DOI: 10.48011/asba.v2i1.1356

Identificação de Sistemas Dinâmicos Não Lineares Utilizando Modelos Neuro-Fuzzy Lineares Locais com um Algoritmo LOLIMOT-PSO

Maria Emília Andrade Borges * **Tiago G. de Oliveira** *
Luiz F. Pugliese * **Fadul F. Rodor** *

* *Instituto de Ciências Tecnológicas - ICT*
Universidade Federal de Itajubá – Campus Itabira
(e-mail: maria.borges96@outlook.com, tgaiba@unifei.edu.br, pugliese@unifei.edu.br, fadulrodor@unifei.edu.br).

Abstract: This work proposes the use of the particle swarm optimization algorithm to determine the subspace division points of a given input dimension using the Neuro-Fuzzy model training algorithm known as LOLIMOT (Local Linear Model Trees). The proposal was evaluated in two nonlinear dynamic systems: a NARX model (Nonlinear Autoregressive Exogenous) and a level system. Monte Carlo simulations were performed to analyze the effect of random initialization of the PSO algorithm. The results were compared with the conventional LOLIMOT algorithm and in all tests it was possible to observe an improvement regarding the cost function.

Resumo: Este trabalho propõe o uso do algoritmo de otimização baseado em enxame de partículas para a determinação dos pontos de divisão do subespaço de uma dada dimensão de entrada utilizando algoritmo de treinamento de modelos *Neuro-Fuzzy* conhecido como LOLIMOT (*Local Linear Model Trees*). A proposta foi avaliada em dois sistemas dinâmicos não lineares, sendo um modelo NARX (*Nonlinear Autoregressive Exogenous*) e um sistema de nível. Simulações de Monte Carlo foram efetuadas para analisar o efeito da inicialização aleatória do algoritmo PSO. Os resultados foram comparados com o algoritmo LOLIMOT convencional e em todos os casos foi possível observar uma melhora com relação a função de custo.

Keywords: Nonlinear Systems Identification; Local Linear *Neuro-Fuzzy* Models; LOLIMOT; PSO.

Palavras-chaves: Identificação de sistemas não lineares; Modelos *Neuro-Fuzzy* Lineares Locais; LOLIMOT; PSO.

1. INTRODUÇÃO

Com o objetivo de obter modelos cada vez mais exatos e parcimoniosos, várias representações foram propostas ao longo dos anos, principalmente no que tange a representação de sistemas não lineares. Uma dessas são os chamados modelos *Neuro-Fuzzy* com modelos lineares locais (MLL) ou modelos *Fuzzy* Takagi-Sugeno (Nelles, 2001). Esses modelos podem ser interpretados como redes neurais de base radial (*Radial Basis Funcion* - RBF) normalizadas sob algumas condições. Neste contexto, Nelles (1997) propôs o algoritmo conhecido como LOLIMOT (*Local Linear Model Tree*) sendo esse o foco desse trabalho.

Desde sua concepção, o algoritmo sempre chamou atenção por sua construção intuitiva e vem sendo aplicado em diversos trabalhos. Em Petchinathan et al. (2014) esta técnica foi comparada ao ANFIS (*Adaptive Neuro-Fuzzy Inference System*) para modelagem de um experimento envolvendo neutralização de pH, apesar de em termos gerais ambas se mostrarem válidas para modelagem de sistemas em tempo real complexos e não lineares, LOLIMOT apresentou resultados melhores, precisando de menor tempo de treinamento e apresentando erro quadrático médio menor.

Schaffnit et al. (2000) evidenciam a facilidade de controle da complexidade do algoritmo e a boa interpretabilidade dos resultados ao utiliza-lo para obter o modelo de um turbo compressor com turbina de geometria variável.

Esse trabalho propõe uma metodologia para treinamento de modelos *Neuro-Fuzzy* com modelos lineares locais baseada no algoritmo LOLIMOT em conjunto com uma otimização por enxame de partículas (*Particle Swarm Optimization* - PSO) (Kennedy e Eberhart, 1995), tendo em vista a velocidade de convergência, facilidade de configuração e dos poucos parâmetros da técnica. A proposta será aplicado em diferentes sistemas e comparada com a metodologia convencional.

