UNIVERSIDADE FEDERAL DO PIAUÍ

CENTRO DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

MESTRADO ACADÊMICO EM CONTROLE E AUTOMAÇÃO DE SISTEMAS

ANTONIO GALENO PEREIRA NETO

EMPREGO DE TÉCNICA DE MODELAGEM LOCAL MÚLTIPLA INVERSA PARA IDENTIFICAÇÃO DE SISTEMAS DINÂMICOS COM OUTLIERS

TERESINA

2023

ANTONIO GALENO PEREIRA NETO

EMPREGO DE TÉCNICA DE MODELAGEM LOCAL MÚLTIPLA INVERSA PARA IDENTIFICAÇÃO DE SISTEMAS DINÂMICOS COM OUTLIERS

Dissertação apresentada ao Curso de Mestrado Acadêmico em Controle e Automação de Sistemas do Programa de Pós-Graduação em Engenharia Elétrica do Centro de Tecnologia da Universidade Federal do Piauí, como requisito parcial à obtenção do título de mestre em Engenharia Elétrica. Área de Concentração: Engenharia de Controle e Automação de Sistemas

Orientador: Prof. Dr. Luis Gustavo Mota Souza

Co-Orientador: Prof. Dr. Francisco de Assis da Silva Mota

TERESINA

2023

FICHA CATALOGRÁFICA
Universidade Federal do Piauí
Biblioteca Comunitária Jornalista Carlos Castello Branco
Divisão de Representação da Informação

P436e Pereira Neto, Antonio Galeno.
Emprego de técnica de modelagem local múltipla inversa para identificação de sistemas dinâmicos com outliers / Antonio Galeno Pereira Neto. -- 2023.
84 f.

Dissertação (Mestrado) – Universidade Federal do Piauí, Programa de Pós-Graduação em Engenharia Elétrica, 2023.
"Orientador: Prof. Dr. Luís Gustavo Mota Souza".
"Co-orientador: Prof. Dr. Francisco de Assis da Silva Mota".

1. Modelos Locais. 2. Estimação M. 3. SOM. 4. Sistemas Dinâmicos. I. Souza, Luís Gustavo Mota. II. Mota, Francisco de Assis da Silva. III. Título.

CDD 621.3

Bibliotecária: Francisca das Chagas Dias Leite – CRB3/1004

ANTONIO GALENO PEREIRA NETO

EMPREGO DE TÉCNICA DE MODELAGEM LOCAL MÚLTIPLA INVERSA PARA IDENTIFICAÇÃO DE SISTEMAS DINÂMICOS COM OUTLIERS

Dissertação apresentada ao Curso de Mestrado Acadêmico em Controle e Automação de Sistemas do Programa de Pós-Graduação em Engenharia Elétrica do Centro de Tecnologia da Universidade Federal do Piauí, como requisito parcial à obtenção do título de mestre em Engenharia Elétrica. Área de Concentração: Engenharia de Controle e Automação de Sistemas

Aprovada em: 20/12/2022

BANCA EXAMINADORA

Prof. Dr. Luis Gustavo Mota Souza (Orientador)
Universidade Federal do Piauí (UFPI)

Prof. Dr. Francisco de Assis da Silva Mota (Co-Orientador)
Universidade Federal do Piauí (UFPI)

Prof. Dr. José Maria Pires de Menezes Júnior
Universidade Federal do Piauí (UFPI)

Prof. Dr. Otacílio da Mota Almeida

Prof. Dr. Amauri Holanda de Souza Júnior
Instituto Federal do Ceará (IFCE)

Ao Espiríto Santo de Deus, que me mantém de pé todos os dias e me sustenta com divino Amor.
À minha família e amigos, que sempre estiveram ao meu lado, me apoiando e acreditando em mim durante todos esses anos.

AGRADECIMENTOS

Ao Prof. Dr. Luís Gustavo Mota Souza por sua orientação durante esse trabalho.

Ao Prof. Dr. Francisco de Assis da Silva Mota, co-orientador e coordenador do Grupo de Estudos Avançados em Processos Industriais, onde iniciei este trabalho.

Aos meus pais Kléber Barros e Renata Maria por sempre me incetivarem a estudar e garantirem todas as condições necessárias.

À minha noiva Ana Tatiane por sempre permanecer ao meu lado e me incentivar nos momentos de desânimo e dificuldade.

Aos meus amigos e irmãos da Renovação Carismática Católica por suas orações e conselhos em todos os momentos.

Aos meus amigos e colegas de trabalho da Drogarias Globo por estarem ao meu lado nesse trajeto.

“Não é o evitar o sofrimento, a fuga diante da dor, que cura o homem, mas a capacidade de aceitar a tribulação e nela amadurecer, de encontrar o seu sentido através da união com Cristo, que sofreu com infinito amor.”

(Papa Emérito Bento XVI)

RESUMO

Com o crescimento de sistemas e processos não lineares, principalmente em ambientes industriais, os estudos por novas e mais precisas técnicas de Identificação de Sistemas Dinâmicos têm aumentado nas pesquisas. Dentre essas técnicas, as que permitem a geração de modelos locais apresentaram grande avanço, permitindo a geração de diversos modelos lineares locais para representar sistemas mais complexos. Também é preciso se preocupar com a inserção de ruídos nos dados, dependendo da fonte os processos podem acabar absorvendo informações que não são relevantes para a geração do modelo (outliers), o que pode interferir significativamente no processo de identificação. Neste trabalho, os modelos locais múltiplos D-MKSOM e P-MKSOM foram modificados para tratar dados discrepantes através da técnica de estimação M, passando assim a realizar uma estimação mais robusta, originando os algoritmos RD-MKSOM e RP-MKSOM, sendo então aplicados em bases de dados com fortes não linearidades. Os dados utilizados foram contaminados com ruído para avaliar quantitativa e qualitativamente a robustez dos algoritmos e o comportamento do erro obtido nos testes. Para fins de comparação, a rede neural global ELM também foi aplicada aos dados com a aplicação da técnica de estimação M. A partir dos resultados foi observado que os algoritmos propostos levam a uma melhor generalização a partir da análise de resíduos e robustez para inserção de outliers nos dados de teste.

Palavras-chave: Modelos Locais. Estimação M. SOM. Sistemas Dinâmicos.

ABSTRACT

With the growth of non-linear systems and processes, especially in industrial environments, studies for new and more accurate techniques for Dynamical Systems Identification have increased in research. Among these techniques, the ones that allow the generation of local models showed great progress, allowing the generation of several local linear models to represent more complex systems. It is also necessary to be concerned with the insertion of noise in the data, depending on the source, the processes may end up absorbing information that is not relevant to the generation of the model (outliers), which can significantly interfere in the identification process. In this work, the multiple local models D-MKSOM and P-MKSOM were modified in order to handle outlier data through the M estimation technique, thus starting to perform a more robust estimation, generating the RD-MKSOM and RP-MKSOM algorithms then they were applied to databases with strong non-linearities. The data used were contaminated with noise in order to quantitatively and qualitatively evaluate the robustness of the algorithms and the behavior of the error obtained in the tests. For comparison purposes, the ELM global neural network was also applied to the data with the application of M estimation. In the results was observed that the proposed algorithms leads to better generalization from the residual analysis and robustness to outliers insertion in test data.

Keywords: Local Models. M Estimation. SOM. Dynamical Systems.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SÍMBOLOS

t	Tempo discreto
$\{u(t)\}$	Sequência de símbolos de dados de entrada da planta industrial
$\{y(t)\}$	Sequência de símbolos de dados de saída da planta industrial
q	Número de memória da sequência de entrada, $u(t)$
p	Número de memória da sequência de saída, $y(t)$
$f(\cdot)$	Mapeamento não-linear desconhecido
$f^{-1}(\cdot)$	Mapeamento inverso não-linear desconhecido
$\mathbf{x}(t)$	Vetor de entrada no instante t
N	Número total de amostras do conjunto de dados utilizado pela rede neural
N_1	Número de amostras do conjunto de dados utilizado na etapa de treinamento da rede neural
N_2	Número de amostras do conjunto de dados utilizado na etapa de teste da rede neural
h_1	Número de neurônios existentes na camada escondida da rede ELM
w_{ij}	Peso do i-ésimo neurônio oculto ligado a j-ésima componente de entrada
θ_i	Limiar associado ao neurônio i
u_i	Ativação do i-ésimo neurônio da camada oculta
$\mathbf{W}$	Matriz de pesos sinápticos da camada escondida
$\mathbf{y}(t)$	Vetor de saída dos neurônios da camada escondida do modelo ELM no instante t
$\mathbf{Y}$	Matriz de saída dos neurônios da camada escondida
$\mathbf{m}$	Vetor de pesos sinápticos do neurônio na camada de saída
$J(\cdot)$	Função custo do erro quadrático instantâneo
Z	Potência do sinal na saída
V_i	Célula de Voronoi do i-ésimo neurônio
$\mathbf{w}_i(t)$	Vetor de pesos associado ao neurônio i no instante t

N_i	Número de vetores de dados pertencentes à célula de Voronoi do i-ésimo protótipo
$\alpha(t)$	Taxa de aprendizagem usada pelos algoritmos
N_{i^*}	Número de vetores de dados associados ao protótipo $\mathbf{w}_{i^*}$
$i^*(t)$	Índice do neurônio vencedor na rede no instante t
α_0, α_T	Valores inicial e final de $\alpha(t)$, respectivamente
$\mathscr{X}$	Espaço contínuo dos dados de entrada
$\mathscr{A}$	Espaço de saída discreto
Φ	Transformação não-linear
g	Número de neurônios existentes na rede SOM
$\mathbf{r}_i$	Coordenadas do neurônio i em um arranjo uni- ou bi-dimensional
$\mathbf{r}_{i^*}$	Coordenadas do neurônio vencedor i^* em um arranjo uni- ou bi-dimensional
$h(i^*, i; t)$	Função vizinhança da rede SOM no instante t
K_{i^*}	Conjunto vizinhança topológico que contém os neurônios vizinhos de i^*
$\sigma(t)$	Abertura da vizinhança topológica
σ_0, σ_f	Valores inicial e final de $\sigma(t)$, respectivamente
$\mathbf{c}_i$	Vetor de coeficientes do filtro transversal linear associado ao neurônio i
$\mathbf{x}^{in}(t)$	Vetor de entrada do mapeamento dinâmico usado pela rede VQTAM
$x^{out}(t)$	Escalar de saída do mapeamento dinâmico usado pela rede VQTAM
$\mathbf{w}_i^{in}(t)$	Vetor de pesos do neurônio i associado ao espaço de entrada do mapeamento dinâmico
$w_i^{out}(t)$	Escalar do neurônio i associado ao espaço de saída do mapeamento dinâmico
K	Número de neurônios vencedores
$\mathbf{R}$	Matriz de regressão formada pelos vetores de pesos do espaço de entrada na rede KSOM
$\mathbf{p}$	Vetor de predição formada pelos escalares de pesos do espaço de saída na rede KSOM
$j_k^{(i)}$	k-ésimo índice do vetor protótipo mais próximo ao vetor $\mathbf{w}_i^{in}$

$\mathscr{J}_i$	Conjunto contendo os índices dos K vetores protótipos mais próximos a $\mathbf{w}_i^{in}$, incluindo o neurônio i
$\mathbf{R}_i$	Matriz composta das partes das entradas dos K vetores protótipos cujos os índices pertencem ao conjunto $\mathscr{J}_i$ da rede P-MKSOM ou
$\mathbf{b}_i^{out}$	Vetor composto das partes de saída dos K vetores protótipos cujos os índices pertencem ao conjunto $\mathscr{J}_i$ da rede P-MKSOM
$\mathbf{X}_i^{in}$	Conjunto de vetores de dados de entrada pertencentes à célula de Voronoi do neurônio i na rede D-MKSOM
x_i^{out}	Conjunto de escalares de saída (x^{out}) associados com cada vetor $\mathbf{x}^{in}$ na rede D-MKSOM
$u(t)$	Saída desejada no instante t para modelagem inversa
$\hat{u}(t)$	Saída estimada no instante t
$e(t)$	Erro de aproximação no instante t
$\hat{f}(\cdot)$	Aproximação da função desconhecida $f(\dot{})$
σ_e^2	Variância do erro
σ_u^2	Variância do sinal de entrada
$w(e_n(t-1))$	Peso referido ao erro de estimação no instante $t-1$
$w_n(t-1)$	Vetor de pesos referidos aos erros de estimação acumulados do instante $t-1$
k_e	Função peso de estimação M
s	Constante robusta calculada a partir do desvio padrão dos resíduos
$\mathbf{B}(t-1)$	Matriz de ponderação robusta construída com os valores de diag$(w_n(t-1))$

SUMÁRIO

1 INTRODUÇÃO

Em geral, o processo de modelagem tem demandado técnicas cada vez mais avançadas com o objetivo de produzir modelos confiáveis, melhor conhecimento físico e a realização de processos de controle precisos e estáveis. No entanto, existe demanda para que esses modelos sejam obtidos rapidamente, especialmente em ambientes industriais onde o processo de produção deve permanecer na maior escala possível, fazendo com que a modelagem *caixa branca* (levantamento e equacionamento de todas as variáveis constituintes da física no processo) para o controle desses processos seja frequentemente impraticável (AGUIRRE, 2014).

A modelagem caixa branca vem sendo dificultada ainda com a implementação de processos não-lineares, com múltilplas entradas e múltiplas saídas (MIMO) e variantes no tempo, tornando exaustivo o processo de equacionamento desses modelos. Uma alternativa a esses desafios é a identificação de sistemas, também conhecida como modelagem *caixa preta* dado que apenas os dados de entrada e saída do sistema são conhecidos. Ou ainda, em alguns casos, pode ser extremamente viável a implementação de técnicas de modelagem *caixa cinza*, em que o cientista possui alguma informação física do processo e pode estruturar o modelo com maior confiabilidade (LJUNG, 1999a).

Pode-se dizer ainda que a obtenção e pré-processamento dos dados contituem a base fundante de todo o processo de identificação sistemas, visto que dados desconhecidos podem ser inseridos no sistema a partir de diversas fontes externas e interferir consideravelmente na acurácia dos modelos obtidos, são os chamados Outliers, desse modo essas etapas permitem a construção de modelos de regressão capazes de representar adequadamente e de modo confiável as características do sistema desejado a partir dos dados de entrada e saída obtidos. Portanto, a correta organização, o tratamento adequado e uma observação satisfatória dos dados de um sistema dinãmico são fatores extremamente importantes para a efetividade de sua identificação.

Em modelagem, para a realização do processo de identificação foram desenvolvidos e são comumente utilizados e estudados na literatura algoritmos que fornecem os chamados *modelos globais*, esses modelos representam o sistema como um todo a partir de um único modelo de regressão, sendo abundantemente aplicados em trabalhos científicos e processos de modelagem. Por outro lado, têm crescido na comunidade científica o interesse pela utilização de *modelos locais*, estes são capazes de caracterizar o sistema a partir da geração de diversos modelos de regressão para um mesmo processo, normalmente utilizando técnicas de clusterização para tais propósitos, podendo caracterizar o sistema a partir de informações menores, apresentando

potencial para a caracterização de sistemas com diversos pontos de operação ou muitas não-linearidades (SOUZA; BARRETO; CORONA, 2015). Obviamente a melhor maneira de escolher um modelo ou técnica de modelagem é com a realização de pesquisas e testes.

Os modelos locais mais frequentemente aplicados costumam utilizar técnicas de quantização vetorial, capazes de particionar o espaço de dados através da geração de clusters formados a partir da geração de centroides, vetores ou neurônios capazes de realizar uma correspondência entre os dados, fornecendo algoritmos com a capacidade de mapear até mesmo dados de entrada e saída, como no caso da rede neural SOM (KOHONEN, 2013), capaz de modelar os dados a partir do treinamento de neurônios para a definição de uma vizinhança topológica entre os dados, utilizando princípios como competitividade e cooperação na atualização de seus vetores protótipos.

O estudo e a aplicação de modelos locais já não podem ser considerados recentes na literatura, porém, o horizonte de técnicas a serem desenvolvidas segue em crescimento na pesquisa devido à quantidade de técnicas e abordagens aplicadas para a obtenção dos modelos locais. O desenvolvimento de novas técnicas visa sempre garantir a construção de modelos capazes de abranger de modo satisfatório todas as características do sistema.

Os algoritmos estudados foram analisados utilizando conjuntos de dados industriais apresentados nesse trabalho, e avaliados qualitativa e quantitativamente com a apresentação de curvas de regressão dos dados de teste e dos dados preditos gerados, aqui também são apresentados os comportamentos gaussianos para os dados preditos gerados pela necessidade de avaliar a capacidade de generalização dos modelos que melhor aproximem os dados estudados. Os algoritmos demonstraram aumento na robustez dos modelos em alguns casos e, mais importante, baixa correlação com os dados de treinamento.

Alguns trabalhos mais clássicos envolvendo o uso de modelos locais podem ser encontrados em Johansen e Foss (1993) e Göttsche, Hunt e Johansen (1998) que utilizaram modelos de regressão ARX a fim de encontrar representações locais de sistemas não-lineares. Com a união de modelos de regressão estabelecidos com algoritmos de quantização vetorial houve o surgimento de técnicas de modelagem local utilizando a rede neural SOM para associar modelos ARX aos dados (WALTER; RITTER; SCHULTEN, 1990). Em Barreto e Araújo (2004), por exemplo, houve a implementação de redes neurais competitivas para identificação de sistemas dinâmicos (aqui para séries temporais não estacionárias).

Com o trabalho de Souza (2012), veio o desenvolvimento de modelos múltiplos

locais, capazes de fornecer rápidas e precisas respostas para sistemas altamente não-lineares. Modelos híbridos também podem ser encontrados, como no trabalho de Souza, Barreto e Corona (2015) que realizou a separação do espaço de dados em regiões menores e similares, utilizando técnicas de partição em camadas (VESANTO; ALHONIEMI, 2000), permitindo a realização de uma identifição com maior grau de generalização a partir de modelos globais (MLP, ELM, RBF).

Nos últimos anos, modelos locais têm sido aperfeiçoados com o objetivo de melhor aproximar as curvas de regressão de dados não-lineares que se desejam modelar. Como em Matthiesen *et al.* (2018) onde a rede de modelagem local aplicada é modificada com um método de cálculo que permite rápida avaliação da rede e assim habilita a implementação em controle de máquinas para pequenos ciclos de tempo. Em Yin *et al.* (2021) os autores desenvolveram um controle de torque preditivo (PTC) sem a utilização de sensores para motores de indução utilizando o modelo ultra-local sem utilizar nenhum parâmetro prévio de motores. Também no trabalho de Souza e Santos (2022) o algoritmo KSOM utilizando o estimador OLS e o estimador M foi utilizado para comparar a performance da estimação aplicada a Curva de Potência de Turbinas Eólicas, trabalho que se avalia correlato ao método que está proposto.

1.1 Justificativa

O trabalho apresentado visa mostrar que modelos lineares locais podem ser aplicados de modo satisfatório a sistemas dinâmicos não-lineares, permitindo a modelagem de sistemas industriais de grande porte, que não podem sofrer paradas que demandem uma quantidade considerável de tempo e que possuem pontos de operação variáveis. Com a aplicação das técnicas de minimização do efeito de outliers busca-se alcançar um maior nível de confiabilidade na modelagem, especialmente em sistemas que sofrem com ambientes ruidosos. A estratégia é realizada através da comparação dos resultados com os obtidos pela aplicação de um modelo global robusto de comprovada eficácia aos mesmos bancos de dados. Além disso deseja-se construir modelos robustos inversos, que possibilitem uma ação de controle mais rápida a partir da estimação do valor de entrada desejada para se alcançar a saída do sistema em determinado ponto de operação.

1.2 Objetivos

A seguir são citados os objetivos do desenvolvimento deste trabalho.

1.2.1 Objetivo Geral

O presente trabalho tem como objetivo geral realizar a modelagem de sistemas dinâmicos não-lineares com a presença de outliers nos dados a partir da utilização de modelos locais múltiplos denominados robustos.

1.2.2 Objetivos Específicos

Dentro dos objetivos específicos deste trabalho estão a proposição de algoritmo de modelagem local múltipla utilizando técnica de estimação M para aumentar a robustez da predição realizada, bem como a realização do teste do algoritmo proposto em bancos de dados industriais de sistemas dinâmicos não-lineares. A avaliação quantitativa dos algoritmos a partir dos modelos obtidos também será realizada utilizando o erro de predição estimado sem e com a inserção de outliers.

Outros objetivos específicos do trabalho são o de apresentar os gráficos de predição dos modelos em comparação com os gráficos de dados de saída desejados para os menores valores de erro predição obtidos, evaliar a capacidade de generalização dos algoritmos tanto quantitativamente quanto qualitativamente e comparar os resultados dos modelos locais obtidos com técnica de modelagem global robusta através dos erros de predição gerados.

1.3 Estrutura do Trabalho

Este trabalho está organizado conforme apresentado a seguir. No Capítulo 2 são apresentadas duas abordagens prévias de modelagem local baseadas na rede SOM que servem como sistemas para a construção das abordagens propostas neste trabalho. No Capítulo 3 as duas técnicas propostas para a estimação de parâmetros de modelos múltiplos lineares locais são apresentadas. No Capítulo 4 a proposta de modelos múltiplos lineares locais robustos é então descrita e apresentada, utilizando a técnica de estimação M, com a ponderação interativa dos vetores protótipos para gerar modelos locais menos suscetíveis à presença de outliers. Finalizando a explanação da base teórica desse trabalho no Capítulo 5 é apresentado o modelo global ELM e sua utilização em identificação de sistemas dinâmicos, bem como uma versão robusta do algoritmo utilizando a técnica de estimação M. Nos Capítulos 6 e 7 são apresentados os recursos metodológicos utilizados e os resultados obtidos neste trabalho, respectivamente. A dissertação é finalizada com o Capítulo de 8, fornecendo ainda indicações de trabalhos futuros e

com o apêndice A onde se apresenta os resultados obtidos dos algoritmos propostos para a tarefa de classificação de padrões.

2 ABORDAGENS DE MODELAGEM LOCAL BASEADAS NA REDE SOM

Assumindo que os sistemas dinâmicos que estão sendo tratados podem ser matematicamente descritos através do modelo discreto NARX [1] (LJUNG, 1999b; NORGAARD *et al.*, 2000):

$$y(t) = f\left[y(t-1),\ldots,y(t-p);u(t),u(t-1),\ldots,u(t-q+1)\right]+\xi(t), \tag{2.1}$$

onde $f(\cdot)$ representa uma mapeamento não-linear desconhecido, $u(t)\in\mathbb{R}$ e $y(t)\in\mathbb{R}$ denotam, respectivamente, a entrada e saída do modelo no instante de tempo t, enquanto $p\geq 1$ e $q\geq 1$ ($p\geq q$) são as memórias das ordens de entrada e saída, respectivamente. A variável $\xi(t)$ representa a variável de erro, assumido aqui como ruído gaussiano branco aditivo.

Neste trabalho, o objetivo é avaliar a proposta de modelagem linear local na identificação inversa de sistemas, especialmente para uso posterior em estratégias de controle. Assim, para a tarefa de identificação inversa de sistemas é necessário que os modelos neurais possam aproximar a seguinte função de mapeamento inverso:

$$u(t) = f^{-1}\left[u(t-1),\ldots,u(t-q);y(t-1),\ldots,y(t-p)\right]+\xi(t). \tag{2.2}$$

Esse tipo de modelo inverso não linear do sistema é necessário, por exemplo, na implementação de muitas estratégias de controle, como o Controle Inverso-Direto (DAOSUD *et al.*, 2005), Controle de Modelo Interno (MU; SUN; YU, 2011; NORGAARD *et al.*, 2000; HUSSAIN; KERSHENBAUM, 2000; LIGHTBODY; IRWIN, 1997a), Aprendizado Inverso Generalizado (HUNT *et al.*, 1992; HUSSAIN, 1996) e outros (CHO *et al.*, 2007; CHO *et al.*, 2006; ANDRÁSIK; MÉSZÁROS; AZEVEDO, 2004). A arquitetura de Controle de Modelo Interno utiliza, por exemplo, ambos modelos direto e inverso para a ação de controle de um dado sistema.

Modelos inversos devem fornecer uma representação confiável do mapeamento inverso a partir das regiões operacionais de interesse. Modelagem inversa é uma tarefa de processamento de sinais normalmente considerada muito mais difícil de lidar do que a modelagem direta, desde que múltiplas soluções podem existir. Então, modelagem inversa pertence à classe de problemas mal postados (ill-posed) sendo estes de fase não-mínima, no trabalho proposto a modelagem inversa será tratada a partir da utilização de modelos lineares locais. Um exemplo é a utilização de controle por modelo interno (*Internal Model Control* - IMC) (HUSSAIN; KERSHENBAUM, 2000), conforme ilustrado na Figura 1.

[1] NARX stands for Nonlinear Autoregressive model with exogenous inputs.

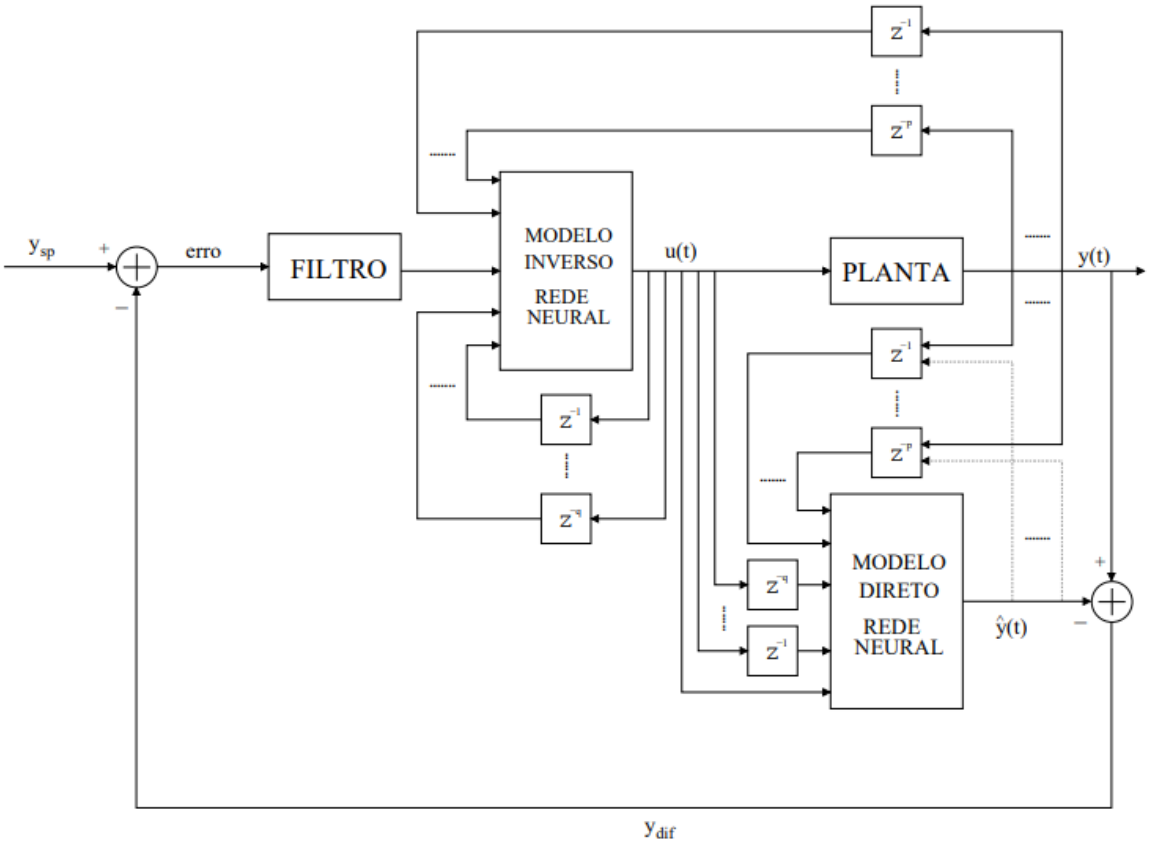


Figura 1 – Estrutura da rede neural na configuração de controle por modelo interno.

