UNIVERSIDADE REGIONAL INTEGRADA DO NOROESTE DO ESTADO DO RIO GRANDE DO SUL – UNIJUÍ

ADENIR SEBASTIÃO DE BRITTO

IDENTIFICAÇÃO DE SISTEMAS UTILIZANDO ALGORITMOS MEMÉTICOS PARA ESTIMAÇÃO DE PARÂMETROS

Ijuí

2007

ADENIR SEBASTIÃO DE BRITTO

IDENTIFICAÇÃO DE SISTEMAS UTILIZANDO ALGORITMOS MEMÉTICOS PARA ESTIMAÇÃO DE PARÂMETROS

Dissertação apresentada ao Curso de Mestrado em Modelagem Matemática da Universidade Regional Integrada do Noroeste do Estado do Rio Grande do Sul – UNIJUÍ, como requisito parcial à obtenção do título de Mestre em Modelagem Matemática.

Orientador: Professor Doutor Gideon Villar Leandro

Ijuí

2007

Aos meus pais Severiano e Angelina, a minha esposa Ilce, as minhas filhas Eliane e Bruna.

Agradecimentos:

- Acima de tudo, a Deus.

- À Universidade de Passo Fundo pelo incentivo e ajuda financeira durante a realização do curso de mestrado.

- À UNIJUI por ter proporcionado a possibilidade de fazer o curso.

- Ao Professor Doutor Gideon Villar Leandro por ter aceitado a minha proposta de trabalho e pela excelente orientação.

- A toda minha família pela compreensão da minha ausência, principalmente a esposa Ilce e as filhas Eliane e Bruna.

- Ao colega Professor Mestre Moacir Carlos Gomes, que cedeu o algoritmo genético no início deste trabalho e auxiliou na implementação dos resultados obtidos neste trabalho.

- A Professora Maria Emilse pela revisão textual.

- Aos colegas professores e funcionários da UPF pelo apoio e incentivo.

- Aos amigos e amigas que tive oportunidade de conhecer durante a realização do curso em Ijuí pelos incentivos e alegrias proporcionadas.

RESUMO

Uma das maiores dificuldades encontradas no trabalho com otimização combinatória é que os métodos utilizados requerem procedimentos específicos para cada situação tratada. Mesmo métodos considerados de uso geral como os algoritmos evolutivos, precisam da definição de operadores especiais para cada problema estudado. Neste trabalho é desenvolvido um algoritmo memético, pertencente ao grupo dos algoritmos evolutivos que estima parâmetros de sistemas lineares e também não lineares. Os sistemas são representados por modelos discretos do tipo ARX e NARX polinomiais.

O algoritmo memético proposto na sua fase de busca local utiliza o Método de Newton Inexato, para refinar a melhor solução até então encontrada. Resultados obtidos com a metodologia proposta são comparados com resultados de algoritmo genético disponível na literatura. Analisando os resultados verifica-se sempre um melhor desempenho do algoritmo memético, pois ele converge num número menor de iterações, economizando tempo computacional e, quando submetido a sinais diferentes dos utilizados na fase de estimação, consegue representar satisfatoriamente as dinâmicas presentes no sistema.

Palavras-chaves: Identificação de sistemas, estimação de parâmetros, algoritmos meméticos, busca local.

ABSTRACT

One of the biggest difficulties found in this work on combinatory optimization is that the used methods require specific procedures for each of the situations being treated. Even the methods which are considered of general use such as the evolutionary algorithms need the definition of special operators for each of the problems studied. In this approach a Memetic Algorithm is developed, belonging to a group of the evolutionary algorithms that estimates not only linear system but also no linear parameters. The systems are represented by discreet models of the ARX and NARX polinomial type.

The Memetic Algorithm, proposed in its local search phase, uses Inexact Newton's Method to refine the best solution found so far. Results obtained with the proposed methodology are compared with Genetic Algorithm results available in the literature. Analyzing the results we can always see a better performance of the Memetic Algorithm, because it converges in a smaller number of iterations saving computational time and when submitted to signs which are different from the ones used in the estimate phase satisfactory succeeds in representing the dynamics presented in the system.

Key words: System Identification, estimate of parameters, memetic algorithms, local search.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

NOTAÇÃO E SIMBOLOGIA

AE	algoritmo do elipsóide;
AG	algoritmo genético;
AM	algoritmo memético;
ARX	modelo auto-regressivo com entradas exógenas (*autoregressive model with exogenous inputs*);
ERR	taxa de redução de erro (*error reduction ration*);
LI	limite inferior do conjunto de dados;
LS	limite superior do conjunto de dados;
LU	triangularização de matrizes (L, de "*lower*" , inferior; U, de "*upper*", superior);
MIMO	multientradas, multisaídas (multi-*input, multi-output*);
MQ	mínimos quadrados;
NARX	modelo não linear auto-regressivo com entradas exógenas (*nonlinear autoregressive model with exogenous inputs*);
NARMAX	modelo não linear auto-regressivo, de média móvel com entradas exógenas (*nonlinear autoregressive moving average model with exogenous inputs*);
PCA	percentual de cruzamento aritmético;
PCE	percentual de cruzamento com os extremos;
PM	percentual de mutação;
P*mnu*	percentual de mutação não uniforme;
QP	quantidade de pontos;
SISO	uma entrada e uma saída (*single-input, single-output*);
$ans_p(x)$	respostas correspondentes ao problema P;
B	parâmetro de curvatura da região factível;

C	coeficiente de correlação;
d	tempo morto;
$e(k)$	erro no instante k;
F	função genérica para o modelo NARX;
F_i	função de avaliação;
$F(x)$	sistema genérico;
f, g	função genérica para saída e para a entrada do sistema, respectivamente;
$f_i(x)$	derivadas primeiras das funções de $F(x)$;
G	numero máximo de gerações;
G_i	função de avaliação associada ao coeficiente de correlação;
$H(s)$	função de transferência;
$H(z)$	transformada de Laplace de $H(s)$;
$\hat{H}(z_G)$	função estimada pelo algoritmo genético;
$\hat{H}(z_M)$	função estimada pelo algoritmo memético;
I_p	domínio de entrada;
$J(xk)$	matriz jacobiana no instante k;
n_u	máximo atraso entre os regressores de entrada;
n_y	máximo atraso entre os regressores de saída;
P	problema computacional;
$Par(i, j)$	gene de um cromossomo (um dos parâmetros da solução);
$P(k)$	conjunto de x_i soluções do sistema;
$p(i)$	estratégia de seleção por ranking geométrico normalizado;
q^{-1}	operador de atraso, $y(k)q^{-1} = y(k - 1)$;
$sol_p(x)$	solução factível para o problema P;
^{T}	transposição de vetores ou matrizes;
Ts	tempo de amostragem;
t	contador de iterações;
$u(k)$	entrada de sistemas não autônomos no instante k;
X, Y	cromossomos pais;
x, y	cromossomos filhos;
x_i	suposta solução do sistema(cromossomo);

$y(k)$	sinal de saída no instante k;
$\hat{y}(k)$	estimativa de saída do sistema no instante k;
$\Re$	conjunto dos números reais;
$\Re^n$	espaço de n dimensões;
$\Re^m$	espaço de m dimensões;
$\propto_i$	parâmetros da saída do sistema;
β_i	parâmetros da entrada do sistema;
θ	vetor de parâmetros a estimar;
$\hat{\theta}$	vetor de parâmetros estimado;
$\varphi(k)$	saídas dos sistema em função dos valores prévios dos sinais de saída e de entrada;
$\xi(k)$	resíduo no instante k;
$\psi(k - 1)$	vetor de regressores que contém observações até o instante $k - 1$;

INTRODUÇÃO

Sistema é o conjunto de componentes (físicos, biológicos, mecânicos, etc.) que interagem de forma definida. Um conjunto de equações é também um sistema, porém, a fim de evitar confusão, reserva-se o termo "sistema" para o processo real, e uma equação, ou um conjunto de equações, será chamado de "modelo". Se este é representativo daquele, dizemos que este é um modelo daquele sistema. Percebendo ou não, o ser humano funciona baseado em modelos. Certamente, a maioria desses modelos são mentais (AGUIRRE, 2004).

A modelagem procura reproduzir, com certa precisão, as principais características destes sistemas. É sabido também que um modelo pode representar bem uma ou mais características de um sistema e, por outro lado, não é viável a sua aplicação a outras características. Com estas constatações, pode-se observar que os modelos de um sistema são, simplesmente, representações aproximadas do mesmo e que a escolha do modelo adequado é fundamental para se obter êxito na representação desejada.

Na modelagem matemática encontra-se uma das formas utilizadas para descrever características de um sistema. Fazendo uso de equações matemáticas, esta forma de modelagem simula saídas para o sistema de acordo com os estímulos aplicados (entradas), permitindo, assim, a descrição do comportamento **dinâmico do sistema.**

A identificação de sistemas não lineares é muito utilizada para descrever e analisar características de sistemas reais e tem sido aplicada a vários campos da ciência (LJUNG e SODERSTROM, 1983). Nas últimas décadas consolidaram-se a teoria e a

prática de identificação de sistemas. Inicialmente, os modelos matemáticos usados para representar os sistemas eram lineares e considerados suficientes, apesar de não reproduzirem o comportamento não linear dos sistemas analisados.

Até a década de 80, a resolução dos problemas de programação linear e não linear era proposta através de métodos de otimização. A escolha do método de resolução a ser utilizado depende principalmente da razão entre a qualidade da solução gerada pelo método e o tempo gasto para encontrar essa solução. O desenvolvimento e sucesso dos métodos heurísticos, em especial as metaheurísticas, fomentaram o interesse dos pesquisadores na década de 90 pela aplicação desses métodos em problemas de otimização combinatória, considerados complexos apesar do uso intensivo e da evolução computacional.

O sucesso de modelos discretos paramétricos na aproximação de sistemas dinâmicos contínuos no tempo é bem conhecido (NORTON, 1986; AGUIRRE, 2000; MENDES, 1995; SWAIN, 1996; LJUNG, 1999). Contudo, a interpretação dos parâmetros do modelo discreto com relação ao correspondente processo físico não é direta. É, portanto, freqüentemente desejável identificar modelos contínuos no tempo, já que podem conter relações diretas com os parâmetros de interesse.

De acordo com Moscato (1999), o uso genérico da denominação algoritmo memético é feito para identificação de metaheurísticas, constituindo um dos procedimentos de maior êxito na resolução de problemas de otimização combinatória.

Algoritmos meméticos vêm sendo amplamente estudados e aplicados em vários problemas de otimização encontrados na literatura, tais como seqüenciamento de máquinas (MOSCATO E NORMAN, 1992; CHENG E GEN, 1997; MENDES, 2003), caixeiro-viajante (BUI E MOON, 1996; TIN JUNIOR, 2001) e outros.

O objetivo desta dissertação é desenvolver e testar a eficácia de um algoritmo memético para problemas de identificação de sistemas utilizando-se para busca local o método de Newton inexato. A razão da escolha deste método deve-se a sua flexibilidade, simplicidade de implementação computacional e eficácia na busca de soluções globais para problemas representados por funções não deriváveis, descontínuas e multimodais. Além dessas características, é possível trabalhar com diversos tipos de sinais.

Esta aplicação foi implementada computacionalmente através do software Matlab, que permitiu avaliar a resposta temporal dos modelos estimados.

A dissertação está organizada em quatro capítulos.

O Capítulo 1 trata da fundamentação teórica sobre identificação de sistemas, apresentando vários aspectos a serem observados em problemas reais de identificação, os quais são agrupados em cinco etapas: coleta de dados, escolha da representação matemática, determinação da estrutura do modelo, estimação de parâmetros e validação do modelo (AGUIRRE, 2004).

No Capítulo 2 é feita uma revisão dos conceitos básicos sobre algoritmos meméticos, descrevendo-se as principais etapas do fluxograma de funcionamento desses algoritmos. Inicialmente, apresentam-se a representação dos cromossomos, a inicialização da população, a função de avaliação e os critérios de parada. Descreve-se, também, a busca local, principal responsável pelas mudanças que ocorrem na população ao passar de uma geração para outra. Os operadores genéticos descritos são: seleção, cruzamento e mutação.

No Capítulo 3 é apresentada a metodologia usada para a identificação de sistemas, bem como a estrutura, parâmetros e funcionamento do algoritmo memético aplicado à estimação de parâmetros de sistemas lineares do tipo ARX e sistemas não lineares, do tipo NARX, ambos invariantes no tempo. A descrição completa do algoritmo memético utilizado é também apresentada neste capítulo. Esta descrição contém: a representação dos cromossomos; o tamanho e a geração da população inicial; a função de avaliação escolhida; os operadores genéticos de seleção, cruzamento e mutação utilizados, a busca local, bem como a freqüência de aplicação dos operadores utilizados.

O Capítulo 4 tem como objetivo principal comparar o desempenho do algoritmo memético proposto. A partir das considerações iniciais, descrevem-se os resultados obtidos para sistemas lineares de segunda, terceira e quarta ordens e, também, para sistemas não lineares. Na fase de estimação foi utilizado sinal aleatório e, na fase de validação, outro sinal aleatório com as mesmas características do sinal usado na estimação; também foi utilizado um sinal em degrau unitário, com a finalidade de verificar se o modelo é global.

Finalmente, no Capítulo 5 são apresentadas as principais conclusões e sugestões para futuros trabalhos.

1 IDENTIFICAÇÃO DE SISTEMAS

1.1 Introdução

O interesse por algoritmos mais eficientes tem se tornado uma das preocupações da comunidade científica. Uma das alternativas que vêm sendo investigadas com significativa ênfase, visando agregar maior poder computacional e proporcionar maior agilidade no estudo de sistemas, é o emprego da modelagem matemática, que consiste na representação do comportamento do processo por meio de equações matemáticas e pela definição de um modelo que explique o grau de dependência da saída do sistema em relação à entrada.

Na literatura existem várias técnicas de modelagem, e uma possível classificação é: *modelagem caixa branca*, *modelagem caixa preta* e *modelagem caixa cinza* (AGUIRRE, 2004). Na modelagem caixa branca, que também é conhecida como "modelagem pela física" ou "natureza do processo", ou, ainda, "modelagem fenomenológica ou conceitual", é indispensável o conhecimento profundo das relações internas do sistema, visto que o modelo matemático resulta do equacionamento de cada relação entre os elementos que constituem o sistema. Esse tipo de modelagem, dependendo da complexidade da planta, pode se tornar inviável, em virtude da necessidade do conhecimento, do tempo disponível ou do custo necessário.

A técnica alternativa, referida como "modelagem caixa preta" ou "modelagem empírica", tem como principal característica necessitar de pouco ou nenhum

conhecimento do sistema, bastando somente os dados de entrada e saída. A área da modelagem que estuda essa técnica é chamada de "identificação de sistemas". O que motiva o estudo da identificação de sistemas é o fato de, normalmente, não serem conhecidas as equações que regem o funcionamento do sistema e, quando conhecidas, seu uso seria impraticável pelo tempo disponível, pelos recursos ou condições de aplicação, além da dificuldade na estimação dos parâmetros envolvidos.

A "modelagem caixa cinza" está entre a modelagem fenomenológica e a modelagem caixa preta, as quais podem ser interpretadas como os dois extremos de técnicas de modelagem. Nela o processo de identificação ocorre com o conhecimento, ao menos parcial, do sistema. Esse processo se caracteriza pelo uso de informações auxiliares que não estão disponíveis nos dados de entrada e saída do sistema. Por permitir a incorporação de informações prévias à identificação, a modelagem caixa cinza normalmente resulta em modelos melhores. Esta é uma área bastante ampla e, por se tratar de assunto relativamente novo, muitos problemas encontram-se na fase de estudos preliminares.

Deve-se notar que os modelos de um sistema são simplesmente representações aproximadas da planta; conseqüentemente, não existe um modelo e, sim, uma família de modelos com características e desempenhos variados. Além disso, o modelo é uma aproximação de algumas características do sistema real. Caso se pretendesse gerar um modelo que busca representar muitas características do sistema real, possivelmente o objetivo seria inatingível, em razão da complexidade exigida pelo modelo (AGUIRRE, 2004).

1.2 Modelos Matemáticos

Modelos matemáticos constituem um poderoso recurso para resumir o conhecimento acerca de um processo ou sistema. Porém, os modelos matemáticos são classificados de várias formas para sua distinção.

É possível distinguir modelos lineares de modelos não lineares. Os modelos lineares satisfazem o princípio da superposição (princípio da homogeneidade mais o princípio da aditividade) ou têm o mesmo tipo de comportamento independentemente do ponto de operação, enquanto que os sistemas não lineares não satisfazem a este princípio. A importância da linearidade em um modelo está nas simplificações obtidas durante sua análise, possibilitando, inclusive, isolar os efeitos das variáveis de entrada para estudo separadamente. Aguirre (2004) afirma que "todo o sistema é em princípio não linear. A dinâmica de sistemas não lineares normalmente depende da amplitude do sinal de entrada, bem como do ponto de operação do sistema. Em torno de um ponto de operação, alguns sistemas não lineares podem ser aproximados por modelos lineares".

Uma segunda forma de classificar modelos é baseada na característica da variável independente temporal. Um modelo é contínuo quando as variáveis são contínuas no tempo, isto é, o sistema possibilita seu acompanhamento em qualquer instante de tempo. Esses modelos geralmente são descritos por equações diferenciais. Os modelos discretos são aqueles nos quais as variáveis do sistema evoluem em instantes isolados de tempo. Os instantes de tempo nos quais as variáveis podem alterar seus valores correspondem a múltiplos do período de amostragem. Modelos discretos são representados por equações a diferenças.

A consideração de invariância no tempo implica que o sistema modelado não varia com o tempo. Isso não significa que as variáveis envolvidas no sistema em estudo têm valores constantes; geralmente os valores das variáveis flutuam com o tempo, sendo tal evolução temporal determinada por uma lei. Ser invariante no tempo não quer dizer que o sistema está "estático", mas certamente implica que a dinâmica que está regulando a evolução temporal é a mesma. (AGUIRRE, 2004).

Finalmente, incertezas poderão ocorrer quanto à natureza probabilística na resposta do modelo. Se essa incerteza for significativa, um modelo estocástico deve ser usado, no qual algumas variáveis são descritas em termos de uma função de distribuição de probabilidade.

1.3 Etapas na Identificação de Sistemas

A identificação de sistemas visa encontrar um modelo matemático que melhor representa a relação entre o sinal de entrada $u(k)$ e a saída $y(k)$ de um sistema real qualquer. Há vários aspectos a serem observados em problemas reais de identificação, os quais podem ser agrupados em cinco etapas: coleta de dados, escolha da representação matemática, determinação da estrutura do modelo, estimação de parâmetros e validação do modelo (LJUNG,1999; AGUIRRE, 2004).

1.3.1 Coleta de Dados

Ao se buscar um modelo matemático que relacione dinamicamente duas variáveis de um sistema, parte-se do pressuposto de que há correlação significativa entre tais variáveis que justifique o modelo. Evidentemente, se não houver nenhuma relação de causa e efeito entre duas variáveis, a identificação de um modelo que as relacione torna-se injustificada (AGUIRRE, 2004). A identificação de sistemas propõe-se obter modelos a partir de dados, tornando-se necessário gerar e coletar esses dados. A coleta de dados é feita diretamente do processo em estudo, obtendo-se os dados de entrada e saída do sistema.

Quando os dados são utilizados imediatamente para a estimação dos parâmetros, diz-se que estão sendo utilizados em tempo real ou *on-line***,** porém nem sempre isso é possível. Neste caso, os dados devem ser armazenados para posteriormente serem utilizados no processo de identificação, o que consiste no processo denominado *off-line* ou por janelas.

A maior parte dos sistemas reais trata de processos contínuos no tempo, no entanto há diversas aplicações científicas confirmando a necessidade de representar variáveis contínuas de forma discreta no tempo, ou seja, deve-se trabalhar com amostras. O período entre duas amostras é chamado de “tempo” ou “período de amostragem”, T_S. É fundamental que o sinal amostrado contenha as características do sinal original, por isso o tempo de amostragem deve ser suficientemente curto para que a dinâmica da planta

seja bem representada. O teorema de Shannon diz que sinais com freqüência até $\frac{1}{2Ts}$ podem ser reconstruídos a partir de amostras separadas por T_S. Na prática, o tempo de amostragem é escolhido de forma que a freqüência seja entre cinco a dez vezes maior que a freqüência de interesse, não igual a duas vezes maior (freqüência de Nyquist) (AGUIRRE, 2004). Por outro lado, se o tempo de amostragem for muito curto (sobreamostragem), a estimação de parâmetros tende a se tornar mal condicionada, levando as colunas da matriz de regressores a se tornarem linearmente dependentes.

A importância da seleção correta do tempo de amostragem para modelos discretos pode ser vista em Billings e Aguirre (1995). Estes autores concluíram que o tempo de amostragem muito pequeno afeta o desempenho do algoritmo de seleção de estrutura e estimação de parâmetros; também, que algumas iterações não lineares só aparecerão e serão reproduzidas se a taxa de amostragem for suficientemente rápida.

1.3.2 Representação Matemática do Modelo

Na *representação matemática* do modelo de um sistema, entre os fatores preponderantes estão as suas características. Dentre as representações mais utilizadas podem-se citar modelos: lineares ou não lineares, variantes ou invariantes no tempo, estáticos ou dinâmicos, contínuos ou discretos, monovariáveis ou multivariáveis, determinísticos ou estocásticos e paramétricos ou não paramétricos. Matematicamente, a escolha do modelo de um sistema real recai sobre uma das tantas possibilidades existentes na literatura, no entanto prevalecerá a escolha do modelo que possibilite uma melhor representação do sistema em estudo.

Na prática, os sistemas dinâmicos, em última análise, são não lineares. Um sistema não linear pode apresentar comportamento linear em faixa estreita de operação. A escolha de modelos não lineares implica um inevitável aumento da complexidade dos algoritmos a serem utilizados, proporcionando uma significativa vantagem na exatidão do modelo e, sobretudo, na produção de regimes dinâmicos, que modelos lineares não conseguem representar.

Há uma grande variedade de representações não lineares, que, a princípio, podem ser utilizadas na identificação de sistemas.

Foram propostas por Narendra e Parthasarathy (1990) quatro classes de modelos para representar sistemas não lineares em aplicações de identificação e controle. Nos quatro modelos apresentados $u(k)$ é a variável de entrada; $y(k)$ é a variável de saída; f é a função que caracteriza a não linearidade na saída; g é a função que caracteriza a não linearidade na entrada; q^{-1} representa valores passados; α_i representa os parâmetros da saída do sistema e β_i representa os parâmetros da entrada do sistema.

