduos melhores, mais aptos a resolver o problema de otimização, e valores menores às soluções pobres. Além disso, são comumente usadas funções não negativas.

Usualmente, para se calcular o *fitness* (valor da função de avaliação) de um indivíduo, simula-se a sua atuação no ambiente do problema sendo tratado. Na maioria dos casos é necessário decodificar o cromossomo e obter a proposta de solução. Assim, caso os cromossomos representem sistemas classificadores, por exemplo, sua avaliação consiste em se testar sua capacidade de classificação sobre um conjunto de padrões de teste. Quanto maior o número de padrões corretamente classificados, maior será seu *fitness*. Se os indivíduos representam os parâmetros de um controlador, durante a avaliação pode-se medir o desempenho deste em malha fechada frente uma trajetória de referência pré-estabelecida. Neste caso, quanto menor a medida do erro quadrático médio entre o sinal de referência e a saída da planta, maior será o valor do *fitness*. Em um problema de roteamento de veículos, os indivíduos que codificam as rotas com menor custo apresentarão também valores de *fitness* maiores.

A função de *fitness* pode ainda incorporar outras informações não relacionadas estritamente ao desempenho do indivíduo codificado. Medidas de penalização, por exemplo, podem ser incluídas. Tais penalizações podem se referir à infactibilidade de uma solução, à violação de restrições mais brandas ou ao número excessivo de parâmetros em uma solução (diretamente relacionado à complexidade da solução). Em problemas de otimização multi-critério, durante a otimização de funções com múltiplos objetivos que sejam conflitantes, a definição da função de *fitness* exige ainda mais atenção, tornando-se necessário um tratamento analítico cuidadoso.

Dentro do contexto dessa dissertação, a avaliação é realizada comparando-se a resposta dos modelos obtida em série sintética (simulação do modelo sem a utilização da saída do sistema real como entrada do modelo) com o comportamento real dos sistemas sendo modelados, ambos sujeitos a um mesmo sinal de excitação. É nessa etapa que o método dos mínimos quadrados estima o vetor $\boldsymbol{\theta}$ dos coeficientes dos conseqüentes das regras, completando assim o modelo para sua avaliação. A função de *fitness* desenvolvida considera tanto o desempenho do modelo quanto sua complexidade.

4.5.1 Critérios para Análise da Complexidade do Modelo

A função de *fitness* desenvolvida inclui um termo de penalização pela complexidade do modelo, além de considerar o desempenho da solução codificada. Vários critérios de penalização já foram propostos, a maioria baseada em métodos estatísticos do desempenho do modelo [7].

O critério de mínimo comprimento de descrição (MDL, *Minimum-Description-Length*) é baseado em um compromisso entre o erro residual do modelo e sua complexidade. Um modelo que minimiza o critério MDL é ótimo no sentido de ser uma estimação consistente do número de parâmetros e ao mesmo tempo alcançar um erro mínimo. Formalmente, supondo que z_i seja uma seqüência de observações de uma variável aleatória Z, caracterizada pela função densidade de probabilidade $p_Z(\theta)$,

o critério MDL tem a forma:

$$\text{MDL}(k) = -\log p(z|\hat{\theta}) + \frac{k}{2}\log N \tag{4.2}$$

sendo $\hat{\theta}$ a estimativa de máxima verossimilhança do vetor de parâmetros θ, $p(z|\hat{\theta})$ a função de densidade de probabilidade condicional de $p_Z(\theta)$, k o número de parâmetros no modelo e N o número de observações. O primeiro termo da equação (4.2) pode ser interpretado como o número de $bits$ necessário para codificar as observações, enquanto o segundo termo como o número de $bits$ necessário para codificar os parâmetros do modelo [105].

Uma proposta semelhante, chamada mínimo comprimento de mensagem (MML, *Minimum Message Length*), possui também dois termos. O primeiro analisa a estimativa dos parâmetros desconhecidos do modelo e o segundo é uma função das observações, a qual utiliza uma codificação ótima baseada na distribuição de probabilidade dos dados obtida pela estimativa dos parâmetros. O critério MML difere do MDL basicamente em sua implementação e na visão probabilística de seus idealizadores [114].

Um terceiro critério é o critério de informação de Akaike (AIC, *Akaike information criterion*), cuja idéia essencial é estabelecer quantos parâmetros incluir no modelo minimizando o erro de modelagem. É definido como [3]:

$$\text{AIC}(n_\theta) = N\ln[\sigma^2_{erro}(n_\theta)] + 2n_\theta \tag{4.3}$$

sendo N o número de observações, $\sigma^2_{erro}(n_\theta)$ a variância do erro de modelagem (erro de predição de um passo à frente ou resíduo) e $n_\theta = \dim[\hat{\theta}]$ o número de parâmetros no modelo.

Nesse critério, à medida que são incluídos termos no modelo, o número de graus de liberdade aumenta permitindo um melhor ajuste aos dados. Dessa forma, $\sigma^2_{erro}(n_\theta)$ diminui quando n_θ aumenta. A partir de um determinado momento, a diminuição da variância dos resíduos devido à inclusão de um novo termo no modelo é insignificante, não sendo justificada a inclusão deste termo. Assim, a primeira parcela da equação (4.3) quantifica a diminuição na variância dos resíduos resultante da inclusão de um termo, enquanto a segunda parcela penaliza a inclusão de cada termo. A ponderação originalmente estipulada para a inclusão de um termo é 2. Outros critérios de informação semelhantes ao AIC são citados na literatura modificando-se as ponderações entre os dois termos da equação (4.3) [3].

As vantagens e desvantagens de cada uma das abordagens supracitadas não estão bem definidas na literatura, ainda sendo motivos de debate [6]. Neste trabalho adota-se o critério de informação de Akaike como base para o cálculo do *fitness* de um indivíduo, como descrito a seguir. A escolha é justificada principalmente pela maior facilidade de cálculo, já que não necessita, por exemplo, estimar

funções densidade de probabilidade condicional como na equação (4.2).

4.5.2 Proposta de Função de *Fitness*

O *fitness* do i-ésimo indivíduo da população é calculado no presente trabalho como:

$$\text{fitness}(i) = \frac{1}{1 + \overline{\text{AIC}}(i)} \tag{4.4}$$

sendo $\overline{\text{AIC}}(i)$ a normalização do critério de Akaike no intervalo $[0, 1]$, calculado para o i-ésimo indivíduo segundo a equação (4.3). Tal normalização é necessária pois, como pode ser visto na equação (4.3), quando a variância do erro de modelagem $\sigma^2_{erro}(n_\theta)$ é muito baixa (o que é natural nas últimas gerações do algoritmo genético), o valor de $\text{AIC}(n_\theta)$ pode se tornar negativo. Tal normalização, no entanto, não interfere no processo evolutivo uma vez que, para o operador de seleção adotado (ver seção 4.6), o importante é a ordem relativa de grandeza entre os valores de *fitness* dos indivíduos da população.

O valor de n_θ presente na equação (4.3) é calculado computando-se a quantidade de parâmetros do modelo como um todo. Para o sistema *fuzzy* TS FBO com pólo único, são 2 parâmetros devido ao pólo complexo ou 1 se o pólo for real, 2 parâmetros para cada função de pertinência, mais o total de parâmetros nos conseqüentes das regras, calculados utilizando o estimador de mínimos quadrados. Para a arquitetura *fuzzy* TS FBO Generalizado, contabilizam-se 2 parâmetros para cada par de pólos complexos conjugados (no caso geral, duas vezes o número de modelos locais identificados), 2 parâmetros para cada função de pertinência nas premissas das regras, mais o número de parâmetros dos conseqüentes nos modelos locais (que neste caso podem ser distintos), estimados utilizando o algoritmo dos mínimos quadrados.

O número de parâmetros dos conseqüentes de cada regra depende do tipo de modelo local adotado. No caso de modelos lineares com n estados tem-se $n + 1$ parâmetros para cada regra. Para os modelos locais não lineares de Volterra de segunda ordem o número de parâmetros para n estados é $[(n + 1)(n + 2)]/2$ em cada regra.

4.6 Operadores de Seleção

Uma vez que os indivíduos da população foram avaliados, o próximo passo é a utilização de um critério de seleção. A maioria dos métodos de seleção é projetada para escolher preferencialmente indivíduos com maiores notas de aptidão (valores de *fitness*), embora não exclusivamente, para manter a diversidade da população. Basicamente, existem três tipos de seleção: determinística, estocástica e híbrida.

O critério de seleção determinística contempla somente os indivíduos que atendem a determinadas características previamente estabelecidas como desejáveis. Os indivíduos que não atenderem o critério de escolha serão eliminados sumariamente. Dessa forma, não há chance de um indivíduo que não satisfaça os requisitos vir a ser escolhido para fazer parte da próxima geração.

Na seleção estocástica os indivíduos que apresentarem maior adaptabilidade terão mais chance de serem escolhidos. Os de menor adaptabilidade terão menos chance, mas poderão eventualmente ser selecionados.

Na seleção híbrida o processo de seleção intercala os critérios de escolha determinísticos e estocásticos. Dessa forma, há a garantia de escolha de parte dos melhores indivíduos da população, porém mantendo a possibilidade de seleção também dentro de todo o espaço de soluções.

Um método de seleção muito utilizado é o Método da Roleta. É um processo eminentemente estocástico, portanto é probabilístico e variante com as gerações.

A Figura 4.6 exemplifica uma roleta para uma determinada geração de uma população com 6 indivíduos.

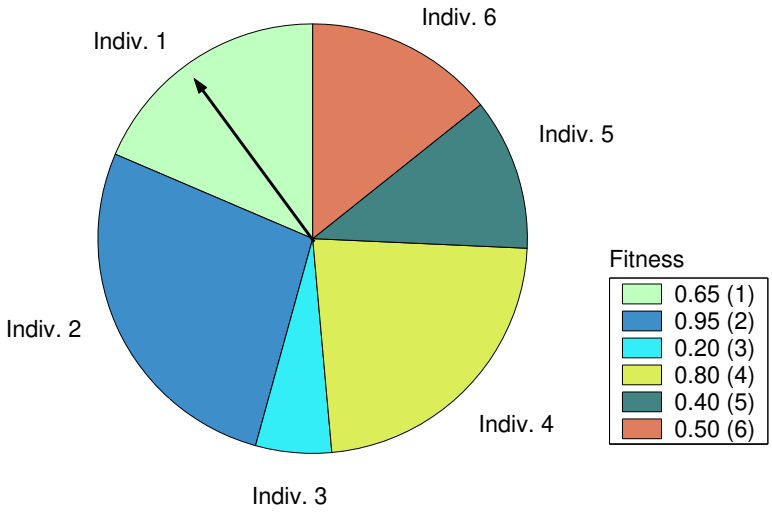

Figura 4.6: Ilustração do método da roleta.

Neste método, cada indivíduo da população é representado na roleta proporcionalmente ao seu índice de aptidão. Assim, os indivíduos com alta adaptabilidade têm uma porção maior da roleta, enquanto aos menos aptos é dada uma porção relativamente menor. Finalmente, a roleta é girada um determinado número de vezes, dependendo do tamanho da população, e aqueles sorteados na roleta são submetidos à atuação dos operadores de reprodução, se for o caso, ou copiados diretamente para a população da geração seguinte.

O método da roleta, assim como todos os métodos de seleção proporcionais ao *fitness*, possui

dois problemas inerentes. Se a população contém uma solução de *fitness* significativamente maior que as demais soluções na população, esta irá ocupar a maior parte da roleta. Assim, a maioria das rodadas da roleta sorteará essa mesma solução. Isto fará com que a população perca sua diversidade e provocará a convergência prematura do algoritmo para uma solução sub-ótima.

O segundo problema pode ocorrer após várias gerações, quando a maioria dos membros da população possui aproximadamente o mesmo *fitness*. Neste caso, as "fatias" da roleta possuem praticamente o mesmo tamanho e as soluções possuem aproximadamente as mesmas probabilidades de serem escolhidas. Obviamente, trata-se de uma busca quase aleatória.

O operador de seleção baseado em *rank* é semelhante ao método da roleta, porém as "fatias" são atribuídas aos indivíduos de acordo com sua posição após a ordenação (ascendente ou descendente) pelo valor do *fitness*. Assim, em uma população de tamanho μ, ao indivíduo com menor valor de *fitness* será atribuído o *rank* zero e ao com o maior valor, o *rank* $\mu - 1$. A seleção por *rank* não apresenta nenhuma das duas desvantagens citadas para o método da roleta.

Após a ordenação, podem ser utilizados mapeamentos lineares ou não lineares para a determinação da probabilidade de seleção de cada indivíduo. Supondo que o *rank* do i-ésimo indivíduo é dado por $rank(i)$, então, para um mapeamento linear, tem-se que sua probabilidade de seleção é dada por [6]:

$$p(i) = \frac{\alpha + \frac{rank(i)}{(\mu-1)}(\beta - \alpha)}{\mu} \tag{4.5}$$

sendo α e β são as quantidades esperadas de filhos dos indivíduos com menor e maior *fitness*, respectivamente.

Um exemplo de mapeamento não linear é:

$$p(i) = \eta(1-\eta)^{\mu-1-rank(i)} \tag{4.6}$$

com $\eta \in (0, 1)$. Esta distribuição está relacionada a uma distribuição de Bernoulli [6].

Outro mecanismo de seleção é o operador de Boltzmann, o qual altera o valor do *fitness* de cada indivíduo segundo a função densidade de probabilidade de Boltzmann, equação (4.7), fornecendo sua probabilidade de seleção Ψ_i:

$$\Psi_i = \frac{1}{1 + \exp(F_i/T)} \tag{4.7}$$

sendo T um parâmetro análogo ao termo de temperatura na distribuição de Boltzmann e F_i o *fitness* do i-ésimo indivíduo. Este termo T é reduzido de uma maneira pré-determinada em iterações sucessivas. Como um valor inicial grande de T é utilizado, as soluções iniciais não possuem grandes

discrepâncias em seu grau de probabilidade de seleção. No entanto, com o progresso do algoritmo genético, o parâmetro T diminui e as melhores soluções são mais evidenciadas.

Um quarto operador de seleção é o torneio. Em um torneio de k indivíduos, um determinado indivíduo é selecionado para pertencer à nova geração determinística ou probabilisticamente, de acordo com seu valor de *fitness*. Após o torneio, há duas opções: ou todos os participantes são colocados de volta na população (podendo vir a participar em outros torneios) ou não o são até que um determinado número de torneios ocorra. Em sua forma mais simples, duas soluções são sorteadas e a melhor delas é escolhida. A pressão seletiva do método é controlada pelo número de elementos que participam em cada torneio. Quanto maior esse número, maior será a pressão seletiva. Quanto maior a pressão seletiva, maior será a taxa de convergência do algoritmo [6].

O torneio também elimina os dois problemas do método da roleta. Além disso, pode ser aplicado tanto a problemas de maximização quanto de minimização, enquanto o método da roleta supõe o objetivo de maximização, sendo necessária a adaptação da função de *fitness* para problemas de minimização. Outro cuidado extra quando do uso do mecanismo da roleta é a impossibilidade de uso de valores de *fitness* negativos, o que não consiste em um problema para o método de torneio.

A implementação do método do torneio é simples e eficiente, não requerendo o ordenamento da população (como no método de *rank*). Como não são necessárias computações globais (como no método da roleta), o método do torneio mostra-se mais adequado para uma implementação paralela.

Tendo em vista as vantagens do método do torneio (simplicidade e eficiência) este foi adotado no presente trabalho.

4.6.1 Proposta de Operador de Seleção

Neste trabalho foi adotado como operador de seleção o torneio, conforme justificativas na seção anterior. Na configuração estabelecida, um total de cinco por cento da população é escolhido aleatoriamente para participar do torneio. Deterministicamente é selecionado o indivíduo com o maior valor de *fitness*. O método é implementado com reposição, ou seja, todos os indivíduos participantes do torneio são recolocados na população, inclusive o selecionado. Foi implementado também elitismo, o qual consiste em selecionar o indivíduo com o maior *fitness* de uma geração e incluí-lo na geração seguinte.

Como exposto por Bäck *et al.* [6], a escolha do operador de seleção deve ser feita em conjunto com as definições dos outros parâmetros do algoritmo genético, principalmente os operadores de reprodução. Dessa forma, ao se aliar um operador de seleção de pressão seletiva moderada, como é o caso, a operadores de recombinação e mutação com alto poder de perturbação (ver seção 4.7.1), uma implementação eficiente é obtida.

4.7 Reprodução

A reprodução é a fase do algoritmo genético em que os indivíduos filhos são gerados com o objetivo de compor a nova população. Consiste na ação de operadores genéticos, os quais são aplicados sobre os indivíduos pais, gerando os filhos. A literatura propõe uma quantidade muito grande de operadores genéticos. A definição dos operadores de reprodução está intrinsecamente ligada ao tipo de representação adotada, a qual, por sua vez, depende da classe de problemas que se estuda. Como exemplo, ao se trabalhar com o problema do caixeiro viajante, os operadores genéticos devem garantir a validade do indivíduo após sua atuação, considerando que um determinado alelo deve ocorrer obrigatoriamente uma única vez em um cromossomo [63].

São dois os operadores genéticos básicos utilizados durante a fase de reprodução nos algoritmos genéticos: Recombinação e Mutação.

O operador de recombinação (ou *crossover*) envolve a participação de dois indivíduos pais. A técnica da recombinação consiste na troca de material genético entre os pais, gerando dois candidatos a filhos. É escolhido aleatoriamente um conjunto de pontos que delimita a região de troca dos genes. As variações desse método referem-se à determinação do número de pontos e sua distribuição dentro do cromossomo, assim como no modo como os genes são trocados. Independentemente de como se processa a recombinação, ela sempre gerará dois filhos. Em algumas abordagens, destes dois filhos apenas um sobrevive. A escolha do filho sobrevivente, o qual se tornará um indivíduo na próxima geração, pode ser feita de diversas formas. Em uma delas, simplesmente é realizado um sorteio entre os dois candidatos a filho. Outra opção é a seleção elitista - a escolha recai sobre o candidato de maior adaptabilidade.

O operador de *crossover* geométrico [80] é um exemplo utilizado para codificação real, isto é, uma na qual os genes assumem valores reais. Dados dois pais, ele gera um único filho conforme ilustrado na Figura 4.7. Este operador atribui a cada gene do filho a raiz quadrada do produto entre os respectivos genes dos pais. Esta operação tende a preservar as características de pais semelhantes e fazer uma combinação daqueles mais diferentes.

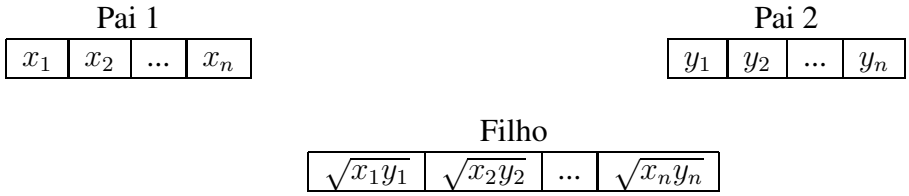

Figura 4.7: Operador de *crossover* geométrico.

Após a recombinação podem ocorrer mudanças aleatórias nos genes do indivíduo. No contexto biológico, estas alterações, genericamente denominadas de mutação, são causadas devido a erros

durante a replicação do material genético dos cromossomos envolvidos em um *crossover*. Uma mutação pode ser caracterizada pela eliminação, duplicação, inversão ou deslocamento de um conjunto de genes [6].

Nos algoritmos genéticos, o operador genético de mutação envolve a participação de apenas um indivíduo. Esta técnica consiste, em analogia à Biologia Genética, na alteração de um ou mais genes de um indivíduo após o *crossover*. O novo indivíduo gerado para a próxima geração pode assim possuir alguma característica adicional que lhe forneça melhor adaptação que seu gerador. Os pontos que identificarão as posições dos genes que sofrerão as transformações são escolhidos aleatoriamente. A motivação para a existência do operador de mutação, além da coerência biológica, é a introdução de diversidade na população. Desta forma, a mutação assegura que a probabilidade de se chegar a qualquer ponto do espaço de busca nunca será nula. Além disso, contorna o problema dos ótimos locais através da leve alteração da direção da busca.

Um exemplo de operador de mutação é a mutação por inversão. Após a escolha aleatória de dois pontos no cromossomo, o operador inverte os genes entre estes pontos. A Figura 4.8 ilustra o comportamento desse operador ao se sortear o primeiro e o último gene do cromossomo para a delimitação da inversão.

Cromossomo original

0.55	0.23	1.70	3.01	0.84	-1.22	6

Cromossomo após mutação

0.55	**-1.22**	**0.84**	**3.01**	**1.70**	**0.23**	6

Figura 4.8: Operador de mutação por inversão.

Cada operador genético tem associado a si uma taxa de ocorrência, tal qual nas ciências biológicas. Os operadores de recombinação possuem altas taxas de ocorrência. Os de mutação, baixa, sob a pena da degeneração do algoritmo em uma busca aleatória.

Uma revisão bibliográfica ampla sobre os diversos operadores genéticos utilizados em algoritmos evolutivos está disponível em [6, 7, 111].

4.7.1 Proposta de Operadores de Reprodução

Crossover

São propostos dois operadores de *crossover* para a arquitetura elaborada. Ambos satisfazem as condições estabelecidas na seção 4.4.1. Consistem em adaptações dos operadores de *crossover* uniforme e *crossover* aritmético, descritos na seqüência.

Dado que a representação proposta é real (em contrapartida à representação binária), o uso do *crossover* aritmético é imprescindível. Os operadores de *crossover* tradicionais de um ponto ou de n-pontos, mesmo em codificações binárias, apresentam o problema de não serem capazes de realizar um exploração eficaz do espaço de busca. Para codificações reais, o único meio de se alcançar essa exploração é permitindo que os indivíduos filhos possuam genes que não estão presentes em nenhum de seus pais [6].

O operador de *crossover* aritmético descrito em [79] atua sobre os genes x_i e y_i de dois indivíduos pais x e y criando o indivíduo filho z segundo a média ponderada:

$$z_i = \beta x_i + (1 - \beta) y_i$$

sendo β um número aleatório no intervalo $[0, 1]$ e $i = 1, ..., L$ os *loci* do cromossomo de comprimento L.

A modificação implementada aqui consiste em ampliar o intervalo de β para $[-\delta; 1 + \delta]$. O parâmetro de expansão percentual δ permite a extrapolação dos limites estabelecidos por x_i e y_i. O efeito imediato dessa alteração é permitir ao operador uma exploração mais efetiva do espaço de busca, dado que em sua versão original o valor de determinado gene de um indivíduo filho é limitado pelos valores dos respectivos genes dos indivíduos pais. Além disso, opera-se com um valor distinto de β para cada elemento, ou seja, β_i. Esta medida, aliada a uma alta taxa de ocorrência de *crossover*, contribui para a manutenção da diversidade da população durante sua evolução. Como a seleção é realizada com reposição (um mesmo indivíduo pode ser selecionado mais de uma vez para reprodução), para a geração de dois indivíduos iguais uma série pouco provável de eventos deveria ocorrer: a seleção do mesmo par de pais e para cada porção considerada do cromossomo os mesmo números β_i deveriam ser sorteados.

