Pontifícia Universidade Católica
do Rio de Janeiro

Bruno dos Santos Costa

Um Estudo da Sensibilidade dos Hiperparâmetros no E2CO

Dissertação de Mestrado

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós–graduação em Matemática, do Departamento de Matemática da PUC-Rio.

Orientador : Prof. Sinesio Pesco
Co-orientador: Prof. Abelardo Borges Barreto Junior

Rio de Janeiro
Abril de 2025

Pontifícia Universidade Católica
do Rio de Janeiro

Bruno dos Santos Costa

Um Estudo da Sensibilidade dos Hiperparâmetros no E2CO

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós–graduação em Matemática da PUC-Rio. Aprovada pela Comissão Examinadora abaixo:

Prof. Sinesio Pesco
Orientador
Departamento de Matemática – PUC-Rio

Prof. Abelardo Borges Barreto Junior
Co-orientador
Departamento de Matemática – PUC-Rio

Prof. Thiago de Menezes Duarte e Silva
Schlumberger Serviços de Petróleo - Matriz

Profa. Malú Grave
UFF

Prof. Emílio José Rocha Coutinho
Petróleo Brasileiro - Rio de Janeiro - Matriz

Rio de Janeiro, 11 de Abril de 2025

Bruno dos Santos Costa

Graduado em Licenciatura em Matemática pela Universidade Federal de Sergipe em 2022.

Ficha Catalográfica

Costa, Bruno dos Santos

Um Estudo da Sensibilidade dos Hiperparâmetros no E2CO / Bruno dos Santos Costa; orientador: Sinesio Pesco; co-orientador: Abelardo Borges Barreto Junior. – 2025.

124 f. : il. color. ; 30 cm

Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Matemática, 2025.

Inclui bibliografia

1. Matemática – Teses. 2. Aprendizado de Máquina. 3. E2CO. 4. Redes Neurais. 5. Estimador de James-Stein. I. Pesco, Sinesio. II. Barreto Junior, Abelardo Borges. III. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Matemática. IV. Título.

CDD: 510

À Deus, por sempre me dar forças diariamente. Afinal, como está escrito:

[29] Ele fortalece o cansado e dá grande vigor ao que está sem forças.
[30] Até os jovens se cansam e ficam exaustos, e os moços tropeçam e caem;
[31] mas aqueles que esperam no Senhor renovam as suas forças.
Voam alto como águias; correm e não ficam exaustos, andam e não se cansam. (Isaías 40:29-31)

Dedico esta obra também aos meus pais, Sebastião e Elizabete; às minhas irmãs, Bruna e Verônica; e aos meus amigos (de pós-graduação, professores e funcionários) do Departamento de Matemática da PUC-Rio, pelo apoio incondicional ao longo desta jornada.

Agradecimentos

A Deus, pela força, determinação e graça concedidas ao longo desta jornada. Aos meus pais, Sebastião e Elizabete, e às minhas irmãs, Bruna e Verônica, meu profundo agradecimento pelo amor incondicional, pelo apoio nos momentos desafiadores e por sempre acreditarem em mim. Agradeço também à minha namorada Stéphanie pela paciência, amor, apoio e carinho ao longo desta jornada.

Expresso minha imensa gratidão aos meus orientadores, professores Sinésio Pesco e Abelardo Barreto, pela orientação indispensável, pelo apoio constante e pela paciência ao longo de todo o percurso. Aos membros da banca de avaliação, agradeço pela generosidade em dedicar seu tempo para analisar este trabalho e pelas sugestões e comentários valiosos que enriqueceram o resultado final. Em particular, agradeço ao professor Emílio Coutinho, pelo auxílio na compreensão das ferramentas principais para o desenvolvimento da dissertação.

Minha gratidão também ao professor Boyan Sirakov, pelo auxílio fundamental e pelas palavras de incentivo durante meu primeiro semestre de mestrado na PUC-Rio.

Aos profissionais do Departamento de Matemática, Creuza, Carlos, Kátia e Mariana, registro meus sinceros agradecimentos pelo suporte excepcional e pela dedicação no trabalho diário, que tanto contribuem para o bom funcionamento do DMAT.

Agradeço de coração aos amigos do DMAT-PUC: Adailton Souza, Adriano Gonçalves, Antonio Andrade, Fabricio Santos, Gabriel Couto, Iago Abalada, Marcelo Pinto, Matheus Setaro, Raul Chavez, Rony Hurtado, Sergio Loiola, Victor Santos e Wladimir Cândido, pela camaradagem, pelas discussões produtivas e pelo apoio mútuo durante esta jornada. Em particular, expresso minha gratidão a Anselmo Pontes, Anthony Guillen, Átila Silva, Claudemir Alcantara, Dimary Moreno, Isabel Gonçalves, Jéssica Neto, Joel Silva, João de Sá, Ortenilton Filho, Victor El Adji e Vinicius Costa, por suas amizades inspiradoras e pelas conversas instigantes sobre Matemática e Programação, que foram fundamentais para o meu crescimento acadêmico e pessoal.

Expresso minha profunda gratidão aos professores Disson dos Prazeres e Fábio dos Santos, da UFS, meus orientadores e mentores na graduação. Suas orientações, incentivo e apoio foram essenciais para minha trajetória acadêmica, inspirando-me a sempre dar o meu melhor como estudante. Também sou imensamente grato ao professor David de Paiva (IFS), que me orientou durante a Iniciação Científica no ensino médio. Seu apoio inestimável e sua inspiração desempenharam um papel crucial na construção do meu caminho acadêmico. Ainda no IFS, gostaria de registrar meu sincero agradecimento aos professores

Paulo André, Hélio Vicente e Rosana, que tiveram um papel fundamental nos primeiros passos dessa jornada.
O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001
Também agradeço ao apoio da PUC-Rio e Petrobras.
A todos que direta ou indiretamente contribuíram para esta conquista, meu mais sincero agradecimento.

Resumo

Costa, Bruno dos Santos; Pesco, Sinesio; Barreto Junior, Abelardo Borges. **Um Estudo da Sensibilidade dos Hiperparâmetros no E2CO**. Rio de Janeiro, 2025. 124p. Dissertação de Mestrado – Departamento de Matemática, Pontifícia Universidade Católica do Rio de Janeiro.

Esta dissertação insere-se no contexto da Modelagem Matemática, com ênfase em problemas relacionados à Engenharia de Reservatórios. Em particular, foi abordado ao longo do texto a aplicação de redes neurais para a predição de importantes dados de poços de petróleo, tais como pressão de fundo de poço, vazão de óleo e água, ao longo de períodos prolongados. Para isso, foi utilizado o método conhecido como *Embed to Control and Observe*. Um dos principais tópicos discutidos no trabalho foi referente a sensibilidade dos hiperparâmetros das redes neurais, que são definidos durante o processo de treinamento. Em especial, houve uma investigação sobre como a variação desses hiperparâmetros impacta na acurácia das predições. Notou-se que os pesos aplicados às funções custo (transição, transição de saída, vazão de água, saturação nos *gridblock* produtores), o número de *batch size*, da *seed* e da versão de Python impactaram significativamente na acurácia das predições.

Palavras-chave

Aprendizado de Máquina; E2CO; Redes Neurais; Estimador de James-Stein.

Abstract

Costa, Bruno dos Santos; Pesco, Sinesio (Advisor); Barreto Junior, Abelardo Borges (Co-Advisor). **Sensitivity Study of Hyperparameters in E2CO**. Rio de Janeiro, 2025. 124p. Dissertação de Mestrado – Departamento de Matemática, Pontifícia Universidade Católica do Rio de Janeiro.

This dissertation is situated within the field of Mathematical Modeling, with an emphasis on problems related to Reservoir Engineering. In particular, the text explores the application of neural networks for the prediction of key oil well data, such as bottom-hole pressure, oil flow rate, and water flow rate over extended periods. To achieve this, the method known as Embed to Control and Observe was employed. One of the main topics discussed in the study concerns the sensitivity of neural network hyperparameters, which are defined during the training process. Specifically, the investigation focused on how variations in these hyperparameters affect the accuracy of the predictions. It was observed that the weights assigned to the cost functions (transition, output transition, water flow rate, saturation in producing gridblocks), the batch size, the seed, and the Python version significantly influenced the prediction accuracy.

Keywords

Machine Learning; E2CO; Neural Network; James-Stein Estimator.

