

Sistemas Bioinspirados Aplicados a Engenharia

Algoritmos Bioinspirados – Sintonia de Controlador PID, Aplicado a um Sistema Motor – Dinamômetro

Leonardo Bezerra Libanio

Programa de Pós – graduação em Sistemas Mecatrônicos

Universidade de Brasília

Brasília, DF, Brasil

E-mail: bezerralibanio@gmail.com

Resumo — O presente artigo tem como objetivo a análise e testes de algoritmos de otimização bioinspirados em inteligência de enxames e evolutiva para resolução de problemas de sintonia das componentes PID na resolução da função custo motor – dinamômetro. A metodologia utilizada é descrita em 02 etapas. Na primeira etapa é descrito os algoritmos bioinspirados; Otimização por Enxame de Partículas – PSO e Otimização por Evolução Diferencial - DE; e a função custo que representa o problema de otimização. Na segunda etapa é apresentado os resultados e discussões da simulação realizada com a ferramenta de desenvolvimento Matlab e Simulink. A função custo selecionada é não linear com representação a partir de uma função de transferência de um sistema motor – dinamômetro. Para melhor observar o desempenho dos algoritmos nos testes, foi apresentado de forma gráfica e estatística os resultados obtidos.

Palavras-chave - Algoritmos bioinspirados, inteligência de enxames, PID, Função Custo.

I. INTRODUÇÃO

A definição de otimização matemática é a capacidade de obter o melhor resultado possível dentro de um conjunto de soluções em um espaço finito ou infinito, visando minimizar os esforços ou maximizar os lucros. Para verificar se uma solução é ótima, é necessário obter uma representação matemática, chamada de “função objetivo” ou “função custo”, que permita avaliar cada solução. Esta representação é formada por uma ou várias variáveis de decisão e um conjunto de restrições que afetam o problema de otimização. Desta forma, o objetivo da otimização é encontrar o melhor valor ou valor aceitável da função custo (o maior valor numérico possível implica em maximização e o menor valor numérico possível implica em minimização) [01].

Algoritmos bioinspirados são técnicas de otimização muito usadas nos últimos anos para resolver problemas complexos com múltiplas variáveis nos quais soluções robustas são difíceis ou impossíveis de encontrar usando métodos clássicos (baseados no cálculo do gradiente). Esse tipo de algoritmo imita o comportamento social de algumas espécies da natureza, mas, precisamente de espécies que possuem inteligência coletiva. A característica principal destes algoritmos é o

processo de interação entre os indivíduos do enxame os quais estão baseados em regras simples, porém, usam suas capacidades de inteligência coletiva para realizar tarefas complexas [02].

Na avaliação dos algoritmos foram utilizadas funções que representam problemas de otimização considerados benchmark (é uma técnica utilizada para medir o desempenho de um sistema) [03]. Estas funções de teste têm características diferentes em termos de complexidade, de composição e interação entre as variáveis de decisão.

No seguimento do artigo é descrito: Os algoritmos bioinspirados PSO e DE, a técnica de melhoria de desempenho do algoritmo OBL, a função custo motor – dinamômetro elétrico, os resultados e discussões dos testes dos algoritmos com base em gráficos e dados estatísticos, a conclusão e as propostas para trabalhos futuros.

II. ALGORITMOS BIOINSPIRADOS EM INTELIGÊNCIA DE ENXAMES E EVOLUTIVA

A. Otimização por Enxame de Partículas - PSO

A técnica PSO (*Particle Swarm Optimization*) é um algoritmo de otimização inspirada no comportamento social do voo dos bandos de pássaros e também no movimento dos cardumes de peixes durante a busca por alimento. O algoritmo foi desenvolvido em 1995 por dois pesquisadores de duas áreas diferentes o psicólogo James Kennedy e o engenheiro Russell Eberhart [04].

A inspiração natural deste algoritmo pode ser explicada da seguinte forma, seja um bando de pássaros que procura uma fonte de alimento em uma área delimitada. No início, os pássaros do bando voam aleatoriamente na área de busca e comunicam-se entre eles quando encontram uma nova fonte de alimento. Ao interagir entre si, o enxame segue o pássaro que estiver mais perto da melhor fonte de alimento. O algoritmo PSO emula este comportamento para resolver problemas de otimização, de forma que as características fundamentais para seu funcionamento são o conhecimento individual,

representado pelo histórico de cada partícula, e o conhecimento social, representado pelo histórico das partículas vizinhas.

A técnica PSO é muito usado em diferentes áreas para resolver problemas de otimização ou como complemento para outras técnicas de controle [05] [06].

Algoritmo PSO básico pode ser descrito, de forma geral, como um conjunto de vetores que contém dados ligados à posição das partículas, as quais se movimentam em uma região ou área definida (vide Figura 01). Neste algoritmo, a nova posição de uma partícula é definida por sua experiência particular, conhecida como memória individual (a partícula lembra-se da melhor posição determinada pela avaliação da função custo). Da mesma forma a partícula é influenciada pela experiência global do enxame, conhecida como memória coletiva (a partícula lembra a melhor posição do enxame determinada pela melhor aptidão ao ser avaliada na função custo) [04].

