UNIVERSIDADE FEDERAL DO PARANÁ

DANIEL KAMINSKI DE SOUZA

ESTIMAÇÃO DE PARÂMETROS POR BUSCA DE CUCO VIA VOOS DE LÉVY

CURITIBA

2014

DANIEL KAMINSKI DE SOUZA

ESTIMAÇÃO DE PARÂMETROS POR BUSCA DE CUCO VIA VOOS DE LÉVY

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Engenharia Elétrica, no Curso de Pós-Graduação em Engenharia Elétrica, Setor de Tecnologia, da Universidade Federal do Paraná.

Orientador: Gideon Villar Leandro
Coorientador: Gustavo Henrique da Costa Oliveira

CURITIBA

2014

Souza, Daniel Kaminski de
Estimação de Parâmetros por Busca de Cuco via voos de Lévy/ Daniel Kaminski de Souza. – Curitiba, 2014
100 f. : il.; grafs., tabs.

Dissertação (mestrado) – Universidade Federal do Paraná, Setor de Tecnologia, Programa de Pós-Graduação em Engenharia Elétrica.
Orientador: Gideon Villar Leandro
Coorientador: Gustavo Henrique da Costa Oliveira
Bibliografia: p. 75-80

1. Algoritmos. 2. Otimização matemática. 3. Métodos de simulação. I. Leandro, Gideon Villar. II. Oliveira, Gustavo Henrique da Costa. III. Estimação de Parâmetros por Busca de Cuco via voos de Lévy.

CDD 621.3

TERMO DE APROVAÇÃO

DANIEL KAMINSKI DE SOUZA

ESTIMAÇÃO DE PARÂMETROS POR BUSCA DE CUCO VIA VOOS DE LÉVY

Dissertação aprovada como requisito parcial para obtenção do grau de Mestre no Curso de Pós-Graduação em Engenharia Elétrica, Setor de Tecnologia, Universidade Federal do Paraná, pela seguinte banca examinadora:

Gideon Villar Leandro - Orientador
Universidade Federal do Paraná

Gustavo Henrique da Costa Oliveira - Coorientador
Universidade Federal do Paraná

Eduardo Parente Ribeiro - Convidado 1
Universidade Federal do Paraná

Leandro dos Santos Coelho - Convidado 2
Universidade Federal do Paraná

Flávio Neves Júnior - Convidado 3
Universidade Tecnológica Federal do Paraná

Curitiba
2014

Dedico esse trabalho a todos os professores. Seus ensinamentos representam um futuro melhor, mais feliz e sustentável para toda a natureza e que compreende também, de maneira especial, nossa própria existência.

Agradecimentos

Agradeço principalmente a minha esposa Janice assim como a minha filha Aline por toda a compreensão que me deram nesses anos. Agradecimentos especiais são direcionados ao departamento de Engenharia Elétrica da Universidade Federal do Paraná, ao grupo de pesquisa em sistemas dinâmicos e controle e aos colegas que contribuíram para o desenvolvimento desse trabalho. Agradeço ainda meus superiores na Electrolux do Brasil S/A pelo suporte a essa pesquisa.

Muito melhor é ousar coisas poderosas, ousar ganhar triunfos gloriosos, mesmo marcado por fracassos. . . a enfileirar-se com aqueles pobres de espírito que nem se alegram muito nem sofrem muito, porque eles vivem num crepúsculo acinzentado que não conhece nem vitória nem derrota.

Strenuous Life

THEODORE ROOSEVELT

Resumo

Nesse trabalho aplica-se o algoritmo de otimização contínua conhecido como Busca de cuco via voos de Lévy na estimação de parâmetros de sistemas não lineares.

Com foco na estimação de parâmetros, propôs-se algumas melhorias no algoritmo Busca de cuco via voos de Lévy que também podem ser estendidas para qualquer problema de otimização contínua. Os resultados de estimação de parâmetros dos sistemas abordados demonstram a superioridade, em relação à acurácia, desse método em comparativo com o algoritmo genético de ordenação não dominado rápido.

Palavras-chaves: estimação de parâmetros. algoritmo de otimização. otimização contínua. busca de cuco via voos de Lévy. algoritmo de ordenação não dominado rápido. NSGAII.

Abstract

In this work the optimization algorithm known as cuckoo search via Lévy flights was applied in the parameters estimation of non-linear systems.

With focus on the parameters estimation, some improvements were made in the Cuckoo search via Lévy flights algorithm that can also be extended to any continuous optimization problem. The parameters estimation results from the systems dealt with show the superiority, regarding accuracy, of this method when compared with the fast nondominated sorting genetic algorithm.

Palavras-chaves: parameters estimation. optimization algorithm. continuous optimization. cuckoo search via Lévy flights. nondominated sorting genetic algorithm. NSGAII.

Lista de ilustrações

Lista de tabelas

Lista de símbolos

$C(\alpha)$ parametro C da transformação não linear de Mantegna. 57

$C_1(\alpha)$ primeira solução da Equação 3.5. 57

$C_2(\alpha)$ segunda solução da Equação 3.5. 57

D parâmetro relacionado à dimensão fractal. 43

J função de custo. 31, 32, 55

$K(\alpha)$ parametro K da transformação não linear de Mantegna. 57

M modelo matemático. 31

N Distribuição Gaussiana. 45

O notação matemática do Grande-O. 15, 43, 44

S Distribuição α-estável. 45, 46

S sistema real. 17, 31, 62, 64–66

U distribuição dos comprimentos dos passos. 43

Z conjunto de dados que inclui tanto sinais de entrada e saída do sistema. 31, 32, 55, 62

Z_M conjunto de dados de saída. 31, 32, 55

Γ função de Euler Γ. 56

α expoente característico da distribuição α-estável. 45, 46

α índice de estabilidade da distribuição estável de Lévy. 19, 57

β assimetria da distribuição α-estável. 45, 46

δ localização da distribuição α-estável. 45, 46

γ escala da distribuição α-estável. 45, 46

$\hat{\theta}$ parâmetros de estimação. 31, 55

μ média. 45

σ desvio padrão. 45

p_a probabilidade de descoberta de ovo estranho pela anfitriã. 37, 38, 62

v variável estocástica intermediária da transformação não linear. 57

w variável estocástica independente resultante da transformação não linear de Mantegna. 57

Lista de acrônimos

AG algoritmos genéticos. 15, 16, 27, 31, 37, 48–52, 55, 67, 68, 71, 73

ASIC circuito integrado de aplicação específica (*Application Specific Integrated Circuit*). 57

BC Busca de Cuco via voos de Lévy(*Cuckoo Search via Lévy Flights*). 15, 16, 21, 28, 29, 31, 36, 40, 49–53, 55, 56, 61, 62, 64, 65, 67–71, 73, 74

cdf função de distribuição cumulativa (*Cumulative Distribution Function*). 45

DNA ácido desóxi ribonucleico (*DeoxyriboNucleic Acid*). 48

EE erro na equação (*equation error*). 66

EP enxame de partículas. 27, 29, 37

EPBC estimador de parâmetros por Busca de Cuco via voos de Lévy(*Cuckoo Search via Lévy Flights*). 17, 61, 62, 64–67, 69

EPNSGAII estimador de parâmetros com NSGA-II. 17, 61, 65, 66

IIR resposta ao impulso infinita (*Infinite Impulsive Response*). 27

MOEA algoritmo evolucionário multi-objetivo. 61

MQ mínimos quadrados. 27

MSE erro médio quadrático (*Mean Squared Error*). 62

NARMAX média móvel auto regressiva não linear com entrada exógena(*Nonlinear Auto Regressive Moving Average with eXogenous input*). 27, 32, 34, 63

NARX auto regressivo não linear com entrada exógena (*Nonlinear Auto Regressive with eXogenous input*). 27

NSGA algoritmo genético de ordenação não dominado rápido(*Nondominated Sorting Genetic Algoritm*). 21, 61

NSGA-II algoritmo genético de ordenação não dominado rápido(*Nondominated Sorting Genetic Algoritm*) II. 21, 61, 65, 66, 73

OE erro na saída (*output error*). 15, 17, 22, 24, 61, 63–67, 69

OEP sistema erro na saída (*output error*) polinomial. 15, 17, 61, 63–65, 67

OER sistema erro na saída (*output error*) racional. 15, 24, 61, 65, 69

PAES estratégia de evolução pareto-arquivada. 61

pdf função densidade de probabilidade (*Probability Density Function*). 45

RNA rede neural artificial. 27

SISO única entrada única saída (*Single Input Single Output*). 62

SPEA Pareto-força EA. 61

UML linguagem de modelagem universal (*Universal Modeling Language*). 15, 32, 38, 39

Sumário

1 Introdução

1.1 Contexto e Motivação

A engenharia de controle é a área da engenharia que aplica a teoria de controle para projetar sistemas com comportamentos desejados. A prática usa sensores para medir o desempenho de saída do sistema que está sendo controlado e suas medidas podem ser usadas para dar realimentação aos atuadores de entrada que fazem correções em direção ao desempenho desejado. Quando um aparelho é projetado para desempenhar uma atividade sem a necessidade de entradas humanas para uma eventual correção ele é chamado controle automático (como o piloto automático do carro para regular a velocidade). Multi disciplinar por natureza, a engenharia de controle de sistemas foca na construção de sistemas de controle principalmente derivados da modelagem matemática de um amplo espectro de sistemas dinâmicos.

A modelagem matemática é o processo de desenvolvimento de um modelo matemático. Que por sua vez é a descrição de um sistema utilizando-se linguagem e conceitos matemáticos. Modelos matemáticos são usado não somente nas ciências naturais (como física, biologia, ciência da terra, meteorologia) e disciplinas da engenharia (como ciência da computação, inteligência artificial), mas também nas ciências sociais (como economia, psicologia, sociologia e ciência política); físicos, engenheiros, estatísticos, analistas de pesquisa de operações e economistas estão dentre os que mais os utilizam extensivamente. Um modelo pode ajudar a explicar um sistema e estudar os efeitos de diferentes componentes, e para fazer previsões sobre comportamento.

Modelos matemáticos podem tomar muitas formas, incluindo mas não limitado a sistemas dinâmicos, modelos estatísticos, equações diferenciais, ou modelos teóricos de jogo. Esses e outros tipos de modelos podem se sobrepor, com um modelo envolvendo uma variedade de estruturas abstratas. Em geral, modelos matemáticos podem incluir modelos lógicos, desde que a lógica seja tomada como parte da matemática. Em muitos casos, a qualidade de um campo científico depende de quão bem os modelos matemáticos desenvolvidos no lado teórico concordam com os resultados de experimentos reproduzíveis.

Embora os termos engenharia de controle e cibernética (WIENER, 1948) estejam frequentemente relacionados a artefatos de alta tecnologia, os animais há tempo aplicam conceitos de controle com destreza invejável. Por exemplo, o cérebro de um mico leão dourado, além de outras funcionalidades, desempenha o papel de um controlador capaz de receber sinais dos olhos (monitor) sobre a distância entre sua mãozinha e um fruto a ser catado. A informação enviada pelo monitor ao controlador é chamada de realimentação, e

baseando-se nela o controlador pode comandar instruções para trazer o comportamento monitorado (alcance da mãozinha) mais perto do comportamento desejado (catar o fruto).

De acordo com (KALMAN, 2013), a teoria de controle não lida diretamente com a realidade, mas sim com modelos matemáticos. O mico-leão, por exemplo, antes mesmo que a operação catar o fruto seja desempenhada, ele já tem uma previsão do resultado. Isso é possível porque ele tem construído em seu cérebro um modelo relativamente preciso da realidade (mãozinha, fruto, ambiente) e que o ajuda no controle do movimento.

Os mesmos princípios são seguidos na Engenharia de Controle, e por isso a fidelidade dos modelos é essencial para atender os requisitos de desempenho. Porém nem sempre os modelos estão disponíveis de antemão, gerando-se a necessidade de construí-los com base em observações da realidade. A área responsável por obter os modelos de um sistema na engenharia é a Identicação de Sistemas.

O campo da identificação de sistemas utiliza métodos estatísticos para construir modelos matemáticos de sistemas dinâmicos a partir de dados medidos. A identificação de sistemas também inclui o dimensionamento ótimo de experimentos para gerar eficientemente dados informativos para encaixar esses modelos assim como para redução de modelos.

Nessa área, a maioria dos fenômenos são primeiramente modelados como sistemas lineares para facilitar o entendimento dos seus comportamentos com uma matemática menos complexa. Assim que a tecnologia evolui, melhores modelos em termos de consumo de recursos e qualidade são necessários para atingir as novas demandas dos processos. Como resultado, uma pesquisa mais profunda sobre o comportamento do sistema frequentemente leva a modelos não lineares. Por isso e além de outros motivos, a identificação de sistemas não lineares tem sido uma área de grande pesquisa nas últimas décadas (LJUNG, 2010; BILLINGS, 1980; LJUNG, 2007; SUBUDHI; JENA, 2011b; SANANDAJI et al., 2011; UGALDE et al., 2013; SUBUDHI; JENA, 2011a; ERITEN et al., 2013; SINGH; CHATTERJEE, 2011; TESLIC et al., 2011; PATCHARAPRAKITI et al., 2010; BAI; DEISTLER, 2010; BAI, 2010).

Um sistema não linear é definido como um sistema que não satisfaz o princípio da superposição. Há vários tipos diferentes de sistemas não lineares. Historicamente, a identificação de sistemas não lineares (NELLES, 2001; BILLINGS, 2013) tem se desenvolvido focando em classes específicas de sistemas e pode ser largamente categorizada em abordagens básicas cada uma definida por uma classe de modelo, dentre eles:

1. Modelos de séries de Volterra
2. Modelos estruturados de bloco (Wiener, Hammerstein e combinação)
3. Modelos de redes neurais

4. Modelos média móvel auto regressiva não linear com entrada exógena(*Nonlinear Auto Regressive Moving Average with eXogenous input*) (NARMAX)

Como os modelos de séries de Volterra, os modelos estruturados de bloco e muitas arquiteturas de redes neurais podem todos ser considerados subconjuntos do modelo NARMAX, esse trabalho utiliza a abordagem NARMAX para a identificação de modelos não lineares. Desde que NARMAX foi introduzida, provando-se qual classe de sistemas não lineares podem ser representadas por esse modelo, muitos resultados e algoritmos foram derivados baseando-se nessa descrição. A maioria dos trabalhos inicialmente estava baseada em expansões polinomiais do modelo NARMAX. Esses são ainda os métodos mais populares, porém outras formas mais complexas baseadas em *wavelets* e outras expansões têm sido introduzidas para representar sistemas não lineares altamente complexos e severamente não-lineares. Uma proporção significativa dos sistemas não-lineares pode ser representada por um modelo NARMAX inclusive sistemas com comportamentos exóticos como caos, bifurcações, e subharmônicas (BILLINGS, 2013).

Dentre as atividades da identificação de sistemas não lineares uma atividade chave é a identificação de parâmetros que faz com que o modelo represente a dinâmica pretendida.

Para estimar parâmetros de sistemas não lineares o método dos mínimos quadrados (MQ) foi amplamente utilizado (MOORE, 1978; GOODWIN, 1977; SOLO, 1978) devido à sua simplicidade, velocidade e matemática bem entendida num tempo em que o poder computacional disponível era relativamente baixo. Mais tarde foi descoberto que problemas que requerem um modelo de ruído tornariam o modelo não linear nos parâmetros, prejudicando a estimação de parâmetros sem tendências por estimadores baseados em MQ (KUMAR; MOORE, 1981).

Para trabalhar com não linearidades nos parâmetros, alguns algoritmos heurísticos de otimização avançados foram empregados. (SAMAD; MATHUR, 1992) utilizaram estimadores de parâmetros com redes neurais artificiais (RNAs) para uma dada estrutura de modelo parametrizada treinada por aprendizado supervisionado. (YAO; SETHARES, 1994) usaram algoritmos genéticos (AG) para resolver o problema de identificação de parâmetros para filtros digitais IIR lineares e não lineares com atraso de transporte ou não lineares nos parâmetros. Mais recentemente, (SCHWAAB et al., 2008) propuseram o uso de otimização por enxame de partículas (EP) para estimação de parâmetros não lineares apresentando bons resultados estatísticos.

Mais recentemente, algoritmos genéticos foram empregados para identificação de modelos auto regressivos não lineares com entrada exógena (*Nonlinear Auto Regressive with eXogenous input*) (NARXs) (CHEN et al., 2007). (COELHO; PESSôA, 2009) usaram programação genética para seleção da estrutura no procedimento de identificação de sistemas baseado sobre uma representação NARX.

Baseando-se nos primeiros princípios, poderia-se construir um modelo chamado caixa branca, mas em muitos casos esses modelos são complexos demais e possivelmente difíceis de obter em tempo razoável devido à natureza complexa de muitos sistemas e processos.

Uma abordagem muito mais comum é portanto partir das medições do comportamento do sistema e das influências externas (entradas do sistema) e tentar determinar uma relação matemática entre elas sem entrar nos detalhes do que está na verdade acontecendo dentro do sistema. Dois tipos de modelos são comuns no campo da identificação de sistemas:

- **modelo caixa cinza**: embora as peculiaridades do que está acontecendo dentro do sistema não são inteiramente conhecidas, um certo modelo baseado em ambas informações, intuição sobre o sistema e dados experimentais, é construído. Entretanto esse modelo ainda tem um número de parâmetros livres que podem ser estimados utilizando-se técnicas de identificação de parâmetros (NIELSEN et al., 2000; GRAUPE, 1976). Um exemplo, (EYKHOFF, 1974) usa o modelo de saturação Monod para crescimento microbial. O modelo contém um relacionamento hiperbólico simples entre a concentração do substrato e a taxa de crescimento, mas isso pode ser justificado pela ligação das moléculas a um substrato sem entrar em detalhes dos tipos de moléculas ou tipos de ligação. A modelagem caixa cinza é também conhecida como modelagem semi-física (FORSSELL et al., 1997).

- **modelo caixa preta**: Nenhum modelo anterior está disponível. A maioria dos algoritmos de identificação de sistemas trabalham com modelos desse tipo.

Como os algoritmos de otimização tem um papel importante na identificação de sistemas caixa preta, e também motivados pelos recentes avanços na área dos algoritmos de otimização, esta pesquisa propõe uma nova técnica para estimar parâmetros de modelos utilizando-se um algoritmo evolucionário baseado na inteligência de enxames conhecido como Busca de Cuco via voos de Lévy(*Cuckoo Search via Lévy Flights*) (BC). Com foco na estimação de parâmetros, duas melhorias no algoritmo de otimização de busca de cuco via voos de Lévy foram realizadas. Essas melhorias podem ser estendidas também para qualquer problema de otimização contínua.

Neste trabalho, problemas bem discutidos anteriormente por (PIRODDI; SPINELLI, 2003), (ZHU, 2005) e (AGUIRRE; BARBOSA; BRAGA, 2010) são explorados de forma a avaliar a qualidade de estimação do algoritmo aqui proposto em comparação com técnicas recentes apresentadas na literatura. O algoritmo BC original será comparado com o algoritmo BC melhorado para destacar os avanços alcançados.

1.2 Objetivos

Este trabalho tem como objetivo fazer a estimação de parâmetros para sistemas lineares e principalmente não lineares através da utilização da metaheurística de otimização Busca de Cuco via voos de Lévy(*Cuckoo Search via Lévy Flights*) (BC) a qual obteve destaque nos últimos anos devido aos resultados alcançados em comparativos com outros algoritmos baseados em enxame como o enxame de partículas (EP).

Um objetivo secundário é a proposição de alterações no algoritmo BC com foco na atividade de estimação de parâmetros. Caso as alterações sejam relevantes também para problemas de otimização em geral, as melhorias podem ser estendidas de forma a promover a evolução da ferramenta de maneira mais ampla.

E por último numa escala de prioridade, objetiva-se divulgar a ferramenta de otimização BC para pesquisadores futuros que venham a necessitar de técnicas de otimização de tal forma a proporcionar uma utilização rápida da metaheurística BC sem grandes entraves no aprendizado e na adoção da nova sistemática.

1.3 Organização

Este trabalho está dividido em cinco capítulos. No Capítulo 1 objetiva-se situar o leitor a respeito da literatura relacionada às principais técnicas que compõem esse trabalho e de seu encaixe no contexto da linha de pesquisa. No Capítulo 2 alguns conceitos fundamentais são explorados com o auxílio de um exemplo prático para facilitar a compreensão e adoção da recente técnica de otimização Busca de Cuco via voos de Lévy(*Cuckoo Search via Lévy Flights*). No Capítulo 3, a metodologia utilizada no desenvolvimento desse trabalho é colocada. Esse capítulo está dividido em duas seções. Primeiramente na seção 3.1 demonstra-se como a metaheurística de otimização BC é aplicada na resolução do problema de identificação de parâmetros. E na seção 3.2, as modificações realizadas no algoritmo BC são detalhadas. Já no Capítulo 4, alguns problemas de estimação de parâmetros são atacados de forma a avaliar o desempenho da ferramenta criada e destacar algumas características importantes, os resultados são então mostrados e discutidos. No Capítulo 5, as conclusões são colocadas objetivando-se ressaltar os elementos mais importantes observados que possam ser úteis na tomada de decisão dos pesquisadores futuros.

2 Fundamentação

Objetiva-se aqui fundamentar os conceitos essenciais ao entendimento do trabalho desenvolvido. Esse capítulo está dividido em seis seções. Primeiramente na seção 2.1 o problema da identificação de parâmetros é detalhado. Na seção 2.2 o conceito de otimização matemática é abordado, bem como a classificação de problemas de otimização quanto à continuidade. Na seção 2.3, o algoritmo de otimização que será utilizado para resolver esse problema é colocado. Na seção 2.4 a distribuição de Lévy é explorada. Na seção 2.6 um exemplo prático de otimização por Busca de Cuco via voos de Lévy(*Cuckoo Search via Lévy Flights*) (BC) é explorado. E por último, na seção 2.5 alguns conceitos importantes dos algoritmos genéticos são revisados.

2.1 Definição do Problema

O processo de identificação de sistemas é iterativo com essas atividades principais: Testes dinâmicos e aquisição de dados - Escolha da classe do modelo e estrutura - Estimação de parâmetros - Validação do modelo conforme pode ser visualizado na Figura 1.

