REDE NEURO-FUZZY EVOLUTIVA PARA ESTIMAÇÃO EM TEMPO REAL DE POSIÇÃO DE UM SISTEMA DE LEVITAÇÃO MAGNÉTICA

Alisson Marques Silva†*, Walmir Caminhas†, André Lemos†, Fernando Gomide‡

*Centro Federal de Educação Tecnológica de Minas Gerais, CEFET-MG - Campus V
Rua Álvares de Azevedo, 400, 35503-822, Divinópolis, MG, Brasil

†Programa de Pós-Graduação em Engenharia Elétrica, UFMG
Av. Antônio Carlos 6627, 31270-901, Belo Horizonte, MG, Brasil

‡Faculdade de Engenharia Elétrica e Computação, UNICAMP
Av. Albert Einstein, 400, 13083-852, Campinas, SP, Brasil

Emails: alissonmarques@cpdee.ufmg.br, caminhas@cpdee.ufmg.br, andrepl@cpdee.ufmg.br, gomide@dca.fee.unicamp.br

Abstract— This paper presents the use of an evolving neuro-fuzzy network (eNFN) for real time position estimation of a magnetic levitation system. The magnetic levitation is a fast, nonlinear, open loop unstable dynamic system. Control and position estimation requires high sampling rates, typically in the order of milliseconds. Experimental results show that the eNFN attains good accuracy, and suggests the neuro-fuzzy network as an effective approach to estimate state variables of fast nonlinear dynamic processes in real time.

Keywords— Evolving Neuro-Fuzzy Network, Magnetic Levitation, Real Time Estimation.

Resumo— Este trabalho apresenta a implementação de uma rede *neuro-fuzzy* evolutiva (eNFN) para estimar a posição de uma esfera em um sistema de levitação magnética em tempo real. O sistema de levitação magnética é não-linear, instável em malha aberta e possui uma dinâmica rápida, o que requer taxas de amostragem da ordem de milésimos de segundo. Os resultados experimentais mostram que a rede eNFN obtém uma boa acurácia na estimação da posição da esfera, sugerindo a rede como uma abordagem promissora para estimar o estado de processos dinâmicos rápidos em tempo real.

Palavras-chave— Rede *Neuro-Fuzzy* Evolutiva, Levitação Magnética, Estimação de Estado em Tempo Real.

1 Introdução

A levitação magnética (MagLev) é uma tecnologia utilizada para eliminar o contato mecânico entre as partes móveis e fixas de um sistema para impedir problemas com atrito, desgaste mecânico, fricção, ruído, e geração de calor e pó metálico. Essa tecnologia é empregada com sucesso em diversas aplicações de engenharia incluindo mancais magnéticos, sistemas de isolamento de vibração, rolamento sem atrito, trens de alta velocidade, modelos de levitação em túnel de vento, dentre outras.

Os sistemas evolutivos são caracterizados por sua capacidade de extrair conhecimento e adaptar sua estrutura e parâmetros em tempo real (tempo de execução) a partir de fluxos de dados. O propósito é tratar e continuamente se adaptar às variações que ocorrem em processos (Kasabov and Filev, 2006).

Este trabalho ilustra a aplicação de uma rede *neuro-fuzzy* evolutiva (eNFN - *Evolving Neo-Fuzzy Neuron Network* - Rede Evolutiva *Neo-Fuzzy Neuron* (Silva et al., 2013)) para estimar em tempo real a posição de uma esfera em um sistema de levitação magnética. A eNFN tem como objetivo estimar a posição da esfera p passos à frente. A posição da esfera é uma das variáveis de estado cujo valor depende da tensão aplicada na bobina que gera o campo magnético.

O restante deste trabalho é organizado como se segue: Seção 2 apresenta o sistema de levitação magnética e o seu modelo matemático; Seção 3 resume a estrutura e a aprendizagem da rede eNFN; Seção 4 descreve em detalhes a metodologia empregada nos experimentos, as simulações e os resultados experimentais; Seção 5 conclui o artigo resumindo suas contribuições e sugerindo trabalhos futuros.

