UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Estudo Comparativo de Técnicas de Inteligência de Enxame na Redução da Ordem de Sistemas Dinâmicos Lineares

Marlon John Pinheiro Silva

DM: 47/2019

UFPA/ITEC/PPGEE
Campus Universitário do Guamá

Belém - Pará - Brasil
2019

UNIVERSIDADE FEDERAL DO PARÁ

INSTITUTO DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Marlon John Pinheiro Silva

Estudo Comparativo de Técnicas de Inteligência de Enxame na Redução da Ordem de Sistemas Dinâmicos Lineares

Dissertação submetida à banca examinadora do Programa de Pós-Graduação em Engenharia Elétrica da UFPA, para a obtenção do grau de Mestre em Engenharia Elétrica na área de Sistemas de Energia Elétrica.

Orientador: Prof. Dr. Carlos Tavares da Costa Júnior

Co-orientador: Prof. Dr. Orlando Fonseca Silva

UFPA/ITEC/PPGEE

Campus Universitário do Guamá

Belém – PA – Brasil

2019

Dados Internacionais de Catalogação na Publicação (CIP) de acordo com ISBD
Sistema de Bibliotecas da Universidade Federal do Pará
Gerada automaticamente pelo módulo Ficat, mediante os dados fornecidos pelo(a) autor(a)

P654e Pinheiro Silva, Marlon John
Estudo comparativo de técnicas de Inteligência de Enxame na redução da ordem de sistemas dinâmicos lineares / Marlon John Pinheiro Silva. — 2019.
xv, 114 f. : il. color.

Orientador(a): Prof. Dr. Carlos Tavares da Costa Júnior
Coorientador(a): Prof. Dr. Orlando Fonseca Silva
Dissertação (Mestrado) - Programa de Pós-Graduação em Engenharia Elétrica, Instituto de Tecnologia, Universidade Federal do Pará, Belém, 2019.

1. Metaheurísticas. 2. Inteligência de Enxame. 3. Sistemas de Controle. 4. Redução de Ordem. I. Título.

CDD 629.8312

UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

"ESTUDO COMPARATIVO DE TÉCNICAS DE INTELIGÊNCIA DE ENXAME NA REDUÇÃO DA ORDEM DE SISTEMAS DINÂMICOS LINEARES"

AUTOR: MARLON JOHN PINHEIRO SILVA

DISSERTAÇÃO DE MESTRADO SUBMETIDA À BANCA EXAMINADORA APROVADA PELO COLEGIADO DO PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA, SENDO JULGADA ADEQUADA PARA A OBTENÇÃO DO GRAU DE MESTRE EM ENGENHARIA ELÉTRICA NA ÁREA DE SISTEMAS DE ENERGIA ELÉTRICA.

APROVADA EM: 17/12/2019

BANCA EXAMINADORA:

Prof. Dr. Carlos Tavares da Costa Júnior
(Orientador – PPGEE/UFPA)

Prof. Dr. Orlando Fonseca Silva
(Co-Orientador – FEEB/UFPA)

Prof. Dr. Antônio da Silva Silveira
(Avaliador Interno – PPGEE/UFPA)

Prof. Dr. Gustavo Sobral Toscano
(Avaliador Externo ao Programa – FEEB/UFPA)

VISTO:

Prof.ª Dr.ª Maria Emília de Lima Tostes
(Coordenadora do PPGEE/ITEC/UFPA)

“É permanecendo firmes que
ganharemos a vida”

DEDICATÓRIA

Dedico este trabalho,
todo o esforço e
tempo para construí-lo
à minha família.

AGRADECIMENTOS

Agradeço, acima de tudo, a Deus Pai, Todo-Poderoso, Criador de todas as coisas, cujo sopro de vida me mantem e me sustém. Agradeço a meu Senhor Jesus Cristo, meu único salvador, que sempre intercede por mim, o qual deposito toda a minha fé. Agradeço por mais esta conclusão de etapa em minha vida. Pela elaboração e conclusão deste trabalho. Pelas pessoas que surgiram em meio ao caminho e contribuíram fortemente para a construção deste. Pela força, pelo amparo, proteção, por tudo. Meu Deus, meu Senhor Jesus Cristo, muito obrigado por tudo!

Agradeço, em particular, a minha família, que sempre esteve presente em minha vida, incentivando e me apoiando em minha vida acadêmica, escolar e profissional. Agradeço a minha mãe, Maria de Jesus Pinheiro Silva, por seu imenso amor. Agradeço a meu pai, Ananias da Costa Coelho, cuja memória me é fortíssima. Agradeço aos meus irmãos, Madson Bruno Silva Coelho e Ananias da Costa Coelho Júnior, que estiveram ao meu lado quando precisei. Agradeço a Darlene Rocha Lima, que me incentivou e me apoiou a crescer desde o começo, depositando seu carinho e admiração. Agradeço-os por participarem ativamente de minhas viagens, na publicação do artigo que gerou esta dissertação, demonstrando toda a sua preocupação e atenção.

Agradeço, em especial, aos meus professores. Agradeço ao professor Orlando Fonseca Silva, “Nick”, tutor de longa data que me iniciou à pesquisa acadêmica e até hoje participa dos trabalhos, sendo meu co-orientador na elaboração deste. Agradeço ao professor Carlos Tavares da Costa Junior, meu orientador, cuja paciência e oportunidade me foi dada. E claro, agradeço a Juan Vidal, colega de trabalho, que me apoiou a reescrever e continuar esta dissertação.

A todos, meu Muito Obrigado!!

RESUMO

A redução de ordem de modelos tem se mostrado um problema bastante recorrente e diversas técnicas surgiram ao longo dos anos, quando, do ponto de vista do projeto de controladores, se tornou inadequada a elaboração e construção destes, visto o alto grau de redundância, que sistemas físicos reais de grande porte podem possuir. No âmbito da matemática determinística, muitos trabalhos, já consagrados na literatura, se propuseram a resolver tal problemática. Recentemente, técnicas que envolvem métodos metaheurísticos em um espaço de busca pré-determinado, utilizando Inteligência de Enxames, vêm sendo utilizados com bastante êxito e tem se mostrado uma nova ferramenta como solução. Com base neste contexto, este trabalho apresenta a compreensão do problema sob o ponto de vista da teoria de sistemas lineares; realizando um estudo comparativo entre as Inteligências de Enxames: *Firefly Algorithm*, enxame de partículas (PSO do inglês - *Particle Swarm Optimization*) e SFLA (do inglês - *Shuffled Frog Leaping Algorithm*).

Palavras-chave: Metaheurísticas, Inteligência de Enxame, Sistemas de Controle, Redução de Ordem.

ABSTRACT

Model order reduction has been a recurring problem and several techniques have emerged over the years, when, from the point of view of controller design, their elaboration and construction became inadequate, considering the high degree of redundancy, which large real physical systems may possess. In the field of deterministic mathematics, many works, already consecrated in the literature, have proposed to solve such problem. Recently, techniques involving metaheuristic methods in a predetermined search space using Swarm Intelligence (SI) have been used quite successfully and a new tool has been shown as a solution. Based on this context, this paper presents the understanding of the problem from the point of view of linear systems theory; conducting a comparative study between the Swarm Intelligences: *Firefly Algorithm*, PSO - *Particle Swarm Optimization* and SFLA - *Shuffled Frog Leaping Algorithm*.

Keywords: Metaheuristics, Swarm Intelligence, Control Systems, Order Reduction.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

CAPÍTULO 1
INTRODUÇÃO

1.1- Revisão Bibliográfica

A simplificação ou redução de ordem de modelos preocupa-se em desenvolver técnicas que permitam, dado um modelo G(s) de ordem n, obter um modelo reduzido, R(s) de ordem m, tal que m < n, que se aproxime do modelo original, ou seja, R(s) ≈ G(s). Há diversos aspectos em que dois modelos podem ser aproximados e qual usar dependerá grandemente dos objetivos da aplicação.

A redução de ordem de modelos dinâmicos tem se mostrado uma técnica efetiva para a simulação de sistemas de grande porte, como por exemplo, sistemas de geração de energia elétrica interligados por linhas de transmissão, já que estes modelos de ordem elevada normalmente possuem um alto grau de redundância e complexidade, o que pode dificultar o processo de simulação, análise ou projeto de controladores. Deste modo torna-se útil e muitas vezes necessário representar tais sistemas usando modelos de baixa ordem que representem adequadamente as características dinâmicas dos mesmos (Bansal *et al.*, 2011; Vasu *et al.*, 2012; Sambariya & Sharma, 2016).

A redução de ordem de modelos, ou aproximação de modelos de sistemas físicos complexos de ordem elevada por modelos de menor ordem mais simples, tem sido objeto de muitas pesquisas há bastante tempo tendo atraído grande atenção em meados da década de sessenta, quando a capacidade de processamento de computadores era bastante limitada, até o fim da década de oitenta, sendo que desde meados da década de oitenta, especial atenção foi dada à redução de ordem para o projeto de controladores (Mansour & Mehrotra, 2003).

Muitas técnicas vêm sendo propostas na literatura para realizar a redução de ordem. De acordo Bottura e Munaro (1994), tais técnicas ou métodos

propostos são diversificados e com diferentes abordagens para derivar modelos reduzidos, como por exemplo: agregação modal, onde procura-se eliminar modos pouco dominantes no sistema (Davison,1966; Aoki,1968; Wilson, Fisher e Seborg, 1972; Arbel e Tse, 1979; Marshall, 1966); Métodos baseados na descrição no espaço de estados também são muito aplicados, sendo que os mais utilizados são fundamentados na Transformação Balanceada (Moore, 1981; Pernebo e Silverman, 1982; Muscato, 2000) e Agregação em Cadeia que envolvem a determinação de subsistemas pouco controláveis/observáveis para serem eliminados do sistema original (Tse, Medanic e Perkins, 1977; Drenick, 1975; Jamshidi, 1983).

Métodos baseados em otimização também foram propostos, onde alguma função é minimizada ou maximizada com respeito aos parâmetros do modelo (El-Attar e Vidyasagar, 1978; Hsia, 1972).

Uma vez que os sistemas encontrados apresentam características próprias, não foi possível, ainda, o desenvolvimento de um procedimento sistemático que seja aplicável a todos os casos. O que se observa é que cada método é melhor aplicado em uma situação específica e tendo suas próprias vantagens e desvantagens. O foco de cada técnica pode variar de acordo com a aplicação, como por exemplo, pode se ter mais interesse em produzir modelos que se aproximem do comportamento do modelo original em baixas frequências ou até mesmo produzir respostas com bons resultados de aproximação para entradas do tipo degrau ou impulsiva (Bansal *et al.*, 2011).

Entre as técnicas clássicas propostas na literatura, tem-se, por exemplo, a aproximação de Padé (Aguirre, 2007), o método de expansão de fração contínua (Chen & Shieh, 1968), o método de correspondência de momentos (Paynter & Takahashi, 1956), o método de aproximação de Routh (Hutton & Friendland, 1975) e o método de retenção de polos dominantes (Davison, 1966). Embora muitas dessas abordagens clássicas produzam modelos reduzidos com respostas temporais estáveis, em algumas situações o modelo obtido pode vir a apresentar a característica de fase não mínima, o que é indesejável. Na tentativa de se obter melhores modelos de ordem reduzida tem-se utilizado técnicas de otimização em conjunto com técnicas clássicas, sendo que entre os métodos

propostos destacam-se os que utilizam algoritmos de inteligência computacional (IC) (Lopes & Takahashi, 2011).

Entre os algoritmos pertencentes ao campo de IC, as metaheurísticas são técnicas eficientes para problemas de otimização em espaço de busca complexo, visando a produção de soluções aceitáveis em tempos hábeis. Tais características as tornam excelentes candidatas para o uso no problema da redução de ordem de modelos. O trabalho de Ferreira *et al.* (2011) apresenta uma abordagem para a otimização da norma do coeficiente da função de erro entre o modelo original e o reduzido utilizando algoritmos genéticos (AG's). Assadi e Abut (2010) utilizam AG's para realizar a redução de ordem de modelos, tendo uma função objetivo baseada nos coeficientes da Transformada Rápida de Fourier. Na literatura estão disponíveis outros trabalhos que utilizam AG's para solucionar este tipo de problema (Parmar *et al.*, 2007; Saini & Prasad, 2010).

Outra classe de metaheurísticas que vem sendo bastante utilizada para a redução de ordem de modelos são os algoritmos de inteligência de enxame (IE) que estão incluídos como técnicas de IC (Sambariya & Sharma, 2016; Hachino *et al.*, 2015; Marella *et al.*, 2014; Nadi *et al.*, 2011). No trabalho de Vasu *et al.* (2012) é realizada a redução de ordem de sistemas com uma entrada e uma saída (SISO do inglês – *single input single output*), usando uma abordagem que combina o método dos mínimos quadrados com o método de otimização de enxame de partículas (PSO do inglês - *Particle Swarm Optimization*), para obter, respectivamente, os coeficientes dos polinômios do denominador e numerador da função de transferência do modelo reduzido.

Ainda em se tratando de algoritmos de inteligência de enxame, tem-se atualmente o algoritmo de vagalume (FA do inglês - *Firefly Algorithm*), que é um algoritmo inspirado no comportamento social de vagalumes, proposto por Xin-She Yang (Yang, 2008).

Neste trabalho, apresenta-se uma metodologia para realizar a redução de ordem de modelos de sistemas dinâmicos lineares utilizando três técnicas de Inteligência de Enxames a saber: o FA, PSO e SFLA (do inglês, *Shuffled Frog*

Leaping Algorithm), com o auxílio do *software* de simulação computacional Matlab (2017).

A metodologia se baseia na minimização do erro das respostas, do sistema original e do modelo reduzido, para um sinal de entrada do tipo degrau unitário. Propõe-se manter as características dinâmicas do modelo original e para avaliar a técnica proposta faz-se uma comparação com uma técnica clássica, determinística, para verificar a eficácia de cada técnica de IE. Apresenta-se também o índice de dominância modal (IDM) como ferramenta para avaliar primeiramente a ordem do modelo reduzido. Por fim, é feita a análise comparativa dos três algoritmos estocásticos propostos.

1.2- Objetivos

Como objetivos gerais para o trabalho tem-se:

- A compreensão do problema de redução de ordem de sistemas dinâmicos sob o ponto de vista da teoria de sistemas lineares;
- Apresentar uma revisão bibliográfica sobre várias técnicas existentes e suas aplicações;
- Apresentar o IDM como ferramenta para avaliação da possível ordem de um modelo reduzido.
- Apresentar uma técnica de redução de ordem clássica baseada na Minimização da Norma dos Coeficientes Polinomiais do Erro.

Como objetivos específicos do trabalho tem-se:

- Apresentar as principais características dos Algoritmos de Inteligência de Enxame, detalhando os algoritmos FA, PSO e SFLA que foram utilizados neste trabalho.

- Elaborar um algoritmo em Matlab para determinação de modelos reduzidos de sistemas dinâmicos, contínuos no tempo, e monovariáveis via IE.

- A avaliar o desempenho e esforço computacional dos três algoritmos propostos, comparativamente a um método clássico determinístico, tomando a resposta ao degrau e a resposta em frequência para esta avaliação.

- Documentar o estudo e seus resultados através da dissertação e publicação de artigo em congresso nacional.

1.3- Estrutura do Trabalho

O trabalho está organizado em cinco capítulos a partir desta introdução, como apresentado a seguir:

O **Capítulo 2** aborda os conceitos de redução de ordem de sistemas e apresenta um método clássico determinístico que será usado na comparação com os métodos estocásticos, cujas características e componentes são destacados no **Capítulo 3** onde se apresenta a estrutura do FA, PSO e SFLA na redução de ordem de sistemas dinâmicos.

O **Capítulo 4** destaca os resultados numéricos obtidos.

O **Capítulo 5** faz uma discussão e observações sobre os resultados obtidos e apresenta as conclusões sobre os métodos utilizados. Mostra, ainda, que a estratégia utilizando Inteligência de Enxames é uma boa alternativa para o problema de redução de ordem, e apresenta algumas propostas futuras de trabalho.

CAPÍTULO 2
REDUÇÃO DA ORDEM DE SISTEMAS DINÂMICOS

2.1- Introdução

Como a redução de ordem tanto pode aplicar-se a modelos de larga escala, ou seja, de ordem muito elevada, como para modelos de ordem moderada, em aplicações típicas de sistemas de controle, neste capítulo apresenta-se o método proposto por Araújo (2008) que utiliza-se de otimização para redução de modelos estáveis, de fase mínima, e com ordem moderada.

A técnica proposta consiste em analisar uma função de erro entre o modelo real e o de ordem reduzida, para, a partir da minimização da norma dos coeficientes do polinômio do numerador desta função, encontrar os parâmetros desconhecidos do modelo de ordem reduzida (Araújo, 2008).

Como a obtenção de um modelo matemático, geralmente, é o ponto de partida para as aplicações de sistemas de controle, primeiramente uma breve revisão de conceitos e definições de modelagem e suas representações será apresentada para um melhor entendimento do assunto.

Sendo ainda um modelo matemático uma representação aproximada de um sistema real, geralmente, pode se ter uma “família” de modelos e não uma única representação que reproduza tal sistema.

Assim, um sistema físico pode ser expresso de diversas maneiras. Dependendo do sistema em questão e das características de interesse, uma representação pode ser mais adequada do que outra. A seguir, são apresentadas as representações lineares mais comuns para sistemas físicos.

2.2- Equação Diferencial

O comportamento dinâmico de muitos sistemas contínuos no tempo pode ser expresso por equações diferenciais ordinárias. A equação diferencial por sua vez é normalmente obtida considerando-se as leis físicas que regem tal sistema.

Quando o modelo é uma representação válida de um sistema, informações significativas podem ser retiradas sobre sua dinâmica ou seu desempenho. Portanto pode-se definir modelo como uma representação física, matemática, lógica ou computacional qualquer de um sistema, processo, fenômeno ou entidade. Segundo a sua natureza, os modelos são classificados em físicos, matemáticos, lógicos e, recentemente, computacionais (Trivelato, 2003).

Lembrando, ainda, que um sistema linear invariante no tempo é aquele em que um deslocamento temporal no sinal de entrada causa o mesmo deslocamento temporal no sinal de saída (sua estrutura e parâmetros não se alteram com o tempo) e pode ser representado pela equação diferencial ordinária da Equação (2.1), para $n \geq m$, onde x é a entrada do sistema, y é sua saída e os coeficientes b_0 , b_1 , … , b_{n-1} , b_n e a_0 , a_1 , … , a_{m-1} , a_m são constantes dadas por combinações dos parâmetros dos componentes do sistema.

$$b_0 y^n + b_1 y^{n-1} + .. + b_{n-1} y' + b_n y = a_0 x^m + a_1 x^{m-1} + .. + a_{m-1} x' + a_m x \qquad (2.1)$$

2.3- Função de Transferência

A função de transferência (FT) é uma representação muito utilizada e importante para sistemas lineares. Ela descreve a relação dinâmica de causa e efeito entre uma entrada e uma saída de um determinado sistema, no domínio da frequência (Aguirre, 2007).

A FT de um sistema é obtida aplicando-se a transformada de Laplace ($\mathcal{L}$) à equação diferencial que descreve tal sistema, considerando-se condições iniciais nulas. A aplicação do conceito de função de transferência se restringe a sistemas lineares invariantes no tempo.

Aplicando a transformada de Laplace na Equação (2.1), obtém-se a FT G(s) dada na Equação (2.2).

$$G(s) = \frac{\mathcal{L}[Saída]}{\mathcal{L}[Entrada]}\Big| = \frac{Y(s)}{X(s)}_{CI=0} = \frac{a(s)}{b(s)} = \frac{a_0 s^m + a_1 s^{m-1} + \cdots + a_{m-1} s + a_m}{b_0 s^n + b_1 s^{n-1} + \cdots + b_{n-1} s + b_n} \quad (2.2)$$

As raízes do polinômio do numerador são chamadas de zeros e as raízes do polinômio do denominador de polos. A ordem do sistema corresponde ao grau do polinômio do denominador, ou seja, n.

2.4- Representação no Espaço de Estados

Uma função de transferência descreve apenas a relação de entrada/saída de um sistema, não fornecendo informações a respeito do que ocorre no interior do sistema (entre os pontos de entrada e os pontos de saída). Por outro lado, a representação no espaço de estados fornece tais informações, pois evidencia também relações entre as variáveis internas ao sistema.

A análise em espaço de estados envolve três tipos de variáveis na modelagem de sistemas dinâmicos: variáveis de entrada, variáveis de saída e variáveis de estado. Observa-se que a representação em espaço de estados de um sistema não é única, ou seja, pode-se ter mais de um modelo em espaço de estados para o mesmo sistema. A representação em espaço de estados tem a estrutura mostrada nas Equações (2.3) e (2.4), sendo chamadas respectivamente, equação de estados e equação de saída.

$$\dot{x}(t) = Ax(t) + Bu(t) \quad (2.3)$$

$$y(t) = Cx(t) + Du(t) \quad (2.4)$$

$x(t) \in \mathbb{R}^n$ é denominado vetor de estado, e é constituído pelas variáveis de estados $x_1(t)$, $x_2(t)$, ..., $x_n(t)$. conforme a Equação (2.5). $\dot{x}(t)$ é a derivada temporal de $x(t)$, ou seja, $\dot{x}(t) = dx(t)/dt$

$$x(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \\ \dots \\ x_n(t) \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix} \equiv x \tag{2.5}$$

$u(t) \in \mathbb{R}^r$, é o vetor de entradas, que pode ser constituído de r funções temporais de entrada, conforme a Equação (2.6), e $y(t) \in \mathbb{R}^m$ é o vetor m-dimensional de saídas medidas, Equação (2.7).

.

$$u(t) = \begin{bmatrix} u_1(t) \\ u_2(t) \\ \dots \\ u_r(t) \end{bmatrix} \tag{2.6}$$

$$y(t) = \begin{bmatrix} y_1(t) \\ y_2(t) \\ \dots \\ y_m(t) \end{bmatrix} \tag{2.7}$$

⇒ A : Matriz de estado $(n \text{ x } n)$.

⇒ B : Matriz de entrada $(n \text{ x } r)$.

⇒ C : Matriz de saída $(m \text{ x } n)$.

⇒ D : Matriz de transmissão direta $(m \text{ x } r)$.

O sistema representado pelas Equações (2.3) e (2.4) será multivariável se $r > 1$ e/ou $m > 1$ e monovariável, caso $r = 1$ e $m = 1$.

A partir da representação em espaço de estados é possível obter a função de transferência de um sistema a partir da Equação (2.8).

$$\frac{Y(s)}{U(s)} = G(s) = C(Is - A)^{-1} B + D \tag{2.8}$$

O sistema considerado para a obtenção de (2.8) é monovariável. No caso de sistemas multivariáveis, a relação Y(s)/U(s) não será apenas a razão entre dois polinômios em s, mas poderá ser representada como a razão entre polinômios de matrizes em s (Aguirre, 2007). Em geral, a representação em espaço de estados é mais conveniente para representar sistemas multivariáveis do que a função ou matriz de transferência.

2.5- Índice de Dominância Modal - IDM

No problema de redução de ordem um procedimento comumente utilizado é reter os modos com as maiores constantes de tempo, denominados de polos dominantes (mais próximos ao eixo $j\omega$). Apesar de normalmente gerar bons resultados, esse procedimento pode apresentar problemas, tais como (Aguirre, 1993):

- Alguns modelos, apesar de terem modos dominantes, têm polos confinados a uma região limitada do plano s, o que dificulta a escolha baseada no critério de distância ao eixo imaginário;
- Os modos mais lentos podem não ser os mais dominantes;
- No caso de polos complexos conjugados a decisão nem sempre é fácil.

Para quantificar a dominância modal, usam-se alguns índices denominados Índices de Dominância Modal (IDM) que, apesar de bastante simples, são normalmente efetivos em determinar os modos dominantes em modelos, pois, além de levar em consideração as constantes de tempo de cada modo, informação contida nos polos, também usam informação sobre a localização dos zeros da FT contidas nos resíduos da expansão em frações parciais (Aguirre, 1993).

2.5.1- Modelos em Função de Transferência

Os IDM's aplicam-se tanto a funções de transferência como para representação de modelos na forma de espaço de estados (Aguirre, 2007). Seja, por exemplo, a FT na forma fatorada de seu denominador, da Equação (2.9).

$$G(s) = \frac{a_0 + a_1 s + \ldots + a_r s^r}{(s-\lambda_1)\ldots(s-\lambda_n)} \tag{2.9}$$

Pressupõe-se que G(s) não tem polos com multiplicidade maior que um, e $n > r$. De acordo com a técnica de expansão em frações parciais (Anexo I), pode-se escrever a função de transferência conforme a Equação (2.10) (Aguirre, 2007).

$$G(s) = \frac{J_1}{(s-\lambda_1)} + \cdots + \frac{J_k}{(s-\lambda_k)} + \frac{J_{k+1}}{(s-\lambda_{k+1})} + \frac{J^*_{k+1}}{(s-\lambda^*_{k+1})} + \cdots + \frac{J_{k+q}}{\left(s-\lambda_{k+q}\right)} + \frac{J^*_{k+q}}{(s-\lambda^*_{k+q})} \tag{2.10}$$

Em que J_i é o i-ésimo resíduo correspondente ao polo λ_i. Os asteriscos indicam complexos conjugados, k o número de polos reais e q o número de polos complexos conjugados. Logo $k + 2q = n$. Além de polos com multiplicidade um, assume-se que $Re(\lambda_i) < 0$, para todo i. Assim, definem-se os IDM's para modos reais conforme a Equação (2.11), em que γ_i é o i-ésimo IDM.

$$\gamma_i = -\frac{J_i}{\lambda_i} \quad , \quad i = 1,2,\ldots,k \tag{2.11}$$

Para polos complexos conjugados, a definição é dada de acordo com a Equação (2.12).

$$\gamma_i = \frac{-[J_{k+l}\,\lambda^*_{k+l} + J^*_{k+l}\,\lambda_{k+l}\,]}{2\lambda_{k+l}\,\lambda^*_{k+l}} = -\frac{Re\{J_{k+l}\,\lambda^*_{k+l}\}}{\lambda_{k+l}\,\lambda^*_{k+l}} \tag{2.12}$$

Para: $i = k + 2l - 1,\ k + 2l$ e $l = 1, 2, \ldots, q$

Da Equação (2.12) constata-se que polos complexos conjugados possuem IDM's idênticos, ou seja, $\gamma_{k+2l-1} = \gamma_{k+2l}$ (para $l = 1\,, 2\,, \ldots, q$) e que, ainda, podem ser tanto positivos quanto negativos. Assim, o IDM é uma indicação quantitativa da amplitude da contribuição de cada modo bem como a sua direção e distância em relação ao eixo imaginário para posteriormente se obter uma FT de ordem reduzida a partir dos modos de maior IDM.

Em termos percentuais, o cálculo do IDM percentual (IDM%) é dado pela Equação (2.13).

$$\gamma_{\%(i)} = \frac{|\gamma_i|}{\Gamma} \times 100\% \tag{2.13}$$

em que:

$$\Gamma = \sum_{i=1}^{n} (|\gamma_i|) \quad ; \quad para\ i = 1\,,2\,,\dots\,,n \tag{2.14}$$

Para o cálculo do IDM percentual acumulado ($\gamma_{\%a(i)}$), tem-se a Equação (2.15).

$$\gamma_{\%a(i)} = \sum_{i=1}^{n} \gamma_{\%(i)} \quad ; \quad para\ i = 1\,,2\,,\dots\,,n \tag{2.15}$$

Os polos e resíduos que farão parte da FT reduzida são aqueles correspondentes aos elementos, cujos valores são realmente significativos e possuem maior influência sobre a resposta do sistema, sendo esta, uma questão subjetiva e de análise (Aguirre, 2007). Os demais valores serão “descartados”, sendo assim, os polos e resíduos correspondentes a estes valores, também, serão eliminados.

Para modelos na forma de espaços de estados das Equações (2.3) e (2.4), os IDM’s para o par de entrada-saída entre a i-ésima entrada e a j-ésima saída são dados pela Equação (2.16).

$$diag\left[\gamma_1^{ij}\ \gamma_2^{ij}\ \dots\ \gamma_n^{ij}\right] = -Re\left\{\hat{C}_j\ \hat{B}_i\ \bar{A}^{-1}\right\}\ ;\ i = 1\,,2\,,\dots\,,v \quad j = 1\,,2\,,\dots\,,u \tag{2.16}$$

Onde $Re\{\ .\ \}$ indica a parte real e os demais termos são dados pelas Equações (2.17) a (2.24) (Aguirre, 2007), onde: v_i são os autovetores de A, C_j é a j-ésmia linha de C e $diag$ representa uma matriz diagonal com os elementos indicados.

$$V = [\, v_1\ v_2\ \ldots\ v_n\,] \tag{2.17}$$

$$\bar{A} = V^{-1} AV \tag{2.18}$$

$$C_j\, V = \left[\, \bar{c}^{\,j}_{\,1}\ \ \bar{c}^{\,j}_{\,2} \ldots\ \bar{c}^{\,j}_{\,n}\,\right], j = 1\,, 2\,, \ldots\ \,, u \tag{2.19}$$

$$\hat{C}_j = diag\left[\, \bar{c}^{\,j}_{\,1}\ \ \bar{c}^{\,j}_{\,2} \ldots\ \bar{c}^{\,j}_{\,n}\,\right] \tag{2.20}$$

$$C = [\, C_1\, C_2\ \ldots\ C_u\,]^T \tag{2.21}$$

$$V^{-1}B_i = \left[\, \bar{b}^{\,i}_{\,1}\ \ \bar{b}^{\,i}_{\,2} \ldots\ \bar{b}^{\,i}_{\,n}\,\right]^T \tag{2.22}$$

$$\hat{B}_i = diag\left[\, \bar{b}^{\,i}_{\,1}\ \ \bar{b}^{\,i}_{\,2} \ldots\ \bar{b}^{\,i}_{\,n}\,\right]^T, i = 1\,, 2\,, \ldots\,, v \tag{2.23}$$

$$B = [\, B_1\, B_2\ \ldots\ B_v\,] \tag{2.24}$$

2.6- Redução de ordem por minimização da norma dos coeficientes polinomiais do erro

Sendo a função de transferência de um sistema linear e invariante no tempo, dada pela Equação (2.26).

$$G(s) = \frac{a(s)}{b(s)} = \frac{a_n s^n + \cdots + a_2 s^2 + a_1 s + a_0}{b_m s^m + \cdots + b_2 s^2 + b_1 s + b_0} \tag{2.26}$$

O problema da redução de ordem é determinar uma nova função de transferência $\hat{G}(s)$, Equação (2.27), onde os graus dos polinômios $\hat{a}(s)$ e $\hat{b}(s)$ sejam menores que os graus dos polinômios $a(s)$ e $b(s)$, respectivamente e ainda, que seja equivalente a G(s), seja quanto a sua resposta no domínio do tempo para uma mesma entrada ou quanto a sua resposta no domínio da frequência:

$$\hat{G}(s) = \frac{\hat{a}(s)}{\hat{b}(s)} \ \cong\ G(s) \tag{2.27}$$

Destaca-se que certos polos do sistema, como os polos dominantes, podem ser retidos na solução, assim como alguns zeros, ou o grau relativo da

função de transferência original pode ser preservado (diferença entre o número de polos e zeros), o que resulta em diferentes formas de redução. Entretanto, em muitos casos não há polos dominantes e nem sempre é possível realizar o cancelamento de polos e zeros.

