UNIVERSIDADE ESTADUAL DE CAMPINAS

Faculdade de Engenharia Química

FELIPE MATHEUS MOTA SOUSA

MODELO EMPÍRICO DA DINÂMICA DE PRODUÇÃO DE ETANOL BASEADO EM REDES DE APRENDIZAGEM PROFUNDA LSTM

CAMPINAS
2021

FELIPE MATHEUS MOTA SOUSA

MODELO EMPÍRICO DA DINÂMICA DE PRODUÇÃO DE ETANOL BASEADO EM REDES DE APRENDIZAGEM PROFUNDA LSTM

Dissertação de Mestrado apresentada à Faculdade de Engenharia Química da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de Mestre em Engenharia Química.

Orientador: Prof. Dr. Flávio Vasconcelos da Silva
Coorientador: Prof. Dr. Rodolpho Rodrigues Fonseca

ESTE EXEMPLAR CORRESPONDE À VERSÃO FINAL DA DISSERTAÇÃO DEFENDIDA PELO ALUNO FELIPE MATHEUS MOTA SOUSA E ORIENTADA PELO PROF. DR. FLÁVIO VASCONCELOS DA SILVA.

CAMPINAS
2021

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca da Área de Engenharia e Arquitetura
Rose Meire da Silva - CRB 8/5974

So85m

Sousa, Felipe Matheus Mota, 1995-
Modelo empírico da dinâmica de produção de etanol baseado em redes de aprendizagem profunda LSTM / Felipe Matheus Mota Sousa. – Campinas, SP : [s.n.], 2021.

Orientador: Flávio Vasconcelos da Silva.
Coorientador: Rodolpho Rodrigues Fonseca.
Dissertação (mestrado) – Universidade Estadual de Campinas, Faculdade de Engenharia Química.

1. Fermentação. 2. Modelagem. 3. Aprendizado profundo. 4. Redes neurais recorrentes. 5. Memória de longo e curto prazo. I. Silva, Flávio Vasconcelos da, 1971-. II. Fonseca, Rodolpho Rodrigues, 1987-. III. Universidade Estadual de Campinas. Faculdade de Engenharia Química. IV. Título.

Informações para Biblioteca Digital

Título em outro idioma: Empirical modeling of ethanol production dynamics using long short-term memory recurrent neural networks
Palavras-chave em inglês:
Fermentation
Modeling
Deep learning
Recurrent neural networks
Long short-term memory
Área de concentração: Engenharia Química
Titulação: Mestre em Engenharia Química
Banca examinadora:
Flávio Vasconcelos da Silva [Orientador]
Sidney Nascimento Givigi Junior
Maurício Bezerra de Souza Junior
Data de defesa: 23-02-2021
Programa de Pós-Graduação: Engenharia Química

Identificação e informações acadêmicas do(a) aluno(a)
- ORCID do autor: https://orcid.org/0000-0001-8095-4258
- Currículo Lattes do autor: http://lattes.cnpq.br/6236819130743047

Dissertação de Mestrado defendida por Felipe Matheus Mota Sousa e aprovada em 23 de fevereiro de 2021 pela Comissão Examinadora constituída pelos doutores:

Prof. Dr. Flávio Vasconcelos da Silva
Faculdade de Engenharia Química
Universidade Estadual de Campinas

Prof. Dr. Sidney Nascimento Givigi Junior
Queen's School of Computing
Queen's University

Prof. Dr. Maurício Bezerra de Souza Junior
Departamento de Engenharia Química
Universidade Federal do Rio de Janeiro

A ata da defesa com as respectivas assinaturas dos membros encontra-se no SIGA/Sistema de Fluxo de Dissertação/Tese e na Secretaria do Programa da Unidade.

Agradecimentos

Aos meus pais, Luiz e Joelma, que nunca mediram esforços para prover a melhor criação possível, apesar de todas as adversidades enfrentadas. Crescer sabendo que tinha um alicerce de amor, respeito e confiança foi imprescindível para me tornar quem sou hoje. Não existem palavras suficientes para agradecer.

À toda minha família, mas em especial ao meu irmão Lucas e às minhas madrinhas Valdeci e Carmen que nunca deixaram de me incentivar a perseguir os meus sonhos, mesmo quando eu não acreditava mais neles.

Ao meu orientador Flávio Vasconcelos da Silva que aceitou me acompanhar nessa jornada. Sua confiança em mim para desenvolver este e outros projetos é uma dádiva que jamais poderei agradecer corretamente.

Ao meu coorientador Rodolpho Rodrigues Fonseca, o responsável por me fazer seguir a carreira acadêmica. Agradeço sempre pelo momento que nossos caminhos cruzaram e começamos a trabalhar juntos. Se um dia chegar a ser um décimo do profissional e ser humano que você é, já me considerarei vitorioso.

Aos amigos que fiz na UFS que, apesar da enorme distância que nos separa, estão presentes na minha vida, ouvindo minhas angústias e partilhando risadas. Se perguntarem o que levei de melhor da universidade, serei obrigado a falar seus nomes.

Aos amigos que fiz na Unicamp e que me ampararam durante o mestrado. Vocês preencheram em tão pouco tempo o vazio deixado pela vida que ficou atrás com alegria e uma boa xícara de café.

Aos companheiros do LCAP/LESQ que contribuíram com discussões, ideias e conversas durante o mestrado.

O presente trabalho foi realizado com apoio do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq).

As consequências de nossos atos são sempre tão complexas, tão diversas, que predizer o futuro é uma tarefa realmente difícil.

J.K. Rowling

Resumo

A biotecnologia tem papel fundamental no desenvolvimento de produtos específicos devido à alta complexidade estrutural das substâncias desejadas que impede a obtenção via rota química, à implementação fácil e barata dos processos e ao menor custo de operação. Apesar do amplo uso dos processos biotecnológicos, a operação contínua está sujeita a alta variabilidade dos produtos finais em consequência da grande susceptibilidade dos sistemas biológicos aos efeitos de diversas variáveis. Desse modo, a modelagem fenomenológica torna-se menos precisa por simplificações assumidas no modelo tais como invariabilidade dos parâmetros cinéticos ou por não contabilizar os efeitos de algumas variáveis importantes do sistema, porém, não contempladas no modelo matemático e que muitas vezes estão disponíveis nos bancos de dados. Assim, foram desenvolvidos modelos empíricos baseados em redes recorrentes com topologia *Long Short-Term Memory* para predizer o comportamento dinâmico das variáveis endógenas volume e concentrações de células, de substrato e de etanol em um processo de fermentação alcoólica. O banco de dados foi obtido por meio de simulação numérica empregando um modelo fenomenológico com parâmetros cinéticos variantes por meio de redes neurais artificiais adaptados às condições operacionais propostas. Três variáveis exógenas foram modificadas para gerar o banco de dados e descrever o comportamento das variáveis de processo. O melhor modelo obtido para predição apresentou vinte neurônios e três passos temporais, alta capacidade de generalização como visto pelo baixo valor de MSE de $2{,}21 \cdot 10^{-4}$, grande correlação linear representado pelos coeficientes de determinação R^2 próximos à unidade e não apresentou repetição do valor das entradas endógenas como predições. Entretanto, esse modelo não foi capaz de operar em simulação recursiva, ou seja, empregando suas predições como entradas posteriores, para três diferentes bancos de dados. O MSE total foi calculado em $6{,}98 \cdot 10^{-2}$ e os altos erros relativos indicam a grande influência das variáveis endógenas no desempenho do modelo. Assim, novas redes com maior complexidade da topologia foram treinadas para verificar a viabilidade de usá-las em modo recursivo. Entre os novos modelos, a rede com 30 neurônios, 7 passos temporais e duas unidades LSTM empilhadas apresentou melhor desempenho com MSE de $1{,}32 \cdot 10^{-3}$. Os erros percentuais foram reduzidos para um valor máximo de 12%, e as respostas preditas distam dos valores reais em aproximadamente a faixa de precisão dos sensores comerciais.

Palavras-chave: Fermentação; modelagem empírica; aprendizagem profunda; redes neurais recorrentes; LSTM.

Abstract

Biotechnology has a central role in the manufacturing of specific products due to the high structural complexity of the desired substances that prevents obtaining them via chemical route, the easy and inexpensive implementation and the lower cost of operation. Despite the widespread use of biotechnology processes, the continuous operation is subject to high variability of final products as result of the susceptibility of biological systems to the effects of several variables. In this way, phenomenological modeling becomes less precise due to the simplifications of the proposed model such as kinetic parameters invariability or because it does not account for the effects of some important system variables that are not included in the mathematical model, although they are often available in databases. Thereby, empirical models based on Long Short-Term Memory recurrent neural networks were developed to predict the dynamic behavior of the endogenous variables volume and cell, substrate and ethanol concentrations in a alcoholic fermentation process. The database was generated through numerical simulation applying a phenomenological model in which the kinetic parameters were varied by an artificial neural network according to the operational conditions. Three exogenous variables were modified to generate the database that describes the process variables behavior. The best prediction model obtained had ten neurons and one time-step, high generalization capability as shown by its low MSE of $2{,}21 \cdot 10^{-4}$, high linear correlation indicated by the determination coefficients R^2 near to unity and it did not show repetition of the endogenous input values as predictions. However, this model was not able to operate in a recursive simulation, that is, using its predictions as future inputs, for three different datasets. The total MSE calculated as $6.98 \cdot 10^{-2}$ and the high relative errors indicate that the endogenous inputs have great influence on the model performance. Thereby, more complex networks were trained to verify the viability of using them in recursive mode. Amongst the new models, the 2 stacked LSTM, 30 neurons and 7 time-steps network showed the best results with MSE equals to $1.32 \cdot 10^{-3}$. Maximum relative errors were reduced to 12% and the predicted responses differ from the real values in approximately the accuracy range of commercial sensors.

Keywords: Fermentation; empirical model; deep learning, recurrent neural networks; LSTM.

Lista de Figuras

Lista de Tabelas

Nomenclaturas

Abreviações e siglas

ANN	Rede neural artificial
BPTT	*Backpropagation through time*
LSTM	*Long-Short Term Memory*
MSE	Média dos erros quadráticos
RAP	Rede de aprendizagem profunda
RNN	Rede neural recorrente

Letras gregas

α	Coeficiente de energia para crescimento [g_P/g_X]
β	coeficiente de energia para manutenção [$g_P/(g_X \cdot h)$]
γ_1	Peso da regularização L1
γ_2	Peso da regularização L2
μ	Taxa de crescimento específica [h^{-1}]
μ_{max}	Taxa de crescimento específica máxima [h^{-1}]
$\Phi(\cdot)$	Função de ativação tangente hiperbólica
$\Psi(\cdot)$	Função de ativação genérica
$\sigma(\cdot)$	Função de ativação sigmoide

Símbolos

$\odot$	Multiplicação de Hadamard ou ponto-a-ponto
$\mathbf{W}$	Matriz de pesos da rede neural
$\mathbf{X}$	Matriz de variáveis de entrada
b	Viés da rede neural
c	Célula de memória
D	Tempo de detenção hidráulico [h]

f	*Forget gate*
h	Estado oculto
i	*Input gate*
J	Função custo
K_d	Taxa específica de morte celular [h^{-1}]
K_P	Concentração de inibição por produto [g_P/L]
K_S	Constante de saturação [g_S/L]
$K_S^{'}$	Concentração de inibição por substrato [g_S/L]
m_S	Coeficiente de energia de manutenção [$g_S/(g_X \cdot h)$]
o	*Output gate*
P	Concentração do etanol [g_P/L]
Q	Vazão volumétrica [L/h]
r_P	Taxa de produção de etanol [$g_P/(L \cdot h)$]
r_S	Taxa de consumo de substrato [$g_S/(L \cdot h)$]
r_X	Taxa de crecimento celular [$g_X/(L \cdot h)$]
S	Concentração de substrato [g_S/L]
V	Volume reacional [L]
X	Concentração celular [g_X/L]
$Y_{P/S}$	Coeficiente de rendimento do produto [g_P/g_S]
$Y_{X/S}$	Coeficiente de rendimento celular [g_X/g_S]

Sumário

1 Introdução

A biotecnologia é caracterizada como o emprego de organismos e de sistemas vivos para a obtenção de produtos de interesse em diversas áreas como alimentícia, farmacológica, ambiental, entre outras. Seu desenvolvimento acompanha a história humana, datando desde 6000 a.C. com a fermentação de bebidas alcoólicas até os dias atuais com avanços nas fronteiras do conhecimento tais como a terapia gênica (ENCYCLOPÆDIA BRITANNICA, 2019).

O uso continuado e priorizado da biotecnologia é explicado pelas possibilidades somente obtidas por esta via. Muitos produtos indispensáveis possuem estrutura complicada para ser sintetizada por via química tal como o fármaco imunossupressor Tacrolimo, obtido no caldo fermentativo de bactérias do gênero *Streptomyces*, e utilizado para evitar rejeição de órgãos transplantados (SINGH *et al.*, 2017).

No aspecto ambiental, o tratamento biológico de águas residuárias cresceu em expressividade frente aos tratamentos químico ou termo-oxidativos para remoção de compostos orgânicos solúveis devido à equidade em eficiência aliada aos menores custos em implantação do tratamento e operação. Além disso, a versatilidade do tratamento biológico permite seu emprego em uma ampla variedade de efluentes, reaproveitando a matéria orgânica economicamente na obtenção de subprodutos (MITTAL, 2011; CHRISTENSON; SIMS, 2012; OTHMAN *et al.*, 2013; VERVERIS *et al.*, 2016).

A crescente necessidade de obtenção de combustíveis renováveis aliada à redução do uso de combustíveis fósseis impulsionou a utilização dos biocombustíveis obtidos pela fermentação biológica. No Brasil, o programa ProÁlcool implementado pelo governo federal em 1975 impulsionou a produção de etanol por meio de fontes renováveis e insumos agroindustriais a fim de substituir e reduzir a dependência dos combustíveis fósseis (ANDRADE; CARVALHO; SOUZA, 2010). Para o ano de 2019, foram produzidos no país aproximadamente 25 Mm^3 de etanol hidratado para uso automotivo de acordo com dados divulgados pela (ANP, 2020).

Apesar da expressa empregabilidade da via biológica na produção de bens e serviços, a biotecnologia apresenta grande variabilidade nos processos devido à sensibilidade dos sistemas biológicos. Pequenas mudanças nas condições de trabalho tais como pH, agitação do meio, concentração de substrato, origem da fonte de carbono, nitrogênio e oxigênio, presença de produtos inibidores, entre outros fatores, modificam o comportamento celular e/ou enzimático e, consequentemente, o processo como um todo (GHALY; EL-TAWEEL, 1997; GHALY; KAMAL; CORREIA, 2005; OLIVEIRA *et al.*, 2017).

Assim, a modelagem fenomenológica dos sistemas bioquímicos apresenta diferenças quanto aos resultados experimentais, especialmente por muitos assumirem invariabilidade dos parâmetros cinéticos. Frente a esse impasse, a modelagem empírica se mostra como uma possível alternativa para a construção de um modelo matemático capaz de representar os processos biotecnológicos devido à facilidade em se obter dados experimentais de parâmetros operacionais

importantes do processo, muitas vezes já disponíveis no banco de dados das empresas.

Dentre as diferentes técnicas para obtenção de modelos empíricos, as redes neurais de aprendizagem profunda ou *deep learning* (RAP) ganharam notoriedade nas últimas décadas devido a sua grande capacidade de mapeamento de funções complexas a partir de dados reais, possibilitando a resolução de problemas antes inviáveis. No campo da biotecnologia, trabalhos demonstram sua aplicabilidade na obtenção de modelos para a determinação da produção de bioetanol tanto em regime permanente (AHMADIAN-MOGHADAM; ELEGADO; NAYVE, 2013; ESFAHANIAN *et al.*, 2013; MOHAMED *et al.*, 2013; GRAHOVAC *et al.*, 2016) quanto transiente (LINKO; ZHU, 1992; NAGY, 2007; BETIKU; TAIWO, 2015), sendo o uso de redes perceptron multicamadas (MLP) observado em quase totalidade dos trabalhos.

Apesar da grande empregabilidade, as redes MLP apresentam menor eficiência para o desenvolvimento de modelos em regime transiente devido à sua incapacidade de incorporar a dependência temporal das variáveis. Nesse paradigma, opta-se por redes neurais recorrentes (RNN) que permitem incorporar nas variáveis de entrada do modelo um padrão de recorrência dos dados preditos por meio dos chamados estados ocultos, incorporando memória ao modelo (DREYFUS, 2005; PATTERSON; GIBSON, 2017).

Entre as arquiteturas de redes recorrentes, uma das mais aplicadas é a denominada *Long Short-Term Memory* (LSTM), uma rede de aprendizagem profunda modular cujo treinamento é menos passível aos problemas frequentemente observados para as redes recorrentes convencionais. Em sua estrutura, além dos estados ocultos encontrados em RNN, existe a geração e a manutenção da chamada célula de memória que é capaz de lidar com problemas que apresentam dependência da informação a longo prazo, ampliando a capacidade de aprendizagem do modelo neural.

Dessa forma, pretende-se no presente trabalho investigar o desempenho de redes neurais recorrentes *Long Short-Term Memory* (RNN-LSTM) com diferentes topologias capazes de aprender padrões intricados de um grande volume de dados a fim de determinar modelos empíricos para um processo fermentativo contínuo em que os parâmetros cinéticos são variantes ao longo da operação.

2 Fundamentação Teórica e Revisão Bibliográfica

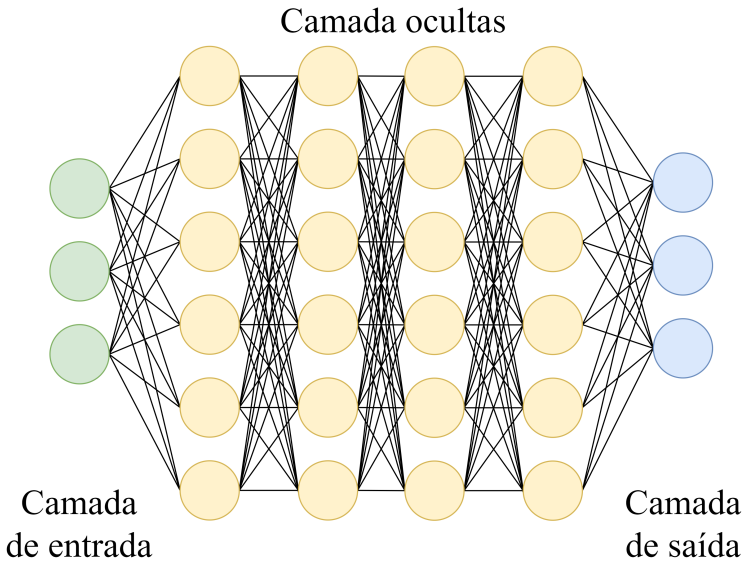
Para a melhor compreensão da escolha da técnica empregada neste trabalho e dos resultados obtidos, são apresentados os principais aspectos e conceitos referentes aos modelos de aprendizagem profunda. Os fundamentos das técnicas de *deep learning* são esclarecidos na Seção 2.1, com maior detalhamento da estrutura de redes de aprendizagem profunda recorrentes na Seção 2.2 e da topologia LSTM investigada neste trabalho na Seção 2.3.

2.1 Aprendizagem profunda

A aprendizagem profunda ou *deep learning* é caracterizada como uma ferramenta baseada em redes neurais artificiais (ANN, do inglês *Artificial Neural Networks*) capaz de obter representações com alto grau de abstração por meio de dados brutos sem a necessidade de intervenção de um operador humano. Sua principal diferença para as ANN tradicionais está na maior profundida das redes e, consequentemente, no maior grau de complexidade advindo do grande número de neurônios altamente interconectados em multicamadas, característica que garante a ampliação do mapeamento de funções não-lineares cada vez mais complexas (LECUN; BENGIO; HINTON, 2015; GOODFELLOW; BENGIO; COURVILLE, 2016; SHRESTHA; MAHMOOD, 2019).

Baseada na fisiologia do sistema neural de animais, a estrutura das redes de aprendizagem profunda é baseada na interconexão de diversos neurônios em diferentes camadas nomeadas entrada, oculta e saída. O número de camadas ocultas depende da rede projetada, sendo um hiperparâmetro definido pelo projetista, enquanto que as camada de entrada e de saída são únicas. As informações das variáveis de entrada do processo são definidas na camada de entrada, ao passo que a camada de saída provê a previsão das variáveis desejadas. As camadas ocultas são responsáveis por decifrar os padrões presentes no dados apresentados a fim de determinar o resultado final, seja uma classificação ou regressão de dados (LECUN; BENGIO; HINTON, 2015; PATTERSON; GIBSON, 2017). A Figura 1 representa simbolicamente a estrutura de uma rede de aprendizagem profunda em que cada linha conectando dois neurônios representa um peso sináptico w.

Figura 1 – Estrutura de uma rede profunda *feedforward*.



Segundo Aggarwal (2018), o aumento da profundidade da rede (*i.e.*, o número de camadas ocultas), além de ampliar a capacidade de aprendizagem e mapeamento de funções, representa uma estratégia de regularização, forçando as camadas posteriores a seguirem um padrão imposto pelas anteriores e, assim, um menor número de neurônios por camada é requerido para um adequado desempenho da RAP. Porém, essa estratégia apresenta ônus durante a etapa de treinamento da rede como será explicado posteriormente.

De acordo com Haykin (2009), cada neurônio representa uma unidade de processamento de informação da rede neural, recebendo as informações de todos os neurônios da camada anterior e transferindo sua informação para todos os neurônios da camada subsequente pelas sinapses ponderadas. Os valores ponderados são combinados linearmente com um parâmetro externo denominado viés ou *bias* referente ao neurônio, e o resultado é geralmente limitado em uma faixa específica por uma função de ativação não linear. A Equação 1 ilustra o cálculo feito por um neurônio j e explicado anteriormente

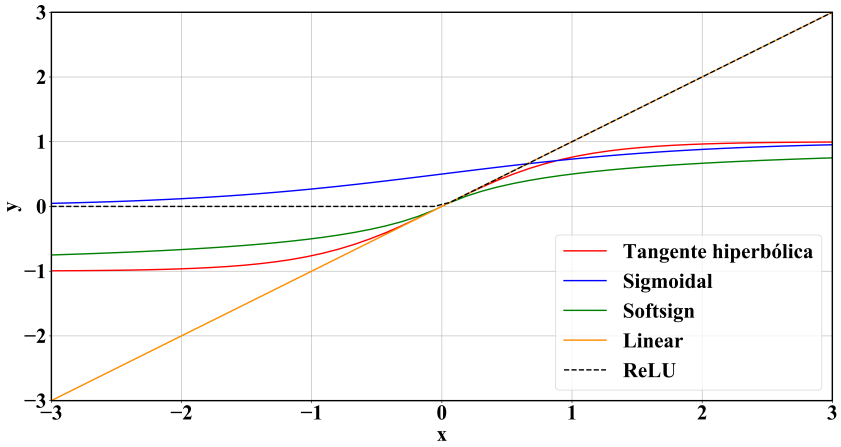
$$o_j = \Psi\left(\sum_{i=1}^{n} w_{i,j}x_{i,j} + b_j\right) \tag{1}$$

em que:

- o_j: saída ou resultado do neurônio j;
- $x_{i,j}$: variáveis de entrada da camada i;
- $w_{i,j}$: pesos sinápticos;
- b_j: viés do neurônio j;
- $\Psi(\cdot)$: Função de ativação.

As funções de ativação atuam como limitantes do sinal de saída dos neurônios e usualmente são funções não lineares, sendo a escolha desse hiperparâmetro indicativo da forma mais adequada de tratamento dos vetores de entrada da rede neural (HEATON, 2015). A Figura 2 ilustra o domínio das funções de ativação mais empregadas em RAP.

Figura 2 – Funções de ativação.



Em relação aos valores dos parâmetros (pesos e *bias*) de uma rede de aprendizagem profunda, existem diferentes formas de treinamento para determiná-los, tais como os treinamentos supervisionado, não-supervisionado, semi-supervisionado e aprendizagem por reforço. Cada um destes funciona de um modo diferente a fim de otimizar o desempenho da RAP e, considerando o objetivo final deste trabalho, somente o treinamento supervisionado será melhor detalhado na Seção 2.1.1.

2.1.1 Treinamento supervisionado

No treinamento supervisionado, a rede é inicializada com valores de pesos aleatórios ou seguindo algum tipo de distribuição estatística e é treinada com exemplos do tipo entrada-saída (sabe-se, *a priori*, o comportamento da função real para um determinado conjunto de variáveis de entrada). Para um dado vetor de entrada i, todos os valores dos nódulos da rede são calculados no sentido direto a fim de se obter o resultado predito pela rede (o). Este é comparado com a saída real para determinar o erro da predição e uma função custo $J(o)$.

O treinamento supervisionado consiste em um problema de otimização multivariável cujo objetivo é minimizar a função custo J por meio da modificação dos valores dos pesos sinápticos w. Essa etapa é usualmente realizada por algoritmos como gradiente descendente estocástico ou alguma adaptação deste que exige o conhecimento dos gradientes locais de $J(o)$ em relação a todos os pesos da RAP. Porém, entre um peso sináptico e a saída da rede existe diferentes caminhos P dentro de um conjunto Υ possível e, assim, a função $o(w)$ representa uma composição de diferentes funções ao longo de todos os caminhos possíveis. Desse modo, os

gradientes locais *z(i,j)* são obtidos por meio da aplicação da regra da cadeia como indicado na Equação 2 (HAYKIN, 2009; AGGARWAL, 2018).

$$\frac{\partial o}{\partial w} = \sum_{P \in \Upsilon} \prod_{(i,j) \in P} z(i,j) \tag{2}$$

Devido à estrutura interconectada das redes neurais, o número de caminhos entre um peso qualquer da rede e a saída cresce exponencialmente, dificultando o cálculo dos gradientes. Para tornar o cálculo possível, usa-se a chamada programação dinâmica que consiste em uma estratégia numérica de cálculo dos gradientes de modo recursivo, iniciando-o pelos neurônios diretamente relacionados à saída *o* para os quais o resultado é conhecido e aplicando a estratégia camada a camada, seguindo o sentido inverso da rede. O conjunto de operações feitas no sentido direto e inverso da rede é denominado de algoritmo de retro-propagação (em inglês, *backpropagation*), e consiste no ponto central para o treinamento supervisionado das redes, sendo sua aplicação possível devido ao caráter acíclico da RAP (AGGARWAL, 2018).

Apesar de ser uma solução para a otimização dos parâmetros de redes neurais, o algoritmo de retro-propagação apresenta reveses na estabilidade, especialmente para o caso de redes com grande profundidade, devido à estrutura que envolve sucessivas multiplicações de gradientes ao longo dos caminhos da RAP.

Os gradientes locais são matematicamente representados pelo produto entre a derivada da função de ativação e o peso e, como indicado na Equação 2, compõem o gradiente total. Caso as multiplicações sejam muito inferiores à unidade, os gradientes propagados tendem a desaparecer ao longo das camadas da rede neural e, dessa forma, os pesos das primeiras não são otimizados. Este problema conhecido como *vanishing grandients* é extremamente comum em RAP e, especialmente em redes recorrentes, impossibilitando a adequada modelagem de sistemas com dependências sequenciais da informação ou temporais a longo prazo. O extremo oposto, nomeado *exploding gradients*, surge devido ao aumento sucessivo do valor dos gradientes ao longo das camadas. De modo semelhante, esse problema impossibilita a otimização da RAP (MANDIC, 2001; GOODFELLOW; BENGIO; COURVILLE, 2016; AGGARWAL, 2018).

Uma solução prática para o problema de *exploding gradients* é o método de *gradient clipping* que consiste em reescalonar o gradiente quando seu valor se torna consideravelmente alto. Essa metodologia prática é considerada adequada e suficiente desde que a rede não seja muito complexa. Em relação ao problema de *vanishing gradient*, diferentes técnicas podem ser empregadas tais como mudança da função de ativação, inicialização inteligente dos valores de pesos e vieses e, a mais eficaz, alteração da arquitetura da rede (GOODFELLOW; BENGIO; COURVILLE, 2016).

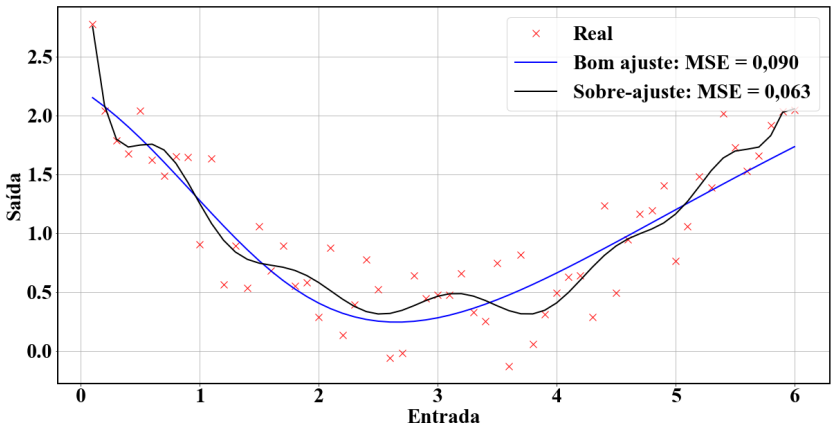
Durante a fase de treinamento, é necessário que informação suficiente seja apresentada à RAP a fim de se obter o conjunto de parâmetros mais adequado para o modelo. Entretanto, a otimização pode se tornar um processo computacionalmente custoso devido ao grande número de vetores de treinamento. Para reduzir o esforço computacional, é prática comum a aplicação da técnica de *mini batch* que consiste no emprego de um determinado número de vetores de entrada

para calcular a tendência dos gradientes locais para este conjunto, ou seja, um grupo de vetores compõem a função custo a ser minimizada. Usualmente são utilizados conjuntos múltiplos de 2, sendo 32 vetores de entrada um valor usualmente empregado na literatura com bons resultados (KESKAR *et al.*, 2017).

2.1.2 Sobre-ajuste em modelos

A imensa quantidade de hiperparâmetros possíveis aliada à presença de ruídos e erros no banco de dados podem gerar o problema de sobre-ajuste, ou seja, a obtenção de um modelo extremamente complexo que engloba tanto a informação da função que se pretende mapear quanto as incertezas supracitadas. Para os dados apresentados na fase de treinamento da rede, o modelo com sobre-ajuste prediz quase perfeitamente, porém, torna-se incapaz de generalizar, predizendo erroneamente para entradas que não foram apresentadas previamente e impossibilitando o emprego do modelo neural obtido (MARSLAND, 2015; BROWNLEE, 2019a). A Figura 3 ilustra o fenômeno supracitado.

Figura 3 – Sobre-ajuste em modelos neurais.



Uma forma simples de minimizar esse problema é a chamada parada antecipada. Nessa metodologia, parte do banco de dados do treinamento é separada para validar a generalização do modelo. Dessa forma, os dados de validação não são apresentados durante o treinamento da rede e, ao final de cada época, é calculada a função custo com estes vetores. Caso o comportamento da função seja crescente após um determinado número de épocas, implica-se que o modelo sofre sobre-ajuste e utilizam-se os parâmetros que tenham fornecido o menor custo para o conjunto de validação.

Outra forma de contornar o fenômeno de sobre-ajuste é o uso das regularizações que atuam como penalizações do valor da função custo empregada para minimização da fase de treinamento. O emprego das regularizações reduzem a complexidade do modelo ajustado, beneficiando a capacidade de generalização. As formas de regularização mais comuns são as chamadas

normas L_1 e L_2.