Caso 1:

A equação (1.1) e a Figura 1.1 representam um mapeamento $f: \Re^m \rightarrow \Re$, no qual toda a não linearidade do sistema está em função da entrada *u(k)*.

$$y(k) = \sum_{i=1}^{n_y} \alpha_i y(k-i) + g[u(k-1), u(k-2), \ldots, u(k-n_u)] \tag{1.1}$$

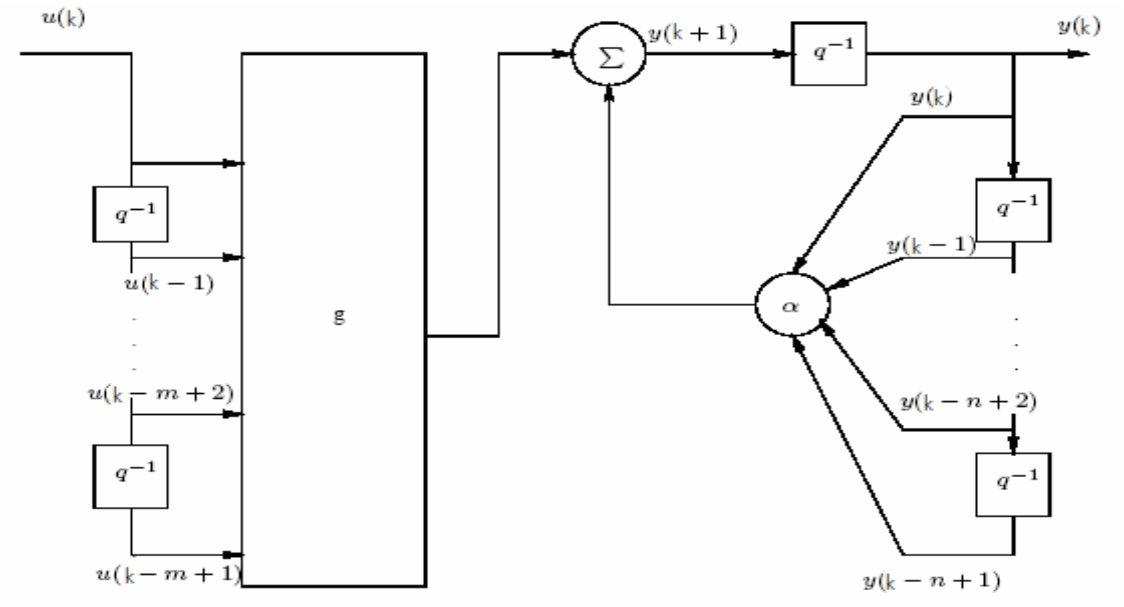


Figura 1.1 - Modelo I

Caso 2:

A equação (1.2) e a Figura 1.2 representam um mapeamento f: $\Re^n \rightarrow \Re$, onde a não linearidade do sistema está em função da saída *y(k)*.

$$y(k) = f[y(k-1), y(k-2), \ldots, y(k-n_y)] + \sum_{i=1}^{n_u} \beta_i (k-i) u(k-i) \qquad (1.2)$$

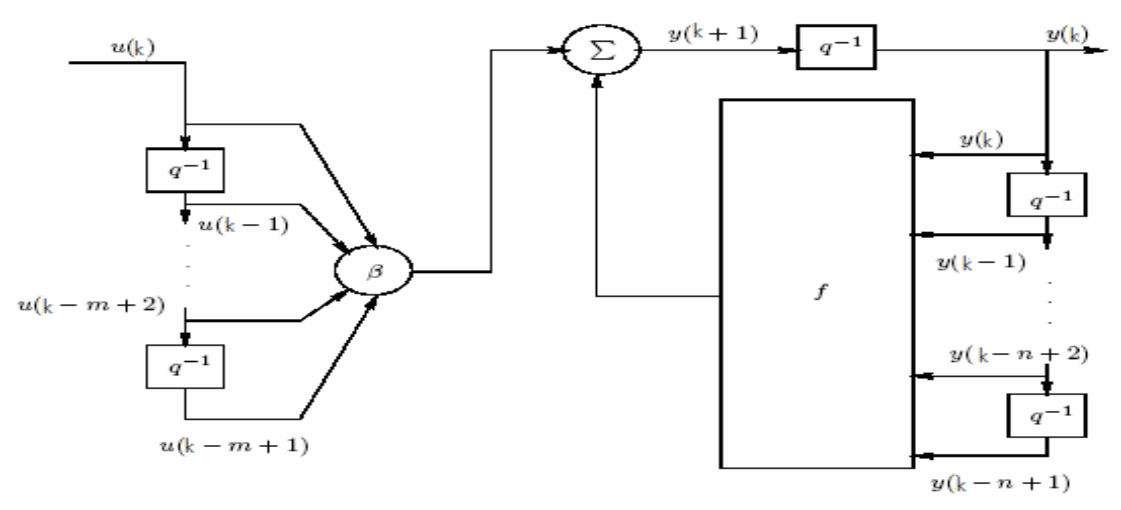


Figura 1.2 - Modelo II

Caso 3:

A equação (1.3) e a Figura 1.3 representam um mapeamento f: $\Re^n \rightarrow \Re$, no qual aparecem não linearidades tanto para a entrada *u(k)* quanto para a saída *y(k)*.

$$y(k) = f[y(k-1), \ldots, y(k-n_y)] + g[u(k-1), u(k-2), \ldots, u(k-n_u)] \qquad (1.3)$$

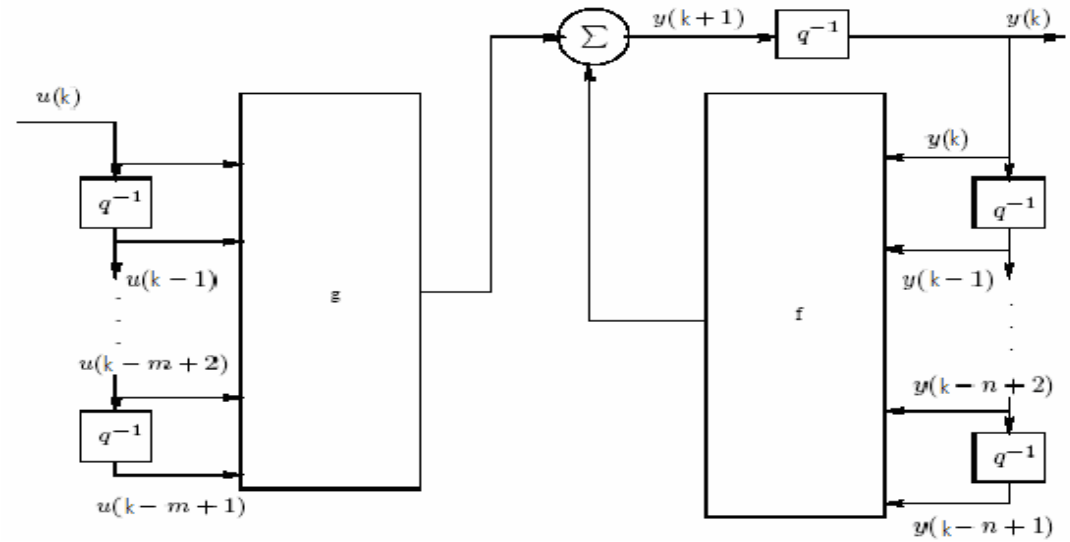


Figura 1.3 - Modelo III

Caso 4:

A equação (1.4) e a Figura 1.4 representam um mapeamento f: $\Re^n \rightarrow \Re$, no qual aparecem não linearidades tanto para a entrada *u(k)* quanto para a saída *y(k)*.

$$y(k) = f[y(k-1),\ldots, y(k-n_y); u(k-d), \ldots, u(k-n_u)] \tag{1.4}$$

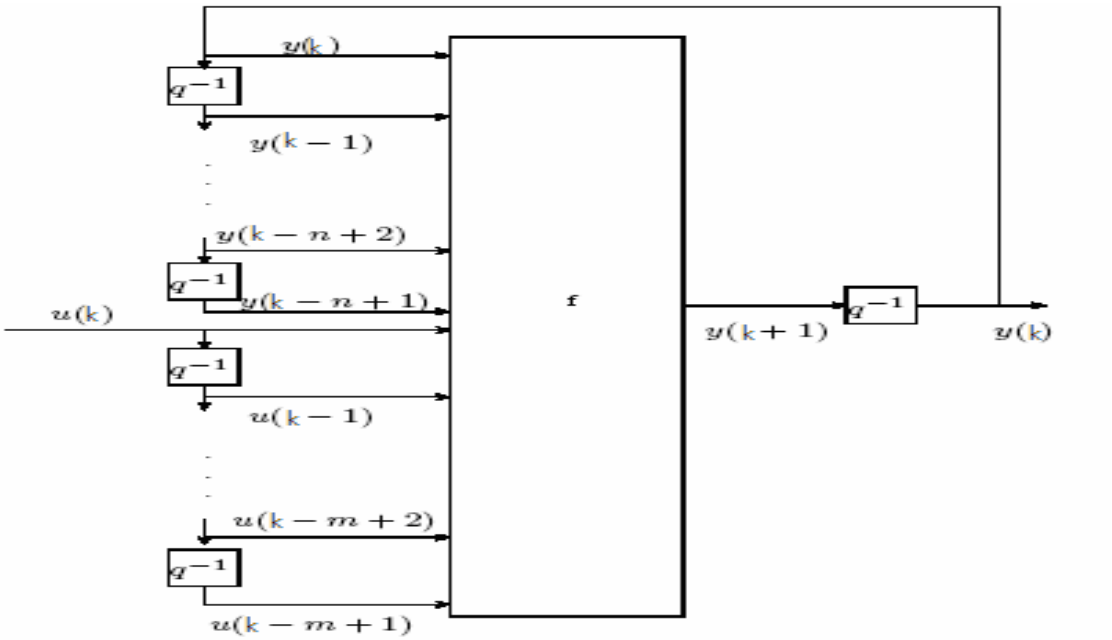


Figura 1.4 - Modelo IV

Nos quatro modelos, *u(k)* e *y(k)* representam a entrada e a saída de um sistema SISO (*Single Input – Single Output*) no instante *k; m* é o número de atrasos em *u;* e *n,* o número de atrasos em *y* com $m \leq n$. Essas quatro expressões podem ser generalizadas para o caso de sistemas MIMO (*Multiple Input – Multiple Output*), considerando **u** e **y** vetores *s* e *p*-dimensionais, respectivamente.

Nessas condições assume-se que as funções *f* e *g* são diferenciáveis em relação aos seus argumentos e, também, que a saída *y(k+1)* depende dos seus *n* valores passados, assim como dos *m* valores passados da entrada *u*. Na identificação de um sistema desconhecido, é necessário assumir que é BIBO (*Bounded Input – Bounded Output*) estável, isto é, que toda entrada limitada produz uma saída também limitada. Assim, o sistema pode ser identificado com base nos dados entrada-saída.

1.3.2.1 Modelo NARX

Os modelos NARX *(Nonlinear AutoRegressive model with eXogenous variables)* são modelos discretos no tempo que explicam o valor da saída *y(k)* em função dos valores prévios dos sinais de saída e de entrada do sistema. Um modelo NARX é um subconjunto dos modelos NARMAX, contendo funções de atraso de entrada e saída sozinhas. Utilizam metodologia de pontos fixos, agrupamentos de termos e coeficientes de agrupamentos de termos para auxiliar na definição da estrutura da função estática não linear que descreve a característica estática da representação. Normalmente são representados da seguinte forma:

$$y(k) = F[y(k - 1), \ldots , y(k - n_y), u(k - d), \ldots, u(k – n_u)] \quad (1.5)$$

Na equação (1.5), n_y representa o maior atraso em y; n_u representa o maior atraso em u e d representa o tempo morto.

1.3.3 Determinação da Estrutura do Modelo

Um dos aspectos mais importantes na determinação da estrutura de modelos é a escolha da ordem do modelo. A escolha da ordem errada para um modelo pode, por um lado, não representar a sua complexidade estrutural e, por outro, torná-lo mal condicionado. Assim, a correta escolha da ordem do modelo é de fundamental importância para uma boa estimação de parâmetros.

Os métodos de determinação da estrutura para sistemas monovariáveis consistem na obtenção das ordens das equações a diferenças que descrevem a dinâmica do sistema. Isso corresponde à escolha do número de parâmetros a serem estimados. A utilização de modelos subparametrizados (número de parâmetros menor que o real) aumenta o grau de incerteza, pois estes modelos de ordem reduzida deixam de representar certas características do sistema, aumentando, assim, o grau de incerteza já existente devido às perturbações. O uso de um modelo sobreparametrizado (número de parâmetros maior que o real) provoca aumento do esforço computacional e maior imprecisão na estimação dos parâmetros.

As não linearidades de modelos NARX são determinadas por produtos entre entradas, entre saídas ou por produtos entre elas. A escolha da estrutura destes modelos consiste em determinar que termos farão parte do modelo, ou seja, que produtos de termos devem estar presentes em sua estrutura.

Sistema não linear. Sistema que não satisfaz o princípio da superposição. (AGUIRRE, 2004).

Entre as técnicas para a determinação da estrutura destaca-se o procedimento denominado ERR (*error reduction ratio*) (BILLINGS et al., 1989b), que associa a cada termo-candidato um índice que corresponde à contribuição deste na explicação da variância dos dados de saída. Dessa forma, é possível ordenar os termos-candidatos de acordo com a contribuição de cada um. Este algoritmo gera uma lista em ordem decrescente com os termos e suas contribuições na identificação do sistema.

O ERR é um critério com base num algoritmo ortogonal que eficientemente combina seleção de estrutura e estimação de parâmetros, originalmente desenvolvido por Korenberg (1985). Foi estendido para a identificação de sistemas estocásticos de uma única entrada e única saída por Korenberg et al. (1988) e posteriormente aplicado a sistemas com múltiplas entradas e múltiplas saídas por Billings et al. (1989b). O ERR pode ser aplicado a problemas escritos na forma matricial

$$Y = P\hat{\theta} + \xi, \tag{1.6}$$

sendo P a matriz de regressores, $\hat{\theta}$ o vetor que contém os parâmetros a serem estimados e ξ o vetor de resíduos.

A escolha da ordem de um modelo aparece como um compromisso entre a capacidade de representação das dinâmicas do sistema e a simplicidade do modelo que representa um menor esforço computacional para os algoritmos de estimação e controle.

1.3.4 Estimação de Parâmetros

Após a coleta dos dados de entrada $u(k)$ e de saída $y(k)$ do sistema, escolhido o modelo e determinada a ordem desse modelo, chega-se ao momento de estimar os

parâmetros do modelo matemático escolhido.

Há vários métodos de estimação de parâmetros, sendo alguns determinísticos e outros estocásticos. Podem-se citar os métodos dos mínimos quadrados, máxima verossimilhança e filtro de Kalman (LJUNG, 1999), sendo o mais conhecido e aplicado o método dos mínimos quadrados.

Nas últimas duas décadas, técnicas que apresentaram bons resultados na resolução de problemas de otimização combinatória foram adaptadas à estimação de parâmetros, dentre as quais se destacam os algoritmos genéticos, redes neurais, simulated anneling, entre outras.

Na resolução de sistemas, a análise dos algoritmos tem como função determinar os recursos necessários para executar um determinado algoritmo. Destaca-se também a importância da análise criteriosa na escolha do mesmo visando à resolução dos problemas propostos. Nessa etapa alguns critérios devem ser considerados, tais como respostas apresentadas, tempo de execução, precisão dos comandos, legibilidade, estrutura, possibilidade de correção e de reutilização. O objetivo final é elaborar não apenas algoritmos que funcionem, mas também com eficiência.

1.3.5 Validação

Como etapa final, no processo de identificação, é necessária a validação dos resultados obtidos pelo modelo para verificar se esse modelo é suficientemente bom, pois normalmente as respostas têm um caráter relativo, isto é, o resultado deverá ser interpretado dentro de um determinado contexto. Naturalmente, o modelo deve ser avaliado, em suas características, de acordo com o uso pretendido, pois um modelo pode representar bem o sistema para algumas aplicações, mas não para outras, isto é, nenhum modelo representa o sistema real em todos os aspectos. Por isso, é interessante usar algumas ferramentas ou procedimentos que permitem verificar a qualidade do modelo encontrado. Análise semelhante, porém um tanto mais trabalhosa, deverá ser feita quando se tratar de uma família de modelos, pois em primeiro lugar deve-se identificar qual modelo melhor se adapta às finalidades pretendidas.

Uma maneira comum de validar um modelo é verificar se ele reproduz, ao longo do tempo, os dados observados do sistema, ou seja, comparar os dados gerados pelo modelo com a saída real do sistema. Porém, não devem ser usados os mesmos dados para estimação e validação, porque o objetivo é gerar um modelo com capacidade de generalização.

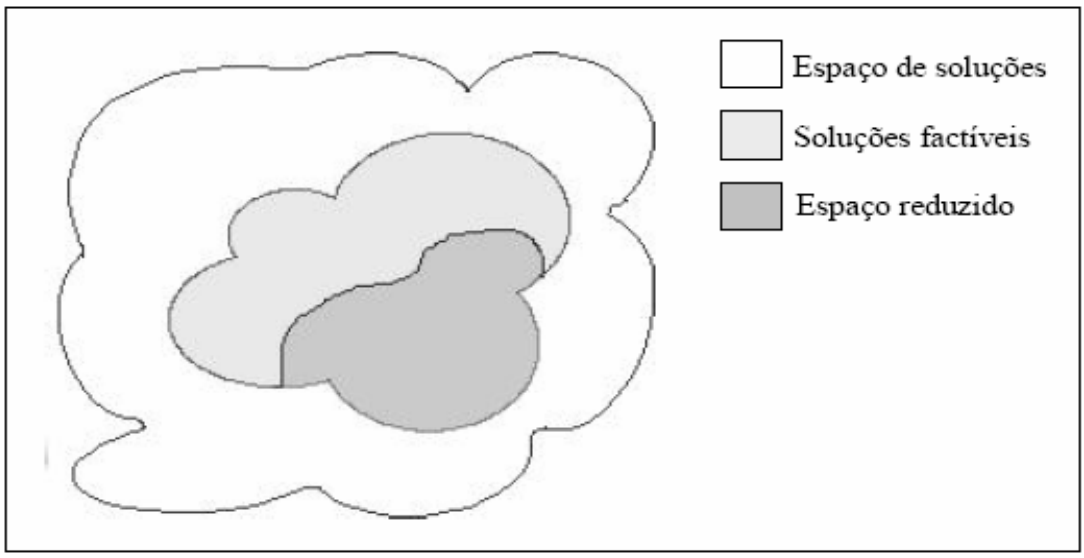
Um procedimento é gerar dois conjuntos de dados a partir do sistema funcionando em situações idênticas, sendo um usado para a estimação e o outro para a validação. Na impossibilidade de realizar dois testes nas mesmas condições, pode-se dividir o conjunto de dados em duas partes e usar a primeira para a estimação e a segunda para a validação (AGUIRRE, 2004).

2 ALGORITMOS MEMÉTICOS

2.1 Introdução

Um procedimento heurístico em geral encontra rapidamente uma solução para um problema, mas sem garantias de que seja ótima. Uma metaheurística diferencia-se de uma heurística por adicionar mais inteligência ao processo de busca por soluções. Tal metodologia procura escapar de ótimos locais e percorrer áreas mais amplas dentro do espaço de soluções possíveis. Busca Tabu (GLOVER, 1989), Simulated Annealing (KIRKPATRICK et al. 1983) e algoritmos genéticos (HOLLAND, 1975) são exemplos de metaheurísticas consagradas por bons resultados obtidos para uma vasta gama de problemas.

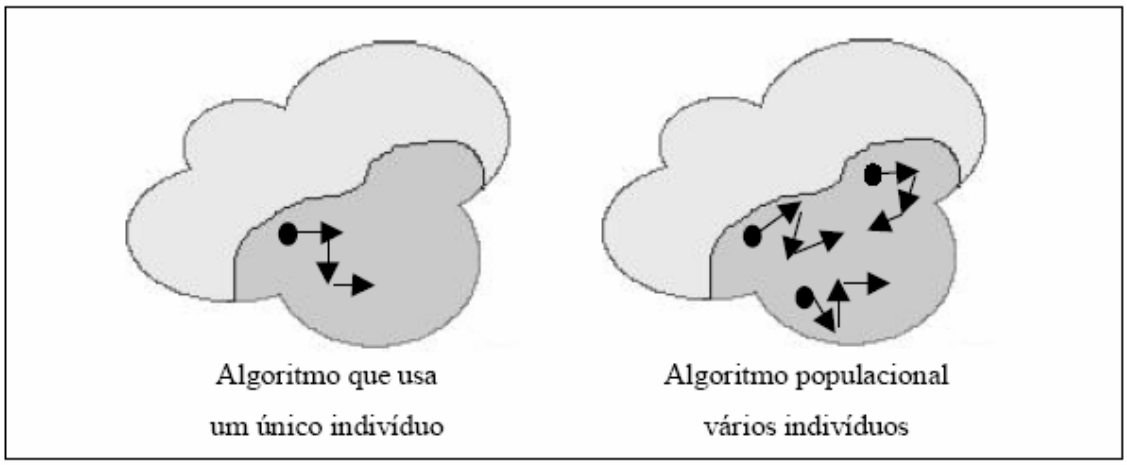
As heurísticas têm sido muito eficientes na redução do espaço de busca de problemas (Figura 2.1) para os quais algoritmos enumerativos levariam até anos para obter uma solução ótima. As heurísticas, especialmente as metaheurísticas, são aplicadas em problemas em que uma solução aproximada, cerca de 2% ou 3% distante do ótimo, traz mais benefícios do que uma solução exata, principalmente em relação à redução do tempo necessário para obtê-la.



FONTE: (TIN JUNIOR, 2001)

Figura 2.1 – Espaço de busca do domínio do problema

Algoritmo memético é uma metaheurística, uma estratégia que guia outras heurísticas em busca de soluções factíveis, pertencente à classe dos algoritmos populacionais. Algoritmo populacional usa vários indivíduos na procura de soluções factíveis no espaço de busca, conforme ilustração da Figura 2.2.



FONTE: (TIN JUNIOR, 2001)

Figura 2.2 - Algoritmo populacional *versus* algoritmo não populacional.

O termo “algoritmo memético” surgiu pela primeira vez em 1989, quando Moscato procurou definir os limites e diferenças entre algoritmo memético e algoritmo genético. R. Dawkins (1976), em seu livro *The selfish gene,* define o termo “meme” como uma unidade de imitação durante a transmissão cultural. Segundo Moscato (1989), algoritmo memético é uma definição mais apropriada para algoritmo genético híbrido, pois o termo “híbrido” retira o algoritmo genético de suas raízes da biologia. Algoritmo memético está mais associado a algoritmo evolutivo híbrido do que a algoritmo genético híbrido, primeiro em função da representação utilizada, que na maioria dos casos envolve inteiros, reais e também binários; segundo, pelo uso de heurísticas que exploram o espaço de busca por meio de movimentos nos indivíduos.