2 Sistema de Levitação Magnética - MagLev

O sistema de levitação magnética considerado neste trabalho consiste no módulo do sistema MagLev produzido pela Feedback Instruments Limited (Feedback, 2006), o qual se destina ao estudo do fenômeno da levitação de uma esfera de aço decorrente da força magnética gerada por uma corrente elétrica circulando em uma bobina. O MagLev é um processo não-linear e instável em malha aberta. Sob o ponto de vista de controle, o objetivo é manter a esfera em uma posição especificada. O módulo MagLev é um processo SISO (*Single Input / Single Output*), isto é, possui uma única entrada, a tensão aplicada à bobina, e uma

única saída, a posição vertical da esfera.

O módulo, ilustrado na Figura 1, possui uma bobina para gerar o campo magnético e um sensor óptico infravermelho para medir a posição vertical da esfera. O controle da posição da esfera é feito somente dentro da faixa de trabalho desse sensor.

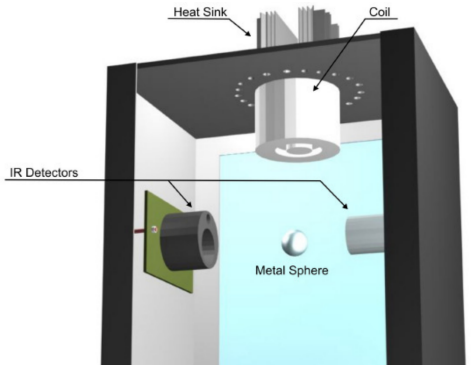

Figura 1: Módulo MagLev - Fonte: (Feedback, 2006).

O software de aquisição de dados e controle da posição da esfera é executado em um microcomputador com sistema operacional Windows e utiliza uma placa de aquisição de dados Advantech, modelo PCI-1711. As unidades de controle são implementadas em ambientes Matlab e Simulink. O sistema operacional Windows não foi desenvolvido para aplicações em tempo real. Por isso utiliza-se o *Real-Time Workshop* do Matlab (MathWorks, 2009) para simular o processamento em tempo real. O *Real-Time Workshop* não executa um processamento determinístico com taxa de amostragem fixa, mas possibilita ao sistema a mais alta prioridade entre todas as tarefas executas pelo sistema operacional. Quando uma tarefa é executada pelo núcleo de processamento em tempo real, o mecanismo de alocação de código do Windows é bloqueado. As tarefas de prioridade mais baixa são executadas nos intervalos de tempo restante. Caso não seja possível obter características de tempo real para uma determinada taxa de amostragem, o núcleo de processamento em tempo real emite um alerta.

2.1 Modelo Matemático do Sistema MagLev

O modelo MagLev, conforme o módulo descrito, possui uma variável de entrada, a tensão (u), e uma variável de estado que é a própria saída, a posição vertical da esfera (x). O modelo não-linear mais simples do sistema de levitação magnética relativo à posição da esfera (x) e a corrente da bobina (i) é dado por (1):

$$\ddot{x} = g - \frac{k}{m}\frac{i^2}{x^2}, \tag{1}$$

onde k é uma constante que depende dos parâmetros da bobina, g é a constante gravitacional e m é a massa da esfera. A Equação 1 mostra que o modelo tem dinâmica tal que o aumento da corrente leva a redução da aceleração. Para apresentar o modelo fenomenológico completo deve ser introduzida uma relação entre a tensão u (variável de controle) e a corrente da bobina i. O MagLev está equipado com um circuito de controle interno que fornece uma corrente proporcional à tensão de controle u:

$$i = k_1 u. \tag{2}$$

As equações 1 e 2 constituem o modelo não-linear implementado no Simulink. Os limites para a variável de controle foram definidos com a amplitude entre $-5V$ e $5V$.

