UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE TECNOLOGIA E GEOCIÊNCIAS
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
CURSO DE GRADUAÇÃO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

JOÃO PAULO DE BARROS SANTOS

MODELAGEM DE SISTEMAS DINÂMICOS BASEADA EM REDES NEURAIS ARTIFICIAIS: uma comparação com métodos clássicos ARX e ARMAX.

Recife
2025

JOÃO PAULO DE BARROS SANTOS

MODELAGEM DE SISTEMAS DINÂMICOS BASEADA EM REDES NEURAIS ARTIFICIAIS: uma comparação com métodos clássicos ARX e ARMAX.

Trabalho de Conclusão de Curso apresentado ao Departamento de Engenharia Elétrica da Universidade Federal de Pernambuco, como requisito parcial para obtenção do grau de Engenheiro de Controle e Automação.

Orientador(a): Prof. Dr. Rafael Cavalcanti Neto

Recife
2025

Ficha de identificação da obra elaborada pelo autor,
através do programa de geração automática do SIB/UFPE

Santos, João Paulo de Barros.
Modelagem de sistemas dinâmicos baseada em redes neurais artificiais: uma comparação com métodos clássicos ARX e ARMAX. / João Paulo de Barros Santos. - Recife, 2025.
71 p : il., tab.

Orientador(a): Rafael Cavalcanti Neto
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Pernambuco, Centro de Tecnologia e Geociências, Engenharia de Controle e Automação - Bacharelado, 2025.
Inclui referências, apêndices.

1. Identificação de sistemas. 2. Redes neurais. 3. Modelagem de sistemas dinâmicos. 4. Modelo NARX. 5. Sistemas de controle. 6. Conversor Buck. I. Cavalcanti Neto, Rafael. (Orientação). II. Título.

620 CDD (22.ed.)

JOÃO PAULO DE BARROS SANTOS

MODELAGEM DE SISTEMAS DINÂMICOS BASEADA EM REDES NEURAIS ARTIFICIAIS: uma comparação com métodos clássicos ARX e ARMAX.

Trabalho de Conclusão de Curso apresentado ao Departamento de Engenharia Elétrica da Universidade Federal de Pernambuco, como requisito parcial para obtenção do grau de Engenheiro de Controle e Automação.

Aprovado em: 17/12/2025

BANCA EXAMINADORA

_____

Prof. Dr. Rafael Cavalcanti Neto
Universidade Federal de Pernambuco

_____

Prof. Dr. Douglas Contente Pimentel Barbosa
Universidade Federal de Pernambuco

_____

Eng. M.Sc. Pablo Luiz Tabosa da Silva
Universidade Federal da Paraíba

Dedico este trabalho à minha mãe, Valéria Barros, cuja garra e cuidado sempre impulsionaram os meus passos. Mesmo com pouco acesso aos estudos, ela me ofereceu o maior presente que poderia: a oportunidade de ser transformado pela educação.

AGRADECIMENTOS

Posso dizer que sou extremamente responsável por todos os meus passos nesse curso, mas preciso ser justo afirmando que não poderia chegar aonde cheguei sozinho. Em todos os meus passos, embora difíceis, pude contar com o apoio de muitos, e isso tudo começa de berço. Serei eternamente grato a minha mãe por todo o esforço dedicado desde a escola, que mesmo com poucos recursos me fez acreditar em mim mesmo e me deu o necessário para que eu alcançasse meus objetivos.

Aqui, também agradeço às minhas tias, Mariinha e Paê, por cada centavo e tempo investido em mim para que eu me tornasse quem sou hoje. Agradeço à minha avó, Terezinha, eu te amo demais e você é minha maior inspiração de força e de alegria. Agradeço a minha irmã, Andreza, que sempre fez com que eu me sentisse amado; tudo teria sido muito mais difícil sem você por perto. Agradeço a Tia Neia, Tio Marcos e Maria Rita, amo vocês. E, tão importante quanto, agradeço a Rafael, meu maior parceiro da vida, sempre do meu lado me apoiando em tudo; também teria sido um desafio maior sem você nessa reta final. Agradeço a todos meus familiares que em algum momento, de alguma forma, investiram na minha educação em especial ao meu Tio Valdeque, Muito obrigado.

Um grande agradecimento a todos os professores do departamento que pude aprender junto, em especial ao meu professor orientador Rafael Cavalcanti e ao professor Fabrício Bradachia, que foram os docentes que mais marcaram a minha jornada na universidade.

Não poderia deixar de agradecer aos grandes amigos que pude formar na faculdade: Hewerton, Letícia e Nathalia. Mas, também agradeço aos meus demais amigos da vida: Thaynara, Mykaella, Rogério, Mª Letícia, Karol, Priscila, Valter e Day. Todos vocês tornaram o processo mais leve e me ajudaram em algum momento.

Agradeço também à Galva, na figura de Kater, Filipe e Aline, que sempre me deram oportunidades de crescimento, mas meu maior agradecimento aqui dedico a Glauber por me impulsionar nas minhas atividades e enxergar potencial no meu trabalho e por ter sido aquele que mais me motivou para que este trabalho acontecesse. E, também deixo um agradecimento especial a todos meus amigos de trabalho que tornam todos os meus dias melhores.

Por fim, agradeço a todos os negros da história que lutaram para que hoje eu, e tantos outros jovens negros, pudéssemos acessar uma educação de qualidade. Concluo este curso sendo o primeiro da minha família materna a obter um diploma em uma universidade pública federal. Não escrevo isso por orgulho pessoal, mas com a esperança de estar abrindo caminho para muitos outros. Como Paulo Freire, acredito que a educação transforma pessoas, e pessoas transformam o mundo, e é com esse mesmo pensamento que quero ver meus primos e, um dia, meus filhos, tendo as oportunidades para transformar as realidades que ainda precisam ser transformadas.

“A essência da inteligência é ver o futuro e não saber das coisas.”
— *Michio Kaku.*

RESUMO

Este trabalho tem como objetivo comparar técnicas de identificação de sistemas, especificamente os modelos ARX, ARMAX e uma rede neural NARX, aplicadas a uma planta simulada no *Simulink*, que para este trabalho foi usado o conversor Buck. A identificação foi realizada utilizando dados provenientes de variações reais no sinal de entrada, e o desempenho dos modelos foi avaliado por meio das métricas MSE e R^2. Testes iniciais com ruído no sinal mostraram que a planta atenua naturalmente essas perturbações, o que levou à utilização dos dados limpos para garantir uma comparação equilibrada entre os métodos. Os resultados obtidos evidenciam que os modelos lineares ARX e ARMAX representam satisfatoriamente o comportamento dinâmico observado, enquanto a rede NARX apresentou desempenho superior, alcançando menor erro e maior precisão na previsão da saída. Dessa forma, o estudo mostra que, embora métodos lineares sejam eficientes em diversas situações, abordagens não lineares baseadas em redes neurais podem oferecer maior fidelidade na identificação de sistemas, especialmente quando a dinâmica envolve características que os modelos lineares não capturam completamente.

Palavras-chave: Identificação de sistemas; ARX; ARMAX; NARX; Redes neurais; Conversor Buck; Modelagem dinâmica.

ABSTRACT

This work aims to compare different system identification techniques, specifically the ARX, ARMAX, and NARX models, applied to a plant simulated in Simulink, for which a Buck converter was used as the study case. The identification was carried out using data obtained from realistic variations in the input signal, and the performance of the models was evaluated through the MSE and R^2 metrics. Initial tests with noise showed that the plant naturally attenuates such disturbances, which led to the use of clean data to ensure a fair comparison between the methods. The results demonstrate that the linear ARX and ARMAX models satisfactorily represent the observed dynamic behavior, while the NARX neural network achieves superior performance, presenting lower error and greater accuracy in predicting the system output. Therefore, the study shows that although linear models are effective in many situations, nonlinear approaches based on neural networks can offer higher fidelity in system identification, especially when the dynamics involve characteristics that linear models cannot fully capture.

Keywords: System identification; ARX; ARMAX; NARX; Neural networks; Buck converter; Dynamic modeling.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ARX	*AutoRegressive with eXogenous Input*
ARMAX	*AutoRegressive Moving Average with eXogenous Input*
NARX	*Nonlinear AutoRegressive with eXogenous Input*
RNA	Rede Neural Artificial
MLP	*Multi-Layer Perceptron*
MSE	*Mean Squared Error (Erro Quadrático Médio)*
R^2	Coeficiente de Determinação
PWM	*Pulse Width Modulation* (Modulação por Largura de Pulso)
CC–CC	Conversor de Corrente Contínua para Corrente Contínua
LC	Indutor e Capacitor
Ts	Tempo de amostragem
FPE	*Final Prediction Error*
WoS	*Web of Science*
MATLAB	*Matrix Laboratory*
UFPE	Universidade Federal de Pernambuco
MOSFET	*Metal-Oxide-Semiconductor Field-Effect Transistor*
DC	*Direct Current* (Corrente Contínua)
LTI	*Linear Time-Invariant* (Sistema Linear Invariante no Tempo)
IDDATA	Formato de dados do MATLAB para identificação
IA	Inteligência Artificial
PWM	*Pulse Width Modulation* (Modulação por Largura de Pulso)

LISTA DE SÍMBOLOS

V_{in}	Tensão de entrada do conversor *Buck*
V_o	Tensão de saída do conversor *Buck*
L	Indutância do indutor
C	Capacitância do capacitor de filtro
R	Resistência da carga
f_s	Frequência de chaveamento
D	*Duty cycle* aplicado ao PWM
I_L	Corrente no indutor
t	Tempo
s	Variável da transformada de Laplace
$y(t)$	Saída do sistema (tensão simulada)
$u(t)$	Entrada do sistema *(duty cycle)*
a_i	Coeficientes do polinômio A(z) nos modelos ARX/ARMAX
b_i	Coeficientes do polinômio B(z) nos modelos ARX/ARMAX
c_i	Coeficientes do polinômio C(z) no modelo ARMAX
n_a	Ordem do polinômio A(z)
n_b	Ordem do polinômio B(z)
n_k	Atraso entre entrada e saída
$e(t)$	Erro ou ruído da modelagem
$\hat{y}(t)$	Saída prevista pelo modelo
MSE	*Mean Squared Error*
R^2	Coeficiente de determinação

SUMÁRIO

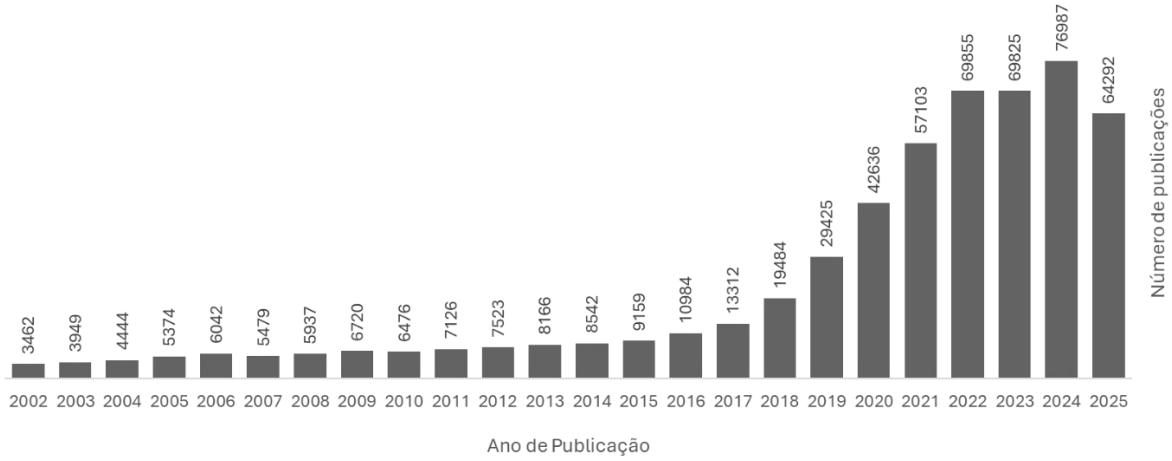
1 INTRODUÇÃO

Nas últimas décadas, o avanço da tecnologia e a crescente complexidade dos sistemas dinâmicos têm impulsionado a busca por métodos de modelagem e controle mais precisos e robustos. Quando o modelo dinâmico de um sistema controlado é conhecido exatamente, torna-se possível projetar um controlador ideal capaz de reproduzir a trajetória de referência desejada. Entretanto, na prática, a presença de ruídos, incertezas paramétricas e não linearidades torna difícil obter um modelo exato, o que compromete a precisão e a estabilidade do controle (JAMI'IN et al., 2016; NAYANASIRI; LI, 2022).

Para lidar com essas limitações, métodos clássicos de modelagem, como os modelos ARX (Auto-Regressive with eXogenous Input) e ARMAX (Auto-Regressive Moving Average with eXogenous Input), têm sido amplamente utilizados por sua simplicidade e eficiência na representação de sistemas lineares. No entanto, esses modelos apresentam desempenho limitado quando aplicados a sistemas não lineares, cuja resposta depende de variações complexas das variáveis de estado. Nesse contexto, as redes neurais artificiais (RNA) surgem como uma alternativa promissora, por sua capacidade de aproximar funções não lineares e representar comportamentos dinâmicos complexos a partir de dados experimentais (JAMI'IN et al., 2016; ZHANG et al., 2018).

A relevância científica das redes neurais pode ser observada por meio de uma pesquisa bibliométrica realizada na base de dados *Web of Science*, utilizando o tópico *"neural network*"*. Para isso, a Figura 1 apresenta, por meio de um gráfico de barras, a evolução do número de publicações ao longo dos anos, evidenciando um crescimento considerável desde 2002 e um acentuamento nesse aumento a partir de 2015. Observa-se também que o volume anual de publicações passou de menos de 4 mil trabalhos em 2001 para quase 77 mil em 2024, indicando a consolidação e a crescente importância das redes neurais na comunidade científica.

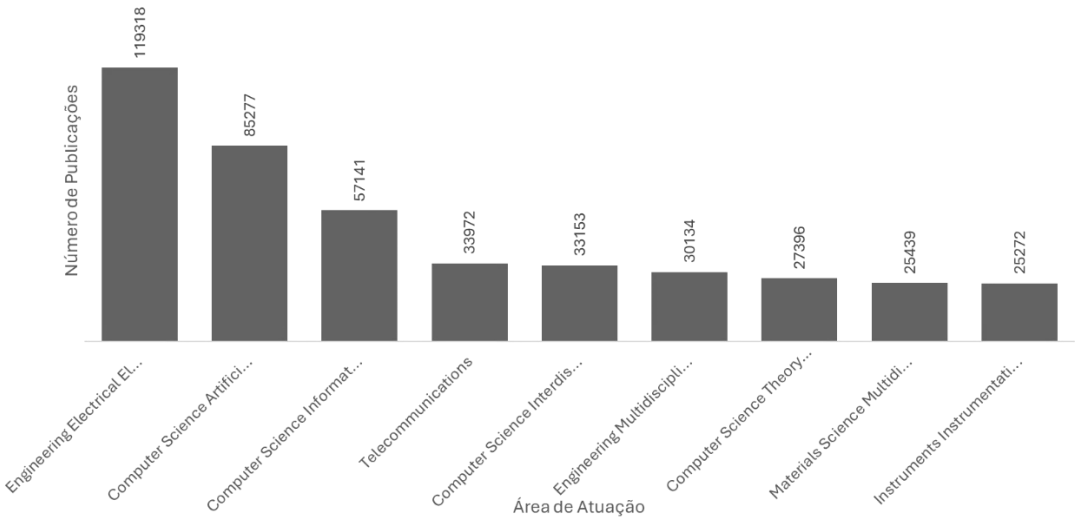
Figura 1 – Número de publicações no *Web of Science* para o tópico *"neural network*".*



Fonte: AUTOR.

A Figura 2 apresenta a distribuição das publicações por área de conhecimento, com destaque para *“Engineering, Electrical & Electronic”,* que lidera o número de trabalhos publicados. Essa predominância demonstra a forte presença das redes neurais em aplicações voltadas à engenharia elétrica, eletrônica, automação, sistemas de controle e processamento de sinais, áreas nas quais a modelagem de sistemas dinâmicos e o controle adaptativo desempenham papel central no avanço tecnológico.

Figura 2 – Quantidade de publicações por área de atuação.



Fonte: AUTOR.

Tabela 1 – Número de publicações totais no *Web of Science* sobre redes neurais.

Área	Número de publicações	Percentual sobre o total.
Engineering, Electrical & Electronic	119.318	21,857%
Computer Science, Artificial Intelligence	85.277	15,621%
Computer Science, Information Systems	57.141	10,467%
Telecommunications	33.972	6,223%
Computer Science, Interdisciplinary Applications	33.153	6,073%
Outros	67.789	56,489%

Fonte: AUTOR.