Os parâmetros e mapeamento do modelo NARX podem ser obtidos utilizando diversas abordagens de inteligência computacional que vem sendo estudadas e melhoradas com o tempo devido à necessidade de melhor entendimento e controle de uma grande demanda de sistemas dinâmicos não-lineares. Entre os métodos utilizados, podem ser citadas as redes neurais (BILLINGS; CHEN, 1992; LAWRENCE; TSOI; BACK, 1996; LIGHTBODY; IRWIN, 1997b; BARRETO; ARAÚJO, 2004), modelos fuzzy Takagi-Sugeno-Kang (TAKAGI; SUGENO, 1985; TSAI; CHEN, 2022; HONGWEI; PENGLONG, 2020) e modelos híbridos (FEI *et al.*, 2022; ZHAO; LIN, 2019; BABUŜKA; VERBRUGGEN, 2003; NAVABI; HOSSEINI, 2020).

Neste trabalho, as abordagens de modelagem local baseadas em quantização vetorial propostas são avaliadas na produção de mapeamentos inversos confiáveis de dois sistemas de dados entrada-saída. Para esse propósito, é adotada uma estratégia de aprendizagem inversa generalizada.

2.1 A Abordagem VQTAM

A abordagem de Memória Associativa Temporal por Quantização Vetorial (*Vector Quantized Temporal Associative Memory* - VQTAM) (BARRETO; ARAÚJO, 2004) tem o objetivo de construir um mapeamento entrada-saída utilizando a técnica neural de *Mapa Auto-Organizável* (*Self Organizing Map* - SOM). A VQTAM é então uma generalização para o domínio temporal de uma técnica de memória associativa com a rede SOM que vem sendo aplicada por diversos autores para realizar a aprendizagem de mapeamentos entrada-saída estáticos (sem memória), especialmento no campo de robótica (BARRETO; ARAÚJO; RITTER, 2003).

A SOM aprende a partir de exemplos um mapeamento (projeção) de espaço de entrada contínuo de alta dimensão $\mathscr{X}$ em um espaço discreto de baixa dimensão (topologia) $\mathscr{A}$ de N neurônios que estão arranjados em formas topológicas fixas, e.g., como uma matriz retangular de duas dimensões. O mapa $i^*(\mathbf{x}) : \mathscr{X} \to \mathscr{A}$ é definido pelos vetores de pesos $\mathscr{W} = \{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_N\}, \mathbf{w}_i \in \mathbb{R}^p \subset \mathscr{X}$, e suas coordenadas correspondentes $\mathbf{r}_i \in \mathbb{R}^2$ na topologia $\mathscr{A}$.

De acordo com a técnica VQTAM, o vetor de entrada para a rede SOM no instante de tempo t, $\mathbf{x}(t)$, é composto de duas partes. A primeira parte, denotada $\mathbf{x}^{in}(t) \in \mathbb{R}^{p+q}$, é composta dos dados de entrada do mapeamento do sistema dinâmico a ser aprendido. A segunda parte, denotada $x^{out}(t) \in \mathbb{R}$, contém dados pertencentes à saída desejada do mapeamento. O vetor de pesos do neurônio i, $\mathbf{w}_i(t)$, apresenta sua dimensão incrementada de acordo. Essas mudanças podem ser formuladas como segue:

$$\mathbf{x}(t) = \begin{pmatrix} \mathbf{x}^{in}(t) \\ x^{out}(t) \end{pmatrix} \quad \text{and} \quad \mathbf{w}_i(t) = \begin{pmatrix} \mathbf{w}_i^{in}(t) \\ w_i^{out}(t) \end{pmatrix} \tag{2.3}$$

onde $\mathbf{w}_i^{in}(t) \in \mathbb{R}^{p+q}$ e $w_i^{out}(t) \in \mathbb{R}$ são, respectivamente, as porções dos vetores (protótipos) de pesos que armazenam informação sobre as entradas e saídas do mepaeamento desejado. Dependendo das variáveis escolhidas para compor o vetor $\mathbf{x}^{in}(t)$ e escalar $x^{out}(t)$ o algoritmo SOM pode ser utilizado (ou qualquer algoritmo de quantização vetorial) para realizar o aprendizado do mapeamento direto ou inverso de um dado sistema dinâmico.

Para a tarefa de identificação inversa que se tem interesse, são dadas as seguintes definições:

$$\mathbf{x}^{in}(t) = [u(t-1), \ldots, u(t-q); y(t-1), \ldots, y(t-p)]^T, \tag{2.4}$$

$$x^{out}(t) = u(t). \tag{2.5}$$

O neurônio vencedor no instante t é determinado baseando-se unicamente em $\mathbf{x}^{in}(t)$:

$$i^*(t) = \arg\min_{\forall i \in \mathscr{A}} \{\|\mathbf{x}^{in}(t) - \mathbf{w}_i^{in}(t)\|\}. \tag{2.6}$$

Para a atualização dos pesos, ambos $\mathbf{x}^{in}(t)$ e $x^{out}(t)$ são utilizados:

$$\mathbf{w}_i^{in}(t+1) = \mathbf{w}_i^{in}(t) + \alpha(t)h(i^*,i;t)[\mathbf{x}^{in}(t) - \mathbf{w}_i^{in}(t)], \tag{2.7}$$

$$w_i^{out}(t+1) = w_i^{out}(t) + \alpha(t)h(i^*,i;t)[x^{out}(t) - w_i^{out}(t)], \tag{2.8}$$

onde $0 < \alpha(t) < 1$ é a taxa de aprendizagem, e $h(i^*,i;t)$ é uma função de vizinhança Gaussina variante no tempo definida como

$$h(i^*,i;t) = \exp\left(-\frac{\|\mathbf{r}_i(t) - \mathbf{r}_{i^*}(t)\|^2}{2\sigma^2(t)}\right) \tag{2.9}$$

onde $\mathbf{r}_i(t)$ e $\mathbf{r}_{i^*}(t)$ são respectivamente, as coordenadas dos neurônios i e i^* na matriz de saída, e $\sigma(t) > 0$ define o raio da função de vizinhança no tempo t. As variáveis $\alpha(t)$ e $\sigma(t)$ devem ambar decair com o tempo a fim de garantir convergência dos vetores de peso para estados estacionários estáveis. Neste trabalho, são adotadas quedas exponenciais para ambas as variáveis:

$$\alpha(t) = \alpha_0\left(\frac{\alpha_T}{\alpha_0}\right)^{(t/T)} \quad \text{and} \quad \sigma(t) = \sigma_0\left(\frac{\sigma_T}{\sigma_0}\right)^{(t/T)} \tag{2.10}$$

onde α_0 (σ_0) e α_T (σ_T) são os valores inicial e final de $\alpha(t)$ e ($\sigma(t)$), respectivamente.

Conforme o treinamento procede, a rede SOM aprende a associar os vetores de pesos da entrada $\mathbf{w}_i^{in}$ com os pesos de saída correspondentes w_i^{out}. O ajuste de pesos é realizado até que um estado estacionário de organização global dos vetores de pesos seja alcançada. Nesse caso, se diz que o mapa convergiu.

Uma vez que a rede SOM tenha sido treinada, a entrada estimada $\hat{u}(t)$ é computada simplesmente como

$$\hat{u}(t) = w_{i^*}^{out}(t). \tag{2.11}$$

Na Figura 2 é possível visualizar a arquitetura e o mapeamento realizado pelo modelo VQTAM a partir do par entrada-saída apresentado. Vale a pena notar que nas redes MLP e RBF, o vetor $\mathbf{x}^{in}(t)$ é apresentado para a rede de entrada, enquanto $x^{out}(t)$ é usada como saída desejada permitindo ao programa obter um sinal de erro que conduz o aprendizado. O método VQTAM, por outro lado, permite aos algoritmos de quantização vetorial, como a SOM, incluir a

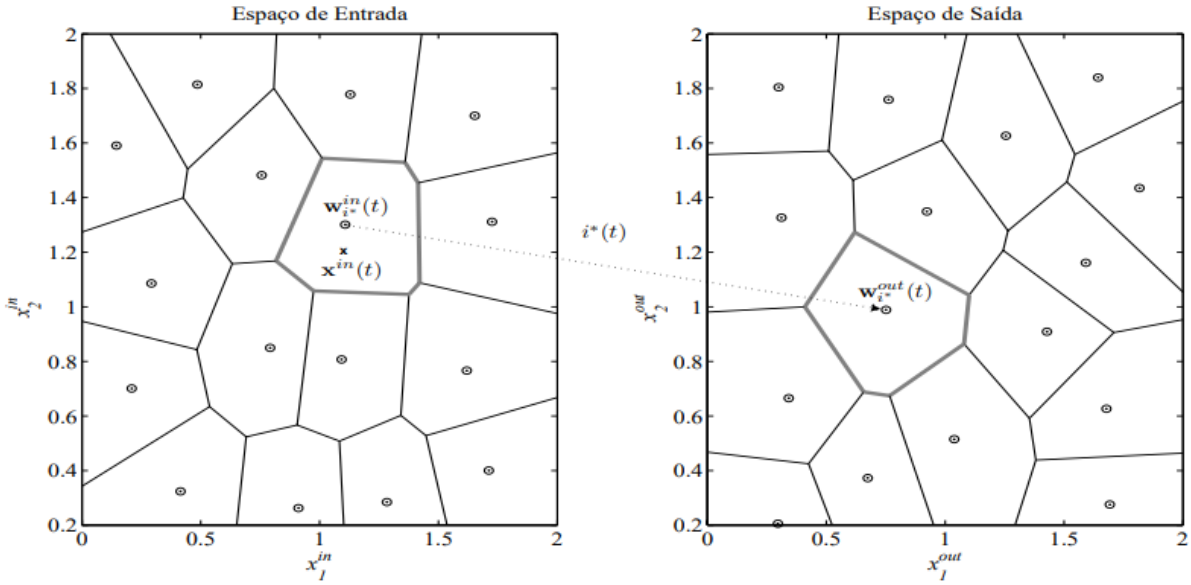


Figura 2 – Representação da arquitetura do modelo VQTAM

saída desejada $x^{out}(t)$ como parte da rede do vetor de entrada $\mathbf{x}(t)$. Isto permite a interpretação do termo $e_i(t) = x^{out}(t) - w_i^{out}(t)$ na Eq. (2.8) como o valor de erro causado pelo neurônio i-th. Assim, usando Eq. (2.11), a estimação do erro devida ao neurônio vencedor é dada por

$$e_{i^*}(t) = x^{out}(t) - w_{i^*}^{out}(t) = u(t) - \hat{u}(t). \tag{2.12}$$

2.2 O Modelo KSOM

Em Souza e Barreto (2010) o modelo KSOM é introduzido e avaliado para a tarefa de identificação inversa de sistemas. O modelo KSOM utiliza um modelo linear local único cujo vetor de coeficientes é variante no tempo, i.e. o mesmo é a partir dos vetores protótipos dos K vizinhos espaciais mais próximos ao vetor protótipo do neurônio vencedor ao vetor de entrada na iteração t. Detalhes são fornecidos a seguir.

A ideia por trás do KSOM é realizar primeiramente o treinamento da VQTAM conforme descrito na Seção 2.1 usando apenas uma pequena quantidade de neurônios (normalmente menos que 100 unidades) com o objetivo de fornecer uma representação reduzida do mapeamento entrada-saída armazenado nos vetores de peso da VQTAM. Então, para um novo vetor de entrada apresentado no instante de tempo t, os coeficientes do modelo linear local variante no tempo são estimados utilizando os vetores de peso dos K primeiros neurônios vencedores $\{i_1^*, i_2^*, \ldots, i_K^*\}$.

Tabela 1 – Algoritmo de Aprendizagem do Modelo VQTAM

MODELO VQTAM	
Constantes	**Valores Típicos**
α : Taxa de aprendizagem	$0.001 < \alpha < 0.5$
σ : Abertura da vizinhança topológica	$0.001 < \sigma < (g/2)$
Entradas	
$\mathbf{x}^{in}(t)$: vetor de entrada, dimensão$(p+q)\times 1$	$x^{out}(t)$: variável de saída
Algoritmo	
1. Inicialização $(t=0)$	
$\mathbf{w}_i^{in}, w_i^{out} \sim U(0,1), i=1,...,g$	
2. Aprendizagem do modelo $(t=1,2,...,N_1)$	
2.1 Escolha do neurônio vencedor:	
$i^*(t) = \text{argmin}_i\{\|\mathbf{x}^{in}(t) - \mathbf{w}_i^{in}(t)\|\}$.	
2.2 Atualização dos vetores de pesos e de coeficientes:	
$h(i^*,i;t) = \exp\left(-\frac{\|\mathbf{r}_i(t)-\mathbf{r}_{i^*}(t)\|^2}{2\sigma^2(t)}\right)$,	
$\mathbf{w}_i^{in}(t+1) = \mathbf{w}_i^{in}(t) + \alpha(t)h(i^*,i;t)[\mathbf{x}^{in}(t) - \mathbf{w}_i^{in}(t)]$,	
$w_i^{out}(t+1) = w_i^{out}(t) + \alpha(t)h(i^*,i;t)[x^{out}(t) - w_i^{out}(t)]$.	
3. Saída do modelo $(t=1,2,...,N_2)$	
3.1 Escolha do neurônio vencedor:	
$i^*(t) = \text{argmin}_i\{\|\mathbf{x}^{in}(t) - \mathbf{w}_i^{in}(t)\|\}$.	
3.2 Cálculo da saída estimada:	
$\hat{u}(t) = w_{i^*}^{out}(t)$.	
3.3 Cálculo do erro:	
$e(t) = u(t) - \hat{u}(t)$.	
Saídas	
$\hat{u}(t)$: saída estimada	$e(t)$: erro

Esses neurônios são selecionados conforme segue:

$$\begin{aligned} i_1^*(t) &= \arg\min_{\forall i}\left\{\|\mathbf{x}^{in}(t) - \mathbf{w}_i^{in}(t)\|\right\}, \\ i_2^*(t) &= \arg\min_{\forall i\neq i_1^*}\left\{\|\mathbf{x}^{in}(t) - \mathbf{w}_i^{in}(t)\|\right\}, \\ &\vdots \qquad\qquad \vdots \qquad\qquad \vdots \\ i_K^*(t) &= \arg\min_{\forall i\neq\{i_1^*,\ldots,i_{K-1}^*\}}\left\{\|\mathbf{x}^{in}(t) - \mathbf{w}_i^{in}(t)\|\right\}. \end{aligned} \tag{2.13}$$

Desde que o conjunto de K vetores de peso vencedores no tempo t seja denotado por $\{\mathbf{w}_{i_1^*}, \mathbf{w}_{i_2^*}, \ldots, \mathbf{w}_{i_K^*}\}$. Associando isso ao estilo de treinamento da VQTAM, cada vetor de pesos $\mathbf{w}_i(t)$ apresenta uma porção associada com $\mathbf{x}^{in}(t)$ e outra associada com $x^{out}(t)$. Então, o KSOM utiliza os K correspondentes pares de vetores protótipos $\{\mathbf{w}_{i_k^*}^{in}(t), w_{i_k^*}^{out}(t)\}_{k=1}^K$, com o objetivo de construir um mapeamento linear local no instante t:

$$w_{i_k^*}^{out} = \mathbf{c}^T(t)\mathbf{w}_{i_k^*}^{in}(t), \qquad k=1,\ldots,K \tag{2.14}$$

onde $\mathbf{c}(t) = [b_1(t), \ldots, b_q(t), a_1(t), \ldots, a_p(t)]^T$ é um vetor de coeficientes variante no tempo. A Equação (2.14) pode ser escrita na forma matricial como

$$\mathbf{w}^{out}(t) = \mathbf{R}(t)\mathbf{c}(t), \tag{2.15}$$

onde o vetor de saída $\mathbf{w}^{out}(t) \in \mathbb{R}^K$ e a matriz de regressão $\mathbf{R}(t) \in \mathbb{R}^{K\times(p+q)}$ no instante t são definidas a seguir

$$\mathbf{w}^{out}(t) = \left[w^{out}_{i^*_1}(t) \;\; w^{out}_{i^*_2}(t) \;\; \cdots \;\; w^{out}_{i^*_K}(t) \right]^T \tag{2.16}$$

e

$$\mathbf{R}(t) = \begin{pmatrix} w^{in}_{i^*_1,1}(t) & w^{in}_{i^*_1,2}(t) & \cdots & w^{in}_{i^*_1,p+q}(t) \\ w^{in}_{i^*_2,1}(t) & w^{in}_{i^*_2,2}(t) & \cdots & w^{in}_{i^*_2,p+q}(t) \\ \vdots & \vdots & \vdots & \vdots \\ w^{in}_{i^*_K,1}(t) & w^{in}_{i^*_K,2}(t) & \cdots & w^{in}_{i^*_K,p+q}(t) \end{pmatrix}. \tag{2.17}$$

Na prática, desde que se tem normalmente $p+q > K$, a matriz $\mathbf{R}$ não é quadrada. Nesse caso, estima-se o vetor de coeficientes $\mathbf{c}(t)$ através do método de Mínimos Quadrados Ordinário (*Ordinary Least Squares* - OLS):

$$\mathbf{c}(t) = \left(\mathbf{R}^T(t)\mathbf{R}(t) + \lambda\mathbf{I}\right)^{-1}\mathbf{R}^T(t)\mathbf{w}^{out}(t), \tag{2.18}$$

onde $\mathbf{I}$ é uma matriz identidade de ordem K e $\lambda > 0$ (e.g. $\lambda = 0.001$) é uma pequena constante acrescentada na diagonal de $\mathbf{R}^T(t)\mathbf{R}(t)$ a fim de garantir que a matriz apresente posto cheio. Uma vez que $\mathbf{c}(t)$ é estimada, pode-se aproximar localmente a saída do mapeamento não linear por meio do seguinte mapeamento linear:

$$\begin{aligned} \hat{u}(t) &= \sum_{k=1}^{q} b_k(t)u(t-k) + \sum_{l=1}^{p} a_l(t)y(t-l), \\ &= \mathbf{c}^T(t)\mathbf{x}^{in}(t). \end{aligned} \tag{2.19}$$

Uma abordagem muito parecida com a do algoritmo KSOM foi introduzida por Principe, Wang e Motter (1998) e utilizada para modelagem local inversa em Cho *et al.* (2007), Cho *et al.* (2006). Nessa arquitetura, os vetores protótipos necessários não são selecionados como os K protótipos mais próximos do vetor de entrada apresentado, mas selecionados praticamente de forma automática como o protótipo vencedor no instante t e seus $K-1$ vizinhos topológicos. Se a preservação de topologia perfeita é alcançada durante o treinamento da rede SOM, os neurônios na vizinhança topológica do protótipo vencedor também são aqueles próximos do vetor de entrada

apresentado. No entanto, se defeitos topológicos estão presentes, como geralmente ocorre para dados multidimensionais, essa propriedade não pode ser garantida. Assim, a utilização dessa arquitetura é limitada a algoritmos de quantização vetorial para preservação de topologia. O modelo KSOM, no entanto, é geralmente suficiente para ser usado com diferentes tipos de algoritmos de quatização vetorial.

No próximo capítulo serão apresentadas as técnicas de modelos lineares locais múltiplos, construídas a partir das técnicas aqui abordadas, como será explanado posteriormente.

Tabela 2 – Algoritmo de Aprendizagem do Modelo KSOM

MODELO KSOM

Constantes	**Valores Típicos**
α : Taxa de aprendizagem	$0.001 < \alpha < 0.5$
σ : Abertura da vizinhança topológica	$0.001 < \sigma < (g/2)$
λ : Constante de regularização	$\lambda = 0.001$

Entradas

$\mathbf{x}^{in}(t)$: vetor de entrada, dimensão$(p+q)\times 1$ — $x^{out}(t)$: variável de saída

Algoritmo

1. Inicialização $(t=0)$

$\mathbf{w}_i^{in}, w_i^{out} \sim U(0,1), i = 1,...,g$

2. Aprendizagem do modelo $(t = 1,2,...,N_1)$

2.1 Escolha do neurônio vencedor:

$i^*(t) = \text{argmin}_i\{\|\mathbf{x}^{in}(t) - \mathbf{w}_i^{in}(t)\|\}$.

2.2 Atualização dos vetores de pesos e de coeficientes:

$h(i^*, i; t) = \exp\left(-\frac{\|\mathbf{r}_i(t) - \mathbf{r}_{i^*}(t)\|^2}{2\sigma^2(t)}\right)$,

$\mathbf{w}_i^{in}(t+1) = \mathbf{w}_i^{in}(t) + \alpha(t)h(i^*, i; t)[\mathbf{x}^{in}(t) - \mathbf{w}_i^{in}(t)]$,

$w_i^{out}(t+1) = w_i^{out}(t) + \alpha(t)h(i^*, i; t)[x^{out}(t) - w_i^{out}(t)]$.

3. Saída do modelo $(t = 1,2,...,N_2)$

3.1 Escolha dos K neurônios mais próximos (vencedores) ao vetor de entrada $\mathbf{x}^{in}(t)$:

$\{i_1^*, i_2^*, \ldots, i_K^*\}$.

3.2 Obtenção do vetor $\mathbf{w}^{out}(t)$ e da matriz de regressão $\mathbf{R}(t)$:

$$\mathbf{w}^{out}(t) = \begin{bmatrix} w_{i_1^*}^{out}(t) & w_{i_2^*}^{out}(t) & \cdots & w_{i_K^*}^{out}(t) \end{bmatrix}^T,$$

$$\mathbf{R}(t) = \begin{pmatrix} w_{i_1^*,1}^{in}(t) & w_{i_1^*,2}^{in}(t) & \cdots & w_{i_1^*,p+q}^{in}(t) \\ w_{i_2^*,1}^{in}(t) & w_{i_2^*,2}^{in}(t) & \cdots & w_{i_2^*,p+q}^{in}(t) \\ \vdots & \vdots & \vdots & \vdots \\ w_{i_K^*,1}^{in}(t) & w_{i_K^*,2}^{in}(t) & \cdots & w_{i_K^*,p+q}^{in}(t) \end{pmatrix}.$$

3.3 Obtenção do vetor de coeficientes no instante t:

$\mathbf{c}(t) = \left(\mathbf{R}^T(t)\mathbf{R}(t) + \lambda\mathbf{I}\right)^{-1}\mathbf{R}^T(t)\mathbf{w}^{out}(t)$.

3.4 Cálculo da saída estimada:

$\hat{u}(t) = \mathbf{c}^T(t)\mathbf{x}^{in}(t)$.

3.5 Cálculo do erro:

$e(t) = u(t) - \hat{u}(t)$.

Saídas

$\hat{u}(t)$: saída estimada — $e(t)$: erro

3 MODELAGEM LOCAL MÚLTIPLA

Como mencionado anteriormente, os dois modelos podem ser entendidos como modelos múltiplos do modelo KSOM, sendo construídos a partir do modelo VQTAM, os modelos múltiplos são definidos após a geração das células de Voronoi com as regiões dos neurônios vitoriosos, após esta etapa são construídos vetores de coeficientes e obtidos os resultados estimados através dos vetores protótipos treinados ou dos dados associados aos vetores. Em outras palavras, enquanto o modelo KSOM requer apenas um único modelo linear local, cujo vetor de coeficientes é estimado a cada iteração, as abordagens propostas constroem modelos múltiplos lineares locais (uma para cada neurônio na rede SOM).

Outra maneira de entender as diferenças entre as abordagens descritas e o modelo KSOM é que enquanto a abordagem KSOM o vetor de coeficientes único é variante no tempo, nas abordagens descritas o vetor de coeficientes de cada modelo local é fixado após uma fase de aprendizagem. Os modelos propostos diferem basicamente na maneira que os coeficientes são estimados. Um deles utiliza os vetores protótipos (pesos) dos neurônios SOM, enquanto o outro utiliza os vetores de dados mapeados a esses protótipos.

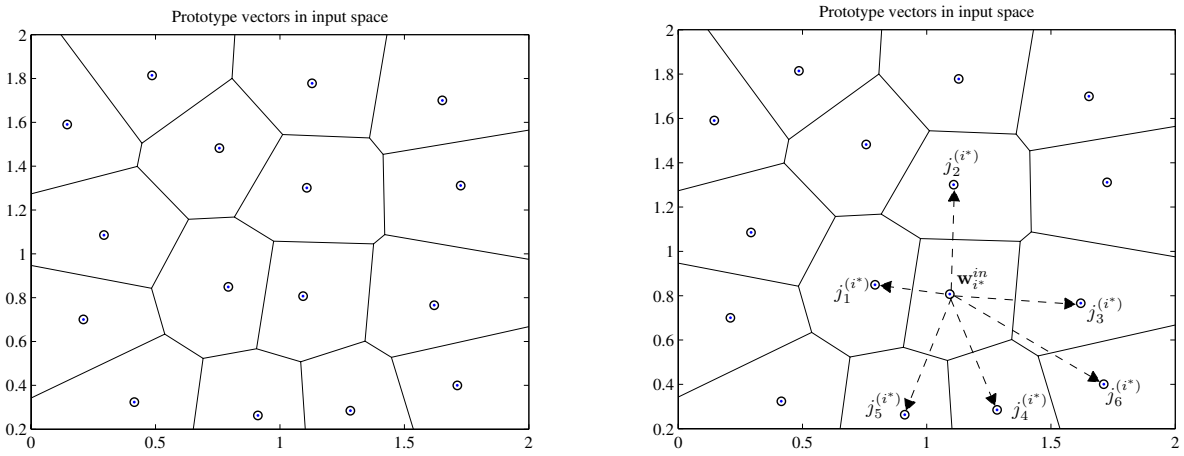
3.1 Modelo Linear Local Baseado nos *K* Protótipos Mais Próximos

O primeiro algoritmo a ser descrito é chamado Modelo KSOM Múltiplo Baseado em Protótipos (*Prototype-based Multiple KSOM Model* - P-MKSOM). O primeiro passo na construção do modelo P-MKSOM requer a abordagem VQTAM (veja a Seção 2.1). A construção do modelo P-MKSOM inicia assim que o treinamento da VQTAM finaliza.

Primeiramente, deixe $j_k^{(i)}$ denota o k-ésimo vizinho mais próximo do neurônio i. Então, encontre os K vizinhos mais próximos do vetor protótipo $\mathbf{w}_i^{in}$ como segue:

$$\begin{aligned} i_1 &= \arg\min_{\forall j \neq i} \left\{ \|\mathbf{w}_i^{in} - \mathbf{w}_j^{in}\| \right\}, \\ i_2 &= \arg\min_{\forall j \neq \{i, i_1\}} \left\{ \|\mathbf{w}_i^{in} - \mathbf{w}_j^{in}\| \right\}, \\ &\vdots \qquad\qquad \vdots \qquad\qquad \vdots \\ i_K &= \arg\min_{\forall j \neq \{i, i_1, \ldots, i_{K-1}\}} \left\{ \|\mathbf{w}_i^{in} - \mathbf{w}_j^{in}\| \right\}, \end{aligned} \tag{3.1}$$

onde $\mathscr{J}_i = i \cup \{i_k\}_{k=1}^{K}$ é o conjunto contendo os índices dos K vizinhos mais próximos do vetor protótipo $\mathbf{w}_i^{in}$, incluindo o neurônio i. A Figura 3 mostra o processo.



(a) (b)

Figura 3 – Construção do modelo P-MKSOM: (a) Células de Voronoi associadas com cada vetor protótipo após o treinamento do modelo VQTAM; (b) o conjunto de $K = 6$ vizinhos mais próximos de um dado vetor protótipo $\mathbf{w}_i^{in}$, cujos índices estão inclusos no conjunto $(\mathscr{J}_{i^*})$.

Uma vez que o conjunto $\mathscr{J}_i$ é determinado para cada neurônio i, constrõe-se N modelos locais de regressão utilizando os vetores protótipos cujos índices pertencem a $\mathscr{J}_i$. Dessa forma, associado ao neurônio i, existe o vetor de coeficientes $\mathbf{c}_i \in \mathbb{R}^{p+q}$ computado utilizando o método dos mínimos quadrados:

$$\mathbf{c}_i = \left(\mathbf{R}_i^T \mathbf{R}_i + \lambda \mathbf{I}\right)^{-1} \mathbf{R}_i^T \mathbf{b}_i^{out}, \tag{3.2}$$

onde $\mathbf{I}$ é a matriz identidade de ordem $(p+q)\times(p+q)$ e $\lambda > 0$ (e.g. $\lambda = 0.001$) é uma pequena constante de regularização. O vetor $\mathbf{b}_i^{out} \in \mathbb{R}^{K+1}$ é composto das partes de saída dos K vetores protótipos cujos índices pertencem a $\mathscr{J}_i$, i.e.

$$\mathbf{b}_i^{out} = \begin{bmatrix} w_i^{out} & w_{i_1}^{out} & \cdots & w_{i_K}^{out} \end{bmatrix}^T, \tag{3.3}$$

e a matriz $\mathbf{R}_i \in \mathbb{R}^{(K+1)\times(p+q)}$ é composta das partes de entrada dos mesmos K vetores protótipos:

$$\mathbf{R}_i = \begin{pmatrix} \left(\mathbf{w}_i^{in}\right)^T \\ \left(\mathbf{w}_{i_1}^{in}\right)^T \\ \vdots \\ \left(\mathbf{w}_{i_K}^{in}\right)^T \end{pmatrix} = \begin{pmatrix} w_{i,1}^{in} & w_{i,2}^{in} & \cdots & w_{i,p+q}^{in} \\ w_{i_1,1}^{in} & w_{i_1,2}^{in} & \cdots & w_{i_1,p+q}^{in} \\ \vdots & \vdots & \vdots & \vdots \\ w_{i_K,1}^{in} & w_{i_K,2}^{in} & \cdots & w_{i_K,p+q}^{in} \end{pmatrix}, \tag{3.4}$$

onde o sobrescrito T denota o(a) vetor/matriz transposto(a).

Uma vez que N modelos de regressão locais tenham sido criados, eles podem ser usados para aproximar a saída do mapeamento não linear de interesse. Relembrando que o

modelo P-MKSOM requer um único modelo local (e assim, um vetor de coeficientes) por neurônio. Qualquer que seja utilizado no instante t é definido pelo índice do neurônio vencedor, determinado como mostrado na Eq. (2.6).

Desde que o interesse é de identificação inversa de sistemas, o modelo P-MKSOM estima a entrada corrente $u(t)$ por meio da seguinte equação:

$$\hat{u}(t) = \mathbf{c}_{i^*}^T \mathbf{x}^{in}(t), \tag{3.5}$$

onde o erro de estimação (residual) no instante t é definido como $e(t) = u(t) - \hat{u}(t)$.

3.2 Modelo Linear Local Baseado em Vetores de Dados Mapeados aos *K* Protótipos Mais Próximos

A segunda abordagem descrita, chamada Modelo KSOM Baseado em Dados (*Data-based Multiple KSOM Model* - D-MKSOM), é similar ao modelo P-MKSOM, diferindo apenas na maneira que os vetores de coeficientes $\mathbf{c}_i$, $i = 1, ..., N$, são estimados. Ao invés de utilizar o vetor protótipo do neurônio i e seus K vizinhos mais próximos, o modelo D-MKSOM computa o vetor de coeficientes $\mathbf{c}_i$ do neurônio i usando os *vetores de dados* (de treinamento) que são mapeados para o neurônio e seus K vizinhos mais próximos. Em outras palavras, com o objetivo de estimar o vetor $\mathbf{c}_i$, o modelo D-MKSOM utiliza todos os vetores de dados (de treinamento) pertencentes à região formada pelas células de Voronoi do neurônio i e de seus K vizinhos mais próximos.

A primeira e segunda etapas na construção do modelo D-MKSOM são as mesmas que as do P-MKSOM: (*i*) treinar o modelo VQTAM utilizando os dados de treinamento disponíveis. (*ii*) Então, encontre o conjunto $\mathscr{J}_i = i \cup \{i_k\}_{k=1}^K$ contendo os índices dos K vizinhos mais próximos do vetor protótipo $\mathbf{w}_i^{in}$, $i = 1, ..., N$, como definido em (3.1).

Um terceiro passo é necessário e consiste em encontrar o conjunto de vetores de dados (de treinamento) que são mapeados aos protótipos $\mathbf{w}_i^{in}$, $\mathbf{w}_{i_1}^{in}$, $\mathbf{w}_{i_2}^{in}$, $\ldots$, $\mathbf{w}_{i_K}^{in}$, para $i = 1, \ldots, N$.

Faça $n^{(i)}$ ser o número de vetores de entrada $\mathbf{x}^{in} \in \mathbb{R}^{p+q}$ mapeados para a célula de Voronoi do neurônio i. Similarmente, faça com que $n^{(i_k)}$ seja o número de vetores de entrada $\mathbf{x}^{in} \in \mathbb{R}^{p+q}$ mapeados para a célula de Voronoi do k-ésimo vizinho mais próximo do neurônio i. Desse modo, o número total de vetores mapeados i e seus K vizinhos mais próximos é dado por

$$n_i = n^{(i)} + n^{(i_1)} + n^{(i_2)} + \cdots + n^{(i_K)}. \tag{3.6}$$

Tabela 3 – Algoritmo de Aprendizagem do Modelo P-MKSOM

MODELO P-MKSOM

Constantes	**Valores Típicos**
α : Taxa de aprendizagem	$0.001 < \alpha < 0.5$
σ : Abertura da vizinhança topológica	$0.001 < \sigma < (g/2)$
λ : Constante de regularização	$\lambda = 0.001$

Entradas

$\mathbf{x}^{in}(t)$: vetor de entrada, dimensão$(p+q)\times 1$ — $x^{out}(t)$: variável de saída

Algoritmo

1. Inicialização $(t=0)$

$\mathbf{w}_i^{in}, w_i^{out} \sim U(0,1), i = 1,...,g$

2. Aprendizagem do modelo $(t = 1,2,...,N_1)$

2.1 Escolha do neurônio vencedor:

$i^*(t) = \text{argmin}_i\{\|\mathbf{x}^{in}(t) - \mathbf{w}_i^{in}(t)\|\}$.

2.2 Atualização dos vetores de pesos e de coeficientes:

$h(i^*,i;t) = \exp\left(-\frac{\|\mathbf{r}_i(t)-\mathbf{r}_{i^*}(t)\|^2}{2\sigma^2(t)}\right)$,

$\mathbf{w}_i^{in}(t+1) = \mathbf{w}_i^{in}(t) + \alpha(t)h(i^*,i;t)[\mathbf{x}^{in}(t) - \mathbf{w}_i^{in}(t)]$,

$w_i^{out}(t+1) = w_i^{out}(t) + \alpha(t)h(i^*,i;t)[x^{out}(t) - w_i^{out}(t)]$.

3. Estimação do vetor de coeficientes do neurônio i

3.1 Determinação dos índices dos K vetores protótipos mais próximos ao vetor $\mathbf{w}_i^{in}$:

$\mathscr{J}_i = i \cup \{i_k\}_{k=1}^K$.

3.2 Obtenção do vetor $\mathbf{b}_i^{out}$ e da matriz de regressão $\mathbf{R}_i$:

$$\mathbf{b}_i^{out} = \begin{bmatrix} w_i^{out} & w_{i_1}^{out} & \cdots & w_{i_K}^{out} \end{bmatrix}^T,$$

$$\mathbf{R}_i = \begin{pmatrix} \left(\mathbf{w}_i^{in}\right)^T \\ \left(\mathbf{w}_{i_1}^{in}\right)^T \\ \vdots \\ \left(\mathbf{w}_{i_K}^{in}\right)^T \end{pmatrix} = \begin{pmatrix} w_{i,1}^{in} & w_{i,2}^{in} & \cdots & w_{i,p+q}^{in} \\ w_{i_1,1}^{in} & w_{i_1,2}^{in} & \cdots & w_{i_1,p+q}^{in} \\ \vdots & \vdots & \vdots & \vdots \\ w_{i_K,1}^{in} & w_{i_K,2}^{in} & \cdots & w_{i_K,p+q}^{in} \end{pmatrix}.$$

3.3 Cálculo do vetor de coeficientes associado a w_i^{in}:

$\mathbf{c}_i = \left(\mathbf{R}_i^T\mathbf{R}_i + \lambda\mathbf{I}\right)^{-1}\mathbf{R}_i^T\mathbf{b}_i^{out}$.

4. Saída do modelo $(t = 1,2,...,N_2)$

4.1 Escolha do neurônio vencedor:

$i^*(t) = \text{argmin}_i\{\|\mathbf{x}^{in}(t) - \mathbf{w}_i^{in}(t)\|\}$.

4.2 Cálculo da saída estimada: $\hat{u}(t) = \mathbf{c}_{i*}^T\mathbf{x}^{in}(t)$

4.3 Cálculo do erro: $e(t) = u(t) - \hat{u}(t)$.

Saídas

$\hat{u}(t)$: saída estimada — $e(t)$: erro

Também, fazendo que $\mathbf{X}_i^{in}$ seja uma matriz de dados $(p+q)\times n^{(i)}$ cujas colunas são os vetores $\mathbf{x}^{in}$ mapeados para a célula de Voronoi do neurônio i. Finalmente, faça $\mathbf{x}_i^{out} \in \mathbb{R}^{n^{(i)}}$ sero vetor contendo as saídas desejadas x^{out} associadas aos vetores $\mathbf{x}^{in} \in \mathbf{X}_i^{in}$.

Seguindo o mesmo raciocínio, $\mathbf{X}_{i_k}^{in}$ é uma matriz de dados $(p+q)\times n^{(i_k)}$ cujas colunas são os vetores $\mathbf{x}^{in}$ mapeados para as células de Voronoi do neurônio i_k, $k = 1,\ldots,K$. De acordo, $\mathbf{x}_{i_k}^{out} \in \mathbb{R}^{n^{(i_k)}}$ é o vetor contendo as saídas a serem alcançadas x^{out} associadas com os vetores $\mathbf{x}^{in} \in \mathbf{X}_{i_k}^{in}$.

Uma vez que os pares $\{\mathbf{X}_i^{in},\mathbf{x}_i^{out}\}$, $\{\mathbf{X}_{i_1}^{in},\mathbf{x}_{i_1}^{out}\}$, $\{\mathbf{X}_{i_2}^{in},\mathbf{x}_{i_2}^{out}\}$, ..., $\{\mathbf{X}_{i_K}^{in},\mathbf{x}_{i_K}^{out}\}$, estão determinados para o neurônio i e seus K vizinhos mais próximos, nós podemos construir o modelo linear local para o neurônio i.

Para esse propósito, assumindo que os conjuntos $\mathbf{x}_i^{out}$ e $\mathbf{x}_{i_k}^{out}$, $k = 1,\ldots,K$ estão arranjados como vetores colunas, pode-se construir o vetor $\mathbf{b}_i^{out} \in \mathbb{R}^{n_i}$ da seguinte forma:

$$\mathbf{b}_i^{out} = \begin{bmatrix} \mathbf{x}_i^{out} \\ \mathbf{x}_{i_1}^{out} \\ \vdots \\ \mathbf{x}_{i_K}^{out} \end{bmatrix}_{n_i\times 1} . \tag{3.7}$$

Similarmente, a matriz de regressão $\mathbf{R}_i \in \mathbb{R}^{n_i\times(p+q)}$ é construída utilizando as matrizes de dados $\mathbf{X}_i^{in}$ e $\mathbf{X}_{i_k}^{in}$, $k = 1,\ldots,K$, como segue:

$$\mathbf{R}_i = \begin{pmatrix} \left(\mathbf{X}_i^{in}\right)^T \\ \left(\mathbf{X}_{i_1}^{in}\right)^T \\ \vdots \\ \left(\mathbf{X}_{i_K}^{in}\right)^T \end{pmatrix}_{n_i\times(p+q)} , \tag{3.8}$$

onde o sobrescrito T denota o(a) vetor/matriz transposto(a).

Assim, o vetor de coeficientes do neurônio i, $\mathbf{c}_i \in \mathbb{R}^{p+q}$, é estimado utilizando o método dos mínimos quadrados ordinário como

$$\mathbf{c}_i = \left(\mathbf{R}_i^T\mathbf{R}_i + \lambda\mathbf{I}\right)^{-1}\mathbf{R}_i^T\mathbf{b}_i^{out}, \tag{3.9}$$

onde $\mathbf{I}$ é uma matriz identidade de dimensão $(p+q)\times(p+q)$ e $\lambda > 0$ (e.g. $\lambda = 0.001$) é uma pequena constante de regularização.

Uma vez que os N modelos de regressão locais são construídos, eles podem ser utilizados para aproximar a saída do mapeamento não linear de interesse. Lembrando que no

trabalho de Souza (2012) os modelos são utilizados para o problema de identificação inversa de sistemas. Assim, o modelo D-MKSOM estima $u(t)$ usando a Equação (3.5).

Tabela 4 – Algoritmo de Aprendizagem do Modelo D-MKSOM

MODELO D-MKSOM

Constantes	**Valores Típicos**
α : Taxa de aprendizagem	$0.001 < \alpha < 0.5$
σ : Abertura da vizinhança topológica	$0.001 < \sigma < (g/2)$
λ : Constante de regularização	$\lambda = 0.001$

Entradas

$\mathbf{x}^{in}(t)$: vetor de entrada, dimensão$(p+q)\times 1$ — $x^{out}(t)$: variável de saída

Algoritmo

1. Inicialização $(t = 0)$

$\mathbf{w}_i^{in}, w_i^{out} \sim U(0,1), i = 1,...,g$

2. Aprendizagem do modelo $(t = 1,2,...,N_1)$

2.1 Escolha do neurônio vencedor:

$i^*(t) = \text{argmin}_i\{\|\mathbf{x}^{in}(t) - \mathbf{w}_i^{in}(t)\|\}$.

2.2 Atualização dos vetores de pesos e de coeficientes:

$h(i^*, i; t) = \exp\left(-\frac{\|\mathbf{r}_i(t) - \mathbf{r}_{i^*}(t)\|^2}{2\sigma^2(t)}\right)$,

$\mathbf{w}_i^{in}(t+1) = \mathbf{w}_i^{in}(t) + \alpha(t)h(i^*, i; t)[\mathbf{x}^{in}(t) - \mathbf{w}_i^{in}(t)]$,

$w_i^{out}(t+1) = w_i^{out}(t) + \alpha(t)h(i^*, i; t)[x^{out}(t) - w_i^{out}(t)]$.

3. Estimação do vetor de coeficientes do neurônio i

3.1 Determinação dos índices dos K vetores protótipos mais próximos ao vetor $\mathbf{w}_i^{in}$:

$\mathscr{J}_i = i \cup \{i_k\}_{k=1}^{K}$.

3.2 Obtenção do vetor $\mathbf{b}_i^{out}$ e da matriz de regressão $\mathbf{R}_i$:

$$\mathbf{b}_i^{out} = \begin{bmatrix} \mathbf{x}_i^{out} \\ \mathbf{x}_{i_1}^{out} \\ \vdots \\ \mathbf{x}_{i_K}^{out} \end{bmatrix},$$

$$\mathbf{R}_i = \begin{pmatrix} \left(\mathbf{X}_i^{in}\right)^T \\ \left(\mathbf{X}_{i_1}^{in}\right)^T \\ \vdots \\ \left(\mathbf{X}_{i_K}^{in}\right)^T \end{pmatrix}.$$

3.3 Cálculo do vetor de coeficientes associado a w_i^{in}:

$\mathbf{c}_i = \left(\mathbf{R}_i^T\mathbf{R}_i + \lambda\mathbf{I}\right)^{-1}\mathbf{R}_i^T\mathbf{b}_i^{out}$.

4. Saída do modelo $(t = 1,2,...,N_2)$

4.1 Escolha do neurônio vencedor:

$i^*(t) = \text{argmin}_i\{\|\mathbf{x}^{in}(t) - \mathbf{w}_i^{in}(t)\|\}$.

4.2 Cálculo da saída estimada: $\hat{u}(t) = \mathbf{c}_{i*}^T\mathbf{x}^{in}(t)$

4.3 Cálculo do erro: $e(t) = u(t) - \hat{u}(t)$.

Saídas

$\hat{u}(t)$: saída estimada — $e(t)$: erro

No capítulo a seguir será apresentada a proposta de modelos múltiplos robustos a partir do modelos P-MKSOM e D-MKSOM abordados neste capítulo, com o objetivo de reduzir a influência de outliers no erro de predição estimado.

4 PROPOSTA: MODELAGEM LOCAL MÚLTIPLA ROBUSTA

Geralmente, ao usar algoritmos de regressão, o mesmo valor de contribuição é considerado para todas as amostras de erro. No entanto, o valor de erro pode ser originado a partir de dados com outliers, o que afeta negativamente a capacidade de generalização dos algoritmos. Esse problema pode ser corrigido através da remoção de todos os outliers presentes nos dados, porém esse processo pode ser cansativo e complicado. Uma alternativa viável é a aplicação da chamada regressão robusta, utilizando métodos de estimação que não são tão sensitivos a outliers, permitindo a adaptação e reponderamento de algoritmos como a ELM e o OLS (BARRETO; BARROS, 2016; SOUZA; BARRETO; CORONA, 2015).

O processo de regressão robusta é baseado na técnica de estimação M proposta por Huber (HUBER, 1964) - em que M está relacionado a "máxima verossimilhança", a qual fornece a geração de um aumento na robustez do modelo minimizando uma função de custo que difere da tradicional soma do erros quadráticos e que permite a ponderação e filtragem da contribuição de cada erro para a função. A seguir será mostrada a aplicação da técnica ao algoritmo de Mínimos Quadrados Ordinário, o modelo pode ser implementado à partir dos seguintes passos:

4.1 Estimação inicial de parâmetros

Primeiramente, deve ser gerado o vetor de parâmetros $\hat{\theta}(0)$ conforme proposto para os algoritmos P-MKSOM e D-MKSOM, através do cálculo da Pseudo-Inversa e da utilização do Método dos Mínimos Quadrados (*OLS*). Esse vetor corresponde às matrizes de coeficientes calculadas através das Equações 3.2 e 3.9, ou seja, inicialmente segue-se com a geração dos vetores protótipos através do algoritmo VQTAM e da associação dos protótipos vencedores para o algoritmo P-MKSOM e dos dados associados aos protótipos vencedores para a técnica D-MKSOM.

4.2 Geração de pesos

A cada iteração t de treinamento do algoritmo, devem ser coletados os resíduos da iteração anterior $e_n(t-1)$, com $n = 1, ..., N_1$ e então armazenados os valores dos pesos que foram gerados $w_n(t-1) = w(e_n(t-1))$. O valor dos pesos podem ser computados através da utilização de uma gama de funções custo, apresentadas na Tabela 5, como por exemplo Andrews, Cauchy,

ou a função de Huber que será utilizada neste trabalho e é descrita a seguir,

$$w(e_n) = \begin{cases} \frac{k_e}{e_n} \text{ se } \|e_n\| > k_e, \\ 1 \text{ por outro lado.} \end{cases} \tag{4.1}$$

A constante k_e se trata de um valor de sintonia, então pode-se entender que, dada a seguinte condição $\|e_n\| > k_e$, quanto maior o valor do resíduo menor o valor do peso associado.

Quanto menor o valor da constante de sintonia k_e, mais resistentes os outliers se tornam, porém se os erros apresentarem uma distribuição normal, o algoritmo se torna menos eficiente. Para a função de Huber $k_e = 1,345s$, sendo s uma constante de robustez calculada do desvio padrão dos resíduos. Neste trabalho foi utilizada a expressão $s = MAD/0,6745$, sendo MAD o desvio absoluto mediano do resíduo.

Nome	Função Custo (ρ)	Função Peso (w)	Limiar (k)
Andrews	$\begin{cases} k^2\left[1-cos(\frac{e_n}{k})\right], \lvert\frac{e_n}{k}\rvert \leq \pi \\ 2k^2, \lvert\frac{e_n}{k}\rvert > \pi \end{cases}$	$\begin{cases} (\frac{e_n}{k})^{-1} sen(\frac{e_n}{k}), \lvert\frac{e_n}{k}\rvert \leq \pi \\ 0, \lvert\frac{e_n}{k}\rvert > \pi \end{cases}$	1,339
Bisquare	$\begin{cases} \frac{k^2}{6}\{1-\left[1-(\frac{e_n}{k})^2\right]^3\}, \lvert\frac{e_n}{k}\rvert \leq 1 \\ \frac{k^2}{6}, \lvert\frac{e_n}{k}\rvert > 1 \end{cases}$	$\begin{cases} \left[1-(\frac{e_n}{k})^2\right]^2, \lvert\frac{e_n}{k}\rvert \leq 1 \\ 0, \lvert\frac{e_n}{k}\rvert > 1 \end{cases}$	4,685
Cauchy	$\frac{k^2}{2} log(1+(\frac{e_n}{k})^2)$	$\frac{1}{1+(\frac{e_n}{k})^2}$	2,385
Fair	$k^2\left[\frac{\lvert e_n\rvert}{k} - log(1+(\frac{\lvert e_n\rvert}{k}))\right]$	$\frac{1}{1+\frac{\lvert e_n\rvert}{k}}$	1,400
Logistic	$k^2 log\left[cosh(\frac{e_n}{k})\right]$	$(\frac{e_n}{k})^{-1} tanh(\frac{e_n}{k})$	1,205
OLS	e_n^2	1	-
Talwar	$\begin{cases} \frac{e_n^2}{2}), \lvert\frac{e_n}{k}\rvert \leq 1 \\ \frac{k^2}{2}, \lvert\frac{e_n}{k}\rvert > 1 \end{cases}$	$\begin{cases} 1, \lvert\frac{e_n}{k}\rvert \leq 1 \\ 0, \lvert\frac{e_n}{k}\rvert > 1 \end{cases}$	2,795
Welsch	$\frac{k^2}{2}\left[(1-exp(-(\frac{e_n}{k})^2))\right]$	$exp(-(\frac{e_n}{k})^2)$	2,985

Tabela 5 – Lista de funções custo (ρ) e suas correspondentes funções peso de vários estimadores M comumente encontrados na literatura de regressão robusta. Valores padrão do parâmetro de limiar k estão mostrados conforme fornecidos pelo comando *robustfit* do Matlab

4.3 Estimação de parâmetros robustos

No último passo é necessária a solução da equação de mínimos quadrados ponderado para que seja então alcançado um novo vetor de parâmetros,

$$\hat{\theta}(t) = (\mathbf{X}^T\mathbf{B}(t-1)\mathbf{X})^{-1}\mathbf{X}^T\mathbf{B}(t-1)\mathbf{y}, \tag{4.2}$$

onde $\mathbf{B}(t-1) = \text{diag}(w_n(t-1))$ é uma matriz $N_1 \times N_1$.

Considerando as definições acima, o vetor de coeficientes do modelo P-MKSOM passa a ser calculado conforme a expressão a seguir

$$\mathbf{c}_i = (\mathbf{R}_i^T\mathbf{B}(t-1)\mathbf{R}_i + \lambda\mathbf{I})^{-1}\mathbf{R}_i\mathbf{B}(t-1)\mathbf{b}_i^{out}, \tag{4.3}$$

para o modelo D-MKSOM, a etapa de treinamento agora se mantém no cálculo do vetor de coeficientes através da mesma expressão apresentada na Equação (4.3).

A aplicação do algoritmo apresentada à técnica de Mínimos Quadrados Ordinário é chamada de Mínimos Quadrados Iterativamente Reponderado (*Iterative Reweighted Least Squares* - IRLS), proposto em (SOUZA; BARRETO; CORONA, 2015), podendo ser aplicada a diversos algoritmos como MLP, ELM (BARRETO; BARROS, 2016), SOM (FOX; WEISBERG, 2010) etc.

De agora em diante, os modelos propostos acima serão utilizados na modelagem de sistemas industriais com comportamentos não-lineares, sendo avaliados os modelos obtidos a partir de dados sem e com a contaminação de outliers a fim de entender o impacto da aplicação da estimação M aos modelos, que serão chamados a partir de agora de P-MKSOM Robusto (RP-MKSOM) e D-MKSOM Robusto (RD-MKSOM).

No capítulo a seguir serão apresentadas as condições básicas para a construção do algoritmo ELM e do algoritmo ELM robusto, rede neural global que será aplicada aos bancos de dados em sua versão robusta para a obtenção de modelos e comparada com os modelos aqui propostos.

Tabela 6 – Algoritmo de Aprendizagem do Modelo RP-MKSOM

MODELO RP-MKSOM

Constantes	**Valores Típicos**
α : Taxa de aprendizagem	$0.001 < \alpha < 0.5$
σ : Abertura da vizinhança topológica	$0.001 < \sigma < (g/2)$
λ : Constante de regularização	$\lambda = 0.001$

Entradas

$\mathbf{x}^{in}(t)$: vetor de entrada, dimensão$(p+q)\times 1$ $\quad$ $x^{out}(t)$: variável de saída

Algoritmo

1. Inicialização $(t=0)$

$\mathbf{w}_i^{in}, w_i^{out} \sim U(0,1), i = 1,...,g$

2. Aprendizagem do modelo $(t = 1,2,...,N_1)$

2.1 Escolha do neurônio vencedor:

$i^*(t) = \text{argmin}_i\{\|\mathbf{x}^{in}(t) - \mathbf{w}_i^{in}(t)\|\}$.

2.2 Atualização dos vetores de pesos e de coeficientes:

$h(i^*,i;t) = \exp\left(-\frac{\|\mathbf{r}_i(t)-\mathbf{r}_{i^*}(t)\|^2}{2\sigma^2(t)}\right)$,

$\mathbf{w}_i^{in}(t+1) = \mathbf{w}_i^{in}(t) + \alpha(t)h(i^*,i;t)[\mathbf{x}^{in}(t) - \mathbf{w}_i^{in}(t)]$,

$w_i^{out}(t+1) = w_i^{out}(t) + \alpha(t)h(i^*,i;t)[x^{out}(t) - w_i^{out}(t)]$.

3. Estimação do vetor de coeficientes robusto do neurônio i

3.1 Determinação dos índices dos K vetores protótipos mais próximos ao vetor $\mathbf{w}_i^{in}$:

$\mathscr{J}_i = i \cup \{i_k\}_{k=1}^K$.

3.2 Obtenção do vetor $\mathbf{b}_i^{out}$ e da matriz de regressão $\mathbf{R}_i$:

$$\mathbf{b}_i^{out} = \begin{bmatrix} w_i^{out} & w_{i_1}^{out} & \cdots & w_{i_K}^{out} \end{bmatrix}^T,$$

$$\mathbf{R}_i = \begin{pmatrix} \left(\mathbf{w}_i^{in}\right)^T \\ \left(\mathbf{w}_{i_1}^{in}\right)^T \\ \vdots \\ \left(\mathbf{w}_{i_K}^{in}\right)^T \end{pmatrix} = \begin{pmatrix} w_{i,1}^{in} & w_{i,2}^{in} & \cdots & w_{i,p+q}^{in} \\ w_{i_1,1}^{in} & w_{i_1,2}^{in} & \cdots & w_{i_1,p+q}^{in} \\ \vdots & \vdots & \vdots & \vdots \\ w_{i_K,1}^{in} & w_{i_K,2}^{in} & \cdots & w_{i_K,p+q}^{in} \end{pmatrix}.$$

3.3 Cálculo do vetor de coeficientes associado a w_i^{in}:

$\mathbf{c}_i = \left(\mathbf{R}_i^T\mathbf{R}_i + \lambda\mathbf{I}\right)^{-1}\mathbf{R}_i^T\mathbf{b}_i^{out}$.

3.4 Cálculo do erro de treinamento: $e(t) = u(t) - \hat{u}(t)$.

3.5 Ponderação de pesos $w(e_n(t-1))$:

$$w(e_n) = \begin{cases} \frac{k_e}{e_n} \text{ se } \|e_n\| > k_e, \\ 1 \text{ por outro lado.} \end{cases}.$$

3.6 Cálculo da matriz de ponderação:

$\mathbf{B}(t-1) = \text{diag}(w_n(t-1))$.

3.7 Cálculo do vetor de coeficientes robusto associado a w_i^{in}:

$\mathbf{c}_i = (\mathbf{R}_i^T\mathbf{B}(t-1)\mathbf{R}_i + \lambda\mathbf{I})^{-1}\mathbf{R}_i\mathbf{B}(t-1)\mathbf{b}_i^{out}$.

4. Saída do modelo $(t = 1,2,...,N_2)$

4.1 Escolha do neurônio vencedor:

$i^*(t) = \text{argmin}_i\{\|\mathbf{x}^{in}(t) - \mathbf{w}_i^{in}(t)\|\}$.

4.2 Cálculo da saída estimada: $\hat{u}(t) = \mathbf{c}_{i^*}^T\mathbf{x}^{in}(t)$

4.3 Cálculo do erro: $e(t) = u(t) - \hat{u}(t)$.

Saídas

$\hat{u}(t)$: saída estimada $\quad$ $e(t)$: erro

Tabela 7 – Algoritmo de Aprendizagem do Modelo RD-MKSOM

MODELO RD-MKSOM

Constantes e Valores Típicos: Idem modelo RP-MKSOM

Entradas

$\mathbf{x}^{in}(t)$: vetor de entrada, dimensão$(p+q)\times 1$ $\quad$ $x^{out}(t)$: variável de saída

Algoritmo

1. Inicialização $(t=0)$

$\mathbf{w}_i^{in}, w_i^{out} \sim U(0,1), i=1,...,g$

2. Aprendizagem do modelo $(t=1,2,...,N_1)$

2.1 Escolha do neurônio vencedor:

$i^*(t) = \text{argmin}_i\{\|\mathbf{x}^{in}(t) - \mathbf{w}_i^{in}(t)\|\}$.

2.2 Atualização dos vetores de pesos e de coeficientes:

$h(i^*,i;t) = \exp\left(-\frac{\|\mathbf{r}_i(t)-\mathbf{r}_{i^*}(t)\|^2}{2\sigma^2(t)}\right)$,

$\mathbf{w}_i^{in}(t+1) = \mathbf{w}_i^{in}(t) + \alpha(t)h(i^*,i;t)[\mathbf{x}^{in}(t) - \mathbf{w}_i^{in}(t)]$,

$w_i^{out}(t+1) = w_i^{out}(t) + \alpha(t)h(i^*,i;t)[x^{out}(t) - w_i^{out}(t)]$.

3. Estimação do vetor de coeficientes do neurônio i

3.1 Determinação dos índices dos K vetores protótipos mais próximos ao vetor $\mathbf{w}_i^{in}$:

$\mathscr{J}_i = i \cup \{i_k\}_{k=1}^K$.

3.2 Obtenção do vetor $\mathbf{b}_i^{out}$ e da matriz de regressão $\mathbf{R}_i$:

$$\mathbf{b}_i^{out} = \begin{bmatrix} \mathbf{x}_i^{out} \\ \mathbf{x}_{i_1}^{out} \\ \vdots \\ \mathbf{x}_{i_K}^{out} \end{bmatrix},$$

$$\mathbf{R}_i = \begin{pmatrix} \left(\mathbf{X}_i^{in}\right)^T \\ \left(\mathbf{X}_{i_1}^{in}\right)^T \\ \vdots \\ \left(\mathbf{X}_{i_K}^{in}\right)^T \end{pmatrix}.$$

3.3 Cálculo do vetor de coeficientes associado a w_i^{in}:

$\mathbf{c}_i = \left(\mathbf{R}_i^T\mathbf{R}_i + \lambda\mathbf{I}\right)^{-1}\mathbf{R}_i^T\mathbf{b}_i^{out}$.

3.4 Cálculo do erro de treinamento: $e(t) = u(t) - \hat{u}(t)$.

3.5 Ponderação de pesos $w(e_n(t-1))$:

$$w(e_n) = \begin{cases} \frac{k_e}{e_n} \text{ se } \|e_n\| > k_e, \\ 1 \text{ por outro lado.} \end{cases}.$$

3.6 Cálculo da matriz de ponderação:

$\mathbf{B}(t-1) = \text{diag}(w_n(t-1))$.

3.7 Cálculo do vetor de coeficientes robusto associado a w_i^{in}:

$\mathbf{c}_i = (\mathbf{R}_i^T\mathbf{B}(t-1)\mathbf{R}_i + \lambda\mathbf{I})^{-1}\mathbf{R}_i\mathbf{B}(t-1)\mathbf{b}_i^{out}$.

4. Saída do modelo $(t=1,2,...,N_2)$

4.1 Escolha do neurônio vencedor:

$i^*(t) = \text{argmin}_i\{\|\mathbf{x}^{in}(t) - \mathbf{w}_i^{in}(t)\|\}$.

4.2 Cálculo da saída estimada: $\hat{u}(t) = \mathbf{c}_{i^*}^T\mathbf{x}^{in}(t)$

4.3 Cálculo do erro: $e(t) = u(t) - \hat{u}(t)$.

Saídas

$\hat{u}(t)$: saída estimada $\quad$ $e(t)$: erro

5 MÁQUINA DE APRENDIZADO EXTREMO (ELM)

O algoritmo da Máquina de Aprendizado Extremo (*Extreme Learning Machine* - ELM) foi desenvolvido para a realização do treinamento de SFLNs (redes neurais sem retroalimentação de camada escondida única, do inglês) (HUANG; ZHU; ZIEW, 2006), sendo uma proposta extremamente simples capaz de aprimorar o processo de modelagem frente algumas dificuldades verificadas na implementação do algoritmo back-propagation. Dada a aplicação desse algoritmo em modelos globais e a utilização em trabalhos com a aplicação da estimação M, a rede neural será aqui apresentada, bem como a sua versão robusta apresentada em Barreto e Barros (2016).

5.1 Particularidades da ELM

Convecionalmente, ao realizar o treinamento de SFLN's, objetiva-se encontrar o grupo de parâmetros que satisfaça a condição elencada abaixo,

$$\| \mathbf{H}(\hat{\mathbf{w}}_1,...,\hat{\mathbf{w}}_{\tilde{N}},\hat{b}_1,...,\hat{b}_{\tilde{N}})\hat{\beta} - \mathbf{Y} \| = \min_{\mathbf{w}_i,b_i,\beta} \| \mathbf{H}(\hat{\mathbf{w}}_1,...,\hat{\mathbf{w}}_{\tilde{N}},\hat{b}_1,...,\hat{b}_{\tilde{N}})\beta - \mathbf{Y} \|, \tag{5.1}$$

sendo $\mathbf{H}$ a matriz de pesos da camada escondida de neurônios, constituída de $\hat{\mathbf{w}}_1,...,\hat{\mathbf{w}}_{\tilde{N}}$ valores de pesos dos neurônios da camada escondida e $\hat{b}_1,...,\hat{b}_{\tilde{N}}$ valores de bias, com β sendo o vetor de parâmetros e $\mathbf{Y}$ o vetor desejado de saída do sistema.

Para um valor desconhecido de $\mathbf{H}$ geralmente se utiliza um método de aprendizado baseado em gradiente para rastrear o valor mínimo da expressão $\| \mathbf{H}\beta - \mathbf{Y} \|$. Os algoritmos de gradiente mínimo que são geralmente utilizados realizam o ajuste do iterativo vetor $\mathbf{W}$ composto dos parâmetros de pesos $(\mathbf{w}_i, \beta_i)$ e biases (b_i), conforme a seguinte expressão:

$$\mathbf{W}_t = \mathbf{W}_{t-1} - \eta \frac{\partial E(\mathbf{W})}{\partial \mathbf{W}}, \tag{5.2}$$

em que η é a taxa de aprendizagem do algoritmo. De forma estabelecida os gradientes são computados utilizando o algoritmo de backpropagation, que apresenta algumas características indesejáveis e desafiadoras.

Um exemplo é a própria taxa de aprendizagem η que se for muito pequena leva o algoritmo a convergir de forma muito lenta, porém se for muito grande o algoritmo se

torna instável e diverge, então a sintonia da taxa de aprendizagem deve ser cuidadosa. Outra questão relativa ao BP é a presença de um mínimo local na superfície de erro, podendo levar a aprendizagem a estagnar antes de alcançar o mínimo global do sistema. O algoritmo de BP corre o risco de sobretreinamento, levando a uma pior generalização dos dados, fazendo com que sejam necessárias a utilização de técnicas de validação e avaliação de parada. Além desses fatores sabe-se que o algoritmo BP demanda um alto custo computacional.

O algoritmo ELM visa estabeler uma forma eficiente de tratar os desafios elencados acima, mantendo a performance quando do treinamento de SFLN's. Nas seções a seguir serão elencados os passos para a implementação da rede ELM:

5.1.1 Inicialização Aleatória dos Pesos da Camada Oculta

Esta etapa consiste na geração aleatória de pesos sinápticos w_{ij} com $i = 0, ..., h_1$ e $j = 0, ..., p+q$ sendo $h_1 (2 < h_1 < \infty)$ a quantidade de neurônios presentes na camada oculta. Então a matriz de pesos **W** é dada por:

$$\mathbf{W} = \begin{bmatrix} w_{10} & w_{11} & \cdots & w_{1(p+q)} \\ w_{20} & w_{21} & \cdots & w_{2(p+q)} \\ \vdots & \vdots & \vdots & \vdots \\ w_{h_1 0} & w_{h_1 1} & \cdots & w_{h_1(p+q)} \end{bmatrix}_{h_1 \times (p+q+1)} = \begin{bmatrix} w_1^T \\ w_2^T \\ \vdots \\ w_{h_1}^T \end{bmatrix}. \tag{5.3}$$

A arquitetura de pesos da rede é semelhante à arquitetura da rede MLP como pode ser observado na Figura 4, os algoritmos se diferem nas etapas posteriores a partir da etapa de treinamento.

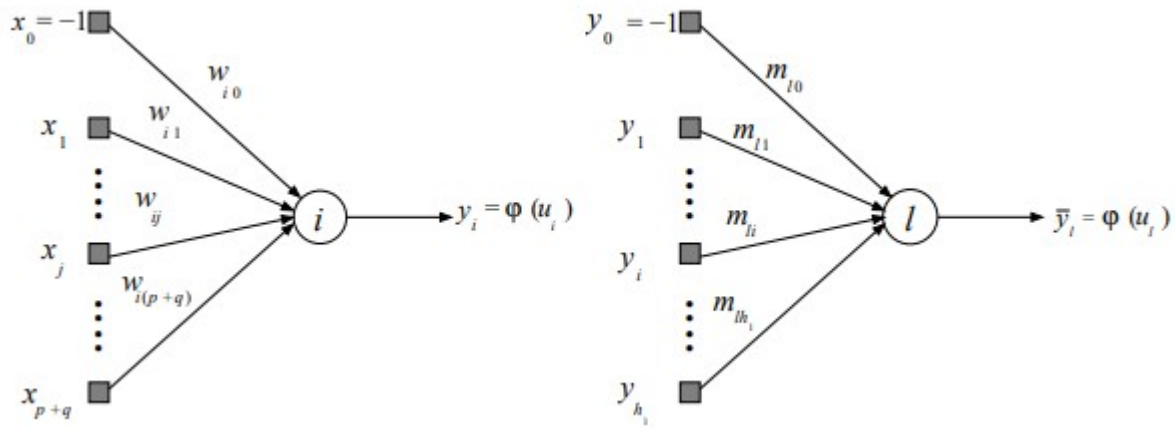


Figura 4 – Rede ELM Pesos e Neurônios.

Os pesos utilizados para a inicialização do processo de treinamento são tipicamente distribuídos de forma uniforme ou normal:
$w_{ij} \sim U(a,b)$ ou $w_{ij} \sim N(0,\sigma^2)$.

5.1.2 Acúmulo das Saídas dos Neurônios Ocultos

Essa etapa consiste no cálculo das ativações dos neurônios da camada oculta e suas respectivas saídas. Primeiramente, para as ativações referentes aos vetores de entrada apresentados calcula-se:

$$u_i(t) = \sum_{j=0}^{p+q} w_{ij} x_j(t) = \mathbf{w}_i^T \mathbf{x}(t), \tag{5.4}$$

onde t representa a iteração atual e varia conforme a quantidade de dados de treinamento N_1 existentes, ou seja, $t = 1,...,N_1$. Deste modo, a matriz com o valor de ativação de todos os neurônios ocultos é dada por:

$$\mathbf{u}(t) = \mathbf{W}\mathbf{x}(t), \tag{5.5}$$

assim sendo, as saídas dos neurônios podem ser calculadas:

$$\mathbf{y}(t) = \varphi(\mathbf{u}(t)) = \varphi(\mathbf{W}\mathbf{x}(t)). \tag{5.6}$$

O operador φ representa um função não-linear do tipo sigmoidal, em que geralmente se utilizam dois tipos de função: a sigmóide, apresentada na Eq. 5.7, e a tangente hiperbólica, apresentada na Eq. 5.8.

$$\varphi(v) = \frac{1}{1+exp\{-v\}}, \tag{5.7}$$

$$\varphi(v) = \frac{1-exp\{-v\}}{1+exp\{-v\}}. \tag{5.8}$$

Os vetores de saída constituirão uma matriz $\mathbf{Y}$, apresentada em (5.9), que será utilizada no passo 3 para calcular os pesos do neurônio de saída.

$$\mathbf{Y} = \begin{bmatrix} -1 & -1 & ... & -1 \\ y_1(1) & y_1(2) & ... & y_1(N_1) \\ y_2(1) & y_2(2) & ... & y_2(N_1) \\ \vdots & \vdots & \vdots & \vdots \\ y_{h_1}(1) & y_{h_1}(2) & ... & y_{h_1}(N_1) \end{bmatrix}_{h_1+1 \times N_1} \tag{5.9}$$

5.1.3 Determinação dos Pesos do Neurônio de Saída

Dado um conjunto de dados de entrada de treino $\mathbf{x}(t)$, existe um conjunto de dados de saída desejados $\mathbf{d}(t)$ que podem ser organizados matricialmente como:

$$\mathbf{d} = \begin{bmatrix} d(1) \\ d(2) \\ \vdots \\ d(N_1) \end{bmatrix}_{N_1 \times 1} . \tag{5.10}$$

A partir dos dados de saída desejados, pode-se chegar a uma relação que melhor represente a relação entre as saídas dos neurônios das camadas ocultas e o valor de saída desejado da rede,

$$\mathbf{d}(t) = \mathbf{m}^T \mathbf{y}(t), \tag{5.11}$$

sendo $\mathbf{m}$ o vetor de pesos de saída da rede, que podem ser calculados utilizando o método dos mínimos quadrados (OLS), também conhecido como pseudoinversa (AGUIRRE, 2014) (PRINCIPE; EULIANO; LEFEBVRE, 2000), dado pela expressão:

$$\mathbf{m} = (\mathbf{Y}^T \mathbf{Y})^{-1} \mathbf{Y}^T \mathbf{D}. \tag{5.12}$$

Alguns trabalhos envolvendo aplicações, adaptações e utilização da ELM em diferentes áreas podem ser vistos em Soria-Olivas *et al.* (2011), Barreto e Barros (2016), Zhang *et al.* (2017), Liu *et al.* (2020) e Yang *et al.* (2021).

Tabela 8 – Algoritmo de Aprendizagem do Modelo ELM

MODELO ELM

Entradas

$\mathbf{x}(t)$: vetor de entrada, dimensão$(p+q+1)\times 1$ $u(t)$: variável observada

Algoritmo

1. Inicialização aleatória dos pesos ocultos $(t=0)$

$w_{ij} \sim U(a,b)$ ou $w_{ij} \sim N(0,\sigma^2)$, $i=1,...,h_1$

2. Obtenção do vetor de pesos de saída $(t=1,2,...,N_1)$

2.1 Cálculo das ativações dos neurônios ocultos:

$u_i(t) = \mathbf{w}_i^T \mathbf{x}(t)$,

$\mathbf{y}(t) = \varphi(\mathbf{u}(t)) = \varphi(\mathbf{W}\mathbf{x}(t))$,

onde$\varphi(\cdot)$ é uma função sigmoidal:

$\varphi(v) = \frac{1}{1+exp\{-v\}}$, (função logística)

$\varphi(v) = \frac{1-exp\{-v\}}{1+exp\{-v\}}$, (tangente hiperbólica)

2.2 Acúmulo das ativações dos neurônios ocultos:

$$\mathbf{Y} = \begin{bmatrix} -1 & -1 & ... & -1 \\ y_1(1) & y_1(2) & ... & y_1(N_1) \\ y_2(1) & y_2(2) & ... & y_2(N_1) \\ \vdots & \vdots & \vdots & \vdots \\ y_{h_1}(1) & y_{h_1}(2) & ... & y_{h_1}(N_1) \end{bmatrix}.$$

2.3 Cálculo do vetor de pesos $\mathbf{m} \in \mathbb{R}^{h_1+1}$:

$\mathbf{d}(t) = \mathbf{m}^T \mathbf{y}(t)$,

$\mathbf{m} = (\mathbf{Y}^T\mathbf{Y})^{-1}\mathbf{Y}^T\mathbf{D}$.

3. Etapa de teste $(t=1,2,...,N_2)$

3.1 Cálculo das ativações dos neurônios ocultos:

$$\mathbf{y}(\mathbf{t}) = \begin{bmatrix} -1 \\ y(t) \end{bmatrix} = \begin{bmatrix} -1 \\ \varphi(\mathbf{W}\mathbf{x}(t)) \end{bmatrix}.$$

3.2 Cálculo da saída da rede:

$\hat{\mathbf{u}}(t) = \mathbf{m}^T \mathbf{y}(t)$.

Saídas

$\hat{u}(t)$: saída estimada $e(t)$: erro

5.2 ELM Robusto

No trabalho de Barreto e Barros (2016) foi realizada a extensão da ELM através da utilização da Estimação M com a aplicação do algoritmo IRLS aqui apresentado na atualização dos pesos dos modelos de regressão gerados. Foi construída a partir disso a técnica chamada de RELM-B, voltada para aplicação do IRLS na versão batelada da rede ELM.

Em linhas gerais, para a implementação do algoritmo RELM-B, primeiramente deve-se estimar o vetor de parâmetros $\hat{\beta}_i(0)$ utilizado para estimar o vetor de saída da rede, o vetor deve ser estimado normalmente com a técnica OLS. Após a aquisição do vetor, para cada iteração do algoritmo IRLS deve-se computar os resíduos da iteração anterior, denotados como $e_{in}(t-1), n = 1, ..., N_1$ e associados ao i-ésimo neurônio de saída, com os resíduos obtidos podem-se encontrar os seus pesos correspondentes $w_{in}(t-1) = w[e_{in}(t-1)]$. O novo vetor de parâmetros é estimado através da equação a seguir,

$$\hat{\beta}_i(t) = (\mathbf{Y}^T\mathbf{B}(t-1)\mathbf{Y})^{-1}\mathbf{Y}^T\mathbf{B}(t-1)\mathbf{D}_i^T, \tag{5.13}$$

onde $\mathbf{B}(t-1) = diagw_{in}(t-1)$ é uma matriz de pesos $N_1 \times N_1$.

Como pode ser observado o processo de geração de modulo robusto utilizando a rede neural ELM a partir da estimação M é semelhante ao que foi utilizado nos modelos locais propostos nesse trabalho. No trabalho em que foi proposto, os autores utilizaram o algoritmo desenvolvido para tratar outliers presentes em problemas de classificação de padrões. No trabalho foi desenvolvida uma versão batelada e uma versão sequencial do algoritmo, bem como houve a utilização do algoritmo de Otimização de Enxame de Partículas (PSO, do inglês), para selecionar o melhor modelo em conjuntos de dados artificiais e reais utilizando diferentes funções de custo para a ponderação do erro.

A partir desse ponto o trabalho será dedicado à apresentação dos bancos de dados, metodologia de aquisição dos resultados e de avaliação dos melhores modelos.

Tabela 9 – Algoritmo de Aprendizagem do Modelo RELM-B

MODELO RELM-B

Entradas

$\mathbf{x}(t)$: vetor de entrada, dimensão$(p+q+1)\times 1$ $u(t)$: variável observada

Algoritmo

1. Inicialização aleatória dos pesos ocultos $(t=0)$

$w_{ij} \sim U(a,b)$ ou $w_{ij} \sim N(0,\sigma^2)$, $i=1,...,h_1$

2. Obtenção do vetor de pesos de saída $(t=1,2,...,N_1)$

2.1 Cálculo das ativações dos neurônios ocultos:

$u_i(t) = \mathbf{w}_i^T \mathbf{x}(t)$,

$\mathbf{y}(t) = \varphi(\mathbf{u}(t)) = \varphi(\mathbf{W}\mathbf{x}(t))$,

onde$\varphi(\cdot)$ é uma função sigmoidal:

$\varphi(v) = \frac{1}{1+exp\{-v\}}$, (função logística)

$\varphi(v) = \frac{1-exp\{-v\}}{1+exp\{-v\}}$, (tangente hiperbólica)

2.2 Acúmulo das ativações dos neurônios ocultos:

$$\mathbf{Y} = \begin{bmatrix} -1 & -1 & ... & -1 \\ y_1(1) & y_1(2) & ... & y_1(N_1) \\ y_2(1) & y_2(2) & ... & y_2(N_1) \\ \vdots & \vdots & \vdots & \vdots \\ y_{h_1}(1) & y_{h_1}(2) & ... & y_{h_1}(N_1) \end{bmatrix}.$$

2.3 Cálculo do vetor de pesos $\mathbf{m} \in \mathbb{R}^{h_1+1}$:

$\mathbf{d}(t) = \mathbf{m}^T \mathbf{y}(t)$,

$\mathbf{m} = (\mathbf{Y}^T\mathbf{Y})^{-1}\mathbf{Y}^T\mathbf{D}$.

2.4 Cálculo do erro de treinamento: $e(t) = u(t) - \hat{u}(t)$.

2.5 Ponderação de pesos $w(e_n(t-1))$:

$$w(e_n) = \begin{cases} \frac{k_e}{e_n} \text{ se } \|e_n\| > k_e, \\ 1 \text{ por outro lado.} \end{cases}.$$

2.6 Cálculo da matriz de ponderação:

$\mathbf{B}(t-1) = \text{diag}(w_n(t-1))$.

2.7 Cálculo do vetor de coeficientes robusto associado a w_i^{in}:

$\hat{\beta}_i(t) = (\mathbf{Y}^T\mathbf{B}(t-1)\mathbf{Y})^{-1}\mathbf{Y}^T\mathbf{B}(t-1)\mathbf{D}_i^T$.

3. Etapa de teste $(t=1,2,...,N_2)$

3.1 Cálculo das ativações dos neurônios ocultos:

$$\mathbf{y}(\mathbf{t}) = \begin{bmatrix} -1 \\ y(t) \end{bmatrix} = \begin{bmatrix} -1 \\ \varphi(\mathbf{W}\mathbf{x}(t)) \end{bmatrix}.$$

3.2 Cálculo da saída da rede:

$\hat{\mathbf{u}}(t) = \mathbf{m}^T\mathbf{y}(t)$.

Saídas

$\hat{u}(t)$: saída estimada $e(t)$: erro

6 CONJUNTOS DE DADOS E AVALIAÇÃO DOS ALGORITMOS

O trabalho aqui desenvolvido consiste na implementação de uma técnica de tratamento de outliers em modelos múltiplos locais desenvolvida por Souza (2012), tomando o método de *Estimação M* com o objetivo de aumentar a robustez dos modelos obtidos. O tratamento de outliers foi realizado utilizando um método iterativo que acompanhou a etapa de treinamento dos algoritmos através da inserção de pesos nos erros gerados a partir dos dados preditos obtidos nesta etapa. Para a validação dos modelos aqui propostos, é realizada a identificação de sistemas dinâmicos industriais, que serão apresentados a seguir.

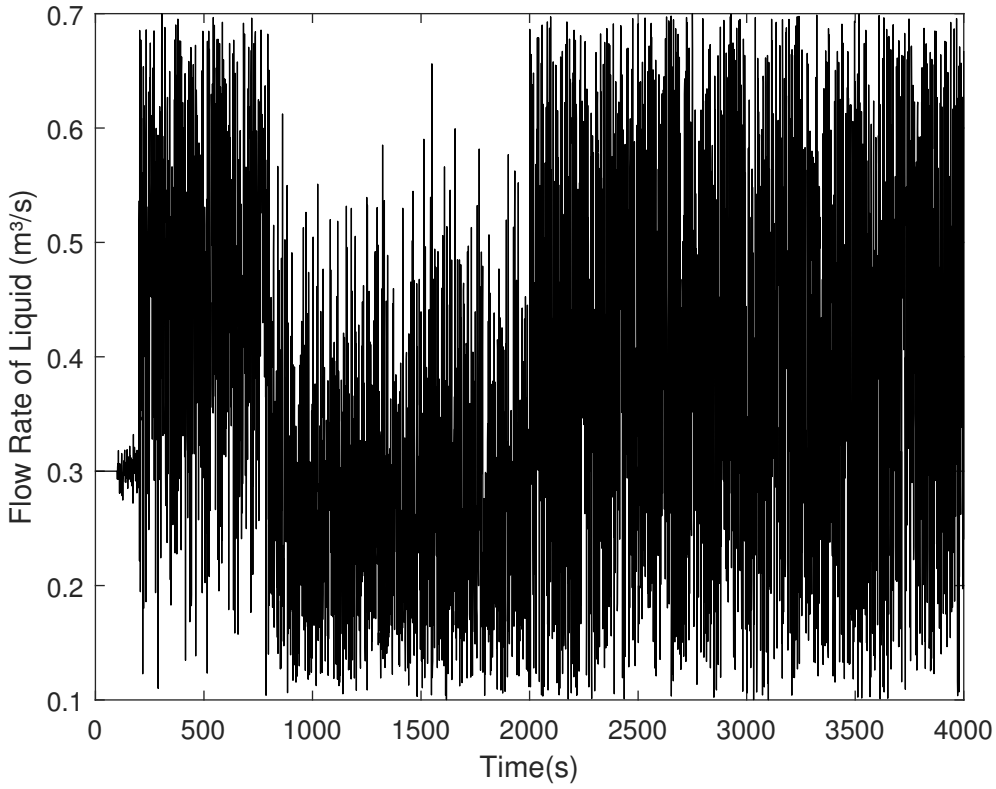
Neste estudo, modelos locais múltiplos robusto foram obtidos a partir de dois conjuntos de dados, tomados do repositório *DaISy* (Database for the Identification of Systems), que foram estudados e modelados no trabalho de Souza (2012), através dos modelos P-MKSOM e D-MKSOM sem realizar o tratamento de outliers através da estimação M proposto no presente trabalho.

6.1 Descrição dos Dados

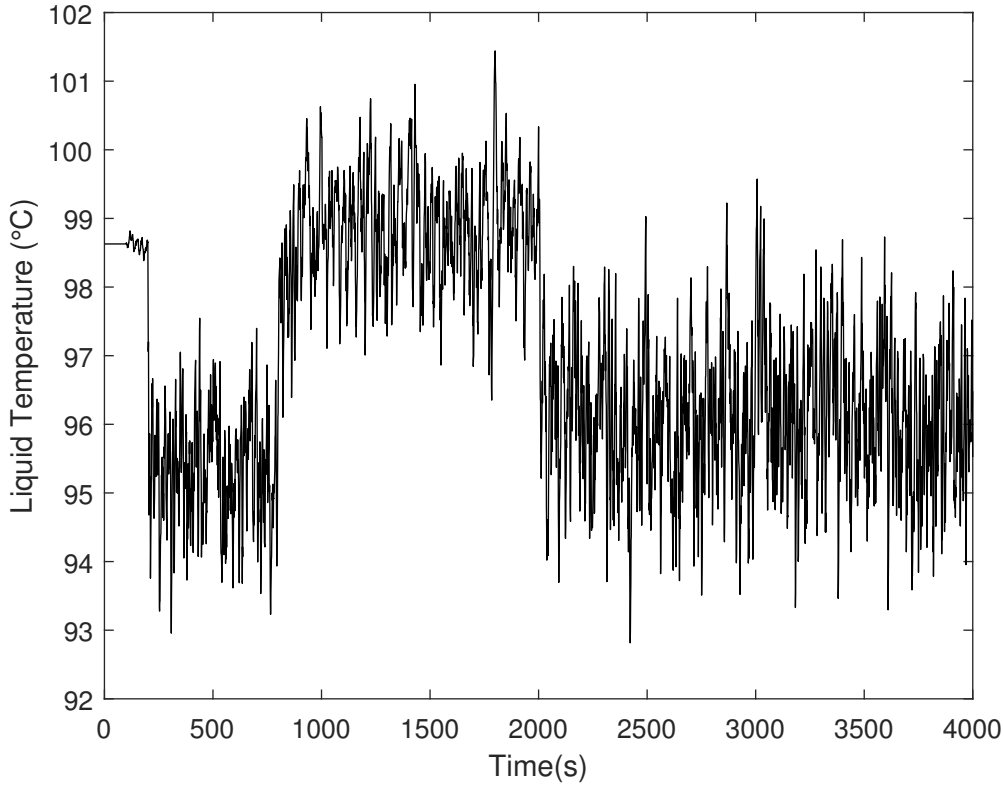
Os conjuntos de dados estudados aqui são comumente analisados na literatura por suas simplicidade e facilidade de representação, sendo um banco de dados industrial: Trocador de Calor, e um banco de dados mecânico: Atuador Hidráulico.

6.1.1 Trocador de Calor

O dados analisados aqui pertencem a um sistema de troca de calor a vapor de líquido saturado. No experimento, apresentado no trabalho de Bittanti e Piroddi (1996), o trocador de calor permite que água seja aquecida por meio de vapor saturado pressurizado transmitido através de um tubo de cobre. A variável de saída é a temperatura final do líquido, as variáveis de entrada são a taxa de vazão, a temperatura do vapor e a temperatura inicial do líquido. No experimento que gerou o banco de dados, as temperaturas do vapor e inicial do líquido foram mantidas constantes em seus valores nominais. Na Fig. 5 são apresentadas as séries temporais de entrada e saída dos dados a partir dos quais foram realizados os experimentos. Esse banco de dados é considerado referência para estudos de controle não-linear devido ao seu comportamento de fase não mínima, conforme mostrado nos gráficos da relação entrada-saída.



(a) (1)



(b) (2)

Figura 5 – Séries temporais com os valores de entrada (1) e saída (2) para o trocador de calor.

6.1.2 Atuador Hidráulico

Os dados aqui utilizados para identificação e modelagem se referem ao comportamento do atuador de movimento do braço mecânico de uma estrutura hidráulica - conhecida como Grua, composta de quatro atuadores no total: um para mover a estrutura, um para mover o braço, um para mover o antebraço e um para mover uma extensão telescópica do antebraço. O estudo tem como foco o atuador que movimenta o braço, visto que é a parte mais afetada pelo movimento oscilatório da estrutura. A Grua está apresentada (com ênfase no atuador do braço) na Figura 6.

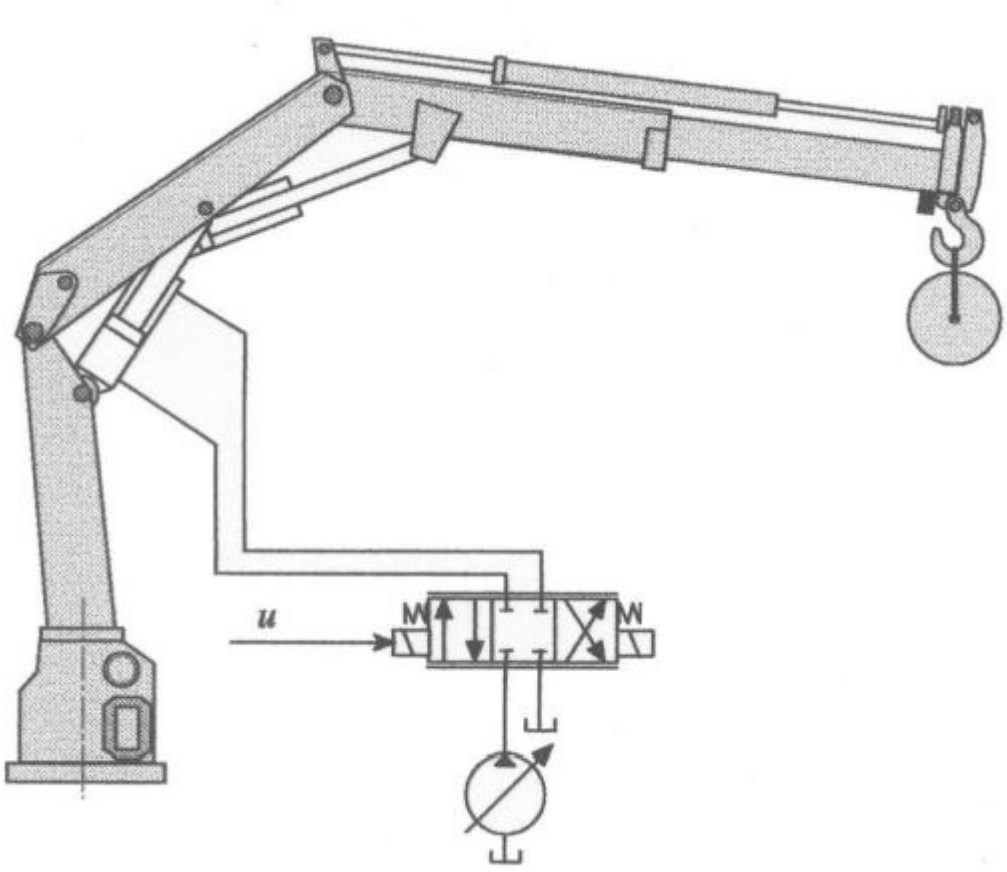
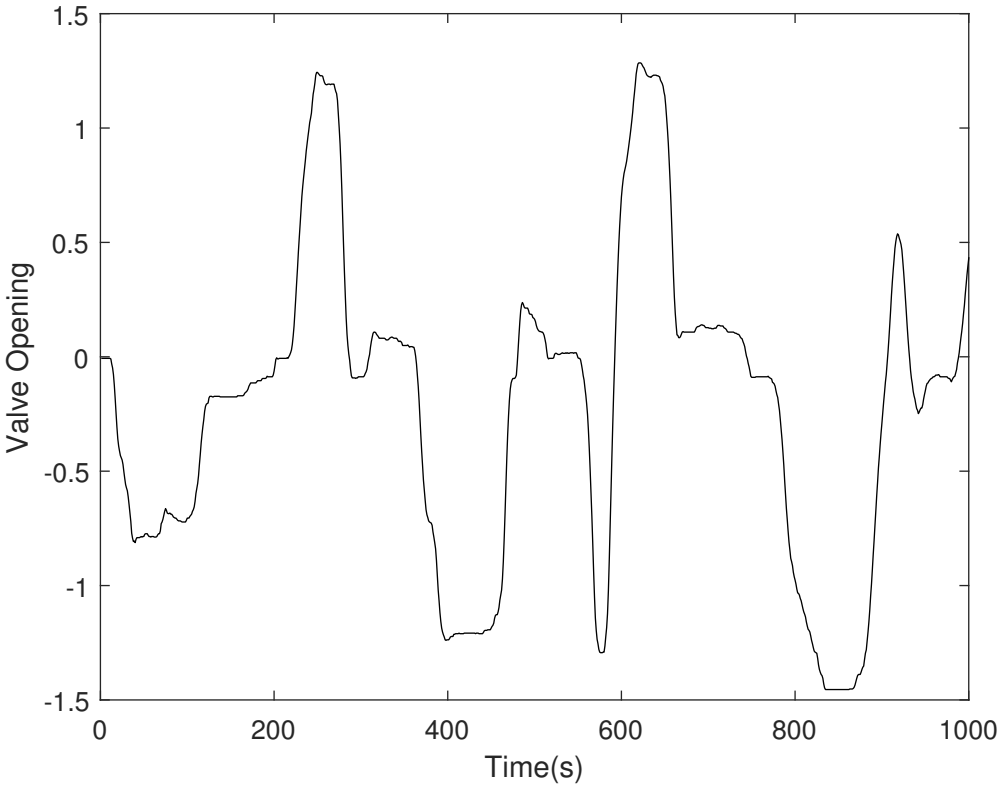
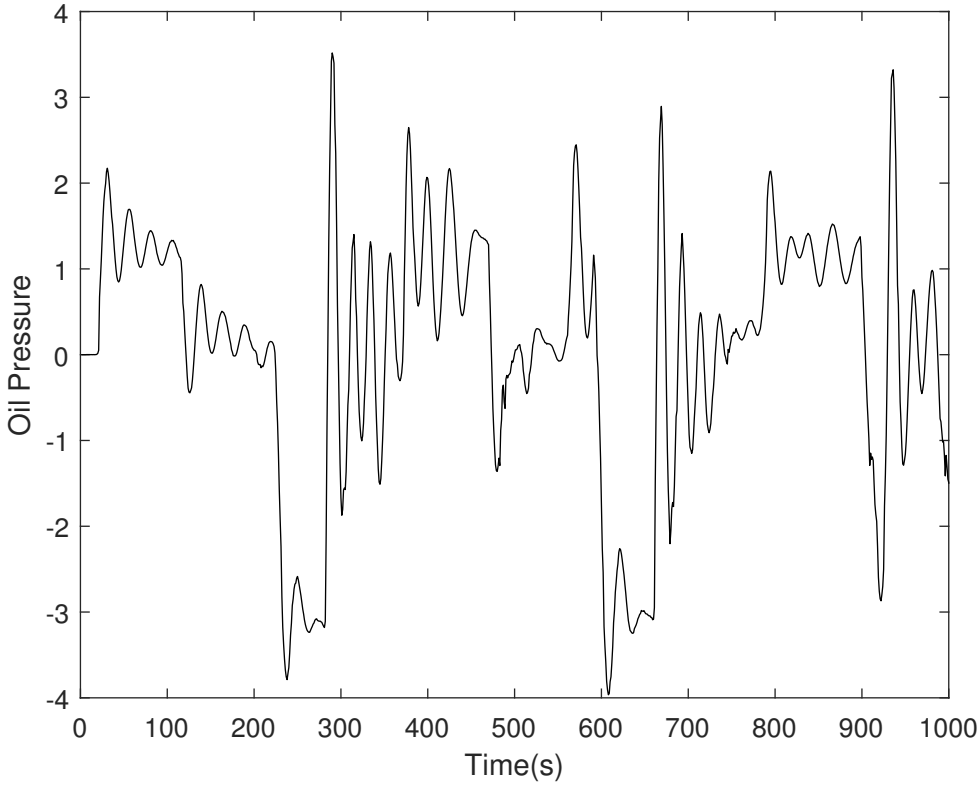


Figura 6 – Esquemático Atuador Hidráulico

Os dados analisados correspondem a posição da válvula (utilizada como série temporal de entrada $u(t)$) e pressão do óleo (utilizado como série temporal de saída $y(t)$) o qual apresenta comportamento fortemente oscilatório devido às ressonâncias mecânicas, um problema que dificulta a modelagem do sistema. Na Fig. 7 são apresentadas as séries temporais de entrada e saída dos dados a partir dos quais foram realizados os experimentos.



(a) (1)



(b) (2)

Figura 7 – Séries temporais com os valores de entrada (1) e saída (2) para o atuador hidráulico.

6.2 Condições de Treinamento e Avaliação do Erro

Os dados tomados do repositório Daisy foram primeiramente analisados sem a inserção de outliers, depois os dados foram poluídos com outliers de distribuição normal seguindo a proporção de *5%*, *10%* e *15%*, sempre aplicados à proporção de dados utilizados para teste. A separação de dados utilizados para treinamento, validação e teste seguiu a seguinte proporção: *50%*; *30%*; *20%* respectivamente.

Para a seleção dos hiperparâmetros, etapas de 100 rodadas de treinamento e validação foram realizadas, variando a quantidade de neurônios na camada escondida entre $Nh = 1, ..., 30$, com o número de neurônios vizinhos variando entre $K = 1, ..., Nh$ para cada quantidade de neurônios fixada por etapa. Dessa maneira, para cada nova quantidade de neurônios na camada escondida, Nh rodadas de treinamento e validação, e finalmente a quantiade total de $Nh * Nh$ etapas de treinamento e validação. Em seguida, com os melhores hiperparâmetros definidos, o estágio de treinamento e teste foi realizado, também consistindo de *100* rodadas, com o objetivo de obter os resíduos necessários para a etapa de análise.

A comparação entre os algoritmos de um mesmo conjunto de dados foi realizada através da análise do erro de generalização, a partir dos valores de Erro Quadrático Médio Normalizado (*Normalized Mean Square Error* NMSE):

$$\textbf{NMSE} = \frac{\sum_{t=1}^{N_2} e^2(t)}{N_2 \widehat{\sigma}_u^2}, \tag{6.1}$$

sendo N_2 a quantidade de amostras do conjunto de teste, $e(t)$ o erro de predição calculado como $e(t) = u(t) - \widehat{u}(t)$ e $\widehat{\sigma}_u^2$ a variância da série temporal de teste original.

Os modelos também serão avaliados a partir da utilização de diferentes algoritmos de quantização vetorial, sendo estes: **SOM, Fuzzy K-means, FSCL, K-means, PLSOM e WTA**. Com o objetivo de avaliar a efetividade da aplicação de modelos locais nos bancos de dados, a rede neural **ELM** foi utilizada a fim de gerar modelos globais robustos, desde que ela também foi treinada com a aplicação do algoritmo *IRLS* aqui proposto, a mesma metodologia de treinamento, validação e teste explanada previamente foi utilizada aqui para o algoritmo ELM robusto. Uma versão do ELM robusto similar a que está sendo utilizada neste artigo pode ser encontrada em Barreto e Barros (2016).

6.3 Análise dos Resíduos

A validação dos modelos lineares locais e globais obtidos é extremamente importante no processo de identificação de sistemas como os que foram gerados nesse trabalho, desse ponto em diante essa validação será feita a partir da análise dos resíduos (AGUIRRE, 2014). Através da aplicação de métodos estatísticos aos resíduos obtidos na etapa de teste é possível avaliar a capacidade de generalização alcançada pelos modelos, que representa quanta informação o modelo foi capaz de extrair dos dados e se ainda existe informação relevante que não foi absorvida pelos modelos neurais.

O processo de validação estatística aplicado aqui é baseado na utilização de funções de autocorrelação e correlação cruzada, que permitem avaliar se os resíduos obtidos na identificação durante a etapa de teste apresentam ruído gaussiano branco e variância σ^2, indicando que os dados não estão correlacionados com a entrada apresentada. Os métodos aplicados seguem as relações apresentadas nas equações abaixo (BILLINGS; VOON, 1983), (BILLINGS; VOON, 1986), (BILLINGS; ZHU, 1994):

$$\Phi_{ee}(\tau) = E\{e(t-\tau)e(t)\} = \delta(t), \tag{6.2}$$

$$\Phi_{ue}(\tau) = E\{u(t-\tau)e(t)\} = 0, \forall_\tau, \tag{6.3}$$

$$\Phi_{u^2e}(\tau) = E\{[u^2(t) - \bar{u^2}(t)]e(t-\tau)\} = 0, \forall_\tau, \tag{6.4}$$

$$\Phi_{u^2e^2}(\tau) = E\{[u^2(t) - \bar{u^2}(t)]e^2(t-\tau)\} = 0, \forall_\tau, \tag{6.5}$$

Os gráficos de autocorrelação e correlação cruzada linear e não-lineares serão apresentados para todos os algoritmos propostos, considerando a poluição de outliers em zero. Em seguida, os histogramas para os modelos serão apresentados considerando os modelos que tiveram dados poluídos com outliers em suas respectivas porcentagens.

7 EXPERIMENTOS E RESULTADOS

Para cada banco de dados apresentado, durante as etapas de treinamento, validação e teste, foi utilizado um modelo *NNARX* a fim de seguir o comportamento não-linear dos dados. As ordens de memória q e p utilizadas, apresentadas posteriormente para cada conjunto de dados estudado, basearam-se no trabalho realizado por Souza (2012), que validou esses valores para os mesmos bancos aqui modelados. Os valores de hiperparâmetros utilizados (sendo g a quantidade de neurônios, K o número de vetores protótipos associados aos neurônios, α a taxa de aprendizagem do algoritmo e σ o raio de vizinhança) parâmetros validados, assim como os resultados obtidos, serão apresentados nas seguintes seções, com os resultados armazenados por conjunto. Os experimentos e resultados foram obtidos utilizando a interface MATLAB.

7.1 Experimentos e Resultados: Trocador de Calor

O Trocador de Calor apresenta dados com valores altamente não-lineares, constituindo um desafio para algoritmos de modelagem. Os dados foram normalizados dentro dos limites [-1,+1] com ordens de memória: q = 3 and p = 6. Os hiperparâmetros escolhidos para a realização do processo de modelagem a partir dos dados apresentados, calculados durante o processo de validação como explicado anteriormente, estão listados na Tabela 10.

Tabela 10 – PARÂMETROS TROCADOR DE CALOR

CONJUNTO DE PARÂMETROS			
g	K	$(\alpha_0);(\alpha_T)$	$(\sigma_0);(\sigma_T)$
30	20	(0.5);(0.01)	(g/2);(0.001)

As Tabelas 11, 12 e 13 apresentam os valores de erro obtidos para os modelos RP-MKSOM, RD-MKSOM e ELM Robusto, respectivamente, considerando as aplicações dos algoritmos nos dados sem a aplicação de outliers e com a inserção de outliers nos dados de teste nas proporções de 5%, 10% e 15%.

Entre as técnicas aplicadas, o modelo global ELM demonstrou melhor aplicabilidade ao banco de dados aqui descrito, sempre mantendo os menores valores médios de NMSE com baixa variância, dessa maneira indicando um certo nível de robustez. Com a inserção de outliers nos dados, houve um pequeno aumento dos valores de erro, no entanto permaneceram abaixo dos valores apresentados por outros algoritmos.

Tabela 11 – Resultados NMSE RP-MKSOM Trocador de Calor

RP-MKSOM 0%				
VQ Algoritmo	**NMSE**			
	Min	*Max*	*Média*	*Variância*
SOM	**0.3814**	**0.8385**	**0.4867**	**0.0040**
FKM	1.1680	1.2650	1.2003	2.0982e-04
FSCL	1.4409	2.2722	1.6054	0.0201
KMEANS	0.4723	0.5407	0.5310	1.4607e-04
PLSOM	0.6469	2.1617	1.2474	0.1089
WTA	0.9452	2.0787	1.4219	0.0409
RP-MKSOM 5%				
VQ Algoritmo	**NMSE**			
	Min	*Max*	*Média*	*Variância*
SOM	1.9523	7.7119	3.8549	1.3995
FKM	**1.1721**	**1.2309**	**1.1990**	**1.4457e-04**
FSCL	1.4604	2.5584	1.7219	0.0368
KMEANS	9.7938	14.9527	14.2512	0.4390
PLSOM	5.9410	29.9508	14.4801	18.4978
WTA	2.2374	4.5127	2.8733	0.1442
RP-MKSOM 10 %				
VQ Algoritmo	**NMSE**			
	Min	*Max*	*Média*	*Variância*
SOM	3.4115	13.4462	7.8786	5.2672
FKM	**1.1824**	**1.2371**	**1.2029**	**1.2674e-04**
FSCL	1.4881	2.4343	1.7393	0.0233
KMEANS	15.6019	24.3887	23.1812	2.2126
PLSOM	10.5994	41.6388	24.2482	50.2341
WTA	2.5033	5.5983	3.9462	0.3575
RP-MKSOM 15 %				
VQ Algoritmo	**NMSE**			
	Min	*Max*	*Média*	*Variância*
SOM	4.4159	17.3181	10.2764	10.0511
FKM	**1.1874**	**1.2692**	**1.2056**	**1.6056e-04**
FSCL	1.7998	2.3591	2.0237	0.0130
KMEANS	18.7420	28.0860	26.9255	1.1680
PLSOM	10.9106	53.0454	31.8718	82.5023
WTA	2.9974	6.9831	5.0749	0.7929

Tabela 12 – Resultados NMSE RD-MKSOM Trocador de Calor

RD-MKSOM 0%				
VQ Algoritmo	**NMSE**			
	Min	*Max*	*Média*	*Variância*
SOM	**0.3215**	**0.3605**	**0.3389**	**5.6903e-05**
FKM	0.4061	0.4468	0.4299	7.3007e-05
FSCL	0.4040	0.7602	0.4763	0.0031
KMEANS	0.4301	0.4800	0.4565	1.6087e-04
PLSOM	0.4344	0.5201	0.4652	2.7482e-04
WTA	0.3915	0.6314	0.4652	0.0017
RD-MKSOM 5%				
VQ Algoritmo	**NMSE**			
	Min	*Max*	*Média*	*Variância*
SOM	5.0560	6.0800	5.6152	0.0355
FKM	5.5411	7.7570	6.4857	0.1705
FSCL	3.9849	13.0692	6.3773	3.1824
KMEANS	3.3604	6.0302	5.4684	0.1313
PLSOM	8.7381	11.0697	9.5172	0.2475
WTA	**3.7111**	**8.1042**	**4.8717**	**0.3080**
RD-MKSOM 10 %				
VQ Algoritmo	**NMSE**			
	Min	*Max*	*Média*	*Variância*
SOM	9.5910	10.9354	10.2391	0.0857
FKM	10.0679	13.2334	11.5087	0.3534
FSCL	**5.6107**	**18.7386**	**9.0966**	**4.7330**
KMEANS	9.7949	13.9109	12.2502	0.5247
PLSOM	14.2881	17.9067	15.6049	0.7544
WTA	6.9575	19.1629	10.9350	5.0795
RD-MKSOM 15 %				
VQ Algoritmo	**NMSE**			
	Min	*Max*	*Média*	*Variância*
SOM	**10.1412**	**11.8914**	**11.0952**	**0.1526**
FKM	11.8887	15.6436	13.9479	0.6129
FSCL	8.5378	21.9005	14.2621	9.6710
KMEANS	11.2809	14.2092	13.0235	0.4169
PLSOM	16.8356	20.9517	18.1897	0.8618
WTA	7.5651	18.5274	11.9492	5.8004

Tabela 13 – Resultados NMSE ELM Robusto Trocador de Calor

ELM Robusto 0%				**ELM Robusto 5%**			
NMSE				**NMSE**			
Min	*Max*	*Média*	*Variância*	*Min*	*Max*	*Média*	*Variância*
0.0031	1.2321	0.0907	0.0153	0.1544	0.1552	0.1548	4.6847e-08
ELM Robusto 10%				**ELM Robusto 15%**			
NMSE				**NMSE**			
Min	*Max*	*Média*	*Variância*	*Min*	*Max*	*Média*	*Variância*
0.1457	0.1544	0.1500	4.6847e-08	0.1450	0.1457	0.1453	4.7341e-08

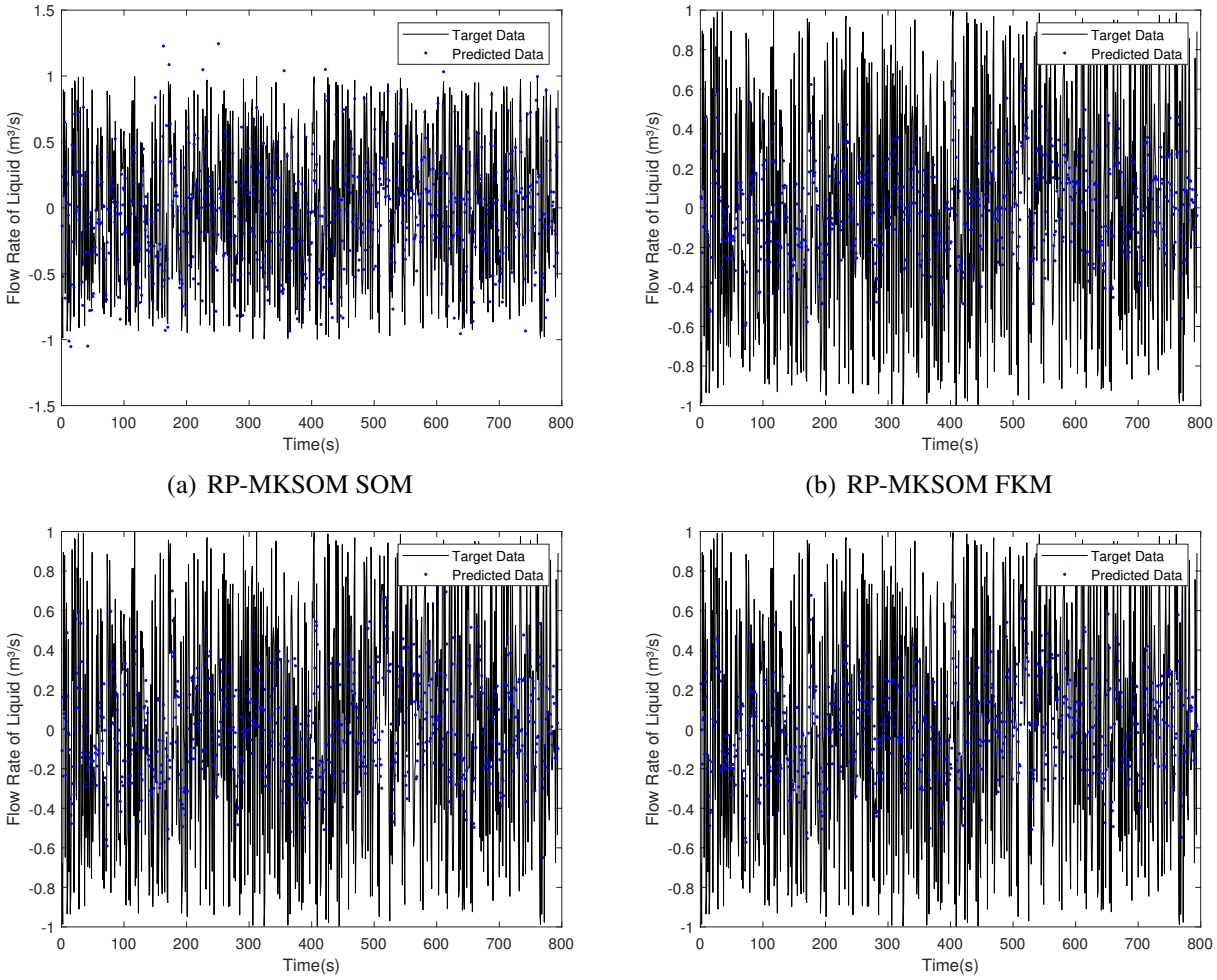
Entre os modelos locais utilizados, o modelo RD-MKSOM apresentou baixos valores para os dados sem outliers, enquanto para níveis maiores de inserção de outliers pode-se notar que ambos os modelos levam a maiores valores de erro, sem necessariamente obedecerem a um padrão. Desse modo, para dados sem outliers, é recomendado o uso do modelo RD-MKSOM com o algoritmo SOM para quantização vetorial, já para níveis maiores de outliers, é recomendada a utilização do modelo RP-MKSOM com o uso do algoritmo FKM.

Para propósitos de visualização das curvas de regressão para os modelos, temos na Figura 8 as curvas de regressão para os menores valores de NMSE obtidos durante os testes com os algoritmos para cada nível de inserção de outliers nos modelos RP-MKSOM.

Para os modelos RD-MKSOM nós apresentamos na Figura 9 as curvas de regressão para os menores valores de NMSE obtidos durante os testes com os algoritmos para cada nível de inserção de outliers.

O RP-MKSOM manteve a curva de regressão com valores abaixo dos dados de teste sendo conduzido pela mesma distribuição. Acredita-se que devido à natureza do algoritmo Fuzzy K-Médias, este algoritmo apresenta uma maior robustez e capacidade preditiva para os dados. Está claro que uma abordagem que possui uma visualização mais global desses dados pode apresentar menores valores quantitativos de erro.

Para acrescentar, durante os testes para conjunto de dados do Trocador de Calor, percebeu-se que os algoritmos de quantização que apresentaram um valor de erro inicialmente maior obtiveram uma menor progressão de erro durante a contaminação com outliers. Analisando os dados, está claro que a inserção de outliers teve um efeito altamente nocivo, aumentando o valor de erro dos algoritmos de forma quase exponencial em alguns casos. No entanto, para dados sem nenhuma intervenção, o nível pode ser considerado baixo e permaneceu sem grandes variações para os algoritmos de quantização utilizados.



(a) RP-MKSOM SOM

(b) RP-MKSOM FKM

(c) RP-MKSOM FKM

(d) RP-MKSOM FKM

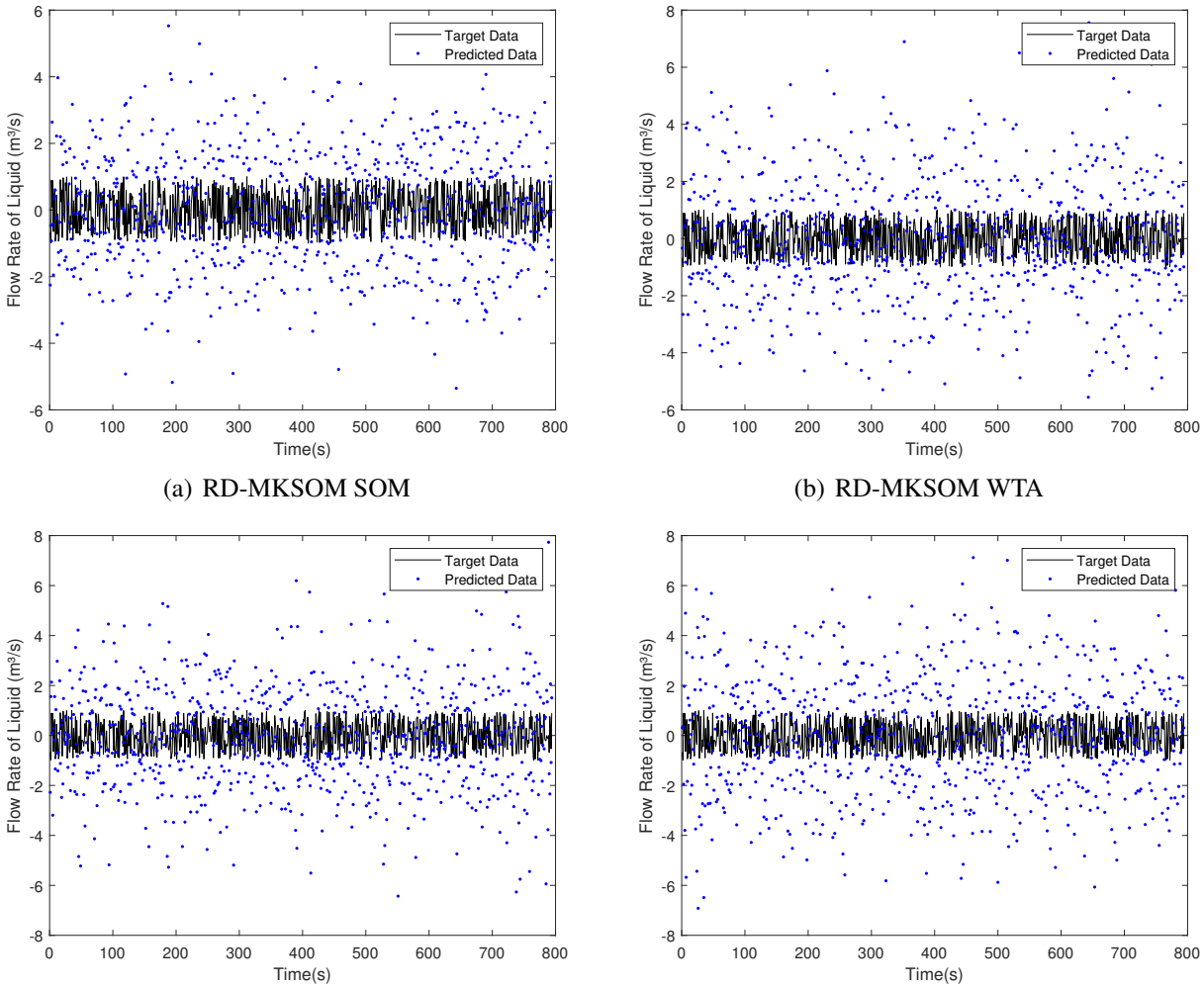
Figura 8 – Melhores curvas de regressão para cada nível de inserção de Outliers: (a) 0% com Algorimo SOM, (b) 5% com Algorimo FKM, (c) 10% com Algorimo FKM, (d) 15% com Algorimo FKM.

7.1.1 Análise dos Resíduos: Trocador de Calor

Para o trocador, os gráficos das funções para os algoritmos RD-MKSOM e RP-MKSOM são apresentados usando a rede neural SOM como algoritmo de quantização vetorial e para o algoritmo ELM Robusto, mostrados na Figura 10.

Como observado os modelos apresentam um nível similar de correlação entre os dados. Os modelos gerados apresentam grande parte de seus dados descorrelacionados, apresentando pouca informação da série de entrada como influência, o modelo ELM ainda possui uma maior quantidade de correlação, ainda que tivesse a capacidade de gerar baixos níveis de erro de predição mostrando que a capacidade de generalização do modelo está relacionada aos dados de entrada.

Para avaliar a progressão do erro conforme a quantidade de outliers nos dados



(a) RD-MKSOM SOM

(b) RD-MKSOM WTA

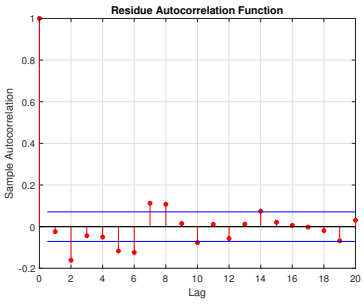
(c) RD-MKSOM FSCL

(d) RD-MKSOM SOM

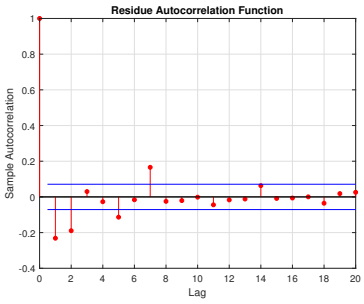
Figura 9 – Melhores curvas de regressão para cada nível de inserção de Outliers: (a) 0% com Algoritmo SOM, (b) 5% com Algoritmo WTA, (c) 10% com Algoritmo FSCL, (d) 15% com Algoritmo SOM.

aumenta, os histogramas gerados a partir dos resíduos serão apresentados, mostrando em quais modelos se comportaram conforme o ruído gaussiano branco, desse modo validando os gráficos das funções de autocorrelação e correlação cruzada apresentados.

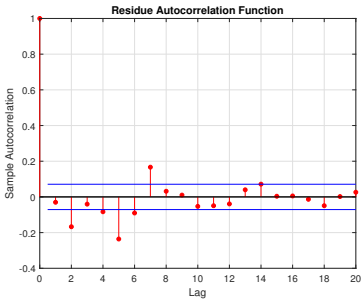
A partir da análise dos histogramas, é possível perceber o que foi dito acima, os resíduos apresentados para o ELM Robusto não apresentam um comportamento gaussiano uniforme, mostrando ainda uma certa quantidade de dados correlacionados entre a entrada e a saída. Por outro lado os modelos robustos RP-MKSOM e RD-MKSOM apresentam uma clara distribuição normal para as amostras, podendo ser utilizada aquela que apresenta os menores valores de erro de predição.



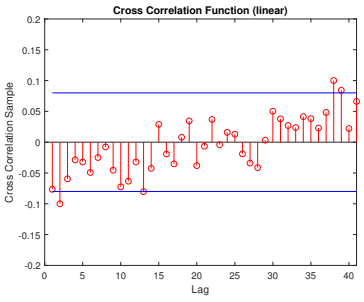
(a) $\Phi_{ee}(\tau)$ - Robust ELM



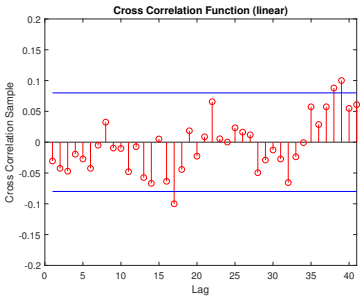
(b) $\Phi_{ee}(\tau)$- RD-MKSOM



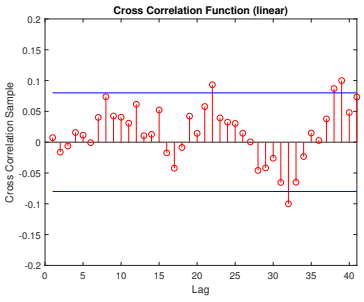
(c) $\Phi_{ee}(\tau)$ - RP-MKSOM



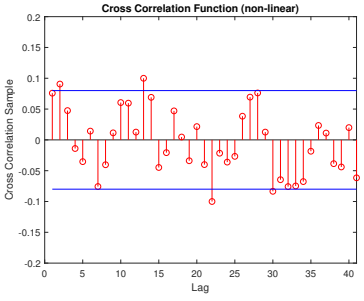
(d) $\Phi_{ue}(\tau)$ - Robust ELM



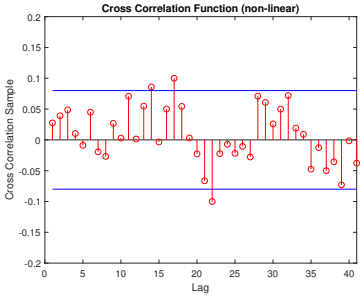
(e) $\Phi_{ue}(\tau)$ - RD-MKSOM



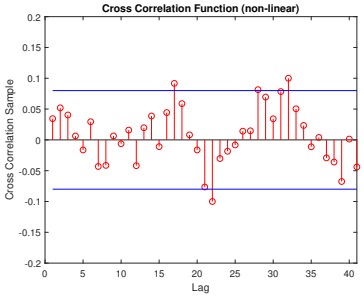
(f) $\Phi_{ue}(\tau)$ - RP-MKSOM



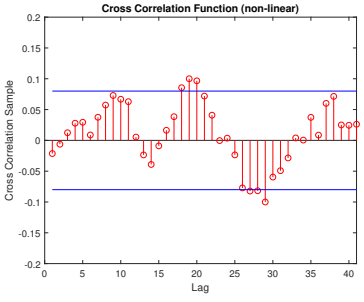
(g) $\Phi_{u^2e}(\tau)$ - Robust ELM



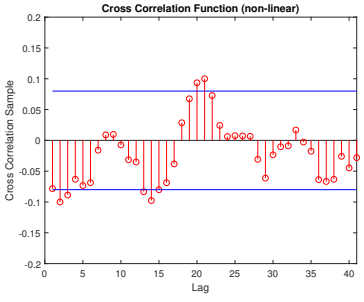
(h) $\Phi_{u^2e}(\tau)$ - RD-MKSOM



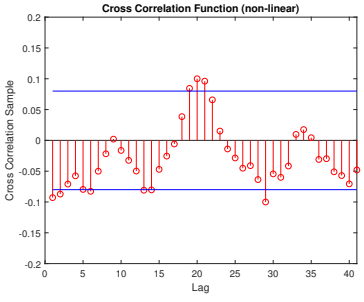
(i) $\Phi_{u^2e}(\tau)$ - RP-MKSOM



(j) $\Phi_{u^2e^2}(\tau)$ - Robust ELM

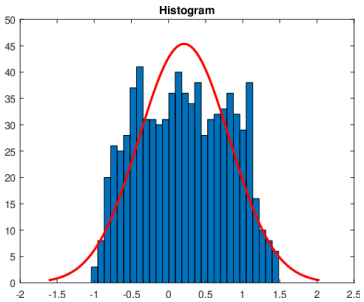


(k) $\Phi_{u^2e^2}(\tau)$ - RD-MKSOM

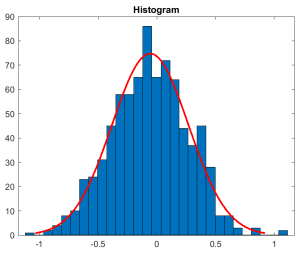


(l) $\Phi_{u^2e^2}(\tau)$ - RP-MKSOM

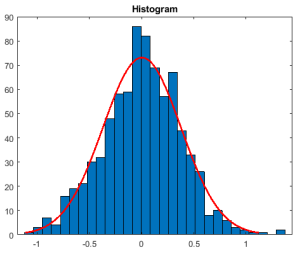
Figura 10 – Funções de Autocorrelação e Correlação Cruzada dos Algoritmos ELM Robusto (a, d, g, j), RD-MKSOM (b, e, h, k) e RP-MKSOM (c, f, i, l) para o conjunto de dados do Trocador de Calor.



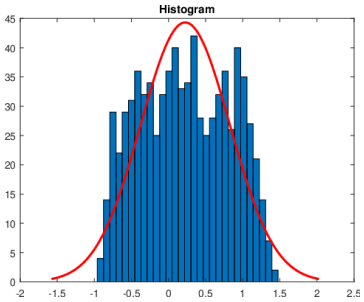
(a) Robust ELM - 0%



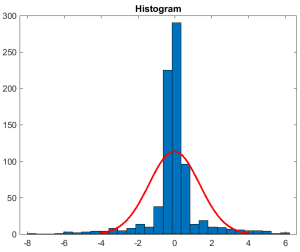
(b) RD-MKSOM - 0%



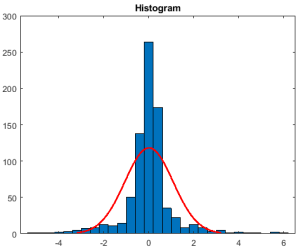
(c) RP-MKSOM - 0%



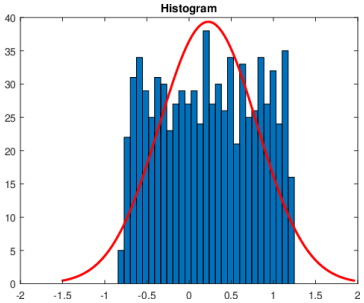
(d) Robust ELM - 5%



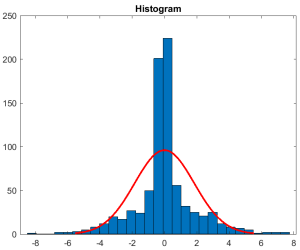
(e) RD-MKSOM - 5%



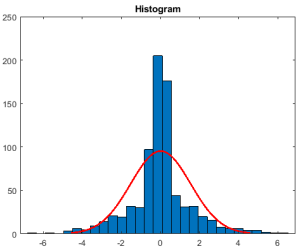
(f) RP-MKSOM - 5%



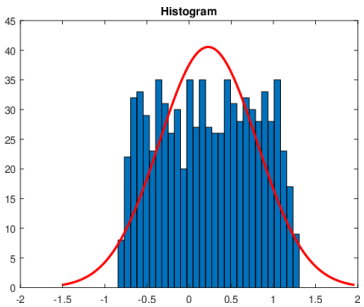
(g) Robust ELM - 10%



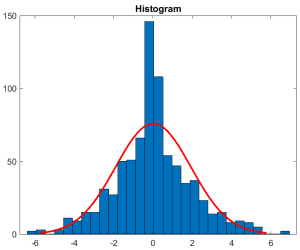
(h) RD-MKSOM - 10%



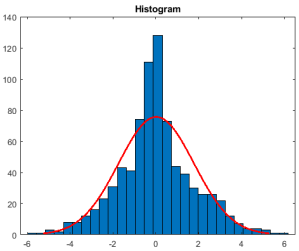
(i) RP-MKSOM - 10%



(j) Robust ELM - 15%



(k) RD-MKSOM - 15%



(l) RP-MKSOM - 15%

Figura 11 – Histogramas dos Algoritmos ELM Robusto (a, d, g, j), RD-MKSOM (b, e, h, k) e RP-MKSOM (c, f, i, l) para o conjunto de dados do Trocador de Calor em diferenes níveis de contaminação de outliers.

7.2 Experimentos e Resultados: Atuador Hidráulico

Como mencionado anteriormente, os dados aqui utilizados se referem ao controle do atuador do braço, que sofre com o comportamento oscilatório da Grua. Os dados foram normalizados para os limites [-1,+1] com ordens de memória: q = 4 e p = 5. Os hiperparâmetros escolhidos para realizar a modelagem a partir dos dados apresentados estão listados na Tabela 14.

Tabela 14 – PARÂMETROS ATUADOR HIDRÁULICO

CONJUNTO DE PARÂMETROSPARAMETERS SET			
g	K	$(\alpha_0);(\alpha_T)$	$(\sigma_0);(\sigma_T)$
20	15	(0.5);(0.01)	(g/2);(0.001)

As Tabelas 15, 16 e 17 apresentam os valores de erro obtidos para os modelos RP-MKSOM, RD-MKSOM e ELM Robusto, respectivamente, considerando as aplicações dos algoritmos nos dados sem a aplicação de outliers e com a inserção de outliers nos dados de teste nas proporções de 5%, 10% e 15%.

Como pode ser observado em todos os algoritmos, baixos valores de erro foram encontrados, mostrando a capacidade dos modelos de lidar com os dados apresentados, mesmo com altos valores de contaminação. O modelo apresentou menores valores de erro, sendo recomendado para a predição de dados dessa natureza, em todos os modelos o algoritmo SOM obteve maior robustez e acurácia na realização do processo de identificação.

O modelo ELM robusto também mostrou inicialmente uma boa capacidade de modelagem do banco de dados, a frente inclusive do modelo RP-MKSOM, no entanto, com a inserção de outliers nos dados, o primeiro começou a mostrar valores de erro maiores que o segundo, com variâncias mais amplas. Na aplicação dos métodos, pode-se perceber que a utilização da rede neural SOM como algoritmo de quantização vetorial é satisfatória se comparada aos outros algoritmos aplicados.

Para propósitos de visualização das curvas de regressão para os modelos, temos na Figura 12 as curvas de regressão para os menores valores de NMSE obtidos durante os testes com os algoritmos para cada nível de inserção de outliers nos modelos RP-MKSOM.

Para os modelos RD-MKSOM nós apresentamos na Figura 13 as curvas de regressão para os menores valores de NMSE obtidos durante os testes com os algoritmos para cada nível

Tabela 15 – Resultados NMSE RP-MKSOM Atuador Hidráulico

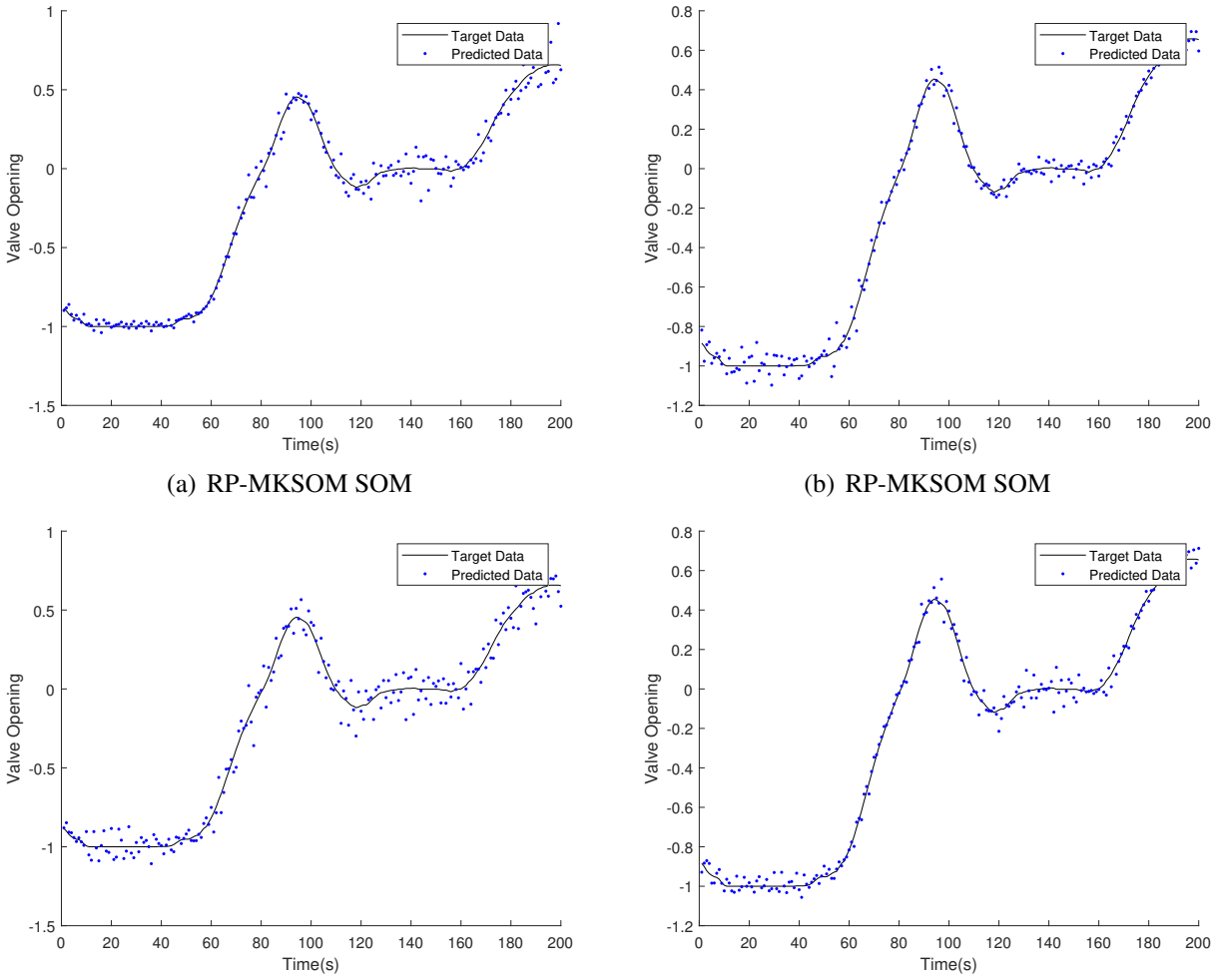
RP-MKSOM 0%				
VQ Algoritmo	**NMSE**			
	Min	*Max*	*Média*	*Variância*
SOM	**1.6801e-04**	**0.0011**	**3.6638e-04**	**2.2562e-08**
FKM	0.0087	0.0772	0.0200	1.3449e-04
FSCL	0.0029	0.0250	0.0111	2.0660e-05
KMEANS	2.3263e-04	7.2384e-03	3.8473e-04	9.9682e-09
PLSOM	1.5830e-04	7.4697e-04	3.7110e-04	2.3191e-08
WTA	0.0027	0.0237	0.0115	1.5048e-05
RP-MKSOM 5%				
VQ Algoritmo	**NMSE**			
	Min	*Max*	*Média*	*Variância*
SOM	**3.6237e-04**	**0.1343**	**0.0098**	**2.0345e-04**
FKM	0.0105	0.0657	0.0227	1.4004e-04
FSCL	0.0056	0.0506	0.0212	8.1944e-05
KMEANS	6.4624e-04	0.0920	0.0152	1.6006e-04
PLSOM	0.0014	0.0446	0.0117	1.0435e-04
WTA	0.0109	0.0509	0.0262	6.4921e-05
RP-MKSOM 10 %				
VQ Algoritmo	**NMSE**			
	Min	*Max*	*Média*	*Variância*
SOM	**8.4487e-04**	**0.0493**	**0.0110**	**7.3288e-05**
FKM	0.0109	0.0797	0.0231	1.1498e-04
FSCL	0.0077	0.0602	0.0259	1.0714e-04
KMEANS	8.6048e-04	0.1323	0.0245	4.4586e-04
PLSOM	0.0030	0.1327	0.0366	0.0011
WTA	0.0118	0.0573	0.0288	8.2842e-05
RP-MKSOM 15 %				
VQ Algoritmo	**NMSE**			
	Min	*Max*	*Média*	*Variância*
SOM	**0.0013**	**0.3333**	**0.0183**	**0.0011**
FKM	0.0113	0.1355	0.0271	3.3514e-04
FSCL	0.0181	0.1049	0.0494	2.7948e-04
KMEANS	0.0025	0.1122	0.0328	4.2155e-04
PLSOM	0.0026	0.1016	0.0346	7.4223e-04
WTA	0.0165	0.0825	0.0390	2.0037e-04