O artigo será dividido da seguinte forma: a Seção 2 apresenta os modelos *Neuro-Fuzzy* e seu treinamento utilizando o LOLIMOT, a Seção 3 apresenta uma breve descrição do algoritmo PSO, a Seção 4 apresenta a metodologia proposta, a Seção 5 expõe os resultados encontrados com

a metodologia proposta e a Seção 6 as considerações finais do trabalho.

2. MODELOS *NEURO-FUZZY* LINEARES LOCAIS

2.1 Estrutura

A estrutura da rede de um modelo *Neuro-Fuzzy* MLL é formada por neurônios compostos por um MLL e uma função de validação associada, que determina a região de ativação deste, conforme a Figura 1.

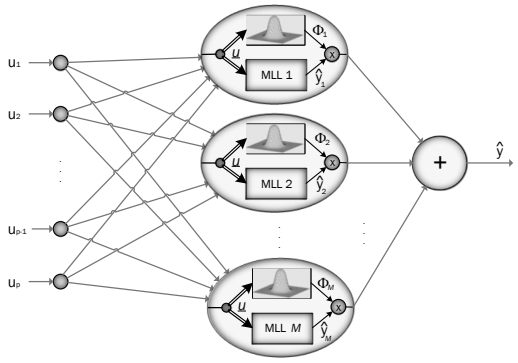

Figura 1. Estrutura de uma rede *Neuro-Fuzzy* com modelos lineares locais com M neurônios e p entradas.

A saída $\hat{y}_i$ dos MLLs é definida na Equação (1),

$$\hat{y}_i = w_{i0} + w_{i1}u_1 + w_{i2}u_2 + ... + w_{ip}u_p \quad (1)$$

onde w_{ij} são os parâmetros do i-ésimo neurônio.

As funções de validação são normalmente escolhidas como Gaussianas normalizadas, e caso essas sejam ortogonais, são descritas conforme Equação (2),

$$\Phi_i(\underline{u}) = \frac{\mu_i(\underline{u})}{\sum_{j=1}^{M} \mu_j(\underline{u})} \quad (2)$$

sendo que, $\mu_i(\underline{u})$ é o parâmetro que mede o grau de associação de uma amostra com determinado *cluster*, e é dado pela Equação (3).

$$\mu_i(\underline{u}) = exp\left\{-\frac{1}{2}\sum_{j=1}^{p}\frac{(u_j - c_{ij})^2}{\sigma_{ij}^2}\right\} \quad (3)$$

A função descrita em (2) é normalizada de forma que a contribuição local da validação dos M neurônios resulte em uma contribuição global, ou seja, some 100%, conforme Equação (4),

$$\sum_{i=1}^{M} \Phi_i(\underline{u}) = 1 \quad (4)$$

onde $\underline{u}$ é o vetor de entradas $\underline{u} = [u_1\ u_2\ ...\ u_p]^T$.

Com isso, a saída $\hat{y}$ da rede é calculada somando a contribuição dos M modelos lineares, conforme Equação (5).

$$\hat{y} = \sum_{i=1}^{M} \hat{y}_i(\underline{u})\Phi_i(\underline{u}) \quad (5)$$

2.2 Algoritmo LOLIMOT

O LOLIMOT é um algoritmo incremental baseado na construção de modelos utilizando a decomposição das regiões de pior desempenho em relação ao espaço de dados de entradas em regiões retangulares ortogonais aos eixos, conforme ilustrado na Figura 2 (Schaffnit et al., 2000; Nelles, 2001).

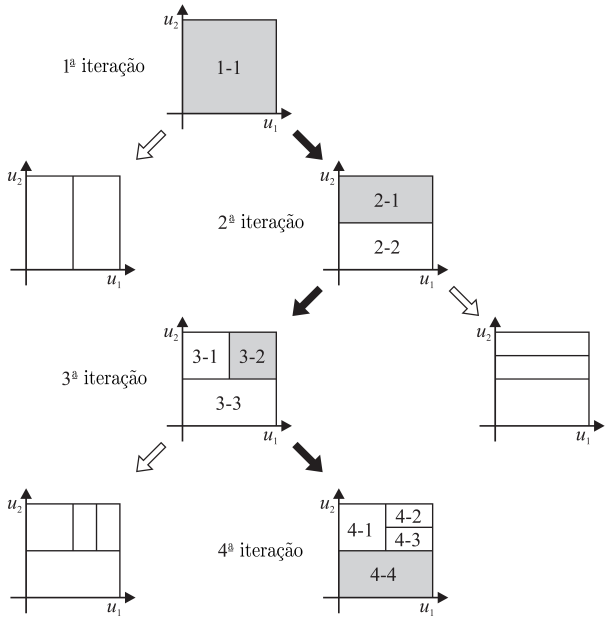

Figura 2. Estrutura de treinamento do algoritmo LOLIMOT.