Dentro desse contexto, este trabalho propõe comparar o desempenho dos modelos ARX, ARMAX e NARX (rede neural ARX) aplicados à modelagem e identificação de sistemas dinâmicos. O estudo é realizado utilizando como planta de referência o conversor *Buck*, um conversor CC–CC amplamente empregado em sistemas de eletrônica de potência e frequentemente utilizado como modelo experimental em pesquisas de controle (NAYANASIRI; LI, 2022). Por sua estrutura simples e comportamento dinâmico não linear, o conversor *Buck* constitui uma excelente base para a análise comparativa de técnicas de identificação, permitindo avaliar a capacidade dos diferentes modelos em representar com fidelidade o comportamento dinâmico de sistemas reais.

Dessa forma, o objetivo deste trabalho é avaliar a eficácia e a precisão dos modelos ARX, ARMAX e NARX na representação do comportamento do conversor Buck, comparando os erros de estimação e a resposta dinâmica simulada. Espera-se, com isso, demonstrar a contribuição das redes neurais na melhoria da capacidade de modelagem de sistemas não lineares, fornecendo subsídios para a aplicação dessas técnicas em projetos de controle mais robustos e eficientes.

1.1 Objetivos

1.1.1 Geral

Avaliar o desempenho de modelos clássicos de identificação (ARX e ARMAX) e de redes neurais artificiais do tipo NARX na modelagem do conversor *Buck*, comparando sua capacidade de representar o comportamento dinâmico do sistema.

1.1.2 Específicos

Para alcançar o objetivo geral proposto, foram definidos os seguintes objetivos específicos:

- Implementar, no ambiente *MATLAB/Simulink*, a simulação de um conversor Buck para coleta de dados de entrada e saída;
- Aplicar métodos de identificação clássicos ARX e ARMAX a partir dos dados simulados;
- Desenvolver e treinar uma rede neural NARX com base nos mesmos dados, analisando seu desempenho de aprendizado;
- Comparar quantitativamente os resultados obtidos entre os três modelos, utilizando métricas de desempenho como o erro médio quadrático (MSE);
- Avaliar graficamente a resposta dinâmica simulada de cada modelo em relação à resposta real da planta;
- Discutir as vantagens e limitações de cada método de modelagem, considerando aspectos de precisão, robustez e complexidade computacional.

1.2 Organização do Trabalho

Este trabalho está estruturado em quatro capítulos principais, que se complementam de forma lógica e progressiva para alcançar os objetivos propostos.

O Capítulo 1 apresenta a introdução ao tema, destacando a relevância da modelagem de sistemas dinâmicos e a motivação para o uso de técnicas baseadas em redes neurais artificiais. Também são definidos os objetivos geral e específicos que orientam o desenvolvimento do estudo.

O Capítulo 2 aborda a fundamentação teórica, na qual são revisados os principais conceitos relacionados à identificação de sistemas, com ênfase nos modelos clássicos ARX e ARMAX e na estrutura das redes neurais artificiais. Nesse contexto, é apresentada a arquitetura NARX, destacando sua aplicação na modelagem de sistemas não lineares.

O Capítulo 3 descreve a metodologia e o desenvolvimento do trabalho, detalhando o processo de simulação do conversor Buck, a geração dos sinais de entrada e saída, e a implementação dos modelos ARX, ARMAX e NARX no ambiente *MATLAB/Simulink*. São apresentadas ainda as etapas de treinamento, validação e comparação dos resultados obtidos por cada modelo.

Por fim, o Capítulo 4 reúne as conclusões e propostas de continuidade, sintetizando os principais resultados alcançados e apontando possíveis direções para trabalhos futuros que busquem aprimorar ou expandir as técnicas de modelagem analisadas.

2 FUNDAMENTAÇÃO TEÓRICA

A modelagem de sistemas dinâmicos é um dos pilares fundamentais da engenharia de controle e automação, pois permite representar matematicamente o comportamento de processos físicos a partir de suas variáveis de entrada e saída. A partir dessa representação, torna-se possível projetar controladores, prever respostas e realizar simulações sem a necessidade de ensaios diretos sobre o sistema real.

Entre as diversas técnicas disponíveis para modelagem, destacam-se os métodos baseados em identificação de sistemas, que têm por objetivo estimar os parâmetros de um modelo formulado a partir de dados de entrada e saída. A área de *system identification* é ampla e está centrada justamente nesse processo de estimação de parâmetros de um modelo matemático previamente definido (LJUNG, 1999).

Esses métodos são amplamente utilizados em situações em que não se dispõe de um modelo analítico exato do sistema físico, ou quando o comportamento real é complexo. Nesse contexto, a identificação de sistemas é uma ferramenta essencial tanto em estudos de modelagem de sistemas do tipo caixa preta, nos quais a estrutura interna do sistema é desconhecida, quanto em abordagens de validação de modelos, nas quais se busca verificar a coerência entre o comportamento teórico e o observado experimentalmente.

Nos últimos anos, o avanço das técnicas de inteligência artificial impulsionou o desenvolvimento de métodos de modelagem capazes de lidar com sistemas de natureza não linear e de comportamento complexo. Dentre essas técnicas, destacam-se as redes neurais artificiais (RNAs), que se baseiam em estruturas computacionais inspiradas no funcionamento do cérebro humano e possuem a capacidade de aprender padrões e relações não lineares diretamente a partir dos dados experimentais.

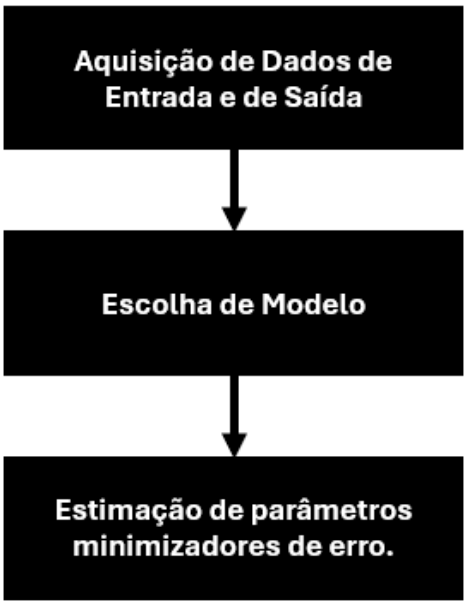
O presente capítulo apresenta os fundamentos teóricos que sustentam o estudo, abordando os conceitos de identificação de sistemas, os modelos ARX e ARMAX e os princípios das redes neurais artificiais aplicadas à modelagem de sistemas dinâmicos. Por fim, são discutidas as principais métricas de desempenho utilizadas para comparar a precisão dos modelos identificados.

2.1 Identificação de sistemas dinâmicos

A identificação de sistemas é uma área fundamental da área de controle, voltada para a obtenção de modelos matemáticos capazes de representar o comportamento dinâmico de um sistema a partir de dados observados de entrada e saída. Essa abordagem é amplamente utilizada quando não se dispõe de um modelo físico completo do processo, ou quando o sistema apresenta comportamentos complexos e de difícil descrição analítica (LJUNG, 1999).

De forma geral, o processo de identificação envolve três etapas principais: (i) a aquisição dos sinais de entrada e saída; (ii) a escolha de uma estrutura de modelo apropriada; e (iii) a estimação dos parâmetros que minimizam a diferença entre o comportamento real e o previsto. O resultado é um modelo capaz de reproduzir a dinâmica do sistema, o que permite sua aplicação em simulações, controle e previsão de estados.

Figura 3 – Etapas para o processo de identificação de modelo.



Fonte: AUTOR.

Nos últimos anos, a identificação paramétrica de sistemas, que consiste em determinar as dinâmicas internas de um sistema com base em um modelo ajustado

sobre dados experimentais, tem recebido crescente atenção da comunidade científica. Esse tipo de abordagem busca não apenas descrever a resposta observada, mas também capturar as relações dinâmicas entre as variáveis envolvidas. A incorporação de redes neurais artificiais (RNA) nesse campo tem se tornado cada vez mais comum, impulsionada pela capacidade dessas estruturas de modelar relações não lineares complexas, pela menor necessidade de conhecimento prévio sobre o processo e pelo avanço do poder computacional disponível para treinamento (DONG; STARR; ZHAO, 2023).

A escolha de um modelo matemático adequado depende diretamente das características da planta em estudo, dos objetivos da modelagem e do nível de conhecimento disponível sobre o sistema. Na prática, essa decisão também envolve experiência e intuição do engenheiro, uma vez que diferentes estruturas são frequentemente testadas até que se obtenha uma representação satisfatória do comportamento real. Em projetos de controle baseados em modelos, há uma tendência em se utilizar representações lineares, em razão da ampla variedade de técnicas de controle desenvolvidas a partir dessas formulações. No entanto, para sistemas que apresentam não linearidades significativas, modelos mais sofisticados, tornam-se indispensáveis para capturar a dinâmica com maior fidelidade (TAVARES, 2012).

As técnicas de identificação podem ser classificadas conforme o grau de conhecimento prévio sobre o sistema:

- **Modelos caixa-branca,** baseados em leis físicas conhecidas;
- **Modelos caixa-cinza,** que combinam conhecimento físico parcial com parâmetros ajustados experimentalmente;
- e **modelos caixa-preta,** que dependem exclusivamente de dados experimentais para representar o comportamento dinâmico.

Os modelos ARX *(AutoRegressive with eXogenous Input)* e ARMAX *(AutoRegressive Moving Average with eXogenous Input)* são exemplos clássicos de abordagens lineares utilizadas em identificação paramétrica. Já as estruturas baseadas em redes neurais, como a NARX *(Nonlinear AutoRegressive with eXogenous Input),* estendem esse conceito para sistemas não lineares, permitindo uma descrição mais fiel de processos reais com dinâmicas complexas. Assim, a

identificação de sistemas se torna uma ferramenta essencial não apenas para fins de modelagem, mas também como base para o desenvolvimento de controladores mais robustos e adaptativos.

2.2 Modelos clássicos de identificação (ARX, ARMAX)

Os modelos ARX e *ARMAX* estão entre as estruturas mais utilizadas na identificação de sistemas lineares invariantes no tempo, devido à sua simplicidade e eficiência na modelagem de sistemas dinâmicos. Esses modelos são amplamente empregados para aproximar o comportamento dinâmico de sistemas lineares, sendo aplicados em tarefas de controle e detecção de falhas, por sua capacidade de representar de forma direta as relações entre entrada e saída (AGGOUNE; CHETOUANI; RADJEAI, 2014).

Existem diversas formas de representar um sistema dinâmico, como funções de transferência, modelos em espaço de estados e representações polinomiais (TAVARES, 2012). No campo da identificação de sistemas, as representações polinomiais são as mais recorrentes, pois descrevem o comportamento dinâmico do sistema com base em dados experimentais e operadores de atraso. Dentre elas, destacam-se os modelos AR *(AutoRegressive)*, ARX *(AutoRegressive with eXogenous input)*, ARMAX *(AutoRegressive Moving Average with eXogenous input),* NARX *(Nonlinear AutoRegressive with eXogenous input)* e NARMAX *(Nonlinear AutoRegressive Moving Average with eXogenous input)*, cuja escolha depende do tipo de planta, do objetivo da modelagem e do nível de conhecimento prévio sobre o sistema (TAVARES, 2012).

Em aplicações de controle baseadas em modelos, observa-se uma preferência por representações lineares, em virtude da grande quantidade de métodos de análise e projeto já consolidados para esse tipo de estrutura (TAVARES, 2012).

2.2.1 Modelo ARX

O modelo ARX descreve a saída de um sistema como uma combinação linear de saídas e entradas passadas, acrescida de um termo de erro. Sua formulação geral é dada por:

$$\mathrm{A}(q^{-1})y(k) = B(q^{-1})u(k) + e(k) \tag{2.1}$$

onde $y(k)$ é a saída, $u(k)$ é a entrada, $e(k)$ é o termo de erro, e q^{-1} representa o operador de atraso, tal que $q^{-1}y(k) = y(k-1)$. Os polinômios $A(q^{-1})$ e $B(q^{-1})$ são definidos como:

$$\mathrm{A}(q^{-1}) = 1 - a_1 q^{-1} - a_2 q^{-2} - \cdots - a_{n_a} q^{-n_a} \tag{2.2}$$

$$B(q^{-1}) = b_1 q^{-1} + a_2 q^{-2} - \cdots - b_{n_b} q^{-n_b} \tag{2.3}$$

em que a_i e b_i são coeficientes associados aos regressores de saída e entrada respectivamente, e n_a e n_b são as ordens dos polinômios.

A expressão também pode ser reescrita como:

$$\mathrm{y(k)} = \frac{B(q^{-1})}{A(q^{-1})}u(k) + \frac{1}{A(q^{-1})}e(k) \tag{2.4}$$

O modelo ARX é classificado como um modelo de erro na equação, pois o termo de erro é tratado como ruído branco filtrado por um processo autorregressivo, o que faz com que o ruído na saída apresente correlação temporal (TAVARES, 2012; AGUIRRE, 2004). Essa estrutura é adequada para sistemas em que o ruído não domina o comportamento dinâmico e pode ser considerado uma pequena perturbação.

2.2.2 Modelo ARMAX

O modelo ARMAX é uma extensão do ARX, incluindo um termo adicional que representa a média móvel *(Moving Average)*, permitindo modelar de forma mais precisa a influência do ruído. Essa característica torna o modelo mais adequado para sistemas nos quais o ruído apresenta correlação temporal significativa. Sua forma geral é:

$$\mathrm{A}(q^{-1})y(k) = B(q^{-1})u(k) + \mathrm{C}(\mathrm{q}^{-1})e(k) \tag{2.5}$$

onde, $C(q^{-1})$ é o polinômio de média móvel, expresso como:

$$\mathrm{C}(q^{-1}) = 1 + c_1 q^{-1} + c_2 q^{-2} + \cdots + c_{n_c} q^{-n_c} \tag{2.6}$$

os coeficientes c_i representam a influência dos erros passados, e n_c é a ordem do polinômio $C(q^{-1})$. O modelo pode ser reescrito como

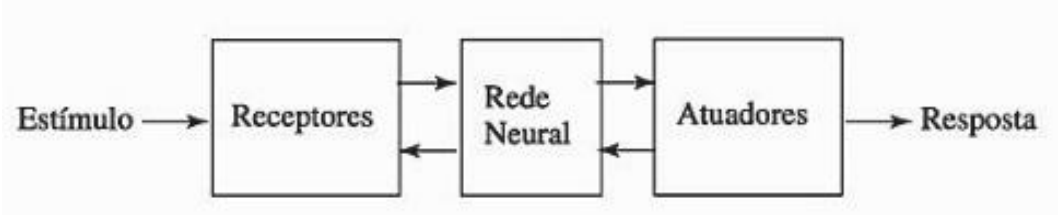
$$y(k) = \frac{B(q^{-1})}{A(q^{-1})}u(k) + \frac{C(q^{-1})}{A(q^{-1})}e(k) \tag{2.7}$$

Ao contrário do modelo ARX, o ARMAX não é linear nos parâmetros, o que torna sua estimação mais complexa. Entretando, essa estrutura é mais robusta e flexível, especialmente em plantas industriais sujeitas a ruídos correlacionados (TAVARES, 2012; AGUIRRE,2004). Esse tipo de modelo é apropriado quando se busca uma representação mais fiel da dinâmica estocástica do sistema.

2.3 Redes neurais artificiais

As redes neurais artificiais (RNAs) são modelos computacionais inspirados na estrutura e no funcionamento do sistema nervoso humano. O cérebro pode ser compreendido como um sistema composto por três estágios principais: receptores, rede neural e atuadores (HAYKIN, 1999). Os receptores convertem estímulos provenientes do ambiente externo em impulsos elétricos que são transmitidos ao cérebro, onde a rede neural realiza o processamento da informação e a tomada de decisões. Em seguida, os atuadores transformam os impulsos gerados pelo cérebro em respostas observáveis, como movimentos ou reações fisiológicas. Esse ciclo de percepção, processamento e ação serve como base conceitual para o desenvolvimento de modelos neurais artificiais.

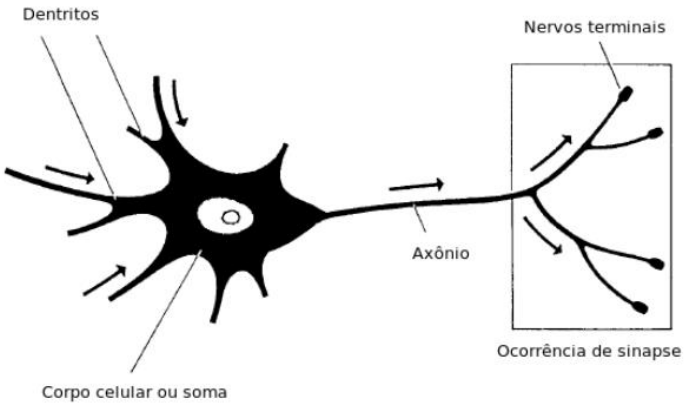
Figura 4 – Representação do sistema nervoso em blocos



Fonte: (GONÇALVES)

Segundo Gonçalves (2023), o neurônio biológico é a unidade fundamental do cérebro humano, especializado na transmissão e processamento de informações por meio de impulsos elétricos. Ele é composto por três partes principais: o corpo celular (ou soma), responsável por integrar os sinais recebidos; os dendritos, que captam estímulos de outros neurônios; e o axônio, uma ramificação mais longa que conduz os sinais elétricos até outras células. Nas extremidades do axônio localizam-se os terminais sinápticos, que realizam a comunicação entre neurônios através de fenômenos conhecidos como sinapses (ARBIB, 2002).