Pelo exposto, sendo a função $\hat{G}(s)$ a função reduzida de $G(s)$, esta pode ou não gerar um determinado erro; logo, pode-se escrever a Equação (2.28), ou seja, a função de transferência original como sendo a soma de sua função reduzida com um possível erro:

$$G(s) = \hat{G}(s) + e(s) \tag{2.28}$$

Substituindo as Equações (2.26) e (2.27) na Equação (2.28), resulta na Equação (2.29).

$$e(s) = \frac{a(s).\hat{b}(s) - \hat{a}(s).b(s)}{b(s).\hat{b}(s)} = \frac{N(s)}{D(s)} \tag{2.29}$$

Portanto o numerador da função erro, é dado pela Equação (2.30).

$$N(s) = a(s).\hat{b}(s) - \hat{a}(s).b(s) \tag{2.30}$$

Desta forma, a métrica proposta em Araújo (2008) para a redução de ordem de modelos consiste em minimizar a norma ao quadrado dos coeficientes do polinômio N(s), ou, compactamente, resolver a Equação (2.31).

$$\textit{min norm}^2 \textit{ \{ coef. [N(s)] \}} \tag{2.31}$$

Ressalta-se que a norma referente à Equação (2.31), nada mais é do que um método analítico para se calcular a distância entre dois pontos, Equação (2.32) e, considerando que um desses pontos se encontra na origem, tem-se a Equação (2.33).

$$norm = \sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2} \tag{2.32}$$

$$norm^2 = (x_1 - 0)^2 + (y_1 - 0)^2 \tag{2.33}$$

Como se quer o mínimo da norma ao quadrado dos coeficientes de N(s), implica afirmar que as derivadas (parciais) em relação aos coeficientes desta serão igualadas a zero, resultando no sistema de Equações (2.34) e (2.35).

$$\begin{cases} \dfrac{\partial\, norm^{2}\{coeficientes\ de\ \ N(s)\}}{\partial coef\ numerador\ de\ \hat{G}} = 0 & (2.34) \\ \dfrac{\partial\, norm^{2}\{coeficientes\ de\ \ N(s)\}}{\partial coef\ denominador\ de\ \hat{G}} = 0 & (2.35) \end{cases}$$

Para um completo entendimento, serão apresentados no Capítulo 4, os estudos de casos, utilizando esta técnica.

2.7- Conclusões

Observa-se, que o método apresentado na seção 2.6, como a grande maioria dos existentes, não fornece uma indicação sobre a ordem que se deve adotar para o modelo reduzido, sendo esta escolha por tentativa e erro; o método pode falhar quando aplicado a um sistema estável e de baixa ordem gerando um modelo reduzido instável; em algumas situações o modelo reduzido obtido pode ser de fase não-mínima apesar do modelo original ser de fase mínima.

Os IDM's são uma alternativa para o problema da determinação da ordem de modelos de ordem reduzida, particularmente quando se considera que certas características intrínsecas do sistema original devem ser mantidas, tais como: a estabilidade ou fase mínima, visto que em certas situações, como é o caso de projeto de controladores, é desejável que o modelo reduzido seja compatível em relação a estes dois itens com o sistema original.

O IDM e o método da seção 2.6, serão utilizados em exemplos de obtenção de modelos reduzidos no Capítulo 4, tendo como critério de avaliação a resposta ao degrau unitário e em frequência, aplicados aos sistemas originais e reduzidos.

CAPÍTULO 3
ALGORITMOS DE INTELIGÊNCIA DE ENXAMES

3.1- Algoritmos de Otimização

Não é exagero dizer que a otimização está em toda parte, do *design* de engenharia ao planejamento de negócios e do roteamento da Internet ao planejamento de férias. Em quase todas essas atividades, tenta-se atingir certos objetivos ou otimizar algo como lucro, qualidade e tempo. Como recursos, tempo e dinheiro são sempre limitados em aplicações do mundo real, busca-se encontrar soluções para otimizar o uso desses recursos valiosos sob planejamento e *design* usando ferramentas matemáticas (Yang, 2008).

Atualmente, a Inteligência Computacional (IC) tornou-se uma ferramenta indispensável para resolver problemas de otimização com vários algoritmos de busca eficientes. Assim, apresenta-se, neste capítulo, uma breve abordagem a respeito da IC e evolutiva, afim de aplicá-las aos métodos de redução da ordem de sistemas lineares que serão propostos, fazendo uma breve distinção entre o método determinístico clássico e o estocástico, utilizando inteligência de enxames.

3.2- Introdução à Inteligência Computacional

A IC é uma área recente das ciências teóricas e aplicadas, com enfoque em ciência da computação e matemática. É um conjunto de metodologias computacionais e abordagens inspiradas na natureza para lidar com problemas complexos do mundo real nos quais as abordagens tradicionais são ineficazes ou inviáveis.

A característica de inteligência é geralmente atribuída aos seres humanos, mas em IC qualquer forma de vida pode ser usada como inspiração, como formigas. Em IC a inteligência está diretamente ligada à tomada de decisão e raciocínio (Kruse, 2013).

De acordo com Zadeh (1994), a IC inclui principalmente redes neurais artificiais, além de outros algoritmos com inspiração biológica como a inteligência coletiva, o sistema imunológico, e inclui campos mais amplos, como processamento de imagem, além de outros formalismos como a Teoria de Dempster-Shafer, teoria do caos, lógica polivalente e lógica fuzzy, são utilizados na construção de modelos computacionais, visto que, os modelos tradicionais muitas vezes não conseguem lidar com incertezas, ruído, e a presença de um contexto em permanente alteração.

As Redes Neurais, introduzidas em 1940 (com mais desenvolvimentos na década de 1980) imitam o cérebro humano e representam um mecanismo computacional baseado em um modelo matemático simplificado dos neurônios e sinais que eles processam (Jamshidi, 2003).

A Computação evolucionária, introduzida na década de 1970 e mais popular desde os anos 1990 imita a evolução das espécies (com base populacional) através da reprodução das gerações. Ela também imita a genética nos chamados algoritmos genéticos (Liden, 2008).

A lógica fuzzy ou nebulosa ou difusa, foi introduzida em 1965 como uma ferramenta para formalizar e representar o processo de raciocínio e os sistemas de lógica difusa, que são baseados em lógica fuzzy e possuem muitas características atribuídas à inteligência. Com a incerteza, que é comum para o raciocínio humano, a percepção e a inferência e, ao contrário de alguns equívocos, tem uma espinha dorsal matemática muito formal e estrita (“é bastante determinista, em si, ainda que permitindo que as incertezas sejam efetivamente representadas e manipuladas”) Zadeh (1994).

Como forma de demonstrar o impacto da área, recentemente, a IC tem estudado, de forma profunda, a inteligência coletiva proporcionada por certas

comunidades de animais e insetos, isto é, inspiradas na natureza, com a capacidade de busca, comunicação e localização entre os indivíduos destas. A título de exemplo, tem-se, como apresentado em Wang *et al.* (2003), a utilização de soluções por otimização de colônias de formigas.

Sendo assim, como pré-requisito, ou base, para o estudo inicial destas comunidades, faz-se necessário, a abordagem de alguns conceitos fundamentais de heurísticas e metaheurísticas, como apresentado a seguir.

3.3- Heurísticas versus Metaheurísticas

Depois que um problema de otimização é formulado corretamente, a tarefa principal é encontrar as soluções ótimas por algum procedimento de solução usando as técnicas matemáticas corretas.

De fato, como se observa em quase todos os algoritmos metaheurísticos modernos, tenta-se usar as melhores soluções ou agentes, e randomizar (ou substituir) os não tão bons, enquanto avalia-se a competência de cada indivíduo (aptidão) em combinação com o histórico do sistema (uso de memória). Com esse equilíbrio, pretende-se projetar algoritmos de otimização melhores e mais eficientes (Yang, 2008).

Deste modo, a classificação do algoritmo de otimização pode ser realizada de várias maneiras. Uma maneira simples é observar a natureza do algoritmo e isso divide os algoritmos em duas categorias: algoritmos determinísticos e algoritmos estocásticos. Algoritmos determinísticos seguem um procedimento rigoroso, e seu caminho e valores de variáveis de *design* e funções são repetíveis. Por exemplo, *hill-climbing* é um algoritmo determinístico e, para o mesmo ponto de partida, eles seguirão o mesmo caminho, independentemente se o programa é executado hoje ou amanhã. Por outro lado, os algoritmos estocásticos sempre possuem alguma aleatoriedade (Goldbarg, 2017). Os algoritmos genéticos são um bom exemplo, as *strings* ou soluções na população serão diferentes cada vez que o programa é executado, pois os algoritmos usam alguns números pseudo-aleatórios. Embora os resultados finais possam não ter

grandes diferenças, mas os caminhos de cada indivíduo não são exatamente repetíveis.

Deste modo, de acordo com Gigerenzer (2011), as heurísticas são métodos de busca que tiram proveito de características e informações do próprio problema a ser explorado, facilitando o encontro de um mínimo global no espaço de busca. São limitadas e fornecem sempre a mesma solução quando iniciadas de um mesmo ponto de partida. Em contrapartida, as metaheurísticas vem suprir esta deficiência e tem como objetivo principal explorar um espaço de pesquisa de forma inteligente, ou seja, encontrar soluções de alta qualidade movendo-se para áreas não exploradas quando necessário.

Contudo, a área que estuda as metaheurísticas é considerada um subcampo primário da área de otimização estocástica, classe geral de algoritmos e técnicas que empregam algum grau de aleatoriedade para encontrar soluções tão ótimas quanto possível para problemas reconhecidamente difíceis.

Segundo a definição original, metaheurísticas são métodos de solução que coordenam procedimentos de busca locais com estratégias de mais alto nível, de modo a criar um processo capaz de escapar de mínimos locais e realizar uma busca robusta no espaço de soluções de um problema. Posteriormente, a definição passou a abranger quaisquer procedimentos que empregassem estratégias para escapar de ótimos locais em espaços de busca de soluções complexas (Geem *et al.*, 2001). Em especial, foram incorporados procedimentos que utilizam o conceito de vizinhança para estabelecer meios de fugir dos ótimos locais. Uma metaheurística, portanto, visa produzir um resultado satisfatório para um problema de otimização, focando na eficiência e uma maior exploração do espaço de busca, porém sem qualquer garantia de otimalidade.

Como apresentado em (Geem *et al.*, 2001), as metaheurísticas são aplicadas para encontrar respostas a problemas sobre os quais há poucas informações: não se sabe como é a aparência de uma solução ótima, há pouca informação heurística disponível e força-bruta é desconsiderada devido ao espaço de solução ser muito grande. Porém, dada uma solução candidata ao problema, esta pode ser testada e sua otimalidade, averiguada.

Algumas de suas características são:

- Utiliza estratégias para guiar o processo de busca;
- Explora de maneira eficiente o espaço de busca, com o objetivo de encontrar uma solução ótima próxima ao ponto atual;
- Utiliza de técnicas de buscas locais a complexos processos de aprendizagem;
- Possui mecanismos que evitam o aprisionamento dos mesmos em áreas restritas do espaço de busca;
- Faz uso de um domínio especifico de conhecimento com uma heurística para estratégias de busca;
- Armazena experiências de buscas, utilizando-as para guiar o algoritmo nos futuros processos de buscas.

Há diversos métodos de metaheurísticas. Alguns podem ser vistos como extensões de algoritmos de busca local que procuram sair de regiões com poucas possibilidades de encontrar ótimas soluções e ir para locais onde as melhores soluções podem estar presentes. Isso é proposto por algoritmos como Busca Tabu, Busca Local Iterativa (ILS, do inglês *Iterated Local Search*), Busca em Estrutura de Vizinhança Variável (VNS, do inglês *Variable Neighborhood Search*), Recozimento Simulado (*Simulated Annealing*) e GRASP (do inglês *Greedy Randomized Adaptive Search Procedures*, em tradução livre, “Procedimentos ágeis de Pesquisa Adaptativa Aleatória”).

Há também outras técnicas inspiradas na capacidade da natureza de adaptação dos seres vivos ao meio onde vivem, através da recombinação e mutação de indivíduos, mais precisamente, recombinar soluções atuais (soluções pai) para melhorar futuras soluções (soluções filho). Nessa classe estão algoritmos de computação evolutiva, como os Algoritmos Genéticos. Outros algoritmos são inspirados no comportamento de indivíduos em interação com o meio onde habitam, como Colônia de Formigas e Nuvem de Partículas (Kennedy & Eberhart, 1995). A Colônia de Formigas é uma abordagem inspirada

no comportamento das formigas para encontrar o menor caminho entre a origem do alimento e seu ninho, onde as mesmas depositam feromônio para marcar a trajetória. A Nuvem de Partículas é motivada pela simulação do comportamento social de organismos existentes na natureza.

Em Geem *et al.* (2001), para que uma metaheurística explore um espaço de busca de forma inteligente, obtenha soluções de ótima qualidade e consiga mover-se para áreas não exploradas quando necessário, os conceitos de intensificação e diversificação devem ser atingidos. A intensificação consiste em concentrar as buscas em regiões promissoras em torno de boas soluções. A diversificação corresponde em fazer buscas em regiões ainda não exploradas. Toda metaheurística deve possuir esses dois componentes que devem ser balanceados e bem utilizados, tornando necessário integrar metaheurísticas por uma hibridização, que pode levar a melhores desempenhos do que uma metaheurística sozinha.

3.4- Metaheurística inspirada na natureza

De acordo com Zhu e Tang (2010), existe uma variedade de organismos na natureza que possuem a habilidade de buscar alimento de maneira cooperativa enquanto tentam evitar predadores e outros riscos, o que é chamado de “comportamento de enxame”. Esse comportamento é encontrado em pássaros, peixes, formigas, abelhas, cupins e outros tipos de insetos. A vida em sociedade oferece mais chances de sobrevivência a esses organismos do que se eles vivessem de forma isolada.

Segundo Zhu e Tang (2010) e Rosendo (2010) esse tipo de comportamento geralmente não segue comandos de um líder e não possui um sistema hierárquico, mas mesmo não havendo um controle centralizado ou um plano global, cada organismo do enxame segue regras locais de interação para comandar suas ações, o que pode gerar um padrão global de comportamento, dessa maneira o enxame acaba atingindo seus objetivos. Deve-se ressaltar, contudo, que os agentes individuais não têm conhecimento explícito de

resolução de problemas, sendo que o comportamento inteligente surge (ou emerge) por causa das ações sociais dos agentes (White & Pagurek,1998).

Os indivíduos do enxame interagem entre si e com o ambiente para alcançar um determinado objetivo. Por exemplo, na busca por alimento, os indivíduos constantemente trocam informações, para saber que direção seguir, utilizando sua experiência individual e também a experiência de seus companheiros, com base na melhor posição que já ocuparam em relação ao alimento que buscam.

De acordo com Zuben e Attux (2008) esse tipo de comportamento social inspira pesquisadores a desenvolver diversas ferramentas computacionais para a resolução de problemas e estratégias de coordenação e controle de robôs. Assim surgiu a Inteligência de Enxame (SI, do inglês *Swarm Intelligence*) no fim da década de 1980, para se referir a sistemas robóticos formados por uma coleção de agentes simples que interagem em um ambiente seguindo regras locais (Rosendo, 2010).

Ainda segundo Rosendo (2010), a Inteligência de Enxame é uma técnica de inteligência computacional que estuda o comportamento coletivo de agentes descentralizados. E com base nestes sistemas naturais de comportamento emergente, com o passar do tempo diversos sistemas artificiais de otimização têm sido desenvolvidos e aprimorados. Segundo Pereira (2007), tais algoritmos possuem características que os tornam mais eficazes do que outros algoritmos na pesquisa de solução ótima, das quais se destacam:

- A capacidade de trabalhar com uma população de soluções simultaneamente, introduzindo assim uma perspectiva global e uma maior diversidade de pesquisa. Tal característica proporciona uma grande capacidade de encontrar ótimos globais em problemas que possuem diversos ótimos locais.

- Os algoritmos de enxame trabalham com soluções aleatórias e operadores probabilísticos, o que possibilita uma maior capacidade de fuga de ótimos locais, além de também manter uma maior diversidade da população.

- Quanto ao domínio da pesquisa, não é necessário um conhecimento prévio, podendo este ser multidimensional, com ou sem restrições, lineares ou não lineares.

Alguns dos algoritmos com inteligência de enxame utilizados são o de otimização por grupo de partículas (PSO, do inglês *Particle Swarm Optimization*) introduzido por Kennedy e Ebehart (1995) e o *Firefly Algorithm* (FA) proposto por Xin-She Yang (2008), baseados no movimento de grupos de animais, tais como bandos de aves e cardumes de peixe. As observações de como esses animais se movimentam em conjunto de forma sincronizada e harmoniosa, levou a tentativa de criação de modelos computacionais que descrevessem o princípio por trás dessa comunicação.

3.5- Algoritmo de Enxame *Firefly*

O *Firefly Algorithm* (FA) é um algoritmo de otimização bioinspirado, baseado no comportamento social de vagalumes na natureza, insetos bioluminescentes conhecidos por suas emissões luminosas, e proposto originalmente por Xin-She Yang (Yang, 2008).

Os padrões de luminosidade podem ser usados tanto para atrair suas presas, se comunicar com outros vagalumes, emitir alertas sobre a presença de predadores ou para atrair possíveis parceiros reprodutivos em rituais de acasalamento, sendo essa atração mais forte quanto maior for a intensidade da luz. Esta capacidade de comunicação diminui à medida que a distância entre os vagalumes aumenta (Xing & Gao, 2014).

O FA simula o comportamento social entre os indivíduos de uma população de vagalumes no verão em regiões tropicais. Nesta situação, cada vagalume desloca-se no hiperespaço produzindo seu próprio padrão de luminosidade atraindo ou sendo atraído pelo padrão de outros vagalumes que considere mais atrativo. A ideia é que os vagalumes convirjam para aquele com maior brilho na população.

No FA os comportamentos dos vagalumes, tais como sua atratividade e comunicação, são usados para realizar uma pesquisa no espaço de busca pelas melhores soluções para o problema a ser otimizado. Neste algoritmo, cada possível solução é conhecida como “vagalume” e seu brilho está associado com o seu valor da função objetivo. Os vagalumes são atraídos pelos vizinhos que possuem o brilho mais intenso, ou seja, aqueles com os melhores valores da função objetivo. Quando não existem vizinhos mais “brilhantes” do que o vagalume em questão, este irá se movimentar de forma aleatória pelo espaço de busca (Xing & Gao, 2014).

A ideia do algoritmo é calcular o valor da função objetivo em diversos pontos do domínio, escolhidos inicialmente de forma aleatória, considerando que em cada um destes pontos existia um vagalume e fazer com que esse valor da função nestes pontos esteja relacionado com a intensidade da luz gerada pelos vagalumes. Em seguida são feitas interações, seguindo certas regras, com o objetivo de fazer com que os valores convirjam para o ponto que gere mais brilho, ou seja, no ponto onde a função apresente o valor ótimo (Ribeiro, 2014).

A atratividade está ligada à intensidade da luz (I) enxergada pelo vagalume, que diminui com o aumento da distância (r), de maneira que a intensidade de luz é inversamente proporcional ao quadrado da distância, $I \propto r^{-2}$, portanto os pontos que apresentam os menores valores para a função objetivo vão sendo atraídos em direção aos que apresentam os maiores valores.

De acordo com Yang (2008) o algoritmo deve seguir algumas regras:

- Cada vagalume é atraído por outros vagalumes independentemente do sexo.

- A atratividade é proporcional ao brilho. O vagalume de menor brilho sempre irá se mover em direção ao de maior brilho.

- A atratividade é proporcional à intensidade da luz, que é inversamente proporcional à distância, pois o algoritmo tem o objetivo de encontrar o ponto que a intensidade da luz seja máxima.

- Quando não houver nenhum outro com brilho maior que o seu, o vagalume se moverá de forma aleatória.

- O brilho emitido por um vagalume é determinado pela sua avaliação frente à função objetivo, por consequência, quanto melhor avaliado, mais brilhante.

No FA assume-se a existência de uma população de vagalumes e que a mesma é usada para solucionar o problema, fazendo com que os indivíduos se movimentem de forma interativa pelo espaço de busca.

A posição, x_i, de cada vagalume utilizado no algoritmo define uma solução candidata X_i, Equação (3.1).

$$X_i = (x_1, x_2, \dots, x_d) \tag{3.1}$$

A distância euclidiana entre dois vagalumes (*i* e *j*) é dada pela Equação (3.2), onde n é o número de dimensões que o vagalume pode se deslocar.

$$r_{ij} = \|x_i - x_j\| = \sqrt{\sum_{k=1}^{n} \left(x_{i,k} - x_{j,k}\right)^2} \tag{3.2}$$

No algoritmo, dois conceitos distintos são importantes: a intensidade da luz (ou brilho), que é um parâmetro individual de cada vagalume, e a atratividade, que depende da distância que o vagalume está sendo observado e indica o quão forte ele irá atrair outros vagalumes do enxame.

A intensidade da luz relativa, I, observada por um vagalume i, a uma distância, r_{ij}, de outro vagalume j, é calculada usando a Equação (3.3) onde γ é o coeficiente de absorção da luz no meio que varia de 0 a 1 e I_0 é a intensidade original da luz em *r* = 0.

$$I = I_0 e^{-\gamma r_{ij}} \tag{3.3}$$

Assume-se que a atratividade, ou função de atração, β, de um vagalume, é proporcional à intensidade de seu brilho enxergada pelos vagalumes

adjacentes E inversamente proporcional à distância entre dois vagalumes, conforme a Equação (3.4), onde β_0 é atratividade em r = 0.

$$\beta = \beta_0 e^{-\gamma r_{ij}^2} \tag{3.4}$$

Existem dois casos específicos importantes:

a. Para $\gamma \to 0$, a atratividade será sempre constante, o que seria equivalente a vagalumes espalhados num espaço ideal, onde todos podem ser observados, de qualquer distância e, portanto, sejam sempre atraídos em direção ao que apresenta a maior intensidade de luz.
b. Para $\gamma \to \infty$, tem-se uma situação completamente oposta: nenhum vagalume pode ser observado por outro, fazendo com que eles se movam de forma completamente aleatória. Este caso corresponde a um método de busca aleatória.

O movimento de um vagalume i em direção a um vagalume j, causado por esta atração, pode ser calculado usando a Equação (3.5), onde $X_i(t)$ é a posição corrente do vagalume i, $X_j(t)$ é a posição do vagalume j, α é um coeficiente aleatório e ε_i é um vetor aleatório com distribuição gaussiana.

$$X_i(t+1) = X_i(t) + \beta_0 e^{-\gamma r_{ij}^2}\left(X_j - X_i\right) + \alpha\varepsilon_i \tag{3.5}$$

Considerando a posição inicial de cada componente no enxame, a Equação (3.5) pode ser escrita como em (3.6).

$$x_i = x_{i0}(t) + \beta_0 e^{-\gamma r_{ij}^2}\left(x_j(t) - x_{i0}(t)\right) + \alpha\left(rand - \frac{1}{2}\right) \tag{3.6}$$

Onde:

x_{i0} é a posição inicial do vagalume i ;

$\beta_0 e^{-\gamma r_{ij}^2}$ é a parcela do movimento devido à atratividade gerada pelo vagalume j;

$\alpha\left(rand - \frac{1}{2}\right)$ é a parcela aleatória do movimento, com *rand* sendo um número aleatório entre 0 e 1.

Os passos para implementar o FA podem ser sintetizados como no pseudocódigo adaptado de Yang (2008) em Serapião e Rocha (2012) apresentado a seguir:

1. Definir a função objetivo $J(x)$, $x = (x_i, \dots, x_d)^T$.
2. Definir os parâmetros $n, \alpha, \beta_0, \gamma, MaxGerações$.
3. Gerar a população inicial de vagalumes x_i $(i = 1, 2, \dots, n)$.
4. Calcular a intensidade de luz I, para x_i proporcionalmente a $J(x_i)$, para cada vagalume x_i:
5. Calcular o fator de atratividade β de acordo com $e^{-\gamma^2}$.
6. Mover o vagalume x_i em direção aos vagalumes mais brilhantes.
7. Se o critério de convergência for satisfeito, termine, senão, volte ao passo 4.

A Figura 3.1 ilustra o funcionamento do algoritmo na forma de fluxograma.

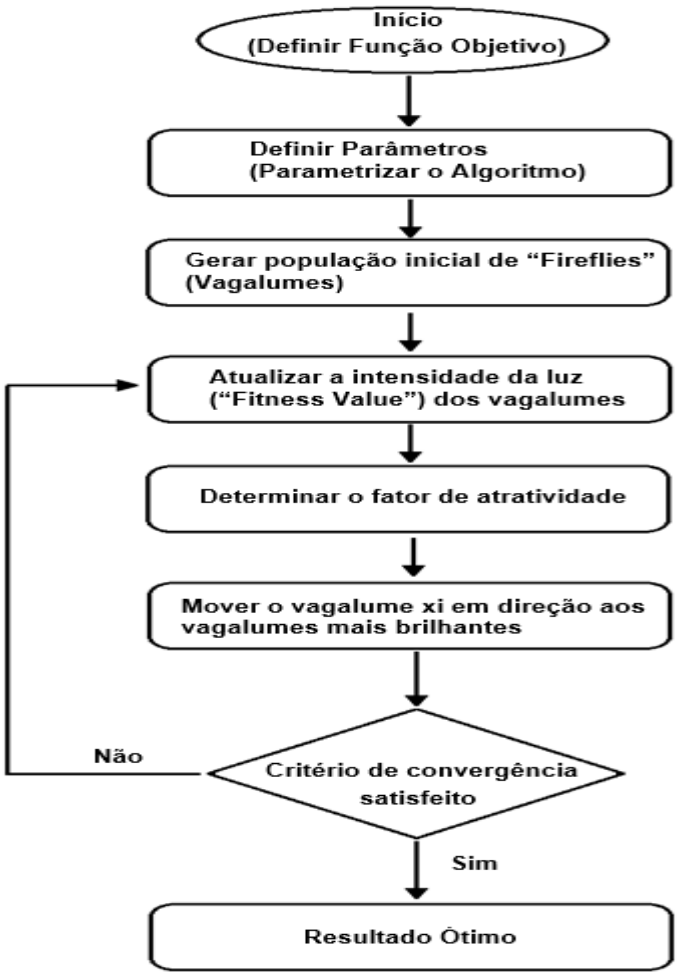

Figura 3.1 - Fluxograma de soluções de problemas de otimização com *Firefly Algorithm*.

3.6- *Particle Swarm Optimization* - PSO

A metaheurística do enxame de partículas PSO (*Particle Swarm Optimization*) é uma técnica de otimização estocástica, inicialmente voltada para funções contínuas, seu desenvolvimento foi inspirado na simulação de um sistema social simplificado (Kennedy & Eberhart, 1995). Ele procura reproduzir o deslocamento de um conjunto de indivíduos (bando de pássaros, cardume de peixes ou enxame de insetos) que pode ser caracterizado por um comportamento individualmente aleatório, mas globalmente direcionado. Foi concebida a partir de estudos relacionados ao comportamento destas espécies dentro de uma sociedade tendo como base os trabalhos de observação comportamental do biólogo Frank Heppner (Souza, 2014).

Embora seja classificado, por alguns autores, como um algoritmo do tipo evolutivo, por possuir similaridades com outros algoritmos desta classe, como os Algoritmos Genéticos (AG's) por exemplo, pois seu processo de otimização baseia-se, também, na manipulação de uma população de soluções que é inicializada de forma aleatória consistindo na busca (da posição) com maior aptidão (*fitness*), em um espaço de busca predeterminado, correspondente ao conjunto de todas as possíveis soluções, cujos problemas são resolvidos iterativamente ao tentar-se melhorar a solução candidata com respeito a uma dada medida de qualidade e critérios de parada (Silva, 2014); para outros autores, este algoritmo não pode ser classificado ou inserido na computação evolutiva por não possuir os mesmos operadores de seleção, recombinação e mutação, por exemplo. Por outro lado, assemelha-se ao método da colônia de formigas (ACO do inglês, *Ant Colony Optimization*) e aproxima-se desta quanto ao quesito enxames. De fato, de acordo com Engelbrecht (2007), trata-se de um algoritmo pertencente ao grupo dos algoritmos bioinspirados e pode-se, então, finalmente classifica-lo e inseri-lo na categoria de algoritmos baseados em inteligência de enxames ou populações. Adiante, a Figura 3.2 ilustra a situação.