A cada iteração são avaliadas as condições para decomposição do espaço e é adicionado um novo MLL otimizado por técnicas de mínimos quadrados ponderada (MQP). Sendo necessário o ajuste a *priori* do fator de proporcionalidade entre a extensão dos hiper-retângulos e os desvios padrão, k_σ.

O valor ideal de k_σ depende da aplicação específica, mas geralmente adotando-se $k_\sigma = 1/3$ bons resultados são alcançados (Nelles, 2001). Os desvios padrão são calculados pela Equação (6)

$$\sigma_{ij} = k_\sigma \Delta_{ij} \quad (6)$$

em que Δ_{ij} representa a extensão do hiper-retângulo do modelo local i na j-ésima dimensão das entradas $\underline{u}$, e c_{ij} é o centro do mesmo.

O algoritmo LOLIMOT consiste em um *loop* externo no qual a estrutura da regra é determinada e um *loop* interno alinhado, no qual os parâmetros consequentes da regra são otimizados por estimativa local (Nelles et al., 2000).

Para implementação do algoritmo, Nelles (2001) propõe as seguintes etapas fundamentais:

- Comece com um modelo inicial: São construídas as funções de validação para o particionamento do espaço de entrada e estimado os parâmetros dos MLL pelo MQP. Como nenhum particionamento de espaço de entrada está disponível a *priori*, nesta etapa utiliza-se $M = 1$ e o MLL passa a ser um modelo

linear global, pois sua função de validação cobre todo o espaço de entrada, $\Phi_1(\underline{u}) = 1$;

- Determinação do pior MLL: É calculado uma função de custo local para cada um dos MLLs, ponderando os erros do modelo quadrado com o grau de validade do modelo local correspondente, de acordo com Equação (7);

$$I_i = \sum_{j=1}^{N} e^2(j)\Phi_i(\underline{u}(j)) \tag{7}$$

  É atribuído a l o MLL de pior desempenho, $max(I_i)$;

- Refinamento adicional: O l-ésimo MLL é selecionado para refinamento, realizando-se uma divisão ortogonal adicional, sendo testada em todas as dimensões;

- Verificação das divisões: A melhor dentre as alternativas da etapa anterior é selecionada, atualizando a função de validação e incrementando o MLL, $M = M+1$;

- Teste de Convergência: Nesta etapa é verificado se todos os critérios foram atendidos e o resultado desejado encontrado. Caso positivo o algoritmo é finalizado, caso negativo retoma-se a etapa de determinação do pior MLL e é realizada uma nova divisão.

Na proposta original, a divisão do subespaço formado pela dimensão de entrada avaliada é sempre dividido pela metade (Nelles, 2001).

3. OTIMIZAÇÃO POR ENXAME DE PARTÍCULAS

O algoritmo de otimização por enxame de partículas foi originalmente introduzido por Eberhart (1995), simulando o comportamento de espécies socialmente organizadas em grupos (Clerc, 2002, 2010), como alcateias, enxames, matilhas, cardumes, etc.

3.1 Algoritmo PSO

O algoritmo é composto por M indivíduos (partículas), que se movem em um espaço de busca de forma a influenciar uns aos outros. Cada partícula possui N propriedades (dimensões do espaço de busca), que definem três vetores N-dimensionais associados a cada um desses indivíduos: vetor de posição atual ($X_{i,n}$), vetor de velocidade ($V_{i,n}$) e vetor melhor posição individual ($P_{i,n}$). Para n-ésima iteração, os vetores da i-ésima, ($1 \leq i \leq M$) são:

(1) $X_{i,n} = (X^1_{i,n}, X^2_{i,n}, \ldots, X^j_{i,n}, \ldots, X^N_{i,n})$, sendo que cada componente do vetor é uma variável de decisão do problema ($1 \leq j \leq N$);

(2) $V_{i,n} = (V^1_{i,n}, V^2_{i,n}, \ldots, V^j_{i,n}, \ldots, V^N_{i,n})$, na qual cada componente indica o incremento da velocidade atual ($1 \leq j \leq N$);

(3) $P_{i,n} = (P^1_{i,n}, P^2_{i,n}, \ldots, P^j_{i,n}, \ldots, P^N_{i,n})$, também representando incrementos da posição atual ($1 \leq j \leq N$).