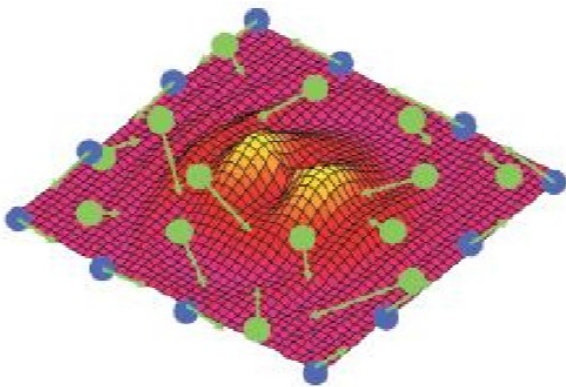


Fig. 01 – Conjunto de vetores em movimento [07]

Os parâmetros do algoritmo PSO são descritos, como:

- Partícula: indivíduos do enxame.
- Enxame: conjunto de partículas.
- Posição (x): coordenadas de uma partícula no espaço N-dimensional (possível solução de um problema)
- Aptidão: valor que representa quão boa é uma solução. Resultado da avaliação de uma posição por meio de uma função custo.
- $Pbest$ (y_i): memória individual da partícula determinada pela posição do melhor valor de aptidão encontrado pela partícula
- $Gbest$ (y_s): memória coletiva do enxame determinada pela posição do melhor valor de aptidão encontrado entre todas as partículas.
- $Vmax$: velocidade máxima possível para uma partícula.

Estes parâmetros são a base para o algoritmo. É importante descrever as equações que os representam e definem as mudanças de posições das partículas no tempo, quando elas se movimentam procurando a melhor solução em um espaço de N dimensões. As equações (1) e (2), mostram a atualização da posição da i -ésima (i th) partícula na j -ésima (j th) dimensão para um enxame com S partículas.

$$v_{ij}^{(t+1)} = wv_{ij}^{(t)} + c_1U_{1j}(y_{ij}^{(t)} - x_{ij}^{(t)}) + c_2U_{2j}(y_{sj}^{(t)} - x_{ij}^{(t)}) \quad (1)$$

$$x_{ij}^{(t+1)} = x_{ij}^{(t)} + v_{ij}^{(t+1)} \quad (2)$$

- U_{1j} e U_{2j} são números aleatórios uniformemente distribuídos entre 0 e 1
- c_1 e c_2 são os coeficientes cognitivo individual e social, respectivamente. Um valor grande de c_1 indica partículas com alta confiança na sua experiência, enquanto um valor grande de c_2 proporciona maior confiança das partículas na experiência do enxame.
- w é denominado o fator de inércia. Este parâmetro é aplicado durante o cálculo da velocidade, sendo utilizado como um fator de escala para a velocidade atual de cada partícula. Desta forma define a influência da velocidade na medida em que o algoritmo vai sendo executado.
- y_{ij} é a melhor posição individual da i th partícula na j th dimensão.
- y_{sj} é a melhor posição global entre todas as partículas na j th dimensão.
- v_{ij} estão limitadas na faixa $[-Vmax, Vmax]$ evitando assim que as partículas abandonem o espaço de busca. Nesta equação, pode-se observar que a posição para o próximo instante de tempo, depende da posição atual e do cálculo da velocidade. Entretanto, a velocidade para o próximo instante de tempo da partícula depende diretamente da velocidade atual, da melhor posição encontrada pela partícula (y_i) e da melhor posição encontrada pelo enxame (y_s).

A nova posição de uma partícula pode se observar como uma soma vetorial dos parâmetros y_{sj} e y_{ij} (vide Figura 02).

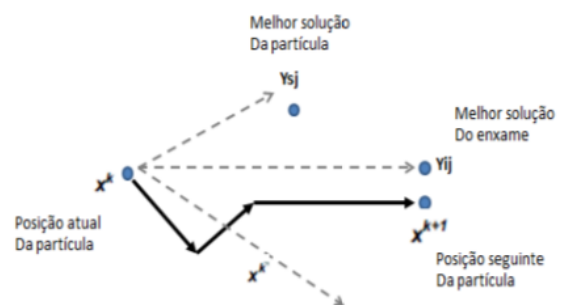


Fig. 02 – Otimização da posição da partícula [07]

Demonstração do pseudocódigo do algoritmo PSO na sua forma básica:

Pseudocódigo: PSO básico

```

Entrada: S, N, f, c1, c2, xmax, vmax, Maxiter
Saída: pos. da partícula: x e seu melhor valor de aptidão: f(x)
Início
-----//Inicializa o enxame
-----Para k=1: S faça
-----Para j=1: N faça
-----vij = - vmax + 2U[0,1] vmax
-----xij = - xmax + 2U[0,1] xmax
-----Fim para
Fim para
Repita
-----//Avaliação e detecção
-----Para k=1: S faça
-----Se f(xk) = fmin então
-----yk = xk
-----fmin = - f(xk)
-----Fim se
-----Fim para
-----//Calcule y, usando os S valores de aptidão f(yik)
-----//Atualização usando equações
-----Para k=1: S faça
-----Para j=1: N faça

-----vij(t+1) = wvij(t) + c1U1j(yij(t) - xij(t)) + c2U2j(ysj(t) - xij(t))
-----xij(t+1) = xij(t) + vij(t+1)

-----Fim para
-----Fim para
-----Até iterações = Maxiter
Fim

```

B. Otimização por Evolução Diferencial – DE

Este algoritmo utiliza NP vetores de parâmetros D-dimensionais x_{i,G}, i = 1, . . . , NP, como população em cada geração G.

O conjunto inicial de vetores é gerado aleatoriamente e deve cobrir todo o espaço de busca. Na ausência de qualquer conhecimento acerca do espaço de busca (regiões promissoras ou mesmo soluções parciais), utiliza-se uma distribuição uniforme para a população inicial [10].

DE gera novos vetores de parâmetros através da adição da diferença ponderada entre dois vetores de parâmetros a um terceiro indivíduo. Considere esta operação como uma mutação.