Sumário

Lista de figuras

Lista de tabelas

Lista de algoritmos

Lista de Abreviaturas

Terminologias gerais

EDPs – Equações Diferenciais Parciais

EDOs – Equações Diferenciais Ordinárias

HFS – *High Fidelity Simulation* (Simulação de alta fidelidade)

MOR – *Model order reduction* (Modelo de ordem reduzida)

E2C – *Embed to Control* (Incorporar para Controle)

E2CO – *Embed to Control and Observe* (Incorporar para Controle e Observação)

POD – *Proper Orthogonal Decomposition* (Decomposição Ortogonal Própria)

SciML – *Scientific Machine Learning* (Aprendizado de Máquina Científico)

TPWL – *Trajectory Piecewise Linearization* (Linearização por Partes da Trajetória)

MSE – *Scalar Mean Squared Error Criterion* (Critério de erro quadrático médio escalar)

OLS – *Ordinary Least Squares Estimator* (Estimador de mínimos quadrados ordinários)

BHP – *Bottom-hole pressure (Pressão de fundo de poço)*

JS – Estimador de James-Stein

Estado e Dados Estimados

$\hat{x}$ – Estado estimado no espaço original

$\hat{y}$ – Dados observados estimados

$\hat{z}$ – Estado estimado no espaço latente

Funções de Perda

$\mathcal{L}_{flux,pred}$ – Perda para previsão do fluxo

$\mathcal{L}_{flux,rec}$ – Perda para reconstrução do fluxo

$\mathcal{L}_{flux2Ph,pred}$ – Perda para previsão do fluxo bifásico

$\mathcal{L}_{flux2Ph,rec}$ – Perda para reconstrução do fluxo bifásico

$\mathcal{L}_{flux2Ph}$ – Perda do fluxo bifásico

$\mathcal{L}_{flux}$ – Perda do fluxo

$\mathcal{L}_{pred}$ – Perda para previsão

$\mathcal{L}_{rec}$ – Perda para reconstrução

$\mathcal{L}_{pres}$ – Perda para a pressão nos blocos de malha dos injetores

$\mathcal{L}_{i\_sat}$ – Perda para a saturação nos blocos de malha dos injetores

$\mathcal{L}_{p\_pres}$ – Perda para a pressão nos blocos de malha dos produtores

$\mathcal{L}_{p\_sat}$ – Perda para a saturação nos blocos de malha dos produtores

$\mathcal{L}_{trans}$ – Perda para transição

$\mathcal{L}_{well\_data3}$ – Perda dos dados de poço para modelo E2CO

Pesos Aplicados às Funções de Perda

γ_{flux} – Aplicado à γ_{flux}

$\gamma_{flux2Ph}$ – Aplicado à $\mathcal{L}_{flux2Ph}$

γ_{i_pres} – Aplicado à $\mathcal{L}_{i\_pres}$

γ_{i_sat} – Aplicado à $\mathcal{L}_{i\_sat}$

γ_{p_pres} – Aplicado à $\mathcal{L}_{p\_pres}$

γ_{p_sat} – Aplicado à $\mathcal{L}_{p\_sat}$

γ_{well_data2} – Aplicado à $\mathcal{L}_{well\_data2}$

γ_{well_data3} – Aplicado à $\mathcal{L}_{well\_data3}$

Matrizes

$\mathbf{A}$ – Matriz de estado discretizada

$\mathbf{A}_c$ – Matriz de estado contínua

$\mathbf{A}_r$ – Matriz de estado reduzida discretizada

$\mathbf{B}$ – Matriz de entrada discretizada

$\mathbf{B}_c$ – Matriz de entrada contínua

$\mathbf{B}_r$ – Matriz de entrada reduzida discretizada

$\mathbf{C}$ – Matriz de saída discretizada

$\mathbf{C}_c$ – Matriz de saída contínua

$\mathbf{C}_r$ – Matriz de saída reduzida discretizada

$\mathbf{D}$ – Matriz de alimentação direta discretizada

$\mathbf{D}_c$ – Matriz de alimentação direta contínua

$\mathbf{D}_r$ – Matriz de alimentação direta reduzida discretizada

Erro relativo

e_j^p – Referente à vazão de fase j para o poço produtor p

E_o – Referente à vazão de óleo para todos os produtores

E_v – Referente à variável de estado v

E_w – Referente à vazão de água para todos os produtores

E_{BHP} – Referente à pressão no fundo do poço para todos os injetores

e_i – Referente à pressão no fundo do poço do poço injetor i

$E_{\mathrm{cum},j}$ – Referente ao fluxo acumulado da fase j para todos os produtores

$e_{\mathrm{cum,j}}^p$ – Referente ao fluxo acumulado da fase j para o poço produtor p

Demais parâmetros e/ou variáveis

μ – Viscosidade

∇p – Gradiente de pressão dos blocos de malha adjacentes

ϕ – Porosidade

ρ – Densidade

SKIN – Fator *Skin*

WI – *Well index* (Índice de Poço)

B – Fator de volume de formação

h – Altura do bloco de malha do poço

k – Permeabilidade

$k_{r;j}$ – Permeabilidade relativa à fase j

l_x – Dimensão do estado no espaço original

l_z – Dimensão do estado no espaço reduzido

n_i – Número de poços injetores

n_p – Número de poços produtores

P – Pressão

q_{inj} – Taxa de injeção de água

q_o – Taxa de produção de óleo

q_w – Taxa de produção de água

r_o – Raio equivalente do bloco de malha do poço

r_w – Raio do poço

S – Saturação

S_w – Saturação de água

u – Entrada ou controle do sistema

x – Estado no espaço original

y – Saída do sistema

z – Estado no espaço latente

A beleza da Matemática só se mostra aos seguidores mais pacientes.

Maryam Mirzakhani

1
Introdução

Sabe-se que a indústria de óleo e gás representa um dos pilares da economia mundial devido à sua importância estratégica para o desenvolvimento socioeconômico de diversos países. Sendo assim, para companhias/empresas que realizam a extração de petróleo, é crucial conhecer dados que forneçam informações sobre as características dos poços de petróleo, a fim de identificar, por exemplo, a viabilidade de extração desse recurso energético.

Nesse contexto, é importante que as companhias possuam dados confiáveis relativos à pressão de fundo de poço para poços injetores, além da vazão/-taxa de óleo e água para poços produtores. De posse disso, há conhecimento de simuladores que procuram reproduzir situações similares aos casos reais, a exemplo do IMEX (CMG, 2024). Este simulador foi lançado em 1980 e é um dos principais simuladores de reservatório para modelagem de processos de recuperação primária, secundária e terciária de petróleo. Todavia, é de conhecimento geral que quanto mais rápido for possível obter informações sobre os dados do poço, melhor para a tomada de decisões, sejam elas de cunho técnico e/ou econômico, respeitando sempre a relação custo-benefício.

Sob essa ótica de obtenção de informações confiáveis a respeito dos dados de poços mencionados, torna-se relevante comentar sobre os modelos *proxy*. Uma descrição simples de tais modelos é dada por (BAHRAMI; MOGHADDAM; JAMES, 2022), que os define como tabelas de interpolação avançadas, a partir das quais podemos interpolar rapidamente intervalos de dados não lineares para encontrar uma solução aproximada. Segundo (MOTAEI; GANAT, 2023), os modelos *proxy* se destacam por compensar a lentidão dos modelos de simulação numérica, oferecendo atualizações rápidas com novos dados. Os autores ressaltam que esses modelos são particularmente úteis em situações em que o tempo de decisão é crítico, proporcionando respostas ágeis aos eventos do processo. Entre as principais abordagens para a construção de modelos *proxy* estão a regressão, as redes neurais artificiais (ANNs), a lógica fuzzy e as máquinas de vetores de suporte (SVM).

No artigo, (MOTAEI; GANAT, 2023) também explicam que o potencial dos modelos *proxy* em previsão e otimização ainda não foi totalmente explorado. São necessários avanços para aprimorar a qualidade desses modelos, incorporar dados ao núcleo dos modelos *proxy*, aumentar sua robustez e desenvolver métricas de desempenho sólidas, com o objetivo de avaliar sua performance, confiabilidade e pontualidade. Embora atualmente esses modelos

sejam utilizados como suporte para a otimização de processos e a tomada de decisões, projeta-se que venham a se tornar protagonistas, tanto na vigilância e monitoramento diários quanto no próprio processo de otimização.

Em particular, nesta dissertação, discutiremos a construção de um modelo *proxy* baseado em redes neurais artificiais, conhecido como *Embed to Control and Observe*, desenvolvido por (COUTINHO, 2022). Queremos prever a longo prazo dados de poço (pressão de fundo de poço, vazão de óleo e água), ou seja, estaremos prevendo dados considerando um sistema bifásico (óleo-água). Mais ainda, temos como o principal problema de pesquisa investigar a questão da sensibilidade aos hiperparâmetros das redes neurais, posto que, conforme veremos, pequenas variações em tais hiperparâmetros podem pôr em risco a qualidade das predições realizadas. Todavia, antes de chegarmos a esse método, será relevante entender, principalmente, como as equações de simulação de reservatório podem ser reescritas como sistemas (o que será tratado no capítulo 2). No capítulo 3, compreenderemos os métodos *Embed to Control*, *Embed to Control* acoplado com função de perda física para simulação numérica de reservatórios de petróleo, e *Embed to Control* acoplado com dados de poço e funções de perda de fluxo bifásico. Para isso, exploraremos os trabalhos de (WATTER et al., 2015), (JIN; LIU; DURLOFSKY, 2020) e (COUTINHO, 2022).