Figura 5 – Modelo simplificado de neurônio biológico.



Fonte: (GONÇALVES)

A partir do entendimento dessa estrutura, pesquisadores buscaram reproduzir computacionalmente o comportamento do sistema nervoso humano, originando o conceito de neurônio artificial. O modelo mais influente nesse sentido foi proposto por McCulloch e Pitts (1943), considerado o marco inicial das redes neurais artificiais. O

modelo, conhecido como Perceptron, implementa de forma simplificada os principais componentes e mecanismos de funcionamento do neurônio biológico.

Nesse modelo, os estímulos provenientes de outros neurônios são representados por sinais de entrada (x_j), e a intensidade com que cada estímulo influencia o neurônio receptor é definida por pesos sinápticos (w_{kj}). Cada peso representa o grau de importância ou "força sináptica" associada a uma conexão, por isso quanto maior o peso, mais significativo é o impacto do sinal sobre o neurônio de saída.

O neurônio artificial realiza inicialmente uma soma ponderada dos sinais de entrada, multiplicando cada valor x_j pelo seu respectivo peso w_{kj}. Essa operação é expressa pela seguinte equação:

$$u_k = \sum_{j=1}^{n} w_{kj} \cdot x_j + bias \tag{2.8}$$

onde n é o número total de sinais de entrada incidentes no neurônio k. O resultado dessa soma, u_k, representa o nível de ativação do neurônio.

Em seguida, aplica-se uma função de ativação $\varphi(u_k)$ que transforma o valor de entrada em uma saída normalizada, conforme:

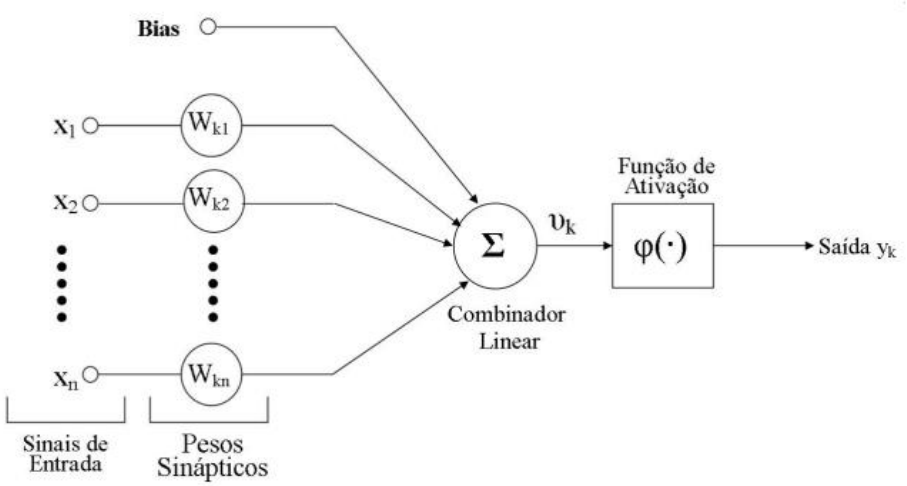
$$y_k = \varphi(u_k) \tag{2.9}$$

A função de ativação é responsável por introduzir não linearidade no modelo, permitindo que o neurônio artificial simule comportamentos mais complexos do que simples combinações lineares (ARBIB, 2002). A literatura apresenta diversas funções de ativação, cada uma com propriedades específicas para diferentes aplicações. Entre as mais comuns estão:

- **Função degrau (*Heaviside*)** — usada nas primeiras redes, com saídas binárias (0 ou 1);
- **Função sigmoide logística** — contínua e suave, amplamente utilizada em problemas de classificação;
- **Função tangente hiperbólica** $(\tanh)$ — semelhante à sigmoide, mas centrada em zero, facilitando o aprendizado em certas redes;

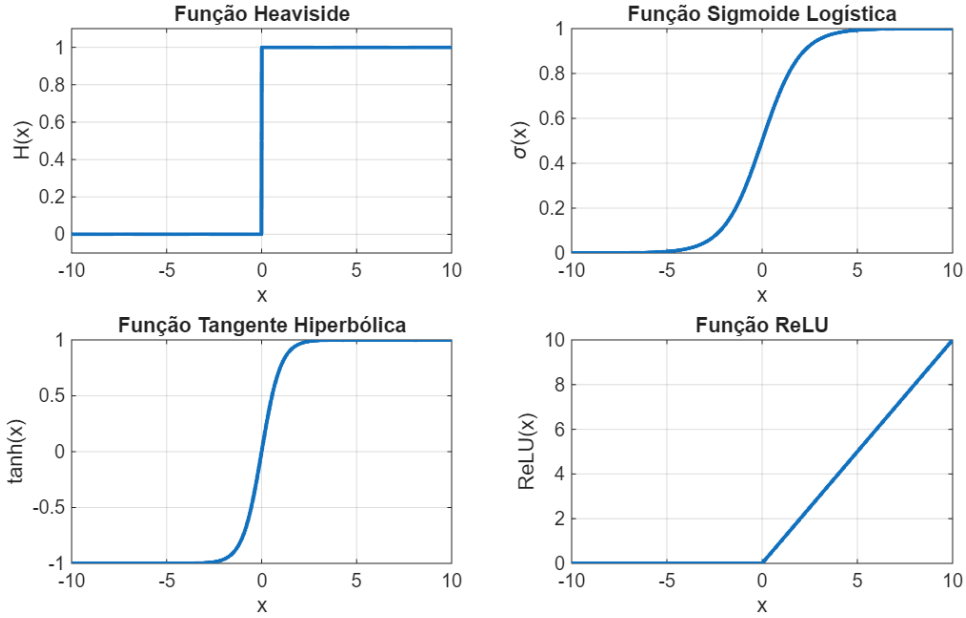
- **Função ReLU (*Rectified Linear Unit*)** — amplamente empregada em redes profundas, define a saída como zero para valores negativos e linear para positivos.

Figura 6 – Modelo de neurônio artificial proposto por McCulloch e Pitts



Fonte: (GONÇALVES)

Figura 7 – Funções de ativação.



Fonte: AUTOR.

Essas funções determinam como o neurônio reage a diferentes níveis de estímulo, controlando a propagação da informação dentro da rede. Ao interligar diversos neurônios artificiais em camadas, forma-se uma rede neural, capaz de aprender padrões e relacionamentos complexos entre variáveis a partir de dados. Esse aprendizado é realizado através de algoritmos de ajuste iterativo dos pesos sinápticos, que minimizam o erro entre a saída desejada e a saída produzida pela rede.

Em síntese, conforme destaca Gonçalves (2023), o modelo matemático das redes neurais artificiais tem inspiração biológica, refletindo a tentativa de traduzir o mecanismo de funcionamento do cérebro humano em uma estrutura computacional capaz de processar informações, reconhecer padrões e aprender com experiências passadas.

2.3.1 Rede Neural NARX (Nonlinear AutoRegressive with eXogenous Input)

As redes neurais do tipo NARX representam uma classe específica de redes neurais recorrentes desenvolvidas para lidar com problemas de previsão em séries temporais e modelagem de sistemas dinâmicos não lineares. Diferentemente das redes alimentadas apenas por entradas externas, a NARX incorpora mecanismos de retroalimentação, permitindo que a rede utilize não apenas os sinais de entrada, mas também valores passados da própria saída para prever estados futuros do sistema. Essa característica confere à rede um comportamento dinâmico, em que as saídas anteriores influenciam diretamente os resultados subsequentes, refletindo de forma mais realista a natureza temporal dos sistemas físicos (AQUIZE et al., 2023).

De forma geral, o modelo NARX pode ser descrito pela seguinte relação funcional:

$$y(k) = f\big(u(k-1), u(k-2) \dots u(k-n), y(k-1), y(k-2), \dots, y(k-m)\big) \qquad (2.10)$$

em que:

- $u(k)$ representa o vetor de entradas exógenas (sinais de entrada do sistema),

- $y(k)$ é o vetor de saídas observadas,
- n e m são, respectivamente, os atrasos associados às entradas e às saídas (ARAUJO, 2024).

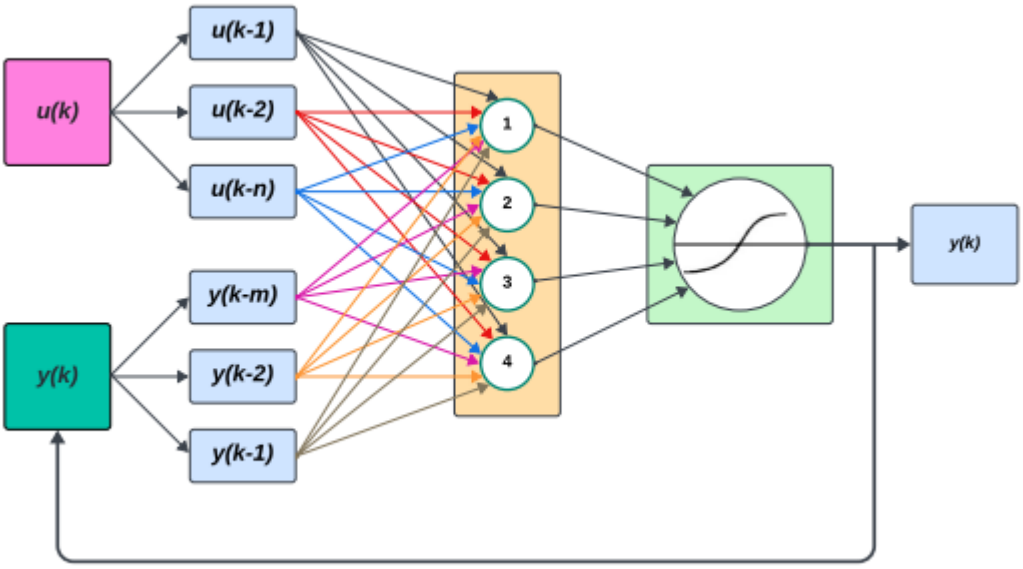
A principal vantagem do modelo NARX está na sua capacidade de aprendizado contínuo. Ao combinar os dados originais de entrada com as saídas geradas durante o processo de treinamento, a rede é capaz de aprimorar iterativamente sua habilidade de predição, ajustando seus pesos sinápticos de modo a minimizar o erro entre o valor previsto e o valor real (KHALED et al., 2020). Essa propriedade torna o modelo particularmente eficiente na representação de sistemas complexos e não lineares.

Internamente, a NARX é estruturada sobre uma Perceptron Multicamadas (MLP – *Multi-Layer Perceptron*), composta por uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída. Os neurônios são interconectados por pesos adaptativos e operam por meio de funções de ativação não lineares, como sigmoide, tangente hiperbólica ou ReLU (funções já apresentadas neste capítulo), o que confere à rede a capacidade de representar relações complexas entre as variáveis. A presença das linhas de atraso *(memory lines)* e da realimentação das saídas faz com que a NARX capture de forma eficiente a dependência temporal das variáveis envolvidas, característica essencial na modelagem de sistemas dinâmicos, (SOUZA et al., 2019).

Além disso, diversos estudos destacam que as redes NARX apresentam convergência mais rápida e maior estabilidade de treinamento em comparação com outras arquiteturas recorrentes tradicionais. Essa eficiência está relacionada à forma como as realimentações são incorporadas, permitindo que a rede aprenda de maneira mais direta as relações causais entre entradas e saídas do sistema (ARAUJO, 2024).

A Figura 8 ilustra a estrutura genérica de uma rede neural NARX, onde $u(k)$ representa a entrada exógena, $y(k)$ é a saída estimada e os blocos de atraso indicam a memória temporal do modelo. Essa representação evidencia o fluxo de informações e a interação entre as variáveis passadas e presentes, fundamentais para a predição do comportamento futuro do sistema.

Figura 8 – Sistema dinâmico da rede neural NARX.



Fonte: (ARAUJO, V. G., 2024).

2.4 Métricas de desempenho para comparação de modelos

A análise quantitativa do desempenho de um modelo é fundamental para avaliar sua capacidade de representar o comportamento real de um sistema dinâmico. As métricas de desempenho permitem quantificar o erro entre os valores medidos e os valores estimados, bem como a capacidade do modelo de responder adequadamente diante de ruídos ou variações nas entradas.

Neste trabalho, são adotadas três métricas principais: Erro Quadrático Médio (MSE), Coeficiente de Determinação (R^2) e Filtragem, conforme metodologia inspirada em Araújo. T. (2022).

2.4.1 Erro Quadrático Médio (MSE)

O Erro Quadrático Médio (MSE — *Mean Squared Error*) é uma métrica estatística amplamente utilizada para quantificar a precisão de um modelo de predição. Ele mede o desvio médio entre os valores reais e os valores estimados, elevando ao quadrado as diferenças para penalizar erros maiores.

Matematicamente, é expresso como:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_1)^2 \tag{2.11}$$

onde:

- n é o número total de observações;
- y_i representa o valor real do sistema;
- $\hat{y}_i$ é o valor estimado pelo modelo.

O valor do MSE é sempre não negativo e varia entre 0 e ∞, sendo que valores próximos de zero indicam uma maior precisão do modelo. Essa métrica será usada como parâmetro comparativo principal entre os modelos ARX, ARMAX e NARX desenvolvidos neste estudo.

2.4.2 Coeficiente de Determinação (R²)

O Coeficiente de Determinação (R^2) mede o quanto o modelo é capaz de explicar a variabilidade dos dados observados em relação à média. Ele fornece uma medida intuitiva de ajuste do modelo aos dados reais, indicando o percentual da variação da saída que é corretamente reproduzida pela predição.

Sua formulação é dada por:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{2.12}$$

onde $\bar{y}$ é a média dos valores reais da saída.

O coeficiente R^2 varia de 0 a 1, sendo 1 o valor ideal, que representa um modelo perfeitamente ajustado. Embora um R^2 alto não garanta sozinho a excelência do modelo, valores baixos indicam que o modelo possui baixo poder explicativo sobre os dados de saída (ARAÚJO, T., 2022).

2.4.3 Filtragem

A filtragem avalia a resposta do modelo diante da presença de ruído no sinal de entrada. Esse ruído, normalmente modelado como ruído branco de média nula e

variância unitária, é introduzido artificialmente para verificar o quanto ele se propaga até a saída estimada.

Modelos com boa capacidade de filtragem mantêm a estabilidade e a precisão das estimativas, apresentando menor variação na saída mesmo quando submetidos a entradas ruidosas.

Segundo Araújo, T. (2022), a filtragem é essencial na análise de sistemas reais, nos quais o ruído de medição pode comprometer o desempenho do estimador, sendo, portanto, um bom indicador da robustez do modelo.

3 METODOLOGIA E DESENVOLVIMENTO

Este capítulo apresenta a metodologia adotada para a modelagem e análise comparativa dos modelos ARX, ARMAX e NARX, descrevendo detalhadamente as etapas de simulação, coleta e tratamento de dados, identificação e validação dos modelos propostos. O conteúdo também permeia por aspectos fundamentais do desenvolvimento do trabalho, destacando o conversor Buck como a plataforma de aplicação prática escolhida para demonstrar o desempenho das técnicas de modelagem abordadas.

O conversor Buck é um dispositivo amplamente utilizado em sistemas de eletrônica de potência devido à sua eficiência na conversão de tensão contínua e sua dinâmica relativamente simples, o que o torna ideal para estudos de identificação de sistemas. A escolha desse conversor como planta de estudo permite analisar com clareza o comportamento dinâmico de um sistema não linear, cuja resposta depende diretamente do ciclo de trabalho *(duty cycle)* aplicado ao chaveamento.

Dessa forma, o capítulo abrange tanto os procedimentos experimentais e computacionais utilizados na obtenção dos dados de entrada e saída quanto as ferramentas matemáticas e computacionais aplicadas na estimação e comparação dos modelos.