No capítulo 5 apresentar-se-á uma comparação dos resultados obtidos com essa modificação no *crossover* aritmético e sem a mesma. Em suma, essa modificação permitiu ao algoritmo genético manter de fato a diversidade da população por um número maior de gerações, fornecendo resultados médios melhores.

O valor do parâmetro de expansão percentual não deve ser muito grande, sob pena de degeneração do algoritmo genético em uma busca aleatória. Este valor deve ser tal que permita a exploração mais efetiva do espaço de busca em torno dos dois genes sendo combinados (e não apenas entre eles), porém deve também respeitar a propriedade inerente aos operadores de *crossover* de manter nos indivíduos filhos características presentes em seus pais. Para ilustrar a sensibilidade do algoritmo genético a mudanças nesse parâmetro, considere o seguinte cenário hipotétipo. Os genes sendo combinados pelo *crossover* aritmético residem no intervalo $[0, 1]$, obtidos por uma distribuição uni-

forme. Um cromossomo possui um número de genes n elevado, digamos 10.000[5]. Seja a seguinte norma para medida de diferença entre um filho f e seus pais p_1 e p_2:

$$d = \frac{1}{n} \sum_{i=1}^{2} \sum_{j=1}^{n} |f_j - p_{ij}| \tag{4.8}$$

sendo j o índice do gene no cromossomo. A Figura 4.9 exibe o cálculo da norma (4.8) para o *crossover* entre dois pais sorteados aleatoriamente gerando um filho e para diferentes valores do parâmetro δ. Para $\delta = 0$ tem-se o *crossover* aritmético original descrito em [79]. Observa-se que

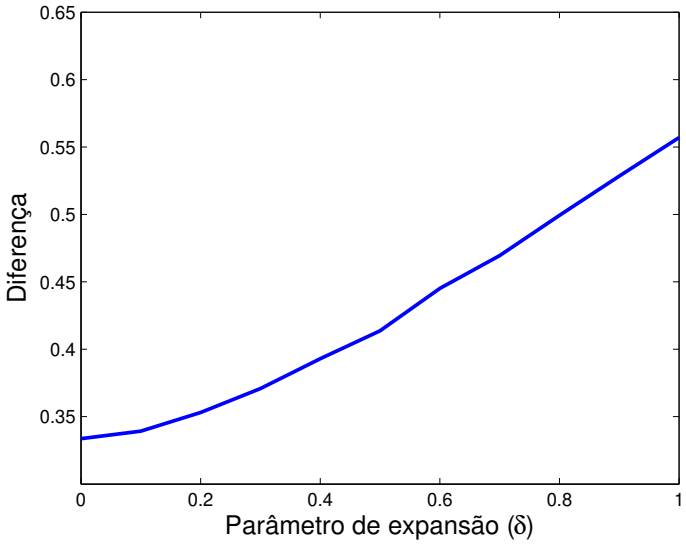

Figura 4.9: Sensibilidade do *crossover* aritmético ao parâmetro de expansão δ.

para valores de δ até aproximadamente $0,2$ (correspondendo a uma expansão de 20%) a medida de diferença é aproximadamente igual ao caso original, ou seja, o filho gerado herda características de ambos os pais. No entanto, conforme pode ser observado na Figura 4.10, alguns filhos (representados pelos asteriscos) estão fora do intervalo entre os dois pais, porém próximos a um deles. Esta é a justificativa para a alteração introduzida no *crossover* aritmético tradicional. A Figura 4.10 foi obtida para $\delta = 0,2$. O capítulo 5 apresenta uma série de comparações de simulações com e sem o parâmetro de expansão δ, evidenciando a eficiência dessa modificação.

Para o modelo *fuzzy* TS FBO Generalizado, o *crossover* aritmético é aplicado entre todos os genes, observando-se apenas que as últimas posições do cromossomo são número inteiros, sendo necessário então o arredondamento dos valores finais obtidos para o inteiro mais próximo.

[5]Este valor elevado é utilizado para explicitar a sensibilidade aos valores do parâmetro δ.

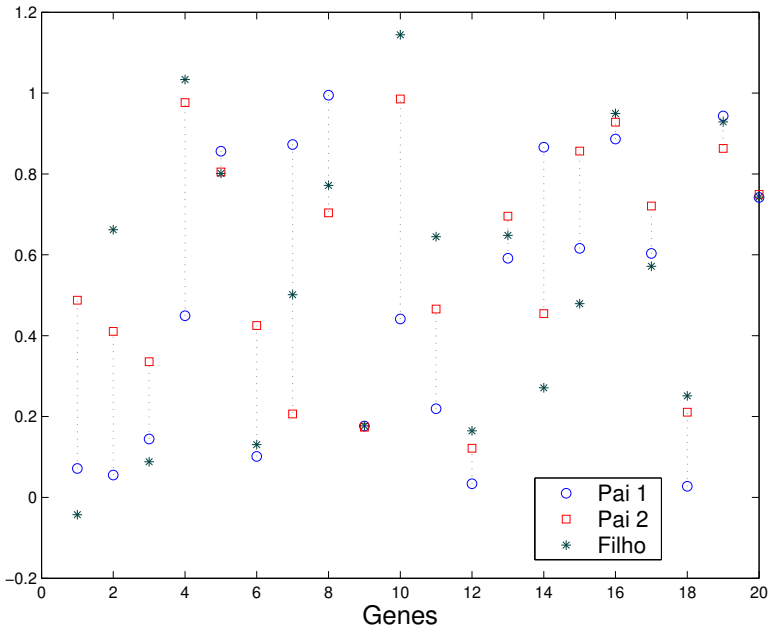

Figura 4.10: Resultado do *crossover* aritmético modificado em um conjunto de genes.

Para o modelo *fuzzy* TS FBO com pólo único, cuja representação é mostrada na Figura 4.3, o *crossover* aritmético opera da seguinte maneira.

1. Os pólos são combinados da mesma forma que para o modelo anterior;
2. Para cada variável de entrada faz-se:

   Para cada termo lingüístico (função de pertinência) faz-se:

   (a) Se os dois indivíduos possuem o termo lingüístico, aplica-se o *crossover* às suas funções de pertinência, combinando os respectivos parâmetros (aberturas e centros);

   (b) Se apenas um indivíduo possui o termo lingüístico, copia-se tal termo para o respectivo indivíduo filho;

3. Aplica-se o *crossover* aritmético ao número de estados nos conseqüentes das regras, realizando-se o devido arredondamento.

Duas verificações de validade simples são necessárias em ambas arquiteturas após a aplicação desse operador: estabilidade do pólo e abertura mínima das funções de pertinência. Caso o pólo resultante seja instável, é gerado aleatoriamente um novo pólo estável. No caso de a abertura da função de pertinência resultante ser menor que o limite especificado, então atribui-se à abertura este

valor mínimo. Além dessas duas verificações, existe outra opcional, relativa ao uso da segunda representação para as funções de pertinência explicada na seção 4.4.1, isto é, posição relativa (e não absoluta) dos centros. Como os parâmetros ξ representam a distância entre dois centros consecutivos das funções gaussianas, seu valor deve ser maior que o mínimo estabelecido pelo projetista, garantindo um grau máximo de sobreposição das funções de pertinência. Se o valor resultante for menor que o permitido, atribui-se este mínimo ao respectivo gene do indivíduo.

No modelo *fuzzy* TS FBO Generalizado, outra verificação trivial necessária consiste em garantir que os índices dos estados nas premissas sejam válidos, ou seja, valores menores ou iguais ao menor dos parâmetros indicando a quantidade de estados dos modelos locais. Caso esta condição não seja satisfeita, sorteiam-se novos valores válidos para estes índices.

Uma análise semelhante também é prevista para o modelo *fuzzy* TS FBO com pólo único. O número de estados nos conseqüentes deve ser igual ou maior ao número de estados (variáveis lingüísticas) dos antecedentes. Esta verificação é necessária pois o número de estados nos modelos FBO locais (ordem dos modelos) é dado pelo número de estados nos conseqüentes das regras. Caso este número fosse inferior ao número de estados dos antecedentes, uma regra iria referenciar em sua premissa uma variável inexistente. Assim como no modelo Generalizado, a correção do indivíduo consiste no sorteio de um valor válido para o parâmetro em questão. É válido esclarecer que, mesmo que o método dos mínimos quadrados atribua a um estado dos conseqüentes um coeficiente nulo, este estado em si possui uma dinâmica própria, podendo assim estar presente nas premissas das regras. Da mesma forma, pode acontecer de um estado não estar presente nas premissas das regras, e sim apenas nos conseqüentes. Neste caso, embora tal estado não atue na etapa de seleção das regras que serão ativadas, ele contribue com sua dinâmica para a representação do sistema como um todo.

O operador de *crossover* uniforme é agora descrito. Ele não utiliza pontos de cruzamento propriamente ditos. Em seu lugar, determina, por meio de uma máscara binária, que genes de cada pai serão herdados pelos filhos. A Figura 4.11 ilustra a atuação desse operador. Caso o i-ésimo elemento da máscara seja 1, então o valor do i-ésimo gene do filho 1 será igual ao gene correspondente do pai 1. Se tal elemento da máscara for 0, o valor do i-ésimo gene do filho 1 será igual ao gene correspondente do pai 2. A atribuição dos genes do filho 2 segue o processo inverso.

O operador de *crossover* uniforme é aplicado no modelo *fuzzy* TS FBO Generalizado diretamente como descrito anteriormente, verificando-se apenas os requisitos relativos à estabilidade do pólo e índice dos estados nas premissas das regras.

Para sua aplicação no modelo *fuzzy* TS FBO com pólo único, é preciso se estipular o que será considerado como unidade básica para intercâmbio entre os cromossomos, dado o sorteio da máscara binária. Tais unidades serão compostas por um conjunto de genes com uma semântica bem definida. São consideradas como unidades para o *crossover* uniforme:

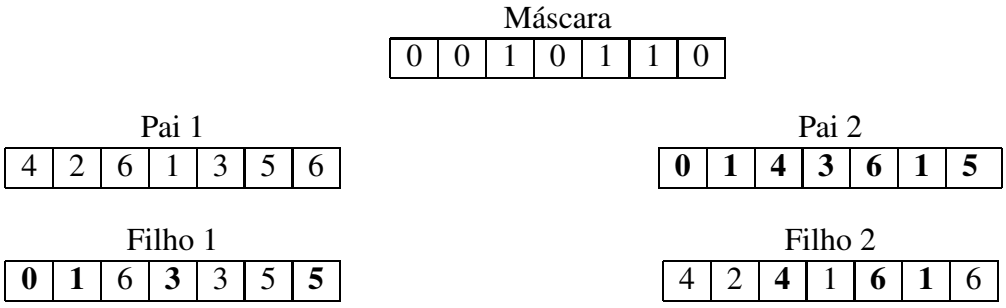

Figura 4.11: Operador de recombinação uniforme.

- Os dois genes que compõem o pólo complexo;
- Os dois genes que definem uma função de pertinência gaussiana;
- O gene determinando a quantidade de estados nos conseqüentes das regras.

Dessa forma, garante-se um tratamento a essa representação matricial equivalente ao dado aos cromossomos tradicionais na forma de um vetor. Para cada variável de entrada dos dois indivíduos pais, as unidades correspondentes às suas funções de pertinência são trocadas de acordo com a máscara binária sorteada, a qual também pode ser tida agora como matricial.

A verificação de validade necessária para os indivíduos filhos se restringe ao número de estados nos conseqüentes das regras. A análise das distâncias entre os centros das funções de pertinência também deve ser realizada, em ambas arquiteturas, caso a abordagem para controle de sobreposição seja utilizada.

Ainda considerando o modelo *fuzzy* TS FBO com pólo único, o operador de *crossover* uniforme é responsável pelo possível aumento do número de funções de pertinência bem como do número de variáveis de entrada para determinado indivíduo. Esta é uma função importante deste operador, já que os operadores de *crossover* aritmético e de mutação (descritos na próxima seção) não são capazes de incluir novos termos lingüísticos em um indivíduo e as medidas de similaridade (descritas na seção 4.9), pelo contrário, eliminam funções de pertinência redundantes. Para aumentar o número de funções de pertinência, basta que a máscara binária sorteada atribua a um dos indivíduos filhos uma função de pertinência do outro pai, função esta originalmente inexistente. Para a inclusão de uma nova variável de entrada, a máscara deve ser tal que atribua ao menos duas funções de pertinência à uma variável que inicialmente não possuia nenhuma.

Cada vez que uma operação de *crossover* deve ser efetuada, realiza-se um sorteio entre eles, sendo que o *crossover* aritmético tem probabilidade de ocorrência igual a 70%, e o uniforme, 30%. Estes valores são justificados pela maior capacidade do operador de *crossover* aritmético de manter a diversidade da população (uma vez que tende a gerar indivíduos filhos distintos dos pais).

Mutação

O operador de mutação tradicional em representações com números reais atua sobre um gene y_i gerando um elemento y_i' segundo a equação (4.9):

$$y_i' = y_i + M \tag{4.9}$$

sendo M uma realização de uma variável aleatória. As diferentes versões de mutação são dadas pelas diferentes distribuições da variável aleatória M. O operador de mutação implementado nesse trabalho é baseado em uma distribuição Gaussiana.

A função densidade de probabilidade para uma variável aleatória com distribuição Gaussiana é dada pela equação (4.10):

$$g(x) = \frac{1}{\sigma\sqrt{2\pi}}\exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right) \tag{4.10}$$

sendo μ o valor médio da distribuição e σ^2 sua variância. É convenção fazer $\mu = 0$ durante a aplicação do operador de mutação Gaussiana, o que também é implementado no presente trabalho. Assim, o controle sobre a atuação do operador de mutação, em termos de quão diferentes de seus pais os indivíduos filhos podem ser gerados, concentra-se em um único parâmetro, a variância σ^2, ou, equivalentemente, no desvio padrão σ. São descritas na literatura muitas abordagens nas quais este parâmetro é variável, sendo função da taxa de convergência do algoritmo genético (diversidade da população), da efetividade de mutações anteriores ou do número de gerações já processadas. Além disso, outras funções de densidade de probabilidade já foram propostas, tais como a distribuição de Cauchy e a de Laplace [6]. Neste trabalho, a variância é, além de variável, também diferenciada para componentes distintos do cromossomo sendo mutado.

Embora a distribuição inicial (e possivelmente em todas gerações) da população não seja Gaussiana (ver seção 4.4.1), o desvio padrão dos genes de um indivíduo é uma medida da diversidade da população e é por isso adequado como parâmetro para ajustar o nível de perturbação do operador de mutação. Uma série de estudos corroboram a idéia de que o nível de perturbação do operador de mutação deve diminuir com o tempo, seguindo de alguma forma a tendência da própria diversidade da população, tornando a busca mais acurada à medida que o algoritmo genético tende a convergir [6, 34, 35, 44].

Após essa pequena introdução é possível detalhar como o operador de mutação pode ser implementado nas arquiteturas propostas. Como as representações agrupam num mesmo cromossomo elementos de domínios diferentes, um tratamento diferenciado é dado a cada um deles.

Primeiramente, é calculado o desvio padrão σ_i de cada gene i na população. Para a arquitetura *fuzzy* TS FBO com pólo único, é preciso desconsiderar os eventuais genes nulos para determinados

parâmetros. Por exemplo, algumas funções de pertinência podem não estar presentes em determinado indivíduo. O valor zero no cromosso, que indica a ausência de uma função de pertinência, não é considerado no cálculo do desvio padrão para aquela função de pertinência em particular. Para o modelo *fuzzy* TS FBO Generalizado, o cálculo do desvio padrão dos genes resume-se ao desvio padrão de cada coluna da matriz numérica que codifica a população. Em seguida, para os parâmetros com codificação real, a mutação consiste na adição da realização de uma variável aleatória Gaussiana com média zero e desvio padrão σ_i. Para os parâmetros com codificação inteira, apenas efetua-se o arredondamento da perturbação para o inteiro mais próximo antes de esta ser adicionada ao gene em questão.

As mesmas verificações de validade descritas para o *crossover* aritmético são realizadas após uma mutação, para ambas as arquiteturas.

4.8 Condições de Parada

Para encerrar a descrição dos componentes do algoritmo genético resta explicitar a condição de parada adotada. O algoritmo procede a evolução até que uma das duas condições seguintes seja atendida:

- Um número máximo de gerações definido pelo usuário é atingido;
- A diversidade da população, calculada em termos do desvio padrão dos valores codificados nos cromossomos (soma dos desvios padrão de cada gene), torna-se menor que um limite inferior pré-estabelecido.

Uma alternativa simples para a definição do limite inferior descrito na segunda condição anterior seria simplesmente arbritar um valor "pequeno" comparado à diversidade inicial da população. No entanto, este valor poderia ter impactos diferentes para as duas arquiteturas propostas aqui. Em outras palavras, um mesmo valor poderia significar uma diversidade relativa muito menor em uma representação cromossômica do que em outra.

Dessa forma, o limite inferior de diversidade deve ter uma magnitude tal que considere a arquitetura específica utilizada. Uma forma eficiente de alcançar esse objetivo é estipular uma percentagem da diversidade inicial da população como o limite inferior para se considerar a convergência da polulação. Assim, considera-se ainda a eventual diferença nas diferentes inicializações das populações em execuções distintas do algoritmo genético para uma mesma arquitetura.

Finalizando esse capítulo, a seção seguinte descreve um método para a simplificação dos modelos *fuzzy* TS FBO baseado em medidas de similaridade.

4.9 Medidas de Similaridade

Métodos automatizados para a geração de funções de pertinência em um sistema *fuzzy*, tais como redes neurais, algoritmos genéticos e métodos de agrupamento, podem produzir funções de pertinência redundantes, ou seja, funções de pertinência que englobem porções semelhantes no universo de discurso, como acontece nos trabalhos [30, 65, 71, 91, 98]. Um número de funções de pertinência maior que o necessário implica em aumento da complexidade do sistema *fuzzy* sem a correspondente melhoria de desempenho. Um dos modos de se detectar estas redundâncias é o uso de medidas de similaridade.

Uma medida de similaridade é uma função que determina a similaridade entre dois conjuntos *fuzzy*, de modo a indicar o quão similar ou iguais eles são. Nesse contexto, o termo similar ou igual não deve ser entendido como igualdade de forma, e sim como o grau de sobreposição entre as funções de pertinência de dois conjuntos *fuzzy*.

Para ser usada em um mecanismo automático de simplificação de funções de pertinência uma medida de similaridade deve possuir as seguintes características [99]:

1. Dois conjuntos *fuzzy* que não apresentem qualquer sobreposição possuem grau de similaridade nulo;

2. Dois conjuntos *fuzzy* que se sobrepõem possuem um grau de similaridade maior que zero;

3. Somente conjuntos *fuzzy* iguais (em termos de sobreposição) devem ter o grau de similaridade máximo, no caso 1;

4. O grau de similaridade não deve ser influenciado por fatores de escala ou de deslocamento no domínio em que os conjuntos estão definidos.

As medidas de similaridade podem ser divididas em:

- Medidas de Similaridade Geométricas: São mais adequadas na determinação da medida de similaridade entre conjuntos *fuzzy* distintos;

- Medidas de Similaridade baseadas na Teoria de Conjuntos: São mais adequadas para capturar a similaridade entre conjuntos *fuzzy* que se sobrepõem. Baseiam-se em operações sobre conjuntos, como a união e a interseção.

Um exemplo de uma medida de similaridade baseada nas operações de união e interseção é dado pela equação (4.11):

$$S(A,B) = \frac{\mid A \cap B \mid}{\mid A \cup B \mid} \tag{4.11}$$

onde S indica a similaridade entre os dois conjuntos *fuzzy* A e B, $|\ .\ |$ indica a cardinalidade de um conjunto, e $\cap$ e $\cup$ representam a interseção e a união, respectivamente. A equação acima em termos de função de pertinência pode ser escrita na seguinte forma:

$$S(A,B) = \frac{\sum_{j=1}^{m}[\mu_A(x_j) \wedge \mu_B(x_j)]}{\sum_{j=1}^{m}[\mu_A(x_j) \vee \mu_B(x_j)]} \tag{4.12}$$

definida em um universo discreto $X = \{x_j \mid j = 1, 2, ..., m\}$. $\wedge$ e $\vee$ são os operadores mínimo e máximo, respectivamente, e $\mu_Z(x_i)$ é o grau de pertinência do valor x_i ao conjunto Z. Quando da implementação computacional, domínios contínuos devem ser discretizados.

Uma vez medida a similaridade entre os conjuntos *fuzzy* A e B e detectado que seu valor está acima de um limite pré-estabelecido, pode-se substituir A e B por A, por B ou criar um novo conjunto *fuzzy* C a partir de A e B. Desse modo, efetua-se a simplificação da base de regras do sistema *fuzzy* através da eliminação da redundância nas funções de pertinência de suas variáveis lingüísticas.

4.9.1 Proposta de Método para Simplificação de Sistemas *Fuzzy*

Propõe-se a aplicação das medidas de similaridade durante a execução do algoritmo genético como método para a simplificação dos sistemas *fuzzy* TS FBO com pólo único. Ao final de cada geração, após a aplicação dos operadores genéticos de reprodução, é calculada a similaridade entre cada par de funções de pertinência de uma mesma variável lingüística. Então, caso o valor da medida de similaridade seja superior a um limite, as operações de simplificação devem ser efetuadas.

Duas atitudes são tomadas para a simplificação de um sistema *fuzzy*. Primeiramente, caso haja mais de duas funções de pertinência para a variável lingüística em questão, é realizada a fusão das duas funções de pertinência com maior grau de similaridade acima do limite estabelecido. As duas funções de pertinência são substituídas por uma com a seguinte configuração: o centro da nova função de pertinência é a média aritmética dos centros das duas originais e a abertura é também a média das duas aberturas originais. Essa estratégia faz com que a nova função de pertinência preserve ao máximo a forma e disposição das funções originais.

Caso existam apenas duas funções de pertinência para a variável lingüística sendo analisada, com grau de similaridade acima do limite, a própria variável lingüística é eliminada do sistema. Caso fosse efetuada a fusão, restaria apenas uma função de pertinência. Assim como afirmado na seção 4.4.1, esta variável é desnecessária ao sistema, podendo ser removida sem prejuízo à performance do modelo.

Um aspecto crucial é a definição do limite de similaridade para se incorrer em simplificação. Um valor elevado implica em poucas simplificações, podendo ainda existir ao final do processo evolutivo funções redundantes. Um valor demasiadamente baixo faz com que muitas fusões/eliminações sejam

efetuadas, diminuindo a diversidade da população e acelerando a convergência do algoritmo provavelmente em direção a um ótimo local. Com essas preocupações em mente, elaborou-se uma equação para se tornar dinâmico o valor desse limite. Nas gerações iniciais o valor é próximo ao máximo, 1. À medida que a diversidade da população cai, o limite de similaridade também cai, porém bem mais lentamente. Este comportamento é requerido para evitar que a realização de muitas fusões e/ou eliminações nas gerações iniciais do algoritmo genético incorra em decréscimo ainda mais acentuado da diversidade da população.