Após essa parte introdutória, ainda no capítulo 3, investigaremos o treinamento e predição de dados de poço, a estruturação e implementação das redes neurais, a aplicação de camadas convolucionais parciais e a geração de dados. Também discutiremos a análise dos erros relativos das predições e proporemos a aplicação de uma ferramenta estatística, conhecida como Estimador de James-Stein (JAMES; STEIN, 1961), aplicada aos dados de poço, na tentativa de obtermos informações ainda mais confiáveis.

Por fim, no capítulo 4, analisaremos os cenários observados a partir das predições realizadas, compararemos os resultados obtidos com aqueles disponíveis na literatura. Encerraremos no capítulo 5 com uma breve conclusão e sugestões para futuros estudos nesta linha de pesquisa.

2
Simulação Numérica de Reservatórios

A simulação de reservatório pode ser definida como o processo de inferir o comportamento de um reservatório real através do desempenho de um modelo de tal reservatório (PEACEMAN, 1977). Existem dois tipos de modelos, o físico (por exemplo, um modelo de laboratório em escala) e o matemático. Um modelo matemático é um conjunto de EDPs (Equações Diferenciais Parciais) juntamente com um conjunto apropriado de condições de contorno, que acreditamos descrever adequadamente os processos físicos significativos que ocorrem em tal sistema.

Peaceman (PEACEMAN, 1977) também informa que nos reservatórios de petróleo ocorrem basicamente fluxo de fluidos e transferência de massa. Pode-se fluir simultaneamente até três fases imiscíveis (água, óleo e gás), já a transferência de massa pode acontecer principalmente entre as fases de gás e óleo. No processo de fluxo de fluidos, tem-se que levar em consideração ação de forças gravitacionais, capilares e viscosas. O autor reforça que as equações que regem o modelo devem levar em consideração as tais forças supracitadas e considerar uma descrição arbitrária do reservatório com respeito à heterogeneidade e geometria. As EDs (equações diferenciais) são obtidas via combinação da lei de Darcy (ver em Apêndice A) para cada fase com um balanço material diferencial simples para cada fase.

Em uma EDP que descreve a física do fluxo de fluidos num meio poroso, em geral, as variáveis independentes são o tempo, t, e as coordenadas espaciais x, y, z. Denotando a variável dependente arbitrária com um ponto grosso ($\bullet$), as equações podem ser expressas como (JANSEN, 2013):

$$\underbrace{\varphi(t,x,y,z,\bullet) \times L(\bullet)}_{\text{termo de transporte e/ou fluxo}} + \underbrace{\psi(t,x,y,z,\bullet)}_{\text{termo fonte}} - \underbrace{\varepsilon(t,x,y,z,\bullet) \times \frac{\partial(\bullet)}{\partial t}}_{\text{termo de acumulação}} = 0, \qquad (2\text{-}1)$$

em que φ, ψ e ε são parâmetros que podem ser funções de tempo e espaço, e L é um operador diferencial espacial. Notemos que ε, φ e ψ podem ser funções de variável dependente, em que a equação é não linear. Além da EDP (2-1), é necessário especificar o domínio espacial Ω e o domínio temporal T nos quais ela é válida. Na fronteira Γ de Ω, necessita-se definir condições de contorno (CC), e em um ponto específico no tempo, uma condição inicial (CI), para formular completamente o problema. Para geometrias que se aproximam mais da realidade, buscamos soluções numéricas para as equações, o que requer

alguma forma de discretização das equações.

Como já apontado na Introdução, neste trabalho será abordado o caso bifásico (água-óleo). Nesse sentido, é importante destacar o trabalho de (COUTINHO, 2022) que nos informa que um simulador de reservatório bidimensional bifásico (óleo e água) é baseado nas equações de fluxo:

$$\underbrace{\nabla\left[\frac{\vec{k}k_{r,j}(S_j)}{\mu_j}\rho_j\nabla P_j\right]}_{\text{termo de fluxo}} + \underbrace{q_j}_{\text{termo fonte}} - \underbrace{\frac{\partial}{\partial t}(\phi S_j\rho_j)}_{\text{termo de acumulação}} = 0, \tag{2-2}$$

em que o subscrito j denota a fase óleo ou água, k é a permeabilidade absoluta, k_r é a permeabilidade relativa, μ é a viscosidade, ρ é a densidade do fluido e ϕ é a porosidade. Convém comentar que o uso de $\vec{}$ denota que estamos lidando com um tensor. A solução almejada é referente à pressão (P_j) e à saturação (S_j) da fase j.

Em seu trabalho, (COUTINHO, 2022) expõe que as equações de simulação de reservatório podem ser reescritas como sistemas, em que as equações não lineares podem ser linearizadas num modelo de espaço de estado variável no tempo. Tal sistema pode ser expresso como

$$\xrightarrow{u(t)} \boxed{\begin{array}{l}\dot{x} = \mathbf{A}x + \mathbf{B}u \\ y = \mathbf{C}x + \mathbf{D}u\end{array}} \xrightarrow{y(t)}$$

Figura 2.1: Representação do sistema. Fonte: (COUTINHO, 2022).

O autor salienta que embora os vetores de controle u sejam impostos aos poços, a evolução do estado x é calculada baseada no processo de discretização da EDP (2-2), e a saída y é observada nos poços.

Verificaremos nas próximas três seções que tal sistema de fato apresenta a configuração exposta na Figura 2.1. Para isso, utilizaremos amplamente os conhecimentos provenientes do livro de (JANSEN, 2013). Inicialmente trataremos do caso de fluxo monofásico e, em seguida, mostraremos o caso para o fluxo bifásico.

2.1 Equações do sistema

Primeiramente, deve-se tentar realizar a discretização espacial das EDPs, deixando a discretização temporal para um estágio posterior (JANSEN, 2013). A discretização no espaço pode ser realizada utilizando diferenças finitas, volumes finitos ou elementos finitos. É válido frisar que todos os métodos de discretização resultam num sistema de equações diferenciais ordinárias (EDOs) que são representadas por

$$\begin{cases} \hat{e}_1\left(\bullet_1, \dfrac{d\left(\bullet_1\right)}{dt}\right) = \hat{f}_1\left(t, \bullet_1, \bullet_2, \ldots, \bullet_n, \psi_1\right), \\ \hat{e}_2\left(\bullet_2, \dfrac{d\left(\bullet_2\right)}{dt}\right) = \hat{f}_2\left(t, \bullet_1, \bullet_2, \ldots, \bullet_n, \psi_2\right), \\ \qquad\qquad \vdots \\ \hat{e}_n\left(\bullet_n, \dfrac{d\left(\bullet_n\right)}{dt}\right) = \hat{f}_n\left(t, \bullet_1, \bullet_2, \ldots, \bullet_n, \psi_n\right), \end{cases} \tag{2-3}$$

em que a variável dependente contínua ($\bullet$) e o termo fonte contínuo ψ da equação (2-1) são agora denotados por número finito de valores discretos $\bullet_i$ e ψ_i, correspondendo a pontos discretos no espaço, e $\hat{e}_i$, $\hat{f}_i$, são, no caso geral, funções não lineares. Em geral, as funções $\hat{e}_i$ são lineares nas derivadas $\dfrac{d(\bullet_i)}{dt}$, resultando em:

$$\begin{cases} \dfrac{d\left(\bullet_1\right)}{dt} = f_1\left(t, \bullet_1, \bullet_2, \ldots, \bullet_n, \psi_1\right), \\ \dfrac{d\left(\bullet_2\right)}{dt} = f_2\left(t, \bullet_1, \bullet_2, \ldots, \bullet_n, \psi_2\right), \\ \qquad\qquad \vdots \\ \dfrac{d\left(\bullet_n\right)}{dt} = f_n\left(t, \bullet_1, \bullet_2, \ldots, \bullet_n, \psi_n\right), \end{cases} \tag{2-4}$$

em que as funções f_i são diferentes das funções $\hat{f}_i$ no sistema de equações (2-3).