3 Rede Evolutiva *Neo-Fuzzy Neuron* - eNFN

Esta seção descreve a rede *neuro-fuzzy* evolutiva eNFN (Silva et al., 2013). A eNFN foi construída utilizando a estrutura do *Neo-Fuzzy Neuron* (Yamakawa et al., 1992) e utiliza um algoritmo de aprendizado incremental para, simultaneamente, granularizar o espaço de entrada e atualizar os pesos da rede. A estrutura da rede e seus parâmetros evoluem simultaneamente de acordo com o fluxo de dados de entrada.

A estrutura, o algoritmo para atualização dos pesos e o algoritmo evolutivo para a eNFN são resumidos a seguir.

3.1 Neo-Fuzzy Neuron - *NFN*

No caso geral, o modelo *Neo-Fuzzy Neuron* (NFN) pode ser visto como um conjunto de modelos locais do tipo Takagi-Sugeno de ordem zero, um para cada variável de entrada, cada um contendo m_i regras. A saída y_t do NFN em t é computada pela soma das saídas dos modelos individuais y_{ti}, isto é

$$y_t = \sum_{i=1}^{n} y_{ti} = \sum_{i=1}^{n}\sum_{j=1}^{m_i} \mu_{A_{ij}}(x_{ti}) q_{ij}, \tag{3}$$

onde n é o número de variáveis de entrada, m_i é a granularidade do domínio da i-ésima variável de entrada, x_{ti}. As funções de pertinência são triangulares e complementares, portanto no máximo duas das m_i funções de pertinência são ativadas para uma dada entrada x_{ti}. As duas funções ativas são indexadas por k_i e k_i+1, respectivamente. Portanto a saída dos modelos individuais y_{ti} pode ser calculada por (4):

$$y_{ti} = \mu_{A_{ik_i}}(x_{ti}) q_{ik_i} + \mu_{A_{ik_i+1}}(x_{ti}) q_{ik_i+1}. \tag{4}$$

3.2 Atualização dos Pesos da Rede

O algoritmo de aprendizagem é um processo incremental, supervisionado, que ajusta os pesos da rede usando um procedimento baseado no gradiente com taxa de aprendizado ótima. O aprendizado é realizado pela atualização dos pesos q_{ij} para minimizar a função do erro quadrático E_t (5):

$$E_t = \frac{1}{2}(y_t - \hat{y}_t)^2 = E_t(q_{ij}), \tag{5}$$

onde $\hat{y}_t$ é a saída da rede e y_t a saída desejada em t. Como para a i-ésima entrada x_{ti} em t somente as funções de pertinência $\mu_{ik_i}(x_{ti})$ e $\mu_{ik_i+1}(x_{ti})$ estão ativas, a minimização de E_t envolve somente o ajuste dos pesos referentes às funções ativas. Portanto, estes pesos, q_{ik_i}, são atualizados usando

$$q_{ik_i} = q_{ik_i} - \alpha_t(y_t - \hat{y}_t)\mu_{ik_i}(x_{ti}), \tag{6}$$

onde a fórmula para a taxa de aprendizado α (Caminhas and Gomide, 2000) é

$$\alpha_t = \frac{1}{\sum_{i=1}^{n} \mu_{ik_i}(x_{ti})^2 + \mu_{ik_i+1}(x_{ti})^2}. \tag{7}$$

3.3 Evolução da Estrutura da Rede

No algoritmo evolutivo da rede eNFN, os valores modais das funções de pertinência, o número de funções de pertinência e, consequentemente, o número de neurônios podem ser modificados simultaneamente durante a atualização dos pesos. O aprendizado evolutivo consiste nas cinco etapas descritas a seguir.

3.3.1 Inicialização das Funções de Pertinência

A eNFN usa funções de pertinência triangulares e complementares. Uma função de pertinência triangular pode ser definida por três parâmetros: o limite inferior (a), o valor modal (b) e o limite superior (c). Tendo em conta que as funções de pertinência são complementares, a k-ésima função tem (b_k) como seu valor modal e seu limite inferior é o valor modal da função de pertinência adjacente inferior, que é $a_k = b_{k-1}$, e o seu limite superior é o valor modal da função de pertinência adjacente superior, isto é, $c_k = b_{k+1}$.