Algoritmos meméticos são algoritmos genéticos que incorporam um procedimento de busca local, permitindo intensificar a busca por soluções dentro da vizinhança dos indivíduos no algoritmo genético. A palavra “memético” origina-se de *meme,* definido como a menor unidade de conhecimento capaz de ser transmitida (MOSCATO, 1989).

Nesse algoritmo básico podem-se destacar duas fases: a primeira fase inicia-se com a geração da população inicial. Em seguida, a população inicial é submetida à função de avaliação para determinar a aptidão ou *fitness* de cada indivíduo (cromossomo); após, é verificado se foi atingido o critério de parada, que normalmente está relacionado à precisão da resposta e ao número máximo de gerações. Caso o critério de parada não seja satisfeito, o algoritmo entra no laço onde ocorrem os incrementos das gerações e as aplicações dos operadores genéticos de seleção, cruzamento e mutação. Em seguida, submetem-se os novos cromossomos à função de avaliação, finalizando-se a primeira fase ou etapa correspondente ao algoritmo genético. A segunda fase inicia-se selecionando o melhor cromossomo (indivíduo), que em seguida é submetido à busca local. Esta fase é específica do algoritmo memético.

A estrutura básica de um algoritmo memético é mostrada na Figura 2.3.

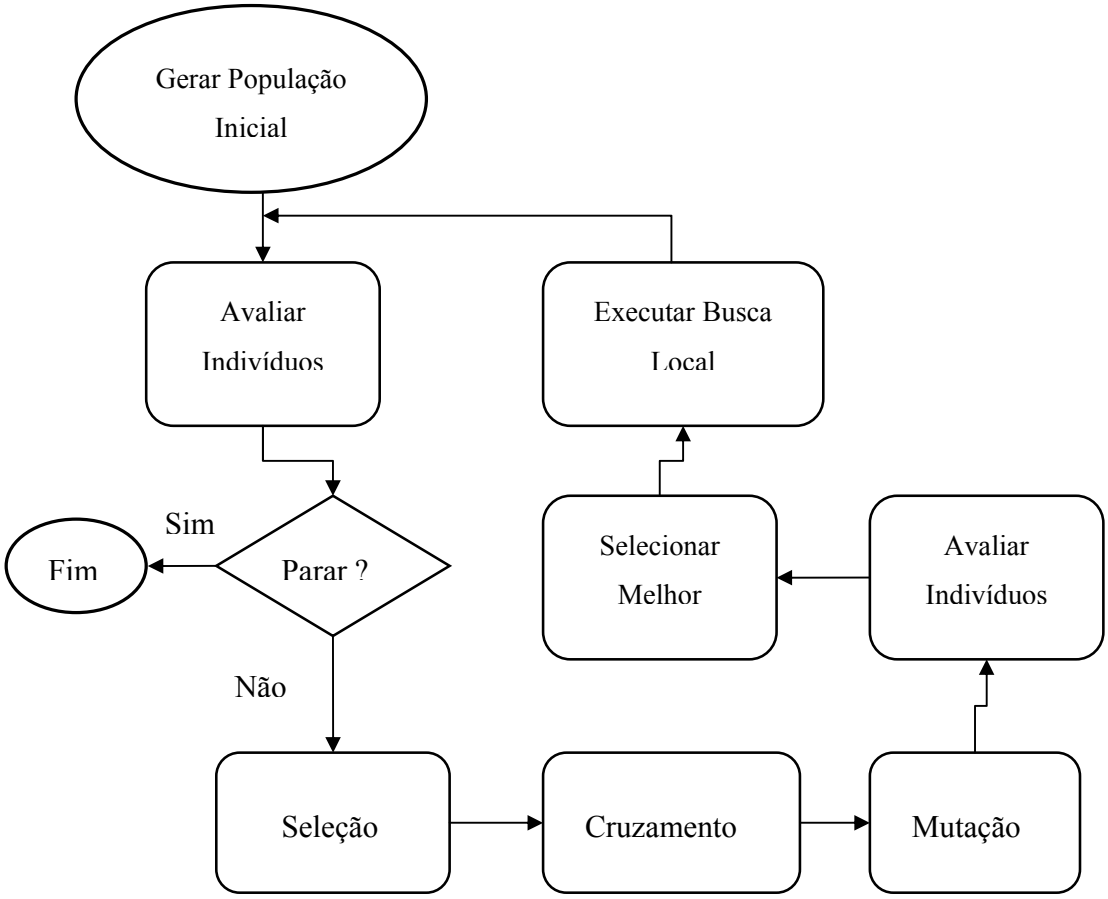
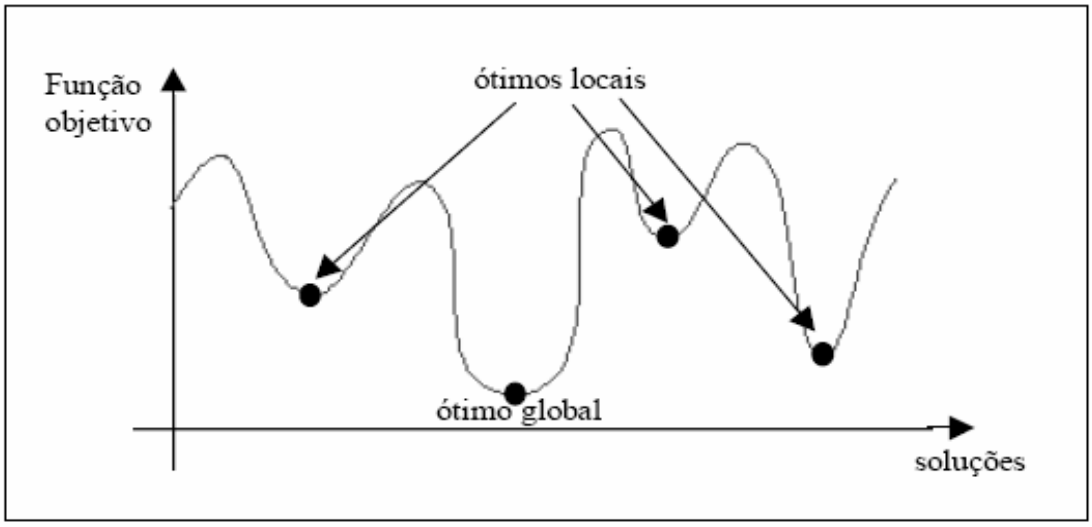


Figura 2.3 – Fluxograma de um algoritmo memético

2.2 Representação da Solução

Um algoritmo aplicado repetidas vezes a uma solução inicial tende a levar as soluções subseqüentes a um ótimo local. Uma vez obtido um ótimo local, a aplicação deste algoritmo não consegue melhorar a solução, fazendo com que o valor da função objetivo tenha tendência a piorar, conforme ilustrado na Figura 2.4.



FONTE: (TIN JUNIOR, 2001)

Figura 2.4 – Ótimo global e ótimos locais para um problema de minimização.

A escolha da representação correta para a solução do problema, bem como a estratégia de busca e os movimentos no espaço de soluções, pode criar ou evitar os ótimos locais (MOSCATO, 1989).

2.3 Estrutura Populacional

Uma das características da maior parte dos algoritmos evolucionários é adotar uma população hierarquicamente estruturada, na qual os indivíduos são classificados de acordo com sua qualidade, pois testes comparativos entre populações estruturadas e não estruturadas mostram que essa abordagem pode gerar um grande salto de desempenho em algoritmos tanto genéticos quanto meméticos. Outro ponto a ser destacado é que o número de indivíduos utilizados pelo método pode ser alterado consideravelmente sem que haja perda de desempenho causada pelo efeito da convergência. Isso também leva a uma significativa redução do esforço computacional e a um aumento no número de gerações executadas num mesmo intervalo de tempo (MENDES, 1999).

2.4 Algoritmo Genético

Os algoritmos genéticos são algoritmos computacionais que, a partir de uma amostra da população, procuram evoluir conforme o desempenho de seus indivíduos, privilegiando os mais aptos. Esta técnica faz parte de um grupo de algoritmos chamados "algoritmos evolucionários". Podem ser citadas, ainda, outras metodologias que fazem parte desses algoritmos evolucionários, tais como a programação evolucionária, estratégias evolucionárias, programação genética e sistemas classificadores, entre outros. Entretanto, os algoritmos genéticos são mais estudados pela sua flexibilidade, simplicidade de implementação e eficácia na busca global em ambientes adversos (TANOMARU, 1995). Ao longo do tempo, vários estudos foram realizados aplicando-se algoritmo genético, que, por meio de um método computacional, simula processos biológicos de recombinação genética, mutação e seleção, gerando soluções para um dado problema. O primeiro trabalho sobre algoritmo genético foi apresentado por Holland, (1975) e detalhes sobre estes algoritmos podem ser encontrados em Goldberg (1989), Davis (1991), Michalewicz (1996), Back et al. (2000a), Back et al., (2000b).

Diversos mecanismos que permitem guiar uma busca por soluções são adaptados do contexto da teoria da evolução dos seres vivos. Assim, indivíduo ou cromossomo no contexto do algoritmo genético é a representação de uma provável solução do problema matemático associado. As informações que fornecem as soluções estão de alguma forma codificadas no indivíduo, onde os genes representam cada atributo ou informação codificada nesse indivíduo. O conjunto de indivíduos forma uma população, e um algoritmo genético pode ter uma ou várias populações.

A estrutura de um algoritmo genético foi descrita por Michalewicz (1996) da seguinte maneira: durante a iteração *t,* um algoritmo genético mantém uma população de soluções potenciais (cromossomos, vetores), $P(k) = \{x_1^t, \ldots, x_n^t\}$; cada solução x_i^t passa por um processo de avaliação e produz um valor correspondente a sua adaptação ou *fitness;* é gerada uma nova população (iteração $k + 1$) contemplando a participação dos indivíduos mais qualificados.

O primeiro passo no algoritmo genético é responsável por gerar a população inicial sobre a qual o processo evolutivo será executado.

2.4.1 Indivíduos e Codificações

Os indivíduos ou cromossomos representam possíveis soluções para o problema. Os diferentes atributos que caracterizam cada indivíduo são chamados “genes” e as informações que descrevem uma solução do problema estão codificadas nos genes de cada indivíduo. Holland (1975) apresenta o algoritmo genético clássico, no qual as soluções candidatas são codificadas em arranjos binários de tamanho fixo. Entretanto, em diversas aplicações práticas a utilização de codificação binária leva a um desempenho insatisfatório (VON ZUBEN, 1996).

Na resolução de problemas de otimização numérica, os algoritmos genéticos podem ser apresentados através de representação inteira (cada cromossomo é um vetor cujos elementos são números inteiros) ou em ponto flutuante (cada cromossomo é um vetor de números na representação em ponto flutuante). Simulações computacionais comparando o desempenho de algoritmos genéticos com codificação binária e com ponto flutuante aplicados a um problema de controle foram apresentadas por Michalewicz, (1996), nas quais os resultados apresentados mostram uma significativa superioridade da codificação em ponto flutuante.

2.4.2 Inicialização da População

O método mais comum utilizado na definição da população inicial é a inicialização aleatória dos indivíduos, com a precaução de que todo o espaço de busca do problema seja atingido. Se algum conhecimento inicial a respeito do problema estiver disponível, este poderá ser empregado na inicialização da população, ou seja, os indivíduos são gerados de acordo com alguma regra inicial, com a preocupação de não gerar indivíduos inválidos na etapa de inicialização.

2.4.3 Função de Avaliação

Seguindo a inspiração biológica, o algoritmo genético avalia cada indivíduo de forma a identificar aqueles mais aptos a permanecerem na população. A aptidão de um indivíduo é definida por uma função chamada *fitness*. O *fitness* procura indicar quais indivíduos representam as melhores soluções para o problema em questão.

Quanto à avaliação do *fitness*, ao contrário da evolução natural, os algoritmos evolutivos não têm um ambiente real no qual a "sobrevivência" ou "aptidão" possa ser testada. Enquanto o *fitness* de um indivíduo na natureza depende da integração de todas as suas interações com o meio e com os demais indivíduos da mesma e de outras espécies, o *fitness* de um indivíduo em uma simulação computacional de um processo evolutivo pode envolver desde apenas a aplicação de um mapeamento estático, que leva cada conjunto de atributos que caracteriza um indivíduo ao seu valor de *fitness*, até alguma simulação computacional que reflita algum tipo de analogia com as mencionadas interações, com o meio e com outros indivíduos. O tempo de processamento da maioria dos algoritmos evolutivos é desprezível em comparação ao tempo de avaliação, havendo, freqüentemente, uma forte ênfase em reduzir o número de avaliações durante a evolução, ou, então, realizar a sua implementação em paralelo (MOSCATO, 1989; EBY et al., 1999). As funções que permitem obter um valor de *fitness* podem envolver um único ou múltiplos objetivos; podem ser unimodais ou multimodais, contínuas ou descontínuas, suaves ou "ruidosas", estáticas ou dinâmicas.

Algoritmos evolutivos são reconhecidamente eficientes em encontrar boas soluções para quase todos os tipos de funções de *fitness*, mas técnicas especializadas são freqüentemente requeridas para funções multimodais, com múltiplos objetivos, ruidosas e dinâmicas. A avaliação não é sempre executada por funções de *fitness* explícitas. Alguns algoritmos evolutivos empregam avaliadores humanos para avaliar e julgar as soluções. O *fitness* também pode ser determinado por competição entre soluções. Por exemplo, cada solução pode representar uma estratégia de um jogo, e o *fitness* de cada estratégia depende de quantas outras soluções na população de estratégias corrente podem ser derrotadas (AXELROD, 1987).

2.4.4 Critério de Parada

Já em relação ao critério de parada, também existem várias opções, embora nenhuma delas se mostre adequada a todos os casos:

i) monitoramento do tempo de execução: normalmente a evolução é terminada após um número específico de gerações, avaliações, ou tempo de processamento. Para algoritmos que usam funções de *fitness* custosas computacionalmente, ou para algoritmos que devem gerar soluções rapidamente, o critério de término primário deve estar baseado no tempo. Principal desvantagem: nenhuma garantia de que haja evolução significativa no processo no tempo especificado;

ii) monitoramento do nível de diversidade da população ao longo das gerações: finaliza o processo evolutivo assim que a diversidade cai abaixo de um determinado limiar. Principal desvantagem: calcular a diversidade da população a cada geração pode ser extremamente custoso computacionalmente;

iii) monitoramento da qualidade das soluções candidatas ao longo das gerações: finaliza o processo evolutivo assim que se encontrar um indivíduo com um valor de *fitness* acima de um determinado limiar. Principal desvantagem: pode ser muito difícil definir este limiar de qualidade, principalmente quando não é conhecido o valor de *fitness* para a solução ótima;

iv) monitoramento da qualidade das soluções candidatas ao longo das gerações: finaliza o processo evolutivo assim que se detectar a interrupção do processo de melhora do valor de *fitness*, o que pode ser medido por um número fixo de gerações sem melhora significativa. Principal desvantagem: se ainda houver diversidade suficiente na população, etapas de estagnação temporária do *fitness* podem ser seguidas por etapas de rápida evolução, sendo difícil definir um número de gerações capaz de indicar estagnação permanente. A detecção de estagnação pode ser utilizada também para reinicializar a população (WHITLEY & STARKWEATHER, 1990);

v) finalizar o processo evolutivo após transcorrido um número predeterminado de gerações. Principal desvantagem: por ser predeterminado, é difícil definir *a priori* um número adequado de gerações.

O critério v) é, certamente, o mais utilizado, por ser barato computacionalmente e por permitir o reinício do processo evolutivo (a partir do estado da última parada) caso se detecte a necessidade de prosseguir com a evolução. Um problema comum a todos os algoritmos diz respeito às "taxas dos operadores", isto é, quão freqüentemente aplicar cada operador. Uma resposta possível para esse problema é deixar o algoritmo calcular, ou seja, as taxas de aplicação de cada operador evoluirão e se adaptarão com o tempo, mudando de acordo com o nível de sucesso ou falha. Há muitos possíveis esquemas para esse processo de adaptação (TUSON & ROSS, 1998; BÄCK *et al.*, 2000a).

Em se tratando de identificação de sistemas, quando da estimação de parâmetros determina-se o momento de parada do algoritmo em função do erro entre a saída real calculada e a saída estimada pelo modelo, no qual este erro deve levar em consideração a população amostrada.

2.4.5 Operadores Genéticos

Os algoritmos genéticos, as estratégias evolutivas, a programação evolutiva e a programação genética consistem em classes de referência para algoritmos evolutivos. São ilimitadas as variações que podem ser introduzidas junto a cada algoritmo evolutivo, impedindo, inclusive, de se definir uma fronteira clara entre essas quatro classes. Todos os algoritmos evolutivos realizam a reprodução dos indivíduos, ou pela clonagem direta dos progenitores, ou usando operadores de recombinação e mutação para permitir herança com variação. Esses operadores podem realizar muitas tarefas diferentes, desde uma simples modificação de alelos selecionados aleatoriamente até operações de busca local. A evolução dos algoritmos genéticos é baseada na aplicação dos operadores genéticos sobre a população, com a finalidade de selecionar os mais aptos, cruzar os cromossomos entre si e provocar mutações nos descendentes. Os operadores genéticos são: seleção e reprodução, cruzamento ou recombinação e mutação.

2.4.5.1 Seleção e Reprodução

A seleção determina quais indivíduos irão reproduzir e quais irão permanecer na população. Dentre as técnicas de seleção mais empregadas encontram-se a seleção por torneio e pela roleta. A seleção por torneio estabelece a probabilidade de uma solução ser selecionada em termos de quantos outros indivíduos aleatoriamente selecionados ela pode derrotar. A seleção proporcional de *fitness*, ou roleta, estabelece a probabilidade de seleção como sendo diretamente proporcional ao *fitness* de cada indivíduo, devendo o *fitness* do indivíduo ser normalizado com base no *fitness* total da população naquela geração (GOLDBERG, 1989). Uma vez que a descendência foi gerada durante a etapa de reprodução, deve ser inserida na população. Algoritmos evolutivos usualmente mantêm populações de tamanho fixo. Assim, para cada novo indivíduo que é inserido na população, outro indivíduo existente deve ser eliminado.

Os algoritmos evolutivos mais simples eliminam todos os indivíduos da geração corrente e os substituem por novos descendentes, respeitando o tamanho fixo da população. Outros algoritmos mais elaborados realizam a substituição parcial da população corrente, proporcional ao *fitness*. Desse modo, os descendentes sempre substituem soluções menos adaptadas do que eles mesmos, ou os mais fracos da população são substituídos por descendentes mais adaptados. De fato, o uso de substituição baseada em *fitness* exemplifica o famoso argumento de Darwin: sobrevivência dos mais adaptados. Mas a substituição não precisa, necessariamente, ser baseada em *fitness*; ela pode ser baseada na satisfação de restrições, na similaridade de genótipos, na idade dos indivíduos, ou em qualquer outro critério pertinente, desde que uma pressão baseada em *fitness* seja exercida em alguma outra etapa do algoritmo evolutivo.

2.4.5.2 Cruzamento ou Recombinação

Criada uma população e avaliado cada indivíduo, o processo evolutivo é efetuado a cada geração (iteração do algoritmo). A seleção determina quais indivíduos irão reproduzir e quais irão permanecer na população. Os indivíduos com melhor valor de

fitness são, em geral, selecionados, mas isso dependerá do tipo de seleção adotada pelo algoritmo genético implementado. O operador de *crossover* (ou recombinação) procura efetuar trocas de informações codificadas nos indivíduos para que novas e, preferencialmente, melhores soluções sejam criadas. A pressão seletiva pode ocasionar uma rápida homogeneização da população, o que leva a uma prematura convergência do algoritmo. A idéia intuitiva que norteia o operador *crossover* é a troca de informações entre as diferentes soluções candidatas à solução do sistema.

O operador de cruzamento mais comumente empregado é o *crossover* de um ponto, que consiste na seleção de dois indivíduos (pais); a partir desses cromossomos são gerados dois novos indivíduos (filhos). Para a geração dos filhos seleciona-se aleatoriamente um mesmo ponto de corte nos cromossomos dos pais; os genes, a partir do ponto de corte, são trocados, gerando os filhos.

O valor da probabilidade de cruzamento depende do problema em estudo, mas geralmente são utilizados valores em torno de 40% do tamanho da população para a taxa de cruzamento.

Outros operadores de recombinação são descritos em Michalewicz e Schönauer (1996).

2.4.5.3 Operador de Mutação

O operador de mutação modifica aleatoriamente um ou mais genes de um cromossomo e pode ser aplicado antes ou depois do operador de cruzamento. A probabilidade de mutação em um gene é denominada “taxa de mutação”.

Intuitivamente, o operador de mutação cria uma variabilidade extra na população, mas sem destruir o progresso já obtido. Goldberg (1989) afirma que a mutação consiste em modificar aleatoriamente um ou mais genes do cromossomo pai e tem como objetivo restaurar a diversidade genética eventualmente perdida durante o processo evolutivo.

No caso de problemas com codificação binária, o operador de mutação padrão simplesmente troca o valor de um gene em um cromossomo (MICHALEWICZ, 1996).

Já, nos problemas com codificação em ponto flutuante, o operador de mutação seleciona aleatoriamente um componente do cromossomo e gera um indivíduo.

O operador de mutação procura evitar a homogeneização alterando a informação codificada em alguns indivíduos para restaurar uma saudável heterogeneização da população. O procedimento termina quando um número adequado de gerações foi atingido. Geralmente, são utilizados valores pequenos para taxa de mutação, abaixo de 5% do tamanho da população.

Outros operadores de mutação são descritos em Michalewicz e Schönauer (1996).

2.5 Algoritmo Memético

Moscato & Norman (1992) introduziram o termo "algoritmo memético" para descrever um processo evolutivo que possua uma busca local como parte decisiva na evolução. Genericamente, os algoritmos meméticos utilizam os operadores evolutivos que determinam regiões promissoras no espaço de busca, combinando com busca local nessas regiões. Merz e Freisleben (1999) afirmam que esse processo tem sido aplicado com sucesso em vários problemas de otimização, o que induz a concluir que algoritmo memético corresponde à união de um método de busca global e uma heurística local aplicada a cada indivíduo.

Merz e Freisleben (1999) afirmam também que num ambiente que emprega algoritmo memético os operadores de recombinação e mutação agem como estratégias de diversificação. Os indivíduos da população podem estar localizados em uma região do espaço de busca contendo um ótimo local, chamada "base de atração" do ótimo local. Utilizando a informação contida na população, novos pontos de partida podem ser descobertos após a busca local. Os operadores de recombinação e mutação podem gerar indivíduos da população que estejam localizados em bases de atração de ótimos locais ainda não explorados, de modo que um novo pico deva ser alcançado (maximização) ou um vale deva ser explorado (minimização).