Notemos que as equações (2-3) e (2-4) são ambas sistemas acoplados de EDOs. Caso as funções f_i sejam lineares nas variáveis dependentes, uma simplificação adicional é possível que leve ao desacoplamento das equações, chegando em

$$\frac{d\left(\tilde{\bullet}_i\right)}{dt} = \tilde{f}_i\left(t, \tilde{\bullet}_i, \tilde{\psi}_i\right), \quad i = 1, 2, \ldots, n\ , \tag{2-5}$$

em que as variáveis dependentes transformadas $\tilde{\bullet}_i$ são combinações lineares das variáveis originais $\bullet_i$. Nas simulações de reservatório, as funções $\hat{e}_i$ e $\hat{f}_i$ são tipicamente lineares nas derivadas e não lineares nas variáveis dependentes, o que *a priori* torna o sistema de equações (2-4) mais importante. Vale ressaltar que, quando se trata de realizar cálculos em larga escala, é mais eficiente usar o formato do sistema (2-3).

2.1.1
Espaço de estados

Podemos reescrever o sistema (2-4) utilizando a notação vetorial, de modo a seguir a convenção adotada em Sistemas e Controle. Assim, a variável dependente contínua ($\bullet$) da equação (2-1), representada por um número finito

de valores discretos $\bullet_i$, pode ser expressa pelo vetor $\mathbf{x}$. Similarmente, o termo fonte contínuo ψ da equação (2-1), denotado por ψ_i, que corresponde a pontos discretos no espaço, pode ser representado pelo vetor $\mathbf{u}$, que atua como termo fonte. Tais vetores $\mathbf{x}$ e $\mathbf{u}$ são definidos da seguinte forma:

$$\mathbf{x} = \underbrace{\begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}}_{n \text{ elementos}}, \quad \mathbf{u} = \underbrace{\begin{bmatrix} u_1 \\ \vdots \\ u_m \end{bmatrix}}_{m \text{ elementos}}. \tag{2-6}$$

Notemos que os elementos do vetor $\mathbf{x}$ não são referentes à coordenadas espaciais. Como $\mathbf{u}$ tem m elementos, ao invés de n, isso antecipa o caso em que muitos dos termos fontes são nulos, de tal sorte que $m \ll n$, caso em que pode ser computacionalmente proveitoso fazer uso de um vetor $\mathbf{u}$ mais curto. Por (VIANA; ESPINAR, 2021), sabemos que podemos representar o sistema (2-4) por

$$\dot{\mathbf{x}}(t) = \mathbf{f}(t, \mathbf{u}(t), \mathbf{x}(t)), \tag{2-7}$$

em que $\mathbf{f}$ é uma função vetorial não linear de $\mathbf{x}, \mathbf{u}$ e t, e mais, é claro que $\mathbf{x}, \mathbf{u}$ dependem de t. Em simulações de reservatório, as equações são normalmente não lineares, todavia com coeficientes que não dependem diretamente do tempo, ou seja,

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{u}(t), \mathbf{x}(t)). \tag{2-8}$$

Caso $\mathbf{f}$ seja uma função linear, de $\mathbf{x}$ e $\mathbf{u}$, podemos usar uma notação vetor-matriz e escrever (2-8) como uma equação diferencial vetorial linear variante no tempo (LTV) - sigla em inglês, isto é,

$$\dot{\mathbf{x}}(t) = \mathbf{A}(\mathbf{t})\mathbf{x}(\mathbf{t}) + \mathbf{B}(\mathbf{t})\mathbf{u}(\mathbf{t}), \tag{2-9}$$

em que os coeficientes da matriz $\mathbf{A}_{n\times n}$, conhecida como matriz sistema e da matriz $\mathbf{B}_{n\times m}$, denominada matriz de entrada podem ser funções de t. Se $\mathbf{A}$ e $\mathbf{B}$ independem de t, obtemos uma equação linear invariante no tempo (LTI) - sigla em inglês, dada por

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(\mathbf{t}) + \mathbf{B}\mathbf{u}(\mathbf{t}). \tag{2-10}$$

A partir daqui, não indicaremos explicitamente a dependência temporal das variáveis, e escreveremos, por exemplo, simplesmente que $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{u}, \mathbf{x})$. Em Sistemas e Controle (NISE, 2014), afirmamos que sistemas de equações de primeira ordem, como o tipo (2-10), são chamados de equações de estado. Neste tipo de representação, os elementos do vetor $\mathbf{x}$ são as variáveis de estado que definem totalmente o estado dinâmico do sistema. Definimos trajetória no

espaço de estado, como uma sequência contínua de valores de $\mathbf{x}$ sobre um certo intervalo de tempo. Em aplicações de Engenharia de Reservatórios, às vezes é melhor começar utilizando o sistema (2-3) no lugar do sistema (2-4), mesmo que as funções e_i sejam lineares. Diremos que equações do tipo (2-3) são ditas equações de estado generalizadas. Em LTI, podemos escrevê-las como

$$\hat{\mathbf{E}}\dot{\mathbf{x}} = \hat{\mathbf{A}}\mathbf{x} + \hat{\mathbf{B}}\mathbf{u}, \tag{2-11}$$

em que devemos notar que $\mathbf{u}$ é o vetor de entrada, e $\hat{\mathbf{E}}$, $\hat{\mathbf{A}}$ e $\hat{\mathbf{B}}$ são as matrizes associadas ao sistema descrito em (2-3) (JANSEN, 2013).

Podemos definir a relação entre o vetor de saída $\mathbf{y}$ e o vetor de estado $\mathbf{x}$ de um sistema dinâmico em dois casos:

No caso linear, temos que

$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u}, \tag{2-12}$$

em que $\mathbf{C}_{p\times n}$ é a matriz de saída, e $\mathbf{D}_{p\times m}$ é a matriz de transmissão direta, considerando que $\mathbf{y}$ tem p elementos.

Para o caso não linear, consideramos

$$\mathbf{y} = \mathbf{h}(\mathbf{u}, \mathbf{x}). \tag{2-13}$$

Também é possível consultar em (JANSEN, 2013) a representação das equações não lineares implícitas, situadas no contexto do uso de funções não lineares ainda mais gerais.

É relevante discutir termos de erro, dado que introduzí-los em aplicações de Sistemas e Controle com o objetivo de caracterizar aproximações de processos reais. Na simulação de reservatório, comumente se considera os parâmetros das equações do sistema, em particular as permeabilidades dos blocos da malha, como incertos. De modo geral, as incertezas dos parâmetros são tão significativas que prevalecem sobre os erros do modelo.

2.1.2
Equações linearizadas

Com fins de analisar a natureza de sistemas não lineares $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{u}, \mathbf{x})$, ou para aproximar sua solução numérica, em geral é preciso linearizá-las em torno de um ponto no espaço estado-entrada. Para tal tarefa, usaremos a expansão em séries de Taylor, começando do sistema de equações (2-8), e fazendo as devidas adaptações de notação empregadas por (JANSEN, 2013), podemos escrever

$$\dot{\overline{\mathbf{x}}} + \underbrace{\dot{\mathbf{x}}^0}_{\substack{\| \\ \mathbf{f}(\mathbf{u}^0,\mathbf{x}^0)}} \approx \mathbf{f}\left(\mathbf{u}^0,\mathbf{x}^0\right) + \frac{\partial \mathbf{f}\left(\mathbf{u}^0,\mathbf{x}^0\right)}{\partial \mathbf{u}}\overline{\mathbf{u}} + \frac{\partial \mathbf{f}\left(\mathbf{u}^0,\mathbf{x}^0\right)}{\partial \mathbf{x}}\overline{\mathbf{x}}, \tag{2-14}$$

isto é, as equações do sistema linearizado são da forma

$$\dot{\overline{\mathbf{x}}} = \overline{\mathbf{A}}\left(\mathbf{u}^0,\mathbf{x}^0\right)\overline{\mathbf{x}} + \overline{\mathbf{B}}\left(\mathbf{u}^0,\mathbf{x}^0\right)\overline{\mathbf{u}}, \tag{2-15}$$

em que as matrizes Jacobianas $\overline{\mathbf{A}}$ e $\overline{\mathbf{B}}$ são:

$$\overline{\mathbf{A}}\left(\mathbf{u}^0,\mathbf{x}^0\right) \triangleq \frac{\partial \mathbf{f}\left(\mathbf{u}^0,\mathbf{x}^0\right)}{\partial \mathbf{x}}, \tag{2-16}$$

$$\overline{\mathbf{B}}\left(\mathbf{u}^0,\mathbf{x}^0\right) \triangleq \frac{\partial \mathbf{f}\left(\mathbf{u}^0,\mathbf{x}^0\right)}{\partial \mathbf{u}}. \tag{2-17}$$

Similarmente, podemos linearizar uma função de saída não linear $\mathbf{y} = \mathbf{h}(\mathbf{u},\mathbf{x})$, onde obtemos

$$\overline{\mathbf{y}} = \overline{\mathbf{C}}\left(\mathbf{u}^0,\mathbf{x}^0\right)\overline{\mathbf{x}} + \overline{\mathbf{D}}\left(\mathbf{u}^0,\mathbf{x}^0\right)\overline{\mathbf{u}}, \tag{2-18}$$

em que os Jacobianos $\overline{\mathbf{C}}$ e $\overline{\mathbf{D}}$ são dados por:

$$\overline{\mathbf{C}}\left(\mathbf{u}^0,\mathbf{x}^0\right) \triangleq \frac{\partial \mathbf{h}\left(\mathbf{u}^0,\mathbf{x}^0\right)}{\partial \mathbf{x}}, \tag{2-19}$$

$$\overline{\mathbf{D}}\left(\mathbf{u}^0,\mathbf{x}^0\right) \triangleq \frac{\partial \mathbf{h}\left(\mathbf{u}^0,\mathbf{x}^0\right)}{\partial \mathbf{u}}. \tag{2-20}$$

Para o caso em que as equações do sistema e de saída são dadas de modo implícito, consultar (JANSEN, 2013).