O modelo inicia-se com o mínimo de duas funções de pertinência para granularizar cada uma das n variáveis de entrada. Novas funções de pertinência podem ser criadas e inseridas de acordo com o fluxo dos dados de entrada e com o erro de modelagem. O valor modal inicial das funções de pertinência são encontrados como se segue

$$\begin{aligned} b_{i1} &= x_{min_i}, \\ b_{i2} &= x_{max_i}, \end{aligned} \tag{8}$$

onde x_{min_i} é o limite inferior e x_{max_i} o limite superior do domínio da i-ésima variável de entrada.

3.3.2 Adaptação de Contexto

Durante a operação de um processo podem ocorrer mudanças que impliquem em dados de entrada com valores diferentes dos definidos inicialmente como limites máximo e mínimo. Portanto, é importante que os valores de x_{min_i} e x_{max_i} sejam atualizados para se adaptarem às condições prevalecentes. O procedimento para atualizar os limites das variáveis de entrada pode ser feito da seguinte forma:

$$\begin{aligned} &se\, x_{ti} < x_{min_i}\, então\, x_{min_i} = x_{ti}\, e\, b_{i1} = x_{min_i}, \\ &se\, x_{ti} > x_{max_i}\, então\, x_{max_i} = x_{ti}\, e\, b_{im_i} = x_{max_i}. \end{aligned} \tag{9}$$

3.3.3 Adaptação dos Valores Modais das Funções de Pertinência

Este passo encontra b_i^*, o índice da função de pertinência mais ativada por x_{ti}. Se $b_i^* > 1$ e $b_i^* < m_i$, então a b_i^*-ésima função de pertinência terá seu valor modal atualizado utilizando (10). Caso contrário, se a função mais ativa for tal que $b_i^* = 1$ ou $b_i^* = m_i$, então o valor modal será mantido pois, neste caso, os valores modais correspondem a x_{min_i} e x_{max_i}, respectivamente.

$$b_{ib_i^*}^{novo} = b_{ib_i^*}^{antigo} + \beta(x_{ti} - b_{ib_i^*}^{antigo}), \tag{10}$$

onde β é a taxa de aprendizado. O valor típico adotado nos experimentos computacionais apresentados neste trabalho é $\beta = 0.01$.

3.3.4 Criação de Funções de Pertinência

A criação e inserção de funções de pertinência refina a granularidade do espaço de entrada visando reduzir o erro de saída de maneira uniforme. Neste trabalho a granularidade do espaço de entrada é feita com base no erro da função de pertinência mais ativa. O valor médio do erro de saída local das regras correspondentes às funções mais ativas são comparados com o valor médio do erro de modelagem global. Se o valor do erro médio local for maior do que o erro médio global, então a região local é refinada com a adição de funções de pertinência como segue.

O valor médio $\hat{\mu}_m$ e a variância $\hat{\sigma}_m^2$ do erro global do modelo são calculados recursivamente para a entrada x_t por (11) e (12):

$$\hat{\mu}_{m_t} = \hat{\mu}_{m_{t-1}} - \beta(\hat{\mu}_{m_{t-1}} - e_t), \tag{11}$$

$$\hat{\sigma}_{m_t}^2 = (1-\beta)(\hat{\sigma}_{m_{t-1}}^2 + \beta(\hat{\mu}_{m_t} - e_t)^2), \tag{12}$$

onde $e_t = y_t - \hat{y}_t$.