No PSO, assim como em outros algoritmos, existe uma população de indivíduos, chamados de nuvem (ou enxame) de partículas, que ao invés de utilizar operadores genéticos, evoluem através da cooperação e competição entre si por diversas gerações (isto é, as partículas ou indivíduos não são

descartados após cada iteração). Os componentes do grupo favorecem-se de suas experiências individuais, ou seja, armazenam informação de suas melhores posições já visitadas, como também se beneficiam da experiência coletiva, isto é, de outros membros do grupo, como a experiência de seus vizinhos, por exemplo, durante a busca de uma melhor localização (*fitness*) em relação ao seu alvo, como o objetivo por busca de comida, local para pouso, proteção de predadores e outros. A Figura 3.3 ilustra o caso.

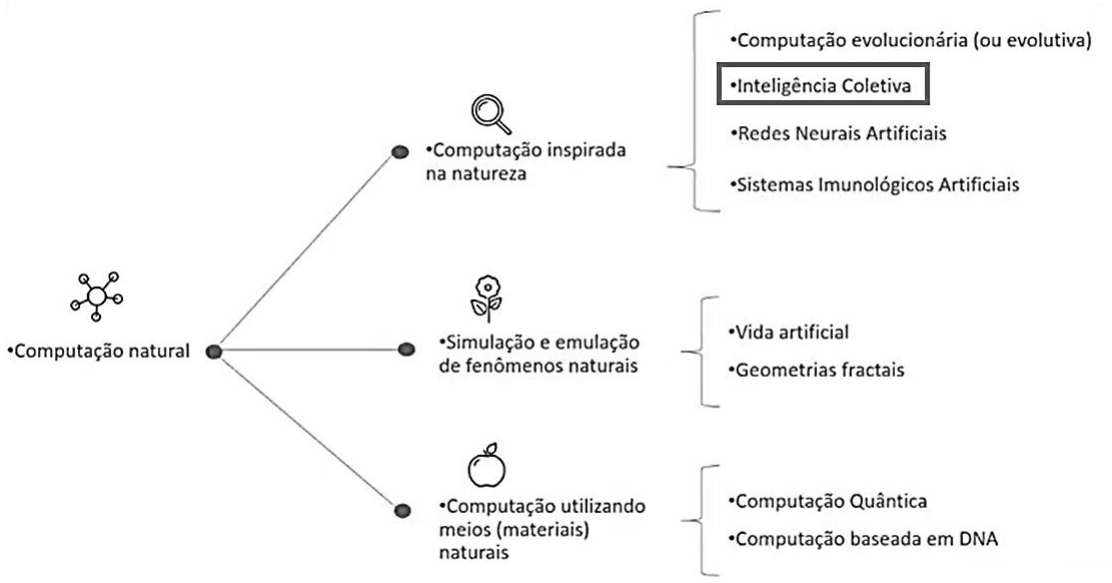

Figura 3.2 – Classificação do algoritmo PSO quanto a sua natureza.

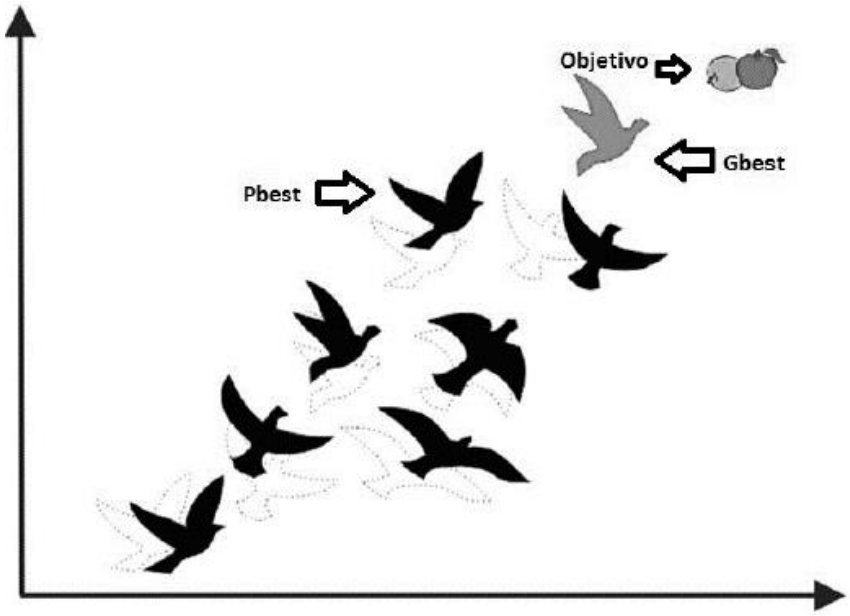

Figura 3.3 – Ilustração da nuvem de partículas (bando de pássaros) em busca de comida, guiados por um "líder" o qual possui a melhor posição (*Gbest*) do bando em relação ao objetivo. Fonte: Garcia (2016).

Observada a Figura 3.3, pode-se entender o funcionamento do algoritmo PSO por meio de uma analogia de um bando de pássaros em pleno voo na busca por alimento ou abrigo, em que é possível notar que há um pássaro que se destaca em relação aos demais, ou seja, este pássaro possui a melhor posição do grupo de tal forma que os outros pássaros desta população tendem a segui-lo. Devido ao fato de todos os indivíduos deste grupo estarem em movimento (em voo), há sempre a atualização das posições de cada componente deste.

Comparativamente, no PSO, este pássaro que se evidencia, sendo o líder do grupo e guiando todos os outros, trata-se da partícula de melhor posição dentro do espaço de busca considerado, denominada na literatura por *Gbest*. Da mesma forma, as melhores posições dos demais pássaros, já visitadas até o momento, tratam-se das melhores posições das partículas, chamadas *Pbest*. Por fim, o alimento ou abrigo, equivale a função objetivo do algoritmo.

Em outras palavras, a metodologia tende a preservar as posições com melhor aptidão e descartar as outras. Para tanto, o movimento de cada partícula é afetado tanto pela informação sobre sua melhor posição quanto pelo registro da melhor posição obtida considerando todas as outras partículas. A posição e a velocidade iniciais de cada partícula são definidas aleatoriamente. A cada nova iteração esses valores são atualizados. Esse processo é repetido até ser atingido o número máximo de iterações, ou que seja atingido o erro máximo pré-determinado. A Figura 3.4 ilustra o fluxograma do PSO.

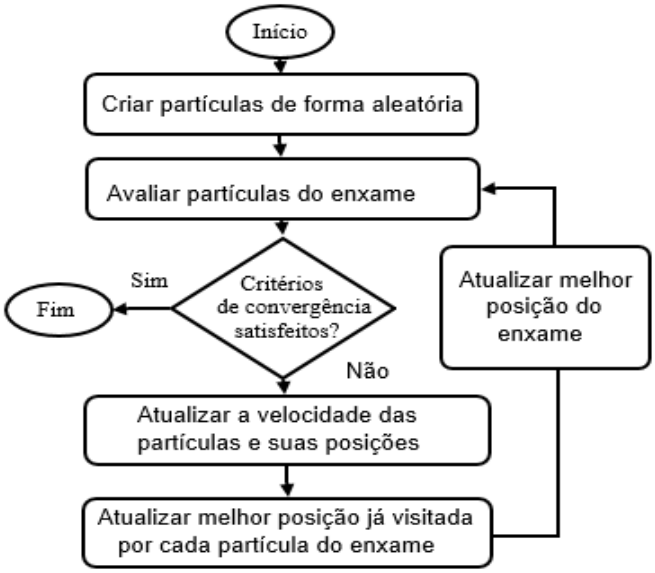

Figura 3.4 - Fluxograma de soluções de problemas de otimização com PSO.

3.6.1- Estrutura do PSO Clássico

De acordo com o modelo clássico proposto por Kennedy e Eberhart (1995), as partículas circulam pelo espaço de busca, tendo suas velocidades atualizadas de forma dinâmica com base no histórico das experiências individuais e coletiva de todo o enxame. Logo, a evolução do algoritmo PSO está associada à trajetória percorrida pelo enxame e ao tempo gasto para encontrar a melhor solução do problema. As estruturas de armazenamento dos valores, isto é, os vetores posição (*X*) e velocidade (*V*) da i-ésima partícula, no instante t, no espaço de busca D-dimensional podem ser representados como visto nas Equações (3.7) e (3.8), respectivamente.

$$X_i(t) = [x_{i1}(t),\ x_{i2}(t),\ x_{i3}(t), \dots ,\ x_{iD}(t)] \tag{3.7}$$

$$V_i(t) = [v_{i1}(t),\ v_{i2}(t),\ v_{i3}(t), \dots ,\ v_{iD}(t)] \tag{3.8}$$

Da mesma forma, sendo cada uma destas partículas uma solução potencial para o problema, a sua melhor posição (individual ou prévia), isto é, a posição que fornece o melhor valor de aptidão da i-ésima partícula (*Pbest*), é representada por:

$$Pbest_i(t) = [pbest_{i1}(t),\ pbest_{i2}(t),\ pbest_{i3}(t), \dots ,\ pbest_{iD}(t)] \tag{3.9}$$

Coletivamente, a posição da partícula mais apta, ou seja, da partícula com o melhor desempenho segundo a função objetivo encontrada até então no instante *t*, ou ainda, a melhor posição encontrada pelo enxame (*Gbest*) é dada por:

$$Gbest(t) = [gbest_1(t),\ gbest_2(t),\ gbest_3(t), \dots ,\ gbest_D(t)] \tag{3.10}$$

Por fim, de posse dos valores de *Pbest* e *Gbest,* pode-se obter a atualização e iteração do algoritmo por meio das Equações (3.11) e (3.12).

$$V_i(t+1) = V_i(t) + \ c_1.r_1\ [Pbest_i - X_i(t)] + c_2.r_2\ [Gbest - X_i(t)] \tag{3.11}$$

$$X_i(t+1) = X_i(t) + V_i(t+1) \tag{3.12}$$

Onde:

c_1 e c_2 - são duas constantes positivas que correspondem as componentes cognitivas e sociais do enxame, respectivamente, (também chamados de taxas de aprendizado);

r_1 e r_2 - são duas funções randômicas definidas no intervalo [0,1]; as quais diminuem a possibilidade das partículas ficarem presas em um ótimo local, uma vez que proporcionam uma natureza estocástica ao algoritmo.

Para o produto de $c_1.r_1$ maior que o produto $c_2.r_2$, as partículas sofrerão maior influência de seu fator cognitivo individual, *Pbest*, isto é, a iteração entre as partículas é menor, o que dificulta a convergência do algoritmo. Por outro lado, para um produto $c_2.r_2$ maior que $c_1.r_1$, cada partícula está sujeita a uma influência maior do fator social (de aprendizagem), *Gbest*, e provavelmente ficará presa em ótimos locais (Esmin, 2005).

Kennedy e Eberhart (1995), propuseram que c_1 e c_2 devem ser iguais e que $c_1 = c_2 = 2$, do contrário, caso a soma destes coeficientes seja maior que 4, a primeira parcela da Equação (3.11), isto é, $V_i(t)$, pouco influenciaria na convergência do algoritmo para um ótimo global, dispersando, desta forma, as partículas pelo espaço de busca.

Shi e Eberhart (1998), acrescentaram ainda, à velocidade inicial das partículas, o fator de inércia (ω), empregado para controlar o impacto da velocidade anterior na velocidade atual, influenciando assim as habilidades de exploração global e local das partículas, conforme mostra a Equação (3.13).

$$V_i(t+1) = \omega . V_i(t) + c_1 . r_1 \left[Pbest_i - X_i(t)\right] + c_2 . r_2 \left[Gbest - X_i(t)\right] \qquad (3.13)$$

Por fim, os valores de posição e velocidade devem estar inseridos dentro dos limites mínimo e máximo de cada variável presente na partícula. Esses limites determinam o espaço de busca do enxame, evitando que as partículas gerem resultados inválidos (Aloise, 2005).

A implementação deste algoritmo encontra-se no Anexo IV deste trabalho.

3.7- *Shuffled Frog Leaping Algorithm* - SFLA

O “*Shuffled Frog Leaping Algorithm*” é um método de busca randômica, pertencente a categoria de Inteligência de Enxames, assim como descrito anteriormente para o PSO (Ver Figura 3.2). Propõem-se a resolver problemas de otimização multi-objetivo, abordados em primeira instância por Eusuff e Lansey (2003).

O SFLA, segundo Eusuff e Lansey (2003), é uma metaheurística desenvolvida para resolver, em suma, problemas de otimização combinatória. O algoritmo contém elementos de busca local e troca de informações globais, consistindo em um conjunto de populações virtuais de rãs interativas distribuídas em diferentes agrupamentos denominados “memeplex”, onde cada “meme” representa uma unidade de evolução cultural.

Para garantir uma exploração global, as rãs virtuais são periodicamente misturadas, isto é, são trocadas as suas posições originais e reorganizadas em novos agrupamentos em uma técnica similar àquela usada no algoritmo de evolução complexa aleatória (Duan, 1992). Além disso, para fornecer a oportunidade de geração randômica de informações aprimoradas, rãs aleatórias são geradas e substituídas na população (Xingyu, 2019).

O algoritmo foi inspirado na evolução memética de um grupo de sapos a procura por comida. Neste método, uma solução para um determinado problema é apresentada na forma de uma “*string*”, inerente aos sapos da população considerada (Bouazza, 2017).

Ressalta-se que um algoritmo memético é aquele que reflete as ações de determinadas características de um grupo social de indivíduos por meio da imitação (e, possivelmente, posterior melhoria na técnica, método ou ação em si). O termo “algoritimo memético” foi cunhado por Moscato (1989), derivado de “meme” (Dawkins, 1976), este por sua vez, infere sobre um padrão de informações que se propagam de forma “contagiosa” ou, como dito, por “imitação”, de modo a alterar e influenciar o comportamento social individual dos membros de um determinado grupo.

Dawkins (1976) ratifica, ainda, que todo conhecimento é memético e que uma ideia ou um padrão de informações não se caracteriza um “meme” até que haja um outro indivíduo que possa ser capaz de levar adiante, isto é, replicar, o conhecimento adquirido de forma consecutiva. A título de exemplo tem-se a música, ideias, frases de efeitos, moda, dentre outros. Para tal, define “meme” como, simplesmente, uma unidade de informação intelectual ou cultural que sobrevive ao tempo e que se pode passar de “mente em mente”, ou seja, de geração em geração.

De acordo com Eusuff e Lansey (2003), o conteúdo real de um meme é análogo ao conteúdo do cromossomo de um gene nos Algoritmos Genéticos. A evolução memética e genética estão sujeitas aos mesmos princípios básicos, isto é, possíveis soluções são criadas, selecionadas de acordo com algum critério de “*Fitness”*, combinada com outras soluções e possivelmente modificada (mutada).

A evolução memética, no entanto, é um mecanismo muito mais flexível. Enquanto os genes só podem ser transmitidos pelos pais (ou pai nos casos de reprodução assexuada) à prole; os memes podem, em princípio, serem transmitidos entre dois indivíduos. Outro ponto a saber: os genes são transmitidos entre gerações ao longo do tempo, podendo levar vários anos para se propagarem. Os memes, por sua vez, podem ser transmitidos no espaço de minutos (Eusuff; Lansey; Pasha, 2006).

A replicação genética é restrita pelo número bastante pequeno de filhos de um único pai, enquanto o número de indivíduos que podem assumir um meme de um único indivíduo é quase ilimitado. Além disso, parece muito mais fácil para os memes sofrerem variações, já que os indivíduos podem entrar em contato com muitas fontes diferentes de novos memes. Portanto, a disseminação de memes é muito mais rápida que a disseminação de genes. A outra diferença entre "memes" e "genes" é que os primeiros são processados e possivelmente melhorados pelo próprio indivíduo que os detém, algo que não acontece com os genes (Eusuff; Lansey; Pasha, 2006).

3.7.1- Estrutura do Algoritmo SFLA

De acordo com o explicitado, a população inicial de sapos, após ser gerada de forma aleatória, é particionada em grupos ou subconjuntos (memeplexes) onde o número de sapos, em cada subconjunto desta divisão, deve ser igual.

Feito isso, o SFLA baseia-se em duas técnicas de busca: busca local e técnicas globais de troca de informações (Tavakolan, 2011). Com base na pesquisa local, os sapos de cada subconjunto melhoram suas posições para obter mais alimentos (para alcançar a melhor solução). Na segunda técnica, as informações obtidas entre subconjuntos são comparadas entre si (após cada pesquisa local em subconjuntos). As etapas são:

1. Gerar uma população de sapos P, aleatoriamente, de acordo com a Equação (3.14)

$$P = [U(1), U(2), \dots., U(P)] \tag{3.14}$$

Em que um sapo é representado como um vetor de valores (memótipo) de variáveis de decisão (d), dentro de um espaço de busca d-dimensional, como mostra a Equação (3.15).

$$U(i) = [u_i^1, u_i^2, u_i^3, \dots, u_i^d] \tag{3.15}$$

Para i = 1 até P.

2. Calcular o valor de desempenho, *Fitness*, F(i), para cada sapo U(i); classifica-los em ordem decrescente de desempenho e armazena-los em um vetor, X, de posições.

$$X = [U(i), F(i), i = 1, \dots. U(P), F(P), i = P] \tag{3.16}$$

Onde a melhor posição é pertencente ao sapo U(i=1), por consequência, a posição menos favorecida é referente ao último indivíduo, U(P).

3. Distribuir os indivíduos do vetor X (população organizada) em m memeplex. Cada memeplex contêm n rãs, de tal forma que:

$$Y^k = [U(j)^k, F(j)^k | U(j)^k = U\big(k + m(j-1)\big), F(j)^k = F\big(k + m(j-1)\big)] \quad (3.17)$$

Para: $j = 1, \dots, n$ e $k = 1, \dots, m$

Isto é, a primeira rã vai para o primeiro memeplex, a segunda ao segundo memeplex, a rã “m” vai para o memeplex “m” e a rã “m+1” vai para o primeiro memeplex e assim por diante, conforme ilustra a Figura 3.5.

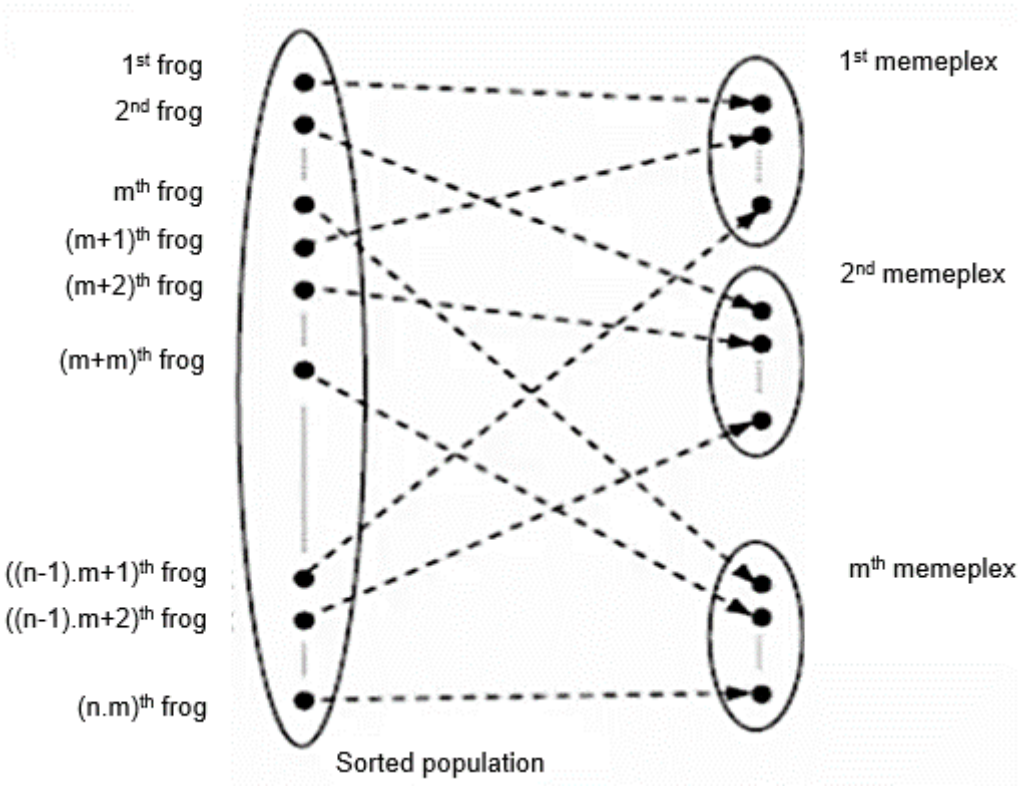

Figura 3.5 - Particionamento de memeplexes - Adaptado de Xingyu (2019).

Observa-se que a população inteira (P) de rãs é dada pelo produto de m memeplexes, por n rãs (número de rãs por subconjunto), de tal forma que:

$$P = m \times n \quad (3.18)$$

O algoritmo é iterativo de acordo com o fluxograma ilustrado na Figura 3.6.

Figura 3.6 - Fluxograma de soluções de problemas de otimização com SFLA.

O próximo passo é baseado na pesquisa local. Dentro de cada memeplex se identifica a melhor rã (*Fitness*), Xb, e a pior, Xw. Além disso, o sapo que contém a melhor aptidão global em toda a população será identificado como Xg. Em seguida, a posição da pior rã é ajustada de acordo com as Equações (3.19) e (3.20).

$$D = rand \times (X_b - X_g) \tag{3.19}$$

$$X_w^{novo} = X_w^{atual} + D \tag{3.20}$$

Onde -Dmax ≤ D ≤ Dmax, rand é um número aleatório no intervalo de [0 1] e D é o tamanho do vetor do salto (mudança de posição). Dmax é a alteração máxima permitida na posição de um sapo.

Se a nova posição do sapo não melhorar, as Equações (3.19) e (3.20) são repetidas em relação ao melhor sapo global (ou seja, Xg substitui Xb). Se nenhuma melhoria ocorrer neste último caso, um sapo aleatório será gerado para substituir a posição antiga do sapo anterior. O processo de embaralhamento continua até que os critérios de convergência sejam atendidos. O desenvolvimento deste algoritmo encontra-se no anexo V.

3.8- Estrutura e Organização dos Algoritmos para Redução de Ordem

Para utilizar os algoritmos apresentados nas seções anteriores na redução da ordem de sistemas dinâmicos, os indivíduos do “enxame” devem representar os coeficientes dos polinômios para um possível modelo reduzido. Inicialmente, as posições dos destes indivíduos são geradas de forma aleatória e avaliadas para que se encontrem as melhores posições a cada interação. Cada solução representada pelos indivíduos da população é avaliada para que o algoritmo possa encaminhar o grupo para as regiões com os membros mais aptos (ou adaptados, de acordo com a função *Fitness*). A avaliação é feita com base no erro entre a resposta do sistema original e a resposta do sistema reduzido proposto. Os passos do algoritmo são:

1- Definir a estrutura do modelo reduzido (ordem).

2- Inicializar aleatoriamente os membros da população.

3- Calcular o erro entre as respostas do modelo original $\mathrm{y(t)}$ e a do modelo reduzido $\mathrm{y_n(t)}$, representado pelo vagalume/partícula/rã n a uma entrada do tipo degrau, Equação (3.21), e em seguida calcular a função custo ou erro médio quadrático, Equação (3.22), onde m é o tamanho do vetor de erro $e_n(t)$.

$$e_n(t) = y_n(t) - y(t) \tag{3.21}$$

$$F_n = \frac{\sum_{t=1}^{m}[e_n(t)]^2}{m} \tag{3.22}$$

4- Efetuar a iteração do algoritmo, a fim de melhorar a posição de todos os indivíduos do enxame.

5- Verificar se algum componente atende ao critério de parada, isto é, ao número máximo de iterações ou ao valor de erro máximo admitido entre a resposta do modelo original e a do modelo reduzido do melhor indivíduo: caso sim, retornar a melhor solução encontrada; caso não, voltar ao passo 3, como mostra o fluxograma da Figura 3.7.

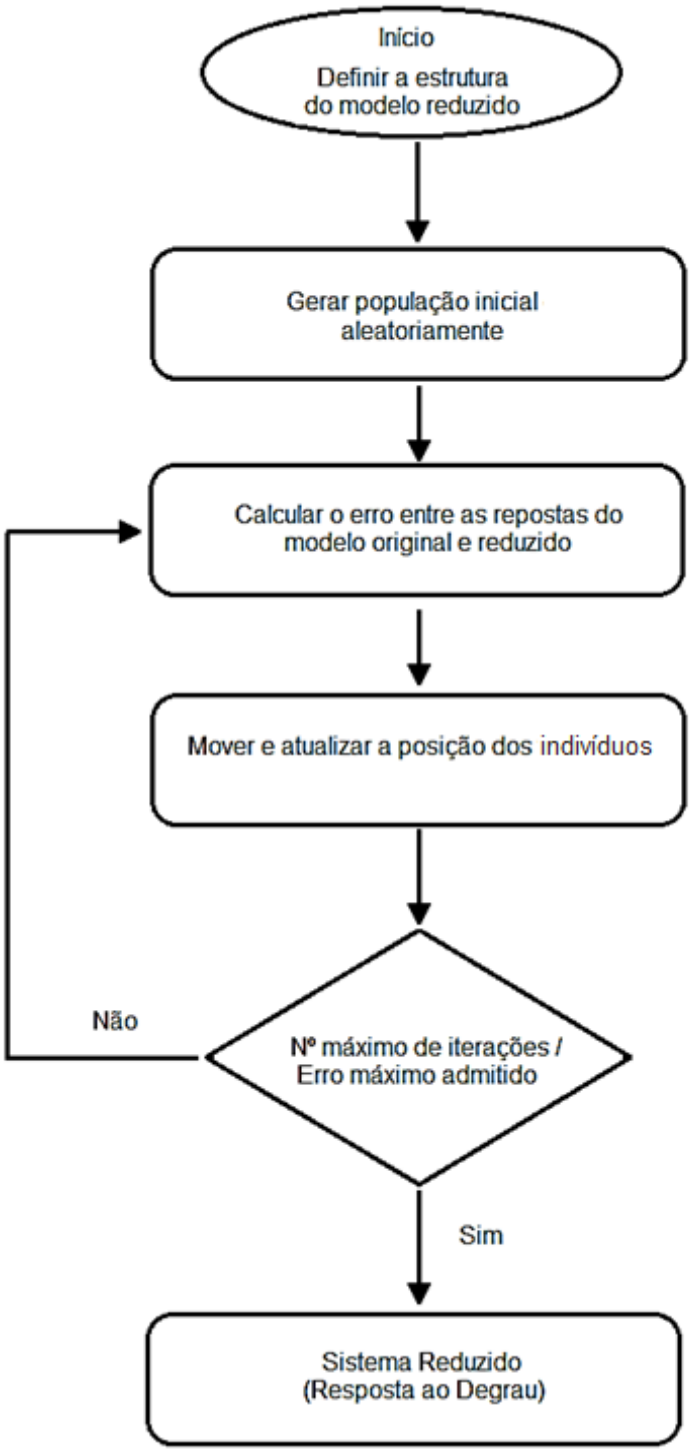

Figura 3.7 - Fluxograma para solução e determinação de modelos reduzidos utilizando os Algoritmos de Enxames abordados.

3.9- Conclusões

Neste capítulo abordou-se a diferença fundamental entre os métodos determinísticos e estocásticos de busca; confrontando a metodologia clássica de otimização e minimização para problemas que possuem uma dada função objetivo. Evidenciou-se, desta forma, o estudo dos algoritmos bioinspirados utilizados para resolução de tais problemáticas.

Foram abordados os conceitos básicos sobre os algoritmos de Inteligência de Enxame como o FA, PSO e SFLA, além de apresentar a metodologia necessária para utilização dos mesmos.

Por fim, mostrou-se a forma de se adaptar os três algoritmos para a utilização e aplicação em sistemas de controle, visando a obtenção de modelos reduzidos.

No próximo capítulo serão apresentados os resultados numéricos dos métodos de redução da ordem de sistemas dinâmicos, expressos por uma Função de Transferência, via abordagem clássica determinística (por minimização) e estocástica (Inteligência de Enxame) para fins de comparação.

CAPÍTULO 4
RESULTADOS NUMÉRICOS

4.1- Caso 1: Sistema perfeitamente reduzível para primeira ordem, $G_1(s)$

Em Araújo (2008) foi obtida a redução do sistema representado pela Equação (4.1) para um sistema de primeira ordem, com a finalidade de testar a eficiência do método de redução por minimização da norma dos coeficientes polinomiais do erro (ROMNCPE). Tal modelo apresenta o cancelamento perfeito de polos e zeros do sistema original, gerando assim o modelo exato reduzido proposto na Equação (4.2).

$$G_1(s) = \frac{4s^3 + 28s^2 + 68s + 60}{s^4 + 8s^3 + 24s^2 + 32s + 15} \tag{4.1}$$

$$\hat{G}_1(s) = \frac{a}{s + b} \tag{4.2}$$

Testada a veracidade da metodologia, segue-se então, a determinação de sua ordem mais baixa, devido a influência dos polos mais dominantes do sistema, isto é, dos polos mais significativos e que realmente contribuem de forma expressiva à resposta do sistema. Deste modo, recorre-se a teoria dos Índices de Dominância Modais, apresentados no Capítulo 2.

4.1.1- Aplicação dos IDM's – $G_1(s)$

Para se determinar os IDM's, faz-se necessário conhecer-se as raízes do polinômio característico da função a ser analisada, ou seja, conhecer os polos inerentes ao sistema. Para tanto, levando-se em consideração o modelo original representado pela Função de Transferência, mostrada na Equação (4.1) tem-se que seus polos são: $P_1 = -3$; $P_2 = -1$; $P_3 = -2 + j$ e $P_4 = -2 - j$.