2.2
Fluxo monofásico

Nesta seção, exploraremos o caso de fluxo monofásico. Discorreremos a respeito de: sistemas de equações, considerando algumas hipóteses no fluxo do líquido; pressões e vazões prescritas, em que discutiremos a possibilidade de controlar o sistema via variáveis de estado; e modelos de poço, onde mostraremos como estabelecer uma relação algébrica entre a pressão e a taxa de fluxo no bloco da malha que contém o poço.

2.2.1
Sistemas de equações

Considerando o fluxo de um líquido monofásico fracamente compressível através de um meio poroso, e usando quaisquer métodos de discretização, por exemplo diferenças finitas, chega-se a um sistema de EDOs, escrito como

$$\mathbf{V}\dot{\mathbf{p}} + \mathbf{T}\mathbf{p} = \mathbf{q}, \tag{2-21}$$

em que $\mathbf{V}$ é a matriz de acumulação (diagonal); $\mathbf{T}$ é a matriz de transmissibilidade (simétrica de faixas); $\mathbf{p}$ é um vetor de pressões; $\mathbf{q}(\mathrm{m}^3/\mathrm{s})$ é um vetor de taxas de fluxo volumétrico. Cabe ressaltar que $\mathbf{V}$ e $\mathbf{T}$ são matrizes com entradas dependentes de propriedades dinâmicas e estáticas de reservatório.

Notemos que para a equação (2-21) ser reformulada na forma de variáveis de estado (2-10), definimos:

$$\mathbf{u} \triangleq \mathbf{L}_{uq}\mathbf{q}, \tag{2-22}$$

$$\mathbf{x} \triangleq \mathbf{p}, \tag{2-23}$$

em que $\mathbf{L}_{uq}$ denota a matriz de localização (contém zeros e uns nos lugares apropriados), $\mathbf{u}$ representa o vetor de entradas do sistema (elementos não nulos de $\mathbf{q}$), e dizer que $\mathbf{x} = \mathbf{p}$, significa que as variáveis de estado $\mathbf{x}$ são iguais às pressões $\mathbf{p}$.

Para uma descrição da representação generalizada do espaço de estados (2-11) e da forma ordinária do espaço de estado (2-10), ver (JANSEN, 2013).

2.2.2
Pressões e vazões prescritas

No que tange ao controle de sistema por pressões ou vazões, o mesmo pode ser regulado por vazões (taxas de fluxo) ou por pressões nos poços (JANSEN, 2013). Cabe comentar que não é possível prescrever simultaneamente pressão e vazão em um poço; deve-se fixar uma delas ou definir uma relação matemática entre ambas. No que concerne à Engenharia de Reservatórios, o método mais frequentemente utilizado é o de definir um *modelo de poço*, que veremos em detalhes na próxima subseção.

Outra forma de regulação do sistema é através da prescrição direta de uma das pressões, causando uma redução do comprimento do vetor de estado em um elemento. Para fazer isso, é conveniente reordenar as variáveis da equação (2-21), de forma que possa-se agrupar os valores prescritos e não prescritos. Os detalhes de como promover tal estruturação podem ser observados em (JANSEN, 2013).

2.2.3
Modelos de poço

Uma abordagem comum em simulação de reservatórios para prescrever pressão de fundo de poço é via definição de um *modelo de poço* (JANSEN, 2013). Os principais conceitos e equações são relativos à:

- Taxa de fluxo q em um bloco da malha que é determinada pela equação

$$q = J_{well}(\breve{p}_{well} - p), \tag{2-24}$$

em que J_{well} é o índice de poço, que depende da geometria do bloco da malha e reflete o efeito do fluxo próximo ao poço, que em geral não é representado adequadamente pela discretização via diferenças finitas. Já $\breve{p}_{well}$ é a pressão de fundo prescrita e p é a pressão do bloco da malha. Podemos pensar que o uso da equação (2-24) pode ser visto como a especificação de uma relação algébrica entre a pressão e a taxa de fluxo no bloco da malha que contém o poço;

• Generalização para múltiplos poços, em que pode-se expandir a equação (2-24) para

$$\mathbf{q}_3 = \mathbf{J}_3(\breve{\mathbf{p}}_{well} - \mathbf{p}_3), \tag{2-25}$$

com $\mathbf{J}_3$ sendo uma matriz diagonal de índices de poços J_{well}, e $\breve{\mathbf{p}}_{well}$ é um vetor de pressões de fundo prescritas;

• Representação no espaço de estados, em que o modelo é reescrito na forma de espaço de estados, como:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}, \tag{2-26}$$

$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u}, \tag{2-27}$$

em que o formato das matrizes $\mathbf{A}, \mathbf{B}, \mathbf{C}$ e $\mathbf{D}$, assim como os detalhes para obter as equações (2-26) e (2-27) podem ser consultados em (JANSEN, 2013).

De fato, ao exibir como obter (2-26) e (2-27) mostramos que o esquema de representação da Figura 2.1 é satisfeito para o caso monofásico. Diante disso, na próxima seção apresentaremos como verificar isso para o caso bifásico.

2.3 Fluxo bifásico

Descreveremos os sistemas de equações não lineares para o caso bifásico (óleo e água). Nessa descrição, veremos que os elementos-chave a serem observados estão relacionados às matrizes de acumulação, transmissibilidade e fluxo fracionário. Quanto à discussão do modelo de poço, novamente exibiremos o tipo de relação algébrica, já comentada anteriormente, referente à pressão do bloco da malha e à taxa de fluxo do poço. Posteriormente, mostraremos como incorporar o que foi dito no modelo de poço ao espaço de estado. Por fim, exibiremos o formato do vetor de saída estendido, que possui como elementos medições relativas às variáveis de saída e de entrada, respectivamente.

2.3.1
Equações do sistema

No contexto de sistema de equações não-lineares, podemos destacar a equação (2-28), que é relativa a uma descrição simplificada do fluxo bifásico de óleo e água, em que é relacionado variáveis como pressão ($\mathbf{p}$) e saturação ($\mathbf{s}$) com termos que dependem de várias propriedades do reservatório e das fases fluídicas, como compressibilidade, permeabilidade e viscosidade (JANSEN, 2013):

$$\underbrace{\begin{bmatrix} \mathbf{V}_{wp}(\mathbf{s}) & \mathbf{V}_{ws} \\ \mathbf{V}_{op}(\mathbf{s}) & \mathbf{V}_{os} \end{bmatrix}}_{\mathbf{V}} \begin{bmatrix} \dot{\mathbf{p}} \\ \dot{\mathbf{s}} \end{bmatrix} + \underbrace{\begin{bmatrix} \mathbf{T}_w(\mathbf{s}) & \mathbf{0} \\ \mathbf{T}_o(\mathbf{s}) & \mathbf{0} \end{bmatrix}}_{\mathbf{T}} \begin{bmatrix} \mathbf{p} \\ \mathbf{s} \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{F}_w(\mathbf{s}) \\ \mathbf{F}_o(\mathbf{s}) \end{bmatrix}}_{\mathbf{F}} \mathbf{q}_{well,t}. \quad (2\text{-}28)$$

Nesta equação, destaca-se:

• $\mathbf{V}$ - Matriz de acumulação, cujas entradas dependem da porosidade ϕ e compressibilidade das fases (c_o, c_w e c_r, isto é, compressibilidade do óleo, água e rocha, respectivamente);

• $\mathbf{T}$ - Matriz de transmissibilidade, que possui entradas dependentes da permeabilidade da rocha k, permeabilidades relativas (k_{ro}, k_{rw}) e viscosidades (μ_o, μ_w) das fases;

• $\mathbf{F}$ - Matriz de fluxo fracionário, que contém termos dependentes da saturação que relacionam as taxas de fluxo de óleo e água nos poços com as taxas de fluxo totais;

• $\mathbf{q}_{well,t}$ - É um vetor que representa as taxas de fluxo total dos poços, com valores não nulos nos elementos correspondentes aos blocos da malha penetrados por um poço.