A equação (4.13) mostra como o limite L é calculado em função da diversidade da população na i-ésima geração, σ_i, da diversidade inicial da população, σ_0 e do limite mínimo de similaridade ϵ:

$$L = \epsilon + (1 - \epsilon)(1 - (1 - \min(1, \sigma_i/\sigma_0))^{\rho}) \tag{4.13}$$

sendo ρ uma potência par e $\min(.)$ o operador de mínimo (retorna o menor de seus argumentos). Este operador de mínimo é usado para limitar superiormente em 1 o valor resultante desta equação na eventualidade da diversidade da população em uma geração intermediária (σ_i) se tornar maior do que a diversidade inicial (σ_0).

A Figura 4.12 ilustra a atuação dessa função. Os dados da diversidade da população foram simulados como uma função exponencial decrescente. A curva em azul no gráfico já é o resultado da normalização σ_i/σ_0. As curvas em vermelho são o resultado do uso de diferentes potências ρ, de 2 até 10. A seta indica o sentido crescente de ρ, ou seja, se inicia em $\rho = 2$ e aponta para $\rho = 10$. O limite mínimo de similaridade nestas curvas é $\epsilon = 0,2$. Valores desse limite acima de aproximadamente $0,8$ implicam em poucas fusões de funções de pertinência (a menos que estas sejam extremamente semelhantes). Valores abaixo de $0,3$ se mostram mais adequados nas últimas gerações do algoritmo genético, concluindo a simplificação do conjunto de funções de pertinência. Sendo assim, conclui-se que os valores mais adequados para a potência ρ na equação (4.13) são $\rho = 8$ ou $\rho = 10$. No entanto, estes valores não são críticos, no sentido que sua escolha afeta pouco o desempenho do algoritmo genético, já que as várias curvas definindo o limite L, para valores de ρ elevados, são semelhantes entre si e apresentam o comportamento desejado.

4.10 Resumo

Este capítulo detalhou como um algoritmo genético é aplicado na tarefa de otimização de modelos *fuzzy* TS FBO com pólo único e *fuzzy* TS FBO Generalizado. Após uma breve introdução sobre o tema, levantando os princípios biológicos motivadores dos algoritmos genéticos, sua estrutura básica e áreas de aplicação, além de uma revisão sobre os chamados Sistemas Genético-*Fuzzy*, descrevem-se

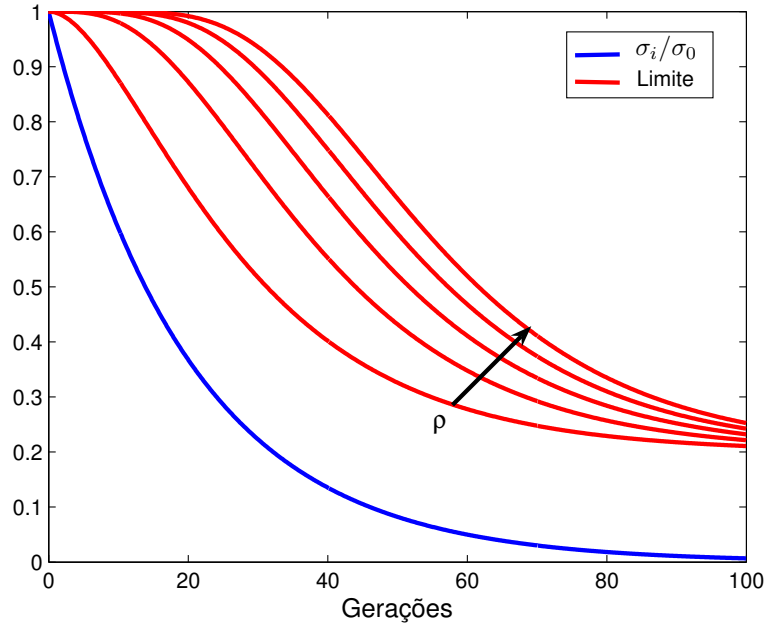

Figura 4.12: Função para a definição dinâmica do limite de similaridade.

os vários componentes e processos que integram um AG.

As principais contribuições deste capítulo são a proposta de uma representação cromossômica para modelos *fuzzy* TS FBO (seção 4.4.1), a extensão do operador de *crossover* aritmético através da introdução de um coeficiente de expansão (seção 4.7.1) e a proposta de uma metodologia para se aplicar medidas de similaridade com um limite dinâmico na tarefa de simplificação das funções de pertinência da arquitetura *fuzzy* TS FBO com pólo único (seção 4.9.1).

Uma perspectiva para continuidade do presente trabalho é a automação da determinação do número de modelos locais na arquitetura *fuzzy* TS FBO Generalizado. Um ponto de partida seria o trabalho [81], no qual são descritas várias abordagens de modelagem que tratam de múltiplos modelos locais, nas quais são propostos métodos como clusterização, estados funcionais e cadeias de Markov.

Para finalizar este capítulo apresenta-se a seguir um pseudocódigo para o algoritmo genético descrito, com as referências às seções nas quais cada etapa é abordada. Os dois operadores de *crossover* descritos são utilizados nos experimentos apresentados no próximo capítulo, seguindo a proporção apresentada na seção 4.7.1.

Algoritmo 4.1 Pseudocódigo para o Algoritmo Genético

```
g = 0;
cria população P(g);                                              (seção 4.4.1)
calcula diversidade da população inicial;
faça
   avalia fitness de P(g);                                        (seção 4.5.2)
   seleciona indivíduos para reprodução;                          (seção 4.6.1)
   operadores de crossover, gerando P'(g);                        (seção 4.7.1)
   operador de mutação Gaussiana, gerando P(g+1);                 (seção 4.7.1)
   medida de similaridade em P(g+1);                              (seção 4.9.1)
   calcula diversidade da população P(g+1);
   g = g + 1;
enquanto (g < Máximo de Gerações) E (diversidade da população P(g) > Diversidade Mínima)
mostrar resultados;
```

Capítulo 5

Resultados de Modelagem

Este capítulo apresenta os resultados obtidos ao se aplicar o método descrito no capítulo 4 para a modelagem de sistemas dinâmicos não lineares. Foram abordados dois sistemas: um levitador magnético e um processo bioquímico de polimerização. A análise qualitativa dos modelos encontrados baseia-se principalmente em seu desempenho (considerando o erro quadrático médio calculado em série sintética) e em sua complexidade.

A próxima seção descreve em mais detalhes os sistemas físicos abordados. As seções seguintes apresentam os resultados de modelagem.

5.1 Descrição dos Sistemas

5.1.1 Levitador Magnético

O sistema, ilustrado na Figura 5.1, é composto por duas bobinas, uma inferior e outra superior, que geram campos magnéticos pela passagem de correntes elétricas. Essas bobinas interagem através do campo com discos magnéticos que se deslocam em uma barra de vidro que serve como guia. Variando-se a magnitude da corrente na bobina inferior, pode-se controlar a posição do magneto inferior, fazendo-o levitar através de uma força magnética repulsiva. Similarmente, o magneto superior é posicionado através de uma força magnética de atração, gerada através de um valor adequado de corrente na bobina superior. Com a proximidade dos discos surge também interação magnética (força de repulsão) entre os dois magnetos. Dois sensores óticos baseados em feixes de laser são utilizados para medir a posição dos magnetos.

Considerando o movimento do disco magnético inferior (disco 1), sendo m_1 a massa do disco, y_1 a altura do disco 1, c_1 o coeficiente de atrito entre o disco e a barra de vidro e g a aceleração da

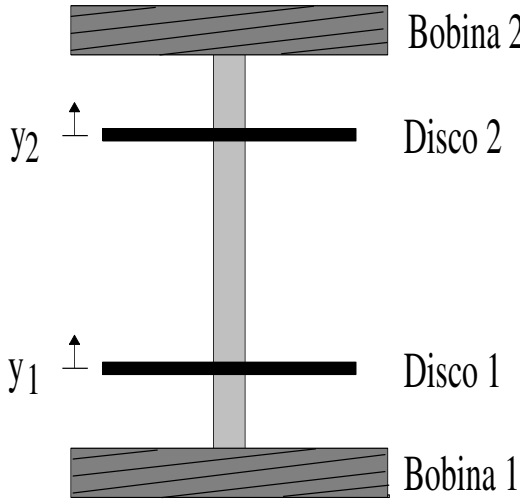

Figura 5.1: Levitador magnético.

gravidade, a equação que descreve o movimento do disco é:

$$m_1\ddot{y}_1 + c_1\dot{y}_1 + F_{m_{12}} = F_{u_{11}} - F_{u_{21}} - m_1 g \tag{5.1}$$

sendo:

$F_{m_{12}}$: força de interação magnética entre os discos.

$F_{u_{11}}$: força de interação magnética entre a bobina 1 e o disco 1,

$F_{u_{21}}$: força de interação magnética entre a bobina 2 e o disco 1,

Estas forças são descritas pelo seguinte conjunto de equações [95]:

$$\begin{aligned} F_{m_{12}} &= \frac{c}{(y_c + y_2 - y_1 + d)^4} \\ F_{u_{11}} &= \frac{i_1}{a(100y_1 + b)^4} \\ F_{u_{21}} &= \frac{i_2}{a(y_c + 100y_1 + b)^4} \end{aligned}$$

sendo i_1 a corrente aplicada à bobina 1, i_2 a corrente aplicada à bobina 2, y_2 a altura do disco 2, y_c a distância entre as bobinas 1 e 2 e $a, b, c,$ e d coeficiente reais.

As simulações foram realizadas mantendo-se uma corrente elétrica constante na bobina inferior e aplicando-se degraus de amplitudes variáveis de corrente na bobina 2. Assume-se então como variável controlada a posição y_1 do disco 1 e como variável de controle a corrente na bobina 2, i_2. A não linearidade do modelo é devida aos termos proporcionais à inversa da quarta potência da variável

controlada, y_1, no conjunto de equações anterior [92].

5.1.2 Processo de Polimerização

Trata-se de um processo de polimerização do tipo *Continuous Stirred Tank Reactor*, CSTR, descrito em detalhes em [74]. A variável controlada nesse processo é o número médio do peso molecular do polímero resultante da reação, $y(t)$, cuja unidade é $Kg/kmol$. A variável de controle é a taxa de influxo de uma substância iniciadora, $u(t)$, dada em m^3/h. O modelo em espaço de estados desse processo é [15]:

$$\begin{aligned}
\dot{x}_1(t) &= 60 - x_1(t)(2,4568\sqrt{x_2(t)} + 10) \\
\dot{x}_2(t) &= 80u(t) - 10,1022x_2(t) \\
\dot{x}_3(t) &= 0,002412x_1(t)\sqrt{x_2(t)} + 0,11218x_2(t) - 10x_3(t) \\
\dot{x}_4(t) &= 245,9811x_1(t)\sqrt{x_2(t)} - 10x_4(t) \\
y(t) &= x_4(t)/x_3(t)
\end{aligned} \tag{5.2}$$

5.1.3 Organização dos Resultados

Existem basicamente duas arquiteturas para a obtenção de modelos de sistemas dinâmicos neste trabalho, descritas no capítulo 3: aquela com pólo único e a Generalizada. Esta última permite duas sub-classes de modelos, dependendo do modo de estimação dos parâmetros dos conseqüentes das regras, se local ou global. Sendo assim, existem três abordagem possíveis. Todas foram aplicadas na modelagem do levitador magnético, porém apenas a arquitetura com pólo único foi utilizada no processo de polimerização, uma vez que este último possui uma dinâmica menos complexa. Para cada uma dessas arquiteturas, consideram-se aqui modelos locais lineares e não lineares nos conseqüentes das regras. Os modelos não lineares são modelos de Volterra de segunda ordem.

Antes da exibição dos resultados propriamente ditos, apresenta-se a atribuição dada aos parâmetros do algoritmo genético. As configurações exibidas foram obtidas considerando-se o conhecimento prévio sobre os sistemas (como a definição de dois modelos locais no modelo Generalizado para o levitador magnético), o número de parâmetros sendo otimizado (influenciando o tamanho da população, por exemplo) além de ajustes empíricos direcionados por práticas comumente aplicadas no uso de algoritmos genéticos (como o uso de uma pequena taxa de ocorrência do operador de mutação) [6, 7, 79].

Em particular, os limites mínimo e máximo das aberturas das funções de pertinência gaussianas são apenas valores extremos dos cromossomos codificados na população e não restrições que possam

vir a direcionar o processo de busca em determinado sentido. Além disso, os dados de entrada/saída de todos os sitemas analisados, e portanto seus universos de discurso, estão normalizados. Se este não fosse o caso, então de fato os valores apresentados para os limites das funções de pertinência representariam grandezas distintas em cada problema.

O valor do parâmetro de expansão percentual do operador de *crossover* aritmético foi escolhido baseado nos argumentos formulados na seção 4.7.1.

Finalmente, os resultados apresentados foram obtidos ao se executar o algoritmo genético em um computador com processador dual Intel Pentium® 4 de 3,4 GHz e 1 GB de memória RAM.

5.2 Modelos para o Levitador Magnético

Os dados de treinamento para o algoritmo genético foram obtidos excitando-se o sistema real em laboratório com um sinal periódico da forma mostrada na Figura 5.2.

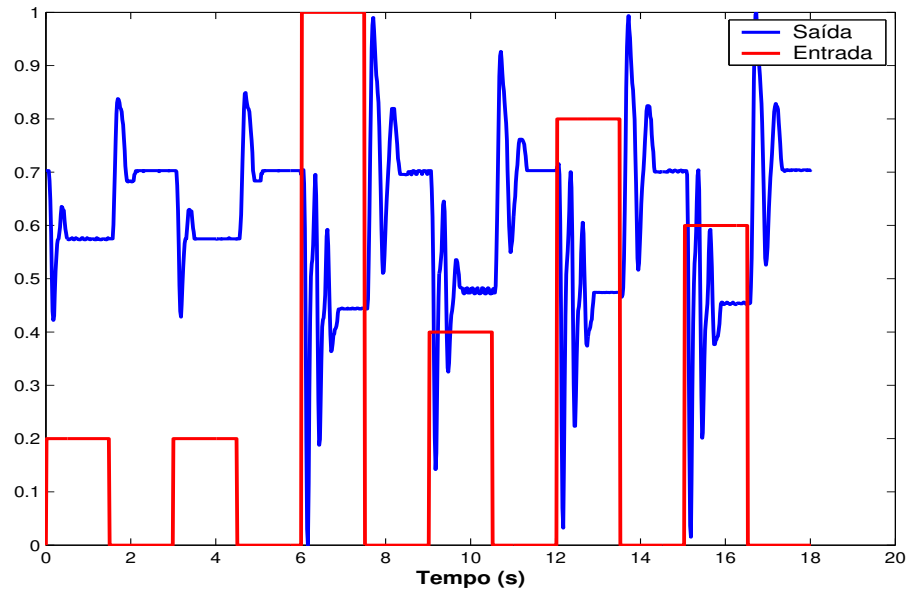

Figura 5.2: Sinais normalizados de entrada e saída para treinamento (levitador magnético).

O sinal de entrada foi projetado de forma a atender aos requisitos de ser persistentemente excitante, tornando possível a tarefa de se extrair dos dados de entrada/saída informações sobre as dinâmicas dominantes do sistema [3]. Trata-se de um sinal pseudo-aleatório composto por uma série de degraus de amplitudes diferentes. Quando o objetivo é obter modelos não lineares de sistemas dinâmicos, é importante que os sinais de entrada, além de possuir uma grande faixa de freqüência, devem também ter excursões em diferentes amplitudes, o que leva o sistema a atingir diferentes pon-

tos de operação [3]. De fato, observam-se na Figura 5.2 dois comportamentos distintos no sistema, um mais oscilatório que o outro, dependendo se a variação do sinal de entrada foi positiva ou negativa. Esta informação foi utilizada no momento de se definir como 2 a quantidade de modelos locais na arquitetura *fuzzy* TS FBO Generalizado. Isso implica que o modelo deve possuir apenas um estado (variável de entrada) na premissa das regras e duas funções de pertinência (ver seção 3.2.4).

Como os algoritmos genéticos são métodos não determinísticos de otimização, para avaliar seu desempenho na tarefa de sintonia de modelos *fuzzy* TS FBO não é adequado analisar apenas uma simulação em particular. Os resultados apresentados na seqüência foram coletados após a execução do AG por um número de rodadas igual a 35. Os gráficos exibidos correspondem a uma rodada cujos resultados estão próximos àqueles da média de todas as rodadas. Para o levitador magnético, a configuração do AG em todas as rodadas foi a exibida na tabela 5.1

Parâmetros de Execução	**TS FBO Pólo Único**	**TS FBO Generalizado**
Tamanho da população	100	100
Número máximo de gerações	80	80
Taxa de *crossover*	0,85	0,85
Taxa de mutação	0,20	0,20
Expansão percentual máxima do operador de *crossover* (δ)	0,20	0,20
Percentual da diversidade inicial para convergência	3	3
Controle de sobreposição	não	não
Abertura mínima das FP gaussianas	0,3	0,3
Abertura máxima das FP gaussianas	3	3
Número máximo de estados nos conseqüentes das regras	7	7
Número máximo de variáveis de entrada (max_{vl})	4	-
Número máximo de FP por variável de entrada (max_{tl})	3	-
Limite inferior para aplicação das medidas de similaridade (ϵ)	0,2	-
Potência para uso na equação do limite de similaridade (ρ)	8	-

Tabela 5.1: Configuração do AG para o sistema levitador magnético.

O *Tamanho da população* é o número de indivíduos (cromossomos) presente em cada geração. O

processo evolutivo continua até que o número de gerações alcance o *Número máximo de gerações* ou a diversidade da população atinja o *Percentual da diversidade inicial para convergência*. Cada indivíduo deve respeitar os limites *Número máximo de estados nos conseqüentes das regras*, *Número máximo de variáveis de entrada* e *Número máximo de FP por variável de entrada*. Além disso, as aberturas das funções de pertinência são restritas ao intervalo entre *Abertura máxima das FP gaussianas* e *Abertura mínima das FP gaussianas*. Finalmente, as probabilidades de ocorrência das operações de *crossover* (que utiliza o valor *Expansão percentual máxima do operador de crossover* para o *crossover* aritmético) e mutação são dadas pelos parâmetros *Taxa de crossover* e *Taxa de mutação*, respectivamente. Como o operador de *crossover* aritmético possui maior capacidade de manter a diversidade da população (pois tende a gerar indivíduos filhos distintos dos pais) este é aplicado com probabilidade de ocorrência maior que o operador de *crossover* uniforme. Cada vez que uma operação de *crossover* deve ser efetuada, realiza-se um sorteio entre o operador aritmético, com probabilidade de ocorrência de 70%, e o operador uniforme, com probabilidade de 30%. A opção de controle de sobreposição não foi usada uma vez que a interpretação adotada em todas as simulações para a representação cromossômica foi a com distâncias absolutas entre os centros das funções de pertinência, e não a com distâncias relativas. Esta última opção seria uma perspectiva para trabalhos futuros.

Seguindo a ordem de apresentação das representações cromossômicas do capítulo 4, serão primeiramente exibidos os resultados para a arquitetura *fuzzy* TS FBO com pólo único e em seguida a Generalizada.

5.2.1 Modelo *fuzzy* TS FBO com Pólo Único

A tabela 5.2 resume os resultados obtidos após 35 rodadas do algoritmo genético. As informações mostradas na tabela referem-se principalmente ao compromisso entre a qualidade do modelo (valor pequeno para o EQM, Erro Quadrático Médio) e sua complexidade (número de parâmetros). Além do valor do EQM, também é mostrado o valor calculado do critério de Akaike (AIC) segundo a equação (4.3). As duas últimas colunas da tabela apresentam os resultados com o uso de modelos locais respectivamente lineares e não lineares nos conseqüentes das regras[1].

A fim de ilustrar a vantagem da modificação implementada no operador de *crossover* aritmético, a tabela 5.3 exibe um resumo dos resultados obtidos em um conjunto de simulações com a mesma configuração apresentada na tabela 5.1 com a única modificação de se zerar o parâmetro de expansão δ. A tabela apresenta os resultados correspondentes a modelos locais lineares nos conseqüentes das regras, para os quais será apresentado na seqüência um exemplo de resultado.

[1]Na seqüência do texto, para a simplicidade da escrita, será usada a notação de "modelo linear" e "modelo não linear" quando o mais apropriado seria respectivamente "modelo local linear nos conseqüentes das regras" e "modelo local não linear nos conseqüentes das regras."

	Linear	**Não Linear**
EQM médio	0,0016962	0,00093761
Número médio de parâmetros	52,54	111,43
AIC médio	-3996,40	-4201,63
Menor EQM	0,0005747	0,0005061
Número de parâmetros do menor EQM	213	154
AIC do menor EQM	-4228,85	-4416,22
Maior EQM	0,0074675	0,0035242
Número de parâmetros do maior EQM	23	94
AIC do maior EQM	-2994,77	-3318,51
Desvio padrão do EQM	0,0015281	0,0006193
Desvio padrão do AIC	326,04	298,16
Número médio de gerações	46,00	35,11
Tempo médio de processamento por rodada (min)	26,05	51,38

Tabela 5.2: Resumo dos resultados do modelo *fuzzy* TS FBO com pólo único (levitador magnético).

	Linear
EQM médio	0,0019517
Número médio de parâmetros	50,43
AIC médio	-3954,78
Número médio de gerações	32,83

Tabela 5.3: Resumo dos resultados do modelo *fuzzy* TS FBO com pólo único sem o parâmetro de expansão δ (levitador magnético).

Observa-se que o algoritmo genético forneceu, com o parâmetro de expansão, um EQM médio menor ao caso sem tal parâmetro. Visto de outra forma, os resultados sem o parâmetro de expansão (*crossover* aritmético original) possuíam em média 4% menos parâmetros, porém fornecendo um EQM médio 15% maior. O valor do AIC médio confirma as comparações anteriores, uma vez o valor menor foi obtido com o parâmetro de expansão (lembrando que um menor AIC corresponde a um melhor resultado). Além disso, o número médio de gerações com o parâmetro de expansão foi sensivelmente maior que sem o mesmo, corroborando a afirmação da seção 4.7.1 da manutenção da diversidade da população.

A tabela 5.4 apresenta os resultados do melhor indivíduo de todas as rodadas, considerando-se o menor EQM. Ao contrário do que será observado nos resultados para a arquitetura *fuzzy* TS FBO Generalizado, aqui ambos os modelos linear e não linear apresentaram valores próximos um do outro para o melhor EQM de todas as rodadas.