Cada atividade representada na Figura 1 está sujeita a conter erros. Por esse motivo um sistema somente é considerado identificado uma vez que ele esteja razoavelmente validado. Como as modelagens de sistemas caixa branca e caixa cinza utilizam informações do modelo anterior, a qualidade dessas informações impacta diretamente nos resultados. Desta forma os erros das informações de entrada poderiam inviabilizar o atingimento dos critérios de desempenho desejados. Para evitar a análise de qualidade dessas informações, a modelagem caixa preta de sistemas foi apontada nessa pesquisa.

Este trabalho foca na atividade de estimação de parâmetros (penúltima etapa da Figura 1) e tem, como pré-requisito, a escolha da classe do modelo e estrutura.

A identificação de sistemas é embasada na construção de um modelo matemático M que se comporta aproximadamente como o sistema real S (AGUIRRE; BARBOSA; BRAGA, 2010).

Conforme pode ser observado na Figura 2, existe um conjunto de dados Z que inclui tanto sinais de entrada e saída do sistema S. Se os sinais de entrada de Z forem tomados e aplicados às entradas do modelo M, o conjunto de dados Z_M é produzido.

Então se torna possível definir uma função de custo $J(Z,Z_M)$ que mede quão longe o conjunto de dados Z_M está do conjunto de dados medidos Z. Minimizando-se $J(Z,Z_M)$, pela escolha dos parâmetros $\hat{\theta}$ do modelo M, obtém-se a estimação $\hat{\theta}$ que melhor aproxima M de S.

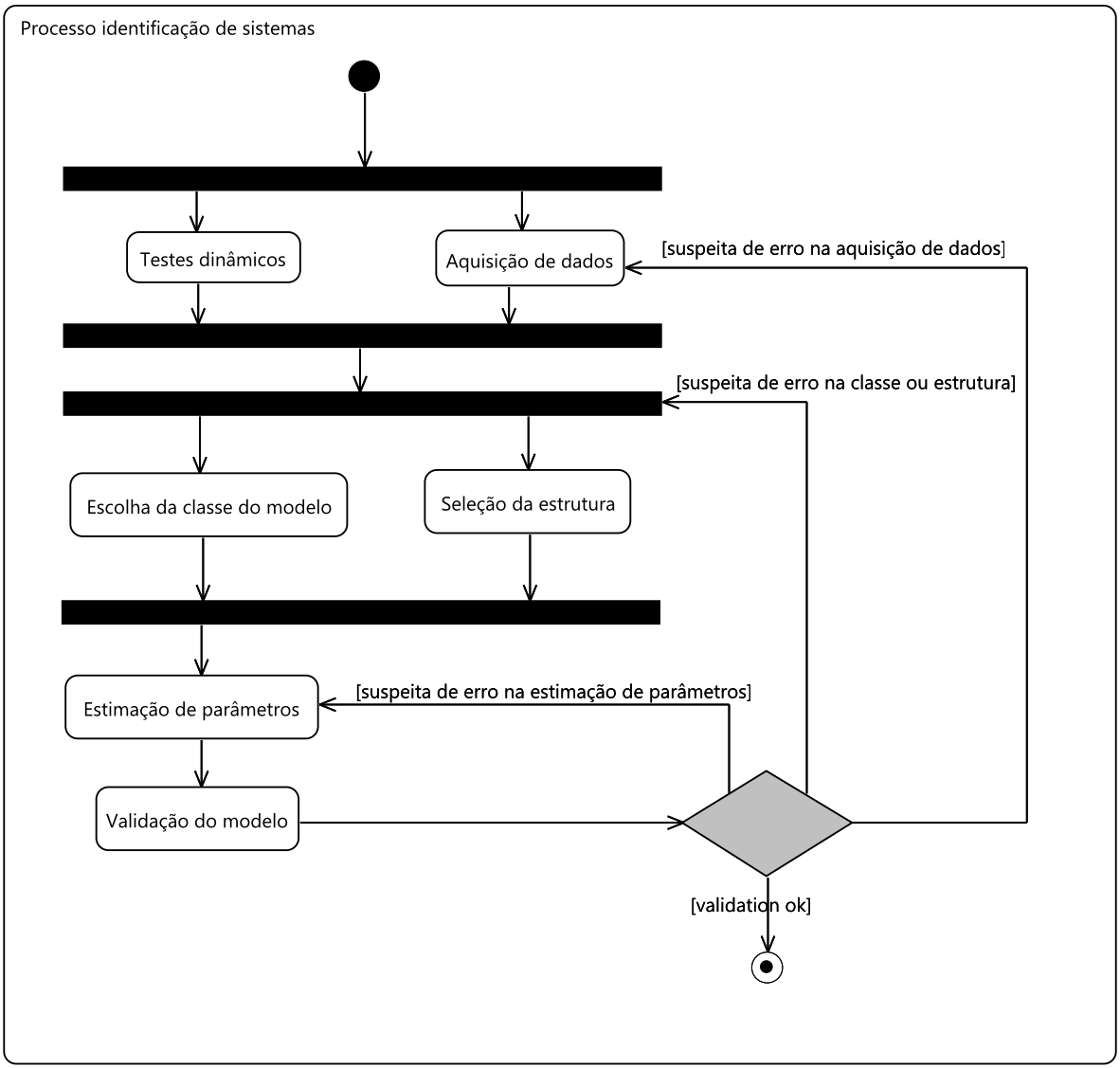

Figura 1 – Diagrama de atividade UML do processo de identificação de sistemas. Fonte o autor.

A minimização de $J(Z,Z_M)$ pode ser classificada como um problema de otimização contínua do tipo minimização. Essa classe de problemas da otimização matemática será abordada na próxima seção para consolidação de alguns conceitos importantes desse trabalho.

A abordagem média móvel auto regressiva não linear com entrada exógena(*Nonlinear Auto Regressive Moving Average with eXogenous input*) (NARMAX) será utilizada nesse trabalho porque além de permitir representar um amplo espectro de sistemas não lineares, ela é considerada uma filosofia de identificação de sistemas não lineares (BILLINGS, 2013).

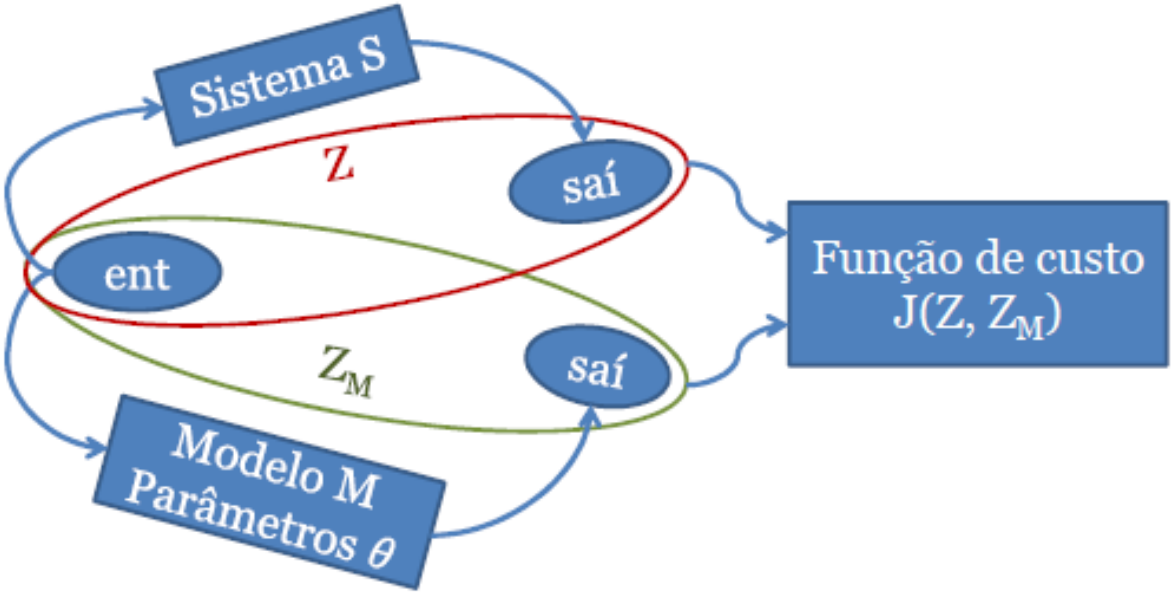

Figura 2 – Representação da identificação de sistemas. Fonte: o autor.

2.1.1 Abordagem NARMAX

A abordagem NARMAX é definida por uma classe de modelos do tipo NARMAX. Esses modelos podem representar uma ampla classe de sistemas não lineares (BILLINGS, 2013), e são definidos como

$$\begin{aligned} y(k) = F[&y(k-1), && y(k-2), \ldots, && y(k-n_y), \\ &u(k-d), && u(k-d-1), \ldots, && u(k-d-n_u), \\ &e(k-1), && e(k-2), \ldots, && e(k-n_e)] + e(k) \end{aligned}$$

onde $y(k)$, $u(k)$ e $e(k)$ são a saída do sistema, entrada e sequências de ruído respectivamente; n_y, n_u, e n_e são os atrasos máximos para a saída do sistema, entrada e ruído; $F[\cdot]$ é alguma função não linear, d é um atraso de transporte tipicamente configurado para $d = 1$. O modelo é essencialmente uma expansão de termos de entradas passadas, saídas e de ruído. Porque o ruído é modelado explicitamente, estimativas sem tendências do modelo do sistema podem ser obtidas na presença de ruído não observado altamente correlacionado e não linear. Desde que o NARMAX foi introduzido, provando-se qual classe de sistemas não lineares podem ser representados por esse modelo, muitos resultados e algoritmos tem sido derivados baseados nas expansões polinomiais do modelo NARMAX. Esses ainda são os métodos mais populares hoje, mas outras formas mais complexas baseadas em wavelets e outras expansões têm sido introduzidas para representar sistemas não lineares altamente complexos e severamente não lineares. Uma proporção significativa de sistemas não lineares podem ser representados por um modelo NARMAX inclusive sistemas com comportamentos exóticos como caos, bifurcações, e sub-harmônicas. Enquanto NARMAX começou como o nome de um modelo, ele agora desenvolveu-se para uma filosofia de identificação de sistemas não lineares (BILLINGS, 2013).

A identificação de sistemas pode ser dividida em dois propósitos. O primeiro envolve aproximação onde o propósito chave é desenvolver um modelo que aproxima o conjunto de dados de forma que boas previsões possam ser feitas. Existem muitas aplicações onde essa abordagem é apropriada, por exemplo na previsão de séries temporais do clima, preços de ações, fala, rastreamento de alvos, classificação de padrões, etc. Em aplicações assim, a forma do modelo não é tão importante. O objetivo é encontrar um esquema de aproximação que produza um mínimo de erros de previsão. Um segundo objetivo da identificação de sistemas, que inclui o primeiro objetivo como um subconjunto, envolve muito mais que somente encontrar um modelo para alcançar os menores erros. Esse segundo propósito é o motivo pelo qual a filosofia NARMAX foi desenvolvida e está ligada à idéia de encontrar a estrutura de modelo mais simples entre todas. O propósito aqui é desenvolver modelos que reproduzam as características dinâmicas do sistema em questão, para encontrar o modelo mais simples possível, e se possível relacionar isso a componentes e comportamentos do sistema sob estudo.

2.2 Otimização matemática

Na matemática, ciência da computação, ou ciência do gerenciamento, a otimização matemática (alternativamente, otimização ou programação matemática) é a seleção de um elemento melhor (em relação a alguns critérios) de algum conjunto de alternativas disponíveis (HOLDER, 2006–14).

No caso mais simples de todos, um problema de otimização consiste em maximizar ou minimizar uma função real escolhendo-se sistematicamente valores de entrada de dentro de um conjunto permitido e calcular o valor da função. A generalização da teoria de otimização e técnicas para outras formulações compreende uma ampla área da matemática aplicada. Mais geralmente, otimização inclui a busca dos valores melhores disponíveis de alguma função objetiva dado um domínio definido (ou um conjunto de limitações), incluindo uma variedade de diferentes tipos de funções objetivas e diferentes tipos de domínios.

2.2.1 Problema de otimização

Na matemática, e ciência da computação, um problema de otimização é o problema de encontrar-se a melhor solução dentre todas as soluções factíveis. Problemas de otimização podem ser divididos em duas categorias dependendo do domínio das variáveis se contínuo ou discreto. Um problema de otimização com variáveis discretas é conhecido como um problema de otimização combinacional. Em um problema desse tipo, busca-se um objeto como um inteiro, permutação ou gráfico de um conjunto finito (ou possivelmente infinito contável).

Esse trabalho utiliza uma técnica de otimização em que o domínio das variáveis é contínuo. A seguir, para efeito de comparação, os problemas de otimização contínua e discreta são brevemente abordados.

2.2.1.1 Problema de otimização combinacional

Formalmente, um problema de otimização combinacional A é um quádruplo (I, f, m, g), onde

- I é um conjunto de instâncias;
- dada uma instância $x \in I$, $f(x)$ é o conjunto de soluções factíveis;
- dada uma instância x e uma solução factível y de x, $m(x, y)$ denota a medida de y, que é normalmente um número real positivo;
- g é a função objetivo, e é min ou max.

O objetivo é então encontrar para alguma instância x uma solução ótima, que é, uma solução factível y com

$$m(x, y) = gm(x, y')|y' \in f(x).$$

Para cada problema de otimização combinacional, há um problema de decisão que pergunta se há uma solução factível para alguma medida particular m_0. Por exemplo, se há um gráfico G que contém vértices u e v, um problema de otimização poderia ser "encontrar um caminho de u para v que usa a menor quantidade de bordas". A resposta desse problema pode ser quatro, por exemplo. Um problema de decisão correspondente seria "há um caminho de u a v que usa dez ou menos bordas?" Esse problema pode ser resolvido com um simples 'sim' ou 'não'.

No campo dos algoritmos de aproximação, algoritmos são projetados para encontrar soluções próximas à ótima para problemas difíceis. A versão de decisão convencional é então uma definição inadequada do problema visto que ela somente especifica soluções aceitáveis. Embora problemas de decisões adequadas possam ser introduzidos, o problema é mais naturalmente caracterizado como um problema de otimização (AUSIELLO, 1999).

2.2.1.2 Problema de otimização contínua

A forma padrão de um problema de otimização é (BOYD, 2004)

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimizar}} \quad & f(\mathbf{x}) \\ \text{sujeito a} \quad & g_i(\mathbf{x}) \leq 0, \ i = 1, \ldots, m \\ & h_i(\mathbf{x}) = 0, \ i = 1, \ldots, p \end{aligned}$$

onde

- $f(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$ é a função objetivo a ser minimizada sobre a variável $\mathbf{x}$,
- $g_i(\mathbf{x}) \leq 0$ são chamadas restrições de desigualdade, e
- $h_i(\mathbf{x}) = 0$ são chamadas restrições de igualdade.

Por convenção, a forma padrão define um problema de minimização. Um problema de maximização pode ser tratado negando-se a função objetiva.

Para resolver o problema da estimação de parâmetros descrito na seção 2.1, um algoritmo de otimização contínua se faz necessário. Por esse motivo e devido a suas características de desempenho e inovação, o algoritmo selecionado foi o Busca de Cuco via voos de Lévy(*Cuckoo Search via Lévy Flights*), e que será introduzido na próxima seção.

2.3 Busca de cuco via voos de Lévy

Foi originalmente proposto por (YANG; DEB, 2009) e comparado com otimização por enxame de partículas (EP) e algoritmos genéticos (AG) onde mostrou resultados superiores sobre uma ampla gama de problemas de otimização contínua.

2.3.1 Comportamento de Procriação dos Cucos

Os cucos são pássaros que não criam seus filhotes. Ao invés, as mamães cuco colocam seus ovos nos ninhos de fêmeas de outras espécies esperando que elas incubarão os filhotes de cuco como seus. Porém se a anfitriã descobre que os ovos não são dela, ou ela simplesmente descarta os ovos estranhos, ou abandona o ninho e constrói um novo ninho em outro lugar.

2.3.2 Algoritmo Busca de Cuco

O algoritmo segue três regras ideais (YANG; DEB, 2009):

1. Cada cuco bota um ovo de cada vez, e bota seu ovo num ninho escolhido aleatoriamente. Isso é feito substituindo-se o ovo antigo por outro gerado através de um voo de Lévy aplicado sobre o ovo anterior.

2. Os melhores ninhos são transferidos às próximas gerações. Essa regra é responsável por garantir a convergência da solução através de elitismo.

3. O número de ninhos anfitriões é fixo, e o ovo botado pelo cuco é descoberto pela anfitriã com uma probabilidade p_a.

2.3.2.1 Pseudo Código

Segue o pseudo código que explica o funcionamento do algoritmo (YANG; DEB, 2009).

. . . . . .
início
*Função objetivo f(**x**), **x** = (x_1, ..., x_d)^T;*
Gera uma população inicial de
n ninhos anfitriões $\mathbf{x}_i$ (i = 1, 2, ..., n);
enquanto ($t <$GeraçãoMáxima) **ou** (critério de parada)
Toma um cuco aleatoriamente por voos de Lévy;
avalia a sua qualidade/pertinência Pi;
Escolhe um ninho j entre n aleatoriamente;

se ($P_i > P_j$),
substitui j pela nova solução;
fim se
Uma fração (p_a) dos piores ninhos são abandonados
e novos são construídos;
Mantém melhores soluções
(ou ninhos com soluções de qualidade);
Ordena as soluções e acha a melhor atual;
fim enquanto
Pós processa os resultados e visualização;
fim início

A análise dos arquivos fonte disponibilizados pelos autores (YANG, 2010a), possibilitou o levantamento do diagrama da Figura 3, representado na linguagem de modelagem de software UML (linguagem de modelagem universal (*Universal Modeling Language*)).

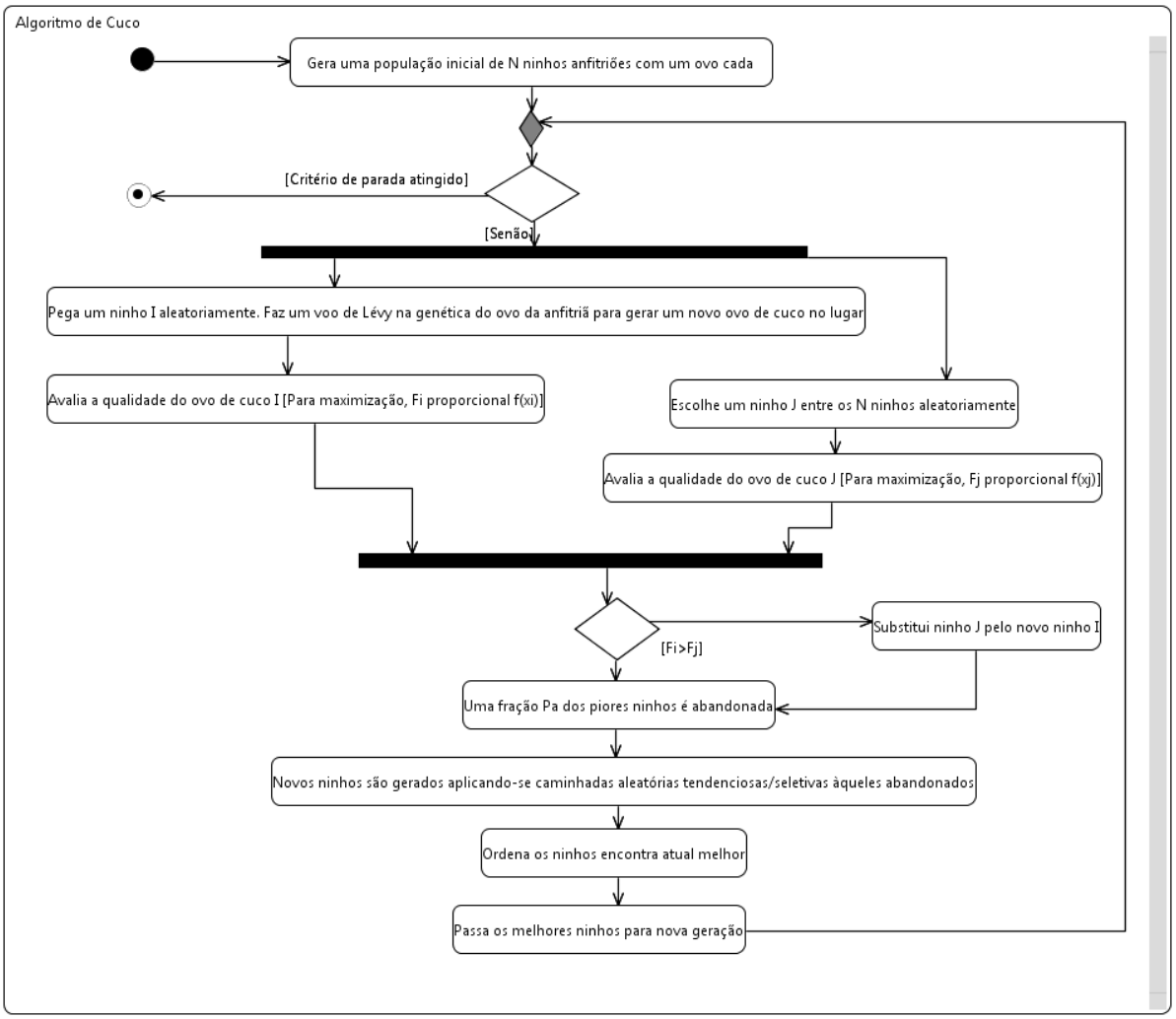

Figura 3 – Diagrama de atividade UML do algoritmo busca de cuco via voos de Lévy.

2.3.3 Voo de Lévy

Um contribuidor do desempenho do algoritmo BC é o voo de Lévy que além de estar intrínseco ao método BC, é um tipo de caminhada aleatória responsável por explorar o espaço de busca por soluções. Muitos animais às vezes desenvolvem voos com características típicas do voo de Lévy. O voo de Lévy é uma busca espacial eficiente em termos de distância percorrida em que as mudanças de direção são isotrópicas (probabilidade de escolha de determinada direção igual a qualquer outra direção) e aleatórias, os segmentos da trajetória são retilíneos e seus comprimentos seguem uma distribuição de Lévy (MANDELBROT, 1983). Os tubarões, por exemplo, alternam entre o movimento browniano quando a caça é abundante em uma determinada região próxima e o movimento por voo de Lévy quando a caça está escassa nessa região(HUMPHRIES et al., 2010). Um exemplo do voo de Lévy com cincoenta passos pode ser observado na Figura 4 com a origem como ponto de partida. Já a Figura 5 demonstra um voo do tipo movimento browniano.