A posição inicial de cada partícula, $X_{i,0}$ é definida de forma aleatória dentro do espaço de busca da j-ésima dimensão, sendo $1 \leq j \leq N$ e o espaço limitado por $[X^j_{min}, X^j_{max}]$ no qual X^j_{min} e X^j_{max} são, respectivamente, os limites inferior e superior da posição das partículas. De forma análoga, a velocidade inicial de cada partícula $V_{i,0}$ pode ser escolhido de maneira randômica dentro do intervalo $[-V^j_{max}, V^j_{max}]$ sendo ($1 \leq j \leq N$) e $P_{i,n}$ pode ser inicializado como o vetor de posição inicial $P_{i,0} = X_{i,0}$.

Na n-ésima iteração, avalia-se a função objetivo para o vetor de melhor posição individual de cada partícula. O melhor resultado é armazenado em $P_{g,n}$, no qual g denota o indexador da melhor partícula. Esse valor é armazenado no vetor de melhor posição global ($G_n = G^1_n, G^2_n, \ldots, G^N_n$).

O vetor $P_{i,n}$ é dado pela relação (8)

$$P_{i,n} = \begin{cases} X_{i,n} & \text{se } f(X_{i,n}) < f(P_{i,n-1}), \\ P_{i,n-1} & \text{se } f(X_{i,n}) \geq f(P_{i,n-1}) \end{cases} \tag{8}$$

logo, G_n é definido pela Equação (9)

$$G_n = P_{g,n}, \text{ na qual } g = \arg \min_{(1 \leq i \leq M)} [f(P_{i,n})]. \tag{9}$$

A atualização do vetor velocidade é dada pela Equação (10) e do vetor de posições pela Equação (11).

$$V^j_{i,n+1} = V^j_{i,n} + c_1 r^j_{i,n}(P^j_{i,n} - X^j_{i,n}) + c_2 R^j_{i,n}(G^j_n - X^j_{i,n}) \tag{10}$$

$$X^j_{i,n+1} = X^j_{i,n} + V^j_{i,n+1} \tag{11}$$

nas quais, $i = 1, 2, \ldots, M$, $j = 1, 2, \ldots, N$, c_1 representa o parâmetro de aceleração cognitivo e c_2 representa o parâmetro de aceleração social. Os parâmetros $r^j_{i,n}$ e $R^j_{i,n}$ são duas sequências diferentes de números aleatórios compreendidos entre $[0, 1]$.

Uma variação da atualização do vetor de velocidades é dada pela equação (12) (Shi e Eberhart, 1998).

$$V^j_{i,n+1} = wV^j_{i,n} + c_1 r^j_{i,n}(P^j_{i,n} - X^j_{i,n}) + c_2 R^j_{i,n}(G^j_n - X^j_{i,n}) \tag{12}$$

na qual w, dado pela Equação (13), é chamado de fator de inércia e seu valor determina o comportamento do algoritmo, sendo um valor alto uma busca global e um valor baixo uma busca local.

$$w = w_{max} - \frac{(w_{max} - w_{min})}{n_{max}} n, \tag{13}$$

no qual w_{max} e w_{min} correspondem aos valores máximos e mínimos, respectivamente, do fator de inércia, n_{max} é o número máximo de iterações e n é a iteração atual.

4. METODOLOGIA PROPOSTA

Neste trabalho é proposta uma maneria alternativa para a obtenção dos pontos de divisão dos subespaços formados pelos dados de entrada para a construção de modelos lineares locais *Neuro-Fuzzy*, uma vez que na proposta original os cortes são sempre realizados na metade do subespaço em questão. Para tanto, foi utilizada como base a *toolbox* desenvolvida em (Nelles et al., 2000),

modificando o código para inserir o processo de seleção do ponto de corte com base no resultado do PSO.