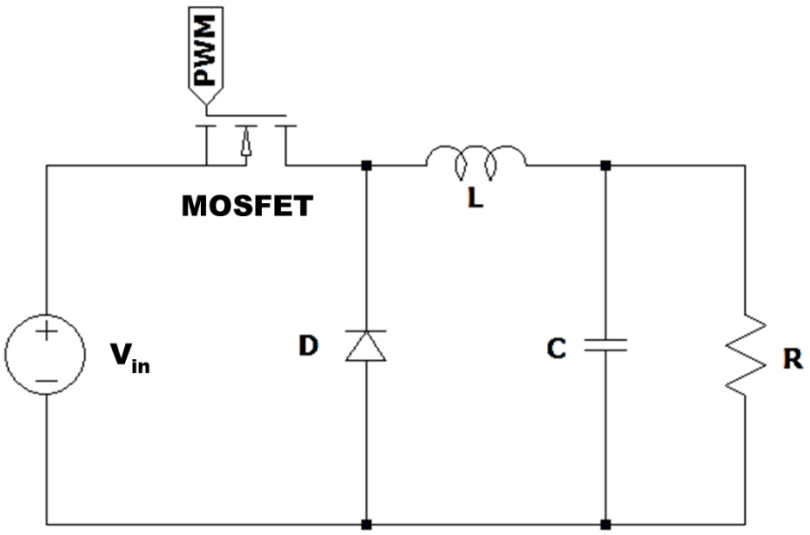
3.1 Definição da planta e simulação do sistema

O conversor *Buck* foi selecionado como planta de estudo neste trabalho por se tratar de um sistema dinâmico não linear de ampla utilização em pesquisas de controle e eletrônica de potência. Sua estrutura relativamente simples, composta por elementos reativos e chaveamento em alta frequência, torna-o uma excelente base para avaliar o desempenho de técnicas de identificação e modelagem de sistemas. De acordo com Nayanasiri e Li (2022), os conversores CC–CC redutores são amplamente empregados em aplicações que exigem alta eficiência e estabilidade, e sua dinâmica típica envolve relações não lineares entre tensão e corrente.

A primeira etapa que envolve esse conversor no trabalho consistiu na modelagem e simulação da planta escolhida. O conversor *Buck* tem como principal

característica reduzir a tensão de entrada para um valor de saída inferior, mantendo a polaridade desta tensão (MOHAN; UNDELAND; ROBBINS, 1995). Sua operação é baseada na comutação controlada de um dispositivo semicondutor (geralmente um MOSFET), em conjunto com um diodo, um indutor e um capacitor de filtro.

Figura 9 – Circuito esquemático de um conversor Buck.



Fonte: AUTOR.

A modelagem, por sua vez, foi desenvolvida no ambiente *Simulink/MATLAB*, por meio do *Simscape Electrica – Specialized Power Systems*. O circuito foi construído a partir de blocos que representam os elementos físicos do conversor, incluindo (MOHAN; UNDELAND; ROBBINS, 1995):

- Fonte de tensão contínua (V_{in}) — Fornece a tensão de entrada para o sistema, fixada em 24 V;
- Chave semicondutora (MOSFET) — responsável pela comutação, controlada por um sinal PWM;
- Diodo (D) — conduz a corrente durante o período em que o MOSFET está desligado;
- Indutor (L) — armazena energia magnética e suaviza a corrente;
- Capacitor de filtro (C) — reduz a ondulação na tensão de saída;

- Carga resistiva (R) — representa a carga conectada ao conversor.

Os valores utilizados para os componentes foram definidos conforme a Tabela 2, de modo a garantir uma resposta dinâmica adequada à frequência de chaveamento escolhida.

Tabela 2 – Parâmetros dos elementos do conversor Buck.

Parâmetro	Símbolo	Valor	Unidade
Tensão de entrada	V_{in}	50	V
Indutância	L	1,2	mH
Capacitância	C	15,3	μF
Resistência de carga	R	4	Ω
Frequência de chaveamento	f_s	12	kHz

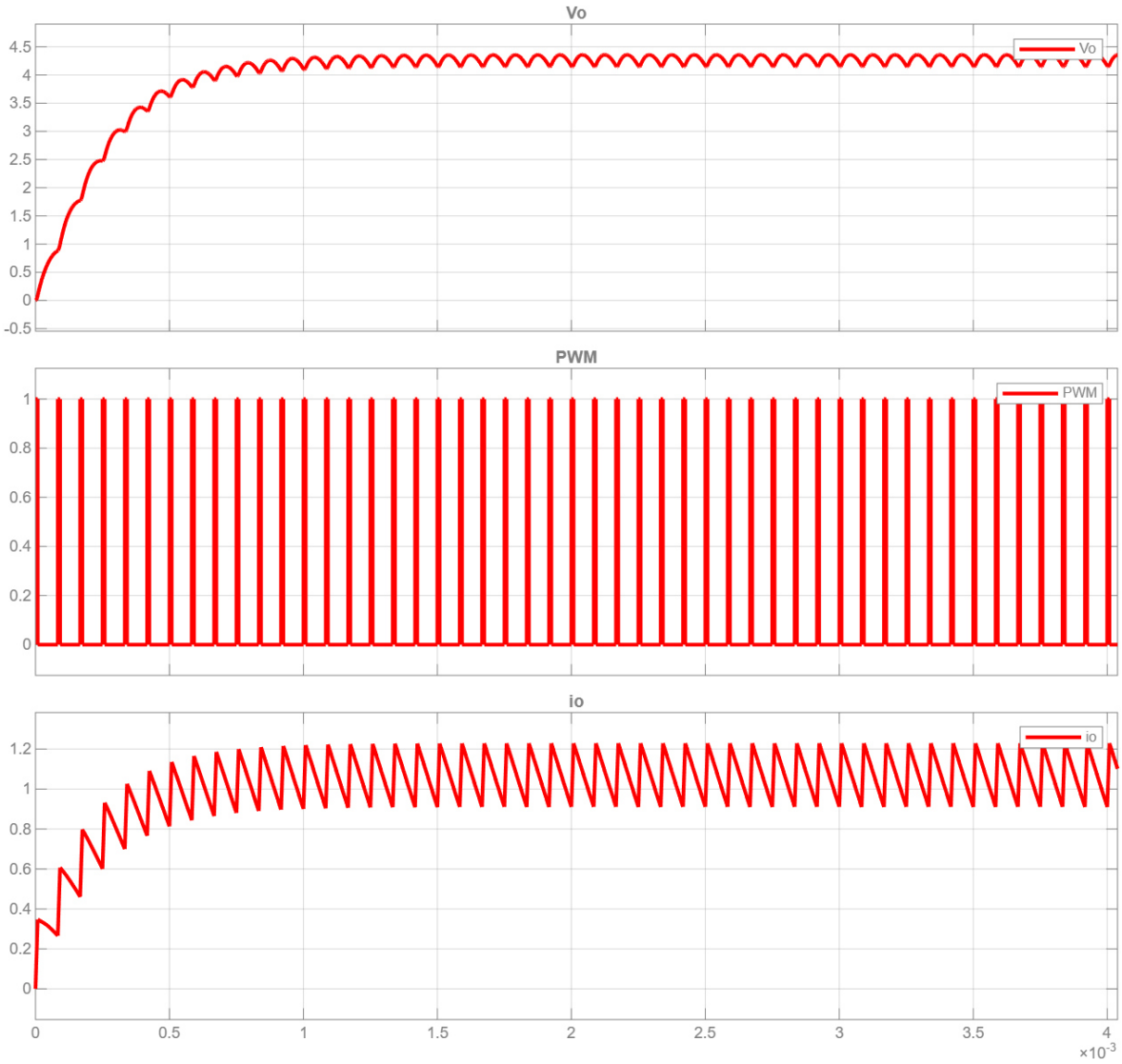
Fonte: AUTOR.

Inicialmente, para avaliar o comportamento do circuito, foram aplicados valores fixos de *duty cycle* de 10%, 50% e 100%, de forma isolada, com o objetivo de observar o comportamento do conversor *Buck* e verificar se a simulação estava respondendo conforme o esperado. Essa etapa inicial foi importante para validar o modelo do conversor, garantindo que a tensão de saída variasse de maneira coerente com o princípio de funcionamento do *Buck*, isto é, apresentando uma tensão proporcional ao ciclo de trabalho aplicado, segundo a relação $V_o = D \cdot V_{in}$, em que D é o *duty cycle* e V_o é a tensão na carga (MOHAN; UNDELAND; ROBBINS, 1995).

A partir dessa verificação, confirmou-se que o modelo respondia adequadamente, permitindo então a utilização de um sinal variável de *duty cycle* para a coleta dos dados necessários à identificação dos modelos. O comportamento do conversor nessas condições pode ser observado nas figuras a seguir (Figura 10, Figura 11 e Figura 12), que mostram as respostas da tensão de saída para os três valores de ciclo de trabalho aplicados.

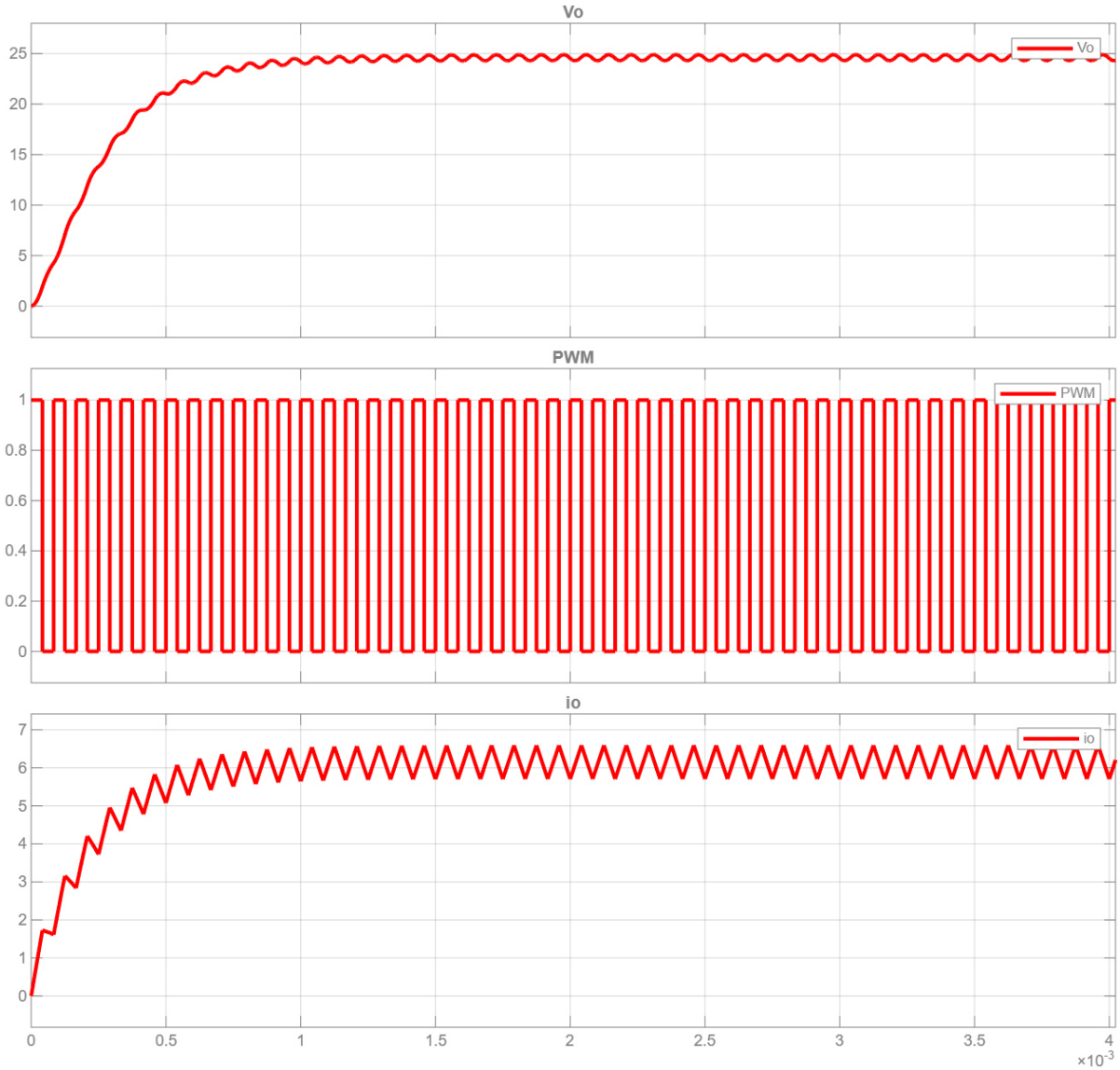
Assim, com o circuito finalizado, é possível simular respostas para vários valores de ciclos de trabalho para o levantamento dos dados que serão úteis para a devida continuidade do desenvolvimento deste trabalho.

Figura 10 – Curvas de tensão, PWM e corrente no indutor para um ciclo de trabalho de 10%.



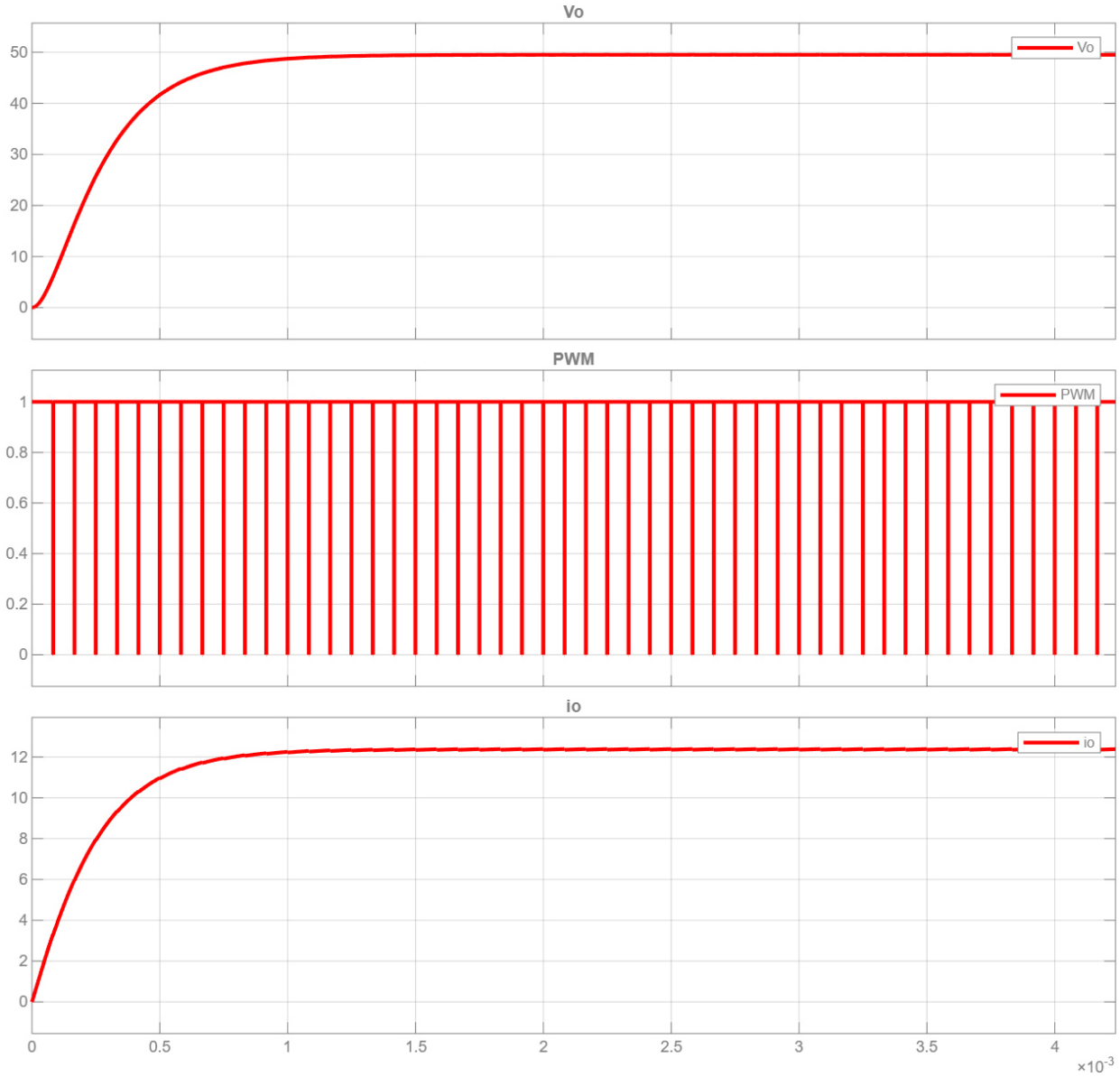
Fonte: AUTOR.

Figura 11 – Curvas de tensão, PWM e corrente no indutor para um ciclo de trabalho de 50%.



Fonte: AUTOR.

Figura 12 – Curvas de tensão, PWM e corrente no indutor para um ciclo de trabalho de 100%.

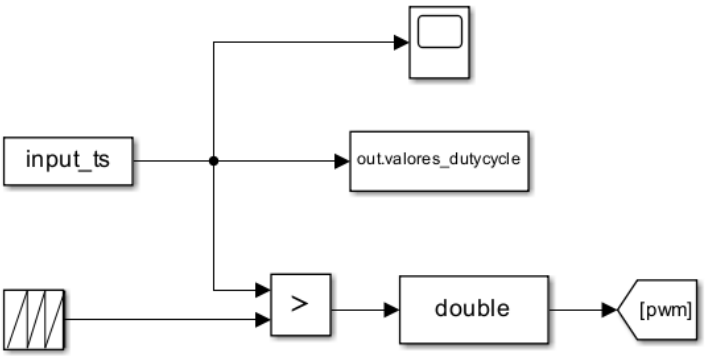


Fonte: AUTOR.