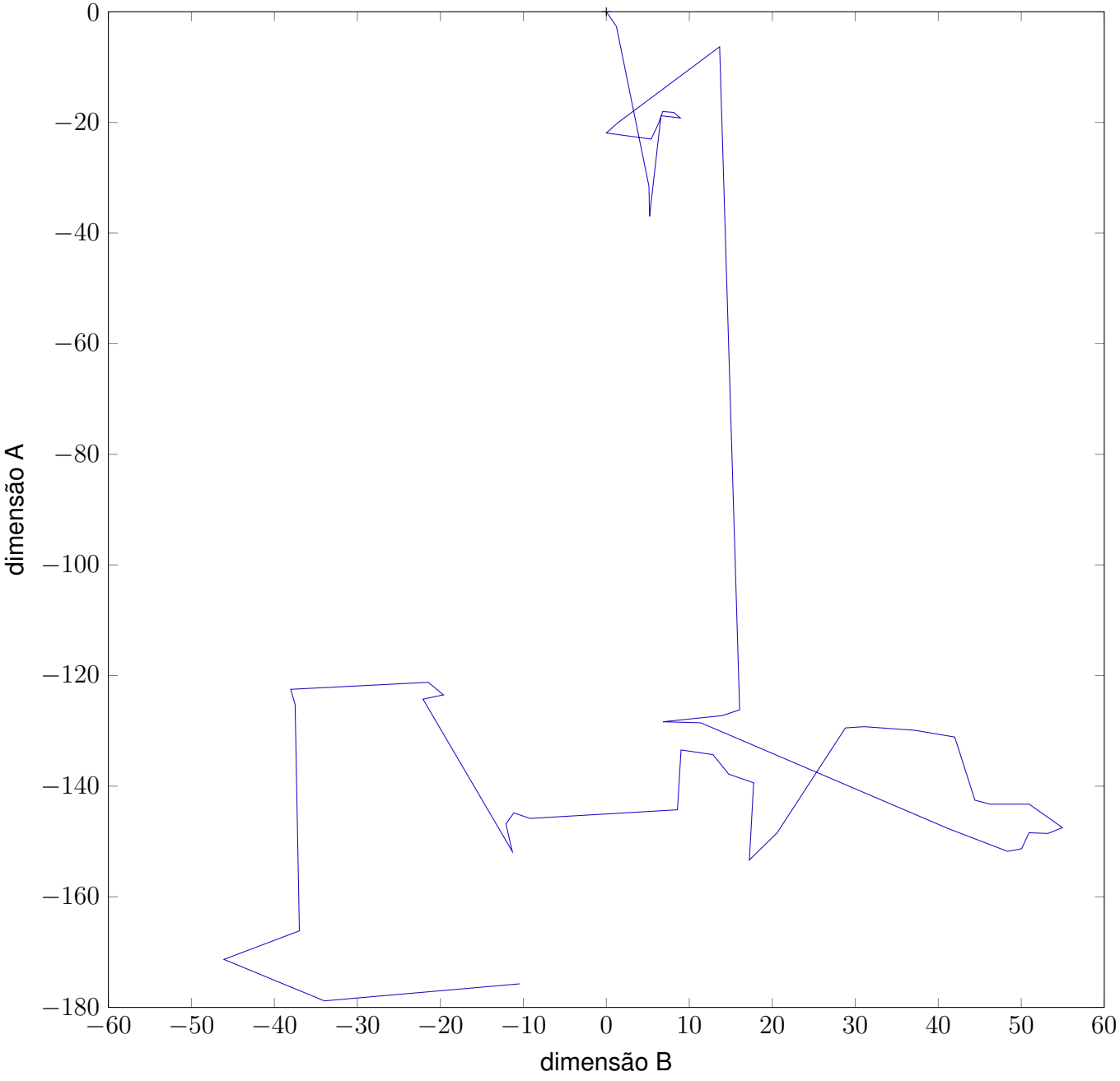

Figura 4 – Um exemplo de 50 passos de um voo de Lévy em duas dimensões. Fonte: O autor.

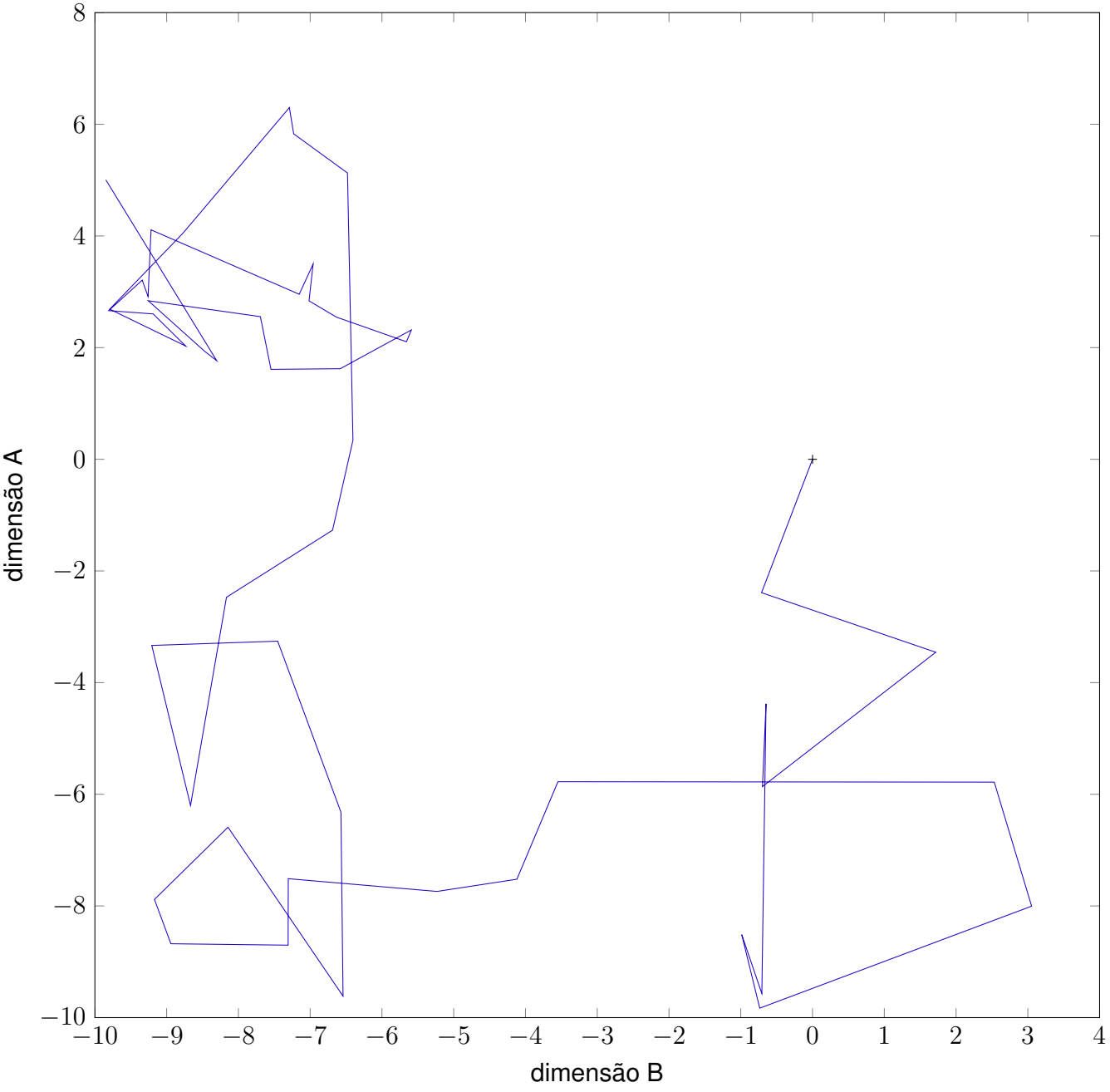

Figura 5 – Um exemplo de 50 passos de um voo do tipo movimento browniano em duas dimensões. Fonte: O autor.

O caso particular para o qual Mandelbrot (MANDELBROT, 1983) usou o termo “voo de Lévy” é definido pela função de sobrevivência da distribuição dos comprimentos dos passos, U, como pode ser visto na Equação 2.1, tal que

$$Pr(U > u) = \begin{cases} 1 & : u < 1, \\ u^{-D} & : u \geq 1. \end{cases} \tag{2.1}$$

Ou seja, a probabilidade de que a distribuição dos comprimentos dos passos U seja maior que o comprimento u é ora 1, ora u^{-D}. Onde D é um parâmetro relacionado à dimensão fractal e a distribuição é um caso particular da distribuição de Pareto (**??**).

Os voos de Lévy são processos estocasticamente auto-similares (**??**), e por isso apresentam um parâmetro de dimensão fractal. Uma dimensão fractal é uma razão que fornece um índice estatístico de complexidade que compara como o detalhe em um padrão muda com a escala em que é medido (**??**). O termo fractal é definido como um fenômeno natural ou conjunto matemático que exibe um padrão de repetição em cada escala (**??**).

Pesquisadores, mais tarde, reformularam a Equação 2.1 permitindo que a distribuição dos comprimentos dos passos seja qualquer distribuição para a qual a função de sobrevivência tem uma cauda do tipo de potência (**??**) conforme pode ser visualizado na Equação 2.2.

$$Pr(U > u) = O(u^{-k}), \quad \text{para } k \text{ que satisfaz } 1 < k < 3. \tag{2.2}$$

Aqui a notação O é a notação matemática do Grande-O (BEIGI, 2011). Na matemática, a notação O descreve o comportamento limitante de uma função quando o argumento tende a um valor ou infinito, geralmente em termos de funções mais simples. Ele é um membro de uma família maior de notações chamada notação Landau, Bachmann-Landau ou notação assintótica. Um exemplo da notação O pode ser observado na Figura 6.

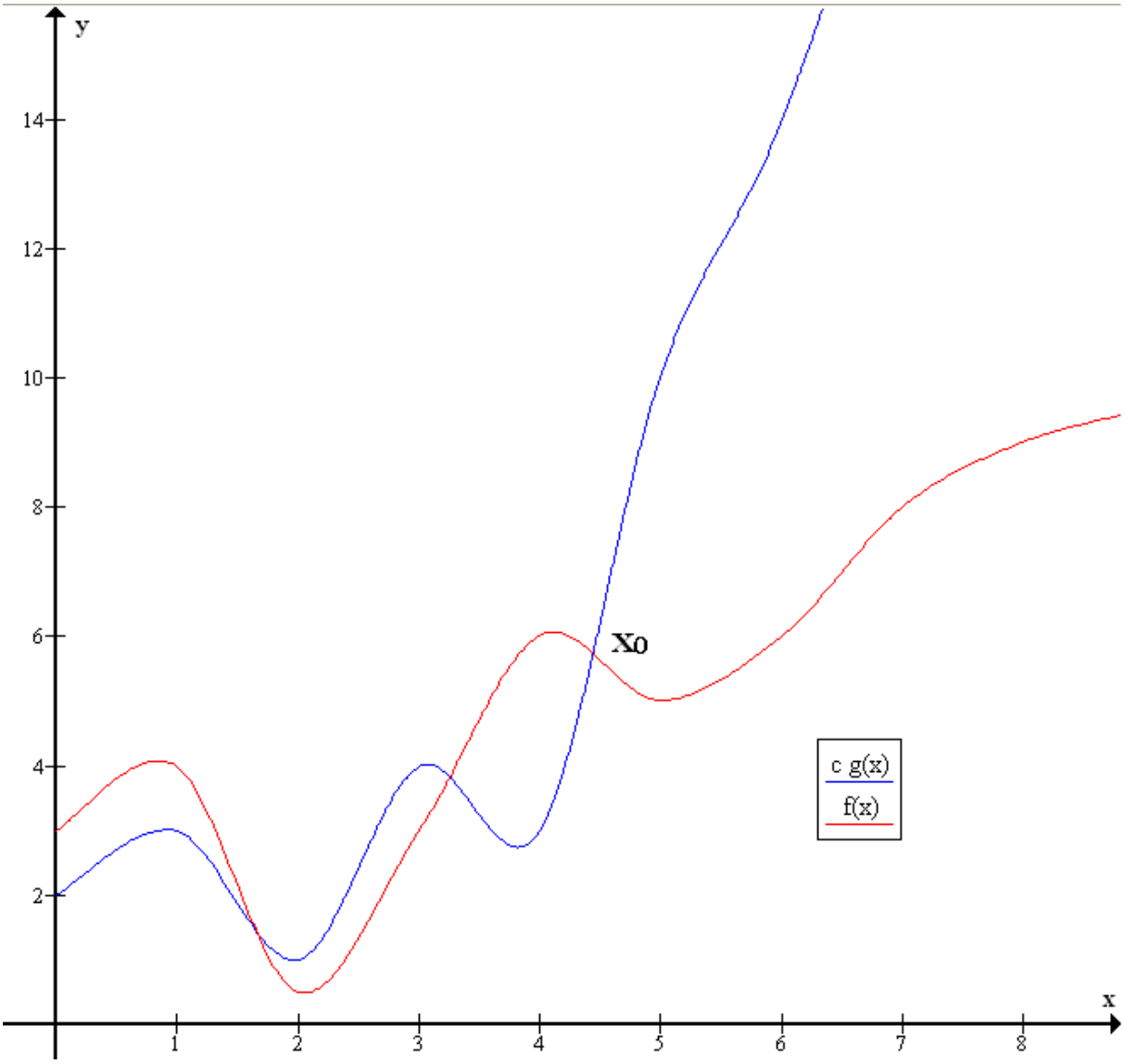

Figura 6 – Exemplo de notação O: $f(x) \in O(g(x))$ assim como existe $c > 0$ (ex., $c = 1$) e x_0 (ex., $x_0 = 5$) de forma que $f(x) < cg(x)$ sempre que $x > x_0$.

2.4 Distribuição de Lévy

A distribuição de Lévy é um caso particular das distribuições α-estáveis (SAMORADNITSKY; TAQQU, 1994). As distribuições α-estáveis apresentam várias características que fazem sua utilização complicada, incluindo médias e variâncias infinitas, e o fato de que a função densidade de probabilidade (*Probability Density Function*) (pdf) ou a função de distribuição cumulativa (*Cumulative Distribution Function*) (cdf) não poderem ser escritas na forma fechada.

Distribuições α-estáveis aparecem naturalmente no estudo das distribuições de cauda pesada, e têm encontrado aplicações na economia e física como modelos de eventos raros, mas extremos (como terremotos ou quebras de bolsas de valores). Muitos acreditam que alguns dos problemas recentes financeiros aconteceram porque os analistas limitaram-se a utilizar modelos gaussianos (que não possuem caudas pesadas) (**??**).

2.4.1 Parametrização

Na literatura, a parametrização de distribuições α-estáveis não é consistente (há mais de seis de parametrizações diferentes). A parametrização escolhida aqui está em linha com (SAMORADNITSKY; TAQQU, 1994). A maneira mais comum de especificar a parametrização é observar a função característica da variável aleatória α-estável, que será feito na próxima subseção.

2.4.2 Introdução à distribuição α-estável

A distribuição α-estável é na verdade uma família de distribuições de quatro parâmetros e é usualmente denotada por $S(\alpha,\beta,\gamma,\delta)$. O primeiro parâmetro $\alpha\in[0,2]$ é chamado expoente característico, e descreve a cauda da distribuição. O segundo $\beta\in[-1,1]$ é a assimetria, e conforme o nome implica, especifica se a distribuição é assimétrica à direita ($\beta>0$) ou à esquerda ($\beta<0$). Os últimos dois parâmetros são a escala, $\gamma>0$, e a localização $\delta\in\mathbb{R}$. Pode-se considerar essas duas como sendo similares à variância e à média na distribuição normal no seguinte sentido - se $Z\sim S(\alpha,\beta,1,0)$, então se $\alpha=1$, $\gamma.Z+\delta\sim S(\alpha,\beta,\gamma,\delta)$. A variável Z é usualmente chamada uma variável aleatória α-estável padrão. A família de distribuições α-estáveis é uma classe rica, e inclui as seguintes distribuições como subclasses (SAMORADNITSKY; TAQQU, 1994):

1. Distribuição gaussiana $N(\mu,\sigma^2)$ é dada por $S(2,\beta,\frac{\sigma^2}{\sqrt{2}},\mu)$. Note que β não importa nesse caso.

2. Distribuição de Cauchy com escala γ e localização δ é dada por $S(1,0,\gamma,\delta)$.

3. Distribuição de Lévy (também conhecida como gaussiana inversa ou Pearson V), com escala γ e localização δ é dada por $S(\frac{1}{2}, 1, \gamma, \delta)$.

2.4.3 Função STBLRND

Utilizando-se os conceitos de (CHAMBERS; MALLOWS; STUCK, 1976) e (Weron; Weron, 1995), (VEILLETTE, 2012) construiu no MATLAB a função SBTLRND para gerar números aleatórios com uma distribuição α-estável. Essa função é chamada conforme Figura 7:

```
X = stblrnd(alpha, beta, gama, Δ) % Gera uma única variável aleatória
% S(alfa, beta, gama, Δ).
X = stblrnd(alfa, beta, gama, Δ, M) %Gera uma matriz M por M de
% variáveis aleatórias S(alfa, beta, gama, Δ).
X = stblrnd(alfa, beta, gama, Δ, M, N,..) %Gera uma matriz M por N de
% variáveis aleatórias S(alfa, beta, gama, Δ).
X = stblrnd(alfa, beta, gama, Δ, [M,N,..]) %Gera uma matriz M por N
% por.. de variáveis aleatórias S(alfa, beta, gama, Δ).
```

Figura 7 – Função STBLRND.

Na Figura 7, as entradas alfa, beta, gama e Δ devem ser escalares e pertencer aos seus intervalos apropriados. Se a escolha do parâmetro corresponder a uma distribuição especial (Gaussiana, Cauchy, Lévy), então um método mais rápido é utilizado.

2.4.4 Utilização da função STBLRND para desempenhar voos de Lévy

O código da Figura 8 foi construído a fim de testar um voo característico de Lévy. Seu resultado para os parâmetros $\alpha = 0.5$, $\beta = 1$, $\gamma = 1.5$, $\delta = 1$ pode ser observado na Figura 4. Modificando-se a variável voo para $Brownian$, é possível também desempenhar um movimento do tipo browniano em que o resultado para os parâmetros $\alpha = 1.5$, $\beta = 0$, $\gamma = 1.5$, $\delta = 1$ pode ser observado na Figura 5. Observe que nesse movimento a distribuição α-estável é parametrizada como uma distribuição Gaussiana.

```
config = 'new';
%config = 'keep';

voo = 'Levy';
%voo = 'Brownian';

switch config
case 'new'
    switch voo
    case 'Brownian'
        alpha = 2;
        beta = 0;
    case 'Levy'
        alpha = 1/2;
        beta = 1;
    end
    gamma = 1.5;
    Δ  = 1;

    N = 50;
    s = [0 0];

    steps = stblrnd(alpha,beta,gamma,Δ,N, 2);
    %   step = mantegna(1.5, 1, 1, zeros(N, 2));

    vin = randn(N, 2);
    randomDistributedVersors = bsxfun(@rdivide,vin,sqrt(sum(vin.^2, ...
        2)));

    trajectory(1,:) = s;
    for i = 2 : N
        aStep = steps(i, :);
        aRandomDistributedVersor = randomDistributedVersors(i,:);

        s=s+aStep.*aRandomDistributedVersor;
        trajectory(i,:) = s;
    end
     save(strcat(voo,'.mat'), 'trajectory');
case 'keep'
     load(strcat(voo,'.mat'), 'trajectory');
end


figure;
plot(trajectory(:,1), trajectory(:,2));
hold
plot(0, 0, 'k+');

ylabel('dimensão A');
xlabel('dimensão B');
```

Figura 8 – Código utilizado para fazer voo de Lévy na Figura 4 e movimento Browniano na Figura 5.

2.5 Algoritmos Genéticos

Os fundamentos e aplicações dos algoritmos genéticos podem ser encontrados em muitos artigos como (RAWLINS, 1991; GREFENSTETTE; BAKER, 1989; GOLDBERG, 1989). O AG se refere à computação evolucionária baseada nas idéias da genética. O AG é inspirado pela teoria da evolução de Darwin que resolve um problema imitando um processo evolucionário. O espaço de todas as soluções factíveis é chamado de espaço de busca. Com um AG, o problema a ser resolvido é projetado no espaço de busca, e a melhor solução entre um número de possíveis soluções é identificada durante o procedimento de evolução. Um cromossomo que consiste de uma sequência de genes representa uma solução do problema. Primeiramente um cromossomo deve ser codificado, e que será bem específico para um problema particular devido ao fato de conter informações da natureza do problema. Um cromossomo também deveria de alguma forma conter informação sobre a solução que ele representa. A forma mais comum de codificação é uma sequência binária. Na genética um cromossomo consiste de genes, i.e. blocos de DNA. Então um gene é um segmento de um cromossomo e carrega algumas informações genéticas particulares. De acordo com esses genes, uma função de pertinência e uma função de codificação são estabelecidas. A função de codificação traduz os genes (códigos binários) em uma solução real. A função de pertinência avalia a pertinência da solução baseada na solução atual e no conjunto de dados de treinamento. As formas gerais de uma função de pertinência e função de codificação são representadas como

$$V_f = f(t_i, X), \tag{2.3}$$

$$t_i = C(ch_i), \qquad i = 0, 1, 2 \ldots n \tag{2.4}$$

onde V_f é o valor de pertinência, e normalmente não deveria ser menor que zero, X o conjunto de dados de treinamento e t_i o valor de codificação de um cromossomo, ch_i.

Operações são desempenhadas sobre esses códigos que fazem a população inteira evoluir. São os operadores elementares do efeito AG sobre a população:

- **Seleção**: selecionar um cromossomo que tem um valor de pertinência elevado da população atual, e copiá-lo na nova população.
- **Cruzamento**: selecionar dois cromossomos que tem valores de pertinência elevados da população atual, trocar alguns bits deles e copiá-los na nova população. As localizações desses bits são aleatórias.
- **Mutação**: selecionar um cromossomo que tem um valor de pertinência elevado da população atual, alterar alguns bits dele e copiá-lo na nova população.