Os vetores de parâmetros mutados são então combinados com outros vetores pré-determinados, denominados *target vectors*, a fim de gerar os *trial vectors*. Esta combinação de parâmetros é referida como crossover em DE. É importante ressaltar que cada vetor presente na atual população deve ser usado uma vez como *trial vector*.

Caso o *trial vector* forneça um valor de *fitness* maior (maximização) que aquele associado ao respectivo *target vector*, este último dará lugar ao primeiro na próxima geração. Esta operação corresponde à seleção.

Mutação

Para cada *target vector* x_{i,G}, i = 1, . . . , NP, um novo vetor é gerado por meio da equação 8.

$$v_{i,G+1} = x_{r1,G} + F(x_{r3,G} - x_{r2,G}) \quad (8)$$

Onde r₁, r₂, r₃ ∈ 1, 2, . . . , NP são índices mutuamente distintos e também diferentes do índice i.

F é uma constante real ∈ [0, 2] que determina o tamanho do passo a ser dado na direção definida pelo vetor diferença x_{r3,G} - x_{r2,G}. Seja x_{i,G} o atual *target vector* (vide Figura 03).

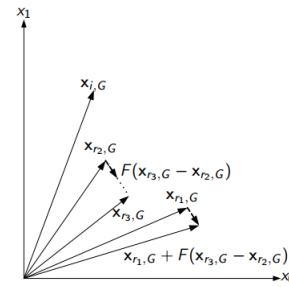


Fig. 3 - Processo bidimensional de mutação [10].

Crossover

Com a finalidade de aumentar a diversidade dos vetores de parâmetros mutados, um procedimento similar ao crossover é utilizado. Seja x_{i,G} o *target vector* sob análise e v_{i,G+1} o respectivo vetor modificado obtido por meio da equação 8.

O vetor u_{i,G+1} = (u_{1i,G+1} u_{2i,G+1} . . . u_{Di,G+1}), denominado *trial vector*, é obtido conforme equação 9.

$$u_{ji,G+1} = \begin{cases} v_{ji,G+1}, & \text{se } r_j \leq CR \text{ ou } j = I_i \\ x_{ji,G}, & \text{se } r_j > CR \text{ e } j \neq I_i \end{cases} \quad (9)$$

onde j = 1, . . . , D, r_j ~ U(0,1), CR ∈ [0, 1] é uma constante definida pelo usuário e I_i é um índice aleatoriamente escolhido ∈ 1, . . . , D, o que garante que u_{i,G+1} recebe pelo menos uma componente de v_{i,G+1}. Seja x_{i,G} o *target vector* sob análise e v_{i,G+1} o respectivo vetor mutado obtido por meio da equação 8. A Figura 04 detalha o processo de geração do *trial vector* u_{i,G+1}.

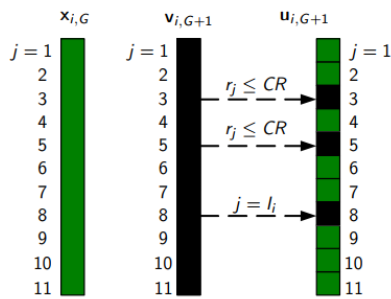


Fig. 04 - Processo de *Crossover* [10]

Seleção

Após as etapas de mutação e *crossover*, nas quais todos os NP vetores serviram como *target vector*, a seleção dos vetores que serão preservados para a próxima geração é feita usando um critério elitista.

Seja $x_{i,G}$ o *target vector* sob análise e $u_{i,G+1}$ seu respectivo trial vector.

Se $f(u_{i,G+1}) > f(x_{i,G})$, então $x_{i,G+1} = u_{i,G+1}$. (Max.)

Caso contrário, $x_{i,G+1} = x_{i,G}$.

Pseudocódigo: DE básico

Função $x = \text{DE}(\text{NP}, \text{CR}, \text{F}, \text{range}, \text{f})$

$x = \text{random}(\text{range}, \text{NP})$

$\text{fit}_x = f(x)$

while critério de parada não for *do*

-----*for* $i = 1$ até NP *do*

-----*if* $\text{fit}_u(i) > \text{fit}_x(i)$ *then*

----- $x_{i,G+1} = u_{i,G+1}$

-----*else*

----- $x_{i,G+1} = x_{i,G}$

-----*end if*

-----*end for*

end while

C. Técnicas de Melhoria de Desempenho dos Algoritmos Bioinspirados.

Estes métodos são modificações realizadas por pesquisadores na busca do aumento do desempenho dos algoritmos bioinspirados baseados em populações. Métodos denominados na literatura como o de adição de diversidade artificial, tentam evitar o problema de convergência prematura, especialmente quando se faz uso de topologias em que os agentes do enxame seguem exclusivamente o indivíduo com melhor desempenho. Para esses casos existe uma forte tendência a encontrar soluções sub-ótimas como consequência do melhor indivíduo encontrar-se preso em um mínimo local [8].

É importante saber que, com estas modificações pode-se obter a melhoria no desempenho dos algoritmos, porém, incrementa-se também a complexidade computacional do algoritmo. Os métodos que são apresentados neste trabalho dos muitos que podem ser encontrados na literatura são: 1) Fator de inércia; 2) Fatores de constrição; 3) Variações de

Aprendizagem por Oposição (OBL); e 4) Atrativo-Repulsivo (AR).