Similarmente, a média do erro local $\hat{\mu}_{b_{ti}}$ correspondente a função de pertinência mais ativa (b_i^*) é recursivamente calculada para a entrada x_{ti} usando (13):

$$\hat{\mu}_{b_{ti}} = \hat{\mu}_{b_{t-1i}} - \beta(\hat{\mu}_{b_{t-1i}} - e_t). \quad (13)$$

Para evitar uma granularidade muito fina de uma determinada região do domínio de entrada, insere-se um limitador (τ) para controlar a menor distância entre valor modal da função criada e o valor modal das funções adjacentes. Se $b_i^* \neq 1$ e $b_i^* < m_i$, então a distância entre os valores modais das funções de pertinência após uma nova inserção de funções é dada por (15). Caso contrário, se a função mais ativa é tal que $b_i^* = 1$ ou $b_i^* = m_i$, então a distância é calculada usando (18) e (20), respectivamente. Observe que o número de regras não é determinado *a priori*, pois este é um resultado do algoritmo. A limitação da excessiva granularidade de uma região particular do domínio de entrada é um mecanismo para evitar modelos complexos e *overfitting*. O limitador (τ), contudo, indiretamente permite o controle do número de regras.

Mais precisamente, se $\hat{\mu}_{b_{ti}} > \hat{\mu}_{m_t} + \hat{\sigma}^2_{m_t}$ e $dist > \tau$, então uma nova função de pertinência é criada e inserida. O limitador (τ) é calculado usando (14):

$$\tau = \frac{x_{max_i} - x_{min_i}}{\eta}, \quad (14)$$

onde η é um parâmetro escolhido empiricamente. Um valor típico, o adotado nos resultados computacionais apresentados neste trabalho, é $\eta = 10$.

A criação e inserção de funções de pertinência atualiza a granularidade da i-ésima variável do domínio de entrada como se segue.

Se $b_i^* \neq 1$ e $b_i^* < m_i$, então a função mais ativa será substituída por duas novas funções de pertinência cujos valores modais são dados por (16) e (17).

$$dist = \frac{(b_{ib_i^*+1} - b_{ib_i^*-1})}{3}. \quad (15)$$

$$novo\_b_1 = b_{ib_i^*-1} + dist. \quad (16)$$

$$novo\_b_2 = b_{ib_i^*-1} + 2 * dist. \quad (17)$$

Se $b_i^* = 1$, então uma nova função é criada e inserida entre a primeira e a segunda. O valor modal da nova função de pertinência é dado por (19).

$$dist = \frac{(b_{ib_i^*+1} - b_{i,b_i^*})}{2}. \quad (18)$$

$$novo\_b = b_{ib_i^*} + dist. \quad (19)$$

Se $b_i^* = m_i$, então a nova função de pertinência é criada e inserida entre a última e a penúltima com valor modal dado por (21).

$$dist = \frac{(b_{ib_i^*} - b_{ib_i^*-1})}{2}. \quad (20)$$

$$novo\_b = b_{ib_i^*} - dist. \quad (21)$$

Nos três casos, o erro médio da função de pertinência mais ativa e o número de funções referentes àquela variável de entrada devem ser atualizados. Observe que a criação de uma nova função de pertinência modifica os limites inferiores e superiores das funções adjacentes, mas seus valores modais permanecem inalterados. As modificações nos limites superior e inferior são feitas para manter as funções de pertinência complementares.

3.3.5 Exclusão de Funções de Pertinência

O método para exclusão automática de funções de pertinência e regras *fuzzy* utiliza o conceito de idade da regra (Angelov and Filev, 2005; Lughofer and Angelov, 2011). A idade da regra é determinada pela idade das funções de pertinência do antecedente da regra. A idade de uma função de pertinência é o período de tempo em que ela fica inativa. Portanto, a idade é determinada por $idade_i = t - a_i$, onde a_i é o passo em que a i-ésima função de pertinência foi ativada e t o passo atual.

Para cada variável de entrada i, determina-se b_i^-, o índice da função de pertinência menos ativa habilitada por x_{ti}. A função de pertinência menos ativa é excluída se $idade_{b_i^-} > \omega \; e \; m_i > 2$, onde ω indica a idade limite para uma função de pertinência. O valor utilizado nos experimentos computacionais e nos respectivos resultados apresentados neste artigo é $\omega = 100$.