2.5.1 Busca Local

Aarts e Verhoeven (1997) consideram a busca local como sendo uma aproximação geral para problemas de otimização combinatória baseados na exploração de vizinhanças. De maneira geral, é a principal diferença entre os algoritmos meméticos e os algoritmos genéticos, pois nestes nenhum indivíduo passa por qualquer processo de otimização via aplicação de operadores de busca local. Métodos de busca local para otimização combinatória geralmente se baseiam em uma definição de vizinhança que estabelece uma relação entre soluções no espaço de configuração do problema. Essa busca deverá ser realizada até que uma condição de parada seja satisfeita; essa condição deve ser atendida sempre que o processo de busca não tenha mais a capacidade de melhorar a solução atual.

De um modo geral, quanto mais conhecimento do problema se utiliza na elaboração da busca local, melhores são os resultados obtidos. Isso ocorre especialmente porque, em vizinhanças grandes, algumas possibilidades que são avaliadas são totalmente desnecessárias. Regras empíricas baseadas nas características esperadas das boas soluções permitem uma redução do tamanho da vizinhança. Outra alternativa consiste em não testar movimentos que não afetem a qualidade da solução.

Moscato (1999) apresenta uma definição formal de busca local nos seguintes termos: um problema computacional P tem domínio de entrada I_p, com $x \in I_p$, podendo ser estabelecido um conjunto $ans_p(x)$ de respostas correspondentes. No entanto, é preciso garantir que exista um subconjunto $sol_p(x) \subseteq ans_p(x)$, que identifica as soluções factíveis de P. Um algoritmo soluciona um problema P se, para a entrada $x \in I_p$, apresentar como saída qualquer $y \in sol_p(x)$ – solução factível – ou, em caso de $sol_p(x) = \{\}$, indicando que não existe y. A otimização combinatória é um tipo especial de problema de busca, em que cada $x \in I_p$ tem um conjunto $sol_p(x)$ de cardinalidade finita e cada solução $y \in sol_p(x)$ tem um valor de *fitness* $m_p(y, x)$. A busca, nesse tipo de problema, será responsável por encontrar uma solução factível $y^* \in sol_p(x)$, que maximize o *fitness* $m_p(y, x)$.

É um trabalho consideravelmente complicado a criação de buscas locais. Por um lado, o principal objetivo é conseguir o maior poder de busca possível, porém, por outro, é necessário manter a complexidade computacional sob controle.

2.5.2 Seleção Para Recombinação

A recombinação é uma forma de busca que usa um ou mais indivíduos para explorar o espaço de soluções do problema. Um operador de recombinação pode ser assexuado ou sexuado, isto é, poderá ser com dois ou mais indivíduos. Durante a recombinação os indivíduos trocam informações ou genes. Geralmente, as informações/genes similares entre ambos são preservados na sua descendência.

Hadamard (1949) *apud* (MENDES, 2003) afirma que as descobertas de novas invenções não ocorrem simplesmente por acaso, mas, sim, por uma combinação de idéias misturadas com alguma irracionalidade. A recombinação justamente adota esse comportamento ao combinar idéias parciais de soluções distintas para criar novas soluções.

Após a criação de um indivíduo pela recombinação, este sofre um processo de aprendizado ou evolução que explora o máximo de conhecimento, tornando-o apto para se reproduzir e gerar novos indivíduos, que herdarão novas características genéticas e culturais. Esse processo de evolução geralmente é realizado por algoritmos de busca local, mas também pode ser realizado por outras heurísticas ou algoritmos exatos.

Um indivíduo obtém o máximo de aprendizado quando atinge o ótimo local do espaço por ele explorado.

2.5.3 Representação e Recombinação

O êxito do algoritmo memético depende diretamente da representação genética utilizada para descrever as soluções na forma de cromossomos. A literatura recomenda a utilização de representações que sejam compactas, completas e estáveis. Entende-se por representação compacta aquela que utiliza o menor número possível de variáveis para

representar de forma única uma solução; representação completa, aquela que apresenta todas as soluções possíveis para o problema, inclusive a solução ótima; representação estável, aquela que, com pequenas mudanças no cromossomo, implica pequenos ajustes de adaptabilidade. Caso essas mudanças exijam uma gama significativa de adaptabilidade, poderão ocorrer problemas na evolução da população, tendo em vista que informações importantes aprendidas durante o processo evolutivo estão sendo continuamente perdidas.

A seleção é responsável por escolher pais dentre a população para realizar a recombinação e gerar um filho. Esse filho gerado passa por um aprendizado através de uma busca local, um mecanismo de exploração da vizinhança. Depois que o filho evolui é acrescentado à população original somente se não existirem indivíduos semelhantes a ele, ou seja, se não houver indivíduos com o mesmo valor de função objetivo. Sempre que um indivíduo entra na população, outro sai para manter o tamanho da população fixo. Esse critério para incluir indivíduos na população serve para garantir a sua diversidade.

2.5.4 Mutação

A mutação consiste em inserir ruído numa solução e, com isso, proporcionar diversidade à população de indivíduos. Geralmente, ocorre com uma pequena probabilidade, já que ruído em excesso pode prejudicar a convergência do algoritmo, e consiste em pequena mudança em parte, ou partes, do código genético do indivíduo. Essa natureza puramente aleatória faz a mutação apresentar características destrutivas, gerando indivíduos piores ou não viáveis. Indivíduos assim são eliminados pelos mecanismos de seleção natural. A mutação bem-sucedida pode originar características excelentes ou, mesmo, fazer um indivíduo saltar de um mínimo local ou melhorar sua adaptabilidade, criando uma nova solução, que seria virtualmente inacessível via operação de recombinação.

Tin Junior (2001) afirma que, quando um indivíduo é escolhido para sofrer mutação, uma cópia dele é feita e o operador de mutação é aplicado. A cópia que sofreu

mutação passa pelo mesmo processo de aprendizado através da busca local. Depois que o indivíduo atingiu o seu ótimo local, é acrescentado à população no lugar do original, que deu origem à solução alterada, somente se não existirem indivíduos com o mesmo valor de função objetivo.

2.5.5 Inserção de Novos Indivíduos

A inserção de novos indivíduos na população é outra etapa considerada crítica na dinâmica do algoritmo memético. Em geral, na literatura encontram-se duas estratégias. A primeira sugere uma percentagem fixa dos novos indivíduos criados, independentemente do *fitness* obtido. A parte relevante dessa política é que mantém a diversidade da população por mais tempo, porém, dado que a população em geral tem tamanho fixo, pode ocorrer a situação de um bom indivíduo preexistente ser substituído por um novo indivíduo muito pior. Isso pode ser parcialmente resolvido por meio de um ajuste na percentagem de indivíduos aceitos para valores baixos e/ou substituindo os indivíduos na seqüência do pior para o melhor, o que caracteriza uma política elitista, preservando sempre o melhor indivíduo.

A segunda política também define percentagem dos novos indivíduos. No entanto, a escolha de quais serão aceitos e quais serão descartados segue o critério da probabilidade, a qual é proporcional ao *fitness* de cada indivíduo criado. Essa é uma representação com características que se aproximam mais da realidade que ocorre na natureza.

3 ESTIMAÇÃO DE PARÂMETROS UTILIZANDO ALGORITMOS MEMÉTICOS

3.1 Introdução

O processo de identificação de sistemas acontece por meio de várias etapas, conforme visto no capítulo 1. Tendo definido quais serão a representação matemática (modelo) e a estrutura do modelo, um dos passos seguintes é a determinação de parâmetros. Considerando que toda a massa de dados está disponível antes de se começar a estimar parâmetros (processo *off-line*), o objetivo é garantir que o modelo escolhido reproduza fielmente características previamente conhecidas. Técnicas recursivas são muito úteis para a realização desta etapa pela possibilidade da estimação dos parâmetros à medida que os dados do processo são disponibilizados (processo *on-line*).

Várias técnicas vêm sendo usadas para estimar parâmetros de modelos NARX e NARMAX polinomial (BILLINGS et al., 1998; CORRÊA et al., 2000). Em geral, todas as técnicas constantes na literatura utilizam alguma variação do método dos mínimos quadrados. A utilização de tais métodos tem como principal vantagem a possibilidade de combinar estimação de parâmetros com escolha de estrutura. Como alternativa a este método, este capítulo apresenta a metodologia de um algoritmo memético usado para a estimação de parâmetros na identificação de sistemas não lineares, invariantes no tempo e do tipo NARX.

3.2 Descrição do Problema

Para se ter uma boa estimação de parâmetros é necessário que o sinal de entrada do sistema seja persistentemente excitante. Um sinal branco tem potência espectral numa ampla faixa de freqüência, isto é, excita o sistema em todas as faixas de freqüência (ruído branco). Para que isso ocorra é necessário um número ilimitado de pontos do sinal de entrada (AGUIRRE, 2004).

Pela impossibilidade prática de se trabalhar com um número ilimitado de pontos, os sistemas são geralmente excitados com sinais de tamanho limitado e normalmente com certa aleatoriedade (ruído colorido). Um dos métodos mais conhecidos e usados na estimação de parâmetros é o dos mínimos quadrados. Contudo, este método apresenta polarização em razão de o sinal de entrada não ser um ruído branco, ou seja, o sinal não possui uma potência espectral em todas as freqüências.

Para contornar esse problema, é sugerido neste trabalho um algoritmo de estimação de parâmetros baseado na técnica evolutiva dos algoritmos meméticos, os quais apresentam bons resultados em trabalhos envolvendo sinais estocásticos e determinísticos. Posto isso, deseja-se encontrar um modelo que apresente a saída estimada tão próxima quanto possível da saída real do sistema e com o erro de saída no tempo minimizado satisfatoriamente.

3.3 Modelo Matemático

Um modelo NARX, para um sistema dinâmico com uma entrada e uma saída, conforme foi visto no capítulo 1, pode ser expresso na forma da equação (3.1),

$$\hat{y}(k) = f[y(k-1),...,y(k-n_y); u(k-d), ...,u(k-n_u)] = y(k) - e(k) \qquad (3.1)$$

onde $u(k)$, $y(k)$ e $e(k)$ são a entrada, a saída e erro do sistema para o instante de tempo k; n_y, n_u e d denotam os maiores atrasos em y, em u e o tempo morto, respectivamente; f é uma função não linear descrevendo o comportamento dinâmico do sistema; $\hat{y}(k)$

corresponde à estimativa da saída do sistema em qualquer instante k e $e(k)$ é o erro calculado entre a saída real do sistema e o saída estimada pelo modelo. Sendo o sistema estável, todos os sinais utilizados no procedimento de identificação são finitos.

Este trabalho está voltado para modelos discretos, no qual a estrutura NARX é descrita a seguir:

Vetor regressor:

$$\varphi(k) = [y(k - 1), \ldots, y(k - n_a), u(t - n_m), \ldots, u(k - n_b - n_m + 1)]^T \tag{3.2}$$

onde $\varphi(k)$ representa as saídas do sistema em função dos valores prévios dos sinais de saída e de entrada.

Preditor:

$$\hat{y}(k|\theta) = \hat{y}(k|k - 1, \theta) = g(\varphi(k), \theta) \tag{3.3}$$

onde θ representa o vetor de parâmetros reais do sistema.

Para estimar parâmetros é conveniente colocar o modelo na forma

$$\hat{y}(k) = \varphi(k) \cdot \hat{\theta} \tag{3.4}$$

onde $\hat{\theta}$ representa o vetor de parâmetros estimados pelo modelo.

No caso onde há mais de uma entrada ou saída a representação utilizada é dada por

$$y(k) = \psi^T(k - 1)\hat{\theta} + \xi(k) \tag{3.5}$$

onde

$$y(k) = \begin{bmatrix} y_1(k) \\ y_2(k) \\ M \\ y_m(k) \end{bmatrix}, \quad u(k) = \begin{bmatrix} u_1(k) \\ u_2(k) \\ M \\ u_r(k) \end{bmatrix} \tag{3.6}$$

são vetores de saída e entrada, respectivamente, m é o número de saídas, r é o número de entradas; $\psi(k - 1)$ é o vetor de regressores, possibilitando combinações lineares e não lineares dos sinais $u(k)$ e $y(k)$ e do vetor de resíduos $\xi(k)$ até o instante $(k - 1)$.

Pretende-se estimar, através dos algoritmos meméticos, o vetor de parâmetros $\hat{\theta}$, de maneira tal que a saída estimada $\hat{y}(k)$ seja o mais próximo possível da saída real $y(k)$, sendo a saída estimada dada por:

$$\hat{y}(k) = \Psi(k) \cdot \hat{\theta}. \tag{3.7}$$

3.4 Descrição do Algoritmo Memético

A seguir é descrita a estrutura geral dos algoritmos memético e genético aplicado neste trabalho a partir do fluxograma apresentado na Figura 2.3. O algoritmo genético básico é composto por cinco fases principais: inicialização da população, avaliação, seleção, cruzamento e mutação. A diferença entre algoritmos genéticos e meméticos está apenas na aplicação da busca local, cuja estratégia de aplicação varia de um problema para outro e será descrita posteriormente neste capítulo. Também existem alguns procedimentos, tais como a representação dos cromossomos, tamanho da população, inserção de novos indivíduos e o critério de parada, que são indispensáveis na programação.

3.4.1 Representação dos Cromossomos

Nos problemas de estimação de parâmetros a representação cromossômica, possível solução do problema, pode ser binária ou por ponto flutuante (número pertencente aos reais). Entretanto, a representação real apresenta dois fatores favoráveis em relação à binária.

1. *Cobertura:* a completa cobertura das possíveis soluções dentro do espaço de busca do problema.

2. *Avaliação:* a avaliação é realizada na mesma base, isto é, não há necessidade de mudança na base quando os valores gerados passarem pelo processo de avaliação, operação indispensável quando os cromossomos são gerados na forma binária. Com isso há uma redução no tempo de processamento e na complexidade do algoritmo. Trabalhos comparativos entre a representação binária e ponto flutuante foram feitos por Michalewicz (1999).

A partir de orientações constantes na literatura, optou-se pela representação decimal ou por ponto flutuante, na qual cada cromossomo é representado por um vetor com a quantidade de elementos igual ao número de parâmetros estimados.

A estrutura utilizada neste trabalho para representar a população de indivíduos foi a forma matricial, na qual os cromossomos ou indivíduos estão dispostos em linhas e os parâmetros estimados ou genes estão em colunas. As dimensões da matriz são definidas por *P* linhas, que representam o tamanho da população, e *NP* colunas, que é determinado pelo número de parâmetros. A Tabela 3.1 mostra um exemplo de uma população com cinco indivíduos de quatro parâmetros cada um, com valores no intervalo [-2 2].

Tabela 3.1 – População de cromossomos na forma matricial.

$$Gene = Par(i,\ j) = \begin{bmatrix} 1{,}39 & -0{,}08 & 1{,}05 & 0{,}91 \\ -1{,}01 & 1{,}56 & 0{,}77 & -0{,}77 \\ 0{,}05 & 1{,}39 & -0{,}99 & -1{,}82 \\ -1{,}45 & -0{,}19 & 1{,}30 & 0{,}45 \\ 0{,}70 & 0{,}92 & -0{,}89 & 1{,}72 \end{bmatrix},$$

onde cada vetor linha corresponde a uma solução (cromossomo) e cada componente deste vetor (parâmetro) corresponde a um gene.

A Tabela 3.2 mostra um exemplo de uma população genérica *m* indivíduos de *n* parâmetros cada um.

Tabela 3.2 – População genérica de cromossomos na forma matricial.

$$Gene = Par(i, j) = \begin{bmatrix} x_{11} & x_{12} & \Lambda & x_{1n} \\ x_{21} & x_{22} & \Lambda & x_{2n} \\ M & M & O & M \\ x_{m1} & x_{m2} & \Lambda & x_{mn} \end{bmatrix},$$

com $i = 1, 2, 3, \ldots, m$ e $j = 1, 2, 3, \ldots, n$.

3.4.2 Tamanho da População

Na literatura não há um consenso quanto ao tamanho ideal da população. Grandes populações proporcionam uma cobertura maior no espaço de busca do problema, aumentando também o tempo de processamento. Já pequenas populações tendem a não proporcionar total cobertura no espaço de busca e, dependendo do problema, elevar excessivamente o tempo de processamento para a obtenção da solução.

Salienta-se a necessidade de aumentar o número de indivíduos (cromossomos) com o aumento do número de parâmetros a serem estimados.

No presente trabalho foi utilizada a seguinte estratégia: geram-se aleatoriamente dois mil cromossomos e calcula-se o *fitness* selecionando os oitenta melhores. Estes são os cromossomos utilizados no processo iterativo, mostrado na Figura 2.3

3.4.3 Inicialização da População

Nos problemas abordados, os indivíduos iniciais são todos gerados aleatoriamente dentro do intervalo de busca do problema por meio da equação (3.8),

$$Par(i, j) = LI + r(LS - LI), \tag{3.8}$$

onde *i* varia de um até o número máximo de cromossomos (tamanho da população); *j* varia entre um e o número total de parâmetros; *LI* e *LS* são os limites inferiores e superiores, respectivamente, para o domínio ou espaço de busca do problema; *r* é um número aleatório gerado no intervalo entre [0 1] e *Par* é o valor inicial para o parâmetro *j* do cromossomo *i*.

Em alguns casos, a aleatoriedade é tendenciosa, requerendo mecanismo especial na geração dos valores. Uma forma de reduzir o espaço de busca do problema consiste em normalizar os parâmetros, estratégia que foi adotada neste trabalho, utilizando-se para divisor comum de todos os dados o maior valor absoluto entre a entrada *u*, a saída *y*, o número de atrasos na entrada *u* e o número de atrasos na saída *y*. Com isso, tem-se uma região viável para os parâmetros no intervalo [-1,1].

3.4.4 Função de Avaliação

Uma das características do algoritmo memético utilizado é que para todos os problemas adotou-se uma população hierarquicamente estruturada, na qual os indivíduos são classificados de acordo com sua qualidade. Uma função de avaliação bastante utilizada na bibliografia é dada pela equação (3.9), que representa o erro médio quadrático entre a saída estimada e a saída real do sistema.

$$F_i = \frac{1}{N}\sqrt{\sum_{t=1}^{N}(y(t) - \hat{y}(t))^2} \tag{3.9}$$

onde F_i representa a aptidão em função do erro estimado; $y(t)$, a saída real do sistema; $\hat{y}(t)$, a saída estimada e N, o número total de amostras.

Com o objetivo de melhorar a correlação entre a saída real e estimada, foi introduzido um termo na equação (3.9), que calcula o coeficiente de correlação entre as saídas. Para se chegar à função de avaliação mostrada na equação (3.10), aproveitou-se a sugestão feita em Fávaro (1999) com algumas alterações.

$$G_i = \frac{\frac{1}{N}\sqrt{\sum_{t=1}^{N}(y(t)-\hat{y}(t))^2} \quad + \quad |1-C|}{2} \tag{3.10}$$

onde *C* é o coeficiente de correlação entre a saída real do sistema e a estimada pelo modelo.

Com o acréscimo da segunda parcela houve uma melhora na homogeneidade nas respostas encontradas pelo algoritmo estimador dos parâmetros, encontrando-se valores semelhantes nas várias simulações feitas.

3.4.5 Método de Seleção

A seleção para cruzamento é uma das fases mais críticas na dinâmica do algoritmo, pois uma seleção mal-elaborada tem grande chance de originar indivíduos ruins, independentemente da estratégia de cruzamento utilizada. O processo de seleção usa o valor de *fitness* de cada indivíduo, por meio da função de avaliação, para escolher os melhores cromossomos e gerar a população seguinte.

Dentre as várias estratégias de seleção disponíveis na literatura, neste trabalho foi escolhida a seleção por *ranking* geométrico normalizado, cuja seleção dos indivíduos é realizada aplicando-se a equação (3.11),

$$p(i) = \frac{q(1-q)^{rank(i)-1}}{1-(1-q)^{P}} \tag{3.11}$$

onde *q* representa a taxa unitária do cromossomo com maior aptidão; *P,* o tamanho da população; *rank*(*i*), a posição do cromossomo ao ser classificado em ordem decrescente de *fitness* e *p*(*i*) é a probabilidade de seleção do cromossomo *i*. É importante ressaltar que a equação (3.11) não usa o valor de *fitness* de cada indivíduo, e, sim, a posição relativa deste indivíduo dentro da população quando ordenado pelo valor da aptidão. Isso significa que a primeira posição associada ao *rank*(*1*) terá a maior probabilidade de seleção, independentemente do valor de *fitness* encontrado pela função de avaliação.

A função de avaliação utilizada neste trabalho encontra o erro entre a saída real do sistema e a estimada pelo modelo, e o valor de *fitness* de cada indivíduo está associado a esse erro; logo, o objetivo é a minimização da função de avaliação. Portanto, deve-se vincular a maior probabilidade de seleção ao indivíduo que possui o menor *fitness*, caracterizando a estimação de parâmetros como um problema de minimização.

Os algoritmos meméticos são estruturados para a maximização de problemas, e sua alteração para minimização foi feita por meio de um artifício na seleção, na qual os cromossomos foram enfileirados em ordem crescente de *fitness*, fazendo-se com que a maior probabilidade de seleção seja associada ao indivíduo de menor erro.

3.4.6 Cruzamento

A operação de cruzamento é a que permite a troca genética entre os cromossomos e sua importância na evolução da população aumenta nas gerações finais. Neste momento há um grande número de cromossomos próximos do ponto ótimo, sendo necessária a troca de informações para explorar novas regiões pela combinação entre os indivíduos.

Os dois métodos de cruzamento utilizados neste trabalho são descritos a seguir, juntamente com a freqüência de aplicação:

- *Cruzamento aritmético* – os cromossomos da população que devem ser submetidos ao cruzamento aritmético são selecionados através do percentual de *crossover* (*PCA*). Neste trabalho, utilizou-se o percentual *PCA* de 40% ou 0,4. O cruzamento entre cada parâmetro (gene) dos cromossomos selecionados é feito pela equação (3.12) e (3.13), onde x e y são os filhos dos cromossomos X e Y e r é um número aleatório no intervalo [0 1].

$$x = r.X + (1 - r)Y \tag{3.12}$$

$$y = r.Y + (1 - r)X \tag{3.13}$$

Observa-se nas equações anteriores que os filhos são gerados na região entre os genitores, fazendo a população tender para o centro do espaço de busca. No início das gerações os cromossomos estão dispersos dentro do espaço de busca do problema, provocando a geração de filhos, pelo cruzamento aritmético, distante dos pais. Porém, no final das gerações os cromossomos estão próximos um dos outros e, com isso, os filhos também serão gerados próximos aos pais. Como os filhos substituem os pais na geração seguinte, isso pode provocar a perda de material genético de bons cromossomos.