1) Fator de inércia: é aplicado durante o cálculo da velocidade, sendo utilizado como um fator de escala para a velocidade atual de cada partícula, vide equações 10 e 11. [09]

$$v_{ij}^{(t+1)} = \underbrace{w}_{\text{inércia}} v_{ij}^{(t)} + \underbrace{c_1 U_{1j}[0,1]}_{\text{componente cognitivo}} (y_{ij}^{(t)} - x_{ij}^{(t)}) + \underbrace{c_2 U_{2j}[0,1]}_{\text{componente social}} (y_{sj}^{(t)} - x_{ij}^{(t)}) \quad (10)$$

$$x_{ij}^{(t+1)} = x_{ij}^{(t)} + v_{ij}^{(t+1)} \quad (11)$$

Esta técnica é amplamente utilizada, configurando o fator de inércia w para decrescer linearmente, desde valores grandes até valores pequenos durante a execução do algoritmo. Valores típicos: $w=[0.9 \text{ a } 0.1]$.

- O fator de inércia controla a capacidade de exploração das partículas.

- Valores grandes de w resultam em uma busca global

- Valores pequenos de w permitem às partículas explorarem localmente a vizinhança de uma possível solução.

- O algoritmo PSO com o uso do fator de inércia w e coeficientes c_1 e c_2 é conhecido academicamente como algoritmo PSO canônico. Neste relatório é utilizado o PSO canônico.

2) Fatores de constrição: úteis para assegurar a convergência no algoritmo. O fator de constrição facilita a escolha dos parâmetros w , c_1 e c_2 mediante as seguintes relações (vide equações 12 e 13).

$$v_{ij}^{(t+1)} = x(v_{ij}^{(t)} + c_1 U_{1j}(y_{ij}^{(t)} - x_{ij}^{(t)}) + c_2 U_{2j}(y_{sj}^{(t)} - x_{ij}^{(t)})) \quad (12)$$

$$x = 2 / |2 - \phi - \sqrt{(\phi^2 - 4\phi)}| \quad (13)$$

sendo, $\phi = c_1$ e c_2 , $\phi > 4$

Desta forma é possível aplicar o PSO sem impor restrições na trajetória das partículas, especificamente no valor de v_{\max} .

- O método de constrição mais encontrado na literatura é utilizar $f=4.1$ com constantes $c_1=c_2=2.05$ ($c=0.729$) [02].

3) Variação de Aprendizagem por Oposição (OBL)

O método de aprendizado em oposição, o qual direciona a busca da melhor solução pelos algoritmos bioinspirados, na direção oposta da busca atual. O processo consiste em que em um determinado momento os agentes encontram-se juntos ao redor da melhor posição encontrada, com um decremento da diversidade. Nesse caso, esta técnica permite trocar as posições de alguns dos seus agentes às coordenadas opostas, explorando novas possibilidades no espaço de busca [11].

A abordagem OBL está baseada no conceito do número oposto, definido pela equação 14.

$$\dot{x} = a + b - x \quad (14)$$

sendo x um número real definido na faixa $[a,b]$ e \dot{x} o número oposto de x . Esta definição também é válida para pontos N -dimensionais x_i definidos na faixa $[a_i,b_i]$, $i = 1,2,...,N$ [07].

D. Função Custo

No estudo de algoritmos de otimização é comum a realização de comparações entre diferentes algoritmos usando uma série de funções teste. As características das funções teste são as mais diversas, no entanto, em síntese a maior parte delas pode ser implementada no espaço n dimensional, exigindo do algoritmo a estimação de n parâmetros. Além disso, muitas delas são multimodais o que dificulta a otimização pelo excesso de mínimos locais, tornando-as eficientes para avaliar o desempenho dos algoritmos de otimização nas mais diversas situações. A seguir são apresentadas as principais funções teste utilizadas na literatura junto de suas principais características.

1) Função PID -

a) *Descrição do problema* - O controle PID (Proporcional Integral Derivativo) é uma das técnicas mais empregadas quando se deseja realizar o controle de variáveis contínuas. O controle PID consiste em um algoritmo matemático, que tem por função o controle preciso de uma variável em um sistema, permitindo ao sistema operar de forma estável no ponto de ajuste desejado, mesmo que ocorram variações ou distúrbios que afetariam sua estabilidade, conforme Figura 05 [12].

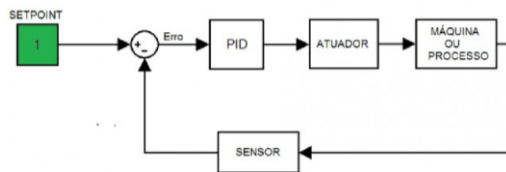


Fig. 05 – Diagrama de blocos controlador PID [12]

Aplicação dos controladores PID podem ser encontradas em qualquer área que necessite de controle de variáveis contínuas como: rotação, nível, pressão, vazão, temperatura, posicionamento, controle de tensão em fontes chaveadas, etc.

O controle PID pode ser descrito pela equação 15:

$$MV = K_p \times E + K_i \int_0^t E \times dt + K_p K_d \frac{dE}{dt} S_0 \quad (15)$$

Onde: MV: Variável manipulada. K_p : Ganho proporcional. K_i : Ganho integral. K_d : Ganho derivativo. E: Erro ou desvio. S_0 : Saída inicial do controlador.

O erro é a diferença entre o valor desejado (*setpoint*) e o valor real da variável, aplicado a uma planta, conforme Figura 06.