Após a exclusão de uma função de pertinência a granularidade do espaço de entrada é atualizado conforme descrito a seguir.

Se $b_i^- \neq 1$ e $b_i^- < m_i$, então a função de pertinência menos ativa é eliminada e os limites superiores e inferiores das funções de pertinência adjacentes são ajustados. Os limites são ajustados a fim de manter as funções de pertinência complementares.

Se $b_i^- = 1$, então a função de pertinência menos ativa é eliminada e o valor modal da função adjacente é definido para x_{min_i}. Neste caso, a exclusão da função de pertinência modifica o limite inferior da função de pertinência adjacente a que teve seu valor modal modificado.

Se $b_i^- = m_i$, então a função de pertinência menos ativa é eliminada e o valor modal da função adjacente é definido para x_{max_i}. Aqui, a exclusão da função de pertinência modifica o limite superior da função adjacente a que teve seu valor modal modificado.

4 Experimentos

Esta seção apresenta os experimentos realizados para avaliar o desempenho da eNFN na estimação em tempo real da posição da esfera do módulo MagLev. O controle da posição da esfera é feito por um controlador PID cuja entrada é a diferença entre a posição atual (medida) e a posição desejada, e a saída é a tensão.

Na estimação em tempo real da posição da esfera p passos à frente, a rede eNFN é executada em duas etapas. A primeira etapa visa o ajuste dos parâmetros e a evolução da estrutura. Nesta etapa, a eNFN tem como entrada a posição desejada em $(t-5)$[1] e as medidas da posição em $(t-5)$, $(t-4)$ e $(t-3)$. O valor estimado da posição é comparado com a posição medida atual (y_t) e a diferença (erro) é utilizada para atualizar os parâmetros e adaptar a estrutura da rede. Conclui-se esta etapa fixando parâmetros e estrutura da eNFN. Na segunda etapa estima-se a posição p passos à frente. Nesta etapa, o modelo tem como entrada a posição desejada em $(t-5+p)$ e as medidas da posição em $(t-5+p)$, $(t-4+p)$ e $(t-3+p)$. Utilizam-se estas entradas e os parâmetros e estrutura fixados na primeira etapa para estimar a posição p passos à frente, isto é, $(\hat{y}_{t+p})$. A Figura 2 mostra o diagrama do modelo desenvolvido no Simulink para fazer a estimação em tempo real, enfatizando o diagrama e o funcionamento da rede eNFN.

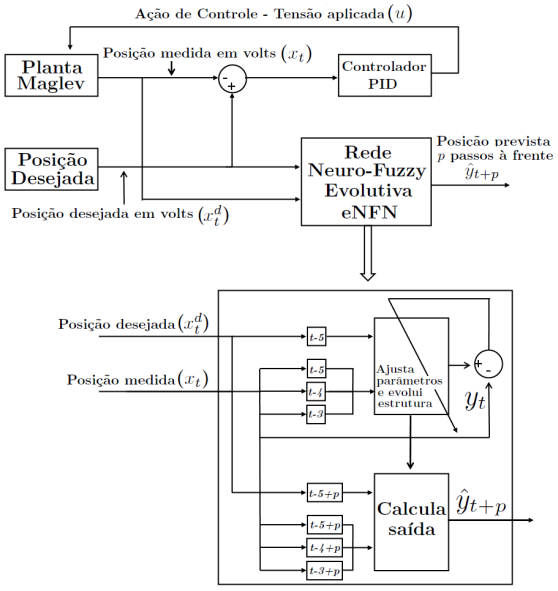

Figura 2: Estimador de estado com eNFN.