- *Cruzamento aritmético com os extremos* – este operador usa o mesmo procedimento que o cruzamento aritmético, no entanto utiliza os extremos do intervalo de busca do problema para cruzar com os cromossomos selecionados. Este operador foi introduzido para compensar o efeito de centralização da população provocado pelo cruzamento aritmético. A freqüência de aplicação é dada por *PCE*, definido em 35% ou 0,35. A equação (3.14) mostra como é encontrado o descendente do cromossomo selecionado.

$$x = \begin{cases} r_1.X + (1 - r_1)a & se \quad r_2 \leq 0{,}5 \\ r_1.X + (1 - r_1)b & se \quad r_2 > 0{,}5 \end{cases} \tag{3.14}$$

Onde r_1 e r_2 são números aleatórios gerados no intervalo [0 1], que definem a distância entre o cromossomo pai e filho e o extremo a ser cruzado, respectivamente. Optou-se pelo cruzamento aritmético com os extremos, ao invés do cruzamento heurístico, porque sempre gera descendente dentro do espaço de busca do problema, ao passo que no cruzamento heurístico é possível ter filhos fora desse domínio.

Observa-se que a soma dos percentuais de cruzamento aplicados na população supera 100%, o que implica dizer que toda a população seria submetida a algum tipo de cruzamento, gerando a população seguinte completamente diferente da atual. Porém, nem toda a população é substituída, pois para cada operador de cruzamento é gerado um novo vetor aleatório *V*. Isso possibilita que um mesmo indivíduo seja selecionado mais de uma vez para o cruzamento e outros não sejam escolhidos nenhuma vez. Além disso, o

método de seleção geométrico normalizado favorece a geração de cópias dos melhores cromossomos que podem passar para a geração seguinte sem sofrer cruzamento ou mutação. Contudo, para não haver dúvidas quanto à reprodução do melhor cromossomo para as gerações seguintes, foi utilizada a técnica elitista, na qual o melhor cromossomo da população é guardado para assumir o lugar do pior indivíduo na geração seguinte.

3.4.7 Mutação

A mutação tem papel fundamental na evolução e consiste em mudança aleatória de pares ou partes do código genético do indivíduo, procurando manter a diversidade da população. Sua importância é maior nas primeiras gerações, quando o algoritmo ainda não encontrou os melhores cromossomos e tem muito espaço para procurar. Quando bem-sucedida, a mutação pode originar indivíduos com características excelentes, ou mesmo fazer o indivíduo saltar de mínimos locais, facultando melhorar sua adaptabilidade. Uma população está presa em um mínimo local quando está enfrentando dificuldades para obter soluções melhores que a atual. Um bom movimento de mutação eliminará essa situação, criando uma nova solução, que seria virtualmente inacessível apenas por cruzamento.

A seleção dos parâmetros que devem sofrer mutação é feita com o auxílio de uma matriz, composta de elementos $r(i,j)$ gerados aleatoriamente no intervalo [0 1], com ordem P x NP, de forma que cada valor $r(i,j)$ corresponda a um parâmetro da população, conforme item 3.4.1 e Tabela 3.1. A escolha dos parâmetros ocorre com a comparação dos elementos $r(i,j)$ dessa matriz e a probabilidade de mutação, sendo selecionado para a mutação o parâmetro associado ao elemento da matriz aleatória, cujo valor é menor ou igual à probabilidade de mutação definida.

Os operadores de mutação usados neste trabalho foram de mutação uniforme e mutação não-uniforme, que são descritos a seguir juntamente com as probabilidades de aplicação.

- *Mutação uniforme* – a mutação uniforme permite que o parâmetro selecionado através da probabilidade de mutação *PM* seja substituído por outro valor qualquer

dentro do espaço de busca do problema, gerando diversidade à população. A probabilidade de mutação uniforme deve ser baixa, razão por que foi utilizado neste trabalho o valor $PM = 0{,}01$, correspondente a 1%.

- *Mutação não-uniforme* – neste operador ocorre a redução da região viável para os cromossomos gerados aleatoriamente em torno do indivíduo selecionado para a mutação no decorrer das gerações. Essa região é definida pela equação (3.15),

$$x = \begin{cases} X + (b - X)(r_2(1 - \frac{g}{G}))^B & \text{se} \quad r_1 < 0{,}5 \\ X - (X - a)(r_2(1 - \frac{g}{G}))^B & \text{se} \quad r_1 \geq 0{,}5 \end{cases} \tag{3.15}$$

onde r_1 e r_2 são números aleatórios uniformemente distribuídos entre [0 1]; a e b são os limites inferior e superior para o domínio de busca do problema; g é a geração atual; G, o número máximo de gerações e B, um parâmetro que define a curvatura limite da região factível para x, cujo valor usado foi 1.879. A freqüência de aplicação do operador de mutação não-uniforme também é ajustável no decorrer das gerações, conforme a equação (3.16). Uma mudança inserida neste trabalho é a introdução da probabilidade variável, com a qual é possível definir o valor inicial p_i e final p_f no processo de mutação. Com isso, a probabilidade de mutação não-uniforme *pmnu1* inicia com valor $p_i = 0{,}05$ (5%) e aumenta para *pmnu2* com valor $p_f = 0{,}7$ (70%) no fim das gerações.

$$pmnu = p_f - (p_f - p_i)\left(1 - \frac{g}{G}\right)^B \tag{3.16}$$

A combinação entre a redução do espaço viável para os cromossomos e o aumento da probabilidade de mutação no final do processamento permite o refinamento dos valores encontrados, provocando uma busca local em torno dos melhores pontos da população.

3.4.8 Busca Local

Os algoritmos evolutivos normalmente não incluem métodos para identificar e transferir conhecimento adicional visando auxiliar na implementação de processos de busca local. Um aspecto a ser considerado é o valor atual do *fitness* de um dado indivíduo da população; outro é o potencial de aumento do *fitness* a partir de variações localizadas e determinísticas nos atributos do indivíduo. Caso se saiba o que pode ser feito para aumentar o *fitness* de um indivíduo, via uma busca local determinística, a inclusão desta etapa num processo evolutivo pode representar ganhos de desempenho significativos ao longo das gerações de um processo evolutivo. É evidente que o custo computacional da busca local não pode ser excessivo, principalmente quando aplicada a cada indivíduo da população e a cada geração.

Moscato (1989) introduziu os algoritmos meméticos, que foram, inicialmente, comparados à evolução das artes marciais, em particular, o *kung-fu*. Estudos do comportamento humano mostraram que, como os outros primatas, os seres humanos tendem a lutar usando uma seqüência de movimentos muito desordenada; ao contrário, os movimentos de um mestre de *kung-fu* são uma extraordinária combinação de simplicidade
e efetividade. Até onde se sabe, todas as artes marciais exploram a habilidade do cérebro humano de associar eventos seqüenciais. Assim, o conhecimento básico é transmitido pelo aprendizado de um conjunto de seqüências selecionadas de movimentos, denominadas "formas". A forma, como um cromossomo, não é uma entidade indivisível, sendo composta de uma seqüência de subunidades defensivas e agressivas que também podem ser divididas. Todavia, dentro da forma há alguns movimentos que podem ser entendidos como unidades indivisíveis, os quais são os realmente importantes. São esses movimentos não decomponíveis que devem ser considerados memes (MOSCATO, 1989).

Os indivíduos podem calcular a sua função de *fitness* pela avaliação de seu desempenho na execução dos movimentos das formas e em competições ou torneios. É interessante verificar que a informação melhora através das gerações. É muito importante lembrar que nem todos os indivíduos podem ensinar, somente aqueles que têm a faixa preta. Isso se equipara aos processos de cruzamento nos algoritmos evolutivos, que atribuem maior probabilidade de seleção aos indivíduos com maior *fitness*.

Retornando ao contexto da implementação computacional, os algoritmos meméticos são um "casamento" entre uma busca global baseada em populações e uma busca local heurística realizada junto a cada um de seus indivíduos. Goldberg (1989) classifica variações de algoritmos meméticos como "algoritmos genéticos híbridos". Em termos gerais, pode-se descrever um algoritmo memético como a seguir: dada uma representação de um problema de otimização, uma população de indivíduos é criada; o estado de cada um desses indivíduos pode ser aleatoriamente escolhido ou derivar de um certo procedimento de inicialização; após, cada indivíduo realiza uma busca local, utilizando como mecanismo um ótimo local ou apenas aplicar alguns passos que levem a melhoras pontuais; em seguida, cada indivíduo interage com os outros membros da população, interação que pode ser competitiva ou cooperativa. O comportamento cooperativo pode ser compreendido como os mecanismos de *crossover* nos algoritmos evolutivos ou outros tipos de cruzamento que resultam na criação de um novo indivíduo. Mais genericamente, pode-se compreender cooperação como uma troca de informações. A competição pode ser similar aos processos de seleção dos sistemas evolutivos. A busca local e a cooperação (cruzamento, troca de informações) são repetidas até que um critério de parada seja satisfeito (MOSCATO, 1999).

A busca local é a principal diferença entre os algoritmos meméticos implementados e os algoritmos genéticos e sua influência sobre o desempenho do algoritmo é notável; após os testes com os problemas abordados, realizando a busca local a cada dez iterações, é difícil imaginar um método eficiente que não utilize uma busca com base numa vizinhança definida.

3.4.8.1 Método de Newton Inexato

No âmbito da análise da busca da solução não linear através de qualquer forma do método de Newton, há de se considerar a importância da solução aproximada do sistema de equações lineares. Entretanto, as estratégias de solução podem se tornar inviáveis, principalmente pelo esforço computacional e demanda de memória.

Os métodos iterativos de solução são adequados às novas arquiteturas de computadores, onde estão disponíveis facilidades de vetorização e, eventualmente, a

realização de tarefas paralelas no processo de solução. Considerando-se a utilização de métodos iterativos para a solução do sistema de equações lineares produzido durante uma iteração do processo não linear, surge a idéia de ajustar a tolerância da solução linear de acordo com o estágio corrente de convergência da iteração não linear. O desenvolvimento desta idéia e como ela afeta o andamento do processo não linear são a base do método conhecido por *Método de Newton truncado ou inexato*.

Os diversos métodos tipo Newton são baseados na equação da expansão da série convergente de Taylor, onde o cálculo das soluções lineares no início das iterações não lineares não necessita de grande precisão. Esta necessidade aumenta à medida que as iterações não lineares aproximam-se da solução no equilíbrio. A idéia desta variação do método de Newton é minimizar o esforço empregado para a solução do sistema de equações lineares quando a direção de busca não linear estiver longe da solução no equilíbrio.

Considere-se o caso $F(x) = 0$, cuja solução é considerada difícil e vai sendo aproximada por uma seqüência de pontos $\{x_k\}$. Dada cada aproximação x_k, constrói-se com as informações disponíveis nesse ponto um problema mais acessível. A aproximação x_{k+1} é a solução do problema mais acessível. Este muda de uma iteração para a seguinte e, via de regra, sua solução está cada vez mais próxima da solução do problema que inicialmente se considerava difícil.

No problema em estudo neste trabalho, o *k*-ésimo problema resolvido vem de considerar a aproximação de Taylor de primeira ordem de $F(x)$ numa vizinhança do ponto atual x_k :

$$F(x) \approx L_k(x) = F(x_k) + J(x_k)\,(x - x_k). \tag{3.17}$$

onde $J(x_k)$ é a matriz jacobiana.

Seguindo o princípio descrito acima, o ponto seguinte x_{k+1} é uma solução de

$$L_k(x) = 0. \tag{3.18}$$

Se $J(x_k)$ é não singular, (3.17) tem solução única; então, a iteração consiste em resolver um sistema linear:

$$J(x_k)s_k = -F(x_k) \tag{3.19}$$

$$x_{k+1} = x_k + s_k.$$

A implementação de (3.19) pressupõe o cálculo de $J(x_k)$, isto é, a avaliação das derivadas primeiras das funções $f_i(x)$, $i = 1, \ldots, n$. Num passado não muito distante, o cálculo de derivadas era considerado não só difícil, mas também muito suscetível a erros humanos. Atualmente, a possibilidade de falha humana pode ser evitada através das diferenciações simbólicas e automáticas. Em geral, quando se calculam efetivamente as derivadas, muitos cálculos usados na avaliação podem ser reaproveitados. Quando n é muito grande e a estrutura da matriz $J(x)$ não é favorável para uma fatoração LU esparsa, a resolução do sistema linear newtoniano (3.19) por métodos diretos fica impraticável.

Os métodos de Newton inexatos representam um ponto de vista radicalmente diferente. Substituem a resolução proposta em (3.19) por outra mais manejável e que aborda a resolução do sistema linear newtoniano através de métodos iterativos lineares, os quais, como se sabe, são geralmente econômicos em termos de memória e custo computacional, isto é, para resolver

$$J(x_k)s = -F(x_k) \tag{3.20}$$

utiliza-se uma seqüência $s^0, s^1, s^2, \ldots$, produzida por um método iterativo linear, para solução de (3.20) em vez de fatoração LU, no qual os sucessivos iterandos s^λ são calculados com um custo moderado.

3.4.9 Inserção de Novos Indivíduos

A inserção de novos indivíduos na população é outra etapa considerada crítica na dinâmica do algoritmo. A literatura geralmente apresenta duas estratégias para tal. Uma refere-se à aceitação de percentagem fixa de novos indivíduos criados, independentemente do valor obtido na sua avaliação. Esta estratégia, no geral, mantém a diversidade da população por mais tempo, porém, em razão de a população ser fixa, pode ocorrer a substituição de um bom indivíduo preexistente por um indivíduo muito pior.

A segunda política também estabelece que se deva aceitar uma percentagem fixa de novos indivíduos na população também fixa. Todavia, a escolha de quais serão aceitos e quais serão descartados segue uma probabilidade que é proporcional ao valor obtido na avaliação de cada indivíduo criado. Esta é a representação que mais se aproxima da realidade que ocorre na natureza, pois, assim que nascem, os filhos já têm de enfrentar a competição por alimento, além dos predadores naturais. É também a que mais se aproxima da estratégia adotada neste trabalho, levando-se em consideração que os indivíduos somente serão substituídos se apresentarem melhor *fitness*.

3.4.10 Critério de Parada

O critério de parada utilizado nesta dissertação para o algoritmo memético está relacionado com a precisão do melhor cromossomo. Também se adotou como critério de parada o número máximo de iterações para situações com maior esforço computacional.

O resumo dos principais parâmetros aplicados no algoritmo memético utilizado neste trabalho está apresentado na Tabela 3.2. O valor de cada parâmetro foi encontrado empiricamente após diversas simulações realizadas.

Tabela 3.2 – Resumo dos parâmetros aplicados no algoritmo memético

Operadores	Valores
Quantidade de dados	QP=80
Número de gerações	N=500
Tamanho da população	*fam*=80
Seleção geométrica normalizada	q = 0,10
Cruzamento aritmético	PCA = 0,40
Aritmético com extremos	PCE = 0,35
Mutação uniforme	PM = 0,01
Mutação não uniforme	$Pmnu1$ = 0,05

Mutação não uniforme	*pmnu 2* = 0,70

4 RESULTADOS OBTIDOS

4.1 Introdução

Neste capítulo são apresentados resultados de cinco sistemas. Para efetivar a obtenção desses resultados foi utilizado um computador pessoal com um único processador *Móbile Intel Pentium 4*, 3.2 GHz e 512 Mb de memória DDR-RAM e sistema operacional *Windows XP*. O software utilizado para o desenvolvimento dos programas computacionais foi o Matlab 7.0.

A validação dos resultados obtidos pelo algoritmo memético (AM) na estimação de parâmetros no processo de identificação de sistemas foi realizada pela comparação entre as respostas do sistema real e dos modelos no domínio do tempo e, nos casos de sistemas lineares, também da freqüência. Para a entrada dos sistemas em análise, foram utilizados sinais aleatórios e em degrau unitário. A validação dos modelos no domínio da freqüência foi realizada com auxílio do diagrama de Bode da magnitude. A resposta em freqüência foi gerada a partir das funções de transferência discretas do sistema e dos modelos.

As simulações realizadas estão divididas em duas fases: uma onde são utilizados sistemas lineares e outra utilizando sistemas não lineares. Os sistemas lineares utilizados são de segunda, terceira e quarta ordem e foram também utilizados por Gomes (2005). Os sistemas não lineares utilizados provêm da literatura; o sistema com o sinal de entrada

não linear consta em Narendra e Parthasaraty (1990) e o conversor Buck, em Aguirre (2004).

4.2 Sistemas Lineares

As funções de transferências escolhidas são da literatura da área, estando inicialmente representadas no espaço contínuo do tempo; são, então, discretizadas através do software Matlab utilizando-se a função **c2d,** que possibilita a passagem de uma função de transferência em *s* (transformada de Laplace), contínua no tempo, para a forma discreta em z (transformada Z). Finalmente, os sistemas são escritos em forma recursiva utilizando-se o operador de atraso (q).

O sinal de entrada utilizado na estimação é aleatório, gerado através da função **rand** do software Matlab. Sua escolha é motivada por apresentar características semelhantes ao ruído branco. Os sinais de entrada utilizados na validação são: um aleatório e um degrau unitário. O sinal degrau unitário foi utilizado para verificar a capacidade de o modelo ser global (apresentar saída semelhante ao sistema real para sinais diferentes daquele usado na estimação).

4.2.1 Sistema Linear de Segunda Ordem

O sistema de segunda ordem estudado é representado pela função de transferência no tempo contínuo, dado na equação (4.1). Já a equação (4.2) apresenta a função de transferência discretizada pelo Matlab com um tempo de amostragem de 0,1 s. A forma recursiva é apresentada em (4.3), onde é verificada a necessidade de estimação de quatro parâmetros. O sistema em estudo apresenta dois pólos complexos.

$$H(s) = \frac{25}{s^2 + 4s + 25} \tag{4.1}$$

$$H(z) = \frac{0{,}1077z + 0{,}09414}{z^2 - 1{,}469z + 0{,}6703} \tag{4.2}$$

$$y(k) = 1{,}469\,y(k-1) - 0{,}6703\,y(k-2) + 0{,}1077\,u(k-1) + 0{,}09414\,u(k-2) \qquad (4.3)$$

Os resultados obtidos pelo algoritmo genético (AG) do sistema (4.1) são apresentados em (4.4) e do algoritmo memético (AM), em (4.5). Foram encontrados utilizando-se mil pontos (dados de entrada e saída), com uma população de oitenta indivíduos (cromossomos) que se reproduziram por no máximo quinhentas gerações.

$$\hat{\mathrm{H}}(\mathrm{G}) = \frac{0{,}10763\mathrm{z} + 0{,}09422}{\mathrm{z}^2 - 1{,}4679\mathrm{z} + 0{,}66989} \qquad (4.4)$$

$$\hat{\mathrm{H}}(M) = \frac{0{,}10766\mathrm{z} + 0{,}094136}{\mathrm{z}^2 - 1{,}46851\mathrm{z} + 0{,}67032} \qquad (4.5)$$

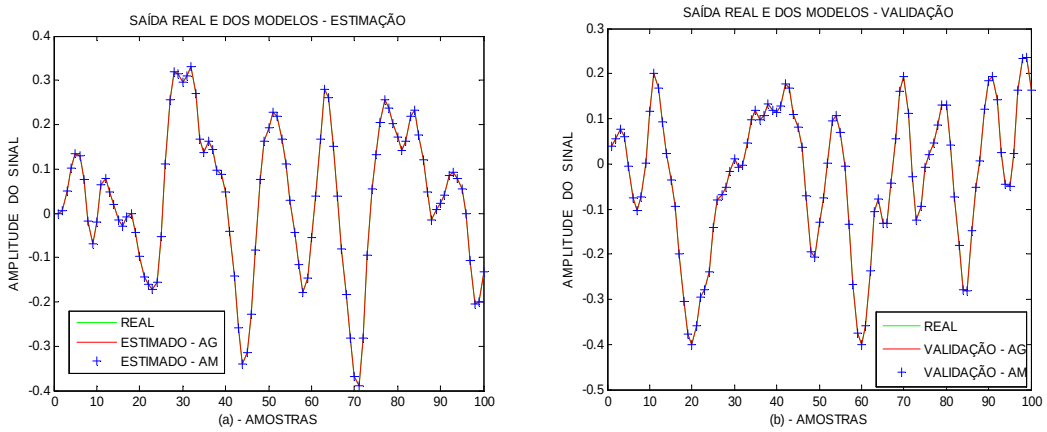


Figura 4.1 – Resposta temporal do sistema e dos modelos: (a) estimação; (b) validação.

A Figura 4.1(a) apresenta a saída do sistema e dos modelos na fase de estimação, utilizando os primeiros mil pontos de dados do grupo gerado inicialmente com dois mil pontos. A segunda parte dos pontos gerados inicialmente é utilizada para validação, cujas saídas do sistema e dos modelos estão representadas na Figura 4.1(b), onde se observa o excelente desempenho dos modelos encontrados, produzindo uma resposta temporal que se sobrepõe à saída real.

As evoluções dos parâmetros do melhor indivíduo no decorrer das gerações no AG e no AM estão apresentadas, respectivamente, nas Figuras 4.2(a e b). Cabe ressaltar a diferença nas Figuras 4.2(a e b), enquanto o trabalho computacional desenvolvido por

quinhentas iterações no AG consegue uma estabilidade com erro na ordem de 10^{-5}; apesar de aparente estabilidade com o número de iterações em torno de cem, o AM com dez iterações atinge erro na ordem de 10^{-9}, estabelecido como critério de parada por ter chegado à resposta global.

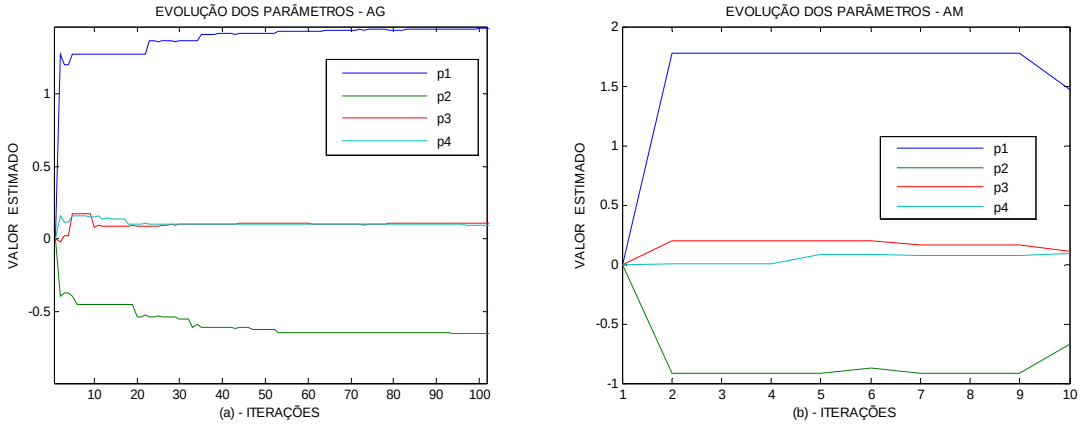


Figura 4.2 – Comportamento dos parâmetros: (a) parâmetros do AG; (b) parâmetros do AM.