A cada rodada do algoritmo, novos pontos de corte são testados, gerando-se um valor de aptidão *(fitness)*. A convergência será alcançada ao atingir o número máximo de iterações pré-determinado ou outro critério de parada (um erro mínimo, por exemplo). Esse procedimento é realizado em cada dimensão da entrada, para só então decidir em qual dimensão a divisão será realizada com a taxa determinada pelo algoritmo PSO.

5. RESULTADOS

Com o intuito de demonstrar a metodologia proposta a identificação de dois sistemas dinâmicos não lineares foram desenvolvidas. Para cada sistema foi gerada uma simulação de Monte Carlo com 100 inicializações aleatórias do algoritmo LOLIMOT-PSO, extraindo dessas simulações o número médio de parâmetros e o erro quadrático médio (*Mean Squared Error* - MSE) dos dados de validação para cada modelo obtido. Além disso, são apresentados os gráficos das respostas do melhor modelo obtido para cada uma das simulações, sendo os resultados comparados com o algoritmo LOLIMOT convencional por meio de tabelas para cada exemplo.

Para todos os casos, o número máximo de modelos lineares locais foi limitado em 50. Os parâmetros do PSO foram mantidos iguais em todos os exemplos, sendo:

- Os parâmetros cognitivo e social (c_1 e c_2) foram selecionados como 2 (Shi e Eberhart, 1998);
- O número de partículas foi escolhido como 10 e o valor máximo de velocidade (V_{max}) foi determinado empiricamente como 0,1, uma vez que os valores do parâmetro otimizado (as taxas de divisão) deve estar limitado no intervalo [0,1];
- O valor de inércia do PSO foi determinado a partir de outros trabalhos (Shi e Eberhart, 1998), variando de forma linear, conforme a Equação (13), com $w_{max} = 1,2$, $w_{min} = 0,6$;
- Como critérios de parada considerou-se uma variação mínima de 10^{-4} nos valores do *fitness* ou um número de passos de 100.

5.1 Modelo NARX

No exemplo aqui apresentado utiliza-se um sistema não linear que exibe comportamentos dinâmicos distintos em diferentes regiões da zona de operação (Lazar, 2001).

$$y(k) = \frac{2,5y(k-1)y(k-2)}{1+y(k-1)^2+y(k-2)^2} + \\ + 0,3\cos\{(0,5(y(k-1)+y(k-2))) + 1,2u(k-1)\} \quad (14)$$

Foram coletados dois conjuntos de dados contendo 300 pares de amostras de entrada e saída, com entradas variando entre 0 e 2 em degraus aleatórios de distribuição uniforme e duração de 10 amostras. A Figura 3 apresenta os dados de entrada e saída utilizados no processo de estimação do modelo.

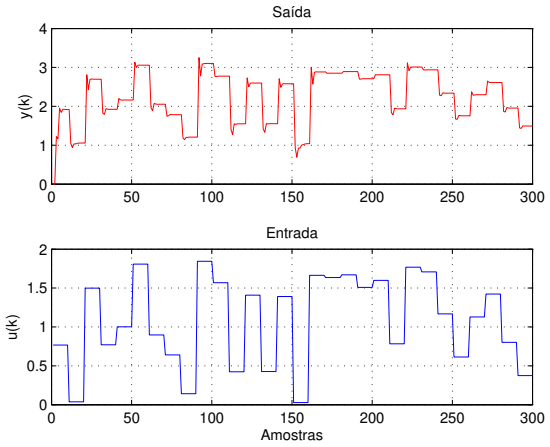

Figura 3. Dados de identificação do sistema NARX.

Os dados utilizados no processo de validação podem-se observar na Figura 4.

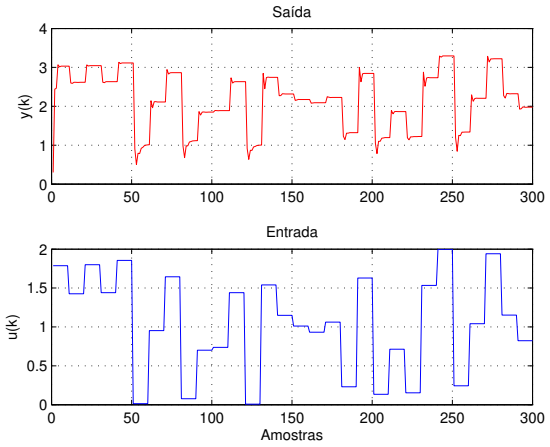

Figura 4. Dados de validação do sistema NARX.