A norma L_1 penaliza a soma dos valores absolutos dos pesos da rede, gerando modelos de baixa complexidade e robustos a entradas atípicas. A tendência do modelo que emprega essa regularização é a obtenção de soluções esparsas ao anular os pesos de vários neurônios. Já a norma L_2 penaliza a soma do quadrado dos pesos, levando a valores pequenos dos pesos, porém, não nulos. Esta é oposta à norma L_1, ou seja, mais complexa, vulnerável a valores atípicos e não esparsa. Seu uso é mais recomendado quando todas as entradas influenciam a variável de saída e os pesos apresentam valores próximos. A prática mais comum dita o uso da combinação das duas regularizações para a fase de treinamento (KHANDELWAL, 2019).

De acordo com Patterson e Gibson (2017), a generalização de redes de aprendizagem profunda também pode ser elevada por meio da técnica de desligamento ou *dropout*. Esta consiste em ajustar a função de ativação de um percentual de neurônios aleatórios em zero a cada iteração do treinamento, de modo que esses não contribuam para a propagação do erro no sentido direto e do gradiente no inverso. A estratégia evita a coadaptação dos neurônios, melhorando a generalização do modelo neural. Ressalta-se que o uso do *dropout* deve ser acompanhado de outras técnicas de regularização como as já comentadas anteriormente.

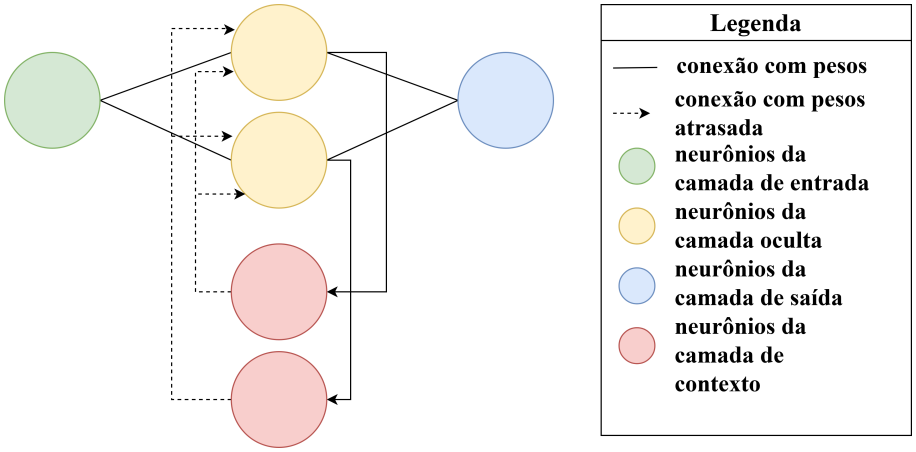
As estruturas das redes neurais e as técnicas de treinamento mencionadas anteriormente constituem a base de redes de *deep learning* com maior complexidade e capacidade de aprendizado, usualmente aplicadas para problemas mais sofisticados. Entre estas, destacam-se as redes neurais recorrentes.

2.2 Redes neurais recorrentes

As redes neurais recorrentes são amplamente empregadas em problemas que requerem o processamento de informação de modo sequencial ou que possuem dependência temporal pois suas camadas neuronais apresentam em sua estrutura um padrão de recorrência dos dados calculados por determinado neurônio. Além das informações transmitidas pela camada anterior, elas utilizam as saídas atrasadas dos neurônios da própria camada ponderadas pelos pesos sinápticos W_h dos chamados neurônios de contexto como variáveis de entrada (LECUN; BENGIO; HINTON, 2015; PATTERSON; GIBSON, 2017). Esse padrão de recorrência é ilustrado na Figura 4.

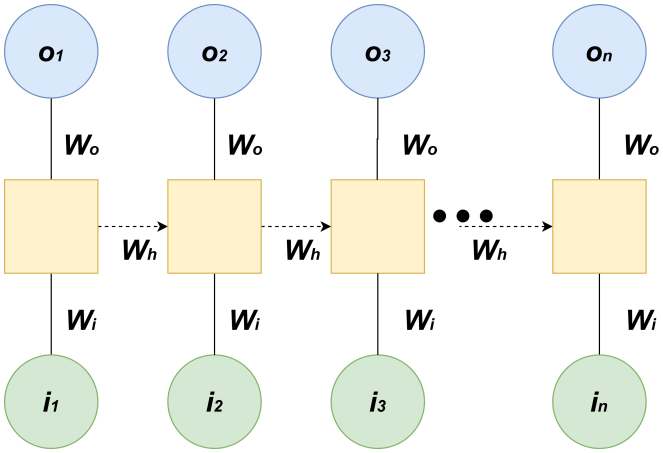
A presença dos neurônios de contexto são responsáveis por salvar as informações dos estados anteriores e gerar os denominados estados ocultos h. A depender da informação previamente processada pela rede neural recorrente, os resultados calculados para uma mesma entrada podem diferir consideravelmente (HEATON, 2015).

Figura 4 – Estrutura de redes recorrentes convencionais.



Quando observado o cálculo das RNN sequencialmente, nota-se a estrutura em cadeia apresentada na Figura 5. A rede é composta por unidades neurais iguais repetidas temporalmente em n diferentes instantes. Ressalta-se que foram condensadas as camadas oculta e de contexto na representação e os termos W indicam os pesos das camadas. Como visto na Figura 5, essa arquitetura de rede é adequadamente compatível com modelos com dependência temporal, como é o caso do modelo determinado na proposta deste trabalho, devido ao seu fluxo ininterrupto de informação entre as unidades neurais.

Figura 5 – Representação do cálculo sequencial em uma RNN.

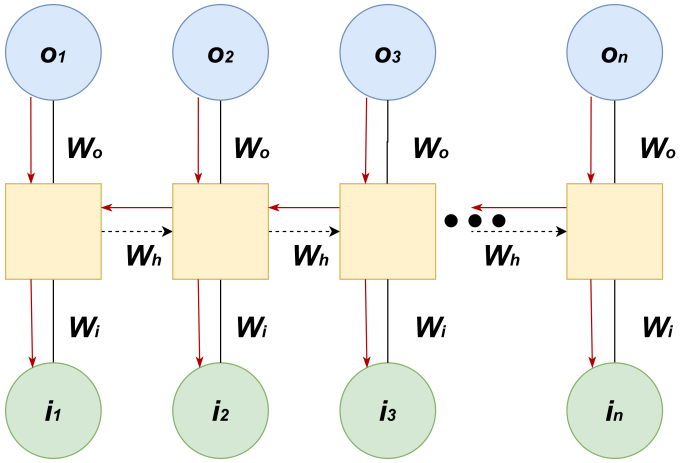


2.2.1 Treinamento de redes recorrentes

A determinação dos pesos e vieses de uma rede neural recorrente é feita por um treinamento supervisionado empregando uma extensão do algoritmo *backpropagation* conhecida como *backpropagation through time* ou BPTT. Na direção direta da RNN são determinados os erros entre os valores do modelo predito e os reais usados para treinar a rede para cada instante de tempo, e na inversa são propagados os gradientes determinados (NIELSEN, 2015; PATTERSON; GIBSON, 2017). Apesar do caráter cíclico das informações do estado oculto, a RNN

apresenta uma estrutura acíclica como mostra a Figura 5 e, assim, a aplicação da programação dinâmica é possível. O diferencial do algoritmo BPTT consiste na propagação dos gradientes não somente dentro das unidades das redes, mas também no espaço temporal, como ilustra a Figura 6.

Figura 6 – Treinamento de rede RNN empregando BPTT.



*As setas em vermelho indicam o sentido de propagação dos gradientes.

De acordo com Aggarwal (2018), apesar de os parâmetros das RNN serem compartilhados pelas unidades temporais (*i.e.*, os pesos e vieses são iguais em todas as unidades), o algoritmo BPTT os considera como independentes e distintos para as diferentes cópias da rede durante o treinamento. Os gradientes locais das cópias são, então, determinados de modo semelhante à retro-propagação e somados para determinar o gradiente local instantâneo para determinado nódulo a fim de gerar uma atualização unificada para cada parâmetro.

De acordo com Goodfellow, Bengio e Courville (2016), um dos grandes reveses durante o treinamento de RNN é representado pela difícil inclusão das dependências temporais ao longo prazo, sendo causado, primordialmente, pelo problema de *vanishing gradients*. A solução mais adequada consiste em alterar a topologia da rede recorrente. Entre as diferentes arquiteturas modificadas para uma rede neural profunda recorrente, destaca-se a denominada LSTM (do inglês *Long Short-Term Memory*).

2.3 Rede recorrente LSTM

A arquitetura de rede *Long Short-Term Memory* foi criada por Hochreiter e Schmidhuber (1997) com o intuito de solucionar os problemas de otimização encontrados em redes recorrentes. Seu diferencial durante a otimização consiste em forçar um erro constante por meio do chamado *constant error carousel* gerado pelos portões da estrutura, permitindo o aprendizado ao longo prazo.

A RNN-LSTM é caracterizada como uma rede neural modular, ou seja, é composta por

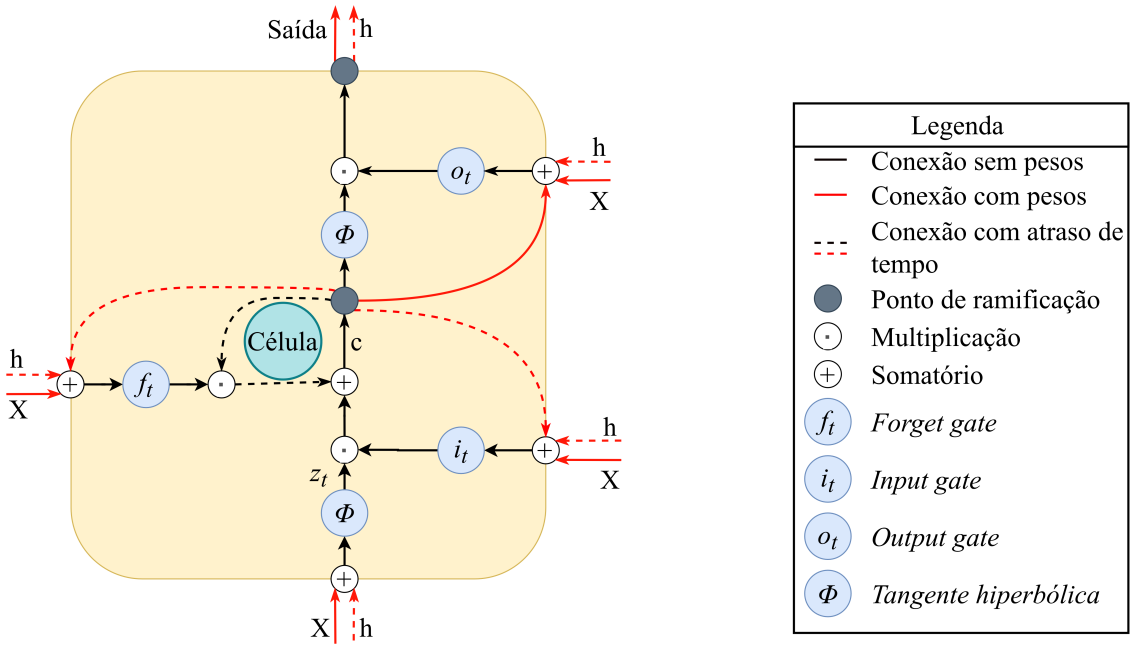
redes menores independentes e com atribuições distintas que atuam concomitantemente para se obter o resultado final. O diferencial da sua estrutura está na existência de uma célula de memória gerada pelas redes independentes e cujo estado é calculado por simples multiplicações de Hadamard (*i.e.*, ponto-a-ponto) e adições, sendo mantida ao longo do tempo. Dessa forma, a RNN-LSTM consegue preservar uma memória a longo prazo, distinguindo-a da RNN convencional (PATTERSON; GIBSON, 2017; SHRESTHA; MAHMOOD, 2019).

As informações empregadas no cálculo do estado da célula são advindas de um ciclo de recorrência interna capaz de modular o fluxo de dados por meio dos denominados portões (*gates*) *input, forget* e *output*. Cada um desses portões representa uma rede neural *feedforward* com função de ativação sigmoide que seletivamente permite a passagem de informação para a atualização do estado da célula (GOODFELLOW; BENGIO; COURVILLE, 2016; PATTERSON; GIBSON, 2017).

De acordo com Patterson e Gibson (2017), cada portão possui uma função específica durante o gerenciamento da informação. O *input gate* evita que dados irrelevantes sejam introduzidos nos cálculos, enquanto que o *forget gate* elimina informações anteriores da célula que não são mais importantes. Por fim, o *output gate* permite ou não que o conteúdo de memória da célula seja enviado para a camada de saída. Dessa forma, a arquitetura LSTM opera sempre em ciclos de exclusão, atualização e filtração de dados da célula de memória, evitando que as informações a longo prazo sejam ignoradas.

Os cálculos executados pela RNN-LSTM obedecem uma sequência específica. Para melhor compreensão, considere a estrutura ilustrada na Figura 7. Em uma RNN-LSTM, existem três fontes distintas de dados: duas externas representadas pelas variáveis de entrada X_t e estados ocultos de todas as unidades da rede LSTM h_t e uma interna constituída pela célula de memória c_t. As fontes externas são alimentadas nos quatro terminais da rede (*i.e.*, nos três portões e na entrada da rede) enquanto que a interna somente atua nos portões (ABDEL-NASSER; MAHMOUD, 2017; PATTERSON; GIBSON, 2017).

Figura 7 – Representação da célula de memória em uma RNN-LSTM.



Fonte: Adaptada de Patterson e Gibson (2017).

Semelhantemente aos portões, a entrada da RNN-LSTM é configurada como uma rede neural *feedforward* com único diferencial de ter a função de ativação do tipo tangente hiperbólica (Φ) responsável por ponderar a informação como indicado na Equação 3. Como mencionado anteriormente, nesse terminal somente são empregados os estados ocultos anteriores e as variáveis de entrada atuais.

$$z_t = \Phi\left(W_{x,z}X_t + W_{h,z}h_{t-1} + b_z\right) \tag{3}$$

Os portões *input* e *forget* são redes neurais *feedforward* com função de ativação sigmoide e calculados como mostram as Equações 4 e 5. Dessa forma, as saídas dos portões são limitadas no intervalo [0, 1] pela função de ativação, característica responsável pelo correto funcionamento da estrutura RNN-LSTM.

$$i_t = \sigma\left(W_{x,i}X_t + W_{h,i}h_{t-1} + W_{c,i}c_{t-1} + b_i\right) \tag{4}$$

$$f_t = \sigma\left(W_{x,f}X_t + W_{h,f}h_{t-1} + W_{c,f}c_{t-1} + b_f\right) \tag{5}$$

As restrições da função de ativação sigmoide representam a principal ferramenta para determinação da célula de memória. O cálculo envolve uma parcela de eliminação das informações da célula e outra de incremento representadas pelo primeiro e segundo termos da Equação 6, respectivamente. Caso a saída do *forget gate* seja unitária, toda a informação acumulada na célula de memória até o passo anterior é mantida. Para o caso de f_t ser nulo, toda a informação da célula é eliminada. Já para o *input gate*, o valor unitário representa que todas as novas informações são

importantes e devem ser utilizadas para a atualização da célula, enquanto que o valor nulo indica que nenhuma informação das novas variáveis de entrada possui relevância.

$$c_t = f_t \odot c_{t-1} + i_t \odot z_t \tag{6}$$

Após a atualização da célula de memória, esta é utilizada juntamente com as fontes externas para determinar a resposta da RNN-LSTM em duas etapas. Inicialmente o *output gate* determina quais informações são relevantes para a saída como indica a Equação 7 e, de modo semelhante aos portões anteriores, elimina as informações desnecessárias para a resposta na etapa seguinte representada pela Equação 8. Nessa etapa, a saída é ponderada pela função de ativação tangente hiperbólica, indicando sua importância para a resposta final da rede.

$$o_t = \sigma\left(W_{x,o}X_t + W_{h,o}h_{t-1} + W_{c,o}c_t + b_o\right) \tag{7}$$

$$h_t = o_t \odot \Phi(c_t) \tag{8}$$

Para a determinação dos parâmetros da RNN-LSTM, é empregado o treinamento supervisionado com algoritmo *backpropagation through time* elucidado anteriormente. Entretanto, o problema de *vanishing gradient* não acomete essa topologia devido ao fluxo ininterrupto do gradiente entre as células de memória mantidas separadas que envolve somente multiplicações ponto a ponto e adições, diferentemente das RNN convencionais que utilizam multiplicações matriciais durante a fase de treinamento.

2.4 Revisão da literatura

A eficácia da arquitetura LSTM tem sido notoriamente reconhecida no meio científico, especialmente na última década, sendo aplicada em problemas de natureza diversa. Qing e Niu (2018) estudaram a aplicação de RNN-LSTM na determinação do comportamento temporal da irradiação solar com base em dados meteorológicos (temperatura, ponto de orvalho, velocidade de vento, umidade, visibilidade e classificação climática). O banco de dados foi obtido em base de horas durante um período de 30 meses e empregado para treinar e validar tanto a RNN-LSTM quanto redes *feedforward* multicamadas e um modelo de regressão linear multivariável por meio de mínimos quadrados. Os resultados mostraram que a rede RNN-LSTM apresentou maior generalização dos dados, obtendo uma redução significativa da raiz da média dos erros quadráticos em relação aos outros algoritmos propostos.

No trabalho de Abdel-Nasser e Mahmoud (2017), diferentes modelos para a previsão, de hora em hora, da geração de energia fotovoltaica foram desenvolvidos a partir de diferentes arquiteturas de RNN-LSTM. Os autores empregaram a arquitetura básica da rede LSTM, além das técnicas de janela deslizante, de passos temporais e de memória entre *batches*. Também foi desenvolvida uma rede baseada na sobreposição de unidades LSTM com memória entre

batches. Os cinco modelos gerados foram treinados com tamanho de *batch* igual a 1, números de épocas 20, 50 e 100 e somente entradas endógenas. Entre os modelos criados, a RNN-LSTM para regressão com passos temporais empregando 50 épocas apresentou o melhor desempenho com base no critério RMSE. Os autores também verificaram o desempenho de metodologias citadas na literatura como regressão linear múltipla e redes neurais, porém, o modelo baseado em LSTM apresentou melhor desempenho devido a sua estrutura de recorrência e memória interna.

O desempenho da topologia LSTM para predição da velocidade de automóveis em autoestradas foi estudado por Ma *et al.* (2015). Além da rede baseada em LSTM, foram obtidos modelos com redes neurais do tipo Elman, com atraso de tempo e não-linear autorregressiva com entradas exógenas, além de máquina de vetores de suporte, ARIMA e filtro de Kalman, algoritmos que são usualmente empregados para predição temporal. Os autores analisaram modelos utilizando número variável de intervalos de tempo, excetuando as RNN-LSTM, e o benefício de empregar uma entrada exógena (volume de carros). Os resultados indicaram que as redes LSTM superaram bastante o desempenho obtido pelos outros modelos, com uma melhoria mínima de 28% do valor do MSE para caso mais próximo. Além disso, os autores ressaltam que a estrutura LSTM automaticamente ajusta o intervalo de tempo ótimo, sendo um ponto benéfico durante o treinamento da rede em comparação à dificuldade em ajustar os parâmetros dos outros algoritmos.

No trabalho desenvolvido por Wöllmer *et al.* (2010), o problema de reconhecimento de emoções humanas foi explorado. Os autores empregaram uma RNN-LSTM bidirecional para a obtenção de um modelo para a classificação de três classes (raiva, neutralidade/tristeza e alegria) baseado em expressões audiovisuais. A informação visual adquirida por meio de marcadores faciais e normalizada para minimização das características individuais foi utilizada para a obtenção de uma representação de vetores de entrada com baixa dimensionalidade por meio da análise de característica principal, enquanto que as entradas da informação de áudio foram extraídas das características da onda acústica. O desempenho do modelo LSTM bidirecional foi comparado a técnicas de cunho estatístico como o modelo oculto de Markov e as máquinas de vetores de suporte, além de redes LSTM unidirecional. Os resultados indicam que ambas as redes com topologia LSTM superaram o desempenho dos modelos estatísticos na classificação de três estados emocionais, com melhoria mínima de 6,5%. Em relação às RNN-LSTM, o modelo bidirecional obteve resultados ligeiramente melhores, indicando que o emprego do modelo concatenado mapeia melhor as informações apresentadas na fase de treinamento.

Zhao, Sun e Jin (2018) propuseram o emprego de redes neurais LSTM com o incremento da técnica de reparametrização conhecida como *batch normalization* para o diagnóstico de falhas sequenciais no *benchmark Tennessee Eastman Process* ou TEP. No estudo, o desempenho da rede proposta foi comparado aos obtidos pelas técnicas de análise dinâmica das componentes principais acoplada à máquina de vetores de suporte (DPCA+SVM), análise discriminante linear dinâmica também aliada à máquina de vetores de suporte (DLDA+SVM) e perceptron multicamadas (MLP). Diferentes casos foram analisados, variando-se as falhas do sistema causadas por até 21 variáveis distintas. Em todos os casos analisados, a eficiência da classificação da

falha aplicando a rede LSTM com *batch normalization* foi superior às obtidas pelas outras técnicas. Além do melhor desempenho, a rede proposta possui uma aprendizagem dinâmica da informação por meio do sistema de portões da RNN-LSTM, e o emprego da reparametrização *batch normalization* auxiliou na estabilização da rede, aumentando a eficiência de correlação e a velocidade de convergência em comparação a uma rede LSTM sem o emprego da técnica e com o mesmo conjunto de hiperparâmetros.

Althelaya, El-Alfy e Mohammed (2018) desenvolveram modelos de predição dos valores de fechamento das ações de empresas empregando como entradas os valores diários de volume monetário, fechamento, abertura, menor e maior das ações em uma janela de 10 dias. Foram verificadas as RNN-LSTM, assim como redes *gated recurrent unit* (GRU) e MLP. Para as rede LSTM e GRU, foram empregados modelos tanto uni quanto bidirecionais. Os autores observaram bons resultados das redes, com destaque para as RNN-LSTMs que obtiveram desempenhos melhores em comparação às outras redes, tanto no modelo uni quanto bidirecional, sendo a última a melhor rede desenvolvida no estudo.

Apesar de os trabalhos expostos anteriormente não ilustrarem a aplicação das redes LSTM em engenharia de processos, esses demonstram a capacidade que as redes recorrentes de aprendizagem profunda com topologia *Long Short-Term Memory* apresentam como aproximadores universais de funções de natureza complexa e de difícil determinação do modelo determinístico, sendo indicativo da potencialidade para aplicação na modelagem empírica do processo de fermentação alcoólica.

3 Hipótese

Apesar da complexidade envolvida no processo fermentativo e da variabilidade dos parâmetros cinéticos que o caracterizam, as dinâmicas das variáveis de processo podem ser efetivamente preditas por meio do conhecimento dos parâmetros operacionais macroscópicos ao aplicar modelos baseados em redes de aprendizagem profunda com topologia *Long Short-Term Memory*.

3.1 Objetivos

- Desenvolver um modelo matemático para descrever o comportamento dinâmico do processo fermentativo;
- Obter modelo empírico baseado em ANN para parâmetros cinéticos variáveis;
- Gerar banco de dados que caracterizem a dependência das variáveis de processo em relação às variáveis operacionais;
- Desenvolver a rotina de treinamento de RNN-LSTM em linguagem Python;
- Treinar e validar os modelos;
- Comparar os desempenhos das redes desenvolvidas com base nos critérios quantitativos de desempenho MSE e teste de regressão;
- Analisar a capacidade das redes trabalharem com retroalimentação dos dados preditos;
- Desenvolver uma interface gráfica de utilizador para rápida simulação do comportamento dinâmico do processo empregando RNN-LSTM em modo *offline*.

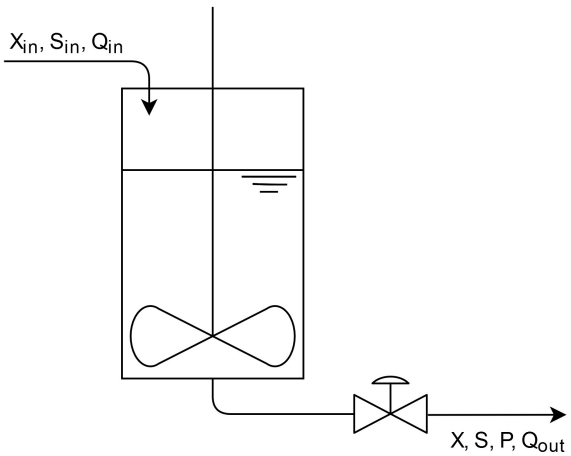
4 Materiais e Métodos

Para o desenvolvimento do presente trabalho, dividiu-se a metodologia em quatro seções distintas. Na Seção 4.1, apresenta-se o modelo matemático utilizado para simular o comportamento dinâmico de um biorreator contínuo empregado no processo fermentativo de produção de etanol. A Seção 4.2 elucida a criação e a escolha de modelos empíricos baseados em redes neurais artificiais para a determinação dos valores dos parâmetros cinéticos variantes do sistema. Na Seção 4.3, apresenta-se a estratégia para a obtenção do comportamento dinâmico do sistema e, consequentemente, do banco de dados utilizado para o treinamento das redes recorrentes com topologia LSTM. Por fim, a Seção 4.4 explora a determinação dos modelos empíricos baseados em aprendizagem profunda utilizados para descrever o comportamento dinâmico do sistema estudado e a metodologia de avaliação de desempenho tanto para o modo de operação *online* quanto *offline*.

4.1 Modelo matemático para o processo fermentativo

O modelo matemático empregado para descrever o processo de fermentação alcoólica é baseado no trabalho de Ghaly e El-Taweel (1997). Os autores exploram a via fermentativa para a obtenção de etanol utilizando o soro de leite como substrato, subproduto da produção de queijos formado em grande volume. A depender da matéria-prima, cerca de 60 a 90% (m/m) é convertido em soro com até 70% dos açúcares iniciais. Devido à alta demanda biológica de O_2, o soro de leite é altamente poluente, chegando a ser 100 vezes mais nocivo ambientalmente que o esgoto doméstico. Assim, empregá-lo na produção de etanol é proveitoso economicamente e reduz seu impacto ambiental (IRKIN, 2019; LOPES *et al.*, 2019).

Figura 8 – Esquema do biorreator de fermentação contínua.



Considere o esquema do biorreator ilustrado na Figura 8 em que:

- X: concentração de células;
- S: concentração de substrato;
- P: concentração de etanol/produto;
- Q: vazão volumétrica.

Para simplificar o modelo fenomenológico, foram empregadas as hipóteses de mistura perfeita, massa específica do fluido constante assim como a área transversal do biorreator.

Aplicando o princípio de conservação da matéria, o balanço de massa global é descrito como

$$\begin{bmatrix} \text{Taxa de} \\ \text{acúmulo de} \\ \text{matéria} \end{bmatrix} = \begin{bmatrix} \text{Entrada de} \\ \text{matéria por} \\ \text{convecção} \end{bmatrix} - \begin{bmatrix} \text{Saída de} \\ \text{matéria por} \\ \text{convecção} \end{bmatrix}$$

ou matematicamente como expressa a Equação 9. Usando a hipótese de massa específica constante, a Equação 9 é reduzida como indica a Equação 10.

$$\frac{dm}{dt} = \frac{d}{dt}(\rho V) = W_{in} - W_{out} \tag{9}$$

$$\frac{dV}{dt} = \frac{W_{in} - W_{out}}{\rho} = Q_{in} - Q_{out} \tag{10}$$

Optou-se por empregar como força motriz da vazão de saída a pressão estática da coluna de líquido no biorreator (Equação 11). Os termos C_V, g e A_t representam o produto do coeficiente de descarga e área do orifício de saída, a aceleração gravitacional e a área transversal do biorreator, respectivamente.

$$Q_{out} = C_v\sqrt{2gh} = C_v\sqrt{\frac{2gV}{A_t}} \tag{11}$$

O balanço material para a massa de células presentes no biorreator é expresso como

$$\begin{bmatrix} \text{Taxa de} \\ \text{acúmulo de} \\ \text{células} \end{bmatrix} = \begin{bmatrix} \text{Entrada de} \\ \text{células por} \\ \text{convecção} \end{bmatrix} - \begin{bmatrix} \text{Saída de} \\ \text{células por} \\ \text{convecção} \end{bmatrix} + \begin{bmatrix} \text{Taxa de} \\ \text{crescimento} \\ \text{celular} \end{bmatrix} - \begin{bmatrix} \text{Taxa de} \\ \text{mortalidade} \\ \text{celular} \end{bmatrix}$$

e matematicamente escrito como mostra a Equação 12.

$$\frac{d}{dt}(XV) = Q_{in}X_{in} - Q_{out}X + r_xV - r_dV \tag{12}$$

No modelo desenvolvido, tanto o volume quanto a concentração de células são funções do tempo. Assim, aplicando-se a regra da cadeia no termo a esquerda da igualdade da Equação 12 e substituindo o resultado expresso em 10, obtém-se

$$\frac{d}{dt}(XV) = V\frac{dX}{dt} + X\frac{dV}{dt} = V\frac{dX}{dt} + X(Q_{in} - Q_{out}) \tag{13}$$

Substituindo a expressão 13 em 12 e reorganizando, o balanço final para a concentração celular é determinado pela Equação 14.

$$\frac{dX}{dt} = \frac{Q_{in}}{V}(X_{in} - X) + r_x - r_d \tag{14}$$

A taxa de crescimento celular r_X representa o aumento do número de células viáveis presentes no meio, sendo matematicamente calculada como o produto da taxa específica de crescimento das células μ e da concentração celular como expressa na Equação 15.

$$r_x = \mu X \tag{15}$$

Diferentes formas de taxas específicas de crescimento podem ser encontradas na literatura. Para o presente trabalho, optou-se pelo uso da equação de Monod modificada para contemplar a inibição do desenvolvimento celular pela presença do produto fermentativo e pelo efeito da pressão osmótica causado pelo excesso de substrato como indicado na Equação 16

$$\mu = \mu_{max}\frac{S}{K_S + S}\frac{K_P}{K_P + P}\frac{K_S^{'}}{K_S^{'} + S} \tag{16}$$

em que:

- μ_{max}: velocidade máxima de crescimento celular;
- K_S: constante de saturação;
- K_P: concentração de inibição por etanol;
- $K_S^{'}$: concentração de inibição por substrato.

De acordo com Taherzadeh e Karimi (2011), o principal fator para a inibição por etanol é a redução da atividade da água responsável por causar estresse nas células e possível desnaturação das enzimas glicolíticas. Segundo os autores, uma solução com 20% m/v de etanol é responsável por reduzir a atividade para 0,895, valor inferior à faixa ótima de atividade que está entre 0,975 e 0,999 para a espécie empregada usualmente na indústria *Saccharomyces cerevisiae*.

Além disso, o etanol presente no meio consegue reduzir o pH intracelular e romper ligações de pontes de hidrogênio da estrutura celular, contribuindo para a redução da atividade celular e, consequentemente, do bio-produto.

Por fim, a cinética de morte microbiológica escolhida para esse modelo é dada pela Equação 17, na qual K_d representa a taxa específica de morte celular.

$$r_d = K_d X \tag{17}$$

Similarmente ao balanço de células, desenvolveu-se um balanço de conservação do produto levando em consideração que não há alimentação desta espécie no biorreator e expresso como

$$\begin{bmatrix} \text{Taxa de} \\ \text{acúmulo do} \\ \text{produto} \end{bmatrix} = - \begin{bmatrix} \text{Saída do} \\ \text{produto por} \\ \text{convecção} \end{bmatrix} + \begin{bmatrix} \text{Taxa de produção} \\ \text{do produto por} \\ \text{atividade celular} \end{bmatrix}$$

Em termos matemáticos, o balanço é calculado como indica a Equação 18.

$$\frac{d}{dt}(PV) = V\frac{dP}{dt} + P\frac{dV}{dt} = -Q_{out}P + r_P V$$

$$\frac{dP}{dt} = -\frac{Q_{in}}{V}P + r_P \tag{18}$$

A expressão da cinética de produção de etanol foi definida como indica a equação 19

$$r_P = \alpha r_X + \beta X \tag{19}$$

em que:

- α: coeficiente energético para crescimento;
- β: coeficiente energético para manutenção.