Feito isto, evidenciando-se os seus polos e realizando-se a expansão em frações parciais, tem-se a Equação (4.3), cujos resíduos A e B, são dados pelas Equações (4.4) e (4.5).

$$G_1(s) = \frac{4s^3 + 28s^2 + 68s + 60}{(s+3)(s+1)(s^2+4s+5)} = \frac{A}{s+3} + \frac{B}{s+1} + \frac{Cs+D}{(s^2+4s+5)} \tag{4.3}$$

$$A = \frac{4s^3 + 28s^2 + 68s + 60}{(s+3)(s+1)(s^2+4s+5)}\,(s+3)\Big|_{s=-3} = 0 \tag{4.4}$$

$$B = \frac{4s^3 + 28s^2 + 68s + 60}{(s+3)(s+1)(s^2+4s+5)}\,(s+1)\Big|_{s=-1} = 4 \tag{4.5}$$

A determinação dos resíduos C e D é obtida pelas Equações (4.6) e (4.7).

$$\frac{A}{(s+3)}\,(s^2+4s+5)\Big|_{s=-2-j} + \frac{B}{(s+1)}\,(s^2+4s+5)\Big|_{s=-2-j} + \frac{Cs+D}{(s^2+4s+5)}\,(s^2+4s+5)\Big|_{s=-2-j} = -2C + D - jC \tag{4.6}$$

$$\frac{4s^3 + 28s^2 + 68s + 60}{(s+3)(s+1)(s^2+4s+5)}\,(s^2+4s+5)\Big|_{s=-2-j} = 0 \tag{4.7}$$

Comparando-se as Equações (4.6) e (4.7), vem que: $C = 0$ e $D = 0$. Logo, a expansão mostrada na Equação (4.3) resume-se apenas a Equação (4.8), ou seja, o modelo proposto por Araújo (2008) e apresentado na Equação (4.2).

$$\hat{G}_1(s) = \frac{4}{s+1} \tag{4.8}$$

Seguindo, ainda, com a metodologia dos IDM's e de posse dos polos e resíduos da FT, constrói-se a Tabela 4.1, onde são identificados os seus IDM's, percentuais e acumulados. As colunas 3, 4 e 5 desta, atendem diretamente as Equações (2.11) a (2.15) do Capítulo 2; e indicam a contribuição de cada polo na dinâmica do sistema, referindo-se, portanto, à determinação da ordem adequada para a redução de $G_1(s)$, onde observa-se que o único polo realmente significativo é o polo localizado em -1 do plano complexo, destacado na coluna IDM%($\gamma_{\%(i)}$), cuja contribuição é de 100% na influência da resposta ao estímulo de entrada.

Polos	**Resíduos**	**IDMs** (γ_i)	**IDM%**($\gamma_{\%(i)}$)	**IDM% Acumulado**
$-2+j$	0	0	0,00	0,00
$-2-j$	0	0	0,00	0,00
-3	0	0	0,00	0,00
-1	4	4	100,00	100,00

Tabela 4.1 – IDM's para a FT, $G_1(s)$, apresentada na Equação (4.1).

Concluindo esta etapa, descartam-se os polos complexos conjugados e o polo real em -3, das linhas 1, 2 e 3, respectivamente, cujas contribuições são de 0,0%, isto é, em nada alteram a resposta do sistema, desta forma retornando à Função de Transferência de primeira ordem.

A seguir, aplicam-se as técnicas de redução abordadas nos Capítulos 2 e 3, orientadas pela Tabela 4.1; a primeira, referente ao método determinístico de redução da ordem por minimização da norma dos coeficientes polinomiais do erro (ROMNCPE) gerada entre o modelo original e o modelo proposto reduzido e a segunda, referente ao algoritmo de busca metaheurística, *Firefly Algorithm* (FA).

4.1.2- Aplicação da Metodologia ROMNCPE – $G_1(s)$

A redução por minimização da norma dos coeficientes polinomiais é dada a partir da diferença entre o modelo original do sistema e o modelo reduzido inicialmente proposto, em que um possível erro, caso gerado, deve ser minimizado a fim de manter as respostas e características dos sistemas próximas ou equivalentes umas as outras (original e reduzido). Com base nisto, dispondo da informação pré-estabelecida pelos IDM's, segue-se a redução de $G_1(s)$, pelo método exposto na seção 2.6, a partir do erro dado na Equação (4.9) e (4.10).

$$e(s) = G_1(s) - \hat{G}_{\mathbf{1}_{ROMNCPE}}(s) = \frac{4s^3 + 28s^2 + 68s + 60}{s^4 + 8s^3 + 24s^2 + 32s + 15} - \frac{a}{s + b} \tag{4.9}$$

$$e(s) = \frac{(4s^3 + 28s^2 + 68s + 60).(s + b) - a.(s^4 + 8s^3 + 24s^2 + 32s + 15)}{(s^4 + 8s^3 + 24s^2 + 32s + 15).(s + b)} \tag{4.10}$$

Evidenciando-se os termos em s, o numerador da função erro, N(s), é dado pela Equação (4.11).

$$N(s) = (4 - a)s^4 + (28 + 4b - 8a)s^3 + (68 + 28b - 24a)s^2 + (60 + 68b - 32a)s + (60b - 15a) \tag{4.11}$$

Aplicando a Equação (2.31), min norm^2 { coef. [N(s)] }, tem-se a Equação (4.12).

$$f(a,b) = (4 - a)^2 + (28 + 4b - 8a)^2 + (68 + 28b - 24a)^2 + (60 + 68b - 32a)^2 + (60b - 15a)^2 \tag{4.12}$$

Resolvendo as Equações (2.34) e (2.35), obtêm-se o sistema de Equações (4.13) e (4.14).

$$\begin{cases} 3780a - 7560b - 7560 = 0 & (4.13) \\ -7560a + 18048b + 12192 = 0 & (4.14) \end{cases}$$

Desta forma, encontram-se: a = 4 e b = 1 . Portanto a forma reduzida de $G_1(s)$ é dada pela Equação (4.15).

$$\hat{G}_{\mathbf{1}_{ROMNCPE}}(s) = \frac{a}{s+b} \quad \Rightarrow \quad \hat{G}_{\mathbf{1}_{ROMNCPE}}(s) = \frac{4}{s+1} \tag{4.15}$$

Tem-se, assim, uma função perfeitamente reduzida, ou seja, com erro nulo; com respostas ao degrau idênticas, para o modelo original e reduzido.

Para ilustração dos resultados por meios gráficos, utilizou-se o *software* Matlab para se obter as respostas dos sistemas, original e reduzido, para uma entrada degrau unitário, Figura 4.1. As linhas de comando para este caso e os demais, encontram-se no Anexo II.

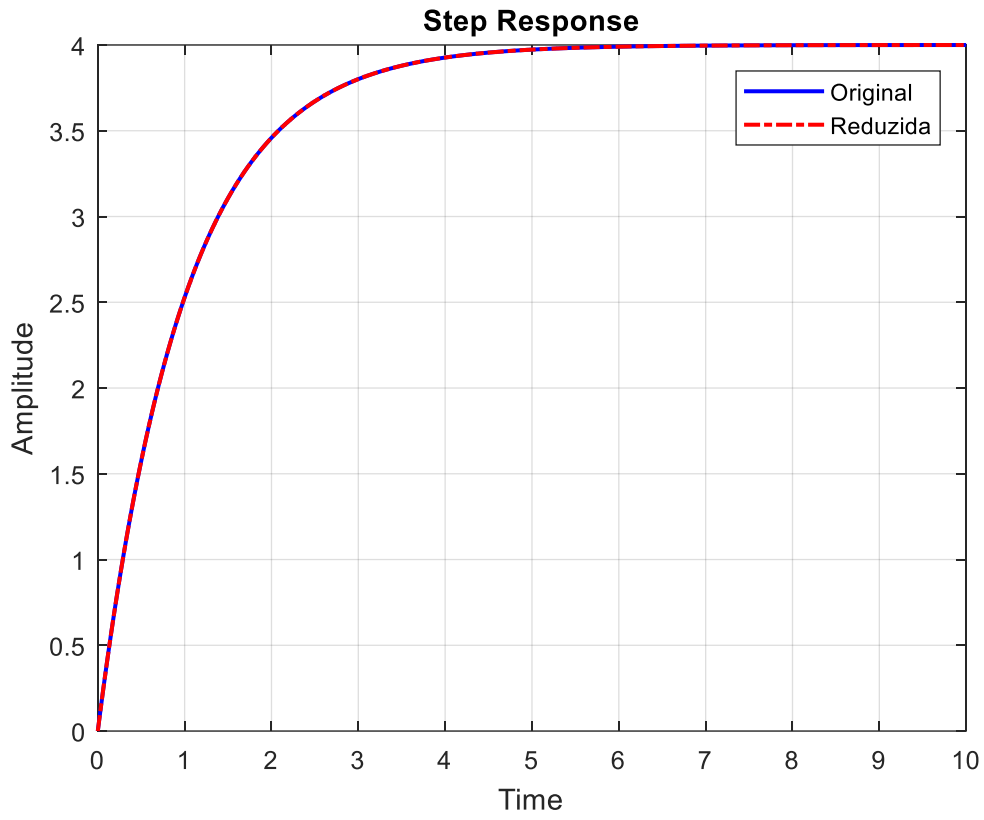

Figura 4.1 - Comparação entre as respostas ao degrau unitário, por ROMNCPE, para o modelo original, $G_1(s)$, e o modelo reduzido, $\hat{G}_{\mathbf{1}_{ROMNCPE}}(s)$ – Caso 1.

Comparando as duas respostas da Figura 4.1 observa-se que elas se sobrepõem, portanto, o erro é nulo, como esperado e mostrado na Figura 4.2 e que, ainda, a função em questão fora perfeitamente reduzida, mostrando a validade do método.

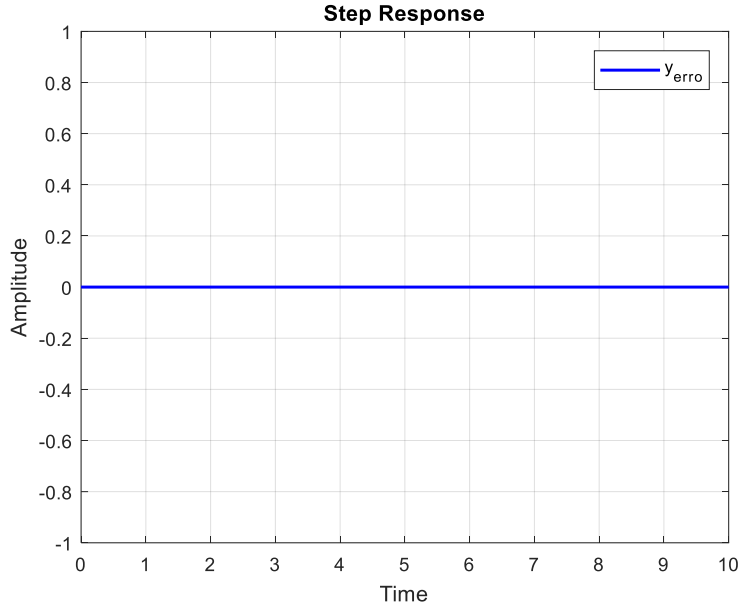

Figura 4.2 - Erro entre as respostas ao degrau para G1(s) e $\hat{G}_{\mathbf{1}_{ROMNCPE}}(s)$ – Caso 1.

Considerando, ainda este exemplo, verifica-se na Figura 4.3 que suas respostas em frequência também são iguais.

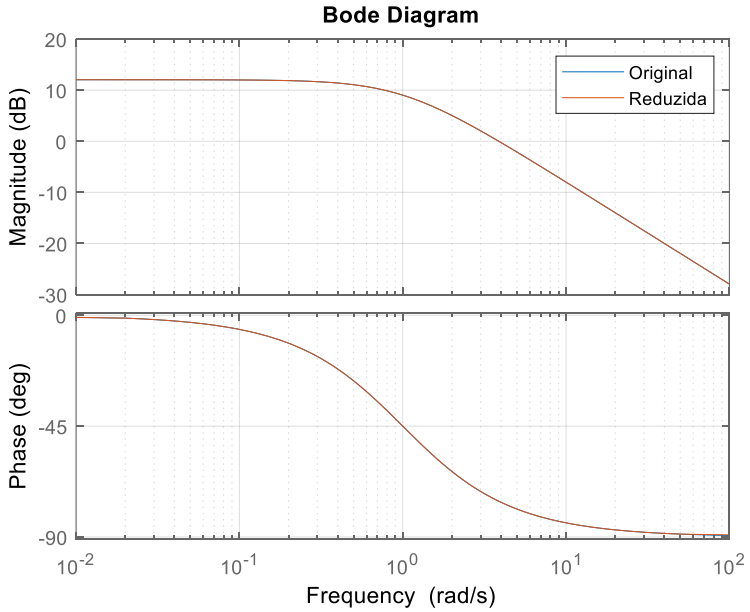

Figura 4.3 - Comparação em magnitude e fase para G1(s) e $\hat{G}_{\mathbf{1}_{ROMNCPE}}(s)$ – Caso 1.

Observa-se novamente, que há uma superposição das respostas de tais modelos. Porém o erro não será nulo para todo e qualquer caso, com isso o seu modelo reduzido não terá as mesmas características do sistema original, contudo, deverá possuir uma resposta aproximada, para que a metodologia adotada seja válida.

Adiante, para efeito de comparação, segue a metodologia abordada no Capítulo 3, em que o algoritmo de busca se mostra, também, bastante eficaz.

4.1.3- Aplicação do *Firefly Algorithm* – $G_1(s)$

Para analisar o desempenho do *Firefly Algorithm* (FA), para este caso, foi realizada a redução do modelo da Equação (4.1) para um modelo de primeira ordem representado pela Equação (4.2), onde o FA deverá encontrar os melhores valores de a e b que conservem as características dinâmicas do modelo original. A Equação (4.16) mostra o modelo reduzido obtido com o uso do FA.

$$\hat{G}_{\mathbf{1}_{FA}}(s) = \frac{4{,}013}{s + 1{,}001} \tag{4.16}$$

Vale ressaltar que uma das características das metaheurísticas é que a correta parametrização dos algoritmos leva a desempenhos ótimos, enquanto que, uma configuração de um dos parâmetros de forma errada influenciaria de forma negativa no desempenho do mesmo. Deste modo, com base nos valores encontrados na literatura e através dos ensaios feitos, chegou-se aos parâmetros de referência apresentados na Tabela 4.2 (Xing & Gao, 2014). Adotou-se o espaço de busca positivo para que os modelos reduzidos gerados fossem de fase mínima.

Parâmetros	Valores
População	25
β_0	2
γ	0,4
Espaço de busca	[0 , 50]
Iteração Máxima	100
Erro Máximo	6 x 10^{-5}

Tabela 4.2 - Parâmetros do FA ajustados e inseridos no algoritmo em Matlab no Anexo III.

A resposta ao degrau unitário do modelo original e do reduzido constam na Figura 4.4. A Figura 4.5 faz a comparação entre as suas respostas em frequências; a Figura 4.6 mostra o erro existente entre as repostas dos dois modelos e a Figura 4.7 mostra o gráfico referente ao número de iterações necessárias para se encontrar os melhores indivíduos que comporão a função de transferência reduzida.

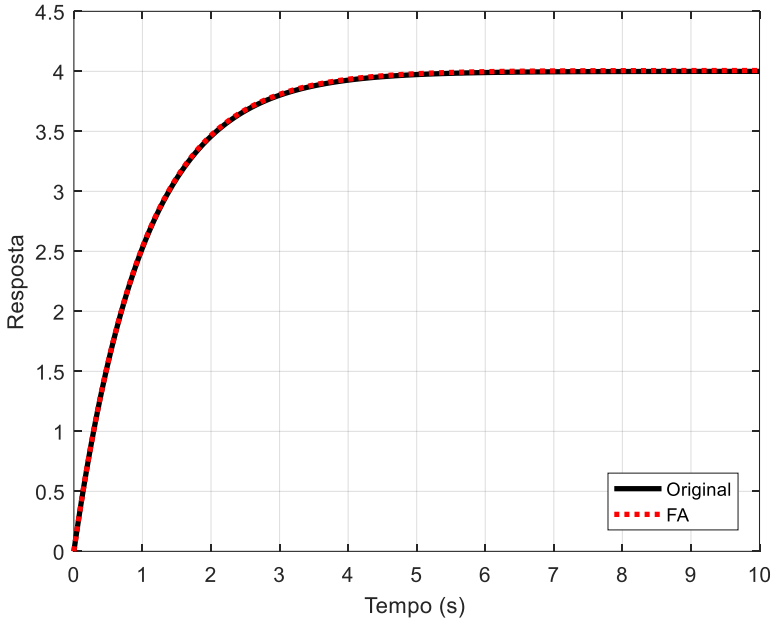

Figura 4.4 - Comparação entre as respostas, ao degrau unitário, do modelo original e do modelo reduzido encontrado pelo FA – Caso 1.

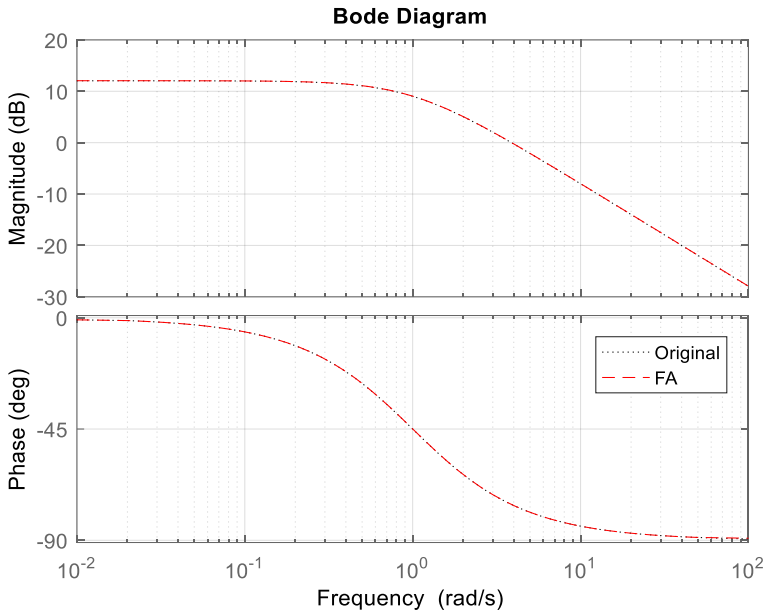

Figura 4.5 - Comparação entre os diagramas de Bode do modelo original e do modelo reduzido encontrado pelo FA – Caso 1.

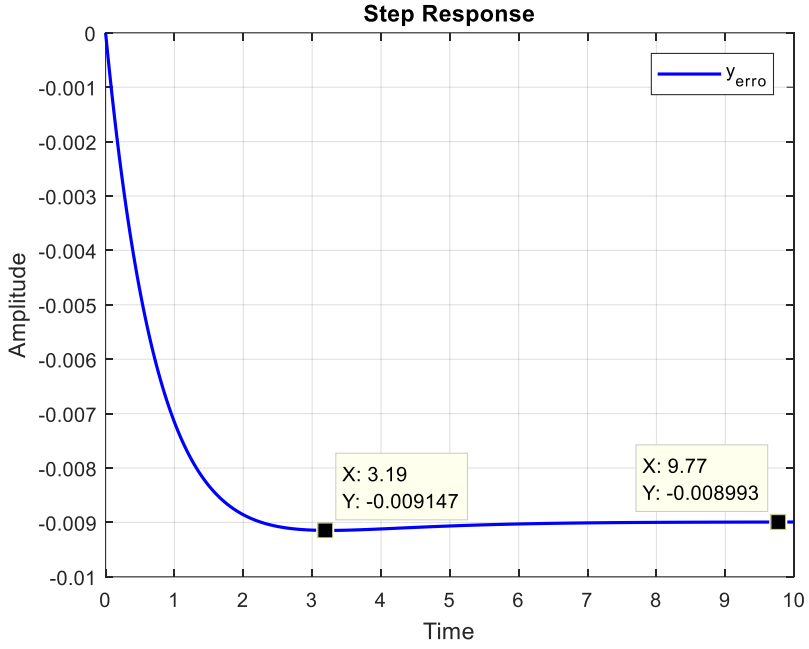

Figura 4.6 - Erro entre a resposta ao degrau unitário da função original e do reduzido encontrado pelo FA – Caso 1.

Em termos de erro para a resposta ao degrau, quando comparado com o erro apresentado na Figura 4.2 (erro nulo), a diferença apresentada na Figura 4.6 se dá em virtude dos parâmetros da Tabela 4.2. Seu erro máximo (em módulo) é de aproximadamente 0,0091 e de regime, 0,0090.

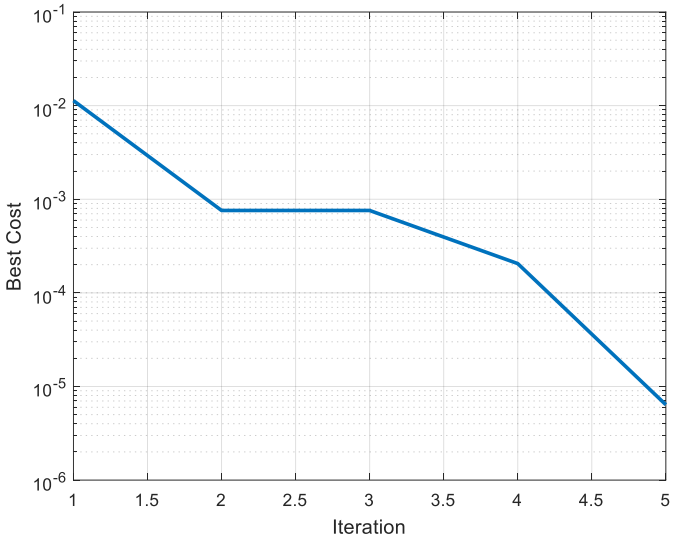

Figura 4.7 - Valor otimizado do erro do melhor indivíduo por iteração – Caso 1.

Dos resultados obtidos, observa-se que o FA gerou um modelo reduzido muito próximo do modelo original, considerando o cancelamento dos polos e zeros, o que demonstra a precisão do método para este caso, em que se tem um modelo reduzido exato. O tempo decorrido para se obter o modelo reduzido, para este exemplo, foi de 10,7 segundos.

O gráfico apresentado na Figura 4.7 mostra os melhores resultados de acordo com cada iteração feita pelo algoritmo. Nota-se que entre a quarta iteração e a quinta, o valor, isto é, o erro, continua a decair, se aproximando de zero, como se esperava, para uma função que apresenta redução exata, porém como o critério de parada utilizado foi o atendimento as condições de erro menor que 6 x 10^{-5} ou ao se atingir o número máximo de iterações especificadas, no caso 100, o algoritmo encerra, então, a sua busca, o que justifica o gráfico da Figura 4.6, retornando a função reduzida apresentada na Equação (4.16). A Tabela 4.3, mostra os resultados obtidos pela busca.

Iteration 1: Best Cost = 11314e-06	Iteration 4: Best Cost = 205,47e-06
Iteration 2: Best Cost = 758,9e-06	Iteration 5: Best Cost = 6,3863e-06
Iteration 3: Best Cost = 758,9e-06	Time: 10.699905 seconds

Tabela 4.3 - Número de Iterações feitas pelo FA e seus respectivos valores.

O tempo demandado foi de, aproximadamente, 10,7 segundos, tempo este, relativamente curto, o que pode não ocorrer para outros casos, onde se exige um esforço computacional um pouco maior do algoritmo de busca.

4.2- Caso 2: Sistema com redução de terceira para segunda ordem, $G_2(s)$

Para a FT da Equação (4.17) apresentada em Araújo (2008), deseja-se determinar um modelo de ordem reduzida equivalente de modo a manter as características intrínsecas do sistema.

$$G_2(s) = \frac{1}{s^3 + 3{,}09s^2 + 3{,}179s + 1{,}089} \tag{4.17}$$

Utilizando a função “residue” do Matlab, obtêm-se os resultados expostos na Tabela 4.4.

r (resíduos)	p (polos)	k (ganho)
90,9091	-1,1	[ ]
-1000	-1,0	
909,0909	-0,99	

Tabela 4.4 - Valores retirados do Matlab para expansão de $G_2(s)$ do caso 2.

Neste caso, como os polos estão extremamente próximos entre si e próximos da origem do plano s, não é evidente se algum deles pode ser considerado dominante ou não. Para tanto, segue a obtenção de ordem adequada por IDM, seguindo os mesmos procedimentos anteriormente descritos.

4.2.1- Aplicação dos IDM’s – $G_2(s)$

A Tabela 4.5 resume os resultados obtidos para os IDM’s (Equações (2.13), (2.14) e (2.15)).

Polos	Resíduos	IDMs (γ_i)	IDM%($\gamma_{\%(i)}$)	IDM% Acumulado
$-1{,}1$	$90{,}9091$	$82{,}64$	$4{,}1303$	$4{,}1303$
$-1{,}0$	-1000	-1000	$49{,}9771$	$54{,}1074$
$-0{,}99$	$909{,}0909$	$918{,}2736$	$45{,}8926$	100

Tabela 4.5 - Valores para obtenção de $\hat{G}_2(s)$.

Da coluna 4 da Tabela 4.5 observa-se que o polo situado em -1.1, pode ser considerado desprezível em relação aos polos -1 e -0,99 , devido ao seu baixo IDM% assim, a função de transferência tem apenas dois polos que contribuem de forma considerável para a sua resposta, e, portanto a função de transferência reduzida será de 2ª ordem.

Determinada a ordem adequada desta, segue a redução com base na metodologia proposta em Araújo (2008) e em Silva (2017).

4.2.2- Aplicação da Metodologia ROMNCPE – $G_2(s)$

Objetiva-se a redução para o seguinte modelo dado pela Equação (4.18).

$$\hat{G}_{2_{ROMNCPE}}(s) = \frac{cs + d}{s^2 + as + b} \tag{4.18}$$

De tal forma que o numerador do erro é dado pela Equação (4.19) e aplicando a Equação (2.31) resulta nas Equações (4.20) a (4.24) e no sistema de Equações (4.25).

$$N(s) = (-c)s^4 + (-3{,}09c - d)s^3 + (1 - 3{,}179c - 3{,}09d)s^2 + (a - 1{,}089c - 3{,}179d)s + (b - 1{,}089d) \tag{4.19}$$

$$f(a,b,c,d) = (-c)^2 + (-3{,}09c - d)^2 + (1 - 3{,}179c - 3{,}09d)^2 + (a - 1{,}089c - 3{,}179d)^2 + (b - 1{,}089d)^2 \tag{4.20}$$

$$\frac{\partial f(a,b,c,d)}{\partial a} = 0 \quad \Rightarrow \quad 2a - 2{,}178c - 6{,}358d = 0 \tag{4.21}$$

$$\frac{\partial f(a,b,c,d)}{\partial b} = 0 \Rightarrow \quad 2b - 2{,}178d = 0 \tag{4.22}$$

$$\frac{\partial f(a,b,c,d)}{\partial c} = 0 \Rightarrow -2{,}178a + 43{,}680124c + 32{,}750082d = 6{,}358 \tag{4.23}$$

$$\frac{\partial f(a,b,c,d)}{\partial d} = 0 \Rightarrow -6{,}358a - 2{,}178b + 32{,}750082c + 43{,}680124d = 6{,}18 \tag{4.24}$$

$$\begin{cases} 2a + 0b - 2{,}178c - 6{,}358d = 0 \\ 0a + 2b + 0c - 2{,}178d = 0 \\ -2{,}178a + 0b + 43{,}680124c + 32{,}750082d = 6{,}358 \\ -6{,}358a - 2{,}178b + 32{,}750082c + 43{,}680124d = 6{,}18 \end{cases} \tag{4.25}$$

Usando o Matlab para resolver o referido sistema de equações, tem-se: a = 1,2805; b = 0,4851; c = -0,1246; d = 0,4455. Portanto, o modelo reduzido é dado pela Equação (4.26).

$$\hat{G}_{2_{ROMNCPE}}(s) = \frac{-0{,}1246s + 0{,}4455}{s^2 + 1{,}2805s + 0{,}4851} \tag{4.26}$$

Observa-se que o modelo reduzido é um sistema de fase não-mínima, ou seja, o zero está posicionado no semi-plano direito do plano s enquanto que o modelo original é de fase mínima.

Adiante, na seção 4.2.3, procura-se enfatizar a questão do posicionamento deste zero e de como a tentativa de elimina-lo afetará consideravelmente a resposta ou mesmo a busca pela FT adequada, na redução por *Particle Swarm Optimization* - PSO.

Seguindo a metodologia, as respostas no tempo e em frequência, bem como o erro, para a função original e a reduzida são mostradas nas Figuras 4.8, 4.9 e 4.10.

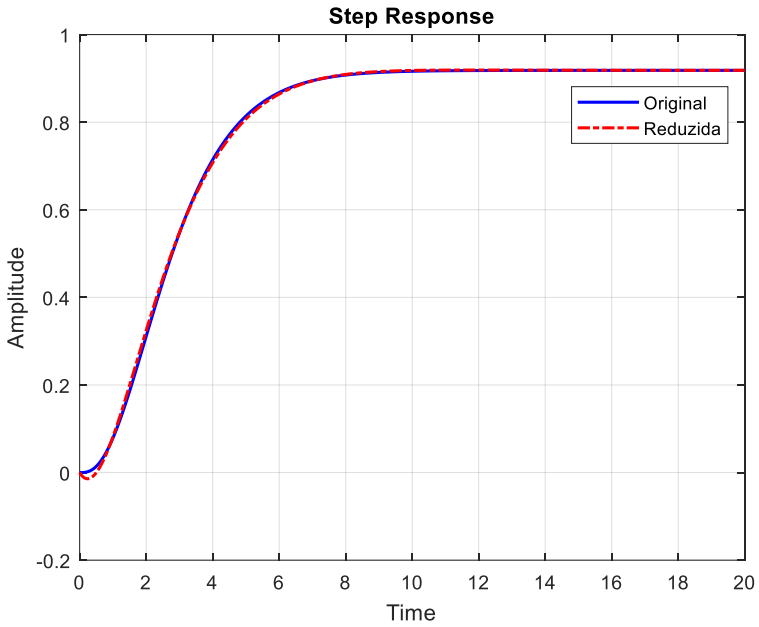

Figura 4.8 - Resposta ao degrau unitário para $G_2(s)$, modelo original, e para $\hat{G}_{2_{ROMNCPE}}(s)$, modelo reduzido – Caso 2.

Como trata-se de sistemas aproximados, o erro máximo foi de, aproximadamente, 0,0168 enquanto o erro em regime tende para zero, conforme observa-se no gráfico da Figura 4.9.