Os valores de EQM apresentados na tabela 5.4 são consideravelmente menores que aqueles obti-

	Linear	**Não Linear**
Erro quadrático médio	0,0005747	0,0005061
AIC	-4228,85	-4416,22
Número de regras	24	4
Número de parâmetros nos conseqüentes	8	36
Pólos	0,85702 $\pm j$ 0,31570	0,80883 $\pm j$ 0,23776
Número de parâmetros do modelo	213	154

Tabela 5.4: Melhor resultado (menor EQM) do modelo *fuzzy* TS FBO com pólo único (levitador magnético).

dos para a arquitetura Generalizada apresentada na próxima seção. No entanto, deve ser notado que a quantidade de parâmetros dos modelos com pólo único é bem superior. Por se tratar de uma arquitetura mais flexível, durante o processo evolutivo surgem indivíduos que codificam modelos com elevado número de parâmetros. A função de *fitness* implementada e as medidas de similaridade impedem que o número de parâmetros de todos os indivíduos aumente arbitrariamente. Assim, embora os melhores indivíduos[2] de todas as rodadas de fato apresentem uma grande quantidade de parâmetros, os indivíduos com EQM próximo à média da tabela 5.2 apresentam consideravelmente um menor número de parâmetros. Na verdade, considerando os parâmetros definidos na tabela 5.1, a arquitetura *fuzzy* TS FBO com pólo único teria em princípio liberdade para gerar modelos com até 675 parâmetros, no caso de modelos locais lineares, e até 2943 parâmetros, para modelos locais não lineares.

Como ilustração, exibem-se na seqüência os resultados detalhados de uma rodada cujo EQM se aproximou da média dos resultados apresentados na tabela 5.2. Em particular, considera-se o uso de modelos locais lineares nos conseqüentes das regras. Estes modelos são escolhidos pois o valor médio do EQM é similar aos encontrados com a arquitetura Generalizada, de forma que seja possível comparar o desempenho de cada proposta. Ao se escolher modelos com valores próximos de EQM (e aceitáveis), o julgamento pelo melhor será baseado na complexidade de cada um, a qual é avaliada em termos do seu número de parâmetros. Em suma, o melhor modelo dentre um conjunto com valores próximos de EQM será aquele com o menor número de parâmetros.

Na simulação ora apresentada, o AG consumiu 43 gerações, fornecendo ao final do processo evolutivo um valor de diversidade da população igual a 0,31 (a diversidade inicial foi 25,34). A Figura 5.3 exibe a curva da diversidade da população ao longo das gerações, com um decaimento característico.

O limite para atuação das medidas de similaridade, como exposto na seção 4.9, toma como base

[2]A não ser que seja explicitado o contrário, o termo "melhor indivíduo" referir-se-á ao indivíduo que representa o modelo mais preciso, ou seja, com o menor EQM.

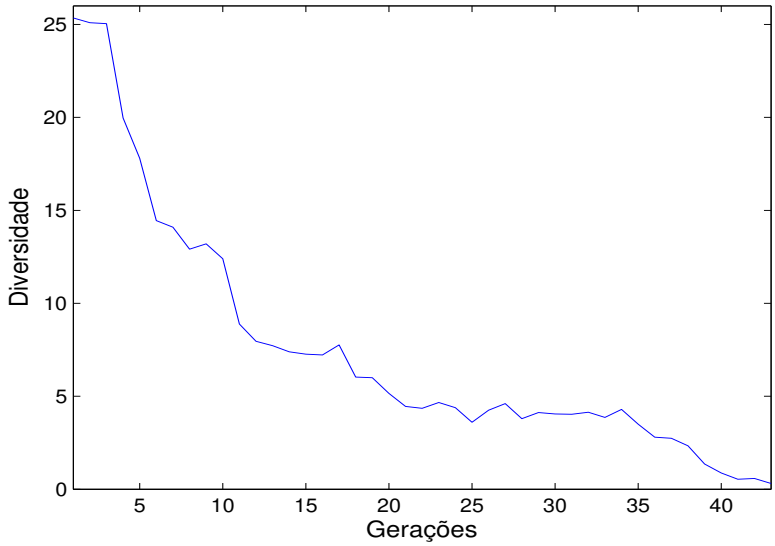

Figura 5.3: Diversidade da população para o modelo *fuzzy* TS FBO com pólo único e modelos locais lineares (levitador magnético).

de cálculo o valor da diversidade corrente da população. Utilizando os parâmetros definidos na tabela 5.1, o AG realizou simplificações nas funções de pertinência respeitando o limite de similaridade mostrado na Figura 5.4.

O comportamento do limite de similaridade exibido na Figura 5.4 é semelhante ao da Figura 4.12, obtida de forma analítica. O resultado é exatamente o esperado, poucas simplificações de funções de pertinência nas primeiras gerações do AG e um progressivo aumento à medida que o AG tende a convergir.

A convergência da população, indicada pela Figura 5.3, faz com que o valor do critério de Akaike médio da população, base para o cálculo do *fitness*, também tenda a se estabilizar, o que é confirmado pelo gráfico da Figura 5.5.

Ao final da rodada, o AG forneceu a solução apresentada na tabela 5.5. Tal solução codifica um modelo com um total de 23 parâmetros. Para fins de comparação, note-se que a melhor solução de todas as rodadas possuia aproximadamente 9 vezes mais parâmetros, porém reduzindo o valor do EQM em apenas 3 vezes.

A Figura 5.6 exibe as funções de pertinência para os termos lingüísticos de entrada, completando a solução fornecida pelo AG. O algoritmo genético selecionou como variável única de entrada o primeiro estado ortonormal[3]. Desse modo, com uma variável de entrada e dois termos lingüísticos

[3]Na verdade, no lugar de "primeiro estado ortonormal" seria mais correto falar "variável resultante da inferência *fuzzy* TS aplicada aos sinais obtidos na filtragem da entrada pela primeira função da base ortonormal", porém a expressão

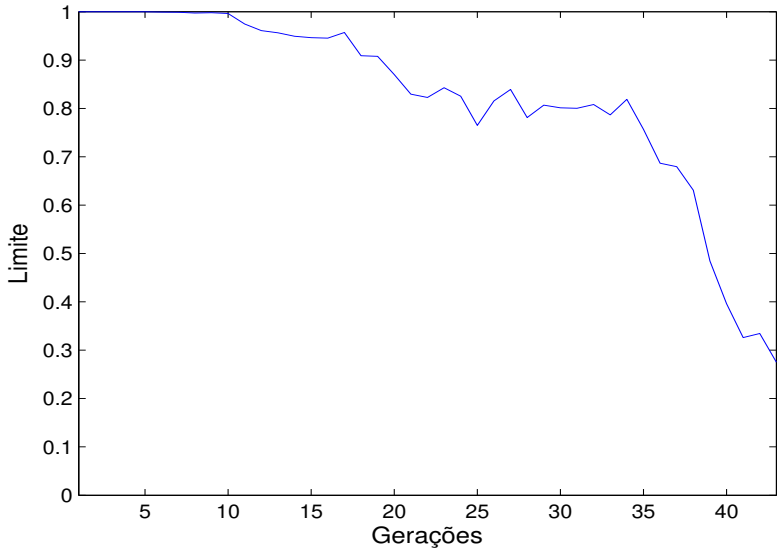

Figura 5.4: Limite de similaridade para o modelo *fuzzy* TS FBO com pólo único e modelos locais lineares (levitador magnético).

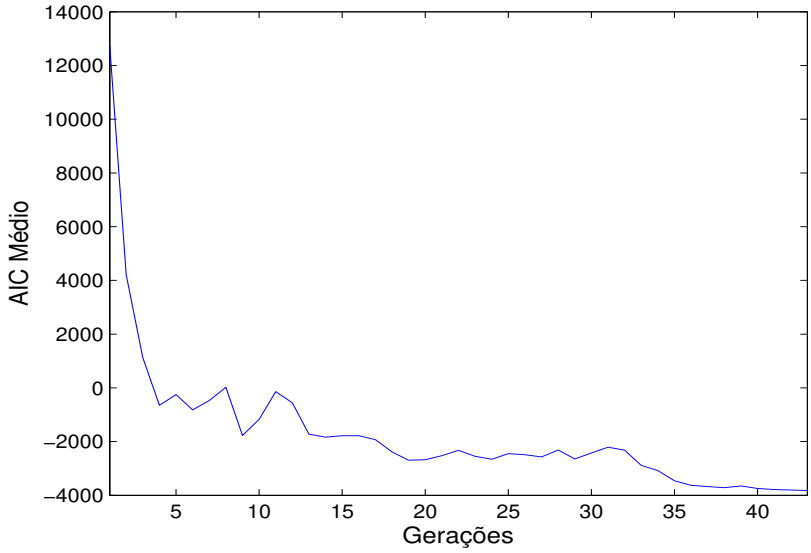

Figura 5.5: Evolução do AIC médio da população para o modelo *fuzzy* TS FBO com pólo único e modelos locais lineares (levitador magnético).

Pólo	0,79924 $\pm j$ 0,26822
Número de regras	2
Número de parâmetros nos conseqüentes	8
EQM	0,0019020

Tabela 5.5: Solução fornecida pelo AG para o modelo *fuzzy* TS FBO com pólo único e modelos locais lineares (levitador magnético).

forma-se uma base com duas regras.

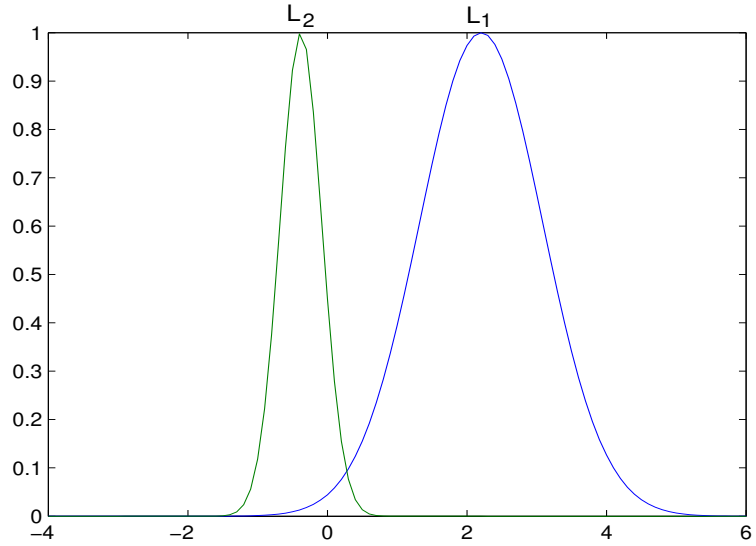

Figura 5.6: Funções de pertinência de entrada para o modelo *fuzzy* TS FBO com pólo único e modelos locais lineares (levitador magnético).

Finalmente, a Figura 5.7 exibe o resultado final obtido. São comparados os dados de saída do sistema real com os fornecidos pelo modelo *fuzzy* TS FBO com pólo único, para um conjunto de dados de validação diferente do utilizado na estimação dos parâmetros dos modelos locais. Para essa simulação o valor do EQM foi de 0,0019020.

Para fins de comparação, ao se adotar a metodologia tradicional de distribuição homogênea das funções de pertinência no universo de discurso proposta em [39] e aplicada em [16], e considerando um modelo com a mesma estrutura do exemplificado anteriormente (número de regras, estados ortonormais, etc.), obtém-se um EQM de 0,0025306, ou seja, 33% maior que a solução fornecida pelo algoritmo genético.

"estado ortonormal" será utilizada por conveniência de escrita.

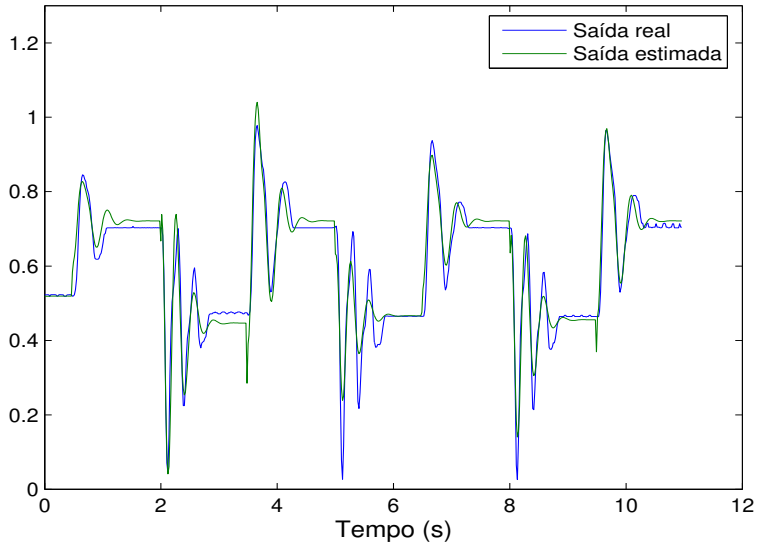

Figura 5.7: Saída real e a fornecida pelo modelo *fuzzy* TS FBO com pólo único e modelos locais lineares (levitador magnético).

5.2.2 Modelo *fuzzy* TS FBO Generalizado com Estimação Local

Esta seção e a seguinte apresentam os resultados obtidos com a arquitetura Generalizada, variando-se o modo de estimação dos parâmetros dos modelos locais nos conseqüentes das regras. A tabela 5.6 resume os resultados obtidos após 35 rodadas do algoritmo genético, considerando-se estimação local.

Assim como para o caso da arquitetura com pólo único, a tabela 5.7 exibe um resumo dos resultados obtidos em um conjunto de simulações sem o parâmetro de expansão δ no operador de *crossover* aritmético. Apresentam-se somente os resultados correspondentes a modelos locais lineares nos conseqüentes das regras, que serão exemplificados a seguir.

Embora nesse caso o número médio de gerações tenha sido aproximadamente o mesmo nos dois conjuntos de simulações, aquele utilizando o parâmetro de expansão no *crossover* aritmético forneceu tanto um EQM médio quanto um número médio de parâmetros menores ao caso sem o parâmetro de expansão, resultando em um AIC médio também melhor.

A tabela 5.8 apresenta os resultados do melhor indivíduo de todas as rodadas, considerando-se o menor EQM. Embora o modelo não linear tenha fornecido o menor EQM, deve-se observar que este possui um número de parâmetros mais de três vezes maior que o modelo linear.

Exibem-se na seqüência os resultados detalhados de uma rodada cujo desempenho se aproximou da média dos resultados apresentados na tabela 5.6. Em particular, considera-se o uso de modelos

	Linear	**Não Linear**
EQM médio	0,0017423	0,0011960
Número médio de parâmetros	25,86	61,26
AIC médio	-3901,70	-4128,91
Menor EQM	0,0015965	0,0005823
Número de parâmetros do menor EQM	26	83
AIC do menor EQM	-3952,70	-4465,75
Maior EQM	0,0024847	0,0019559
Número de parâmetros do maior EQM	24	45
AIC do maior EQM	-3676,50	-3782,26
Desvio padrão do EQM	0,0002055	0,0005619
Desvio padrão do AIC	65,05	286,72
Número médio de gerações	36,20	34,17
Tempo médio de processamento por rodada (min)	9,28	17,15

Tabela 5.6: Resumo dos resultados do modelo *fuzzy* TS FBO Generalizado com estimação local (levitador magnético).

	Linear
EQM médio	0,0021789
Número médio de parâmetros	26,20
AIC médio	-3769,00
Número médio de gerações	37,29

Tabela 5.7: Resumo dos resultados do modelo *fuzzy* TS FBO Generalizado com estimação local sem o parâmetro de expansão δ (levitador magnético).

locais lineares nos conseqüentes das regras.

Na simulação ora apresentada, o AG consumiu 29 gerações, fornecendo ao final do processo evolutivo um valor de diversidade da população igual a 0,20 (a diversidade inicial foi 11,26). A Figura 5.8 exibe a curva da diversidade da população ao longo das gerações, com um decaimento característico.

A convergência da população, indicada pela Figura 5.8, faz com que o valor do critério de Akaike médio da população, base para o cálculo do *fitness*, também tenda a se estabilizar, o que é confirmado pelo gráfico da Figura 5.9.

Com o progresso da evolução, o erro quadrático médio do melhor indivíduo da população tende a diminuir, conforme pode ser observado na Figura 5.10.

Ao final da rodada, o AG forneceu a solução apresentada na tabela 5.9. Tal solução codifica um modelo com um total de 25 parâmetros.

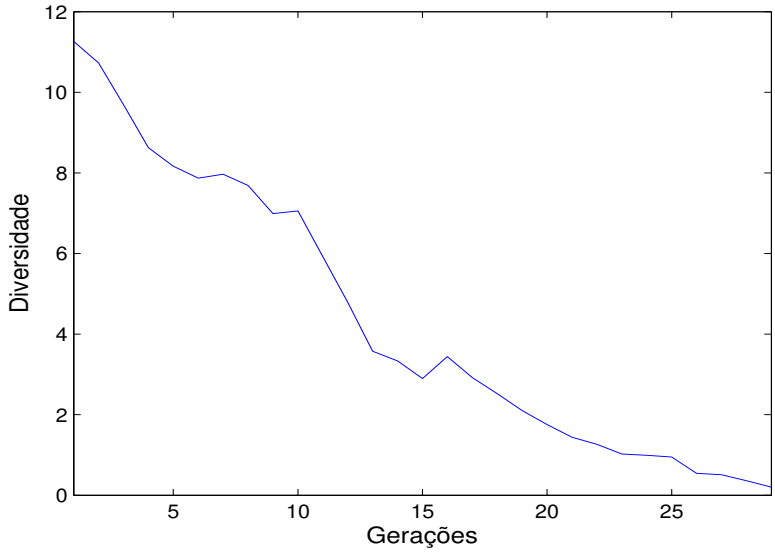

Figura 5.8: Diversidade da população para o modelo *fuzzy* TS FBO Generalizado com estimação local e modelos locais lineares (levitador magnético).

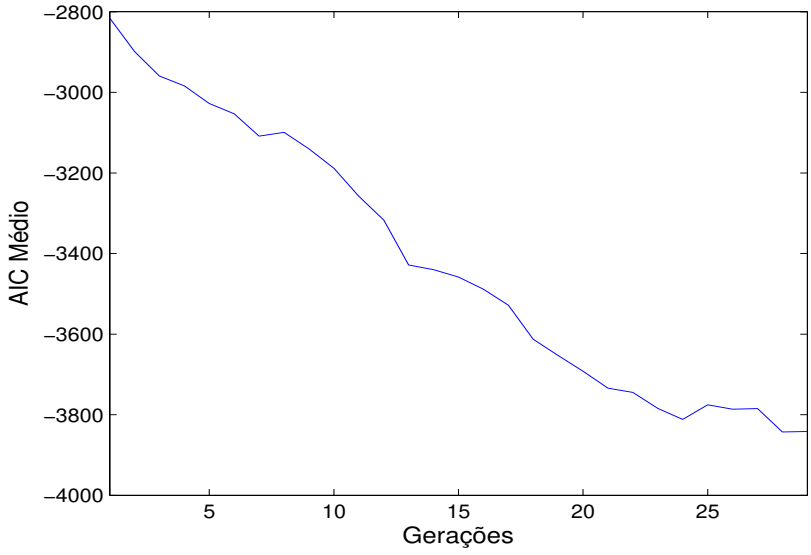

Figura 5.9: Evolução do AIC médio da população para o modelo *fuzzy* TS FBO Generalizado com estimação local e modelos locais lineares (levitador magnético).

	Linear	**Não Linear**
Erro Quadrático Médio	0,0015965	0,0005823
AIC	-3952,70	-4465,75
Número de Regras	2	2
Pólos	0,82869 $\pm j$ 0,27907	0,82271 $\pm j$ 0,27665
	0,80999 $\pm j$ 0,26396	0,80093 $\pm j$ 0,24151
Número de Parâmetros do modelo	26	83
Número de Estados no 1º modelo local	6	7
Número de Estados no 2º modelo local	7	7

Tabela 5.8: Melhor resultado do modelo *fuzzy* TS FBO Generalizado com estimação local (levitador magnético).

Primeiro pólo	0,89091 $\pm j$ 0,16241
Segundo pólo	0,82444 $\pm j$ 0,26901
Número de estados no 1º modelo	5
Número de estados no 2º modelo	7
Índice do estado nas premissas	2
EQM	0,0017220

Tabela 5.9: Solução fornecida pelo AG para o modelo *fuzzy* TS FBO Generalizado com modelos locais lineares e estimação local (levitador magnético).

Antes de apresentar o conjunto das funções de pertinência para os termos lingüísticos de entrada, a Figura 5.11 exibe a dinâmica da variável de estado resultante da inferência *fuzzy* TS (equação (2.31) do capítulo 2) envolvendo o segundo estado da base de funções ortonormais de cada modelo local.

Observam-se duas "regiões de operação" bem distintas: uma em torno de zero e outra entre um e dois. As funções de pertinência encontradas pelo AG, mostradas na Figura 5.12, são capazes de classificar corretamente tais regiões de operação. Note-se que a FP "L2" é praticamente nula em torno de zero, ocorrendo o mesmo para a FP "L1" após aproximadamente o valor 1.

A capacidade do AG de realizar otimização com múltiplos objetivos é evidente nestes resultados. Como explicitado na seção 4.3.2, desde que os operadores reprodutivos e a função de *fitness* levem em conta a diversidade semântica codificada em um mesmo cromossomo, durante o processo evolutivo os parâmetros se adequam uns aos outros, e de forma ótima. Esta busca multidimensional eficiente permite a obtenção de resultados indicativos da grande viabilidade da metodologia ora proposta.

No caso presente, estes resultados podem ser resumidos nas figuras 5.13 e 5.14. A Figura 5.13 exibe as saídas dos dois modelos locais juntamente com a saída da planta. Ao se analisar esta figura juntamente com as Figuras 5.11 e 5.12 conclui-se que cada modelo local de fato representa o sistema em uma região de operação bem determinada. A base de regras do modelo *fuzzy* TS FBO Gene-

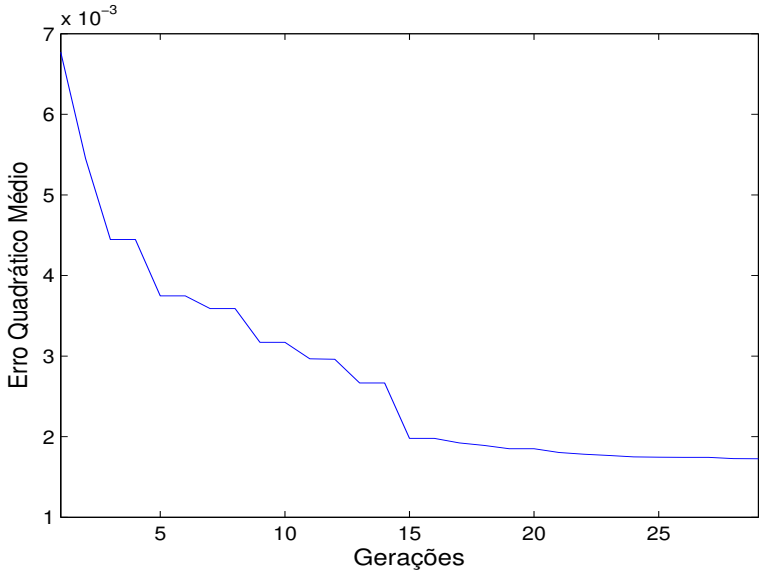

Figura 5.10: Evolução do EQM do melhor indivíduo da população para o modelo *fuzzy* TS FBO Generalizado com estimação local e modelos locais lineares (levitador magnético).