Os coeficientes α e β são determinados com base nos valores dos coeficientes de rendimento de células $Y_{X/S}$, de rendimento de produto $Y_{P/S}$ e de manutenção celular m_S e calculados pelas Equações 20 e 21, respectivamente.

$$\alpha = \frac{Y_{P/S}}{Y_{X/S}} \tag{20}$$

$$\beta = m_S Y_{P/S} \tag{21}$$

O desenvolvimento do balanço material para o substrato é semelhante aos desenvolvidos anteriormente e determinado como

$$\begin{bmatrix} \text{Taxa de} \\ \text{acúmulo do} \\ \text{substrato} \end{bmatrix} = \begin{bmatrix} \text{Entrada do} \\ \text{substrato por} \\ \text{convecção} \end{bmatrix} - \begin{bmatrix} \text{Saída do} \\ \text{substrato por} \\ \text{convecção} \end{bmatrix} - \begin{bmatrix} \text{Taxa de consumo} \\ \text{do substrato por} \\ \text{atividade celular} \end{bmatrix}$$

O balanço de conservação do substrato é expresso matematicamente como indicado na Equação 22.

$$\frac{d}{dt}(SV) = V\frac{dS}{dt} + S\frac{dV}{dt} = Q_{in}S_{in} - Q_{out}S - r_S V$$

$$\frac{dS}{dt} = \frac{Q_{in}}{V}(S_{in} - S) - r_S \tag{22}$$

A taxa de consumo do substrato r_S é composta por três termos diferentes que contemplam a formação de células novas, a manutenção celular e a geração do produto fermentativo e é calculada como expressa pela Equação 23. Os três termos a direita da igualdade representam, respectivamente, os fenômenos previamente mencionados.

$$r_S = \frac{r_X}{Y_{X/S}} + m_S X + \frac{r_P}{Y_{P/S}} \tag{23}$$

4.2 Modelo neural para os parâmetros cinéticos

Para o modelo desenvolvido na Seção 4.1, os comportamentos dos parâmetros cinéticos $Y_{X/S}$, $Y_{P/S}$ e m_S são funções do tempo de detenção hidráulico D - razão entre o volume reacional e a vazão volumétrica de saída - e da concentração de substrato da entrada. Para calcular esses valores, foram desenvolvidos modelos empíricos baseados em redes neurais artificiais do tipo *feedforward*. Os dados empregados para o treinamento foram obtidos em Ghaly e El-Taweel (1997) e estão sumarizados na Tabela 1. Observa-se que os valores dos parâmetros cinéticos variam até 157% entre os valores mínimo e máximo, indicando grande sensibilidade do processo frente às variações das condições operacionais.

Tabela 1 – Parâmetros cinéticos experimentais.

S_{in} [g_S/L]	D [h]	m_S [g_S/($g_X \cdot h$)]	$Y_{X/S}$ [g_X/g_S]	$Y_{P/S}$ [g_P/g_S]
50	18	3,50	0,0280	0,252
50	24	3,67	0,0480	0,264
50	30	4,32	0,0590	0,311
50	36	4,68	0,0720	0,337
50	42	5,13	0,0720	0,369
100	18	6,31	0,0450	0,454
100	24	6,44	0,0520	0,464
100	30	6,47	0,0490	0,466
100	36	6,51	0,0510	0,469
100	42	6,59	0,0510	0,467
150	18	6,10	0,0425	0,439
150	24	6,42	0,0468	0,462
150	30	6,46	0,0465	0,465
150	36	6,49	0,0456	0,467
150	42	6,54	0,0439	0,471

Fonte: Adaptada de Ghaly e El-Taweel (1997).

Devido ao pequeno volume de vetores do banco de dados, optou-se por empregar a técnica de validação cruzada *k-fold* para a avaliação do modelo gerado por um determinado conjunto de hiperparâmetros. Para o treinamento, foi utilizado um *k-fold* igual a 5, valor típico encontrado na literatura (KUHN, 2013). Assim, o banco de dados foi dividido em 5 subconjuntos

igualmente espaçados e com dados aleatoriamente selecionados do banco original, e 5 modelos neurais foram determinados para cada conjunto de hiperparâmetros. Para cada modelo, k-2 subconjuntos foram empregados para o treinamento, um utilizado como conjunto de validação para evitar o sobre-ajuste do modelo por meio da técnica de parada antecipada e um para testar quantitativamente o desempenho da ANN. O resultado final é definido como a média dos k modelos determinados acrescido do desvio padrão da média. Após a seleção do conjunto de hiperparâmetros que melhor ajustam os dados experimentais, uma nova rede é treinada com o banco de dados em completude.

Devido à diferença de magnitude das variáveis de saída, optou-se por escaloná-las dentro do intervalo de 0 a 1 de modo a evitar uma maior influência da variável com maior ordem de grandeza durante a fase de treinamento. De modo semelhante, as variáveis de entrada foram escalonadas no mesmo intervalo. A função custo definida para minimização e treinamento das redes foi o erro médio quadrático e, a fim de evitar o sobre-ajuste dos dados, optou-se por implementar as regularizações L_1 e L_2 na função custo expressa na Equação 24

$$J = \frac{1}{k \cdot n} \sum_{j=1}^{k} \sum_{i=1}^{n} \left(Y_{j,i} - \hat{Y}_{j,i} \right)^2 + \gamma_1 \sum_{m=1}^{p} |w_m| + \gamma_2 \sum_{m=1}^{p} w_m^2 \tag{24}$$

em que:

- $Y_{j,i}$: variáveis de saída reais;
- $\hat{Y}_{j,i}$: variáveis de saída preditas pelas redes;
- w_m: pesos sinápticos;
- γ_1: peso da regularização L_1;
- γ_2: peso da regularização L_2;
- k: número de variáveis de saída;
- n: número total de vetores de treinamento;
- p: número total de pesos da ANN.

A primeira parcela a direita da igualdade na Equação 24 representa o MSE (do inglês *mean squared error*), enquanto os termos seguintes representam as regularizações L1 e L2, respectivamente.

Para a etapa de treinamento, foi definido um número de épocas máximo igual a 200, parada antecipada caso o desempenho calculado para o conjunto de validação não melhore após 10 atualizações sucessivas dos pesos sinápticos, delta mínimo de melhoria igual a 10^{-5} e algoritmo de otimização tipo Adam. A Tabela 2 sumariza todos os hiperparâmetros verificados nessa etapa.

Tabela 2 – hiperparâmetros analisados.

Hiperparâmetro	Valor
Camadas ocultas	1 e 2
Neurônios	1, 2, 3, 4, 5, 6, 7, 8, 9, 10 , 11, 12, 13, 14 e 15
Inicialização dos pesos	Aleatória uniforme, aleatória normal, identidade e LeCun uniforme
Função de ativação	Tangente hiperbólica, sigmoidal, linear, softsign e ReLU
γ_1	0, 10^{-3}, 10^{-2}, 10^{-1} e 1
γ_2	0, 10^{-3}, 10^{-2}, 10^{-1} e 1
Dropout	0; 0,05; 0,10; 0,15 e 0,20

Após a seleção do modelo neural final, seu desempenho também foi avaliado com base no teste de regressão que representa uma regressão linear entre as variáveis de saída reais e preditas pelo modelo. Idealmente, os valores reais e preditos são iguais e, assim, o coeficiente angular da reta determinada tem valor unitário e o linear é nulo. Também foi calculado o coeficiente de determinação R^2 como indicado na Equação 25. Este parâmetro representa o grau de correlação linear entre os dados, sendo desejável valores próximos à unidade.

$$R^2 = 1 - \frac{\sum_{i=1}^{n} \left(Y_i - \hat{Y}_i\right)}{\sum_{i=1}^{n} \left(Y_i - \overline{Y}_i\right)^2} \tag{25}$$

4.3 Resolução do modelo matemático para determinação do comportamento dinâmico do processo fermentativo

A resolução conjunta do sistema de equações diferenciais representado pelas Equações 10, 14, 18 e 22 juntamente às demais equações constitutivas apresentadas na Seção 4.1 descrevem o comportamento dinâmico do processo fermentativo. O modelo desenvolvido foi implementado em linguagem Python versão 3.7.7, e simulações foram feitas a fim de determinar o comportamento das variáveis de processo V, X, P e S frente a perturbações nas variáveis operacionais X_{in}, S_{in} e Q_{in}. Também foi incluído nessa etapa o modelo neural desenvolvido como descrito na Seção 4.2 para a modificação dos valores dos parâmetros m_S, $Y_{X/S}$ e $Y_{P/S}$ de modo a reproduzir a natureza variante destes ao longo da operação.

O regime permanente inicial do processo foi determinado resolvendo o problema estacionário gerado pelas Equações 10, 14, 18 e 22 e utilizando a ferramenta *fsolve* da biblioteca *scipy*. A Tabela 3 sumariza os valores dos parâmetros operacionais e cinéticos empregados no modelo e os iniciais das variáveis de entrada do sistema.

Tabela 3 – Parâmetros operacionais e cinéticos empregados na simulação.

Parâmetro	Valor	Unidade
At	100	cm^2
Cv	$1{,}01 \cdot 10^{-4}$	cm^2
Q_{in}	0,1086	L/h
X_{in}	0,055	g/L
S_{in}	50	g/L
μ_{max}	0,051	h^{-1}
K_d	0,005	h^{-1}
K_S	1,9	g/L
K_P	20,65	g/L
K_S'	112,51	g/L

Fonte: Adaptada de Ghaly e El-Taweel (1997).

Por fim, os comportamentos dinâmicos das variáveis controladas frente às variações das variáveis operacionais foram determinados para um conjunto de valores pré-definidos. Somente uma variável manipulada era modificada por vez em intervalos espaçados de 200 h, excetuando no início do processo.

A simulação foi desenvolvida em linguagem Python utilizando a função *odeint* do pacote *scipy*. A escolha do *solver* foi baseada no fato de que este emprega o algoritmo *lsoda* que monitora os dados calculados e alterna entre os métodos Adams e BDF caso o sistema de equações diferenciais apresente rigidez, possivelmente encontrado para o modelo fermentativo devido à grande disparidade das ordens de grandeza das variáveis envolvidas.

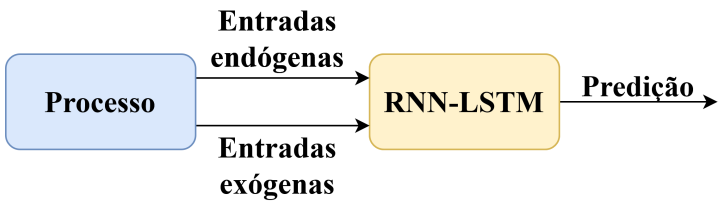
A taxa de aquisição dos dados foi definida em 3 h devido às características dinâmicas das variáveis de processo observadas em testes preliminares. Além disso, foi acrescido ruído ao resultado das simulações para simular a aquisição de dados reais de sensores. As amplitudes dos ruídos foram definidas como $\pm$0,5%, $\pm$1,5%, $\pm$1,5% e $\pm$1,0% dos *spans* dos sensores de volume, de concentração de substrato, de concentração de etanol e do contador de células, respectivamente. As faixas de precisão foram selecionadas de acordo com os valores encontrados em catálogos de diferentes equipamentos disponíveis comercialmente.

4.4 Modelos empíricos

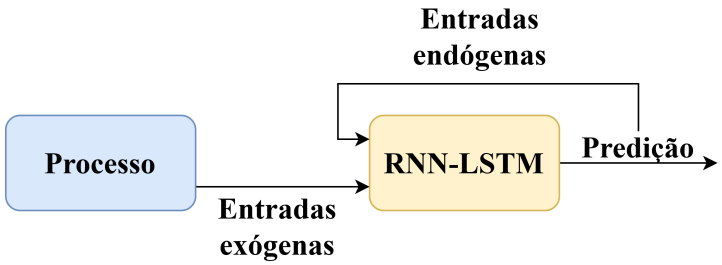
Neste tópico, será feita a distinção dos dois modos de operação a que foram submetidos os modelos baseados em aprendizagem profunda. Para tal, é necessário fazer a distinção entre variáveis de entrada endógenas e exógenas. São ditas endógenas as variáveis que são ao mesmo tempo entrada e saída do modelo, ou seja, os valores anteriores da sequência ou série temporal são empregados para determinar os seguintes. Já as variáveis exógenas são aquelas que atuam somente como entrada do modelo de predição.

Foi denominado modo *online* aquele em que todas as entradas endógenas do modelo neural são os valores reais obtidos por meio da simulação apresentada na Seção 4.3. Já o modo *offline* é dito aquele em que as entradas endógenas são os valores calculados pelo próprio modelo neural, excetuando-se o primeiro exemplo de entrada. A Figura 9 apresenta sucintamente a rotina de cálculos empregada nos dois modos de operação. A obtenção e a avaliação dos desempenhos destes são explanadas nas subseções seguintes.

Figura 9 – Modos de operação para predição empregando as RNN-LSTM.



(a) Modo *online*



(b) Modo *offline*

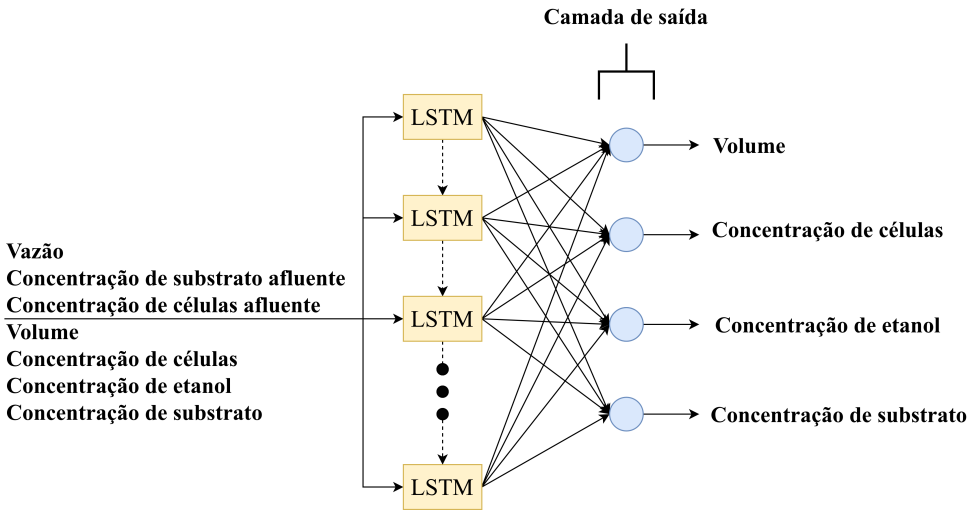
4.4.1 Modelo *online*

Os modelos baseados em redes de aprendizagem profunda com topologia LSTM foram desenvolvidos em linguagem Python versão 3.7.7, empregando as bibliotecas especializadas Tensorflow e Keras nas versões 1.14.0 e 2.3.1, respectivamente. A primeira representa uma biblioteca desenvolvida para avaliação matemática simbólica, especialmente de problemas com matrizes multidimensionais, atingindo uma rápida avaliação de expressões matemáticas complexas calculadas repetidamente. Sua capacidade de reduzir consideravelmente o tempo de resposta a torna recomendável para problemas em que o tempo de avaliação do modelo é uma variável importante (TENSORFLOW, n.d.).

A biblioteca Keras consiste em uma interface de programação de aplicativo voltada para a construção de redes de aprendizagem profunda. Esta é capaz de utilizar as funcionalidades de outras bibliotecas como a Tensorflow e permite rápida elaboração de redes profundas, criando-as e definindo todos os seus hiperparâmetros camada a camada (KERAS, n.d.).

A Figura 10 ilustra o diagrama simplificado da LSTM-RNN desenvolvida neste trabalho com suas respectivas variáveis de entrada e saída. Ressalta-se que cada unidade LSTM representada indica um passo temporal do modelo, e a dimensionalidade do vetor de saída dessa

Figura 10 – Diagrama esquemático da RNN-LSTM.



é determinada pelo número de neurônios da camada de saída dos portões.

O banco de dados obtido na etapa anterior foi dividido nos denominados conjuntos de treinamento e de teste empregados para otimizar os parâmetros das redes recorrentes e para validar os modelos, respectivamente. Dada a natureza dinâmica do sistema, os vetores foram separados de modo sequencial, reservando 80% para a etapa de treinamento e 20% para a de teste. Ressalva-se que 20% dos vetores de treinamento foram retidos no chamado conjunto de validação para a verificação da habilidade de generalização dos modelos durante a otimização dos parâmetros e evitar o sobre-ajuste por meio da técnica de parada antecipada. Caso a função custo do conjunto de validação não obtivesse melhorias após 10 atualizações consecutivas, o treinamento era cessado, e o modelo com melhor desempenho era salvo para comparações futuras.

Semelhantemente à etapa de desenvolvimento do modelo neural para os parâmetros cinéticos, as variáveis de entrada e de saída foram normalizadas no intervalo de 0 a 1 para promover equidade dos efeitos causados na função custo durante a etapa de treinamento.

Uma árvore de busca foi empregada para variar os hiperparâmetros selecionados para análise e que podem ser conferidos na Tabela 4. Ressalva-se aqui a importância do chamado passo temporal ou *timesteps*, hiperparâmetro que contempla o número de vetores temporais usados para cada exemplo durante a fase de treinamento e predição.

Tabela 4 – Hiperparâmetros analisados para redes LSTM.

Hiperparâmetro	Valor
Passo temporal	1, 2 e 3
Neurônios	1, 3, 5, 10, 15 e 20
Inicialização dos pesos	Aleatória uniforme, aleatória normal e identidade
γ_1	0, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}
γ_2	0, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}
Dropout	0; 0,05; 0,10; 0,15 e 0,20

O banco de dados obtido para o treinamento e para o teste foram reorganizados em uma matriz tridimensional (n,m,p) em que n representa o número de exemplos para o treinamento ou predição, m consiste na dimensão temporal do vetor e p indica as variáveis de entrada ou saída. Para os modelos desenvolvidos, a predição é feita para um passo temporal, ou seja, três horas em relação ao último vetor de entrada, independentemente do número de passos temporais utilizados para as variáveis de entrada.

O desempenho das redes foi analisado com base no critério MSE comentado anteriormente tanto para a fase de treinamento quanto de teste. A rede final selecionada também foi avaliada com base no teste de regressão e R^2 comentados na Seção 4.2. Todos os treinamentos foram desenvolvidos por meio de um processador Intel® Core™ i7 e memória RAM de 8 GB.

4.4.2 Modelo *offline*

A fim de determinar se a RNN-LSTM possui a capacidade de predizer eficientemente o comportamento do processo fermentativo com base somente em um ponto de regime inicial e nas variáveis exógenas, desenvolveu-se uma rotina de cálculo recursivo aqui denominado modo *offline*.

Após a determinação da melhor rede com topologia LSTM em modo *online*, as entradas endógenas usadas para a predição foram aplicadas recursivamente na rede de aprendizagem profunda, ou seja, os valores calculados por esta eram utilizados na predição do próximo ponto, excetuando o primeiro vetor de entrada.

Para a determinação da dinâmica do processo, as entradas exógenas do teste foram divididas em três conjuntos menores a fim de evitar grande acúmulo de erros ao longo do cálculo recursivo. O desempenho quantitativo dessa etapa foi calculado com base nos critérios MSE e no erro relativo percentual.

4.5 Interface gráfica do usuário

Com o intuito de desenvolver uma plataforma de rápido acesso aos resultados empregando as RNN-LSTM em modo *offline*, foi criada uma interface gráfica de usuário ou GUI (do inglês *Graphical User Interface*). O programa foi desenvolvido empregando a biblioteca *tkinter* disponível para Python.

O programa permite a visualização da simulação de 200 h do processo ao serem fornecidos os dados para a primeira predição e um modelo RNN-LSTM previamente treinado. Ressalva-se que foi assumido para o desenvolvimento da rotina de cálculos da GUI que as últimas entradas exógenas seriam mantidas ao longo do tempo simulado. O número de *timestep* empregado depende da estrutura da rede LSTM desenvolvida e carregada no programa e deve ser informado inicialmente para o funcionamento correto.

5 Resultados e discussões

Os resultados obtidos nesta dissertação são apresentados em 5 tópicos distintos, porém dependentes entre si. Na Seção 5.1 são apresentados os modelos neurais obtidos para a predição dos valores dos parâmetros cinéticos do modelo matemático apresentado na Seção 4.2. A geração do banco de dados necessário para posterior desenvolvimento de modelos de RNN-LSTM e uma breve análise da dinâmica do processo estão descritas na Seção 5.2. A Seção 5.3 compreende os resultados dos treinamentos supervisionados e as validações das redes recorrentes com topologia LSTM em modo de operação *online*. Na Seção 5.4 é investigada a possibilidade de aplicação das redes LSTM em modo *offline* para predizer a dinâmica do processo fermentativo a partir de uma única condição inicial e diferentes variações das entradas exógenas. Por fim, na Seção 5.5 é apresentada uma interface gráfica do usuário desenvolvida para auxiliar na obtenção da resposta dinâmica do sistema aplicando o modelo *offline* desenvolvido.

5.1 Obtenção da ANN para determinação dos valores dos parâmetros cinéticos do modelo fermentativo

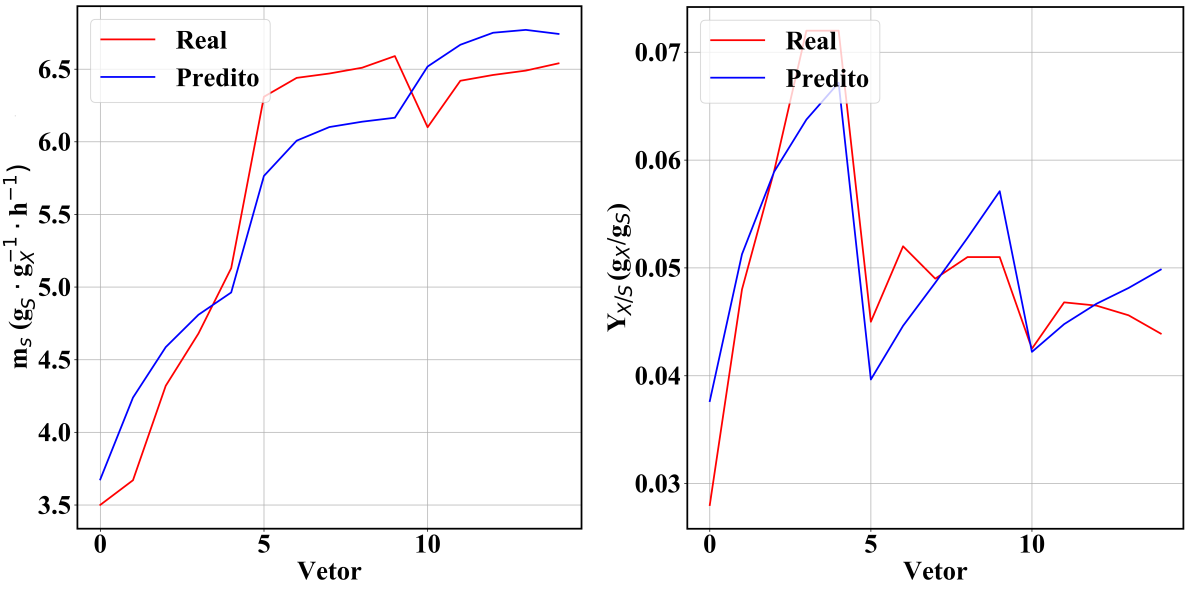
Diferentes modelos neurais foram obtidos ao variar os hiperparâmetros analisados utilizando a ferramenta *gridsearchcv* presente na biblioteca *scikit-learn*. Essa permite obter os diferentes modelos neurais automaticamente, como pode ser conferido no código do programa em linguagem Python implementado para esta etapa e disponibilizado no Apêndice A.

Como visto na Tabela 2, foram propostas variações em sete hiperparâmetros (número de neurônios, número de camadas ocultas, função de ativação, pesos para as regularizações L_1 e L_2, percentual de *dropout* e distribuição inicial dos pesos), originando 75.000 combinações possíveis para os modelos neurais, cada uma sendo utilizada para treinar 5 conjuntos de treinamento no método de validação cruzada. Por simplicidade, somente o modelo com melhor desempenho, ou seja, menor valor para o MSE médio, será apresentado e comentado a seguir.

Entre as combinações averiguadas, o modelo com função de ativação *softsign*, valores de desligamento, γ_1 e γ_2 iguais a zero, 15 neurônios em uma única camada oculta e distribuição inicial LeCun uniforme apresentou o melhor desempenho com média e desvio padrão do MSE na validação cruzada de $0{,}0459 \pm 0{,}0274$. Após treinamento com todos os exemplos do banco de dados, a rede resultante obteve um MSE de 0,0129, resultado 71,9% menor que a média obtida, reflexo do maior número de vetores empregado para o treinamento da rede final. Também verificou-se que redes com maior número de camadas ocultas não apresentaram melhoria significativa do desempenho em comparação aos modelos menos profundos, possivelmente devido ao pequeno volume de vetores para o treinamento das ANN.

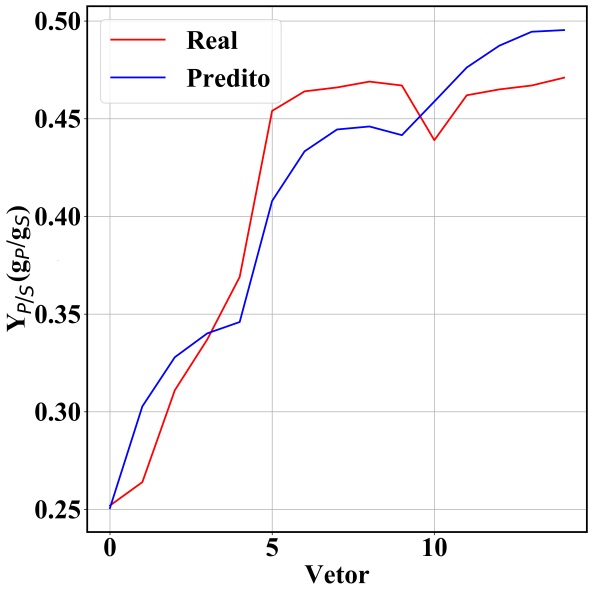
Na Figura 11 está ilustrada a comparação entre os valores reais obtidos experimentalmente no trabalho de Ghaly e El-Taweel (1997) e os preditos pela rede neural escolhida. Como visto, as predições conseguem simular a tendência dos dados experimentais, porém, a diferença entre os valores reais e preditos é considerável, especialmente para o coeficiente de rendimento celular. O resultado aquém da idealidade é fruto do restrito número de vetores de entrada disponibilizado para o treinamento que conta com somente 15 vetores.

Figura 11 – Predição dos parâmetros cinéticos do modelo utilizando a ANN.



(a) Coeficiente de manutenção.

(b) Coeficiente de rendimento celular.

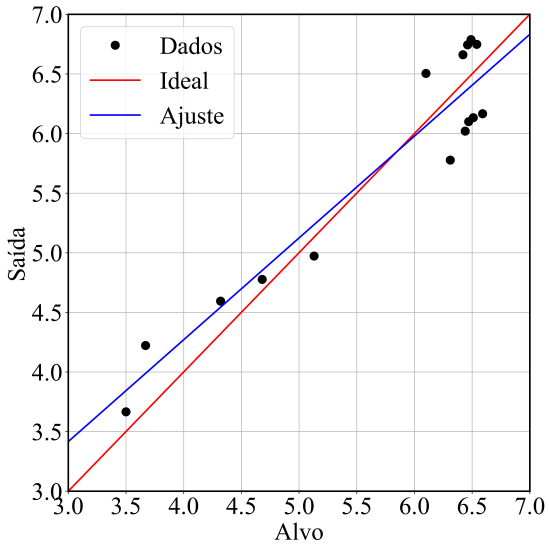


(c) Coeficiente de rendimento do produto.

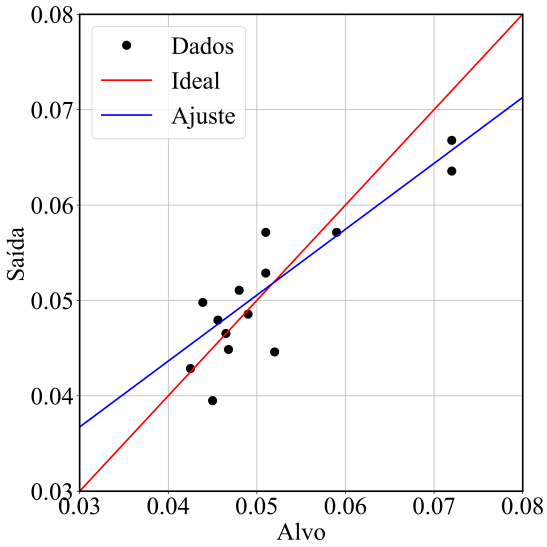
Os testes de regressão para o modelo neural são ilustrados na Figura 12. Como evidenciado pelos gráficos, há uma considerável distinção entre a regressão obtida pelo modelo e a

idealidade, resultante do pequeno banco de dados que não permite um treinamento adequado das redes neurais.

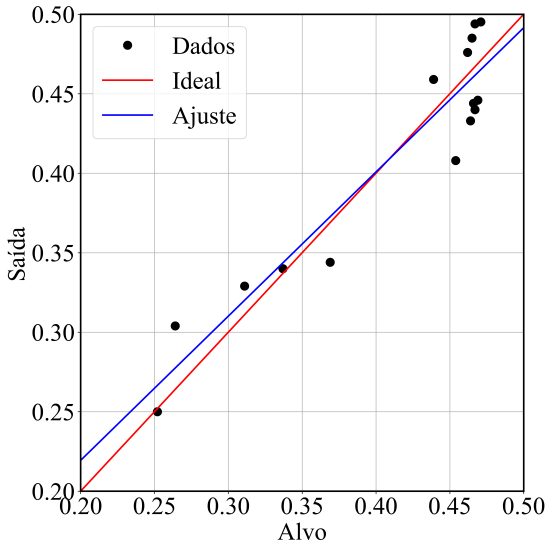
Figura 12 – Testes de regressão da ANN para predição dos parâmetros cinéticos.



(a) Coeficiente de manutenção.



(b) Coeficiente de rendimento celular.



(c) Coeficiente de rendimento do produto.

Os resultados quantitativos do teste de regressão estão sumarizados na Tabela 5. Comprovando os resultados qualitativos mencionados anteriormente, verifica-se que os coeficientes angulares estão abaixo da unidade, em especial aquele determinado para $Y_{X/S}$ e, de modo semelhante, os resultados para o coeficiente de determinação R^2.

Tabela 5 – Resultado quantitativo para o teste de regressão da ANN empregada para predizer os parâmetros cinéticos do modelo fermentativo.