A Figura 5 apresenta a resposta do modelo com menor MSE obtido após a simulação de Monte Carlo. Os resultados das simulações estão apresentados na Tabela 1.

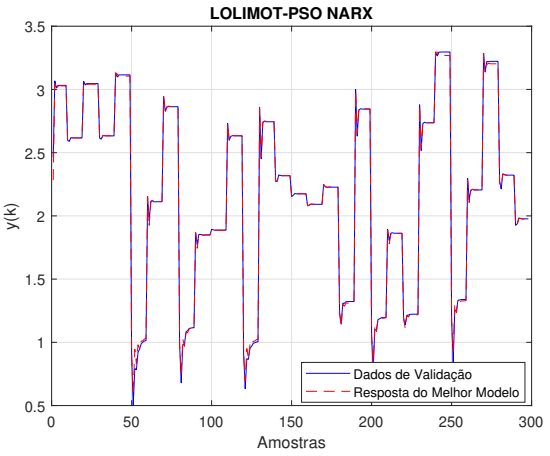

Figura 5. Resposta do modelo com menor MSE para o sistema NARX.

Tabela 1. Resultados dos algoritmos - NARX

Algoritmo	**Número Médio de Parâmetros**	**MSE**
LOLIMOT	35	$6,3327 \times 10^{-4}$
LOLIMOT-PSO	31,1	$5,2825 \times 10^{-4}$

O histograma da Figura 6 apresenta o comportamento dos resultados das simulações.

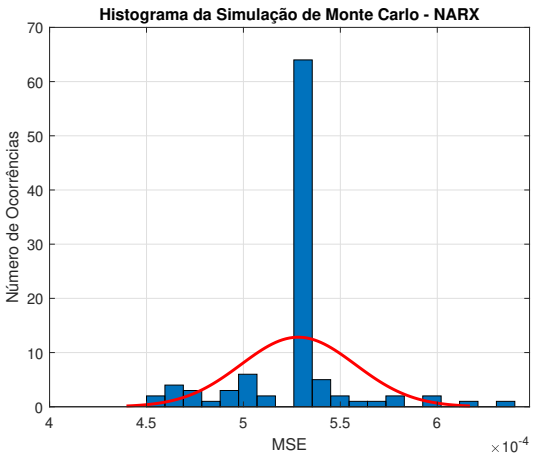

Figura 6. Histograma da simulação de Monte Carlo para o sistema NARX.

Os resultados apresentados acima comprovam a efetividade da metodologia proposta. O número médio de parâmetros das simulações foi inferior ao modelo obtido pelo algoritmo convencional, resultando ainda em modelos com maior exatidão.

5.2 Sistema de Nível Líquido

A planta de nível tem seu esquema de funcionamento retratado na Figura 7, na qual pode-se observar a transferência de líquido entre dois tanques por uma bomba. A saída y é o nível de água no tanque (centímetros) e a entrada u é a tensão (Volts) aplicada ao sistema que bombeia a água para o tanque superior (Lindskog, 1996).

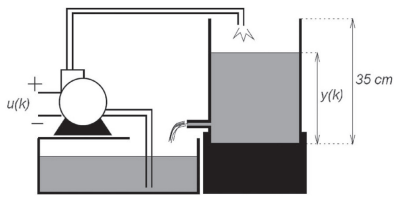

Figura 7. Sistema de Nível.