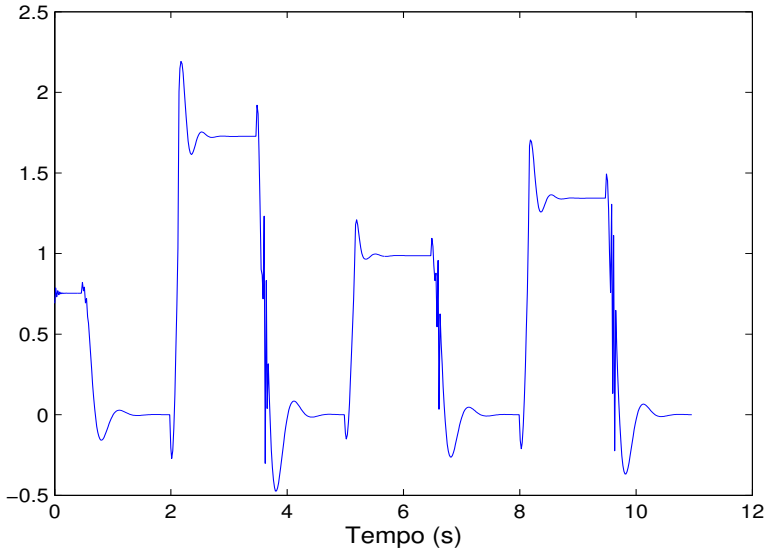

Figura 5.11: Dinâmica do estado na premissa das regras para o modelo *fuzzy* TS FBO Generalizado com estimação local e modelos locais lineares (levitador magnético).

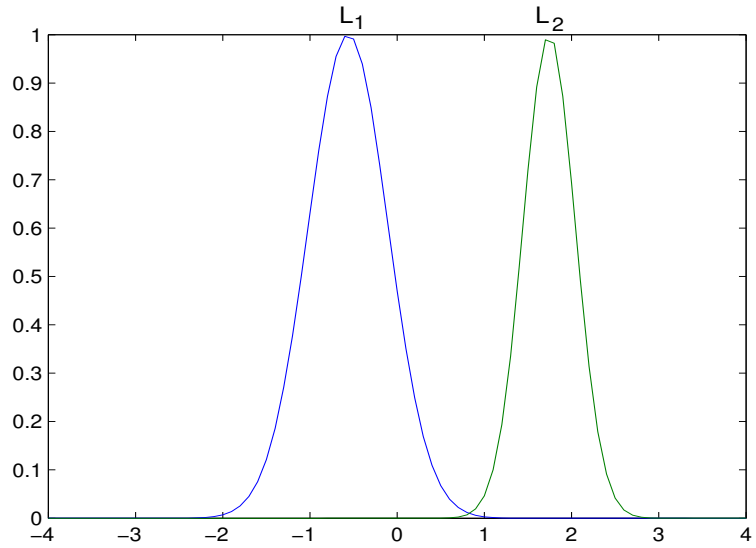

Figura 5.12: Funções de pertinência de entrada para o modelo *fuzzy* TS FBO Generalizado com estimação local e modelos locais lineares (levitador magnético).

ralizado tem como função selecionar cada modelo local tomando como base o valor do estado nas premissas das regras e as configurações das funções de pertinência fornecidos pelo AG.

Finalmente, a Figura 5.14 exibe o resultado final obtido. São comparados os dados de saída do sistema real com os fornecidos pelo modelo *fuzzy* TS FBO Generalizado com estimação local dos conseqüentes das regras. Para essa simulação o valor do EQM foi de 0,0017220. Observa-se que o modelo obtido consiste nas melhores aproximações de cada um dos modelos locais lineares. Por outro lado, o método não otimizado de distribuição homogênea das funções de pertinência no universo de discurso fornece, para um modelo com a mesma estrutura do descrito acima, um EQM de 0,0028360, 65% maior que a solução fornecida pelo algoritmo genético.

5.2.3 Modelo *fuzzy* TS FBO Generalizado com Estimação Global

A tabela 5.10 é análoga à tabela 5.6, porém obtida utilizando estimação global dos parâmetros dos modelos locais.

Para fins de comparação, a tabela 5.11 exibe um resumo dos resultados obtidos em um conjunto de simulações sem o parâmetro de expansão δ no operador de *crossover* aritmético. Apresentam-se somente os resultados correspondentes a modelos locais não lineares nos conseqüentes das regras, os quais serão exemplificados na seqüência.

Assim como no caso do modelo Generalizado com estimação local, o número médio de gerações

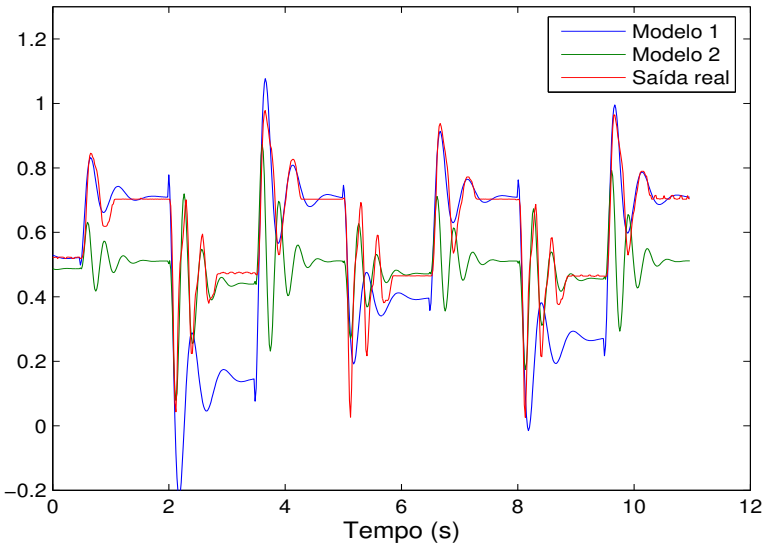

Figura 5.13: Modelos locais para o sistema *fuzzy* TS FBO Generalizado com estimação local e modelos locais lineares (levitador magnético).

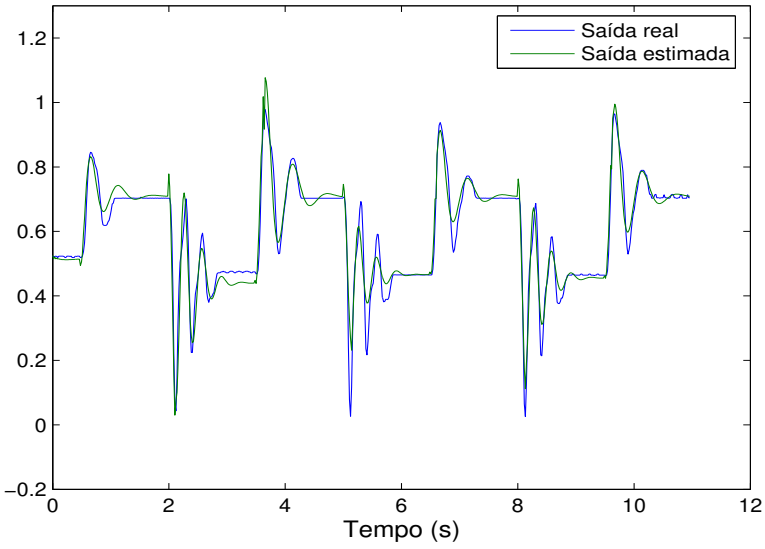

Figura 5.14: Saída real e a fornecida pelo modelo *fuzzy* TS FBO Generalizado com estimação local e modelos locais lineares (levitador magnético).

	Linear	**Não Linear**
EQM médio	0,0083562	0,0018611
Número médio de parâmetros	20,71	43,03
AIC médio	-3178,80	-3879,90
Menor EQM	0,0055663	0,0015395
Número de parâmetros do menor EQM	22	45
AIC do menor EQM	-3237,60	-3941,30
Maior EQM	0,0141410	0,0027026
Número de parâmetros do maior EQM	20	36
AIC do maior EQM	-3200,40	-3808,60
Desvio padrão do EQM	0,00218605	0,0002675
Desvio padrão do AIC	81,17	45,00
Número médio de gerações	25,94	22,29
Tempo médio de processamento por rodada (min)	5,21	6,30

Tabela 5.10: Resumo dos resultados do modelo *fuzzy* TS FBO Generalizado com estimação global (levitador magnético).

	Linear
EQM médio	0,0019060
Número médio de parâmetros	44,89
AIC médio	-3893,40
Número médio de gerações	21,54

Tabela 5.11: Resumo dos resultados do modelo *fuzzy* TS FBO Generalizado com estimação global sem o parâmetro de expansão δ (levitador magnético).

com ou sem o parâmetro de expansão no *crossover* aritmético foi aproximadamente o mesmo. No entanto o uso do parâmetro de expansão forneceu ao mesmo tempo resultados médios melhores em termos de um menor EQM médio e um menor número médio de parâmetros, embora com o valor do AIC médio ligeiramente maior.

A tabela 5.12 apresenta os resultados do melhor indivíduo de todas as rodadas, considerando-se o menor EQM. Embora novamente o modelo não linear tenha fornecido o menor EQM, o número de parâmetros deste é aproximadamente o dobro do modelo linear, e não três vezes, como no caso de estimação local. Ou seja, uma menor diferença no número de parâmetros dos modelos causou uma maior diferença entre os menores EQMs. Em outras palavras, o modelo com estimação global melhora mais rapidamente seu desempenho com a inclusão de mais parâmetros do que o modelo com estimação local. No entanto, ao se analisar a tabela 5.12 juntamente com a tabela 5.8, observa-se que o modelo Generalizado linear com estimação local fornece aproximadamente o mesmo EQM do

modelo Generalizado não linear com estimação global, possuindo pouco mais da metade do número total de parâmetros deste último.

	Linear	**Não Linear**
Erro Quadrático Médio	0,0055663	0,0015395
AIC	-3237,60	-3941,30
Número de Regras	2	2
Pólos	0,79327 $\pm j$ 0,25582	0,84744 $\pm j$ 0,25934
	0,16555 $\pm j$ 0,03512	0,42894 $\pm j$ 0,72567
Número de Parâmetros do modelo	22	45
Número de Estados no 1º modelo local	4	6
Número de Estados no 2º modelo local	5	2

Tabela 5.12: Melhor resultado do modelo *fuzzy* TS FBO Generalizado com estimação global (levitador magnético).

Para facilitar a comparação do desempenho dos diferentes modelos abordados neste trabalho, exibem-se na seqüência os resultados detalhados de uma rodada cujo desempenho se aproximou da média dos resultados apresentados na tabela 5.10, considerando-se o uso de modelos locais não lineares nos conseqüentes das regras, os quais apresentaram um EQM médio próximo ao detalhado anteriormente para o modelo com estimação local.

Na simulação ora apresentada, o AG consumiu 18 gerações, fornecendo ao final do processo evolutivo um valor de diversidade da população igual a 0,31 (a diversidade inicial foi 10,26). A Figura 5.15 exibe a curva da diversidade da população ao longo das gerações. A estabilização do valor do critério de Akaike médio da população, exibido na Figura 5.16, confirma a convergência da população.

A Figura 5.17 apresenta a evolução do erro quadrático médio do melhor indivíduo da população. Neste ponto deve ser notado que, embora a estratégia de elitismo esteja sendo utilizada, pode acontecer de o melhor indivíduo de uma geração apresentar um EQM maior que o melhor da geração anterior. A explicação para tal evento é simples: como está sendo utilizado um critério de avaliação que, além do desempenho (em termos de um baixo erro) considera também a complexidade do modelo, um certo indivíduo, por codificar um modelo com menos parâmetros, pode possuir um valor de *fitness* mais elevado, mesmo com um EQM maior. Tal comportamento ocorre nessa simulação em particular, conforme pode ser observado na Figura 5.17.

Ao final da rodada, o AG forneceu a solução apresentada na tabela 5.13, a qual codifica um modelo com um total de 36 parâmetros.

Ao se proceder com a estimação global dos parâmetros dos modelos locais de cada regra, não é esperado que ao se analisar cada um destes modelos locais isoladamente seja observada alguma

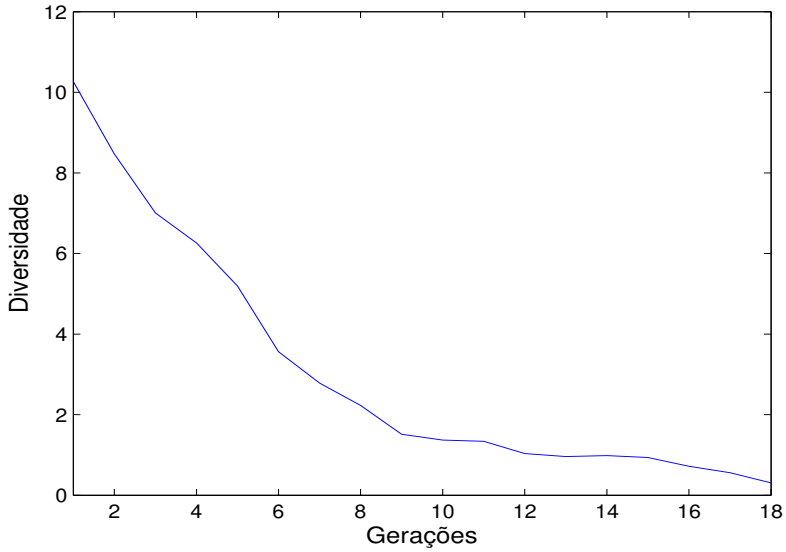

Figura 5.15: Diversidade da população para o modelo *fuzzy* TS FBO Generalizado com modelos locais não lineares e estimação global (levitador magnético).

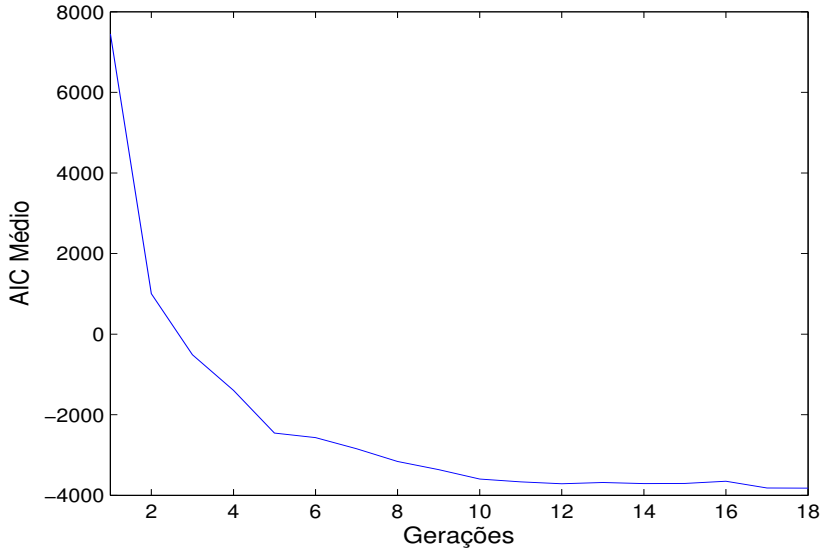

Figura 5.16: Evolução do AIC médio da população para o modelo *fuzzy* TS FBO Generalizado com modelos locais não lineares e estimação global (levitador magnético).

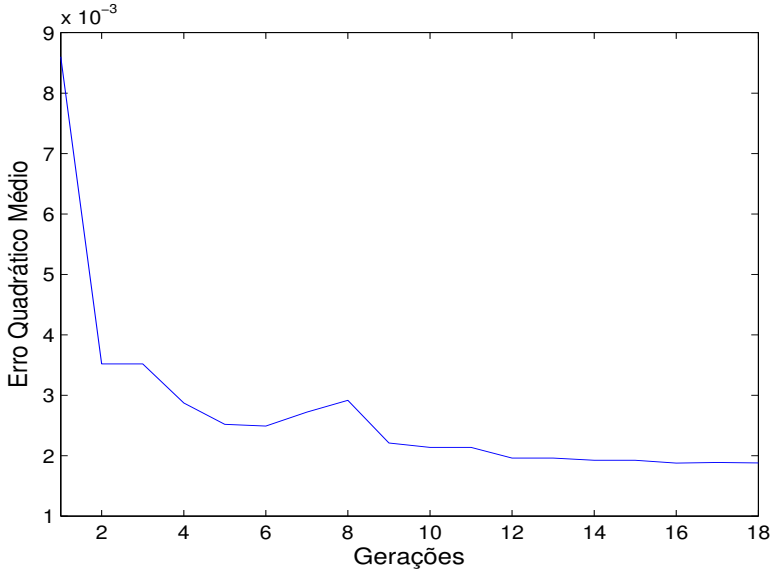

Figura 5.17: Evolução do EQM do melhor indivíduo da população para o modelo *fuzzy* TS FBO Generalizado com modelos locais não lineares e estimação global (levitador magnético).

Primeiro pólo	0,90461 $\pm j$ 0,19269
Segundo pólo	0,84671 $\pm j$ 0,32033
Número de estados no 1º modelo	3
Número de estados no 2º modelo	4
Índice do estado nas premissas	1
EQM	0,0018657

Tabela 5.13: Solução fornecida pelo AG para o modelo *fuzzy* TS FBO Generalizado com modelos locais não lineares e estimação global (levitador magnético).

dinâmica particular do sistema real, como é evidente no caso de estimação local analisado na seção anterior. Os pólos de cada modelo local exibidos na tabela 5.13 não caracterizam única e completamente uma região de operação específica do sistema real (ver Figura 5.20), e apenas após a interpolação dos modelos locais, através da inferência *fuzzy* TS, obtém-se um modelo representativo do sistema real.

Dessa forma, embora o comportamento da variável lingüística de entrada, exibido na Figura 5.18, seja semelhante ao observado no modelo com estimação local, as funções de pertinência encontradas pelo AG não particionam o universo de discurso do mesmo modo anterior. Ao contrário, são praticamente sobrepostas (Figura 5.19).

Esta sobreposição quase completa das funções de pertinência indica que é mais razoável aplicar

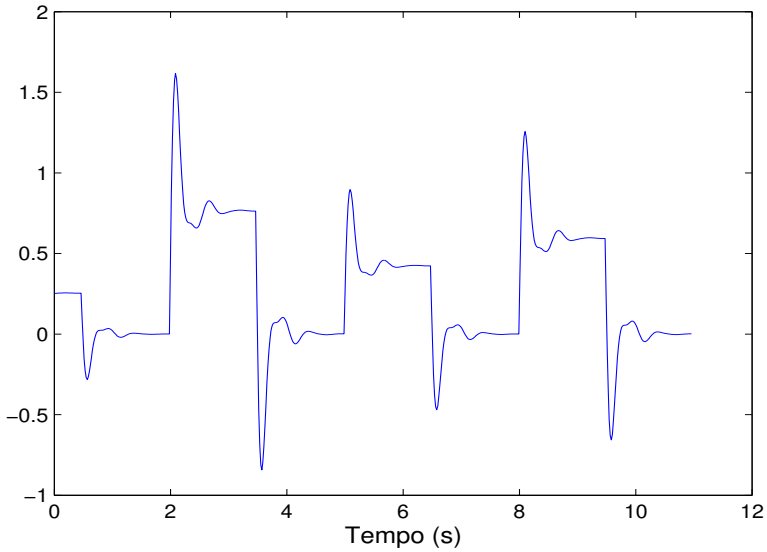

Figura 5.18: Dinâmica do estado na premissa das regras para o modelo *fuzzy* TS FBO Generalizado com modelos locais não lineares e estimação global (levitador magnético).

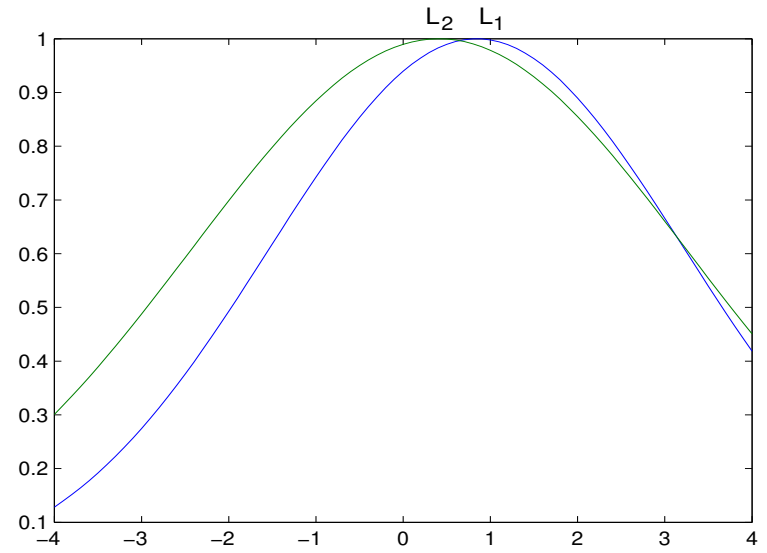

Figura 5.19: Funções de pertinência de entrada para o modelo *fuzzy* TS FBO Generalizado com modelos locais não lineares e estimação global (levitador magnético).

a arquitetura *fuzzy* TS FBO Generalizada com a estimação local dos parâmetros dos modelos locais, tendo em vista sua motivação e definição originais descritas na seção 3.2.4. O efeito da sobreposição é a incapacidade de o processo de inferência *fuzzy* TS de diferenciar as regiões de operação distintas do sistema modeladas individualmente por cada regra, uma vez que praticamente o mesmo grau de ativação será atribuído a cada uma destas regras.

A Figura 5.20 corrobora os comentários anteriores, ao exibir as saídas dos dois modelos locais juntamente com a saída da planta. Observa-se que não há correlação direta entre cada modelo local e alguma região de operação específica do sistema real.

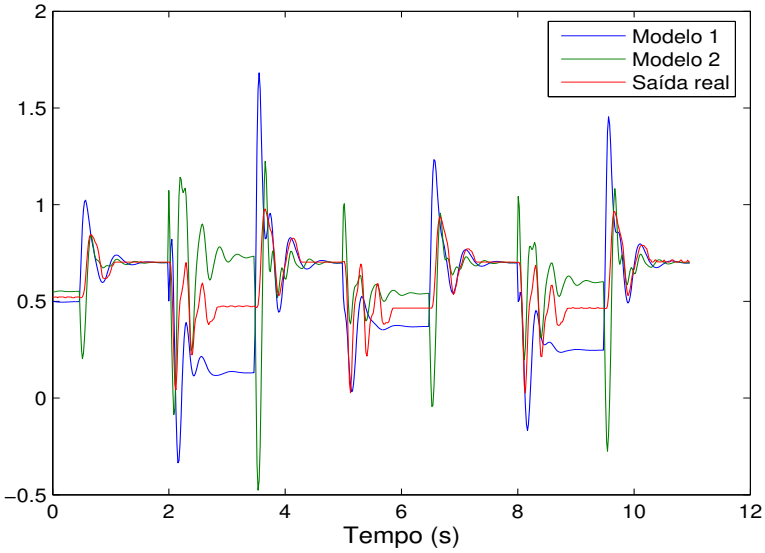

Figura 5.20: Modelos locais para o sistema *fuzzy* TS FBO Generalizado com modelos locais não lineares e estimação global (levitador magnético).