Variável	Coef. angular	Coef. linear	R^2
m_S	0,854	0,854	0,896
$Y_{X/S}$	0,691	0,016	0,786
$Y_{P/S}$	0,907	0,038	0,896

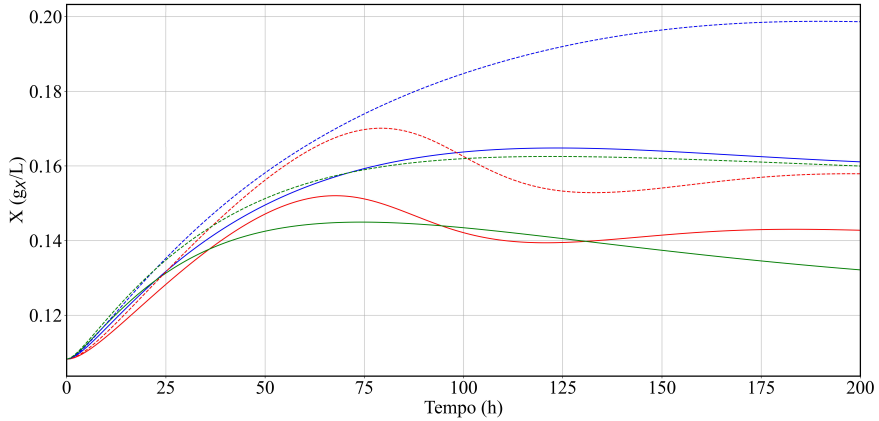
Apesar do resultado um pouco aquém do desejado, o modelo neural promove a variação dos valores dos parâmetros cinéticos baseados nas condições de processo dentro de um intervalo esperado e seguindo a tendência dos dados experimentais, permitindo a simulação de um sistema mais próximo da realidade se em comparação aos modelos com parâmetros constantes.

A fim de comparar as diferenças no comportamento dinâmico do processo fermentativo ao empregar as redes neurais para modificar os valores dos parâmetros cinéticos frente a um modelo com parâmetros constantes, foram causadas perturbações do tipo degrau nas variáveis de entrada Q_{in} e S_{in} em um sistema inicialmente em regime permanente para taxas de diluição de 42 h (correspondente a uma vazão de 0,1086 L/h) e concentração de substrato na alimentação de 50 g_S/L. Para o modelo com parâmetros cinéticos constantes, foram mantidos os valores referentes ao ponto de regime permanente inicial que correspondem a 5,13 $g_S/(g_X \cdot h)$, 0,072 g_X/g_S e 0,369 g_P/g_S para m_S, $Y_{X/S}$ e $Y_{P/S}$, respectivamente. As respostas dos modelos estão ilustradas nas Figuras 13 e 14 para as variações em S_{in} e Q_{in}, respectivamente. Ressalta-se que para o modelo a parâmetros variantes, as respostas estão ilustradas com linhas contínuas enquanto que o modelo com parâmetros constantes está representado com linhas tracejadas.

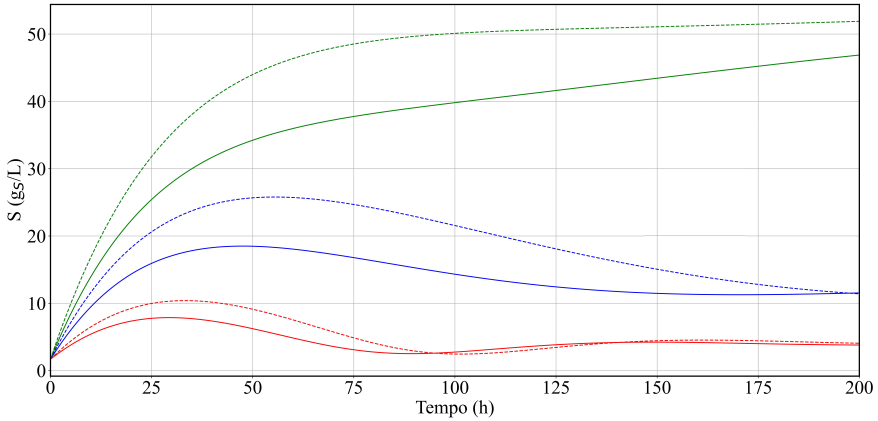
Como observado nas Figuras 13 e 14, o modelo fermentativo apresenta não linearidade ilustrada pelos diferentes ganhos do processo K calculados como a razão entre a variação da variável de processo pela variação da variável de entrada do modelo. Todos os valores calculados estão sumarizados na Tabela 6. Além disso, nota-se que o modelo apresenta diferentes comportamentos dinâmicos que variam entre amortecidos e oscilatórios para uma mesma variável de processo a depender da magnitude do degrau de entrada - por exemplo, Figura 13 (a).

Devido à aplicação da ANN para ajustar os parâmetros cinéticos, notam-se sensíveis variações nos valores dos ganhos do processo fermentativo que chegam a ser 212% maiores que os valores calculados para o modelo com parâmetros constantes. Também foram verificadas inversões dos sinais de ganhos, indicando sentidos opostos das respostas como visto para o crescimento celular e para a concentração de etanol quando alterada a vazão de alimentação. Além disso, os comportamentos dinâmicos foram modificados, com alteração da amplitude e da frequência de oscilação, por exemplo. Por fim, destaca-se que 5 das 6 perturbações causadas no processo levaram a valores muitos próximos em regime permanente para a concentração de substrato, entretanto, as respostas observadas para as outras variáveis sugerem mudanças no metabolismo celular e na empregabilidade da fonte de carbono.

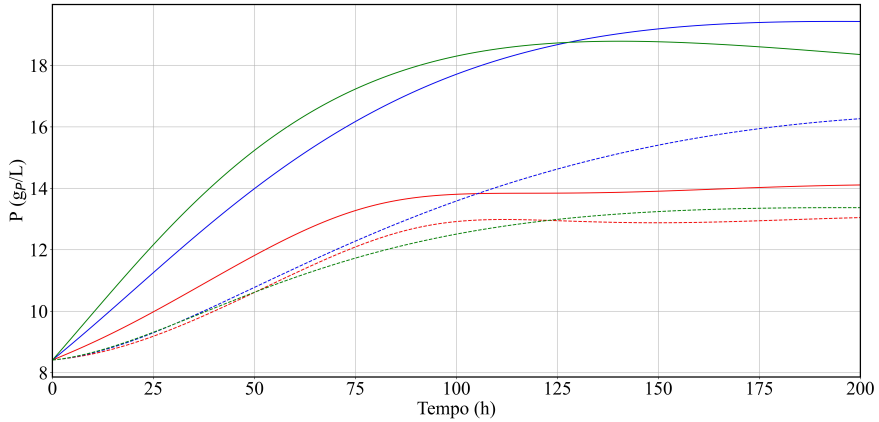
Figura 13 – Comparação das respostas a diferentes degraus na concentração de substrato da alimentação empregando o modelo fermentativo com parâmetros constantes e com parâmetros variantes.



(a) Concentração celular.



(b) Concentração de substrato.



(c) Concentração de produto.

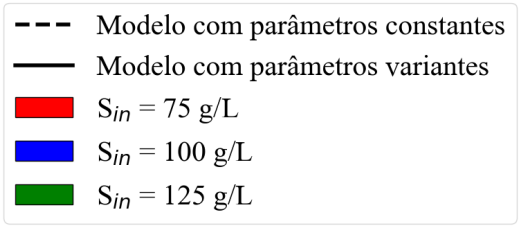
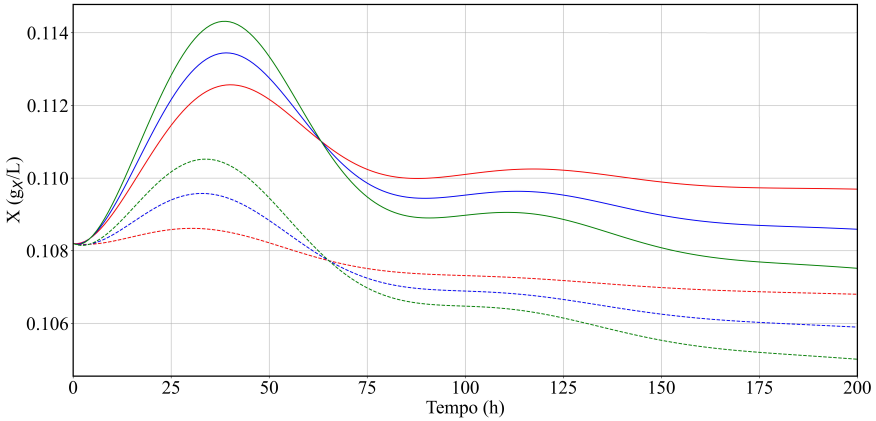
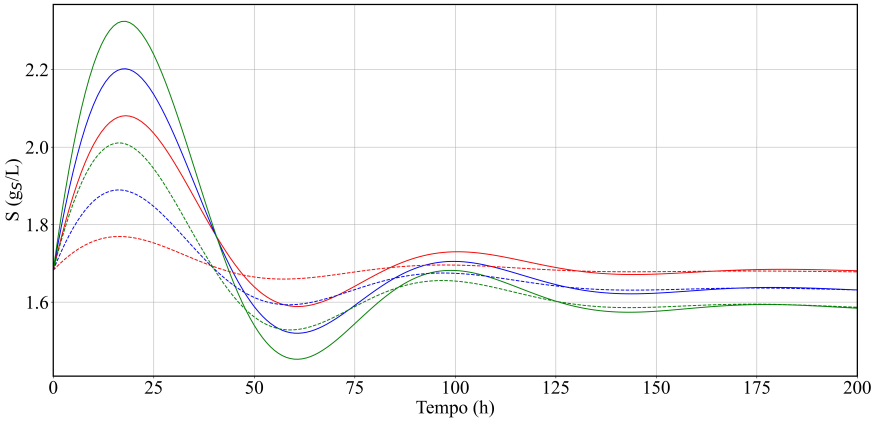


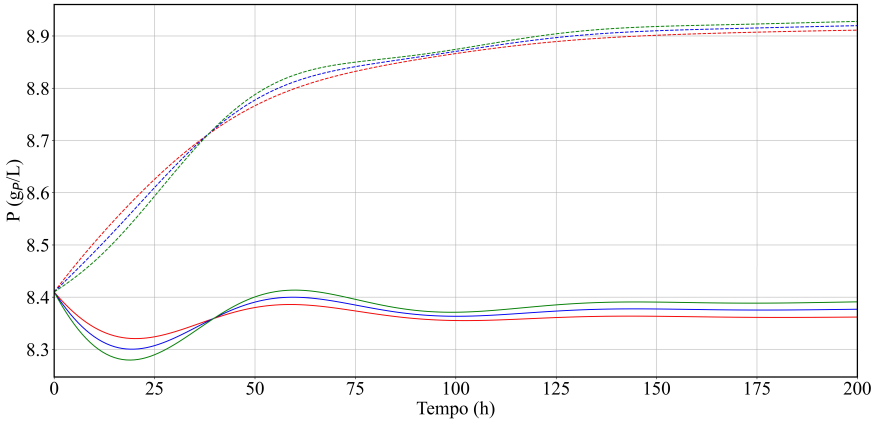
Figura 14 – Comparação das respostas a diferentes degraus na vazão da alimentação empregando o modelo fermentativo com parâmetros constantes e com parâmetros variantes.



(a) Concentração celular.



(b) Concentração de substrato.



(c) Concentração de produto.

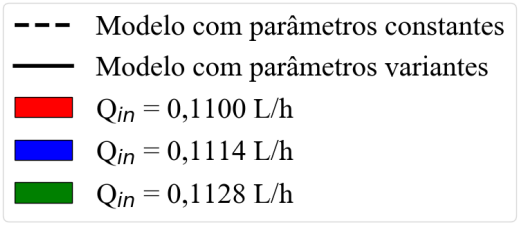


Tabela 6 – Ganhos calculados para o modelo fermentativo.

Modelo com parâmetros constantes				**Modelo com parâmetros variantes**			
Q_{in} [L/h]	K_X [$g_X \cdot h/L^2$]	K_S [$g_S \cdot h/L^2$]	K_P [$g_P \cdot h/L^2$]	Q_{in} [L/h]	K_X [$g_X \cdot h/L^2$]	K_S [$g_S \cdot h/L^2$]	K_P [$g_P \cdot h/L^2$]
0,1100	-0,98	4,29	355,10	0,1100	1,10	4,29	-35,71
0,1114	-0,83	-15,71	180,10	0,1114	0,23	-15,71	-13,78
0,1128	-0,77	-23,33	122,45	0,1128	-0,16	-23,33	-0,68
S_{in} [g_S/L]	K_X [g_X/g_S]	K_S [g_S/g_S]	K_P [g_P/g_S]	S_{in} [g_S/L]	K_X [g_X/g_S]	K_S [g_S/g_S]	K_P [g_P/g_S]
75	$1,98 \cdot 10^{-3}$	0,11	0,18	75	$1,38 \cdot 10^{-3}$	0,11	0,23
100	$1,79 \cdot 10^{-3}$	0,20	0,16	100	$1,05 \cdot 10^{-3}$	0,20	0,22
125	$6,85 \cdot 10^{-4}$	0,67	0,07	125	$3,17 \cdot 10^{-4}$	0,63	0,13

5.2 Obtenção do banco de dados para treinamento das RNN-LSTM

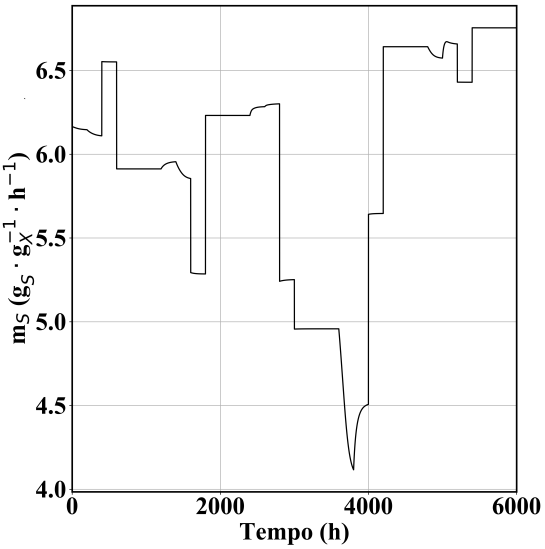
Os comportamentos dinâmicos das variáveis de processo estudadas neste trabalho quando o sistema é submetido a diferentes perturbações nas variáveis de entrada do modelo Q_{in}, S_{in} e X_{in} foram determinados pela resolução do sistema de equações diferenciais e constitutivas apresentado na Seção 4.1 juntamente ao modelo neural escolhido na Seção 5.1. Os valores selecionados para as entradas e seus pontos de mudança estão sumarizados na Tabela 7.

Tabela 7 – Perturbações nas variáveis exógenas para geração do banco de dados.

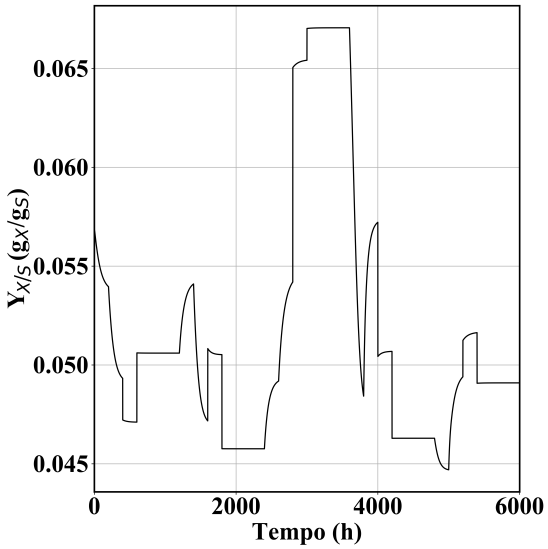
t (h)	Q (L/h)	Si (g/L)	Xi (g/L)	t (h)	Q (L/h)	Si (g/L)	Xi (g/L)
0	0,0969	100	0,025	3000	0,1086	50	0,055
200	0,0798	100	0,025	3200	0,1086	50	0,035
400	0,0798	130	0,025	3400	0,1086	50	0,065
600	0,0798	90	0,025	3600	0,0570	50	0,065
800	0,0798	90	0,045	3800	0,0741	50	0,065
1000	0,0798	90	0,060	4000	0,0744	80	0,065
1200	0,0912	90	0,060	4200	0,0744	140	0,065
1400	0,0684	90	0,060	4400	0,0744	140	0,030
1600	0,0684	70	0,060	4600	0,0744	140	0,050
1800	0,0684	110	0,060	4800	0,0627	140	0,050
2000	0,0684	110	0,075	5000	0,1026	140	0,050
2200	0,0684	110	0,055	5200	0,1026	120	0,050
2400	0,0855	110	0,055	5400	0,1026	150	0,050
2600	0,1083	110	0,055	5600	0,1026	150	0,040
2800	0,1086	60	0,055	5800	0,1026	150	0,070

Os valores empregados dos parâmetros cinéticos ao longo da simulação em decorrência das modificações impostas ao sistema estão ilustrados na Figura 15, enquanto as respostas dinâmicas das variáveis de processo estão descritas na Figura 16. O código implementado para a resolução desta etapa pode ser conferido no Apêndice B.

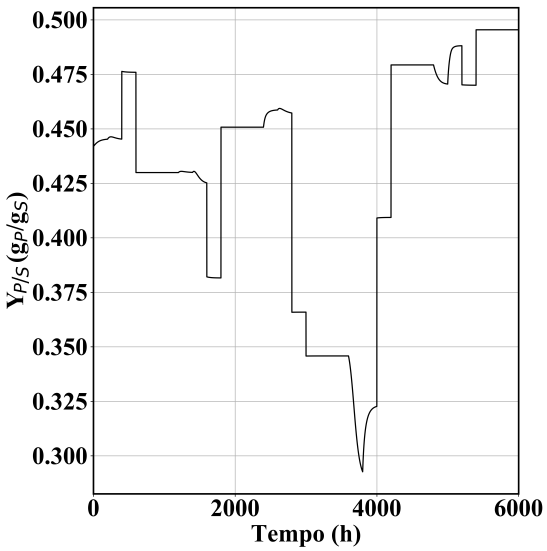
Figura 15 – Valores dos parâmetros cinéticos empregados na simulação.



(a) Coeficiente de manutenção.



(b) Coeficiente de rendimento celular.



(c) Coeficiente de rendimento do produto.

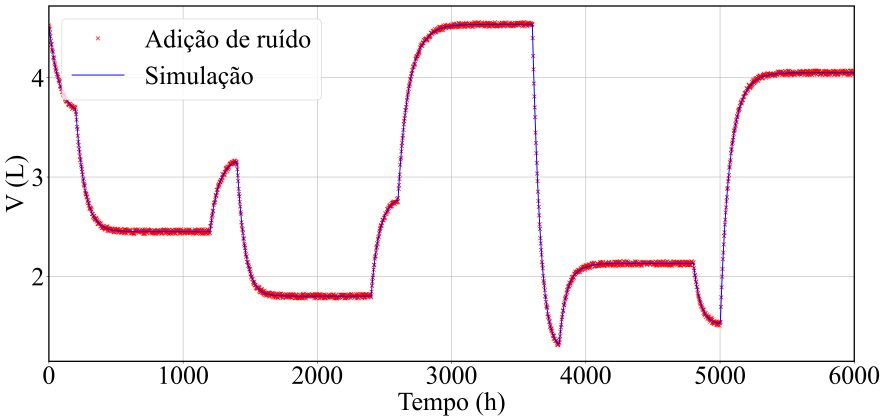
Como observado na Figura 15, os valores dos parâmetros cinéticos ao longo simulação são modificados em até 65% dentro de algumas horas, adaptando-se às condições operacionais empregadas na determinação do modelo neural (concentração de substrato da corrente de ali-

mentação e tempo de detenção hidráulico). Nota-se a presença de variações de grande amplitude nos valores, sendo resultantes da súbita entrada degrau aplicada na variável S_{in}, enquanto que as modificações no tempo de detenção hidráulico ocorrem de modo mais gradual devido à sua dependência com o volume reacional, originando uma dinâmica mais suave para os parâmetros. Estas modificações podem ser vistas em 4.000 e 3.800 h, respectivamente.

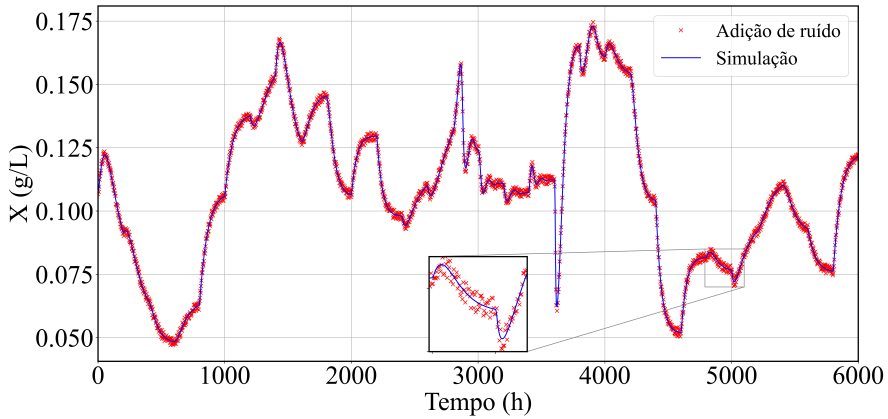
Pela inspeção da Figura 16, nota-se que dentro do intervalo de 200 h entre as mudanças, o sistema não atinge um novo regime estacionário, possivelmente devido à interação intrínseca entre todas as variáveis, excetuando-se o volume do sistema que somente depende da vazão de entrada e do seu próprio valor, aliada à constante variação dos parâmetros cinéticos do sistema. Dessa forma, ambos os regimes puderam ser empregados para a análise do desempenho dos modelos recorrentes de aprendizagem profunda.

Um aspecto relevante para o processo é a presença de uma aparente resposta inversa das concentrações em relação à vazão de entrada do sistema como pode ser notada em 4.800 e 5.000 h, por exemplo. Ressalta-se que o efeito observado na Figura 16 não representa a influência completa de Q_{in} uma vez que o sistema ainda não havia atingido o regime permanente para as entradas anteriores a essa mudança. Acredita-se que o fenômeno seja causado pelo efeito de diluição do conteúdo quando modificado o volume reacional para as concentrações celular e de etanol. Para o substrato, assume-se que seja causada como consequência da variação do número de células aliada à entrada do substrato na corrente de alimentação.

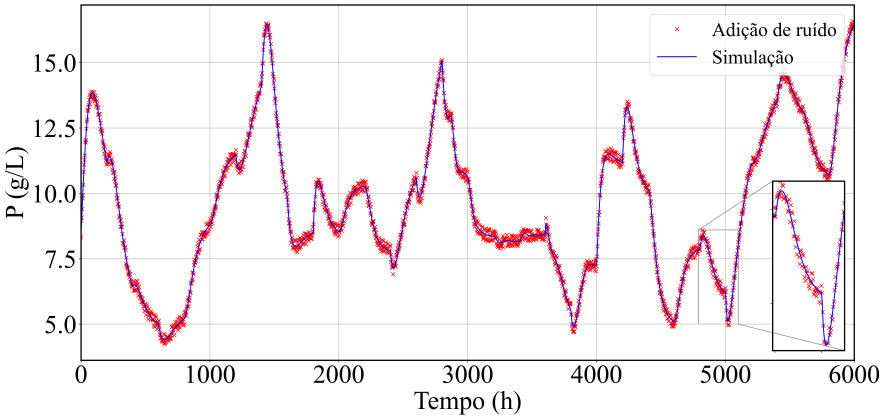
Figura 16 – Comportamento dinâmico das variáveis endógenas empregadas para o desenvolvimento dos modelos de aprendizagem profunda.



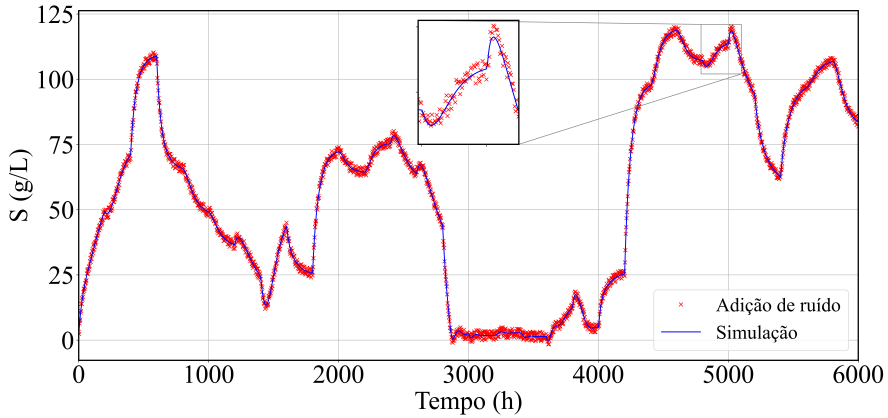
(a) Volume.



(b) Concentração celular.



(c) Concentração de produto.



(d) Concentração de substrato.

5.3 Obtenção do modelo de RNN-LSTM para operação em modo *online*

A simulação acima apresentada compõe o banco de dados empregado para o treinamento e para a validação dos modelos empíricos de aprendizagem profunda com topologia LSTM propostos neste trabalho. O banco de dados é constituído por 2.000 vetores, dos quais 1.600 foram destinados para o conjunto de treinamento e 400 para o de teste. Como mencionado na Seção 4.3, a taxa de aquisição selecionada foi de 3 h entre os vetores, assim o tempo total do banco de dados é de 6.000 h. Dos 1.600 exemplos do banco de dados de treinamento, 320 foram destinados para o conjunto de validação a fim de empregar a estratégia de parada antecipada.

Devido ao carácter temporal do problema, os vetores foram separados de modo sequencial assim como apresentados na Figura 16, ou seja, os dados entre 0 e 3.840 h foram empregados para os treinamentos das redes, entre 3.840 e 4.800 h para a validação e entre 4.800 e 6.000 h como banco de dados de teste para avaliar a capacidade de generalização dos modelos. Além disso, toda a informação temporal salva na célula de memória ao empregar os dados da fase de treinamento foi apagada previamente à predição usando o conjunto de teste e, assim, nenhuma informação cruzou os dois conjuntos de dados.

Como indicado na Tabela 4, 6.750 redes com topologia LSTM foram geradas e, por simplicidade, somente o melhor resultado para cada par de número de neurônios e *timesteps* está apresentado na Tabela 8. Ressalva-se que o parâmetro de desempenho MSE indicado nos resultados a seguir são referentes somente ao calculado para o conjunto de teste.

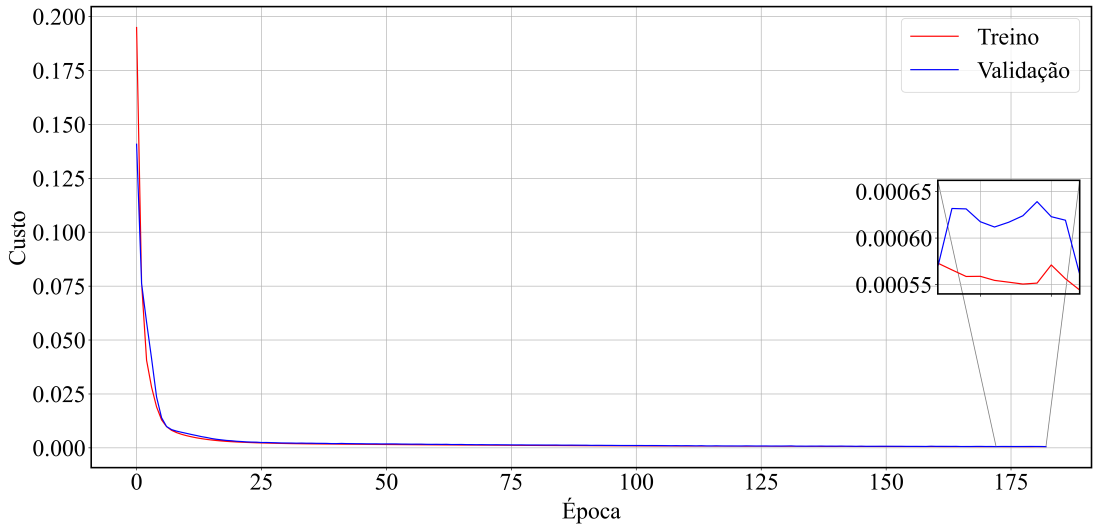
Tabela 8 – Melhores RNN-LSTM obtidas para o modelo fermentativo.

Rede	*Timesteps*	Neurônios	Inicialização	γ_1	γ_2	*Dropout*	MSE
1	1	3	Aleatória uniforme	10^{-3}	0	0	$4{,}36 \cdot 10^{-3}$
2	1	5	Identidade	0	10^{-4}	0	$8{,}85 \cdot 10^{-4}$
3	1	10	Identidade	0	0	0	$6{,}77 \cdot 10^{-4}$
4	1	15	Identidade	10^{-4}	10^{-3}	10%	$1{,}33 \cdot 10^{-3}$
5	1	20	Aleatória normal	10^{-3}	0	0	$1{,}32 \cdot 10^{-3}$
6	2	3	Identidade	0	10^{-1}	0	$3{,}18 \cdot 10^{-3}$
7	2	5	Aleatória uniforme	10^{-4}	10^{-4}	0	$5{,}64 \cdot 10^{-4}$
8	2	10	Aleatória normal	0	0	0	$3{,}98 \cdot 10^{-4}$
9	2	15	Identidade	0	0	0	$2{,}63 \cdot 10^{-4}$
10	2	20	Aleatória normal	10^{-3}	0	0	$7{,}99 \cdot 10^{-4}$
11	3	3	Identidade	10^{-2}	10^{-1}	0	$2{,}85 \cdot 10^{-3}$
12	3	5	Identidade	0	0	0	$1{,}55 \cdot 10^{-3}$
13	3	10	Aleatória normal	0	0	0	$5{,}34 \cdot 10^{-4}$
14	3	15	Aleatória normal	10^{-3}	10^{-3}	5%	$5{,}05 \cdot 10^{-4}$
15	**3**	**20**	**Identidade**	$\mathbf{10^{-4}}$	**0**	**0**	$\mathbf{2{,}21 \cdot 10^{-4}}$

Como observado na Tabela 8, a rede denominada 15 com três passos temporais, vinte neurônios compondo cada portão e rede neural das entradas, inicialização dos pesos com distribuição identidade, sem taxa de desligamento e pesos das regularizações L_1 e L_2 iguais a 10^{-4} e 0, respectivamente, apresentou o menor valor de custo para a fase de teste, com MSE na ordem de $2{,}21 \cdot 10^{-4}$. Assim, foi selecionada a rede 15 para a verificação das características de desempenho do modelo tanto para a fase de treinamento quanto de teste.

Está ilustrado na Figura 17 o valor da função custo ao longo das épocas de treinamento. Nota-se que devido às dez falhas sucessivas em reduzir o valor do MSE do conjunto de validação em um valor mínimo de 10^{-5} entre duas épocas consecutivas, o treinamento da RNN-LSTM 15 foi cessado após 183 épocas. Recorda-se que o conjunto de parâmetros selecionado para o modelo neural corresponde à otimização da centésima septuagésima terceira época. Ressalta-se novamente a importância da parada antecipada na redução do esforço computacional e, sobretudo, na mitigação do efeito de sobre-ajuste.

Figura 17 – Custos da RNN-LSTM 15 durante a fase de treinamento.



As predições feitas pela RNN-LSTM 15 para as 4 variáveis estudadas utilizando o conjunto de treinamento em sua totalidade estão representadas na Figura 18, juntamente com as curvas dos dados reais para fins de comparação. Percebe-se que as predições são extremamente próximas aos dados reais, indicando que o treinamento da rede foi adequado. Os erros observados são de baixa amplitude, sendo desconsideráveis em relação às magnitudes das faixas de trabalho das variáveis de processo.