Para o funcionamento do circuito, o controle do chaveamento foi implementado por meio de um gerador PWM, como pode ser visto na Figura 13, formado pela comparação entre uma onda triangular periódica e um sinal de referência variável *(duty cycle).*

A referência de valores de ciclo de trabalho foi fornecida a partir de um vetor criado no *MATLAB* e importado para o *Simulink* pelo bloco *From Workspace*, permitindo a variação ao longo da simulação. O código para gerar esse vetor de variação de *duty cycle* pode ser avaliado no Apêndice A deste trabalho.

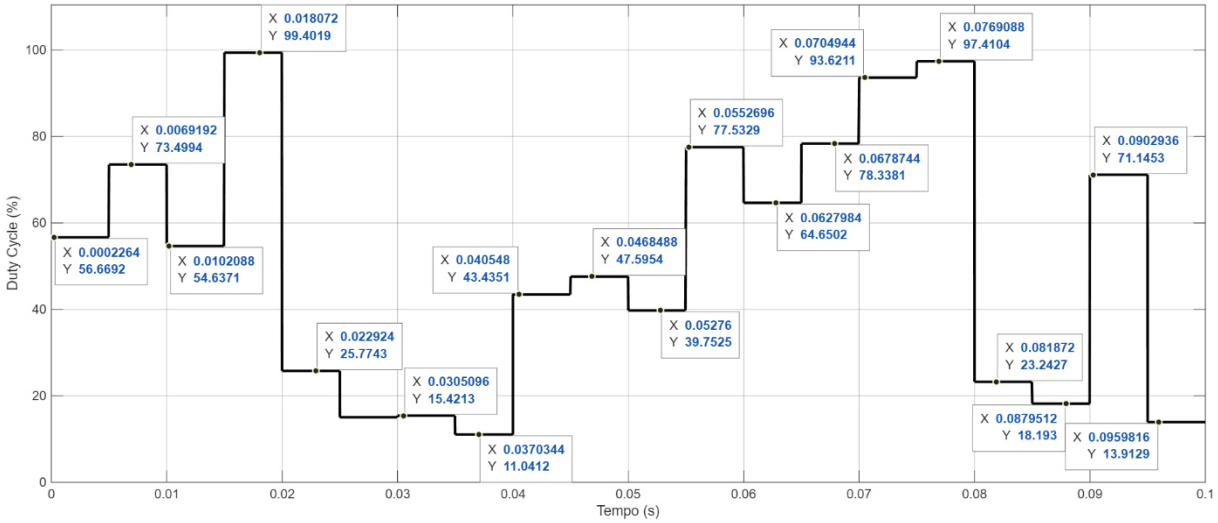
Figura 13 – Diagrama do gerador PWM no MATLAB.



Fonte: AUTOR.

O sinal de *duty cycle* gerado apresenta variações aleatórias ao longo do tempo, formadas por níveis constantes em pequenos intervalos. Esse tipo de sinal foi escolhido porque permite testar a resposta do conversor *Buck* em diferentes condições de operação, o que é importante para uma boa identificação do sistema. Cada degrau permanece constante por cerca de 5 milissegundos e depois muda para um novo valor aleatório, variando entre aproximadamente 5% e 95% de ciclo ativo. O gráfico que mostra essa variação aleatória do *duty cycle* é apresentado na Figura 14, onde é possível observar o comportamento em degraus do sinal utilizado como entrada da planta simulada.

Figura 14 – Varação aleatória do sinal de *duty cycle* utilizado como entrada na simulação do conversor Buck.



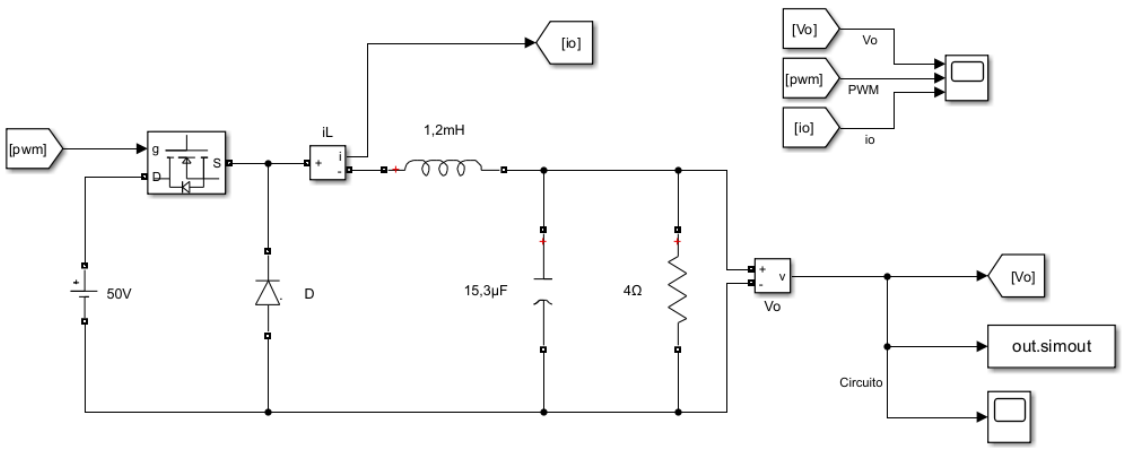
Fonte: AUTOR.

Durante a execução da simulação, as principais variáveis monitoradas foram:

- o *duty cycle* aplicado à chave;
- a tensão de saída do conversor (V_{out}).

A tensão de saída foi capturada por meio de um bloco *Voltage Sensor* e exportada para o *MATLAB* utilizando o bloco *To Workspace* — item *out.simout* na Figura 15. Esse bloco salvou, no *Workspace,* os dados de simulação. Esses dados, por sua vez, serviram como base para as etapas seguintes de identificação dos modelos ARX e ARMAX e posterior treinamento da rede neural NARX.

Figura 15 – Circuito do conversor *Buck* simulado no *MATLAB*.



Fonte: AUTOR.

3.2 Modelagem com ARX

Nesta etapa, foi desenvolvido um modelo de identificação do tipo ARX, com o objetivo de obter uma representação aproximada da dinâmica do conversor Buck. Esse procedimento foi conduzido de forma híbrida: inicialmente, o modelo ARX foi estimado utilizando-se apenas o primeiro degrau presente no sinal de entrada, de modo a capturar o comportamento transitório fundamental do sistema em torno de uma variação real de entrada. Para isso, o algoritmo percorreu automaticamente o vetor de *duty cycle*, identificando o primeiro salto significativo e extraindo uma janela temporal que inclui tanto uma pequena parcela anterior ao degrau quanto o trecho

completo de subida da tensão de saída. Em seguida, esse conjunto reduzido foi estruturado no formato *iddata* e submetido a uma busca sistemática das ordens do modelo ARX *(na, nb, nk),* variando de 1 a 5, adotando-se como critério de seleção o menor valor de FPE *(Final Prediction Error).* O modelo resultante a partir desse primeiro degrau foi então considerado representativo da dinâmica local do conversor e utilizado como base para a próxima etapa.

Nessa etapa, realizou-se uma busca das ordens do modelo ARX, variando os parâmetros na, nb e nk de 1 a 5. No contexto de modelos ARX, na corresponde à ordem do polinômio associado às saídas passadas, isto é, quantas amostras anteriores de $y(t)$ o modelo utiliza para prever o valor atual. Esse parâmetro está relacionado ao número de polos do modelo discreto. Já nb representa a ordem do polinômio aplicado às entradas passadas, definindo quantas amostras anteriores de $u(t)$ influenciam a saída; este parâmetro está relacionado ao número de zeros do modelo. Por fim, nk corresponde ao atraso puro entre entrada e saída, indicando o número de instantes de amostragem necessários para que uma variação em $u(t)$ provoque efeito observável em $y(t)$. Esses três parâmetros definem completamente o modelo ARX e são fundamentais para capturar a dinâmica do sistema. Como critério de escolha do melhor modelo, adotou-se o menor valor de FPE *(Final Prediction Error)*, que fornece uma estimativa estatística da qualidade do ajuste penalizando modelos excessivamente complexos.

Após definida a estrutura ótima do modelo, este ARX identificado a partir do primeiro degrau foi aplicado ao conjunto completo de dados do experimento, de modo a avaliar sua capacidade de generalização. Essa abordagem permite verificar se um modelo linear ajustado localmente em torno de um único transitório é capaz de representar adequadamente a dinâmica global do sistema, mesmo considerando que o conversor Buck apresenta comportamento inerentemente não linear. A saída prevista pelo modelo ARX foi comparada diretamente com a saída real do *Simulink* ao longo de todo o intervalo de simulação.

O código completo adotado para identificação e simulação encontra-se disponibilizado no Apêndice C.

3.3 Modelagem com ARMAX

Após a identificação do modelo ARX, foi desenvolvido também um modelo ARMAX. O modelo ARMAX é uma extensão natural do ARX, pois além dos polinômios que representam a dinâmica da planta (A e B), ele incorpora o polinômio $C(z^{-1})$, responsável por modelar a dinâmica do ruído como um processo de média móvel. Essa característica o torna particularmente útil em sistemas sujeitos a perturbações, flutuações de chaveamento ou medições ruidosas.

Seguindo a mesma metodologia utilizada para o modelo ARX, o processo de identificação do ARMAX foi realizado com base somente no primeiro degrau presente no sinal de entrada. Primeiramente, o código identificou automaticamente o instante de ocorrência desse degrau, extraindo em seguida uma janela contendo tanto uma fração anterior ao salto quanto todo o transitório associado à subida da saída. Esse procedimento concentra a identificação em uma região dinâmica significativa.

Com esse trecho selecionado, realizou-se uma varredura das ordens na, nb, nc e nk, variando de 1 a 5, utilizando como critério de escolha o valor mínimo do FPE (*Final Prediction Error*). O modelo ARMAX escolhido apresentou estrutura $(na, nb, nc, nk) = (5,2,5,1)$, indicando que tanto a parte autoregressiva quanto a parte de ruído possuem quinta ordem, enquanto a influência da entrada foi representada por dois coeficientes com atraso puro de uma amostra.

Depois da identificação local, o modelo ARMAX foi submetido à aplicação global. Para isso, ele foi simulado utilizando-se todo o vetor de entrada do experimento, permitindo avaliar sua capacidade de generalização para além do primeiro degrau utilizado na estimação. Em seguida, a resposta prevista foi comparada com a saída real, gerando tanto o MSE global quanto o coeficiente de determinação R^2. Os resultados globais poderão ser vistos no próximo capítulo deste trabalho.

3.4 Treinamento e validação da rede neural NARX

Nesta etapa foi desenvolvida a modelagem do sistema a partir dos dados obtidos na simulação do conversor Buck, empregando-se uma rede neural do tipo NARX *(Nonlinear AutoRegressive with eXogenous Input)*. Essa arquitetura foi escolhida por

sua capacidade de representar sistemas dinâmicos não lineares e por considerar tanto os valores passados da saída quanto os da entrada, permitindo capturar adequadamente o comportamento temporal do sistema. A implementação e o treinamento foram realizados no ambiente *MATLAB*, utilizando a *toolbox Neural Network*.

O script completo foi elaborado de forma modular, iniciando-se pela importação dos dados simulados e pela preparação dos sinais para o treinamento. O arquivo *"saida_simulacao.mat"*, exportado do *Simulink*, contém duas variáveis principais: o vetor de tempo (*tout*) e o vetor de saída (*simout*), que representam respectivamente o domínio temporal e o valor da tensão de saída do conversor. Esses dados foram carregados e tratados conforme o trecho de código a seguir.

Figura 16 – Código de carregamento de dados.

```
7   %% Carregar dados da simulação
8   load("C:\Users\joaop\Downloads\img\saida_simulação.mat");
9
10  t = out.tout;          % tempo
11  y = out.simout;        % sinal de saída
12  Ts = mean(diff(t));    % tempo de amostragem médio
13
```

Fonte: AUTOR.

O tempo médio de amostragem (T_s) é calculado automaticamente a partir da diferença entre amostras consecutivas do vetor t, o que permite reproduzir no *MATLAB* a mesma relação de tempo utilizada no *Simulink*. Essa etapa é importante pois garante a coerência entre as bases de dados e o modelo neural que será treinado.

Em seguida, foi adicionado ao sinal de saída um ruído de intensidade controlada, simulando a presença de incertezas ou medições ruidosas, comuns em sistemas reais. Esse ruído segue uma distribuição normal com média zero e variância proporcional à amplitude do sinal. O parâmetro *"nivel_ruido"* (Figura 17) pode ser ajustado para introduzir diferentes intensidades de ruído, permitindo analisar posteriormente a robustez do modelo NARX frente a sinais degradados. No caso inicial, o ruído foi mantido nulo para avaliar o desempenho puro da rede com dados ideais. No entanto, em outras simulações, esse valor de ruído será importante para testes de estresse na rede.

Figura 17 – Código para adicionar ruido no sinal de saída.

```
14    % Adicionar ruído à saída simulada
15    nivel_ruido = 0;
16    amplitude_ruido = nivel_ruido * max(y);
17    y = y + amplitude_ruido * randn(size(y));
18
```

Fonte: AUTOR.

Após o pré-processamento, os dados foram divididos em conjuntos de treinamento e teste, correspondendo a 40% e 60% do total de amostras, respectivamente. Essa divisão é feita de forma sequencial, garantindo que o treinamento utilize apenas as primeiras amostras do sinal, enquanto as demais são reservadas para validação do modelo após o aprendizado. É importante saber também que o total de amostras é de 12.501 medições, garantindo que 40% para aprendizagem ainda seja bem eficiente.

Figura 18 – Código para separação de dados entre treino e teste.

```
23    %% Dividir em treino e teste
24    nTreino = round(0.4 * length(t));
25    u_train = u(1:nTreino);
26    y_train = y(1:nTreino);
27    u_test  = u(nTreino+1:end);
28    y_test  = y(nTreino+1:end);
```

Fonte: AUTOR.

A criação da rede foi realizada com o comando *narxnet*, definindo atrasos de entrada e realimentação de 1 a 5 amostras, e uma camada oculta com 10 neurônios. A configuração de divisão interna dos dados foi ajustada para 60% de treinamento, 20% de validação e 20% de teste, como apresentado na Figura 19.

Figura 19 – Código sobre uso do *narxnet*.

```
%% 4) Criar a rede NARX
inputDelays = 1:5;
feedbackDelays = 1:5;
hiddenNeurons = 10;
net = narxnet(inputDelays, feedbackDelays, hiddenNeurons);

net.divideParam.trainRatio = 0.6;
net.divideParam.valRatio = 0.2;
net.divideParam.testRatio = 0.2;
```

Fonte: AUTOR.

Os parâmetros de divisão controlam como o *MATLAB* separa os dados dentro do próprio conjunto de treinamento, garantindo um monitoramento contínuo do desempenho da rede ao longo das iterações. O conjunto de validação é usado para detectar *overfitting*, interrompendo o treinamento quando o erro de validação deixa de diminuir.

No Apêndice B deste trabalho, será encontrado o código completo para a criação e treino da rede neural NARX aplicados nesta esta pesquisa.

3.5 Comparação e análise dos resultados

Nesta seção é finalmente apresentada a comparação e análise dos resultados obtidos a partir da aplicação dos modelos ARX, ARMAX e NARX à resposta do conversor *Buck*. O objetivo neste capítulo é avaliar o desempenho de cada abordagem de identificação, considerando critérios como a capacidade de representação da dinâmica do sistema, o erro de predição e o comportamento dos modelos. As subseções seguintes detalham individualmente os resultados alcançados por cada modelo permitindo uma análise comparativa entre as estruturas utilizadas.

3.5.1 ARX

As métricas quantitativas calculadas incluem o erro médio quadrático (MSE) e o coeficiente de determinação (R^2). O MSE expressa o desvio médio entre as respostas real e prevista, constituindo uma medida direta do erro de modelagem. O coeficiente

R^2, por sua vez, indica o quanto da variabilidade da saída real é explicada pelo modelo ARX; valores próximos de 1 caracterizam elevada fidelidade da aproximação linear. A combinação dessas métricas, somada à inspeção do gráfico comparativo final, permite concluir sobre a adequação e as limitações do modelo ARX quando aplicado globalmente.