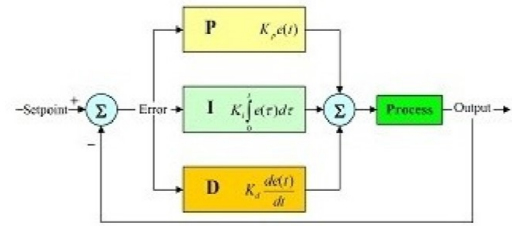


Fig. 06 – Diagrama de blocos de um controlador PID [12]

b) Variáveis de um controlador PID

A **ação proporcional** elimina as oscilações da variável, tornando o sistema estável, mas não garante que a mesma esteja no valor desejado (*setpoint*), esse desvio é denominado off-set e atua corrigindo o erro do sistema, multiplicando o ganho proporcional pelo erro, dessa forma agindo com uma maior amplitude de correção a fim de manter a estabilidade da variável, conforme equação 16.

$$MV = K_p \times E \quad (16)$$

A **ação integral** elimina o desvio de *off-set*, fazendo com que a variável permaneça próximo ao valor desejado para o sistema mesmo após um distúrbio, ou seja a variável permanece próximo ao setpoint mesmo que ocorra uma variação brusca nas condições de operação, atua integrando o erro no tempo, portanto quanto maior for o tempo de permanência do erro no sistema, maior será a amplitude da ação integral, conforme a equação 17.

$$MV = K_i \int_0^t E \times dt \quad (17)$$

A **ação derivativa**: A ação derivativa fornece ao sistema uma ação antecipativa evitando previamente que o desvio se torne maior quando o processo se caracteriza por ter uma correção lenta comparada com a velocidade do desvio. A ação derivativa tem sua resposta proporcional à taxa de variação da variável do processo, aumentando a velocidade de resposta do sistema caso a presença do erro seja detectada, conforme equação 18.

$$MV = K_d \frac{dE}{dt} S_0 \quad (18)$$

c) Formulação do problema de otimização

A função custo pode ser formulada a partir das variáveis da resposta ao degrau unitário de uma planta a ser controlada, a avaliação de desempenho pode ser medida em relação ao valor desejado (*set point*) em termos de; sobrepasso (*overshoot*),

erro de estado estacionário (*steady-state error*), tempo de subida (*rise time*) e tempo de estabilização (*settling time*), conforme Figura 07.

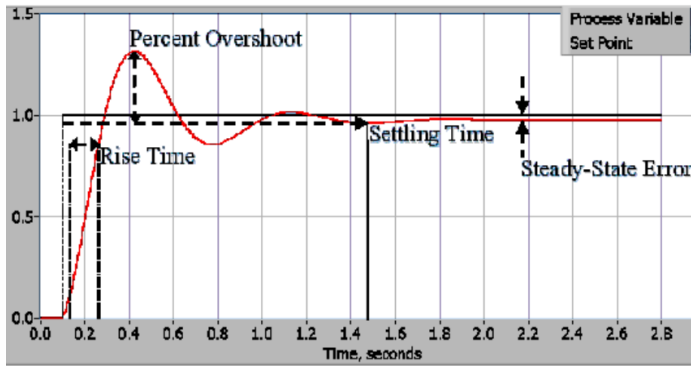


Fig. 07 – Resposta de um sistema PID em malha fechada [13].

2) Modelagem da Planta

O motor fornece um deslocamento de saída para uma tensão de entrada, uma saída mecânica gerada por uma entrada elétrica. A Figura 08 (a) demonstra um sistema eletromecânico e a Figura 08 (b) a representação de deslocamento de saída gerado por uma entrada elétrica.

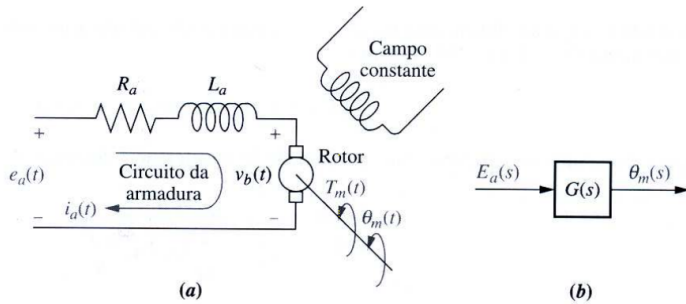


Fig. 08 – Representação de um sistema eletromecânico [14]

O campo magnético é produzido por ímãs permanentes estacionários ou por meio de um eletroímã estacionário chamado de campo fixo.

Um circuito rotativo denominado armadura, através do qual circula a corrente $i_a(t)$, corta o campo magnético segundo um ângulo reto e experimenta uma força, $F = B l i_a(t)$, sendo B a intensidade do campo magnético e l o comprimento do condutor.

O torque resultante aciona o rotor, o qual é o elemento girante do motor, representado pela equação 26:

$$V_b(t) = K_b \frac{\partial \theta_m(t)}{\partial t} \quad (19)$$

Sendo $V_b(t)$ a força contra eletromotriz (fcm), K_b a constante de fcm e $\frac{\partial \theta_m(t)}{\partial t} = \omega_m(t)$ é a velocidade angular. Aplicando-se a transformada de Laplace na equação 19, considerando-se as condições iniciais nulas, tem-se a equação 20:

$$V_b(s) = K_b s \theta_m(s) \quad (20)$$

A equação 21 da transformada de Laplace, considerando-se as condições iniciais nulas, da equação de malha do circuito de armadura é:

$$R_a I_a(s) + L_a s I_a(s) + V_b(s) = E_a(s) \quad (21)$$

O torque produzido pelo motor é proporcional à corrente de armadura, conforme equação 22.

$$T_m(s) = K_t I_a(s) \quad (22)$$

Sendo K_t uma constante de torque do motor. Rescrevendo, tem-se a equação 23.

$$I_a(s) = \frac{T_m(s)}{K_t} \quad (23)$$

Substituindo 19 e 22 em 21, tem-se a equação 31.

$$\frac{(R_a + L_a s) T_m(s)}{K_t} + K_b s \theta_m(s) = E_a(s) \quad (24)$$

A Figura 09 demonstra o carregamento típico de um motor.