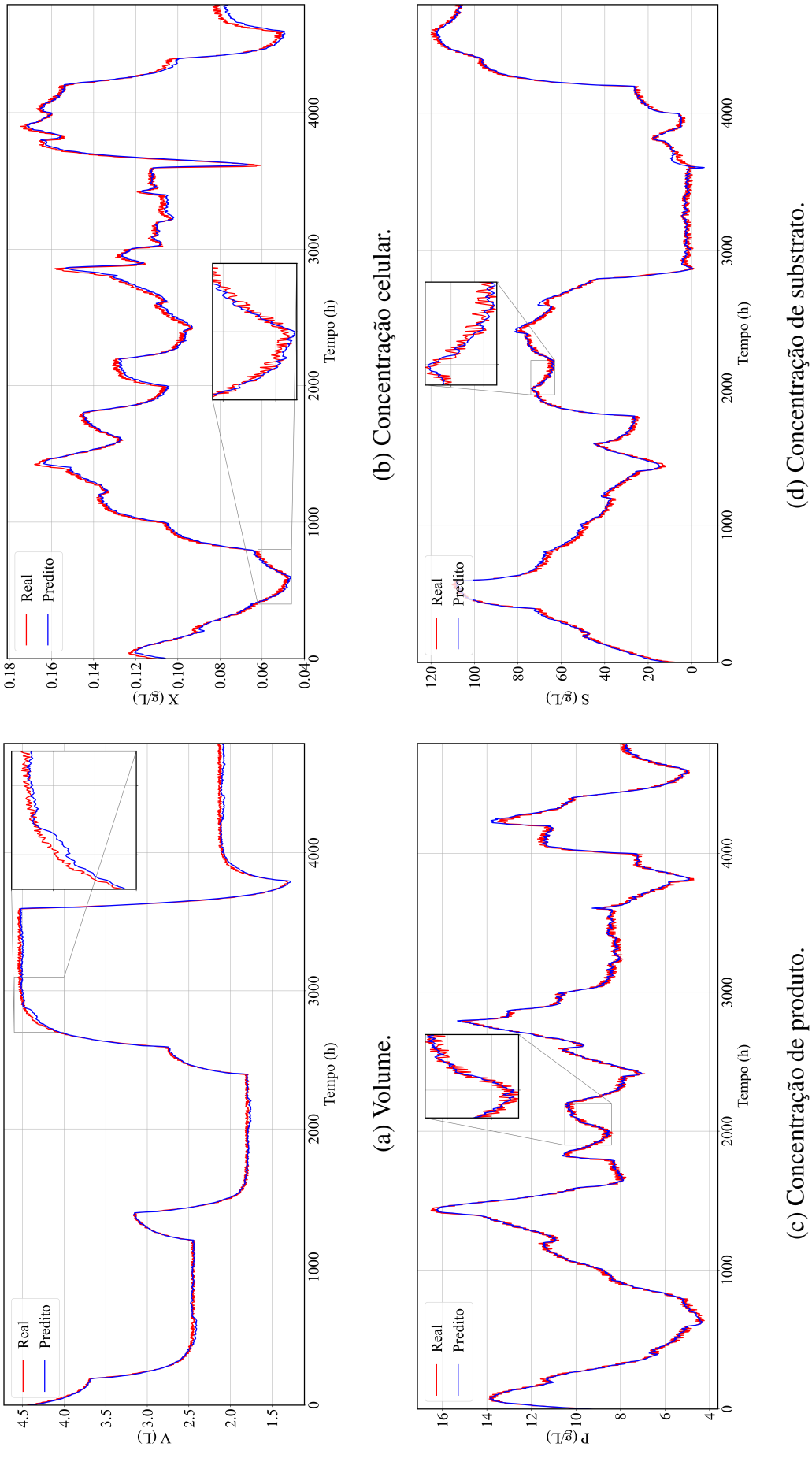
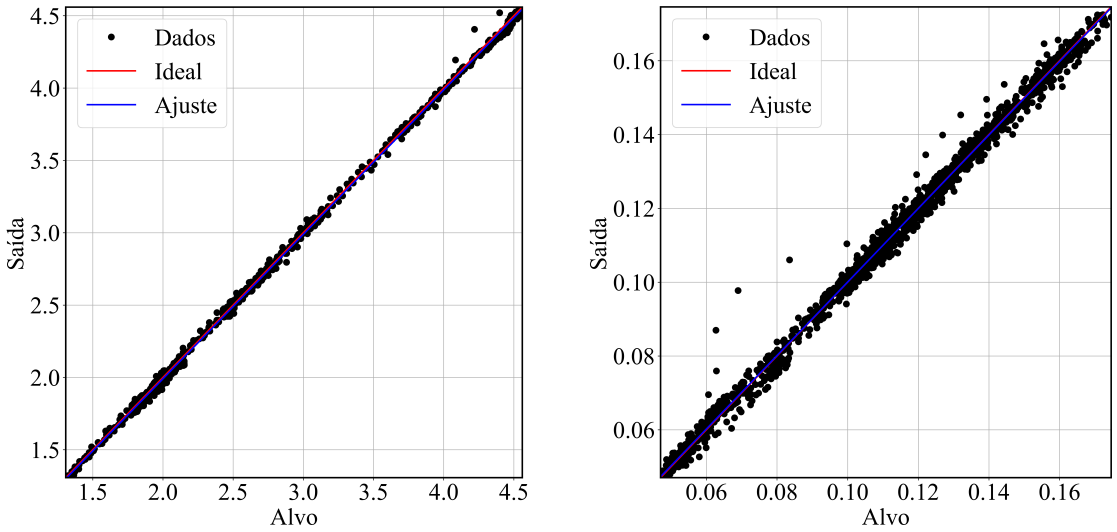


Figura 18 – Predições da RNN-LSTM 15 para banco de dados de treinamento.

Ao trabalhar com redes recorrentes, uma preocupação existente é a possibilidade de o modelo simplesmente repetir o valor apresentado da entrada endógena como predição, especialmente quando a diferença entre dois valores sucessivos é pequena devido a uma alta taxa de amostragem e/ou dinâmica lenta do processo aliada ao fato de a predição ser única e estar espaçada em um passo temporal (BROWNLEE, 2019b). Para averiguar se esse fenômeno ocorreu, foram destacadas faixas menores para cada uma das variáveis como mostrado na Figura 18.

Observa-se na Figura 18 que as curvas real e predita apresentam visíveis variações no comportamento tais como amplitude dos picos em pontos próximos e inversão do posicionamento vertical, além de apresentarem um comportamento mais suave em comparação aos dados reais com ruídos. A presença de ruídos durante a fase de treinamento aumenta o volume de informações contidas no banco de dados, atua como regularização e melhora a generalização de redes neurais, originando predições mais próximas aos valores reais do sistema (BISHOP, 1995; AN, 1996; GOODFELLOW; BENGIO; COURVILLE, 2016). Diante destas características, é possível inferir que a rede recorrente não simplesmente reproduz os dados de entrada como predição, indicando a validade do treinamento e da empregabilidade da RNN-LSTM em testes posteriores.

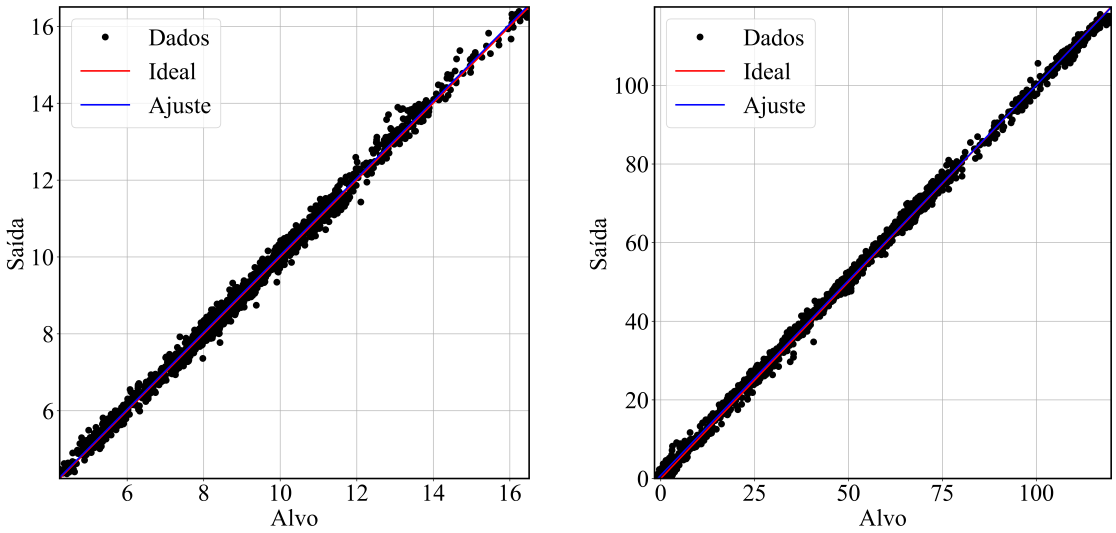
Figura 19 – Testes de regressão para o banco de dados de treinamento em modo *online*.



(a) Volume.

(b) Concentração celular.

Figura 18 – Testes de regressão para o banco de dados de treinamento em modo *online*.



(c) Concentração de produto.

(d) Concentração de substrato.

Os testes de regressão para as variáveis de processo utilizando como entradas todos os 1.600 exemplos contidos no banco de dados de treinamento estão ilustrados na Figura 19. Como observado, o ajuste real dos dados e a reta ideal são quase idênticos, com coeficientes de determinação R^2 próximos à unidade. Os coeficientes angulares obtidos na regressão foram muito próximos a um e os lineares praticamente zero, excetuando aquele determinado para o substrato. Entretanto, levando-se em consideração a faixa de valores para esta variável, o resultado é satisfatório. Os valores calculados para os testes de regressão estão expressos na Tabela 9.

Tabela 9 – Desempenho da RNN-LSTM 15 para fase de treinamento em modo *online*.

Variável	Coef. Angular	Coef. Linear	R^2
V	0,998	-0,006	0,999
X	0,996	0,000	0,994
P	1,002	0,022	0,994
S	0,992	0,779	0,998

Estes resultados indicam a capacidade da rede de predizer eficientemente as variáveis de saída do processo fermentativo utilizando os dados treinados, entretanto, deve-se averiguar sua capacidade de generalização para condições não observadas durante o treinamento. Assim, foi empregado o banco de dados de teste para gerar a predição utilizando a RNN-LSTM 15. Como mencionado anteriormente, toda a informação da célula de memória armazenada durante os cálculos da etapa de treinamento foi apagada de modo que a informação não cruzasse os bancos de dados. Os valores preditos pela RNN-LSTM são apresentados na Figura 20.

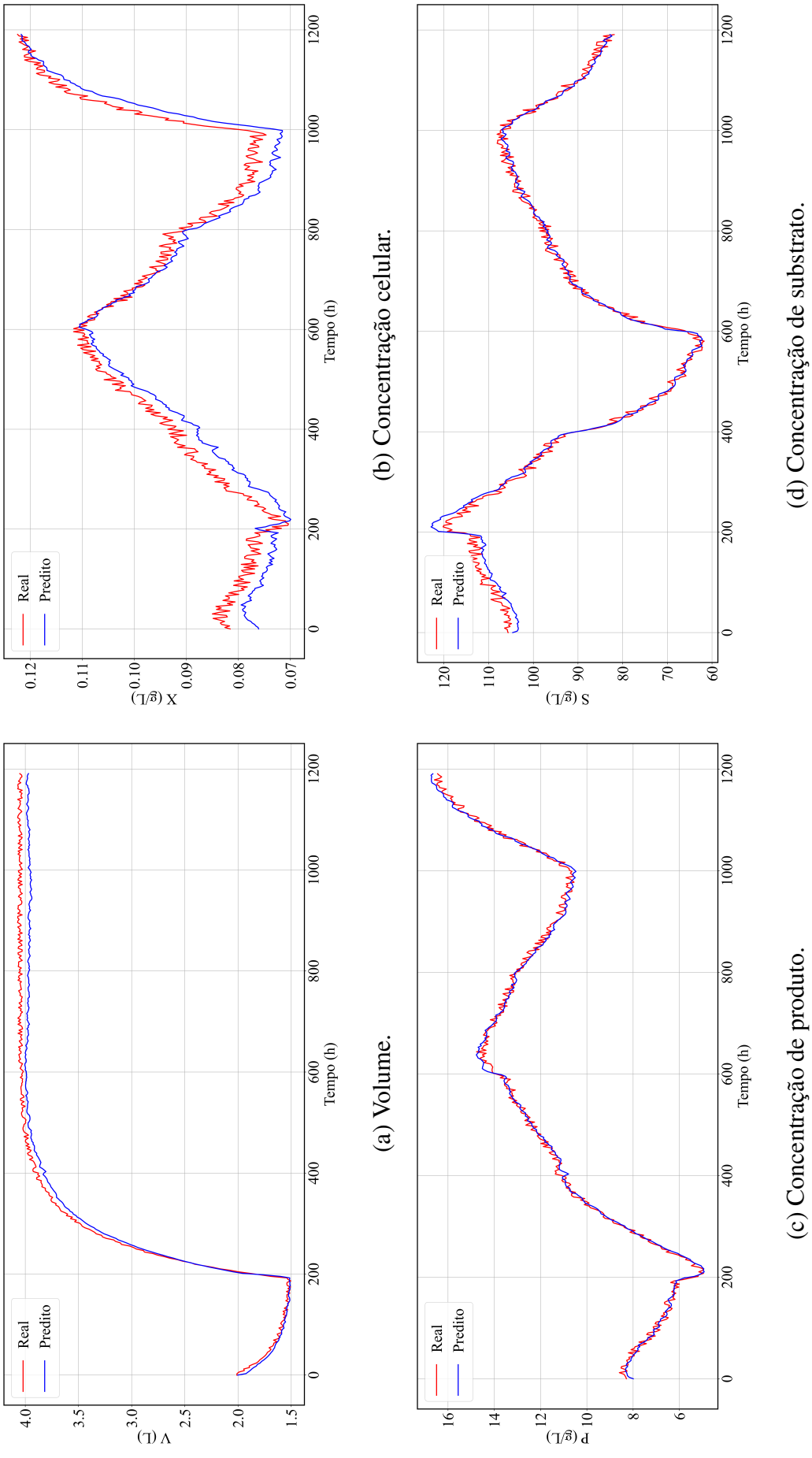


Figura 20 – Predições da RNN-LSTM 15 para banco de dados de teste em modo *online*.

(a) Volume.

(b) Concentração celular.

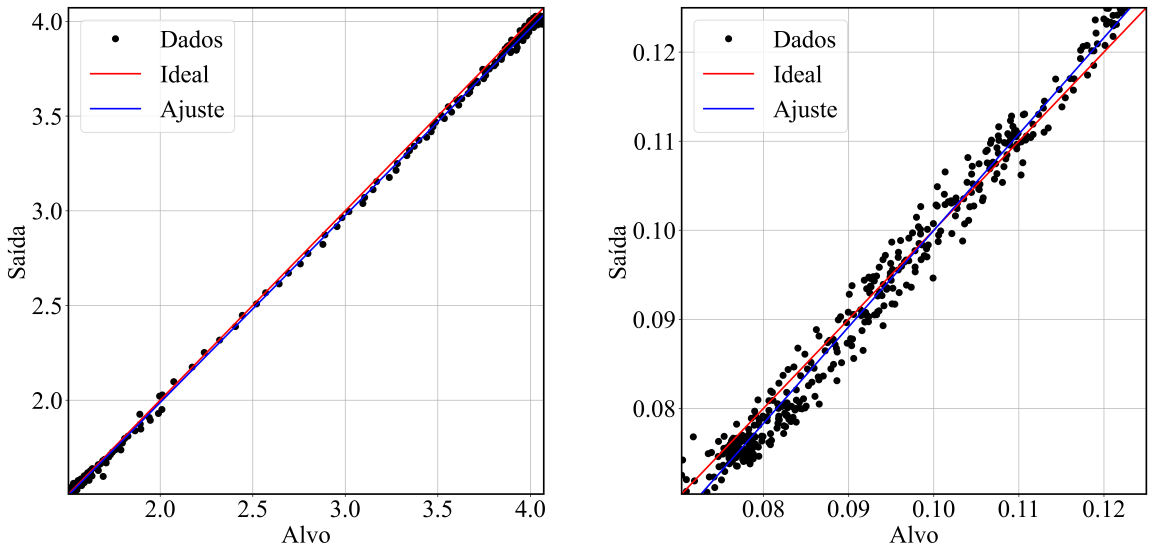
(c) Concentração de produto.

(d) Concentração de substrato.

Como verificado na Figura 20, as predições da RNN-LSTM se distanciam muito pouco dos valores reais e acompanham todo o comportamento dinâmico eficientemente, incluindo a resposta inversa presente em 200 h. Igualmente ao comentado anteriormente, as predições não são resultados de uma repetição dos valores de entrada da rede e incluem perceptíveis distinções, cruzamentos e superposições das curvas durante o regime transiente do sistema. Além disso, a rede prediz o regime permanente do volume com erros inferiores a 3%, indicando a habilidade da RNN-LSTM modelar tanto o regime transiente quanto permanente eficientemente mesmo em situações que não foram apresentadas na fase de treinamento.

Os testes de regressão para o banco de dados de teste estão ilustrados na Figura 21. Semelhantemente ao que foi averiguado para os vetores de treinamento, as retas ajustadas são extremamente próximas do resultado ideal, indicando boa concordância das predições da RNN-LSTM e dos valores reais. Os parâmetros quantitativos para o teste de regressão estão sumarizados na Tabela 10 e comprovam a alta correlação linear do modelo neural com os dados simulados, apresentando valores de R^2 e de coeficientes angulares muito próximos à unidade, além de coeficientes lineares aproximadamente nulos, excetuando para S, porém o valor é desprezável frente à faixa de valores da variável.

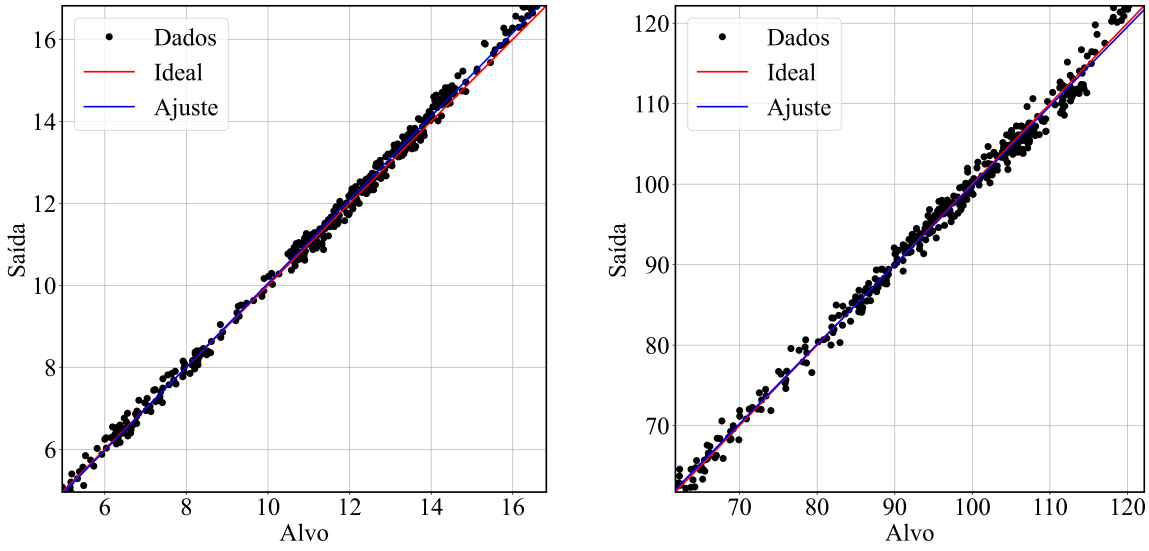
Figura 21 – Testes de regressão para o banco de dados de teste em modo *online*.



(a) Volume.

(b) Concentração celular.

Figura 20 – Testes de regressão para o banco de dados de teste em modo *online*.



(c) Concentração de produto.

(d) Concentração de substrato.

Tabela 10 – Desempenho da RNN-LSTM 15 para a fase de teste em modo *online*.

Variável	Coef. Angular	Coef. Linear	R^2
V	0,988	0,012	0,998
X	1,083	-0,008	0,970
P	1,019	-0,143	0,995
S	0,987	1,091	0,992

5.4 Aplicação da RNN-LSTM em modo de operação *offline*

Para verificar a habilidade da rede LSTM em trabalhar em modo *offline*, a RNN-LSTM escolhida na Seção 5.3 foi empregada para a predição aplicando suas próprias previsões recursivamente como variáveis endógenas. Para isso, três bancos de dados com 62 vetores denominados BD_1, BD_2 e BD_3 foram utilizados para simular o comportamento dinâmico do processo. Ressalta-se que os vetores empregados nesta etapa não foram apresentados na fase de otimização dos parâmetros da RNN-LSTM para averiguar a habilidade de generalização das redes mesmo em modo *offline*. Além disso, optou-se por dividir os bancos de dados pois o uso de ANN em modo recursivo leva a um acúmulo contínuo de erros que degeneram a resposta para períodos longos de predição (SOUZA, 2019). Ressalta-se que os bancos de dados selecionados não possuem continuidade entre si e, por isso, as informações da célula de memória da RNN-LSTM foram excluídas entre as predições de cada banco.

Para inicializar as predições, o primeiro conjunto de variáveis endógenas de cada banco de dados foi empregado, e suas variáveis exógenas foram usadas como entradas em totalidade das predições. A escolha do melhor modelo em modo *online* obtido anteriormente se baseia

na premissa de que este deve manter o bom desempenho em modo *offline*, já que as variações mesmo para o banco de dados não apresentado anteriormente foram mínimas. A modificação no código implementado para a configuração *offline* pode ser conferida no Apêndice D.

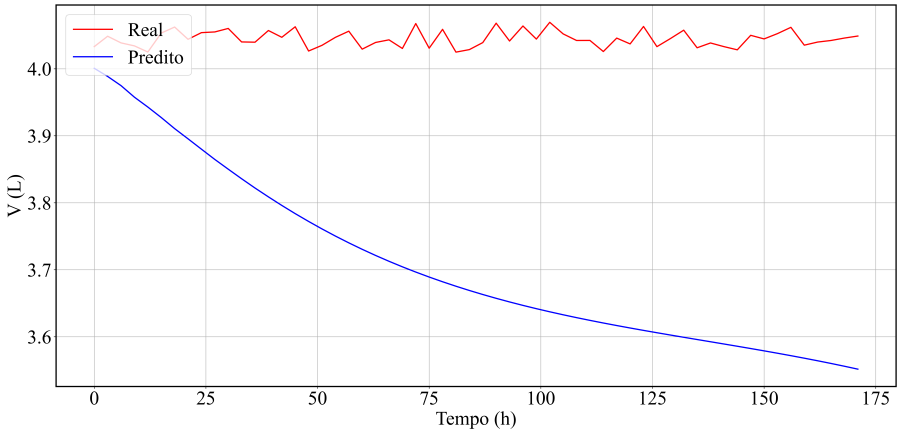
Comparações entre os valores reais das quatro variáveis e as predições da RNN-LSTM 15 são mostradas nas Figuras 22 a 24 para os bancos de dados BD_1, BD_2 e BD_3, respectivamente. Destaca-se que os volumes ilustrados para BD_1 e BD_2 representam faixas em regime permanente, assim, ambos os regimes foram analisados em operação *offline*. Além disso, os valores de entrada da variável exógena S_{in} para o BD_1 consistem em uma faixa de extrapolação do treinamento, situação reconhecidamente mais difícil de lidar pela rede (HETTIARACHCHI; HALL; MINNS, 2005).

Para os dois testes em regime permanente, as predições da RNN-LSTM 15 falharam em identificar a característica do estado do sistema, predizendo valores cada vez mais distantes do valor de regime ao longo das simulações. Para todos os bancos de dados, os menores erros das predições ocorreram para as primeiras calculadas cujos valores das variáveis endógenas empregadas no modelo são os dados reais obtidos dos bancos de dados. Os comportamentos de todas as variáveis foram muito distintos do esperado com altas diferenças de valores em grande totalidade das simulações. É interessante destacar que para BD_3, todos os resultados preditos apresentaram tendência oposta aos valores reais. Todas essas características implicam na ineficiência desse modelo neural em predizer o processo fermentativo em modo *offline*. Para fins de comparação posterior, o MSE calculado para cada banco de dados está sumarizado na Tabela 11.

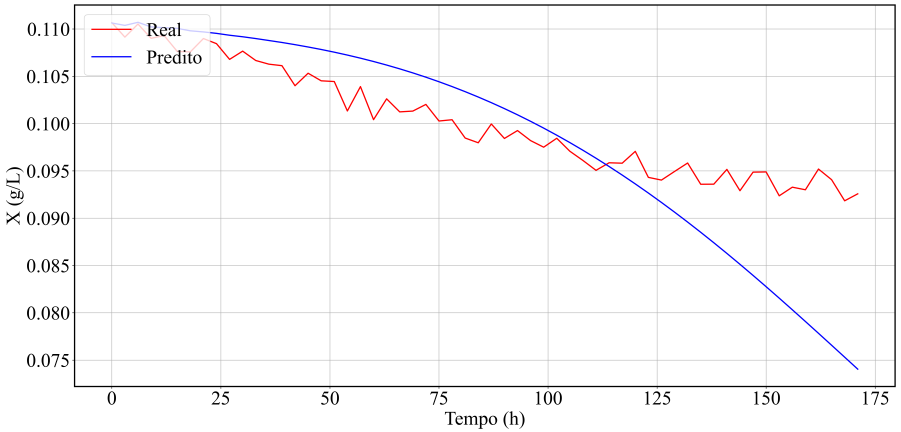
Tabela 11 – MSE calculado para operação *offline* empregando a RNN-LSTM 15.

BD	MSE
1	$6{,}55 \cdot 10^{-3}$
2	$5{,}45 \cdot 10^{-3}$
3	$5{,}78 \cdot 10^{-2}$
Total	$\mathbf{6{,}98 \cdot 10^{-2}}$

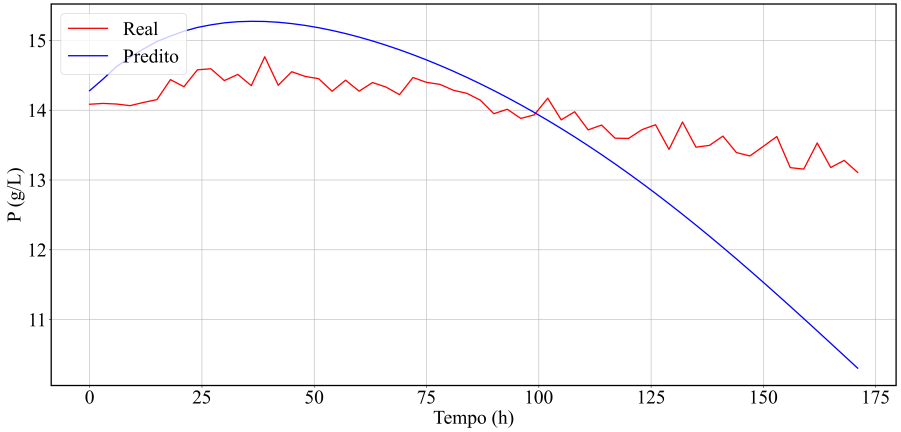
Figura 22 – Predições da RNN-LSTM para BD_1 em modo *offline*.



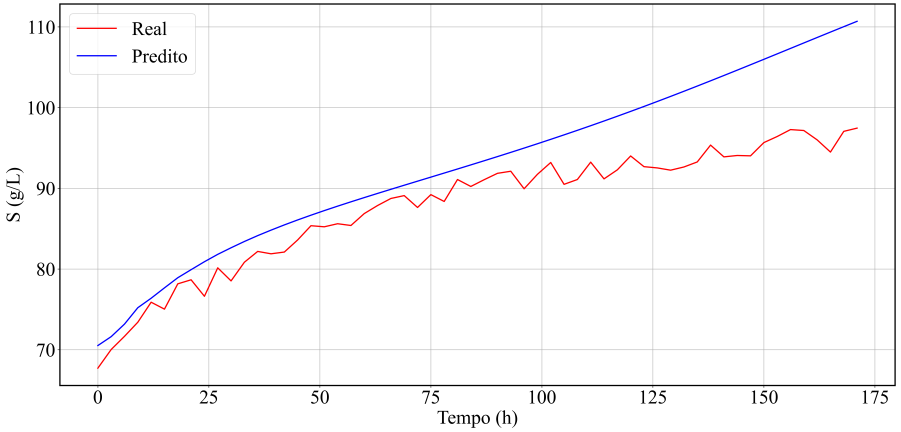
(a) Volume.



(b) Concentração celular.

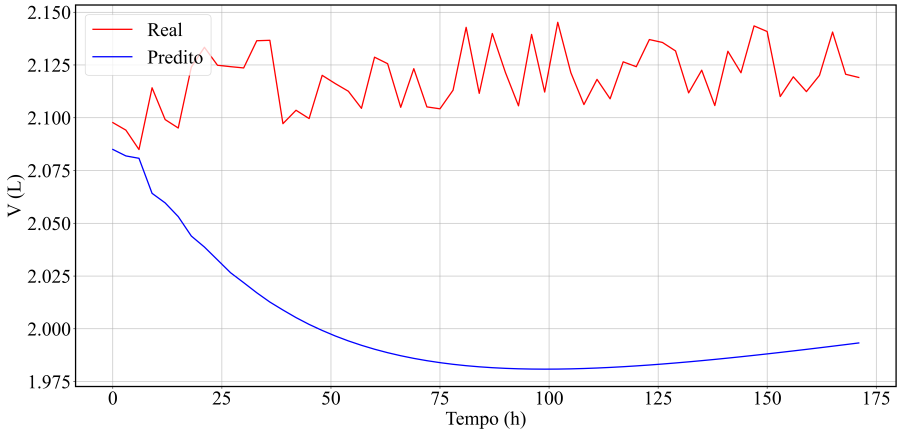


(c) Concentração de produto.

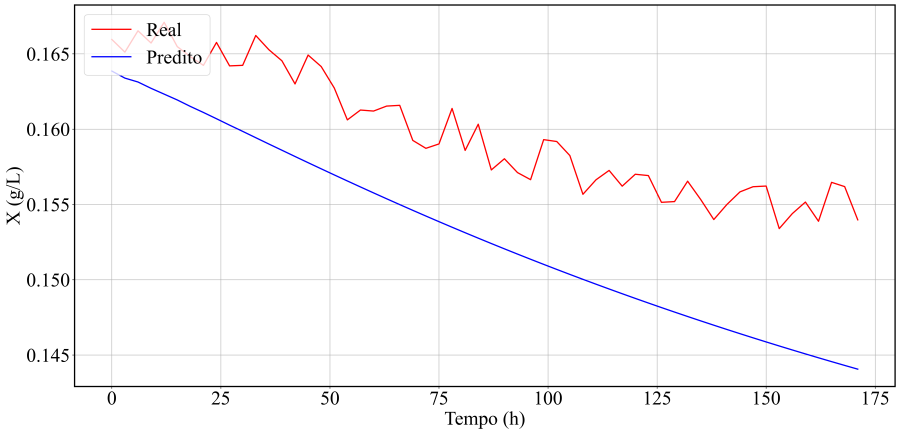


(d) Concentração de substrato.

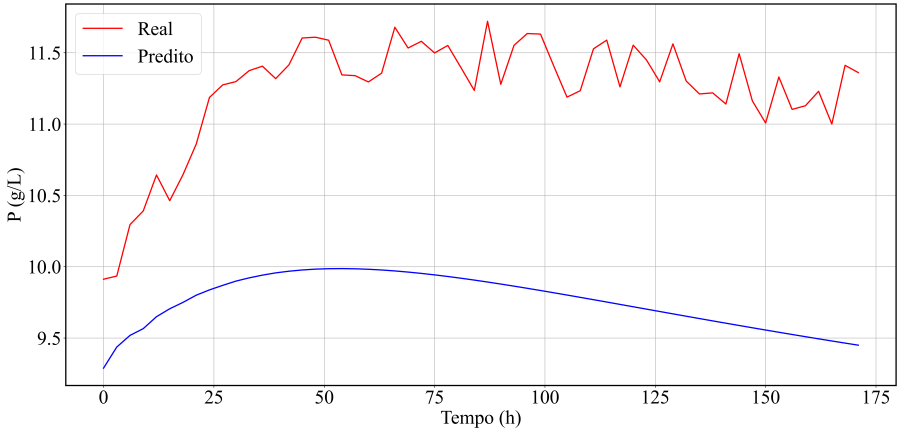
Figura 23 – Predições da RNN-LSTM para BD_2 em modo *offline*.