Por fim, a Figura 5.21 exibe o modelo final obtido. O valor do EQM para esse modelo com 36 parâmetros foi de 0,0018657. Para fins de comparação, os resultados exibidos anteriormente para o modelo Generalizado com estimação local foram um EQM de 0,0017220 para um modelo com 25 parâmetros, ou seja, um menor EQM com um modelo mais simples. Ainda mais, considerando o método não otimizado de distribuição homogênea das funções de pertinência no universo de discurso, o modelo apresenta um EQM de 0,0064353, 245% maior que a solução fornecida pelo algoritmo genético para o modelo *fuzzy* TS FBO Generalizado com modelos locais não lineares e estimação global.

Com fins ilustrativos, a Figura 5.22 apresenta os modelos locais obtidos em outra rodada do AG cujo EQM para um modelo também com 36 parâmetros foi de 0,0018440, próximo ao obtido no modelo previamente descrito. Observa-se que os modelos locais obtidos são praticamente anti-

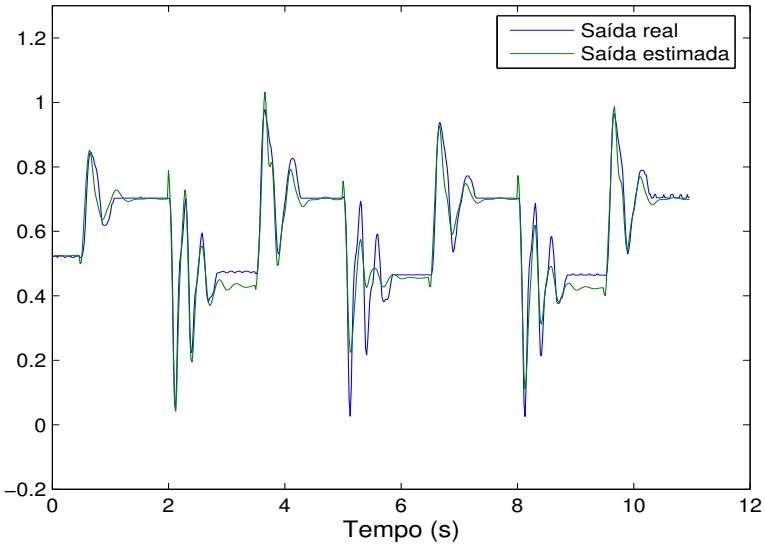

Figura 5.21: Saída real e a fornecida pelo modelo *fuzzy* TS FBO Generalizado com modelos locais não lineares e estimação global (levitador magnético).

simétricos. Como as funções de pertinência para este caso também estavam sobrepostas, ocorre uma interpolação dos modelos locais obtidos, resultando em uma estimativa global razoável.

5.2.4 Comentários sobre os Modelos do Levitador Magnético

O desempenho do modelo *fuzzy* TS FBO Generalizado com estimação local foi superior ao caso com estimação global, para a arquitetura elaborada no presente trabalho. Embora este resultado não seja em princípio o esperado[4], a estimação local deve facilitar a busca evolutiva direcionando melhor o algoritmo genético para a seleção dos demais parâmetros do modelo (funções de pertinência, pólos, etc.). Em outras palavras, como a estimação local tende "separar" os modelos locais nas diferentes regiões de operação do sistema, o AG acaba tendo maior facilidade em otimizar os parâmetros desses modelos independentemente, fornecendo uma solução final com melhor capacidade de representação. As figuras das funções de pertinência e estados das premissas, para o modelo local, evidenciam a eficiência da abordagem. Os pólos encontrados modelam o sistema em duas regiões de operação distintas (como pré-suposto). O algoritmo genético encontra os valores adequados para os dois pólos, seleciona o estado para as premissas que identifica as duas regiões de operação e distribui as funções

[4]Na verdade, a estimação global deve em teoria fornecer um modelo com melhor capacidade de representação em relação aquele obtido com estimação local caso ambos os modelos sejam idênticos [2, 3, 51]. Porém este não é o caso aqui, uma vez que como os modelos estavam em populações distintas do AG possuíam estruturas distintas (número de estados, pólos, etc.).

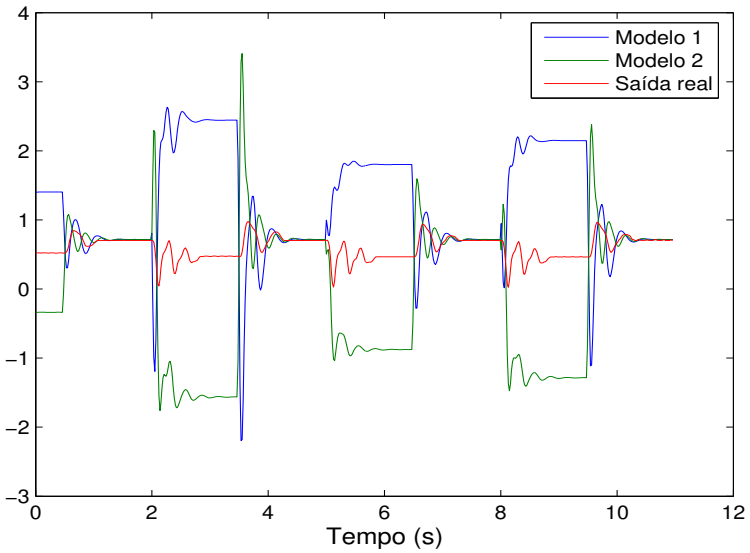

Figura 5.22: Modelos locais anti-simétricos para o sistema *fuzzy* TS FBO Generalizado com modelos locais não lineares e estimação global (levitador magnético).

de pertinência de entrada no universo de discurso de modo que cada modelo local atue eficazmente na tarefa de representação do comportamento do sistema.

Além de apresentar um resultado final melhor, a estimação local é mais vantajosa em termos de cálculos numéricos. A estimação dos parâmetros dos modelos locais é feita de forma separada, ou seja, são calculadas duas matrizes pseudo-inversas[5] de dimensão menor que aquela presente na estimação global dos parâmetros. As conseqüencias são, para a estimação local, um menor tempo de processamento para o cálculo das matrizes pseudo-inversas, as quais são ainda, em média, melhor condicionadas.

No algoritmo genético todas estas características acabam por favorecer a obtenção de melhores soluções (melhores modelos) ao se utilizar a estimação local.

5.3 Modelos para o Processo de Polimerização

Tomando-se como base os mesmos argumentos apresentados para o sistema levitador magnético, os dados de treinamento utilizados no sistema CSTR foram obtidos a partir de um sinal de entrada pseudo-aleatório tal qual o exibido na Figura 5.23.

A Figura 5.23 mostra que este sistema apresenta uma dinâmica menos complexa que a do sistema

[5]No caso geral, serão calculadas tantas matrizes pseudo-inversas quantos forem os modelos locais.

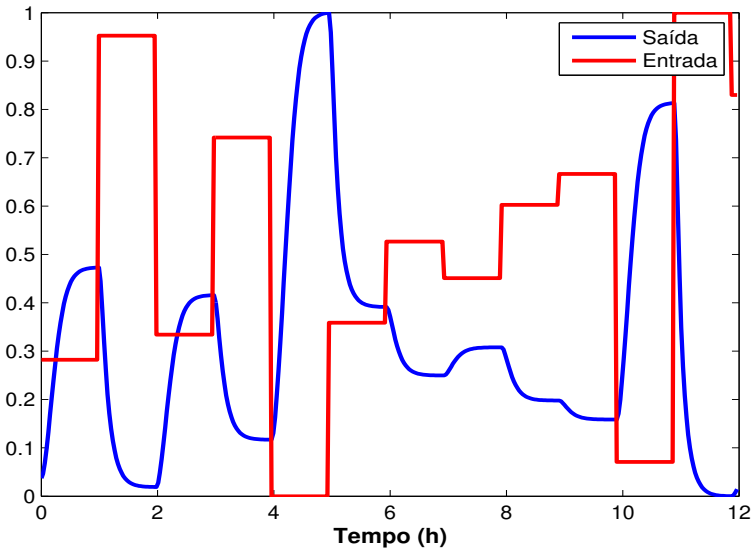

Figura 5.23: Sinais normalizados de entrada e saída para treinamento (CSTR).

levitador magnético, podendo inclusive ser aproximada por um filtro discreto de primeira ordem [14], exceto pelo ganho, que é altamente não linear. Esta é a razão para não se aplicar o modelo *fuzzy* TS FBO Generalizado na tarefa de modelagem deste processo de polimerização.

Diferentemente do sistema levitador magnético, observa-se que o sistema CSTR apresenta uma resposta temporal sobreamortecida, sem oscilações. Portanto, um pólo real é mais adequado para modelar esta dinâmica que um par de pólos complexos conjugados. Para se trabalhar com um pólo real na arquitetura *fuzzy* FS TBO com pólo único basta fazer com que o AG desconsidere a primeira posição da segunda linha da matriz que representa um indivíduo da população (ver Figura 4.3 da seção 4.4.1), posição essa correspondente à parte imaginária do pólo.

Novamente, os resultados apresentados foram obtidos após a execução do AG por um número de rodadas igual a 35. Para o sistema CSTR, a configuração do AG em todas as rodadas foi a mesma que para o levitador magnético, reproduzida aqui na tabela 5.14

A tabela 5.15 é análoga à tabela 5.2, resumindo os resultados obtidos após 35 rodadas do algoritmo genético. Observa-se que neste caso o algoritmo genético obteve soluções mais simples (modelos com um número menor de parâmetros) e mais rapidamente (menor tempo de processamento por rodada) se comparado ao sistema levitador magnético. Isto é decorrência do que foi afirmado anteriormente, isto é, o sistema CSTR é mais simples que o letivador magnético. Logo, existe uma dependência direta entre a complexidade do sistema sendo modelado e a complexidade e demanda computacional para a obtenção de seu modelo.

Para analisar a influência da modificação no operador de *crossover* aritmético, a tabela 5.16 exibe

Parâmetros de Execução	**TS FBO Pólo Único**
Tamanho da população	100
Número máximo de gerações	80
Taxa de *crossover*	0,85
Taxa de mutação	0,20
Expansão percentual máxima do operador de *crossover* (δ)	0,20
Percentual da diversidade inicial para convergência	3
Controle de sobreposição	não
Abertura mínima das FP gaussianas	0,3
Abertura máxima das FP gaussianas	3
Número máximo de estados nos conseqüentes das regras	7
Número máximo de variáveis de entrada (max_{vl})	4
Número máximo de FP por variável de entrada (max_{tl})	3
Limite inferior para aplicação das medidas de similaridade (ϵ)	0,2
Potência para uso na equação do limite de similaridade (ρ)	8

Tabela 5.14: Configuração do AG para o sistema CSTR.

um resumo dos resultados obtidos em um conjunto de simulações sem o parâmetro de expansão δ.

Observa-se que o uso do parâmetro de expansão no *crossover* aritmético permitiu que o algoritmo genético mantivesse a diversidade da população por um número maior de gerações, fornecendo ao final soluções melhores (comparar o AIC médio das tabelas 5.15 e 5.16).

A tabela 5.17 apresenta os resultados do melhor indivíduo de todas as rodadas, considerando-se novamente o menor EQM, para modelos locais lineares e não lineares.

A análise apenas do melhor indivíduo pode levar a conclusão errônea de que um modelo para este sistema necessita de um grande número de parâmetros. No entanto, assim como para o sistema levitador magnético, embora o melhor indivíduo de todas as rodadas de fato apresente uma grande quantidade de parâmetros, os indivíduos com EQM próximo à média da tabela 5.15 apresentam consideravelmente menos parâmetros. Mais ainda, dentre todas as rodadas é possível encontrar soluções com EQM menor que a média e ao mesmo tempo um número de parâmetros também inferior à média

	Linear	**Não Linear**
EQM médio	0,0009744	0,0005689
Número médio de parâmetros	48,11	60,69
AIC médio	-2779,64	-2857,39
Menor EQM	0,0000766	0,0000752
Número de parâmetros do menor EQM	78	138
AIC do menor EQM	-3491,90	-3398,80
Maior EQM	0,0054286	0,0026339
Número de parâmetros do maior EQM	22	78
AIC do maior EQM	-2032,20	-2130,30
Desvio padrão do EQM	0,0012133	0,0004744
Desvio padrão do AIC	339,18	262,45
Número médio de gerações	41,49	56,69
Tempo médio de processamento por rodada (min)	12,41	21,49

Tabela 5.15: Resumo dos resultados do modelo *fuzzy* TS FBO com pólo único (CSTR).

	Linear	**Não Linear**
EQM médio	0,0021007	0,0005885
Número médio de parâmetros	37,83	62,31
AIC médio	-2747,16	-2847,70
Número médio de gerações	38,34	44,17

Tabela 5.16: Resumo dos resultados do modelo *fuzzy* TS FBO com pólo único sem o parâmetro de expansão δ (CSTR).

de todas as soluções.

As duas próximas seções apresentam resultados de modelagem para o sistema CSTR que ilustram a afirmação anterior, para modelos locais lineares e não lineares, respectivamente.

5.3.1 Modelo *fuzzy* TS FBO com Pólo Único - Modelos Locais Lineares

Como exemplo, exibem-se na seqüência os resultados detalhados de uma rodada cujo EQM foi aproximadamente metade da média dos resultados apresentados na tabela 5.15, ou seja, da mesma ordem de grandeza dos resultados obtidos com modelos locais não lineares. Assim, considerando dois modelos com desempenhos semelhantes em termos de suas capacidades de representação, é possível classificar como "melhor" aquele que seja menos complexo (menor número de parâmetros).

Nesta rodada em particular, o AG consumiu 55 gerações, interrompendo sua execução pela convergência da população. O valor da diversidade ao final do processo evolutivo foi igual a 0,91 (a

	Linear	**Não Linear**
Erro quadrático médio	0,0000766	0,0000752
AIC	-3491,90	-3398,80
Número de regras	8	6
Número de parâmetros nos conseqüentes	8	21
Pólos	0,72843	0,74245
Número de parâmetros do modelo	78	138

Tabela 5.17: Melhor resultado do modelo *fuzzy* TS FBO com pólo único (CSTR).

diversidade inicial foi 31,55). A Figura 5.24 exibe a curva da diversidade da população ao longo das gerações.

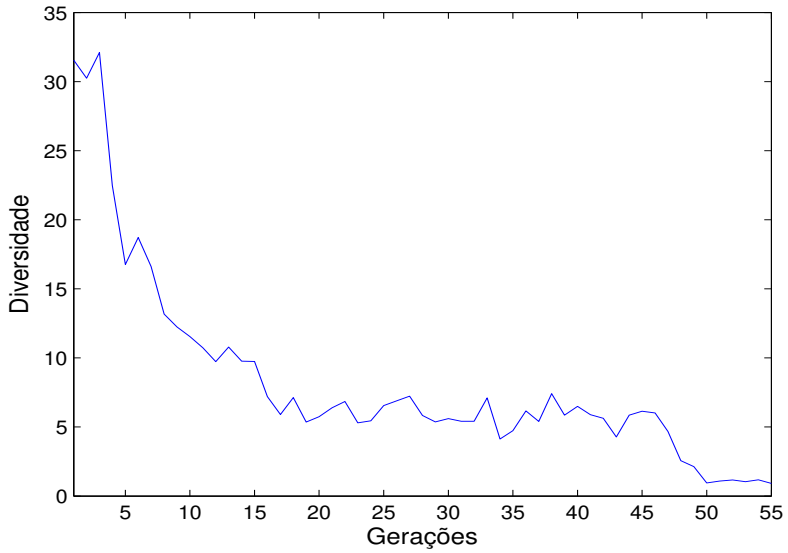

Figura 5.24: Diversidade da população para o modelo *fuzzy* TS FBO com pólo único e modelos locais lineares (CSTR).

O limite de similaridade calculado em função da diversidade da população está exibido na Figura 5.25. Nesta simulação, as fusões e eliminações de variáveis redundantes foram efetuadas com maior força nas dez últimas gerações.

Assim como a diversidade, a curva do critério de Akaike médio da população também tende a se estabilizar, como mostrado na Figura 5.26.

Ao final desta rodada, o AG forneceu a solução apresentada na tabela 5.18. Para este caso, o número de parâmetros do modelo foi 22, enquanto a média das soluções foi 48,11. O modelo obtido adotando-se a abordagem da distribuição homogênea das funções de pertinência (método sem

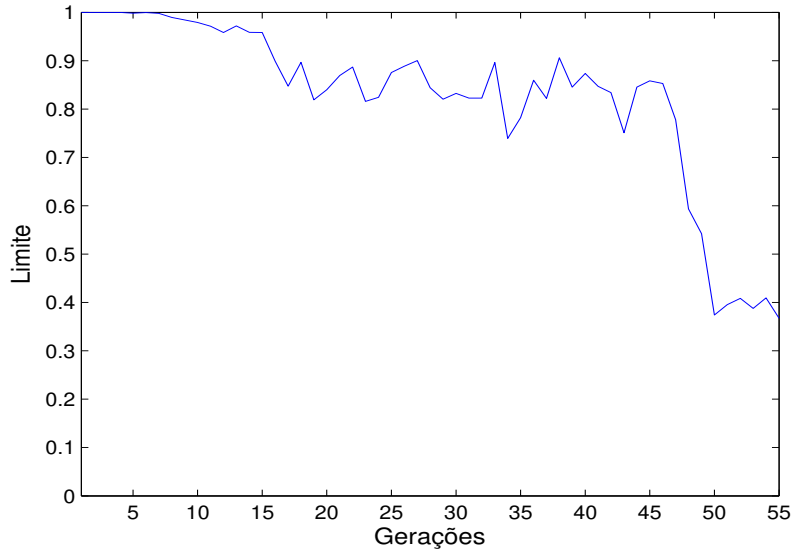

Figura 5.25: Limite de similaridade para o modelo *fuzzy* TS FBO com pólo único e modelos locais lineares (CSTR).

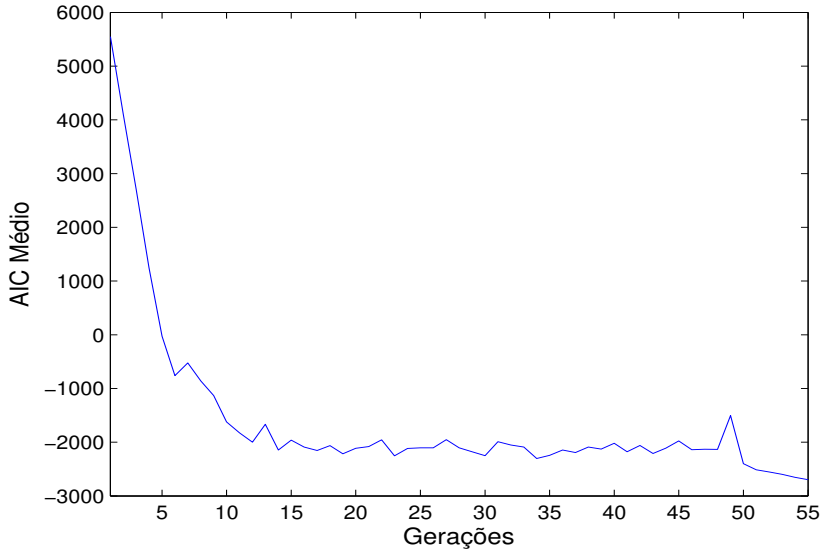

Figura 5.26: Evolução do AIC médio da população para o modelo *fuzzy* TS FBO com pólo único e modelos locais lineares (CSTR).

otimização) apresentou um EQM de 0,0019952, quase 300% maior que aquele obtido com o algoritmo genético proposto.

Pólo	0,91911
Número de regras	2
Número de parâmetros nos conseqüentes	8
EQM	0,0005068

Tabela 5.18: Solução fornecida pelo AG para o modelo *fuzzy* TS FBO com pólo único e modelos locais lineares (CSTR).

A Figura 5.27 exibe as funções de pertinência para os termos lingüísticos de entrada, completando a solução fornecida pelo AG. O algoritmo genético selecionou como variável de entrada o primeiro estado ortonormal.

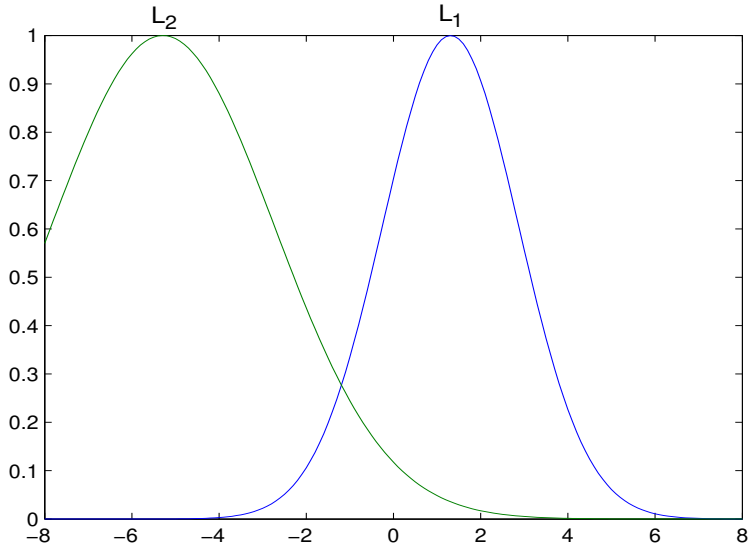

Figura 5.27: Funções de pertinência de entrada para o modelo *fuzzy* TS FBO com pólo único e modelos locais lineares (CSTR).

A Figura 5.28 exibe o resultado final obtido, comparando os dados de saída do sistema real com os fornecidos pelo modelo *fuzzy* TS FBO com pólo único e modelos locais lineares.

5.3.2 Modelo *fuzzy* TS FBO com Pólo Único - Modelos Locais Não Lineares

Para a solução ora apresentada, o AG não convergiu antes de executar o limite máximo de 80 gerações. Porém, como pode ser observado na Figura 5.29, a diversidade da população de fato se

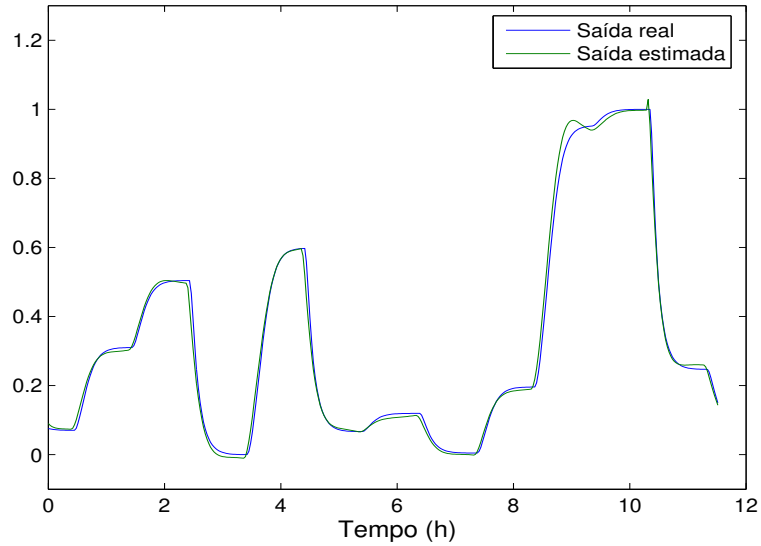

Figura 5.28: Saída real e a fornecida pelo modelo *fuzzy* TS FBO com pólo único e modelos locais lineares (CSTR).

estabilizou em torno de 2,5, apenas não alcançou um valor inferior a 3% da diversidade inicial, a qual foi igual a 31,68. Assim, a interrupção do AG, neste caso, não implicou em uma interrupção do processo evolutivo, pois a população não apresentava melhorias significativas nas últimas gerações.