A simulação foi executada durante 60 segundos com uma taxa de amostragem de 10^{-4} segundos (1 milissegundo), resultando em um total de 60.000 amostras. A posição desejada foi definida inicialmente por uma função seno com amplitude e frequência iguais a 0.5 Hertz, aos 17 segundos alterou-se a posição desejada para uma função quadrada com amplitude de 0.4 Hertz e frequência 0.5 Hertz, aos 31 segundos de simulação a posição desejada passou a ser um função degrau (com valor definido entre -1 e -2), aos 41 segundos voltou a ter como posição desejada a função seno e, por fim, aos 51 segundos, novamente a função quadrada. O objetivo da alternância entre os valores da posição desejada é analisar o comportamento do modelo evolutivo em diferentes condições operacionais. Espera-se que a eNFN consiga computar a saída para atualização dos parâmetros, evoluir sua estrutura e estimar a posição p passos à frente em tempo inferior ao da taxa de amostragem e com precisão.

Para avaliar o desempenho da eNFN na estimação com $p = 1$, $p = 2$ e $p = 3$ passos à frente utilizou-se o erro quadrático E_t (5). A Figura 3 mostra: (A) a posição medida da esfera; (B) a evolução da estrutura da eNFN; (C), (D) e (E) o erro quadrático instantâneo E_t para a estimação da posição com 1, 2 e 3 passos à frente, respectivamente.

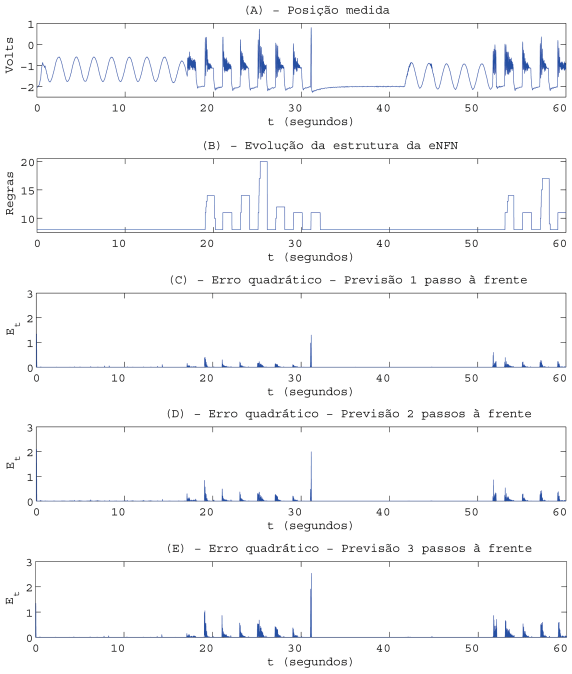

Figura 3: (A) Posição medida da esfera, (B) Evolução da estrutura da eNFN, (C) Erro quadrático na estimação com 1 passo à frente, (D) Erro quadrático na estimação com 2 passos à frente, (E) Erro quadrático na estimação com 3 passos à frente.

Analisando os resultados apresentados na Figura 3 dois momentos podem ser destacados:

1. *Posição desejada é dada pela função seno ou degrau*: a eNFN mantém uma taxa de erro

[1] t é o passo atual

baixa e a estrutura do modelo se mantém estável com o número mínimo de regras.

2. *Posição desejada é dada pela função quadrada*: ocorre um aumento significativo na taxa de erro e logo após o início desse aumento a eNFN evolui sua estrutura para reduzir o erro.

Note que na Figura 3 a eNFN adapta/evolui sua estrutura em função dos dados de entrada e da taxa de erro, buscando manter uma estrutura compacta e uma taxa de erro baixa. Nas figuras 3c, 3d e 3e que mostram os valores da medida erro E_t, observa-se que a eNFN obteve uma melhor acurácia na estimação com 1 passo à frente, seguido pela estimação com 2 e 3 passos à frente. A Figura 4 mostra a posição medida e a estimada (1 passo à frente) pela eNFN durante o experimento, enquanto que a 4a ilustra a posição desejada e a estimada para a função quadrada, a 4b para a função degrau e a Figura 4c para a função seno.