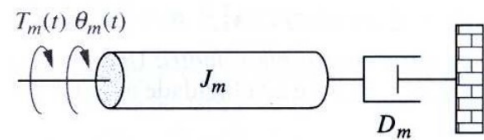


Fig. 09 – Carregamento típico de um motor [14]

Sendo J_m o momento de inércia equivalente na armadura e D_m o amortecimento viscoso equivalente na armadura. A equação 25 de movimento de um motor é demonstrado.

$$T_m(s) = (J_m s^2 + D_m s) \theta_m(s) \quad (25)$$

Substituindo 25 em 24, tem-se a equação 26.

$$\frac{(R_a + L_a s)(J_m s^2 + D_m s)}{K_t} \theta_m(s) + K_b s \theta_m(s) = E_a(s) \quad (26)$$

Considerando $R_a \gg L_a$, tem-se a equação (27).

$$\left[\frac{R_a}{K_t} (J_m s + D_m + K_b) \right] s \theta_m(s) = E_a(s) \quad (27)$$

Determina-se a função de transferência desejada $\frac{\theta_m(s)}{E_a(s)}$, conforme equação 28.

$$\frac{\theta_m(s)}{E_a(s)} = \frac{\frac{K_t}{(R_a J_m)}}{s \left[s + \frac{1}{J_m} \left(D_m + \left(\frac{K_b K_t}{R_a} \right) \right) \right]} \quad (28)$$

A Figura 10 demonstra, um motor de inércia J_a e de amortecimento D_a na armadura acionando uma carga de inércia J_L e amortecimento D_L .

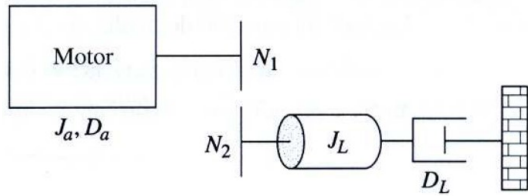


Fig. 10 – Demonstração das variáveis de inércia e amortecimento de um sistema típico [14]

Considerando as impedâncias da carga para a entrada temo a equação 29.

$$J_m = J_a + J_L \left(\frac{N_1}{N_2} \right)^2; D_m = D_a + D_L \left(\frac{N_1}{N_2} \right)^2 \quad (29)$$

Considerando a equação (24), com $R_a \gg L_a$, tem-se a equação 30.

$$\frac{R_a}{K_t} (T_m s + K_b s \theta_m(s) = E_a(s) \quad (30)$$

Aplicando-se a transformada inversa de Laplace na equação 30, tem-se a equação 31.

$$\frac{R_a}{K_t} T_m(t) + K_b \omega_m(t) = e_a(t) \quad (31)$$

Isolando-se $T_m(t)$, tem-se a equação 32

$$T_m(t) = -\frac{K_b K_t}{R_a} \omega_m + \frac{K_t}{R_a} e_{at}(t) \quad (32)$$

Da equação 32, encontra-se o torque de partida, conforme equação 33 e Figura 11.

$$T_{bloq} = \frac{K_t}{R_a} e_{at}(t) \quad (33)$$

E a velocidade angular sem carga, conforme equação 34.

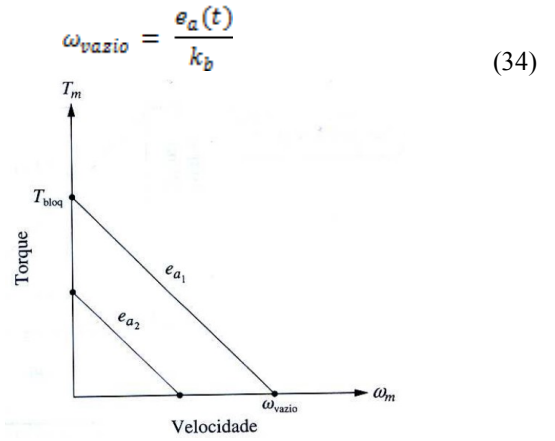


Fig. 11 – Correlação torque – velocidade [14]

As constantes elétricas da função de transferência do motor podem ser determinadas a partir de

$$\frac{K_t}{R_a} = \frac{T_{bloq}}{e_a(t)} \quad \text{e} \quad \frac{K_t}{R_a} = \frac{e_a(t)}{\omega_{vazio}}$$

As constantes elétricas $\frac{K_t}{R_a}$ e K_b , podem ser determinadas como um teste dinamométrico do motor, o qual forneceria T_{bloq} e ω_{vazio} para um dado valor de $e_a(t)$.

O sistema motor – dinamômetro avaliado, pode ser descrito conforme a Figura 12, e sua função transferência é obtida conforme a dedução e aplicação da equação 38, em termos de $\frac{\theta_m(s)}{E_a(s)}$.

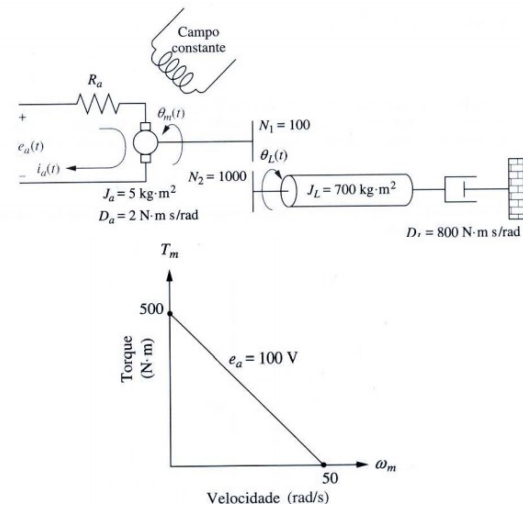


Fig. 12 – Sistema motor – dinamômetro [14]

Inicialmente, calcula-se as impedâncias da carga e armadura do motor, conforme equações 35 e 36;

$$I_m = J_a + J_L \left(\frac{N_1}{N_2} \right)^2 = 5 + 700 \left(\frac{1}{700} \right)^2 = 12 \quad (35)$$

$$D_m = D_a + D_L \left(\frac{N_1}{N_2} \right)^2 = 2 + 800 \left(\frac{1}{10} \right)^2 = 10 \quad (36)$$

Da Figura 12 tem-se representação do gráfico de torque x velocidade; $T_{bloq} = 500$, $\omega_{vazio} = 50$ e $e_a(t) = 100$.