O limite para atuação das medidas de similaridade teve um comportamento semelhante ao da diversidade, não variando consideravelmente quando esta última também se estabilizou. A Figura 5.30 exibe o comportamento desse limite. Como a curva apresentada não se aproximou do limite mínimo estipulado na tabela 5.14, o AG permitiu a existência de funções de pertinência mais sobrepostas que no caso anterior, como será observado na Figura 5.32.

O valor do critério de Akaike médio da população nesta rodada apresentou uma dinâmica um tanto ruidosa, porém seguindo o objetivo de minimização. Atribui-se este comportamento ao fato de a população manter um certo nível de diversidade acima do limite mínimo pré-estabelecido. Esta diversidade permite a contínua renovação de uma parcela da população, através da atuação dos operadores genéticos. Dessa forma, ora a média do critério de Akaike pode aumentar e ora diminuir, oscilando em torno de um nível que pode ser tido como o ótimo para essa rodada em particular. O gráfico da Figura 5.31 exibe tal comportamento para o critério de Akaike médio da população.

Após as 80 gerações, o AG foi interrompido e forneceu a solução apresentada na tabela 5.19, a qual codifica um modelo com um total de 26 parâmetros. Para este modelo em particular a solução fornecida pelo AG aproxima-se daquela obtida quando da distribuição homogênea das funções de pertinência no universo de entrada. Sendo assim, neste caso ambas as estratégias forneceram modelos

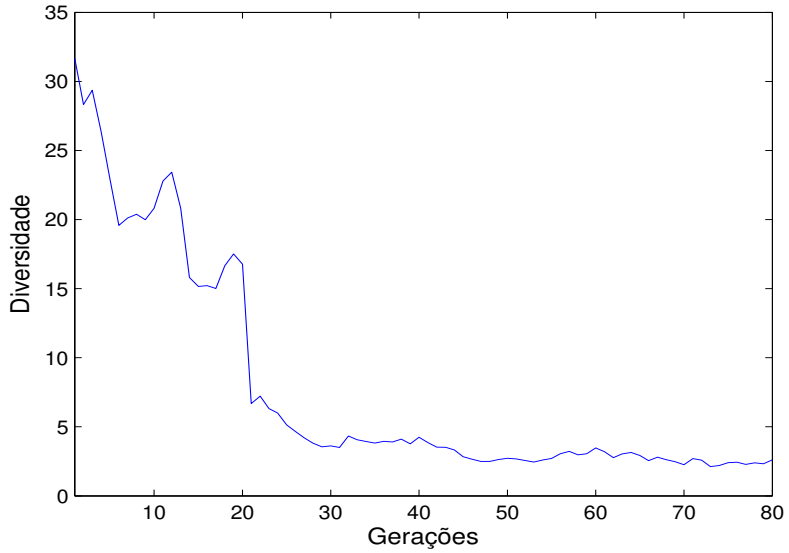

Figura 5.29: Diversidade da população para o modelo *fuzzy* TS FBO com pólo único e modelos locais não lineares (CSTR).

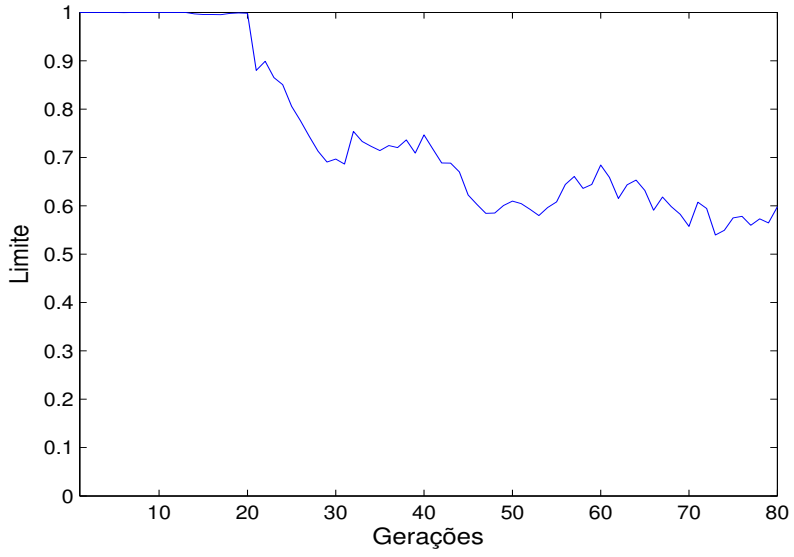

Figura 5.30: Limite de similaridade para o modelo *fuzzy* TS FBO com pólo único e modelos locais não lineares (CSTR).

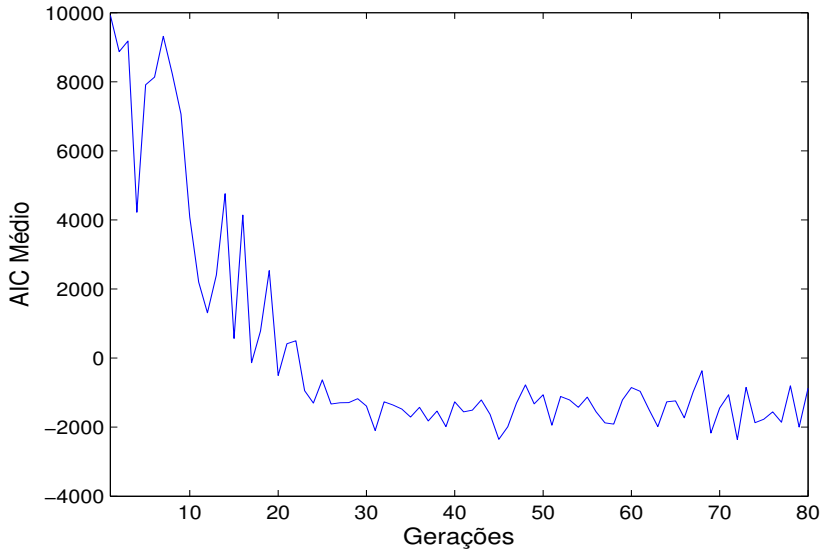

Figura 5.31: Evolução do AIC médio da população para o modelo *fuzzy* TS FBO com pólo único e modelos locais não lineares (CSTR).

equivalentes. Vale ressaltar, no entanto, que este é um caso especial, uma vez que em todas as simulações apresentadas anteriormente, tanto para o CSTR quanto para o levitador magnético, nas duas arquiteturas analisadas (pólo único e Generalizada), os resultados obtidos com o método automático baseado em algoritmos genéticos aqui proposto foram sensivelmente melhores.

Pólo	0.76593
Número de regras	2
Número de parâmetros nos conseqüentes	10
EQM	0.0004894

Tabela 5.19: Solução fornecida pelo AG para o modelo *fuzzy* TS FBO com pólo único e modelos locais não lineares (CSTR).

A Figura 5.32 exibe as funções de pertinência da solução final. Assim como no caso de modelos locais lineares, aqui o algoritmo genético selecionou como variável de entrada o primeiro estado ortonormal. Note-se que as funções de pertinência estão mais sobrepostas que as apresentadas na Figura 5.27.

A Figura 5.33 apresenta os dados de saída do sistema real e os fornecidos pelo modelo *fuzzy* TS FBO com pólo único e modelos locais não lineares. Este resultado é ligeiramente melhor que o obtido com modelos locais lineares, porém, como afirmado anteriormente, este modelo possui um total de 26 parâmetros, contra 22 daquele com modelos locais lineares. Analisando de outra forma, o modelo

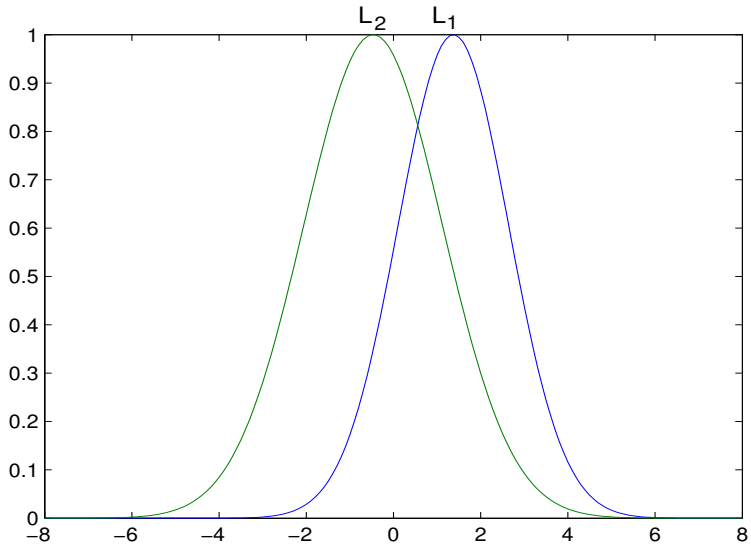

Figura 5.32: Funções de pertinência de entrada para o modelo *fuzzy* TS FBO com pólo único e modelos locais não lineares (CSTR).

atual apresenta um EQM 4% menor que aquele com modelos locais lineares porém com 15% a mais de parâmetros. Logo, este acréscimo de parâmetros não é justificado, sendo então mais adequado o modelo *fuzzy* TS FBO com pólo único e modelos locais lineares para a modelagem do sistema CSTR.

5.3.3 Comentários sobre os Modelos do Processo de Polimerização

Os dois modelos obtidos com o AG, apresentados nas seções anteriores, confirmam a corretude da escolha apenas da arquitetura *fuzzy* TS FBO com pólo único para a modelagem do processo de polimerização CSTR, uma vez que os resultados apresentados demonstram que tais modelos foram capazes de aproximar com grande precisão o sistema real. Em ambos os casos, foram obtidos modelos com EQM da ordem de 5.10^{-4} com um número reduzido de parâmetros.

Ao se comparar as duas opções com modelos locais lineares e não lineares, conforme argumentos da última seção, modelos locais lineares se mostram mais interessantes para o CSTR. Por se tratar de um sistema relativamente simples, o modelo *fuzzy* TS FBO com pólo único e modelos locais lineares nos conseqüentes das regras encontra o melhor compromisso entre capacidade de representação e complexidade do modelo. Além disso, esta arquitetura demanda em média menos processamento computacional até a obtenção de uma solução (ver tabela 5.15).

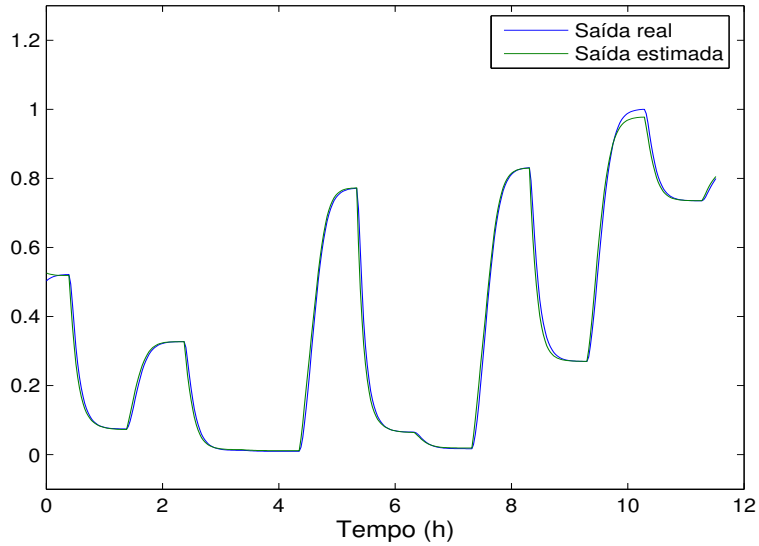

Figura 5.33: Saída real e a fornecida pelo modelo *fuzzy* TS FBO com pólo único e modelos locais não lineares (CSTR).

5.4 Resumo

Este capítulo apresentou os resultados de modelagem para dois sistemas dinâmicos não lineares, um levitador magnético e um processo de polimerização. Exemplificaram-se todas as arquiteturas propostas no presente trabalho, quais sejam, modelo *fuzzy* TS FBO com pólo único, *fuzzy* TS FBO Generalizado com estimação local e *fuzzy* TS FBO Generalizado com estimação global. Para estas arquiteturas, utilizaram-se tanto modelos locais lineares quanto não lineares nos conseqüentes das regras. Por sua menor complexidade, analisou-se para o processo de polimerização apenas o modelo *fuzzy* TS FBO com pólo único.

Os resultados de modelagem obtidos para ambos os sistemas indicam a eficiência do método proposto. Um conjunto de simulações utilizando a abordagem não automática de distribuição homogênea das funções de pertinência no universo de discurso forneceu a base comparativa para a conclusão sobre a eficiência do método baseado em algoritmos genéticos. Outro conjunto de simulações indicou a inferioridade dos resultados ao se utilizar o operador de *crossover* aritmético original, sem o paraâmetro de expansão.

Em particular, para o processo de polimerização, o modelo *fuzzy* TS FBO com modelos locais lineares forneceu os melhores resultados. Para o levitador magnético ambas as arquiteturas tiverem desempenhos compatíveis em termos da minimização do critério de Akaike (base do cálculo do *fitness* das soluções), sendo que o modelo com pólo único apresenta em média valores menores do erro

quadrático médio, porém com um número maior de parâmetros. Além disso, os modelos obtidos com essa arquitetura apresentavam em média um número maior de funções ortonormais (estados), capazes de representar bem qualquer dos dois modos dominantes do sistema (que não são muito distintos) com um único par intermediário de pólos complexos conjugados. A arquitetura com pólo único também demandou maior tempo de processamento computacional, o que pode ser explicado pelo fato de sua representação cromossômica ser muito mais flexível.

Considerando-se apenas a arquitetura Generalizada para o levitador magnético, o método de estimação que forneceu melhor desempenho foi a estimação local dos parâmetros dos conseqüentes das regras. Este resultado merece atenção especial pois concilia os dois objetivos em princípio conflitantes quando da estimação de parâmetros. Como afirmado na seção 3.4, a estimação local permite uma interpretação local do modelo, sendo por exemplo útil nas etapas de análise e validação dos modelos (ferramenta essencial neste capítulo). Naquela seção foi também afirmado que em geral é a estimação global que fornece a melhor aproximação do sistema, em termos de um EQM menor. No entanto, para a arquitetura proposta, o algoritmo genético forneceu modelos ao mesmo tempo interpretáveis localmente e com capacidade de representação do sistema melhor que a obtida com estimação global dos parâmetros dos conseqëntes das regras (resultado semelhante foi obtido em [51]).

Outra observação interessante foi a obtenção de melhores resultados com modelos locais lineares nos conseqüentes das regras, para os dois sistemas analisados. Conjectura-se que modelos locais não lineares (Volterra de segunda ordem) produzem modelos com um maior número de parâmetros sem no entanto introduzir sensíveis melhorias na capacidade de representação devido aos termos não lineares, sofrendo assim maior penalização pelo critério de Akaike.

Capítulo 6

Conclusões

De concepção recente, os sistemas *fuzzy* TS FBO vêm demonstrando sua eficácia na modelagem e controle de sistemas dinâmicos não lineares [89, 13, 14, 16]. Os modelos obtidos com essa arquitetura apresentam uma série de vantagens se comparados a outras possíveis abordagens (ver seção 3.1).

As arquiteturas *fuzzy* TS FBO e *fuzzy* TS FBO Generalizado (apresentadas em detalhe no capítulo 3) são baseadas nos modelos em espaço de estados, linear FBO, de Volterra e *fuzzy* TS descritos no capítulo 2. Em suma, os modelos baseados em funções de base ortonormal possuem as desejáveis características de ausência de realimentação da saída (e eventuais erros de predição), necessidade de um menor número de parâmetros para alcançar uma dada precisão (através da incorporação de conhecimento sobre a dinâmica do sistema nos pólos que definem as funções da base ortonormal) e tolerância a dinâmicas não modeladas ou a pequenas diferenças na ordem dos vetores de regressão. Modelos *fuzzy* TS baseados em funções de base ortonormais, além das características acima, agregam as propriedades de interpretabilidade [32] e facilidade de representação do conhecimento dos sistemas *fuzzy*. No entanto, os resultados promissores já obtidos com essa arquitetura não utilizavam nenhum método computacional inteligente para sua otimização automática.

O objetivo da presente dissertação foi indicar uma possível direção para a realização desta última tarefa, ou mais especificamente, para a definição e otimização da estrutura de modelos *fuzzy* TS FBO e *fuzzy* TS FBO Generalizado.

A solução adotada foi um algoritmo genético, método de otimização bem consolidado na literatura acadêmica e com uma vasta gama de aplicações práticas. Foi possível obter modelos de sistemas dinâmicos não lineares (um processo de polimerização e um levitador magnético) com grande grau de autonomia ao se utilizar a metodologia proposta neste trabalho, confirmando sua eficiência.

No capítulo 3 foram explicitados os parâmetros dos modelos *fuzzy* TS FBO otimizados pelo AG descrito no capítulo 4. A otimização de alguns parâmetros dos modelos *fuzzy* TS já foi abordada em outros trabalhos na literatura acadêmica (ver referências da seção 4.3.1), incluindo a forma e

localização das funções de pertinência [76] e a base de regras de inferência. Trabalhos mais recentes também determinavam de forma automática quantas e quais variáveis de entrada seriam usadas no sistema, bem como quantas funções de pertinência estariam associadas a cada uma delas [26, 94].

As contribuições deste trabalho no contexto acima são:

- Elaboração de uma representação cromossômica para o modelo *fuzzy* TS FBO com pólo único e outra para o modelo *fuzzy* TS FBO Generalizado;
- Inclusão de um coeficiente de expansão no operador de *crossover* aritmético para a manutenção da diversidade da população e obtenção de melhores soluções;
- Uso de um operador de mutação gaussiana com desvio padrão variável e distinto para cada gene;
- Para a arquitetura *fuzzy* TS FBO com pólo único, otimiza-se:
  - O pólo complexo que caracteriza a base de funções ortonormais;
  - O número de variáveis de entrada do modelo;
  - A quantidade de funções de pertinência para cada variável de entrada;
  - Número de funções na base ortonormal;
  - Além desses parâmetros, ainda inclui-se uma etapa de simplificação das funções de pertinência através das medidas de similaridade;
- Para a arquitetura *fuzzy* TS FBO Generalizado, otimiza-se:
  - Um pólo complexo para cada modelo local;
  - Número de funções em cada base ortonormal;
  - Quais funções da base ortonormal estarão presentes nas premissas das regras (variáveis de entrada);
- A disposição e configuração das funções de pertinência em ambas as arquiteturas também foi otimizada.

A arquitetura *fuzzy* TS FBO Generalizado, por sua maior complexidade, teve menos parâmetros otimizados. No entanto, mesmo nesse caso, o grau de autonomia na definição da estrutura do modelo era alto, bastando basicamente ser definido por um especialista o número de modelos locais esperado para o sistema. Ainda para o modelo Generalizado, foram analisadas duas formas de estimação dos coeficientes das regras: estimação local e global. Verificou-se para o sistema levitador magnético que

a estimação local seria mais adequada, uma vez que ao mesmo tempo permite uma interpretação clara do modelo, em termos da correta classificação das diferentes regiões de operação do sistema, além de possibilitar uma melhor representação do mesmo (menor erro de aproximação em série sintética) com um modelo menos complexo (menor número de parâmetros). Este resultado é particularmente interessante pois em princípio a estimação global dos parâmetros deveria fornecer um modelo com melhor capacidade de representação, mesmo a custa de uma perda em sua interpretabilidade (ver seção 5.2.4). Para ambos os sistemas estudados o uso de modelos locais lineares apresentou melhores resultados em comparação aos modelos locais não lineares de Volterra de segunda ordem. Logo, para as arquiteturas propostas neste trabalho, os termos não lineares nos modelos locais são mais penalizados pelo critério de Akaike sem resultar em melhorias significativas na capacidade de representação do modelo como um todo.

Estas conclusões foram obtidas com a análise das simulações documentadas no capítulo 5. Todas as possíveis combinações das arquiteturas propostas foram avaliadas de forma estatística, apresentando-se não os melhores resultados obtidos (que poderiam de certa forma "polarizar" sua interpretação), mas sim aqueles que se aproximaram da média de todas as simulações. Em todos os casos, a corretude das implementações computacionais foi comprovada pela verificação do comportamento esperado do algoritmo genético (como a queda da diversidade da população bem como do valor médio do critério de Akaike com o passar das gerações) e pela obtenção de modelos satisfatórios para os sistemas estudados, resultado este que confirma ainda a eficiência da metodologia aqui proposta. A qualidade dos modelos foi avaliada tanto pela forma e disposição das funções de pertinência obtidas quanto pela comparação das respostas (saídas) fornecidas pelos modelos com aquelas do sistema real, ambos sujeitos a um mesmo sinal de entrada (diferente do utilizado na etapa de otimização). Além disso, foram realizadas várias simulações para comparação de resultados com estratégias sem otimização automática (distribuição homogênea de funções de pertinância) e com o operador de *crossover* aritmético original. Todos os resultados comparativos, também relatados no capítulo 5, indicam a relevância do presente trabalho.

A principal direção para continuidade deste trabalho é a investigação de um método automático para a definição do número de modelos locais na arquitetura *fuzzy* TS FBO Generalizado. Atualmente esta é uma entrada do algoritmo, porém métodos de agrupamento baseados em um conjunto de dados de treinamento, por exemplo, podem ser aplicados nessa tarefa [81]. Outra possibilidade seria a elaboração de uma representação cromossômica flexível, que permitisse o aumento e diminuição do número de modelos locais, alterando o conjunto de funções de pertinência de tal forma que a base de regras completa fosse composta por tantas regras quantos fossem os modelos locais. Seria possível também investigar o uso de base de regras não completas.

Ainda para o modelo *fuzzy* TS FBO Generalizado, podem ser implementados ao mesmo tempo

modelos locais lineares e não lineares, modelando diferentes regiões de operação do sistema real [32]. Logo, caso o sistema real possua regiões de operação distintas, algumas podendo ser modeladas por modelos lineares e as demais por não lineares, então tal implementação da arquitetura *fuzzy* TS FBO Generalizado seria adequada, uma vez que os modelos *fuzzy* permitem a transição suave entre seus modelos locais [93].

Além dessas possíveis continuações na pesquisa da modelagem de sistemas dinâmicos utilizando a arquitetura proposta, outra perspectiva se refere à sua aplicação em controladores baseados em modelos, os controladores preditivos [1, 12, 32, 67, 81, 109].