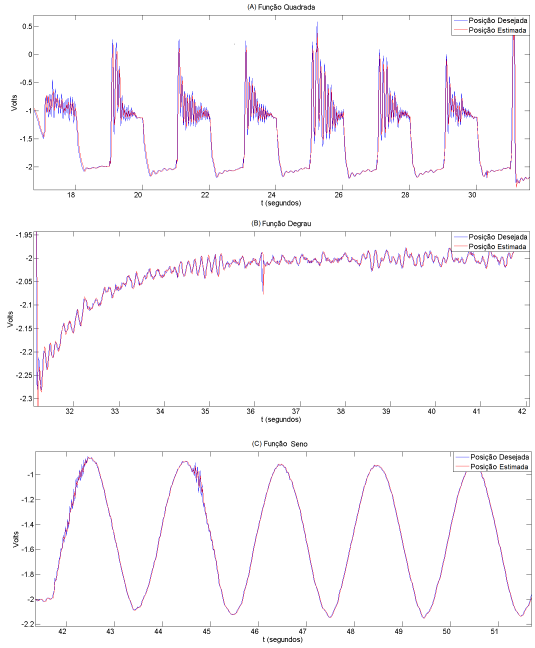

Figura 4: Posição medida e posição estimada (1 passo à frente) pela eNFN.

5 Conclusão

Este trabalho apresentou uma abordagem baseada em uma rede *neuro-fuzzy* evolutiva, assim como a sua implementação na forma de um estimador em tempo real da posição da esfera de um sistema de levitação magnética.

Os experimentos mostram que a eNFN consegue fazer a estimativa em tempo hábil, isto é, abaixo do período da taxa de amostragem, obtendo ao mesmo tempo uma boa acurácia, principalmente para a estimação com 1 passo à frente. Os experimentos também mostram que a estrutura da eNFN evolui de acordo com o fluxo dos dados de entrada para obter o modelo mais adequado e preciso para a condição operacional corrente, mantendo uma estrutura compacta e parcimoniosa.

Os resultados experimentais também sugerem que a eNFN é eficiente para estimar variáveis de processos com dinâmica rápida em tempo real. Evidentemente, a rede também pode ser utilizada para processos lentos.

Trabalhos futuros visam a utilização da eNFN em controle preditivo, em controle baseado na dinâmica inversa em tempo real do módulo de levitação magnética, além de processos não lineares com parâmetros variantes no tempo.

Agradecimentos

Os primeiros autores agradecem à CAPES, CNPq e FAPEMIG pelo apoio. O último autor é grato ao CNPq, processo 304596/2009-4.

Referências

Angelov, P. and Filev, D. (2005). Simpl_ets: A simplified method for learning evolving takagi-sugeno fuzzy models, *Proc. IEEE Int. Conf. Fuzzy Systems*, pp. 1068 – 1073.

Caminhas, W. and Gomide, F. (2000). A fast learning algorithm for neofuzzy networks, *Proc. Inf. Proc. and Manag. of Uncertainty in Knowledge Based Systems*, pp. 1784 – 1790.

Feedback, I. L. (2006). Magnetic levitation control experiments, 33-942s, *UK* .

Kasabov, N. and Filev, D. (2006). Evolving intelligent systems: Methods, learning, & applications, *Proc. Int. Symp. Evolving Fuzzy Systems*, pp. 8 – 18.

Lughofer, E. and Angelov, P. (2011). Handling drifts and shifts in on-line data streams with evolving fuzzy systems, *Applied Soft Computing* **11**(2): 2057–2068.

MathWorks, T. I. (2009). Real-time workshop 7 users guide, *Natick, MA, USA* .

Silva, A. M., Caminhas, W., Lemos, A. and Gomide, F. (2013). A fast learning algorithm for evolving neo-fuzzy neuron, *Applied Soft Computing* (0): Disponível on–line.

Yamakawa, T., Uchino, E., Miki, T. and Kusabagi, H. (1992). A neo fuzzy neuron and its applications to system identification and predictions to system behavior, *Proc. Int. Conf. on Fuzzy Logic and Neural Networks*, Vol. 1, pp. 477 – 484.