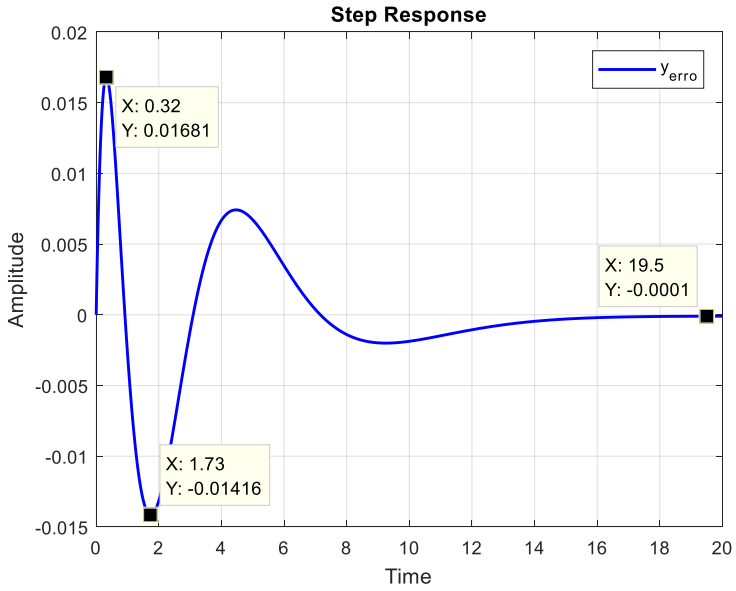

Figura 4.9 - Erro gerado pela reposta ao degrau unitário entre a função original e a reduzida por ROMNCPE – Caso 2.

A influência do zero no semi-plano direito adianta a fase no modelo reduzido, o que pode ser visto no diagrama de bode da Figura 4.10, adiantada de 360º em relação a resposta original.

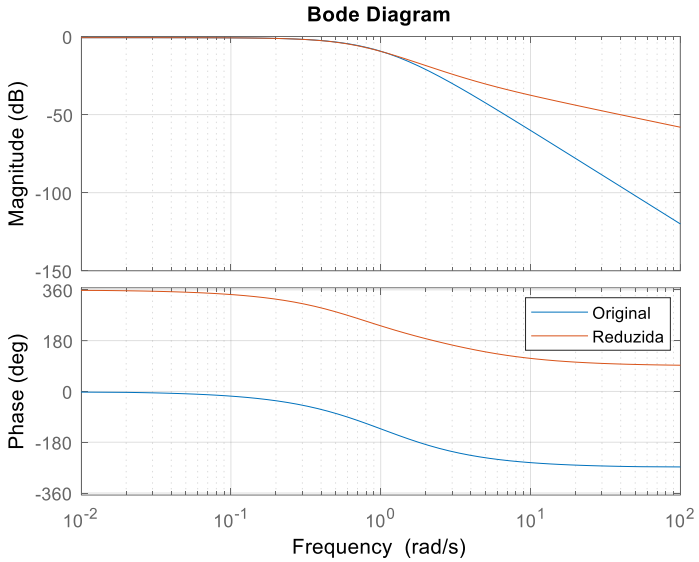

Figura 4.10 - Diagrama de Bode (magnitude e fase) para as funções $G_2(s)$ e $\hat{G}_{2_{ROMNCPE}}(s)$ – Caso 2.

4.2.3- Aplicação do *Particle Swarm Optimization* – $G_2(s)$

Na tentativa de se eliminar possíveis modelos de fase não-mínima, o algoritmo foi parametrizado de acordo com a Tabela 4.6.

Parâmetros	Valores
População	25
ω	0,4
$c_1 = c_2$	2
Espaço de busca	[0 , 5]
Iteração Máxima	200
Erro Máximo	6 x 10^{-5}

Tabela 4.6 - Parâmetros do PSO ajustados e inseridos no algoritmo em Matlab no Anexo IV.

Obtendo-se, desta forma, a Função de Transferência reduzida de segunda ordem representada pelo modelo da Equação (4.27).

$$\hat{G}_{2_{PSO}}(s) = \frac{0{,}9789}{s^2 + 3{,}207s + 1{,}062} \tag{4.27}$$

Como consequência, a resposta ao degrau para a função reduzida não acompanha exatamente a resposta da função original nos primeiros 10 segundos, Figura 4.11. Porém, o erro tende a zero em regime permanente, como mostra a Figura 4.12.

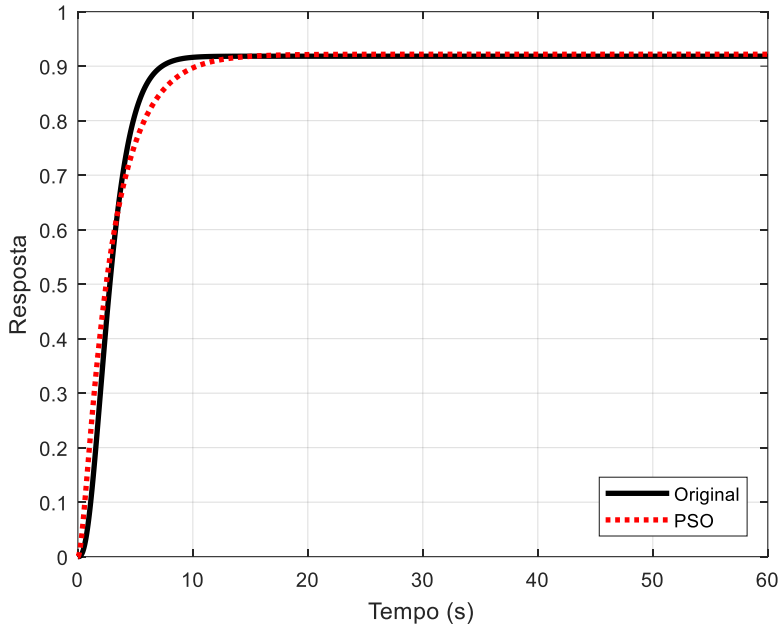

Figura 4.11 - Resposta ao degrau para as funções G2(s) e $\hat{G}_{2_{PSO}}(s)$ – Caso 2.

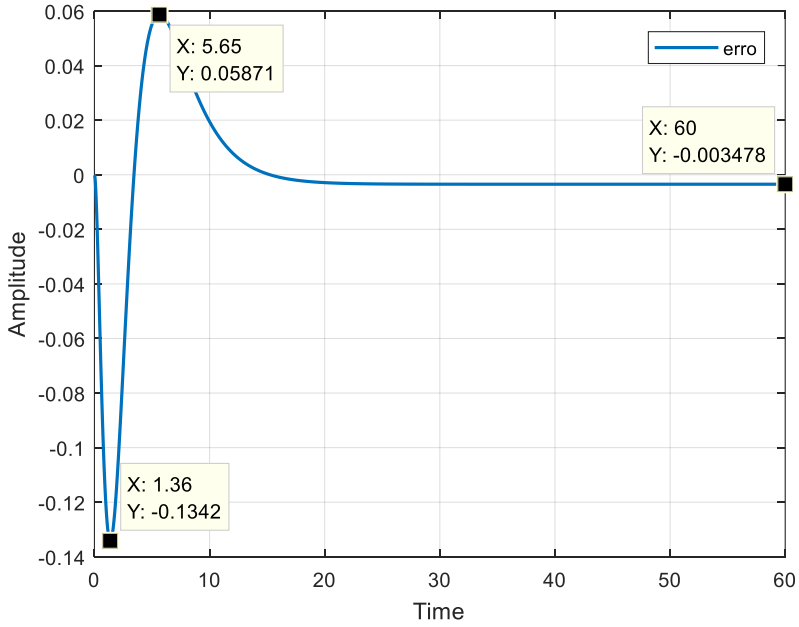

Figura 4.12 - Erro entre a reposta ao degrau unitário da função original e da reduzida por PSO – Caso 2.

Da Figura 4.12, constata-se que o erro máximo é de, aproximadamente, 0,1342 e, em regime permanente, de 3,478e-3, ambos em módulo.

A resposta em frequência é mostrada na Figura 4.13. Percebe-se que, tanto a magnitude quanto a fase, distorcem apenas nas altas frequências, isto é, para as baixas frequências as respostas praticamente se equivalem, observando uma leve diferença entre elas, ou ainda, encontram-se em fase, diferentemente do que ocorrera para a resposta defasada no método anterior.

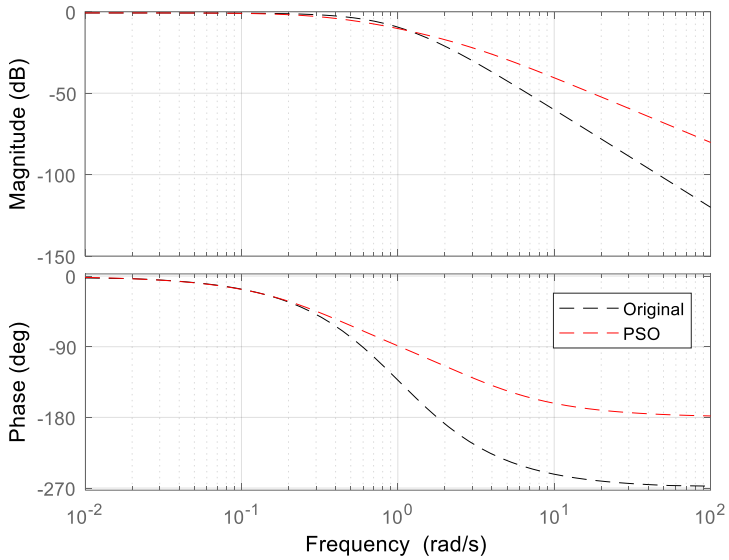

Figura 4.13 - Comparação em magnitude e fase para G2(s) e $\hat{G}_{2_{PSO}}(s)$ – Caso 2.

Para o número de iterações, acompanha-se o gráfico da Figura 4.14.

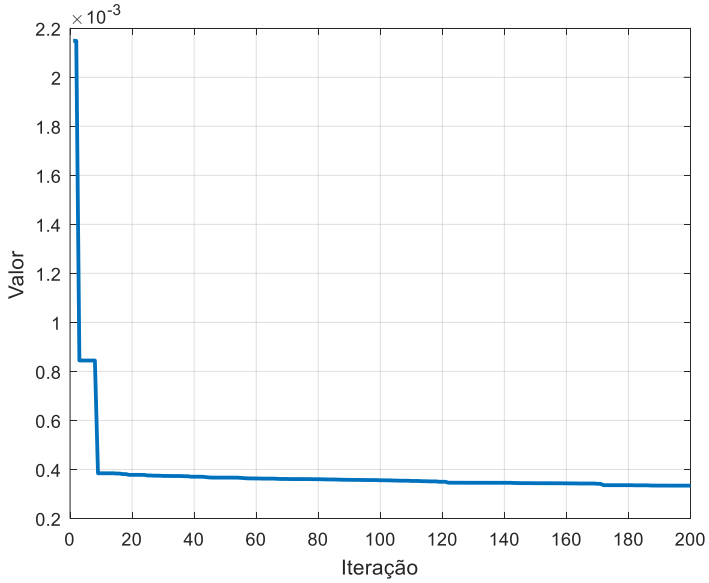

Figura 4.14 - Valor otimizado, por PSO, do erro do melhor indivíduo por iteração – Caso 2.

O tempo decorrido para a completa execução do algoritmo ficou em torno de 54,67 segundos (Tabela 4.7), o que demandou um tempo relativamente maior quando comparado com a busca feita no caso 1, com o *Firefly*. Contudo, observa-se que os melhores indivíduos praticamente convergem por volta da 10ª iteração (Figura 4.14), persistindo, um leve declínio no valor de erro admitido pela parametrização da Tabela 4.6

A Tabela 4.7 mostra as primeiras iterações feitas pelo algoritmo PSO, assim como as últimas.

Iteration 1: Best Cost = 214.91e-05	Iteration 196: Best Cost = 33.391e-05
Iteration 2: Best Cost = 214.91e-05	Iteration 197: Best Cost = 33.391e-05
Iteration 3: Best Cost = 84.443e-05	Iteration 198: Best Cost = 33.391e-05
Iteration 4: Best Cost = 84.443e-05	Iteration 199: Best Cost = 33.391e-05
Iteration 5: Best Cost = 84.443e-05	Iteration 200: Best Cost = 33.373e-05
Elapsed time is 54.6717 seconds.	

Tabela 4.7 - Número de Iterações feitas pelo PSO e seus respectivos valores.

4.3- Caso 3: Sistema com redução de sexta para segunda ordem, $G_3(s)$

A Equação (4.28), apresentada em Silva (2017), mostra a validade de ambos os métodos para sistemas de ordem relativamente altas, as quais possuem possíveis redundâncias em suas modelagens de tal maneira que podem ser eliminadas.

$$G_3(s) = \frac{s^4 + 6s^3 + 96s^2 + 780s + 3250}{s^6 + 13{,}2s^5 + 158{,}6s^4 + 594s^3 + 2765s^2 + 1050s + 2500} \tag{4.28}$$

4.3.1- Aplicação dos IDM's – $G_3(s)$

A Tabela 4.8 resume os Índices de Dominância Modais para este caso.

Polos	Resíduos	IDMs (γ_i)	IDM% ($\gamma_{\%(i)}$)	IDM% Acumulado
$-5+8{,}6603i$	$0{,}0071-0{,}0673i$	0,006183333	0,43357554	0,43357554
$-5-8{,}6603i$	$0{,}0071+0{,}0673i$	0,006183333	0,43357554	0,86715108
$-1{,}5+4{,}7697i$	$-0{,}0228+0{,}158i$	$-0{,}031512456$	2,209654702	3,076805781
$-1{,}5-4{,}7697i$	$-0{,}0228-0{,}158i$	$-0{,}031512456$	2,209654702	5,286460483
$-0{,}1+0{,}995i$	$0{,}0157-0{,}6772i$	0,675367116	47,35676976	52,64323024
$-0{,}1-0{,}995i$	$0{,}0157+0{,}6772i$	0,675367116	47,35676976	100

Tabela 4.8 – IDM's relacionados ao modelo da Equação (4.28), indicando a ordem adequada para obtenção de seu respectivo modelo reduzido.

Observando a quarta coluna da Tabela 4.8, nota-se que apenas as duas últimas linhas possuem IDM% considerável, enquanto os demais podem ser descartados, significando a redução adequada para a segunda ordem, conforme os polos dominantes do sistema demonstram. Para tanto, seguem as metodologias.

4.3.2- Aplicação da Metodologia ROMNCPE – $G_3(s)$

Executando o algoritmo do Anexo II, de acordo com a redução por minimização da norma dos coeficientes polinomiais do erro, chega-se a Equação (4.29).

$$\hat{G}_{3_{ROMNCPE}}(s) = \frac{0{,}01207s + 1{,}255}{s^2 + 0{,}1829s + 0{,}9658} \tag{4.29}$$

A resposta do modelo reduzido acompanha, quase que integralmente, a do sistema original, gerando um erro perceptível apenas na quinta casa decimal, em regime permanente, tendendo a zero, mostrando uma ligeira diferença entre ambas, conforme mostra a Figura 4.15.

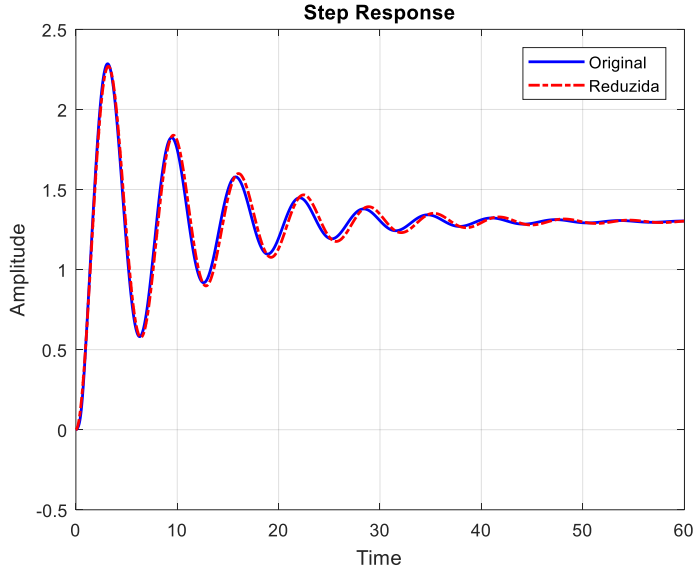

Figura 4.15 - Resposta ao degrau unitário para G3(s) e $\hat{G}_{3_{ROMNCPE}}(s)$ - Caso 3.

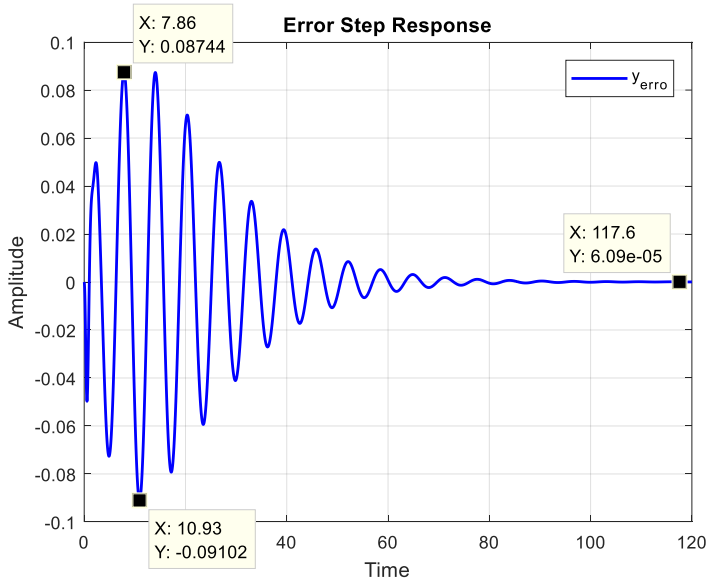

Figura 4.16 - Erro gerado entre as respostas da função original e da reduzida por ROMNCPE – Caso 3.

De acordo com o diagrama de Bode da Figura 4.17, a resposta para o modelo reduzido se encontra defasada, ou ainda, atrasada em 360º com relação a resposta do seu modelo original, distorcida tanto nas baixas quanto nas altas

frequências. Já as suas magnitudes se aproximam, principalmente nas baixas frequências.

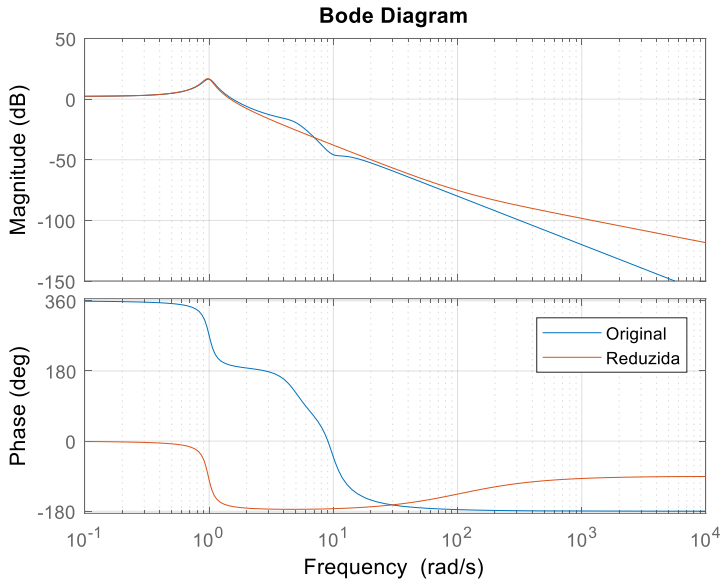

Figura 4.17 - Comparação em magnitude e fase para G3(s) e $\hat{G}_{3_{ROMNCPE}}(s)$ – Caso 3.

4.3.3- Aplicação do *Shuffled Frog Leaping Algorithm* – G3(s)

O modelo encontrado pelo SFLA, utilizando os parâmetros da Tabela 4.9 no algoritmo, não gerou zeros na Função de Transferência reduzida obtida e dada pela Equação (4.30).

Parâmetros	Valores
População	25
m	5
n	5
Espaço de busca	[0 , 5]
Iteração Máxima	200
Erro Máximo	6 x 10^{-5}

Tabela 4.9 - Parâmetros do SFLA para o algoritmo do Anexo V.

$$\hat{G}_{3_{SFLA}}(s) = \frac{1{,}31}{s^2 + 0{,}1937s + 1{,}005} \tag{4.30}$$

As fases, no diagrama de Bode, se equiparam para as altas frequências e suas magnitudes estão bem mais próximas umas das outras. O erro máximo encontrado, entre as respostas ao degrau, foi bem menor, quando analisado frente ao método ROMNCPE. Os resultados gráficos obtidos são mostrados nas Figuras (4.18), (4.19) e (4.20) a seguir, e ilustram a eficiência do método.

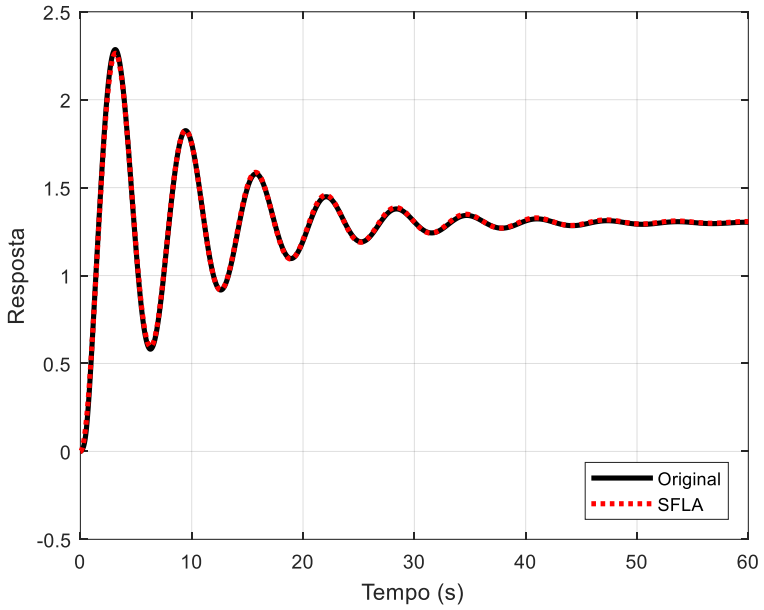

Figura 4.18 - Resposta ao degrau unitário para $G_3(s)$, sistema original, e $\hat{G}_{3_{SFLA}}(s)$, reduzido por SFLA – Caso 3.

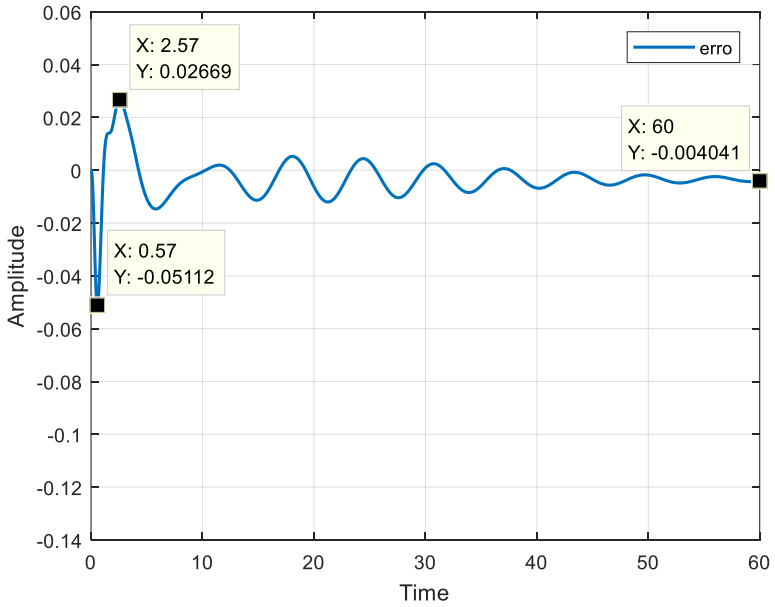

Figura 4.19 - Erro gerado pela reposta ao degrau unitário entre as funções original e reduzida por SFLA – Caso 3.

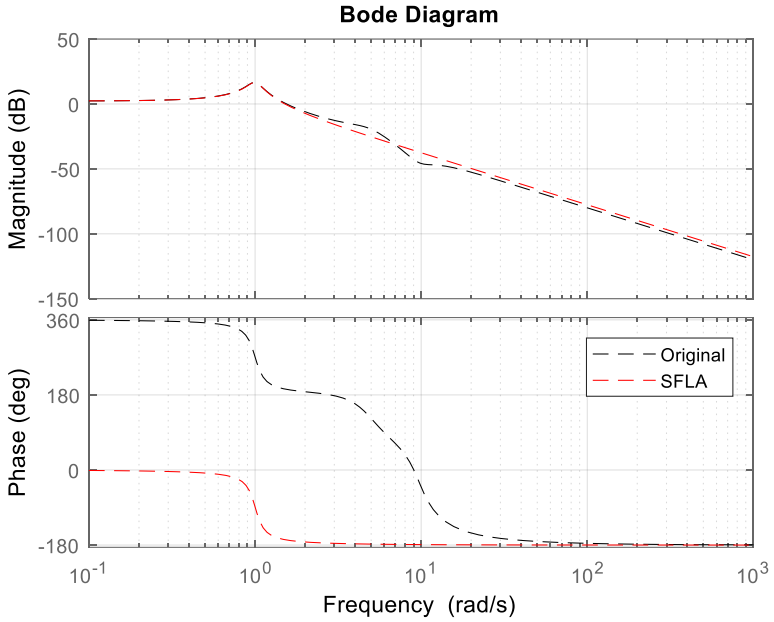

Figura 4.20 - Comparação em magnitude e fase para G3(s) e $\hat{G}_{3_{SFLA}}(s)$ - Caso 3.

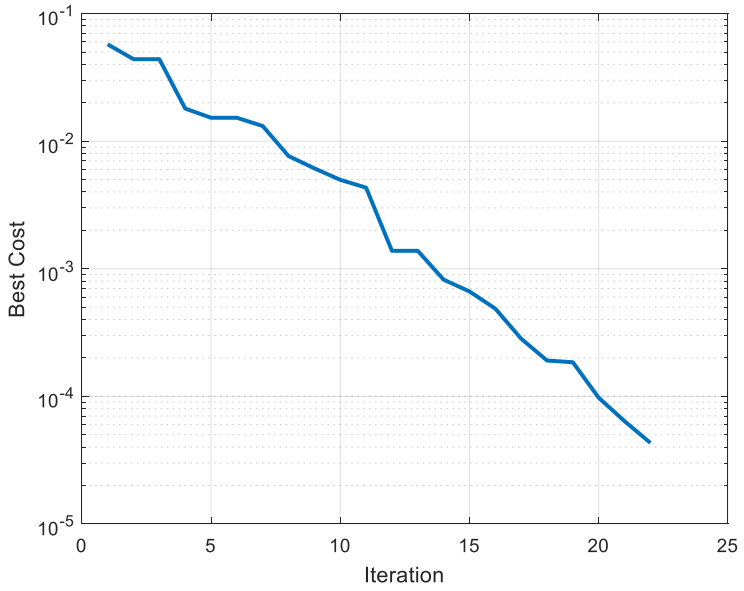

Figura 4.21 - Valor otimizado do erro do melhor indivíduo por iteração - Caso 3.

Iteration 1: Best Cost = 5743.4e-05	Iteration 19: Best Cost = 18.487e-05
Iteration 2: Best Cost = 4393.5e-05	Iteration 20: Best Cost = 9.7491e-05
Iteration 3: Best Cost = 4393.5e-05	Iteration 21: Best Cost = 6.3879e-05
Iteration 4: Best Cost = 1801.6e-05	Iteration 22: Best Cost = 4.3137e-05
Elapsed time is 24.4620 seconds	

Tabela 4.10 - Número de Iterações feitas pelo SFLA e seus respectivos valores.

Verificada e eficácia dos métodos de inteligência de enxame frente ao método determinístico exposto, ou seja, que os modelos reduzidos obtidos por inteligência de enxame, produzem respostas no domínio do tempo continuo muito próximas as dos seus respectivos sistemas originais bem como respostas em frequência, por terem natureza estocástica, faz-se necessário, mensurar e avaliar, quão precisas são as suas respostas, ou seja, sua busca em relação aos melhores indivíduos.

Para tanto, os próximos exemplos seguem de tal forma, a avaliar este quesito, em termos de média do tempo de iteração demandado para cada algoritmo, bem como, o valor de custo e o número de iterações realizadas, considerando as condições de parametrização dos mesmos.

4.4- Caso 4: Sistema com redução de quarta para segunda ordem, $G_4(s)$

Para verificar o desempenho dos algoritmos FA, PSO e SFLA, na obtenção de modelos reduzidos de segunda ordem, considerou-se o modelo de quarta ordem da Equação (4.31), Araújo (2008), e obteve-se, respectivamente, os modelos reduzidos das Equações (4.32), (4.33) e (4.34).

$$G_4(\mathrm{s}) = \frac{s^3 + 7s^2 + 24s + 24}{s^4 + 10s^3 + 35s^2 + 50s + 24} \tag{4.31}$$

$$\hat{G}_{4_{FA}}(\mathrm{s}) = \frac{9{,}014}{s^2 + 9{,}937s + 9{,}013} \tag{4.32}$$

$$\hat{G}_{4_{PSO}}(\mathrm{s}) = \frac{8{,}055}{s^2 + 8{,}669s + 8{,}053} \tag{4.33}$$

$$\hat{G}_{4_{SFLA}}(\mathrm{s}) = \frac{8{,}342}{s^2 + 9{,}202s + 8{,}342} \tag{4.34}$$

Na resposta temporal, Figura 4.22, o erro máximo para o FA foi de 1,0984e-04, em relação a resposta original. Para o PSO o erro máximo foi de 2,6425e-04 e para o SFLA, de 2,7250e-06, apresentando erro a partir da sexta casa decimal, ou ainda, um erro quase cem vezes menor frente aos demais algoritmos.