Também é interessante observar o comportamento simétrico na evolução dos parâmetros (Figura 4.2) quando no mínimo dois cromossomos são alterados. Este comportamento está relacionado com as possibilidades de combinações entre indivíduos, com valores próximos de *fitness*.

A evolução do *fitness* pode ser observada na Figura 4.3.

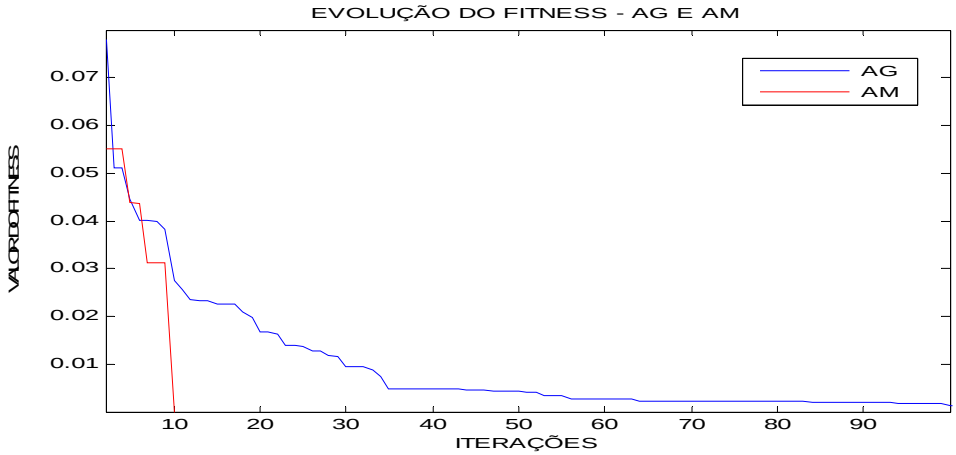


Figura 4.3 – Evolução do *fitness* calculado no AG e no AM.

O comportamento observado na evolução do *fitness* (Figura 4.3) para os dois algoritmos apresentados mostra que o AM satisfaz o critério de parada através da condição de erro, enquanto que o AG satisfaz o critério de parada através do número máximo de iterações. Também é possível uma análise em função do tempo utilizado para realizar o trabalho computacional, onde se constatou que o AM, ao encerrar o processo, utiliza em torno de 3% do tempo utilizado pelo AG.

A Figura 4.4 apresenta a resposta temporal do sistema e dos modelos estimados para um sinal de entrada em degrau e os respectivos diagramas de Bode.

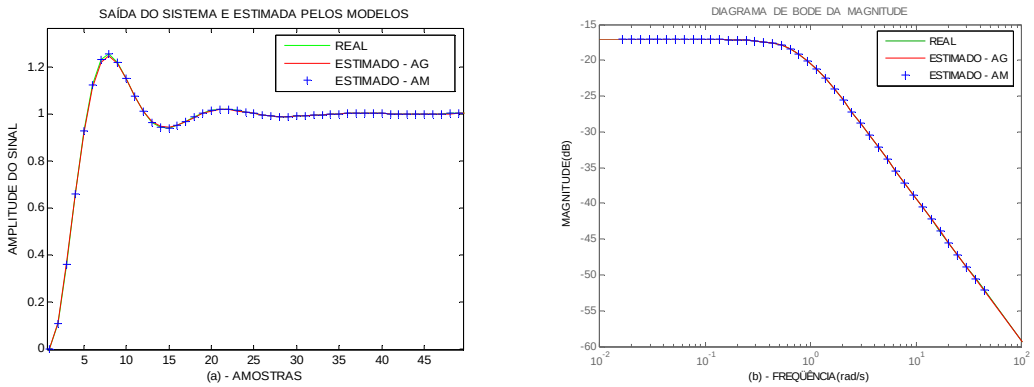


Figura 4.4 – Resposta do sistema e dos modelos: (a) degrau unitário; (b) freqüencial.

Pode-se observar na Figura 4.4 que tanto o AG quanto o AM obtêm excelentes resultados na estimação de parâmetros de sistemas lineares de segunda ordem, tanto na resposta temporal quanto na resposta freqüencial.

4.2.2 Sistema Linear de Terceira Ordem

A função de terceira ordem mostrada pela equação (4.6) é encontrada em Aguirre (2004). Possui dois zeros reais, um pólo real e dois pólos complexos. A discretização é feita com tempo de amostragem de 1s e é apresentada em (4.7). A forma recursiva é vista em (4.8) com a existência de seis parâmetros.

$$H(s) = \frac{0{,}8049s^2 + 0{,}7835s + 0{,}1328}{2{,}45s^3 + 1{,}296s^2 + 0{,}9153s + 0{,}0957} \tag{4.6}$$

$$H(z) = \frac{0{,}3779z^2 - 0{,}4734z + 0{,}1362}{z^3 - 2{,}292z^2 + 1{,}91z - 0{,}5891} \tag{4.7}$$

$$y(k) = 2{,}292y(k-1) - 1{,}91y(k-2) + 0{,}5891y(k-3) + 0{,}3779u(k-1) - 0{,}4734u(k-2) + 0{,}1362u(k-3) \tag{4.8}$$

Os resultados obtidos pelo algoritmo genético (AG) do sistema (4.6) são apresentados em (4.9) e do algoritmo memético (AM), em (4.10). Foram encontrados utilizando-se mil pontos (dados de entrada e saída), com uma população de oitenta indivíduos (cromossomos) que se reproduziram por, no máximo, quinhentas gerações.

$$\hat{H}(G) = \frac{0{,}3780z^2 - 0{,}4432z + 0{,}12624}{z^3 - 2{,}211z^2 + 1{,}8009z - 0{,}54089} \tag{4.9}$$

$$\hat{H}(M) = \frac{0{,}3779z^2 - 0{,}47337z + 0{,}13622}{z^3 - 2{,}2919z^2 + 1{,}9104z - 0{,}58913} \tag{4.10}$$

A Figura 4.5 apresenta a saída do sistema e dos modelos na fase de estimação e de validação.

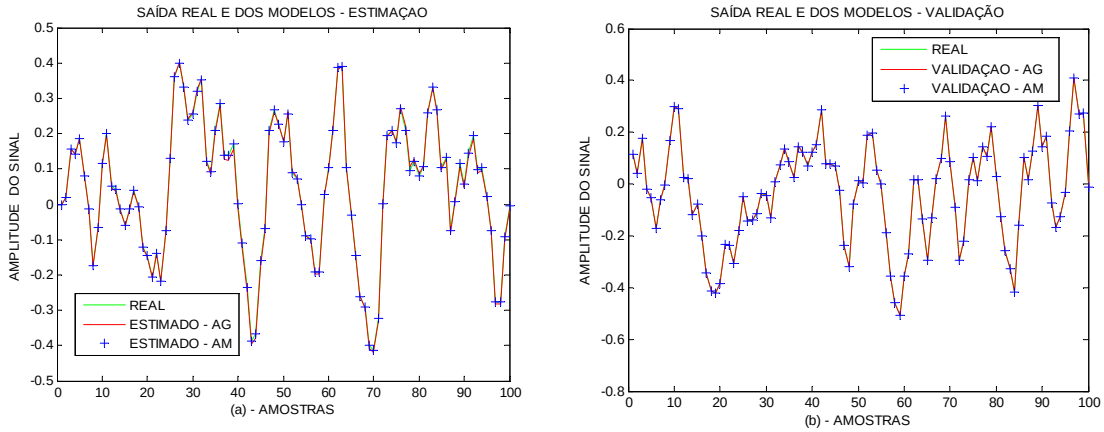


Figura 4.5 – Resposta temporal do sistema e dos modelos: (a) estimação; (b) validação.

A Figura 4.5(a) apresenta a saída do sistema e dos modelos na fase de estimação, utilizando os primeiros mil pontos do grupo gerado inicialmente com dois mil pontos. A segunda parte dos pontos gerados inicialmente é utilizada para validação, cujas saídas do sistema e dos modelos estão representadas na Figura 4.5(b), onde se observa que o

desempenho dos modelos é satisfatório, pois continuam produzindo uma resposta temporal que se sobrepõe à saída real.

A evolução dos parâmetros do melhor indivíduo no decorrer das gerações no AG e no AM está apresentada na Figuras 4.6.

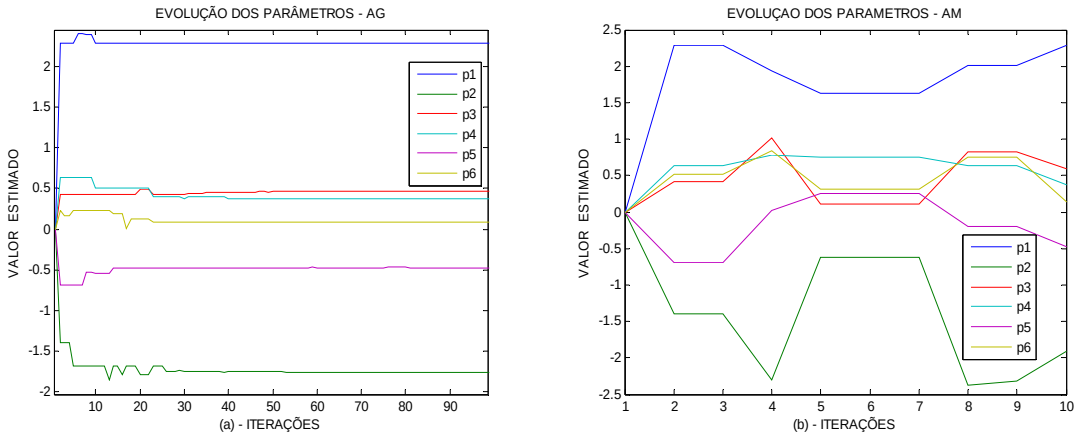


Figura 4.6 – Comportamento dos parâmetros: (a) parâmetros do AG; (b) parâmetros do AM.

Observando-se as figuras 4.6(a) e 4.6(b), percebe-se que, enquanto o trabalho computacional desenvolvido por quinhentas iterações no AG consegue uma estabilidade com erro na ordem de 10^{-3}, o AM encerra o processo computacional com dez iterações, atingindo erro na ordem de 10^{-8}, estabelecido como critério de parada por conseguir a resposta global.

O comportamento observado na evolução do *fitness* (Figura 4.7) confirma a agilidade do AM na busca da solução global, conseguindo convergência e aproximando-se dos valores desejados, ao passo que o AG com o trabalho computacional realizado apenas ameniza o erro. Na análise em função do tempo utilizado para realizar o trabalho computacional, constatou-se que o AM, ao encerrar o processo, utiliza em torno de 3,3% do tempo utilizado pelo AG.

A evolução do *fitness* pode ser observada na Figura 4.7.

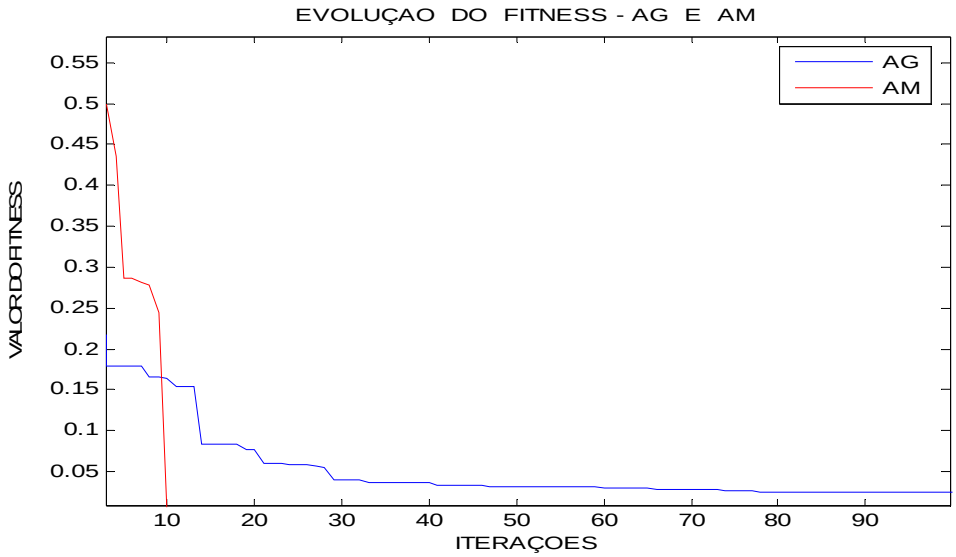


Figura 4.7 – Evolução do *fitness* calculado no AG e no AM.

A Figura 4.8 apresenta a resposta temporal do sistema e dos modelos estimados para um sinal de entrada em degrau e os respectivos diagramas de Bode.

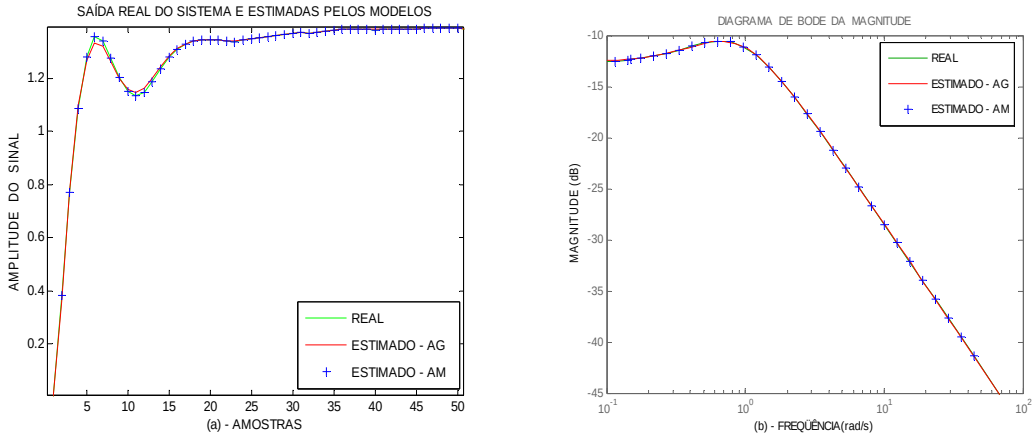


Figura 4.8 – Resposta do sistema e dos modelos: (a) degrau unitário; (b) freqüencial.

Pode-se observar na Figura 4.8 que tanto o AG quanto o AM obtêm excelentes resultados na estimação de parâmetros de sistemas lineares de terceira ordem, tanto na resposta temporal quanto na resposta freqüencial.

4.2.3 Sistema Linear de Quarta Ordem

O sistema de quarta ordem analisado é caracterizado pela função de transferência apresentada na equação (4.11), com oito parâmetros para a estimação. Este sistema possui quatro pólos complexos com um zero real e dois complexos. A função de transferência na forma discreta, com tempo de amostragem de 0,3s, é apresentada na equação (4.12), enquanto a sua forma recursiva está em (4.13).

$$H(s) = \frac{14s^3 + 248s^2 + 900s + 1200}{s^4 + 18s^3 + 102s^2 + 180s + 120} \tag{4.11}$$

$$H(z) = \frac{3{,}318z^3 - 3{,}208z^2 + 0{,}8839z + 0{,}0149}{z^4 - 1{,}543z^3 + 0{,}7385z^2 - 0{,}09885z + 0{,}004517} \tag{4.12}$$

$$\begin{aligned} y(k) = {} & 1{,}543y(k-1) - 0{,}7385y(k-2) + 0{,}09885y(k-3) - 0{,}004517y(k-4) + \\ & 3{,}318u(k-1) - 3{,}208u(k-2) + 0{,}8839u(k-3) + 0{,}0149u(k-4) \end{aligned} \tag{4.13}$$

O conjunto de parâmetros estimados pelo algoritmo genético (AG) do sistema (4.11) é apresentado em (4.14) e do algoritmo memético (AM), em (4.15). Foram encontrados utilizando-se mil pontos (dados de entrada e saída), com uma população de oitenta indivíduos (cromossomos) que se reproduziram por no máximo quinhentas gerações.

$$\hat{\mathrm{H}}(G) = \frac{3{,}3114\mathrm{z}^3 - 3.0482\mathrm{z}^2 + 3{,}2547\mathrm{z} + 0.23396}{\mathrm{z}^4 - 1{,}494\mathrm{z}^3 + 1{,}4229\mathrm{z}^2 - 0{,}45025\mathrm{z} - 0{,}10977} \tag{4.14}$$

$$\hat{\mathrm{H}}(M) = \frac{3{,}3175\ \mathrm{z}^3 - 3.208\ \mathrm{z}^2 + 0{,}8389\ \mathrm{z} + 0.1489}{\mathrm{z}^4 - 1{,}5433\ \mathrm{z}^3 + 0{,}73849\ \mathrm{z}^2 - 0{,}0988\ \mathrm{z} + 0{,}00451} \tag{4.15}$$

A Figura 4.9 apresenta a saída do sistema e dos modelos na fase de estimação e de validação.

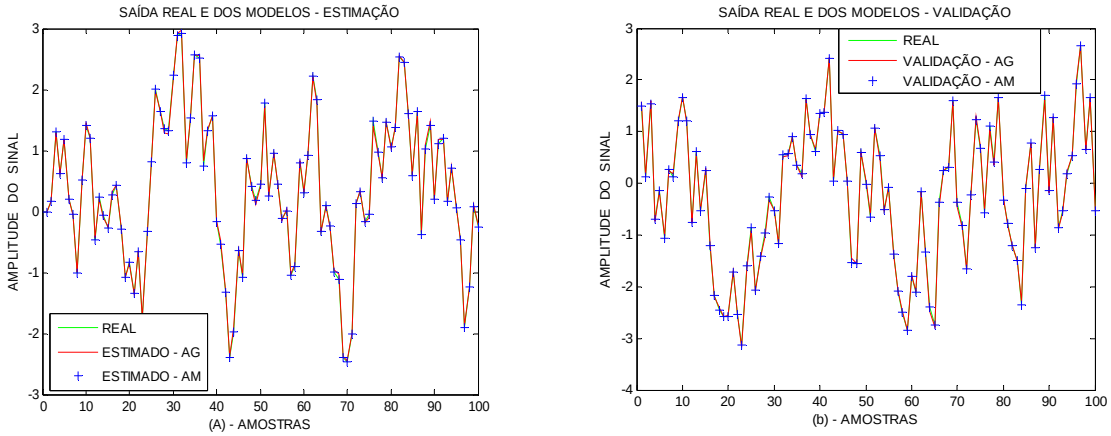


Figura 4.9 – Resposta temporal, do sistema e dos modelos: (a) estimação; (b) validação.

A Figura 4.9(a) apresenta a saída do sistema e dos modelos na fase de estimação, utilizando os primeiros mil pontos do grupo gerado inicialmente com dois mil pontos. A segunda parte dos pontos gerados inicialmente é utilizada para validação, cuja saída do sistema e dos modelos está representada na Figura 4.9(b), onde se observa que o desempenho dos modelos é satisfatório, pois continuam produzindo uma resposta temporal que se sobrepõe à saída real.

A evolução dos parâmetros do melhor indivíduo no decorrer das gerações no AG e no AM está apresentada na Figura 4.10.

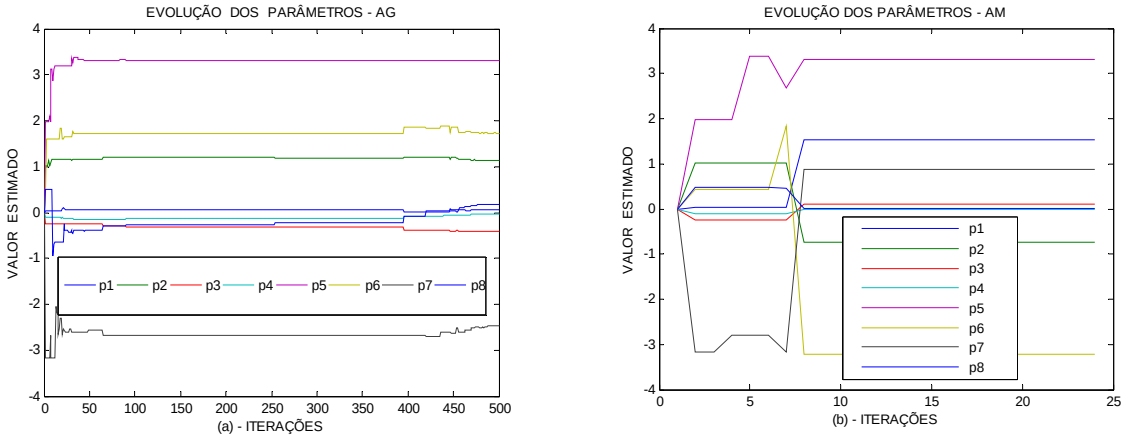


Figura 4.10 – Comportamento dos parâmetros: (a) parâmetros do AG; (b) parâmetros do AM.

Observando-se as figuras 4.10(a) e 4.10(b), percebe-se que no trabalho computacional desenvolvido por quinhentas iterações no AG a estabilidade é obtida, com erro na ordem de 10^{-3}, através de um ponto mínimo local. O AM encerra os procedimentos com dez iterações ao atingir o erro na ordem de 10^{-8}, valor estabelecido como critério de parada.

A evolução do *fitness* poderá ser observada na Figura 4.11.

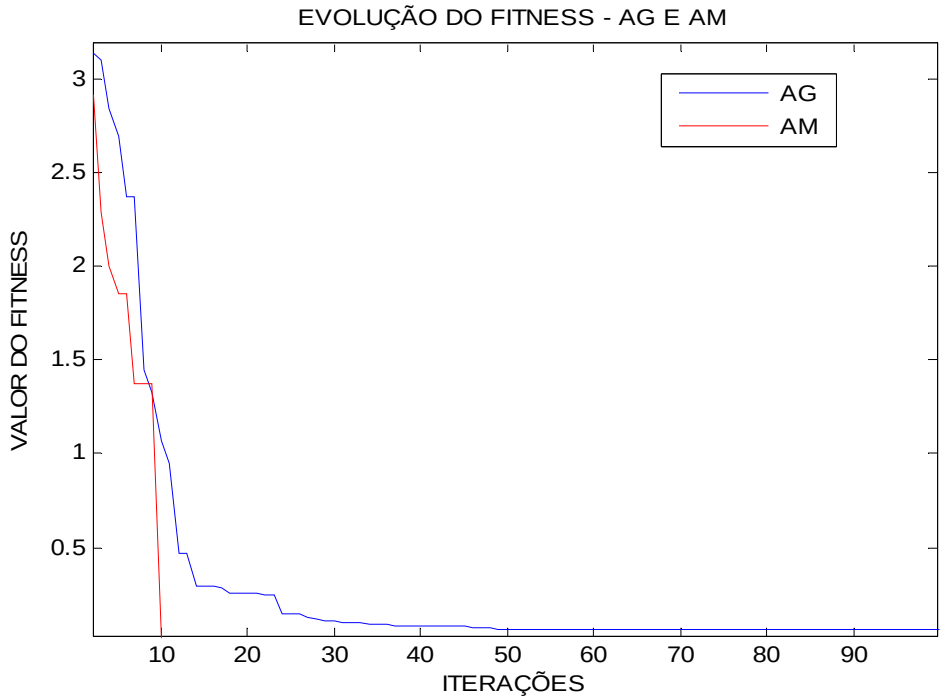


Figura 4.11 – Evolução do *fitness* calculado no AG e no AM.