A vazão de entrada (q_{in}) de um determinado fluido no processo de nível é definida pela Equação (15),

$$q_{in} = kV(t) \tag{15}$$

na qual $V(t)$ é a tensão de acionamento aplicada a servo-bomba do processo, k é uma constante de proporcionalidade relacionada com a tensão $u(t)$ de comando do *drive* da bomba de recalque. A vazão de saída (q_{out}) do sistema é dada pela Equação (16),

$$q_{out} = a_{out}\sqrt{2gy(t)} \tag{16}$$

na qual a_{out} é a área da tubulação de saída, g é a aceleração da gravidade e $y(t)$ é o nível do líquido no tanque no instante t. Assim, a equação que define o nível do tanque em um instante de tempo t é dada pela Equação (17),

$$y(t) = y(0) + \frac{1}{A}\int_0^t (q_{in}(\tau) - q_{out}(\tau))d\tau \tag{17}$$

na qual A é a área transversal do reservatório.

Lindskog (1996), obteve dois conjuntos de dados que serão utilizados neste trabalho, um para identificação e outro para validação, cada um contendo 1000 pares de dados de entrada e saída amostrados a uma taxa de 1 segundo. Os dados utilizados no processo de identificação podem ser observados na Figura 8.

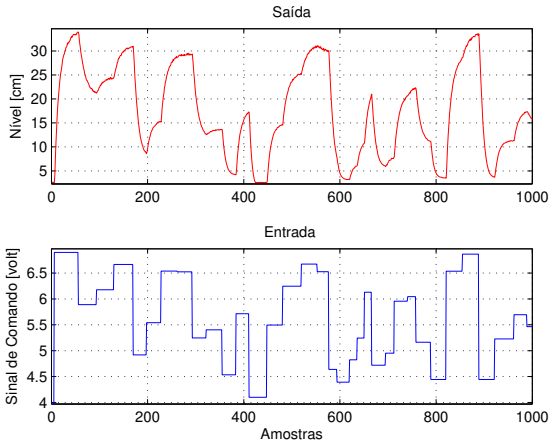

Figura 8. Dados de identificação do sistema de nível.

Enquanto os dados utilizados no processo de validação podem ser observados na Figura 9.

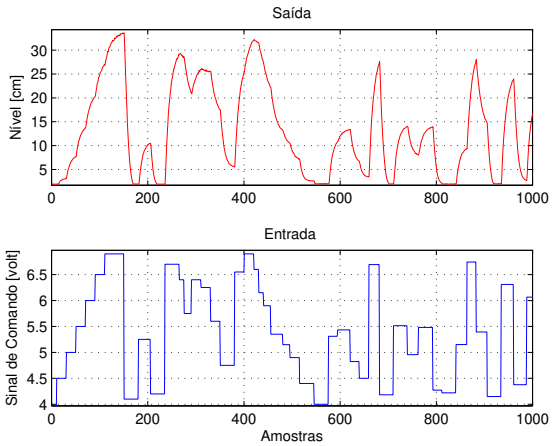

Figura 9. Dados de validação do sistema de nível.

A Figura 10 apresenta a resposta do modelo com menor MSE obtido após a simulação de Monte Carlo. Os resultados das simulações estão apresentados na Tabela 2.

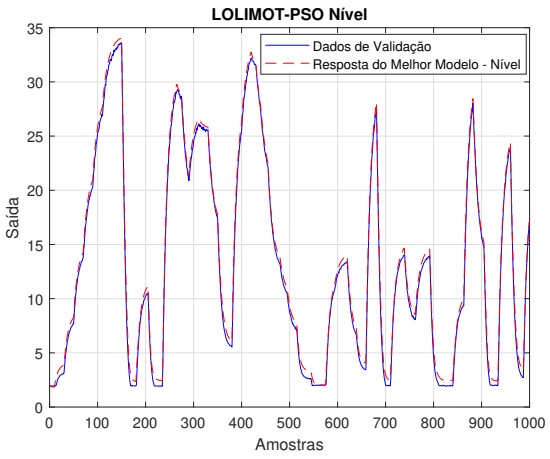

Figura 10. Resposta do modelo com menor MSE para o sistema de nível.

Tabela 2. Resultados dos algoritmos - Nível

Algoritmo	**Número Médio de Parâmetros**	**MSE**
LOLIMOT	33	0,3904
LOLIMOT-PSO	26,40	0,3734

O histograma da Figura 11 apresenta o comportamento dos resultados das simulações.