Referências Bibliográficas

[1] Abonyi, J. (2003). Fuzzy Model Identification for Control. Birkhäuser.

[2] Abonyi, J. e Babuska, R. (2000). Local and global identification and interpretation of parameters in Takagi-Sugeno fuzzy models. Ninth IEEE International Conference on Fuzzy Systems, San Antonio, TX, USA, Vol. 2, 835-840.

[3] Aguirre, L. A. (2004). Introdução a Identificação de Sistemas, 2º Edição, Editora UFMG.

[4] Anderson, J. A. (1995). An introduction to neural networks. MIT Press.

[5] Amaral, J. F. M., Vellasco, M. M., Tanscheit, R. e Pacheco, M. A. C. (2001). A Neuro-Fuzzy-Genetic System for Automatic Setting of Control Strategies. Genetic Fuzzy Systems: New Developments, IFSA/NAFIPS Conference, Vancouver, Canada, 1553-1558.

[6] Bäck, T., Fogel, D. B. e Michalewicz, Z. (2000). Evolutionary Computation 1, Basic Algorithms and Operators. Institute of Physics Publishing, Bristol and Philadelphia.

[7] Bäck, T., Fogel, D. B. e Michalewicz, Z. (2000). Evolutionary Computation 2, Advanced Algorithms and Operators. Institute of Physics Publishing, Bristol and Philadelphia.

[8] Blanton, J. L. Jr. e Wainwright, R. L. (1993) Multiple Vehicle Routing with Time and Capacity Constraints Using Genetic Algorithms. Proceedings of the 5th International Conference on Genetic Algorithms, San Francisco, CA, USA, 452-459.

[9] Bodin, P., Silva, T. O. E. e Wahlberg, B. (1996). On the construction of orthonormal basis functions for system identification. Proceedings of the IFAC World Congress, San Francisco, CA, USA, Vol. 1, 369-374.

[10] Braga, A. P., Ludermir, T. B. e Carvalho, A. C. P. L. F. (2000). Redes Neurais Artificiais. LTC - Livros Técnicos e Científicos Editora.

[11] Brassard, G. e Bratley, P. (1996) Fundamentals of algorithmics. Prentice-Hall.

[12] Camacho, E. F. e Bordons, C. (2004). Model Predictive Control, 2nd Edition. Springer.

[13] Campello, R. J. G. B., Meleiro, L. A. C., Amaral, W. C. e Maciel Filho, R. (2001). Identification of a bioprocess using Laguerre function based models. Proceedings of the Sixth World Congress of Chemical Engineering, Melbourne, Australia, p. CD.

[14] Campello, R. J. G. B. (2002). Arquiteturas e Metodologias para Modelagem e Controle de Sistemas Complexos Utilizando Ferramentas Clássicas e Modernas. Tese de Doutorado, DCA/FEEC/UNICAMP, Campinas, SP, Brasil.

[15] Campello, R. J. G. B. e Amaral, W. C. (2002). Takagi-Sugeno Fuzzy Models within Orthonormal Basis Function Framework and their Application to Process Control. Proceedings of the 11th IEEE Int. Conference on Fuzzy Systems, 1399-1404.

[16] Campello, R. J. G. B., Meleiro, L. A. C. e Amaral, W. C. (2004). Control of a Bioprocess using Orthonormal Basis Function Fuzzy Models. Proceedings of the IEEE International Conference on Fuzzy Systems, Budapest, Hungary, Vol. 2, 801-806.

[17] Cordón, O. (2001). Genetic fuzzy systems : evolutionary tuning and learning of fuzzy knowledge bases. World Scientific.

[18] Cordón, O., Gomide, F., Herrera, F., Hoffmann, F. e Magdalena, L. (2004). Ten years of genetic fuzzy systems: current framework and new trends. Fuzzy Set and Systems, Vol. 141, 5-31.

[19] Cybenko, G. (1988). Continuous Valued Neural Network with two Hidden Layers are Sufficient. Technical Report, Department of Computer Science, Tufts University.

[20] Cybenko, G. (1989). Approximation by Superpositons of a Sigmoid Function. Mathematics of Control, Signals and Systems, Vol. 2, 303-314.

[21] Darwin, C. (1859). The origin of species by means of natural selection, or the preservation of favoured races in the struggle for life. John Murray, London, United Kingdom.

[22] Delgado, M. R., Von Zuben, F. e Gomide, F. (2001). Hierarchical genetic fuzzy systems. Information Sciences, Vol. 136, 29-52.

[23] Delgado, M. R., Von Zuben, F. e Gomide, F. (2001). Local and global estimation of Takagi-Sugeno consequent parameters in genetic fuzzy systems. Joint 9th IFSA World Congress and 20th NAFIPS International Conference, Vancouver, BC, Canada, Vol. 3, 1247-1252.

[24] Delgado, M. R. B. S. (2002). Projeto Automático de Sistemas Nebulosos: Uma Abordagem Co-Evolutiva. Tese de Doutorado, DCA/FEEC /UNICAMP, Campinas, SP, Brasil.

[25] Delgado, M. R., Von Zuben, F. e Gomide, F. (2002). Multi-Objective Decision Making: Towards Improvement of Accuracy, Interpretability and Design Autonomy in Hierarchical Genetic Fuzzy Systems. Proceedings of the IEEE International Conference on Fuzzy Systems, Honolulu, HI, USA. Vol. 2, 1222-1227.

[26] Delgado, M. R., Von Zuben, F. e Gomide, F. (2004). Coevolutionary genetic fuzzy systems: a hierarchical collaborative approach. Fuzzy Sets and Systems, Vol. 141, Issue 1, 89-106.

[27] Dorf, R. C. e Bishop, R.H. (1995). Modern Control Systems, 7th Edition. Addison-Wesley.

[28] Doyle III, F. J., Pearson, R. K. e Ogunnaike, B. A. (2002). Identification and Control using Volterra models. Springer.

[29] DuBois, D. e Prade. H. (2000). Fundamentals of Fuzzy Sets. Kluwer Academic Publishers.

[30] Erden, M. S., Leblebicioglu, K. e Halici, U. (2004). Multi-Agent Systems-Based Fuzzy Controller Design with Genetic Tuning for a Mobile Manipulator Robot in the Hand Over Task. Journal of Intelligent and Robotic Systems. Vol. 39, Issue 3, 287-306

[31] Espíndola, R. P. (1999). Otimização de Regras Fuzzy de Classificação por Algoritmos Genéticos. Dissertação de Mestrado, COPPE/PEC.

[32] Espinosa, J., Vandewalle, J. e Wertz, V. (2004). Fuzzy Logic, Identification and Predictive Control. Springer-Verlag.

[33] Eykoff, P. (1974). System Identification, Parameter and State Estimation. John Willey & Sons.

[34] Fogel, D. B. e Atmar, J. W. (1990). Comparing genetic operators with gaussian mutations in simulated evolutionary processes using linear systems, Formal Aspects of Computing (Historical Archive), Vol. 63, Issue 2, 111-114.

[35] Fogel, D. B., Fogel, L. J. e Atmar, J. W. (1991). Meta-evolutionary programming. Twenty-Fifth Conference on Signals, Systems and Computers, Pacific Grove, CA, USA, Vol. 1, 540-545.

[36] Goldbarg, M. C. e Luna, H. P. (2000). Otimização Combinatória e Programação Linear. Editora Campus.

[37] Goodwin, G. C., Graebe, S. F. e Salgado, M. E. (2001). Control System Design. Prentice Hall.

[38] Goonatilake, S. e Khebbal, S. (1995). Intelligent Hybrid Systems. John Wiley & Sons.

[39] Güven, M. K. e Passino, K. M. (2001). Avoiding exponential parameter growth in fuzzy systems, IEEE Transactions on Fuzzy Systems, Vol. 9, 194-199.

[40] Haykin, S. (1999). Neural Networks: A Comprehensive Foundation, 2nd Edition. Prentice Hall.

[41] Haykin, S. e Veen, B. V. (2000). Sinais e Sistemas. Bookman.

[42] He, L. e Mort, N. (2000). Hybrid Genetic Algorithms for Telecommunications Network Back-Up Routeing. BT Technology Journal, Vol. 18, Issue 4, 42-50.

[43] Herrera, F. e Lozano, M. (1996). Adaptation of Genetic Algorithm Parameters Based on Fuzzy Logic Controllers. Genetic Algorithms and Soft Computing, Physica-Verlag.

[44] Hinterding, R. (1996). Gaussian mutation and self-adaption for numeric genetic algorithms. IEEE International Conference on Evolutionary Computation, Perth, Australia, Vol.1, 384-389.

[45] Holland, J. H. (1962). Outline for a logical theory of adaptive systems. Journal of ACM 9 297-314.

[46] Holland, J. H. (1975). Adaptation in Natural and Artificial Systems. Ann Arbor, MI: University of Michigan Press.

[47] Homaifar, A. e McCormick, E. (1995). Simultaneous Design of Membership Functions and Rule Sets for Fuzzy Controllers Using Genetic Algorithms. IEEE Transactions on Fuzzy Systems, Vol. 3, Issue 2, 129-139.

[48] Jain, L. C. e Jain, R. K. (1997). Advances in Fuzzy Systems - Applications and Theory. Hybrid Intelligent Engineering Systems, Vol. 11, World Scientific.

[49] Jain, L. C. e Martin, N. M. (1998). Fusion of Neural Networks, Fuzzy Sets, and Genetic Algorithms. CRC Press.

[50] Jikai, Y., Hongtao, Y. H. S. e Yuanbin, H. (1997). Fuzzy Control Technique Based on Genetic Algorithms Optimizing and Its Applications. IEEE International Conference on Intelligent Processing Systems, Beijing, China, Vol. 1, 329-333.

[51] Johansen, T. A., Shorten, R. e Murray-Smith, R. (2000). On the interpretation and identification of dynamic Takagi-Sugeno fuzzy models. IEEE Transactions on Fuzzy Systems, Vol. 8, Issue 3, 297-313.

[52] Cavalcante Junior, F. L. (2000). Controle Preditivo Utilizando um Modelo Nebuloso. Dissertação de mestrado, DCA/FEEC/UNICAMP, Campinas, SP, Brasil.

[53] Kailath, T. (1980). Linear Systems. Prentice-Hall.

[54] Karr, C. e Freeman, L. M. (1998) Industrial Applications of Genetic Algorithms. CRC Press.

[55] Kibangou, A.Y., Favier, G. e Hassani, M. M. (2003).Generalized orthonormal basis selection for expanding quadratic Volterra filters. 13th IFAC Symposium on System Identification, The Netherlands, 1119-1124.

[56] Kim, J. e Zeigler, B. P. (1996). Hierarchical Distributed Genetic Algorithms: A Fuzzy Logic Controller Design Application. IEEE Intelligent Systems, Vol. 11, Issue 3, 76-84.

[57] Kumar, K. e Wu, B. (1999). Application of Genetic and Fuzzy Modelling in Time Series Analysis. Proceedings of the Third IEEE International Conference on Computational Intelligence and Multimedia Applications, New Delhi, India, 128-132.

[58] Kumar, P., Chandna, V. K. e Thomas, M. S. (2004). Fuzzy-Genetic Algorithm for Pre-Processing Data at the RTU. IEEE Transactions on Power Systems, Vol. 19, Issue 2, 718-723.

[59] Kikuchi, S., Tominaga, D., Arita, M., Takahashi, K. e Tomita, M. (2003). Dynamic modeling of genetic networks using genetic algorithm ans S-system. Bioinformatics, Vol. 19, 643-650.

[60] Koza, J. R., Bennett III, F. H., Andre, D., Keane, M. A. e Dunlap, F. (1997). Automated Synthesis of Analog Electrical Circuits by Means of Genetic Programming. IEEE Transactions on Evolutionary Computation, Vol. 1, N. 2, 109-128.

[61] Kosko, B. (1997). Fuzzy Engineering, Prentice Hall.

[62] Laabidi, K. e Bouani, F. (2004). Genetic Algorithms for Multiobjective Predictive Control. First International Symposium on Control, Communications and Signal Processing, 149-152.

[63] Larrañaga, P., Kuijpers, C. M. H., Murga, R. H., Inza, I. e Dizdarevic, S. (1999). Genetic Algorithms for the Travelling Salesman Problem: A Review of Representations and Operators. Artificial Intelligence Review, Vol. 13, Issue 2, 129-170.

[64] Lee, M. A. e Takagi, H. (1993). Dynamic Control of Genetic Algorithms using Fuzzy Logic Techniques. Proceedings of the 5th International Conference on Genetic Algorithms, San Francisco, CA, USA, 76-83.

[65] Lee, M. A. e Takagi, H. (1993). Integrating Design Stages of Fuzzy Systems using Genetic Algorithms. Second IEEE International Conference on Fuzzy Systems, San Francisco, CA, USA, Vol. 1, 612-617.

[66] Leung, F. H. F., Lam, H. K., Ling, S. H. e Tam, P. K. S. (2004). Optimal and Stable Fuzzy Controllers for Nonlinear Systems Based on an Improved Genetic Algorithm. IEEE Transactions on Industrial Electronics, Vol. 51, N. 1, 172-182.

[67] Li, N., Li, S. Y. e Xi, Y. G. (2004). Multi-model predictive control based on the Takegi-Sugeno fuzzy models: a case study. Information Sciences, Vol. 165, 247-263.

[68] Lima, C. M. R. R (2002). Otimização da Frota de Veículos na Distribuição de Gás. Monografia de Graduação, Universidade Federal do Rio Grande do Norte, Natal, RN.

[69] Lima, R. N. B., Miranda, M. N., Filho, J. V. S., Pedroza, A. C. P. e Mesquita, A. C. (2002). HW/SW codesign of handoff protocol for wireless ATM networks based on performance optimization using genetic algorithm. Proceedings. 15th Symposium on Integrated Circuits and Systems Design, 29-34.

[70] Linkens, D. A. e Nyongesa, H. O. (1996). Learning Systems in intelligent control: an appraisal of fuzzy, neural and genetic algorithm control applications. Proceedings of the IEEE Control Theory and Applications, Vol. 143, Issue 4, 367-386.

[71] Liska, J. e Melsheimer, S. S. (1994). Complete Design Of Fuzzy Logic Systems Using Genetic Algorithms. IEEE World Congress on Computational Intelligence, Proceedings of the Third IEEE Conference on Fuzzy Systems, Vol. 2, 1377-1382.

[72] Ljung, L. (1999) System identification: Theory for the user, Z Edition. Prentice-Hall.

[73] Mamdani, E. H. (1976). Application of fuzzy logic to approximate reasoning using linguistic synthesis. Proceedings of the Sixth IEEE International Symposium on Multiple-valued Logic, Logan, Utah, USA, 196-202.

[74] Maner, B. R., Doyle III, F. J., Ogunnaike, B. A. e Pearson, R. K. (1996). Nonlinear Model predictive control of a simulated multivariable polymerization reactor using second-order Volterra models. Automatica, Vol. 32, Issue 9, 1285-1301.

[75] McCulloch, W. S. e Pitts, W. H. (1943). A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics, Vol. 5, 115-133.

[76] Medeiros, A. V., Maitelli, A. L. e Gabriel Filho, O. (2001). Otimização das Funções de Pertinência de um Controlador Nebuloso utilizando Algoritmos Genéticos. V Simpósio Brasileiro de Automação Inteligente, Canela, RS.

[77] Medeiros, A. V. (2003). Utilização de Técnicas de Inteligência Artificial no Controle de Sistemas. Monografia de Graduação, Universidade Federal do Rio Grande do Norte, Natal, RN.

[78] Medeiros, A. V., Maitelli, A. L. e Araújo, F. M. U. (2003). Geração das regras de inferência de um Controlador Nebuloso utilizando Algoritmos Genéticos. VI Simpósio Brasileiro de Automação Inteligente, Bauru, SP.

[79] Michalewicz, Z. (1996). Genetic Algorithms + Data Structures = Evolution Programs. 3º Edição, Springer-Verlag.

[80] Michalewicz, Z., Nazhiyath, G., e Michalewicz, M. (1996). A Note on Usefulness of Geometrical Crossover for Numerical Optimization Problems, Proceedings of the 5th Annual Conference on Evolutionary Programming, San Diego, CA. MIT Press, Cambridge, MA, 305-312.

[81] Murray-Smith, R. e Johansen, T. A. (1997). Multiple Model Approaches to Modelling and Control. Taylor and Francis.

[82] Ng, K. C. e Li, Y. (1994). Design Of Sophisticated Fuzzy Logic Controllers Using Genetic Algorithms. IEEE World Congress on Computational Intelligence, Proceedings of the Third IEEE Conference on Fuzzy Systems, Orlando, FL, USA, Vol. 3, 1708-1712.

[83] Ninness, B., Gómez, J.-C. e Weller, S. (1995). MIMO System Identification Using Orthonormal Basis Functions. Proceedings of the 34th IEEE Conference on Decision and Control, New Orleans, LA, USA, Vol. 1, 703-708.

[84] Ninness, B. M. e Gustafsson, F. (1997). A Unifying Construction of Orthonormal Bases for System Identification. IEEE Transactions on Automatic Control. Vol 42, Issue 2, 515-521.

[85] Norgaard, M., Ravn, O., Poulsen, N. K. e Hansen, L. K. (2000). Neural Networks for Modelling and Control of Dynamic Systems. Springer.

[86] Ogata, K. (1978) System Dynamics. Prentice-Hall.

[87] Ogata, K. (1995) Discrete-time control systems, 2nd Edition. Prentice-Hall.

[88] Ogunnaike, B. A. e Ray, W. H. (1994). Process Dynamics, Modeling, and Control. Oxford University Press.

[89] Oliveira, G. H. C., Campello, R. J. G. B. e Amaral, W. C. (1999). Fuzzy Models within Orthonormal Basis Function Framework. IEEE International Fuzzy Systems Conference, Seoul, Korea, Vol. 2, 957-962.

[90] Palm, W. J. (1983). Modeling, Analysis and Control of Dynamic Systems. John Wiley & Sons.

[91] Park, D. Kandel, A. e Langholz, G. (1994). Genetic-Based New Fuzzy Reasoning Models with Application to Fuzzy Control. IEEE Transactions on Systems, Man and Cybernetics, Vol. 24, Issue 1, 39-47.

[92] Parks, T. R. (1999). Manual for Model 730 - Magnetic Levitation System, ECP.

[93] Pedrycz, W. e Gomide, F. A. C. (1998). An Introduction to Fuzzy Sets: Analysis and Design (Complex Adaptive Systems). MIT Press.

[94] Peña-Reyes, C. A. (2002). Coevolutionary Fuzzy Modeling. Tese de doutorado, Section de'informatique, École Polytechnique Fédérale de Lausanne, Lausanne, Suiça.

[95] Rosa, A. (2005). Desenvolvimento de Modelos Discretos de Volterra usando Funções de Kautz. Dissertação de Mestrado, DCA/FEEC/UNICAMP, Campinas, SP, Brasil.

[96] Rosenblatt, F. (1958). The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. Cornell Aeronautical Laboratory, Psychological Review, Vol. 65, No. 6, 386-408.

[97] Ruan, D. (1997). Intelligent Hybrid Systems - Fuzzy Logic, Neural Networks and Genetic Algorithms. Kluwer Academic Publishers.

[98] Russo, M. (1998). FuGeNeSys - A Fuzzy Genetic Neural System for Fuzzy Modeling. IEEE Transactions on Fuzzy Systems, Vol. 6, Issue 3, 373-388.

[99] Setnes, M., Babuska, R., Kaymak, U. e van Nauta Lemke, H. R. (1998). Similarity measures in fuzzy rule base simplification, IEEE Transactions on Systems, Man and Cybernetics - Part B: Cybernetics, Vol. 28, 376-386.

[100] Shaw, I. S. e Simões, M. G. (1999). Controle e Modelagem Fuzzy. Edgard Blücher.

[101] Schaffer, J. D., Whitley, D. e Eshelman, L. J. (1992). Combinations of genetic algorithms and neural networks: a survey of the state of the art. International Workshop on Combinations of Genetic Algorithms and Neural Networks, Baltimore, MD, USA, 1-37.

[102] Su, C. T., Lii, G. R. e Hwung, H. R. (1996). A Neuro-Fuzzy Method for Tracking Control. IEEE International Conference on Industrial Technology, Shanghai, China, 682-686.

[103] Subbu, R. e Bonissone, P. P. (2003). A Retrospective View of Fuzzy Control of Evolutionary Algorithm Resource. IEEE International Conference on Fuzzy Systems, St. Louis, MO, USA, Vol. 1, 143-148.

[104] Takagi, T. e Sugeno, M. (1985). Fuzzy identification of systems and its applications to modeling and control. IEEE Transactions on Systems, Man, and Cybernetics. Vol. 15, 116-132.

[105] Tenorio, M. F. e Lee, W. (1990). Self-organizing network for optimum supervised learning. IEEE Transactions on Neural Networks, NN-1, 100-109.

[106] Tsoukalas, L. H. e Uhrig, R. E. (1997). Fuzzy and Neural Approaches in Engineering. Wiley-Interscience.

[107] United States Patent and Trademark Office. http://www.uspto.gov/index.html. Acessado em julho de 2005.

[108] Van den Hof, P. M. J., Heuberger, P. S. C. e Bokor, J. (1994). System identification with generalized orthonormal basis functions. Proceedings of the 33rd IEEE Conference on Decision and Control, Lake Buena Vista, FL, USA, Vol. 4, 3382-3387.

[109] Verbruggen, H. B. e Babuska, R. (1999). Fuzzy Logic Control, Advances in Applications. World Scientific.

[110] Vonk, E., Jain, L. C. e Johnson, R. P. (1997). Automatic Generation of Neural Network Architecture Using Evolutionary Computation. World Scientific.

[111] Von Zuben, F. J. (2005). Notas de Aula da disciplina Computação Evolutiva, disponível em http://www.dca.fee.unicamp.br/∼vonzuben. Acessado em 13/02/2005.

[112] Wahlberg, B. (1991). System identification using Laguerre models. IEEE Transactions on Automatic Control, Vol. 36, Issue 5, 551-562.

[113] Wahlberg, B. (1994). System identification using Kautz models. IEEE Transactions on Automatic Control, Vol. 39, Issue 6, 1276-1282.

[114] Wallace, C. S. e Freeman, P. R. (1987). Estimation and inference by compact coding. J. R. Stat. Soc. B, Vol. 49, 240-265.

[115] Wang, L.-X. e Mendel, J. M. (1992). Fuzzy basis functions, universal approximation, and orthogonal least-squares learning. IEEE Transactions on Neural Networks, Vol.3, Issue 5, 807-814.

[116] Wu, C. J. e Liu, G. Y. (2000). A Genetic Approach for Simultaneous Design of Membership Functions and Fuzzy Control Rules. Journal of Intelligent and Robotic Systems, Vol. 28, Issue 3, 195-211.

[117] Zadeh, L. A. (1965) Fuzzy Sets. Information and Control, Vol. 8, 338-353.

[118] Zhang, T., Jamshidi, M., Coelho, L. S. e Krohling, R. A. (2002). Robust Control Systems with Genetic Algorithms. CRC Press.