2.6 Exemplo prático otimização por busca de cuco via voos de Lévy

Nessa seção, um exemplo prático de utilização da Busca de Cuco via voos de Lévy(*Cuckoo Search via Lévy Flights*) é abordado de forma a proporcionar aos pesquisadores uma curva de aprendizado mais rápida deste novo algoritmo de otimização. Em paralelo a otimização por algoritmos genéticos também é aplicada para efeito de comparação. Além disso, para facilitar o comparativo, o código original de (YANG, 2010c) foi alterado com o objetivo de tornar as chamadas à função principal (código A.1) mais parecidas possíveis às chamadas de função do algoritmos genéticos presentes no ambiente MATLAB.

A função selecionada para otimização foi a função d-dimensional de Langermann (ADORIO; DILIMAN, 2005; MOLGA; SMUTNICKI, 2005).

2.6.1 Função de Langermann

2.6.1.1 Equação matemática

$$f(x) = \sum_{i=1}^{m} c_i . e^{\left(-\frac{1}{\pi} . \sum_{j=1}^{d} (x_j - A_{ij})^2 \right)} . cos\left(\pi . \sum_{j=1}^{d} (x_j - A_{ij})^2 \right) \tag{2.5}$$

2.6.1.2 Descrição

A função de Langermann é multimodal, com muitos máximos locais distribuídos não uniformemente. Os valores recomendados de m, **c** e **A**, conforme (MOLGA; SMUTNICKI, 2005) são (para $d = 2$): $m = 5$, $\mathbf{c} = (1, 2, 5, 2, 3)$ e:

$$\mathbf{A} = \begin{pmatrix} 3 & 5 \\ 5 & 2 \\ 2 & 1 \\ 1 & 4 \\ 7 & 9 \end{pmatrix}$$

2.6.1.3 Domínio de entrada

A função é normalmente avaliada no hypercubo $x_i \in [0, 10]$, para todo $i = 1, \ldots, d$.

2.6.1.4 Código

O código de disponibilizado por (SURJANOVIC; BINGHAM, 2013) foi utilizado para a geração da Figura 9.

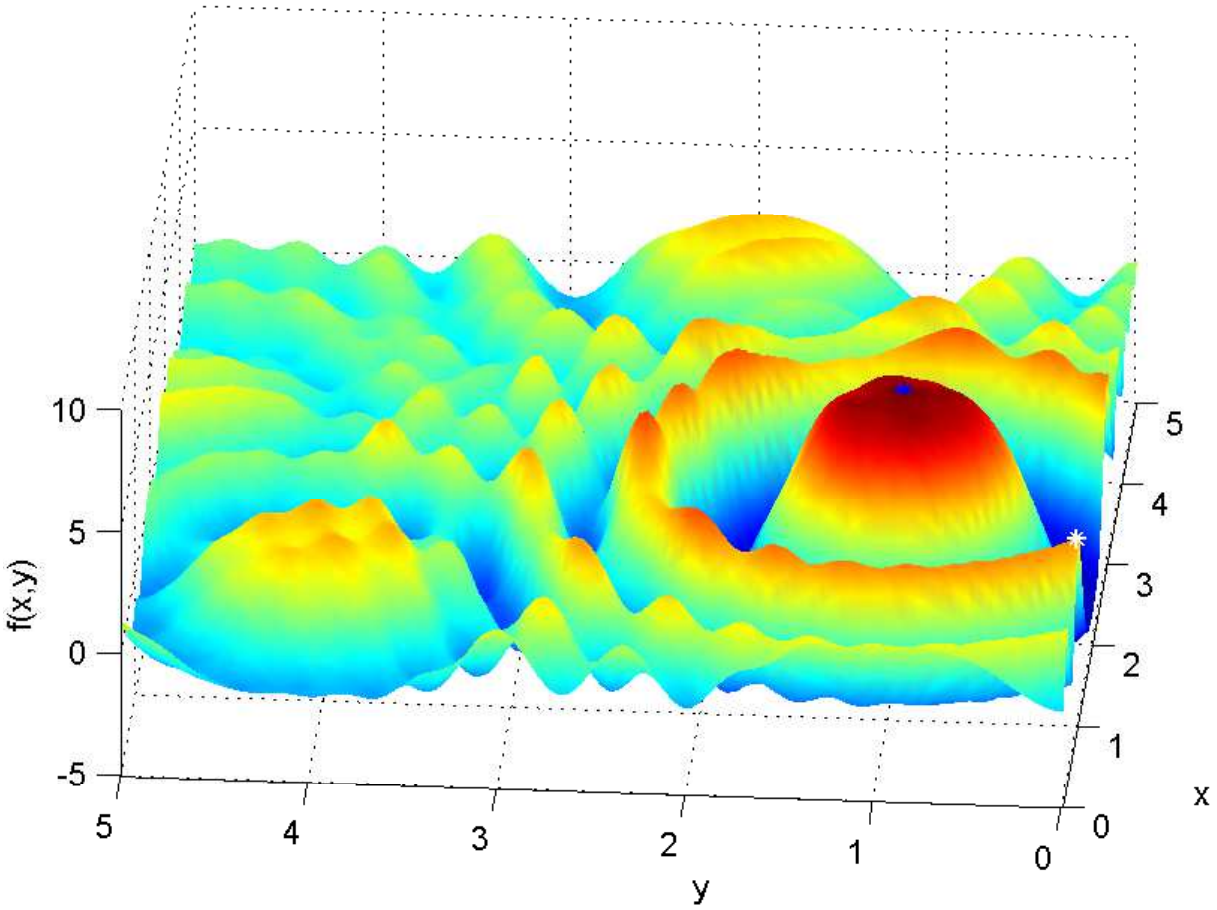

Figura 9 – Função de Langermann com 2 dimensões.

2.6.1.5 Comparativo MATLAB

O comparativo é feito com objetivo de encontrar o máximo global da função de Langermann de duas dimensões. Para rodar as otimizações por algoritmos genéticos e Busca de Cuco via voos de Lévy(*Cuckoo Search via Lévy Flights*), os seguintes arquivos se fazem necessários:

- cuckoo_search.m (código Apêndice A.1)
- mantegna.m (código Anexo A.1)
- mainScript.m (código Apêndice A.4)
- langer.m (código Anexo A.2)
- langer3.m (código Apêndice A.3)
- langer2.m (código Apêndice A.2)

O arquivo principal é o roteiro mainScript.m, que por sua vez faz as chamadas aos outros arquivos. Ou seja, assim que o roteiro mainScript.m é executado, o resultado já pode ser visualizado nas figuras resultantes.

Uma figura como a Figura 10 é atualizada automaticamente enquanto a evolução por Busca de Cuco via voos de Lévy(*Cuckoo Search via Lévy Flights*) estiver em curso.

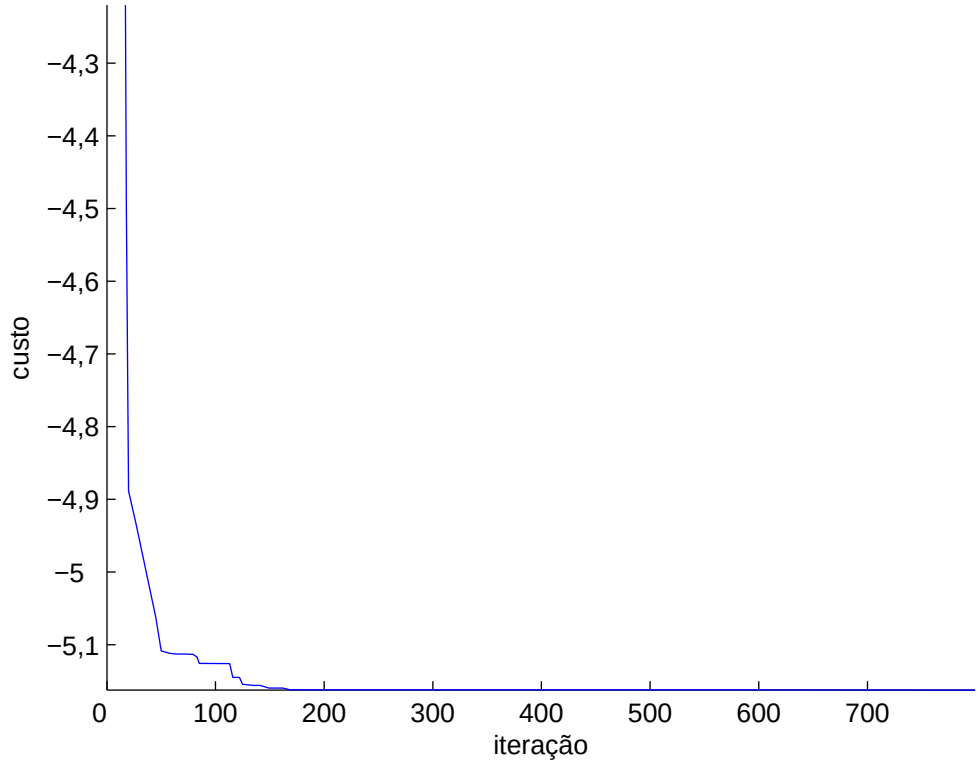

Figura 10 – Gráfico de evolução da Busca de Cuco via voos de Lévy(*Cuckoo Search via Lévy Flights*).

Os resultados da otimizações são inseridos no gráfico da Figura 9 colocando-se um marcador branco para o resultado da otimização por algoritmos genéticos e um marcador azul para o resultado da otimização por Busca de Cuco via voos de Lévy(*Cuckoo Search via Lévy Flights*).

Conforme pode ser observado na Figura 9, a otimização por AG, nessa rodada em particular, indicou um máximo local da função de Langermann de duas dimensões, enquanto a otimização por Busca de Cuco via voos de Lévy(*Cuckoo Search via Lévy Flights*) indicou o seu máximo global.

Para avaliar estatísticamente os resultados, as otimizações por AG e por BC da função de Langermann de duas dimensões foram realizadas mil vezes. Com esses dados os histogramas tridimensionais da Figura 11 e da Figura 12 foram obtidos. Os eixos x e y são as coordenadas apontadas como resultado pelo método de otimização aplicado à

função de Langermann.

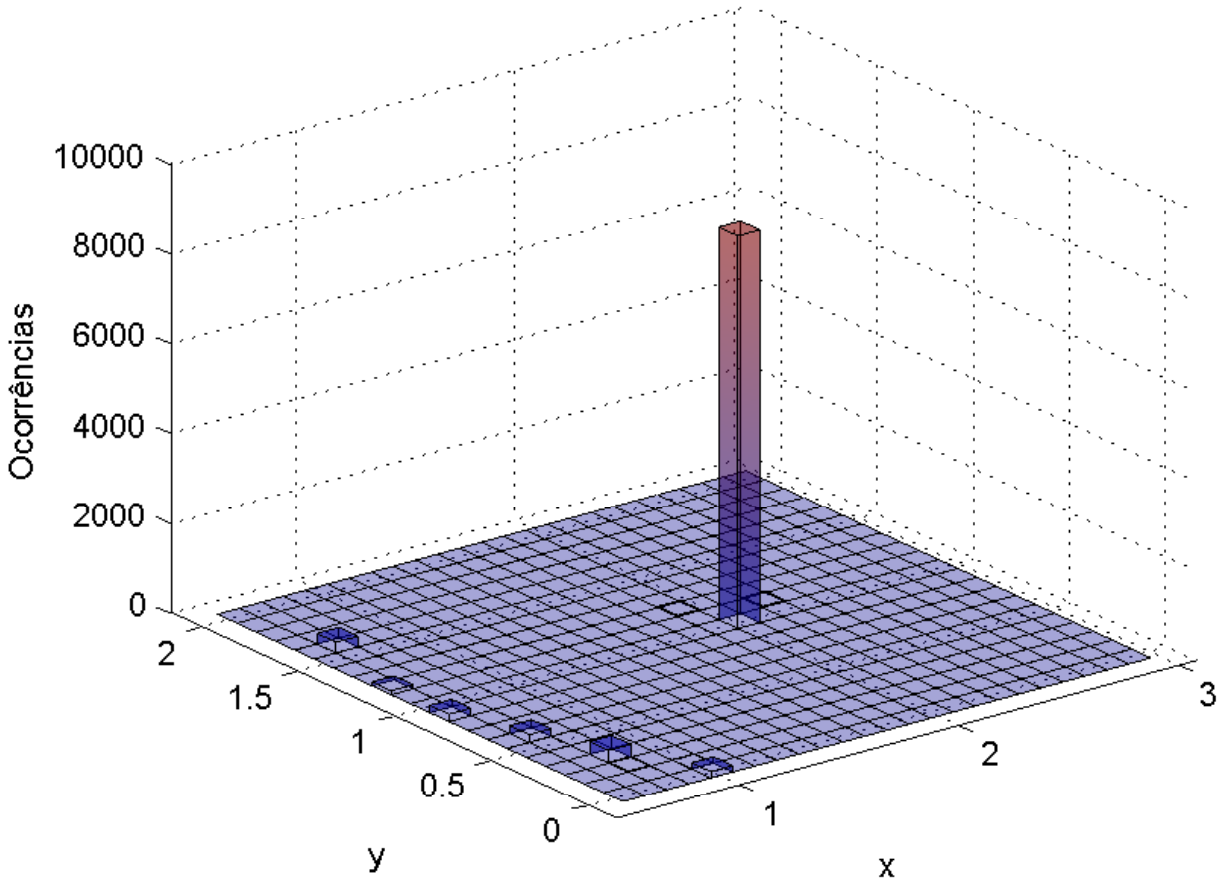

Figura 11 – Histograma para 1000 rodadas de otimização por AG da função de Langermann com 2 dimensões.

Pode-se observar na Figura 11, que com o método AG, alguns mínimos locais foram apontados como resultado. Já na Figura 12 pode-se observar que todas as mil rodadas de otimização com o método BC apontaram para o máximo global da função de Langermann bidimensional.

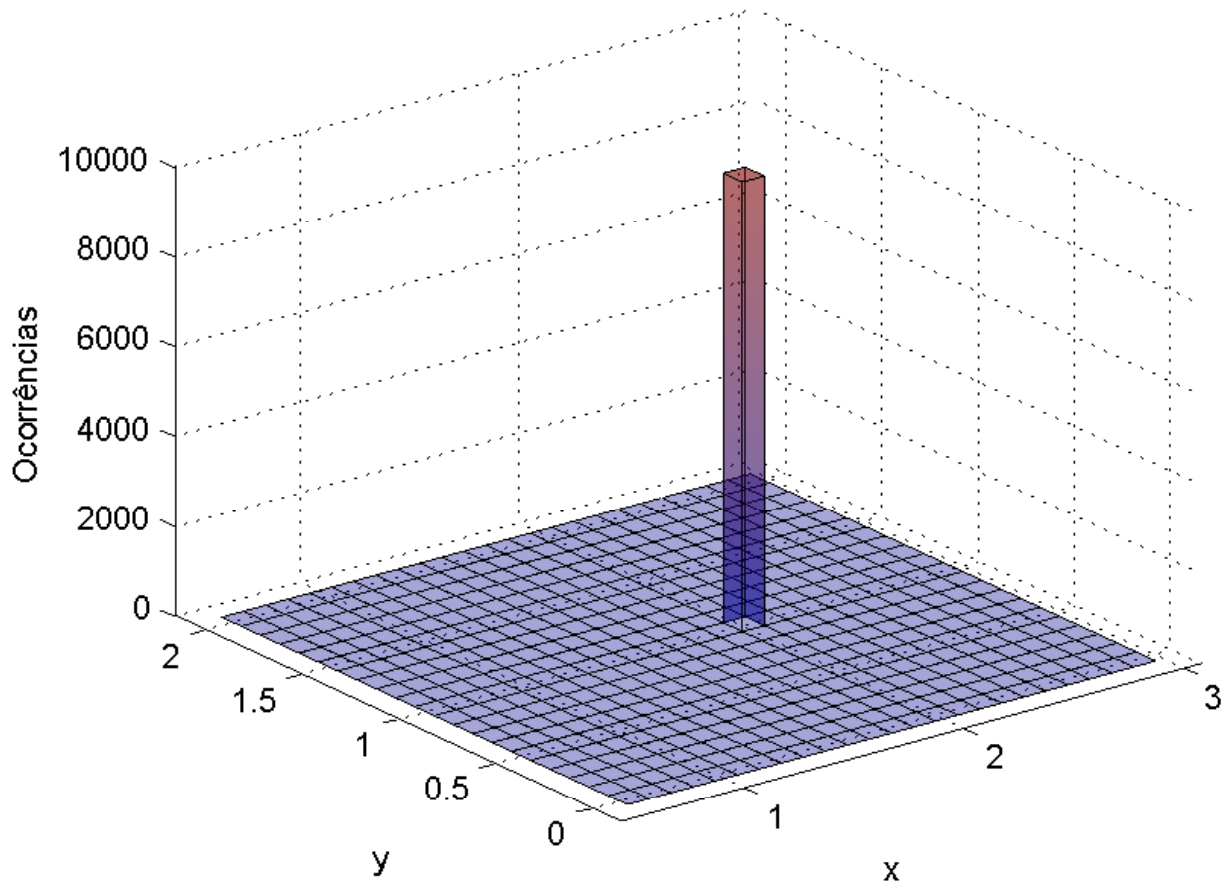

Figura 12 – Histograma para 1000 rodadas de otimização por BC da função de Langermann com 2 dimensões.

3 Metodologia

Objetiva-se aqui demonstrar na seção 3.1, como o BC foi configurado e aplicado na identificação de parâmetros. Já na seção 3.2, demonstra-se o desenvolvimento das melhorias construídas no algoritmo BC com foco em estimação de parâmetros.

3.1 BC aplicada à Identificação de Sistemas

Com uma estrutura matemática já conhecida e constante, a relação entre os sinais de entrada e saída de Z_M será função exclusiva do vetor de parâmetros estimados $\hat{\theta}$. É aqui que entra o Busca de Cuco via voos de Lévy(*Cuckoo Search via Lévy Flights*) (BC). Como um algoritmo de otimização, ele será responsável por resolver o problema apresentado na seção 2.1 através da evolução de $\hat{\theta}$ de forma a minimizar a função de custo $J(Z,Z_M(\hat{\theta}))$ dado o conjunto de dados Z.

3.1.1 Critério de parada

Assim como os algoritmos genéticos, a Busca de Cuco via voos de Lévy(*Cuckoo Search via Lévy Flights*) também precisa de um critério de parada para interromper a evolução da população de soluções. Em geral, dois critérios de parada distintos são usados em algoritmos evolucionários:

1. Parada quando o algoritmo atinge um determinado número de avaliações da função objetivo.
2. Parada quando a evolução da população entra em saturação, i.e., quando a evolução média de cada geração se torna muito pequena.

O critério de parada que é proposto aqui para identificação leva em consideração a saturação da taxa média de evolução. E os motivos são:

1. Geralmente, para a identificação de sistemas práticos, as coordenadas do erro mínimo global são desconhecidas.
2. Quando a evolução da população entra em saturação, é provável que que a melhor solução até então esteja próxima do mínimo global.

A saturação da taxa média de evolução será configurada por dois parâmetros, *StallGenLimit* e *TolFun*. Quando a taxa de evolução das últimas *StallGenLimit* gerações for menor que *TolFun*, o algoritmo interrompe novas gerações.

3.2 Melhorias na BC

Através do estudo aprofundado do algoritmo original (YANG, 2010c) da Busca de Cuco via voos de Lévy(*Cuckoo Search via Lévy Flights*) foi possível identificar duas frentes de melhoria no cálculo dos voos de Lévy:

1. Simulação numérica de processos estocásticos estáveis de Lévy
2. Cálculo de direção isotrópica

Para tal o código disponibilizado pelo próprio autor (YANG, 2010a) foi utilizado como ponto de partida.

3.2.1 Simulação numérica de processos estocásticos estáveis de Lévy

No passo da geração de um novo cuco por voo de Lévy, todos os ovos, exceto o melhor, são substituídos com base na qualidade de novos ovos de cuco produzidos com os voos de Lévy a partir do ovo atual conforme a Equação 3.1:

$$ovo_i^{(t+1)} = ovo_i^{(t)} + f.S.(ovo_i^{(t)} - ovo_{melhor}^{(t)}).r \tag{3.1}$$

onde $ovo_i^{(t)}$ é o i-ésimo ovo, f é o parâmetro de tamanho do passo; r é um número aleatório de uma distribuição normal padrão e ovo_{melhor} é o melhor ovo até então; e S é uma caminhada aleatória baseada nos voos de Lévy.

O método utilizado por Yang em seu código apresenta um cálculo simplificado do comprimento de passo S do voo de Lévy conforme pode ser observado na Equação 3.2.

$$S = \frac{a}{\sqrt[\alpha]{\mid b \mid}} \tag{3.2}$$

Onde a e b são variáveis aleatórias independentes normalmente distribuídas com parâmetros de distribuição $N(0, \sigma_a^2)$ e $N(0, \sigma_b^2)$ respectivamente em que $\sigma_b = 1$ e σ_a é calculado através da Equação 3.3.

$$\sigma_a(\alpha) = \sqrt[\alpha]{\left(\frac{\Gamma(1+\alpha).sin(\frac{\pi.\alpha}{2})}{\Gamma(\frac{1+\alpha}{2}).\alpha.2^{\frac{\alpha-1}{2}}} \right)} \tag{3.3}$$

Onde Γ é a função de Euler Γ.

Para evitar possíveis efeitos colaterais de uma simplificação como essa, foi decidido utilizar na íntegra o método de (MANTEGNA, 1994).

Em comparativo com outros métodos computacionais para geração de números aleatórios com uma distribuição de Lévy, o método de Mantegna apresentou cálculos mais simples (LECCARDI, 2005), o que favorece uma implementação em hardware, como por exemplo um ASIC (circuito integrado de aplicação específica (*Application Specific Integrated Circuit*)), para obtenção de um processamento muito mais rápido. Caso uma implementação por hardware não seja uma opção de projeto, o algoritmo de McCulloch é mais indicado devido à sua maior velocidade de processamento por software. Outra opção seria utilizar o método vislumbrado por (CHAMBERS; MALLOWS; STUCK, 1976) e que foi finalmente codificado por (Weron; Weron, 1995).