O comportamento observado na evolução do *fitness*, Figura 4.11, apesar do aumento no número de parâmetros, confirma a agilidade do AM na busca da solução global, ao passo que o AG, com o trabalho computacional realizado, não consegue sair de um mínimo local; o AM consegue a convergência desejada. Também se constatou que o AM, ao encerrar o processo, utiliza em torno de 3,4% do tempo utilizado pelo AG.

A Figura 4.12 apresenta a resposta temporal do sistema e dos modelos estimados para um sinal de entrada em degrau e os respectivos diagramas de Bode.

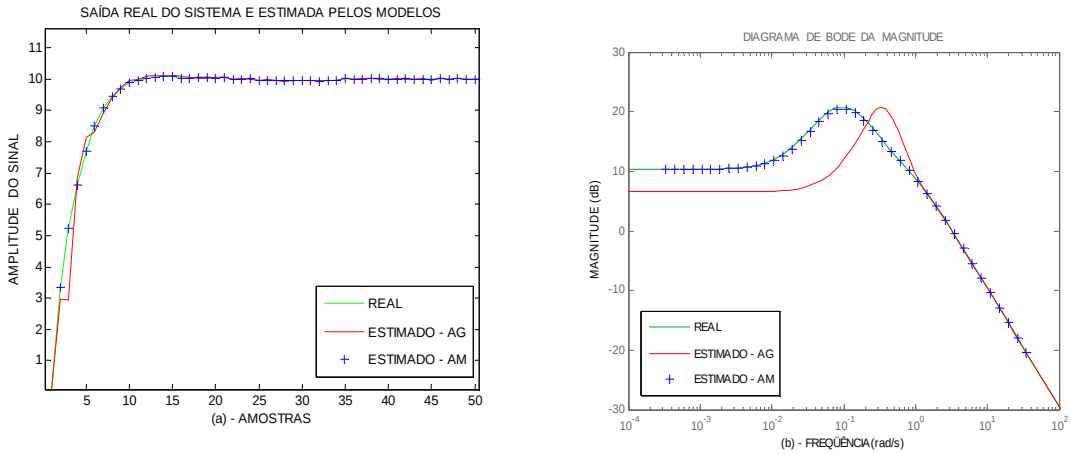


Figura 4.12 – Resposta do sistema e dos modelos: (a) degrau unitário; (b) freqüencial.

Observa-se na Figura 4.12 que o AM apresenta excelentes resultados na estimação de parâmetros de sistemas lineares de quarta ordem, tanto na resposta temporal quanto na resposta freqüencial. Já o AG, por ser um modelo particular, apresenta diferenças nas respostas, em relação ao sistema real, para sinais diferentes dos utilizados na estimação.

4.2.4 Algumas Considerações sobre os Sistemas Lineares

Nos sistemas lineares apresentados, percebe-se que os modelos encontrados representam satisfatoriamente o sistema em regime transitório e em regime permanente. Neste aspecto, cabe ressaltar que no AM, mesmo quando a ordem do sistema em estudo é elevada, com poucas iterações atinge-se um dos critérios de parada, relação do erro entre a saída do sistema e a saída do modelo. A divergência apresentada nos gráficos referentes ao AG em relação à saída real do sistema está associada às estratégias aplicadas ao mesmo. A busca local proporcionou ao AM um excelente desempenho, fazendo com que o modelo acompanhe o comportamento do sistema no intervalo amostrado.

Quanto à resposta na freqüência, apresentada pelo Diagrama de Bode da Magnitude, observa-se que a dificuldade se acentua para o AG, assim como outros métodos, quando se aumenta a ordem do sistema. O AM apresenta resultado satisfatório nos três casos estudados, com um número reduzido de iterações.

O erro, estabelecido como critério de parada, foi reduzido com a elevação do grau do sistema em face das dificuldades associadas ao número de parâmetros que deverão ser estimados pelo algoritmo. Os modelos, quando excitados por um sinal aleatório, apresentam melhor desempenho quando comparados com os resultados obtidos nos mesmos modelos, por meio de entrada com sinal na forma de degrau unitário.

4.3 Sistemas Não Lineares

Nesta seção são apresentados resultados de identificação de dois sistemas não lineares. Os sistemas apresentados podem ser classificados dentre as estruturas de sistemas apresentadas na seção 1.3. Todo sistema é, em princípio, não linear. A dinâmica de sistemas não lineares normalmente depende da amplitude do sinal de entrada, bem como do ponto de operação do sistema. Em torno de um ponto de operação, alguns sistemas não lineares podem ser aproximados por modelos lineares (AGUIRRE, 2004).

4.3.1 Sistema Não Linear com não linearidade na entrada

O sistema não linear com não linearidade na entrada analisado é encontrado em Narendra e Parthasarathy (1990), sendo caracterizado pela equação a diferenças (4.16).

$$y(k) = 0{,}3y(k - 1) + 0{,}6y(k - 2) + f[u(k - 1)] \tag{4.16}$$

A dinâmica desse sistema consta de uma dependência linear de valores passados da saída do próprio sistema y*(k)* e uma dependência não linear com relação à entrada *u(k)* aplicada ao sistema. Essas características correspondem ao modelo I da seção 1.3. A função *f(.)* é não linear, suposta desconhecida, sendo, no citado trabalho, utilizada a função apresentada em (4.17).

$$f(u) = 0{,}6sen(\pi u) + 0{,}3sen(3\pi u) + 0{,}1sen(5\pi u) \tag{4.17}$$

Narendra e Parathasarahy (1990) utilizaram $u = u(k)$ gerado de forma aleatória e com distribuição uniforme no intervalo [-1, 1] e um atraso puro no tempo, para $k = 1, \ldots$

,500. Nas condições iniciais, adotaram $y(1) = y(2) = 0.0$. Então obtiveram o seguinte modelo para *f(.)*.

$$f[u(k)] = u^3(k) + 0{,}3u^2(k) - 0{,}4u(k) \tag{4.18}$$

A equação (4.16) é reescrita na forma recursiva e apresentada em (4.19), tem cinco parâmetros para ser estimados e validados.

$$y(k) = 0{,}3y(k-1) + 0{,}6y(k-2) + u^3 + 0{,}3u^2 - 0{,}4u \tag{4.19}$$

Para a fase de estimação de parâmetros de (4.19), o sinal de entrada utilizado foi

$$u(k) = \begin{cases} sen(2\pi k/250) & 0 \le \mathrm{k} \le 250 \\ sen(2\pi k/250) + sen(2\pi k/25) & 251 \le \mathrm{k} \le 500 \end{cases}$$

O conjunto de parâmetros estimados pelo algoritmo genético (AG) do sistema (4.19) é apresentado em (4.20), do algoritmo memético (AM), em (4.21). Foram encontrados utilizando-se quinhentos pontos (dados de entrada e saída), com uma população de oitenta indivíduos (cromossomos), que se reproduziram por no máximo quinhentas gerações.

$$y(kG) = 0{,}2667y(k-1) + 0{,}6316y(k-2) + 1{,}0207u^3 + 0{,}30708u^2 - 0{,}41557u \tag{4.20}$$

$$y(kM) = 0{,}29998y(k-1) + 0{,}6y(k-2) + u^3 + 0{,}3u^2 - 0{,}4u \tag{4.21}$$

A Figura 4.13 apresenta a saída do sistema e dos modelos para *u(k)* definido acima. A Figura 4.13(a) apresenta o desempenho do AG em relação ao sistema e a Figura 4.13(b) apresenta o desempenho do AM em relação ao sistema. Observando-se as duas figuras, percebe-se que o AM tem excelente performance, sobrepondo-se à saída real do sistema. Constata-se também que o AG mantém as características dinâmicas do sistema, porém não se sobrepõe a ele.

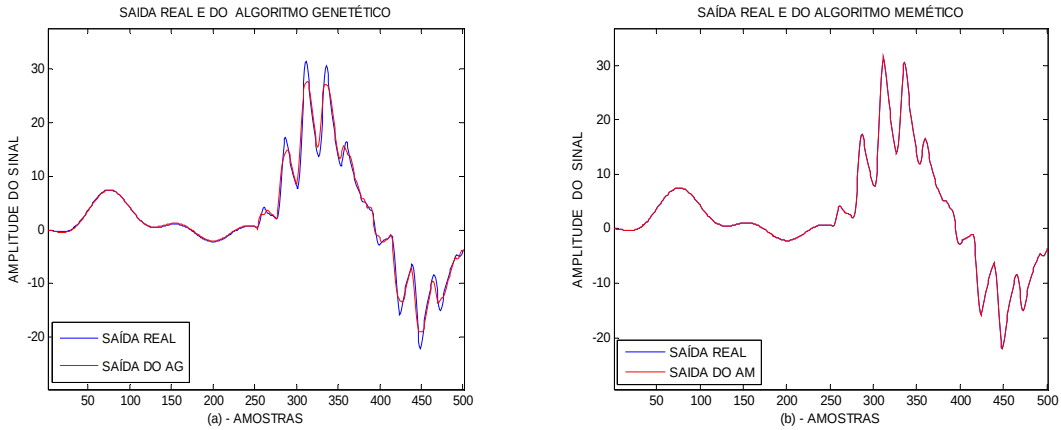


Figura 4.13 – Resposta temporal dos modelos e do sistema: (a) saída real e AG; (b) saída real e AM.

A evolução dos parâmetros do melhor indivíduo no decorrer das gerações no AG e no AM está apresentada na Figura 4.14.

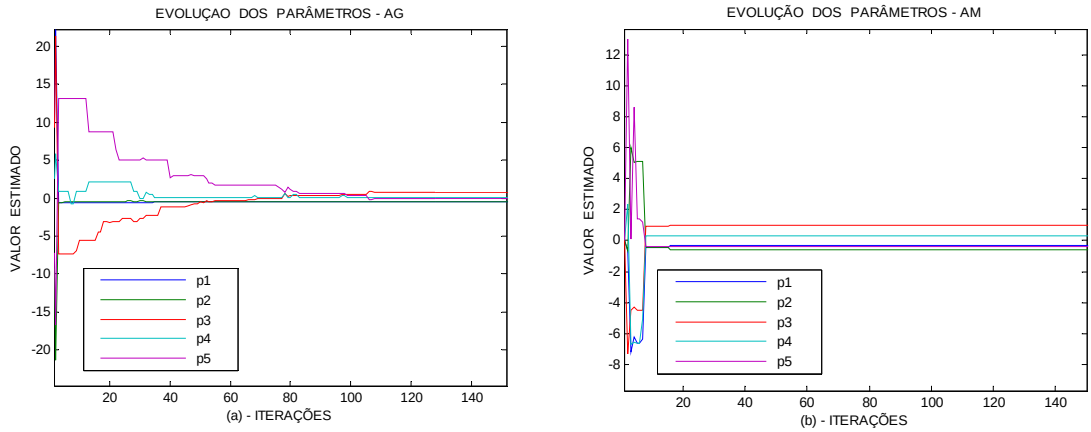


Figura 4.14 – Comportamento dos parâmetros: (a) parâmetros do AG; (b) parâmetros do AM.

Observando-se as Figuras 4.14(a) e 4.14(b), percebe-se que no trabalho computacional desenvolvido por quinhentas iterações no AG a estabilidade obtida, com erro na ordem de 10^{-2}, em torno de cem iterações é um dos pontos de mínimo local, enquanto que o AM estabiliza em torno de vinte iterações, com erro na ordem de 10^{-7}. Na análise em função do tempo utilizado para realizar o trabalho computacional, constatou-se que o AM realiza o trabalho computacional no mesmo tempo que o AG.

A evolução do *fitness* pode ser observada na Figura 4.15.

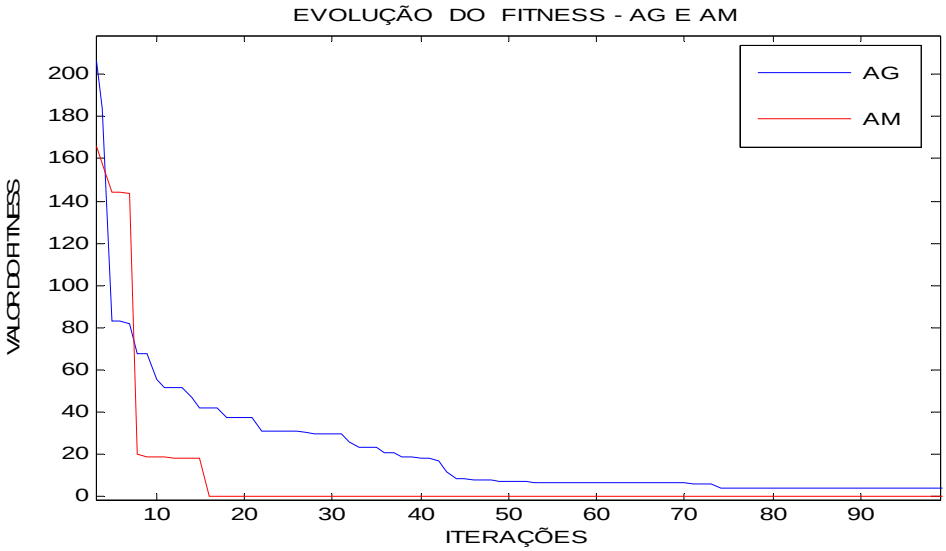


Figura 4.15 – Evolução do *fitness* calculado no AG e no AM.

A Figura 4.16 apresenta a resposta do sistema e dos modelos quando excitados por um sinal degrau unitário.

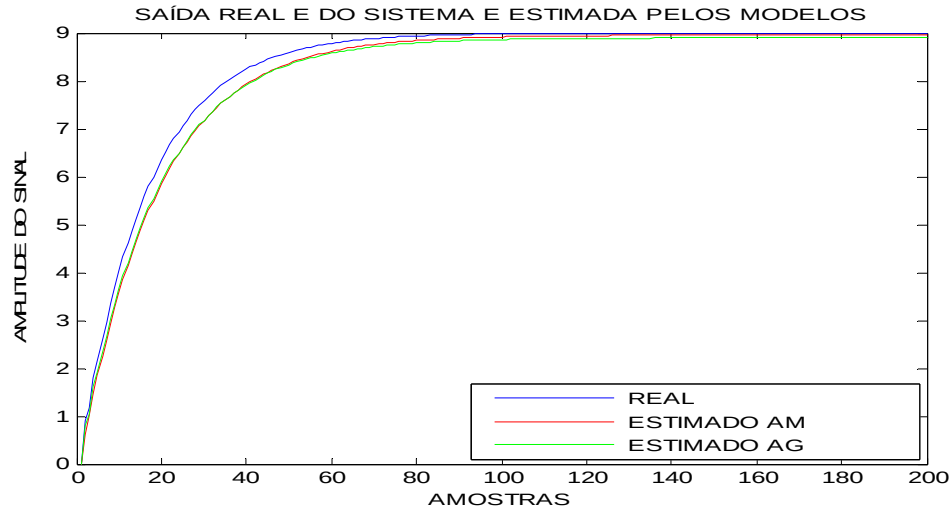


Figura 4.16 - Resposta do sistema e dos modelos com sinal em degrau unitário.

Analisando-se as respostas no domínio do tempo para o modelo não linear com saturação na entrada apresentado Figura 4.16, verifica-se o bom desempenho dos

modelos apresentados. Há apenas um pequeno erro em regime transitório, o qual se torna praticamente nulo em regime permanente.

4.3.2 Sistema Não Linear – Conversor Buck

O conversor Buck usado neste trabalho está representado na Figura 4.17. Durante todo o teste a fonte CC de alimentação, V_d, é mantida constante e igual a 24V. O MOSFET IRF840 é chaveado atuando-se na porta G. A razão cíclica é definida como a proporção de tempo em que a chave está ligada em relação ao período de operação T, isto é, D = T_{ligado}/T. A razão cíclica foi variada usando-se técnicas de modulação por largura de pulso (PWM, do inglês *Pulse Width Modulation*) a uma freqüência de chaveamento de 33 kHz. Esta modulação é realizada pelo circuito integrado LRM3524 (AGUIRRE et al., 2000b).

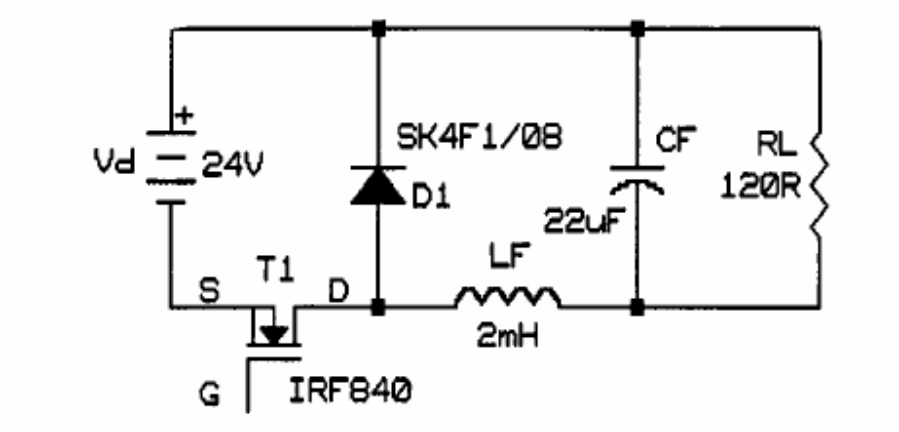


Figura 4.17: Conversor CC-CC Buck

A freqüência de chaveamento utilizada resulta num modo de operação contínuo. Quando o ciclo de trabalho tende à unidade, a corrente flui através de LF e a tensão de saída V_o aumenta, pois V_d é aplicada em série com LF e CF. No entanto, quando D aproxima-se de zero, a tensão da saída diminui de acordo com o regime dinâmico definido por CF, LF e a carga RL.

4.3.2.1 Modelo do Conversor Buck

O conversor Buck apresentado na Figura 4.17 apresenta a seguinte relação de tensão em estado estacionário (AGUIRRE et al., 2000b).

$$V_o = (1 - D)V_d$$

$$V_o = \left(1 - \frac{\bar{u} - 1}{3}\right) V_d \tag{4.22}$$

$$V_o = \frac{4Vd}{3} - \frac{Vd}{3}\bar{u},$$

sendo V_o a tensão na carga, D o ciclo de trabalho, V_d = 24V a tensão constante fornecida pela fonte, e $\bar{u}$ o valor de estado estacionário do sinal de controle $u(k)$.

Em Aguirre et al.(2000b) podem ser vistos vários modelos para esse sistema. O que melhor reproduziu a característica por eles desejada foi o modelo obtido através do uso de informação a *priori* na escolha de termos candidatos. A estrutura selecionada é

$$\begin{aligned} y(k) \;=\; & c_{1,0}(1)y(k-1) + c_{1,0}(2)y(k-2) + c_{1,0}(3)y(k-3) + \\ & + c_{0,2}(1,3)u(k-1)u(k-3) + c_{0,3}(1,1,3)u(k-1)^2\,u(k-3) + \\ & + c_{0,3}(1,3,3)u(k-1)u(k-3)^2 + c_{0,3}(1)u(k-1)^3 + c_{0,2}(3)u(k-3)^3 + c_0. \end{aligned} \tag{4.23}$$

A literatura apresenta resultados da estimação dos parâmetros através de dois métodos, os quais podem ser vistos em Aguirre (2004).

- Método dos Mínimos Quadrados

$$\begin{aligned} y(k) = {} & 1{,}2013y(k-1) - 0{,}2608y(k-2) - 0{,}2080y(k-3) - \\ & - 0{,}6162u(k-1)u(k-3) + 8{,}8699\,u(k-1)^2u(k-3) - \\ & - 9{,}7707u(k-1)u(k-3)^2 - 2{,}6783u(k-1)^3 + 3{,}6636u(k-3)^3 + 6{,}2479 \end{aligned} \tag{4.24}$$

- Método do algoritmo do elipsóide

$$\begin{aligned} y(k) = {} & 0{,}7315y(k-1) - 0{,}0047y(k-2) - 0{,}2495y(k-3) - \\ & - 1{,}7617u(k-1)u(k-3) + 3{,}6774\,u(k-1)^2u(k-3) - \\ & - 4{,}6409u(k-1)u(k-3)^2 - 0{,}8280u(k-1)^3 + 2{,}0210u(k-3)^3 + 13{,}7292 \end{aligned} \tag{4.25}$$

O conjunto de parâmetros estimados pelo algoritmo genético (AG) do sistema (4.23) é apresentado em (4.26), do algoritmo memético (AM), em (4.27). Foram encontrados utilizando-se mil pontos (dados de entrada e saída), com uma população de oitenta indivíduos (cromossomos), que se reproduziram por no máximo quinhentas gerações.

$$\begin{aligned} y(kG) = {} & 0{,}0795y(k-1) + 0{,}1255y(k-2) + 0{,}0733y(k-3) + \\ & + 3{,}5065u(k-1)u(k-3) - 1{,}4228u(k-1)^2u(k-3) + \\ & + 0{,}7266(k-1)u(k-3)^2 - 3{,}4755u(k-1)^3 - 1{,}5986u(k-3)^3 + 4{,}7302 \end{aligned} \tag{4.26}$$

$$\begin{aligned} y(kM) = {} & 0{,}8651y(k-1) - 0{,}2414y(k-2) - 0{,}2137y(k-3) - \\ & - 0{,}1457u(k-1)u(k-3) + 10.696\,u(k-1)^2u(k-3) + \\ & + 4{,}9039u(k-1)u(k-3)^2 - 5{,}1586u(k-1)^3 + 0{,}0750u(k-3)^3 + 3{,}1207 \end{aligned} \tag{4.27}$$

A Figura 4.18 apresenta a saída do sistema (saída real) e as saídas com parâmetros estimados pelo método dos mínimos quadrados e algoritmo do elipsóide, constantes em Aguirre (2004). A Figura 4.18(a) apresenta o desempenho do AG e a Figura 4.18(b) apresenta o desempenho do AM.