Sob a ótica do *modelo de poço*, tem-se que os termos fonte em geral não são as taxas de fluxo nos poços, mas sim as pressões. Pode-se explicar esse fato, reescrevendo a equação (2-28) na forma particionada como

$$\left[\begin{array}{ccc|ccc} \mathbf{V}_{wp,11} & \mathbf{0} & \mathbf{0} & \mathbf{V}_{ws,11} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{V}_{wp,22} & \mathbf{0} & \mathbf{0} & \mathbf{V}_{ws,22} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{V}_{wp,33} & \mathbf{0} & \mathbf{0} & \mathbf{V}_{ws,33} \\ \hline \mathbf{V}_{op,11} & \mathbf{0} & \mathbf{0} & \mathbf{V}_{os,11} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{V}_{op,22} & \mathbf{0} & \mathbf{0} & \mathbf{V}_{os,22} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{V}_{op,33} & \mathbf{0} & \mathbf{0} & \mathbf{V}_{os,33} \end{array}\right] \left[\begin{array}{c} \dot{\mathbf{p}}_1 \\ \dot{\mathbf{p}}_2 \\ \dot{\mathbf{p}}_3 \\ \dot{\mathbf{s}}_1 \\ \dot{\mathbf{s}}_2 \\ \dot{\mathbf{s}}_3 \end{array}\right] +$$

$$\left[\begin{array}{ccc|ccc} \mathbf{T}_{w,11} & \mathbf{T}_{w,12} & \mathbf{T}_{w,13} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{T}_{w,21} & \mathbf{T}_{w,22} & \mathbf{T}_{w,23} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{T}_{w,31} & \mathbf{T}_{w,32} & \mathbf{T}_{w,33} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \hline \mathbf{T}_{o,11} & \mathbf{T}_{o,12} & \mathbf{T}_{o,13} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{T}_{o,21} & \mathbf{T}_{o,22} & \mathbf{T}_{o,23} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{T}_{o,31} & \mathbf{T}_{o,32} & \mathbf{T}_{o,33} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{array}\right] \left[\begin{array}{c} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \\ \mathbf{s}_1 \\ \mathbf{s}_2 \\ \mathbf{s}_3 \end{array}\right] \tag{2-29}$$

$$= \left[\begin{array}{ccc} \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{F}_{w,22} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{F}_{w,33} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{F}_{o,22} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{F}_{o,33} \end{array}\right] \left[\begin{array}{c} \mathbf{0} \\ \breve{\mathbf{q}}_{well,t} \\ \mathbf{J}_3 \left(\breve{\mathbf{p}}_{\text{well}} - \mathbf{p}_3\right) \end{array}\right].$$

Para fins de notação, os elementos do vetores $\mathbf{p}_1, \mathbf{p}_2$ e $\mathbf{p_3}$ correspondem, respectivamente, às:

1. pressões nos blocos da malha que não são penetrados por um poço;
2. pressões nos blocos em que os termos fonte são as taxas de fluxo total do poço prescritas $\breve{\mathbf{q}}_{well,t}$;
3. pressões nos blocos em que os termos fonte são obtidos via prescrição das pressões de fundo de poço $\breve{\mathbf{p}}_{well}$ com o auxílio de uma matriz diagonal de índices de poço $\mathbf{J}_3$.

Para calcular as taxas de fluxo de óleo e água nos poços com pressões prescritas, podemos fazer uso do modelo de poço, dado por:

$$\left[\begin{array}{c} \overline{\mathbf{q}}_{well,\ w} \\ \overline{\mathbf{q}}_{well,\ o} \end{array}\right] = \left[\begin{array}{c} \mathbf{F}_{w,33} \\ \mathbf{F}_{o,33} \end{array}\right] \mathbf{J}_3 \left(\breve{\mathbf{p}}_{well} - \mathbf{p}_3\right). \tag{2-30}$$

Já para calcular as pressões de fundo de poço $\overline{\mathbf{p}}_{well}$ nos poços com taxas de fluxo total prescritas, precisamos de uma matriz diagonal $\mathbf{J}_{q,2}$ de índice de poços de tal sorte que

$$\breve{\mathbf{q}}_{well,t} = \mathbf{J}_2 \left(\overline{\mathbf{p}}_{well} - \mathbf{p}_2 \right), \tag{2-31}$$

donde obtemos

$$\overline{\mathbf{p}}_{well} = \mathbf{J}_2^{-1} \breve{\mathbf{q}}_{well,t} - \mathbf{p}_2. \tag{2-32}$$

Quanto à forma do espaço de estado, com o objetivo de trazer essas equações para o formato do espaço de estado, definimos os vetores $\mathbf{x}$, $\mathbf{u}$ e $\mathbf{y}$, que referem-se, respectivamente, aos vetores de estado, entrada e saída, como:

$$\mathbf{u} \triangleq \begin{bmatrix} \breve{\mathbf{q}}_{well,t} \\ \breve{\mathbf{p}}_{well} \end{bmatrix}, \tag{2-33}$$

$$\mathbf{x} \triangleq \begin{bmatrix} \mathbf{p} \\ \mathbf{s} \end{bmatrix}, \tag{2-34}$$

$$\mathbf{y} \triangleq \begin{bmatrix} \overline{\mathbf{p}}_{well} \\ \overline{\mathbf{q}}_{well,w} \\ \overline{\mathbf{q}}_{well,o} \end{bmatrix}. \tag{2-35}$$

As equações (2-29), (2-30) e (2-32) podem ser reescritas na forma de espaço de estado não linear, ou seja,

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{u}, \mathbf{x}), \tag{2-36}$$

$$\mathbf{y} = \mathbf{h}(\mathbf{u}, \mathbf{x}), \tag{2-37}$$

em que as funções $\mathbf{f}$ e $\mathbf{h}$ são definidas no formato das equações (2-38) e (2-39), ou seja,

$$\mathbf{f} \triangleq \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}, \tag{2-38}$$

$$\mathbf{h} \triangleq \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u}. \tag{2-39}$$

Os detalhes das matrizes secantes dependentes do estado $\mathbf{A(x)}$, $\mathbf{B(x)}$ e $\mathbf{C(x)}$ e $\mathbf{D(x)}$, podem ser consultadas em (JANSEN, 2013). Vale pontuar que as equações (2-38) e (2-39) se assemelham à estrutura das equações (2-26) e (2-27), o que evidencia a similaridade entre os casos monofásico e bifásico.

Para fins de aplicações computacionais, em geral, é preciso expressar as equações do sistema em forma de espaço de estados completamente implícito, isto é,

$$\mathbf{g}(\mathbf{u}, \mathbf{x}, \dot{\mathbf{x}}) = \hat{\mathbf{E}}\dot{\mathbf{x}} - \hat{\mathbf{A}}\mathbf{x} - \hat{\mathbf{B}}\mathbf{u} = \mathbf{0}. \tag{2-40}$$

Para mais detalhes referentes à equação (2-40), consultar (JANSEN, 2013).

Vamos analisar agora o chamado vetor de saída estendido. Até aqui, consideramos saídas do sistema sob a ótica de sinais de resposta, e assumimos que todas as entradas do sistema eram conhecidas. Mas, em campo as entradas e saídas têm que ser mensuradas, sendo assim, podemos definir um vetor de saída estendido $\mathbf{y}$ que possui todos os sinais medidos, como

$$\mathbf{y} \triangleq \begin{bmatrix} \overline{\mathbf{p}}_{well} \\ \overline{\mathbf{q}}_{well,w} \\ \overline{\mathbf{q}}_{well,o} \\ \text{- - - - - - -} \\ \breve{\mathbf{p}}_{well} \\ \tilde{\tilde{\mathbf{q}}}_{well,w} \\ \tilde{\tilde{\mathbf{q}}}_{well,o} \end{bmatrix}, \tag{2-41}$$

em que adicionamos til para denotar que as variáveis foram mensuradas, em vez de prescritas. Em seu livro, (JANSEN, 2013) explica que na equação (2-41), há a inclusão de medições da pressão prescrita $\breve{\mathbf{p}}_{well}$, e que os elementos acima e abaixo da linha pontilhada ilustram medições relativas às variáveis de saída e entrada, respectivamente. Os detalhes do formato das matrizes $\mathbf{C}$ e $\mathbf{D}$ podem ser consultadas em (JANSEN, 2013).