O algoritmo selecionado para fazer o cálculo do comprimento dos passos do voo de Lévy é baseado no método de Mantegna e foi obtido no apêndice do artigo de Leccardi. Mategna, para alcançar uma convergência mais rápida para o processo estável de Lévy de índice α, propôs uma transformação não linear (equação 3.4) com parâmetros $K(\alpha)$ e $C(\alpha)$ apropriados.

$$w = \left\{ [K(\alpha) - 1] \, . e^{\frac{-v}{C(\alpha)}} + 1 \right\} . v \tag{3.4}$$

Onde v é uma variável estocástica intermediária na obtenção de uma segunda variável estocástica indepedente w utilizada para determinar o valor ótimo de $K(\alpha)$ resultando na Equação 3.6 (MANTEGNA, 1994).

É possível observar que o algoritmo de Leccardi, para calcular $C(\alpha)$, utiliza um método de cálculo polinomial de valores dados alguns pontos conhecidos, ao invés de resolver a integral da Equação 3.5 numericamente obtendo-se $C_1(\alpha)$ e $C_2(\alpha)$, conforme proposto por Mantegna.

$$\frac{1}{\pi . \sigma_a(\alpha)} . \int_0^\infty \sqrt[\alpha]{q} . \mathrm{e}^{\left[-\frac{q^2}{2} - \frac{q^{\left(\frac{2}{\alpha}\right)} . C^2(\alpha)}{2 . \sigma_a^2(\alpha)} \right]} \, \mathrm{d}q = \frac{1}{\pi} . \int_0^\infty cos \left[\left(\frac{K(\alpha) - 1}{\mathrm{e}} + 1 \right) . C(\alpha) \right] . \mathrm{e}^{-q^\alpha} \, \mathrm{d}q \tag{3.5}$$

onde:

$$K(\alpha) = \frac{\alpha . \Gamma\left(\frac{\alpha+1}{2.\alpha}\right)}{\Gamma\left(\frac{1}{\alpha}\right)} . \sqrt[\alpha]{\frac{\alpha . \Gamma\left(\frac{\alpha+1}{2}\right)}{\Gamma(1+\alpha) . sen\left(\frac{\pi . \alpha}{2}\right)}} \tag{3.6}$$

3.2.2 Cálculo de direção isotrópica

Um voo de Lévy é uma caminhada aleatória em que os comprimentos dos passos têm uma distribuição de probabilidade do tipo cauda pesada. Quando definida como uma

caminhada num espaço de dimensão maior que um, os passos são realizados em direções aleatórias isotrópicas.

Na implementação do algoritmo original no MATLAB, (YANG, 2010b) sugere a utilização da função randn com dimensão igual ao número de coordenadas do ninho conforme pode ser visto na Figura 13.

```
%% Get a cuckoo and generate new solutions by random walk
function s=get_a_cuckoo(s,star)
    % This is a random walk, which is less efficient
    % than Levy flights. In addition, the step size
    % should be a vector for problems with different scales.
    % Here is the simplified implementation for demo only!
    stepsize=0.05;
    s=star+stepsize*randn(size(s));
```

Figura 13 – Implementação da isotropia do algoritmo original no MATLAB (YANG, 2010b).

Da trigonomia, o versor $\hat{\mathbf{v}}$ de um vetor $\vec{\mathbf{v}}$ carrega a informação de direção conforme pode ser observado na Equação 3.7, enquanto $\|\vec{\mathbf{v}}\|$ é a magnitude desse vetor.

$$\vec{\mathbf{v}} = \|\vec{\mathbf{v}}\|.\hat{\mathbf{v}} \tag{3.7}$$

Portanto o produto interno da escalar $stepsize$ por um vetor gerado pela operação $randn$ tem o objetivo de criar um vetor com direção aleatória isotrópica e magnitude constante igual a 0.05 e que desempenha um passo da caminhada aleatória. No voo de Lévy, esse vetor tem sua magnitude calculada através da simulação numérica de processos estocásticos de Lévy.

Como nessa implementação os vetores euclidianos resultantes da operação $randn$ não apresentam módulo unitário, surge, como efeito colateral, um desvio na distribuição de probabilidade dos comprimentos dos passos através da adição de mais uma variável aleatória, o comprimento do vetor de direção gerado.

Para demonstrar esse efeito, 10000 vetores de direção bidimensionais foram gerados conforme implementação original (Figura 14) em que a amplitude deles foi calculada e o histograma levantado.

Como pode ser observado na Figura 15, a amplitude dos vetores gerados não é unitária. Objetivando-se corrigir essa distorção, esses vetores foram transformados em versores mantendo-se a direção isotrópica e aleatória.

Isso foi alcançado através da divisão dos vetores pelas suas respectivas magnitudes, conforme sugestão de (AINA; JONAS, 2012). A operação pode ser vista na Figura 16.

```
1 A = randn(10000,2);
2 B = A.^2;
3 C= sqrt(B(:,1)+B(:,2));
4
5 hist(C,100)
```

Figura 14 – Código para avaliação do comprimento de vetores bidimensionais gerados através da função randn. Fonte: O autor.

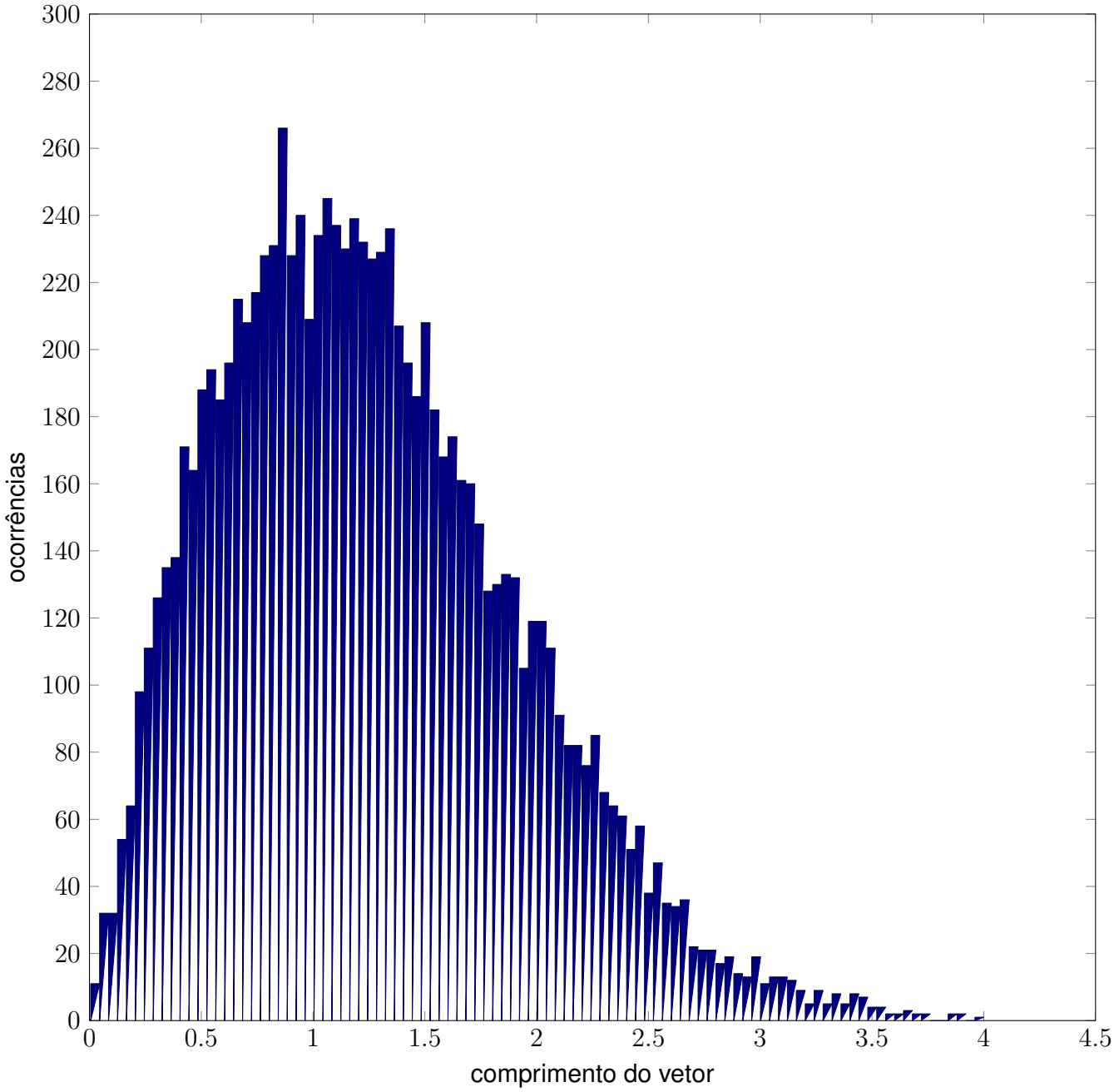

Figura 15 – Histograma do comprimento de 10000 vetores bidimensionais gerados através da função randn no MATLAB.

```
1 v = randn(10000,3);
2 v = bsxfun(@rdivide,v,sqrt(sum(v.^2,2)));
3
4 plot3(v(:,1),v(:,2),v(:,3),'.')
5 axis equal
```

Figura 16 – Código sugerido por Aina e Jonas para geração de vetor unitário de 3 dimensões.

Além de resolver o problema apontado de distorção, esse tipo de implementação ainda permite a geração de um versor n-dimensional, para isso basta alterar a dimensão 3 na linha 1 do código da Figura 16 para a dimensão desejada. O resultado dos vetores tridimensionais utilizando-se como ponto de partida a origem pode ser observado na Figura 17.

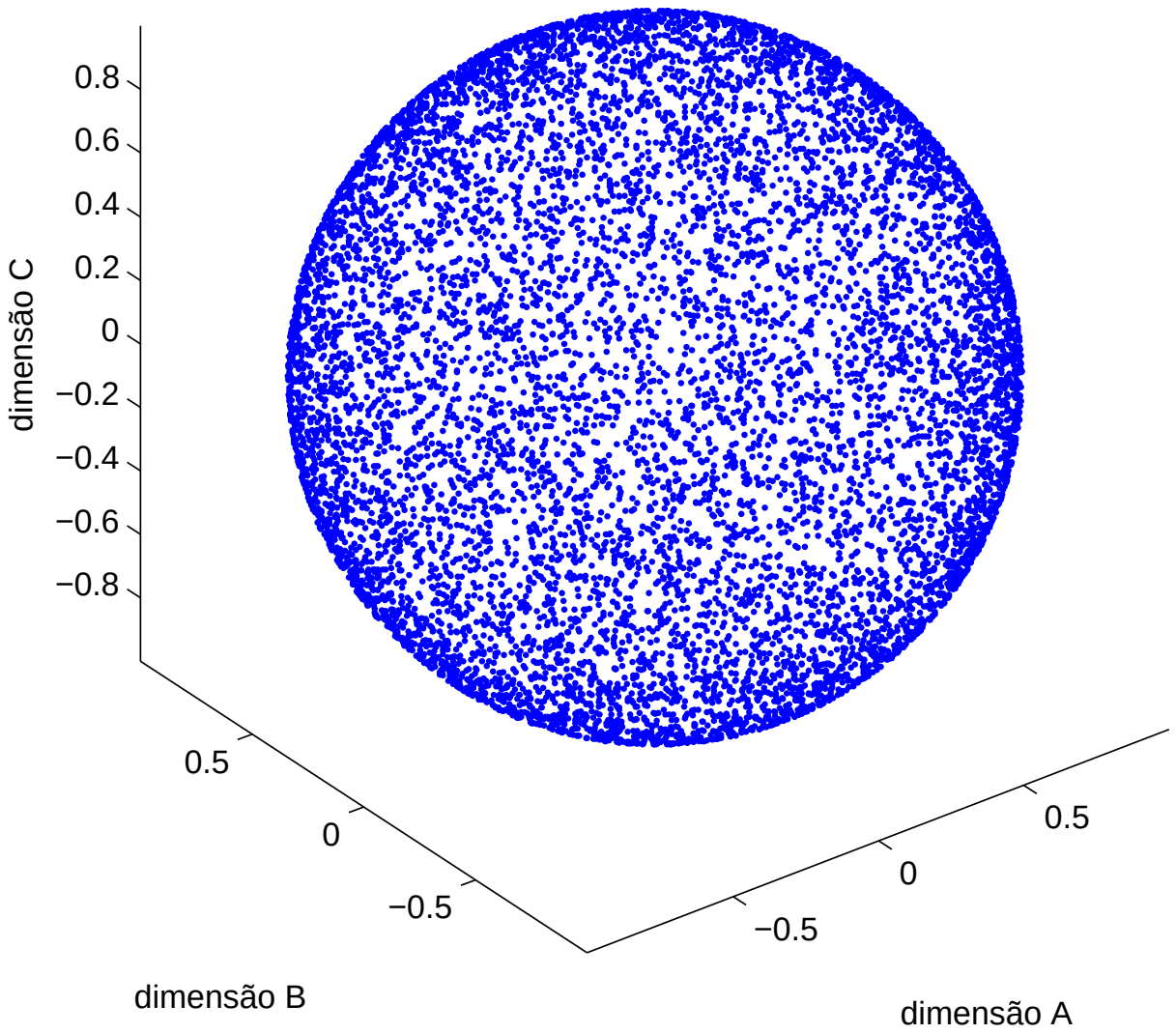

Figura 17 – Dez mil versores tridimensionais gerados isotrópica e aleatoriamente.

4 Resultados e Discussão

Para avaliar os resultados do estimador de parâmetros por Busca de Cuco via voos de Lévy(*Cuckoo Search via Lévy Flights*) (EPBC) proposto, um comparativo será feito com os resultados do estimador de parâmetros com NSGA-II (EPNSGAII) obtidos por (AGUIRRE; BARBOSA; BRAGA, 2010).

No artigo de (AGUIRRE; BARBOSA; BRAGA, 2010) um algoritmo evolucionário multi-objetivo (MOEA) chamado algoritmo genético de ordenação não dominado rápido(*Nondominated Sorting Genetic Algoritm*) II (NSGA-II) (DEB et al., 2002) foi aplicado com sucesso para a estimação de parâmetros de alguns exemplos já explorados na literatura.

Segundo (DEB et al., 2002) o NSGA-II lida bem com os principais pontos de crítica do NSGA, com melhores resultados se comparado com a estratégia de evolução pareto-arquivada (PAES) e Pareto-força EA (SPEA).

Dentre os exemplos abordados, dois foram selecionados para testar o desempenho do algoritmo aqui proposto. Eles possuem as seguintes características:

1. Sistema erro na saída (*output error*) polinomial (OEP) linear nos parâmetros

2. Sistema erro na saída (*output error*) racional (OER) não linear nos parâmetros

4.1 Parâmetros BC utilizados

A otimização busca de cuco por voos de Lévy foi configurada de acordo com a Tabela 1.

Tabela 1 – Parâmetros utilizados no cálculo da distribuição de Lévy.

Parâmetro	Valor
α	1.5
γ	1
n	1
N	matrix $\hat{\boldsymbol{\theta}}$

Conforme pode ser visto na Tabela 1, a dimensão da matriz de resultados será igual à dimensão do vetor de parâmetros a serem identificados $\hat{\boldsymbol{\theta}}$.

Como ambos os exemplos são SISO (única entrada única saída (*Single Input Single Output*)), a função de custo selecionada para ser minimizada foi o erro médio quadrático (*Mean Squared Error*) (MSE) aplicado ao erro de saída conforme descrito pela equação:

$$J(Z, Z_M) = MSE(y - \hat{y}) \tag{4.1}$$

Nestes exemplos o parâmetro StallGenLimit foi configurado relativamente alto e o parâmetro TolFun foi configurado bem baixo para evitar a desistência da evolução muito precocemente assim como um aumento da duração da simulação Monte Carlo.

Para completar a parametrização do algoritmo de cuco, a população inicial foi fixada em 15 elementos (PopulationSize), e a probabilidade de descoberta de ovos estranhos pela anfitriã (p_a) foi fixada em 0,25 (DiscoveryRate). Resumindo, o algoritmo BC foi configurado de acordo com a Tabela 2.

Tabela 2 – Conjunto de configurações da BC para estimação de parâmetros.

Parâmetro BC	Valor
StallGenLimit	1000
PopulationSize	15
TolFun	1e-20
DiscoveryRate	0,25

4.2 Método de Monte Carlo

O método de Monte Carlo será utilizado para avaliar estatisticamente os resultados obtidos com a aplicação do estimador de parâmetros por Busca de Cuco via voos de Lévy(*Cuckoo Search via Lévy Flights*) (EPBC) à ambos os exemplos. Métodos de Monte Carlo geralmente seguem este padrão:

1. Definir um domínio de possíveis entradas
2. Gerar entradas aleatoriamente com uma distribuição de probabilidade sobre o domínio.
3. Fazer uma computação determinística das entradas.
4. Agregar os resultados.

4.3 Construção dos modelos dos exemplos

Os sistemas S dos exemplos, aqui também explorados, foram construídos no Simulink objetivando-se gerar o conjunto de dados Z que inclui tanto os sinais de entrada e saída do sistema S.

Seguindo-se as recomendações do MATLAB (MATLAB, 2013) para melhores resultados, a semente inicial dos blocos geradores de ruído gaussiano foi configurada para a função *randseed*, que gera um número primo aleatório e maior que 30.

4.3.1 Exemplo OE polinomial

O primeiro exemplo aqui abordado trata da identificação de parâmetros de um modelo para um sistema erro na saída (*output error*) polinomial (OEP) proposto por (PIRODDI; SPINELLI, 2003):

$$\begin{cases} w(k) &= 0,75w(k-2) + 0,25u(k-1) \\ & \quad -0,2w(k-2)u(k-1) \\ y(k) &= w(k) + e(k) \end{cases} \tag{4.2}$$

onde a entrada $u(k)$ é um processo auto regressivo de segunda ordem com pólos em $z = 0,9$ e $0,95$ excitado por um ruído branco gaussiano com media zero e variância $\sigma^2 = 0,25$.

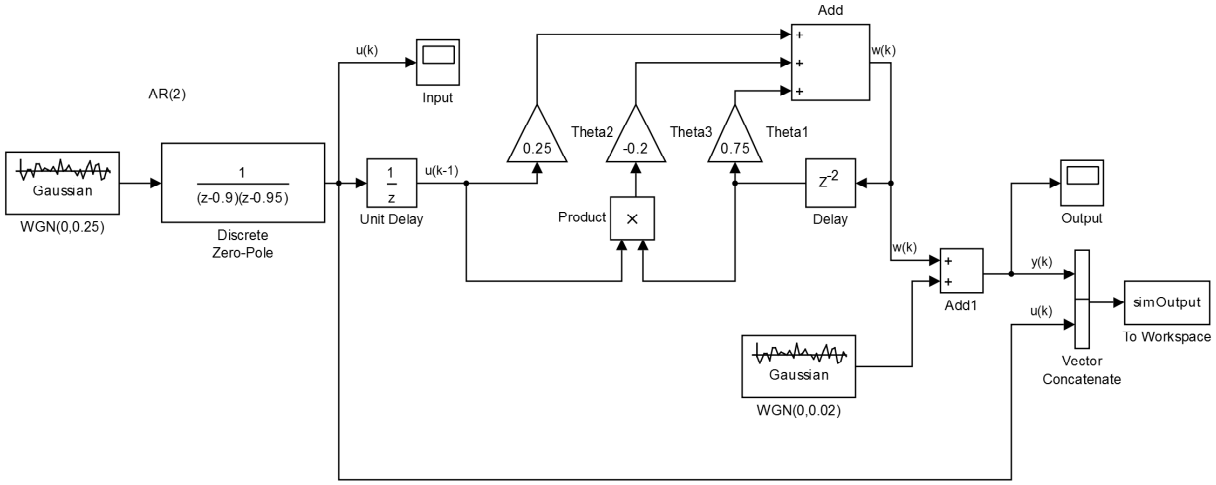

Figura 18 – Sistema erro na saída (*output error*) polinomial (OEP).

Re-escrevendo na representação NARMAX:

$$\begin{aligned} y(k) =& 0,75y(k-2) + 0,25u(k-1) \\ & - 0,2y(k-2)u(k-1) - 0,75e(k-2) \\ & + 0,2e(k-2)u(k-1) + e(k) \end{aligned} \tag{4.3}$$

Na simulação livre, frequentemente esse sistema apresenta instabilidade na saída conforme pode ser observado na Figura 19. Quanto maior a quantidade de amostras utilizadas nos sinais de entrada e saída, maior a probabilidade de o sistema entrar em

oscilação. Esse comportamento traz uma dificuldade adicional na estimação devido à possibilidade de gerar inúmeros pontos de mínimo na função de custo.

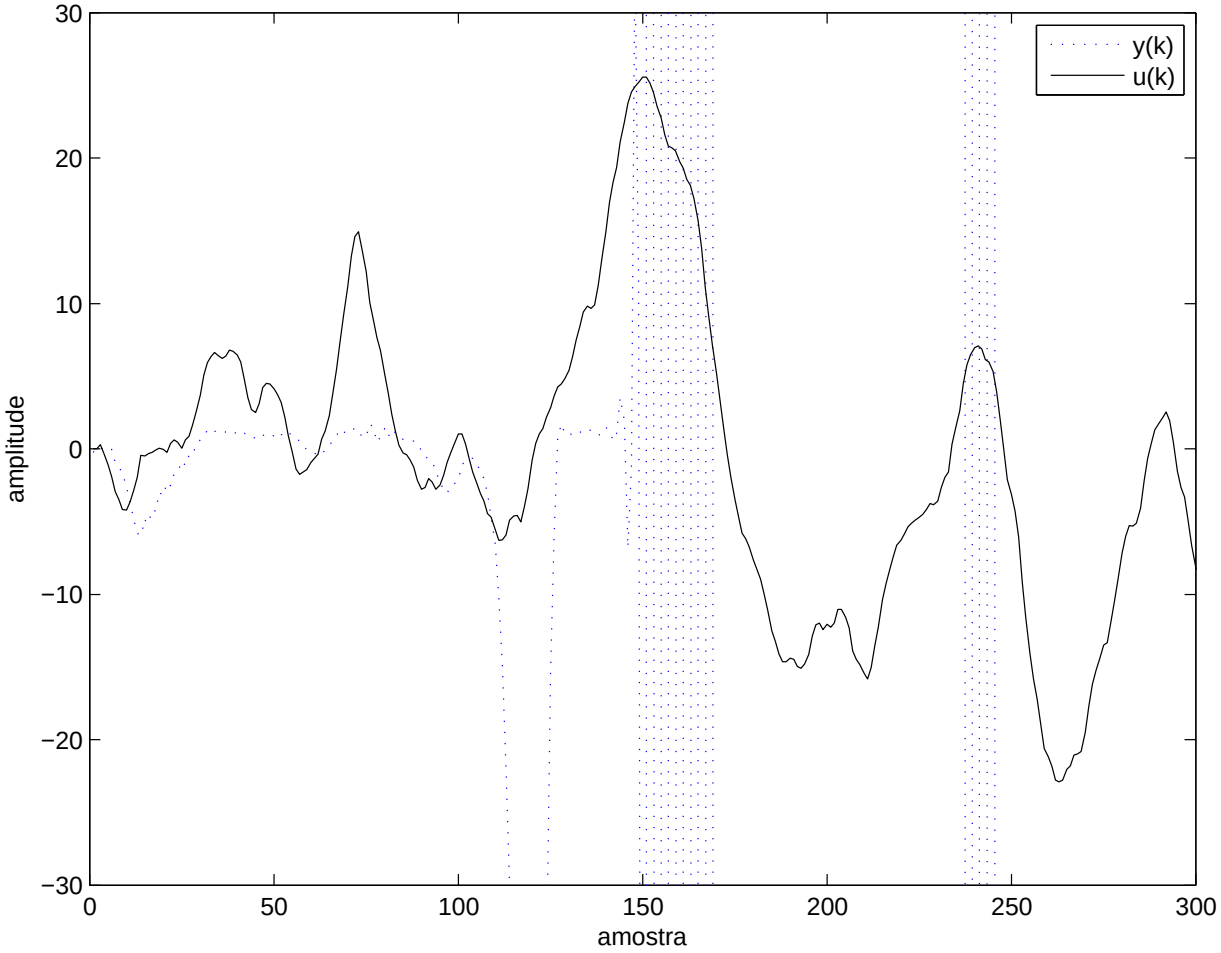

Figura 19 – Instabilidade da saída numa simulação livre do sistema erro na saída (*output error*) polinomial. Valores máximo e mínimo de $y(k)$ são $6,9537 \times 10^{23}$ e $-1,9038 \times 10^{17}$ respectivamente.