Portanto, as constantes elétricas são:

$$\frac{K_t}{R_a} = \frac{T_{bloq}}{e_a(t)} = \frac{500}{100} = 5$$

$$\frac{K_t}{R_a} = \frac{e_a(t)}{\omega_{vazio}} = \frac{100}{50} = 2$$

Assim a função de transferência $\frac{\theta_m(s)}{E_a(s)}$ resulta na equação,

$$\frac{\theta_m(s)}{E_a(s)} = \frac{5/12}{s[s + \frac{1}{12}(10 + (5)(2))]} \quad (37)$$

Para determinar $\frac{\theta_L}{E_a}$, usamos a relação $\frac{N_1}{N_2} = 1/10$, e resulta na função de transferência motor – dinamômetro, utilizada no controlador PID sintonizado via algoritmos bioinspirados PSO, DE e variações de aprendizagem OBL, simulados via ferramenta Matlab e Simulink. A função de transferência está representada conforme equação 38.

$$\frac{\theta_L}{E_a} = \frac{0.0417}{s(s + 1.667)} \quad (38)$$

III. RESULTADOS E DISCUSSÕES DAS SIMULAÇÕES DOS ALGORITMOS BIOINSPIRADOS DE OTIMIZAÇÃO, CONFOME A FUNÇÃO BENCHMARK

Os resultados dos testes dos algoritmos de otimização são apresentados de forma estatística em tabelas que contém a media, mediana, desvio padrão dos dados obtidos, valor mínimo e número de acertos. Para cada algoritmo PSO e DE e variações de aprendizagem OBL os valores de parâmetros de configuração são apresentados na Tabela 01.

Tabela 01. Parâmetros de configuração dos algoritmos DE, PSO, O-PSO e O-DE

DE		PSO		O-PSO		O-DE	
Parâmetros	Valor	Parâmetros	Valor	Parâmetros	Valor	Parâmetros	Valor
Tamanho do enxame	[20]	Tamanho do enxame	[20]	Tamanho do enxame	[20]	Tamanho do enxame	[20]
Dimensões	3	Dimensões	3	Dimensões	3	Dimensões	3
Iterações	100	Iterações	100	Iterações	100	Iterações	100
Fator de mutação	1,2	Peso inércia	.9 - .1	Peso inércia	.9 - .1	Fator de mutação	1,2
Taxa de crossover	1	Coefficient e Cognitivo e Social	C1=C2 = 2.05	Coefficient e Cognitivo e Social	C1=C2 = 2.05	Taxa de crossover	0,95
		Velocidad e máxima	-3,3	Velocidad e máxima	-3,3		
				limit	40	limit	40

Fonte: Autor

Os algoritmos foram executados 32 vezes para cada conjunto de parâmetros (número de partículas e número de dimensões). Para as 32 execuções as posições iniciais dos agentes foram geradas aleatoriamente. Dos resultados obtidos em cada teste foi escolhida a melhor posição encontrada pelas partículas e o respectivo valor de aptidão (valor mínimo encontrado). Com os resultados dos 32 experimentos foram calculados o valor da média, mediana, valor mínimo, desvio padrão e testes de desempenho não paramétricos para cada algoritmo.

A. Resultados dos Algoritmos de Otimização DE, PSO, O-DE, O- PSO, utilizando a função custo PID.

Tabela 02. Resultados estatístico dos algoritmos

Algoritmo	Média	Mediana	Mínimo	Desvio P	goals/32	P	I	D	
S=20	PSO	-1E+38	-1E+38	1E+25	-1E+38	99,5	100	-70,9	100
N=3	O-PSO	72,449	72,4493	72,424	0,03606	0	50,71	-0,03	39,74
	DE	-8,524	12,054	-17,05	0,03125	59,596	65,44	59,87	
	O-DE	72,453	72,453	72,452	0,0015	0	86,34	0	100

Fonte: Autor

As Figuras de 13 a 16 apresentam os gráficos de resposta do controlador PID a função degrau e aos parâmetros de P, I e D, sintonizados pelos algoritmos.



Fig. 13 – Resposta do controlador sintonizado com PSO



Fig. 14 – Resposta do controlador sintonizado com O-PSO



Fig. 15 – Resposta do controlador sintonizado com DE

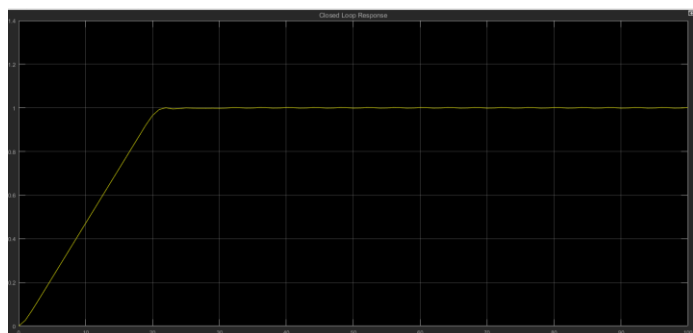


Fig. 16 – Resposta do controlador sintonizado com O-DE

B. Resultados de Convergência

As condições experimentais para cada algoritmo estão descritas na Tabela 01. Os espaços de busca foram limitados conforme Equação 38. O valor de aptidão máximo admissível (*threshold*) foi configurado em 0.01 para a função multimodal.