Já que verificamos principalmente que a estrutura do sistema da Figura 2.1 é válida para os casos de fluxo monofásico e bifásico, iremos a partir daqui entender a trajetória de tópicos/temas que nos levaram ao entendimento téorico do motivo do modelo *Embed to Control and Observe* (E2CO) precisar ter sido formulado. O texto abaixo será baseado de forma significativa na tese doutorado de (COUTINHO, 2022).

Em geral, sabe-se que os poços podem ser controlados pela pressão de fundo de poço (BHP, sigla em inglês para *bottom-hole pressure*). Assumiremos que os poços produtores e injetores são controlados pela BHP e pela taxa de injeção (q_{inj}), respectivamente. Definimos o vetor de controle composto pelos controles de todos os poços no passo de tempo t, como

$$u^t := \begin{bmatrix} \text{BHP}_1^t \\ \vdots \\ \text{BHP}_p^t \\ q_{\text{inj},1}^t \\ \vdots \\ q_{\text{inj},i}^t \end{bmatrix}, \tag{2-42}$$

em que p, i dizem respeito ao número de poços produtores e injetores, respectivamente.

Já a evolução dinâmica do estado será resolvida para cada passo de tempo

t, em que o estado é denotado pelo vetor:

$$x^t = \begin{bmatrix} \mathrm{P}_1^t \\ \vdots \\ \mathrm{P}_n^t \\ \mathrm{S}_1^t \\ \vdots \\ \mathrm{S}_n^t \end{bmatrix}, \tag{2-43}$$

em que n refere-se ao número de blocos da malha.

A saída dos poços produtores é a taxa de fluxo (vazão) de óleo (q_o) e taxa de fluxo de água (q_w). Para os poços injetores, a saída é o BHP. Assim, o vetor de saída pode ser expresso como

$$y^t := \begin{bmatrix} \mathrm{q}_{o,1}^t \\ \vdots \\ \mathrm{q}_{o,p}^t \\ \mathrm{q}_{w,1}^t \\ \vdots \\ \mathrm{q}_{w,p}^t \\ \mathrm{BHP}_1^t \\ \vdots \\ \mathrm{BHP}_i^t \end{bmatrix}, \tag{2-44}$$

e (COUTINHO, 2022) expressa que tais valores são calculados usando a equação de Peaceman (PEACEMAN, 1978).

Notemos que (COUTINHO, 2022) considerou as hipóteses fornecidas por (JANSEN, 2013), e ao realizar a aplicação da discretização totalmente implícita, torna-se possível escrever a forma residual da equação (2-2) como

$$g\left(x^{t+1}, u^{t+1}\right) = \underbrace{\mathbf{F}\left(x^{t+1}\right)}_{\text{termo de fluxo}} + \underbrace{\mathbf{Acc}\left(x^{t+1}, x^t\right)}_{\text{termo de acumulação}} + \underbrace{\mathbf{Q}\left(x^{t+1}, u^{t+1}\right)}_{\text{termo fonte}} = 0. \tag{2-45}$$

Pode-se resolver essa equação via método de Newton. Então, a evolução do estado pode ser calculada através de

$$x^{t+1} = x^t - \frac{\mathbf{F}\left(x^{t+1}\right) + \mathbf{Acc}\left(x^{t+1}, x^t\right) + \mathbf{Q}\left(x^{t+1}, u^{t+1}\right)}{\mathbf{J}^{t+1}}, \tag{2-46}$$

em que $\mathbf{J} = \dfrac{\partial g}{\partial x}$.

Com fins de aplicar técnicas de modelagem de ordem reduzida à simulação de reservatórios de forma eficaz, pode-se linearizar a equação residual (2-45) ou (2-46). Como visto nas discussões acima, através da abordagem do

Sistema de Controle, uma representação de espaço de estado de um sistema linear pode ser escrita como

$$\begin{cases} \dot{x} = \mathbf{A}_c x + \mathbf{B}_c u \\ y = \mathbf{C}_c x + \mathbf{D}_c u \end{cases} . \tag{2-47}$$

Na Tabela 2.1, temos a descrição dos elementos envolvidos na equação (2-47), cuja primeira equação é referente a evolução de estado, em que será aplicado o método de linearização.

Elemento	Descrição
x	estado
$\dot{x}$	derivada de tempo de estado
u	entradas/controles
y	saída do sistema
$\mathbf{A}_c$	matriz de estado em tempo contínuo
$\mathbf{B}_c$	matriz de entrada em tempo contínuo
$\mathbf{C}_c$	matriz de saída em tempo contínuo
$\mathbf{D}_c$	matriz de transmissão em tempo contínuo

Tabela 2.1: Descrição dos elementos da equação (2-47). Fonte: Adaptado de (COUTINHO, 2022).

A seguir, vamos usar o método de linearização por partes da trajetória (ou TPWL - sigla em inglês) para fazer uma aproximação das matrizes $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$ em cada tempo de interesse. Isso se deve, pois de modo geral precisa-se escolher um ponto de linearização no tempo e usar o jacobiano neste momento para calcular as matrizes supracitadas, mas caso a dinâmica do sistema seja fortemente alterada com o passar do tempo, escolher um tempo fixo pode acarretar em grandes erros de estimativa.

2.4 Linearização por partes da trajetória

Tal método é dotado de duas etapas. A primeira é relativa ao processamento *offline* ou procedimento de treinamento, e a segunda é o processamento *online*. A etapa 1, pode ser explicada seguindo o fluxograma exibido na Figura 2.2.

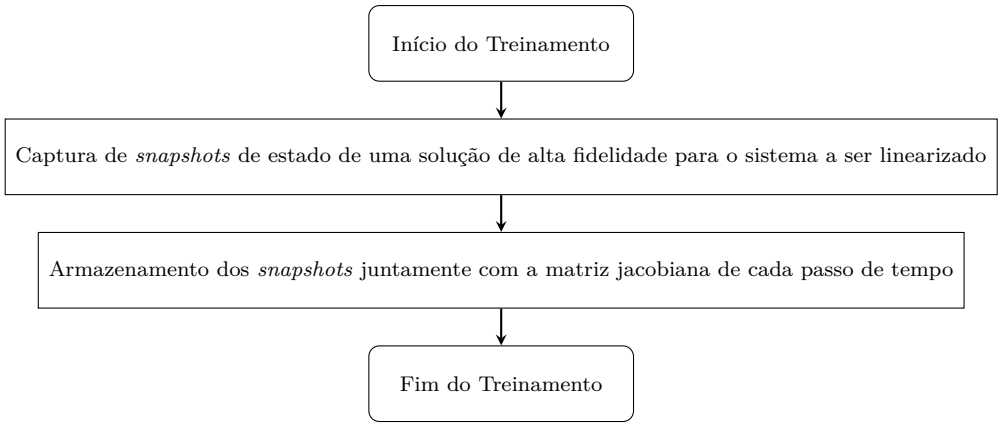

Figura 2.2: Fluxograma do processo de treinamento do TPWL. Fonte: Adaptado de (COUTINHO, 2022).

Já na etapa 2, basicamente usa-se os estados armazenados na etapa de treinamento em ordem para prever o novo estado. Desse modo, se busca prever o estado no próximo passo de tempo x^{t+1}, baseando-se no estado atual x^t e controles u^t. Primeiramente, no grupo de *snapshots* (instantâneos) de estados armazenados, identificamos o *snapshot* x^i próximo a x^t. Então, faz-se uso do estado armazenado e do Jacobiano para i e $i+1$, visando calcular o estado x^{t+1} empregando a equação

$$x^{t+1} = x^{i+1} - \frac{\mathbf{F}^{i+1} + \mathbf{Acc}^{i+1} + \dfrac{\partial \mathbf{Acc}^{i+1}}{\partial x^i}\left(x^t - x^i\right) + \mathbf{Q}\left(x^{i+1}, u^{t+1}\right)}{\mathbf{J}^{i+1}}. \quad (2\text{-}48)$$

A etapa 2 pode ser resumida no fluxograma ilustrado na Figura 2.3.