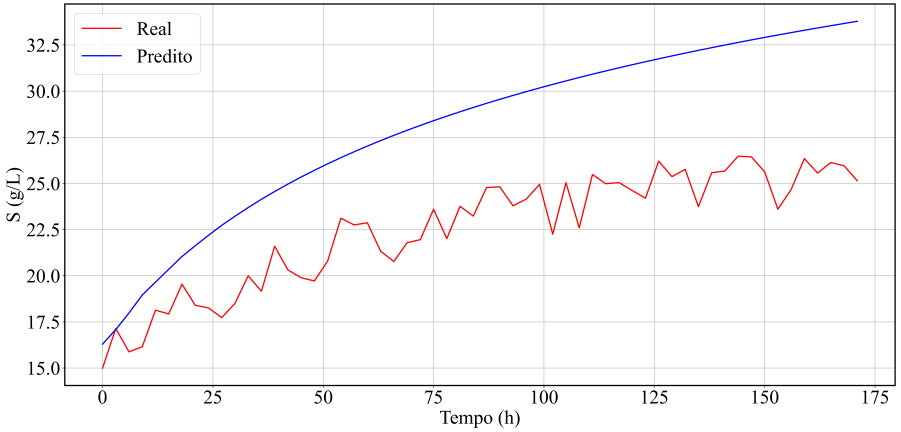
(a) Volume.



(b) Concentração celular.

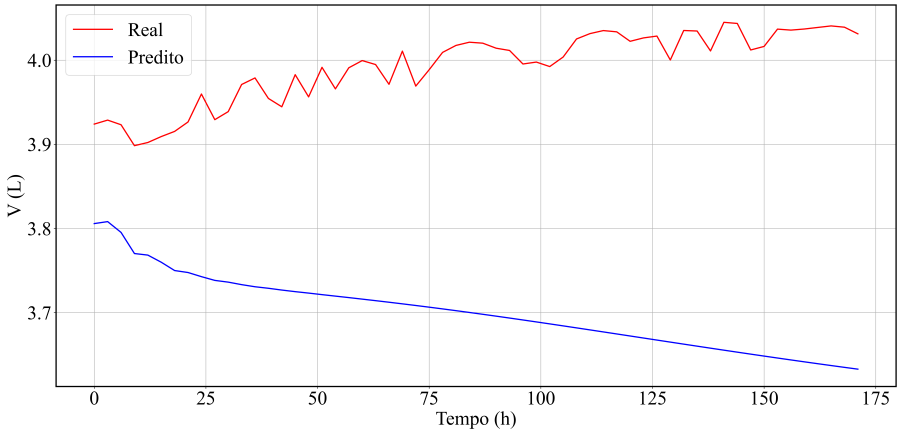


(c) Concentração de produto.

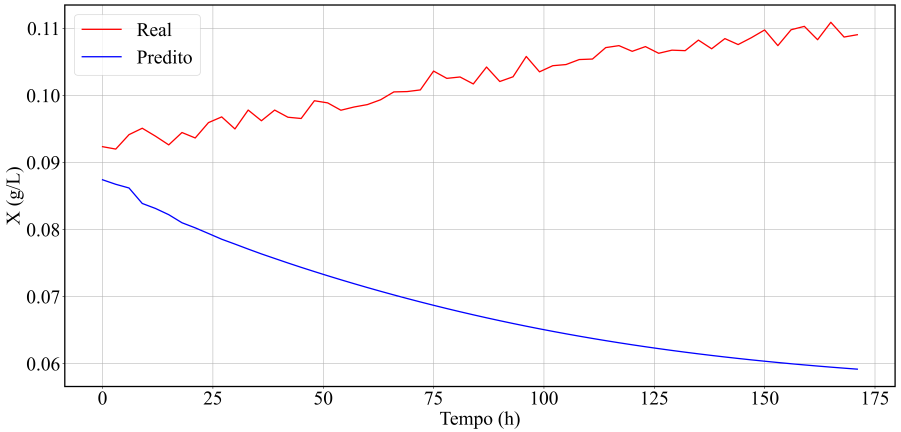


(d) Concentração de substrato.

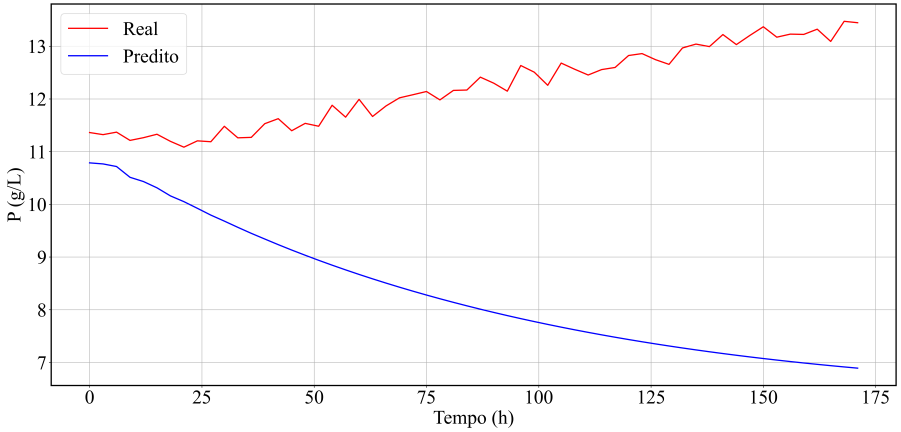
Figura 24 – Predições da RNN-LSTM para BD_3 em modo *offline*.



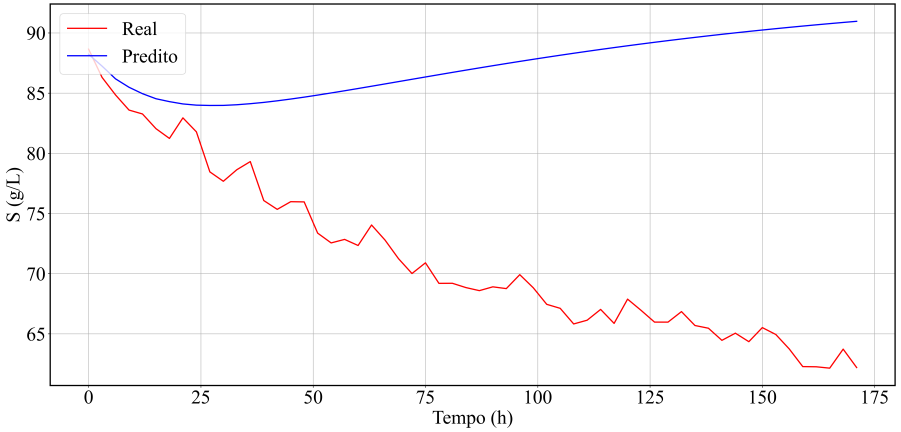
(a) Volume.



(b) Concentração celular.



(c) Concentração de produto.

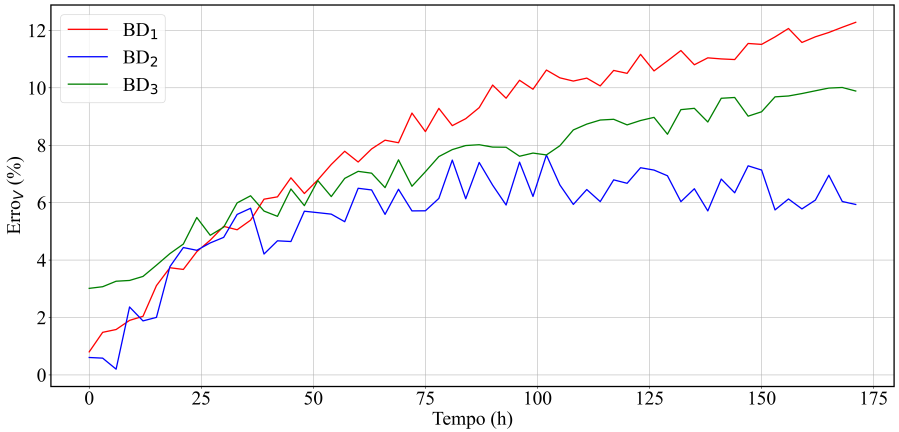


(d) Concentração de substrato.

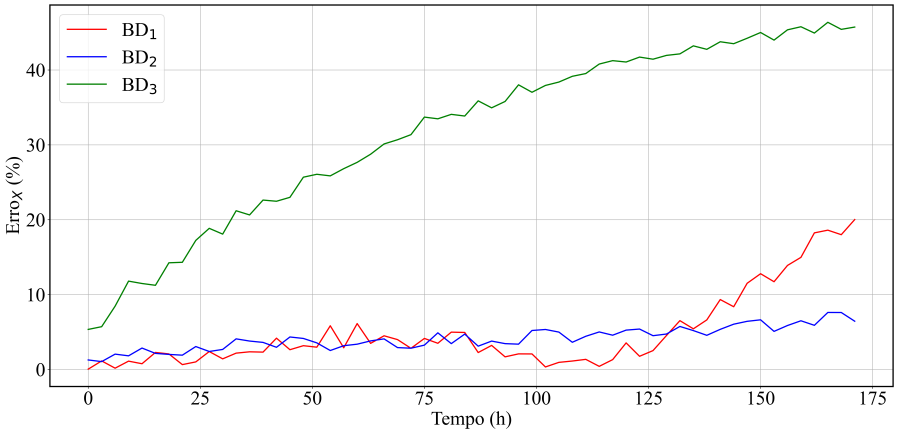
As razões para tais resultados são fundamentadas em dois pontos principais: todas as quatro variáveis endógenas da RNN-LSTM estão intrinsecamente interligadas, assim, os erros da predição de uma variável são usados nos cálculos de todas as variáveis, gerando um efeito cumulativo que degenera rapidamente a qualidade da resposta. Como comentado anteriormente, as variáveis X, S e P influenciam minimamente na resposta de V, sendo uma explicação plausível para o melhor desempenho da RNN-LSTM na predição desta variável. Além disso, o resultado indica que a relevância dada para as variáveis endógenas durante o cálculo das predições é extremamente alta. Durante a fase *online* em que os dados de entrada foram os disponíveis no banco de dados, ou seja, somente informação correta era apresentada para o cálculo, foi observado um erro pouco considerável das predições. Porém, durante o modo *offline* em que as informações de entrada começam a se distinguir das reais aliada ao efeito cumulativo comentado anteriormente, observa-se degeneração da qualidade da resposta após poucas predições, inviabilizando o emprego da RNN-LSTM obtida.

Para fins de comparação visual, os erros percentuais foram calculados e dispostos em gráficos temporais na Figura 25 para os três bancos de dados apresentados anteriormente. Como mencionado anteriormente, os erros apresentam uma tendência de crescimento ao longo do tempo devido à propagação dos erros calculados, especialmente para BD_3 devido às predições em sentidos opostos. Os erros percentuais chegam a valores máximos de 12,3; 46,0, 48,9 e 46,3% para V, X, P e S, respectivamente.

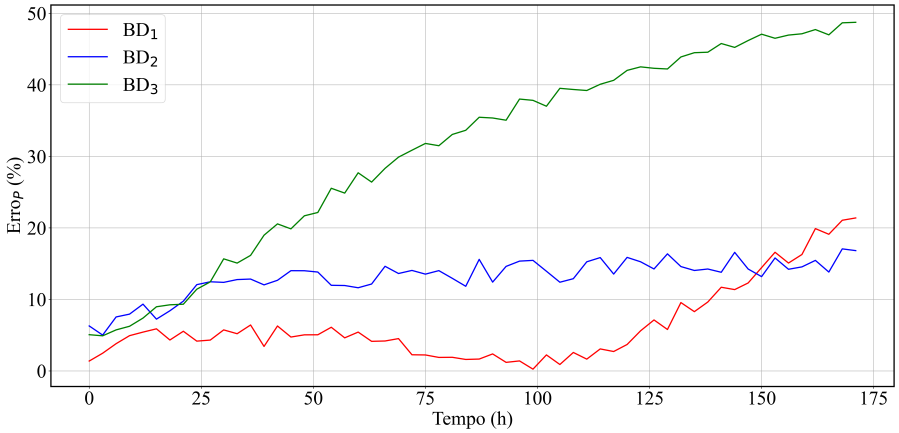
Figura 25 – Progressão dos erros relativos das predições feitas pela RNN-LSTM 15 em modo *offline*.



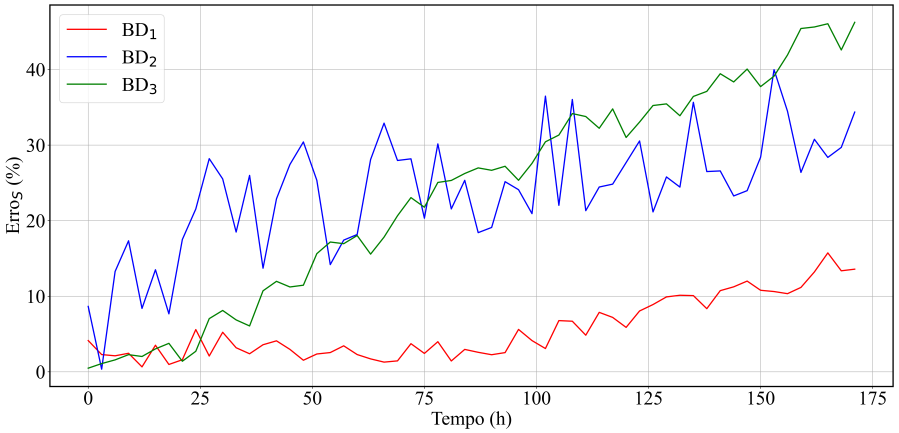
(a) Volume.



(b) Concentração celular.



(c) Concentração de produto.



(d) Concentração de substrato.

Com o intuito de verificar se o aumento da complexidade das redes recorrentes com topologia LSTM permitiria seu emprego em modo *offline* com desempenho aceitável, novas redes foram treinadas e testadas. Nesta etapa, somente foram modificados os números de neurônios da rede neural, de passos temporais e de unidades LSTM empilhadas. Na arquitetura de redes LSTM empilhadas, as saídas de uma unidade são entradas da unidade seguinte. De acordo com Kim *et al.* (2019), o problema de *vanishing gradients* pode ocorrer em RNN-LSTM empilhadas com mais de três unidades e, por isso, o número máximo deste hiperparâmetro foi ajustado para três durante a nova fase de treinamento.

Uma vez que foi verificado que a técnica de desligamento não contribuiu efetivamente para um melhor desempenho da maioria dos modelos de aprendizagem profunda na Seção 5.3, foi decidido não utilizá-la. Também não foram empregadas as regularizações L_1 e L_2 durante essa fase. Para o mesmo conjunto de hiperparâmetros, os pesos iniciais das redes seguiram três diferentes distribuições aleatórias a fim de avaliar os possíveis resultados do treinamento e obter a melhor configuração dos pesos. O treinamento seguiu o mesmo padrão usado anteriormente, empregando conjuntos de treinamento, validação e teste, além da parada antecipada. A ordem da repartição dos conjuntos e percentuais foram mantidos, assim como o número de falhas verificadas para a terminação da otimização dos pesos.

Para essa etapa, optou-se como critério de seleção a avaliação qualitativa da complexidade do modelo frente à melhoria do parâmetro de desempenho quantitativo MSE calculado para as predições feitas pelo novo modelo neural em modo *offline* para os três bancos de dados apresentados anteriormente. Ressalta-se que o aumento dos passos temporais empregados na predição reduz o número de vetores preditos, porém, a redução não chega a ser significativa para o cálculo dos parâmetros de desempenho. Os resultados obtidos para as novas configurações estão sumarizados na Tabela 12.

Tabela 12 – Melhores RNN-LSTM obtidas para operação em modo *offline*.

Rede	Unidades LSTM	*Timesteps*	Neurônios	MSE			
				BD_1	BD_2	BD_3	Total
1	1	3	30	$1{,}14 \cdot 10^{-1}$	$4{,}90 \cdot 10^{-3}$	$3{,}18 \cdot 10^{-3}$	$1{,}22 \cdot 10^{-1}$
2	1	3	40	$1{,}93 \cdot 10^{-1}$	$5{,}91 \cdot 10^{-3}$	$1{,}88 \cdot 10^{-3}$	$2{,}01 \cdot 10^{-1}$
3	1	3	50	$1{,}39 \cdot 10^{-1}$	$3{,}38 \cdot 10^{-3}$	$1{,}50 \cdot 10^{-2}$	$1{,}58 \cdot 10^{-1}$
4	2	3	30	$4{,}13 \cdot 10^{-2}$	$6{,}70 \cdot 10^{-3}$	$3{,}08 \cdot 10^{-2}$	$7{,}88 \cdot 10^{-2}$
5	2	3	40	$1{,}13 \cdot 10^{-2}$	$1{,}39 \cdot 10^{-3}$	$1{,}14 \cdot 10^{-3}$	$1{,}38 \cdot 10^{-2}$
6	2	3	50	$2{,}84 \cdot 10^{-3}$	$1{,}88 \cdot 10^{-3}$	$6{,}27 \cdot 10^{-3}$	$1{,}10 \cdot 10^{-2}$
7	3	3	30	$7{,}12 \cdot 10^{-3}$	$2{,}10 \cdot 10^{-3}$	$2{,}51 \cdot 10^{-3}$	$1{,}17 \cdot 10^{-2}$
8	3	3	40	$1{,}52 \cdot 10^{-2}$	$7{,}09 \cdot 10^{-4}$	$1{,}63 \cdot 10^{-3}$	$1{,}76 \cdot 10^{-2}$
9	3	3	50	$2{,}04 \cdot 10^{-1}$	$1{,}18 \cdot 10^{-2}$	$7{,}51 \cdot 10^{-2}$	$2{,}91 \cdot 10^{-1}$
10	1	5	30	$1{,}13 \cdot 10^{-2}$	$1{,}75 \cdot 10^{-3}$	$9{,}24 \cdot 10^{-4}$	$1{,}40 \cdot 10^{-2}$
11	1	5	40	$1{,}18 \cdot 10^{-3}$	$1{,}09 \cdot 10^{-3}$	$1{,}16 \cdot 10^{-2}$	$1{,}39 \cdot 10^{-2}$
12	1	5	50	$1{,}32 \cdot 10^{-2}$	$1{,}12 \cdot 10^{-3}$	$8{,}82 \cdot 10^{-4}$	$1{,}53 \cdot 10^{-2}$
13	2	5	30	$4{,}75 \cdot 10^{-3}$	$1{,}05 \cdot 10^{-3}$	$2{,}32 \cdot 10^{-3}$	$8{,}13 \cdot 10^{-3}$
14	2	5	40	$5{,}68 \cdot 10^{-3}$	$4{,}02 \cdot 10^{-3}$	$2{,}43 \cdot 10^{-3}$	$1{,}21 \cdot 10^{-2}$
15	2	5	50	$6{,}60 \cdot 10^{-3}$	$3{,}05 \cdot 10^{-4}$	$3{,}47 \cdot 10^{-3}$	$1{,}04 \cdot 10^{-2}$
16	3	5	30	$1{,}28 \cdot 10^{-2}$	$1{,}10 \cdot 10^{-3}$	$6{,}33 \cdot 10^{-3}$	$2{,}02 \cdot 10^{-2}$
17	3	5	40	$1{,}67 \cdot 10^{-2}$	$4{,}95 \cdot 10^{-4}$	$3{,}40 \cdot 10^{-3}$	$2{,}06 \cdot 10^{-2}$
18	3	5	50	$4{,}90 \cdot 10^{-3}$	$4{,}22 \cdot 10^{-3}$	$4{,}77 \cdot 10^{-3}$	$1{,}39 \cdot 10^{-2}$
19	1	7	30	$9{,}38 \cdot 10^{-3}$	$2{,}14 \cdot 10^{-3}$	$1{,}03 \cdot 10^{-3}$	$1{,}25 \cdot 10^{-2}$
20	1	7	40	$1{,}14 \cdot 10^{-2}$	$4{,}92 \cdot 10^{-4}$	$1{,}31 \cdot 10^{-3}$	$1{,}32 \cdot 10^{-2}$
21	1	7	50	$3{,}40 \cdot 10^{-3}$	$6{,}40 \cdot 10^{-4}$	$1{,}02 \cdot 10^{-3}$	$5{,}06 \cdot 10^{-3}$
22	**2**	**7**	**30**	$\mathbf{6{,}95 \cdot 10^{-4}}$	$\mathbf{4{,}40 \cdot 10^{-4}}$	$\mathbf{1{,}83 \cdot 10^{-4}}$	$\mathbf{1{,}32 \cdot 10^{-3}}$
23	2	7	40	$4{,}95 \cdot 10^{-3}$	$2{,}40 \cdot 10^{-4}$	$1{,}51 \cdot 10^{-3}$	$6{,}69 \cdot 10^{-3}$
24	2	7	50	$7{,}76 \cdot 10^{-3}$	$2{,}15 \cdot 10^{-3}$	$9{,}44 \cdot 10^{-4}$	$1{,}09 \cdot 10^{-2}$
25	3	7	30	$2{,}77 \cdot 10^{-3}$	$4{,}83 \cdot 10^{-3}$	$1{,}97 \cdot 10^{-3}$	$9{,}57 \cdot 10^{-3}$
26	3	7	40	$5{,}41 \cdot 10^{-3}$	$1{,}01 \cdot 10^{-3}$	$1{,}90 \cdot 10^{-3}$	$8{,}32 \cdot 10^{-3}$
27	3	7	50	$8{,}62 \cdot 10^{-3}$	$3{,}86 \cdot 10^{-3}$	$1{,}67 \cdot 10^{-2}$	$2{,}92 \cdot 10^{-2}$

Entre os novos modelos determinados, o que apresentou o melhor desempenho para aplicação em modo *offline* é composto por 30 neurônios na topologia, exemplos de entrada com 7 passos temporais e duas unidades LSTM (rede 22). A média do erro quadrático total determinada para esse modelo foi calculada em $1{,}32 \cdot 10^{-3}$, sendo aproximadamente 73,9% menor que a segunda melhor RNN-LSTM obtida (RNN-LSTM 21). Destaca-se que a RNN-LSTM 21 também corresponde à configuração com menor empilhamento, isto é, menor número de unidades LSTM agrupadas, porém, pelo critério de desempenho quantitativo já mencionado, optou-se por manter a RNN-LSTM 22 para os testes posteriores.

Também é interessante notar que as redes denominadas 23 e 24 cujos números de neurônios são progressivamente maiores que a RNN-LSTM 22 apresentaram crescimento dos valores de MSE. O acréscimo de neurônios representa um aumento da capacidade de aprendizagem das redes, entretanto, deve-se suprir este com banco de dados para treinamento maior a fim de

compensá-la. Caso contrário, existe a possibilidade de ocorrer falhas no treinamento e crescimento dos erros, fato que aparentemente ocorreu para as redes mencionadas. Em comparação ao modelo obtido na Seção 5.3, a redução do critério de desempenho quantitativo chega a duas ordens de grandeza para BD_3, e o novo valor calculado para o MSE total dos três bancos é 98% menor.

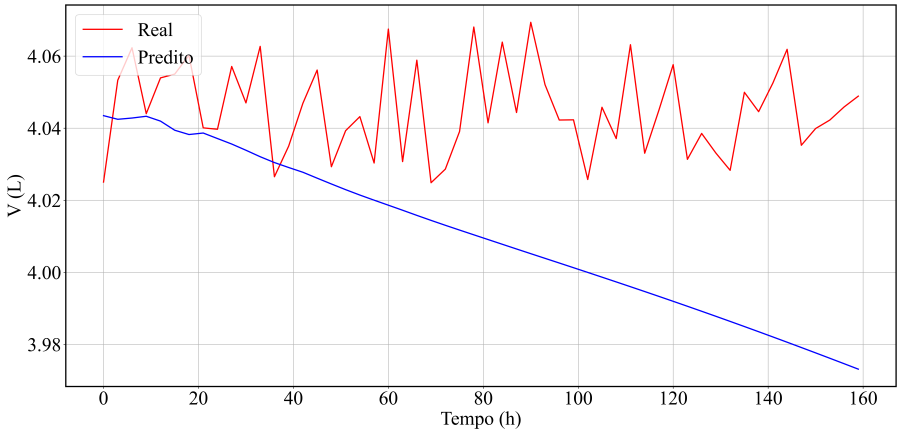
As predições dos comportamentos dinâmicos em modo *offline* empregando a RNN-LSTM 22 estão ilustradas nas Figuras 26 a 28 para os três bancos de dados já apresentados. Para o volume em regime permanente (BD_1 e BD_2), nota-se que as respostas da rede apresentam um perfil transiente, entretanto, a RNN-LSTM ainda foi capaz de atingir um regime permanente para o BD_2 distando aproximadamente 3,0% do valor médio para o regime permanente. Mesmo sem ocorrer estabilização para a predição de V para BD_1, os erros relativos percentuais não ultrapassam 2,0%, indicando uma boa congruência dos resultados.

Em relação às dinâmicas das outras variáveis de processo, os resultados obtidos por meio da RNN-LSTM 22 seguem adequadamente o comportamento esperado, especialmente para o início da simulação, com maiores disparidades para os pontos finais da simulação provavelmente devido ao acúmulo e à propagação dos erros por meio do cálculo recursivo e, para o BD_1, o efeito da extrapolação também deve ser considerado. Diferentemente do resultado gerado em modo *offline* por meio do modelo obtido na Seção 5.3, as predições feitas pela RNN-LSTM 22 reproduzem a tendência de crescimento/decréscimo das variáveis de processo disponíveis em BD_3, gerando uma redução considerável dos erros percentuais relativos ao longo da simulação.

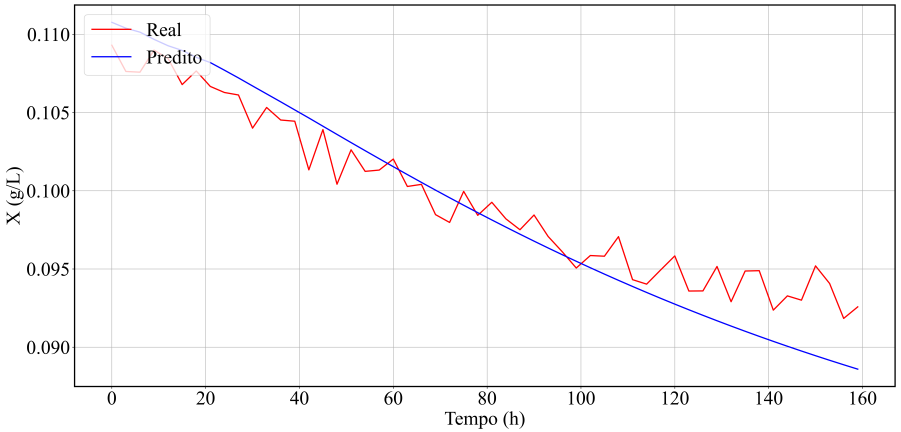
Os gráficos dos erros percentuais ao longo do tempo para os três bancos de dados simulados por meio da RNN-LSTM 22 estão ilustrados na Figura 29. Para as variáveis V, X, P e S, os valores de erros percentuais máximos calculados foram 3,8; 6,0, 8,8 e 12,0%, respectivamente, valores bastante inferiores aos 12,3; 46,0; 48,9 e 46,3% obtidos com a RNN-LSTM anterior que possuía uma arquitetura menos complexa. Novamente nota-se o crescimento dos erros ao longo do tempo, especialmente após 100 h de simulação, com aumento máximo de aproximadamente 4%. Entretanto, esse é consideravelmente menor que o observado para o modelo obtido na Seção 5.3, revelando que a maior complexidade do modelo trouxe melhorias significativas dos valores preditos.

Apesar de o erro percentual ainda atingir valores de 12,0%, considerando a natureza recursiva do cálculo, a interdependência das quatro variáveis endógenas e o emprego de sete passos temporais correspondentes a 21 h de informação real, a RNN-LSTM permite desenvolver os perfis transientes das quatro variáveis do modelo para 159 h com uma precisão próxima a dos sensores, indicando sua aplicabilidade na predição a curto prazo de processos químicos complexos. Assim, pode-se inferir que o aumento da capacidade de aprendizagem da RNN-LSTM possibilita o emprego desta estrutura em operações *offline* tais como controle de processos por meio de controladores de modelo preditivo ou como ferramenta potencial na gestão da cadeia de mantimentos e controle de qualidade. Entretanto, ressalta-se que o crescimento da capacidade de aprendizagem deve ser acompanhada de uma expansão do banco de dados para compensá-lo.

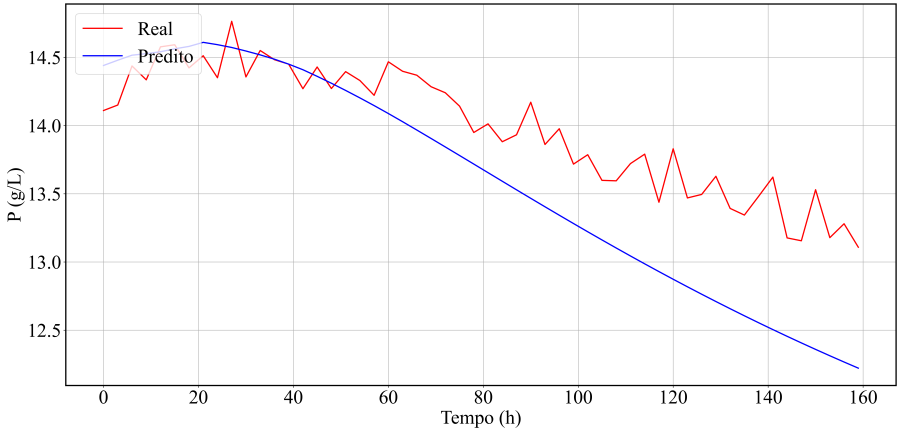
Figura 26 – Predições para BD_1 em modo *offline* aplicando a RNN-LSTM 22.



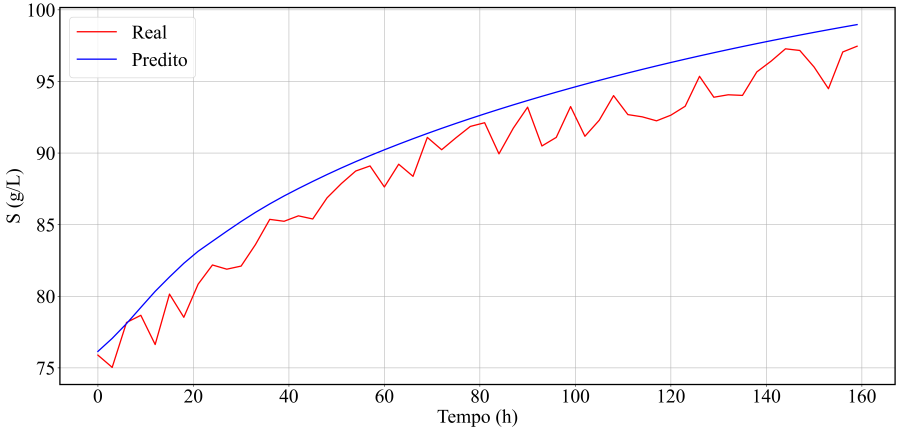
(a) Volume.



(b) Concentração celular.



(c) Concentração de produto.