Figura 20 – Parâmetros do modelo ARX obtido no *MATLAB*.

```
Command Window
Modelo ARX identificado (apenas 1º degrau):
  idpoly with properties:

                A: [1 -3.5886 5.1479 -3.8783 1.6707 -0.3518]
                B: [0 -5.4066e-06 5.2226e-05 -3.1898e-05 -3.2428e-05 2.1573e-05]
                C: 1
                D: 1
                F: 1
   IntegrateNoise: 0
         Variable: 'z^-1'
          IODelay: 0
        Structure: [1x1 pmodel.polynomial]
    NoiseVariance: 1.7914e-08
       InputDelay: 0
      OutputDelay: 0
        InputName: {'u1'}
        InputUnit: {''}
       InputGroup: [1x1 struct]
       OutputName: {'y1'}
       OutputUnit: {''}
      OutputGroup: [1x1 struct]
            Notes: [0x1 string]
         UserData: []
             Name: ''
               Ts: 8.0000e-07
         TimeUnit: 'seconds'
     SamplingGrid: [1x1 struct]
           Report: [1x1 idresults.arx]
```

Fonte: AUTOR.

Figura 21 – Métricas obtidas para o modelo ARX.

```
Command Window
MSE global: 5.939207e-01
R² global:  0.9972
>>
```

Fonte: AUTOR.

O modelo ARX estimado a partir do primeiro degrau resultou em uma estrutura de quinta ordem, com parâmetros $(na, nb, nk) = (5,5,1)$, apresentando coerência com a dinâmica rápida e de múltiplas constantes de tempo do conversor Buck. O polinômio $A(z^{-1})$ possui cinco termos, representando a contribuição das saídas anteriores, enquanto o polinômio $B(z^{-1})$ apresenta cinco coeficientes associados às entradas defasadas, refletindo a influência do sinal de duty cycle sobre a tensão de saída. O atraso identificado, $nk = 1$, é compatível com o comportamento físico do sistema, já que a resposta do conversor não ocorre instantaneamente após a aplicação da entrada devido à dinâmica LC e ao tempo de cálculo do modelo discreto.

Após a identificação local, o modelo foi aplicado ao conjunto completo de dados da simulação, abrangendo todos os degraus e variações presentes no experimento. O desempenho global obtido teve erro médio quadrático (MSE) igual a $0{,}594\ V^2$. O coeficiente de determinação (R^2) calculado para todo o intervalo temporal foi de 0,9972, indicando que 99,72% da variabilidade da saída real é explicada pelo modelo linear. Esses resultados demonstram que, embora baseado exclusivamente no primeiro degrau, o modelo ARX foi capaz de generalizar adequadamente para os demais regimes de operação, apresentando elevada fidelidade em relação à resposta real do conversor.

Tabela 3 – Coeficientes do modelo ARX identificado $(na = 5, nb = 5, nk = 1)$.

Termo	Coeficiente
a_0	1,0000
a_1	-3,5886
a_2	5,1479
a_3	-3,8783
a_4	1,6707
a_5	-0,3518
b_0	0,0000
b_1	-5,4066e-6
b_2	5,2226e-5

b_3	-3,1898e-5
b_4	-3,2428e-5
b_5	2,1573e-5

Fonte: AUTOR.

Tabela 4 – Resumo dos parâmetros adicionais do modelo ARX

Parâmetro	Valor
Ordem na	5
Ordem nb	5
Atraso nk	1 amostra
Variância do ruído	1,7914e-8
Tempo de amostragem T_s	$8 \cdot 10^{-7}$

Fonte: AUTOR

Sendo assim:

$$A(z^{-1})y(t) = B(z^{-1})u(t-1) \tag{3.13}$$

onde,

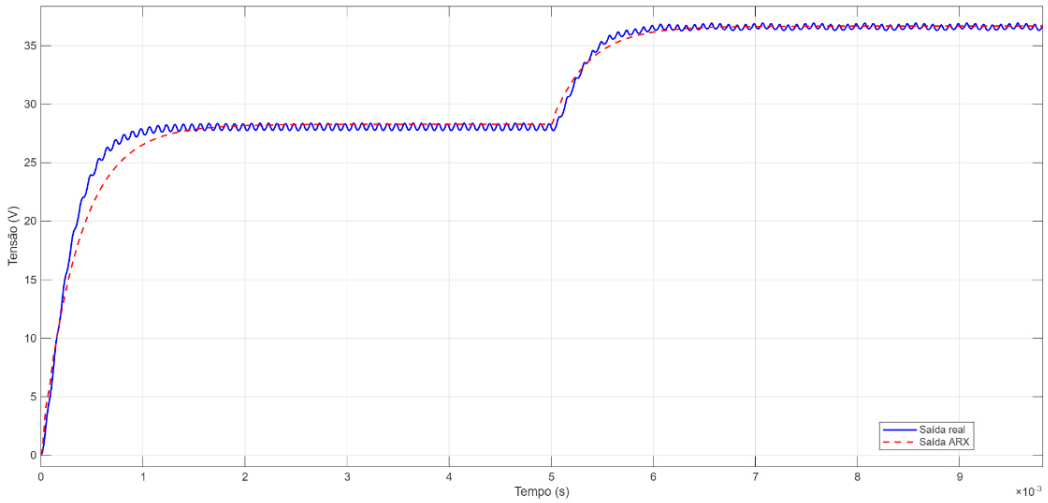
$$A(z^{-1}) = 1 - 3{,}5886z^{-1} + 5{,}1479z^{-2} - 3{,}8783z^{-3} + 1{,}6707z^{-4} - 0{,}3518z^{5} \tag{2.14}$$

e

$$B(z^{-1}) = -5{,}4066 \cdot 10^{-6}z^{-1} + 5{,}2226 \cdot 10^{-5}z^{-1} - 3{,}1898 \cdot 10^{-5}z^{-3} \\ -3{,}2428 \cdot 10^{-5}z^{-4} + 2{,}1573 \cdot 10^{-5}z^{-5} \tag{2.15}$$

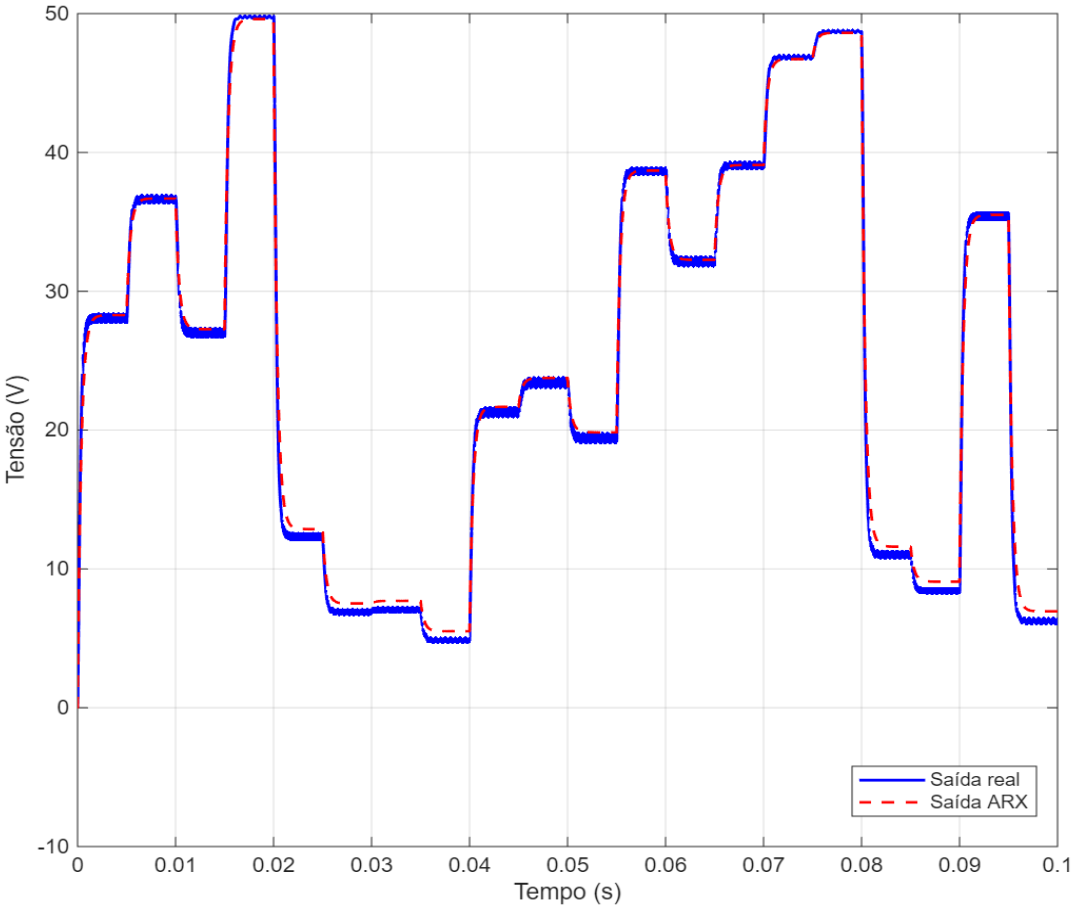
Nas figuras a seguir, pode-se verificar a resposta do modelo para os dados:

Figura 22 – Comparação entre saída real e saída do modelo ARX (Dois primeiros degraus).



Fonte: AUTOR.

Figura 23 – Comparação entre saída real e saída do modelo ARX (Geral)



Fonte: AUTOR.

3.5.2 ARMAX

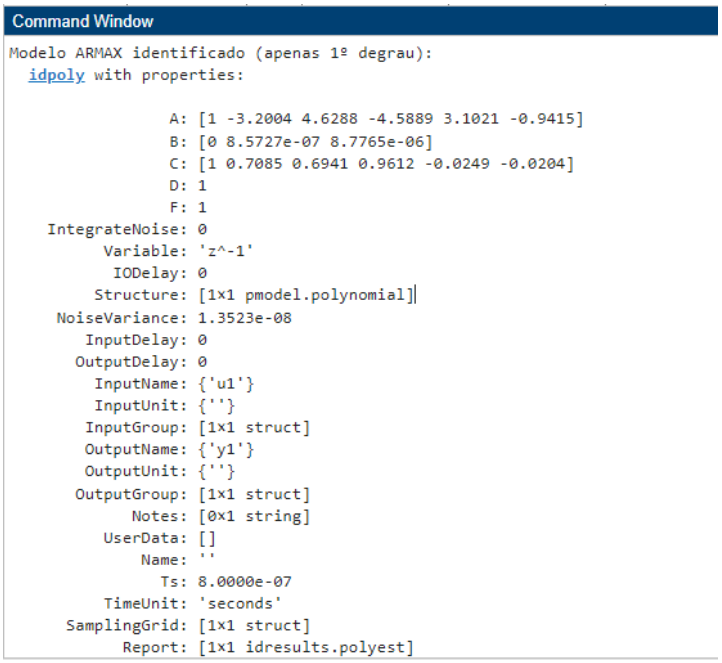
A Tabela 5 apresenta os coeficientes extraídos dos polinômios $A(z^{-1})$, $B(z^{-1})$e $C(z^{-1})$do modelo identificado, de acordo com o que foi observado na Figura 24 – Parâmetros do modelo ARMAX obtido no *MATLAB*..

Tabela 5 – Coeficientes dos polinômios do modelo ARMAX.

Termo	Coeficiente
a_0	1,0000
a_1	-3,2004
a_2	4,6288
a_3	-4,5889
a_4	3,1021
a_5	-0,9415
b_0	0,0000
b_1	8,5727e-7
b_2	8,7765e-6
c_0	1,0000
c_1	0,7085
c_2	0,6941
c_3	0,9612
c_4	-0,0249
c_5	-0,0204

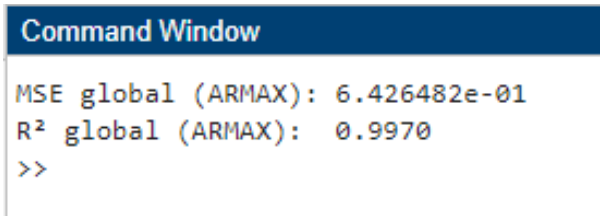
Fonte: AUTOR

Figura 24 – Parâmetros do modelo ARMAX obtido no *MATLAB.*



Fonte: AUTOR.

Figura 25 – Métricas referente ao modelo ARMAX.

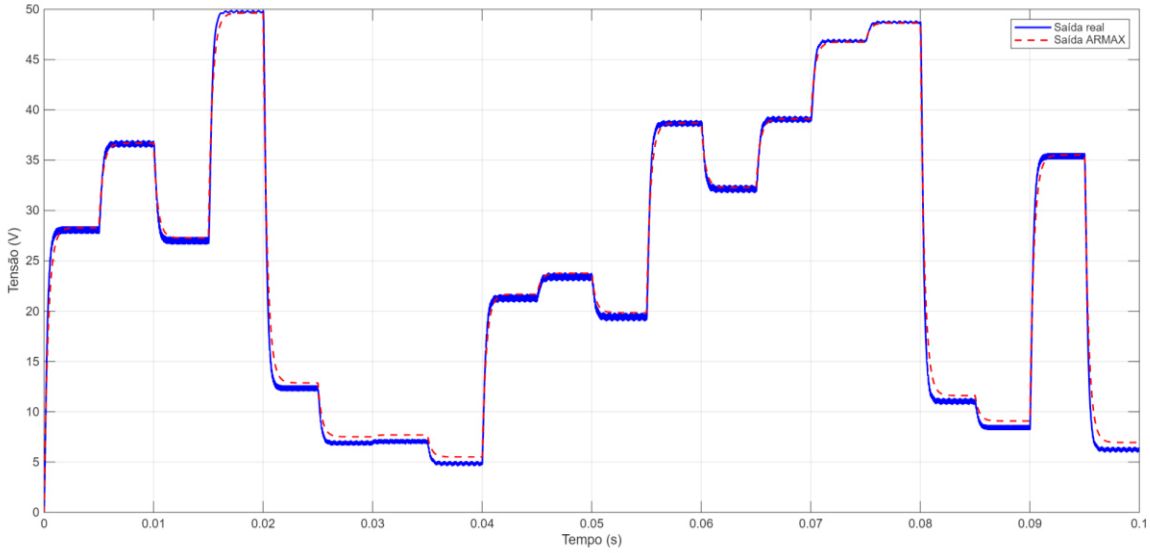


Fonte: AUTOR

Esses valores indicam que o modelo ARMAX explica aproximadamente 99,7% da variabilidade da saída real ao longo de todo o intervalo da simulação, desempenho extremamente elevado para um modelo linear ajustado apenas a partir de um único transitório. Comparativamente, o ARMAX apresentou desempenho muito próximo ao do ARX, o que está de acordo com a baixa variância de ruído dos dados, indicando que o termo adicional $C(z^{-1})$ contribuiu apenas marginalmente para a melhoria do ajuste.

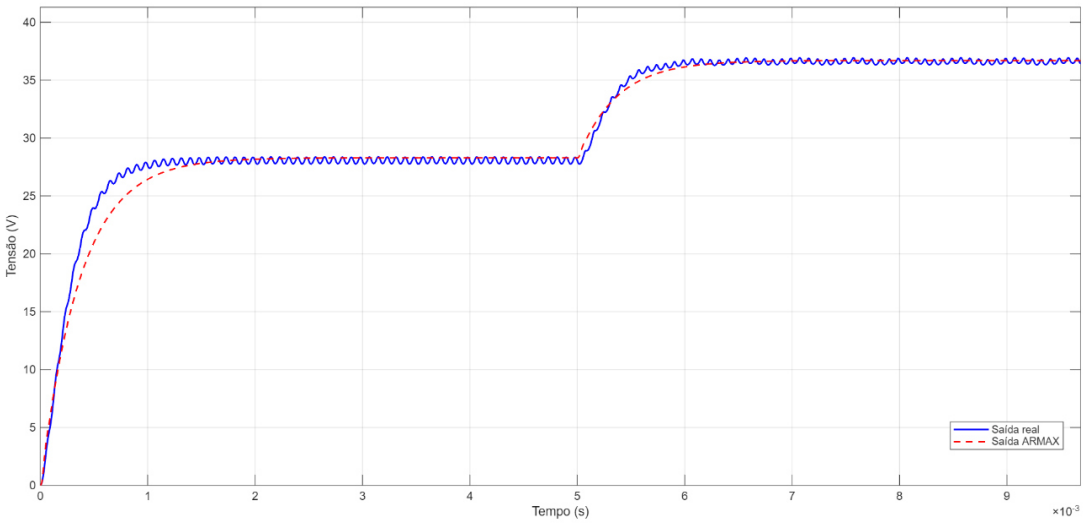
Por fim, o gráfico comparativo entre as curvas real e prevista demonstra que o modelo ARMAX é capaz de acompanhar muito bem tanto o formato quanto a dinâmica da resposta do conversor, apresentando discrepâncias pequenas e concentradas em regiões onde se percebe maior não linearidade na operação chaveada. Dessa forma, o ARMAX se apresenta como uma alternativa linear robusta e coerente dentro do conjunto de técnicas de modelagem avaliadas neste trabalho.

Figura 26 – Comparação entre saída real e saída do modelo ARMAX (Geral).