Os resultados obtidos das 1000 rodadas Monte Carlo mostram que utilizando-se a otimização Busca de Cuco via voos de Lévy(*Cuckoo Search via Lévy Flights*) (BC) para estimação de parâmetros, foi possível estimar os parâmetros do modelo conforme mostra a Tabela 3 na coluna $\bar{\boldsymbol{\theta}}$.

Tabela 3 – Resultados do modelo OEP com EPBC sobre 1000 rodadas Monte Carlo. 300 amostras foram usadas na rodada de simulação livre do sistema S no Simulink.

$\boldsymbol{\theta_n}$	ref	min $\boldsymbol{\theta_m}$	max $\boldsymbol{\theta_M}$	med $\bar{\boldsymbol{\theta}}$	desvio $\boldsymbol{\sigma_\theta}$
θ_1	0,75	0,7497	0,7500	0,7500	0,0000
θ_2	0,25	0,0316	0,3408	0,2336	0,0402
θ_3	-0,2	-0,2001	-0,2000	-0,2000	0,0000

Se os resultados forem comparados agora com aqueles obtidos por (AGUIRRE; BARBOSA; BRAGA, 2010) na Tabela 4, é também possível ver que para 300 amostras, os desvios padrão $\boldsymbol{\sigma_\theta}$ do estimador de parâmetros por Busca de Cuco via voos de Lévy(*Cuckoo Search via Lévy Flights*) para os parâmetros θ_1 e θ_3 são iguais a zero, enquanto o estimador com NSGA-II obteve 0,005. Já o parâmetro θ_2 foi melhor estimado pelo estimador de parâmetros com NSGA-II.

Tabela 4 – Resultados do modelo OEP com EPNSGAII sobre 1000 rodadas Monte Carlo. 1000 amostras foram usadas na rodada de simulação livre do sistema S no Simulink.

$\boldsymbol{\theta_n}$	ref	min $\boldsymbol{\theta_m}$	max $\boldsymbol{\theta_M}$	med $\boldsymbol{\bar{\theta}}$	desvio $\boldsymbol{\sigma_\theta}$
θ_1	0,75	0,7350	0,7690	0,7500	0,0050
θ_2	0,25	0,2330	0,2650	0,2500	0,0050
θ_3	-0,2	-0,2210	-0,1790	-0,2010	0,0050

4.3.2 Sistema erro na saída (*output error*) racional (OER)

O seguinte sistema erro na saída (*output error*) racional (OER) é não linear nos parâmetros e foi tirado de (ZHU, 2005).

$$\begin{cases} w(k) &= \frac{0{,}3w(k-1)w(k-2)+0{,}7u(k-1)}{1+w(k-1)^2+u(k-1)^2} \\ y(k) &= w(k)+e(k) \end{cases} \tag{4.4}$$

onde a entrada $u(k)$ é uma sequência uniformemente distribuída com media zero e variância σ^2 igual a $0,33$ e $e(k)$ é um ruído branco com média zero e variância σ^2 igual a $0,01$. Esse problema foi escolhido porque de acordo com o (AGUIRRE; BARBOSA; BRAGA, 2010) ele representa uma estimação de parâmetros desafiadora.

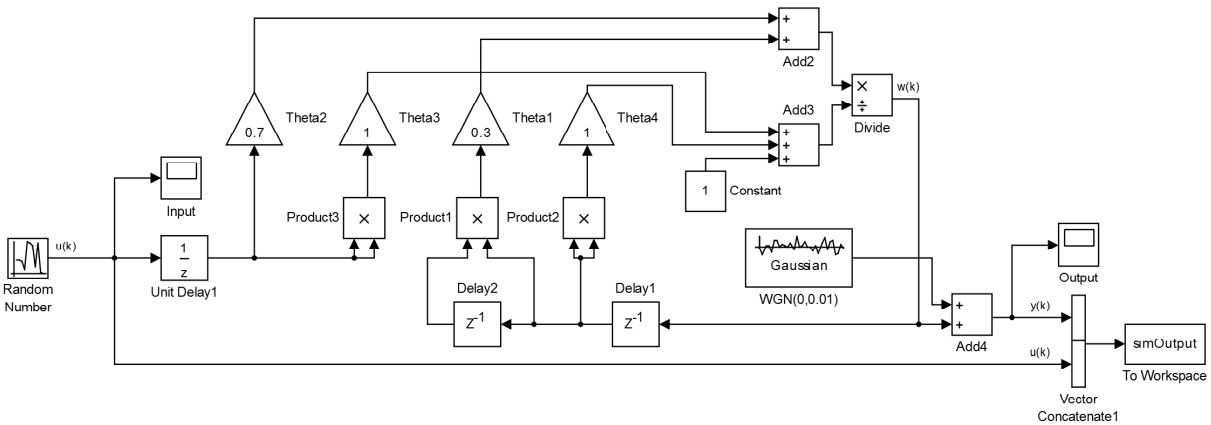

Figura 20 – Sistema erro na saída (*output error*) racional (OER).

Ele pode ser re-escrito usando a representação erro na equação (*equation error*) (EE):

$$y(k) = \frac{0,3y(k-1)y(k-2) + 0,7u(k-1)}{1 + y(k-1)^2 + u(k-1)^2} + e(k) \tag{4.5}$$

Os resultados obtidos das 1000 rodadas Monte Carlo mostram que utilizando-se a otimização BC para estimação de parâmetros, foi possível estimar os parâmetros do modelo conforme mostra a Tabela 5 na coluna $\bar{\theta}$.

Tabela 5 – Resultados do modelo OE racional com EPBC sobre 1000 rodadas Monte Carlo. 1000 amostras foram usadas na rodada de simulação livre do sistema S no Simulink.

$\boldsymbol{\theta_n}$	ref	min $\boldsymbol{\theta_m}$	max $\boldsymbol{\theta_M}$	med $\bar{\boldsymbol{\theta}}$	desvio $\boldsymbol{\sigma_\theta}$
θ_1	0,3	0,0874	0,5552	0,3008	0,0745
θ_2	0,7	0,6182	0,7882	0,6981	0,0266
θ_3	1	-0,4115	2,5834	0,9693	0,5121
θ_4	1	0,7476	1,2415	0,9971	0,0837

Se os resultados forem comparados agora com aqueles obtidos por (AGUIRRE; BARBOSA; BRAGA, 2010) na Tabela 6, é também possível ver que para 1000 amostras, os desvio padrão σ_θ do estimador de parâmetros BC foi aproximadamente metade do estimador de parâmetros com NSGA-II. Isso significa que ele foi mais eficiente na indicação dos valores de parâmetros reais.

Tabela 6 – Resultados do modelo racional OE com EPNSGAII sobre 1000 rodadas Monte Carlo. 1000 amostras foram usadas na rodada de simulação livre do sistema S no Simulink.

$\boldsymbol{\theta_n}$	ref	min $\boldsymbol{\theta_m}$	max $\boldsymbol{\theta_M}$	med $\bar{\boldsymbol{\theta}}$	desvio $\boldsymbol{\sigma_\theta}$
θ_1	0,3	-0,25	0,834	0,293	0,142
θ_2	0,7	0,454	0,905	0,703	0,062
θ_3	1	-2,659	4,175	1,044	0,939
θ_4	1	0,452	1,773	1,016	0,225

4.4 Influência da quantidade de amostras nos resultados

Para efeito de comparação com sistemas reais e tomando-se por exemplo uma taxa de amostragem constante, aumentar a quantidade de amostras implica um aumento do tempo de simulação.

Em geral para sistemas reais, quanto maior o tempo de aquisição de dados, maior a quantidade de informação que pode ser extraída dos experimentos, o que leva a um entendimento maior das relações que regem o sistema em questão.

Por isso, a quantidade de amostras dos sinais de entrada e saída desses sistemas a serem identificados tem uma influência importante nos resultados principalmente em sistemas ruidosos, em que uma quantidade maior de amostras possibilita um entendimento melhor das propriedades estatísticas do ruído e de seu efeito nos sistemas.

Por esse motivo os exemplos de identificação de parâmetros abordados foram também explorados em relação à quantidade de amostras dos sinais. Para efeito de comparação o algoritmos genéticos (AG) e o Busca de Cuco via voos de Lévy(*Cuckoo Search via Lévy Flights*) (BC) foram utilizados na identificação dos parâmetros dos exemplos previamente abordados.

4.4.1 Avaliação quantidade de amostras exemplo OE polinomial

A varredura do exemplo sistema erro na saída (*output error*) polinomial (OEP) foi feita de 20 a 320 amostras em progressão geométrica com razão q=2.

É possível observar na Figura 21 uma curva de histograma de estimação de parâmetros do numerador da Equação 4.2 com o estimador de parâmetros por Busca de Cuco via voos de Lévy(*Cuckoo Search via Lévy Flights*) (EPBC) para cada quantidade de amostras simulada.

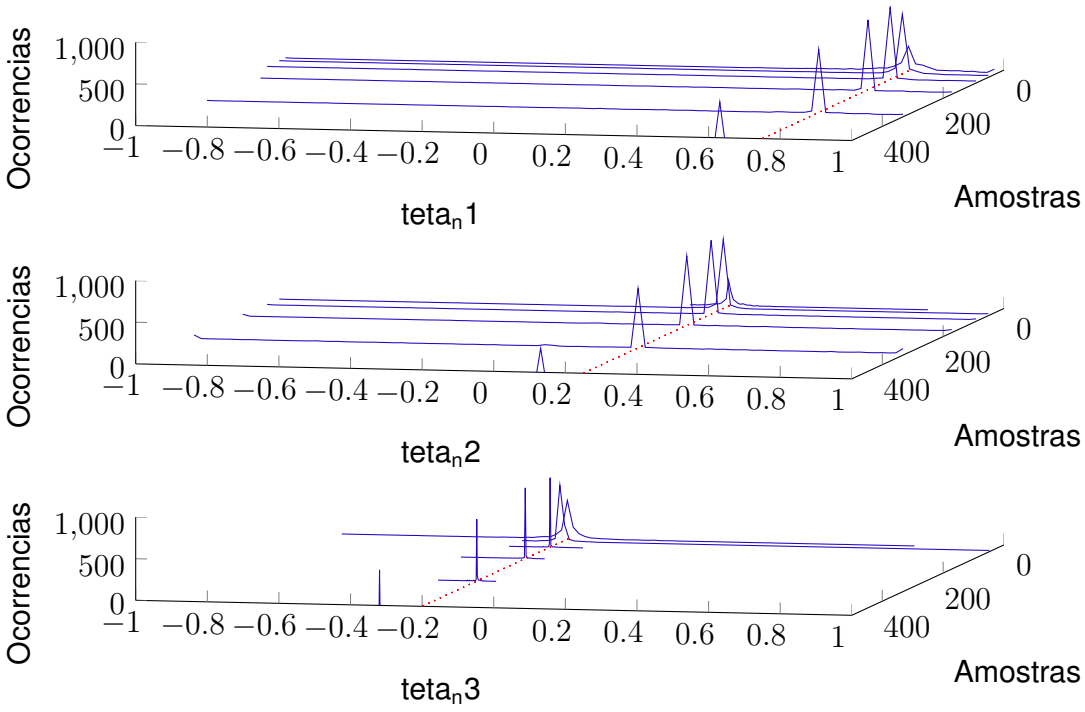

Figura 21 – Histogramas dos 3 parâmetros identificados por BC em 1000 rodadas Monte Carlo para sinais de comprimento 20, 40, 80, 160, 320 amostras.

A linha pontilhada denota o valor de referência $\boldsymbol{\theta}_i$ para a identificação de cada parâmetro $\hat{\boldsymbol{\theta}}_i$ com $\{i \in \mathbb{I}/1 \leq i \leq 3\}$.

À medida que a quantidade de amostras aumenta, a acurácia da estimação também aumenta, conforme pode ser visto mais nitidamente na família de histogramas do parâmetro $\hat{\boldsymbol{\theta}}_3$.

Os resultados da Figura 21 resultantes da estimação por BC agora podem ser comparados com os resultados da Figura 22 advindos da estimação por AG.

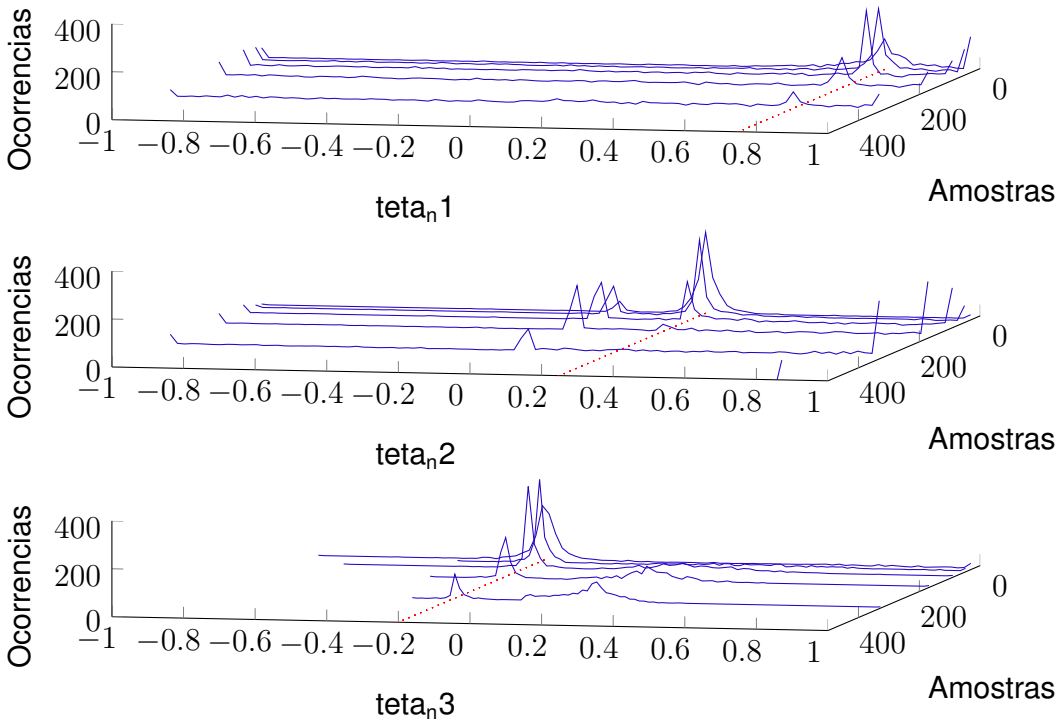

Figura 22 – Histogramas dos 3 parâmetros identificados por AG em 1000 rodadas Monte Carlo para sinais de comprimento 20, 40, 80, 160, 320 amostras. Observe a ocorrência de mínimos locais.

É possível observar na Figura 22 que os resultados de estimação por AG foram menos acurados porque houve indicações de mínimos locais e que estão distantes dos valores de referência $\boldsymbol{\theta}_{ni}$. Esses mínimos locais não foram observados nos resultados de estimação por BC para esse exemplo.

4.4.2 Avaliação quantidade de amostras exemplo OE racional

A varredura do exemplo sistema erro na saída (*output error*) racional (OER) foi feita de 20 a 2560 amostras em progressão geométrica com razão q=2.

É possível observar nas Figuras 23 e 24 uma curva de histograma de estimação de parâmetros do numerador $\hat{\boldsymbol{\theta}}_{ni}$ e denominador $\hat{\boldsymbol{\theta}}_{di}$ da Equação 4.4 com o estimador de parâmetros por Busca de Cuco via voos de Lévy(*Cuckoo Search via Lévy Flights*) (EPBC) para cada quantidade de amostras simulada.

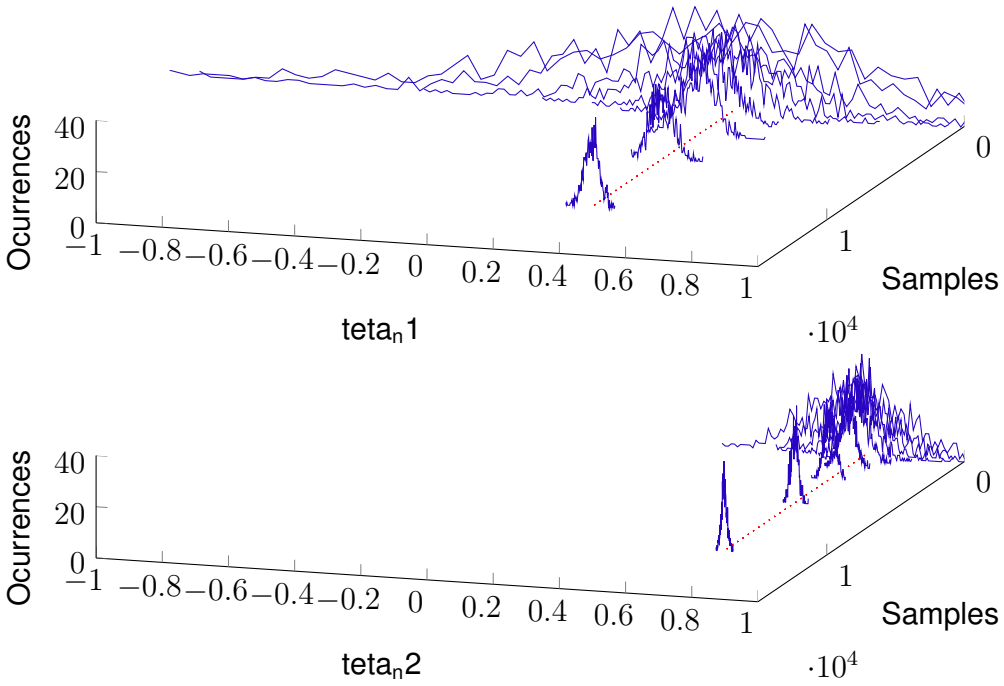

Figura 23 – Histogramas dos 2 parâmetros identificados do numerador por BC em 1000 rodadas Monte Carlo para sinais de comprimento 20, 40, 80, 160, 320, 640, 1280, 2560, 5120, 10240 amostras.

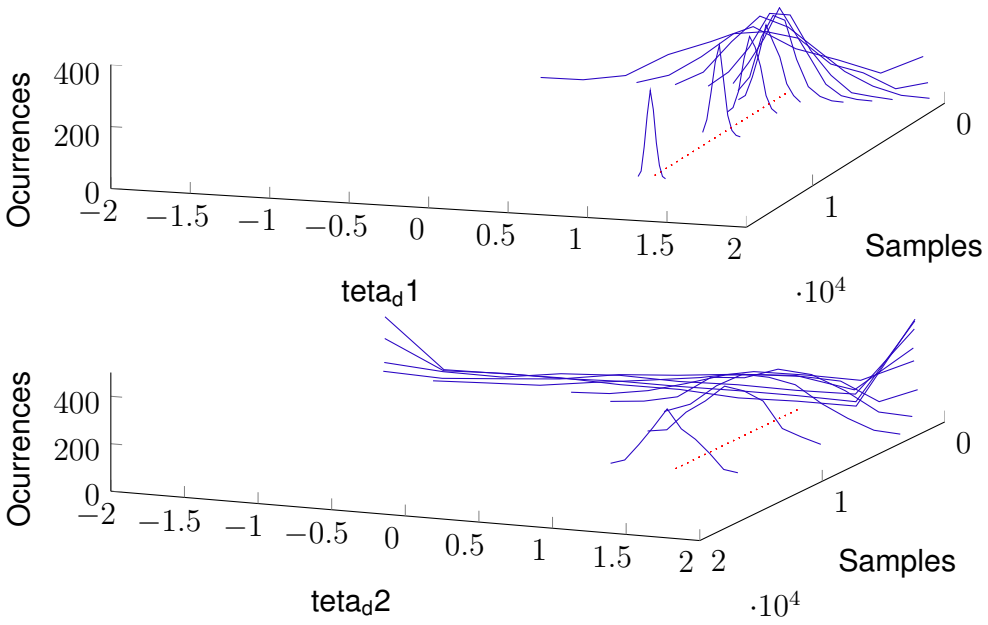

Figura 24 – Histogramas dos 2 parâmetros identificados do denominador por BC em 1000 rodadas Monte Carlo para sinais de comprimento 20, 40, 80, 160, 320, 640, 1280, 2560, 5120, 10240 amostras.

À medida que a quantidade de amostras aumenta, a acurácia da estimação também aumenta, conforme pode ser visto nitidamente em todas as famílias de histogramas para cada parâmetro $\hat{\boldsymbol{\theta}}_{ni}$ ou $\hat{\boldsymbol{\theta}}_{di}$.