(d) Concentração de substrato.

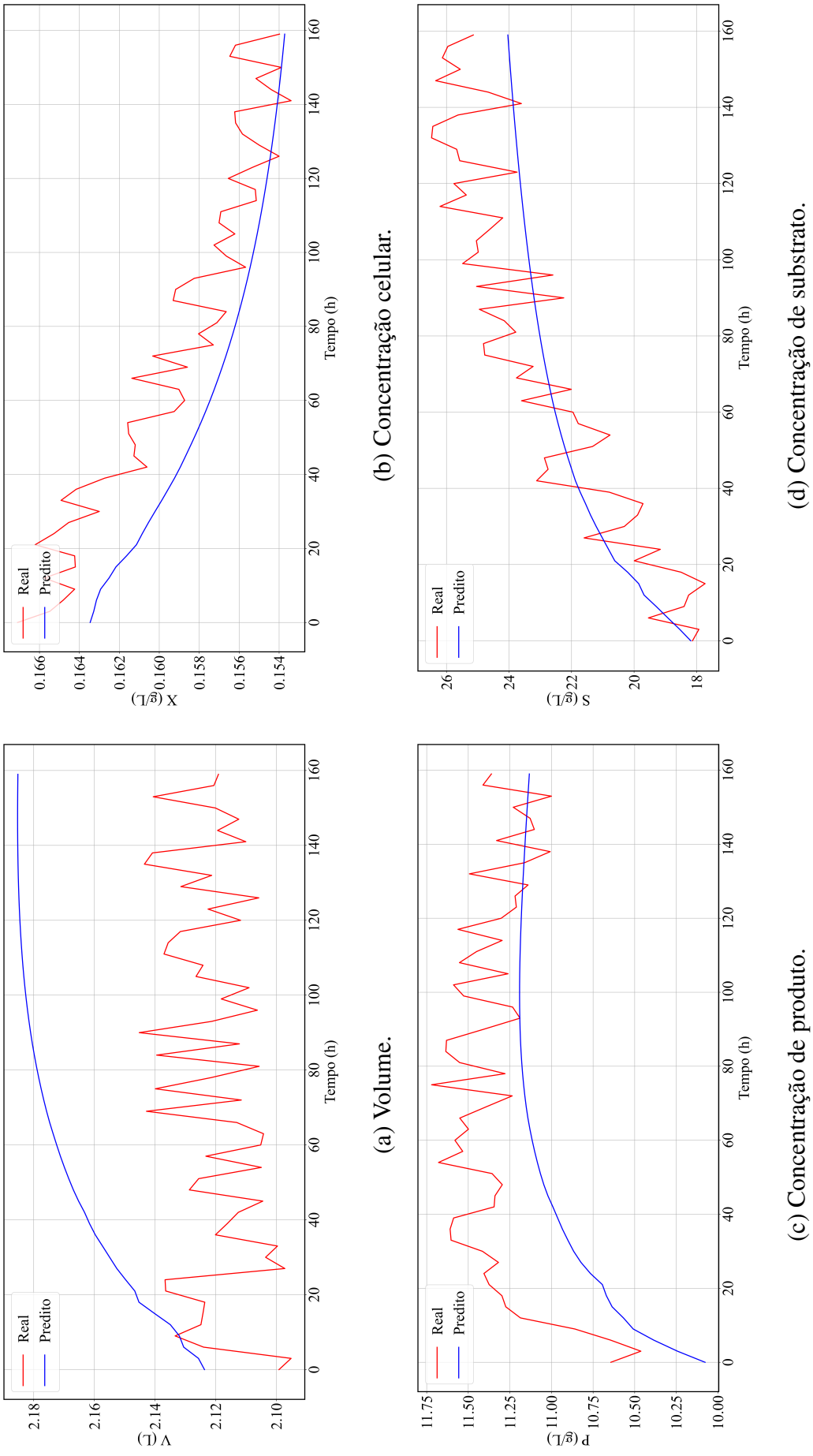


Figura 27 – Predições do BD_2 em modo *offline* aplicando a RNN-LSTM 22.

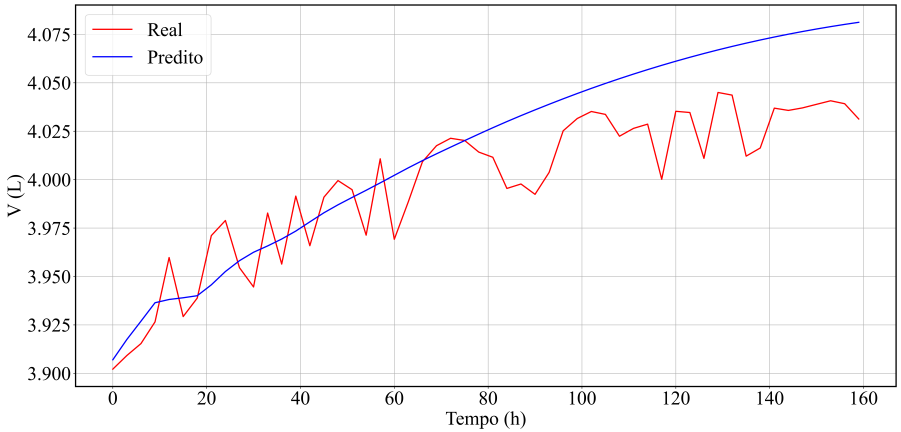
(a) Volume.

(b) Concentração celular.

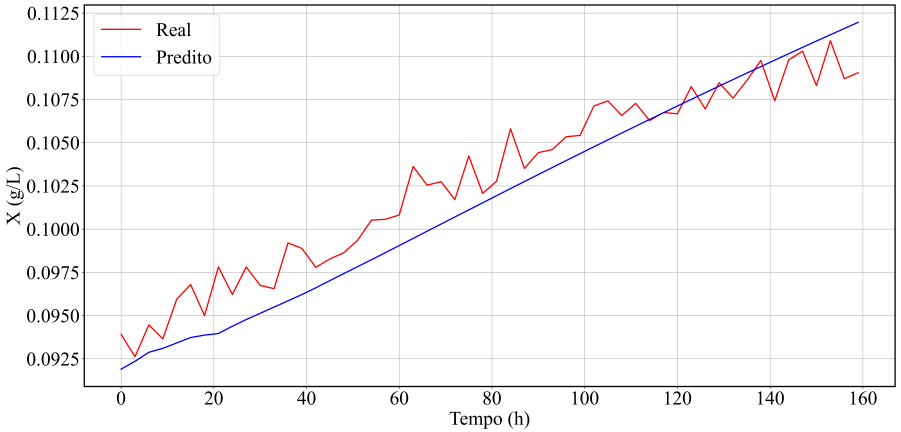
(c) Concentração de produto.

(d) Concentração de substrato.

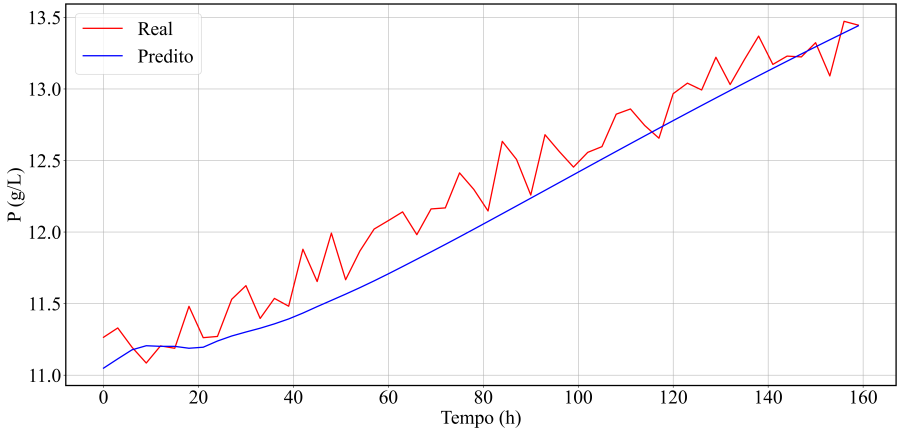
Figura 28 – Predições do BD_3 em modo *offline* aplicando a RNN-LSTM 22.



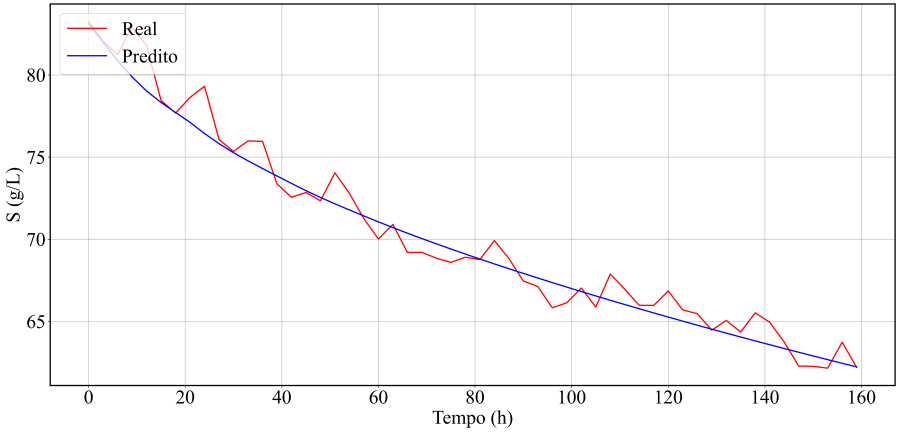
(a) Volume.



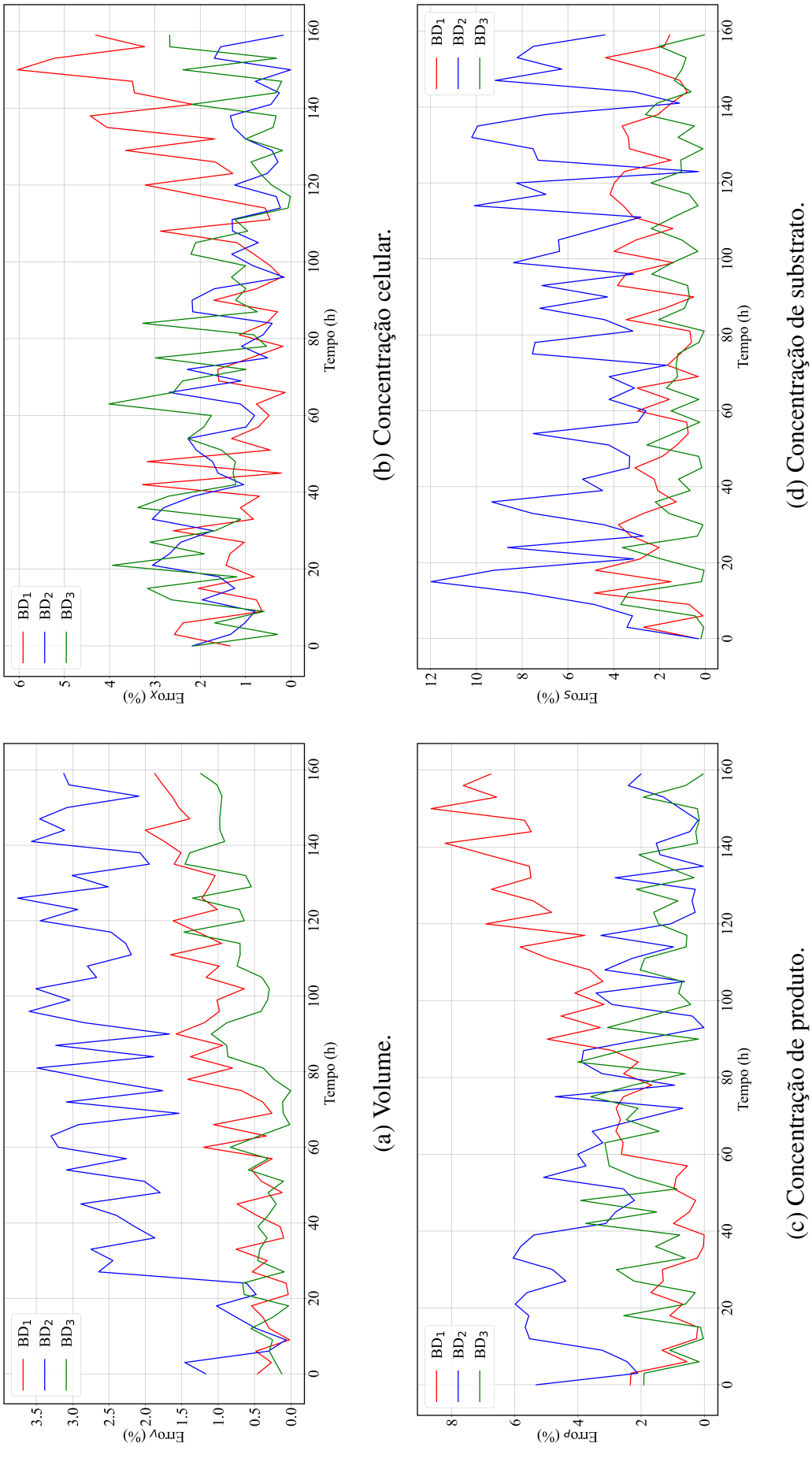
(b) Concentração celular.



(c) Concentração de produto.



(d) Concentração de substrato.



(a) Volume.

(b) Concentração celular.

(c) Concentração de produto.

(d) Concentração de substrato.

Figura 29 – Progressão dos erros relativos das predições feitas pela RNN-LSTM 22 em modo *offline*.

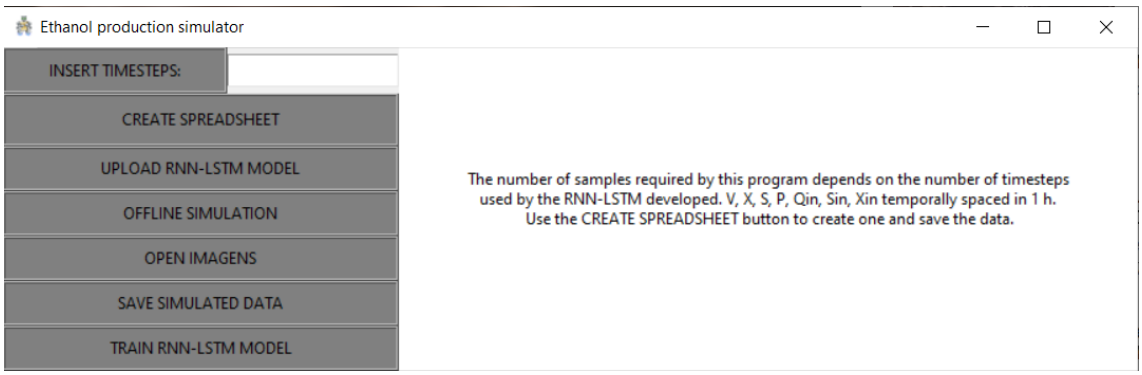
5.5 Interface gráfica do usuário

Toda a lógica de programação empregada para treinar e validar os modelos tanto em operação *online* quanto *offline* apresentadas nas Seções 5.3 e 5.4, respectivamente, foram adaptadas para a criação de uma interface gráfica de usuário. O objetivo principal do programa desenvolvido é gerar uma janela de predição de 150 h utilizando redes de aprendizagem profunda com topologia LSTM em modo de operação *offline* a partir de um estado inicial do sistema fornecido pelo usuário.

Na Figura 30 estão dispostas as principais janelas disponíveis na GUI para a utilização do programa. Na Figura 30 (a) está ilustrada a janela principal que permite ao usuário acessar todas as funcionalidades disponíveis. Para fazer a predição das variáveis de processo, deve-se indicar, a priori, o número de passos temporais empregados pelo modelo de aprendizagem profunda que deve ser carregado no programa. O usuário deve então informar os valores iniciais das variáveis endógenas e exógenas disponíveis em um banco de dados e salvá-las como ilustra a Figura 30 (b). Ressalva-se que essa etapa somente utiliza bancos de dados em planilha Excel®. Os resultados da simulação *offline* podem ser conferidos graficamente como indicado na Figura 30 (c) ou salvos em uma planilha Excel® para utilização posterior.

Caso o usuário deseje gerar outra RNN-LSTM com base no próprio banco de dados, existe a opção de treinamento no programa tendo como única restrição o intervalo de amostragem dos dados que deve ser 3 h. É permitido ao usuário definir os hiperparâmetros *timestep*, número de neurônios, pesos das regularizações L_1 e L_2, percentual de desligamento, distribuição inicial dos pesos, razão de separação dos dados de treinamento e teste, número de épocas, *batch size*, percentual de conjunto de validação e número de falhas para parada antecipada como indicado na 30 (d). Em caso de dúvidas, o usuário pode consultar a janela de ajuda indicada na Figura 30 (e). Para o treinamento, são verificados os desempenhos da rede empregando os conjuntos de treinamento e de teste, além do teste em modo *offline*, tendo como parâmetro quantitativo o MSE. Gráficos comparativos dos dados reais e das predições para os três conjuntos podem ser obtidos, além da função custo ao longo do treinamento como visto na Figura 30 (f).

Figura 30 – Interface gráfica do usuário.



(a) Janela principal.

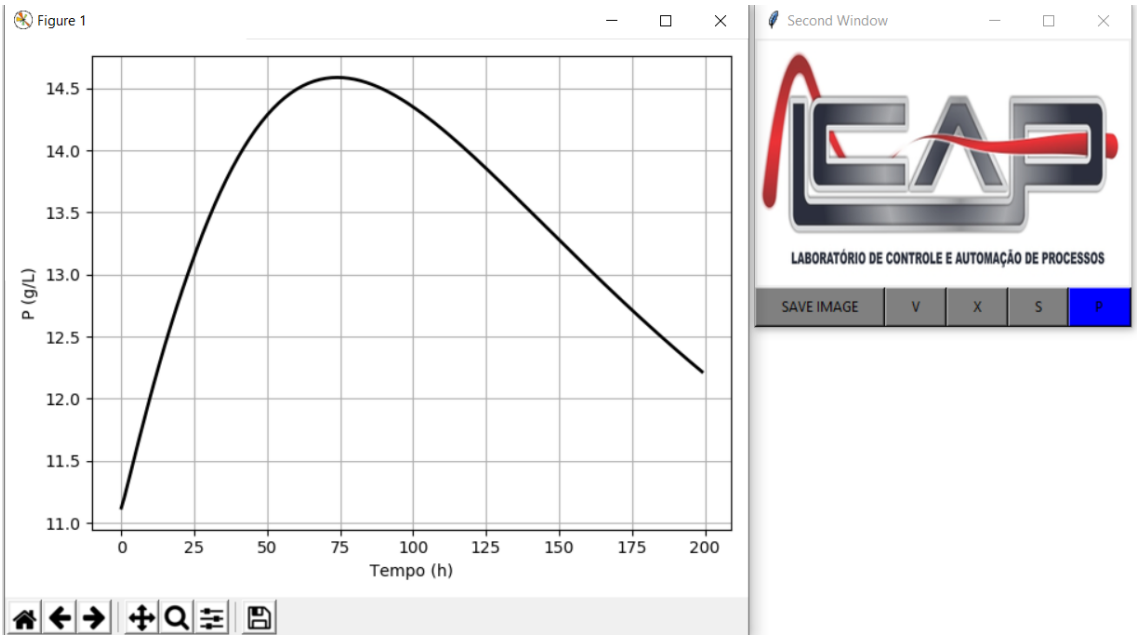
Figura 30 – Interface gráfica do usuário.

DATA WINDOW

V (L)	X (g/L)	S (g/L)	P (g/L)	Qin (L/h)	Sin (g/L)	Xin (g/L)
4.53752893050155	0.108198269432709	1.68212858654039	8.4098231988164	0.1086	50.0	0.055
4.52589875570734	0.107859960712826	2.37590216725086	8.53813050314143	0.0969	100.0	0.025
4.51440710441429	0.107772811054769	3.05404217679561	8.66466626117752	0.0969	100.0	0.025
4.50305249976402	0.107855718192012	3.7168646909253	8.78943675986211	0.0969	100.0	0.025
4.49183347696463	0.10805801170852	4.3644711139687	8.9124952099859	0.0969	100.0	0.025
4.48074858327265	0.108346032848567	4.99691740249292	9.03390439030751	0.0969	100.0	0.025
4.46979637794527	0.108696351585026	5.61427553685396	9.15372302290335	0.0969	100.0	0.025
4.45897543223028	0.109092074742081	6.216653862058	9.27200123791093	0.0969	100.0	0.025
4.44828432933274	0.109520666552536	6.80420157431401	9.38877962619545	0.0969	100.0	0.025
4.43772166438016	0.10997260615594	7.37710646817031	9.5040896735268	0.0969	100.0	0.025
4.42728604444412	0.110440531295573	7.93559068589567	9.61795467762582	0.0969	100.0	0.025
4.41697608836122	0.110918642768338	8.47990525050135	9.73039101763532	0.0969	100.0	0.025
4.40679042676045	0.111402321382995	9.01032556796918	9.84140901473639	0.0969	100.0	0.025
4.39672770225224	0.111887865848272	9.52714738596505	9.95101390719279	0.0969	100.0	0.025
4.38678656908338	0.11237226147138	10.0306820002683	10.0592068736611	0.0969	100.0	0.025
4.37696569322405	0.112853044449874	10.5212532917315	10.1659856473184	0.0969	100.0	0.025
4.36726375232805	0.113328190637 4	10.9991947640706	10.2713451981549	0.0969	100.0	0.025
4.35767943563143	0.113796029347771	11.4648468002635	10.3752783509409	0.0969	100.0	0.025
4.34821144421525	0.114255184351213	11.9185549065873	10.4777760903801	0.0969	100.0	0.025
4.33885849038456	0.114704510453876	12.3606671936284	10.578828128495	0.0969	100.0	0.025
4.32961929810401	0.115143063642997	12.7915332867299	10.6784232522637	0.0969	100.0	0.025
4.32049260267805	0.115570064052021	13.211502829729	10.776549512486	0.0969	100.0	0.025
4.31147715101375	0.115984873975321	13.620924368972	10.8731945091057	0.0969	100.0	0.025
4.30257170132004	0.116386973149555	14.0201441203677	10.9683456363993	0.0969	100.0	0.025

GET DATA FROM AN EXCEL SPREADSHEET | CREATE INITIAL EXCEL SPREADSHEET | SAVE DATA FOR SIMULATION

(b) Obtenção de dados iniciais para simulação *offline*.



(c) Gráficos da simulação *offline*.

Figura 30 – Interface gráfica do usuário.

TRAINING RNN-LSTM MODEL

TIMESTEPS	24	
NEURONS	200	
L1	.0001	
L2	.0001	
DROPOUT	0	
INIT_MODE	random_uniform	
TRAIN-TEST SPLIT	.8	
EPOCHS	300	
BATCH SIZE	32	
VALIDATION SPLIT	.1	
PATIENCE	10	
Train RNN-LSTM model	Save RNN-LSTM model	HELP
Train MSE Test MSE Offline MSE	0.0001328783996198061 0.0002842425523306452 0.008696070425796905	SHOW TRAIN RESULTS

(d) Janela de treinamento da RNN-LSTM.

HELP

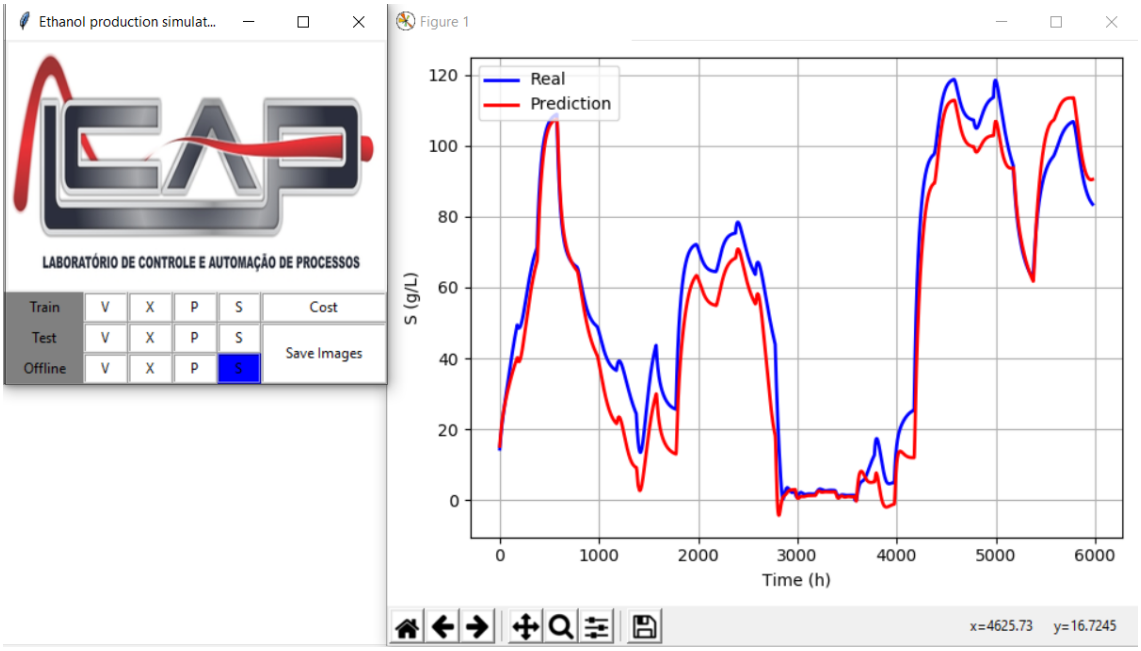
Timesteps	Number of temporal data used as one example, it must be an integer
Neurons	Number of neurons in each LSTM gate, it must be an integer
L1	Weight for L1 regularization, it must be a float
L2	Weight for L2 regularization, it must be a float
Dropout	Percentage of dropout, it must be a float between 0 and 1
Init_mode	Initial weight distribution, it must be a string viable options: random_uniform, random_normal, lecun_uniform, glorot_normal, ones, glorot_uniform, zeros, identity
Train-Test split	Percentage of dataset used for training, it must be a float between 0 and 1
Epochs	Number of epochs for training, it must be an integer
Batch size	Number of samples used at one optimization, it must be an integer
Validation split	Percentage of training dataset used for validation, it must be a float between 0 and 1
Patience	Number of failures for early stopping, it must be an integer

Excel spreadsheet must look like the image below

	A	B	C	D	E	F	G	H
1	t (h)	V (L)	X (g/L)	S (g/L)	P (g/L)	Qi (dm3/h)	Si (g/L)	Xi (g/L)
2	0	4.537529	0.108198	1.682129	8.409823	0.1086	50	0.055
3	1	4.525899	0.10786	2.375902	8.538131	0.0969	100	0.025
4	2	4.514407	0.107773	3.054042	8.664666	0.0969	100	0.025
5	3	4.503052	0.107856	3.716865	8.789437	0.0969	100	0.025
6	4	4.491833	0.108058	4.364471	8.912495	0.0969	100	0.025
7	5	4.480749	0.108346	4.996917	9.033904	0.0969	100	0.025
8	6	4.469796	0.108696	5.614276	9.153723	0.0969	100	0.025
9	7	4.458975	0.109092	6.216654	9.272001	0.0969	100	0.025
10	8	4.448284	0.109521	6.804202	9.38878	0.0969	100	0.025
11	9	4.437722	0.109973	7.377106	9.50409	0.0969	100	0.025
12	10	4.427286	0.110441	7.935591	9.617955	0.0969	100	0.025

(e) Janela de ajuda para treinamento.

Figura 30 – Interface gráfica do usuário.



(f) Gráficos dos resultados das diferentes etapas de treinamento.

6 Conclusões

No presente trabalho, foram desenvolvidos modelos empíricos baseados em redes de aprendizagem profunda com topologia *Long Short-Term Memory* para predizer o comportamento dinâmico de um processo fermentativo alcoólico contínuo em que os parâmetros cinéticos são variantes a depender das condições operacionais.

Redes neurais artificiais foram obtidas para determinar o comportamento dos parâmetros cinéticos do modelo fermentativo de acordo com as condições operacionais. A estratégia de validação cruzada com *k-fold* igual a 5 foi empregada para avaliar o desempenho das redes devido ao pequeno volume disponível de dados experimentais. A melhor topologia encontrada consistia em uma rede com uma camada oculta, 15 neurônios, função de ativação *softsign* e sem desligamento e regularizações L_1 e L_2 durante a fase de treinamento. O erro médio quadrático deste modelo foi de $0{,}0459 \pm 0{,}0274$. O teste de regressão indica que o modelo neural apresenta um desempenho pouco aquém do desejável, com coeficientes de determinação entre 0,786 e 0,896.

Para a obtenção do banco de dados do processo, foi simulada uma operação contínua de 6.000 h com modificações nas variáveis de entrada a cada 200 h, permitindo uma descrição do comportamento dinâmico do sistema. Notou-se que os valores dos parâmetros cinéticos chegaram a variar em 65% em algumas horas de simulação, com variações bruscas devido à entrada degrau da concentração de substrato da corrente de alimentação.

Em relação ao comportamento dinâmico das variáveis estudadas, verificou-se que, excetuando o volume do biorreator, as variáveis de processo não atingiram um novo regime permanente no intervalo de 200 h entre as mudanças, possivelmente pela relação intrínseca das variáveis e constantes modificações dos valores dos parâmetros cinéticos. Além disso, foi observada a presença de resposta inversa das concentrações frente a variações da vazão de alimentação, possivelmente advindas do efeito de diluição causado com a variação do volume reacional.

O banco de dados obtido foi dividido de modo sequencial com uma razão 80/20 para otimizar os parâmetros das redes recorrentes e validar os modelos por meio dos denominados conjuntos de treinamento e teste, respectivamente. Reservou-se 20% dos vetores do banco de treinamento para aplicar a técnica de parada antecipada a fim de reduzir o tempo total da etapa. Foi aplicada uma busca automática para avaliar 6.750 topologias diferentes de RNN-LSTM ao variar 7 hiperparâmetros, e empregado o MSE para o conjunto de teste como parâmetro de desempenho devido ao caráter de generalização implicado neste.

Entre as configurações de RNN-LSTM testadas, o menor MSE ($2{,}21 \cdot 10^{-4}$) foi observado para o modelo com 3 passos temporais, 20 neurônios nas camadas ocultas dos portões, distribuição inicial de pesos identidade, *dropout* nulo e regularizações L_1 e L_2 com pesos iguais a 10^{-4} e 0, respectivamente. Os testes de regressão aplicando os dados de treinamento apresentaram

alta correlação linear, com valores de coeficiente de determinação próximos à unidade. Os coeficientes angulares calculados foram muito próximos a um e os lineares praticamente nulos.

Para avaliar se a RNN-LSTM apresentava problemas de repetição dos valores das entradas como predições, foi analisado os comportamentos das variáveis de saída previstas frentes aos dados reais para faixas reduzidas. Notou-se que a resposta da rede apresentava evidentes variações em relação aos dados reais tais como diferentes amplitudes na resposta, sobreposições e inversões das curvas.

Utilizando o conjunto de teste, notou-se que a RNN-LSTM gerada foi capaz de generalizar adequadamente o comportamento dinâmico de todas as variáveis do processo fermentativo, com excelentes resultados para os testes de regressão. Assim, conclui-se que a topologia de redes recorrentes LSTM apresentam empregabilidade para a descrição de modelos fermentativos quando em operação *online*.

Para a avaliação do uso de modelos recursivos empregando a topologia LSTM, a melhor rede em modo *online* foi utilizada para predizer o comportamento do sistema frente a variações das entradas exógenas dado somente um estado inicial para três bancos de dados descontínuos entre si. Os resultados indicaram que a rede somente foi capaz de predizer adequadamente a dinâmica do processo fermentativo para poucas horas de simulação, enquanto que os valores do volume reacional calculados para o regime permanente apresentaram grandes diferenças do valor real. Desta forma, a aplicação deste modelo não é satisfatória para o cálculo *offline*.