Fonte: AUTOR

Figura 27 – Comparação entre saída real e saída do modelo ARMAX (Dois primeiros degraus).



Fonte: AUTOR

3.5.3 NARX

Durante o processo de treinamento, o *MATLAB* exibe automaticamente a janela *Training Progress,* que inclui tanto gráficos quanto uma tabela com informações detalhadas sobre o estado do aprendizado.

A tabela exibida na Figura 28, apresenta um exemplo desse relatório, que resume parâmetros como desempenho, gradiente, fator adaptativo *(mu)* e número de verificações de validação.

Figura 28 – Relatório de desempenho de treinamento da rede.

Training Progress

Unit	Initial Value	Stopped Value	Target Value
Epoch	0	22	1000
Elapsed Time	-	00:00:06	-
Performance	172	3.53e-07	0
Gradient	1.39e+03	0.0615	1e-07
Mu	0.001	1e-06	1e+10
Validation Checks	0	6	6

Fonte: AUTOR.

A coluna Performance representa o valor do erro médio quadrático *(Mean Squared Error),* que mede a diferença média entre as saídas reais e as saídas previstas pela rede. Observa-se que o valor caiu de 172 para 3.53×10^{-7} durante o processo, evidenciando um ajuste extremamente preciso do modelo à dinâmica do sistema.

O parâmetro *Gradient* indica a taxa de variação do erro em relação aos pesos sinápticos, e quanto menor esse valor, mais próxima a rede está do ponto ótimo de convergência. Ao final do treinamento, o gradiente reduziu de 1.39×10^{3} para 6.15×10^{-2}, o que sugere a estabilização do aprendizado.

O termo Mu corresponde ao fator de ajuste do algoritmo de *Levenberg–Marquardt,* utilizado pelo *MATLAB* como método de otimização. Esse fator regula o tamanho do passo de atualização dos pesos: valores maiores tornam a convergência mais conservadora, enquanto valores menores aceleram o processo.

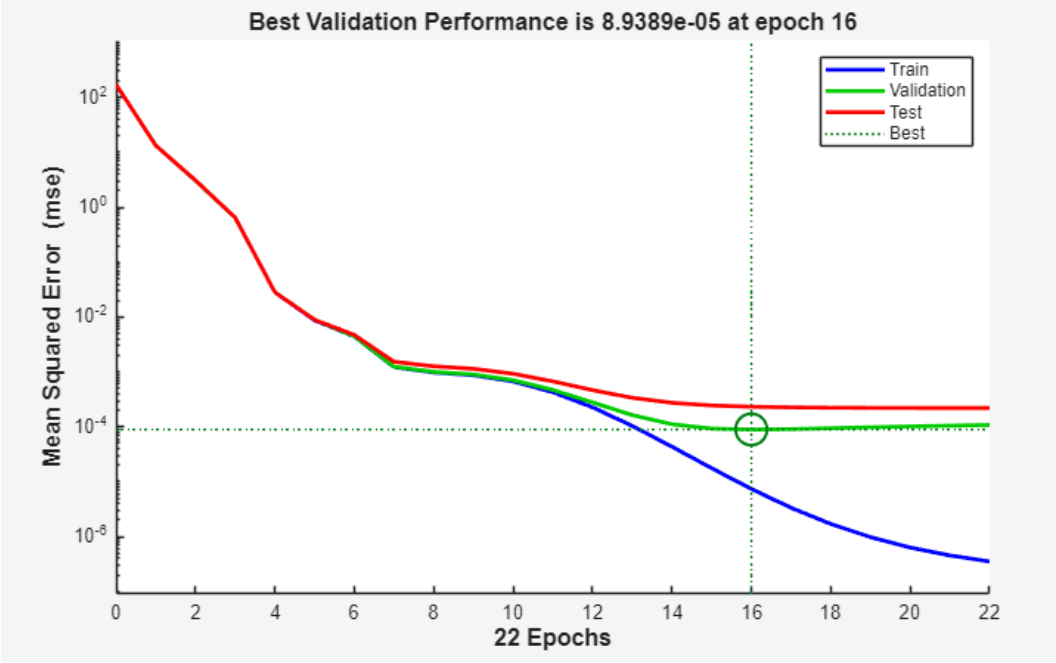
No decorrer do treinamento, *Mu* variou de 0.001 até 1×10^{-6}, indicando que o algoritmo ajustou dinamicamente a taxa de aprendizado conforme o erro diminuía, favorecendo uma convergência suave e estável.

Já o campo *Validation Checks* indica o número de vezes consecutivas em que o erro de validação não melhorou. Quando esse valor atinge o limite máximo (neste caso, 6), o treinamento é interrompido automaticamente, mecanismo conhecido como *early stopping,* que evita o sobreajuste do modelo aos dados de treinamento.

Por fim, o tempo total decorrido *(Elapsed Time)* até o ponto de melhor validação foi de apenas 6 segundos, demonstrando que a rede, mesmo sendo recorrente, apresentou baixo custo computacional para o conjunto de dados utilizado.

Além da tabela numérica, o *MATLAB* fornece automaticamente diversos gráficos de desempenho e diagnóstico. O mais relevante é o gráfico de desempenho *(Training Performance),* Figura 29, que mostra a evolução do erro de treinamento e de validação ao longo das épocas. Esse gráfico é útil para identificar o ponto em que a rede atinge o menor erro de validação, indicando o melhor ajuste possível antes do início de um possível *overfitting*.

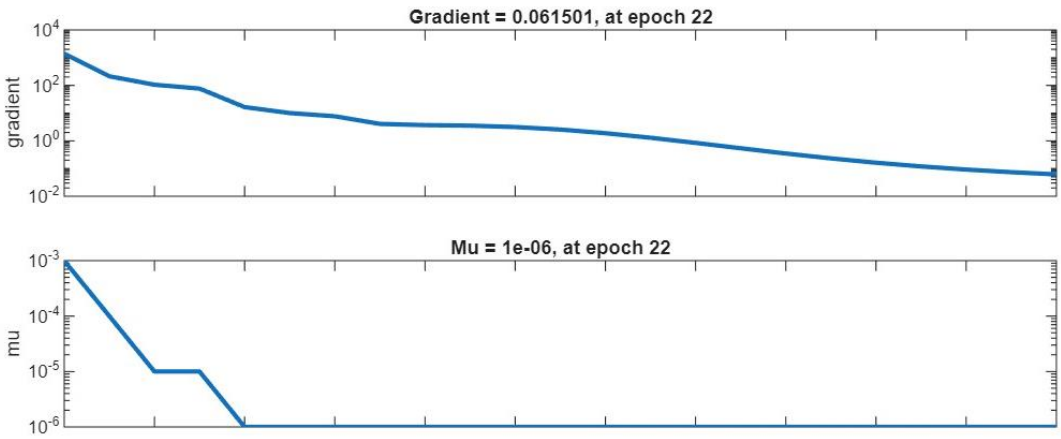
Figura 29 – Validação ao longo das épocas.



Fonte: AUTOR.

Outro gráfico que pode ser apresentado é o *Training State*, que mostra a variação do gradiente e do parâmetro Mu durante o aprendizado. A redução simultânea de ambos confirma a estabilidade do processo de otimização.

Figura 30 – Curvas de *Training State* do *MATLAB*.



Fonte: AUTOR.

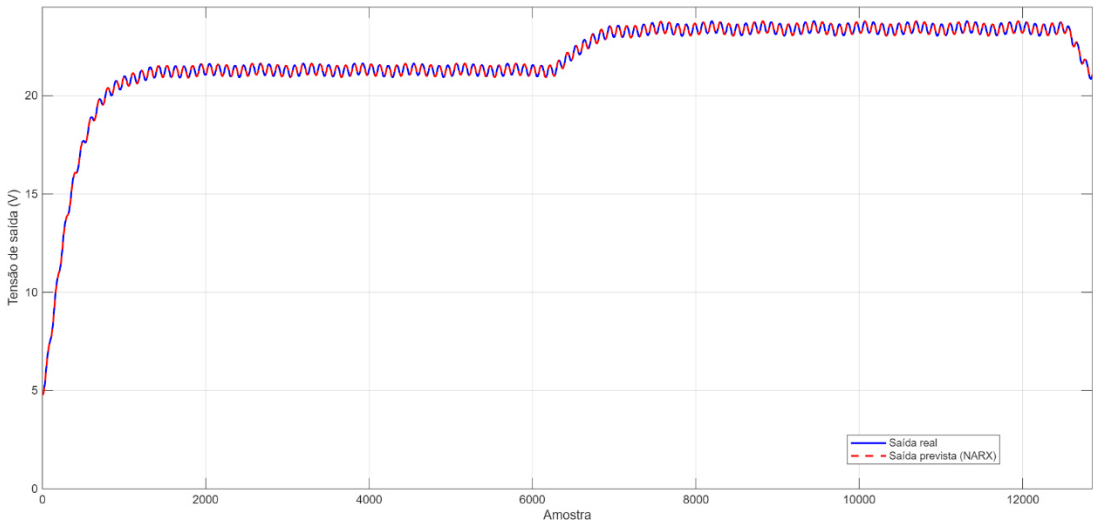
Após o término do treinamento, a rede foi submetida ao conjunto de teste, formado por dados que não foram utilizados no aprendizado, para verificar sua capacidade de generalização.

A saída prevista pela rede *(y_pred)* foi comparada à saída real *(y_real),* e o erro médio quadrático foi calculado conforme a equação (2.11).

A comparação entre as duas saídas mostra que a rede NARX conseguiu prever muito bem o comportamento do conversor Buck. No teste final, o MSE ficou em torno de $94 \cdot 10^{-6}$, um valor bem baixo. Já no gráfico do *Training Progress*, aparece um MSE de $8{,}93 \cdot 10^{-5}$ na época 16, e isso pode gerar dúvida à primeira vista. No entanto, essa diferença é normal, porque o gráfico mostra o erro durante o treinamento e validação, enquanto o código calcula o erro usando o conjunto de teste, que a rede só vê depois de treinada. Como os conjuntos são diferentes, os valores do MSE também variam um pouco. Mesmo assim, ambos os números são bem pequenos, indicando que a NARX representou a planta com muita precisão.

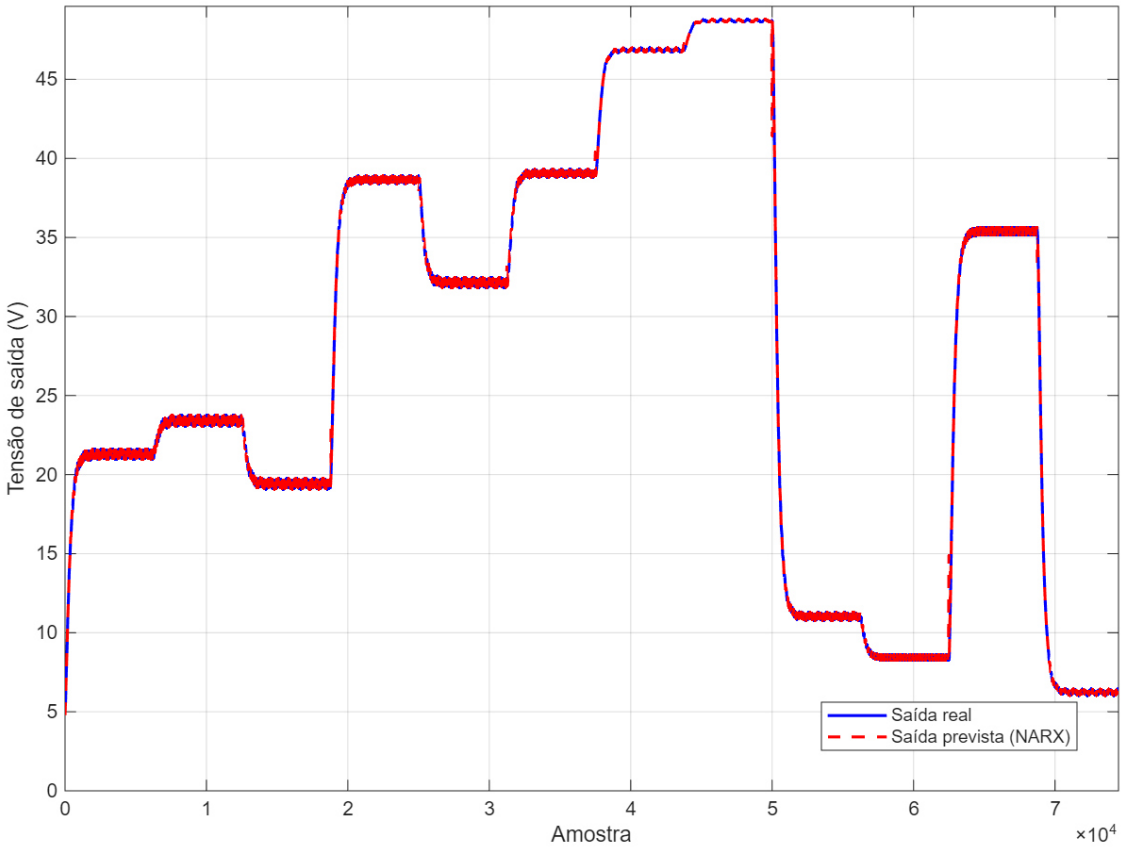
As duas figuras a seguir mostram o resultado de previsão do modelo:

Figura 31 – Comparação entre saída real e saída prevista pela rede NARX (Dois primeiros degraus).



Fonte: AUTOR.

Figura 32 – Comparação entre saída real e saída prevista pela rede NARX (Geral).



Fonte: AUTOR.

3.6 Discussão dos resultados obtidos

Para todos os modelos utilizados neste trabalho ARX, ARMAX e NARX, MSE e R² são apresentados na Tabela 6. No geral, pode-se dizer que os modelos lineares apresentaram a dinâmica geral da planta muito bem, tendo valores de R² muito próximos de 1 e erros relativamente baixos. No entanto, observamos a partir da comparação desses resultados com a rede NARX, a distinção é muito clara. A NARX apresentou um MSE de apenas $9{,}4 \times 10^{-5}$ e R² igual a 1, ou seja, conseguiu praticamente “encaixar” a saída real com pouquíssima diferença. Isso deixa evidente que, apesar de os modelos clássicos funcionarem bem na tendência geral, a rede neural alcança um nível de fidelidade muito maior na reprodução da dinâmica do sistema.

Tabela 6 – Comparação de métricas entre os modelos

Modelo	MSE	R^2	Descrição do desempenho
ARX	$5{,}939207 \times 10^{-1}$	$0{,}9972$	Representa bem a dinâmica geral; modelo linear simples com boa precisão global.
ARMAX	$6{,}426482 \times 10^{-1}$	$0{,}9970$	Semelhante ao ARX, incorpora parte estocástica; desempenho ligeiramente inferior neste caso.
NARX	$9{,}4 \times 10^{-5}$	$1{,}0000$	Melhor desempenho; captura detalhes e não linearidades, reproduzindo a saída com alta fidelidade.

Fonte: AUTOR.

Essa diferença de desempenho não se limita apenas aos indicadores numéricos. Visualmente, é possível observar nas figuras de comparação que a NARX aprende detalhes da dinâmica que não aparecem nas respostas dos modelos lineares. Na Figura 31, nota-se claramente que a rede NARX reproduz com precisão o *ripple* presente na saída real da planta. Esse comportamento não é capturado pelos modelos ARX e ARMAX, como mostram a Figura 22 e a Figura 27. Ambos os modelos suavizam a resposta, deixando de representar as variações rápidas que fazem parte da dinâmica real do sistema.

Essa capacidade da NARX de aprender essas pequenas não linearidades reforça sua vantagem em relação aos modelos clássicos, principalmente em sistemas onde essas oscilações carregam informações importantes sobre o comportamento da planta. Em contrapartida, como o ARX e o ARMAX são modelos lineares, eles tendem a ajustar apenas a tendência principal, reproduzindo a forma geral da curva, mas sem acompanhar os detalhes de alta frequência.

Outro ponto importante é que, durante o desenvolvimento do trabalho, foram realizados testes com diferentes níveis de ruído no sinal de entrada, na tentativa de avaliar a robustez dos modelos. No entanto, observou-se que o conversor naturalmente atenua o ruído devido ao filtro *LC* presente em sua estrutura. Mesmo com ruídos relativamente altos aplicados ao *duty cycle*, a saída da planta não apresentou uma saída relevante, o que impossibilitou uma análise de desempenho dos modelos em cenários ruidosos.

No geral, esses resultados mostram que, embora os modelos lineares sejam ferramentas rápidas, simples e eficazes para muitos tipos de sistemas, a rede NARX se destaca quando o objetivo é capturar com maior precisão os detalhes da dinâmica, incluindo não linearidades e pequenas oscilações que fazem parte do comportamento natural do conversor.