Os resultados das Figuras 23 e 24 resultantes da estimação por BC agora podem ser comparados com os resultados das Figuras 25 e 26 advindos da estimação por AG.

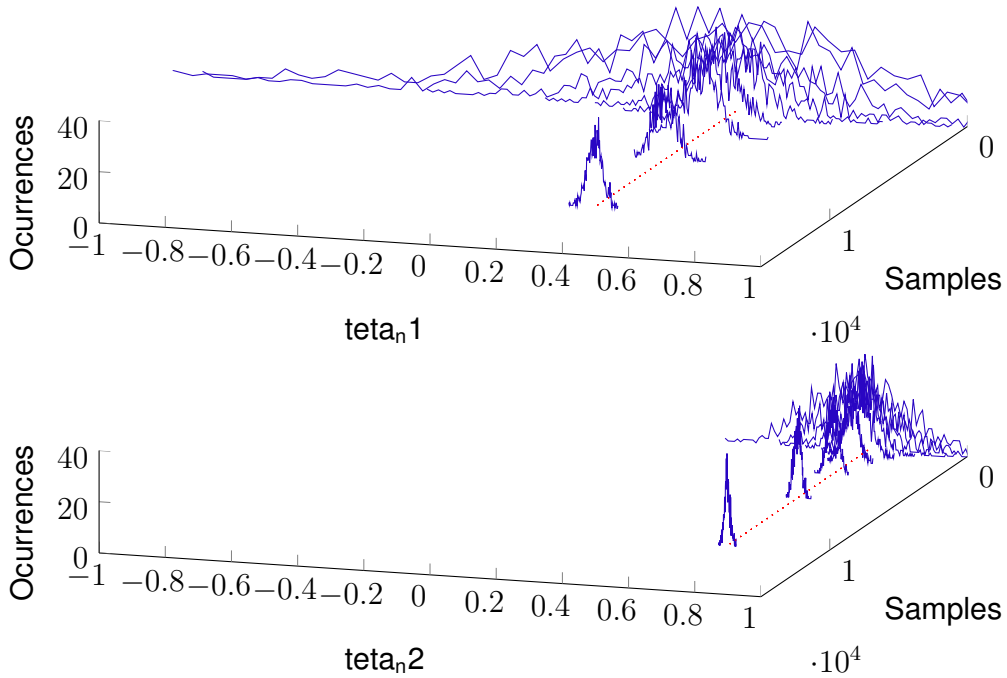

Figura 25 – Histogramas dos 2 parâmetros identificados do numerador por AG em 1000 rodadas Monte Carlo para sinais de comprimento 20, 40, 80, 160, 320, 640, 1280, 2560, 5120, 10240 amostras.

É possível observar no comparativo que mudando-se o método de estimação de parâmetros para esse exemplo, não houve alteração perceptiva nos histogramas resultantes de estimação.

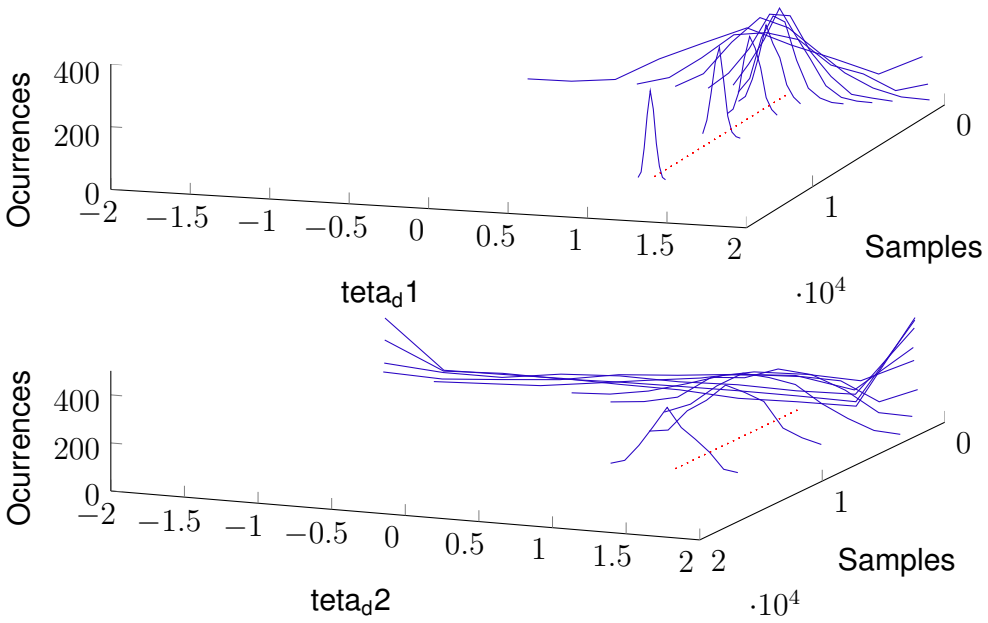

Figura 26 – Histogramas dos 2 parâmetros identificados do denominador por AG em 1000 rodadas Monte Carlo para sinais de comprimento 20, 40, 80, 160, 320, 640, 1280, 2560, 5120, 10240 amostras.

5 Conclusão

Nesse trabalho propôs-se um algoritmo de estimação que aplica a busca de cuco por voos de Lévy na estimação de parâmetros dada a estrutura do sistema. O algoritmo foi acurado, rápido, fácil de implementar e ajustar.

O exemplo prático da função de Langermann foi colocado de forma a evidenciar as características do algoritmo BC em comparativo com o algoritmo AG utilizando-se as ferramentas do MATLAB para fornecer um rápido aprendizado da técnica de otimização BC.

Como o foco dado nesse trabalho é a atividade da estimação de parâmetros dentro da identificação de sistemas, e como o algoritmo BC selecionado para a otimização dos parâmetros é evolucionário, um critério de parada apropriado foi selecionado para interromper a otimização em momento oportuno, evitando-se assim um tempo de processamento muito além do necessário sem comprometer a qualidade do resultado.

Duas oportunidades de melhorias foram exploradas com o objetivo de fazer o cálculo da distribuição de Lévy mais fiel à matemática observada nas buscas espaciais que alguns animais desenvolvem na natureza. Como o algoritmo BC faz a utilização de um cálculo simplificado do método de Mantegna na obtenção da distribuição de Lévy, e o acréscimo de tempo de processamento necessário para a utilização do método integral não é relevante, o último foi utilizado para a obtenção dos resultados dos exemplos. Foi observado também que a forma com que a direção isotrópica é realizada no algoritmo BC distorce a distribuição de Lévy com a amplitude de vetores não unitários responsáveis por aplicar a isotropia à direção. Esse problema foi eliminado com a utilização de versores para essa funcionalidade.

Para destacar o seu desempenho os sistemas dos exemplos escolhidos foram não lineares, sendo o primeiro de saída instável e o segundo não linear também nos parâmetros o que representa uma complexidade adicional para o processo de estimação.

Os resultados demonstram que o algoritmo aqui proposto além de obter sucesso na tarefa da estimação de parâmetros, foi em geral mais acurado no comparativo com o NSGA-II.

Variando-se a quantidade de amostras de simulação dos sistemas foi possível observar que quanto mais amostras utilizadas, melhor a estimação dos parâmetros se torna devido ao fato de as propriedades estatísticas do ruído poderem ser melhor estimadas. Em um dos exemplos, nessa avaliação, foi possível observar nos histogramas dos parâmetros estimados a indicação de mínimos locais pelo algoritmo AG, o que não ocorreu com a utilização do BC.

Trabalhos futuros incluem a estimação da estrutura do modelo através do algoritmo

BC, o estudo da utilização de distribuições α-estáveis para a realização das buscas espaciais ao invés de puramente voos de Lévy, a utilização de parâmetros de busca espacial variáveis de forma a atingir uma rápida convergência à vizinhança do mínimo global sem comprometer sua qualidade de entrega ao final da otimização e a avaliação das condições de fronteira nas otimizações visto que as distribuições de cauda pesada podem gerar comprimentos de passo de grande magnitude extrapolando as fronteiras definidas.

Referências

ADORIO, E. P.; DILIMAN, U. P. *MVF - Multivariate Test Functions Library in C for Unconstrained Global Optimization*. 2005. Disponível em: <http://http://www.geocities.ws/eadorio/mvf.pdf>. Acesso em: Junho de 2013. Citado na página 49.

AGUIRRE, L. A.; BARBOSA, B. H.; BRAGA, A. P. Prediction and simulation errors in parameter estimation for nonlinear systems. *Mechanical Systems and Signal Processing*, v. 24, n. 8, p. 2855 – 2867, 2010. ISSN 0888-3270. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0888327010001469>. Acesso em: 21 de dezembro de 2012. Citado 5 vezes nas páginas 28, 31, 61, 65 e 66.

AINA; JONAS. *How to generate a unit vector pointing in a random direction with isotropic distribution of direction?* 2012. Disponível em: <http://stackoverflow.com/questions/9750908/how-to-generate-a-unit-vector-pointing-in-a-random-direction-with-isotropic-dist>. Acesso em: 31 de dezembro de 2013. Citado 3 vezes nas páginas 15, 58 e 60.

AUSIELLO, G. *Complexity and approximation : combinatorial optimization problems and their approximability properties*. New York: Springer, 1999. ISBN 9783540654315. Citado na página 36.

BAI, E.-W. Non-parametric nonlinear system identification: An asymptotic minimum mean squared error estimator. *Automatic Control, IEEE Transactions on*, v. 55, n. 7, p. 1615–1626, July 2010. ISSN 0018-9286. Citado na página 26.

BAI, E.-W.; DEISTLER, M. An interactive term approach to non-parametric fir nonlinear system identification. *Automatic Control, IEEE Transactions on*, v. 55, n. 8, p. 1952–1957, Aug 2010. ISSN 0018-9286. Citado na página 26.

BEIGI, H. *Fundamentals of Speaker Recognition*. Springer, 2011. (SpringerLink : Bücher). ISBN 9780387775920. Disponível em: <http://books.google.se/books?id=qIMDvu3gJCQC>. Acesso em: 11 de fevereiro de 2014. Citado na página 43.

BILLINGS, S. Identification of nonlinear systems: A survey. *Control Theory and Applications, IEE Proceedings D*, v. 127, n. 6, p. 272–285, November 1980. ISSN 0143-7054. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=04642019>. Acesso em: 11 de fevereiro de 2014. Citado na página 26.

BILLINGS, S. A. *Nonlinear system identification : NARMAX methods in the time, frequency, and spatio-temporal domains*. Chichester, West Sussex: Wiley, 2013. ISBN 978-1-119-94359-4. Citado 4 vezes nas páginas 26, 27, 32 e 34.

BOYD, S. *Convex optimization*. Cambridge, UK New York: Cambridge University Press, 2004. ISBN 9780521833783. Citado na página 36.

CHAMBERS, J. M.; MALLOWS, C. L.; STUCK, B. W. A method for simulating stable random variables. *Journal of the American Statistical Association*, American Statistical Association, v. 71, n. 354, p. pp. 340–344, 1976. ISSN 01621459. Disponível em: <http://www.jstor.org/stable/2285309>. Acesso em: 11 de fevereiro de 2014. Citado 2 vezes nas páginas 46 e 57.

CHEN, Q. et al. Genetic algorithm with an improved fitness function for (n)arx modelling. *Mechanical Systems and Signal Processing*, v. 21, n. 2, p. 994 – 1007, 2007. ISSN 0888-3270. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0888327006000343>. Acesso em: 11 de fevereiro de 2014. Citado na página 27.

COELHO, L. dos S.; PESSôA, M. W. Nonlinear model identification of an experimental ball-and-tube system using a genetic programming approach. *Mechanical Systems and Signal Processing*, v. 23, n. 5, p. 1434 – 1446, 2009. ISSN 0888-3270. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0888327009000600>. Acesso em: 11 de fevereiro de 2014. Citado na página 27.

DEB, K. et al. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *Evolutionary Computation, IEEE Transactions on*, v. 6, n. 2, p. 182–197, 2002. ISSN 1089-778X. Disponível em: <http://ieeexplore.ieee.org/xpl/abstractReferences.jsp?arnumber=996017>. Acesso em: 31 de dezembro de 2013. Citado na página 61.

ERITEN, M. et al. Nonlinear system identification of frictional effects in a beam with a bolted joint connection. *Mechanical Systems and Signal Processing*, v. 39, n. 12, p. 245 – 264, 2013. ISSN 0888-3270. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0888327013001039>. Citado na página 26.

EYKHOFF, P. *System identification : parameter and state estimation*. London New York: Wiley-Interscience, 1974. ISBN 9780471249801. Citado na página 28.

FORSSELL, U. et al. Combining semi-physical and neural network modeling: An example of its usefulness. In: *the 11th IFAC Symposium on System Identification (SYSID'97*. [S.l.: s.n.], 1997. p. 795–798. Citado na página 28.

GOLDBERG, D. E. *Genetic Algorithms in Search, Optimization and Machine Learning*. 1st. ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989. ISBN 0201157675. Citado na página 48.

GOODWIN, G. *Dynamic system identification experiment design and data analysis*. New York: Academic Press, 1977. ISBN 9780080956459. Citado na página 27.

GRAUPE, D. *Identification of systems*. Huntington, N.Y: R.E. Krieger Pub. Co, 1976. ISBN 9780882753591. Citado na página 28.

GREFENSTETTE, J. J.; BAKER, J. E. How genetic algorithms work: A critical look at implicit parallelism. In: *Proceedings of the Third International Conference on Genetic Algorithms*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1989. p. 20–27. ISBN 1-55860-006-3. Disponível em: <http://dl.acm.org/citation.cfm?id=93126.93136>. Citado na página 48.

HOLDER, A. (Ed.). *Mathematical Programming Glossary*. <http://glossary.computing.society.informs.org>: INFORMS Computing Society, 2006–14. Originally authored by Harvey J. Greenberg, 1999-2006. Citado na página 35.

HUMPHRIES, N. E. et al. Environmental context explains lévy and brownian movement patterns of marine predators. *Nature*, Nature Publishing Group, a division of Macmillan Publishers Limited. All Rights Reserved., v. 465, p. 1066 – 1069, june 2010. ISSN 0028-0836. Disponível em: <http://www.nature.com/nature/journal/v465/n7301/full/nature09116.html>. Acesso em: 3 de junho de 2014. Citado na página 40.

KALMAN, R. E. Control theory. In: *Encyclopædia Britannica Online*. Encyclopædia Britannica, 2013. Disponível em: <http://www.britannica.com/EBchecked/topic/135499/control-theory/235464/Biocontrol>. Acesso em: 15 de dezembro de 2013. Citado na página 26.

KUMAR, R.; MOORE, J. B. Detection techniques in least squares identification. *Automatica*, v. 17, n. 6, p. 805 – 819, 1981. ISSN 0005-1098. Disponível em: <http://www.sciencedirect.com/science/article/pii/0005109881900686>. Acesso em: 25 de fevereiro de 2012. Citado na página 27.

LECCARDI, M. Comparison of three algorithms for lévy noise generation. In: *Fifth EUROMECH Nonlinear Dynamics Conference*. [s.n.], 2005. p. 1–6. Disponível em: <http://www.mfn.unipmn.it/~scalas/report1_eng.pdf>. Acesso em: 11 de fevereiro de 2014. Citado na página 57.

LJUNG, L. *Identification of Nonlinear Systems*. Department of Electrical Engineering, Automatic Control, 2007. Disponível em: <http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-56050>. Acesso em: 11 de fevereiro de 2014. Citado na página 26.

LJUNG, L. Perspectives on system identification. *Annual Reviews in Control*, v. 34, n. 1, p. 1 – 12, 2010. ISSN 1367-5788. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1367578810000027>. Acesso em: 3 de junho de 2014. Citado na página 26.

MANDELBROT, B. *The Fractal Geometry of Nature*. San Francisco: Henry Holt and Company, 1983. ISBN 9780716711865. Disponível em: <http://books.google.se/books?id=ijhWngEACAAJ>. Acesso em: 27 de dezembro de 2013. Citado 2 vezes nas páginas 40 e 43.

MANTEGNA, R. N. Fast, accurate algorithm for numerical simulation of lévy stable stochastic processes. *Phys. Rev. E*, American Physical Society, v. 49, p. 4677–4683, May 1994. Disponível em: <http://link.aps.org/doi/10.1103/PhysRevE.49.4677>. Acesso em: 30 de janeiro de 2013. Citado 2 vezes nas páginas 56 e 57.

MATLAB. *Gaussian Noise Generator, Matlab Online Documentation*. Natick, Massachusetts: The MathWorks Inc., 2013. Disponível em: <http://www.mathworks.com/help/comm/ref/gaussiannoisegenerator.html>. Acesso em: 3 de junho de 2013. Citado na página 63.

MOLGA, M.; SMUTNICKI, C. *Test functions for optimization needs*. 2005. Disponível em: <http://www.zsd.ict.pwr.wroc.pl/files/docs/functions.pdf>. Acesso em: 3 de junho de 2013. Citado na página 49.

MOORE, J. B. On strong consistency of least squares identification algorithms. *Automatica*, v. 14, n. 5, p. 505 – 509, 1978. ISSN 0005-1098. Disponível em: <http://www.sciencedirect.com/science/article/pii/0005109878900109>. Citado na página 27.

NELLES, O. *Nonlinear system identification : from classical approaches to neural networks and fuzzy models*. Berlin New York: Springer, 2001. ISBN 3540673695. Citado na página 26.

NIELSEN, H. et al. *Predicting the Heat Consumption in District Heating Systems Using Meteorological Forecasts*. IMM, 2000. (Contract ENS-1323/98-0025). Disponível em: <http://books.google.com.br/books?id=VqBEQwAACAAJ>. Citado na página 28.

PATCHARAPRAKITI, N. et al. Modeling of single phase inverter of photovoltaic system using hammersteinwiener nonlinear system identification. *Current Applied Physics*, v. 10, n. 3, Supplement, p. S532 – S536, 2010. ISSN 1567-1739. 19th International Photovoltaic Science and Engineering Conference and Exhibition (PVSEC-19) 19th International Photovoltaic Science and Engineering Conference and Exhibition (PVSEC-19). Disponível em: <http://www.sciencedirect.com/science/article/pii/S156717391000043X>. Citado na página 26.

PIRODDI, L.; SPINELLI, W. An identification algorithm for polynomial narx models based on simulation error minimization. *International Journal of Control*, v. 76, n. 17, p. 1767–1781, 2003. Disponível em: <http://www.tandfonline.com/doi/abs/10.1080/00207170310001635419>. Acesso em: 27 de dezembro de 2013. Citado 2 vezes nas páginas 28 e 63.

RAWLINS, G. *Foundations of genetic algorithms. [1](1991)*. Elsevier Science & Tech, 1991. (Foundations of genetic algorithms). ISBN 9781558601703. Disponível em: <http://books.google.com.br/books?id=Df12yLrlUZYC>. Citado na página 48.

SAMAD, T.; MATHUR, A. Parameter estimation for process control with neural networks. *International Journal of Approximate Reasoning*, v. 7, n. 3-4, p. 149–164, 1992. ISSN 0888-613X. Disponível em: <http://www.sciencedirect.com/science/article/pii/0888613X9290008N>. Acesso em: 22 de janeiro de 2014. Citado na página 27.

SAMORADNITSKY, G.; TAQQU, S. *Stable Non-Gaussian Random Processes: Stochastic Models with Infinite Variance*. Taylor & Francis, 1994. (Stochastic Modeling Series). ISBN 9780412051715. Disponível em: <http://books.google.com.br/books?id=wTTUfYwjksAC>. Citado na página 45.

SANANDAJI, B. M. et al. Modeling and control of tubular solid-oxide fuel cell systems: Ii. nonlinear model reduction and model predictive control. *Journal of Power Sources*, v. 196, n. 1, p. 208 – 217, 2011. ISSN 0378-7753. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0378775310010621>. Citado na página 26.

SCHWAAB, M. et al. Nonlinear parameter estimation through particle swarm optimization. *Chemical Engineering Science*, v. 63, n. 6, p. 1542–1552, 2008. ISSN 0009-2509. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0009250907008755>. Acesso em: 23 de maio de 2012. Citado na página 27.

SINGH, T. S. D.; CHATTERJEE, A. A comparative study of adaptation algorithms for nonlinear system identification based on second order volterra and bilinear polynomial filters. *Measurement*, v. 44, n. 10, p. 1915 – 1923, 2011. ISSN 0263-2241. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0263224111002910>. Citado na página 26.

SOLO, V. *Time Series Recursions and Stochastic Approximation*. Victor Solo, 1978. Disponível em: <http://books.google.com.br/books?id=NuhgNAAACAAJ>. Citado na página 27.

SUBUDHI, B.; JENA, D. A differential evolution based neural network approach to nonlinear system identification. *Applied Soft Computing*, v. 11, n. 1, p. 861 – 871, 2011. ISSN 1568-4946. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1568494610000116>. Citado na página 26.

SUBUDHI, B.; JENA, D. Nonlinear system identification using memetic differential evolution trained neural networks. *Neurocomputing*, v. 74, n. 10, p. 1696 – 1709, 2011. ISSN 0925-2312. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0925231211001056>. Citado na página 26.

SURJANOVIC, S.; BINGHAM, D. *MATLAB implementation of the Langermann function*. 2013. Simon Fraser University. Disponível em: <http://www.sfu.ca/~ssurjano/Code/langerm.html>. Acesso em: 3 de junho de 2014. Citado na página 49.

TESLIC, L. et al. Nonlinear system identification by gustafson x2013;kessel fuzzy clustering and supervised local model network learning for the drug absorption spectra process. *Neural Networks, IEEE Transactions on*, v. 22, n. 12, p. 1941–1951, Dec 2011. ISSN 1045-9227. Citado na página 26.

UGALDE, H. M. R. et al. Neural network design and model reduction approach for black box nonlinear system identification with reduced number of parameters. *Neurocomputing*, v. 101, n. 0, p. 170 – 180, 2013. ISSN 0925-2312. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0925231212006522>. Citado na página 26.

VEILLETTE, M. *STBL: Alpha stable distributions for MATLAB*. 2012. Disponível em: <http://www.mathworks.com/matlabcentral/fileexchange/37514-stbl--alpha-stable-distributions-for-matlab/content/STBL_CODE/stblrnd.m>. Acesso em: 11 de fevereiro de 2013. Citado na página 46.