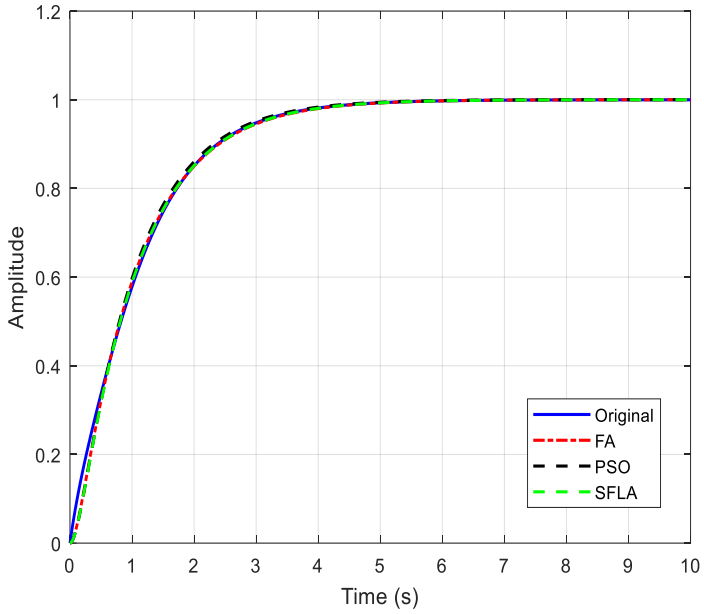

Figura 4.22 - Respostas temporais ao degrau unitário para o sistema original e os reduzidos por IE – Caso 4.

Em relação a resposta em frequência, a principal distorção se dá nas altas frequências, conforme mostra a Figura 4.23.

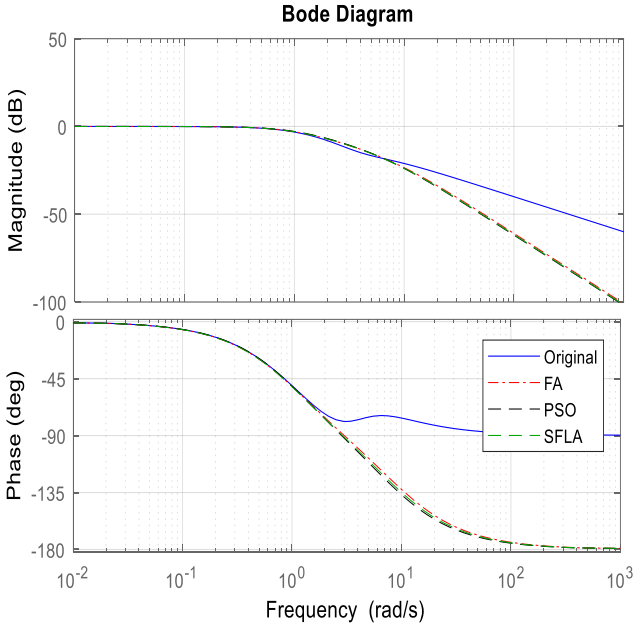

Figura 4.23 - Respostas em frequência (magnitude e fase) para o sistema original e os reduzidos por IE – Caso 4.

A Tabela 4.11 mostra os dados colhidos pela simulação do FA em um campo de busca de [0,10], número máximo de iterações 25, para uma população de 25 indivíduos e critério de parada 9e-6. A mesma parametrização segue para PSO, colhidos os dados na Tabela 4.12.

	Iterações	**Custo**	**Tempo (s)**
Max	25	9,4744e-06	84,534097
Min	8	8,8796e-06	26,541836
Média	15,8	9,00e-06	52,9462701
Desv. Padrão	6,033241252	0,171695e-06	20,94345552

Tabela 4.11 - Dados Fornecidos Pelo FA – Caso 4.

	Iterações	**Custo**	**Tempo (s)**
Max	25	54,688e-06	7,0138
Min	3	8,8603e-06	0,849
Média	21,3	20,2e-06	5,93078
Desv. Padrão	7,180993432	14,8755e-06	1,99655449

Tabela 4.12 - Dados Fornecidos Pelo PSO – Caso 4.

O número de iterações para o SFLA foi de 200 e uma população de 10 indivíduos, mantendo o restante da parametrização em [0,10] e 9e-6, para o espaço de busca e parada, respectivamente. Obteve-se a Tabela 4.13.

	Iterações	**Custo**	**Tempo (s)**
Max	200	17,614e-06	376,3868
Min	22	8,9143e-06	26,8122
Média	155,3	10,8e-06	262,21935
Desv. Padrão	63,56632582	2,9831e-06	120,716039

Tabela 4.13 - Dados Fornecidos Pelo SFLA – Caso 4.

O PSO realizou a busca em menor tempo hábil quando comparado com os outros dois algoritmos de enxame; realizou aproximadamente o mesmo número de iterações que o FA, porém obteve um custo muito alto e seu desvio padrão também se mostrou relativamente alto, o que mostra a grande dispersão dos valores encontrados, que estão diretamente relacionados a precisão do algoritmo para esse caso. Por outro lado, o SFLA apresentou um maior número de iterações e um longo tempo de processamento, com desvio padrão elevado; o seu custo foi moderado, com baixo desvio padrão.

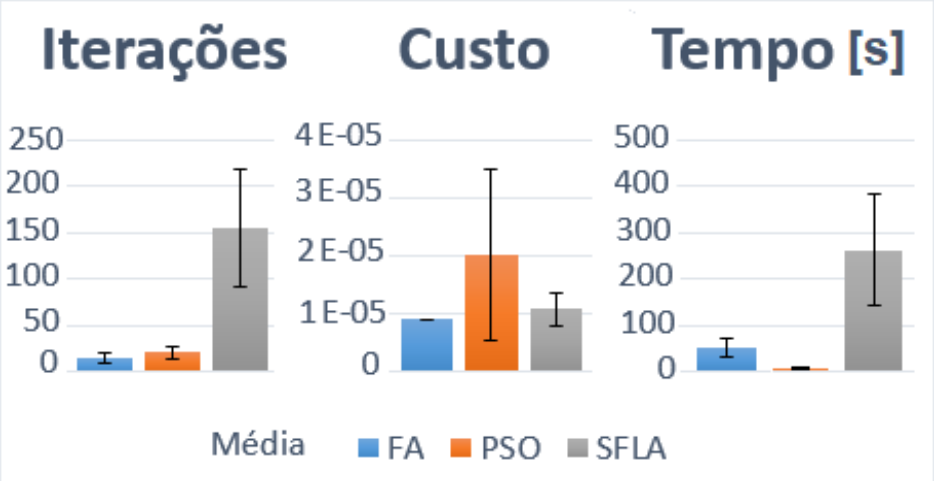

Figura 4.24 - Gráficos realizados no Excel, com base em dez simulações por algoritmo, para efeito de comparação das médias do número de iterações, melhor custo e tempo de execução para o Caso 4.

4.5- Caso 5: Sistema com redução de quarta para segunda ordem, $G_5(s)$

Com base no modelo abordado em Pena (1990), apresentado na Equação (4.35), buscou-se uma redução de quarta para segunda ordem, obtendo-se os modelos das Equações (4.36), (4.37) e (4.38) para o FA, PSO e SFLA respectivamente.

$$G_5(s) = \frac{50s + 100}{s^4 + 25s^3 + 50s^2 + 75s + 100} \tag{4.35}$$

$$\hat{G}_{5_{FA}}(\mathrm{s}) = \frac{2{,}453}{s^2 + 0{,}3458s + 2{,}455} \tag{4.36}$$

$$\hat{G}_{5_{PSO}}(\mathrm{s}) = \frac{2{,}457}{s^2 + 0{,}3471s + 2{,}459} \tag{4.37}$$

$$\hat{G}_{5_{SFLA}}(\mathrm{s}) = \frac{2{,}453}{s^2 + 0{,}3499s + 2{,}458} \tag{4.38}$$

As respostas temporais ao degrau, Figura 4.25, para os modelos reduzidos, pouco divergem entre si no transitório, sendo mais perceptível a diferença em seus tempos de subida. O sobressinal para os modelos reduzidos se mostrou levemente acentuado, sendo o erro máximo para o FA: 0,0289; para o PSO: 0,0281 e para o SFLA: 0,0240.

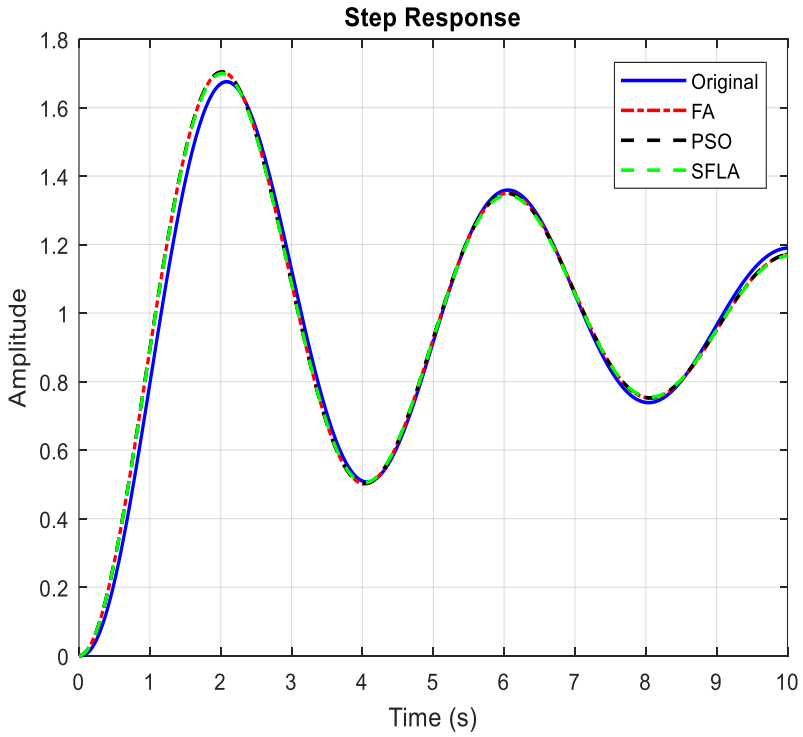

Figura 4.25 - Respostas temporais ao degrau unitário para os sistemas original e reduzidos por IE – Caso 5.

Para este caso as repostas em frequência se aproximam bastante, até mesmo para as altas frequências, em magnitude e permanecendo constante na

fase, conforme se observa na Figura 4.26.

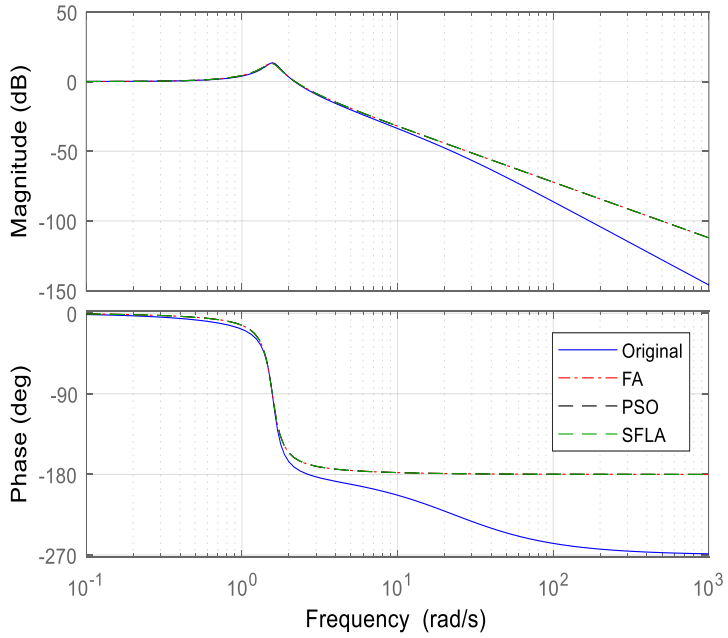

Figura 4.26 - Respostas em frequência (magnitude e fase) para os sistemas original e reduzidos por IE – Caso 5.

Uma população de 25 indivíduos, para um espaço de busca de [0,10] e critério de parada 1,4e-4, tanto para o FA quanto para o PSO, com um número máximo de iterações 25 e 200, respectivamente, geraram as Tabelas 4.14 e 4.15.

	Iterações	**Custo**	**Tempo (s)**
Max	25	139,74e-06	84,054537
Min	9	131,26e-06	30,90556
Média	14,7	135,00e-06	49,539345
Desv. Padrão	4,945255864	3,2316e-06	16,55171007

Tabela 4.14 - Dados Fornecidos Pelo FA – Caso 5.

	Iterações	**Custo**	**Tempo (s)**
Max	165	139,21e-06	45,8295
Min	37	132,85e-06	10,2327
Média	85,4	137,00e-06	23,77045
Desv. Padrão	46,85960117	2,17132e-06	13,10088617

Tabela 4.15 - Dados Fornecidos Pelo PSO – Caso 5.

Com 200 iterações e 10 indivíduos para o SFLA, sendo mantido o restante da parametrização, obteve-se a Tabela 4.16.

	Iterações	**Custo**	**Tempo (s)**
Max	23	139,48e-06	29,5744
Min	13	134,2e-06	16,093
Média	16,3	137,00e-06	20,40483
Desv. Padrão	3,23350515	1,77724e-06	4,056262984

Tabela 4.16 - Dados Fornecidos Pelo SFLA – Caso 5.

Com custos equiparados tem-se como melhor resultado e desempenho, neste caso, o SFLA, pois o mesmo mostrou um menor tempo de processamento, realizando menores buscas pelos melhores indivíduos na população que compunham os coeficientes do modelo reduzido. O custo do FA, apesar de ser o menor destes, mostrou grande dispersão de seus valores, conforme mostra a barra de erros no gráfico de custo da Figura 4.27, o que resulta em uma maior imprecisão de seus resultados. Por sua vez, o SFLA mostrou menor desvio padrão para os três gráficos.

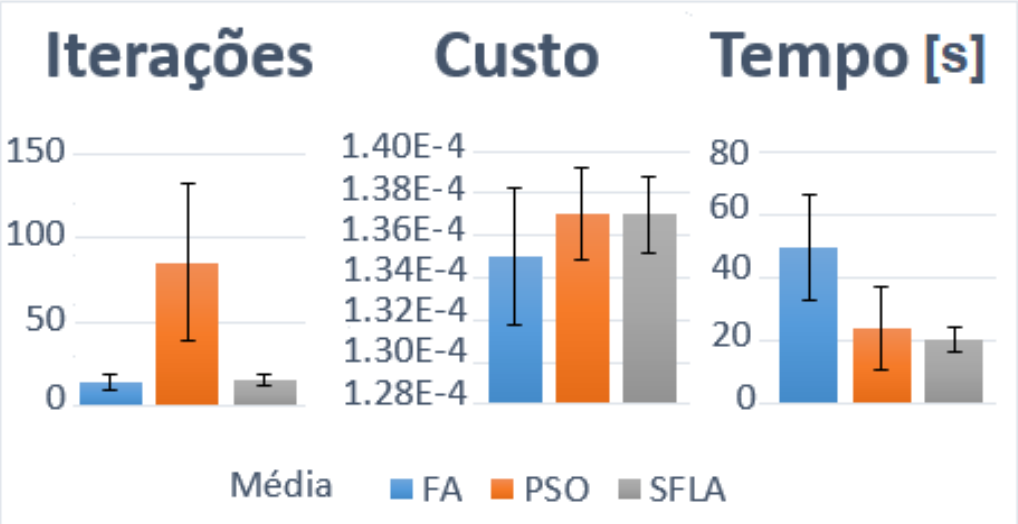

Figura 4.27 - Gráficos realizados no Excel, com base em dez simulações por algoritmo, para efeito de comparação das médias do número de iterações, melhor custo e tempo de execução para o Caso 5.

4.6- Caso 6: Sistema com redução de sexta para terceira ordem, $G_6(s)$

Para efeito de testes, buscou-se reduzir uma função de transferência de grau relativamente alto, conforme mostra a ordem e os valores dos coeficientes do modelo original mostrados na Tabela 4.17.

Coeficientes de	Numerador	Denominador
s6	0	1
s5	0	262
s4	0	25841
s3	0	1,158e6
s2	1	2,094e7
s1	-22e3	4,274e7
s0	1,2e8	9,303e7

Tabela 4.17 - Coeficientes do Polinômio do Numerador e Denominador da Função de Transferência do Modelo Original, $G_6(s)$.

A redução se deu de uma função originalmente de sexta ordem para terceira ordem, contudo, um dos modelos, o gerado pelo PSO, apresentou um zero a menos do que os gerados pelo FA e pelo SFLA.

$$\hat{G}_{6_{FA}}(\mathrm{s}) = \frac{0{,}02852s^2 + 5{,}836s + 8{,}645}{s^3 + 3{,}516s^2 + 7{,}397s + 6{,}708} \tag{4.39}$$

$$\hat{G}_{6_{PSO}}(\mathrm{s}) = \frac{5{,}311s + 7{,}892}{s^3 + 3{,}074s^2 + 6{,}937s + 6{,}115} \tag{4.40}$$

$$\hat{G}_{6_{SFLA}}(\mathrm{s}) = \frac{0{,}005336s^2 + 5{,}302s + 7{,}978}{s^3 + 3{,}097s^2 + 6{,}965s + 6{,}183} \tag{4.41}$$

Apesar do grau apresentado pela função original, os algoritmos conseguiram reduzir de forma bastante satisfatória, encontrando indivíduos bem próximos para os modelos reduzidos. As respostas ao degrau são mostradas nas Figuras 4.28 e 4.29. O erro máximo para o FA, PSO e SFLA, nesta ordem,

são: 0,0175; 0,0055 e 0,0074, sendo o erro do PSO o menor destes.

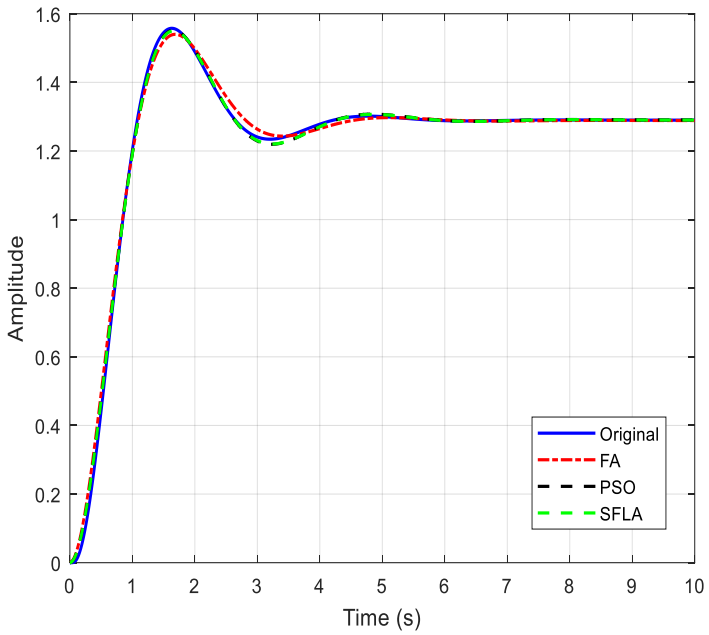

Figura 4.28 - Respostas temporais ao degrau unitário para o modelo original e reduzidos obtidos por IE – Caso 6.

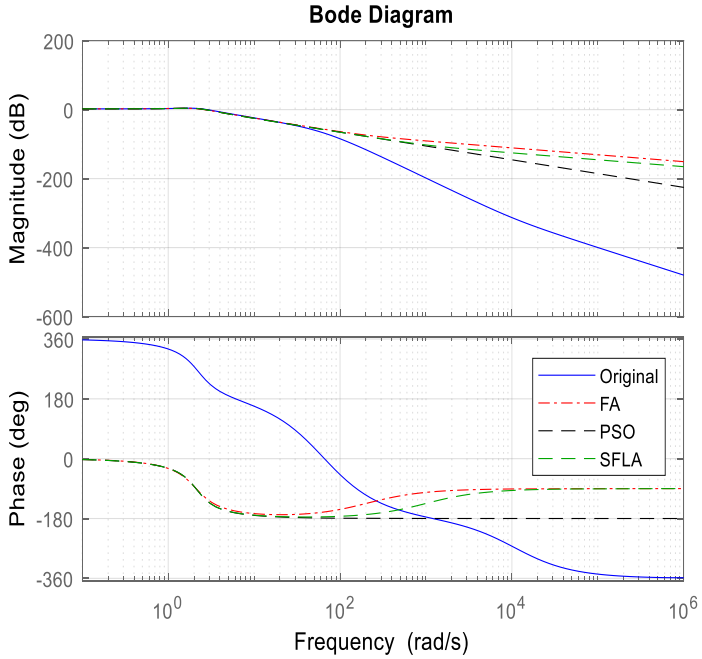

Figura 4.29 - Respostas em frequência (magnitude e fase) do modelo original e dos reduzidos obtidos por IE – Caso 6.

A parametrização para o FA e o PSO ficou em 25 o número de indivíduos que compunham a população, máxima iteração em 200, espaço de busca [0,10] e critério de parada 6e-5, gerando as Tabelas 4.18 e 4.19.

	Iterações	Custo	Tempo (s)
Max	21	59,503e-06	71,383679
Min	5	17,952e-06	16,973042
Média	9	44,8e-06	30,60014627
Desv. Padrão	4,750939757	11,7337e-06	16,1392105

Tabela 4.18 - Dados Fornecidos Pelo FA – Caso 6.

	Iterações	Custo	Tempo (s)
Max	92	10,0e-06	26,6733
Min	25	8,92e-06	7,4732
Média	58,3	9,70e-06	17,02325
Desv. Padrão	22,35595869	0,391637e-06	6,354861007

Tabela 4.19 - Dados Fornecidos Pelo PSO – Caso 6.

Para uma população de 10 indivíduos e critério de parada 1e-5, mantida o restante da parametrização, obteve-se os dados mostrados na Tabela 4.20 para o SFLA.

	Iterações	Custo	Tempo (s)
Max	41	9,99e-06	43,7551
Min	15	7,86e-06	15,9992
Média	24,6	9,47e-06	27,1437
Desv. Padrão	7,275529763	0,712906e-06	7,792024122

Tabela 4.20 - Dados Fornecidos Pelo SFLA – Caso 6.

Em custo, o PSO e o SFLA se equiparam, com desvio padrão pequeno, visto que para o FA o custo se tornou elevado, com um grande desvio padrão. Já para o tempo, o PSO realizou mais iterações em um menor tempo, conforme mostram os gráficos da Figura 4.30.

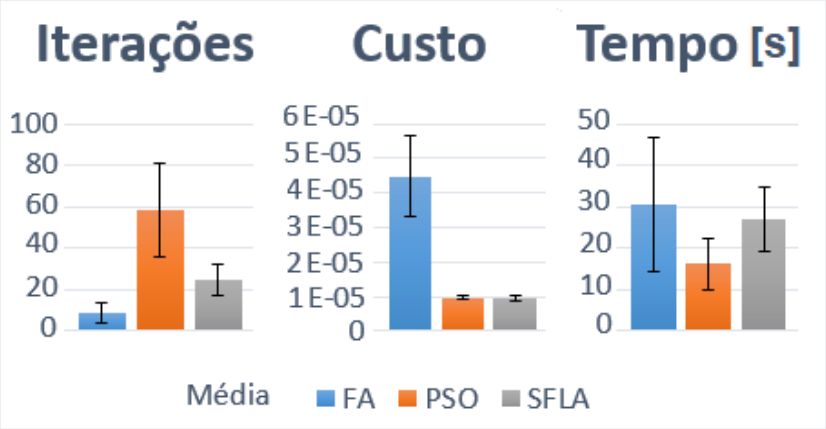

Figura 4.30 - Gráficos realizados no Excel, com base em dez simulações por algoritmo – Caso 6.

4.7- Conclusões

Técnicas para redução da ordem de modelos foram implementadas com o objetivo de estudar como o erro, associado ao processo de redução, é influenciado pelo número de modos truncados e pelo critério de seleção de modos a serem eliminados.

Observou-se que em um sistema de ordem elevada (isto é, n grande e consequentemente com muitos modos) é possível que alguns modos sejam mais importantes para descrever o comportamento do sistema do que outros. Feito isto, uma metodologia para redução de ordem de funções de transferência estáveis de sistemas lineares invariantes no tempo, a parâmetros concentrados, baseada na minimização da norma dos coeficientes do polinômio do numerador do erro foi apresentada (Araújo, 2008), e a aplicação do mesmo em exemplos (estudo de casos) deixou clara a validade do método.

Por fim, os algoritmos bio-inspirados apresentados mostraram-se aptos a efetuar o mesmo procedimento que a técnica clássica determinística, porém dentro do âmbito estocástico, realizando buscas de forma iterativa dentro de um espaço de pesquisa preestabelecido. Por se tratar de uma busca randômica, diversos modelos equivalentes ao original foram encontrados. Com isso, a média entre eles foi retirada, visando a assertividade dos mesmos. Para efeito de comparação, mostrou-se o custo médio, número de iterações e tempo demandado nesta busca, por algoritmo.

CAPÍTULO 5
CONCLUSÃO

Neste trabalho, apresentou-se uma metodologia para realizar a redução de ordem de sistemas dinâmicos, utilizando algoritmos de Inteligência Computacional, para sistemas do tipo SISO. Tal metodologia baseou-se na minimização do erro da resposta a uma entrada do tipo degrau, para o modelo original e para o modelo reduzido. Foi proposto manter-se as características dinâmicas do sistema original. Para avaliar o desempenho da técnica, realizou-se seis estudos de caso. Para avaliação de cada algoritmo utilizado, fez-se uma comparação dos resultados com os obtidos para cada uma das técnicas, em termos de tempo de processamento, número de iterações e o custo obtido. Demonstrou-se que os algoritmos bioinspirados obtiveram um ótimo desempenho e geraram modelos reduzidos com bons resultados.

De tal modo, em consonância com a metodologia clássica de otimização e minimização para problemas que possuem uma dada função objetivo, buscou-se destacar as propriedades estocásticas dos algoritmos de inteligência de enxame. Foram abordados os seus conceitos básicos, além de apresentar a metodologia necessária e a forma de se adaptá-los para a utilização e aplicação na redução da ordem de sistemas dinâmicos lineares, visando eliminar redundâncias do sistema estudado.

Observou-se que os algoritmos de IE mostraram-se tão eficazes quanto os métodos determinísticos, já consagrados pela literatura, levando em consideração que se trata de uma otimização estocástica, isto é, uma classe geral de algoritmos e técnicas que empregam algum grau de aleatoriedade para encontrar soluções tão ótimas quanto possível.

Ainda, desde que devidamente parametrizados, os algoritmos mostraram um maior ou menor esforço computacional, na busca pelos melhores indivíduos que formariam o modelo reduzido, desta forma desprendendo um tempo relativamente maior ou menor na busca dos mesmos, porém com resultados extremamente precisos e erros mínimos.

5.1- Principais Contribuições da Dissertação

De acordo com os resultados obtidos conclui-se que os três algoritmos de Inteligência de Enxame (FA, PSO e SFLA) podem ser usados como uma ferramenta para a obtenção de modelos reduzidos de sistemas do tipo SISO.

5.2 - Publicação Realizada

Os resultados apresentados neste trabalho foram publicados no XIV Congresso Brasileiro de Inteligência Computacional (CBIC, 2019), intitulado “Estudo Comparativo de Técnicas de Inteligência de Enxame na Redução da Ordem de Sistemas Dinâmicos Lineares”, realizado de 03 a 06 de novembro de 2019, em Belém-PA. Cujos autores são: Marlon J. P. Silva, Juan F. Vidal, Carlos T. Costa Jr. e Orlando F. Silva.

Os autores pertencem ao Laboratório de Controle e Sistemas da Faculdade de Engenharia Elétrica e Biomédica, Instituto de Tecnologia, Universidade Federal do Pará - UFPA.

5.3 - Pesquisas Futuras

Como primeira proposta de trabalho futuro tem-se a investigação da possiblidade de generalização de algoritmos bioinspirados capazes de produzir modelos reduzidos multivariáveis (MIMO).

Visto que os estudos de casos apresentados neste trabalho abordaram somente a redução de ordem de sistemas lineares, uma segunda proposta de trabalhos futuros seria testar os diversos algoritmos de enxames para a obtenção de modelos reduzidos aproximados de sistemas não-lineares.

Uma terceira proposta seria investigar a possibilidade de implementação de um algoritmo bioinspirado, que reduza o tempo de busca pelos melhores indivíduos, também pode ser considerado, haja vista que o esforço computacional se torna intenso a medida que a ordem do sistema aumenta.

Por fim, uma quarta sugestão para novas pesquisas, seria investigar o uso da inteligência de enxames aplicada a sistemas fuzzy e redes neurais.

Referências Bibliográficas

Aguirre, L. A., 1993. Quantitative Measure of Modal Dominance for Continuous Systems. Proceedings of the 32nd IEEE Conference on Decision and Control, pp. 2405-2410. San Antonio, Texas, USA.

Aguirre, L. A., 2007. Introdução à Identificação de Sistemas: Técnicas Lineares e Não-Lineares Aplicadas a Sistemas Reais. Editora UFMG 3ª Ed.

Aloise, D. J.; Oliveira, M. C. S.; Silva, T. L., 2005. Otimização Discreta por Nuvem de Partículas Aplicada ao Problema do Caixeiro Viajante. XIII SIMPEP – Bauru, SP.

Aoki, M. (1968). Control of Large Scale Dynamic Systems by Aggregation. IEEE Trans. on A utomatic Control, AC-13, p.246.

Araújo, J. M., 2008. Redução de Ordem no Domínio da Frequência Baseada na Minimização da Norma dos Coeficientes Polinomiais do Erro. Revista Controle e Automação, Vol. 19, No. 3, pp. 235-248.

Arbel, A. e Tse, E. (1979). Reduced Order Models; Canonical Forms and Observers. Int. l. Control, vol.30, p.513.

Assadi, F. and Abut, N., 2010. Method for System Order Reduction Based on Genetic Algorithm and FFT. International Journal of Advanced Research in Electrical Engineering and Technology. Vol. 2, pp. 316-319.

Bansal, J. C.; Sharma, H. and Arya, K. V. (2011). Model Order Reduction of Single Input Single Output Systems Using Artificial Bee Colony Optimization Algorithm. NICSO, SCI 387, pp. 85-100.