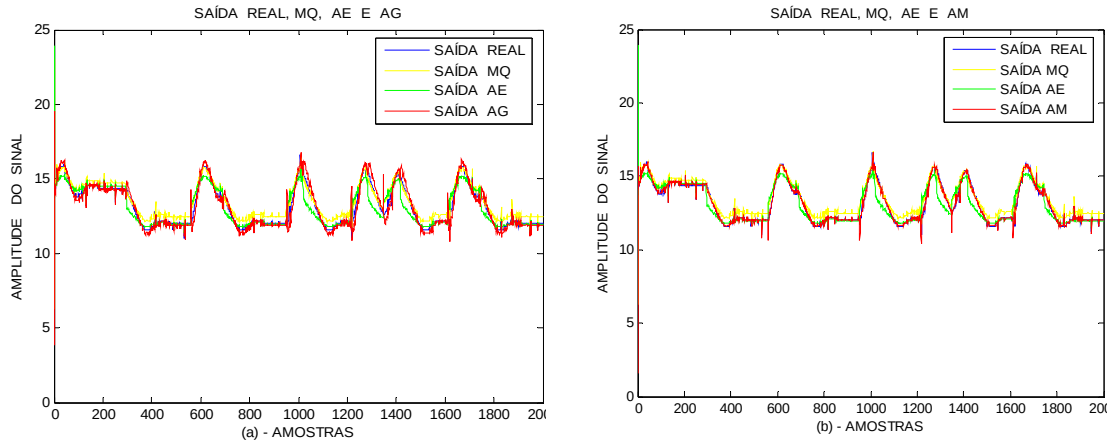


Figura 4.18 – Resposta temporal do modelo e dos sistemas (4.18): (a) saída real, MQ, AE e AG; (b) saída real, MQ, AE e AM.

Observando-se as duas figuras, percebe-se que o AG e o AM mantêm com algumas dificuldades as características do sistema, dificuldades estas que são também observadas nos outros modelos citados.

A evolução dos parâmetros do melhor indivíduo no decorrer das gerações no AG e no AM está reapresentada, respectivamente, nas Figuras 4.19.

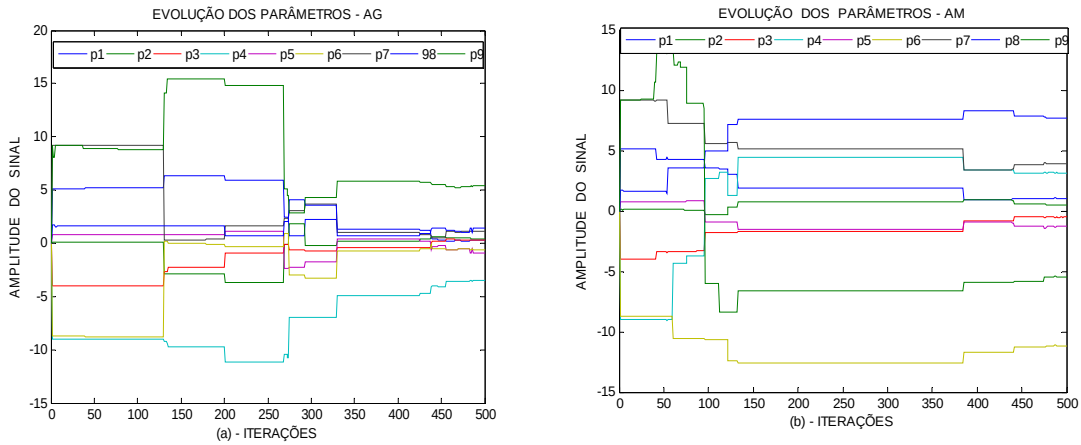


Figura 4.19 – Comportamento dos parâmetros: (a) parâmetros do AG; (b) parâmetros do AM.

Observando-se as Figuras 4.19(a) e 4.19(b), percebe-se que no trabalho computacional desenvolvido por quinhentas iterações os modelos apresentam estabilidade, com o AM evoluindo com maior velocidade, pois apresenta pequenas variações nos parâmetros estimados depois de 150 iterações.

A evolução do *fitness* pode ser observada na Figura 4.20. **N**a análise em função do tempo utilizado para realizar o trabalho computacional, constatou-se que o AM necessita em torno de 40% a mais de tempo em relação ao AG. A redução do erro constatada na Figura (4.20), através da análise do *fitness* de cada algoritmo, compensa a utilização de maior tempo na execução do mesmo.

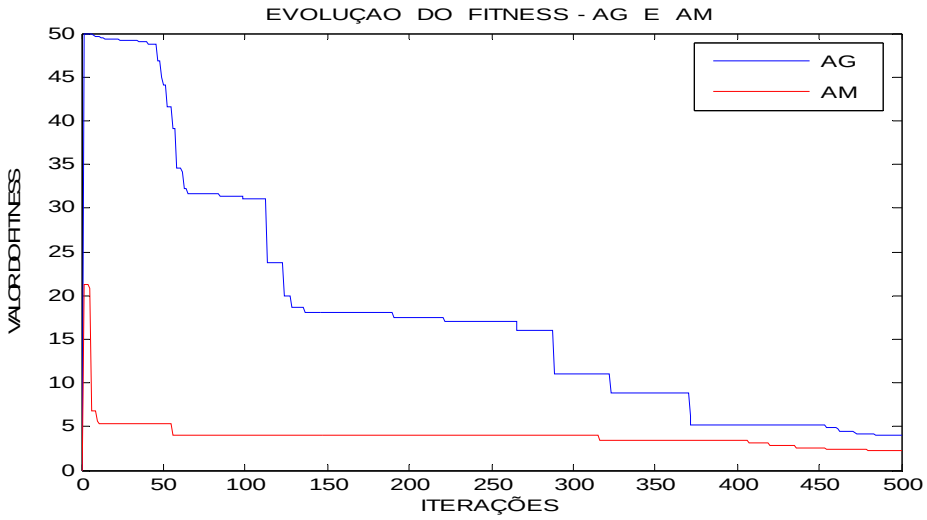


Figura 4.20 – Evolução do *fitness* calculado no AG e no AM.

A Figura 4.21 apresenta a resposta do modelo obtido pelo método dos mínimos quadrados, pelo método do algoritmo do elipsóide, pelo algoritmo genético (AG) e pelo algoritmo memético (AM), quando excitados por um sinal em degrau.

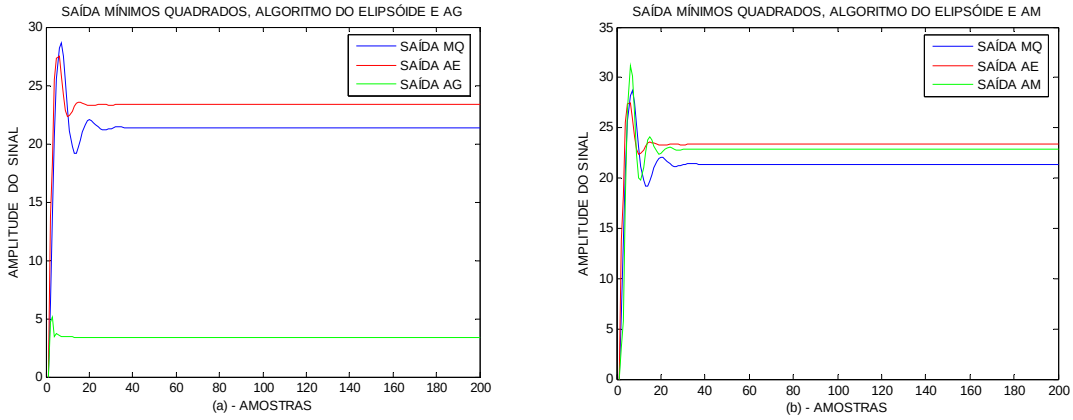


Figura 4.21 - Resposta ao sinal em degrau unitário: (a) mínimos quadrados, algoritmo do elipsóide e AG (b) mínimos quadrados, algoritmo do elipsóide e AM.

A Figura 4.21(a) apresenta o desempenho do AG em relação aos modelos obtidos pelo método dos mínimos quadrados e pelo método do algoritmo do elipsóide e a Figura

4.21(b) apresenta o desempenho do AM em relação aos modelos obtidos pelo método dos mínimos quadrados e pelo método do algoritmo do elipsóide. Observando-se as duas figuras, percebe-se que o AM responde satisfatoriamente com relação aos outros modelos fornecidos pela literatura.

4.3.3 Algumas Considerações sobre os Sistemas Não Lineares

Analisando-se as respostas no domínio do tempo dos sistemas não lineares apresentados, percebe-se que os modelos encontrados para o sistema não lineares representam satisfatoriamente o sistema em regime transitório e em regime permanente. Somente o algoritmo genético não conseguiu generalizar o seu modelo para o caso do converso Buck. Assim, a busca local proporcionou ao AM melhora no desempenho, fazendo com que os modelos acompanhem o comportamento do sistema no intervalo amostrado e sinais diferentes dos usados na fase de estimação.

CONCLUSÃO

Neste trabalho foi desenvolvido um algoritmo memético como metodologia alternativa usada na estimação de parâmetros para identificação de sistemas, utilizando o modelo discreto do tipo ARX para sistemas lineares e NARX para sistemas não lineares. O motivo da escolha desse método deu-se pela sua eficácia na busca de soluções globais, flexibilidade, simplicidade de implementação computacional e possibilidade de trabalhar com diversos tipos de sinais e representações.

Analisando os resultados, o algoritmo memético conseguiu obter de forma eficiente modelos gerais para sistemas lineares e de forma satisfatória para sistemas não lineares. Observa-se que para sistemas de maior ordem e para os sistemas não lineares, o algoritmo encontra maiores dificuldades na obtenção desses modelos gerais, em razão do aumento do número de parâmetros para a estimação. Todavia, encontra modelos que representam de maneira satisfatória o comportamento dinâmico dos sistemas quando estes são excitados por sinais que não apresentam as mesmas características usadas na estimação. O desempenho dos modelos estimados pelo algoritmo memético foi avaliado no domínio do tempo através da saída do modelo e do sistema e os dados obtidos mostram uma alta correlação entre esses sinais. No domínio da freqüência foi utilizado, para os sistemas lineares, o diagrama de Bode da Magnitude, o qual mostra novamente uma alta correlação entre os sinais.

Por se tratar de um algoritmo evolutivo de busca estocástica e por este apresentar comportamento diferente para cada simulação, foi introduzida neste trabalho uma

estratégia que gera a população inicial com grande número de cromossomos. Submetendo-os ao processo de avaliação e organizando-os a partir do desempenho obtido, inicia-se o processo de reprodução. Isso fez o algoritmo diminuir o esforço computacional e encontrar as mesmas respostas em várias simulações.

Um comportamento interessante na evolução dos parâmetros observado durante o processo de estimação foi a melhor convergência dos parâmetros associados às entradas do sistema na forma recursiva, o que representou uma boa resposta do regime transitório desse sistema. De modo geral, o algoritmo memético desenvolvido constitui-se numa boa técnica de estimação de parâmetros para modelos lineares, atingindo uma taxa de acerto em torno de 95% nas instâncias maiores e, para os sistemas não lineares, foi possível atingir várias soluções ótimas, nas quais a busca local mostrou-se primordial.

Como sugestão de trabalhos futuros em termos de aplicação do algoritmo memético na identificação de sistemas não lineares têm-se: variação da metodologia da busca local, estimação de parâmetros de sistemas lineares de ordem superior, estimação *on-line* dos parâmetros do sistema, o uso de outros modelos diferentes do NARX e o uso de formas diferentes da recursiva.

BIBLIOGRAFIA

AGUIRRE, Luiz Antônio. **Introdução à identificação de sistemas**: técnicas lineares e não lineares aplicadas a sistemas reais. 2. ed. Ver. e ampl. Belo Horizonte: Ed. UFMG, 2004.

AGUIRRE, Luiz Antônio; RODRIGUES, Giovani G.; JÁCOME, Cristiano R. F. Identificação de sistemas não-lineares utilizando modelos NARMAX polinomiais - Uma revisão e novos resultados. **SBA Controle & Automação**, v. 09, n. 2, p. 90-106, 1998.

AARTS, E.; VERHOEVEN, M. Operations Research, in Bäck, T., Fogel, D. B. & Michalewicz, Z. Handbook of Evolutionary Computation, Oxford University Press, 1997.

ASSIS, Adilson José. **Identificação e controle de processos não lineares utilizando redes neurais artificiais**. Tese (Doutorado em Engenharia) - Universidade Estadual de Campinas, São Paulo, 2001.

AXELROD, R.; The evolution of strategies in the iterated Prisoner's Dilemma, In Genetic Algorithms and Simulated Annealing. L. Davis; London, Pittman, p. 32-41, 1987.

BÄCK, T.; FOGEL, D. B.; MICHALEWICZ, T. Evolutionary computation 1 e 2, **Institute of Physics Publishing**, 2000.

BARCELLOS, João C. H. **Algoritmos genéticos adaptativos**: um estudo comparativo. Dissertação (Mestrado em Engenharia) - Escola Politécnica da Universidade de São Paulo, São Paulo, 2000.

BARONE, D.A.C. et al. Algoritmo genético no controle da trajetória de um veículo. In Congresso Brasileiro de Redes Neurais III, Florianópolis, 1997.

BILLINGS, S. A; AGUIRRE, L. A. Effects of the sampling time on the dynamics and identification of nonlinear models. **Int. J. Bifurcation and Chaos**, n.6, p.1541-1556, 1995.

BILLINGS, S. A., CHEN, S.; KORENBERG, M. J. Identification of MIMO nonlinear systems using a forward-regression orthogonal estimator. **Int. J. Control**, n.6, p.2157-2189, 1989.

BOLTON, W. **Engenharia de Controle**. Traduzido por Valcere V. Rocha e Silva. São Paulo, Makron Books, 1995.

BUI, T. N.; MOON, B.R. Genetic algorithm and graph partitioning. **IEEE Transations on computers,** v. 7, n. 45, p. 841-855, 1996.

CHENG, R.; GEN, M. Genetic algorithms and engineering design. John Wiley & Sons, New York, 1997.

COELHO, L. S.; COELHO, A. A. R. **Algoritmos evolutivos em identificação e controle de processos**: uma visão integrada e perspectivas, SBA Controle & Automação, v.10, n.1, p. 13-30, 1999.

COELHO, L. S.; COELHO, A. A. R. Estudo comparativo de configurações de Algoritmos Genéticos aplicados à identificação de processo multivariável. In Congresso Brasileiro de Automática XIII, Florianópolis, 2000.

CORRÊA, Marcelo Vieira, **Identificação caixa-cinza de sistemas não-lineares utilizando representações NARMAX racionais e polinomiais**. Tese (Doutorado em Engenharia Elétrica) - Universidade Federal de Minas Gerais, Belo Horizonte, 2001.

DE MORAES LIMA, C. A. **Emprego de teoria de agentes no desenvolvimento de dispositivos neurocomputacionais híbridos e aplicação ao controle e identificação de sistemas dinâmicos**. Dissertação (Mestrado em Engenharia Elétrica) - Universidade Estadual de Campinas, São Paulo, 2000.

EBY, D. et al, **The optimization of flywheels using an injection Island genetic algorithm.** *Evolutionary design by computers*, Morgan Kaufmann, San Francisco,1999.

FÁVARO, Silvio. **Estimação de pólos e zeros de modelos ultizando algoritmos genéticos.** Dissertação (Mestrado em Engenharia Elétrica e Informática Industrial) - Centro Federal de Educação Tecnológica do Paraná. Curitiba, 1999.

GLOVER, F. W. Tabu Search, Part I. **ORSA Journal on Computing**, 1 p. 190-206, 1989.

GLOVER, F. W. Tabu Search, Part II. **ORSA Journal on Computing**, 2 p. 4-32, 1990.

GLOVER, F. W.; LAGUNA, M.; MARTI, R. Fundamentals of scatter search and path relinking, **Control and Cybernetics**, 39, n.3, p. 653-684, 2000.

GOLDBERG, David E. **Genetic algorithms in search, optimization, and machine learning**. Addison Wesley Longman, Inc, 1989.

GOMES, Moacir Carlos. **Identificação de sistemas utilizando algoritmos genéticos para estimação de parâmetros**. Dissertação (Mestrado em Modelagem Matemática) -Universidade Regional do Noroeste do Estado do Rio Grande do Sul, Ijuí, 2005.

HABER, R.; UNBEHAUEN, H. Structure identification of nonlinear dynamic systems-a survey on input/output approaches. **Automatica**, v. 26, n.4, p. 651-677, 1990.

HOLLAND, J.H. **Adaptation in natural and artificial system**. Ann Arbor, 2nd edition. The University of Michigan Press, 1975.

JÁCOME, C.R.F. **Uso de conhecimento prévio na identificação de modelos polinomiais NARMAX**. Dissertação (Mestrado em Engenharia Elétrica) - Universidade Federal de Minas Gerais, Belo Horizonte, 1996.

JOHANSEN, Tor A. Identification of non-linear systems using empirical data and prior knowledge-an optimization approach. **Automatica**, v.32, n.3, p. 337-356, 1996.

JOHANSSON, R. **System modeling identification**, Englewood Cliffs, NJ, Prentice-Hall, 1993.

KIM, S.; KIM, D.; CHANG, K., Using two successive subgradients in the ellipsoid method for nonlinear programming. **Journal of Optimization Theory and Applications**, p. 543-554, 1994.

KIRKPATRICK, S.; GELLAT, C. D.; VECCHI, M. P. Opitimization by simulated anelaling. **Science**, v. 220, p. 671-680, 1983

KORENBERG, M.; BILLINGS, S.; Y. L.; L.; MCLLOY, P. Orthogonal parameter estimation algorithm for non-linear stochastic systems. **Int. J. Control**, Louisville, n.1, p.193-210, 1988.

KRISTINSSON, K.; DUMONT, G. A. System identification and control using genetic algorithms. **IEEE Trans. On Sys., Man, and Cybernetics**, v. 22, n.5, p. 1033-1046, 1992.

LEONTARITIS, I. J.; BILLINGS, S. A. Input-output parametric models for nonlinear systems part I: deterministic nonlinear systems. **Int. J. Control**, n.2, p.303-328, 1985a.

LEONTARITIS, I. J.; BILLINGS, S. A. Input-output parametric models for nonlinear systems part II: stochastic nonlinear systems. **Int. J. Control**, n.2, p.329-344, 1985b.

LI, Y. Modern information technology for control systems design and implementation. **Proc. Second Asiapacific Conference on Control and Measurement**, Wuhan-Chongping, China, 1995.

LJUNG, Lennart. **System identification:** theory for the user. 2nd edition, New Jersey, Prentice-Hall, Englewood Cliffs, 1999.

MATSUMOTO, Élia Yathie. **Matlab 6.5:** fundamentos de programação. São Paulo, Érica, 2002.

MEDEIROS, Anderson Vinícius. **Modelagem de sistemas dinâmicos não lineares utilizando sistemas Fuzzy, algoritmos genéticos e funções de base ortonormal**. Dissertação (Mestrado em Engenharia Elétrica) - Universidade Estadual de Campinas, São Paulo, 2006.

MENDES, Alexandre de Sousa. **O framework NP-Opt e suas aplicações a problemas de otimização**. Tese (Doutorado em Engenharia Elétrica e de Computação) - Universidade Estadual de Campinas, São Paulo, 2003.

MERZ, P.; FREISLEBEN, G. A genetic local search algorithm for solving symmetric and asymmetric traveling salesman problems. **International Conference on Evolutionary Computation**, p. 616-621, 1996.

MICHALEWICZ, Z. **Genetic algorithms + data structures = evolution programs**. Springer Verlag, 1999.

MOSCATO, P. **Problemas de otimização NP, aproximabilidade e computação evolutica:** da prática à teoria**,** PhD Thesis**.** Faculdade de Engenharia Elétrica e de computação, Universidade Estadual de Campinas, São Paulo, 2001.

MOSCATO, P. **On evolution, search, optimiztion, genetic algorithms and martial arts:** towards memetic algorithms**.** Caltech Concurrent Computation Program, C3P Report 826, 1989.

MOSCATO, P.; COLLA, C. An introduction to memetic algorithms, **Revista Iberoamericana de Inteligência Artificial**, n. 19, p. 131-148, 2003.

MOSCATO, P.; NORMAN, M. G. A memetic approach for the traveling salesman problem implementation of a computational ecology for combinatorial optimization on message-passing systems. **Parallel Computing and Transputer Aplications**, IOS PRESS, p 187-194, 1992.

NARENDRA, K. S.; PARTHASARATHY, K. Identification and control of dynamical system's using neural networks. **IEEE Transactions on Neural Networks**, v.1, n.1, p. 04-26, 1990.

NORTON, J. P. An introduction to identification. **Academic Press**: London, 1986.

OGATA, K. **Engenharia de controle moderno**. 3. ed. Livros Técnicos e Científicos Rio de Janeiro, 1998.

SILVA, W. L. A. **Métodos de identificação de sistemas auxiliados por computador**. Dissertação (Mestrado em Engenharia Elétrica), Universidade Federal da Paraíba, 1997.

SWAIN, A. K. Continuous time identification of nonlinear systems using frequency domain transfer functions. Ph. D. diss., University of Sheffield, Sheffield, UK, 1996.

TAN, K. C.; LI, Y.; MURRAY-SMITH, D.J.; SHARMAN, K. C. System identification and linearization using genetic algorithms with simulated annealing. **Proc. IEE/IEEE Int. Conf. On GA in Eng. Syst.: i**nnovations and applications, Sheffield, U.K., p. 164-169, 1995.

TANOMARU, Julio. Motivação, fundamentos e aplicações de algoritmos genéticos. **II Congresso Brasileiro de Redes Neurais**. Curitiba, 1995.

THOMSON, M.; SCHOOLING, S. P.; SOUFIAN, M. The practical application of a nonlinear identification methodology. **Control Eng. Practice**, v.4, n.3, p. 295-306, 1996.

TIN JUNIOR, Gilberto Jorge. **Algoritmos meméticos aplicados ao problema de NO-WAIT FLOWSHOP.** Dissertação (Mestrado em Engenharia Elétrica e de Computação) -Universidade Estadual de Campinas, São Paulo, 2001.

TUSON. A. L.; ROSS, P. M. Adapting operator settings in genetic algorithms, **Evolutionary Computation**, 1998.

UNBEHAUEN, H.; GÖHRING, B. Tests for determining model order in parameter estimation. **Automatica**, v. 10, p 233-244, 1974.

VELLOSO, M.L.; MENDONÇA, J.M.; PACHECO, M.A.; VELLASCO, M.M.B.R. Otimização de planejamento de horários por algoritmos genéticos. In Congresso Brasileiro de Redes Neurais II, Curitiba, 1995.

VON ZUBEN, Fernando J. **Modelos paramétricos e não-paramétricos de redes neurais artificiais e aplicações**. Tese (Doutorado em Engenharia Elétrica e de Computação) - Universidade Estadual de Campinas, São Paulo, 1996.

WIGREN, Torbjörn. Recursive prediction error identification using the nonlinear wierner model. **Automatica**, v. 29, n. 4, p. 1011-1025, 1993.

WHITLEY, D.; STARKWEATHER, T. Optimization small and neural networks using a distributed genetic algorithm. In International Joint Conference on Neural Netwoks. n. 1, p. 206-209, Washington, DC, 1990.

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo