UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA DE TELEINFORMÁTICA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE TELEINFORMÁTICA

MESTRADO ACADÊMICO EM ENGENHARIA DE TELEINFORMÁTICA

RENAN BESSA

IDENTIFICAÇÃO ROBUSTA DE SISTEMAS DINÂMICOS USANDO REDES DE ECOS DE ESTADO

FORTALEZA

2019

RENAN BESSA

IDENTIFICAÇÃO ROBUSTA DE SISTEMAS DINÂMICOS USANDO REDES DE ECOS DE ESTADO

Dissertação apresentada ao Curso de Mestrado Acadêmico em Engenharia de Teleinformática do Programa de Pós-Graduação em Engenharia de Teleinformática do Centro de Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em Engenharia de Teleinformática. Área de Concentração: Engenharia de Teleinformática

Orientador: Prof. Dr. Guilherme de Alencar Barreto

FORTALEZA

2019

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

B465i Bessa, Renan.
Identificação Robusta de Sistemas Dinâmicos Usando Redes de Ecos de Estado / Renan Bessa. – 2019.
96 f. : il. color.

Dissertação (mestrado) – Universidade Federal do Ceará, Centro de Tecnologia, Programa de Pós-Graduação em Engenharia de Teleinformática, Fortaleza, 2019.
Orientação: Prof. Dr. Guilherme de Alencar Barreto.

1. Identificação de sistemas. 2. Rede neural recorrente. 3. Rede de ecos de estados. 4. Robustez a outliers. I. Título.

CDD 621.38

RENAN BESSA

IDENTIFICAÇÃO ROBUSTA DE SISTEMAS DINÂMICOS USANDO REDES DE ECOS DE ESTADO

Dissertação apresentada ao Curso de Mestrado Acadêmico em Engenharia de Teleinformática do Programa de Pós-Graduação em Engenharia de Teleinformática do Centro de Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em Engenharia de Teleinformática. Área de Concentração: Engenharia de Teleinformática

Aprovada em: 17 de Agosto de 2019

BANCA EXAMINADORA

Prof. Dr. Guilherme de Alencar Barreto (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Francesco Corona
Universidade Federal do Ceará (UFC)

Prof. Dr. Fabrício Gonzalez Nogueira
Universidade Federal do Ceará (UFC)

AGRADECIMENTOS

Ao Prof. Dr. Guilherme de Alencar Barreto pela amizade, oportunidade, orientação e confiança em mim para a realização deste trabalho.

À minha família pelo apoio, em especial à minha prima Luana que me ajudou muito na escrita deste trabalho.

Aos meus amigos do programa de pós-graduação em Engenharia de Teleinformática, Michael Duarte, David Coelho, Diego Perdigão, Renan Fonteles, Júlio Peixoto, Haroldo Maya, Kelvin Sales, Policarpo Souza, Manoel Henrique, Thiago Duarte, Rômulo Drumond pela companhia, discussões e colaborações durante este período de estudos.

Às minhas amigas de outros programas de pós-graduação, Jermana Lopes e Rebeca Guerreiro, por todo apoio e ajuda.

À Fundação Cearense de Apoio ao Desenvolvimento (Funcap) pelo financiamento da pesquisa de mestrado via bolsa de estudos.

"A liberdade consiste em conhecer os cordéis que nos manipulam."

(Baruch Spinoza)

RESUMO

O emprego de redes neurais recorrentes na identificação *online* de sistemas é bastante limitada em aplicações do mundo real, principalmente devido à propagação do erro causada pela natureza interativa da tarefa de predição ao longo de vários passos à frente. Tendo isso em mente, revisita-se a arquitetura da rede de ecos de estados (*echo state network*, ESN) a respeito da eficiência na tarefa de aprendizagem *online* usando algoritmo de estimação recursivo e uma variante robusta a *outliers* da mesma. Por meio de um conjunto abrangente de experimentos, mostra-se que o desempenho da ESN é dependente da escolha adequada das vias de realimentação e que a instabilidade da previsão é ampliada pela norma do vetor de pesos da saída, uma questão frequentemente negligenciada em estudos relacionados.

Palavras-chave: Identificação de sistemas. Rede neural recorrente. Rede de ecos de estados. Robustez a *outliers*.

ABSTRACT

The use of recurrent neural networks in online system identification is very limited in real-world applications, mainly due to the propagation of errors caused by the iterative nature of the prediction task over multiple steps ahead. Bearing this in mind, in this paper, we revisit design issues regarding the robustness of the *echo state network* (ESN) model in such online learning scenarios using a recursive estimation algorithm and an outlier robust-variant of it. By means of a comprehensive set of experiments, we show that the performance of the ESN is dependent on the adequate choice of the feedback pathways and that the prediction instability is amplified by the norm of the output weight vector, an often neglected issue in related studies.

Keywords: System identification. Recurrent neural networks. Echo state network. Robustness to outliers.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

APRBS	Sinal binário pseudoaleatório com modulação de amplitude (*amplitude modulated pseudo-random binary signal*)
ARMAX	Autorregressivo de médias móveis com entradas exógenas (*autoregressive moving average with exogenous input*).
BPDC	Retropropagação do erro descorrelacionada (*backpropagation-decorrelation*)
ESN	Rede de ecos de estados (*echo state network*).
FH	Função de Huber.
FHM	Função de Huber.
GP	Processos gaussianos (*gaussian processes*).
KRLS	Mínimos quadrados recursivo kernelizado (*kernel recursive least squares*)
LMS	Mínimos médios quadrados (*least mean squares*).
MIMO	Múltiplas entradas e múltiplas saídas (*multiple-input multiple-output*).
NARMAX	Não linear autorregressivo de médias móveis com entradas exógenas (*non-linear autoregressive moving average with exogenous input*).
NARX	Não linear autorregressivo com entradas exógenas (*non-linear autoregressive with exogenous input*).
OLS	Mínimos quadrados ordinário (*ordinary least squares*).
PRBS	Sinal binário pseudoaleatório (*pseudo-random binary signal*)
SVM	Máquina de vetor suporte (*suport vector machine*).
RLM	Mínimo estimador-*M* recursivo (*recursive least M-estimate*).
RLMCFB	RLM com realimentação cautelosa (RLM *with cautious feedback*)
RLS	Mínimos quadrados recursivo (*recursive least squares*).
RMSE	Raiz do erro quadrático médio (*root mean square error*).
RNN	Rede neural recorrente (*recurrent neural network*).
SIMO	Entrada única e múltiplas saídas (s*ingle-input multiple-output*).
SISO	Entrada única e saída única (s*ingle-input single-output*).

LISTA DE SÍMBOLOS

T_S	Tempo de amostragem.
n	n-ésimo instante.
$u(n), \mathbf{u}(n)$	Entrada amostrada do sistema (escalar ou vetorial).
$t(n), \mathbf{t}(n)$	Saída amostrada do sistema (escalar ou vetorial).
z^{-1}	Operador de atraso unitário.
$A(z^{-1}), B(z^{-1}), C(z^{-1}), D(z^{-1}), P(z^{-1})$	Polinômios do modelo ARMAX.
a, b, c, d, p	Coeficientes dos polinômios do modelo ARMAX.
n_a, n_b, n_c, n_d, n_p	Ordens dos dos polinômios do modelo ARMAX.
τ	Representa o número de atrasos.
$\{\varepsilon(n)\}$	Ruído inerente do sistema.
$y(n), \mathbf{y}(n)$	Saída do modelo (escalar ou vetorial).
$\mathbf{x}(n)$	Vetor de variáveis de estados de um sistema.
$F(\cdot)$	Função não linear qualquer.
$e(n), \mathbf{e}(n)$	Erro da saída do modelo (escalar ou vetorial).
n_y, n_u, n_ε	Atrasos máximos dos sinais do modelo.
N_{all}	Quantidade de amostras de um conjunto.
N_{test}	Quantidade de amostras do subconjunto de teste.
N_{train}	Quantidade de amostras do subconjunto de treinamento.
K	Número de entradas de um sistema.
L	Número de saídas de um sistema.
R	Número de neurônios no reservatório.
$\mathbf{f}(\cdot), \mathbf{f}_{out}(\cdot)$	Função de ativação dos neurônios, elemento a elemento.
$\mathbf{W}$	Matriz de pesos do reservatório da ESN.
$\mathbf{W}_{in}$	Matriz de pesos da entrada da ESN.
$\mathbf{W}_{fb}$	Matriz de pesos da retroalimentação da ESN.
$\mathbf{b}$	Vetor de *bias*.

$\mathbf{W}_{out}$	Matriz de pesos da saída da ESN.
$\mathbf{h}(n)$	Vetor das entradas da camada de saída da ESN.
H	Número de entradas da camada de saída da ESN.
$\rho(\mathbf{W})$	Raio espectral de $\mathbf{W}$.
λ_{max}	Maior autovalor de $\mathbf{W}$.
γ	Taxa de vazamento do reservatório.
J_{OLS}	Função custo do algoritmo OLS.
$\mathbf{v}_1, \mathbf{v}_2$	Vetor opcional de injeção de ruído na ESN.
J_{LS}	Função custo do algoritmo RLS.
λ	Fator de esquecimento dos algoritmos RLS e RLM.
J_ρ	Função custo do algoritmo RLM.
$\rho(\cdot)$	Função estimador-M.
$\rho(\cdot)$	Derivada primeira da função estimador-M.
$q(\cdot)$	Função peso estimador-M.
$\mathbf{R}(n)$	Matriz de correlação de $\mathbf{h}(n)$.
$\mathbf{p}(n)$	Matriz de correlação cruzada de $\mathbf{h}(n)$ e $t(n)$.
$\mathbf{I}$	Matriz identidade.
$\mathbf{S}(n)$	Matriz inversa de correlação de $\mathbf{h}(n)$.
$\mathbf{k}(n)$	Vetor de ganho do algoritmo RLM.
$\hat{\sigma}$	Estimador do desvio padrão dos erros livre de *outliers*.
$\lambda_{\hat{\sigma}}$	Fator de esquecimento do estimador $\hat{\sigma}$.
$med\{\cdot\}$	Operador mediana.
N_W	Quantidade de amostras utilizadas no operador mediana.
ξ, Δ_1, Δ_2	Limiares das funções estimador-M.
δ	Parâmetro de regularização da norma do vetor estimado pelos algoritmos RLS e RLM.
α, β	Referente as vias de retroalimentação da ESN.
$A0, A1, A2, A3$	Arquiteturas da ESN.

ξ_{cfb}	Limiar do algoritmo RLMCFB.
$N1, N2, N3$	Configurações de $\mathbf{v}_1$ e $\mathbf{v}_2$ testadas.

SUMÁRIO

1 INTRODUÇÃO

A rede de ecos de estados (*echo state network*, ESN) (JAEGER, 2001) é uma rede neural recorrente (*recurrent neural network*, RNN) que possui um grande número de neurônios, formando o chamado reservatório, com vias esparsas de interconexão e de realimentação. Os pesos de entrada dos neurônios do reservatório, os internos e os responsáveis pelas conexões com os sinais de entrada ou pelas retroalimentações da saída do modelo, são fixos e aleatoriamente atribuídos. O treinamento dessa rede é constituído de uma estimação dos pesos dos seus neurônios da camada de saída, também chamada de camada de *readout*. Essa estimativa é realizada por meio de regressão linear, geralmente utilizando o conhecido método dos mínimos quadrados ordinário (*ordinary least squares*, OLS). A natureza aleatória da ESN combinada com a estimação linear dos pesos da camada de saída torna a sua concepção muito simples se comparada com as demais RNNs, as quais possuem convergência lenta, quando essa ocorre, e alto custo computacional, principalmente as que empregam o treinamento por retropropagação do erro (*error backpropagation*) (LUKOŠEVIČIUS, 2012).

A ESN tem sido usada como uma poderosa ferramenta para predição de séries caóticas, reconstrução de atratores e identificação de sistemas não lineares em geral (JAEGER, 2003). Com o objetivo de aprimorar essa rede para lidar com dados reais, os quais são passíveis de apresentarem ruídos não gaussianos e pontos inconsistentes com o sistema (*outliers*), alguns trabalhos apresentam métodos robustos para estimação dos pesos da camada de *readout* da ESN. Um dos métodos mais simples é a regularização de Tikhonov, que penaliza o vetor de pesos com norma elevada, reduzindo o sobreajuste do modelo aos dados corrompidos (LUKOŠEVIČIUS; JAEGER, 2009).

Abordagens mais complexas na estimação dos pesos também são propostas, incluindo a inferência bayesiana em que se substitui a usual função de verossimilhança gaussiana, que é bastante sensível aos *outliers*, por uma distribuição laplaciana (LI *et al.*, 2012) ou gaussiana mista (HAN; XU, 2018). Além disso, o emprego de critérios de aprendizagem baseados na teoria da informação, critério da máxima correntropia (*maximum correntropy*) (GUO *et al.*, 2017) e da máxima correntropia generalizada (*maximum generalized correntropy*) (ZHANG *et al.*, 2018), também alcançaram bons resultados na diminuição dessa sensibilidade.

Outros arranjos robustos baseados na ESN substituem a habitual camada de saída linear por ferramentas não lineares, cujas entradas incluem os sinais de ativação dos neurônios do reservatório, em que se busca aproveitar ao máximo a dinâmica gerada por essa estrutura.

Uma dessas abordagens é a formulação com a máquina de vetor suporte (*suport vector machine*, SVM) com função custo robusta, tais como a função perda ε-insensível (ε-*insensitive loss*) ou a função de Huber (SHI; HAN, 2007).

Ademais, o uso de conceitos de filtragem adaptativa kernelizada como no algoritmo dos mínimos quadrados recursivo kernelizado (*kernel recursive least squares*, KRLS), em que se utiliza o critério de atualização do dicionário para lidar com pontos inconsistentes chamado de dependência linear aproximada (*approximate linear dependency*), é capaz de realizar um treinamento *online* e robusto da ESN (ZHOU *et al.*, 2018). Apesar de não ter sido avaliado diretamente, existem referências nas quais arranjos da ESN com algoritmo automapas laplacianos (*laplacian eigenmaps*) (HAN; XU, 2018), mediando a redução da dimensionalidade dos estados do reservatório, e modelos de regressão através de processos gaussianos (*gaussian processes*, GP) (CHATZIS; DEMIRIS, 2011) podem ser capazes de lidar com os efeitos adversos causados pelos *outliers*.

Apesar das relevantes capacidades de modelagem das arquiteturas da ESN mencionadas, sua aplicação à identificação de sistemas não lineares reais é limitada. Isso é particularmente verdadeiro em cenários envolvendo dados reais em previsões iterativas de múltiplos passos à frente de longo prazo. Como a chance de encontrar *outliers* em tais cenários é muito alta, somada a propagação de erros devido à natureza iterativa da tarefa de simular sistemas, a resposta do modelo pode sofrer divergência (ou seja, instabilidade) nos sinais previstos com o passar do tempo. Tal instabilidade é amplificada pela norma do vetor de peso de saída, um problema muitas vezes negligenciado em estudos anteriores.

Tendo em mente a estabilidade da previsão no longo prazo, nesse trabalho é revisitada a ESN combinada com o algoritmo dos mínimos quadrados recursivo (*recursive least squares*, RLS) (HAYKIN, 2014), a RLS-ESN (JAEGER, 2003), para identificação *online* de sistemas na presença de *outliers*. Pela exploração das diferentes arquiteturas de vias de realimentação da ESN e de abordagens fundamentadas na substituição do algoritmo RLS por uma de suas variantes robusta chamada de algoritmo do mínimo estimador-*M* recursivo (*recursive least M-estimate*, RLM) (ZOU *et al.*, 2000), a ESN se mostra capaz de ser usada com segurança para tarefas de predição de múltiplos passos à frente de longo prazo. Um conjunto abrangente de simulações computacionais é realizado usando dados de sistemas reais, sendo três sistemas de entrada única e saída única (s*ingle-input single-output*, SISO) e um sistema de entrada única e múltiplas saídas (s*ingle-input multiple-output*, SIMO), e de um sistema sintético de múltiplas entradas e múltiplas

saídas (*multiple-input multiple-output*, MIMO), em que são aplicadas contaminações por *outliers* em diferentes proporções nas saídas desses sistemas para aferir a robustez dos modelos.

1.0.1 Objetivos da Dissertação

Diante dessa perspectiva, a contribuição principal desse trabalho é a formulação de uma ESN robusta a *outliers*, com treinamento *online* e capaz de realizar previsões em simulação livre de maneira estável e satisfatória. Para alcançar esse objetivo, as seguintes etapas são necessárias:

1. Revisitar e avaliar a RLS-ESN aplicada à identificação de diferentes tipos de sistemas. Além de elencar variações de sua arquiteturas, explorando possíveis vias de realimentação da saída do modelo;
2. Incorporar robustez na estimação da ESN na presença de *outlier*, utilizando o algoritmo RLM, e avaliar o desempenho das arquiteturas e dos estimadores em cenários com contaminação das amostras dos sistemas por *outliers* em diferentes proporções;
3. Propor uma estratégia para minimizar o efeito desestabilizante dos *outliers* na propagação do erro durante o treinamento e o teste da ESN;
4. Analisar as implicações das arquiteturas e dos estimadores no desempenho da ESN na identificação de sistemas na presença de *outliers*.

1.0.2 Produção Científica

Durante o período do mestrado, a seguinte produção científica foi realizada:

1. BESSA, R.; BARRETO, G. A. Robust echo state network for recursive system identification. In: **Advances in Computational Intelligence**. Springer International Publishing, 2019. p. 247–258. Disponível em: https://doi.org/10.1007/978-3-030-20521-8_21. Acesso em: 01/07/2019.

1.0.3 Organização da Dissertação

A organização dessa dissertação é feita da seguinte forma:

Capítulo 2: Apresenta-se o problema de identificação de sistemas com o intuito de descrever as etapa dessa tarefa e introduzir suas ferramentas, suas operações e suas nomenclaras que são aplicadas no decorrer desse trabalho;

Capítulo 3: Elabora-se a fundamentação da ESN, seus princípios teóricos e construtivos, com o foco no problema de estimação robusta aos *outliers*, inclusive se descreve o algoritmo robusto RLM;

Capítulo 4: Defini-se a metodologia dos experimentos que são aplicados como prova de conceito e elencam-se os conjuntos de dados utilizados. Além disso, descreve-se uma proposta para aprimorar a adaptação da RLM-ESN;

Capítulo 5: Expõem-se os resultados obtidos para cada conjunto de dados e apresentam-se discussões sobre os mesmos;

Capítulo 6: Apresentam-se as conclusões sobre os objetivos alcançados e direcionam-se possíveis trabalhos futuros.

2 FUNDAMENTOS DE IDENTIFICAÇÃO DE SISTEMAS DINÂMICOS

Para melhor entendimento de um certo sistema, um modelo matemático pode ser elaborado para predizer ou simular o seu comportamento em diversas situações arbitrárias. Assim, tal modelo pode ser utilizado na análise de processos já existentes ou para concepção de novos. Consequentemente, esse instrumento pode ser aplicado em projeto de controladores, otimização, supervisão, detecção de falhas e diagnóstico de componentes do sistema por exemplo (NELLES, 2001).

De fato, raramente se dispõem de tempo e conhecimento suficientes para elaborar modelos eficazes a partir das equações da física que governa o processo, normalmente denominada de modelagem caixa branca. Todavia, o procedimento chamado de identificação de sistemas possibilita a construção de um mapeamento entrada-saída que emula de forma aproximada uma relação de causa e efeito, apoiando-se em dados coletados da operação normal ou induzida de um sistema real. Ou seja, dispondo de pouco ou nenhum conhecimento prévio do sistema, possuindo apenas um conjunto de amostras dos seus sinais de entrada e de saída, pode-se realizar uma modelagem empírica, também referida como modelagem caixa preta (LJUNG, 1983). Apesar das singularidades de cada sistema real, o esquema de identificação deve ser universal, sendo capaz de descrever uma vasta coleção de processos estruturalmente diferentes (NELLES, 2001).

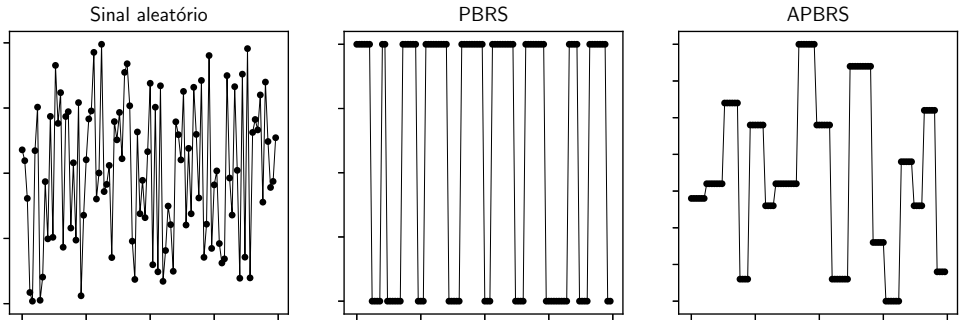
Dito isso, comentam-se as principais etapas e nomenclaturas relativas à tarefa de identificação de sistemas nesse Capítulo.

2.1 Sobre a coleta dos dados

Para realizar a tarefa de identificação de sistemas, é necessário coletar os dados dos sinais relevantes de entrada e de saída do processo. Tendo em mente que a grande parte dos sistemas reais é contínuo, isto é, pode ser aquisitado a qualquer momento, é interessante padronizar a periodicidade da amostragem. Dessa forma, a aquisição dos dados tem uma propriedade denominado de tempo de amostragem, T_S. Assim, busca-se registrar as variáveis em uma frequência alta o suficiente para capturar as características fundamentais do sinal. Na prática, apesar do teorema de Shannon indicar uma frequência de amostragem duas vezes maior do que a maior componente de frequência de interesse contida nas variáveis do sistema, esse valor chega ser de cinco a dez vezes maior normalmente (AGUIRRE, 2015).

Além disso, é usual realizar experimentos aplicando um sinal de excitação no sistema

Figura 1 – Sinais de excitação para identificação de sistemas.



Fonte: Autoria própria.

para amostrar sua resposta. Também, é importante a escolha desse sinal, pois se deve optar por um estímulo de entrada que seja capaz de causar a manifestação da maioria das características dinâmicas, estáticas e não lineares do sistema nos dados coletados. Entretanto, existem casos que não é possível interromper o funcionamento da planta, implicando em uma coleta de amostras durante a operação normal. No caso contrário, a excitação do sistema por meio de sinal aleatório, binário pseudoaleatório (*pseudo-random binary signal*, PRBS) ou binário pseudoaleatório com modulação de amplitude (*amplitude modulated pseudo-random binary signal*, APRBS) são amplamente aplicados para a identificação de sistemas (AGUIRRE, 2015). A Figura 1 apresenta o comportamento desses sinais citados.

Dessa maneira, a realização da aquisição dos sinais do sistema, que por questão de simplicidade é tratado como um sistema SISO, resulta nos conjuntos formados pelas entradas $\{u(n)\}$ e pelas saídas $\{t(n)\}$, em que $n = 1,2,3,...$ representa a sequência dos instantes de tempo em que a amostra é coletada, lembrando que a diferença entre esses instantes é o tempo de amostragem.

2.2 Sobre os modelos

Com o conjunto de dados disponíveis, a etapa seguinte é a escolha de um modelo paramétrico para identificação do sistema, sendo um ponto bastante importante para o sucesso desse método. Esse tipo de modelo possui parâmetros, também referidos como pesos, a serem ajustados de acordo com os dados e pode ser dividido, de forma generalista, em linear e não linear. A princípio, deve-se considerar as estruturas lineares. À vista disso, tem-se o modelo geral chamado de autorregressivos de médias móveis com entradas exógenas (*autoregressive*

moving average with exogenous input, ARMAX), sendo projetado como

$$A(z^{-1})y(n) = \frac{B(z^{-1})}{P(z^{-1})}u(n) + \frac{C(z^{-1})}{D(z^{-1})}\varepsilon(n), \tag{2.1}$$

em que

$$\begin{aligned} A(z^{-1}) &= 1 + a_1 z^{-1} + ... + a_{n_a} z^{-n_a}, \\ B(z^{-1}) &= b_0 + b_1 z^{-1} + ... + b_{n_b} z^{-n_b}, \\ C(z^{-1}) &= 1 + c_1 z^{-1} + ... + c_{n_c} z^{-n_c}, \\ D(z^{-1}) &= 1 + d_1 z^{-1} + ... + d_{n_d} z^{-n_d}, \\ P(z^{-1}) &= 1 + p_1 z^{-1} + ... + p_{n_p} z^{-n_p}; \end{aligned} \tag{2.2}$$

n_a, n_b, n_c, n_d e n_p são inteiros não negativos que representam as ordens dos respectivos polinômios dos pesos do modelo; $\{\varepsilon(n)\}$ denota uma sequência de ruído independente e identicamente distribuída com média zero e variância finita, que refere-se a um ruído inerente a mensuração de um sinal; e $\{y(n)\}$ representa a saída do modelo. Por fim, o símbolo z^{-1} indica o operador de deslocamento unitário, definido por $z^{-\tau}s(n) = s(n-\tau)$, sendo s qualquer sinal e τ um número inteiro (BILLINGS, 2013).

A representação do modelo ARMAX apresentada pode ser simplificada da seguinte forma

$$\begin{aligned} y(n) = a_1 y(n-1) + ... + a_{n_y} y(n-n_y) + b_d u(n-d) + ... + b_{n_u-d} u(n-n_u) + \\ + \varepsilon(n) + c_1 \varepsilon(n-1) + ... + c_{n_\varepsilon} \varepsilon(n-n_\varepsilon), \end{aligned} \tag{2.3}$$

na qual n_y, n_u e n_ε representam os atrasos máximos dos sinais de saída, de entrada e de ruído, respectivamente. Além disso, o valor de d representa o tempo morto, ou seja, o intervalo de tempo que uma variação no sinal de entrada é detectado na saída do sistema, que pode ser desconsiderado fazendo $d = 0$. Porém, se o modelo linear não apresentar um desempenho satisfatório, uma das possíveis justificativas é a não linearidade do processo. Nessa circunstância, uma boa estratégia é seguir para modelos não lineares que englobem, como um caso especial, o modelo linear. Por exemplo, arquiteturas polinomiais ou sistemas *fuzzy* (BILLINGS, 2013).

De forma análoga à estrutura ARMAX, o modelo não linear autorregressivo de médias móveis com entradas exógenas (*non-linear autoregressive moving average with exogenous input*, NARMAX) é denotado por

$$y(n) = F(y(n-1), ..., y(n-n_y), u(n-d), ..., u(n-n_u), \varepsilon(n-1), ..., \varepsilon(n-n_\varepsilon)) + \varepsilon(n), \tag{2.4}$$

em que $F(\cdot)$ representa uma função não linear que tem como argumento os valores de entradas, de saídas e de erros atrasados, também chamados de regressores do modelo. Normalmente, essa função é conhecida a priori e possui parâmetros ajustáveis para se adaptar à dinâmica do sistema (AGUIRRE *et al.*, 1998).

Acrescenta-se que inúmeros paradigmas de modelagem que aplicam os princípios do modelo NARMAX para a identificação de sistemas estão disponíveis na literatura, tendo como exemplo as funções polinomiais e racionais (CHEN; BILLINGS, 1989), as redes neurais artificiais (NARENDRA; PARTHASARATHY, 1990), as redes neurais com função de base radial (CASDAGLI, 1989), os sistemas *fuzzy* (TAKAGI; SUGENO, 1993), os sistemas *neuro-fuzzy* (JANG, 1993), os métodos baseados em SVM (GESTEL *et al.*, 2001) e em GP (KOCIJAN *et al.*, 2005), além de outras técnicas.

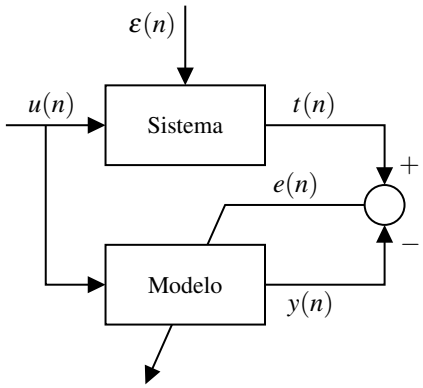
2.3 Sobre estimação e validação dos modelos

O modelo deve reproduzir o comportamento do processo tão próximo quanto possível, ou seja, deve existir uma adaptação do mesmo ao sistema. Com isso em mente, pode-se moldar o modelo por meio dos seus hiperparâmetros, que são relacionado a sua estrutura, e dos seus parâmetros, também referidos por coeficientes ou pesos, que estão presentes nessa estrutura e são adaptados automaticamente seguindo uma função custo a ser otimizada, maximização ou minimização de um ou mais critérios. Exemplos típicos de hiperparâmetros são o número de regressores do modelo ARMAX; o número de neurônios e de camadas de uma rede neural; o número de regras de um sistema *fuzzy*; e assim por diante (NELLES, 2001).

O desempenho do modelo pode ser qualificado essencialmente pelo desvio entre a saída estimada $y(n)$ e o valor aferido $t(n)$, sendo também chamado de valor desejado (*target*), isto é, $e(n) = t(n) - y(n)$. Por conseguinte, esse erro $e(n)$ pode ser utilizado para adaptação do modelo ao sistema em estudo, sendo comumente um dos argumentos de função custo em identificação de sistemas para a atualização dos pesos, tendo sua representação clássica expressa na Figura 2. A essa operação dar-se o nome de estimação ou treinamento do modelo (NELLES, 2001).

Três tipos de abordagens de treinamento podem ser distinguidas. A primeira delas é a descrita no paragrafo anterior, chamada de aprendizado supervisionado, em que é baseada no conhecimento sobre as entradas e as saídas do processo e, consequentemente, sabe-se o exato desvio entre o modelo e o processo para cada amostra. A segunda trata o caso em que se

Figura 2 – Identificação de sistemas: um modelo aprendendo o comportamento do processo.



Fonte: Adaptada de Nelles (2001).

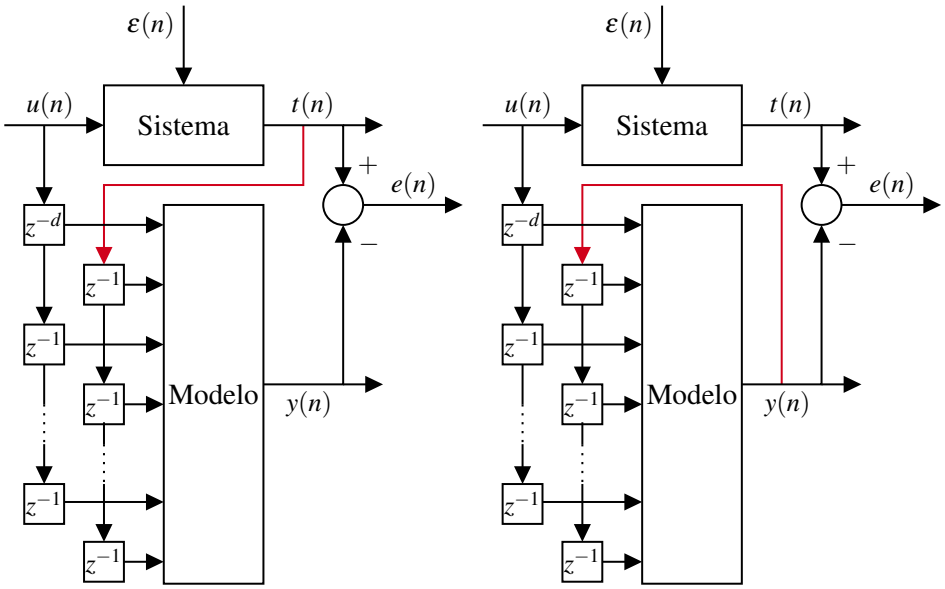
possui conhecimento parcial do processo, denominada de aprendizado por reforço, na qual não é possível uma avaliação a cada entrada, mas se tem apenas a informação de sucesso ou falha da execução do modelo. Além dessas, tem-se o aprendizado não supervisionada, em que apenas utiliza os dados de entrada na tentativa de realizar agrupamentos estatisticamente significativos, explorando a distribuição apresentada por essas entradas, sendo aplicada no pré-processamento dos dados principalmente (NELLES, 2001). Entretanto, as últimas duas abordagens não são discutidas nesse trabalho.

Outra característica do treinamento supervisionado é a forma que os dados são disponibilizados. O aprendizado em lote ou em batelada se particulariza por utilizar todas as amostras disponíveis, que são processadas de uma vez para realizar a estimação. Dessa maneira, se por acaso seja apresentada uma nova amostra, esse método requer uma varredura em todas os dados anteriores. Consequentemente, para grandes conjuntos de dados, demanda-se bastante esforço computacional e espaço de armazenamento para essa operação (BILLINGS, 2013).

Em contraposição, há o aprendizado recursivo que realiza uma estimação a cada amostra apresentada ou em relação a uma janela amostral que se move através dos conjunto de dados. Esse método também é referido como aprendizado *online*, indicando a possibilidade de estimação em tempo real com o processo. Além disso, essa abordagem possibilita a aplicação em sistemas variantes com o tempo, ou seja, adaptar-se às mudanças sofridas pelo sistema ao passar do tempo (BILLINGS, 2013).

Os modelos aplicados ao mapeamento entrada-saída apresentam reiteradamente duas formas de operação: predição de um passo à frente e simulação livre. Além das entradas do sistema $\{u(n-i)\}_{i=d}^{n_u}$, a predição de um passo à frente demanda os valores de saídas passadas do sistema $\{t(n-i)\}_{i=1}^{n_y}$ durante a operação do modelo para predizer a saída seguinte $y(n)$. Já a simulação livre, também referenciada como predição de múltiplos passos à frente, apenas

Figura 3 – Configurações de operação de um modelo.



(a) Um passo à frente. (b) Simulação livre.

Fonte: Adaptada de Nelles (2001).

dispõem das entradas do sistema. A Figura 3 ilustra as distinções entre essas configurações para um caso especial do NARMAX, usualmente empregado na literatura, no qual não se inclui qualquer modelo de dependência do ruído ε, denominado de modelo não linear autorregressivo com entradas exógenas (*non-linear autoregressive moving average with exogenous input*, NARX). Ademais, vale ressaltar que ambas são aplicáveis não só na operação como durante o treinamento do modelo (NELLES, 2001).

Após todas essas definições e procedimentos, o modelo treinado precisa ser validado. Nesse momento, deve-se testar sua capacidade de generalização, ou seja, seu desempenho em relação a dados que não são apresentados durante o treinamento, verificando, assim, se o modelo é capaz de assimilar as características de interesse do sistema. Para mensurar isso, figuras de mérito baseadas no erro quadrático médio, $E[(t(n) - y(n))^2]$, em que $E[\cdot]$ é o operador valor esperado, são empregadas normalmente (AGUIRRE, 2015).

2.4 Conclusão

No decorrer desse Capítulo, são apresentadas de forma breve as principais etapas de identificação de sistemas, a saber, coleta de dados, escolha de modelos paramétricos, seus princípios de estimação e de validação. Além do mais, essa introdução se propõem a reduzir possíveis dubiedades nas terminologias e procedimentos contidos nos próximos capítulos em que se definem as operações propostas para os modelos fundamentados na ESN.

3 ESN PARA IDENTIFICAÇÃO ONLINE E ROBUSTA DE SISTEMAS

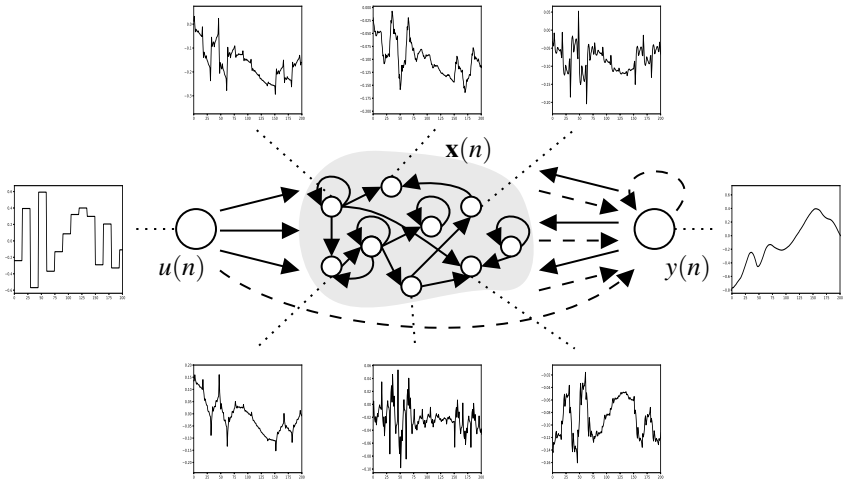
Para melhor fixar as notações, considera-se que a tarefa de identificação de sistemas em tempo discreto necessita da disponibilidade de um conjunto de entradas e de saídas do sistema de interesse: $\{(\mathbf{u}(n),\mathbf{t}(n))\}_{n=1}^{N_{all}}$, em que $\mathbf{u}(n) = (u_1(n),u_2(n),...,u_K(n))^T$ é o vetor de K entradas aplicadas e $\mathbf{t}(n) = (t_1(n),t_2(n),...,t_L(n))^T$ é o vetor de L saídas geradas em um dado instante n do universo de N_{all} amostras. Esse conjunto de dados pode ser dividido entre subconjunto de treino e de teste, $N_{all} = N_{train} + N_{test}$. Após o devido treinamento, o modelo deve ser capaz de inferir um vetor de saída $\mathbf{y}(n) \in \mathbb{R}^{L\times 1}$ que produza o menor desvio possível de $\mathbf{t}(n)$.

Uma das possibilidades para construir modelos de sistemas dinâmicos é a rede neural recorrente, cujos neurônios transmitem sinais de retroalimentação (*feedback*) para si ou para os demais, constituindo uma rede neural artificial complexa e com inúmeras possibilidades de configuração. O arranjo de uma RNN possui o intuito de gerar sinais de estados dos seus neurônios, denotados pelo vetor $\mathbf{x}(n)$, que representem um histórico das entradas apresentadas a rede, ou seja, busca-se construir uma função $\mathbf{F}$, tal que $\mathbf{x}(n) = \mathbf{F}(\mathbf{u}(n),\mathbf{u}(n-1),...)$. À vista disso, as RNNs implementam uma forma diferente de incorporação da dinâmica do sistema quando comparadas aos modelos NARX, em que as recorrências da RNN fornecem informação similar as provindas dos regressores do NARX. Isso possibilita o encurtamento da janela dos regressores para modelos baseado nas RNNs (OZTURK *et al.*, 2007).

Dentre as várias RNNs disponíveis, tais como: Hopfield (HOPFIELD, 1984), Jordan (JORDAN, 1997), Elmam (ELMAN, 1990) e *Long short-term memory* (HOCHREITER; SCHMIDHUBER, 1997), a rede neural de ecos de estados (JAEGER, 2001), nomeada a partir da metáfora de que as suas variáveis de estado $\mathbf{x}(n)$ representam ecos dos seus sinais de entrada, possui uma das mais simples abordagens de treinamento supervisionado. As RNNs normalmente apresentam dificuldade de adaptação dos pesos e vários algoritmos específicos para essas redes são propostos, por exemplo, retropropagação do erro atraves do tempo (*backpropagation through time*) e aprendizado recorrente em tempo real (*real-time recurrent learning*). Entretanto, esses algoritmos experimentam alta complexidade computacional, treinamento lento, possibilidade de instabilidade e decaimento de gradientes através da topologia e do tempo. Normalmente, é necessário aplicar métodos de segunda ordem, como algoritmos baseados no filtro de Kalman, para fornecer um desempenho mais confiável e possibilitar aplicações (GUNDUZ *et al.*, 2007).

Em contraste a isso, tem-se a computação com reservatório. Esse paradigma de RNN, introduzido pela ESN na área de aprendizado de máquina e pela máquina de estado líquido (*liquid*

Figura 4 – Ilustração da estrutura simplificada da ESN. As linhas continuas representam as vias com pesos fixos e as linhas tracejadas representam as vias com pesos ajustáveis.



Fonte: Adaptada de Jaeger (2004).

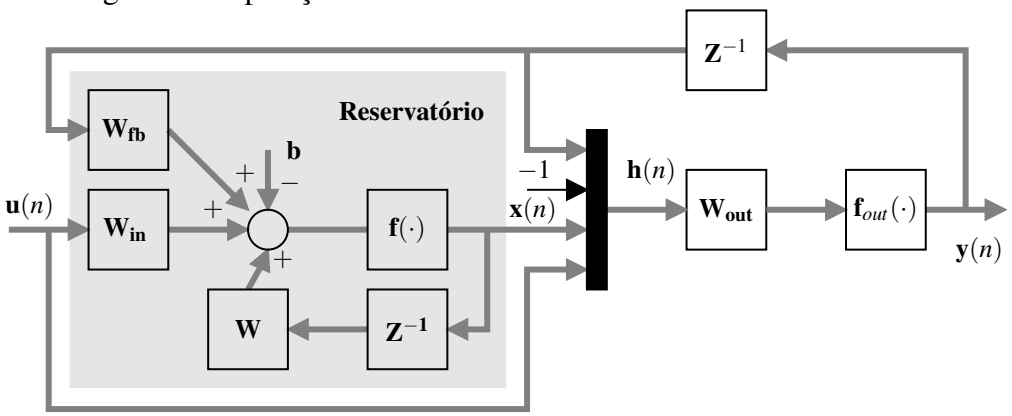
state machines) (MAASS *et al.*, 2002) na área da neurociência computacional, alcança excelentes resultados mesmo sem a completa adaptação dos pesos da rede. Em outras palavras, esse padrão de RNN possui um conjunto de neurônio com pesos fixos e gerados aleatoriamente, chamado de reservatório, e apenas os neurônios pertencentes a camada de saída da rede tem seus pesos ajustados, sendo uma alternativa as redes completamente treinadas (LUKOŠEVIČIUS; JAEGER, 2009). Além disso, é possível trabalhar com a ESN sem janela de atraso das entradas ou das saídas nos seus regressores, sendo uma importante facilidade desse método, já presente desde sua concepção, que é detalhada nesse Capítulo. A estrutura simplificada da ESN é apresentada na Figura 4, onde se adicionam gráficos representativos do comportamento da entrada, da saída e de alguns estados dos neutrônios pertencentes ao reservatório.

Ao longo desse Capítulo, são apresentados os princípios teóricos e construtivos da ESN com o foco no problema de estimação robusta a *outliers*. Inicialmente, os elementos da ESN são apresentados em conjunto com as possíveis abordagens de treinamento. Em seguinda, detalha-se o desenvolvimentos das equações recursivas do algoritmo RLM para a estimação *online* da rede. Por fim, indica-se procedimentos para implementação do treinamento.

3.1 Fundamentos da ESN

A execução normal da ESN na tarefa de simular sistemas é apresentada por meio do diagrama delineado na Figura 5, em sua forma completa nas possíveis vias de conexões entre reservatório, entrada e saída do modelo neural, assim como na sua proposta original (JAEGER,

Figura 5 – Diagrama de operação da ESN.



Fonte: Autoria própria.

2001), onde é destacada a delimitação do reservatório em cinza.

O reservatório da ESN é formado por R neurônios, os quais produzem os sinais de saídas denotados pelas variáveis de estados $\mathbf{x}(n) \in \mathbb{R}^{R\times 1}$, tal que

$$\mathbf{x}(n) = \mathbf{f}(\mathbf{W}\mathbf{x}(n-1) + \mathbf{W}_{in}\mathbf{u}(n) + \mathbf{W}_{fb}\mathbf{y}(n-1) - \mathbf{b}), \tag{3.1}$$

em que $\mathbf{f}(\cdot)$ é uma função de ativação não linear com operação elemento a elemento, normalmente a função tangente hiperbólica ou sigmoide logística; $\mathbf{W} \in \mathbb{R}^{R\times R}$ é a matriz de pesos do reservatório, responsável pela recorrência da rede; $\mathbf{W}_{in} \in \mathbb{R}^{R\times K}$ e $\mathbf{W}_{fb} \in \mathbb{R}^{R\times L}$ são as matrizes de pesos que ligam a entrada e a saída do modelo ao reservatório, respectivamente; e $\mathbf{b} \in \mathbb{R}^{R\times 1}$ é o vetor de *bias*.

A saída do modelo neural é calculada por

$$\mathbf{y}(n) = \mathbf{f}_{out}\left(\mathbf{W}_{out}\mathbf{h}(n)\right), \tag{3.2}$$

na qual $\mathbf{f}_{out}(\cdot)$ é uma função ativação dos neurônios da saída, também com operação elemento a elemento, sendo bastante comum a função identidade; $\mathbf{W}_{out} \in \mathbb{R}^{L\times H}$ é a matriz de pesos da saída; $\mathbf{h}(n) = [-1, \mathbf{u}(n), \mathbf{y}(n-1), \mathbf{x}(n)]^T \in \mathbb{R}^{H\times 1}$ é um vetor concatenado, em que $H = 1+K+R+L$.

3.1.1 Construção do reservatório

Como anteriormente mencionado, os pesos do reservatório, ou seja, os elementos das matrizes $\mathbf{W}$, $\mathbf{W}_{in}$, $\mathbf{W}_{out}$ e do vetor $\mathbf{b}$ têm seus elementos atribuídos de forma aleatória e fixos durante o treinamento e operação da rede. Contudo, existem nuances para a construção do reservatório.

O propósito do reservatório é realizar um mapeamento não linear das suas entradas para um espaço de alta dimensão (i.e. dimensão do vetor $\mathbf{x}(n)$) simultaneamente com um contexto

temporal, isso é, possuir memória das entradas. Essa técnica de expansão dimensional é comum na área de aprendizado de máquina, tendo como objetivo construir um espaço de estados com informação suficiente em $\mathbf{x}(n)$ que resultem em valores próximos da saída desejada de um sistema dinâmico ($\mathbf{t}(n)$) por intermédio de combinações lineares dos seus sinais (LUKOŠEVIČIUS, 2012).

Um ponto importante quando se refere a geração aleatório de valores é qual a distribuição de probabilidade deve ser utilizada. Porém, não há consenso entre os autores sobre a mais adequada, sendo utilizada a distribuição uniforme na maioria dos trabalhos com a ESN. Nessa perspectiva, existem alguns hiperparâmetros a serem definidos para o funcionamento adequado da rede a depender do sistema a ser modelado. O primeiro deles é o número de neurônios do reservatório, R. O desempenho da rede normalmente aumenta quanto maior for o seu reservatório desde que sejam tomadas medidas de regularização para evitar o sobreajuste do modelo no treinamento (LUKOŠEVIČIUS, 2012).

A matriz de recursão $\mathbf{W}$ possui mais dois hiperparâmetros além da sua dimensão dada pelo número de neurônios, a saber, a esparsidade e o raio espectral. Existe a recomendação que $\mathbf{W}$ possua uma restrita porcentagem dos pesos diferentes de zero, o que pode favorecer moderadamente o desempenho e acelerar os cálculos. No entanto, esse parâmetro não é muito sensível e pode variar de valores baixos a 20% das conexões possíveis do reservatório (LUKOŠEVIČIUS; JAEGER, 2009).

O raio espectral de $\mathbf{W}$, $\rho(\mathbf{W})$, é um hiperparâmetro importante da ESN, que se relaciona com a intitulada "propriedade de ecos de estados"(*echo state property*, ESP). Essa propriedade referi-se a faculdade da ESN que ao ser inicializada em dois estados arbitrários e distintos, $\mathbf{x}(0)$ e $\hat{\mathbf{x}}(0)$, e sendo excitada com a mesma sequência de entradas apresenta a convergência entre $\mathbf{x}(n)$ e $\hat{\mathbf{x}}(n)$ ao passar das iterações. Em outras palavras, os estados do reservatório dependem assintoticamente apenas do histórico de entrada (JAEGER, 2003). O $\rho(\mathbf{W})$ é tratado como um parâmetro de escala da matriz e pode ser arbitrariamente atribuído da seguinte de forma:

$$\mathbf{W} = \rho(\mathbf{W}) \left(\frac{1}{|\lambda_{max}(\hat{\mathbf{W}})|} \hat{\mathbf{W}} \right), \tag{3.3}$$

em que $\hat{\mathbf{W}}$ é uma matriz quadrada qualquer e $|\lambda_{max}(\hat{\mathbf{W}})|$ é o seu maior autovalor em valor absoluto. Muitos trabalhos apresentam intervalos desse parâmetro para garantir a ESP para diferentes arquiteturas da ESN. Entretanto, valores de $\rho(\mathbf{W}) < 1$ normalmente asseguram a ESP, mas não é uma condição necessária. Além disso, pode-se otimizar $\mathbf{W}$ por meio de testes, tendo

em mente que seu raio espectral influencia na capacidade de reter memória da ESN, ou melhor, deve-se elevar seu valor quando se deseja aumentar a memória de longo prazo das entradas (é interessante iniciar a busca do valor adequado próximo do círculo unitário) (LUKOŠEVIČIUS, 2012).

Além da memória, um outro aspecto importante de sistemas dinâmicos é a velocidade da resposta à excitação da entrada, ou seja, a rapidez que a entrada influencia a saída do sistema. No caso de alta velocidade, se tem em mãos, por exemplo, o valor de $u(n-1)$ a ser ponderado pela camada de *readout*. Já no caso oposto, em que a dinâmica é muito lenta, há a possibilidade da ESN com integração com vazamento (*leaky integration*), tal que se reformula a Equação 3.1 como

$$\mathbf{x}(n) = (1-\gamma)\mathbf{x}(n-1) + \gamma\mathbf{f}(\mathbf{W}\mathbf{x}(n-1) + \mathbf{W}_{in}\mathbf{u}(n) + \mathbf{W}_{fb}(\mathbf{y}(n-1) - \mathbf{b}), \tag{3.4}$$

onde γ representa a taxa de vazamento do reservatório. Consequentemente, quanto menor essa taxa mais lenta a resposta dinâmica da ESN pode ser obtida (JAEGER *et al.*, 2007).

Por fim, tem-se as especificações dos pesos das conexões de entrada do reservatório, denotados por $\mathbf{W}_{in}$, $\mathbf{W}_{fb}$ e $\mathbf{b}$, que podem ser tratadas, por exemplo, por meio de um valor de escala a qualquer, que representa o intervalo de $[-a,a]$ em que uma distribuição uniforme é amostrada. Os valores de escala muito altos provocam sinais de ativação dos neurônios próximos a saturação de suas funções sigmoidais, resultando no aumento da não linearidade do modelo. Ademais, a elevação do $\rho(\mathbf{W})$ também impacta de maneira similar, mas pode tornar a rede instável antes de alcançar a não linearidade desejada. Outrossim, é importante destacar a importância da normalização, escala e deslocamento, das amostras das entradas e das saídas do sistema para facilitar a atribuição desses parâmetros. Mais detalhes sobre as características do reservatório podem ser encontrados em (LUKOŠEVIČIUS, 2012).

3.1.2 Treinamento da ESN

Após feita uma escolha inicial dos hiperparâmetros do reservatório, a etapa seguinte é o treinamento da ESN. Na abordagem ordinária da ESN, o seu treinamento consiste na estimação dos elementos de $\mathbf{W}_{out}$, que tem o papel de realizar uma combinação linear de $\mathbf{h}(n)$ para resultar na saída $\mathbf{y}(n)$. Para tanto, é necessário uma coleção de valores de $\mathbf{h}(n)$, produzidos a partir da operação da rede com suas devidas entradas, e de seus respectivos valores de saída desejado $\mathbf{t}(n)$,

que podem ser representados pela equação matricial:

$$\mathbf{W}_{out}\mathbf{H} = \mathbf{T}, \tag{3.5}$$

na qual a matriz $\mathbf{H} = [\mathbf{h}(1), \mathbf{h}(2), ..., \mathbf{h}(N_{train})]$, $\mathbf{H} \in \mathbb{R}^{H \times N_{train}}$, e a matriz $\mathbf{T} = [\mathbf{t}(1), \mathbf{t}(2), ..., \mathbf{t}(N_{train})]$, $\mathbf{T} \in \mathbb{R}^{L \times N_{train}}$, sendo formadas com os dados separados para o treinamento. Logo, considera-se tipicamente encontrar um $\mathbf{W}_{out}$ que minimize os erros quadráticos, ou seja, minimizar a função custo $J_{OLS} = \sum_{i=1}^{N_{train}} \mathbf{e}(i)^T \mathbf{e}(i)$, onde $\mathbf{e}(n) = \mathbf{t}(n) - \mathbf{W}_{out}\mathbf{h}(n)$, sendo $\mathbf{e}(n) \in \mathbb{R}^{L \times 1}$ o vetor de erros das saídas. Uma solução numericamente estável para esse problema pode ser formulada aplicado a regularização de Tikhonov, o que resulta na expressão:

$$\mathbf{W}_{out} = \mathbf{T}\mathbf{H}^T(\mathbf{H}\mathbf{H}^T + \delta\mathbf{I})^{-1}, \tag{3.6}$$

em que $\mathbf{I} \in \mathbb{R}^{H \times H}$ é uma matriz identidade e δ é um parâmetro de regularização, que pondera a penalização da norma de $\mathbf{W}_{out}$, diminuindo a sensibilidade ao ruído e o sobreajuste do modelo, além do benefício da estabilidade numérica (LUKOŠEVIČIUS; JAEGER, 2009).

Contudo, quando o vetor $\mathbf{h}(n)$ contém a realimentação da saída da ESN (diretamente como um dos seus elementos ou indiretamente através do reservatório), sendo o caso de sistemas dinâmicos com entrada e saída, no qual se opta pela realimentação da saída, ou em séries temporais, em que é imprescindível a realimentação, essa solução em batelada descrita precisa de um valor que desempenhe a função de uma saída anterior. Para tanto, o método chamado de *teacher forcing* (algo como professor forçante, em português) possibilita a estimação de $\mathbf{W}_{out}$ ao romper o laço de realimentação durante o treinamento, substituindo a saída anterior $\mathbf{y}(n-1)$ da ESN pelo o valor desejado da iteração anterior $\mathbf{t}(n-1)$ nas Equações 3.1 e 3.2. Um outro ponto dessa abordagem é o descarte dos primeiros dados de treinamento motivado pelo transiente inicial do reservatório da ESN e pelas condições iniciais arbitrada para $\mathbf{x}(0)$, normalmente se atribui elementos nulos na inicialização, mas nem sempre é possível devido a aplicação ou quantidade de dados disponíveis (LUKOŠEVIČIUS, 2012).

O método *teacher forcing* também é uma abordagem utilizada em algoritmos de estimação *online*, em que são combinados com o algoritmo RLS (JAEGER, 2003) ou com o algoritmo KRLS (ZHOU *et al.*, 2018), por exemplo. Porém, a disponibilidade de uma previsão, mesmo que imperfeita, da saída da rede a cada iteração possibilita a implementação de algoritmos de estimação especializados na operação com realimentação real. Seguindo essa linha, dois métodos podem ser citados: a retropropagação do erro descorrelacionada (*backpropagation-decorrelation*, BPDC) (STEIL, 2004) e o aprendizado FORCE (SUSSILLO; ABBOTT, 2009).

A BPDC possui baixa complexidade computacional, baixa sensibilidade aos hirparâmetros da rede e é bastante eficiente em se adaptar às mudanças no comportamento do sistema. Todavia, o algoritmo é enviesado pelos dados mais recentes em detrimentos dos dados passados, o que pode ser uma desvantagem dependendo da aplicação. O aprendizado FORCE é baseado no RLS, entretanto, possui um mecanismo que recalcula a saída da ESN a posteriori da estimação, ou melhor, da atualização dos pesos na iteração, tornando o valor previsto mais próximo do desejado. Ademais, esse método apresenta excelentes resultados na construção de geradores de padrões neurais, sendo bastante estáveis e precisos (LUKOŠEVIČIUS *et al.*, 2012).

Em seguida, é detalhado o método *teacher forcing* e o algoritmo RLM com o intuito de fundamentar o treinamento *online* e robusto da ESN proposto para a identificação de sistemas na presença de *outlier*.

3.2 Treinamento *online* e robusto com RLM

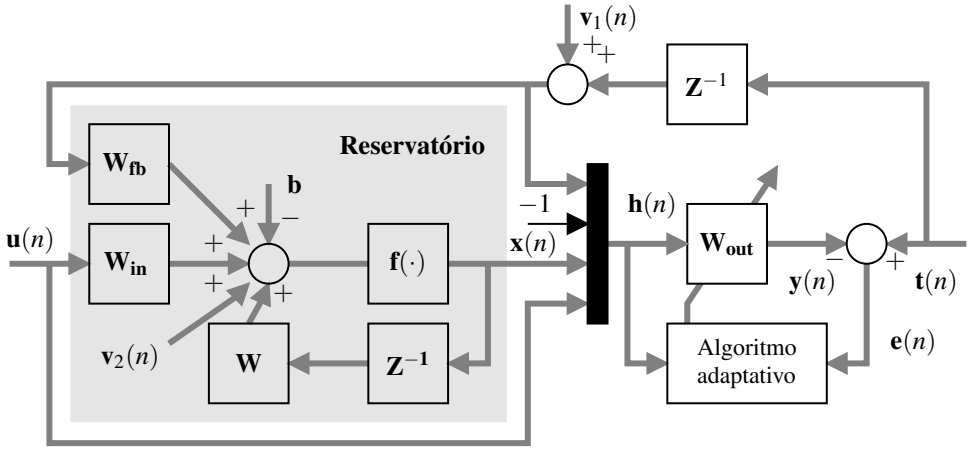
Um algoritmo clássico, simples e eficiente de estimação adaptativa é o dos mínimos médios quadrados (*least mean squares*, LMS) (HAYKIN, 2014). Entretanto, sendo um método de gradiente descendente de primeira ordem, norteado apenas pelo erro instantâneo, o seu desempenho deixa a desejar quando a superfície de curvatura do erro é muito diferente em direções distintas, o que é representado por uma alta variação entre os autovalores de $\mathbf{H}\mathbf{H}^T$. Uma alternativa ao LMS é o algoritmo RLS, que não é sensibilizado pelo espalhamento dos autovalores e apresenta uma convergência mais rápida por ser um método de segunda ordem, apesar de ser computacionalmente mais dispendioso (LUKOŠEVIČIUS; JAEGER, 2009). Dito isso, a escolha natural para tornar o treinamento da ESN robusto a *outlier* é o algoritmo RLM (ZOU *et al.*, 2000; ZOU *et al.*, 2001; CHAN; ZOU, 2004), em razão de ser um método baseado no algoritmo RLS..

3.2.1 Aprendizado por teacher forcing

A estrutura para realizar o aprendizado da ESN *online* junto a abordagem *teacher forcing* é apresentada na Figura 6.

Ao se contrastar com o diagrama presente na Figura 5, o diagrama da Figura 6 apresenta a adição de elementos, a saber, um bloco funcional representando o algoritmo adaptativo, o sinal $\mathbf{t}(n)$ referente ao valor desejado da saída e dois vetores opcionais para injeção de ruído:

Figura 6 – Diagrama de treinamento *online* da ESN.



Fonte: Autoria própria.

$\mathbf{v}_1 \in \mathbb{R}^{L\times 1}$ na saída desejada atrasada e $\mathbf{v}_2 \in \mathbb{R}^{R\times 1}$ nos estados do reservatório. Logo, durante o treinamento, as Equações 3.1 e 3.2 devem ser reformuladas como

$$\mathbf{x}(n) = \mathbf{f}(\mathbf{W}\mathbf{x}(n-1) + \mathbf{W}_{in}\mathbf{u}(n) + \mathbf{W}_{back}(\mathbf{t}(n-1) + \mathbf{v}_1(n)) + \mathbf{v}_2(n) - \mathbf{b}) \tag{3.7}$$

e

$$\mathbf{y}(n) = \mathbf{W}_{out}\mathbf{h}(n), \tag{3.8}$$

em que $\mathbf{h}(n)$ corresponde agora ao vetor $[-1, \mathbf{u}(n), \mathbf{t}(n-1) + \mathbf{v}_1(n), \mathbf{x}(n)]^T$ e atribui-se a função identidade à função $\mathbf{f}_{out}(\cdot)$ da Equação 3.2 a partir desse momento. As grandezas dos vetores de ruído $\mathbf{v}_1(n)$ e $\mathbf{v}_2(n)$ devem ser escolhidos de forma a balancear estabilidade e precisão do modelo, inclusive ajudando na regularização da solução.

3.2.2 Formulação do Algoritmo RLM

Com a estrutura definida, é necessário dispor das equações que regem a atualização dos pesos da camada de saída da ESN com base nos valores de $\mathbf{h}(n)$ e da medida do erro $\mathbf{e}(n)$ a cada iteração. Por conveniência, considera-se o treinamento para um sistema de saída única, em outras palavras, descreve-se a estimação *online* do vetor da camada de *readout* $\mathbf{w}_{out} \in \mathbb{R}^{H\times 1}$ referente a uma certa saída. Tendo em mente que um sistema MIMO pode ser decomposto em sistemas de múltiplas entradas e saída única sem perda de generalidade. Posteriormente, são feitas considerações para um sistema MIMO.

O algoritmo RLS possui a função custo $J_{LS}(n) = \sum_{i=1}^{n} \lambda^{n-i} e^2(i)$ a ser minimizada, onde $0 < \lambda \leq 1$ é o fator de esquecimento e $e(n) = t(n) - \mathbf{w}_{out}^T(n-1)\mathbf{h}(n)$ é o erro do modelo

no instante n, sendo uma função bastante sensível à ruído não gaussiano. Já o algoritmo RLM apresenta sua função custo, definida por

$$J_\rho(n) = \sum_{i=1}^{n} \lambda^{n-i} \rho(e(i)), \tag{3.9}$$

onde $\rho(e)$ é uma função estimador-M, que tem o propósito de limitar o efeito negativo causado por erros muito grandes provindos de uma amostra com ruídos impulsivos. Contudo, pode-se considerar o algoritmo RLS sendo um caso específico do RLM quando $\rho(e) = e^2$. Por esse motivo, o desenvolvimento da equações recursivas é apresentado para o caso geral, Equação 3.9.

3.2.2.1 Equacionamento

Os pesos de saída ótimos podem ser determinados derivando $J_\rho(n)$ em relação a $\mathbf{w}_{out}$ e igualando a zero:

$$\sum_{i=1}^{n} \lambda^{n-i} \psi(e(i)) \mathbf{h}(i) = \sum_{i=1}^{n} \lambda^{n-i} q(e(i)) e(i) \mathbf{h}(i) = 0, \tag{3.10}$$

em que $\psi(e) = \partial\rho(e)/\partial e$ e $q(e) = \psi(e)/e$. Consequentemente, tem-se que

$$\sum_{i=1}^{n} \lambda^{n-i} q(e(i)) \mathbf{h}(i) \mathbf{h}^T(i) \mathbf{w}_{out}(n) = \sum_{i=1}^{n} \lambda^{n-i} q(e(i)) t(i) \mathbf{h}(i), \tag{3.11}$$

podendo ser rescrita na forma:

$$\mathbf{R}(n)\mathbf{w}_{out}(n) = \mathbf{p}(n) \Rightarrow \mathbf{w}_{out}(n) = \mathbf{R}^{-1}(n)\mathbf{p}(n), \tag{3.12}$$

onde

$$\begin{aligned} \mathbf{R}(n) &= \sum_{i=1}^{n} \lambda^{n-i} q(e(i)) \mathbf{h}(i) \mathbf{h}^T(i) \\ &= \lambda \mathbf{R}(n-1) + q(e(n)) \mathbf{h}(n) \mathbf{h}^T(n) \end{aligned} \tag{3.13}$$

e

$$\begin{aligned} \mathbf{p}(n) &= \sum_{i=1}^{n} \lambda^{n-i} q(e(i)) t(i) \mathbf{h}(i) \\ &= \lambda \mathbf{p}(n-1) + q(e(n)) t(n) \mathbf{h}(n), \end{aligned} \tag{3.14}$$

sendo $\mathbf{R}(n) \in \mathbb{R}^{H \times H}$ a matriz de correlação de $\mathbf{h}(n)$ e $\mathbf{p}(n) \in \mathbb{R}^{H \times 1}$ o vetor de correlação cruzada entre $\mathbf{h}(n)$ e $t(n)$.

Assumindo que a matriz de correlação $\mathbf{R}(n)$ é não singular e inversível, pode-se aplicar o lema da inversão de matriz, $(\mathbf{A} + \mu\mathbf{x}\mathbf{y}^T)^{-1} = \mathbf{A}^{-1}\{\mathbf{I} - [(\mu\mathbf{x}\mathbf{y}^T\mathbf{A}^{-1})/(1 + \mu\mathbf{y}^T\mathbf{A}^{-1}\mathbf{x})]\}$,

à Equação 3.13 a fim de construir uma regra iterativa da sua atualização para cada nova entrada. Logo, fazendo $\mathbf{A} = \lambda\mathbf{R}(n-1)$, $\mathbf{x} = \mathbf{y} = \mathbf{h}(n)$, $\mu = q(e(n))$ e considerando

$$\mathbf{S}(n) = \mathbf{R}^{-1}(n), \tag{3.15}$$

encontra-se a seguinte expressão:

$$\mathbf{S}(n) = \lambda^{-1}\mathbf{S}(n-1) - \frac{\lambda^{-2}q(e(n))\mathbf{S}(n-1)\mathbf{h}(n)\mathbf{h}^T(n)\mathbf{S}(n-1)}{1+\lambda^{-1}q(e(n))\mathbf{h}^T(n)\mathbf{S}(n-1)\mathbf{h}(n)}. \tag{3.16}$$

Por conveniência, defini-se um vetor $\mathbf{k}(n) \in \mathbb{R}^{H\times 1}$ como sendo

$$\mathbf{k}(n) = \frac{\lambda^{-1}q(e(n))\mathbf{S}(n-1)\mathbf{h}(n)}{1+\lambda^{-1}q(e(n))\mathbf{h}^T(n)\mathbf{S}(n-1)\mathbf{h}(n)}. \tag{3.17}$$

Assim, a Equação 3.16 pode ser reescrita como

$$\mathbf{S}(n) = \lambda^{-1}\mathbf{S}(n-1) - \lambda^{-1}\mathbf{k}(n)\mathbf{h}^T(n)\mathbf{S}(n-1). \tag{3.18}$$

Bem como, o arranjo da Equação 3.29 pode ser manipulado para se obter:

$$\begin{aligned}\mathbf{k}(n) &= \lambda^{-1}q(e(n))\mathbf{S}(n-1)\mathbf{h}(n) - \lambda^{-1}q(e(n))\mathbf{k}(n)\mathbf{h}^T(n)\mathbf{S}(n-1)\mathbf{h}(n)\\ &= q(e(n))[\lambda^{-1}\mathbf{S}(n-1) - \lambda^{-1}q(e(n))\mathbf{k}(n)\mathbf{h}^T(n)\mathbf{S}(n-1)]\mathbf{h}(n)\\ &= q(e(n))\mathbf{S}(n)\mathbf{h}(n).\end{aligned} \tag{3.19}$$

Ou seja, a Equação 3.18 representa a regra de atualização recursiva da inversa da matriz de correlação $\mathbf{R}(n)$ e $\mathbf{k}(n)$ pode ser interpretado como um vetor de ganho.

Agora, com a finalidade de implementar a regra de atualização *online* dos pesos da camada de saída da rede, pode-se definir $\mathbf{w}_{out}(n)$ em consequência das Equações 3.12, 3.14 e 3.15 como

$$\begin{aligned}\mathbf{w}_{out}(n) &= \mathbf{R}^{-1}(n)\mathbf{p}(n)\\ &= \mathbf{S}(n)\mathbf{p}(n)\\ &= \lambda\mathbf{S}(n)\mathbf{p}(n-1) + q(e(n))\mathbf{S}(n)\mathbf{h}(n)t(n).\end{aligned} \tag{3.20}$$

Em seguida, substituindo $\mathbf{S}(n)$ pela Equação 3.18 na Equação 3.20, encontra-se que

$$\begin{aligned}\mathbf{w}_{out}(n) &= \mathbf{S}(n-1)\mathbf{p}(n-1) - \mathbf{k}(n)\mathbf{h}^T(n)\mathbf{S}(n-1)\mathbf{p}(n-1) + q(e(n))\mathbf{S}(n)\mathbf{h}(n)t(n)\\ &= \mathbf{w}_{out}(n-1) - \mathbf{k}(n)\mathbf{h}^T(n)\mathbf{w}_{out}(n-1) + q(e(n))\mathbf{S}(n)\mathbf{h}(n)t(n).\end{aligned} \tag{3.21}$$

Portanto, pode-se deduzir a equação de atualização dos pesos ao aplicar a definição de $\mathbf{k}(n)$ da Equação 3.19 na Equação 3.21, ou seja,

$$\begin{aligned}\mathbf{w}_{out}(n) &= \mathbf{w}_{out}(n-1) - \mathbf{k}(n)\mathbf{h}^T(n)\mathbf{w}_{out}(n-1) + \mathbf{k}(n)t(n)\\ &= \mathbf{w}_{out}(n-1) + \mathbf{k}(n)[t(n) - \mathbf{h}^T(n)\mathbf{w}_{out}(n-1)]\\ &= \mathbf{w}_{out}(n-1) + \mathbf{k}(n)e(n).\end{aligned} \tag{3.22}$$

Em síntese, o algoritmo RLM é estabelecido pela iteração das Equações 3.29, 3.18 e 3.22, as quais são responsáveis pela atualização do vetor $\mathbf{k}(n)$, $\mathbf{S}(n)$ e $\mathbf{w}_{out}(n)$, respetivamente.

3.2.2.2 Funções estimador-M

Com o objetivo de totalizar a concepção teórica, apresentam-se algumas funções custo $\rho(e)$ e suas correspondentes funções de ponderação $q(e)$ na Tabela 1.

A primeira função da lista é o OLS que corresponde ao caso do algoritmo RLS. Já as seguintes são funções estimador-*M*, as quais são encarregadas de propiciar robustez à abordagem do RLM. Aliás, essas apresentam limiares que regulam a influencia do erro na estimação, podendo ser atribuídos em dependência de uma previsão do desvio padrão $\hat{\sigma}$ de uma distribuição normal do erro livre de ruído impulsivo. Ou seja, o objetivo é atenuar o efeito de erros que possuam valores não condizentes com uma distribuição gaussiana.

O desvio padrão $\hat{\sigma}$ pode ser calculado *online*, simultaneamente com $\mathbf{w}_{out}$, mediante um estimador de variância $\hat{\sigma}^2(n)$, expressado por:

$$\hat{\sigma}^2(n) = c_1 med\{e^2(n), ..., e^2(n-N_w+1)\}, \tag{3.23}$$

onde $c_1 = 1.483(1+5/(N_w-1))$, $med\{\cdot\}$ é a mediana amostral e N_w é o número de amostras ou comprimento da janela. Quanto maior o valor de N_w, maior a acurácia e a complexidade computacional (ROUSSEEUW; LEROY, 1987). Ademais, esse estimador pode ser modificado, resultando em

$$\hat{\sigma}^2(n) = \lambda_{\hat{\sigma}}\hat{\sigma}^2(n-1) + (1-\lambda_{\hat{\sigma}})c_1 med\{e^2(n), ..., e^2(n-N_w+1)\}, \tag{3.24}$$

em que se adiciona um fator de esquecimento $0 < \lambda_\sigma \leq 1$. Em consequência, a natureza recursiva dessa nova formulação torna a janela amostral infinita, deixando a aferição de $\hat{\sigma}^2(n)$ mais estável (ZOU *et al.*, 2000).

É interessante apresentar algumas considerações sobre os limiares. Na função de Huber, a faixa de zero a $1.345\hat{\sigma}$ corresponde ao intervalo de confiança de aproximadamente 90%, assim, essa porcentagem é referente aos erros provindos de um suposta distribuição gaussiana limpa de ruídos impulsivos que são tratados sem ponderação, $q(e) = 1$, ao passo que o restante dos erros é ponderado por $q(e) \to 0$ conforme $|e| \to \infty$. Com o paradigma diferente, a função de Huber modificada exclui a amostra que gere um erro que não corresponda ao intervalo de confiança de 99%, referente à faixa de zero a $2.576\hat{\sigma}$, atribuindo diretamente $q(e) = 0$. Dessa

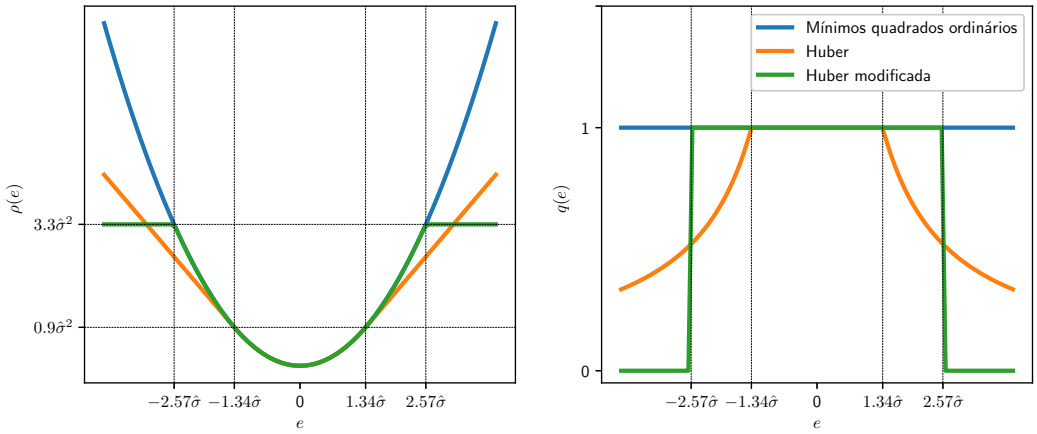
Tabela 1 – Lista de funções custo ($\rho(e)$) e suas respectivas funções de ponderação ($q(e)$).

Nome	**Função custo** ($\rho(e)$)**/ Função peso** ($q(e)$)	**Limiar**
OLS	$\rho(e) = \frac{e^2}{2}$ $q(e) = 1$	
Huber	$\rho(e) = \begin{cases} \frac{e^2}{2}, & 0 < \lvert e\rvert < \xi \\ \xi\lvert e\rvert - \frac{\xi^2}{2}, & \xi \leqslant \lvert e\rvert \end{cases}$ $q(e) = \begin{cases} 1, & 0 < \lvert e\rvert < \xi \\ \frac{\xi}{\lvert e\rvert}, & \xi \leqslant \lvert e\rvert \end{cases}$	$\xi = 1.345\hat{\sigma}$
Huber modificada	$\rho(e) = \begin{cases} \frac{e^2}{2}, & 0 < \lvert e\rvert < \xi \\ \frac{\xi^2}{2}, & \xi \leqslant \lvert e\rvert \end{cases}$ $q(e) = \begin{cases} 1, & 0 < \lvert e\rvert < \xi \\ 0, & \xi \leqslant \lvert e\rvert \end{cases}$	$\xi = 2.576\hat{\sigma}$
Hampel	$\rho(e) = \begin{cases} \frac{e^2}{2}, & 0 < \lvert e\rvert < \xi \\ \xi\lvert e\rvert - \frac{\xi^2}{2}, & \xi \leqslant \lvert e\rvert < \Delta_1 \\ \frac{\xi}{2}(\Delta_1 + \Delta_2) - \frac{\xi^2}{2} + \frac{\xi(\lvert e\rvert - \Delta_2)^2}{2(\Delta_1 - \Delta_2)}, & \Delta_1 \leqslant \lvert e\rvert < \Delta_2 \\ \frac{\xi}{2}(\Delta_1 + \Delta_2) - \frac{\xi^2}{2}, & \Delta_2 \leqslant \lvert e\rvert \end{cases}$ $q(e) = \begin{cases} 1, & 0 < \lvert e\rvert < \xi \\ \frac{\xi}{\lvert e\rvert}, & \xi \leqslant \lvert e\rvert < \Delta_1 \\ \frac{\xi(\lvert e\rvert - \Delta_2)}{\lvert e\rvert(\Delta_1 - \Delta_2)}, & \Delta_1 \leqslant \lvert e\rvert < \Delta_2 \\ 0, & \Delta_2 \leqslant \lvert e\rvert \end{cases}$	$\xi = 1.96\hat{\sigma}$ $\Delta_1 = 2.24\hat{\sigma}$ $\Delta_2 = 2.576\hat{\sigma}$

Fonte: Zou *et al.* (2000), Zou *et al.* (2001), Chan e Zou (2004).

mesma maneira, a função de Hampel exclui da estimação dados que provocam erros fora do intervalo de confiança de 99%. Porém, apresenta uma transição suave entre não ponderar e

Figura 7 – Curvas das funções estimador-M.



Fonte: Autoria própria.

suprimir, sendo consideradas faixas como $[1.96\hat{\sigma}, 2.24\hat{\sigma}]$, representando a probabilidade de ocorrência entre 2.5% a 5% do valor de erro na dada distribuição, e $[2.24\hat{\sigma}, 2.576\hat{\sigma}]$, referente à probabilidade de ocorrência de 1% a 2.5%. Por fim, a Figura 7 apresenta as curvas dessas funções com exceção da função de Hampel.

3.2.2.3 Inicialização das variáveis do algoritmo RLM e pseudocódigo

Seguindo para a implementação do treinamento da rede, por causa das equações recursivas, é necessário inicializar as variáveis iterativas. Normalmente, atribui-se valores nulos para o estado inicial do reservatório, $\mathbf{x}(0) = \mathbf{0}$, para saída passada na primeira iteração, $t(0) = 0$, e para os pesos da camada de saída da rede, $\mathbf{w}_{out}(0) = \mathbf{0}$. Entretanto, a matriz de correlação inversa de $\mathbf{h}(n)$ não pode ser iniciar com uma matriz nula, $\mathbf{S}(0) \neq \mathbf{0}$.

Uma forma de embasar a escolha de $\mathbf{S}(0)$ é adicionando informação a *priori* na função de custo apresentada na Equação 3.9 para aprimorar o mapeamento entre entrada e saída, podendo ser expandida como

$$J_{LS}(n) = \sum_{i=1}^{n} \lambda^{n-i} e^2(i) + \delta\lambda^n \parallel \mathbf{w}_{out}(n) \parallel^2, \tag{3.25}$$

em que a notação $\parallel \star \parallel^2$ representa a norma Euclidiana (l_2) ao quadrado e δ é um número real positivo, sendo um parâmetro de regularização dessa norma. Além disso, a adição do termo

$\delta\lambda^n \parallel \mathbf{w}_{out}(n) \parallel^2$ equivale a reformulação da Equação 3.13 para

$$\begin{aligned}\mathbf{R}(n) &= \sum_{i=1}^{n} \lambda^{n-i} q(e(i))\mathbf{h}(i)\mathbf{h}^T(i) + \delta\lambda^n \mathbf{I} \\ &= \lambda \left[\sum_{i=1}^{n-1} \lambda^{n-i} q(e(i))\mathbf{h}(i)\mathbf{h}^T(i) + \delta\lambda^{n-1}\mathbf{I} \right] + q(e(n))\mathbf{h}(n)\mathbf{h}^T(n),\end{aligned} \tag{3.26}$$

assim, essa matriz de correlação se torna não singular desde a interação $n = 0$, ou seja, $\mathbf{R}(0) = \delta\mathbf{I}$. Consequentemente, tem-se que $\mathbf{S}(0) = \delta^{-1}\mathbf{I}$. Entretanto, quando o fator de esquecimento λ é menor que a unidade, $\lambda^n \to 0$ para valores elevados de n, eliminando o efeito dessa regularização com o tempo. E no caso de $\lambda = 1$, se tem o mesmo efeito da regularização de Tikhonov (HAYKIN, 2014).

Ainda sobre a regularização de $\mathbf{w}_{out}$, vale lembrar que a inserção de ruído durante o treinamento como descrito na Figura 6, representado por $\mathbf{v}_1$ e $\mathbf{v}_2$, evita que os pesos de $\mathbf{w}_{out}$ se elevem ao ponto da ESN alcançar uma zona de instabilidade numérica. Logo, os vetores de ruídos citados são relevantes, principalmente quando se lida com conjuntos de dados extensos ou com estimação adaptativa, tendo o fator de esquecimento menor que a unidade (JAEGER, 2003; LUKOŠEVIČIUS; JAEGER, 2009).

Após feitas as considerações necessárias, apresenta-se o pseudocódigo referente à estimação de $\mathbf{w}_{out}$, no Algoritmo 1, de forma sucinta em que alguns elementos são omitidos.

3.2.2.4 Discussão sobre sistemas com múltiplas saídas

Por fim, o último pormenor na implementação do treinamento que utiliza o algoritmo RLS ou RLM é referente ao tratamento de sistema com múltiplas saídas, mais especificamente na composição da inversa da matriz covariância $\mathbf{S}$ quando há mais de uma saída. Considerando um sistema com L saída, os pesos da camada de *readout* são representados pela matriz $\mathbf{W}_{out} \in \mathbb{R}^{L\times H}$ como anteriormente definida. Logo, para o cenário em que se aplica o RLS, isto é, $q(e) = 1$ para qualquer valor de e, as equações de atualização dos pesos podem ser reescritas como:

$$\mathbf{k}(n) = \frac{\lambda^{-1}\mathbf{S}(n-1)\mathbf{h}(n)}{1 + \lambda^{-1}\mathbf{h}^T(n)\mathbf{S}(n-1)\mathbf{h}(n)}, \tag{3.27}$$

$$\mathbf{S}(n) = \lambda^{-1}\left(\mathbf{I} - \mathbf{k}(n)\mathbf{h}^T(n)\right)\mathbf{S}(n-1) \tag{3.28}$$

Algoritmo 1: Pseudocódigo do treinamento da ESN utilizando o algoritmo RLM

Entrada: δ, $\{(\mathbf{u}(n),t(n))\}_{n=1}^{N_{train}}$, $\mathbf{W}$, $\mathbf{W}_{fb}$, $\mathbf{W}_{in}$, $\mathbf{b}$, λ.
Saída: $\mathbf{w}_{out}(N_{train})$
Resultado: Estimação do vetor $\mathbf{w}_{out}$
início
  $t(0) \leftarrow 0$; $\mathbf{x}(0) \leftarrow \mathbf{0}$; $\mathbf{w}_{out}(0) \leftarrow \mathbf{0}$; $\mathbf{S}(0) \leftarrow \delta^{-1}\mathbf{I}$;
  para $n = 1 : N_{train}$ **faça**
    # Cálculo da execução da ESN
    $\mathbf{x}(n) \leftarrow \mathbf{f}(\mathbf{W}\mathbf{x}(n-1) + \mathbf{W}_{in}\mathbf{u}(n) + \mathbf{W}_{fb}t(n-1) - \mathbf{b})$;
    $\mathbf{h}(n) \leftarrow [-1, \mathbf{u}(n), t(n-1), \mathbf{x}(n)]^T$;
    $y(n) \leftarrow \mathbf{w}_{out}^T(n-1)\mathbf{h}(n)$;
    # Cálculo do erro de previsão da ESN
    $e(n) \leftarrow t(n) - y(n)$;
    # Cálculo da atualização dos pesos pelo RLM
    $\mathbf{k}(n) \leftarrow \dfrac{\lambda^{-1}q(e(n))\mathbf{S}(n-1)\mathbf{h}(n)}{1+\lambda^{-1}q(e(n))\mathbf{h}^T(n)\mathbf{S}(n-1)\mathbf{h}(n)}$;
    $\mathbf{S}(n) \leftarrow \lambda^{-1}\left(\mathbf{I} - \mathbf{k}(n)\mathbf{h}^T(n)\right)\mathbf{S}(n-1)$;
    $\mathbf{w}_{out}(n) \leftarrow \mathbf{w}_{out}(n-1) + \mathbf{k}(n)e(n)$;
  fim
fim

e

$$\mathbf{W}_{out}(n) = \mathbf{W}_{out}(n-1) + \mathbf{k}(n)\mathbf{e}^T(n), \tag{3.29}$$

onde $\mathbf{e}(n) \in \mathbb{R}^{L\times 1}$ representa o vetor de erros das saídas do sistema.

Entretanto, caso se queria aplicar as regularizações distintas ou mesmo diferentes fatores de esquecimento para cada saída do sistema, faz-se necessário estimar as linhas da matriz $\mathbf{W}_{out}$ separadamente. Essa objeção também recai para o caso do algoritmo RLM, pois as estimativas dos limiares são espécificos para cada saída e se supondo que não existe dependência na ocorrência de *outlier* entre as saídas, ou seja, deve ser especificado um ganho $\mathbf{k}_j$, uma matriz $\mathbf{S}_j$ e estimativa de variância do erro $\hat{\sigma}_j^2$ referente a uma saída j qualquer, para $j = 1,2,...,L$. Dessa forma, pode-se representar $\mathbf{W}_{out}(n) = [\mathbf{w}_{out1}^T(n), \mathbf{w}_{out2}^T(n), ..., \mathbf{w}_{outL}^T(n)]^T$.

3.3 Conclusão

Nesse Capítulo, expõe-se a estrutura generalista da ESN e o arcabouço para seu treinamento *online* e robusto para tarefa de identificação de sistemas. Disponibiliza-se, também, a fundamentação teórica para uma boa interpretação da metodologia dos experimentos computacionais descrita no Capítulo seguinte.

4 MATERIAIS E MÉTODOS

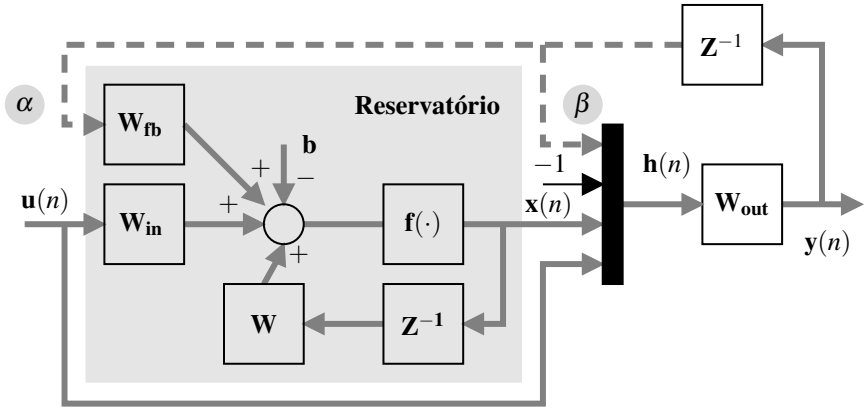
Com a intenção de aferir o desempenho da ESN na tarefa de identificação de sistemas na presença de *outlier*, são realizados experimentos computacionais em que se busca modelar diferentes sistemas a partir de amostras de suas entradas e saídas. Mais especificamente, dispõem-se de três sistemas SISO, um SIMO e um MIMO, em que esses apresentam um variado número de amostras e diferente perfis de sinais de acionamento.

Esses conjuntos de dados têm suas amostras divididas em treino, para estimação dos parâmetros, e teste, para validação o modelo, da ESN. Além disso, os dados referentes às saídas dos sistemas, pertencentes ao treinamento, são contaminadas em diferentes proporções por *outlier*, ocorrendo em 5%, 10% e 15%. Isso emula o caso em que se sabe exatamente quais entradas são aplicadas e se realiza a leitura apenas das saídas, que são passíveis de apresentarem ruído na mensuração. Essa contaminação gradual possibilita aferir o comportamento dos efeitos deletérios causados no desempenho do modelo neural. Por outro lado, busca-se minimizar essas consequências analisando diferentes arquiteturas e algoritmos de estimação durante os testes e em parte do treinamento.

Para realizar as simulações descritas, utiliza-se a plataforma livre de computação científica *Anaconda*, desenvolvida em linguagem *Python*. Dentre suas bibliotecas, destaca-se o pacote *Numpy*, que possibilita manipulação de arranjos, vetores e matrizes, contando com várias funções matemáticas já implementadas, e o pacote *Matplotlib*, que proporciona a elaboração de gráficos de alta qualidade, sendo característico em ambos a sintaxe análoga ao *software* proprietário *Matlab*. Além disso, o ambiente de desenvolvimento integrado escolhido é o *PyCharm* em sua versão gratuita.

O cerne desse Capítulo é a especificação dos experimentos realizados. Para tanto, são descritas as arquiteturas utilizadas da ESN em relação as possíveis vias de realimentação da saída e suas respectivas nomenclaturas na Secção 4.1. Na Secção 4.2, detalham-se as implementações realizadas dos algoritmos de estimação, a saber, o RLS, o RLM e uma proposta de treinamento baseada no algoritmo RLM. Por fim, defini-se a metodologia empregada nas simulações, ou seja, número de repetições de experimentos, sistemática de contaminação, métricas de desempenho, entre outras configurações, na Secção 4.3.

Figura 8 – Diagrama para visualização das arquiteturas da ESN. As linhas tracejas são vias opcionais.



Fonte: Autoria própria.

Tabela 2 – Arquiteturas da ESN referentes às vias de realimentação da saída.

Arquitetura ESN	α	β	**Função de Saída do Modelo ($\mathbf{y}(n) = \mathbf{W}_{out}\mathbf{h}(n)$)**
A0	0	0	$\mathbf{y}(n) = \mathbf{W}_{out}[-1, \mathbf{u}(n), \mathbf{f}(\mathbf{u}(n), \mathbf{u}(n-1), ...)]^T$
A1	0	1	$\mathbf{y}(n) = \mathbf{W}_{out}[-1, \mathbf{u}(n), \mathbf{y}(n-1), \mathbf{f}(\mathbf{u}(n), \mathbf{u}(n-1), ...)]^T$
A2	1	0	$\mathbf{y}(n) = \mathbf{W}_{out}[-1, \mathbf{u}(n), \mathbf{f}(\mathbf{u}(n), \mathbf{u}(n-1), ..., \mathbf{y}(n-1), \mathbf{y}(n-2), ...)]^T$
A3	1	1	$\mathbf{y}(n) = \mathbf{W}_{out}[-1, \mathbf{u}(n), \mathbf{y}(n-1), \mathbf{f}(\mathbf{u}(n), \mathbf{u}(n-1), ..., \mathbf{y}(n-1), \mathbf{y}(n-2), ...)]^T$

Fonte: Autoria própria.

4.1 Descrição das arquiteturas da ESN aplicadas

Desde sua proposta inicial (JAEGER, 2001), apresentam-se inúmeras possibilidades de arquiteturas de conexões da ESN, sendo obtidas a partir da exclusão de vias de sua forma completa ilustrada na Figura 6. Por exemplo, é possível realizar a modelagem de um sistema sintético de décima ordem com componente autoregressívo sem realimentação da saída da ESN de maneira satisfatória (JAEGER, 2003). Como se trata aqui de um sistema com entrada e saída, a arquitetura completa também é uma possibilidade. Em vista disso, propõe-se avaliar a influencia de diferentes arquiteturas na resposta dos modelos.

Como o objetivo é promover a robustez aos *outliers*, considerando sua presença apenas nas amostras de saída do sistema como dito anteriormente, as variantes de arquitetura da ESN confrontadas são referentes as possíveis vias de realimentação de sua saída. Desse modo, o diagrama da Figura 8 explicita essas vias opcionais por linhas tracejadas, as quais são elencadas como vias α e β. Além disso, apresentam-se quatro arquiteturas factíveis pela combinação dessas conexões na Tabela 2, em que se considera a via desabilitada por "0"e habitada por "1". Ademais, é importante salientar que não é aplicado o integrador com vazamento (ou melhor, $\gamma = 1$) nos modelos avaliados.

4.2 Descrição dos métodos de estimação dos pesos da ESN

Em arranjo com as arquiteturas apresentadas, três abordagens de estimação dos pesos da ESN são avaliadas. A primeira utiliza o algoritmo RLS, que é usado como linha de referência para o contraste com os algoritmos robustos. Na segunda, é empregado o algoritmo RLM com as funções de Huber e de Huber modificada tal como descritos na fundamentação teórica. Por fim, a terceira abordagem apresenta uma modificação no treinamento da ESN com o RLM quando uma das vias de realimentação está ativa (no caso das arquiteturas A1, A2 e A3), chamada de RLM com realimentação cautelosa (RLM *with cautious feedback*, RLMCFB), sendo descrita em detalhes a seguir.

4.2.1 Algoritmo RLM com realimentação cautelosa (RLMCFB)

Com o intuito de evitar a realimentação de uma amostra inconsistente, propõe-se utilizar o estimador do desvio padrão do erro $\sigma(n)$ presente no algoritmo RLM para, além de atenuar ou impedir a atualização dos pesos, julgar se a saída desejada $t(n)$ deve ser realimentada como um dos regressores na próxima iteração como é convencional no treinamento pelo método *teacher forcing*. Então, caso a amostra seja um possível *outlier*, a próxima interação opera com uma realimentação real, similar ao aprendizado FORCE (SUSSILLO; ABBOTT, 2009).

Em outras palavras, na iteração n, a saída prevista é calculada por $y(n) = \mathbf{w}_{out}^T(n-1)\mathbf{h}(n)$. Logo, o erro predito resulta de $e(n) = t(n) - \mathbf{w}_{out}^T(n-1)\mathbf{h}(n)$, sendo aplicado na atualização dos pesos como descrito na Secção 3.2. Entretanto, supondo que $|e(n)| > \xi_{cfb}$, em que ξ_{cfb} é o limiar adicional do RLMCFB, também proporcional a $\sigma(n)$, isso ocorre porque o valor desejado $t(n)$ provavelmente é um *outlier*. Assim, esse valor é substituído por $\hat{t}(n) = \mathbf{w}_{out}^T(n)\mathbf{h}(n)$ na realimentação da rede na próxima iteração.

Essa abordagem busca evitar o efeito degradante nas estimações seguintes a realimentação de um *outlier*, podendo ser considerado um método de treinamento híbrido. Porém, mantem-se as características do método *teacher forcing* predominantemente. Para uma comparação isonômica, o algoritmo RLMCFB utiliza as mesmas funções custo do RLM nos testes da ESN, Huber e Huber modificada, sendo atribuído o limiar $\xi_{cfb} = 2.576\hat{\sigma}$ em ambas.

No Algoritmo 2, apresenta-se um pseudocódigo referente ao RLMCFB em conjunto com a função de Huber para o melhor entendimento. Ademais, quando se lida com sistemas de múltiplas saídas, esse algoritmo trata individualmente cada saída, ou seja, o vetor de regressores

Algoritmo 2: Pseudocódigo do treinamento da ESN utilizando o algoritmo RLMCFB

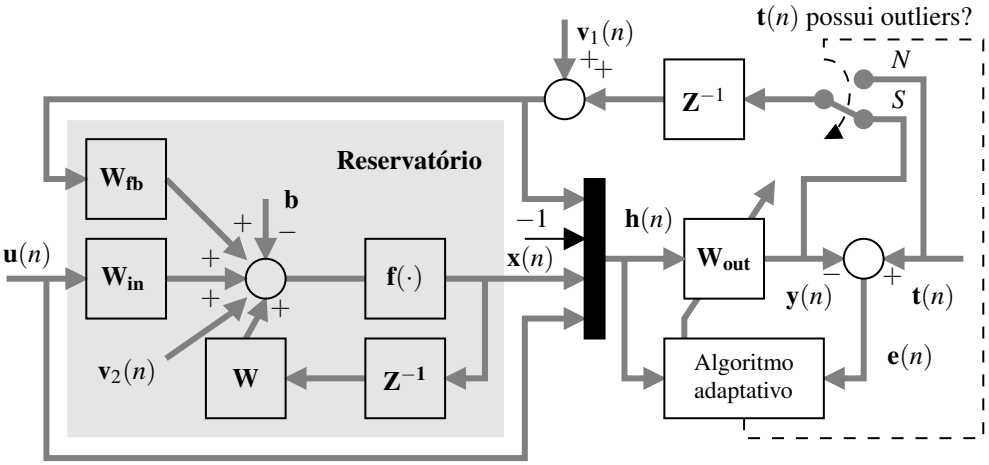
Entrada: δ, $\{(\mathbf{u}(n),t(n))\}_{n=1}^{N_{train}}$, $\mathbf{W}$, $\mathbf{W}_{fb}$, $\mathbf{W}_{in}$, $\mathbf{b}$, λ.
Saída: $\mathbf{w}_{out}(N_{train})$
Resultado: Estimação do vetor $\mathbf{w}_{out}$
início
  $\hat{t}(0) \leftarrow 0$; $\mathbf{x}(0) \leftarrow \mathbf{0}$; $\mathbf{w}_{out}(0) \leftarrow \mathbf{0}$; $\mathbf{S}(0) \leftarrow \delta^{-1}\mathbf{I}$;
  para $n = 1 : N_{train}$ **faça**
    # Cálculo da execução da ESN
    $\mathbf{x}(n) \leftarrow \mathbf{f}(\mathbf{W}\mathbf{x}(n-1) + \mathbf{W}_{in}\mathbf{u}(n) + \mathbf{W}_{fb}\hat{t}(n-1) - \mathbf{b})$;
    $\mathbf{h}(n) \leftarrow [-1, \mathbf{u}(n), aux\_t(n-1), \mathbf{x}(n)]^T$;
    $y(n) \leftarrow \mathbf{w}_{out}^T(n-1)\mathbf{h}(n)$;
    # Cálculo do erro de previsão da ESN
    $e(n) \leftarrow t(n) - y(n)$;
    # Cálculo da atualização dos pesos pelo RLMOD (função de Huber)
    se $|e(n)| > \xi_{fh}(n-1)$ **então**
      $q(e(n)) \leftarrow \xi_{fh}(n-1)/|e(n)|$;
    senão
      $q(e(n)) \leftarrow 1$;
    fim
    $c_\sigma = 1.483(1 + 5/max\{n-1, 1\})$;
    $\hat{\sigma}^2(n) \leftarrow \lambda_\sigma \hat{\sigma}^2(n-1) + (1-\lambda_\sigma)c_\sigma med\{e^2(i)\}_{i=max\{n-N_w+1,1\}}^{n}$;
    $\xi_{fh}(n) \leftarrow 1.345\hat{\sigma}(n)$; $\xi_{cfb}(n) \leftarrow 2.576\hat{\sigma}(n)$;
    $\mathbf{k}(n) \leftarrow \dfrac{\lambda^{-1}q(e(n))\mathbf{S}(n-1)\mathbf{h}(n)}{1 + \lambda^{-1}q(e(n))\mathbf{h}^T(n)\mathbf{S}(n-1)\mathbf{h}(n)}$;
    $\mathbf{S}(n) \leftarrow \lambda^{-1}\left(\mathbf{I} - \mathbf{k}(n)\mathbf{h}^T(n)\right)\mathbf{S}(n-1)$;
    $\mathbf{w}_{out}(n) \leftarrow \mathbf{w}_{out}(n-1) + e(n)\mathbf{k}(n)$;
    se $|e(n)| > \xi_{cfb}(n-1)$ **então**
      $\hat{t}(n) \leftarrow \mathbf{w}_{out}^T(n)\mathbf{h}(n)$;
    senão
      $\hat{t}(n) \leftarrow t(n)$;
    fim
  fim
fim

$\hat{\mathbf{t}}_j(n)$ pode apresentar valores de retroalimentação estimados, $y_j(n)$, ou provindos do conjunto de dados, $t_j(n)$, na mesma iteração n durante o treinamento. Por fim, a Figura 9 mostra o diagrama de operação simplificado desse método.

4.3 Metodologia das simulações e avaliação de desempenho

Ante a todas disposições realizadas para construção de modelos neurais baseados na ESN, a análise da robustez desses modelos na identificação de sistemas dinâmico na presença de *outliers* é realizada através de experimentos computacionais. Para esta prova de conceito,

Figura 9 – Diagrama de treinamento *online* da RLMCFB-ESN.



Fonte: Autoria própria.

selecionam-se quatro sistemas reais e um sintético disponíveis em repositórios, os quais são contaminados com ruído estatístico cuja função densidade de probabilidade é não gaussiana, e afere-se o desempenho dessas estruturas dado um aumento gradual da contaminação. Dessa forma, é necessário esclarecer todos os métodos estabelecidos para realizar essas operações.

4.3.1 Modelos neurais baseados na ESN aferidos

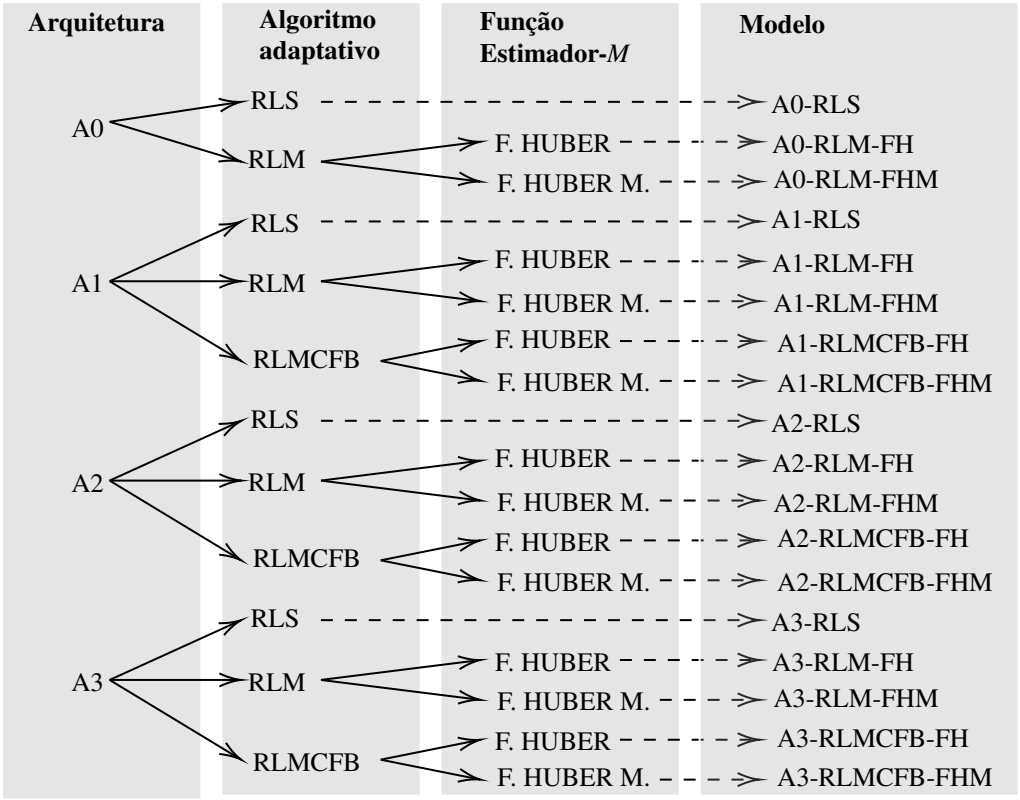
A partir das considerações feitas na Secção 4.1 e 4.2, pode-se construir dezoito modelos da ESN combinando as arquiteturas descritas (A0, A1, A2 e A3) com os algoritmos de estimação propostos (RLS, RLM, RLMCFB), além da escolha da função estimador-*M* quando conveniente, função de Huber ou Huber modificada. Tais modelos são listados na Figura 10.

4.3.2 Enumeração dos bancos de dados

Para realizar os experimentos computacionais, buscam-se amostras referentes a sistemas dinâmicos com características diversas, os quais são listados na Tabela 3 com suas respectivas referências e número de amostras de entrada e de saída.

É importante salientar que o foco desse trabalho não é encontrar o melhor desempenho para cada modelo, mas apresentar os benefícios alcançados na substituição do usual algoritmo RLS por uma versão robusta a *outliers*.

Figura 10 – Lista de modelos avaliados.



Fonte: Autoria própria.

Tabela 3 – Lista de conjunto de dados utilizados nos experimentos computacionais.

Descrição	**Nome**	**# inst. de tempo**	**Referência**
Sistema de secador de cabelo (SISO)	*Dryer*	1000	(MOOR *et al.*, 1997)
Sistema de permutador de calor (SISO)	*Exchanger*	3000[1]	(MOOR *et al.*, 1997)
Sistema de braço robótico flexível (SISO)	*Robot Arm*	1024	(MOOR *et al.*, 1997)
Sistema de tanques em cascata (SIMO)[2]	*Tank*	7500	(WIGREN, 2010)
Sistema de gerador de vapor industrial (MIMO)[3]	*Steam*	9600	(MOOR *et al.*, 1997)

Fonte: Autoria própria.
Nota: 1 - O sistema *Exchanger* possui 4000 amostras, porém são excluídas as 1000 últimas amostras dos experimentos; 2 - O sistema *Tank* possui duas saídas; 3 - O referido sistema MIMO possui quatro entradas e quatro saídas.

4.3.3 Definição dos hiperparâmetros e normalização dos dados

Diante disso, na Tabela 4, apresentam-se os valores dos hiperparâmetros que retornam resultados aceitáveis segundo determinadas figuras de mérito, descritas mais a diante, para as arquiteturas da ESN e os sistemas citados, considerando o cenário sem *outlier* e utilizando algoritmo RLS. Esses hiperparâmetros são utilizados nas simulações, exceto quando indicada alteração do valor.

O algoritmo RLM apresenta dois parâmetros além dos presentes na Tabela 4, isso é, o comprimento de janela amostral N_w e o fator de esquecimento $\lambda_{\hat{\sigma}}$ do estimador da variância $\hat{\sigma}^2$

Tabela 4 – Configuração padrão de hiperparâmetros dos modelos nas simulações.

Configuração ESN	
Raio espectral (**W**)	0.9
Esparcidade (**W**)	5%
Pesos de entrada	$\mathbf{W}_{in} \sim U(-0.1, 0.1)$
Pesos de realimentação	$\mathbf{W}_{back} \sim U(-0.1, 0.1)$
Bias	$\mathbf{b} \sim U(-0.1, 0.1)$
Ruído aditivo 1	$\mathbf{v}_1 = \mathbf{0}$
Ruído aditivo 2	$\mathbf{v}_2 = \mathbf{0}$
Configuração RLS/RLM	
Fator de esquecimento (γ)	1
Parâmetro de regularização (δ)	10^{-3}

Fonte: Autoria própria.

presentes na Equação 3.24. Dada a finitude dos subconjuntos de treinamento, opta-se em se ter a maior acurácia na estimação de $\hat{\sigma}^2(n)$, utilizando todas as amostras disponíveis até a iteração n para o seu cálculo. Consequentemente, o valor de N_w é crescente e varia de 1 ao número total de amostras de treinamento, N_{train}, e não aplica-se o fator de esquecimento, considerando $\lambda_{\hat{\sigma}} = 1$. Ademais, é válido destacar a escolha do fator de esquecimento dos algoritmos RLS e RLM é $\lambda = 1$, pois se decide padronizar os efeitos da regularização das soluções independente do número de amostras do sistema, efeito discutido na Subseção 3.2.2.3, além de considerar os sistemas não variante com o tempo.

Por fim, as entradas e as saídas dos sistemas são normalizadas, subtraindo a média e dividindo por três vezes o valor do desvio padrão, estatísticas das amostras de treinamento. Outro ponto importante é que essa normalização é aplicada mesmo para cenários que há contaminação das amostras, ou seja, essas estatísticas podem ser afetadas pelos *outliers*.

4.3.4 Metodologia de contaminação dos dados por outliers

A robustez dos métodos são avaliados em cenários de contaminação por *outliers* de 5%, de 10% e de 15% das amostras de treinamento. Para isso, adota-se o mesmo procedimento proposto em Mattos *et al.* (2017), em que os *outliers* são artificialmente introduzidos às saídas dos conjuntos de treinamento dos sistemas, sendo gerados por amostragem de $\sigma_{train}\mathscr{T}(0,2)$, ou seja, de uma variável aleatória que segue uma distribuição t de Student, tendo média zero e dois graus de liberdade, e escalonada pelo valor do desvio padrão referente as amostras de treinamento da saída a ser contaminada. Além disso, os valores retornados dessa distribuição $\mathscr{T}(0,2)$ são saturados em ± 20 para evitar valores extremos.

4.3.5 Figuras de mérito

O propósito principal desse trabalho é utilizar a ESN para a tarefa de predição de sistemas em simulação livre de longo prazo de forma estável. Então, opta-se por dividir os conjuntos de dados disponíveis (N_{all} amostradas) pela metade, assumindo os primeiros 50% dos dados (N_{train} amostradas) para o treinamento do modelo e a segunda metade (N_{test} amostradas) para verificar a eficiência na generalização dos modelos aos dados não apresentados. Essa forma de divisão apresenta a vantagem de evitar o transitório inicial do reservatório da ESN durante a previsão, pois utiliza-se os estados apresentados no final do treinamento, $\mathbf{x}(N_{train})$, para realizar a primeira previsão.

Para caracterizar o desempenho desses modelos, são utilizadas duas figuras de mérito. A primeira é a raiz do erro quadrático médio (*root mean square error*, RMSE) das previsões de múltiplos passos à frente, denotada por:

$$RMSE = \sqrt{\frac{\sum_{n=N_{train}+1}^{N_{all}} (t(n) - y(n))^2}{N_{test}}}, \tag{4.1}$$

que evidencia a generalização do modelo. A segunda é a norma Euclidiana (l_2) dos pesos da camada de *readout*, $||\mathbf{w}_{out}||$, que é relacionada com a estabilidade numérica do modelo e suavidade da curva resultante. Essas duas figuras de mérito são importantes para avaliar a qualidade da previsão.

Além dessas, calcula-se o RMSE durante a metade final do treinamento, ou seja, no intervalo entre os instantes $[inteiro(N_{train}/2), N_{train}]$, ainda no decorrer da adaptação da camada de saída, com o intuito de verificar o comportamento da previsão para um passo à frente. Para isso, utiliza-se o valor da saída desejada do subconjunto de treinamento sem *outliers*, o que possibilita verificar o desempenho da rede em modo adaptativo em cenários de contaminação.

4.3.6 Metodologia dos experimentos e número de realizações

Motivada pela natureza aleatória da ESN, é interessante que ocorra aferição dos resultados para diferentes instanciações de pesos da ESN. Da mesma forma, a avaliação de diferentes contaminações dos bancos de dados para a mesma porcentagem de *outliers* corrobora para inferir a robustez do método. Dito isso, decide-se realizar 20 rodadas de treinamento e teste para cada modelo, em que suas respostas são representadas em diagrama de caixa (*bloxplot*). Com essa finalidade, são construídos 20 reservatórios, os quais são combinados com os 18

Tabela 5 – Configuração da injeção de ruído nos teste de hiperparâmetros.

Nomenclatura	**Parâmetros das distribuições**
N1	$\mathbf{v}_1 \sim U(-10^{-4}, 10^{-4})$ e $\mathbf{v}_2 \sim U(-10^{-4}, 10^{-4})$
N2	$\mathbf{v}_1 \sim U(-10^{-3}, 10^{-3})$ e $\mathbf{v}_2 = \mathbf{0}$
N3	$\mathbf{v}_1 = \mathbf{0}$ e $\mathbf{v}_2 \sim U(-10^{-3}, 10^{-3})$

Fonte: Autoria própria.

modelos propostos, possibilitando uma comparação mais justa entre os algoritmos de estimação e as arquiteturas. Ademais, cada cenário de contaminação apresenta 20 perfis de injeção de *outliers* distintos, sendo um para cada rodada de treinamento.

4.3.7 Metodologia de apresentação dos resultados

Os resultados obtidos são apresentados por meio de *boxplot* e gráfico de curvas de simulação, contemplando o final do treinamento e o início da simulação livre, com os valores de norma e RMSE para a respectiva execução. Primeiramente, expõem-se os resultados para verificar o desempenho dos hiperparâmetros e das arquiteturas, em que se variam levemente o raio espectral, o número de neurônios e o parâmetro de regularização em relação aos presentes na Tabela 4. Além disso, adiciona-se os ruídos $\mathbf{v}_1$ e $\mathbf{v}_2$ de acordo com a configuração apresentada na Tabela 5 para verificar o efeito desses na norma.

Em seguida, é realizada a comparação entre os algoritmos RLS e RLM em cenários contaminados, no qual é avaliado uma configuração adicional do RLS com o parâmetro de regularização com valor excessivo para diminuir a norma, $\delta = 10^{-1}$. Assim, como a presença de *outliers* altera a norma da solução, pode-se fazer uma comparação mais rica sobre os efeitos além da norma. Por fim, comparam-se os algoritmos RLM e RLMCFB nas três arquiteturas que possuem retroalimentação da saída da ESN e se apresentam as curvas resultantes de uma execução com a arquitetura completa da ESN, isso é, arquitetura A3, no cenário de 15% de contaminação por *outlier* para todos os algoritmos de estimação.

5 RESULTADOS EXPERIMENTAIS

Neste Capítulo, reportam-se os resultados obtidos nos experimentos realizados com os bancos de dados citados na Tabela 3, sendo apresentados na seguinte ordem:

Secção 5.1: Sistema de secador de cabelo (*Dryer*);

Secção 5.2: Sistema de permutador de calor (*Exchanger*);

Secção 5.3: Sistema de braço robótico flexível (*Robot Arm*);

Secção 5.4: Sistema de tanques em cascata (*Tank*);

Secção 5.5: Sistema de gerador de vapor industrial (*Steam*).

5.1 Sistema de secador de cabelo (*Dryer*)

O sistema indicado como *Dryer* se refere a uma bancada de laboratório que se assemelha a um secador de cabelo (MOOR *et al.*, 1997). Assim, o ar é aquecido por uma malha de resistores na entrada de um tubo e, no seu final, localiza-se um termopar que é responsável pela mensuração da temperatura do ar, sendo coletadas 1000 amostras dessas grandezas com uma frequência de amostragem de 1 Hz. Logo, esse sistema é representado pela:

Entrada: Tensão aplicada à malha de resistores;

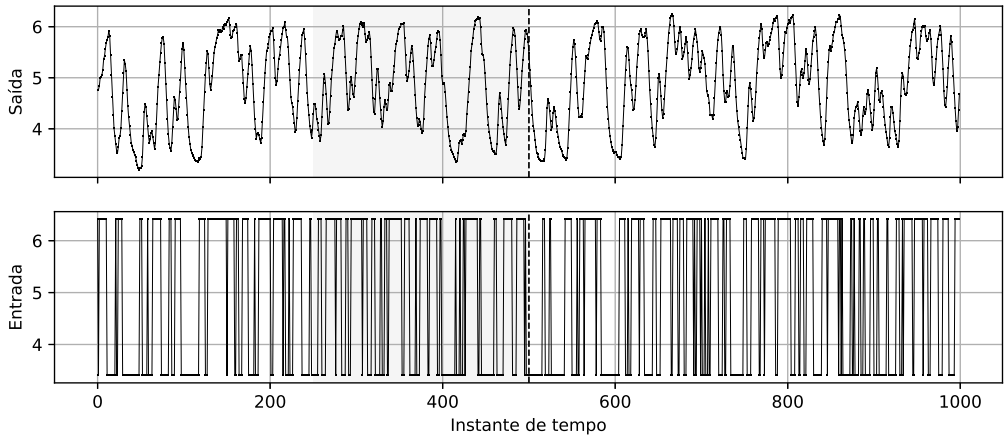
Saída: Tensão induzida pelo termopar.

A Figura 11 mostra as curvas de entrada e de saída do sistema *Dryer*.

Para esse sistema, os resultados obtidos pela ESN com o objetivo de verificar o desempenho dos hiperparâmetros da rede é apresentado na Figura 12. Nesse *boxplot*, pode-se perceber que os valores apresentados na Tabela 4 são subótimos. A partir das curvas apresentadas na Figura 13, nota-se que todas as arquiteturas apresentam bons resultados na identificação do sistema *Dryer*.

Seguindo para os resultados em diferente níveis de contaminação por *outliers*, compara-se o algoritmo RLS com o RLM na Figura 14. Pode-se perceber a robustez do RLM, pois esse mantem os valores de RMSE do teste próximo ao do cenário sem contaminação e atenua o aumento da norma provocado pelos *outliers*, principalmente no caso do RLM-FHM. Além disso, o algoritmo RLS com $\delta = 10^{-1}$ mantem a norma baixa, mas o seu desempenho é inferior ao RLM na comparação do RMSE de treino e de teste. Outro ponto interessante é que as arquiteturas da ESN apresentam um comportamento bastante semelhante, porém, a arquitetura A0 tem desvantagem no RMSE do treino, que se refere a previsão de um passo à

Figura 11 – Sistema *Dryer*: curvas de entrada e saída.



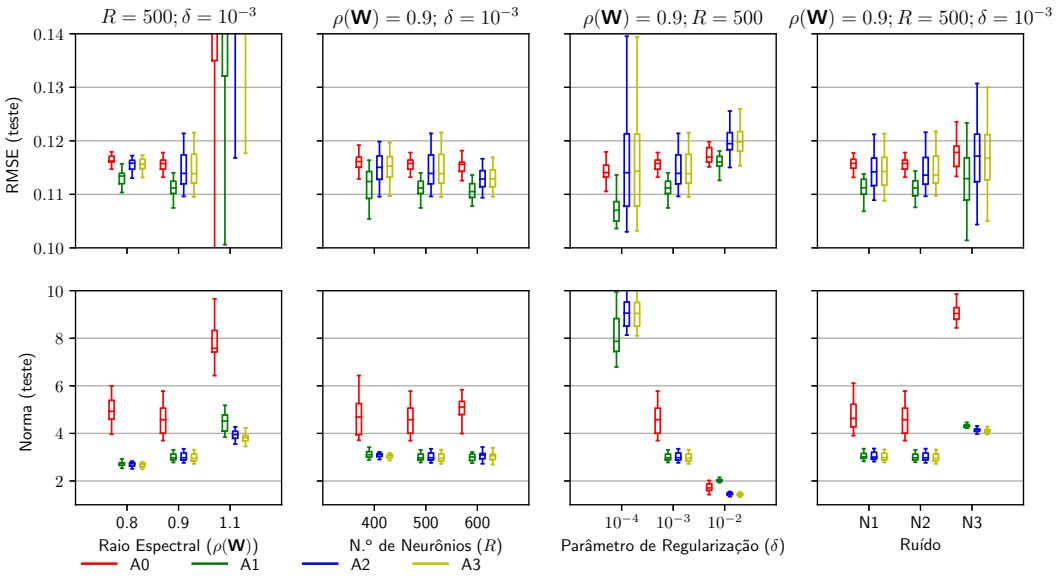
Fonte: Autoria própria.
Nota: A área cinza representa o intervalo que o RMSE é calculado durante o treinamento e a linha vertical tracejada a divisão do conjunto de dados em treinamento e teste.

frente, no cenário sem *outliers*, sendo justificável pela ausência do valor passado da saída real nos regressores.

No *boxplot* da Figura 15, as arquiteturas que apresentam realimentação da saída têm seus resultados apresentados para os estimadores RLM e RLMCFB. Nesses diagramas, evidencia-se o melhor desempenho do RLMCFB, em especial nos critérios de RMSE do treino e da norma da camada de *readout*, deixando aparente a vantagem de não realimentar uma amostra corrompida nesses modelos.

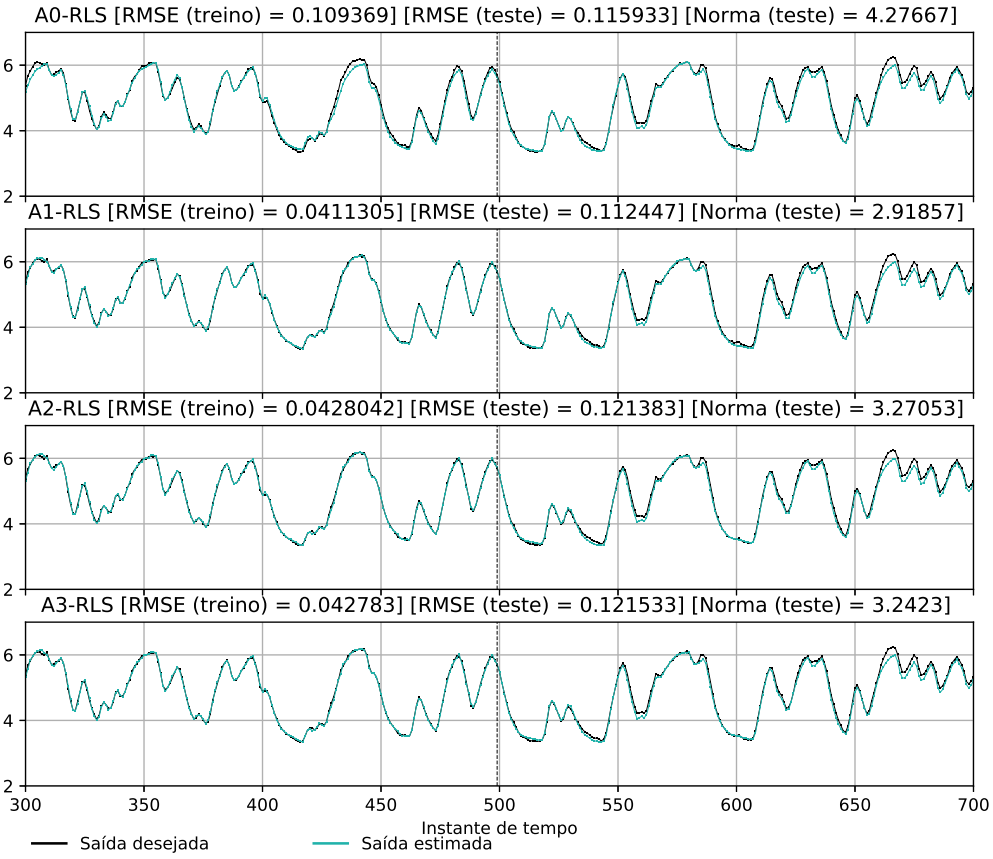
Por fim, apresentam-se as curvas dos modelos com arquitetura A3 na Figura 16, em que se verifica a deformação da curva do modelo estimado com o algorito RLS mesmo aumentando a regularização desse. Ademais, o modelo A3-RLMCFB-FHM exibe uma curva mais suave e a norma reduzida.

Figura 12 – Sistema *Dryer*: gráfico de desempenho entre hiperparâmetros e arquiteturas da ESN.



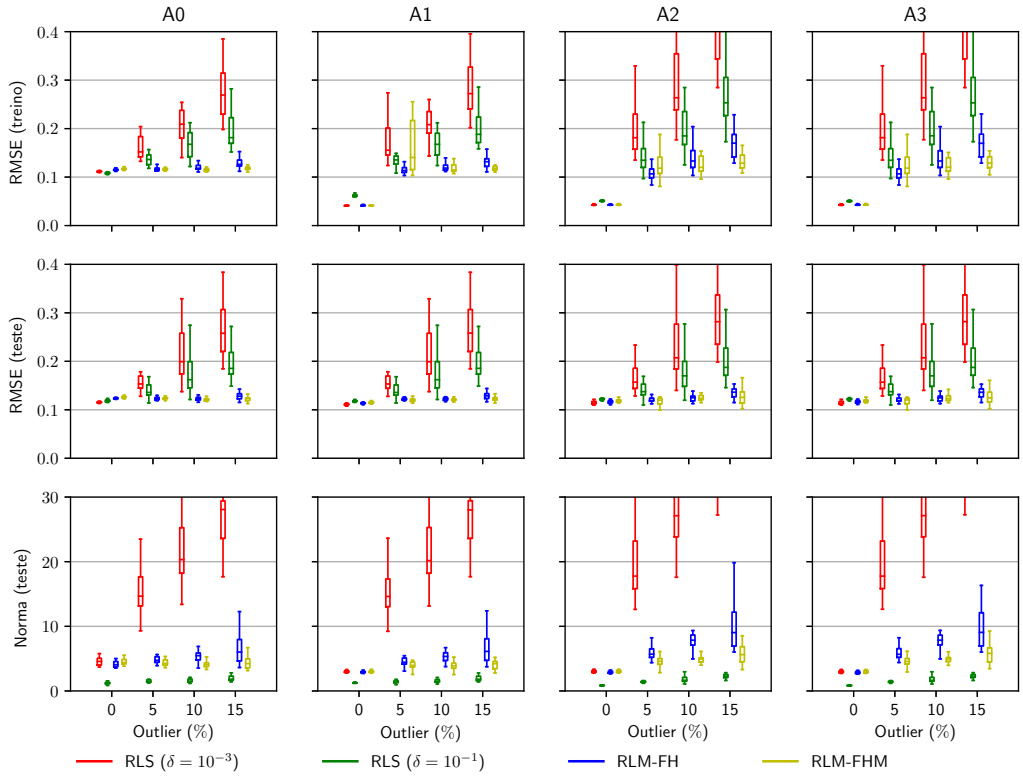
Fonte: Autoria própria.

Figura 13 – Sistema *Dryer*: curvas de previsão considerando diferentes arquiteturas.



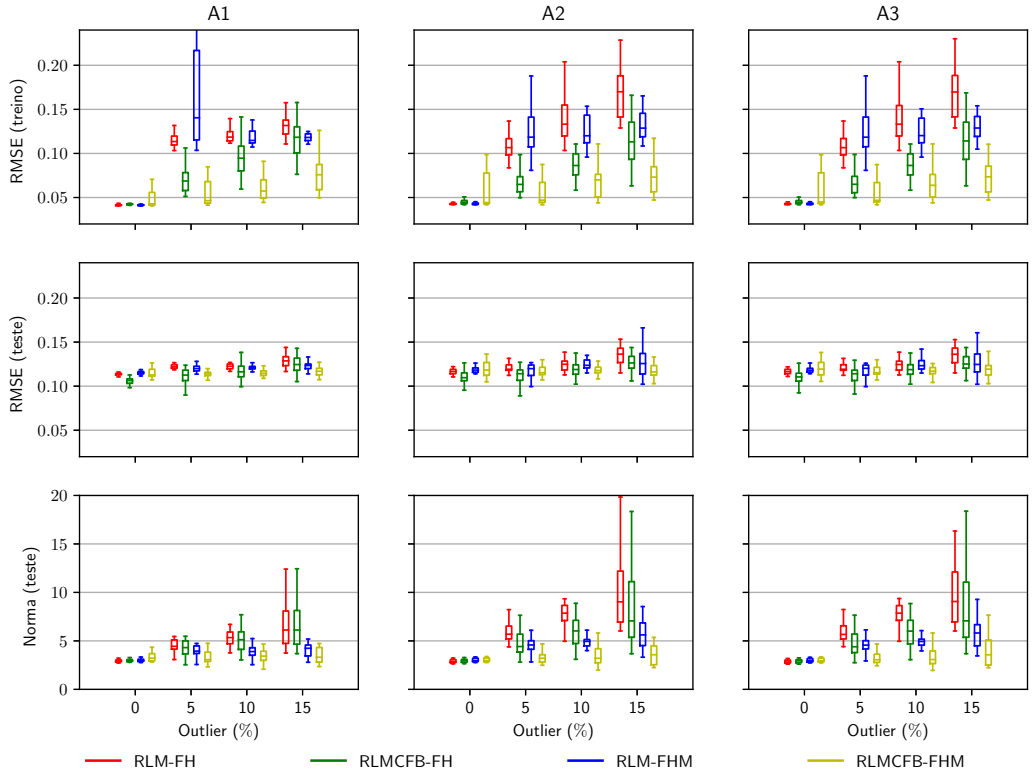
Fonte: Autoria própria.

Figura 14 – Sistema *Dryer*: gráfico de desempenho entre o algoritmo RLS e RLM na presença de *outliers*.



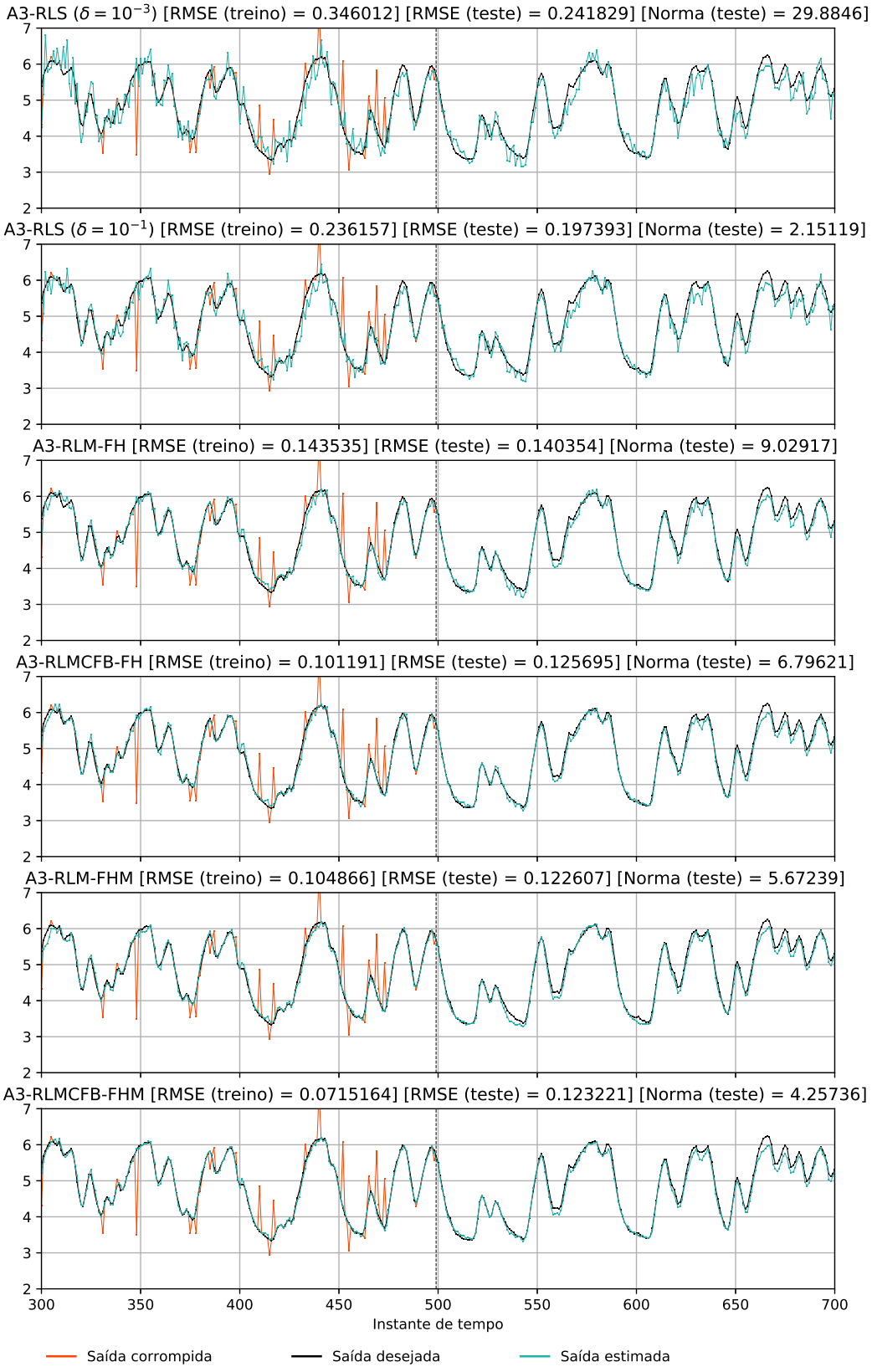
Fonte: Autoria própria.

Figura 15 – Sistema *Dryer*: gráfico de desempenho entre o algoritmo RLM e RLMCFB na presença de *outliers*.



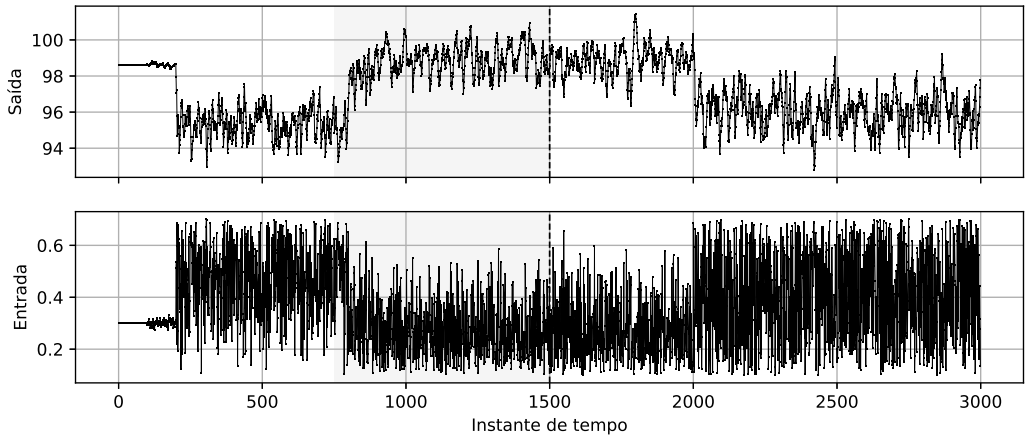
Fonte: Autoria própria.

Figura 16 – Sistema *Dryer*: curvas de previsão considerando diferentes algoritmos de estimação para o cenário de 15% de contaminação por *outlier*.



Fonte: Autoria própria.

Figura 17 – Sistema *Exchanger*: curvas de entrada e saída.



Fonte: Autoria própria.
Nota: A área cinza representa o intervalo que o RMSE é calculado durante o treinamento e a linha vertical tracejada a divisão do conjunto de dados em treinamento e teste.

5.2 Sistema de trocador de calor (*Exchanger*)

Com o intuito de regular a saída da temperatura da água que passa através do trocador de calor a vapor, aquisitam-se os dados desse sistema que é referenciado como *Exchanger* (MOOR *et al.*, 1997). As amostras desse processo são aferidas em um tempo de amostragem de 1 segundo e dispõem de 4000 dados relativos a cada uma das seguintes grandezas:

Entrada: Taxa de fluxo da entrada do líquido no trocador de calor;

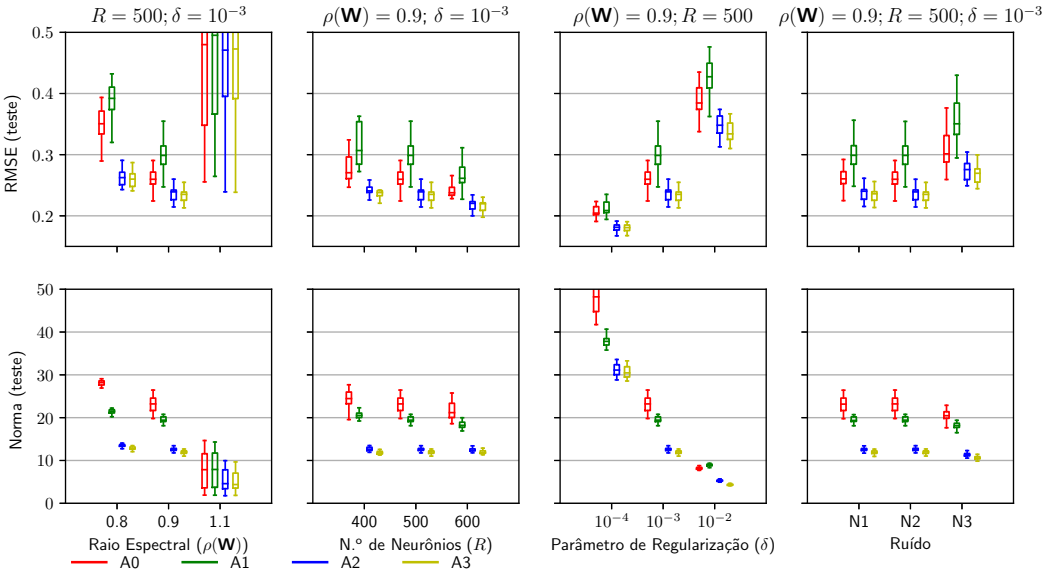
Saída: Temperatura de saída do líquido.

A Figura 17 mostra as curvas de entrada e de saída do sistema *Exchanger*, sendo consideradas apenas as 3000 primeiras amostras de entrada e de saída desse conjunto de dados.

Sobre os resultados para esse conjunto de dados, pode-se identificar uma leve vantagem das arquiteturas que possuem vias de retroalimentação para o reservatório (A2 e A3) pelo diagrama da Figura 18. Contundo, todas as arquiteturas são capazes de modelar bem o sistema como é visto na Figura 19, o que valida a escolha dos hiperparâmetros.

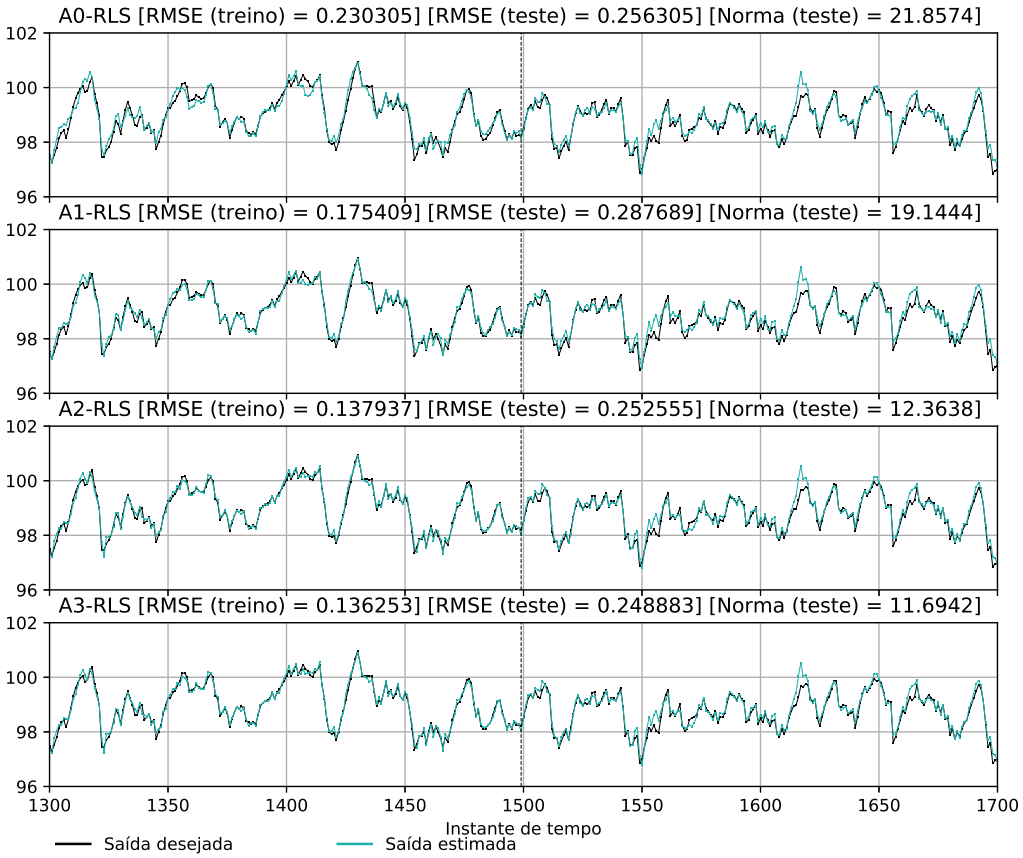
Pela Figura 20, é possível inferir que o estimador RLS não é capaz de lidar com os *outliers*. Já o RLM apresenta bons resultados para as funções estimador-M, que exibem desempenho semelhantes. Isso também é evidenciado no *boxplot* da Figura 21, no qual se compara os algoritmos RLM e RLMCFB. É interessante destacar que o RLMCFB demonstra um comportamento robusto, pois as figuras de mérito não sofrem alterações significativas com o aumento da contaminação dos dados. Além disso, as curvas exibidas na Figura 22 corroboram a afirmação anterior.

Figura 18 – Sistema *Exchanger*: gráfico de desempenho entre hiperparâmetros e arquiteturas da ESN.



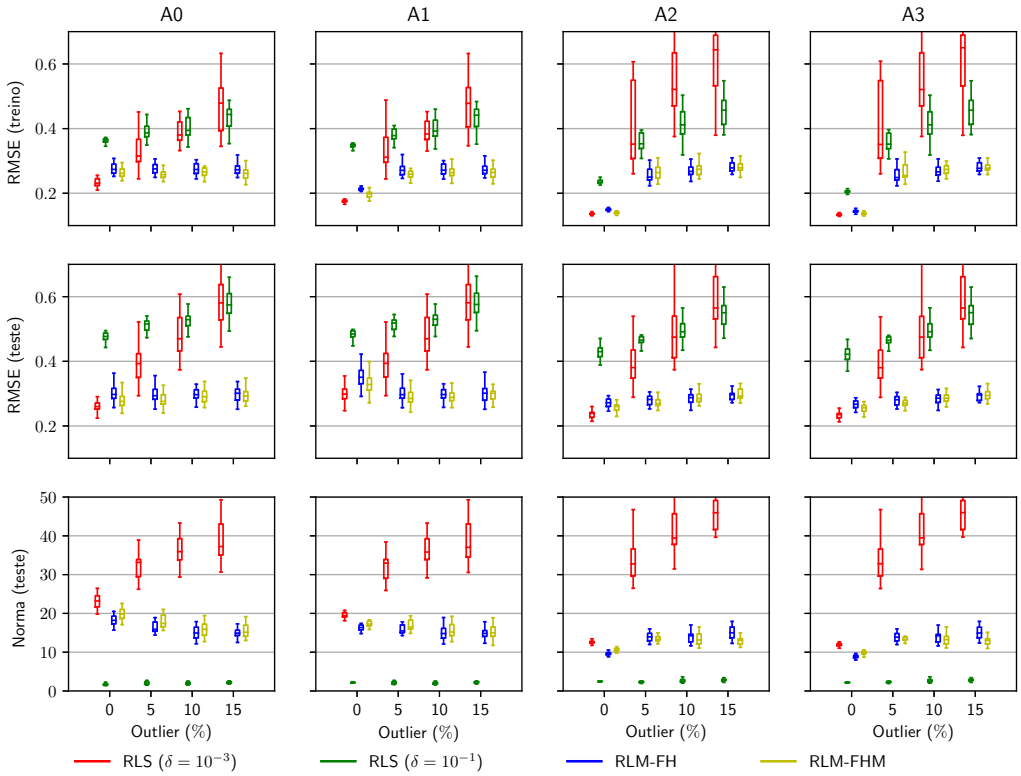
Fonte: Autoria própria.

Figura 19 – Sistema *Exchanger*: curvas de previsão considerando diferentes arquiteturas.



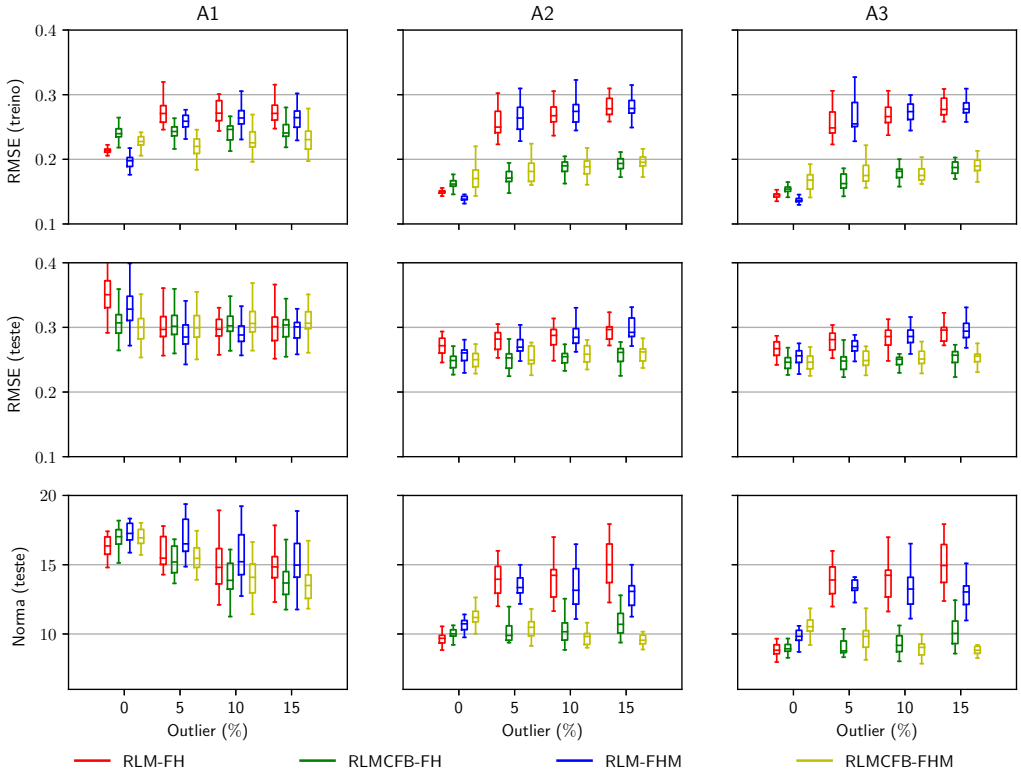
Fonte: Autoria própria.

Figura 20 – Sistema *Exchanger*: gráfico de desempenho entre o algoritmo RLS e RLM na presença de *outliers*.



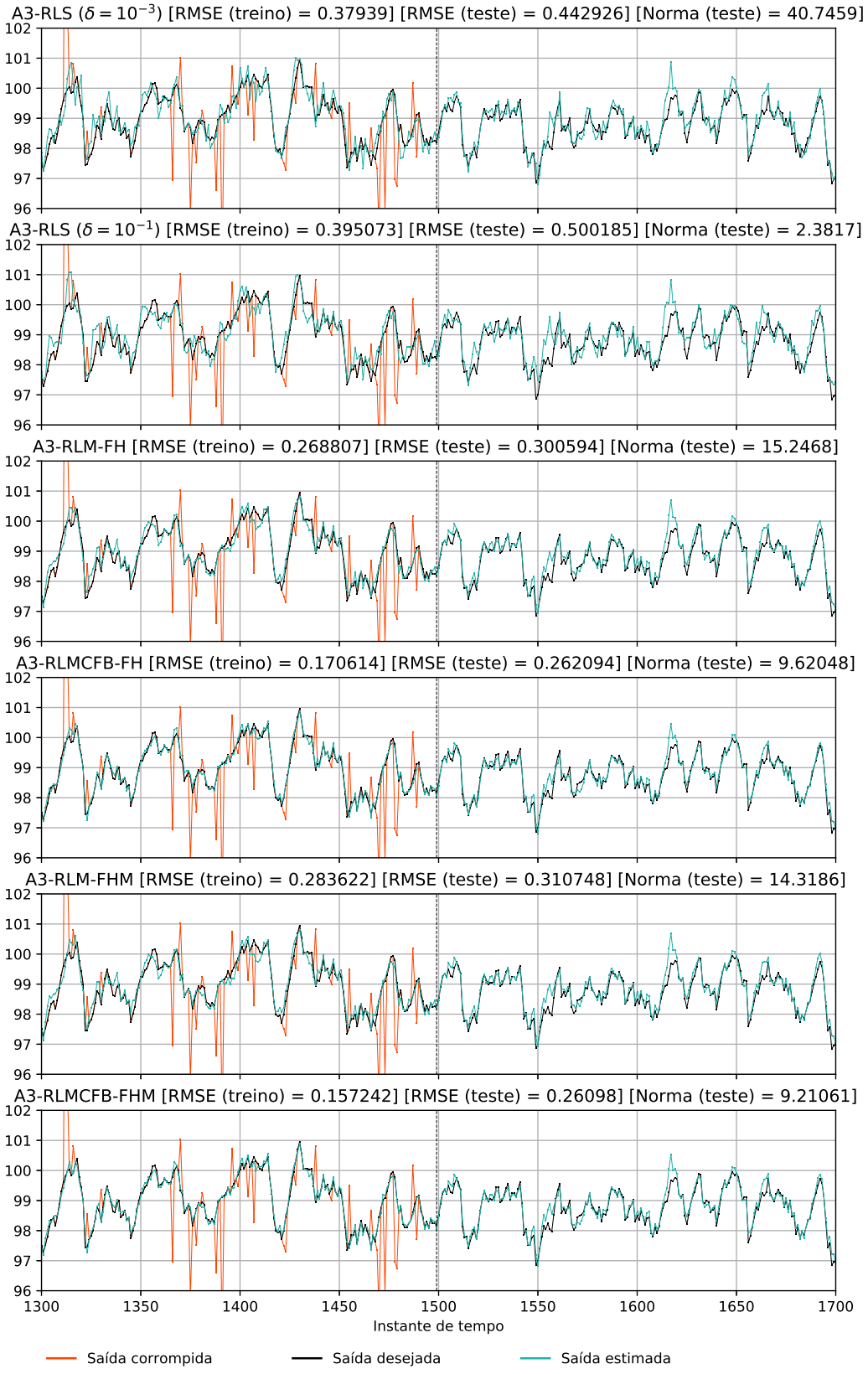
Fonte: o autor.

Figura 21 – Sistema *Exchanger*: gráfico de desempenho entre o algoritmo RLM e RLMCFB na presença de *outliers*.



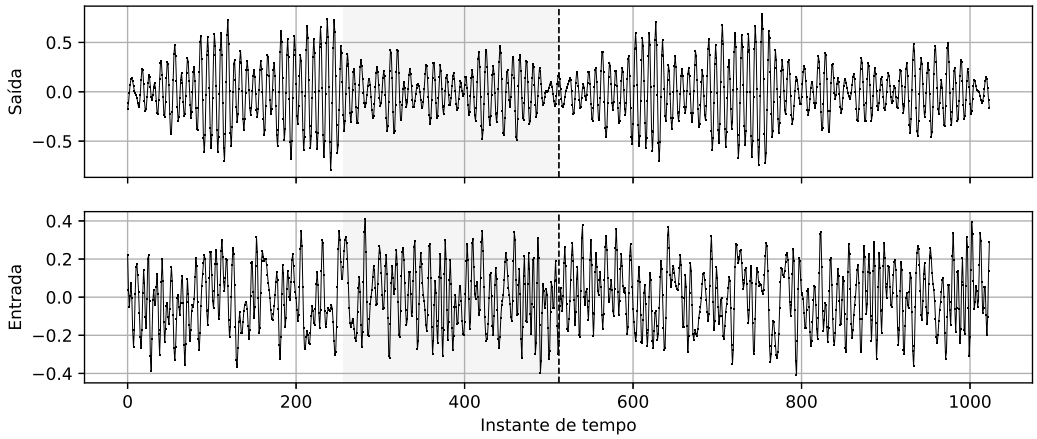
Fonte: Autoria própria.

Figura 22 – Sistema *Exchanger*: curvas de previsão considerando diferentes algoritmos de estimação para o cenário de 15% de contaminação por *outlier*.



Fonte: Autoria própria.

Figura 23 – Sistema *Robot Arm*: curvas de entrada e saída.



Fonte: Autoria própria.
Nota: A área cinza representa o intervalo que o RMSE é calculado durante o treinamento e a linha vertical tracejada a divisão do conjunto de dados em treinamento e teste.

5.3 Sistema de braço robótico flexível (*Robot Arm*)

Relativo a um braço robótico, o conjunto de dados referenciado por *Robot Arm* apresenta 1024 amostras, porém, não há informação sobre sua frequência de amostragem (MOOR *et al.*, 1997). Esse sistema é constituído de um braço conectado a um motor elétrico, em que se afere o torque de reação da estrutura no solo sobre a aceleração do braço flexível. Dessa maneira, as grandezas disponíveis do sistema são:

Entrada: Torque de reação da estrutura;

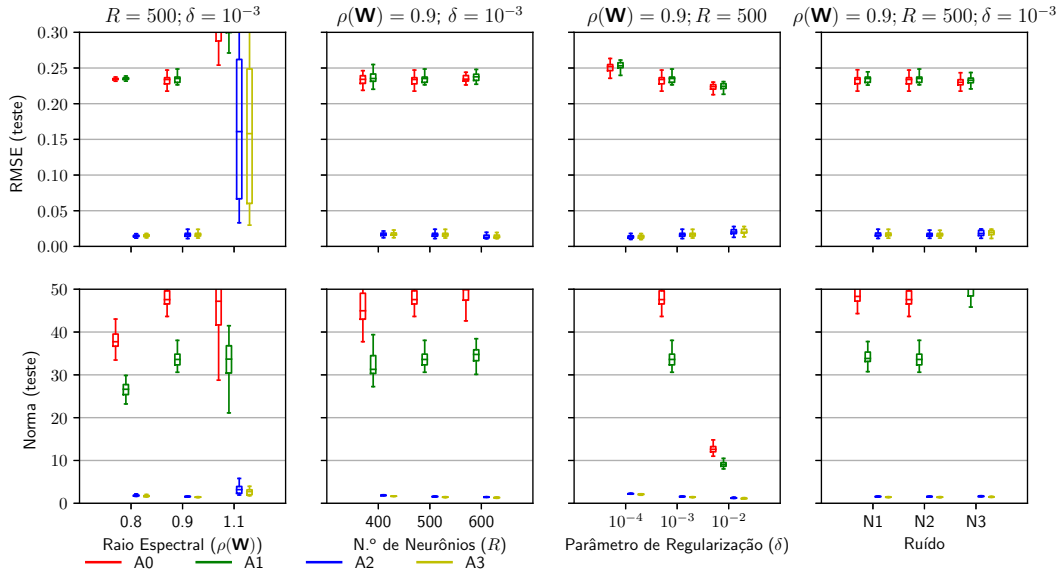
Saída: Aceleração do braço flexível.

A Figura 23 apresenta as curvas de entrada e de saída do sistema *Robot Arm*.

Por meio dos resultados apresentados no *boxplot* da Figura 24 e pelas curvas geradas pelos modelos exibidas na Figura 25, percebe-se que as arquiteturas A0 e A1 não são capazes de identificar o sistema *Robot Arm*. Ademais, as variações nos hiperparâmetros testadas não indicam que seja possível melhorar o desempenho dessas arquiteturas. Porém, as arquiteturas A2 e A3 exibem desempenho semelhante e conseguem modelar satisfatoriamente o sistema.

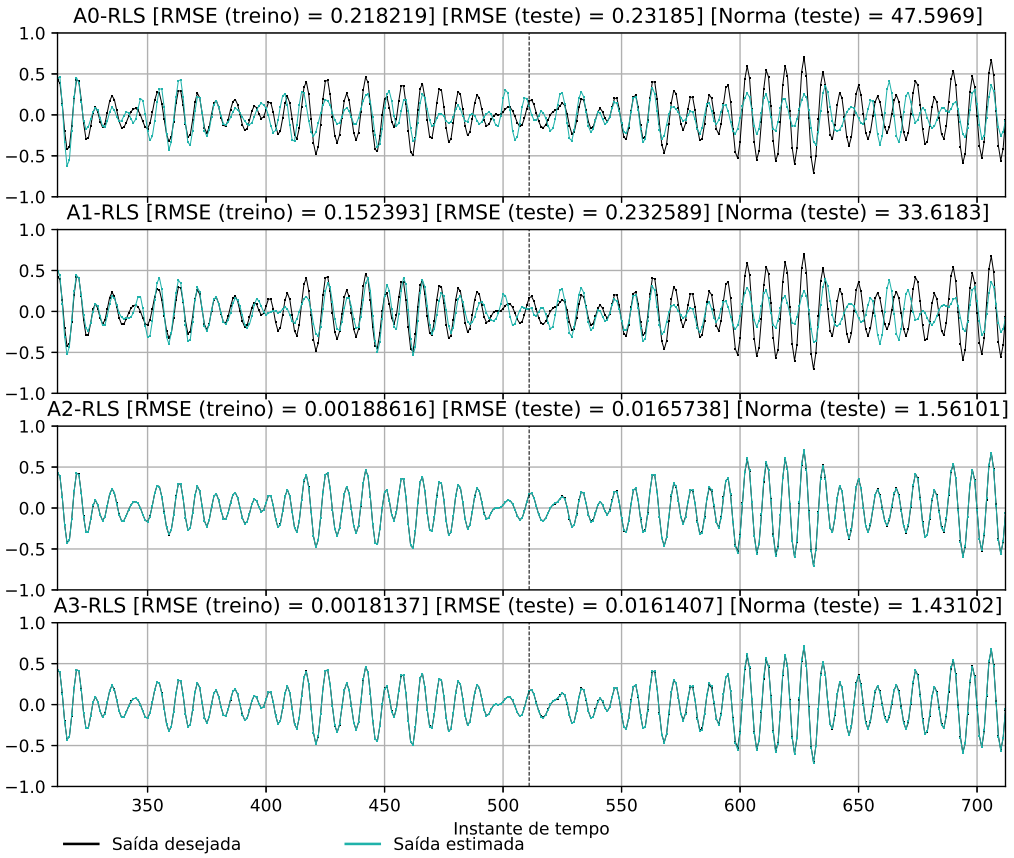
Na presença de *outliers*, o *boxplot* da Figura 26 evidencia que o desempenho do algoritmo RLM está bastante próximo ao do RLS. Entretanto, o algoritmo RLMCFB apresenta robustez à contaminação dos dados como pode ser verificada na Figura 27. Pode-se visualizar a diferença entre as simulações na Figura 28, em que o RLMCFB exibe os melhores resultados independente da função estimador-M, mas o modelo A3-RLMCFB-FHM alcança uma norma menor.

Figura 24 – Sistema *Robot Arm*: gráfico de desempenho entre hiperparâmetros e arquiteturas da ESN.



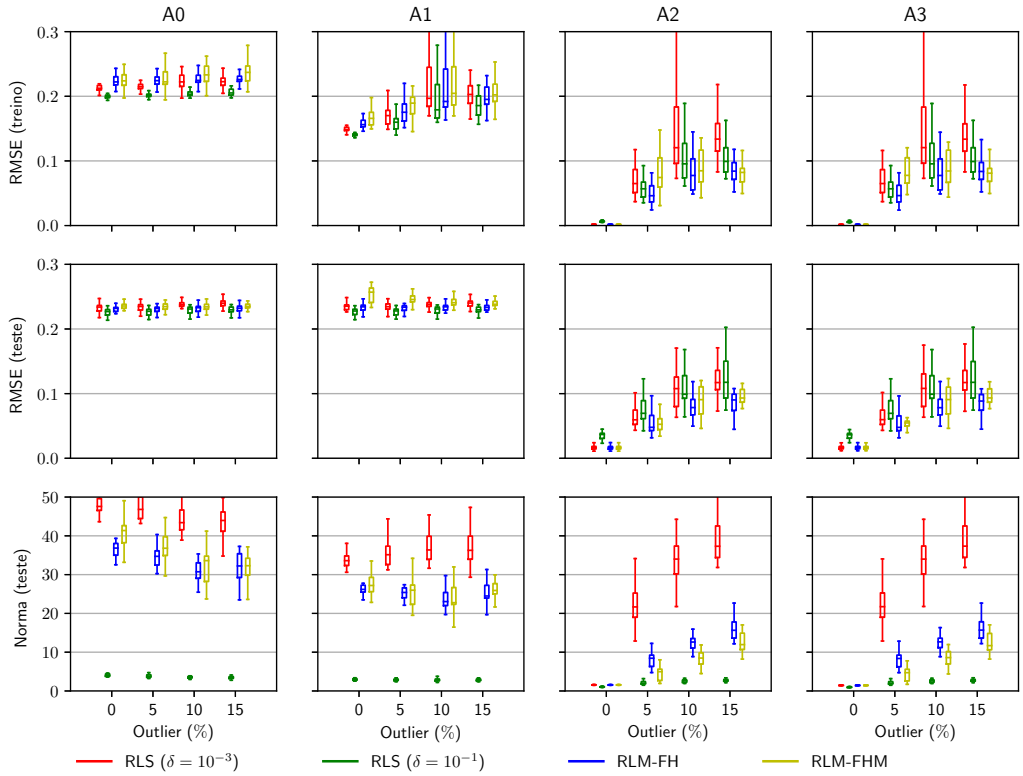
Fonte: Autoria própria.

Figura 25 – Sistema *Robot Arm*: curvas de previsão considerando diferentes arquiteturas.



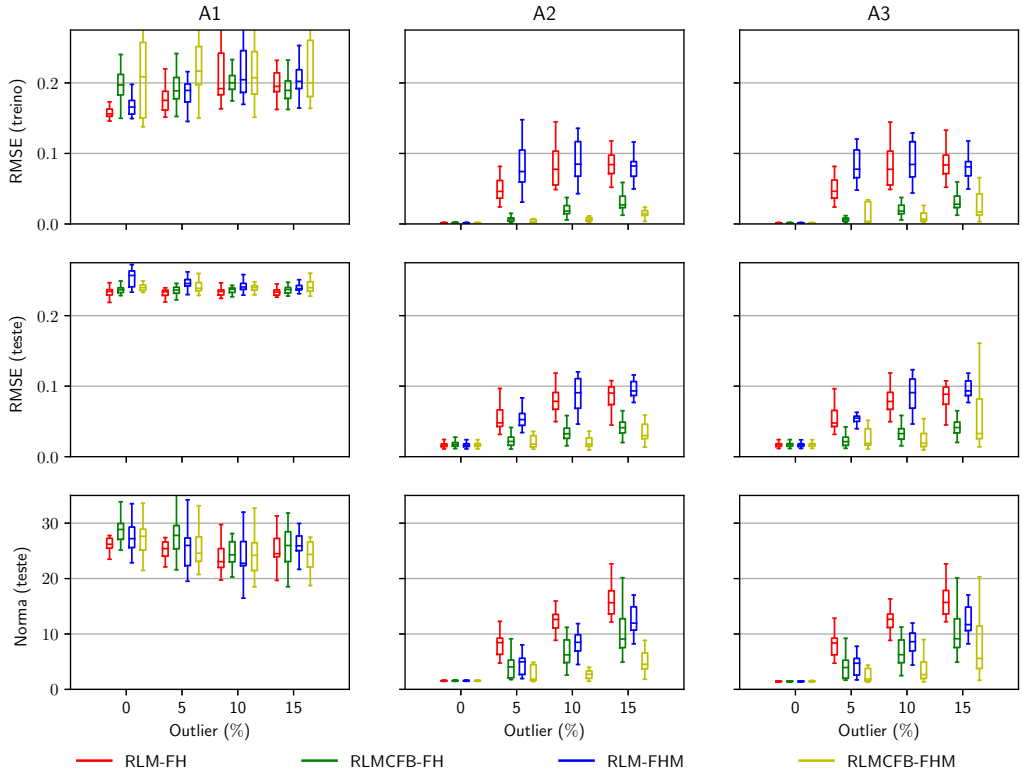
Fonte: Autoria própria.

Figura 26 – Sistema *Robot Arm*: gráfico de desempenho entre o algoritmo RLS e RLM na presença de *outliers*.



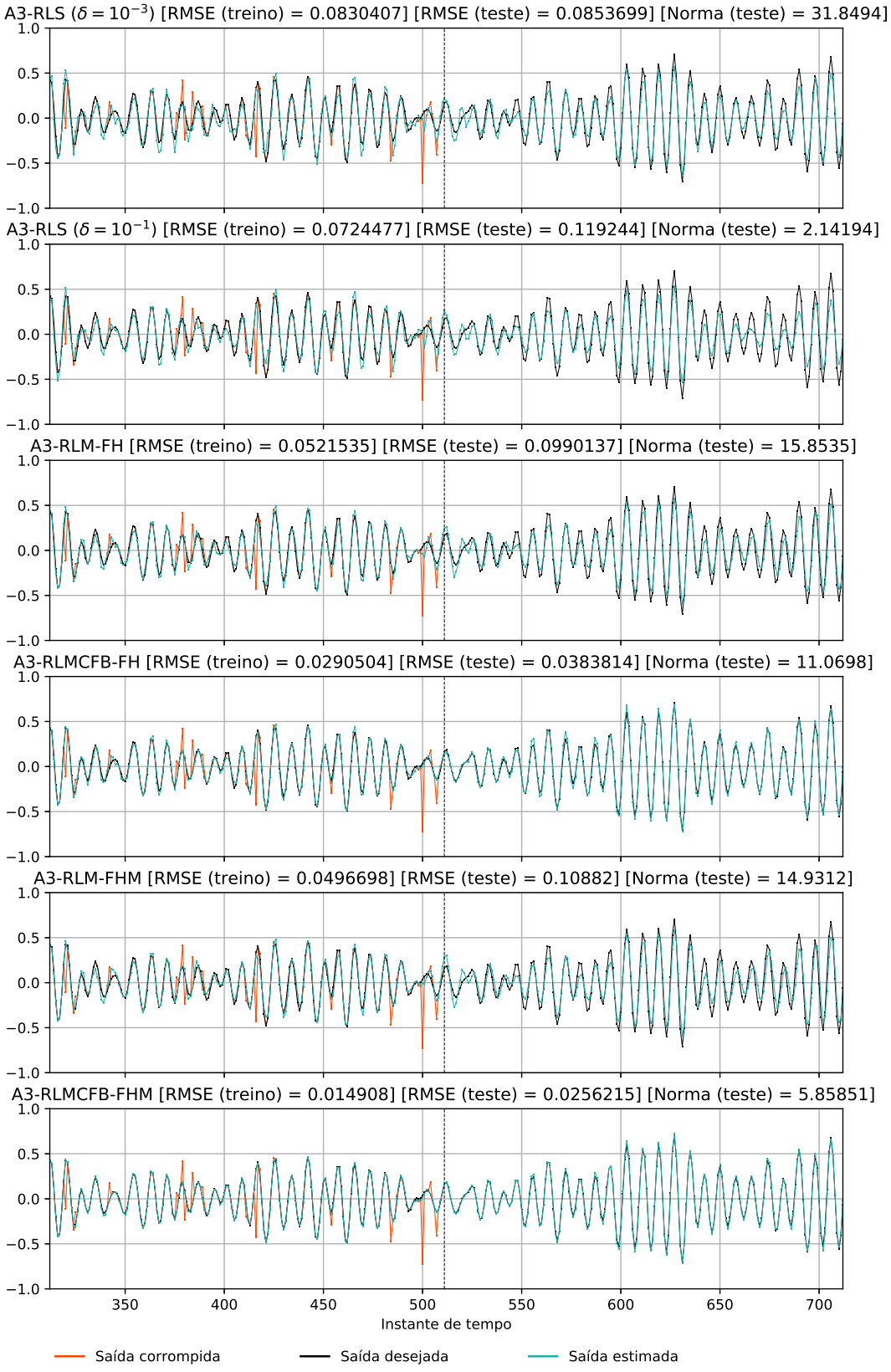
Fonte: Autoria própria.

Figura 27 – Sistema *Robot Arm*: gráfico de desempenho entre o algoritmo RLM e RLMCFB na presença de *outliers*.



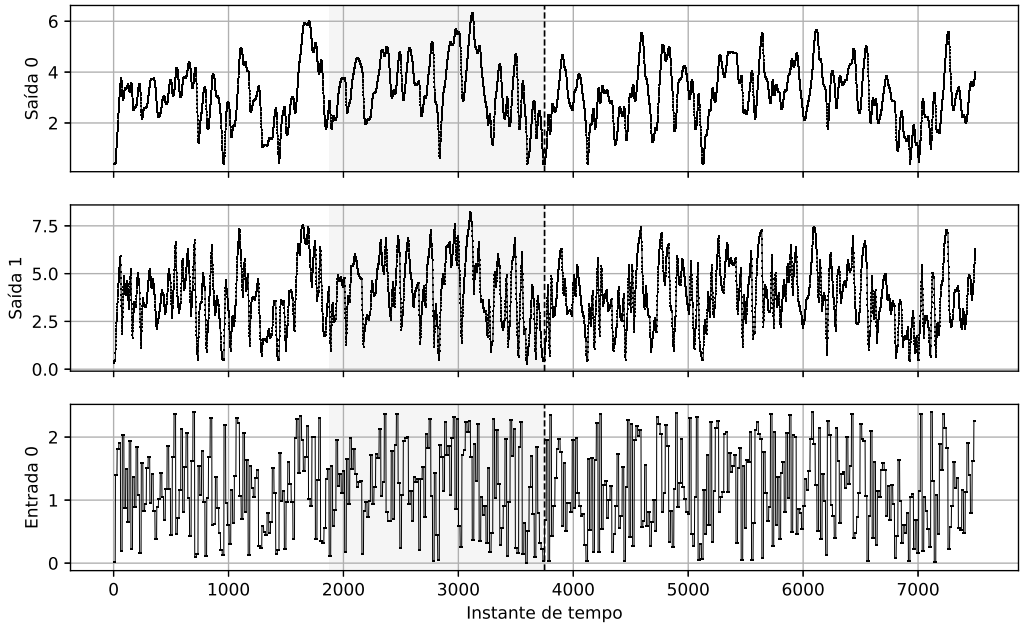
Fonte: Autoria própria.

Figura 28 – Sistema *Robot Arm*: curvas de previsão considerando diferentes algoritmos de estimação para o cenário de 15% de contaminação por *outlier*.



Fonte: Autoria própria.

Figura 29 – Sistema *Tank*: curvas de entrada e de saída.



Fonte: Autoria própria.
Nota: A área cinza representa o intervalo que o RMSE é calculado durante o treinamento e a linha vertical tracejada a divisão do conjunto de dados em treinamento e teste.

5.4 Sistema de tanques em cascata (*Tank*)

O sistema referido como *Tank* (WIGREN, 2010) consiste de um processo de controle de nível de dois tanques interligados em cascata no qual uma bomba hidráulica transporta o líquido para o tanque superior do sistema. Nesse conjundo de dados são disponibilizadas três grandezas: tensão de alimentação da bomba, nível do tanque superior e do inferior, possuindo 7500 amostras cada uma dessas, as quais foram mensuradas com o tempo de amostragem de 4 segundos. Dessa maneira, o conjunto de dados apresenta:

Entrada: Tensão aplicada à bomba;

Saída 0: Nível do tanque inferior;

Saída 1: Nível do tanque superior.

A Figura 29 exibe as curvas de entrada e de saída do sistema *Tank*.

A partir da análise dos *boxplots* presentes nas Figuras 30 e 32, percebe-se uma certa semelhança no desempenho dos hiperparâmetros e das arquiteturas para as duas saídas do sistema. Com exceção da arquitetura A0, que exibe uma resposta melhor para o nível do tanque superior (saída 1). Todavia, as duas saídas alcançam bons resultados para as arquiteturas A2 e

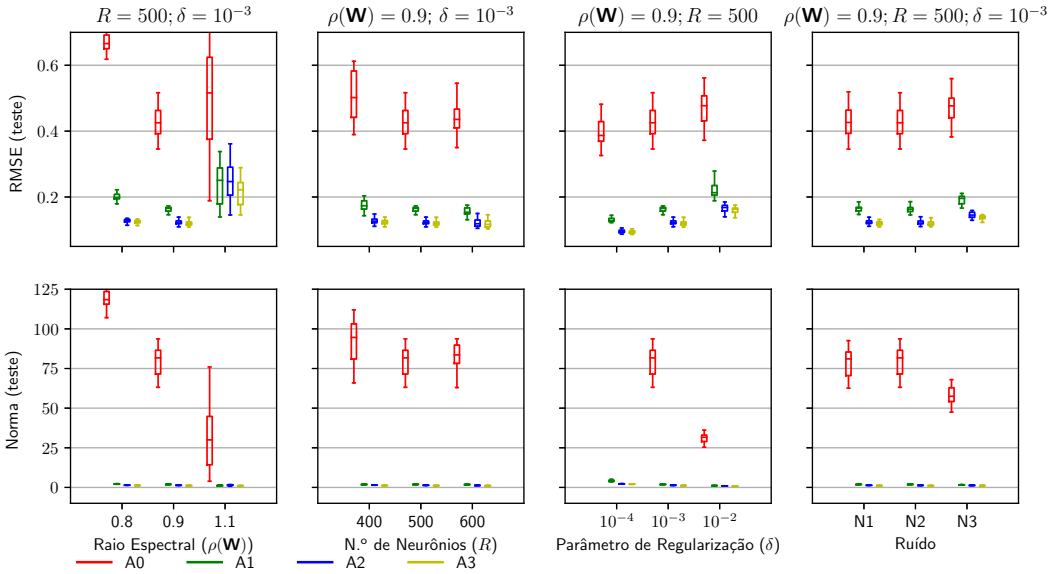
A3, o que pode ser confirmado nos gráficos das Figuras 31 e 33.

Agora, nos cenários contaminados para as arquiteturas A2 e A3, os *boxplots* das Figuras 34 e 37 apresentam uma leve diferença entre os valores de RMSE do teste para os estimadores. Contudo, o algoritmo RLM evita que a norma se eleve como no caso do RLS ($\delta = 10^{-3}$). Ainda, vale ressaltar que a FHM apresenta instabilidade na previsão durante o treinamento.

Seguindo para as Figuras 35 e 38, seus *bloxplots* exibem um desempenho superior do algoritmo RLMCFB-FH, principalmente para o nível do tanque superior (saída 1). Ademais, o estimador RLMCFB-FHM apresenta um comportamento mais estável do que o RLM-FHM.

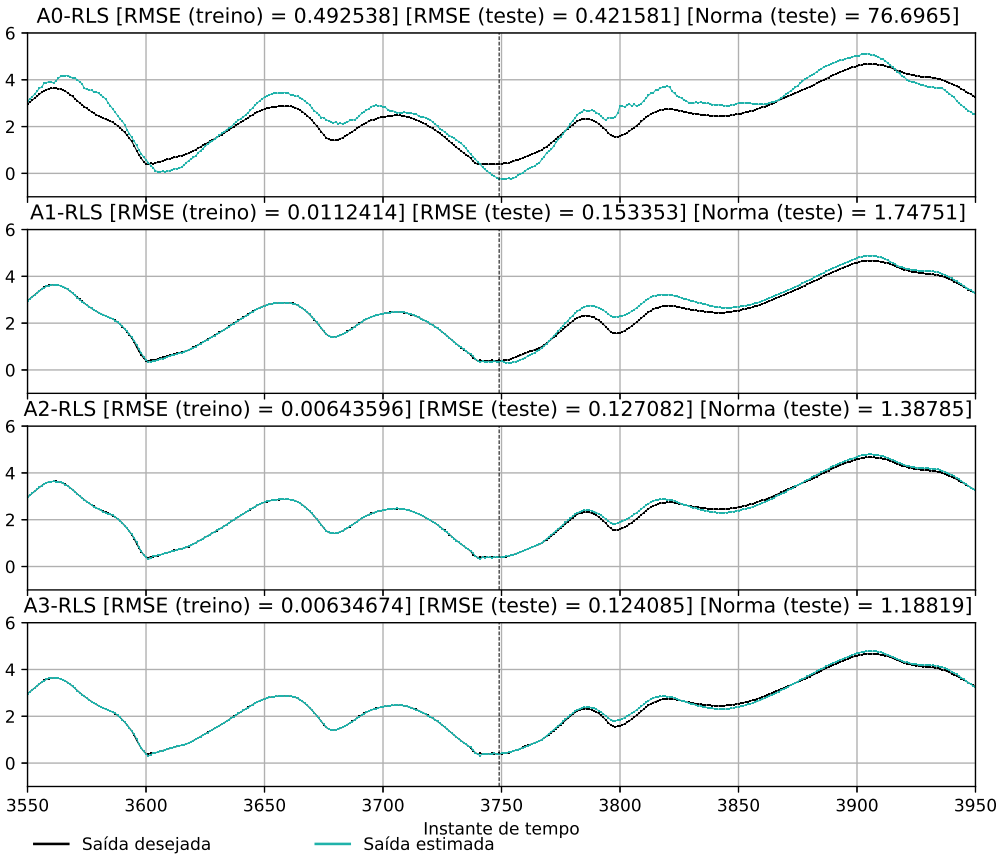
Apesar da pouca diferença do RMSE entre os modelos, as Figuras 36 e 39 mostram as curvas das saídas do sistema para o cenário de 15% de contaminação em que é perceptível a maior suavidade resultante dos modelos baseados no algoritmo RLM, especialmente o modelo A3-RLMCFB-FH.

Figura 30 – Sistema *Tank* (saída 0): gráfico de desempenho entre hiperparâmetros e arquiteturas da ESN.



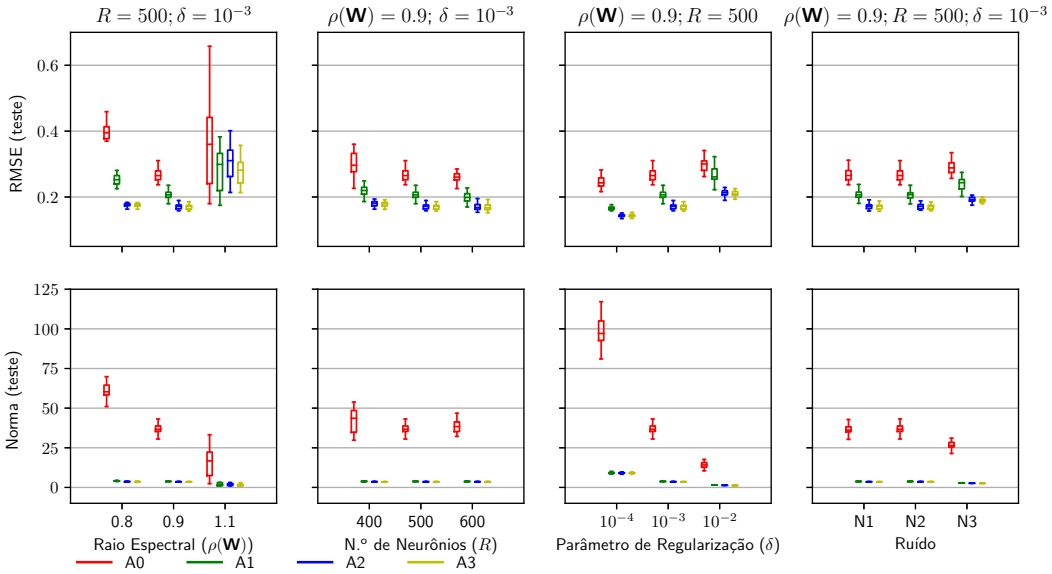
Fonte: Autoria própria.

Figura 31 – Sistema *Tank* (saída 0): curvas de previsão considerando diferentes arquiteturas.



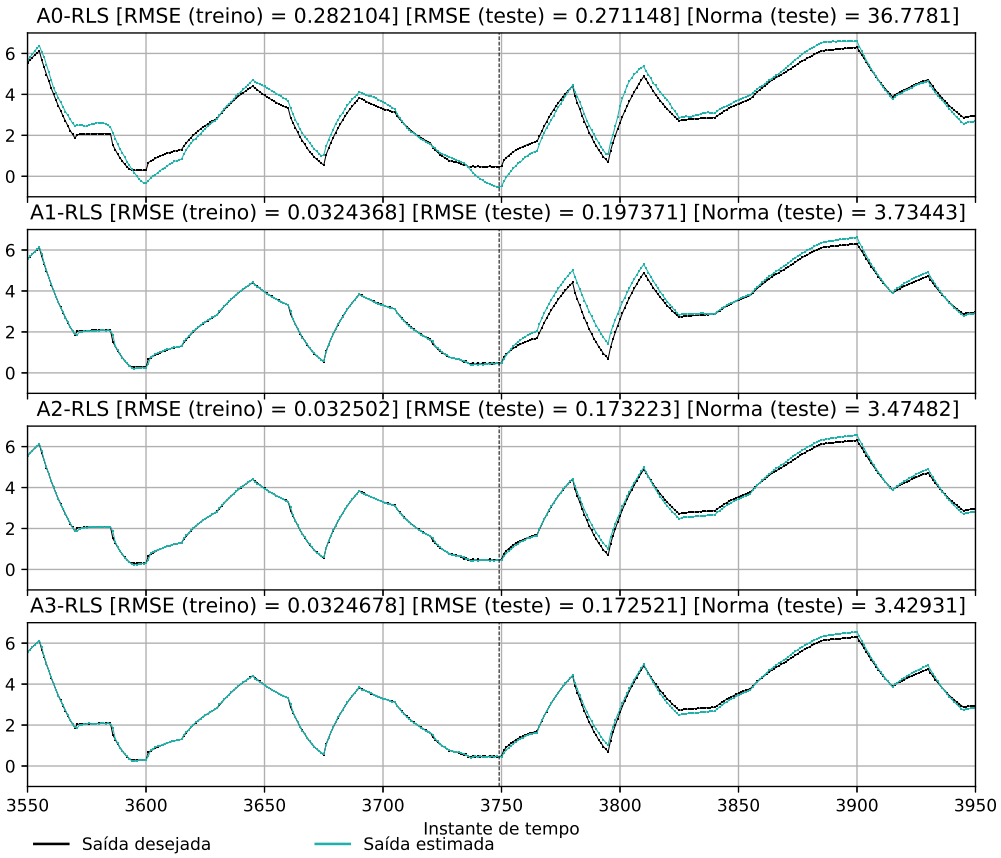
Fonte: Autoria própria.

Figura 32 – Sistema *Tank* (saída 1): gráfico de desempenho entre hiperparâmetros e arquiteturas da ESN.



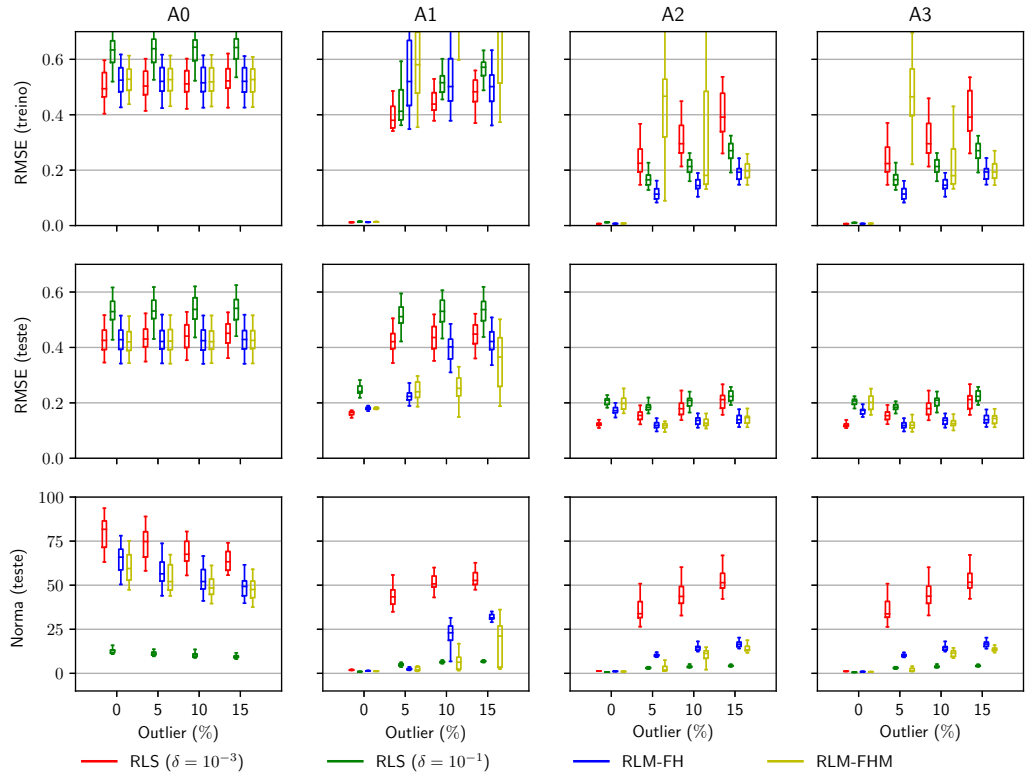
Fonte: Autoria própria.

Figura 33 – Sistema *Tank* (saída 1): curvas de previsão considerando diferentes arquiteturas.



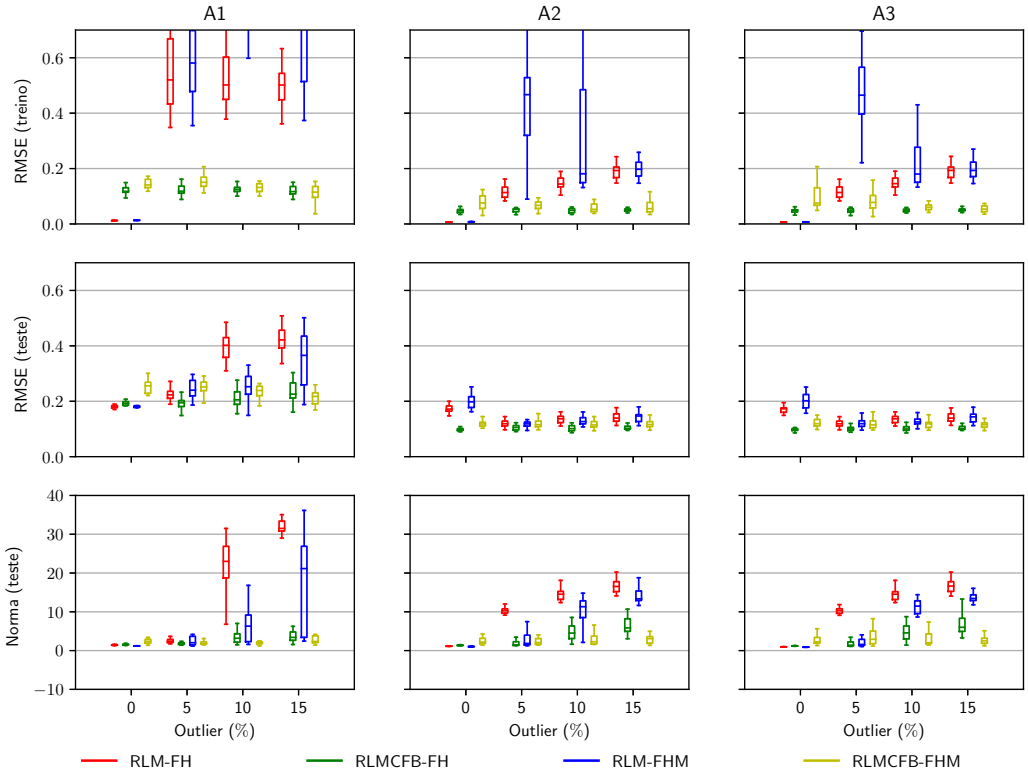
Fonte: Autoria própria.

Figura 34 – Sistema *Tank* (saída 0): gráfico de desempenho entre o algoritmo RLS e RLM na presença de *outliers*.



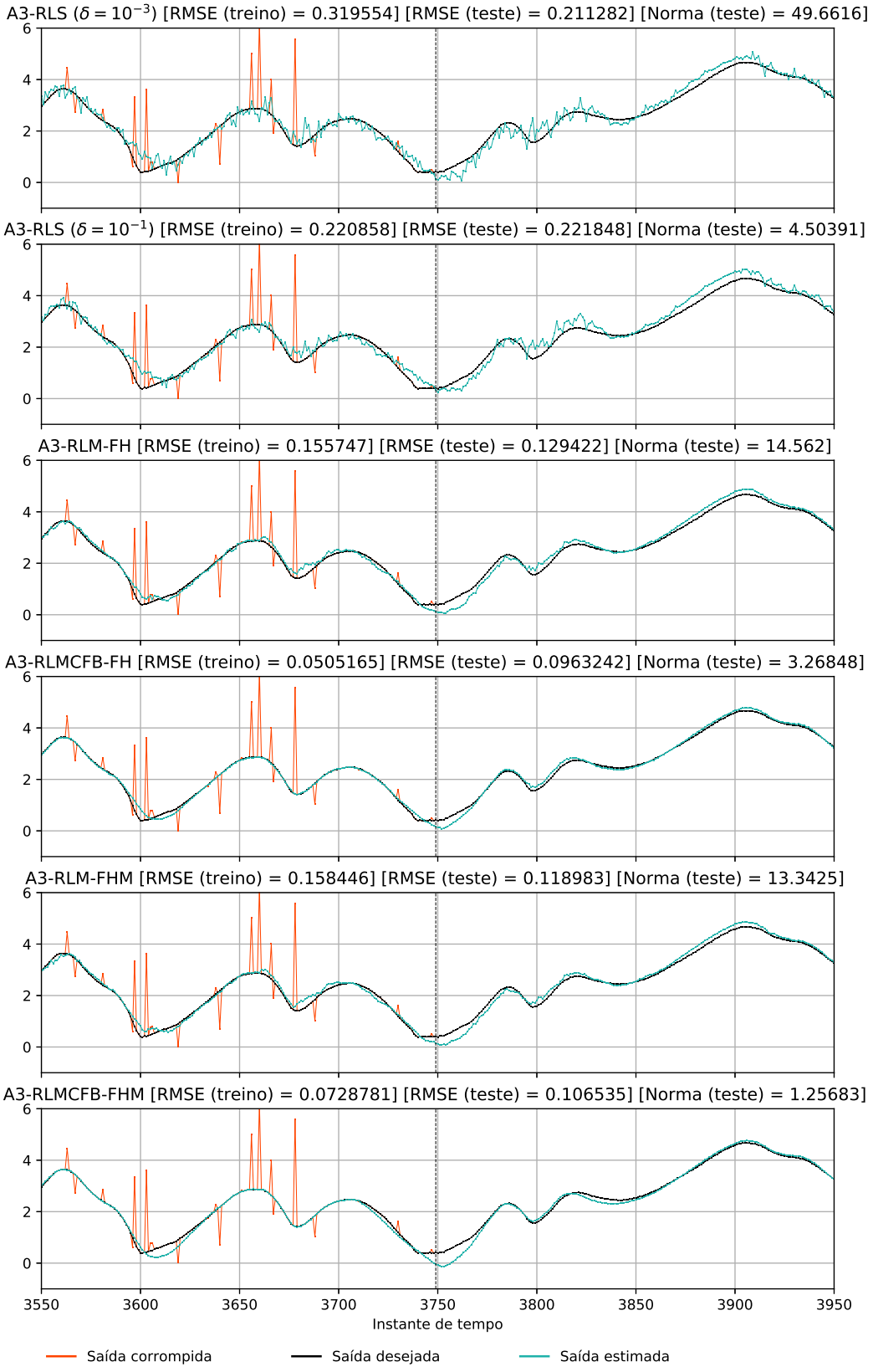
Fonte: Autoria própria.

Figura 35 – Sistema *Tank* (saída 0): gráfico de desempenho entre o algoritmo RLM e RLMCFB na presença de *outliers*.



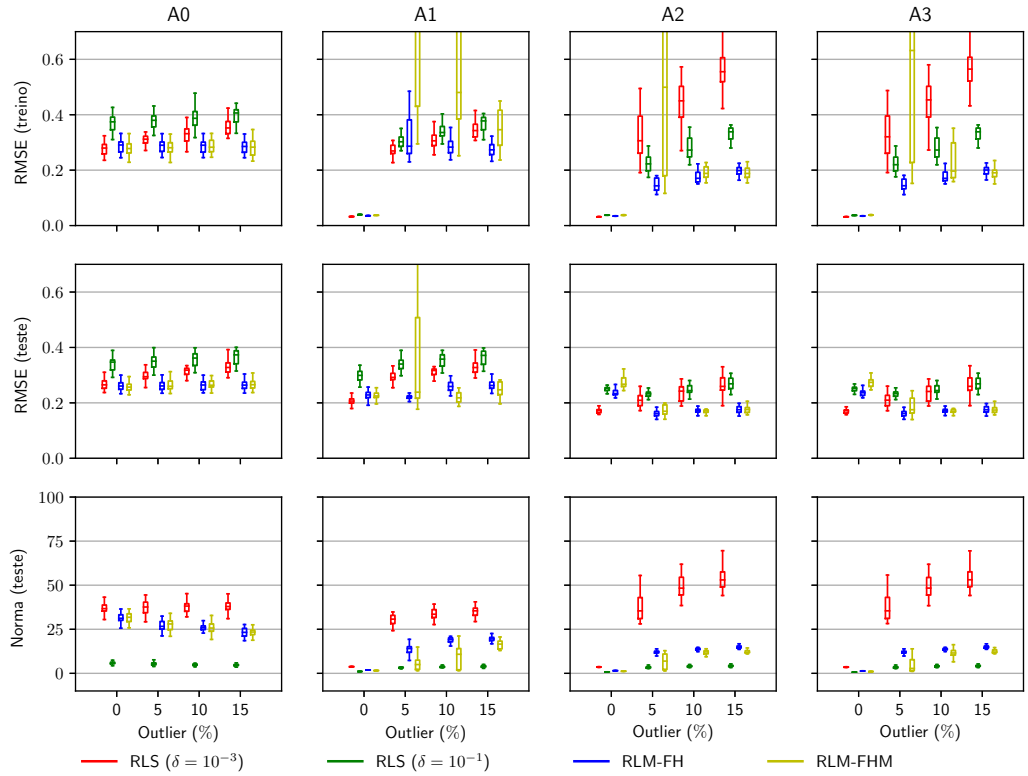
Fonte: Autoria própria.

Figura 36 – Sistema *Tank* (saída 0): curvas de previsão considerando diferentes algoritmos de estimação para o cenário de 15% de contaminação por *outlier*.



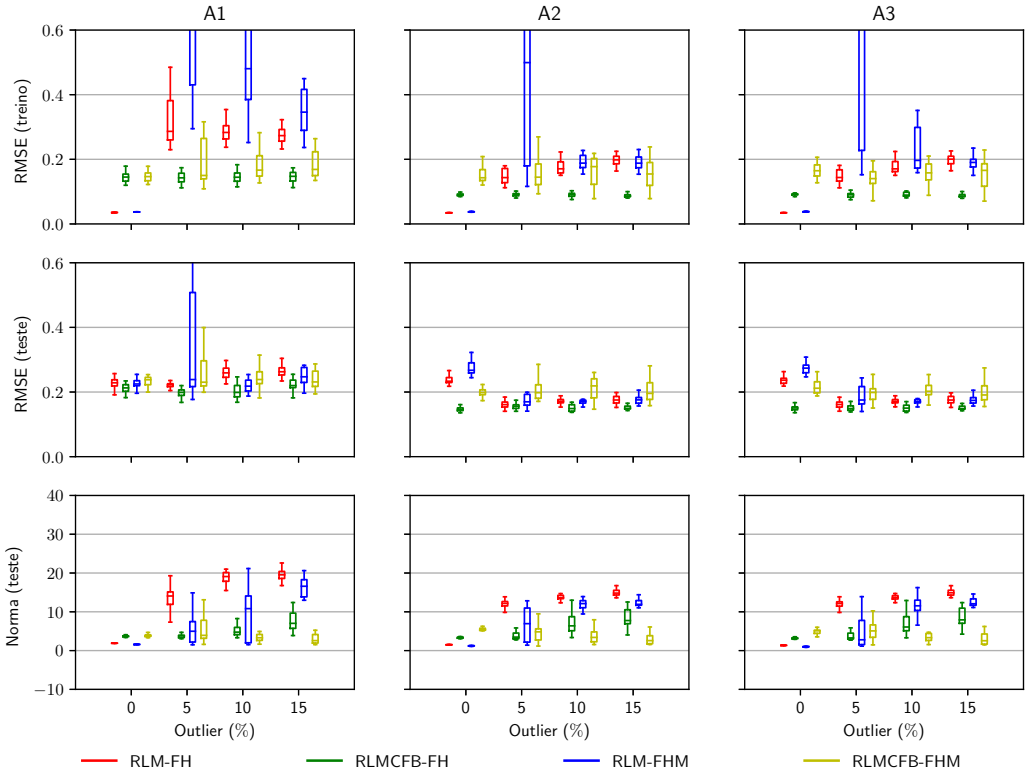
Fonte: Autoria própria.

Figura 37 – Sistema *Tank* (saída 1): gráfico de desempenho entre o algoritmo RLS e RLM na presença de *outliers*.



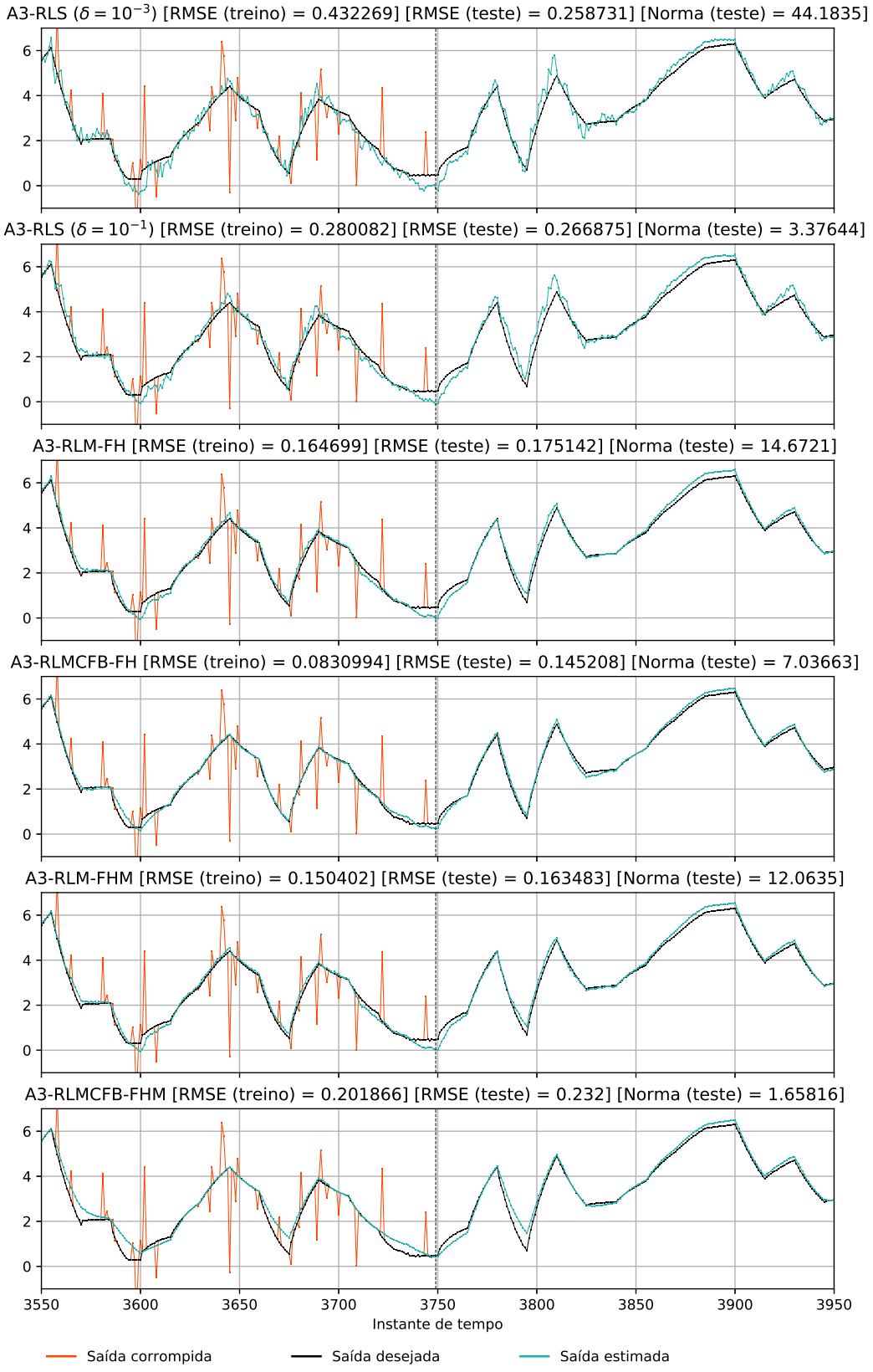
Fonte: Autoria própria.

Figura 38 – Sistema *Tank* (saída 1): gráfico de desempenho entre o algoritmo RLM e RLMCFB na presença de *outliers*.



Fonte: Autoria própria.

Figura 39 – Sistema *Tank* (saída 1): curvas de previsão considerando diferentes algoritmos de estimação para o cenário de 15% de contaminação por *outlier*.



Fonte: Autoria própria.

5.5 Sistema de caldeira industrial (*Steam*)

O conjunto de dados do sistema de caldeira industrial (MOOR *et al.*, 1997), mencionado como *Steam*, é relativo a um modelo da usina de energia Aboott, localizada na Universidade de Illinois em Urbana-Champaign (IL-EUA) elaborado por Pellegrinetti e Bentsman (1996). Mais especificamente, trata-se da simulação de um dos geradores de vapor de recuperação de calor presente na usina, que tem seus dados disponibilizado por Espinosa e Vandewalle (1998). Assim, são disponíveis as principais entradas controláveis:

Entrada 0: Taxa de fluxo de combustível;

Entrada 1: Taxa de fluxo de ar;

Entrada 2: Taxa de fluxo de água;

Entrada 3: Índice de perturbação definido pelo nível de carga,

isto é, o efeito de perturbações (variação na demanda de vapor, ruído do sensor), de incerteza do modelo (por exemplo, variações do poder calorífico do combustível, variações do coeficiente de transferência de calor, dinâmica distribuída da geração de vapor) e de restrições (restrições do atuador, vazões unidirecionais, inundação do tambor). Bem como, as principais variáveis observáveis:

Saída 0: Pressão na caldeira;

Saída 1: Excesso de oxigênio nos gases de escape;

Saída 2: Nível de água na caldeira;

Saída 3: Taxa de fluxo de vapor.

A Figura 40 exibe as curvas de entrada e de saída do sistema *Steam*.

Ao analisar os *boxplots* das Figuras 41, 43, 45 e 47, constata-se que os hiperparâmetros são subótimos, também, para esse sistema MIMO. No que se diz respeito às arquiteturas, o emprego das vias de realimentação do modelo melhora o desempenho da ESN em relação à norma e ao RSME durante o teste. Há exceção a isso apenas no caso da saída 1 (excesso de oxigênio nos gases de escape), em que o RMSE é semelhante para todas as arquiteturas como pode ser visto na Figura 43.

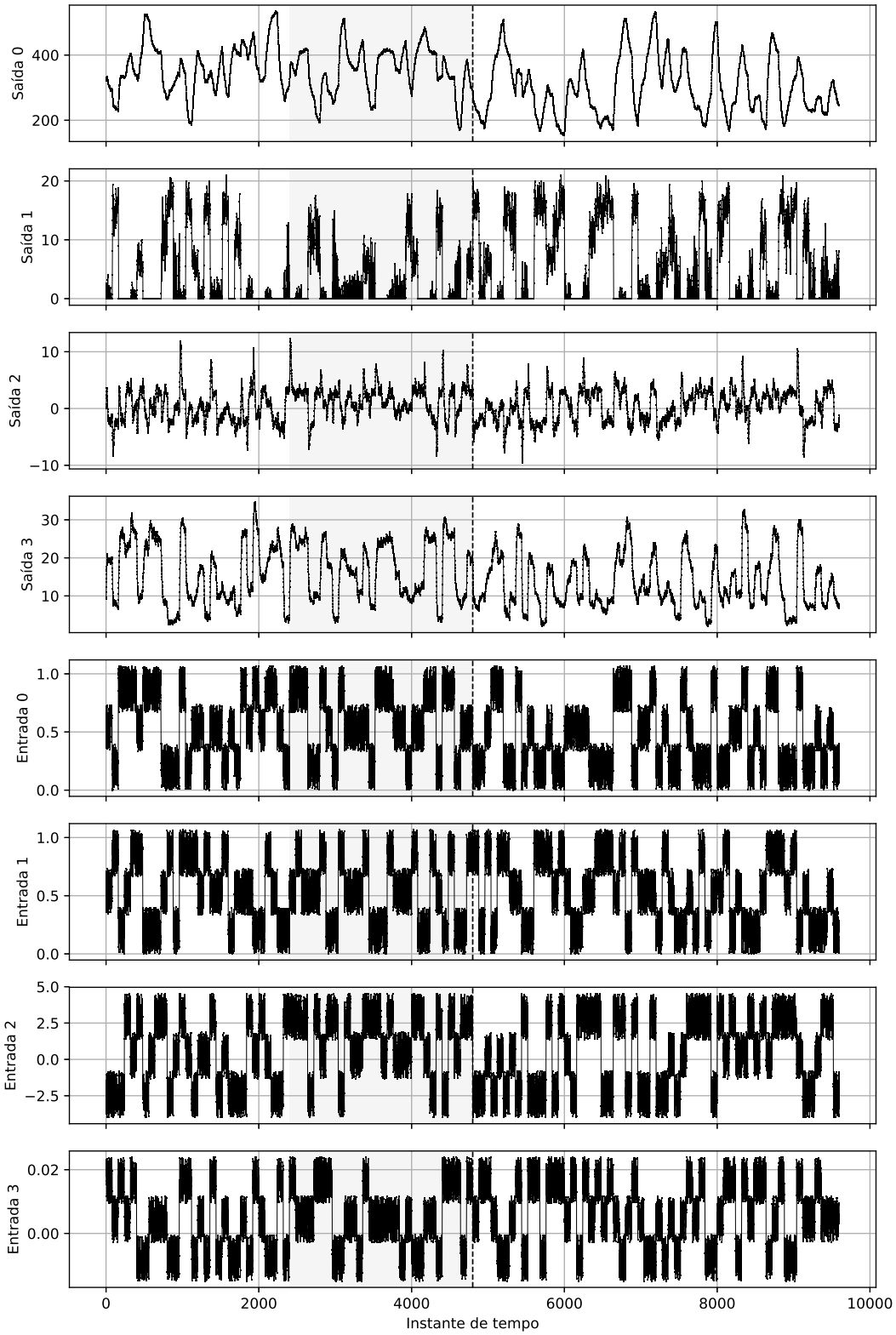
Considerando o sistema completo, a ESN consegue simular bem o sistema como pode ser visto pelas curvas das saídas para a arquitetura A3 ilustradas nas Figuras 42, 44, 46 e 48. Todavia, considerando apenas a saída 1, a ESN não é capaz de identificar essa saída com o mesmo desempenho percebido nas demais saídas do modelo.

Partindo para o exame dos modelos que empregam as arquiteturas A2 e A3 na

presença de *outlier*, os *bloxplots* presentes nas Figuras 49, 52, 55 e 58 evidenciam a robustez do algoritmo RLM-FH ao ser contrastado com os outros estimadores, mesmo no caso do RLM-FHM que apresenta instabilidade nas estimações das saídas 1 e 2 do sistema, Figuras 52 e 55 respectivamente, evidenciado pelos resultados de RMSE do treinamento e do teste.

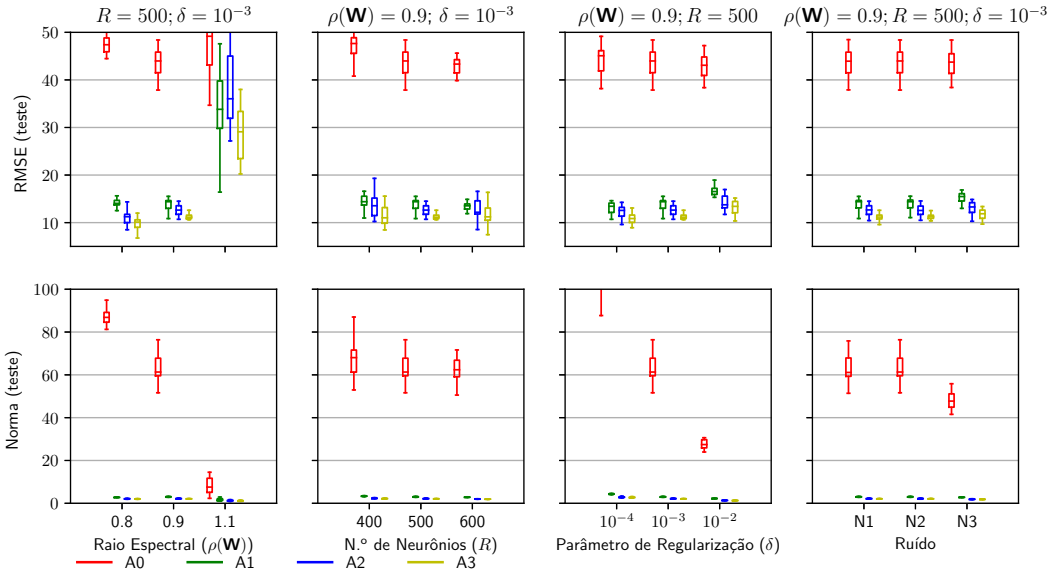
Quanto a comparação entre os algoritmos RLM e RLMCFB, exibidos nas Figuras 50, 53, 56 e 59, pode-se perceber a vantagem na abordagem ao empregar o método RLMCFB-FH, que é capaz de aprimorar o desempenho do RLMCFB-FH para todas as saídas do modelo neural MIMO. Além disso, as curvas das saídas de uma execução para cada algoritmo de estimação são apresentadas nas Figuras 51, 54, 57 e 60, onde o melhor resultado é apresentado pelo modelo A3-RLMCFB-FM para todos os casos.

Figura 40 – Sistema *Steam*: curvas de entradas e saídas.



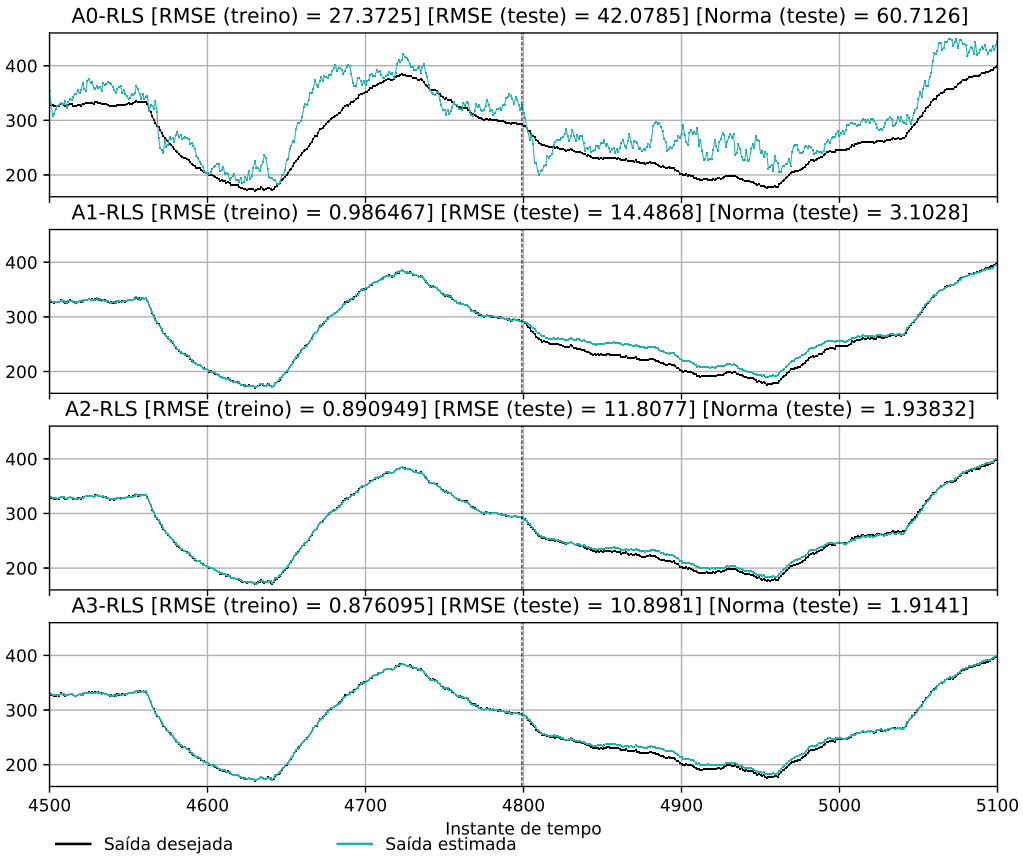
Fonte: Autoria própria.
Nota: A área cinza representa o intervalo que o RMSE é calculado durante o treinamento e a linha vertical tracejada a divisão do conjunto de dados em treinamento e teste.

Figura 41 – Sistema *Steam* (saída 0): gráfico de desempenho entre hiperparâmetros e arquiteturas da ESN.



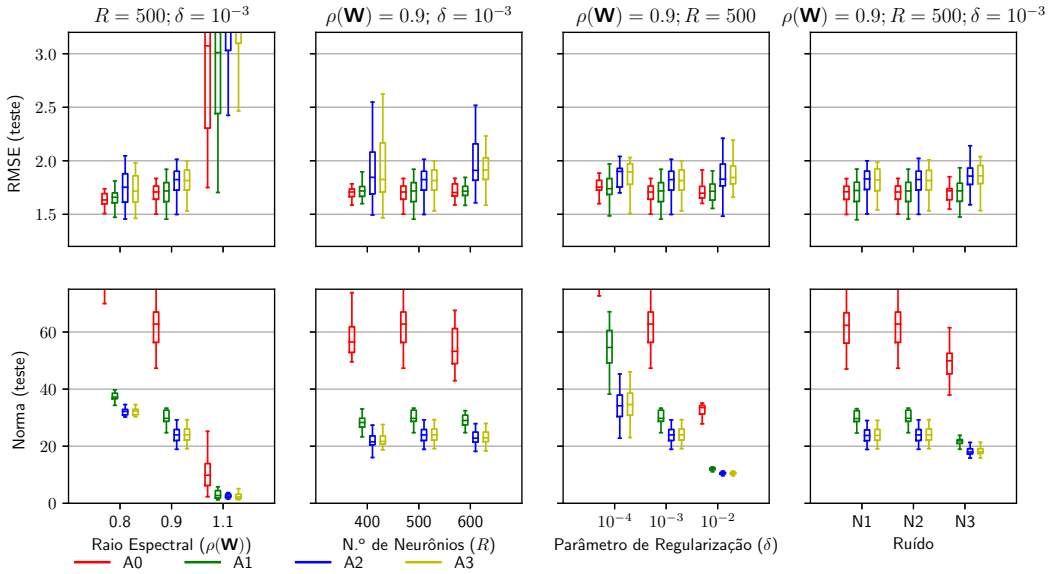
Fonte: Autoria própria.

Figura 42 – Sistema *Steam* (saída 0): curvas de previsão considerando diferentes arquiteturas.



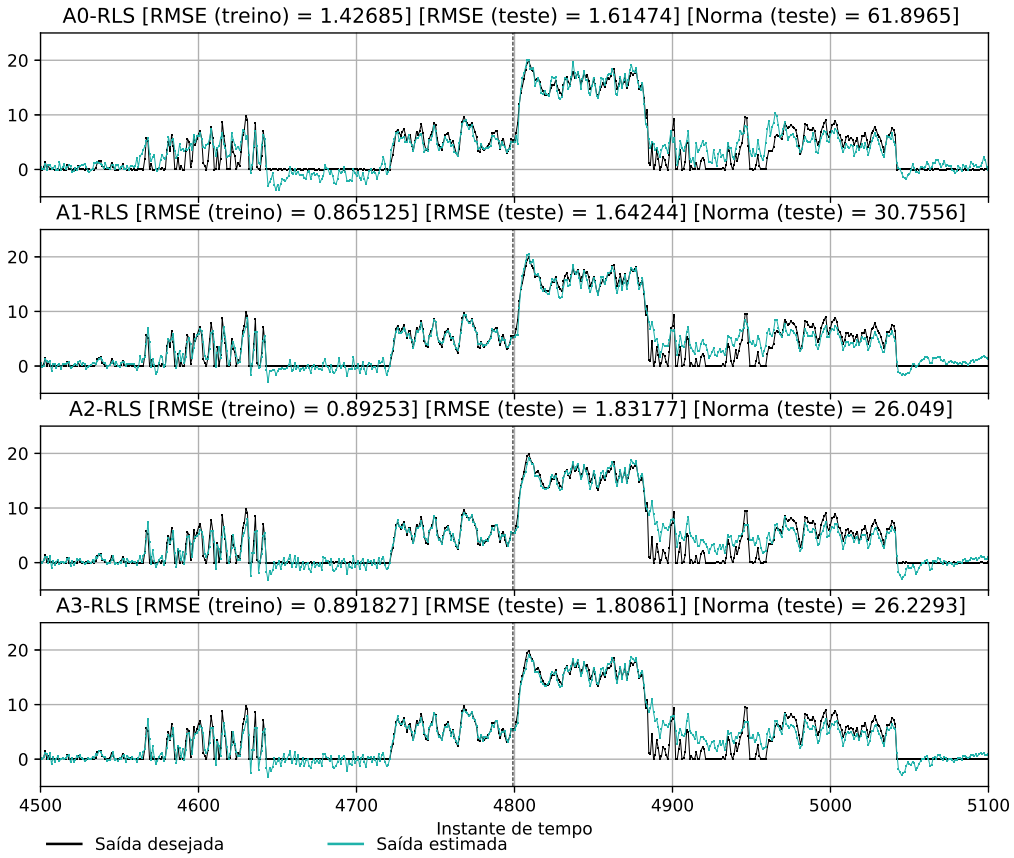
Fonte: Autoria própria.

Figura 43 – Sistema *Steam* (saída 1): gráfico de desempenho entre hiperparâmetros e arquiteturas da ESN.



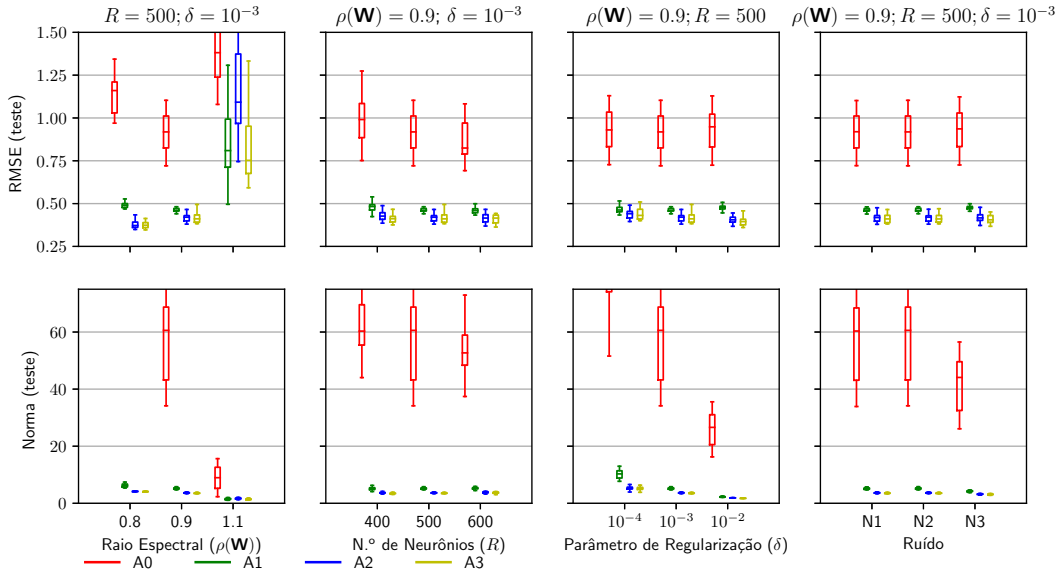
Fonte: Autoria própria.

Figura 44 – Sistema *Steam* (saída 1): curvas de previsão considerando diferentes arquiteturas.



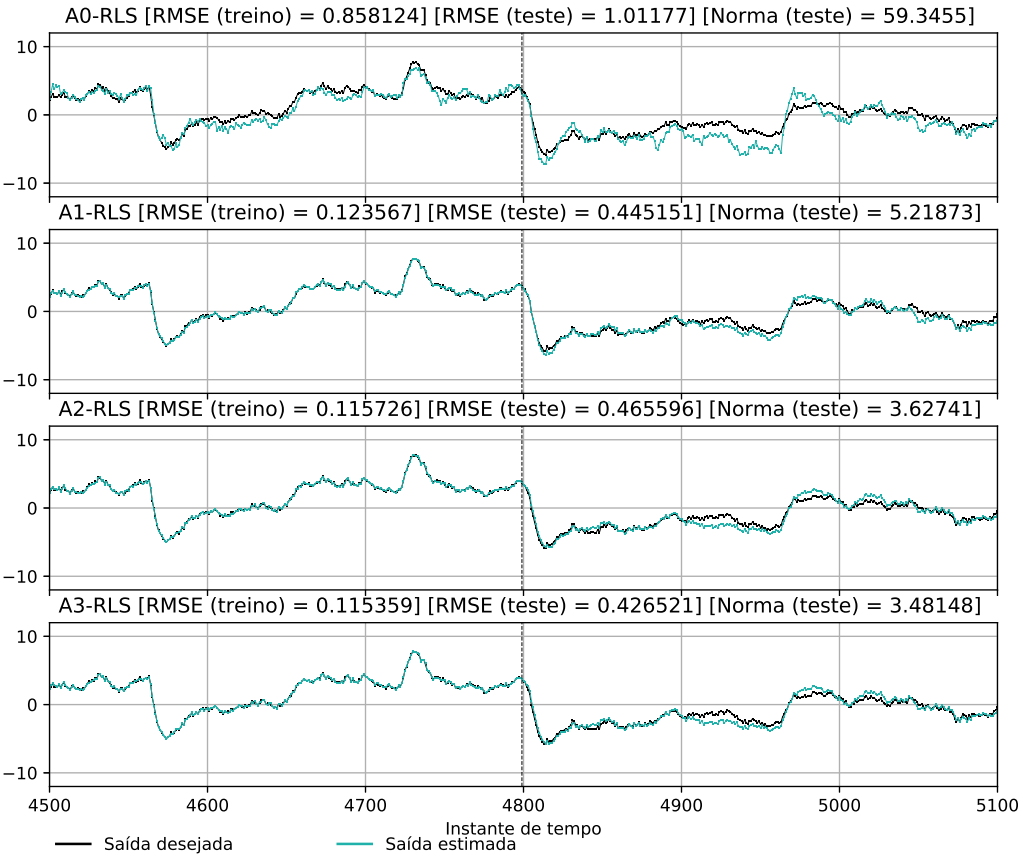
Fonte: Autoria própria.

Figura 45 – Sistema *Steam* (saída 2): gráfico de desempenho entre hiperparâmetros e arquiteturas da ESN.



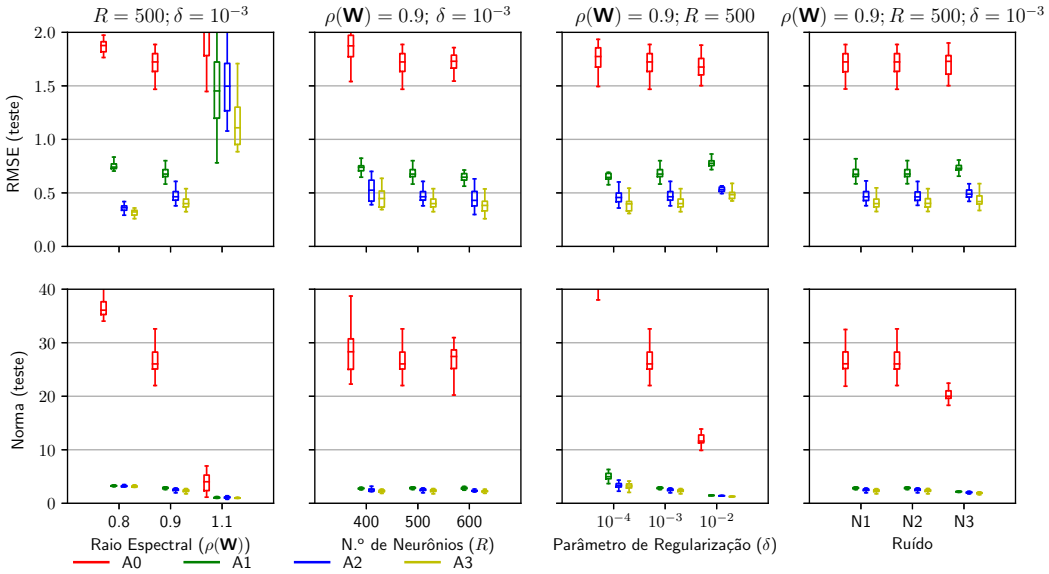
Fonte: Autoria própria.

Figura 46 – Sistema *Steam* (saída 2): curvas de previsão considerando diferentes arquiteturas.



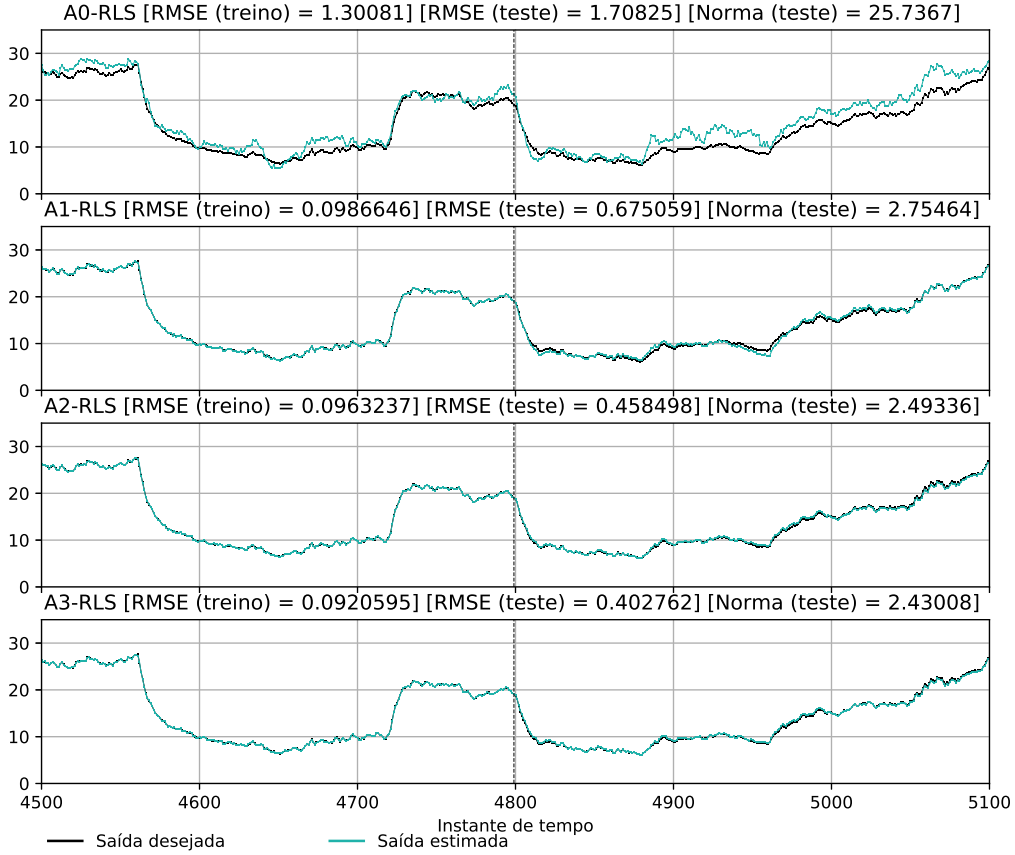
Fonte: Autoria própria.

Figura 47 – Sistema *Steam* (saída 3): gráfico de desempenho entre hiperparâmetros e arquiteturas da ESN.



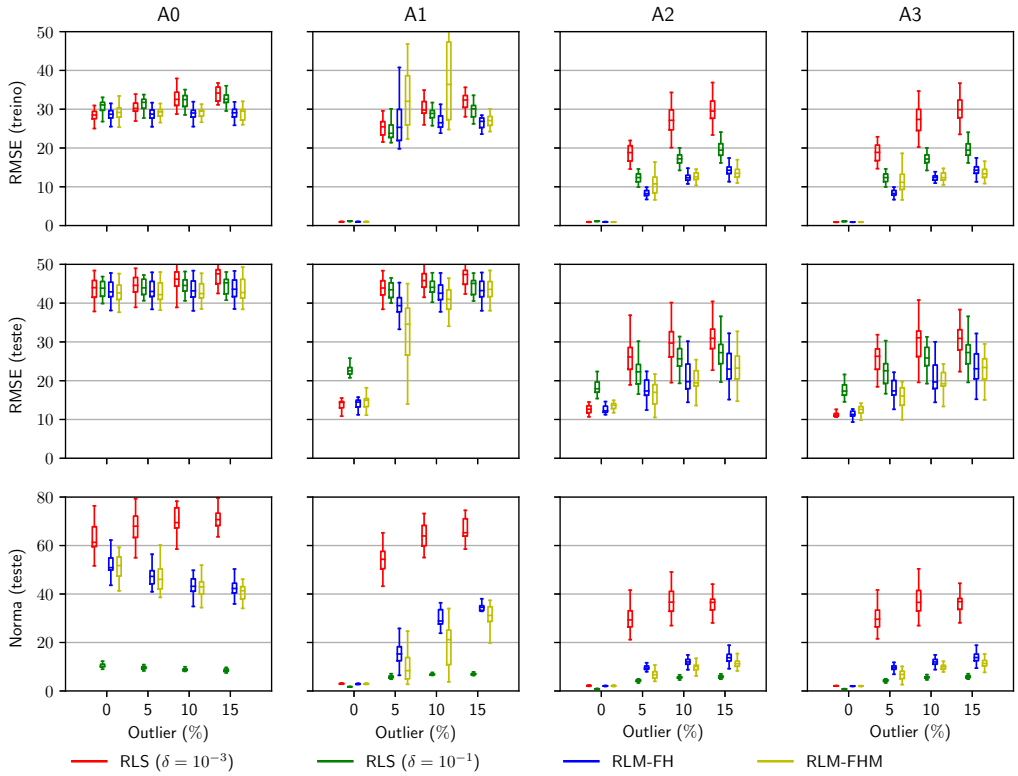
Fonte: Autoria própria.

Figura 48 – Sistema *Steam* (saída 3): curvas de previsão considerando diferentes arquiteturas.



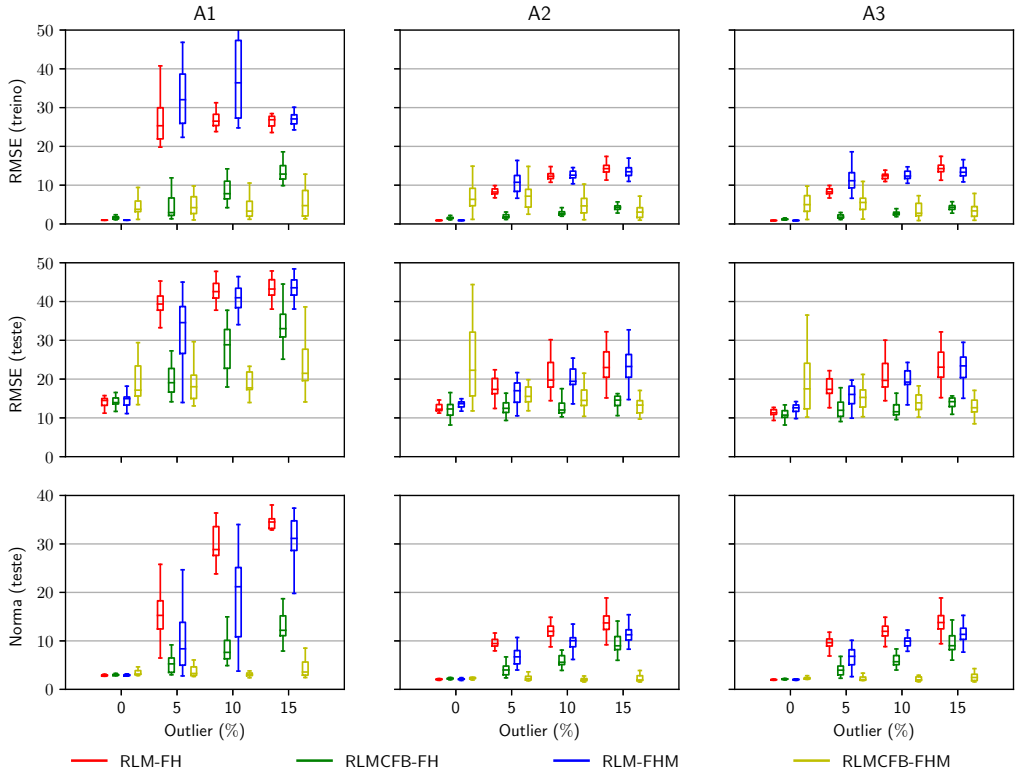
Fonte: Autoria própria.

Figura 49 – Sistema *Steam* (saída 0): gráfico de desempenho entre o algoritmo RLS e RLM na presença de *outliers*.



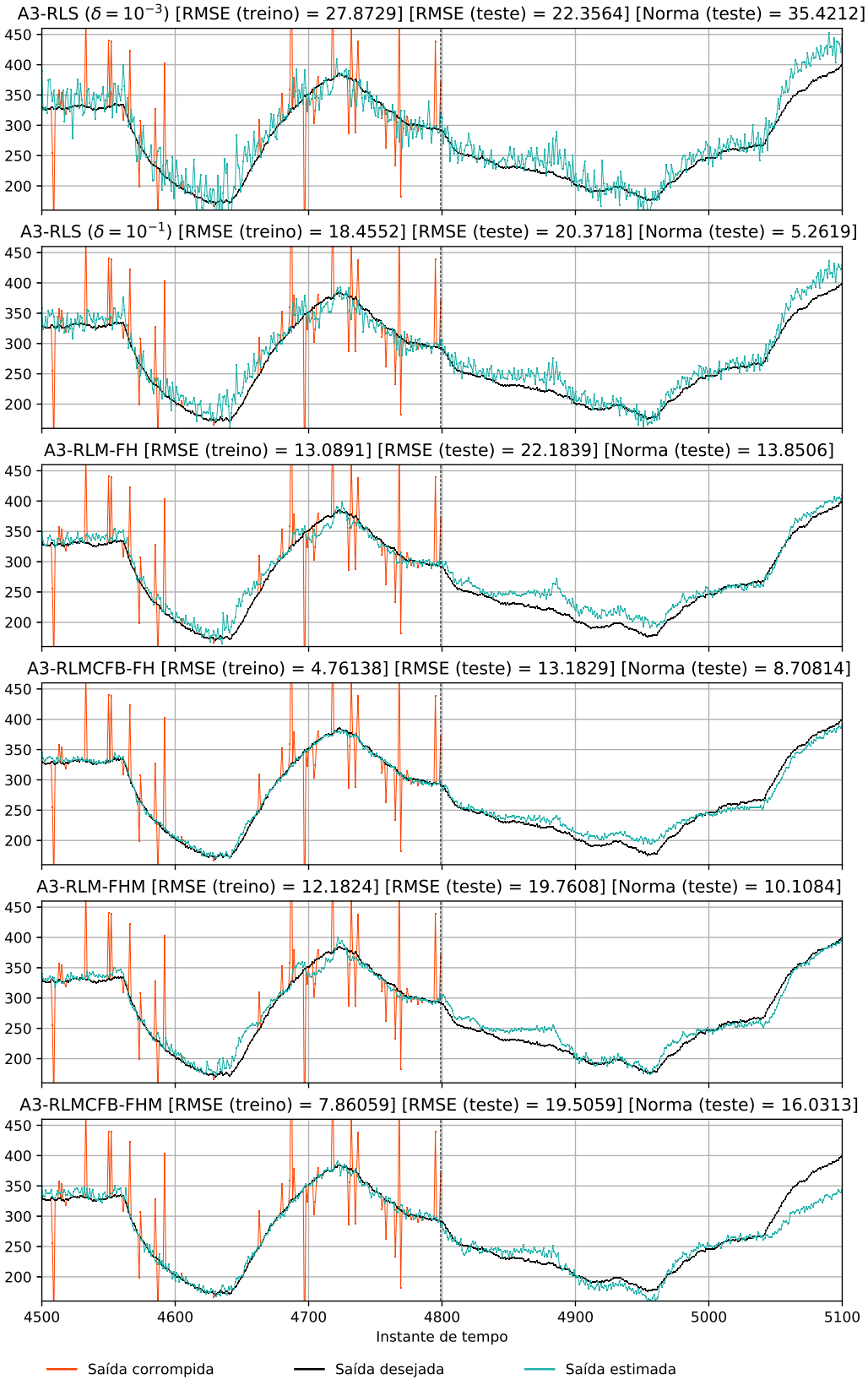
Fonte: Autoria própria.

Figura 50 – Sistema *Steam* (saída 0): gráfico de desempenho entre o algoritmo RLM e RLMCFB na presença de *outliers*.



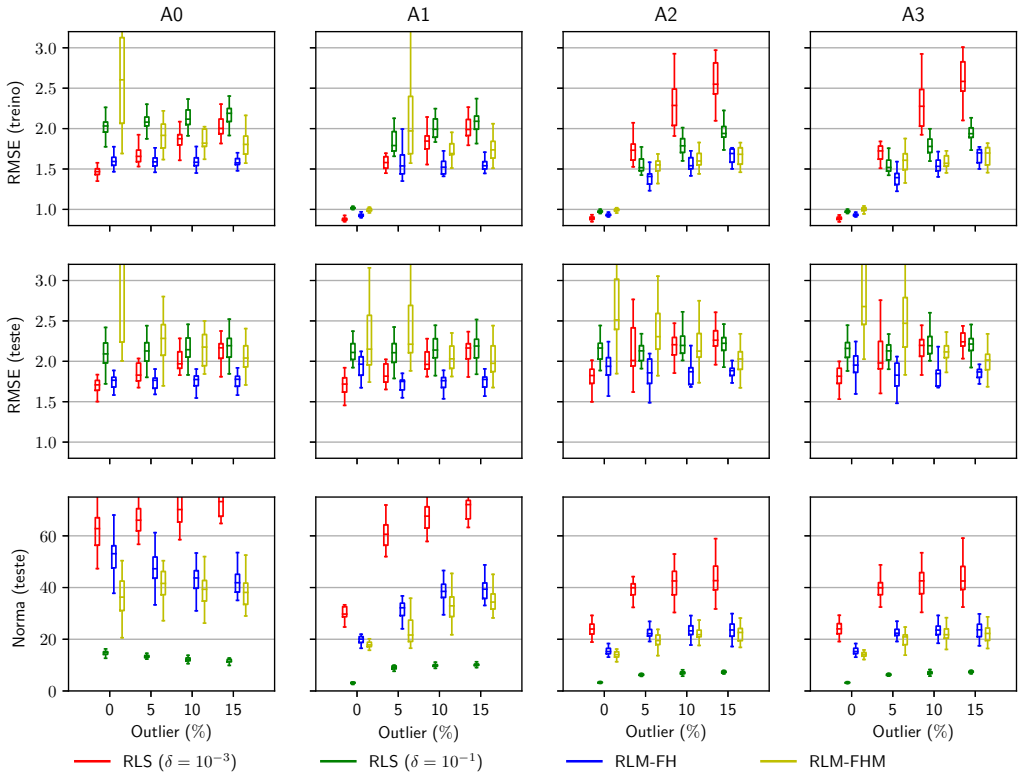
Fonte: Autoria própria.

Figura 51 – Sistema *Steam* (saída 0): curvas de previsão considerando diferentes algoritmos de estimação para o cenário de 15% de contaminação por *outlier*.



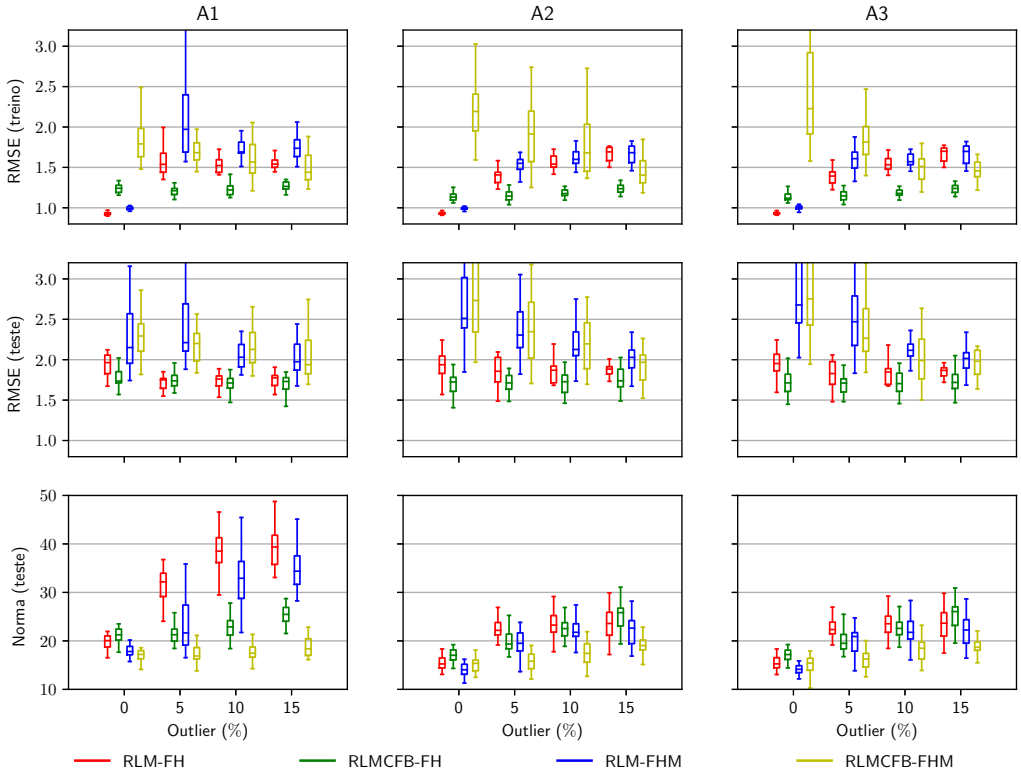
Fonte: Autoria própria.

Figura 52 – Sistema *Steam* (saída 1): gráfico de desempenho entre o algoritmo RLS e RLM na presença de *outliers*.



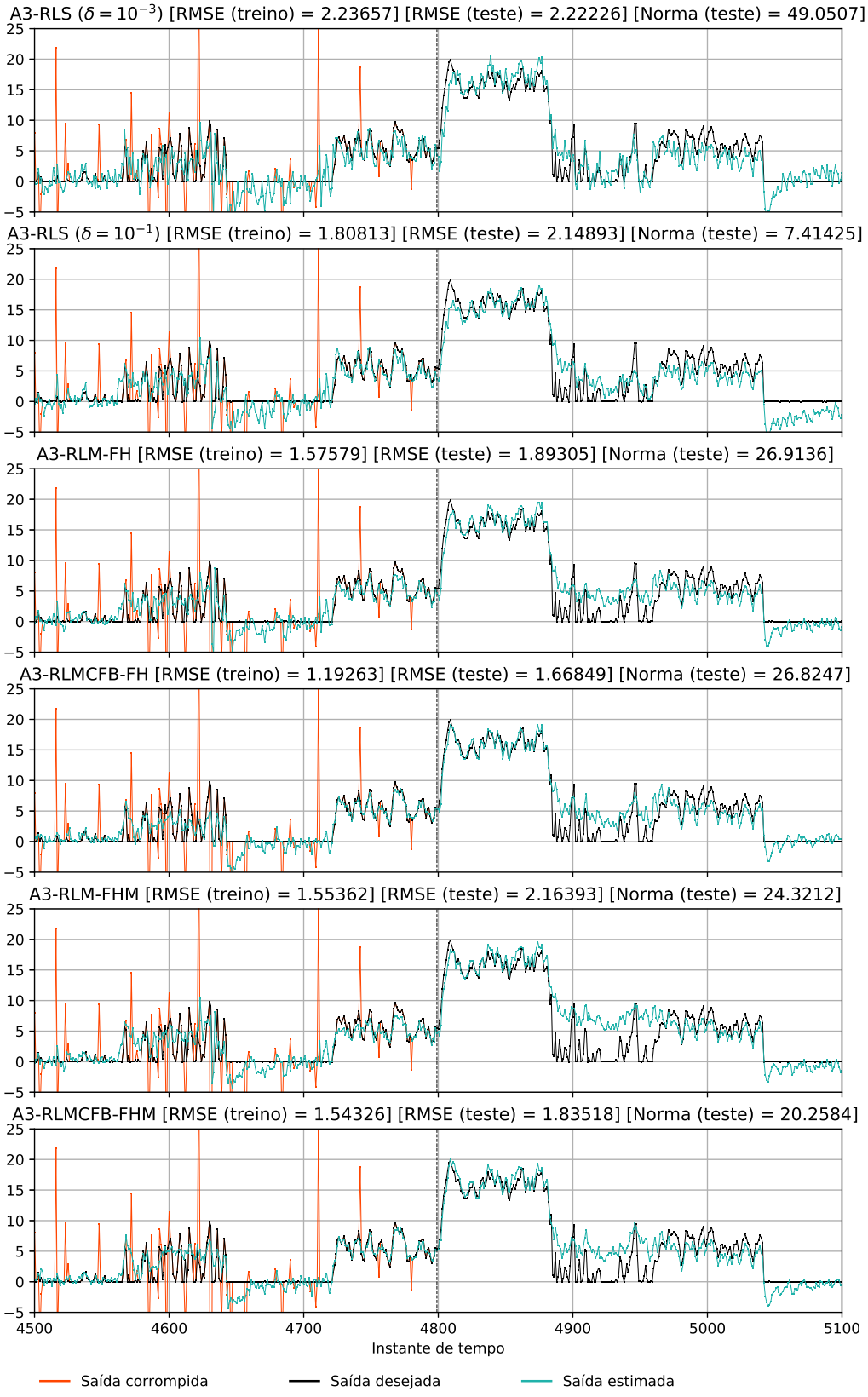
Fonte: Autoria própria.

Figura 53 – Sistema *Steam* (saída 1): gráfico de desempenho entre o algoritmo RLM e RLMCFB na presença de *outliers*.



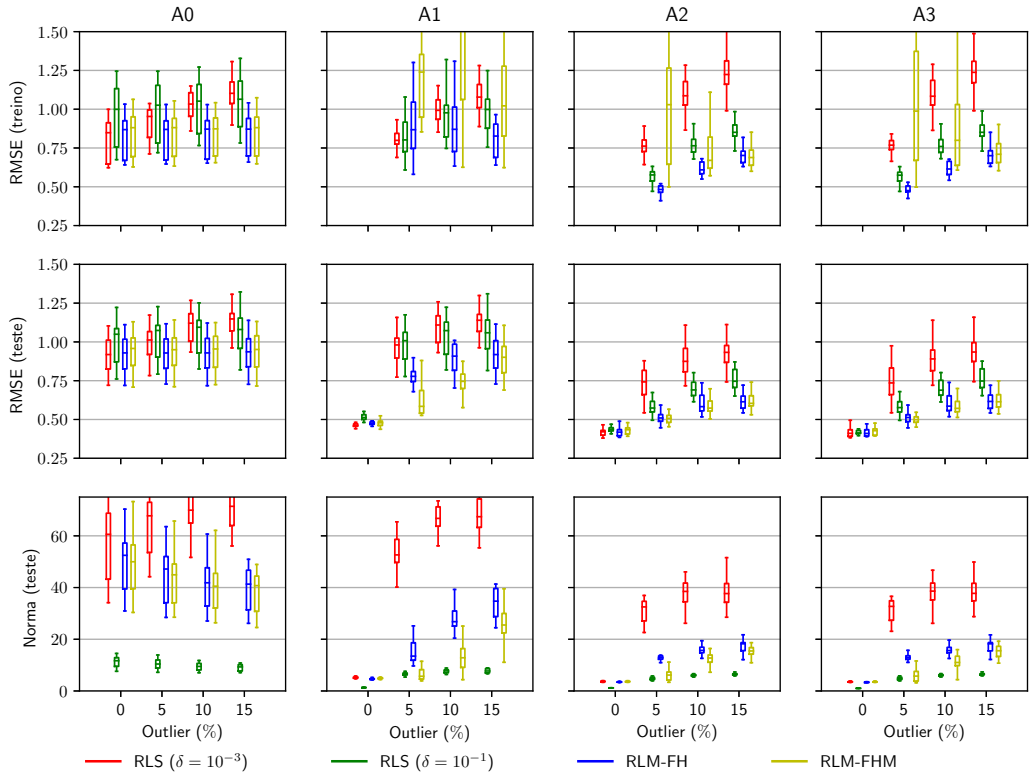
Fonte: Autoria própria.

Figura 54 – Sistema *Steam* (saída 1): curvas de previsão considerando diferentes algoritmos de estimação para o cenário de 15% de contaminação por *outlier*.



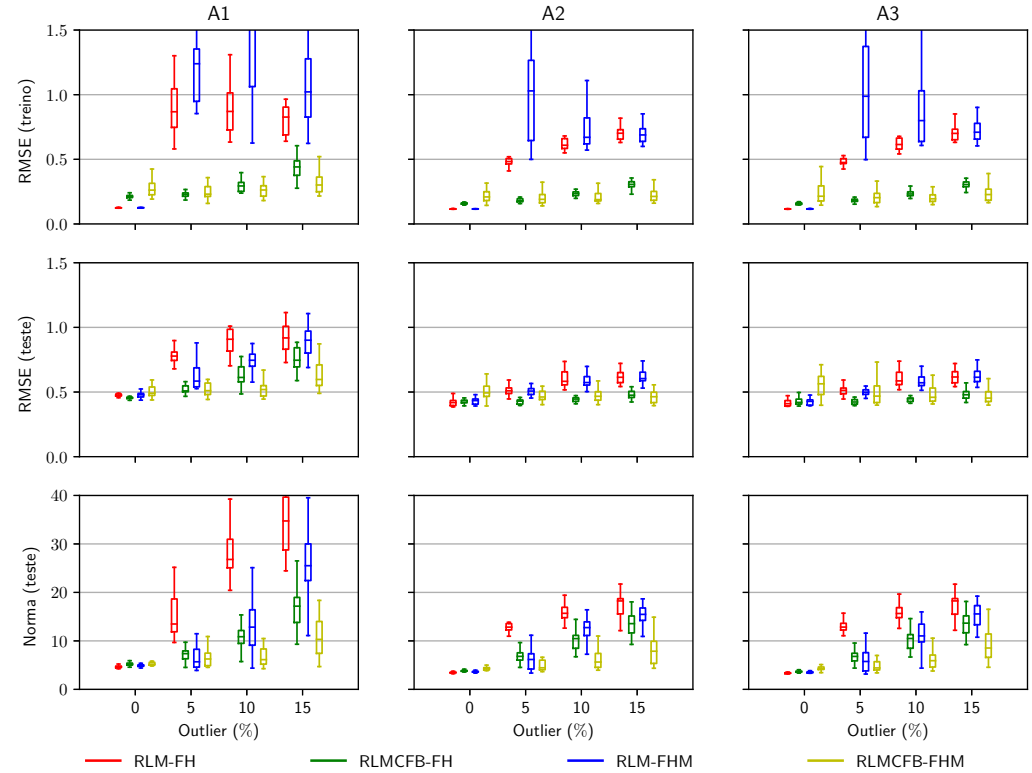
Fonte: Autoria própria.

Figura 55 – Sistema *Steam* (saída 2): gráfico de desempenho entre o algoritmo RLS e RLM na presença de *outliers*.



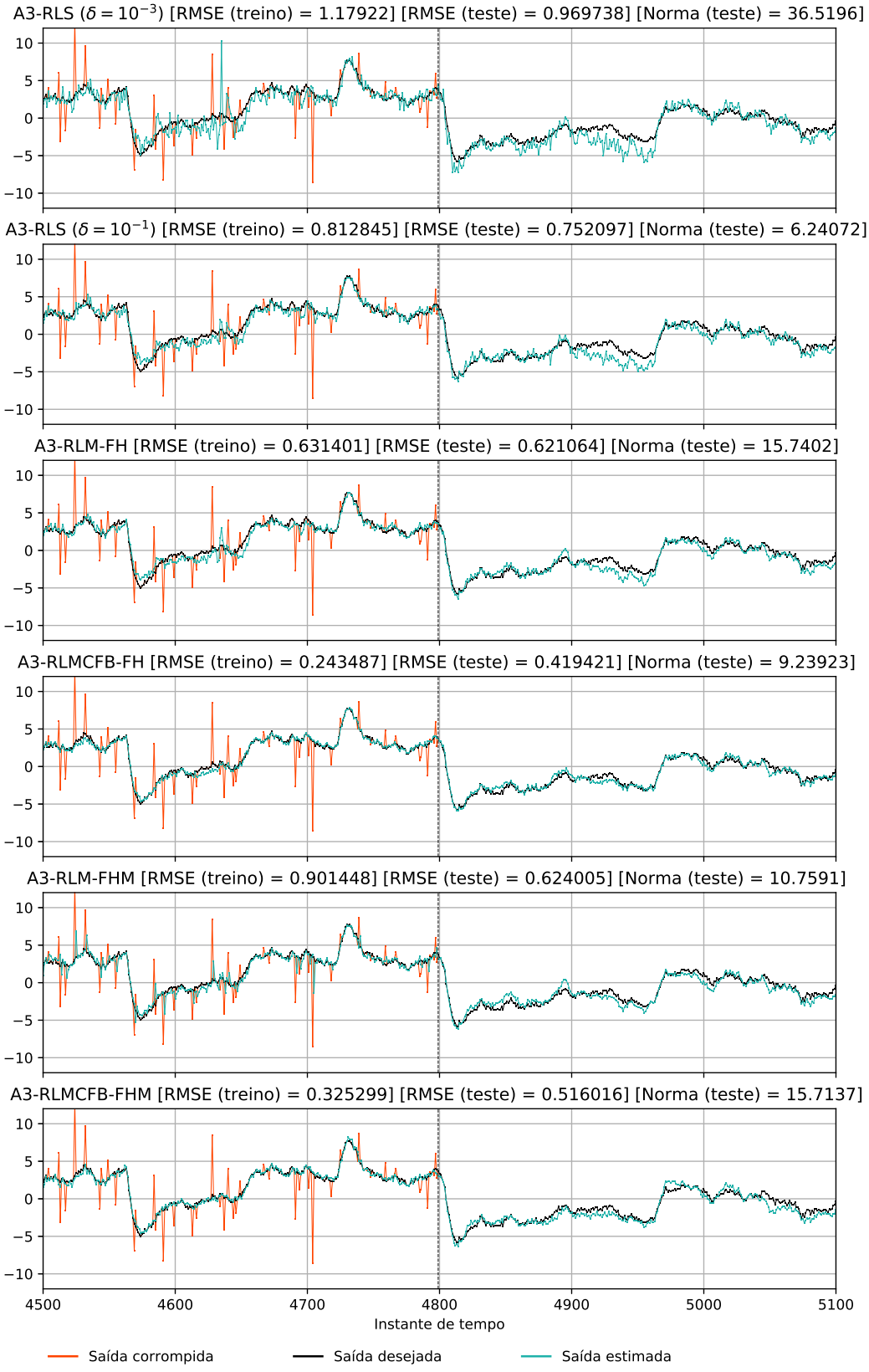
Fonte: Autoria própria.

Figura 56 – Sistema *Steam* (saída 2): gráfico de desempenho entre o algoritmo RLM e RLMCFB na presença de *outliers*.



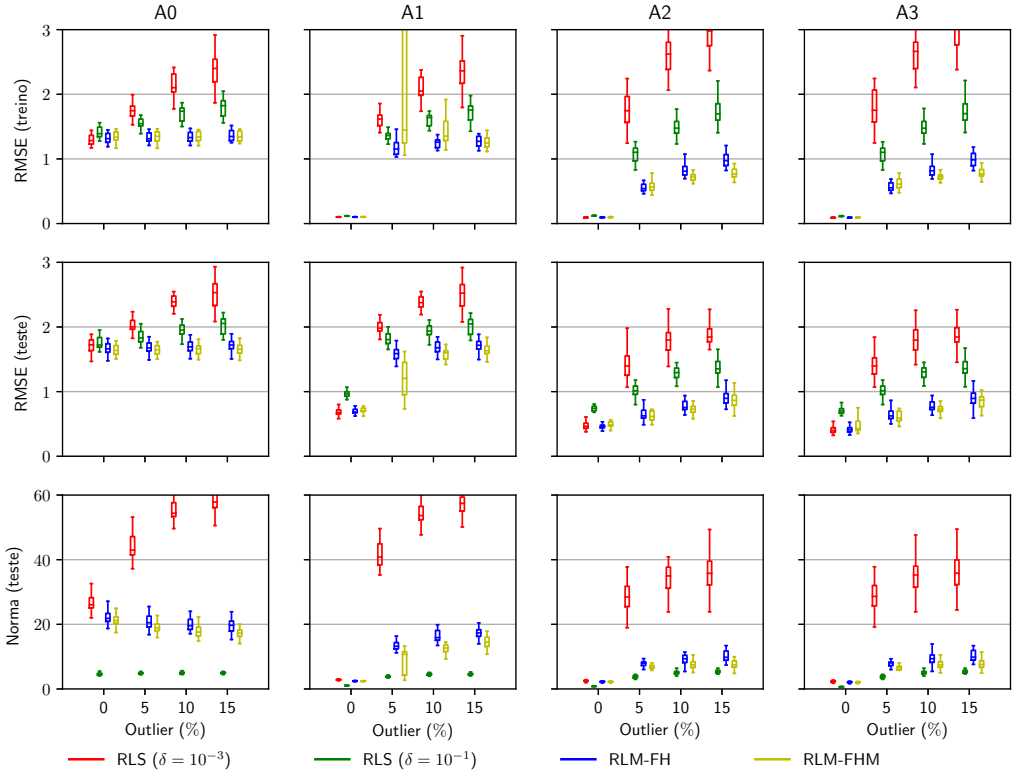
Fonte: Autoria própria.

Figura 57 – Sistema *Steam* (saída 2): curvas de previsão considerando diferentes algoritmos de estimação para o cenário de 15% de contaminação por *outlier*.



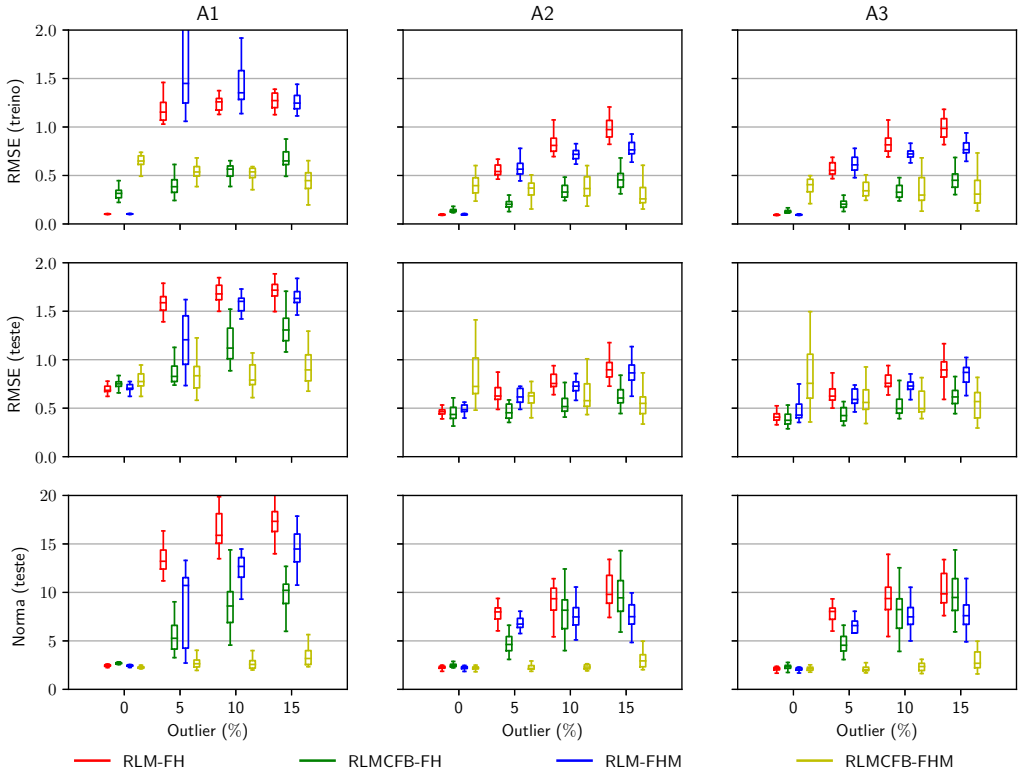
Fonte: Autoria própria.

Figura 58 – Sistema *Steam* (saída 3): gráfico de desempenho entre o algoritmo RLS e RLM na presença de *outliers*.



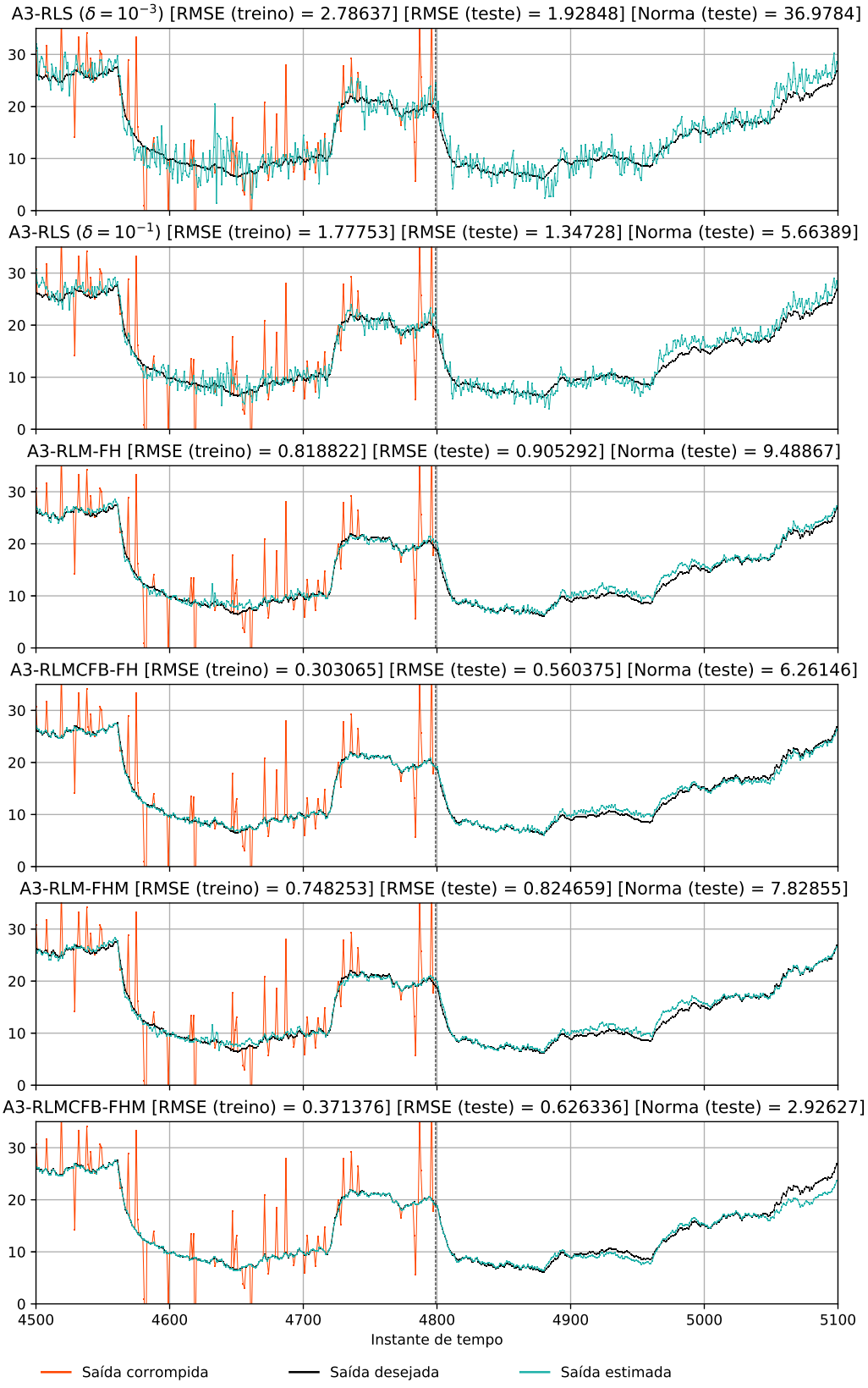
Fonte: Autoria própria.

Figura 59 – Sistema *Steam* (saída 3): gráfico de desempenho entre o algoritmo RLM e RLMCFB na presença de *outliers*.



Fonte: Autoria própria.

Figura 60 – Sistema *Steam* (saída 3): curvas de previsão considerando diferentes algoritmos de estimação para o cenário de 15% de contaminação por *outlier*.



Fonte: Autoria própria.

5.6 Conclusões

A partir da análise de um conjunto abrangente de experimentos computacionais, empregando conjuntos de dados de *benchmarking*, expõem-se algumas conclusões. Com base nos *boxplots* que avaliam as variações nos hiperparâmetros da ESN, percebe-se que as arquiteturas mantêm relativamente a diferença entre os seus desempenhos. Isso pode ser interpretado como uma ligação estreita entre a arquitetura e o sistema modelado. Além do mais, a arquitetura A3 se apresenta mais generalista, obtendo bom desempenho em todos os sistemas, com destaque para os que possuem curvas mais suaves de saída. Essa arquitetura também apresenta soluções mais regularizadas, resultando em melhoria na generalização e evitando sobreajustes aos dados, em que se pode atribuir isso as vias de realimentação, pois esse efeito é percebido desde a arquitetura A1. Ademais, os vetores de ruído, $\mathbf{v}_1$ e $\mathbf{v}_2$, não são necessários dada a normalização imposta pelo parâmetro de regularização δ e o fator de esquecimento unitário λ.

Por outro lado, pode-se inferir que a norma da camada de saída da ESN tende a ser elevada quando o sistema apresenta curvas que se modificam abruptamente, ou melhor, sinais com baixa correlação entre amostras consecutivas. Isso pode ocorrer tanto pela natureza do sistema como pela alta presença de *outliers*. Por isso, ao se avaliar os resultados na presença de *outliers*, fica perceptivo a influência das funções estimador-*M* em evitar a elevação da norma nesses cenários contaminados por atenuar os erros grandes. Em contra partida, a função de Huber modificada suavizou as curvas do modelo neural ao ponto de perder informação em alguns experimentos, sendo evidenciado nas curvas do modelo da saída do nível do tanque superior (saída 1) do sistema *Tank* e da saída de excesso de oxigênio nos gases de escape (saída 1) do sistema *Steam*. Outro ponto para os sistemas com múltiplas saídas é que os *outliers* aparecem nos regressores com maior frequência do que no experimentos com os sistemas de única saída, resultando em mais descarte de informação com os estimadores que utilizam a FHM, o que torna a atenuação da FH mais vantajosa. Todavia, a FHM possui resultados superiores para os sistemas SISOs testados.

O algoritmo RLMCFB proposto obtêm um desempenho superior ao RLM em todas as figuras de mérito empregadas, em especial no caso do RMSE durante o treinamento. Isso é justificável pela capacidade apresentada de evitar que um *outlier* seja realimentado durante o treinamento e possa interferir nas estimativas seguintes.

6 CONCLUSÕES E TRABALHOS FUTUROS

Neste trabalho, avalia-se o desempenho do modelo ESN para tarefa de identificação recursiva de sistemas em cenários contaminados por *outliers*. Duas variantes robustas a *outlier* do algoritmo RLS foram utilizadas para estimar o vetor de peso de saída, ou seja, o algoritmo RLM e o algoritmo RLMCFB proposto, em conjunto com variações da arquitetura da ESN. De acordo com as análises dos experimentos, pode-se afirmar que a ESN demonstra ser uma ferramenta segura para previsão e simulação de sistema dinâmicos. A ESN apresenta resultados bastante promissores e consistentes com o modelo A3-RLMCFB para cenários com e sem a presença de *outlier*. Mais especificamente, o modelo A3-RLMCFB-FHM para sistemas com saída única e o A3-RLMCFB-FH para modelos com mais de uma saída.

6.0.1 Objetivos alcançados

Sobre os objetivos específicos, as seguintes conclusões são apresentadas:

1. A RLS-ESN é aplicada para a identificação de sistema, em que se pode avaliar as possíveis arquiteturas em relação a retroalimentação da sua saída. A avaliação desses testes apresenta a arquitetura A3 como melhor opção para identificação de sistema devido a sua norma reduzida e menor erro de estimação em simulação livre.
2. Visto que a RLS-ESN não alcança resultados satisfatórios para sistemas contaminados por *outliers*, substitui-se o algoritmo de estimação RLS por sua versão robusta RLM.
3. Percebe-se que o *outlier* presente nos regressores da ESN durante o treinamento perturba as estimativas seguintes do modelo, então, propõe-se uma estratégia que evita a realimentação dessa amostra inconsistente, denominada de RLMCFB.
4. A estratégia empregada pelo algoritmo RLMCFB se apresenta promissora na identificação de sistemas. Além disso, a partir da análise dos resultados, não se pode concluir que as arquiteturas da ESN estudadas manifestam influência na robustez a *outlier*. Os casos dos sistema *Dryer* e *Exchanger* corroboram a não influência, pois os resultados na presença de *outliers* são semelhantes em todas as arquiteturas.

6.0.2 Trabalhos futuros

Com o intuito de ampliar os experimentos e o entendimento da ESN, propõem-se os seguintes trabalhos futuros:

- Tendo em vista alguns pontos não abordados nesse trabalho, elenca-se as seguintes questões: a influência do número de amostras disponíveis no subconjunto de treinamento para os modelos; a influência do fator de esquecimento na norma da camada de *readout* da ESN, consequentemente, na estabilidade da rede; a atuação do integrador com vazamento da ESN; o desempenho da ESN com mais de um reservatório; o desempenho dos modelos para hiperparâmetros otimizados especificamente para o sistema modelado; o desempenho dos modelos para sistemas que apresentam *outliers* nos sinais de entrada; a influência de técnicas de otimização do reservatório. Além disso, propor novas figuras de mérito que melhor indiquem a qualidade da simulação; e comparar os resultados com outros algoritmos de estimação robusta não abordados.
- Identificação de séries temporais: verificar a estabilidade e o desempenho dos métodos propostos na presença de *outlier*;
- ESN com aprendizado FORCE (SUSSILLO; ABBOTT, 2009): verificar o desempenho e a estabilidade para identificação de sistemas dinâmicos na presença de *outlier* e incorporar robustez ao método. Durante o treinamento, os dados podem apresentar faixa de operação ainda não aprendida pelo modelo, o que provoca um erro grande. Isso causa uma atualização aguda no modelo pelo método FORCE, porém, no método proposto RLMCRB, essa atualização é atenuada ou evitada. De forma oposta, quando se trata de uma iteração que é apresentada um *outlier*, o método FORCE pode danificar a estimação. Por esse motivo, é proveitoso o estudo para aprimorar esses métodos.
- Identificação adaptativa de sistemas não estacionários na presença de *outliers*: verificar o desempenho e estabilidade dos modelos propostos;
- ESN aplicada em problemas de classificação na presença de *outlier*.

REFERÊNCIAS

AGUIRRE, L. A. **Introdução à identificação de sistemas: Técnicas lineares e não-lineares aplicadas a sistemas reais**. Belo Horizonte - MG, Brasil: Editora UFMG, 2015. ISBN 978-85-423-0079-6.

AGUIRRE, L. A.; RODRIGUES, G. G.; JÁCOME, C. R. Identificação de sistemas não lineares utilizando modelos narmax polinomiais–uma revisão e novos resultados. **SBA Controle e automação**, v. 9, n. 2, p. 90–106, 1998. Disponível em: http://www.sba.org.br/revista/vol9/V9p90.pdf. Acesso em: 01/07/2019.

BESSA, R.; BARRETO, G. A. Robust echo state network for recursive system identification. In: **Advances in Computational Intelligence**. Springer International Publishing, 2019. p. 247–258. Disponível em: https://doi.org/10.1007/978-3-030-20521-8_21. Acesso em: 01/07/2019.

BILLINGS, S. A. **Nonlinear system identification : NARMAX methods in the time, frequency, and spatio-temporal domains**. Chichester, West Sussex, United Kingdom: John Wiley & Sons, Inc, 2013. ISBN 978-1-119-94359-4.

CASDAGLI, M. Nonlinear prediction of chaotic time series. **Physica D: Nonlinear Phenomena**, Elsevier BV, v. 35, n. 3, p. 335–356, maio 1989. Disponível em: https://doi.org/10.1016/0167-2789(89)90074-2. Acesso em: 01/07/2019.

CHAN, S.-C.; ZOU, Y.-X. A recursive least m-estimate algorithm for robust adaptive filtering in impulsive noise: Fast algorithm and convergence performance analysis. **IEEE Transactions on Signal Processing**, Institute of Electrical and Electronics Engineers (IEEE), v. 52, n. 4, p. 975–991, abr. 2004. Disponível em: https://doi.org/10.1109/tsp.2004.823496. Acesso em: 01/07/2019.

CHATZIS, S. P.; DEMIRIS, Y. Echo state gaussian process. **IEEE Transactions on Neural Networks**, Institute of Electrical and Electronics Engineers (IEEE), v. 22, n. 9, p. 1435–1445, set. 2011. Disponível em: https://doi.org/10.1109/tnn.2011.2162109. Acesso em: 01/07/2019.

CHEN, S.; BILLINGS, S. A. Representations of non-linear systems: the NARMAX model. **International Journal of Control**, Informa UK Limited, v. 49, n. 3, p. 1013–1032, mar. 1989. Disponível em: https://doi.org/10.1080/00207178908559683. Acesso em: 01/07/2019.

ELMAN, J. L. Finding structure in time. **Cognitive Science**, Wiley, v. 14, n. 2, p. 179–211, mar. 1990. Disponível em: https://doi.org/10.1207/s15516709cog1402_1. Acesso em: 01/07/2019.

ESPINOSA, J. J.; VANDEWALLE, J. Predictive control using fuzzy models applied to a steam generating unit. In: **PROCEEDINGS OF THE THIRD INTERNATIONAL FLINS WORKSHOP**. Citeseer, 1998. Disponível em: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.45.656. Acesso em: 01/07/2019.

GESTEL, T. V.; SUYKENS, J.; BAESTAENS, D.-E.; LAMBRECHTS, A.; LANCKRIET, G.; VANDAELE, B.; MOOR, B. D.; VANDEWALLE, J. Financial time series prediction using least squares support vector machines within the evidence framework. **IEEE Transactions on Neural Networks**, Institute of Electrical and Electronics Engineers (IEEE), v. 12, n. 4, p. 809–821, jul. 2001. Disponível em: https://doi.org/10.1109/72.935093. Acesso em: 01/07/2019.

GUNDUZ, A.; OZTURK, M. C.; SANCHEZ, J. C.; PRINCIPE, J. C. Echo state networks for motor control of human ECoG neuroprosthetics. In: **2007 3rd International IEEE/EMBS Conference on Neural Engineering**. IEEE, 2007. Disponível em: https://doi.org/10.1109/cne.2007.369722. Acesso em: 01/07/2019.

GUO, Y.; WANG, F.; CHEN, B.; XIN, J. Robust echo state networks based on correntropy induced loss function. **Neurocomputing**, Elsevier BV, v. 267, p. 295–303, dez. 2017. Disponível em: https://doi.org/10.1016/j.neucom.2017.05.087. Acesso em: 01/07/2019.

HAN, M.; XU, M. Laplacian echo state network for multivariate time series prediction. **IEEE Transactions on Neural Networks and Learning Systems**, Institute of Electrical and Electronics Engineers (IEEE), v. 29, n. 1, p. 238–244, jan. 2018. Disponível em: https://doi.org/10.1109/tnnls.2016.2574963. Acesso em: 01/07/2019.

HAYKIN, S. **Adaptive filter theory**. Upper Saddle River, NJ: Pearson, 2014. ISBN 978-0-273-76408-3.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. **Neural Computation**, MIT Press - Journals, v. 9, n. 8, p. 1735–1780, nov. 1997. Disponível em: https://doi.org/10.1162/neco.1997.9.8.1735. Acesso em: 01/07/2019.

HOPFIELD, J. J. Neurons with graded response have collective computational properties like those of two-state neurons. **Proceedings of the National Academy of Sciences**, Proceedings of the National Academy of Sciences, v. 81, n. 10, p. 3088–3092, maio 1984. Disponível em: https://doi.org/10.1073/pnas.81.10.3088. Acesso em: 01/07/2019.

JAEGER, H. The "echo state" approach to analysing and training recurrent neural networks-with an erratum note. **Bonn, Germany: German National Research Center for Information Technology GMD Technical Report**, v. 148, n. 34, p. 13, 2001.

JAEGER, H. Adaptive nonlinear system identification with echo state networks. In: BECKER, S.; THRUN, S.; OBERMAYER, K. (Ed.). **Advances in Neural Information Processing Systems 15**. MIT Press, 2003. p. 609–616. Disponível em: http://papers.nips.cc/paper/2318-adaptive-nonlinear-system-identification-with-echo-state-networks.pdf. Acesso em: 01/07/2019.

JAEGER, H. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. **Science**, American Association for the Advancement of Science (AAAS), v. 304, n. 5667, p. 78–80, abr. 2004. Disponível em: https://doi.org/10.1126/science.1091277. Acesso em: 01/07/2019.

JAEGER, H.; LUKOŠEVIČIUS, M.; POPOVICI, D.; SIEWERT, U. Optimization and applications of echo state networks with leaky- integrator neurons. **Neural Networks**, Elsevier BV, v. 20, n. 3, p. 335–352, abr. 2007. Disponível em: https://doi.org/10.1016/j.neunet.2007.04.016. Acesso em: 01/07/2019.

JANG, J.-S. ANFIS: adaptive-network-based fuzzy inference system. **IEEE Transactions on Systems, Man, and Cybernetics**, Institute of Electrical and Electronics Engineers (IEEE), v. 23, n. 3, p. 665–685, 1993. Disponível em: https://doi.org/10.1109/21.256541. Acesso em: 01/07/2019.

JORDAN, M. I. Serial order: A parallel distributed processing approach. In: **Neural-Network Models of Cognition - Biobehavioral Foundations**. Elsevier, 1997. p. 471–495. Disponível em: https://doi.org/10.1016/s0166-4115(97)80111-2. Acesso em: 01/07/2019.

KOCIJAN, J.; GIRARD, A.; BANKO, B.; MURRAY-SMITH, R. Dynamic systems identification with gaussian processes. **Mathematical and Computer Modelling of Dynamical Systems**, Informa UK Limited, v. 11, n. 4, p. 411–424, dez. 2005. Disponível em: https://doi.org/10.1080/13873950500068567. Acesso em: 01/07/2019.

LI, D.; HAN, M.; WANG, J. Chaotic time series prediction based on a novel robust echo state network. **IEEE Transactions on Neural Networks and Learning Systems**, Institute of Electrical and Electronics Engineers (IEEE), v. 23, n. 5, p. 787–799, maio 2012. Disponível em: https://doi.org/10.1109/tnnls.2012.2188414. Acesso em: 01/07/2019.

LJUNG, L. **Theory and practice of recursive identification**. Cambridge, Mass: MIT Press, 1983. ISBN 0-262-12095-X.

LUKOŠEVIČIUS, M. A practical guide to applying echo state networks. In: **Lecture Notes in Computer Science**. Springer Berlin Heidelberg, 2012. p. 659–686. Disponível em: https://doi.org/10.1007/978-3-642-35289-8_36. Acesso em: 01/07/2019.

LUKOŠEVIČIUS, M.; JAEGER, H. Reservoir computing approaches to recurrent neural network training. **Computer Science Review**, Elsevier BV, v. 3, n. 3, p. 127–149, ago. 2009. Disponível em: https://doi.org/10.1016/j.cosrev.2009.03.005. Acesso em: 01/07/2019.

LUKOŠEVIČIUS, M.; JAEGER, H.; SCHRAUWEN, B. Reservoir computing trends. **KI - Künstliche Intelligenz**, Springer Nature, v. 26, n. 4, p. 365–371, maio 2012. Disponível em: https://doi.org/10.1007/s13218-012-0204-5. Acesso em: 01/07/2019.

MAASS, W.; NATSCHLäGER, T.; MARKRAM, H. Real-time computing without stable states: A new framework for neural computation based on perturbations. **Neural Computation**, MIT Press - Journals, v. 14, n. 11, p. 2531–2560, nov. 2002. Disponível em: https://doi.org/10.1162/089976602760407955. Acesso em: 01/07/2019.

MATTOS, C. L. C.; DAI, Z.; DAMIANOU, A.; BARRETO, G. A.; LAWRENCE, N. D. Deep recurrent gaussian processes for outlier-robust system identification. **Journal of Process Control**, Elsevier BV, v. 60, p. 82–94, dez. 2017. Disponível em: https://doi.org/10.1016/j.jprocont.2017.06.010. Acesso em: 01/07/2019.

MOOR, B. D.; GERSEM, P. D.; SCHUTTER, B. D.; FAVOREEL, W. Daisy: A database for identification of systems. **JOURNAL A**, KONINKLIJKE VLAAMSE INGENIEURSVERENIGING, v. 38, p. 4–5, 1997. [Used dataset: Dryer: (96-006); Exchanger (97-002); Robot arm (96-009); Steam (98-003).]. Disponível em: http://www.esat.kuleuven.ac.be/sista/daisy/. Acesso em: 01/07/2019.

NARENDRA, K.; PARTHASARATHY, K. Identification and control of dynamical systems using neural networks. **IEEE Transactions on Neural Networks**, Institute of Electrical and Electronics Engineers (IEEE), v. 1, n. 1, p. 4–27, mar. 1990. Disponível em: https://doi.org/10.1109/72.80202. Acesso em: 01/07/2019.

NELLES, O. **Nonlinear system identification : from classical approaches to neural networks and fuzzy models**. Berlin New York: Springer, 2001. ISBN 978-3-642-08674-8.

OZTURK, M. C.; XU, D.; PRÍNCIPE, J. C. Analysis and design of echo state networks. **Neural Computation**, MIT Press - Journals, v. 19, n. 1, p. 111–138, jan. 2007. Disponível em: https://doi.org/10.1162/neco.2007.19.1.111. Acesso em: 01/07/2019.

PELLEGRINETTI, G.; BENTSMAN, J. Nonlinear control oriented boiler modeling-a benchmark problem for controller design. **IEEE Transactions on Control Systems Technology**, Institute of Electrical and Electronics Engineers (IEEE), v. 4, n. 1, p. 57–64, 1996. Disponível em: https://doi.org/10.1109/87.481767. Acesso em: 01/07/2019.

ROUSSEEUW, P. J.; LEROY, A. M. **Robust Regression and Outlier Detection**. John Wiley & Sons, Inc., 1987. Disponível em: https://doi.org/10.1002/0471725382. Acesso em: 01/07/2019.

SHI, Z.; HAN, M. Support vector echo-state machine for chaotic time-series prediction. **IEEE Transactions on Neural Networks**, Institute of Electrical and Electronics Engineers (IEEE), v. 18, n. 2, p. 359–372, mar. 2007. Disponível em: https://doi.org/10.1109/tnn.2006.885113. Acesso em: 01/07/2019.

STEIL, J. Backpropagation-decorrelation: online recurrent learning with o(n) complexity. In: **2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No.04CH37541)**. IEEE, 2004. v. 2, p. 843–848. Disponível em: https://doi.org/10.1109/ijcnn.2004.1380039. Acesso em: 01/07/2019.

SUSSILLO, D.; ABBOTT, L. Generating coherent patterns of activity from chaotic neural networks. **Neuron**, Elsevier BV, v. 63, n. 4, p. 544–557, ago. 2009. Disponível em: https://doi.org/10.1016/j.neuron.2009.07.018. Acesso em: 01/07/2019.

TAKAGI, T.; SUGENO, M. Fuzzy identification of systems and its applications to modeling and control. In: **Readings in Fuzzy Sets for Intelligent Systems**. Elsevier, 1993. p. 387–403. Disponível em: https://doi.org/10.1016/b978-1-4832-1450-4.50045-6. Acesso em: 01/07/2019.

WIGREN, T. Input-output data sets for development and benchmarking in nonlinear identification. **Technical Reports from the department of Information Technology**, v. 20, p. 2010–020, 2010. Disponível em: http://www.it.uu.se/research/publications/reports/2010-020/2010-020-nc.pdf. Acesso em: 01/07/2019.

ZHANG, C.; GUO, Y.; WANG, F.; CHEN, B. Generalized maximum correntropy-based echo state network for robust nonlinear system identification. In: **2018 International Joint Conference on Neural Networks (IJCNN)**. IEEE, 2018. Disponível em: https://doi.org/10.1109/ijcnn.2018.8489249. Acesso em: 01/07/2019.

ZHOU, H.; HUANG, J.; LU, F.; THIYAGALINGAM, J.; KIRUBARAJAN, T. Echo state kernel recursive least squares algorithm for machine condition prediction. **Mechanical Systems and Signal Processing**, Elsevier BV, v. 111, p. 68–86, out. 2018. Disponível em: https://doi.org/10.1016/j.ymssp.2018.03.047. Acesso em: 01/07/2019.

ZOU, Y.; CHAN, S.; NG, T. A recursive least m-estimate (RLM) adaptive filter for robust filtering in impulse noise. **IEEE Signal Processing Letters**, Institute of Electrical and Electronics Engineers (IEEE), v. 7, n. 11, p. 324–326, nov. 2000. Disponível em: https://doi.org/10.1109/97.873571. Acesso em: 01/07/2019.

ZOU, Y.; CHAN, S.; NG, T. Robust m-estimate adaptive filtering. **IEE Proceedings - Vision, Image, and Signal Processing**, Institution of Engineering and Technology (IET), v. 148, n. 4, p. 289, 2001. Disponível em: https://doi.org/10.1049/ip-vis:20010316. Acesso em: 01/07/2019.