Bottura, C. P.; Munaro C. J., 1994. Redução de Ordem de Sistemas Discretos Multivariáveis Via Decomposição de Schur. Revista Controle e Automação, Vol. 04, No. 2, pp. 77-81.

Bouazza, A., 2017. Comparison of Singular Perturbations Approximation Method and Meta-Heuristic-Based Techniques for Order Reduction of Linear Discrete Systems. Applied and Computational Mathematics, Vol. 6, No. 4-1, pp. 48-54.

Chen, C. and Shieh, L. (1968). A Novel approach to Linear Model Simplification. International Journal of Control 8(6), pp. 561-570.

Davison, J. E. (1966). A Method for Simplifying Linear Dynamic Systems. IEE Trans. Automatic Control AC 11 (1), pp. 93-101.

Dawkins, R. ,1976. *The Selfish Gene*, Oxford University Press, Oxford.

Drenick, P.E. (1975). On the Decomposition of State Space, IEEE Trans. on Automatic Control, pp.269-271.

Duan, Q., Sorooshian, S. and Gupta, V., Effective and efficient global optimization for conceptual rainfall-runoff models. Water Resources Res., 1992, 28(4), 1015–1031.

El-Attar, R. A. and Vidyasagar, M., 1978. Order reduction by l_1,− and l_∞, − norm minimization. IEEE Transaction on Automatic Control, Vol. 23, n. 04, pp. 731-734.

Engelbrecht, A. P., 2007. *Computational Intelligence: An Introduction*, John Wiley and Sons Inc.

Esmin, Ahmed Ali Abdalla. Estudo de aplicação do algoritmo de otimização por enxame de partícula na resolução de problemas de otimização ligados ao SEP. Tese (Doutorado). Itajubá: Universidade Federal de Itajubá, 2005.

Eusuff, M.M. and Lansey, K.E., Optimization of water distribution network design using the shuffled frog leaping algorithm (SFLA). J. Water Resources Planning Mgmt, Am. Soc. Civ. Engrs, 2003, 129(3), 210–225.

Eusuff, M.; Lansey, K. E.; & Pasha, F., 2006. Shuffled frog-leaping algorithm: a memetic meta-heuristic for discrete optimization. Engineering Optimization. Vol. 38, No. 2, pp. 129–154.

Ferreira, A. O.; Silva, O. F. e Barreiros, J. A. L., 2011. Uso de Algoritmo Genético para Redução de Ordem de Modelos Matemáticos. X-SBAI- Simpósio Brasileiro de Automação Inteligente, São João del-Rei, MG, Brasil.

Garcia, E. P., Onaka, J. M. D., 2016. Uso de técnicas de otimização por enxame de partículas aplicadas na alocação e dimensionamento ótimo de banco capacitores em sistemas de distribuição de energia elétrica. UFAM, 2016.

Geem, Z. W., Kim, J. H., Loganathan, G. V., 2001, “A new heuristic optimization algorithm: harmony search”, *Simulation*, v. 76, n. 2, pp. 60-68.

Gigerenzer, G.; Wolfgang. Heuristic Decision Making. Annual Review of Psychology, 62, 2011. 451-482.

Goldbarg E., Goldbarg, M. e Luna, H., 2017. Otimização Combinatória e Metaheurísticas: Algoritmos e Aplicações. Editora: Elsevier Brasil.

Hachino, T.; Tanigawa, K.; Takata, H.; Fukushina, S. and Igarashi, Y., 2015. Frequency-Weighted Model Reduction Using Firefly Algorithm. Journal of Automation and Control Engineering, vol. 3, No. 3.

Hsia, T.C., 1972. On the simplification of linear systems. IEEE Transaction on Automatic Control, Vol. 17, n. 04, pp. 372-374.

Hutton, M. and Friedland, B. (1975). Routh approximation for Reducing Order of Linear Time Invariant Systems. IEEE Transaction on Automatic Control 20(3), pp. 329-337.

Jamshidi, M. (1983) Large Scale Systems Modeling and Control, North Rolland, New York, pp.23-30.

Jamshidi M, 2003. Tools for intelligent control: fuzzy controllers, neural networks and genetic algorithms.

Kennedy, J.;Eberhart, R. C., 1995. Particle Swarm Optimization. Proceedings of IEEE International Conference on Neural Networks. Piscataway, NJ, USA. pp. 1942–1948.

Kruse, R., Borgelt, C., Klawonn, F., Moewes, C., Steinbrecher, M., & Held, P. Computational intelligence: a methodological introduction. Springer Science & Business Media. 2013.

Liden, R. Algoritmos Genéticos. 2a Ed. Rio de Janeiro: Brasport, 2008. ISBN 8574523739.

Lopes, H. S. & Takahashi, R. H. C., 2011. Computação Evolucionária em Problemas de Engenharia. Curitiba, PR. Editora Omnipax.

Mansour, M. M. and Mehrotra, A., 2003. Model-Order reduction based on PRONY's method. Proceedings of the Design, Automation and Test in Europe Conference and Exhibition.

Marella, T.; Anand, N. V. and Kumar, M. S., 2014. Order Reduction of MIMO System using Firefly Algorithm. International Journal of Electrical Engineering, vol 7, No. 3, pp. 425-438.

Marshall, S.A. (1966). An Approximate Method for Reducing the Order of a Linear System. Control, 10, p.642.

Moore, B. C., 1981. Principal component analysis in linear systems: controlability, observability and model reduction. IEEE Transaction on Automatic Control, Vol. 26, no 01, pp. 17-32.

Muscato, G., 2000. Parametric generalized singular perturbation approximation for model order reduction. IEEE Transactions On Automatic Control, Vol. 45, n. 02, pp. 339-343.

Moscato, P. 1989. On evolution, search, optimization, genetic algorithms and martial arts: towards memetic algorithms. Caltech Concurrent Computation Program 158-79, California Institute of Technology, Pasadena, CA, 1989.

Nadi, D. A. A.; Alsmadi, O. M. and Hammour, Z. S. A., 2011. Reduced Order Modeling of Linear MIMO Systems using Particle Swarm Optimization. International Conference on Automatic and Autonomous Systems.

Parmar, G. Prassad, R. Mukherjee, S. (2007). Order Reduction of Linear Dynamic Systems Using Stability Equation Method and Genetic Algorithm. International Journal of Computer, Information and System Science, pp. 372-378.

Paynter, H. M. and Takahashi, Y. (1956). A new Method of Evaluation Dynamic Response of Counter Flow and Parallel Flow Heat exchangers. Trans. ASME J. Dynam. Syst. Meas. Control 27, pp. 749-753.

Pena, R. T; Aguirre, L. A.; Mendes, E. M. A. M., 1990. A four Fuel Drum Boiler Combustion Control System Study and Redesign, IEEE 29th Conference on Decision and Control, Hawaii, pp. 1567-1572.

Pereira, J. P. G., 2007. Heurísticas computacionais aplicadas à otimização estrutural de treliças bidimensionais. Dissertação (Mestrado) – Diretoria de Pesquisa e Pós-Graduação, Centro Federal de Educação Tecnológica de Minas Gerais, Belo Horizonte, 2007.

Pernebo, L. and Silverman, 1982. Model reduction via balanced state-space representations. IEEE Transaction on Automatic Control, Vol. 27, n. 02, pp. 382-387.

Ribeiro, L. A. D., 2014. Otimização Estrutural de Treliças Utilizando o Algoritmo Firefly. Universidade Federal de Santa Catarina. Florianópolis – SC. Brasil.

Rosendo, M., 2010. Um Algoritmo De Otimização Por Nuvem De Partículas Para Resolução De Problemas Combinatórios. Dissertação (Mestrado) - Universidade Federal do Paraná, Curitiba.

Saini, D. K. and Prasad, R. (2010). Order Reduction of Linear Interval Systems Using Genetic Algorithm. International Journal of Engineering and Technology, Vol. 2, No. 5, pp. 316-319.

Sambariya, D. K. and Shama, O., 2016. Model Order Reduction Using Routh Approximation and Cuckoo Search Algorithm. Journal of Automation and Control, Vol. 4, No. 1, pp. 1-9.

Serapião, A. B., & Rocha, R. K (2012). Algoritmos de otimização bioinspirados baseados em populações para o problema de despacho econômico de carga. Anais do XIX Congresso Brasileiro de Automática.

Shi, Y.; Eberhart, R.; 1998. Parameter selection in particle swarm optimization. In Evolutionary Programming VIZ: Proc. EP98, New York: Springer Verlag pp. 591-600.

Silva, M. J. P.; Vidal, J. F.; Costa JR, C. T. e Silva, O. F., 2017. Redução de Ordem de Sistemas Dinâmicos Utilizando Inteligência Computacional – Uma Abordagem via Firefly Algorithm. XIII-SBAI – Simpósio Brasileiro de Automação Inteligente, Porto Alegre, RS, Brasil.

Silva, A. F. Lemonge, A. C. C. e Lima, B. S. L. P.; 2014. Algoritmo de Otimização com Enxame de Partículas Auxiliado por Metamodelos. XI Simpósio de Mecânica Computacional – SIMMEC/EMMCOMP 2014.

Souza, D. L. 2014. Otimização por Multi-Enxame Evolucionário de Partículas Clássico e Quântico Competitivo sob a Auquitetura Paralela CUDA Aplicado em Problemas de Engenharia. UFPA, 2014.

Tavakolan M.; Baabak, A. and Chiara, N.; 2011. “Applying the Shuffled Frog-Leaping Algorithm to improve scheduling of construction projects with activity splitting allowed,” Management and Innovation for a Sustainable Built Environment, pp. 20-23.

Trivelato, G. da C., 2003. Técnicas de Modelagem e Simulação de Sistemas Dinâmicos. Instituto Nacional de Pesquisas Espaciais, INPE-9665-NTC/358. pp. 06.

Tse, E. C. Y.; J. V. Medanic e Perkins, W.R; 1977. Chained Aggregation of Linear Time Invariant Systems. Proc. lACe, San Francisco.

Vasu, G.; Namratha, J. N. and Murari, P., 2012. Order Reduction of Linear Dynamic Systems Using improved Generalise Least-Squares Method and PSO. International Journal of Engineering Research and Application. Vol. 2, pp. 581-587.

Wang, K.-P.; Huang, L.; Zhou, C.-G.; Pang, W. Particle swarm optimization for traveling salesman problem. In: Proceedings of the 2003 International Conference on Machine Learning and Cybernetics. 2003. p. 1583–1585.

White, T.; Pagurek, B., 1998. Towards multi-swarm problem solving in networks. In: Proceedings of Third International Conference on Multi-Agent Systems (ICMAS’98. IEEE Computer Society, 1998. p. 333–340.

Wilson, R.G.; Fisher, D.G. e Seborg, D.E., (1972). Model Reduction For Discrete Time Systems. Int. J. Control, vol.16, p.519.

Xing, B. and Gao, W. J., 2014. Innovative Computational Intelligence: A Rough Guide to 134 Clever Algorithms. Springer International Publishing Switzerland.

Xingyu, T.; Heng, L., 2019. Developing Shuffled Frog-Leaping Algorithm (SFLA) Method to Solve Power Load-Constrained TCRTO Problems. Hindawi Advances in Civil Engineering. Vol 2019, Article ID 1404636, pp. 16.

Yang, X. S., 2008. Nature-Inspired Metaheuristic Algorithm. UK: Luniver Press. ISBN:978-1-905986-28-6.

Zadeh, Lotfi A., "Fuzzy Logic, Neural Networks, and Soft Computing," Communications of the ACM, March 1994, Vol. 37 No. 3, pages 77-84.

Zhu, Yan-fei; Tang, Xiong-min. Overview of Swarm Intelligence. In: International Conference on Computer Application and System Modeling (ICCASM 2010), Taiyuan, v. 9, p. 400-403, 2010.

Zuben, F. J. V.; Attux, R. R. F. Inteligência de Enxame. DCA/FEEC/Unicamp e DECOM/FEEC/Unicamp, 2008.

ANEXO I

Detalhamento e estudo dos cálculos referentes à expansão em frações parciais com linhas de comando em Matlab para determinação dos IDM's.

- **Polos simples**

$$\frac{18S+60}{(S+2)(S+5)(S+10)} = \frac{A_{10}}{(S+2)} + \frac{A_{20}}{(S+5)} + \frac{A_{30}}{(S+10)}$$

$$A_{10} = \left.\frac{(18S+60)(S+2)}{(S+2)(S+5)(S+10)}\right|_{S=-2} = 1$$

$$A_{20} = \left.\frac{(18S+60)(S+5)}{(S+2)(S+5)(S+10)}\right|_{S=-5} = 2$$

$$A_{30} = \left.\frac{(18S+60)(S+10)}{(S+2)(S+5)(S+10)}\right|_{S=-10} = -3$$

$$H(S) = \frac{18S+60}{(S+2)(S+5)(S+10)} = \frac{1}{(S+2)} + \frac{2}{(S+5)} - \frac{3}{(S+10)}$$

- **Um polo simples, um polo múltiplo**

$$H(S) = \frac{S+3}{(S+5)(S+2)^2} = \frac{A_{10}}{(S+5)} + \frac{A_{20}}{(S+2)^2} + \frac{A_{21}}{(S+2)}$$

$$A_{10} = \left.\frac{(S+3)(S+5)}{(S+5)(S+2)^2}\right|_{S=-5} = -\frac{2}{9}$$

$$A_{20} = \left.\frac{(S+3)(S+2)^2}{(S+5)(S+2)^2}\right|_{S=-2} = \frac{1}{3}$$

$$A_{21} = \left.\frac{d^1}{1!\,dS^1}\left[\frac{(S+3)(S+2)^2}{(S+5)(S+2)^2}\right]\right|_{S=-2} = \frac{2}{9}$$

$$H(S) = \frac{S+3}{(S+5)(S+2)^2} = \frac{-2/9}{(S+5)} + \frac{1/3}{(S+2)^2} + \frac{2/9}{(S+2)}$$

- **Um polo simples, um par de polos complexos conjugados**

$$H(S) = \frac{S+3}{(S+2)(S^2+2S+2)} = \frac{A}{(S+2)} + \frac{BS+C}{(S^2+2S+2)}$$

$$A = \left.\frac{(S+3)(S+2)}{(S+2)(S^2+2S+2)}\right|_{S=-2} = \frac{1}{2}$$

Para a determinação dos resíduos B e C, tem-se: $H(S)(S^2+2S+2)\,|_{S=-1-j} =$

$$= \left.\frac{A(S^2+2S+2)}{(S+2)}\right|_{S=-1-j} + \frac{BS+C}{(S^2+2S+2)}(S^2+2S+2)\,|_{S=-1-j}$$

$$= B(-1-j)+C$$

$$\left.\frac{(S+3)(S^2+2S+2)}{(S+2)(S^2+2S+2)}\right|_{S=-1-j} = \frac{2-j}{1-j} = -B-jB+C \;\; \therefore \; (2-j)$$

$$= (C-B-jB)(1-j)$$

$$\begin{cases} -2B+C=2 \\ -C=-1 \end{cases} \therefore \begin{cases} B=-0{,}5 \\ C=1 \end{cases}, \; \therefore H(S) = \frac{0{,}5}{S+2} + \frac{-0{,}5S+1}{(S+1)^2+1} = \frac{-0{,}5(S+1)+1{,}5}{(S+1)^2+1}$$

- **Dois polos simples, um par de polos conjugados, um polo duplo**

$$H(S) = \frac{19S^5+311S^4+1981S^3+6121S^2+9152S+5120}{(S+1)(S+2)(S^2+6S+10)(S+5)^2}$$

$$H(S) = \frac{A}{(S+1)} + \frac{B}{(S+2)} + \frac{CS+D}{(S+3+j)} + \frac{E_0}{(S+5)^2} + \frac{E_1}{(S+5)}$$

$$A = H(S)(S+1)\,|_{S=-1} = 5$$

$$B = H(S)(S+2)\,|_{S=-2} = 10$$

$$H(S)(S+6S+10)\,|_{S=-3+j} = (CS+D)\,|_{S=-3+j}$$

$$-7+3j = -3C+jC+D \rightarrow \begin{cases} -7=3C+D \\ 3=C \end{cases} \therefore \begin{cases} C=3 \\ D=2 \end{cases}$$

$$E_0 = H(S)(S+5)^2\,|_{S=-5} = -4$$

$$E_1 = \frac{d}{dS}(H(S)(S+5)^2)\,|_{S=-5} = 1$$

$$H(S) = \frac{5}{(S+1)} + \frac{10}{(S+2)} + \frac{3S+2}{(S^2+6S+10)} - \frac{4}{(S+5)^2} + \frac{1}{(S+5)}$$

- **Expansão em Frações Parciais com uso do Matlab**

Dada uma Função de Transferência qualquer, a determinação de seus resíduos, polos e ganho, são dados como no exemplo:

```
=========================================================================
num=[2 5 3 6];
den=[1 6 11 6];
[r,p,k]=residue(num,den)
=========================================================================
```

- **Utilização dos Índices de Dominância Modais**

Para a determinação destes e consequente montagem das tabelas dos IDM's, tem-se o exemplo:

```
=========================================================================
num=1;
den=[1 202.3 10260.62 3064.04 204];
g=tf(num,den);
[r,p,k]=residue(num,den);
idm=-[r(1)/p(1) r(2)/p(2) r(3)/p(3) r(4)/p(4)];
idmperc=abs(-[r(1)/p(1) r(2)/p(2) r(3)/p(3) ...
r(4)/p(4)])*100/(abs(r(1)/p(1))+abs(r(2)/p(2))+abs(r(3)/p(3)) ...
+abs(r(4)/p(4)));
[p r idm' idmperc']
=========================================================================
```

ANEXO II

Linhas de comando no Matlab para os exemplos mostrados nos estudos de casos do Capítulo 4, para determinação do sistema reduzido por meio da ROMNCPE.

==========

- **Algorithm 01:**

```
%_____________________________________________________________________________
%              Reduction by ROMNCPE [MIN 10 To 1]
%              Date Created: Jun 17, 2017
%              By: Silva, Marlon J. P.
%              Electrical Engineer (UFPA)
%              Master in Electrical Engineering (UFPA)
%              Ver. 3.0
%_____________________________________________________________________________

clear, clc, close all

%Case 1:-------------------------------------------------------------------
num=[0 0 0 0 0 0 0 4 28 68 60];                 %Enter eleven elements
den=[0 0 0 0 0 0 1 8 24 32 15];                 %Enter eleven elements
%--------------------------------------------------------------------------


%_____________________________________________________________________________
%                      Original Transfer Function
%_____________________________________________________________________________

Go=tf(num,den)

%_____________________________________________________________________________
%                           Function f(a,b)
%_____________________________________________________________________________

syms a b

f = (num(1)                          )^2 ...
   +(num(2)  +num(1)*a  -den(1)*b)^2 ...
   +(num(3)  +num(2)*a  -den(2)*b)^2 ...
   +(num(4)  +num(3)*a  -den(3)*b)^2 ...
   +(num(5)  +num(4)*a  -den(4)*b)^2 ...
   +(num(6)  +num(5)*a  -den(5)*b)^2 ...
   +(num(7)  +num(6)*a  -den(6)*b)^2 ...
   +(num(8)  +num(7)*a  -den(7)*b)^2 ...
   +(num(9)  +num(8)*a  -den(8)*b)^2 ...
   +(num(10) +num(9)*a  -den(9)*b)^2 ...
   +(num(11)+num(10)*a -den(10)*b)^2 ...
   +(         num(11)*a -den(11)*b)^2;
```

```
%_____________________________________________________________________
%                      Derivate Function f(a,b)
%_____________________________________________________________________

da=diff(f,a);
db=diff(f,b);

[a,b]=solve(da,db);
x=double(a);
y=double(b);

%_____________________________________________________________________
%                      Reduced Transfer Function
%_____________________________________________________________________

Gr=tf([0 y],[1 x])  %Função Reduzida para 1a Ordem do tipo: Gr=b/(s+a)

erro=Go-Gr

%_____________________________________________________________________
%                            Parameters
%_____________________________________________________________________

SimTime=0:0.01:10;               %Vector simulation time of the System
Rs=ones(1,length(SimTime));

yo=lsim(Go,Rs,SimTime);          %System response to a unit-type input
yr=lsim(Gr,Rs,SimTime);

ye=lsim(erro,Rs,SimTime);

%_____________________________________________________________________
%                             Graphics
%_____________________________________________________________________

figure(1)
plot(SimTime,yo,'b','linewidth',1.5)
hold on
plot(SimTime,yr,'r-.','linewidth',1.5)
grid
legend('Original','Reduzida'), title('Step Response'),
xlabel('Time'), ylabel('Amplitude')
hold off


figure(2)
plot(SimTime,ye,'b','linewidth',1.5)
grid
legend('y_e_r_r_o'), title('Error Step Response'),
xlabel('Time'), ylabel('Amplitude')


figure(3)
bode(Go)
hold on
bode(Gr)
grid
legend('Original','Reduzida')
```

```
hold off
```

==========

- **Algorithm 02:**

```
%_____________________________________________________________________
%              Reduction by ROMNCPE [MIN 10 To 2]
%              Date Created: Jun 17, 2017
%              By: Silva, Marlon J. P.
%              Electrical Engineer (UFPA)
%              Master in Electrical Engineering (UFPA)
%              Ver. 3.0
%_____________________________________________________________________

clear, clc, close all

%Case 2:-------------------------------------------------------------
num=[0 0 0 0 0 0 0 0 0 0 1];                    %Enter eleven elements
den=[0 0 0 0 0 0 0 1 3.09 3.179 1.089];         %Enter eleven elements
%--------------------------------------------------------------------

%Case 3:-------------------------------------------------------------
%num=[0 0 0 0 0 0 0 0 0 1.3 1];                 %Enter eleven elements
%den=[0 0 0 0 0 0 0 0 1 0.4 2];                 %Enter eleven elements
%--------------------------------------------------------------------

%Case 4:-------------------------------------------------------------
%num=[0 0 0 0 0 0 1 6 96 780 3250];             %Enter eleven elements
%den=[0 0 0 0 1 13.2 158.6 594 2765 1050 2500]; %Enter eleven elements
%--------------------------------------------------------------------

%_____________________________________________________________________
%                      Original Transfer Function
%_____________________________________________________________________

Go=tf(num,den)

%_____________________________________________________________________
%                        Function f(a,b,c,d)
%_____________________________________________________________________

syms a b c d

f = (num(1)                                                  )^2 ...
   +(num(2)  +num(1)*a             -den(1)*c                 )^2 ...
   +(num(3)  +num(2)*a  +num(1)*b  -den(2)*c  -den(1)*d      )^2 ...
   +(num(4)  +num(3)*a  +num(2)*b  -den(3)*c  -den(2)*d      )^2 ...
   +(num(5)  +num(4)*a  +num(3)*b  -den(4)*c  -den(3)*d      )^2 ...
   +(num(6)  +num(5)*a  +num(4)*b  -den(5)*c  -den(4)*d      )^2 ...
   +(num(7)  +num(6)*a  +num(5)*b  -den(6)*c  -den(5)*d      )^2 ...
   +(num(8)  +num(7)*a  +num(6)*b  -den(7)*c  -den(6)*d      )^2 ...
   +(num(9)  +num(8)*a  +num(7)*b  -den(8)*c  -den(7)*d      )^2 ...
   +(num(10) +num(9)*a  +num(8)*b  -den(9)*c  -den(8)*d      )^2 ...
   +(num(11) +num(10)*a +num(9)*b  -den(10)*c -den(9)*d      )^2 ...
   +(        +num(11)*a +num(10)*b -den(11)*c -den(10)*d)^2 ...
   +(                   +num(11)*b             -den(11)*d)^2 ;
```

```
%_____________________________________________________________________
%                      Derivate Function f(a,b,c,d)
%_____________________________________________________________________

da=diff(f,a);
db=diff(f,b);
dc=diff(f,c);
dd=diff(f,d);

[a,b,c,d]=solve(da,db,dc,dd);
v=double(a);
x=double(b);
y=double(c);
z=double(d);

%_____________________________________________________________________
%                      Reduced Transfer Function
%_____________________________________________________________________

Gr=tf([0 y z],[1 v x])    %Função Reduzida para 2a Ordem do tipo: Gr =
(cs + d) / (s^2 + as + b)

erro=Go-Gr

%_____________________________________________________________________
%                            Parameters
%_____________________________________________________________________

SimTime=0:0.01:10;               %Vector simulation time of the System
Rs=ones(1,length(SimTime));

yo=lsim(Go,Rs,SimTime);          %System response to a unit-type input
yr=lsim(Gr,Rs,SimTime);

ye=lsim(erro,Rs,SimTime);

%_____________________________________________________________________
%                             Graphics
%_____________________________________________________________________

figure(1)
plot(SimTime,yo,'b','linewidth',1.5)
hold on
plot(SimTime,yr,'r-.','linewidth',1.5)
grid
legend('Original','Reduzida'), title('Step Response'),
xlabel('Time'), ylabel('Amplitude')
hold off

figure(2)
plot(SimTime,ye,'b','linewidth',1.5)
grid
legend('y_e_r_r_o'), title('Error Step Response'),
xlabel('Time'), ylabel('Amplitude')

figure(3)
bode(Go)
hold on
bode(Gr)
```

```
grid
legend('Original','Reduzida')
hold off
```

==========

- **Algorithm 03:**

```
%_____________________________________________________________________
%              Reduction by ROMNCPE [MIN 10 To 3]
%              Date Created: Jun 17, 2017
%              By: Silva, Marlon J. P.
%              Electrical Engineer (UFPA)
%              Master in Electrical Engineering (UFPA)
%              Ver. 3.0
%_____________________________________________________________________

clear, clc, close all

%Case 5:-------------------------------------------------------------
%Enter eleven elements
num=[0 0 0 0 0 1 72639 331114 978725 1.816*10^6 907200];
%Enter eleven elements
den=[0 0 1 11233 82957 403753 1.32*10^6 2.794*10^6 3.742*10^6
2.839*10^6 907210];
%--------------------------------------------------------------------

%_____________________________________________________________________
%                       Original Transfer Function
%_____________________________________________________________________

Go=tf(num,den)

%_____________________________________________________________________
%                       Function fc(a,b,c,d,e,f)
%_____________________________________________________________________

syms a b c d e f

fc = (num(1)
)^2 ...
    +(num(2)  +a*num(1)                         -d*den(1)
)^2 ...
    +(num(3)  +a*num(2)  +b*num(1)              -d*den(2)  -e*den(1)
)^2 ...
    +(num(4)  +a*num(3)  +b*num(2)  +c*num(1)   -d*den(3)  -e*den(2)  -
f*den(1) )^2 ...
    +(num(5)  +a*num(4)  +b*num(3)  +c*num(2)   -d*den(4)  -e*den(3)  -
f*den(2) )^2 ...
    +(num(6)  +a*num(5)  +b*num(4)  +c*num(3)   -d*den(5)  -e*den(4)  -
f*den(3) )^2 ...
    +(num(7)  +a*num(6)  +b*num(5)  +c*num(4)   -d*den(6)  -e*den(5)  -
f*den(4) )^2 ...
    +(num(8)  +a*num(7)  +b*num(6)  +c*num(5)   -d*den(7)  -e*den(6)  -
f*den(5) )^2 ...
    +(num(9)  +a*num(8)  +b*num(7)  +c*num(6)   -d*den(8)  -e*den(7)  -
f*den(6) )^2 ...
```

```
    +(num(10) +a*num(9)  +b*num(8)  +c*num(7)  -d*den(9)  -e*den(8)  -
f*den(7) )^2 ...
    +(num(11) +a*num(10) +b*num(9)  +c*num(8)  -d*den(10) -e*den(9)  -
f*den(8) )^2 ...
    +(        +a*num(11) +b*num(10) +c*num(9)  -d*den(11) -e*den(10) -
f*den(9) )^2 ...
    +(                   +b*num(11) +c*num(10)            -e*den(11) -
f*den(10))^2 ...
    +(                              +c*num(11)                       -
f*den(11))^2 ;


%_____________________________________________________________________
%                     Derivate Function fc(a,b,c,d,e,f)
%_____________________________________________________________________

da=diff(fc,a);
db=diff(fc,b);
dc=diff(fc,c);
dd=diff(fc,d);
de=diff(fc,e);
df=diff(fc,f);

[a,b,c,d,e,f]=solve(da,db,dc,dd,de,df);

syms z z1

   a=subs(a,z,0);
   a=subs(a,z1,0);

   b=subs(b,z,0);
   b=subs(b,z1,0);

   c=subs(c,z,0);
   c=subs(c,z1,0);

   d=subs(d,z,0);
   d=subs(d,z1,0);

   e=subs(e,z,0);
   e=subs(e,z1,0);

   f=subs(f,z,0);
   f=subs(f,z1,0);

t=double(a);
u=double(b);
v=double(c);
w=double(d);
x=double(e);
y=double(f);

%_____________________________________________________________________
%                     Reduced Transfer Function
%_____________________________________________________________________

Gr=tf([0 w x y],[1 t u v])           %Função Reduzida para 3a Ordem do
tipo: Gr = (ds^2 + es + f) / (s^3 + as^2 + bs + c)
```

```
erro=Go-Gr

%_____________________________________________________________________________
%                              Parameters
%_____________________________________________________________________________

SimTime=0:0.01:10;                %Vector simulation time of the System
Rs=ones(1,length(SimTime));

yo=lsim(Go,Rs,SimTime);           %System response to a unit-type input
yr=lsim(Gr,Rs,SimTime);

ye=lsim(erro,Rs,SimTime);

%_____________________________________________________________________________
%                               Graphics
%_____________________________________________________________________________

figure(1)
plot(SimTime,yo,'b','linewidth',1.5)
hold on
plot(SimTime,yr,'r-.','linewidth',1.5)
grid
legend('Original','Reduzida'), title('Step Response'),
xlabel('Time'), ylabel('Amplitude')
hold off

figure(2)
plot(SimTime,ye,'b','linewidth',1.5)
grid
legend('y_e_r_r_o'), title('Error Step Response'),
xlabel('Time'), ylabel('Amplitude')

figure(3)
bode(Go)
hold on
bode(Gr)
grid
legend('Original','Reduzida')
hold off
```