Foi avaliada se uma maior complexidade do modelo permitiria o emprego dos cálculos recursivos, e novas redes recorrentes LSTM foram treinadas. Para essa etapa, somente foram variados os números de passos temporais, de neurônios e de unidades LSTM empilhadas, tomando como parâmetro de desempenho o MSE calculado usando a predição em modo *offline*. Entre as novas redes, a que apresentou melhor desempenho com menor complexidade consistia em 7 *timesteps*, 30 neurônios e 2 unidades LSTM, obtendo um MSE total para os três bancos de dados de $1{,}32 \cdot 10^{-3}$.

O novo modelo empírico foi capaz de predizer todas as variáveis de processo dentro de limites aceitáveis tanto para o regime transiente quanto permanente. Notou-se que para a faixa final da simulação, a predição acompanha relativamente bem o comportamento dinâmico, mas apresenta certas diferenças nos valores, possivelmente originadas da propagação do erros durante os cálculos recursivos e da extrapolação dos valores das variáveis exógenas para BD_1. Entretanto, os resultados ainda estão dentro da faixa de precisão de sensores comerciais, indicando uma boa qualidade das predições e possível aplicação do modelo *offline* como ferramenta de gerenciamento de mantimentos e qualidade por meio de controladores de modelo preditivo.

Toda a lógica desenvolvida para o treinamento das RNN-LSTM feita neste trabalho foi adaptada para a criação de uma interface gráfica de utilizador. O objetivo principal do programa é gerar uma simulação com 200 h para as quatro variáveis estudadas no processo fermentativo aplicando o modo de operação *offline* e um único ponto inicial. O modelo neural empregado pode ser desenvolvido no próprio aplicativo por meio da janela de treinamento, permitindo ao usuário escolher vários hiperparâmetros da rede LSTM. O desempenho do treinamento pode ser

conferido graficamente ou por meio do MSE para os conjuntos de treinamento e teste, além de uma simulação *offline* feita para comparação com o banco de dados disponível.

6.1 Sugestões para trabalhos futuros

Os resultados obtidos neste projeto demostram a possibilidade de empregar modelos empíricos baseados em redes recorrentes com topologia *Long Short-Term Memory* na descrição do comportamento dinâmico do processo de fermentação alcoólica aplicando tanto o modo *online* quanto o *offline*. Sugere-se como complemento do estudo:

- a aplicação de banco de dados experimental envolvendo variáveis não presentes no modelo, mas que influenciam no processo tais como pH, taxa de agitação, vazão de gases;
- o emprego de estratégias de *deep reinforcement learning* para constante melhoria do modelo recorrente;
- a atualização da interface gráfica do usuário para permitir o re-treinamento das redes LSTM por meio da estratégia de *deep reinforcement learning*.
- o emprego da RNN-LSTM como ferramenta de predição para estratégias de controle preditivo.

Referências Bibliográficas

ABDEL-NASSER, M.; MAHMOUD, K. Accurate photovoltaic power forecasting models using deep LSTM-RNN. *Neural Computing and Applications*, Springer Science and Business Media LLC, v. 31, n. 7, p. 2727–2740, 2017.

AGÊNCIA NACIONAL DE PETRÓLEO, GÁS NATURAL E BIOCOMBUSTÍVEIS. *Informações de mercado*. 2020. Disponível em: <http://www.anp.gov.br/producao-de-biocombustiveis/etanol/informacoes-mercado-etanol>. Acesso em: 13/04/2020.

AGGARWAL, C. *Neural networks and deep learning : a textbook*. Cham, Switzerland: Springer, 2018. 497 p.

AHMADIAN-MOGHADAM, H.; ELEGADO, F. B.; NAYVE, R. Prediction of ethanol concentration in biofuel production using artificial neural networks. *American Journal of Modeling and Optimization*, Science and Education Publishing, v. 1, p. 31–35, jul. 2013.

ALTHELAYA, K. A.; EL-ALFY, E.-S. M.; MOHAMMED, S. Stock market forecast using multivariate analysis with bidirectional and stacked (LSTM, GRU). In: *2018 21st Saudi Computer Society National Computer Conference (NCC)*. [S.l.]: IEEE, 2018.

AN, G. The effects of adding noise during backpropagation training on a generalization performance. *Neural Computation*, v. 8, n. 3, p. 643–674, 1996.

ANDRADE, E. T. de; CARVALHO, S. R. G. de; SOUZA, L. F. de. Programa do Proálcool e o etanol no Brasil. *Engevista*, Pró Reitoria de Pesquisa, Pós Graduaão e Inovaão - UFF, v. 11, n. 2, 2010.

BETIKU, E.; TAIWO, A. E. Modeling and optimization of bioethanol production from breadfruit starch hydrolyzate vis-à-vis response surface methodology and artificial neural network. *Renewable Energy*, Elsevier BV, v. 74, p. 87–94, 2015.

BISHOP, C. M. Training with noise is equivalent to tikhonov regularization. *Neural Computation*, MIT Press - Journals, v. 7, n. 1, p. 108–116, jan. 1995.

BROWNLEE, J. *Difference between a batch and an epoch in a neural network*. 2019. Disponível em: <https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/>. Acesso em: 12/03/2020.

BROWNLEE, J. *How to Make Baseline Predictions for Time Series Forecasting with Python*. 2019. Disponível em: <https://machinelearningmastery.com/persistence-time-series-forecasting-with-python/>. Acesso em: 20/05/2020.

CHRISTENSON, L. B.; SIMS, R. C. Rotating algal biofilm reactor and spool harvester for wastewater treatment with biofuels by-products. *Biotechnology and Bioengineering*, Wiley, v. 109, n. 7, p. 1674–1684, 2012.

DREYFUS, G. *Neural Networks : Methodology and Applications*. Berlin, Heidelberg: Springer-Verlag Berlin Heidelberg, 2005.

ENCYCLOPÆDIA BRITANNICA. *Biotechnology*. 2019. Disponível em: <https://www.britannica.com/technology/biotechnology>. Acesso em: 15/05/2020.

ESFAHANIAN, M.; NIKZAD, M.; NAJAFPOUR, G.; GHOREYSHI, A. Modeling and optimization of ethanol fermentation using saccharomyces cerevisiae: Response surface methodology and artificial neural network. *Chemical Industry and Chemical Engineering Quarterly*, National Library of Serbia, v. 19, n. 2, p. 241–252, jan. 2013.

GHALY, A.; EL-TAWEEL, A. Kinetic modelling of continuous production of ethanol from cheese whey. *Biomass and Bioenergy*, Elsevier BV, v. 12, n. 6, p. 461–472, jan. 1997.

GHALY, A.; KAMAL, M.; CORREIA, L. Kinetic modelling of continuous submerged fermentation of cheese whey for single cell protein production. *Bioresource Technology*, Elsevier BV, v. 96, n. 10, p. 1143–1152, jul. 2005.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep learning*. Cambridge, Massachusetts: The MIT Press, 2016. 775 p.

GRAHOVAC, J.; JOKIĆ, A.; DODIĆ, J.; VUČUROVIĆ, D.; DODIĆ, S. Modelling and prediction of bioethanol production from intermediates and byproduct of sugar beet processing using neural networks. *Renewable Energy*, Elsevier BV, v. 85, p. 953–958, jan. 2016.

HAYKIN, S. *Neural networks and learning machines*. New York: Prentice Hall/Pearson, 2009. 906 p.

HEATON, J. *Artificial intelligence for humans*. St. Louis, MO: Heaton Research, Inc, 2015. 374 p.

HETTIARACHCHI, P.; HALL, M.; MINNS, A. The extrapolation of artificial neural networks for the modelling of rainfallârunoff relationships. *Journal of Hydroinformatics*, IWA Publishing, v. 7, n. 4, p. 291–296, 2005.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. *Neural Computation*, MIT Press - Journals, v. 9, n. 8, p. 1735–1780, nov. 1997.

IRKIN, R. Natural fermented beverages. In: *Natural Beverages*. [S.l.]: Elsevier, 2019. p. 399–425.

KERAS. *Keras documentation: Keras API reference*. n.d. Disponível em: <https://keras.io/api/>. Acesso em: 17/10/2019.

KESKAR, N. S.; MUDIGERE, D.; NOCEDAL, J.; SMELYANSKIY, M.; TANG, P. T. P. *On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima*. 2017.

KHANDELWAL, R. *L1 and L2 Regularization*. Data Driven Investor, 2019. Disponível em: <https://medium.com/datadriveninvestor/l1-l2-regularization-7f1b4fe948f2>. Acesso em: 17/04/2020.

KIM, H.; PARK, M.; KIM, C. W.; SHIN, D. Source localization for hazardous material release in an outdoor chemical plant via a combination of LSTM-RNN and CFD simulation. *Computers & Chemical Engineering*, Elsevier BV, v. 125, p. 476–489, jun. 2019.

KUHN, M. *Applied predictive modeling*. New York: Springer, 2013. 613 p.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. *Nature*, Springer Science and Business Media LLC, v. 521, n. 7553, p. 436–444, mai 2015.

LINKO, P.; ZHU, Y.-H. Neural network modelling for real-time variable estimation and prediction in the control of glucoamylase fermentation. *Process Biochemistry*, Elsevier BV, v. 27, n. 5, p. 275–283, set 1992.

LOPES, A. C. A.; EDA, S. H.; ANDRADE, R. P.; AMORIM, J. C.; DUARTE, W. F. New alcoholic fermented beverages—potentials and challenges. In: *Fermented Beverages*. [S.l.]: Elsevier, 2019. p. 577–603.

MA, X.; TAO, Z.; WANG, Y.; YU, H.; WANG, Y. Long short-term memory neural network for traffic speed prediction using remote microwave sensor data. *Transportation Research Part C: Emerging Technologies*, Elsevier BV, v. 54, p. 187–197, 2015.

MANDIC, D. *Recurrent neural networks for prediction : learning algorithms, architectures, and stability*. Chichester New York: John Wiley, 2001.

MARSLAND, S. *Machine Learning : an algorithmic perspective*. Boca Raton, FL: CRC Press, 2015. 437 p.

MITTAL, A. Biological wastewater treatment. *Water Today*, p. 32–44, 2011.

MOHAMED, M. S.; TAN, J. S.; MOHAMAD, R.; MOKHTAR, M. N.; ARIFF, A. B. Comparative analyses of response surface methodology and artificial neural network on medium optimization forTetraselmissp. FTC209 grown under mixotrophic condition. *The Scientific World Journal*, Hindawi Limited, v. 2013, p. 1–14, jul. 2013.

NAGY, Z. K. Model based control of a yeast fermentation bioreactor using optimally designed artificial neural networks. *Chemical Engineering Journal*, Elsevier BV, v. 127, n. 1-3, p. 95–109, mar 2007.

NIELSEN, M. A. *Neural Networks and Deep Learning*. Determination Press, 2015. Disponível em: <http://neuralnetworksanddeeplearning.com>. Acesso em: 08/10/2019.

OLIVEIRA, S. C.; STREMEL, D. P.; DECHECHI, E. C.; PEREIRA, F. M. Kinetic modeling of 1-g ethanol fermentations. In: *Fermentation Processes*. [S.l.]: InTech, 2017.

OTHMAN, I.; ANUAR, A. N.; UJANG, Z.; ROSMAN, N. H.; HARUN, H.; CHELLIAPAN, S. Livestock wastewater treatment using aerobic granular sludge. *Bioresource Technology*, Elsevier BV, v. 133, p. 630–634, 2013.

PATTERSON, J.; GIBSON, A. *Deep learning : a practitioner's approach*. Sebastopol, CA: O'Reilly, 2017. 507 p.

QING, X.; NIU, Y. Hourly day-ahead solar irradiance prediction using weather forecasts by LSTM. *Energy*, Elsevier BV, v. 148, p. 461–468, 2018.

SHRESTHA, A.; MAHMOOD, A. Review of deep learning algorithms and architectures. *IEEE Access*, Institute of Electrical and Electronics Engineers (IEEE), v. 7, p. 53040–53065, 2019.

SINGH, B. P.; KUMAR, P.; HAQUE, S.; JAWED, A.; DUBEY, K. K. Improving production of tacrolimus in streptomyces tacrolimicus (ATCC 55098) through development of novel mutant by dual mutagenesis. *Brazilian Archives of Biology and Technology*, FapUNIFESP (SciELO), v. 60, n. 0, ago. 2017.

SOUZA, A. C. O. e. *Modelagem empírica do comportamento dinâmico de um protótipo de flotação por ar dissolvido baseado na aplicação de redes neurais artificiais*. 109 p. Dissertação (Mestrado em Engenharia Química) — Universidade Estadual de Campinas, Campinas, 2019.

TAHERZADEH, M. J.; KARIMI, K. Fermentation inhibitors in ethanol processes and different strategies to reduce their effects. In: *Biofuels*. [S.l.]: Elsevier, 2011. p. 287–311.

TENSORFLOW. *Introduction to TensorFlow*. n.d. Disponível em: <https://www.tensorflow.org/learn>. Acesso em: 17/10/2019.

VERVERIS, C.; CHRISTODOULAKIS, N.; SANTAS, R.; SANTAS, P.; GEORGHIOU, K. Effects of municipal sludge and treated waste water on biomass yield and fiber properties of kenaf (hibiscus cannabinus l.). *Industrial Crops and Products*, Elsevier BV, v. 84, p. 7–12, jun. 2016.

WÖLLMER, M.; METALLINOU, A.; EYBEN, F.; SCHULLER, B.; NARAYANAN, S. Context-sensitive multimodal emotion recognition from speech and facial expression using bidirectional lstm modeling. In: KOBAYASHI, T.; HIROSE, K.; NAKAMURA, S. (Ed.). Japão: Proceedings of the 11th Annual Conference of the International Speech Communication Association, INTERSPEECH, 2010. p. 2362–2365.

ZHAO, H.; SUN, S.; JIN, B. Sequential fault diagnosis based on LSTM neural network. *IEEE Access*, Institute of Electrical and Electronics Engineers (IEEE), v. 6, p. 12929–12939, 2018.

APÊNDICE A – RNA para parâmetros cinéticos

```
########## Libraries ##########

import numpy as np
import pandas as pd
import keras
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras.callbacks import EarlyStopping
from keras.callbacks import ModelCheckpoint
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import GridSearchCV
from keras.wrappers.scikit_learn import KerasRegressor
from sklearn.metrics import mean_squared_error, r2_score

########## Parameters to modify ##########

no = 3  # number of outputs
ni = 2  # number of input variables
en = 200  # epochs
patience = 10  # early stopping check number
neurons = [1,2,3,4,5,6,7,8,9,
10,11,12,13,14,15]  # number of neurons
init_mode = ["random_uniform", "Identity",
"random_normal", "lecun_uniform"]   # weights distribution
l1 = [0, 0.001, 0.01, 0.1, 1]  # L1 regularization weight
l2 = [0, 0.001, 0.01, 0.1, 1]  # L2 regularization weight
do = [0, .05, .1, .15, 0.2]  # dropout percentage rate
activation = ['tanh', 'sigmoid',
'linear', 'softsign']  # activation function

########## import data and scaling ##########

```

```
34 data_xls = pd.read_excel('parametros.xlsx', 'Planilha1',
       index_col=None)
35 input_data = data_xls.iloc[:, 0:2].values
36 output_data = data_xls.iloc[:, 2:5].values
37 sc = MinMaxScaler(feature_range=(0, 1))
38 output = MinMaxScaler(feature_range=(0, 1))
39 input_set_scaled = sc.fit_transform(input_data)
40 output_set_scaled = output.fit_transform(output_data)
41
42 def create_model(init_mode, do, l1, l2, neurons, activation):
43   np.random.seed(1)  # reproducibility
44   regressor = Sequential()
45   # First layer and some dropout regularization
46   regressor.add(Dense(units=neurons, activation=activation,
      input_dim=ni, kernel_initializer=init_mode,
      kernel_regularizer=keras.regularizers.l1_l2(l1=l1, l2=l2)))
47   regressor.add(Dropout(do))
48   # Second layer and some dropout regularization
49   regressor.add(Dense(units=neurons, activation=activation,
      kernel_initializer=init_mode, kernel_regularizer=keras.
      regularizers.l1_l2(l1=l1, l2=l2)))
50   regressor.add(Dropout(do))
51   # Adding the output layer
52   regressor.add(Dense(units=no))
53   # Compiling the RNN
54   regressor.compile(optimizer='adam', loss='mean_squared_error'
      , metrics=['accuracy'])
55   return regressor
56 # simple early stopping and model checkpoint
57 es = EarlyStopping(monitor='val_loss', mode='auto', verbose=0,
       patience=10, min_delta=0.0001)
58 mc = ModelCheckpoint('best_model.h5', monitor='val_loss', mode=
       'auto', verbose=0, save_best_only=True)
59 regressor = KerasRegressor(build_fn=create_model, verbose=0)
60 param_grid = dict(init_mode=init_mode, do=do, l1=l1, l2=l2,
       neurons=neurons, activation=activation)
61 grid = GridSearchCV(estimator=regressor, param_grid=param_grid,
       cv=5)
```

```
62 grid_result = grid.fit(input_set_scaled, output_set_scaled,
       epochs=en, batch_size=bs, verbose=0, callbacks=[es, mc])
63 print("Best: MSE de %f using %s" % (grid_result.best_score_,
       grid_result.best_params_))
64 means = grid_result.cv_results_['mean_test_score']
65 stds = grid_result.cv_results_['std_test_score']
66 params = grid_result.cv_results_['params']
67 for mean, stdev, param in zip(means, stds, params):
68   print("%f (%f) with: %r" % (mean, stdev, param))
69
70 ########## Fitting the RNA ##########
71 activation = grid_result.best_params_.get('activation')
72 do = grid_result.best_params_.get('do')
73 init_mode = grid_result.best_params_.get('init_mode')
74 l1 = grid_result.best_params_.get('l1')
75 l2 = grid_result.best_params_.get('l2')
76 neurons = grid_result.best_params_.get('neurons')
77 regressor = create_model(init_mode=init_mode, do=do, l1=l1, l2=
       l2, neurons=neurons, activation=activation)
78 regressor.fit(input_set_scaled, output_set_scaled, epochs=en,
       batch_size=bs, verbose=0, callbacks=[es, mc])
79 trainPredict_scaled = regressor.predict(input_set_scaled)
80 trainPredict = output.inverse_transform(trainPredict_scaled)
81 MSE = mean_squared_error(output_set_scaled, trainPredict_scaled
       )
```

APÊNDICE B – Simulação do comportamento dinâmico

```
1  ####### Import libraries #######
2
3  import numpy as np
4  import matplotlib.pyplot as plt
5  import pandas as pd
6  from keras.models import model_from_json
7  from sklearn.preprocessing import MinMaxScaler
8  from scipy.optimize import fsolve
9  from scipy.optimize import least_squares
10 from scipy.integrate import odeint
11
12 ####### Import data #######
13
14 data_xls = pd.read_excel('parametros.xlsx', 'Planilha1',
       index_col=None)
15 input_data = data_xls.iloc[:, 0:2].values
16 output_data = data_xls.iloc[:, 2:5].values
17 sc = MinMaxScaler(feature_range=(0, 1))
18 output = MinMaxScaler(feature_range=(0, 1))
19 input_set_scaled = sc.fit_transform(input_data)
20 output_set_scaled = output.fit_transform(output_data)
21 data_step_xls = pd.read_excel('Pasta2.xlsx', 'Planilha1',
       index_col=None)
22 input_step = data_step_xls.iloc[:,0:3].values
23
24 def normalizar (S, R):
25 max1 = 150
26 max2 = 42
27 min1 = 50
28 min2 = 18
29
30 snorm = 1-((max1-S)/(max1-min1))
31 rnorm = 1-((max2-R)/(max2-min2))
```

```
32 return [snorm, rnorm]
33
34 # Variaveis auxiliares
35 Mu = 0.051  # h^-1
36 Kd = 0.005  # h^-1
37 Ks = 1.9  # g/L
38 Kp = 20.650  # g/L
39 Ksl = 112.51  # g/L
40 Cv = 1.0110388566223264e-06  # dm^2
41 g = 9.81*10*3600**2  # dm/h^2
42 At = 1  # dm^2
43 ## Neural Network
44 json_file = open('model3.json', 'r')
45 loaded_model_json = json_file.read()
46 json_file.close()
47 regressor = model_from_json(loaded_model_json)
48 # load weights into new model
49 regressor.load_weights("model3.h5")
50 print("Loaded model from disk")
51
52 def fun(y, t):
53   # Variáveis auxiliares
54   for i in range (len(y))
55     if y[i]<0:
56       y[i]=0
57   V=y[0]
58   X=y[1]
59   S=y[2]
60   P=y[3]
61   Qo = Cv * pow(2 * g * V / At, 0.5)
62   input = np.array(normalizar(Si, V/Qo)).reshape([1, -1])
63   neural_result = output.inverse_transform(regressor.predict(
     input))
64   neural_result.reshape([3,1])
65   ms, YXS, YPS = neural_result[0,0], neural_result[0,1],
     neural_result[0,2]
66   # Equações auxiiares
67   mu = Mu*S/(Ks+S)*Kp/(Kp+P)*Ksl/(Ksl+S)
68   alpha = YPS/YXS
```

```
69    beta = YPS*ms
70    rx = mu*X
71    rd = Kd*X
72    rp = alpha*rx+beta*X
73    # Derivadas
74    dVdt = Qi-Qo
75    dXdt = Qi/V*(Xi-X)+rx-rd
76    dPdt = -Qi/V*P + rp
77    Rxs = rx / YXS
78    Rps = rp / YPS
79    Rms = ms*X
80    dSdt = Qi/V*(Si-S)-(Rxs+Rms+Rps)
81    dydt = [dVdt, dXdt, dSdt, dPdt]
82    return dydt
83 y0 = [4.537528930501552, 0.10819826943270977,
       1.682128586540394, 8.409823198816403] # 0.1086  50  0.055
84 V, X, S, P, t = [], [], [], [], []
85 tspan = 201
86 for i in range(len(input_step)):
87    Qi = input_step[i,0]
88    Si = input_step[i,1]
89    Xi = input_step[i,2]
90    time = np.linspace(i*200, i*200+200, tspan)
91    sol = odeint(fun, y0, time)
92    v, x, s, p = sol[:,0], sol[:,1], sol[:,2], sol[:,3]
93    y0 = [v[-1], x[-1], s[-1], p[-1]]
94    V.append(v), X.append(x), S.append(s), P.append(p), t.append(
       time)
```

APÊNDICE C – Treinamento e validação das RNN-LSTM

```
1  ###### Import libraries #######
2
3  import numpy as np
4  import matplotlib.pyplot as plt
5  import pandas as pd
6  import keras
7  from keras.models import Sequential
8  from keras.models import model_from_json
9  from keras.layers import Dense
10 from keras.layers import LSTM
11 from keras.layers import Dropout
12 from keras.callbacks import EarlyStopping
13 from keras.callbacks import ModelCheckpoint
14 from keras.utils.vis_utils import plot_model
15 from sklearn.preprocessing import MinMaxScaler
16 from sklearn.metrics import mean_squared_error, r2_score
17 import warnings
18
19 #### Parameters to modify #####
20
21 look_back=2  # number of time steps used to predict
22 ni=7  # number of input variables
23 no = 4 # number of output variables
24 f_train = .8  # fraction of dataset used in the training stage
25 en = 300  # epochs
26 bs = 64  # batch size
27 vs = .2  # validation split percentage
28 patience = 10  # early stopping check number
29 neurons = [20] # number of neurons
30 init_mode=["random_uniform", "Identity", "random_normal"] #
       weights distribution
31 l1=[0, .0001, .001, .01, .1]  # L1 regularization weight
32 l2=[0, .0001, .001, .01, .1]  # L2 regularization weight
```

```
33 do=[0, .05, .10, .15, .20]    # dropout percentage rate
34
35 ###### Auxiliar functions ######
36
37 # Creating a data structure with look_back timesteps and 1
    output
38 def create_dataset(training_set_scaled, look_back):
39   x_train, y_train = [], []
40   for i in range(look_back, len(training_set_scaled)):
41   x_train.append(training_set_scaled[i - look_back:i, 0:ni+1])
42   y_train.append(training_set_scaled[i, 0:no])
43   return np.array(x_train), np.array(y_train)
44
45 # Creating the parameters set
46 def model_configs():
47   configs = list()
48   for i in neurons:
49     for k in l1:
50       for l in l2:
51         for m in do:
52           for j in init_mode:
53             cfg = [i, j, k, l, m]
54             configs.append(cfg)
55   return configs
56
57 # Initializing the RNN-LSTM
58 def create_model(cfg_list, x_train, y_train, x_test, y_test):
59   MSE_train = list()
60   MSE_test = list()
61   for i in range (len(cfg_list)):
62   # Variáveis Auxiliares
63   np.random.seed(1)  # reproducibility
64   neurons = cfg_list[i][0]; init_mode=cfg_list[i][1]; l1=
    cfg_list[i][2]; l2=cfg_list[i][3]; do=cfg_list[i][4]
65
66   # Definindo a rede
67   regressor = Sequential()
68   # 1st layer
```

```
69    regressor.add(LSTM(units=neurons, input_shape=(look_back, ni)
      , kernel_initializer=init_mode, kernel_regularizer=keras.
      regularizers.l1_l2(l1=l1, l2=l2)))
70    regressor.add(Dropout(do))
71    # Adding the output layer
72    regressor.add(Dense(units=no))
73    # Compiling the RNN
74    regressor.compile(optimizer='adam', loss='mean_squared_error'
      )
75    # simple early stopping and model checkpoint
76    es = EarlyStopping(monitor='val_loss', mode='auto', verbose
      =0, patience=patience, restore_best_weights=True,
      restore_best_weights=True, min_delta=0.00001)
77    mc = ModelCheckpoint('best_model.h5', monitor='val_loss',
      mode='auto', verbose=0, save_best_only=True)
78    # Fitting the RNN to the training set
79    history = regressor.fit(x_train, y_train, validation_split=vs
      , epochs=en, batch_size=bs, verbose=0, callbacks=[es, mc])
80    train_predict = regressor.predict(x_train, batch_size= None,
      verbose=0)
81    regressor.reset_states()
82    test_predict = regressor.predict(x_test, batch_size= None,
      verbose=0)
83    trainScore = mean_squared_error(y_train, train_predict)
84    testScore = mean_squared_error(y_test, test_predict)
85    MSE_test.append(testScore)
86    MSE_train.append(trainScore)
87    return MSE_train, MSE_test
88
89 ####### Import dataset set #######
90
91 dataset_train = pd.read_excel('simulacao_resultados.xlsx', '
      Sheet1', index_col=None)
92 training_set = dataset_train.iloc[:, 1:8].values # column 0 is
      time
93
94 ## Feature scaling
95 sc = MinMaxScaler(feature_range=(0, 1))
96 output = MinMaxScaler(feature_range=(0, 1))
```

```
training_set_scaled = sc.fit_transform(training_set)
output_scaled = output.fit_transform(np.reshape(training_set[:,
    0:no], [-1,no]))

## Split into train and test sets
train_size = int(len(training_set_scaled)*f_train)
test_size = len(training_set_scaled)-train_size
training_set_splited = training_set_scaled[0:train_size, :]
test_set_splited = training_set_scaled[train_size:len(
   training_set_scaled),:]
x_train, y_train = create_dataset(training_set_splited,
   look_back)
x_test, y_test = create_dataset(test_set_splited, look_back)

######## Search Tree #######

cfg_list = model_configs()

[MSE_train, MSE_test] = create_model(cfg_list, x_train, y_train
   , x_test, y_test)

###### Fitting best model ######
np.random.seed(1)  # reproducibility
i=MSE_test.index(min(MSE_test))
neurons = cfg_list[i][0]; init_mode=cfg_list[i][1]; l1=cfg_list
   [i][2]; l2=cfg_list[i][3]; do=cfg_list[i][4]

# Definindo a rede
regressor = Sequential()
# 1st layer
regressor.add(LSTM(units=neurons, input_shape=(look_back, ni),
   kernel_initializer=init_mode,
kernel_regularizer=keras.regularizers.l1_l2(l1=l1, l2=l2)))
regressor.add(Dropout(do))
# Adding the output layer
regressor.add(Dense(units=no))
# Compiling the RNN
regressor.compile(optimizer='adam', loss='mean_squared_error')
# simple early stopping and model checkpoint
```

```
130 es = EarlyStopping(monitor='val_loss', mode='auto', verbose=0,
      patience=patience)
131 mc = ModelCheckpoint('best_model.h5', monitor='val_loss', mode=
      'auto', verbose=0, save_best_only=True)
132 # Fitting the RNN to the training set
133 history = regressor.fit(x_train, y_train, validation_split=vs,
      epochs=en, batch_size=bs, verbose=0,
134 callbacks=[es, mc])
135 train_predict = regressor.predict(x_train, batch_size=None,
      verbose=0)
136 regressor.reset_states()
137 test_predict = regressor.predict(x_test, batch_size=None,
      verbose=0)
138 trainScore = mean_squared_error(y_train, train_predict)
139 testScore = mean_squared_error(y_test, test_predict)
140
141 # # Denormalization
142
143 train_predict = output.inverse_transform(np.reshape(
      train_predict, [-1, no]))
144 test_predict = output.inverse_transform(np.reshape(test_predict
      , [-1, no]))
145 y_train = output.inverse_transform(np.reshape(y_train, [-1, no
      ]))
146 y_test = output.inverse_transform(np.reshape(y_test, [-1, no]))
147
148 ######## Save RNN (Weights and biases) #########
149
150 # serialize model to JSON
151 model_json = regressor.to_json()
152 with open("model.json", "w") as json_file:
153 json_file.write(model_json)
154 # serialize weights to HDF5
155 regressor.save_weights("model.h5")
156 print("Saved model to disk")
```

APÊNDICE D – Modelo *offline*

```
##### Import libraries #####

import numpy as np
import pandas as pd
from keras.models import model_from_json
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error

##### Parameters to modify #####

look_back = 2  # number of time steps used to predict
ni = 7  # number of input variables
no = 4  # number of output variables

##### Import dataset set #####

dataset_train = pd.read_excel('simulacao_resultados.xlsx', '
    Sheet1', index_col=None)
training_set = dataset_train.iloc[:, 1:8].values # column 0 is
    time

##### Feature scaling #####

sc = MinMaxScaler(feature_range=(0, 1))
output = MinMaxScaler(feature_range=(0, 1))
training_set_offline = sc.fit_transform(training_set)
output_scaled = output.fit_transform(np.reshape(training_set[:,
     0:no], [-1,no]))

# # Load RNN
json_file = open('model.json', 'r')
loaded_model_json = json_file.read()
json_file.close()
regressor = model_from_json(loaded_model_json)
# load weights into new model
regressor.load_weights("model.h5")
```

```
34 print("Loaded model from disk")
35
36 regressor.reset_states()
37 # Initial data
38 x_offline = training_set_offline[0:look_back,:]
39 x_offline = np.reshape(x_offline, (1, look_back, ni))
40 y_data_offline = []
41
42 for i in range(look_back, len(training_set_offline)-1):
43   y_predict_offline = regressor.predict(x_offline, batch_size=
     None, verbose=0)
44   x_data = training_set_offline[i-look_back+1:i+1,:]
45   x_data[look_back-1, 0:no] = y_predict_offline
46   x_offline = np.reshape(x_data, (1, look_back, ni))
47   y_data_offline.append(y_predict_offline)
48
49 y_offline = np.reshape(y_data_offline, [-1,no])
50 y_real = training_set_offline[look_back-1:len(y_offline)+1, 0:
     no]
51
52 offlineScore = mean_squared_error(y_real, y_offline)
53 y_offline = output.inverse_transform(y_offline)
54 y_real = training_set[look_back:-1, 0:no]
```