Weron, A.; Weron, R. Computer simulation of Lévy α-stable variables and processes. In: Garbaczewski, P.; Wolf, M.; Weron, A. (Ed.). *Chaos - The Interplay Between Stochastic and Deterministic Behaviour*. [s.n.], 1995. (Lecture Notes in Physics, Berlin Springer Verlag, v. 457), p. 379–392. Provided by the SAO/NASA Astrophysics Data System. Disponível em: <http://adsabs.harvard.edu/abs/1995LNP...457..379W>. Acesso em: 11 de fevereiro de 2014. Citado 2 vezes nas páginas 46 e 57.

WIENER, N. *Cybernetics: or Control and Communication in the Animal and the Machine*. 2. ed. Cambridge, MA: MIT Press, 1948. Disponível em: <http://www.bibsonomy.org/bibtex/277c5bcc762c988fab8ce3dada97a16ea/idsia>. Acesso em: 15 de dezembro de 2013. Citado na página 25.

YANG, X.-S. *Cuckoo Search (CS) Algorithm*. 2010. Disponível em: <http://www.mathworks.com/matlabcentral/fileexchange/29809-cuckoo-search-cs-algorithm>. Acesso em: 11 de fevereiro de 2014. Citado 2 vezes nas páginas 38 e 56.

YANG, X.-S. *Nature-Inspired Metaheuristic Algorithms*. 2. ed. University of Cambridge, Frome, BA11 6TT, United Kingdom: Luniver Press, 2010. 110 p. ISBN 978-1-905986-28-6. Disponível em: <http://books.google.com.br/books?id=iVB_ETlh4ogC>. Acesso em: 15 de dezembro de 2013. Citado 2 vezes nas páginas 15 e 58.

YANG, X.-S. *Nature-Inspired Metaheuristic Algorithms*. 2. ed. University of Cambridge, Frome, BA11 6TT, United Kingdom: Luniver Press, 2010. 108-111 p. ISBN 978-1-905986-28-6. Disponível em: <http://books.google.com.br/books?id=iVB_ETlh4ogC>. Acesso em: 15 de dezembro de 2013. Citado 2 vezes nas páginas 49 e 56.

YANG, X.-S.; DEB, S. Cuckoo search via lévy flights. In: *Nature Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*. [S.l.: s.n.], 2009. p. 210–214. Citado na página 37.

YAO, L.; SETHARES, W. Nonlinear parameter estimation via the genetic algorithm. *Signal Processing, IEEE Transactions on*, v. 42, n. 4, p. 927–935, Apr 1994. ISSN 1053-587X. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=285655>. Acesso em: 22 de janeiro de 2014. Citado na página 27.

ZHU, Q. An implicit least squares algorithm for nonlinear rational model parameter estimation. *Applied Mathematical Modelling*, v. 29, n. 7, p. 673 – 689, 2005. ISSN 0307-904X. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0307904X04001489>. Acesso em: 21 de dezembro de 2012. Citado 2 vezes nas páginas 28 e 65.

Apêndices

APÊNDICE A – Exemplo prático otimização por busca de cuco via voos de Lévy

A.1 Arquivo cuckoo_search.m

Listagem A.1 – Arquivo cuckoo_search.m

```
% ---------------------------------------------------------------------
% Cuckoo Search (CS) algorithm by Daniel Kaminski de Souza          %
% Based on Xin-She Yang and Suash Deb algoritm                       %
%                                                                   %
% Programmed by: Daniel Kaminski de Souza at:                       %
%   Universidade Federal do Paraná                                  %
%                                                                   %
% Oriented by:                                                      %
%   Gideon Villar                                                   %
%   Gustavo Oliveira                                                %
%                                                                   %
% Programming dates: Nov 2012                                       %
% Last revised: Nov 2012   (simplified version for demo only)       %
% ---------------------------------------------------------------------
% Papers -- Citation Details:
%
% 1) X.-S. Yang, S. Deb, Cuckoo search via Levy flights,
% in: Proc. of World Congress on Nature & Biologically Inspired
% Computing (NaBIC 2009), December 2009, India,
% IEEE Publications, USA,  pp. 210-214 (2009).
% http://arxiv.org/PS_cache/arxiv/pdf/1003/1003.1594v1.pdf
%
% 2) X.-S. Yang, S. Deb, Engineering optimization by cuckoo search,
% Int. J. Mathematical Modelling and Numerical Optimisation,
% Vol. 1, No. 4, 330-343 (2010).
% http://arxiv.org/PS_cache/arxiv/pdf/1005/1005.2908v2.pdf
%
% 3) R. N. Mantegna, Fast, accurate algorithm for numerical ...
    simulation of
% Levy stable stochastic processes. Dipartimento di Energetica ed ...
    Applica-
% zioni di Fisica, Università degli Studi, viale delle Scienze, I-90128
```

```
31 % Palermo, Italy (Received 28 October 1993)
32
33 % ------------------------------------------------------------------------%
34 % This demo program only implements a standard version of                 %
35 % Cuckoo Search (CS), as the Levy flights and generation of               %
36 % new solutions may use slightly different methods.                       %
37 % The pseudo code was given sequentially (select a cuckoo etc),           %
38 % but the implementation here uses Matlab's vector capability,            %
39 % which results in neater/better codes and shorter running time.          %
40 % This implementation is different and more efficient than the            %
41 % the demo code provided in the book by
42 %    "Yang X. S., Nature-Inspired Metaheuristic Algoirthms,               %
43 %     2nd Edition, Luniver Press, (2010).                 "               %
44 % ------------------------------------------------------------------------ %
45
46 % ========================================================================= %
47 % Notes:                                                                   %
48 % Different implementations may lead to slightly different                 %
49 % behavour and/or results, but there is nothing wrong with it,            %
50 % as this is the nature of random walks and all metaheuristics.           %
51 % ------------------------------------------------------------------------
52 % Presentation Data Selection:
53 % 0 = None
54 % 1 = Numbers
55 % 2 = Graphic Evolution
56
57 function [bestnest,fmin, N_iter]=cuckoo_search(fun, nvars, yd, ...
       initialTetaT, lb, ub, options)
58 %CUCKOO_SEARCH    Optimization using Cuckoo Search.
59 % Load default values if needed
60 if isempty(options)
61     %% Estimation Process
62     StallGenLimit_Data = 1000;
63     PopulationSize_Data = 15;
64     TolFun_Data = 1e-20;
65     DiscoveryRate_Data = 0.25;
66
67     % Presentation Data Selection:
68     % 0 = None
69     % 1 = Numbers
70     % 2 = Graphic Evolution
71     Presentation_Data = 2;
72
73     % LevyFlight Data Selection:
74     % 0 = Original
75     % 1 = Mantegna
76     LevyFlight_Data = 0;
```

```
77
78      % StepSize Data Selection:
79      % 0 = Original randn
80      % 1 = Fast randn versor
81      StepSize_Data = 0;
82
83      options = struct( ...
84          'StallGenLimit', StallGenLimit_Data, ...
85          'TolFun', TolFun_Data, ...
86          'PopulationSize', PopulationSize_Data, ...
87          'DiscoveryRate', DiscoveryRate_Data,...
88          'PresentationType', Presentation_Data,...
89          'LevyFlight', LevyFlight_Data,...
90          'StepSize', StepSize_Data...
91      );
92  end
93
94  % Discovery rate of alien eggs/solutions
95  pa=options.DiscoveryRate;
96
97  %% Change this if you want to get better results
98  % Tolerance
99  StallGenLimitOnTwo = options.StallGenLimit/2;
100
101 nd = nvars;
102
103 % initial solutions
104 %nest=lb+(ub-lb).*rand(options.PopulationSize, yd, nd);
105 A = rand(options.PopulationSize, yd, nd);
106 B = lb+(ub-lb);
107 for i=1:nd
108     A(:,:,i) = B(i)*A(:,:,i);
109 end
110 clear B
111 nest = A;
112 clear A
113
114 clones = nest;
115 if (isempty(initialTetaT)~=1)
116     clones(1,:,:) = initialTetaT;
117 else
118     clones = nest;
119 end
120
121 % Get the current best
122 fitness=Inf*ones(options.PopulationSize,1);
123 [fmin,bestnest,nest,fitness]=get_best_nest(fun, nest,clones,fitness);
```

```
124
125 if options.PresentationType == 2    % Check if Graphic Evolution ...
       selected
126     previousFigure = gcf;
127     graphicEvolutionFigure = figure();
128     hold;
129     h = plot(1,0);
130     title('Cucko estimation evolution')
131     xlabel('iteration')
132     ylabel('fmin')
133     xdata_array = [];
134     ydata_array = [];
135     set(h,'XData',xdata_array)
136     set(h,'YData',ydata_array)
137     a = [Inf];
138     b = [0];
139 end
140
141 N_iter=0;
142 i = 0;
143 change = Inf*ones(StallGenLimitOnTwo, 1);
144 changeMean = Inf;
145 %% Starting iterations
146 while (changeMean > options.TolFun),
147     % Generate new solutions (but keep the current best)
148      new_nest=get_cuckoos(nest,bestnest,lb,ub, options);
149      [¬,¬,nest,fitness]=get_best_nest(fun, nest,new_nest,fitness);
150      % Update the counter
151       N_iter=N_iter+options.PopulationSize;
152     % Discovery and randomization
153       new_nest=empty_nests(nest,lb,ub,pa) ;
154     % Evaluate this set of solutions
155     [fnew,best,nest,fitness]=get_best_nest(fun, nest,new_nest,fitness);
156     % Update the counter again
157       N_iter=N_iter+options.PopulationSize;
158     % Find the best objective so far
159     if fnew<fmin,
160         in = i+1;
161         a = [a fnew];
162         b = [b in];
163         currentChange = (a(end-1) - a(end))/(b(end) - b(end-1));
164         bestnest=best;
165         switch options.PresentationType
166             case 1
167             disp(num2str(changeMean))
168             case 2  % Check if Graphic Evolution selected
169                 figure(graphicEvolutionFigure);
```

```
                if length(a) > 60
                    ylim([a(end) a(end-60)]);
                end
                xlim([0 in]);
                set(h,'XData',b);
                set(h,'YData',a);
                refreshdata
                drawnow
            end
            fmin=fnew;
        else
            currentChange = 0;
        end
        index  = (rem(i,StallGenLimitOnTwo)+1);
        change(index) = currentChange;
        changeMean = mean(change);
        i = i + 1;
    end %% End of iterations
    switch options.PresentationType
    case 1
        disp(num2str(changeMean))
    case 2
        figure(previousFigure)
    end
    %% Post-optimization processing
    %% Display all the nests
    %disp(strcat('Total number of iterations=',num2str(N_iter)));
    %fmin
    %bestnest
    %change
    %mean(change)


    %% ---------------- All subfunctions are list below ------------------
    %% Get cuckoos by ramdom walk
    function nest=get_cuckoos(nest,best,lb,ub,options)
    % Levy flights
    sizeNest = size(nest);
    n=sizeNest(1);
    % Levy exponent and coefficient
    % For details, see equation (2.21), Page 16 (chapter 2) of the book
    % X. S. Yang, Nature-Inspired Metaheuristic Algorithms, 2nd Edition, ...
        Luniver Press, (2010).
    reshapeSize = sizeNest(2:end);
    levyFlightOption = options.LevyFlight;
    stepSizeOption = options.StepSize;
```

```
beta=3/2;
for j=1:n,
    s=reshape(nest(j,:,:), reshapeSize);
    % This is a simple way of implementing Levy flights
    % For standard random walks, use step=1;
    %% Levy flights by Mantegna's algorithm
    switch levyFlightOption
    case 0
        sigma=(gamma(1+beta)*sin(pi*beta/2)/(gamma((1+beta)/2)*beta*2^((beta-1)/2)))^(1
        u=randn(size(s))*sigma;
        v=randn(size(s));
        step=u./abs(v).^(1/beta);
    case 1
        step = mantegna(beta, 1, 1, s);
    end

    % In the next equation, the difference factor (s-best) means that
    % when the solution is the best solution, it remains unchanged.
    switch stepSizeOption
    case 0
        stepsize=0.01*step.*(s-best);
        % Here the factor 0.01 comes from the fact that L/100 should ...
           the typical
        % step size of walks/flights where L is the typical lenghtscale;
        % otherwise, Levy flights may become too aggresive/efficient,
        % which makes new solutions (even) jump out side of the ...
           design domain
        % (and thus wasting evaluations).
        % Now the actual random walks or flights
        s=s+stepsize.*randn(size(s));
        % Apply simple bounds/limits
        nest(j,:,:)=simplebounds(s,lb,ub);

    case 1
        if (s ~= best)
            % stepsize = step;
            stepsize=0.01*step.*(s-best);
            % Now the actual random walks or flights
            % From ...
               http://stackoverflow.com/questions/9750908/how-to-generate-a-unit-vector
            v = randn(size(s));
            randomDistributedVersor = ...
               bsxfun(@rdivide,v,sqrt(sum(v.^2)));

            s=s+stepsize.*randomDistributedVersor;
            % Apply simple bounds/limits
            nest(j,:,:)=simplebounds(s,lb,ub);
```

```
259         end
260     end
261 end
262
263 %% Find the current best nest
264 function [fmin,best,nest,fitness]=get_best_nest(fun, ...
       nest,newnest,fitness)
265 % Evaluating all new solutions
266 sizeNest = size(nest);
267 reshapeSize = sizeNest(2:end);
268 for j=1:sizeNest(1),
269     currentNest = reshape(newnest(j,:,:), reshapeSize);
270     fnew=fun(currentNest);
271     if fnew<fitness(j),
272        fitness(j)=fnew;
273        nest(j,:,:)=currentNest;
274     end
275 end
276 % Find the current best
277 [fmin,K]=min(fitness) ;
278 best=reshape(nest(K,:,:), reshapeSize);
279
280 %% Replace some nests by constructing new solutions/nests
281 function new_nest=empty_nests(nest,lb,ub,pa)
282 % A fraction of worse nests are discovered with a probability pa
283 sizeNest = size(nest);
284 n=sizeNest(1);
285 % Discovered or not -- a status vector
286 K=rand(sizeNest)>pa;
287
288 % In the real world, if a cuckoo's egg is very similar to a host's ...
       eggs, then
289 % this cuckoo's egg is less likely to be discovered, thus the ...
       fitness should
290 % be related to the difference in solutions.  Therefore, it is a ...
       good idea
291 % to do a random walk in a biased way with some random step sizes.
292 %% New solution by biased/selective random walks
293 stepsize=rand*(nest(randperm(n),:,:)-nest(randperm(n),:,:));
294 new_nest=nest+stepsize.*K;
295
296 signalsNumber=sizeNest(2);
297 LB = reshape(repmat(lb, n*signalsNumber,1), n, signalsNumber, ...
       length(lb));
298 UB = reshape(repmat(ub, n*signalsNumber,1), n, signalsNumber, ...
       length(ub));
299 new_nest=simplebounds(new_nest,LB,UB);
```

```
300
301 % Application of simple constraints
302 function s=simplebounds(s,Lb_m,Ub_m)
303   % Apply the lower bound
304   ns_tmp=s;
305
306   I=ns_tmp<Lb_m;
307   ns_tmp(I)=Lb_m(I);
308
309   % Apply the upper bounds
310   J=ns_tmp>Ub_m;
311   ns_tmp(J)=Ub_m(J);
312   % Update this new move
313   s=ns_tmp;
```

A.2 Arquivo langer2.m

Listagem A.2 – Arquivo langer2.m

```
1  function [ z ] = langer2( x, y )
2  %LANGER2 Summary of this function goes here
3  %   Detailed explanation goes here
4
5  %function [y] = langer(xx, m, c, A)
6  m = 5;
7  c = [1 2 5 2 3];
8  A = [3 5
9       5 2
10      2 1
11      1 4
12      7 9];
13
14 z = langer([x y], m, c, A);
15
16 end
```

A.3 Arquivo langer3.m

Listagem A.3 – Arquivo langer3.m

```
1  function [ z ] = langer3( x )
2  %LANGER2 Summary of this function goes here
```

```
3  %   Detailed explanation goes here
4
5  %function [y] = langer(xx, m, c, A)
6  m = 5;
7  c = [1 2 5 2 3];
8  A = [3 5
9       5 2
10      2 1
11      1 4
12      7 9];
13
14 z = -langer(x, m, c, A);
15
16 end
```

A.4 Arquivo mainScript.m

Listagem A.4 – Arquivo mainScript.m

```
1  clear all
2  close all
3  clc
4
5  x = linspace(0,5, 100);
6  y = linspace(0,5, 100);
7  [X, Y] = meshgrid(x, y);
8
9  Z = arrayfun(@langer2, X, Y);
10 surf(X, Y, Z);
11 shading interp
12 hold
13
14 %% Genetic Algorithm optimization
15 gaRes = ga(...
16     @langer3, ...  % function handle
17     2, ...         % number of parameters to estimate
18     [], ...
19     [], ...
20     [], ...
21     [], ...
22     [0 0], ...     % lower bound
23     [10 10]...     % upper bounds
24 );
25
26 %% Cuckoo Search via Lévy flights evaluation
```

```
27 [csRes, fmin, N_iter]=cuckoo_search(...
28     @langer3, ...        % function handle
29     2, ...               % number of parameters to estimate
30     1, ...               % number of output signals
31     [], ...              % initial parameters to use
32     [0 0], ...           % lower bounds
33     [10 10], ...         % upper bounds
34     [] ...               % options struct
35 );
36
37 %% Plot results
38 plot3(gaRes(1), gaRes(2), -langer3(gaRes), 'w*');
39 plot3(csRes(1), csRes(2), -langer3(csRes), 'b*');
```

Anexos

ANEXO A – Exemplo prático otimização por busca de cuco via voos de Lévy

A.1 Arquivo mantegna.m

Listagem A.1 – Arquivo mantegna.m

```
1  function z = mantegna(alpha, c, n, matrix)
2  %MANTEGNA Stable random number generator.
3  % Z = MANTEGNA(ALPHA, C, n, N)
4  %
5  % Based on the method of R. N. Mantegna "Fast, accurate algorithm for
6  % numerical simulation of Lévy stable stochastic processes"
7  % Physical Review E 49 4677–83 (1994).
8  %
9  % alpha: defines the index of the distribution and controls the ...
       scale pro–
10 % perties of the stochastic process {z}.
11 %
12 % c: selects the scale unit of the process. It's the gamma letter inside
13 % Mantegna's paper.
14 %
15 % n: number of independent stochastic variables w calculated by equation
16 % (15).
17 %
18 % Z: Stochastic process.
19 matrixSize = size(matrix);
20 % Errortraps:
21 if (alpha < 0.3 || alpha > 1.99)
22     disp('Valid range for alpha is [0.3;1.99].')
23   z = NaN * zeros(matrixSize);
24     return
25 end
26 if (c ≤ 0)
27     disp('c must be positive.')
28     z = NaN * zeros(matrixSize);
29     return
30 end
31 if (n < 1)
32     disp('n must be positive.')
33     z = NaN * zeros(matrixSize);
34     return
```

```
end
if nargin<4
    matrixSize = size(zeros(1,1));
end
if (matrixSize(2) ≤ 0)
    disp('N must be positive.')
    z = NaN;
    return
end

invalpha = 1/alpha;
%% sigmaX calculation in function of alpha. Equation (12).
sigx = ((gamma(1+alpha)*sin(pi*alpha/2))/(gamma((1+alpha)/2)...
*alpha*2^((alpha-1)/2)))^invalpha;

%% v calculation in function of alpha. Equation (6).
sigy = 1;
x = sigx*randn(matrixSize);
y = sigy*randn(matrixSize);
v = x./abs(y).^invalpha;

%% kappa calculation in function of alpha. Equation (20).
kappa = (alpha*gamma((alpha+1)/(2*alpha)))/gamma(invalpha)...
*((alpha*gamma((alpha+1)/2))/(gamma(1+alpha)*sin(pi*alpha/2)))^invalpha;

%% C estimation
p = [-17.7767 113.3855 -281.5879 337.5439 -193.5494 44.8754];
C_a = polyval(p, alpha);

w = ((kappa-1)*exp(-v/C_a)+1).*v;    %   equation (15)
if(n>1)
    z = (1/n^invalpha)*sum(w);
else
    z = w;
end
z = c^invalpha*z;
```

A.2 Arquivo langer.m

Listagem A.2 – Arquivo langer.m

```
function [y] = langer(xx, m, c, A)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
```

```
5  % LANGERMANN FUNCTION
6  %
7  % Authors: Sonja Surjanovic, Simon Fraser University
8  %          Derek Bingham, Simon Fraser University
9  % Questions/Comments: Please email Derek Bingham at ...
       dbingham@stat.sfu.ca.
10 %
11 % Copyright 2013. Derek Bingham, Simon Fraser University.
12 %
13 % THERE IS NO WARRANTY, EXPRESS OR IMPLIED. WE DO NOT ASSUME ANY ...
       LIABILITY
14 % FOR THE USE OF THIS SOFTWARE.  If software is modified to produce
15 % derivative works, such modified software should be clearly marked.
16 % Additionally, this program is free software; you can redistribute it
17 % and/or modify it under the terms of the GNU General Public License as
18 % published by the Free Software Foundation; version 2.0 of the License.
19 % Accordingly, this program is distributed in the hope that it will be
20 % useful, but WITHOUT ANY WARRANTY; without even the implied warranty
21 % of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
22 % General Public License for more details.
23 %
24 % For function details and reference information, see:
25 %
26 %
27 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
28 %
29 % INPUTS:
30 %
31 % xx = [x1, x2, ..., xd]
32 % m = constant (optional), with default value 5
33 % c = m-dimensional vector (optional), with default value [1, 2, 5, ...
       2, 3]
34 % A = (mxd)-dimensional matrix (optional), with default value
35 %     [3, 5; 5, 2; 2, 1; 1, 4; 7, 9]
36 %
37 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
38
39 d = length(xx);
40
41 if (nargin < 4)
42     A = [3, 5; 5, 2; 2, 1; 1, 4; 7, 9];
43 end
44 if (nargin < 3)
45     c = [1, 2, 5, 2, 3];
46 end
47 if (nargin < 2)
48     m = 5;
```

```
49 end
50
51 outer = 0;
52 for ii = 1:m
53     inner = 0;
54     for jj = 1:d
55         xj = xx(jj);
56         Aij = A(ii,jj);
57         inner = inner + (xj-Aij)^2;
58     end
59     new = c(ii) * exp(-inner/pi) * cos(pi*inner);
60     outer = outer + new;
61 end
62
63 y = outer;
64
65 end
```

Índice