Conforme resultados descritos na Tabela 02; melhor posição global e da melhor aptidão obtida para cada experimento, foi calculado o valor médio, mediana, desvio padrão e o valor de aptidão mínimo entre todos os experimento, se apresenta as seguintes considerações para os experimentos:

- Com base nos valores médios da tabela 01 e o gráfico da Figura 13, é possível afirmar que o Algoritmo PSO não apresenta soluções satisfatórias para a sintonia do controlador.
- Com base nos valores médios da tabela 01, desvio padrão e os gráficos das Figuras 14 e 16, é possível afirmar que os algoritmos O-PSO e O-DE apresentam soluções satisfatórias para a sintonia do controlador. Os tempos de subida e de estabilização ficam próximos de 20t e o erro de estado estacionário se mantém estável próximo ao *set point* de valor 1. . A outra observação é a demonstração é que a variação de aprendizagem OBL favoreceu a solução dos algoritmos.
- Com base na tabela 01; mediana, desvio padrão e o gráfico da Figura 15, é possível afirmar que o algoritmo DE apresenta solução satisfatória para a sintonia do controlador. Os tempos de subida próximo a 3t e de estabilização menor que 20t e o erro de estado estacionário se mantém estável próximo ao *set point* de valor 1. Como a componente integral é apresentada com valores relativamente altos, ocorre um baixo tempo de subida e consequente sobrepasso, o que para sistemas que requerem um baixo tempo de resposta é satisfatório, desde que o sobrepasso esteja dentro de parâmetros aceitáveis.
- O desempenho dos algoritmos aumenta conforme o incremento do número de iterações, apresentando valores das componentes P, I e D mais ajustado e consequente melhor resposta ao degrau.

IV. CONCLUSÕES E TRABALHOS FUTUROS

Os resultados obtidos destaca uma visão das potencialidades dos algoritmos para resolver problemas de otimização multimodal aplicado a sintonia de controladores PID. Na perspectiva de otimizar o desempenho é possível realizar um processo de ajuste dos parâmetros de cada algoritmo (valores de inércia, coeficiente cognitivo, entre outras, permitindo explorar as capacidades de busca local e global. Adicionalmente, o incremento do número de iterações

permite que as partículas tenham mais oportunidades de convergir para uma solução. As soluções apresentadas para os parâmetros P, I e D, no processo de sintonia do controlador associada a sua função custo motor-dinamômetro, utilizando os algoritmos bioinspirados O-PSO, O-DE e DE são adequadas.

A. Trabalho Futuro – Realizar aproximação da modelagem da função de transferência motor – dinamômetro, para aplicações Motor de Combustão Interna - Dinamômetro Elétrico com testes em campo para validação das simulações experimentais.

V. REFERÊNCIAS BIBLIOGRÁFICAS

- [01] Adriane Beatriz de Souza Serapião, "Fundamentos de otimização por inteligência de enxames: uma visão geral," Revista Controle & Automação/Vol.20, 2009.
- [02] Diago, J. P. (2015). Otimização de controle de tráfego em grupo de elevadores com algoritmos bioinspirados. Dissertação de Mestrado em Sistemas Mecatrônicos, Publicação ENM.DM-088A/15, Departamento de Engenharia Mecânica, Universidade de Brasília, Brasília, DF, 105p.
- [03] Nelson Guerra Álvarez and Broderick Crawford Labrín, "Optimización de funciones a través de Optimización por Enjambre de Partículas y Algoritmos Genéticos," in Conferencia Latino-americana de Informática, Santiago, Chile, 2006.
- [04] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," International Symposium Micro Machine and Human Science. Nagoya, Japan: IEEE, 1995.
- [05] Simón Da Rosa et al., "A comparison among stochastic optimization algorithms for parameter estimation of biochemical kinetic models," Applied Soft Computing, 2013.
- [06] Jian Liu, Chengdong Wu, Meiju Liu, Enyang Gao, and Guojiang Fu, "RBF Optimization Control Based on PSO for Elevator Group System," International Conference on Information Science and Technology, 2011.
- [07] Daniel Muñoz, otimização por inteligência de enxames usando arquiteturas paralelas para aplicações embarcadas: UnB- Universidade de Brasília, 2012.
- [08] Dervis Karaboga and Bahriye Basturk, "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm," Journal of Global Optimization, v. 39, 2007.
- [09] Shi, Y., Eberhart, R. "A modified particle swarm optimizer", Proc. IEEE Congress on Computational Intelligence, Anchorage, Alaska, USA, 1998. p. 69-73.)
- [10] Rahnamayan, S., Tizhoosh, H. R., Salama, M. M. A., "Opposition-Based Differential Evolution Algorithms", Proceedings of IEEE Congress on Evolutionary Computation, 2006.
- [11] Shahryar Rahnamayan, Hamid R. Tizhoosh, and Magdy M.A. Salama, "Opposition versus randomness in soft computing techniques," Journal Applied Soft Computing, 2008.
- [12] Notas de aula – Sistemas Bioinspirados para Engenharia – Daniel Muñoz – UNB -2017.
- [13] W. Webb Ronald A. Reis, PID Control of Continuous Processes by John Programmable Logic Controllers, Fourth Edition, Prentice Hall PTR.
- [14] Notas de aula – Sistemas Sistema de Controle – Cristiano Quevedo – UTFPR -2012.