==========

- **Algorithm 04:**

```
%_____________________________________________________________________________
%              Reduction by ROMNCPE [MIN 10 To 5]
%              Date Created: Jun 17, 2017
%              By: Silva, Marlon J. P.
%              Electrical Engineer (UFPA)
%              Master in Electrical Engineering (UFPA)
%              Ver. 3.0
%_____________________________________________________________________________
```

```
clear, clc, close all

%Case 6:-------------------------------------------------------------
num=[-2298 -9.845e04 -1.376e06 -6.838e06 -6.1e06 -5.43e05];
den=[1 64.21 1596 1.947e04 1.268e05 5.036e05 1.569e06 3.24e06 4.061e06
2.905e06 2.531e05];
%--------------------------------------------------------------------

%____________________________________________________________________
%                      Original Transfer Function
%____________________________________________________________________

Go=tf(num,den)

%____________________________________________________________________
%                      Function f(a,b,c,d)
%____________________________________________________________________

syms a b c d e f g h i j

ff = (
-den(1)*f                                                 )^2 ...
    +(
-den(2)*f  -den(1)*g                                      )^2 ...
    +(
-den(3)*f  -den(2)*g  -den(1)*h                           )^2 ...
    +(
-den(4)*f  -den(3)*g  -den(2)*h  -den(1)*i                )^2 ...
    +(num(1)
-den(5)*f  -den(4)*g  -den(3)*h  -den(2)*i  -den(1)*j )^2 ...
    +(num(2)  +num(1)*a
-den(6)*f  -den(5)*g  -den(4)*h  -den(3)*i  -den(2)*j )^2 ...
    +(num(3)  +num(2)*a  +num(1)*b
-den(7)*f  -den(6)*g  -den(5)*h  -den(4)*i  -den(3)*j )^2 ...
    +(num(4)  +num(3)*a  +num(2)*b  +num(1)*c
-den(8)*f  -den(7)*g  -den(6)*h  -den(5)*i  -den(4)*j )^2 ...
    +(num(5)  +num(4)*a  +num(3)*b  +num(2)*c  +num(1)*d
-den(9)*f  -den(8)*g  -den(7)*h  -den(6)*i  -den(5)*j )^2 ...
    +(num(6)  +num(5)*a  +num(4)*b  +num(3)*c  +num(2)*d  +num(1)*e  -
den(10)*f  -den(9)*g  -den(8)*h  -den(7)*i  -den(6)*j )^2 ...
    +(        +num(6)*a  +num(5)*b  +num(4)*c  +num(3)*d  +num(2)*e  -
den(11)*f -den(10)*g  -den(9)*h  -den(8)*i  -den(7)*j )^2 ...
    +(                   +num(6)*b  +num(5)*c  +num(4)*d  +num(3)*e
-den(11)*g -den(10)*h  -den(9)*i  -den(8)*j )^2 ...
    +(                             +num(6)*c  +num(5)*d  +num(4)*e
-den(11)*h -den(10)*i  -den(9)*j )^2 ...
    +(                                        +num(6)*d  +num(5)*e
-den(11)*i -den(10)*j )^2 ...
    +(                                                   +num(6)*e
-den(11)*j )^2 ;


%____________________________________________________________________
%                      Derivate Function f(a,b,c,d)
%____________________________________________________________________

da=diff(ff,a);
db=diff(ff,b);
dc=diff(ff,c);
dd=diff(ff,d);
```

```
de=diff(ff,e);
df=diff(ff,f);
dg=diff(ff,g);
dh=diff(ff,h);
di=diff(ff,i);
dj=diff(ff,j);

[a,b,c,d,e,f,g,h,i,j]=solve(da,db,dc,dd,de,df,dg,dh,di,dj);

q=double(a);
r=double(b);
s=double(c);
t=double(d);
u=double(e);
v=double(f);
w=double(g);
x=double(h);
y=double(i);
z=double(j);

%_____________________________________________________________________________
%                      Reduced Transfer Function
%_____________________________________________________________________________

Gr=tf([0 v w x y z],[1 q r s t u])            %Função Reduzida para 5a
Ordem do tipo: Gr = (fs^4 + gs^3 + hs^2 + is + j) / (s^5 + as^4 + bs^3
+ cs^2 + ds + e)

erro=Go-Gr

%_____________________________________________________________________________
%                             Parameters
%_____________________________________________________________________________

SimTime=0:0.01:10;               %Vector simulation time of the System
Rs=ones(1,length(SimTime));

yo=lsim(Go,Rs,SimTime);          %System response to a unit-type input
yr=lsim(Gr,Rs,SimTime);

ye=lsim(erro,Rs,SimTime);

%_____________________________________________________________________________
%                              Graphics
%_____________________________________________________________________________

figure(1)
plot(SimTime,yo,'b','linewidth',1.5)
hold on
plot(SimTime,yr,'r-.','linewidth',1.5)
grid
legend('Original','Reduzida'), title('Step Response'),
xlabel('Time'), ylabel('Amplitude')
hold off

figure(2)
plot(SimTime,ye,'b','linewidth',1.5)
grid
legend('y_e_r_r_o'), title('Error Step Response'),
```

```
xlabel('Time'), ylabel('Amplitude')

figure(3)
bode(Go)
hold on
bode(Gr)
grid
legend('Original','Reduzida')
hold off
```

ANEXO III

Linhas de comando no Matlab para os exemplos mostrados nos estudos de casos do Capítulo 4, para determinação do sistema reduzido por meio do FA.

==========

- **Main Algorithm:**

```
%_____________________________________________________________________
%              Reduction by FireFly Algorithm
%              Date Created: April 07, 2017
%              By: Silva, Marlon J. P.
%              Electrical Engineer (UFPA)
%              Master in Electrical Engineering (UFPA)
%              Ver. 3.1
%_____________________________________________________________________


%PARTE I

clear, clc, close all

type=1;   %Reduction Order

%Caso 1:-------------------------------------------------------------
num=[4 28 68 60]; den=[1 8 24 32 15];
%--------------------------------------------------------------------

%Caso 2:-------------------------------------------------------------
%num=[1]; den=[1 3.09 3.179 1.089];
%--------------------------------------------------------------------

%Caso 3:-------------------------------------------------------------
%num=[1.3 1]; den=[1 0.4 2];
%--------------------------------------------------------------------

%Caso 4:-------------------------------------------------------------
%num=[1 6 96 780 3250]; den=[1 13.2 158.6 594 2765 1050 2500];
%--------------------------------------------------------------------

%Caso 5:-------------------------------------------------------------
%num=[1 72639 331114 978725 1.816e6 907200]; den=[1 11233 82957 403753
1.32e6 2.794e6 3.742e6 2.839e6 907210];
%--------------------------------------------------------------------

%Caso 6:-------------------------------------------------------------
%num=[-2298 -9.845e04 -1.376e06 -6.838e06 -6.1e06 -5.43e05]; den=[1
64.21 1596 1.947e04 1.268e05 5.036e05 1.569e06 3.24e06 4.061e06
2.905e06 2.531e05];
%--------------------------------------------------------------------

Gs=tf(num,den)
```

```
%_____________________________________________________________________
%     Configuração dos parametros para gerar a resposta do sistema
%_____________________________________________________________________

simTime=0:0.01:10;
Rs=ones(1,length(simTime));
y=lsim(Gs,Rs,simTime);

%----------------------------(FIM!! PARTE I)--------------------------

%PARTE II
%_____________________________________________________________________
%                     Parametros de Redução
%_____________________________________________________________________

erro=6e-5;                                              %Erro máximo
admitido
CostFunction=Create_CostFunction(Rs,simTime,y,type)         %Função
Custo
nVar=Number_Decision_Variables(type);     %Número de variáveis de
decisão
VarSize=[1 nVar];              %Tamanho da Matriz de variáveis de
decisão
VarMin=0;                       %Limite inferior das variáveis de
decisão
%VarMin=-100;                 %Habilitar este valor somente para o
Caso 6
VarMax=5;                       %Limite superior das variáveis de
decisão
%VarMax= 100;                 %Habilitar este valor somente para o
Caso 6

%_____________________________________________________________________
%                   Parametros do Algoritmo Firefly
%_____________________________________________________________________

MaxIt=100;                                      %Número Máximo de
Iterações   %MaxIt=800;                    Habilitar este valor
somente para o Caso 6
nPop=25;                          %Número de Fireflies (Tamanho do
Enxame)
gamma=0.4;                                %Coeficiente de Absorção da
Luz
beta0=2;                             %Valor base do coeficiente de
atração
alpha=0.2;                                          %Coeficiente de
Mutação
alpha_damp=0.98;         %Taxa de amortecimento do coeficiente de
Mutação
delta=0.05*(VarMax-VarMin);                    %Faixa uniforme de
Mutação
m=2;

%---------------------------------------------------------------------

if isscalar(VarMin) && isscalar(VarMax)
    dmax=(VarMax-VarMin)*sqrt(nVar);
else
    dmax=norm(VarMax-VarMin);
end
```

```
%-------------------------------------------------------------------------

parameters=[beta0 gamma alpha VarSize delta dmax VarMin VarMax];

%_________________________________________________________________________
%                     Inicialização do Enxame
%_________________________________________________________________________

firefly.Position=[];                                    %Posição do
firefly
firefly.Cost=[];                                  %Valor de custo do
firefly

BestSol.Cost=inf;                  %Inicializa a melhor solução
encontrada
BestCost=zeros(MaxIt,1); %Matriz para manter os melhores valores de
custo

[pop,BestSol]=Inicializar_FireflyEnxames(nPop,VarMin,VarMax,VarSize,Be
stSol.Cost,CostFunction);             %Função para a inicialização do
Enxame

%_________________________________________________________________________
%                  Loop principal do Algoritmo Firefly
%_________________________________________________________________________

tic
for it=1:MaxIt

    newpop=repmat(firefly,nPop,1);

    for i=1:nPop
        newpop(i).Cost=inf;

        for j=1:nPop

            if pop(j).Cost < pop(i).Cost

[newsol]=Movimentar_Firefly(i,j,pop,CostFunction,parameters);

                if newsol.Cost <= newpop(i).Cost
                    newpop(i)=newsol;

                    if newpop(i).Cost <= BestSol.Cost
                        BestSol=newpop(i);
                    end
                end
            end
        end
    end

%_________________________________________________________________________
%                  Salva as novas posições do enxame
%_________________________________________________________________________

[pop,BestCost]=Atualizar_Fireflies(newpop,pop,nPop,BestSol,BestCost,it
);
```

```
disp(['Iteration ' num2str(it) ': Best Cost = '
num2str(BestCost(it))]);   %Mostra a informação de iteração

alpha=alpha*alpha_damp;

parameters=[beta0 gamma alpha VarSize delta dmax VarMin VarMax];

%-------------------------------------------------------------------

if (erro >= BestSol.Cost)
    break;
end

%-------------------------------------------------------------------

end
toc

%_____________________________________________________________________
%           Calcular a resposta para o melhor individuo do sistema
%_____________________________________________________________________

G_FA=ReducedGs(BestSol.Position,type)

y_FA=lsim(G_FA,Rs,simTime);

%-------------------------------------------------------------------

G_err=Gs-G_FA;

y_err=lsim(G_err,Rs,simTime);

%_____________________________________________________________________
%                         Plote Gráficos
%_____________________________________________________________________

%RESPOSTA
figure(1)
plot(simTime,y,'k','linewidth',1.5)
hold on
plot(simTime,y_FA,'r:','linewidth',1.5)
grid
title('Comparação entre a Resposta Original e a Resposta Reduzida')
xlabel('Time (s)'), ylabel('Amplitude')
legend('Sistema Original','Reduzido - FA')
%axis([0 (length(simTime)-1)*0.01 0 max(y)*1.2])

%-------------------------------------------------------------------

%ERRO
figure(2)
plot(simTime,y_err,'k','linewidth',1.5)
grid
title('Erro ao Degrau')
legend('y_e_r_r_o')
xlabel('Time (s)'), ylabel('Amplitude')

%-------------------------------------------------------------------
```

```
%DIAGRAMA DE BODE
figure(3)
bode(Gs,'b')
grid
hold on
bode(G_FA,'r--')
legend('Gs - Original','G_FA - Reduzido')

%--------------------------------------------------------------------

%ITERAÇÃO
figure(4)
semilogy(BestCost,'linewidth',1.5)
title('Iteration x Best Cost')
xlabel('Iteration'), ylabel('Best Cost')
grid

disp('Fim!!')
```

==========

- **Atualizar_Fireflies:**

```
function[pop,BestCost]=Atualizar_Fireflies(newpop,pop,nPop,BestSol,Bes
tCost,it)
%

%_____________________________________________________________________
%         Juntar a posição do enxame antigo com as novas
%_____________________________________________________________________

pop=[pop;newpop]; %Fundir as duas populações

%_____________________________________________________________________
%       Organizar em ordem crescente a posição dos fireflies
%_____________________________________________________________________

[~,SortOrder]=sort([pop.Cost]);   %Ordem de classificação; Ordenar
pop=pop(SortOrder);
pop=pop(1:nPop);                  %Truncar

BestCost(it)=BestSol.Cost;        %Armazena o melhor custo já
encontrado

end
```

==========

- **CompareReducedFA_System1:**

```
function [error]=CompareReducedFA_System1(x,Rs,simTime,y)
%
```

```
%_____________________________________________________________________
%          Criar a Função de Transferência de 1a Ordem G_FA
%_____________________________________________________________________

num=[x(1)];
den=[1 x(2)];

G_FA=tf(num,den);
y_fa=lsim(G_FA,Rs,simTime);

%_____________________________________________________________________
%          Calcular o erro entre o FA e o sistema original
%_____________________________________________________________________

for h=1:length(y)
    desv(h)=((abs(y(h)-y_fa(h)))^2)/2;
end

    error=sum(desv)/length(y);

end
```

==========

- **CompareReducedFA_System2:**

```
function [error]=CompareReducedFA_System2(x,Rs,simTime,y)
%

%_____________________________________________________________________
%          Criar a Função de Transferência de 2a Ordem G_FA
%_____________________________________________________________________

num=[x(1) x(2)];
den=[1 x(3) x(4)];

G_FA=tf(num,den);
y_fa=lsim(G_FA,Rs,simTime);

%_____________________________________________________________________
%          Calcular o erro entre o FA e o sistema original
%_____________________________________________________________________

for h=1:length(y)
    desv(h)=((abs(y(h)-y_fa(h)))^2)/2;
end

    error=sum(desv)/length(y);

end
```

==========

- **CompareReducedFA_System3:**

```
function [error]=CompareReducedFA_System3(x,Rs,simTime,y)
%

%_____________________________________________________________________
%          Criar a Função de Transferência de 3a Ordem G_FA
%_____________________________________________________________________

num=[x(1) x(2) x(3)];
den=[1 x(4) x(5) x(6)];

G_FA=tf(num,den);
y_fa=lsim(G_FA,Rs,simTime);

%_____________________________________________________________________
%          Calcular o erro entre o FA e o sistema original
%_____________________________________________________________________

for h=1:length(y)
    desv(h)=((abs(y(h)-y_fa(h)))^2)/2;
end

    error=sum(desv)/length(y);

end
```

==========

- **CompareReducedFA_System4:**

```
function [error]=CompareReducedFA_System4(x,Rs,simTime,y)
%

%_____________________________________________________________________
%          Criar a Função de Transferência de 4a Ordem G_FA
%_____________________________________________________________________

num=[x(1) x(2) x(3) x(4)];
den=[1 x(5) x(6) x(7) x(8)];

G_FA=tf(num,den);
y_fa=lsim(G_FA,Rs,simTime);

%_____________________________________________________________________
%          Calcular o erro entre o FA e o sistema original
%_____________________________________________________________________

for h=1:length(y)
    desv(h)=((abs(y(h)-y_fa(h)))^2)/2;
end

    error=sum(desv)/length(y);

end
```

==========

- **CompareReducedFA_System5:**

```
function [error]=CompareReducedFA_System5(x,Rs,simTime,y)
%

%_____________________________________________________________________
%          Criar a Função de Transferência de 5a Ordem G_FA
%_____________________________________________________________________

num=[x(1) x(2) x(3) x(4) x(5)];
den=[1 x(6) x(7) x(8) x(9) x(10)];

G_FA=tf(num,den);
y_fa=lsim(G_FA,Rs,simTime);

%_____________________________________________________________________
%          Calcular o erro entre o FA e o sistema original
%_____________________________________________________________________

for h=1:length(y)
    desv(h)=((abs(y(h)-y_fa(h)))^2)/2;
end

    error=sum(desv)/length(y);

end
```

==========

- **Create_CostFunction:**

```
function [CostFunction]=Create_CostFunction(Rs,simTime,y,type)
%

%_____________________________________________________________________
%          Escolha do tipo de Função e Ordem de redução
%_____________________________________________________________________

    if (type==1)
        CostFunction=@(x) CompareReducedFA_System1(x,Rs,simTime,y);
    else
        if (type==2)
            CostFunction=@(x)
CompareReducedFA_System2(x,Rs,simTime,y);
        else
            if (type==3)
                CostFunction=@(x)
CompareReducedFA_System3(x,Rs,simTime,y);
            else
                if (type==4)
                    CostFunction=@(x)
CompareReducedFA_System4(x,Rs,simTime,y);
                else
                    if (type==5)
```

```
                        CostFunction=@(x)
CompareReducedFA_System5(x,Rs,simTime,y);
                    else
                        return
                    end
                end
            end
        end
    end
end
```

==========

- **Inicializar_FireflyEnxames:**

```
function
[pop,BestSol]=Inicializar_FireflyEnxames(nPop,VarMin,VarMax,VarSize,be
stCost,CostFunction)
%

%_____________________________________________________________________
%                       Estrutura vazia do firefly
%_____________________________________________________________________

firefly.Position=[];                                    %Posição do
firefly
firefly.Cost=[];                                     %Valor de custo
firefly

pop=repmat(firefly,nPop,1);              %Inicializar Matriz de
população

%_____________________________________________________________________
%                       Criar fireflies iniciais
%_____________________________________________________________________

    for i=1:nPop
        pop(i).Position=unifrnd(VarMin,VarMax,VarSize);
        pop(i).Cost=CostFunction(pop(i).Position);

        if pop(i).Cost<=bestCost
            BestSol=pop(i);
        end

    end

end
```

==========

- **Movimentar_Firefly:**

```
function [newsol]=Movimentar_Firefly(i,j,pop,CostFunction,parameters)
%
```

```
%_____________________________________________________________________
%                           Parametros
%_____________________________________________________________________

m=2;beta0=parameters(1);
gamma=parameters(2);
alpha=parameters(3);
VarSize=parameters(4:5);
delta=parameters(6);
dmax=parameters(7);
VarMin=parameters(8);
VarMax=parameters(9);

%_____________________________________________________________________
%                       Estrutura vazia de Firefly
%_____________________________________________________________________

rij=norm(pop(i).Position-pop(j).Position)/dmax;
beta=beta0*exp(gamma*rij^m);
e=delta*unifrnd(-1,+1,VarSize);

newsol.Position = pop(i).Position +
beta*rand(VarSize).*(pop(j).Position-pop(i).Position) + alpha*e;

newsol.Position=max(newsol.Position,VarMin);
newsol.Position=min(newsol.Position,VarMax);

newsol.Cost=CostFunction(newsol.Position);

end
```

==========

- **Number_Decision_Variables:**

```
function [nVar]=Number_Decision_Variables(type)
%

%_____________________________________________________________________
%                           Condições
%_____________________________________________________________________

if (type==1)
    nVar=2;
else
    if (type==2)
        nVar=4;
    else
        if (type==3)
            nVar=6;
        else
            if (type==4)
                nVar=8;
            else
                if (type==5)
                    nVar=10;
                else
                    return
```

```
                end
            end
        end
    end
end

end
```

==========

- **ReducedGs:**

```
function [G_FA]=ReducedGs(x,type)
%

%_____________________________________________________________________
%                           Condições
%_____________________________________________________________________

if (type==1)
    num=[x(1)];
    den=[1 x(2)];
    G_FA=tf(num,den);
else
    if (type==2)
        num=[x(1) x(2)];
        den=[1 x(3) x(4)];
        G_FA=tf(num,den);
    else
        if (type==3)
            num=[x(1) x(2) x(3)];
            den=[1 x(4) x(5) x(6)];
            G_FA=tf(num,den);
        else
            if (type==4)
                num=[x(1) x(2) x(3) x(4)];
                den=[1 x(5) x(6) x(7) x(8)];
                G_FA=tf(num,den);
            else
                if (type==5)
                    num=[x(1) x(2) x(3) x(4) x(5)];
                    den=[1 x(6) x(7) x(8) x(9) x(10)];
                    G_FA=tf(num,den);
                else
                    return
                end
            end
        end
    end
end

G_FA=tf(num,den);

end
```

ANEXO IV

Linhas de comando no Matlab para os exemplos mostrados nos estudos de casos do Capítulo 4, para determinação do sistema reduzido por meio do PSO.

==========

- **Main Algorithm:**

```
%_____________________________________________________________________
%              Reduction by Particle Swarm Optimization
%              Date Created: April 15, 2017
%              By: Silva, Marlon J. P.
%              Electrical Engineer (UFPA)
%              Master in Electrical Engineering (UFPA)
%              Ver. 3.0
%_____________________________________________________________________



clc

%_____________________________________________________________________
%  Parameters for Reduction
%_____________________________________________________________________

type=2;                                          % Reduced system order
erro=6e-5;                                     % Maximum allowed error
VarMin=0;                                % Decision Variables Lower Bound
VarMax=5;                                % Decision Variables Upper Bound

nVar=Number_Decision_Variables(type);   % Number of Decision Variables
VarSize=[1 nVar];                        % Decision Variables Matrix Size
CostFunction=Creat_costfuction(Rs,simTime,y,type)



%_____________________________________________________________________
% Parameters of PSO
%_____________________________________________________________________

size=25;                                                  % Swarm size
d=nVar;                                                   % Dimension
w=0.4;                                                % Inertial Factor
c1=2;                                                 % Cognitive Factor
c2=2;                                                    % Social Factor
MaxIt=200;                                           % Maximum iteration

parameters=[d w c1 c2 size MaxIt];                % Parameters of Swarm
```

```
%_____________________________________________________________________
%  Empty Structure of Particle
%_____________________________________________________________________

Particle.Position=[];                          % Position of Particle
Particle.Speed=[];                                % Speed of Particle
Particle.Cost=[];                                  % Cost of Particle

Pg.Position=[];                               % Position of best Particle
Pg.Cost=inf;                                      % Cost of best Particle


BestParticles=repmat(Particle,MaxIt,1);
% Array to Hold Best Particles each iteraction

%_____________________________________________________________________
% Randomly initiate the particles in the Swarm
%_____________________________________________________________________

[swarm,Pg,P]=Initialize_Swarm(size,VarMin,VarMax,VarSize,Pg,CostFuncti
on);

%_____________________________________________________________________
%  PSO Main Loop
%_____________________________________________________________________

tic
for it=1:MaxIt

%_____________________________________________________________________
%  Swarm Update
%_____________________________________________________________________

[swarm]=Swarm_Update(VarMin,VarMax,Pg,P,swarm,parameters,CostFunction)
;

%_____________________________________________________________________
%  Update of the Best Particle
%_____________________________________________________________________

    [Pg,P]=Update_BestParticles(size,Pg,P,swarm);

    BestParticles(it)=Pg;

    disp(['Iteration ' num2str(it) ' Cost = ' num2str(Pg.Cost) ]);


    if(erro>=Pg.Cost)
        break
    end



end
```

```
%_____________________________________________________________________
% End of Algorithm interactions
%_____________________________________________________________________

tempo=toc            % Stores the time (seconds) used by the algorithm
in the search

%_____________________________________________________________________
% Plotar resultado
%_____________________________________________________________________

figure;
Cost=[BestParticles(1:it).Cost];
plot(Cost,'LineWidth',2)
xlabel('Iteração');
ylabel('Valor');
grid on;

%_____________________________________________________________________
% Calculate the output response of the system formed by the best
individual found and plot the signals.
%_____________________________________________________________________

G_PSO=reducedGs(Pg.Position,type)    % Print a transfer function found
on the Workspace.
y_PSO=lsim(G_PSO,Rs,simTime);   % Calculate best system response found

%_____________________________________________________________________
% Plot the response of the real system and that found by the best
individual in the Firefly Algorithm
%_____________________________________________________________________

figure
plot(simTime,y,'k','linewidth',2.5)
hold on
plot(simTime,y_PSO,'r:','linewidth',2.5)
grid
title('Comparação entre a resposta ORIGINAL e a resposta REDUZIDA')
xlabel('Tempo (s)')
ylabel('Resposta')
legend('Original','PSO')

%_____________________________________________________________________
% Plot the Bode diagram
%_____________________________________________________________________

figure
bode(Gs,'k--')              % Frequency response of the original model
grid
hold on
bode(G_PSO,'r--')                   % Reduced model frequency response
legend('Original','PSO')
disp('fim!!!')
```

ANEXO V

Linhas de comando no Matlab para os exemplos mostrados nos estudos de casos do Capítulo 4, para determinação do sistema reduzido por meio do SFLA.

==========

- **Main Algorithm:**

```
%_____________________________________________________________________
%              Reduction by Shuffled Frog Leaping Algorithm
%              Date Created: September 20, 2018
%              By: Silva, Marlon J. P.
%              Electrical Engineer (UFPA)
%              Master in Electrical Engineering (UFPA)
%              Ver. 3.1
%_____________________________________________________________________

clc
%_____________________________________________________________________
%  Parameters for Reduction
%_____________________________________________________________________

type=2;                                          % Reduced system order
erro=6e-5;                                      % Maximum allowed error
CostFunction=Creat_costfuction(Rs,simTime,y,type)      % Cost Function
nVar=Number_Decision_Variables(type);   % Number of Decision Variables
VarSize=[1 nVar];                     % Decision Variables Matrix Size
VarMin=0;                             % Decision Variables Lower Bound
VarMax= 5;                            % Decision Variables Upper Bound

%_____________________________________________________________________
%  SFLA Parameters
%_____________________________________________________________________

MaxIt = 100;                             % Maximum Number of Iterations
nPopMemeplex = 5;                                     % Memeplex Size
nPopMemeplex = max(nPopMemeplex, nVar+1);       % Nelder-Mead Standard
nMemeplex = 5;                                  % Number of Memeplexes
nPop = nMemeplex*nPopMemeplex;                       % Population Size
I=reshape(1:nPop, nMemeplex, []);

%_____________________________________________________________________
%  FLA Parameters
%_____________________________________________________________________

fla_params.q = max(round(0.3*nPopMemeplex),2);     % Number of Parents
fla_params.alpha = 3;                           % Number of Offsprings
fla_params.beta = 5;                    % Maximum Number of Iterations
fla_params.sigma = 2;                                    % Step Size
fla_params.CostFunction = CostFunction;
fla_params.VarMin = VarMin;
fla_params.VarMax = VarMax;
```

```
%_____________________________________________________________________
%  Empty Individual Template
%_____________________________________________________________________
empty_individual.Position = [];
empty_individual.Cost = [];

%_____________________________________________________________________
% Initialize Population Array
%_____________________________________________________________________
pop = repmat(empty_individual, nPop, 1);

% Initialize Population Members
for i=1:nPop
    pop(i).Position = unifrnd(VarMin, VarMax, VarSize);
    pop(i).Cost = CostFunction(pop(i).Position);
end


pop = SortPopulation(pop);                          % Sort Population
BestSol = pop(1);                    % Update Best Solution Ever Found
BestCosts = nan(MaxIt, 1);         % Initialize Best Costs Record Array

%_____________________________________________________________________
% main loop
%_____________________________________________________________________

tic
for it = 1:MaxIt

    fla_params.BestSol = BestSol;

%_____________________________________________________________________
%  Initialize Memeplexes Array
%_____________________________________________________________________

    Memeplex = cell(nMemeplex, 1);
%_____________________________________________________________________
% Form Memeplexes and Run FLA

%_____________________________________________________________________

    for j = 1:nMemeplex

        Memeplex{j} = pop(I(j,:));                % Memeplex Formation
        Memeplex{j} = RunFLA(Memeplex{j}, fla_params);       % Run FLA
        pop(I(j,:)) = Memeplex{j};      % Insert Updated Memeplex into
Population
    end

    pop = SortPopulation(pop);                          % Sort Population

    BestSol = pop(1);                % Update Best Solution Ever Found


    BestCosts(it) = BestSol.Cost;          % Store Best Cost Ever Found
```

```
    disp(['Iteration ' num2str(it) ': Best Cost = '
num2str(BestCosts(it))]);                % Show Iteration Information

    if(erro>=BestSol.Cost)

    break;

  end

end

tempo=toc
%_____________________________________________________________________________
% Results
%_____________________________________________________________________________
figure;
%plot(BestCosts, 'LineWidth', 2);
semilogy(BestCosts, 'LineWidth', 2);
xlabel('Iteration');
ylabel('Best Cost');
grid on;

BestSol.Position

%_____________________________________________________________________________
% Calculate the output response of the system formed by the best
individual found and plot the signals.
%_____________________________________________________________________________

G_SFLA=reducedGs(BestSol.Position,type)    % Print a transfer function
found on the Workspace.
y_SFLA=lsim(G_SFLA,Rs,simTime); % Calculate best system response found

%_____________________________________________________________________________
% Plot the response of the real system and that found by the best
individual in the Firefly Algorithm
%_____________________________________________________________________________
figure
plot(simTime,y,'k','linewidth',2.5)
hold on
plot(simTime,y_SFLA,'r:','linewidth',2.5)
grid
title('Comparação entre a resposta ORIGINAL e a resposta REDUZIDA')
xlabel('Tempo (s)')
ylabel('Resposta')
legend('Original','SFLA')

%_____________________________________________________________________________
% Plot the Bode diagram
%_____________________________________________________________________________
figure
bode(Gs,'k--')              % Frequency response of the original model
grid
hold on
bode(G_SFLA,'r--')                  % Reduced model frequency response
legend('Original','SFLA')
disp('fim!!!')
```