Tabela 16 – Resultados NMSE RD-MKSOM Atuador Hidráulico

RD-MKSOM 0%				
VQ Algoritmo	**NMSE**			
	Min	*Max*	*Média*	*Variância*
SOM	**8.4176e-05**	**9.2106e-05**	**8.8914e-05**	**3.5418e-12**
FKM	1.0851e-04	6.7287e-04	1.2187e-04	4.0543e-09
FSCL	1.1157e-04	7.4850e-04	2.2707e-04	3.7114e-08
KMEANS	1.1242e-04	5.8756e-04	2.4013e-04	2.2549e-08
PLSOM	1.1051e-04	1.1552e-04	1.1292e-04	1.7748e-12
WTA	1.0925e-04	7.3774e-04	2.1174e-04	3.5181e-08
RD-MKSOM 5%				
VQ Algoritmo	**NMSE**			
	Min	*Max*	*Média*	*Variância*
SOM	**1.1134e-04**	**1.6047e-04**	**1.3142e-04**	**1.1850e-10**
FKM	2.4961e-04	0.0847	0.0044	1.3912e-04
FSCL	1.6134e-04	0.1149	0.0131	6.5284e-04
KMEANS	2.1883e-04	0.1663	0.0535	0.0018
PLSOM	1.7394e-04	3.9970e-04	2.7804e-04	1.5516e-09
WTA	1.4196e-04	0.0634	0.0084	1.8453e-04
RD-MKSOM 10 %				
VQ Algoritmo	**NMSE**			
	Min	*Max*	*Média*	*Variância*
SOM	**1.6262e-04**	**3.4409e-04**	**2.0524e-04**	**8.0059e-10**
FKM	4.5381e-04	0.0206	0.0015	6.5713e-06
FSCL	1.8124e-04	0.0977	0.0134	6.0819e-04
KMEANS	1.7038e-04	0.1247	0.0412	0.0013
PLSOM	3.1250e-04	6.5699e-04	4.7117e-04	4.3161e-09
WTA	1.6543e-04	0.0854	0.0168	4.4712e-04
RD-MKSOM 15 %				
VQ Algoritmo	**NMSE**			
	Min	*Max*	*Média*	*Variância*
SOM	**2.2456e-04**	**0.0019**	**4.7596e-04**	**6.8088e-08**
FKM	9.0895e-04	0.0365	0.0029	2.5480e-05
FSCL	1.9262e-04	0.1394	0.0189	0.0013
KMEANS	2.4918e-04	0.2380	0.0923	0.0050
PLSOM	3.7490e-04	9.3815e-04	7.0868e-04	1.6435e-08
WTA	1.7666e-04	0.1283	0.0255	0.0012

Tabela 17 – Resultados NMSE ELM Robusto Atuador Hidráulico

ELM Robusto 0%				**ELM Robusto 5%**			
NMSE				**NMSE**			
Min	*Max*	*Média*	*Variância*	*Min*	*Max*	*Média*	*Variância*
1.4803e-05	0.0015	3.8850e-04	5.6899e-08	8.6245e-04	0.0130	0.0069	1.2591e-05
ELM Robusto 10%				**ELM Robusto 15%**			
NMSE				**NMSE**			
Min	*Max*	*Média*	*Variância*	*Min*	*Max*	*Média*	*Variância*
0.0135	0.0673	0.0404	2.4846e-04	0.0389	0.0670	0.0529	6.8086e-05



(a) RP-MKSOM SOM

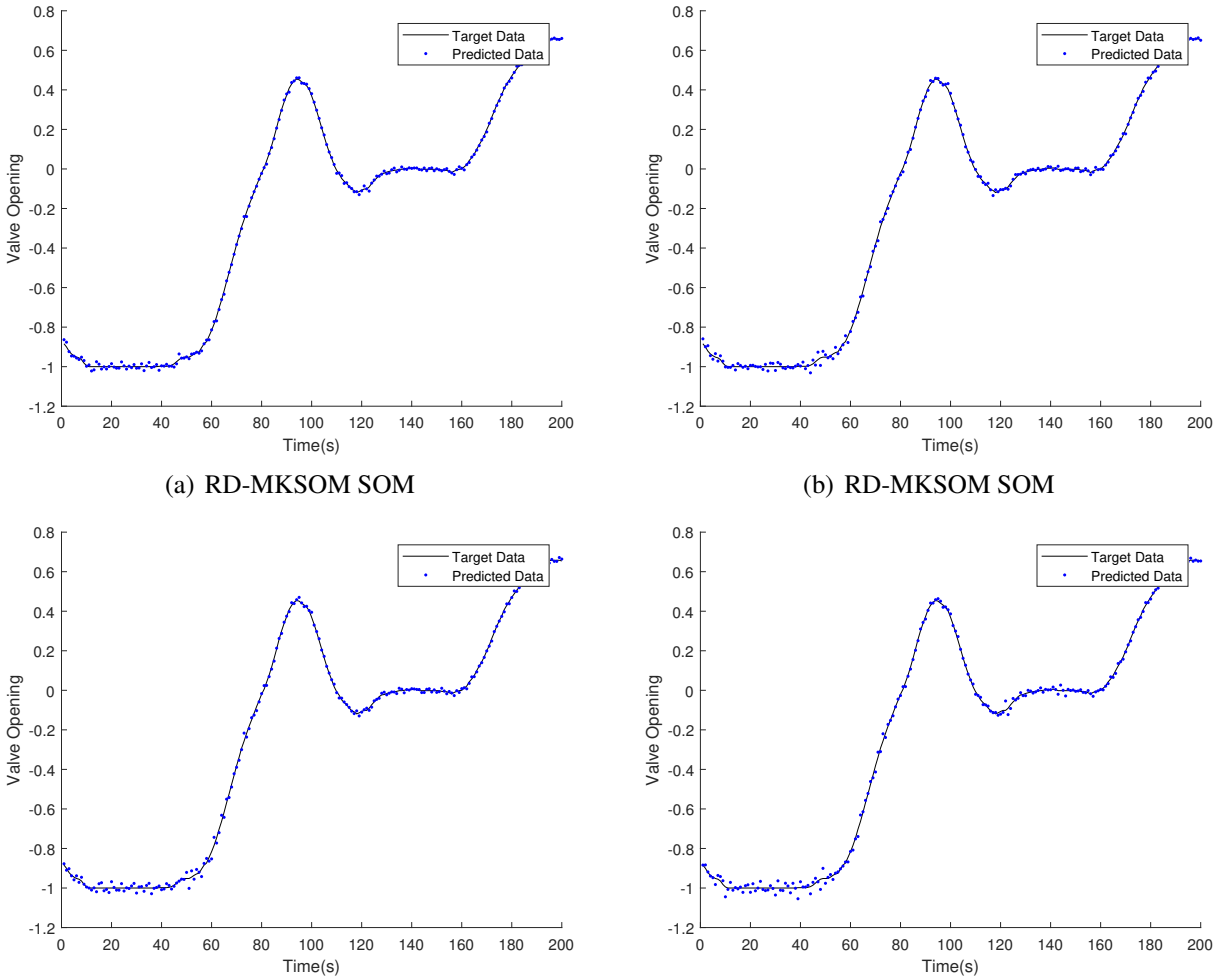
(b) RP-MKSOM SOM

(c) RP-MKSOM SOM

(d) RP-MKSOM SOM

Figura 12 – Melhores curvas de regressão para cada nível de inserção de Outliers: (a) 0% com Algoritmo SOM, (b) 5% com Algoritmo SOM, (c) 10% com Algoritmo SOM, (d) 15% com Algoritmo SOM.

de inserção de outliers.



(a) RD-MKSOM SOM

(b) RD-MKSOM SOM

(c) RD-MKSOM SOM

(d) RD-MKSOM SOM

Figura 13 – Melhores curvas de regressão para cada nível de inserção de Outliers: (a) 0% com Algoritmo SOM, (b) 5% com Algoritmo SOM, (c) 10% com Algoritmo SOM, (d) 15% com Algoritmo SOM.

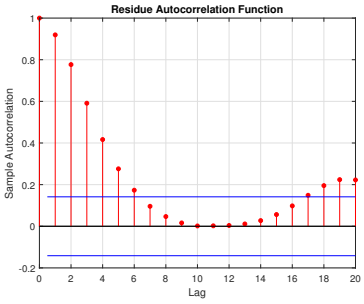
Esse conjunto de dados é mais simples do que o conjunto apresentado anteriormente. Pode ser observado a partir das curvas de regressão que todos os modelos acompanham muito bem os dados de teste, tanto com ou sem contaminação. Porém os modelos RD-MKSOM são claramente menos sensitivos à contaminação do que os demais, mostrando que para esse tipo de dados é recomendada a aplicação de uma modelagem mais local, pelo menos de acordo com os resultados de NMSE. A seguir será apresentada a validação dos resultados a partir da análise dos resíduos.

7.2.1 Análise do Resíduos: Atuador Hidráulico

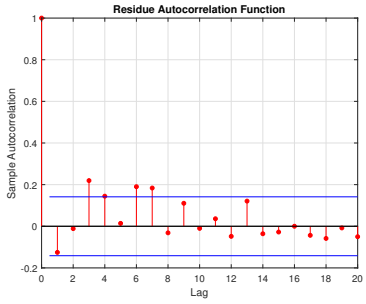
Para o atuador, os gráficos das funções para os algoritmos RD-MKSOM e RP-MKSOM são apresentados usando a rede neural SOM como algoritmo de quantização vetorial e para o algoritmo ELM Robusto, mostrados na Figura 14.

As impressões dos gráficos de autocorrelação e correlação cruzada mostram que os modelos ELM Robusto e RP-MKSOM não satisfazem as condições desejadas de independência entre os dados, ainda que apresentem baixos valores de erro. De acordo com o que pode ser analisado, o modelo RD-MKSOM é aquele que melhor consegue generalizar os dados e possui os menores valores de erro, com os gráficos de autocorrelação e correlação cruzada (linear e não-linear) mostrando que os testes apresentam dados descorrelacionados, assim a etapa de teste adquiriu efetivamente as informações do modelo treinado. Ainda, da análise de histogramas, o comportamento gaussiano e a melhor utilização para os dados do atuador podem ser certificados. Os histogramas mostrando o comportamento dos resíduos são apresentados na Figura 15.

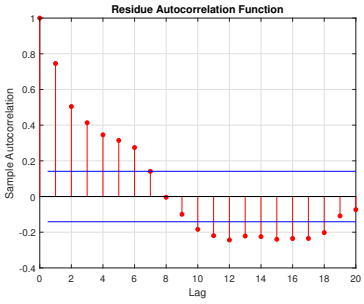
Observando os histogramas, pode-se notar que o algoritmo RD-MKSOM apresenta um comportamento residual mais uniforme para o banco de dados, desse modo pode ser seguramente utilizado para a identificação do modelo. É interessante notar que o aspecto gaussiano permanece até mesmo com o aumento na quantidade de outliers nos dados, mostrando a robustez do algoritmo através da técnica que foi proposta.



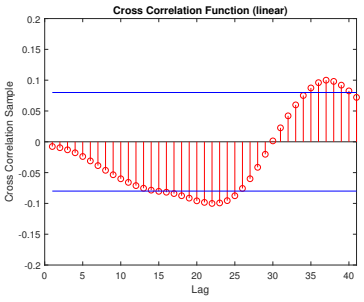
(a) $\Phi_{ee}(\tau)$ - Robust ELM



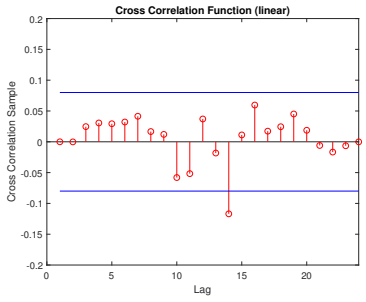
(b) $\Phi_{ee}(\tau)$- RD-MKSOM



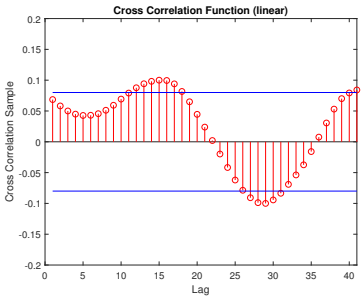
(c) $\Phi_{ee}(\tau)$ - RP-MKSOM



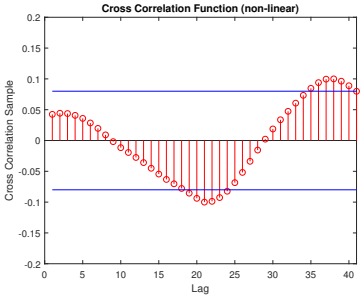
(d) $\Phi_{ue}(\tau)$ - Robust ELM



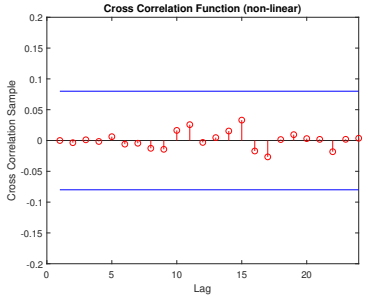
(e) $\Phi_{ue}(\tau)$ - RD-MKSOM



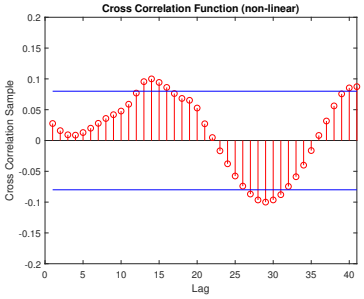
(f) $\Phi_{ue}(\tau)$ - RP-MKSOM



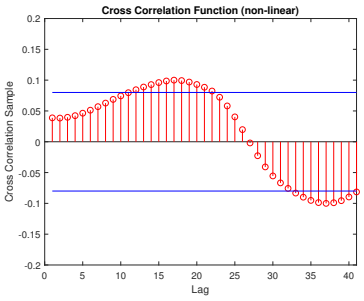
(g) $\Phi_{u^2e}(\tau)$ - Robust ELM



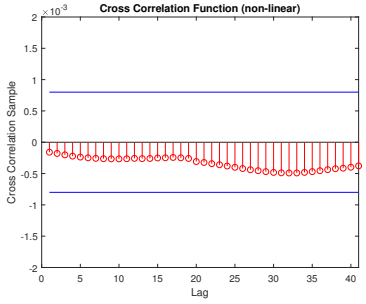
(h) $\Phi_{u^2e}(\tau)$ - RD-MKSOM



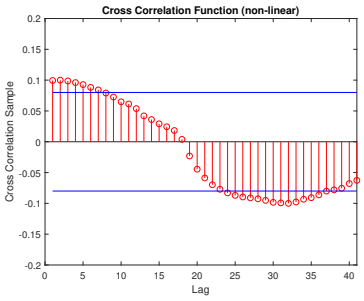
(i) $\Phi_{u^2e}(\tau)$ - RP-MKSOM



(j) $\Phi_{u^2e^2}(\tau)$ - Robust ELM

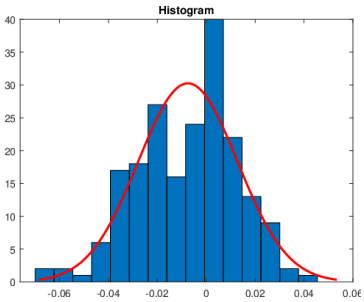


(k) $\Phi_{u^2e^2}(\tau)$ - RD-MKSOM

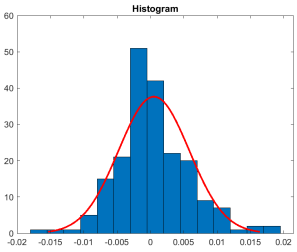


(l) $\Phi_{u^2e^2}(\tau)$ - RP-MKSOM

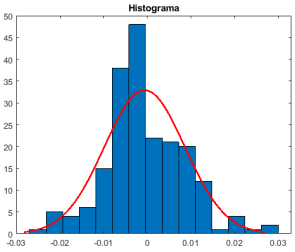
Figura 14 – Funções de Autocorrelação e Correlação Cruzada dos Algoritmos ELM Robusto (a, d, g, j), RD-MKSOM (b, e, h, k) e RP-MKSOM (c, f, i, l) para o conjunto de dados do Atuador Hidráulico.



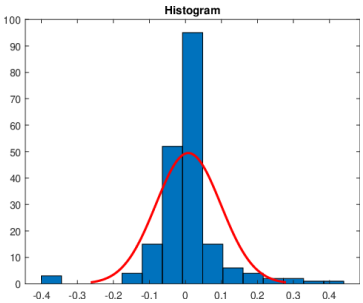
(a) Robust ELM - 0%



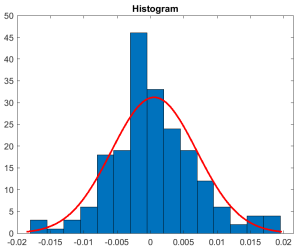
(b) RD-MKSOM - 0%



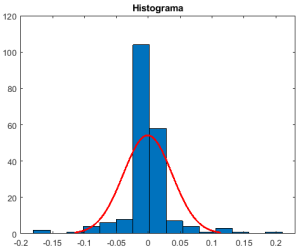
(c) RP-MKSOM - 0%



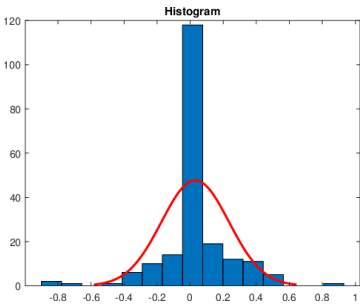
(d) Robust ELM - 5%



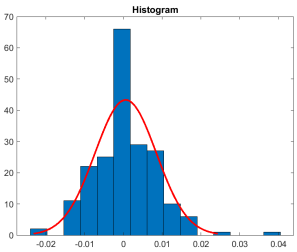
(e) RD-MKSOM - 5%



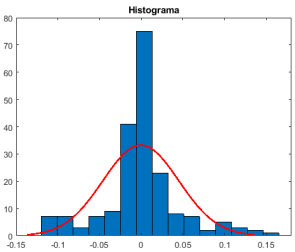
(f) RP-MKSOM - 5%



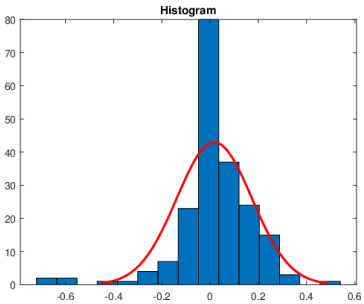
(g) Robust ELM - 10%



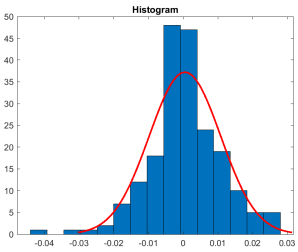
(h) RD-MKSOM - 10%



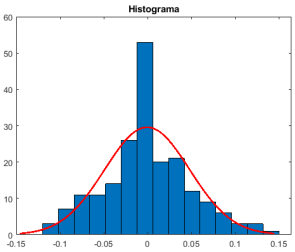
(i) RP-MKSOM - 10%



(j) Robust ELM - 15%



(k) RD-MKSOM - 15%



(l) RP-MKSOM - 15%

Figura 15 – Histogramas dos Algoritmos ELM Robusto (a, d, g, j), RD-MKSOM (b, e, h, k) e RP-MKSOM (c, f, i, l) para o conjunto de dados do Atuador Hidráulico em diferenes níveis de contaminação de outliers.

8 CONCLUSÃO

Nesse trabalho, a aplicação de técnicas de tratamento de outliers com o objetivo de obter uma melhor representação dos dados estudados provou ser satisfatória, especialmente quando os resultados podem ser comparados a resultados de trabalhos correlacionados, com um ganho na acurácia dos modelos apresentados em relação aos aplicados anteriormente. Isso se deve especialmente a ponderamento iterativo aplicado ao erro, de modo que os dados de erros mais relevantes também se tornam mais relevantes na fase de treinamento do algoritmo.

É interessante notar que, em alguns casos, os modelos P-MKSOM Robusto e D-MKSOM Robusto apresentaram, para altos níveis de inserção de outliers, comportamentos similares (ou comportamentos que acompanharam a progressão dos dados) aos modelos que foram apresentados para dados sem contaminação, isto pode ser observado através das figuras das curvas de regressão dos conjuntos de dados. Esses resultados mostram a pequena influência do erro gerado a partir dos dados outliers no comportamento do modelo, mostrando a efetividade da aplicação dos pesos da técnica de estimação M.

Foi percebido que o banco de dados do Atuador Hidráulico pôde ser melhor modelado utilizando técnicas locais e o banco de dados do Trocador de Calor apresentou melhor modelagem através do modelo global mostrado anteriormente. Isso pode ser devido ao fato de que o Atuador Hidráulico apresenta regiões mais bem definidas (ou locais) que o banco de dados do trocador.

A análise dos resíduos, recurso implementado para avaliação dos modelos que são capazes de realizar boa generalização através de comportamentos gaussianos, mostrou que os algoritmos robustos puderam performar eficientemente o mapeamento de entrada-saída a partir do aprendizado das informações apresentadas, ainda que na presença de outliers. Então, para os conjuntos de dados testados nesse trabalho nós recomendamos a utilização dos modelos locais apresentados aqui ao invés dos modelos globais, considerando os baixos valores de Erro Quadrático Médio Normalizado (NMSE) e dos níveis de generalização percebidos utilizando os testes de autocorrelação e de correlação cruzada, embasados pela apresentação dos histogramas.

Sugere-se, para trabalhos futuros, a aplicação dos modelos para sistemas MIMO, desde que se comportaram satisfatoriamente para um sistema de múltiplas entradas. Outro trabalho que pode ser realizado é a aplicação desses modelos associados com modelos globais, assim gerando uma extensão da técnica de modelos regionais. Também consideramos vantajoso a aplicação de outras técnicas de estimação robusta com o objetivo de incrementar a identificação e tratamento de outliers nos dados.

REFERÊNCIAS

AGUIRRE, L. A. **Introdução à Identificação de Sistemas: Técnicas Lineares e Não Lineares: Teoria e Aplicação**. Minas Gerais: Editora UFMG, 2014.

ANDRÁSIK, A.; MÉSZÁROS, A.; AZEVEDO, S. de. On-line tuning of a neural PID controller based on plant hybrid modeling. **Computers and Chemical Engineering**, v. 28, n. 8, p. 1499–1509, 2004.

BABUŜKA, R.; VERBRUGGEN, H. Neuro-fuzzy methods for nonlinear system identification. **Annual Reviews in Control**, v. 27, p. 73–85, 2003.

BARRETO, G. A.; ARAÚJO, A. F. R. Identification and control of dynamical systems using the self-organizing map. **IEEE Transactions on Neural Networks**, v. 15, p. 1244–1259, 2004.

BARRETO, G. A.; ARAÚJO, A. F. R.; RITTER, H. J. Self-organizing feature maps for modeling and control of robotic manipulators. **Journal of Intelligent and Robotic Systems**, v. 36, n. 4, p. 407–450, 2003.

BARRETO, G. A.; BARROS, A. L. B. A robust extreme learning machine for pattern classification with outliers. **Neurocomputing**, v. 176, p. 3–13, 2016.

BILLINGS, S. A.; CHEN, S. Neural netwoks for systems and control: Neural networks and system identification. In: ____. [S.l.]: Peter Peregrinus, London, 1992. cap. 9, p. 181–205.

BILLINGS, S. A.; VOON, W. S. F. Structure detection and model validity tests in the identification of nonlinear systems. **IEE Proceedings, Part D**, v. 130, n. 4, p. 193–199, 1983.

BILLINGS, S. A.; VOON, W. S. F. Correlation based model validity tests for nonlinear models. **International Journal of Control**, v. 44, n. 1, p. 235–244, 1986.

BILLINGS, S. A.; ZHU, Q. M. Nonlinear model validation using correlation tests. **International Journal of Control**, v. 60, n. 6, p. 1107–1120, 1994.

BITTANTI, S.; PIRODDI, L. Nonlinear identification and control of a heat exchanger: a neural network approach. **Journal of the Franklin Institute**, 1996.

CHO, J. *et al.* Modeling and inverse controller design for an unmanned aerial vehicle based on the self-organizing map. **IEEE Transactions on Neural Networks**, v. 17, n. 2, p. 445–460, 2006.

CHO, J. *et al.* Quasi-sliding mode control strategy based on multiple linear models. **Neurocomputing**, v. 70, n. 4-6, p. 962–974, 2007.

DAOSUD, W. *et al.* Neural network inverse model-based controller for the control of a steel pickling process. **Computers and Chemical Engineering**, v. 29, n. 10, p. 2110–2119, 2005.

FEI, J. *et al.* Fuzzy multiple hidden layer recurrent neural control of nonlinear system using terminal sliding-mode controller. **IEEE Transactions on Cybernetics**, v. 52, n. 9, p. 9519–9534, 2022.

FOX, J.; WEISBERG, S. **An R Companion to Applied Regression**. 2nd. ed. Thousand Oaks: Sage Publication, 2010.

GÖTTSCHE, T. H.; HUNT, K. J.; JOHANSEN, T. A. Nonlinear dynamics modelling via operating regime decomposition. **Mathematics and Computers in Simulation**, v. 46, p. 543–550, 1998.

HONGWEI, W.; PENGLONG, F. Fuzzy modeling of multirate sampled nonlinear systems based on multi-model method. **Journal of Systems Engineering and Electronics**, v. 31, n. 4, p. 761–769, 2020.

HUANG, G. B.; ZHU, Q. Y.; ZIEW, C. K. Extreme learning machine: Theory and applications. **Neurocomputing**, v. 70, n. 1–3, p. 489–501, 2006.

HUBER, P. J. Robust estimation of a location parameter. **Ann. Math. Stat.**, v. 35, p. 73–101, 1964.

HUNT, K. J. *et al.* Neural networks for control systems: a survey. **Automatica**, v. 28, n. 6, p. 1083–1111, 1992.

HUSSAIN, M. A. **Inverse model control strategies using neural networks: Analysis, simulation and on-line implementation**. Tese (Doutorado) — University of London, 1996.

HUSSAIN, M. A.; KERSHENBAUM, L. S. Implementation of an inverse-model-based control strategy using neural networks on a partially simulated exothermic reactor. **Chemical Engineering Research and Design**, v. 78, n. 2, p. 299–311, 2000.

JOHANSEN, T. A.; FOSS, B. A. Constructing narmax models using armax models. **Int. J. Control**, v. 58, n. 5, p. 1125–1153, 1993.

KOHONEN, T. Essentials of the self-organizing map. **Neural Networks**, v. 37, p. 52–65, 2013.

LAWRENCE, S.; TSOI, A. C.; BACK, A. D. Function approximation with neural networks and local methods: Bias, variance and smoothness. **Australian Conference on Neural Networks (ACNN)**, p. 16–21, 1996.

LIGHTBODY, G.; IRWIN, G. W. Nonlinear control structures based on embedded neural system models. **IEEE Transactions on Neural Networks**, v. 8, n. 3, p. 553–567, 1997.

LIGHTBODY, G.; IRWIN, G. W. Nonlinear control structures based on embedded neural system models. **IEEE Tran. on Neural Networks**, v. 8, n. 3, p. 553–567, 1997.

LIU, Z.-F. *et al.* Prediction short-term photovoltaic power using improved chicken swarm optimizer - extreme learning machine model. **Journal of Cleaner Production**, v. 248, p. 1–14, 2020.

LJUNG, L. **System Identification: Theory for the User**. 2nd. ed. Englewood Cliffs, NJ: Prentice-Hall, 1999.

LJUNG, L. **System Identification: Theory for the user**. 2nd. ed. Englewood Cliffs, NJ: Prentice-Hall, 1999.

MATTHIESEN, G. *et al.* Design of a semi-physical dynamic model for a pneumatic bulge test using local model networks trained with fem data. In: **2018 Global Fluid Power Society PhD Symposium (GFPS)**. [S.l.: s.n.], 2018. p. 1–6.