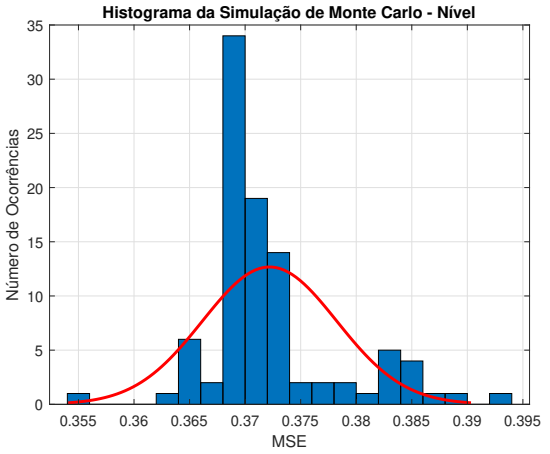

Figura 11. Histograma da simulação de Monte Carlo para o sistema de Nível.

Novamente a metodologia proposta resultou em modelos com menor número de parâmetros que a abordagem convencional, resultando ainda em uma sútil melhoria na exatidão em relação ao LOLIMOT.

6. CONCLUSÃO

Este trabalho apresentou uma proposta de metodologia para a determinação dos pontos de divisão do subespaço de uma determinada dimensão de entrada para o algoritmo de treinamento de modelos *Neuro-Fuzzy* conhecido como LOLIMOT.

A proposta foi avaliada em dois sistemas dinâmicos não lineares. Simulações de Monte Carlo foram efetuadas para analisar o efeito aleatório da inicialização e desenvolvimento do algoritmo PSO utilizado para determinar os pontos de divisão e, tendo em vista as distribuições dos erros dos modelos, o fica comprovada a convergência da metodologia. Em todos os casos foi possível observar uma melhora com relação a função de custo avaliada na comparação dos modelos.

Como citado por Nelles (2001), o efeito da seleção dos pontos de divisão dos modelos locais pode aumentar o esforço computacional demasiadamente, e o ganho em termos da função de custo tende a ser ofuscado com o aumento de modelos locais no algoritmo convencional. Porém, a evolução dos sistemas computacionais e a constante necessidade de obter modelos mais exatos exige uma constante reavaliação dessas técnicas.

REFERÊNCIAS

Clerc, M.and Kennedy, J. (2002). The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *International Journal of Modelling, Identification and Control*, 6(1), 58–73.

Clerc, M. (2010). *Particle swarm optimization*, volume 93. John Wiley & Sons.

Eberhart, R. C.and Kennedy, J. (1995). A new optimizer using particle swarm theory. In *Proceeding of 6th Int. Symp. Micro Machine and Human Science*, 39–43. IEEE.

Kennedy, J. e Eberhart, R.C. (1995). Particle swarm optimization. In *Proceedings of the IEEE International Conference on Neural Networks*, 1942–1948. IEEE.

Lazar, M. (2001). *Nonlinear controller based on the EPSAC approach*. Master's thesis, University of Ghent, Departament of Control, Engineering and Automation, Flanders, Belgium.

Lindskog, P. (1996). *Methods, Algorithms and Tools for System Identification Based on Prior Knowledge*. Ph.D. thesis, Linkoping University, Sweden.

Nelles, O. (1997). Orthonormal basis functions for nonlinear system identification with local linear model trees (LOLIMOT). *IFAC Proceedings Volumes*, 30(11), 639–644.

Nelles, O. (2001). *Nonlinear System Identification:*. Springer-verlag Berlin Heidelberg, 1 edition.

Nelles, O., Fink, A., e Isermann, R. (2000). Local linear model trees (LOLIMOT) toolbox for nonlinear system identification. *IFAC Proceedings Volumes*, 33(15), 845–850.

Petchinathan, G., Valarmathi, K., Devaraj, D., e Radhakrishnan, T.K. (2014). Local linear model tree and neurofuzzy system for modelling and control of an experimental ph neutralization process. *Brazilian Journal Of Chemical Engineering*, 31(2), 483–495.

Schaffnit, J., Nelles, O., Isermann, R., e Schmid, W. (2000). Local linear model trees (LOLIMOT) for nonlinear system identification of a turbocharger with variable turbine geometry (VTG). *IFAC Proceedings Volumes*, 33(15), 615–620.

Shi, Y. e Eberhart, R. (1998). A modified particle swarm. In *Proceedings of 1998 IEEE International Conference on Evolutionary Computation*, 1945–1950. IEEE.