4 CONCLUSÕES E PROPOSTAS DE CONTINUIDADE

Com base nos resultados deste estudo, pode-se concluir que as vantagens dos métodos de identificação são distintas, e a seleção depende muito da aplicação. Os modelos lineares ARX e ARMAX mostraram bom desempenho tanto na representação global do sistema quanto têm a vantagem especial de gerar expressões matemáticas explícitas, o que possibilita interpretá-las e usá-las diretamente na construção de controladores clássicos.

Por outro lado, a rede neural NARX exibiu o melhor desempenho entre os três modelos, conseguindo reproduzir características da dinâmica que foram omitidas nos modelos lineares, como ondulações na saída da planta. Há uma desvantagem nessa abordagem: ela não fornece uma representação analítica do modelo, o conhecimento está "embutido" na rede, o que pode limitar quaisquer aplicações em torno de equações explícitas.

No entanto, mesmo ao aplicar altos níveis de ruído ao sinal de entrada, a planta não mostrou mudanças significativas devido ao seu forte caráter de filtragem. Foi impossível, portanto, testar a robustez dos modelos em cenários ruidosos.

Em uma continuação deste estudo, as mesmas técnicas podem ser aplicadas a qualquer planta com comportamento mais não linear ou menos filtrado, onde o ruído tem um efeito genuíno na saída. Além do exposto, pode ser uma possibilidade investigar redes LSTM, modelos NARMAX ou híbridos para a identificação do sistema.

REFERÊNCIAS

1. AGGOUNE, L.; CHETOUANI, Y.; RADJEAI, H. ***Recursive Identification of the Dynamic Behavior in a Distillation Column by Means of Autoregressive Models.*** Journal of Dynamic Systems, Measurement, and Control, v. 136, n. 6, p. 061009, 2014. DOI: 10.1115/1.4026837.

2. AGUIRRE, L. A. **Introdução à Identificação de Sistemas: Técnicas Lineares e Não Lineares Aplicadas a Sistemas Reais.** 2. ed. Belo Horizonte: UFMG, 2004.

3. AQUIZE, R.; CAJAHUARINGA, A.; MACHUCA, J.; MAURICIO, D.; MAURICIO VILLANUEVA, J. M. **System identification methodology of a gas turbine based on artificial recurrent neural networks.** Sensors, v. 23, n. 4, p. 2231, 2023. DOI: https://doi.org/10.3390/s23042231.

4. ARAÚJO, Thainara de. **Estimação de séries temporais via rede NARX em aplicações industriais.** Dissertação (Mestrado em Engenharia Elétrica) – Universidade Federal de Mato Grosso do Sul, Campo Grande, MS, 2022. Disponível em: https://repositorio.ufms.br/handle/123456789/4482 . Acesso em: 15 nov. 2025.

5. ARAÚJO, Valbério Gonzaga de. **Monitoramento e diagnóstico de falhas em motores de indução trifásicos utilizando rede neural NARX.** Dissertação (Doutorado em Engenharia Elétrica) – Universidade Federal do Rio Grande do Norte, Natal, 2024. Disponível em: https://repositorio.ufrn.br/items/b8454a8f-7621-4466-bfb2-32d08d726337. Acesso em: 14 nov. 2025.

6. ARBIB, M. A. **The Handbook of Brain Theory and Neural Networks. MIT Press,** 2002.

7. DONG, Aoxiang; STARR, Andrew; ZHAO, Yifan. **Neural network-based parametric system identification: a review. International Journal of Systems Science,** v. 54, n. 13, p. 2676-2688, 2023. DOI: 10.1080/00207721.2023.2241957.

8. GONÇALVES, A. R. **Introdução às Redes Neurais Artificiais.** Disponível em: https://andreric.github.io/files/pdfs/redes_neurais.pdf. Acesso em: 6 nov. 2025.

9. HAYKIN, S. **Redes Neurais: princípios e prática.** 2. ed. Porto Alegre: Bookman, 1999.

10. JAMI'IN, Mohammad Abu; HU, Jinglu; MARHABAN, Mohd Hamiruce; SUTRISNO, Imam; MARIUN, Norman Bin. **Quasi-ARX neural network based adaptive predictive control for nonlinear systems. IEEJ Transactions on Electrical and Electronic Engineering,** v. 11, p. 83–90, 2016. DOI: 10.1002/tee.22191

11. KHALED, S.; FAKHRY, M.; MUBARAK, **A. S. Classification of PCG signals using a nonlinear autoregressive network with exogenous inputs (NARX).** In: Proceedings of the 2020 IEEE. [S. l.: s. n.], 2020. p. 98–102.

12. LJUNG, L. **System Identification: Theory for the User.** 2. ed. Upper Saddle River, NJ: Prentice Hall PTR, 1999. ISBN 978-0-13-656695-3.

13. MATHWORKS. **arx (System Identification Toolbox) — MATLAB Function Reference.** Disponível em: https://www.mathworks.com/help/ident/ref/arx.html. Acesso em 15 nov. 2025

14. MATHWORKS. **iddata (System Identification Toolbox) — MATLAB Function Reference.** Disponível em: https://www.mathworks.com/help/ident/ref/iddata.html. Acesso em: 15 nov. 2025.
15. MATHWORKS. **sim (System Identification Toolbox) — MATLAB Function Reference.** Disponível em: https://www.mathworks.com/help/ident/ref/sim.html. Acesso em 15 nov. 2025
16. McCULLOCH, W. S.; PITTS, W. **A Logical Calculus of the Ideas Immanent in Nervous Activity. Bulletin of Mathematical Biophysics,** v. 5, n. 4, p. 115–133, 1943.**
17. MOHAN, Ned; UNDELAND, Tore M.; ROBBINS, William P. **Power Electronics: Converters, Applications and Design.** 2. ed. New York: John Wiley & Sons, 1995.
18. NAYANASIRI, Dulika; LI, Yunwei. **Step-Down DC–DC Converters: An Overview and Outlook. Electronics,** v. 11, n. 11, p. 1693, 2022. DOI: 10.3390/electronics11111693
19. SOUZA, Felipe Maraschin Pereira de. **Aplicação de uma rede neural artificial NARX para obtenção do comportamento dinâmico de um atenuador de impacto de alumínio do tipo honeycomb.** 2019. Dissertação (Mestrado em Engenharia Mecânica) — Programa de Pós-Graduação em Engenharia Mecânica, Universidade Federal da Paraíba, João Pessoa, 2019. Disponível em: https://repositorio.ufpb.br/jspui/bitstream/123456789/19437/1/FelipeMaraschinPereiraDeSouza_Dissert.pdf. Acesso em: 15 nov.2025
20. TAVARES, Marley Fagundes. **Utilização dos modelos ARX e ARMAX em plantas industriais ruidosas**. 2012. Dissertação (Mestrado em Sistemas Dinâmicos) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2012. doi:10.11606/D.18.2012.tde-23012013-144208. Acesso em: 14 nov. 2025.

APÊNDICES

APÊNDICE A – CÓDIGO PARA GERAR VETOR DE VALORES DE DUTY CYCLE.

```
% ============================================================
% Geração de sinal de duty cycle aleatório em degraus
% ============================================================
% Este código cria um sinal de entrada (duty cycle) composto por
% níveis aleatórios constantes em intervalos de tempo fixos,
% utilizado como excitação para o modelo simulado do conversor Buck.
% ============================================================

t = 0 : 800e-9 : 100e-03;   % Vetor de tempo: inicia em 0 e vai até 100 ms,
com passo de 0,8 µs
Ts = 800e-9;                % Passo de amostragem
Tfinal = max(t);            % Tempo total de simulação

duracaoDegrau = 0.005;      % Duração de cada degrau de entrada (5 ms)
Npassos = ceil(Tfinal/duracaoDegrau);  % Quantidade de degraus dentro do
tempo total

% Gerar valores de duty cycle aleatórios entre 5% e 95%
valores = 5 + 95*rand(1, Npassos);

Ns = round(duracaoDegrau/Ts);  % Número de amostras correspondentes a cada
degrau

% Repitir cada valor de duty pelo número de amostras que dura o degrau
u = repelem(valores, Ns);

% Ajustar o tamanho do vetor de entrada para coincidir exatamente com o
tempo total
u = u(1:min(length(u), length(t)));
u = padarray(u, [0, max(0, length(t)-length(u))], 'replicate', 'post');

% Montar a matriz completa com tempo (coluna 1) e duty cycle (coluna 2)
input = [t' u'];

% Converter o sinal em objeto 'timeseries' para uso no Simulink
input_ts = timeseries(u, t);
```

APÊNDICE B – CÓDIGO PARA CRIAÇÃO DE REDE NEURAL (TREINAMENTO, VALIDAÇÃO E TESTE)

```
% ============================================================
%  Identificação com dados reais do Simulink (Rede NARX)
% ============================================================

clear; clc; close all;

%% Carregar dados da simulação
load("C:\Users\joaop\Downloads\img\saida_simulação.mat");

t = out.tout;          % tempo
y = out.simout;        % sinal de saída
Ts = mean(diff(t));    % tempo de amostragem médio

% Adicionar ruído à saída simulada
nivel_ruido = 0;
amplitude_ruido = nivel_ruido * max(y);
y = y + amplitude_ruido * randn(size(y));

% Sinal de entrada (duty cycle)
u = out.valores_dutycycle;
u = u / 100;

%% Dividir em treino e teste
nTreino = round(0.6 * length(t));
u_train = u(1:nTreino);
y_train = y(1:nTreino);
u_test  = u(nTreino+1:end);
y_test  = y(nTreino+1:end);

%% Converter para formato de sequência
u_train = con2seq(u_train');
y_train = con2seq(y_train');
u_test  = con2seq(u_test');
y_test  = con2seq(y_test');

% Criar a rede NARX
inputDelays = 1:5;
feedbackDelays = 1:5;
hiddenNeurons = 10;
net = narxnet(inputDelays, feedbackDelays, hiddenNeurons);

net.divideParam.trainRatio = 0.6;
net.divideParam.valRatio = 0.2;
net.divideParam.testRatio = 0.2;

% Treinar a rede
[Xs, Xi, Ai, Ts_train] = preparets(net, u_train, {}, y_train);
net = train(net, Xs, Ts_train, Xi, Ai);

% Testar a rede
[Xs_test, Xi_test, Ai_test, Ts_test] = preparets(net, u_test, {}, y_test);
y_pred = net(Xs_test, Xi_test, Ai_test);

% Converter e alinhar resultados
y_real = cell2mat(y_test);
y_pred = cell2mat(y_pred);
```

```
delay = max([inputDelays feedbackDelays]);
N = min(length(y_real), length(y_pred)) - delay;
y_real_adj = y_real(delay+1 : delay+N);
y_pred_adj = y_pred(1 : N);

% Plotar comparação
figure;
plot(y_real_adj, 'b', 'LineWidth', 1.5); hold on;
plot(y_pred_adj, 'r--', 'LineWidth', 1.5);
legend('Saída real', 'Saída prevista (NARX)');
xlabel('Amostra');
ylabel('Tensão de saída (V)');
title('Comparação entre saída real e saída prevista pela rede NARX');
grid on;

% Calcular MSE
mse_val = mean((y_real_adj - y_pred_adj).^2);
fprintf('Erro médio quadrático (MSE): %.6f\n', mse_val);
```

APÊNDICE C – CÓDIGO PARA O MODELO ARX

```
% ================================================================
%   Identificação ARX usando apenas o primeiro degrau da entrada
%   Aplicação do modelo ARX identificado em todo o conjunto de dados
% ================================================================

clear; clc; close all;

% Carregar dados
load("C:\Users\joaop\Downloads\img\saida_simulação.mat");

t = out.tout;
y = out.simout;
u = out.valores_dutycycle;
Ts = mean(diff(t));

% Detecção do primeiro degrau do duty cycle

du = diff(u);
tol = max(u) * 0.005;    % 0.5% da escala – evita ruídos pequenos

idx_all = find(abs(du) > tol);   % todos os degraus reais

if isempty(idx_all)
    error("Não foi encontrado nenhum degrau real no sinal de entrada.");
end

idx_step = idx_all(1);   % primeiro degrau de verdade

% Seleção da janela do primeiro degrau
pre_time  = 0.2e-3;                      % 0.2 ms antes
pos_time  = 3e-3;                        % pegar um transitório curto
i1 = max(1, idx_step - round(pre_time/Ts));
i2 = min(length(t), idx_step + round(pos_time/Ts));

u_deg  = u(i1:i2);
y_deg  = y(i1:i2);
t_deg  = t(i1:i2);

% Identificação do modelo arx com o primeiro degrau

data_deg = iddata(y_deg, u_deg, Ts);

melhor_fpe = inf;

for na = 1:5
    for nb = 1:5
        nk = 1;
        try
            sys = arx(data_deg, [na nb nk]);
            if sys.Report.Fit.FPE < melhor_fpe
                melhor_fpe = sys.Report.Fit.FPE;
                model_arx = sys;
                ords = [na nb nk];
            end
        end
    end
end
```

```
% Aplicação do modelo arx a todos os dados

y_arx_total = sim(model_arx, u);

N = min(length(y), length(y_arx_total));
y_real_total = y(1:N);
y_arx_total  = y_arx_total(1:N);
t_total      = t(1:N);

% Métricas globais
mse_global = mean((y_real_total - y_arx_total).^2);
SST = sum((y_real_total - mean(y_real_total)).^2);
SSE = sum((y_real_total - y_arx_total).^2);
R2_global = 1 - SSE/SST;

% Gráfico final – saída real × saída arx (intervalo completo)

figure;
plot(t_total, y_real_total, 'b', 'LineWidth', 1.4); hold on;
plot(t_total, y_arx_total, 'r--', 'LineWidth', 1.3);
xlabel('Tempo (s)');
ylabel('Tensão (V)');
title('Modelo ARX identificado no primeiro degrau e aplicado ao intervalo
completo');
legend('Saída real', 'Saída ARX');
grid on;

% Imprimir os resultados
disp('Modelo ARX identificado (apenas 1° degrau):');
disp(model_arx);
fprintf("MSE global: %.6e\n", mse_global);
fprintf("R² global:  %.4f\n", R2_global);
```

APÊNDICE D – CÓDIGO PARA O MODELO ARMAX.

```
% ================================================================
%   Identificação ARMAX usando apenas o primeiro degrau da entrada
%   Aplicação do modelo ARMAX a todo o conjunto de dados
% ================================================================

clear; clc; close all;

% Carregamento dos dados
load("C:\Users\joaop\Downloads\img\saida_simulação.mat");

t = out.tout;
y = out.simout;
u = out.valores_dutycycle;
Ts = mean(diff(t));

% Detecção do primeiro degrau do duty cycle

du = diff(u);
tol = max(u) * 0.005;      % evita flutuações pequenas

idx_all = find(abs(du) > tol);

if isempty(idx_all)
    error("Nenhum degrau significativo foi encontrado no sinal de
entrada.");
end

idx_step = idx_all(1);      % primeiro degrau real

% Seleção da janela do primeiro degrau
pre_time = 0.2e-3;
pos_time = 3e-3;

i1 = max(1, idx_step - round(pre_time/Ts));
i2 = min(length(t), idx_step + round(pos_time/Ts));

u_deg = u(i1:i2);
y_deg = y(i1:i2);
t_deg = t(i1:i2);

%   Identificação do modelo armax
%   Ordens testadas: na = 1..5, nb = 1..5, nc = 1..5, nk = 1

data_deg = iddata(y_deg, u_deg, Ts);

melhor_fpe = inf;

for na = 1:5
    for nb = 1:5
        for nc = 1:5
            nk = 1;
            try
                sys = armax(data_deg, [na nb nc nk]);
                fpe_atual = sys.Report.Fit.FPE;

                if fpe_atual < melhor_fpe
                    melhor_fpe = fpe_atual;
                    model_armax = sys;
```

```
                    ords = [na nb nc nk];
                end
            end
        end
    end
end

% Aplicação do modelo armax a todo o conjunto de dados

y_armax_total = sim(model_armax, u);

N = min(length(y), length(y_armax_total));
y_real_total = y(1:N);
y_armax_total = y_armax_total(1:N);
t_total = t(1:N);

% Métricas globais
mse_global = mean((y_real_total - y_armax_total).^2);

SST = sum((y_real_total - mean(y_real_total)).^2);
SSE = sum((y_real_total - y_armax_total).^2);
R2_global = 1 - SSE/SST;

% Comparação global – gráfico final

figure;
plot(t_total, y_real_total, 'b', 'LineWidth', 1.4); hold on;
plot(t_total, y_armax_total, 'r--', 'LineWidth', 1.3);
xlabel('Tempo (s)');
ylabel('Tensão (V)');
title('Modelo ARMAX identificado no 1° degrau e aplicado ao intervalo
completo');
legend('Saída real', 'Saída ARMAX');
grid on;

% Resultados
disp('Modelo ARMAX identificado (apenas 1° degrau):');
disp(model_armax);

fprintf("MSE global (ARMAX): %.6e\n", mse_global);
fprintf("R² global (ARMAX):  %.4f\n", R2_global);
```