MU, C.; SUN, C.; YU, X. Internal model control based on a novel least square support vector machines for MIMO nonlinear discrete systems. **Neural Computing and Applications**, v. 20, n. 8, p. 1159–1166, 2011.

MUEZZINOGLU, M. K. *et al.* Acceleration of chemo-sensory information processing using transient features. **Sensors and Actuators B: Chemical**, v. 137, n. 2, p. 507–512, 2009.

NAVABI, M.; HOSSEINI, S. A hybrid pso fuzzy-mrac controller based on eulerint for satellite attitude control. In: **2020 8th Iranian Joint Congress on Fuzzy and intelligent Systems (CFIS)**. [S.l.: s.n.], 2020. p. 033–038.

NORGAARD, M. *et al.* **Neural Networks for Modelling and Control of Dynamic Systems**. [S.l.]: Springer-Verlag, 2000.

PRINCIPE, J. C.; EULIANO, N. R.; LEFEBVRE, W. C. **Neural Adaptive Systems: Fundamentals Through Simulations**. New York, NY: John Willey & Sons, 2000.

PRINCIPE, J. C.; WANG, L.; MOTTER, M. A. Local dynamic modeling with self-organizing maps and applications to nonlinear system identification and control. **Proceedings of the IEEE**, v. 86, n. 11, p. 2240–2258, 1998.

SATHISHKUMAR, E. N.; THANGAVEL, K.; DANIEL, D. A. P. Effective clustering algorithm for gas sensor array drift dataset. **International Journal of Computational Intelligence and Informatics**, v. 3, n. 3, p. 144–150, 2013.

SORIA-OLIVAS, E. *et al.* Belm: Bayesian extreme learning machine. **IEEE Transactions on Neural Networks**, v. 22, p. 505–509, 2011.

SOUZA, A. H.; BARRETO, G. A.; CORONA, F. Regional models: A new approach for nonlinear system identification via clustering of the self-organizing map. **Neurocomputing**, v. 147, p. 31–46, 2015.

SOUZA, L. G. M. **Modelos Lineares Locais Para a Identificação de Sistemas Dinâmicos Usando Redes Neurais Competitivas**. Tese (Doutorado) — Universidade Federal do Ceará, 2012.

SOUZA, L. G. M.; BARRETO, G. A. On building local models for inverse system identification with vector quantization algorithms. **Neurocomputing**, v. 73, n. 10-12, p. 1993–2005, 2010.

SOUZA, L. G. M.; SANTOS, D. C. A performance comparison of robust models in wind turbines power curve estimation: A case study. **Neural Processing Letters**, p. 1–26, 2022.

TAKAGI, T.; SUGENO, M. Fuzzy identification of systems and its applications to modeling and control. **IEEE Transactions on Systems, Man, and Cybernetics**, v. 15, p. 116–132, 1985.

TSAI, S.-H.; CHEN, Y.-W. A novel interval type-2 fuzzy system identification method based on the modified fuzzy c-regression model. **IEEE Transactions on Cybernetics**, v. 52, n. 9, p. 9834–9845, 2022.

VERGARA, A. *et al.* Chemical gas sensor drift compensation using classifier ensembles. **Sensors and Actuators B: Chemical**, v. 166, p. 320–329, 2012.

VESANTO, J.; ALHONIEMI, E. Clustering of the self-organizing map. **IEEE Trans. Neural Netw.**, v. 11, p. 586–600, 2000.

WALTER, J.; RITTER, H.; SCHULTEN, K. **Non-linear prediction with self-organizing map**. 1990. Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN'90). [S.l.: s.n.], 1990. v. 1, p. 587-592.

YAN, K.; KOU, L.; ZHANG, D. Learning domain-invariant subspace using domain features and independence maximization. **IEEE Transactions on Cybernetics**, v. 48, n. 1, p. 288–299, 2018.

YAN, K.; ZHANG, D. Correcting instrumental variation and time-varying drift: A transfer learning approach with autoencoders. **IEEE Transactions on Instrumentation and Measurement**, v. 65, n. 9, p. 2012–2022, 2016.

YANG, P. *et al.* Ensemble of kernel extreme learning machine based random forest classifiers for automatic heartbeat classification. **Biomedical Signal Processing and Control**, v. 63, p. 1–13, 2021.

YIN, Z. *et al.* A sensorless predictive torque control for induction motor using ultra-local model. In: **2021 24th International Conference on Electrical Machines and Systems (ICEMS)**. [S.l.: s.n.], 2021. p. 136–140.

ZHANG, C. *et al.* Modelling of solid oxide electrolyser cell using extreme learning machine. **Electrochimica Acta**, v. 251, p. 137–144, 2017.

ZHAO, J.; LIN, C.-M. Wavelet-tsk-type fuzzy cerebellar model neural network for uncertain nonlinear systems. **IEEE Transactions on Fuzzy Systems**, v. 27, n. 3, p. 549–558, 2019.

APÊNDICE A – APLICAÇÃO DOS MODELOS PROPOSTOS NA TAREFA DE CLASSIFICAÇÃO DE PADRÕES

Durante o processo de validação dos algoritmos, os mesmos foram utilizados para a tarefa de classificação de padrões, avaliando assim a capacidade de versatilidade dos modelos robustos aqui propostos. Aqui os modelos propostos serão aplicados na tarefa de classificação de gases submetidos a uma sequência de sensores que produzem séries temporais que caracterizam cada um dos gases estudados.

A.1 Matriz de Sensores de Gás

A aquisição do banco de dados aqui estudado foi conduzida através da aplicação de diferentes gases em diferentes concentrações a uma matriz de sensores de metal-óxido adquridos e comercialmente manufaturados. O objetivo do estudo é mitigar os desvios presentes nas leituras dos sensores utilizando os modelos múltiplos locais para os dados de bateladas existentes. Os gases foram inseridos em uma câmara controlada em laboratório, constituindo uma planta de estudos, utilizando um sistema supervisório e de aquisição de dados que performou a automação e controle de todo o experimento.

A.1.1 O banco de dados: processamento de dados e extração de características

O banco de dados foi gerado a partir da apresentação de 6 diferentes tipos de gases em concentrações variadas e sem ordem específica à matriz de sensores. Os gases estudados e suas concentrações (em partes por milhão por volume - ppmv) estão apresentados na Tabela 18.

O experimento foi conduzido durante 36 meses, fornecendo um total de 13.910 séries temporais gravadas durante o período. O conjunto de séries temporais obtidas por gases aplicados através dos meses estão listadas na Tabela 19. Dados o espaçamento e aleatoriedade dos experimentos realizados através dos meses, os conjuntos de dados foram separados pelos meses de estudos em bateladas de testes realizados, totalizando 10 bateladas através dos 36 meses.

É interessante notar que em certos meses nenhum dos gases foi aplicado à planta, porque, segundo os autores, a plataforma de experimentos foi utilizada para a realização de outros estudos usando outras misturas de gases, um fator que pode ter influenciado o uso e as subsenquentes leituras dos sensores. Em particular, da batelada de número 9 até a batelada de

Tabela 18 – Concentração dos Gases

Gás Odorante	**Concentrações (ppmv)**
Amônia	50, 60, 70, 75, 80, 90, 100, 110, 120, 125, 130 140, 150, 160, 170, 175, 180, 190, 200, 210 220, 225, 230, 240, 250, 260, 270, 275, 280 290, 300, 350, 400, 450, 500, 600, 700, 750 800, 900, 950, 1000
Acetaldeído	5, 10, 13, 20, 25, 30, 35, 40, 45, 50, 60, 70 75, 80, 90, 100, 120, 125, 130, 140, 150, 160 170, 175, 180, 190, 200, 210, 220, 225, 230 240, 250, 275, 300, 500
Acetona	12, 25, 38, 50, 60, 62, 70, 75, 80, 88, 90, 100 110, 120, 125, 130, 140, 150, 170, 175, 180 190, 200, 210, 220, 225, 230, 240, 250, 260 270, 275, 280, 290, 300, 350, 400, 450, 500 1000
Etileno	10, 20, 25, 30, 35, 40, 50, 60, 70, 75, 90, 100, 110, 120, 125, 130, 140, 150, 160, 170, 175, 180, 190, 200, 210, 220, 225, 230, 240, 250, 275, 300
Etanol	10, 20, 25, 30, 40, 50, 60, 70, 75, 80, 90, 100, 110, 120, 125, 130, 140, 150, 160, 170, 175, 180, 190, 200, 210, 220, 225, 230, 240, 250, 275, 500, 600
Tolueno	10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95, 100

número 10 que foi realizada apenas no mês 36, foi aguardado um período de 5 meses nos quais os sensores foram desligados. Os fatores apresentados fornecem um desafio adicional para os algoritmos que têm o objetivo de satisfatoriamente compensar o desvio presente nas leituras fornecidas pelos sensores através da correta classificação dos gases e favorecer a validação dos resultados obtidos.

As medições realizadas para a geração do banco de dados seguiram um padrão de operação consistindo em 3 passos:

1. A câmara de teste recebe ar seco e os sensores são mantidos em uma temperatura de operação estável de 450 °C.
2. O gás odorante é introduzido na câmara em um regime de fluxo contínuo.
3. O vapor é sugado para fora da câmara e esta é sanitizada com ar seco, um período de 10 minutos é aguardado e então uma nova medição é realizada.

A partir das respostas dos sensores, para propósitos de estudo e modelagem, a

Tabela 19 – EXPERIMENTOS ATRAVÉS DOS MESES

Teste	**Séries Temporais**						
Mês	Amônia	Acetaldeído	Acetona	Etileno	Etanol	Tolueno	Total
1	76	0	0	88	84	0	248
2	7	30	70	10	6	74	197
3	0	0	7	140	70	0	217
4	0	4	0	170	82	5	261
8	0	0	0	20	0	0	20
9	0	0	0	4	11	0	15
10	100	105	525	0	1	0	731
11	0	0	0	146	360	0	506
12	0	192	0	334	0	0	526
13	216	48	275	10	5	0	554
14	0	18	0	43	52	0	113
15	12	12	12	0	12	0	48
16	20	46	63	40	28	0	197
17	0	0	0	20	0	0	20
18	0	0	0	3	0	0	3
19	110	29	140	100	264	9	652
20	0	0	466	451	250	458	1625
21	360	744	630	662	649	568	3613
22	25	15	123	0	0	0	163
23	15	18	20	30	30	18	131
24	0	25	28	0	0	1	54
30	100	50	50	55	61	100	416
36	600	600	600	600	600	600	3600

resposta em regime permanente foi extraída, que pode ser definida a partir da diferença entre o valor máximo de resistência apresentado pelo sensor e seu valor base, gravado no passo 1 da condução dos experimentos,

$$\Delta R = max_t r(t) - min_k r(t), \tag{A.1}$$

e também do valor de sua versão normalizada, dada por,

$$\| \Delta R \| = \frac{max_k r(t) - min_k r(t)}{min_k r(t)}, \tag{A.2}$$

onde $r[k]$ refere-se ao valor de resistência no instante discreto k do experimento.

Respostas transitórias também foram extraídas do sistema, as quais constituem um conjunto chamado média móvel exponencial (ema_α), utilizada na análise da porção transiente presente na resposta da matriz de sensores durante os experimentos, calculada como,

$$y(t) = (1-\alpha)y(t-1) + \alpha(r(t) - r(t-1)), \tag{A.3}$$

sendo y(0) = 0, e a constante $0 < \alpha < 1$ é um parâmetro de suavização. No estudo α foi utilizado com 3 diferentes valores, que alteram o pico e a localização exata no tempo de análise da resposta, baseado em Muezzinoglu *et al.* (2009) os valores utilizados foram $\alpha = 0.1, \alpha = 0.01, \alpha = 0.001$, com (ema_α) sendo computado para fases de subida e descida.

Como mostrado abaixo, 8 características da matriz de sensores foram levantadas, sendo:

- ΔR,
- $\| \Delta R \|$,
- $MAXema_{\alpha=0.001}$,
- $MAXema_{\alpha=0.01}$,
- $MAXema_{\alpha=0.1}$,
- $MINema_{\alpha=0.001}$,
- $MINema_{\alpha=0.01}$,
- $MINema_{\alpha=0.1}$,

desse modo a série temporal de resposta dos 16 sensores consiste em vetor de dimensão 128×1.

A.1.2 Experimentos e Resultados

Nesta subseção, a matriz de confusão obtida para os dados de teste do processo de regressão será apresentada, também serão apresentados o valor de NMSE, a variância do erro durante as 100 realizações e a porcentagem de sucesso dos modelos. Os resultados serão apresentados para os modelos RP-MKSOM e RD-MKSOM. Na Tabela 20 os parâmetros utilizados no algoritmo são apresentados. As ordens de memória utilizadas foram: q = 3 e p = 4, escolhidas a partir do processo de validação, desde que apresentaram um menor valor de NMSE quando comparados a outros valores testados. Os dados foram normalizados na faixa [-1,+1].

Devido ao desvio presente nas leituras dos sensores, os quais se deseja mitigar através da utilização dos algoritmos, nenhum processo de contaminação dos dados com a utilização

Tabela 20 – PARÂMETROS SENSORES DE GÁS

CONJUNTO DE PARÂMETROS			
g	K	$(\alpha_0);(\alpha_T)$	$(\sigma_0);(\sigma_T)$
100	100	(0.5);(0.01)	(g/2);(0.001)

de outliers foi realizado, os modelos serão aplicados com o objetivo de alcançar uma melhor predição diante das séries temporais apresentadas conforme foram coletadas nos experimentos.

Aplicando o algoritmo RP-MKSOM ao conjunto, com a rede neural SOM como vetor de quantização vetorial, os resultados se mostraram extremamente satisfatórios com uma taxa de acertos de *100%*. O valor médio de NMSE foi de **8.5531e-09**, com uma variância de *3.8224e-18*, valor muito baixo de erro, inclusive se comparado aos conjuntos estudados nesse trabalho. Durante os experimentos percebeu-se que a vasta quantidade de dados de treinamento auxiliou o processo de aprendizagem do algoritmo, sendo utilizadas 8 bateladas para treinamento, 1 batelada para validação e 1 para teste, permitindo-o a habilidade de reconhecer as séries temporais pertencentes a determinados dados e aplicar os coeficientes apropriados ao modelado. Na Figura 16 é apresentada a matriz de confusão para os dados de teste modelados.

Para uma maior certificação dos resultados, a modelagem do sistema foi realizada utilizando apenas a primeira batelada para treinamento e a décima para teste. Como esperado, houve uma piora na precisão do algoritmo. Na Figura 17 a matriz de confusão para os dados de teste modelados é apresentada.

Como observado a partir da matriz de confusão, houve um grande aumento na quantidade de dados que foram classificados erroneamente, a taxa de acertos foi des *60.45%*, com um valor médio de NMSE **0.4139** e variância de *0.3080*. Mesmo com a piora nos resultados, percebeu-se que a quantidade de acertos está bem acima daqueles obtidos em trabalhos correlatos para o mesmo conjunto de dados e com configurações semelhantes, ou seja, utilizando a primeira batelada para treinamento e a décima para teste (VERGARA *et al.*, 2012; SATHISHKUMAR; THANGAVEL; DANIEL, 2013; YAN; ZHANG, 2016; YAN; KOU; ZHANG, 2018).

Para o algoritmo RD-MKSOM também foi utilizada a mesma metodologia de treinamento e teste. O valor dos hiperparâmetros são os mesmos apresentados na Tabela 20 para a obtenção dos melhores resultados de teste. Os resultados aqui obtidos se mostraram ainda mais satisfatórios que os resultados obtidos para o algoritmo RP-MKSOM, visto que, tanto no melhor quanto no pior caso os valores de acertos foram de *100%*.

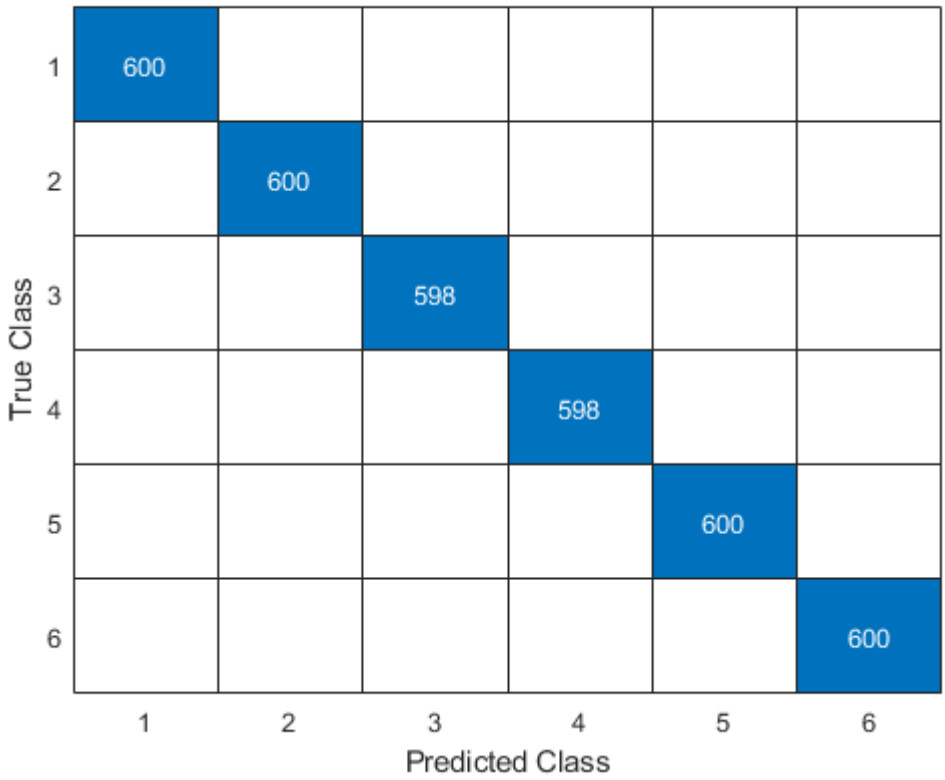


Figura 16 – Matriz de Confusão Resultante RP-MKSOM: 8 bateladas de treinamento e 1 de teste. A identificação dos gases segue da seguinte maneira: 1 - Amônia; 2 - Acetaldeído; 3 - Acetona; 4 - Etileno; 5 - Etanol; 6 - Tolueno.

No melhor caso, considerando 8 bateladas para treinamento e 1 para teste, apresentamos a matriz de confusão na Fig. 18, percebe-se que houve acerto em todas as classificações, o valor médio de NMSE calculado foi de **7.2207e-13** e a variância encontrada atingiu *1.3229e-28*, valores baixos considerando a diversidade e dificuldade de classificação dos dados.

Já para o pior caso, com 1 batelada para treinamento e 1 para teste, apresentamos a matriz de confusão na Fig. 19, também houve acerto em todas as classificações, o valor de NMSE médio calculado foi de **4.4281e-04** e a variância encontrada atingiu *9.9337e-09*, obtendo um resultado extremamente satisfatório considerando a baixa quantidade de dados de treinamento.

Considerando os resultados apresentados, o algoritmo RD-MKSOM pode ser indicado para a realização de tarefas semelhantes, tendo apresentado uma menor taxa de erro e de variância, ainda que ambos tenham obtido valores satisfatórios de acertos com uma quantidade suficiente de dados de treinamento. O leitor pode perceber que a quantidade de dados apresentados na matriz de confusão foi de apenas 3.596, sendo que o banco de dados é de 3.600, isto ocorre devido ao fato de que a série predita apresenta dados a menos do que a série de teste inicial por causa da ordem de memória p = 4, no entanto, não houve nenhuma perca na taxa de

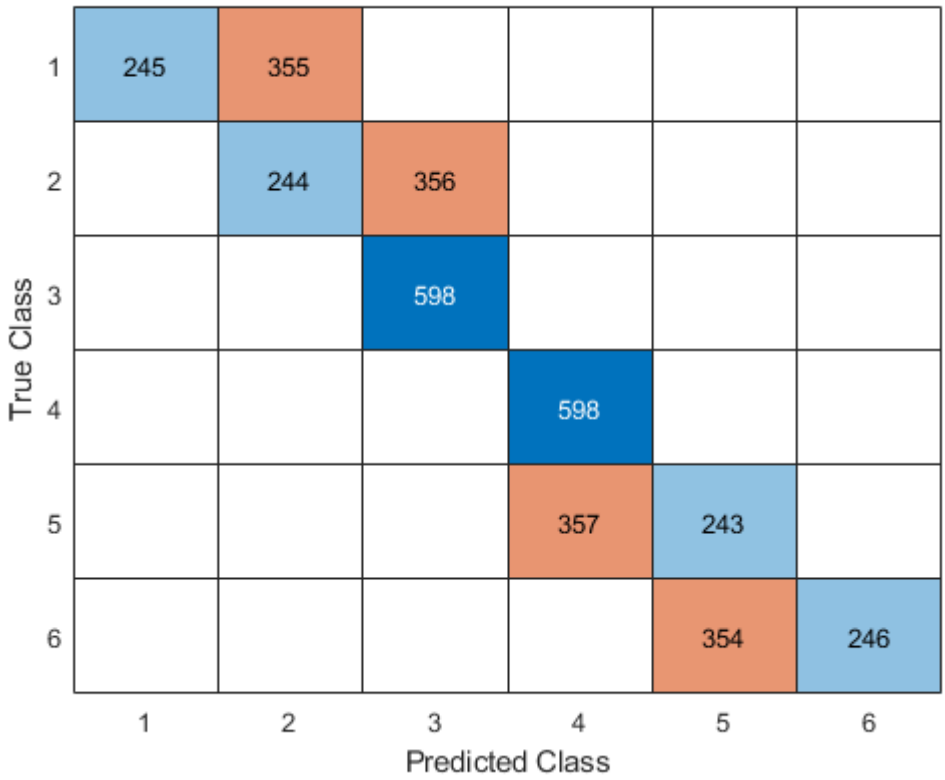


Figura 17 – Matriz de Confusão Resultante RP-MKSOM: 1 batelada de treinamento e 1 de teste. A identificação dos gases segue da seguinte maneira: 1 - Amônia; 2 - Acetaldeído; 3 - Acetona; 4 - Etileno; 5 - Etanol; 6 - Tolueno.

acertos ou na capacidade de predição do algoritmo por conta desse fator.

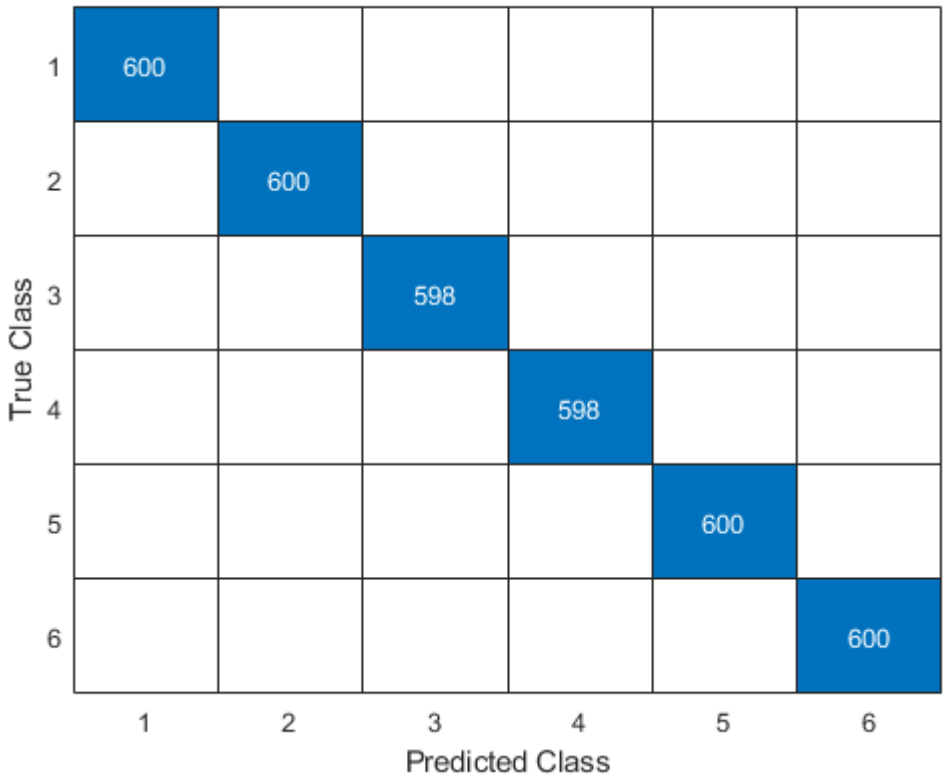


Figura 18 – Matriz de Confusão Resultante RD-MKSOM: 8 bateladas de treinamento e 1 de teste. A identificação dos gases segue da seguinte maneira: 1 - Amônia; 2 - Acetaldeído; 3 - Acetona; 4 - Etileno; 5 - Etanol; 6 - Tolueno.

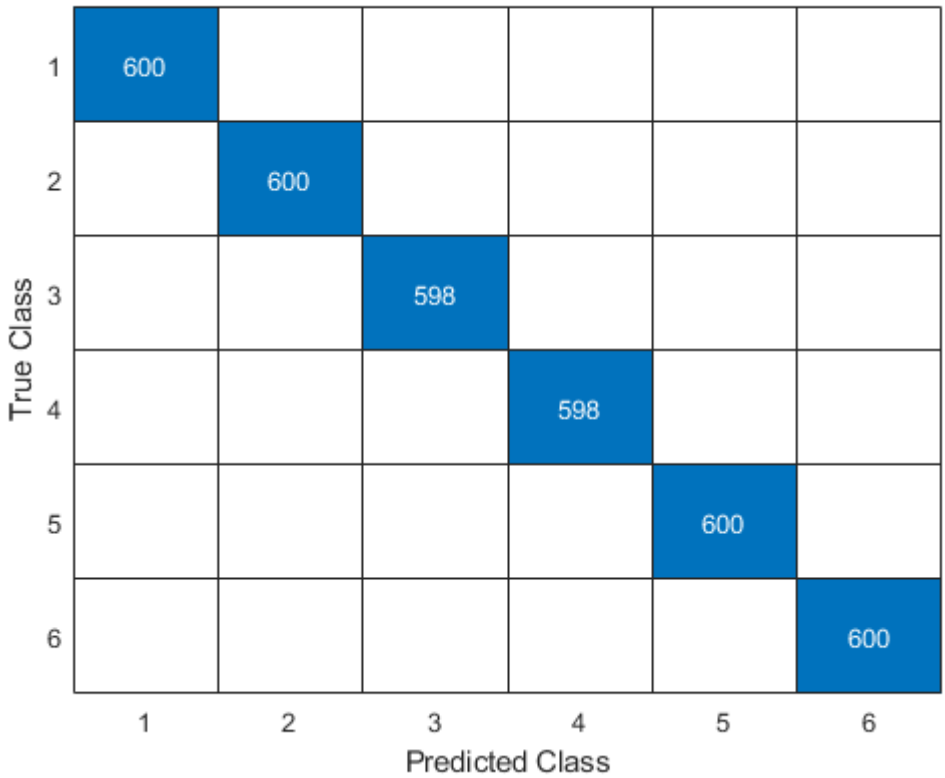


Figura 19 – Matriz de Confusão Resultante RD-MKSOM: 1 batelada de treinamento e 1 de teste. A identificação dos gases segue da seguinte maneira: 1 - Amônia; 2 - Acetaldeído; 3 - Acetona; 4 - Etileno; 5 - Etanol; 6 - Tolueno.