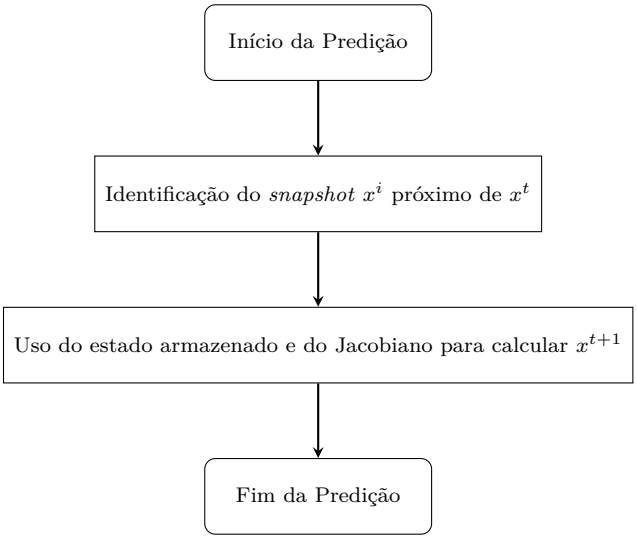

Figura 2.3: Fluxograma do processo de predição do TPWL. Fonte: Adaptado de (COUTINHO, 2022).

Assim como pôde-se escrever a equação (2-46) como um sistema linearizado (2-47), similarmente podemos reescrever a equação TPWL (2-48) como um sistema dado por

$$\begin{cases} x^{t+1} = \mathbf{A}^i x^t + \mathbf{B}^i u^t \\ y^{t+1} = \mathbf{C}^i x^{t+1} + \mathbf{D}^i u^t \end{cases}, \tag{2-49}$$

em que $\mathbf{A}^i, \mathbf{B}^i, \mathbf{C}^i$ e $\mathbf{D}^i$ indicam a versão discretizada no tempo das matrizes contínuas, sendo que o sobrescrito i diz respeito às matrizes linearizadas usando informações do *snapshot* i.

Observemos que para resolver a equação (2-48) e para calcular as matrizes $\mathbf{A}^i, \mathbf{B}^i, \mathbf{C}^i$ e $\mathbf{D}^i$ é necessário inverter o Jacobiano, o que é custoso computacionalmente. Para resolver esse problema de inversão do Jacobiano, usaremos a técnica de decomposição ortogonal própria, que é um método de redução de ordem do modelo.

2.5
Decomposição ortogonal própria

Seja $\mathbf{X} = [\ x^1 \quad x^2 \quad \dots \quad x^n\ ]$ o arranjo dos estados armazenados no passo de treinamento do TPWL. Vamos aplicar a decomposição em valores singulares em $\mathbf{X}$, e escolher apenas os l maiores valores singulares, baseando-se em alguns critérios. Desse modo, a base POD (*Proper Orthogonal Decomposition*) Φ é definida como os l vetores singulares direitos, e o espaço do estado reduzido/latente z é então definido como a projeção do espaço original x baseado na base POD.

Escolhendo $z = \Phi^T x$ e considerando a equação (2-48), podemos reescrevê-la ao utilizar a relação $x \approx \Phi z$ e multiplicar ambos os lados por Φ^T, obtendo:

$$\begin{aligned} \underbrace{\Phi^T \mathbf{J}^{i+1} \Phi}_{\mathbf{J}_r^{i+1}} \left(z^{t+1} - z^{i+1}\right) = -\ [\ \underbrace{\Phi^T \mathbf{F}^{i+1}}_{\mathbf{F}_r^{i+1}} + \underbrace{\Phi^T \mathbf{Acc}^{i+1}}_{\mathbf{Acc}_r^{i+1}} + \underbrace{\Phi^T \frac{\partial \mathbf{Acc}^{i+1}}{\partial x^i} \Phi}_{\left(\frac{\partial \mathbf{Acc}^{i+1}}{\partial x^i}\right)_r} \left(z^t - z^i\right) + \\ + \underbrace{\Phi^T \mathbf{Q}}_{\mathbf{Q}_r} \left(x^{i+1}, u^{t+1}\right)]. \end{aligned} \tag{2-50}$$

Fazendo as devidas adaptações de notação, podemos reescrever a equação (2-50) como uma equação POD-TPWL, dada por

$$z^{t+1} = z^{i+1} - \frac{\left[\mathbf{F}_r^{i+1} + \mathbf{Acc}_r^{i+1} + \left(\frac{\partial \mathbf{Acc}^{i+1}}{\partial x^i}\right)_r (z^t - z^i) + \mathbf{Q}_r\left(x^{i+1}, u^{t+1}\right)\right]}{\mathbf{J}_r^{i+1}}. \tag{2-51}$$

Truncando o espaço latente, trabalhamos com matrizes de dimensão reduzida quando comparadas às equações TPWL. Podemos reescrever (2-51) na estrutura de um sistema de variantes de tempo linear dado por

$$\begin{cases} z^{t+1} = \mathbf{A}_r^i z^t + \mathbf{B}_r^i u^t \\ y^{t+1} = \mathbf{C}_r^i z^{t+1} + \mathbf{D}_r^i u^t \end{cases}, \tag{2-52}$$

em que $\mathbf{A}_r^i, \mathbf{B}_r^i, \mathbf{C}_r^i$ e $\mathbf{D}_r^i$ representam a versão reduzida apresentada em (2-49).

Cabe comentar que (WATTER et al., 2015) desenvolveram um método para o cálculo das matrizes $\mathbf{A}_r^i$ e $\mathbf{B}_r^i$ baseado em *deep-learning*, usando os *snapshots* de estado como dados de treinamento. Já (JIN; LIU; DURLOFSKY, 2020) fizeram uma extensão dessa ideia acrescentando uma função de perda física relacionada ao processo de simulação de reservatório. Uma das contribuições da tese de doutorado de (COUTINHO, 2022) foi expandir e demonstrar a viabilidade de extrair os estados e saídas. Tal feito foi nomeado como *Embed to Control and Observe* (E2CO), mas antes de comentarmos sobre esse método, iremos discorrer no próximo capítulo sobre aplicações em *Machine Learning* que possibilitaram o seu desenvolvimento.

3
O modelo E2CO

3.1
Aplicações em *Machine Learning*

Primeiramente, iremos apresentar aplicações em *Machine Learning* que levaram ao desenvolvimento do E2CO.

3.1.1
Embed to Control

Conhecido como E2C, e originalmente criado por (WATTER et al., 2015), os autores citam que este é um método para aprendizado de modelos e controle de sistemas dinâmicos não lineares com base em imagens em *pixels* brutos. Resumidamente, o E2C expressa-se como um modelo generativo profundo, que se situa na família dos *autoencoders* variacionais, que aprende a gerar trajetórias de imagens via um espaço latente em que a dinâmica é restrita a ser localmente linear. Tal modelo procede diretamente de uma formulação de controle ótimo no espaço latente, suportando previsões de longo prazo de sequências de imagens, apresentando um desempenho robusto numa variedade de problemas complexos de controle.

Na Figura 3.1, tem-se um diagrama que exibe os principais elementos para o treinamento do E2C. A partir de agora, voltaremos a citar o trabalho de (COUTINHO, 2022).

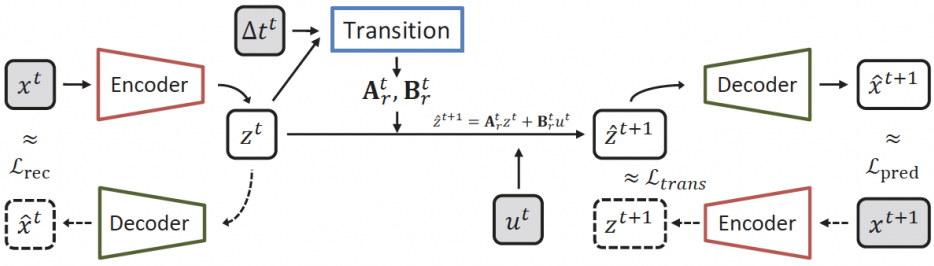

Figura 3.1: Diagrama para treinamento do E2C. Fonte: (COUTINHO, 2022).

Na Tabela 3.1, podemos notar uma descrição detalhada dos elementos expostos no diagrama do E2C.

Elemento(s)	Descrição
$x^t, x^{t+1}, u^t, \Delta t^t$	*inputs* usados para treinar o modelo
$\hat{x}^t$, $\hat{z}^{t+1}$, $\hat{x}^{t+1}$	valores estimados
$\rightarrow$	empregadas nos procedimentos de treinamento e predição
$\dashrightarrow$	empregadas nos procedimentos de treinamento

Tabela 3.1: Descrição dos elementos do diagrama do E2C. Fonte: Adaptado de (COUTINHO, 2022).

Por extenso, o diagrama (Figura 3.1) ilustra que o estado x^t é processado pelo *encoder*, resultando no vetor z^t no espaço reduzido. Em seguida, as informações contidas em z^t são decodificadas pelo *decoder*, levando ao estado previsto $\hat{x}^t$. O elemento de transição faz uso do estado no espaço latente para gerar as matrizes $\mathbf{A}_r^t$ e $\mathbf{B}_r^t$. Essas matrizes são empregadas para lidar com a evolução do sistema dinâmico, usando uma abordagem de sistema de controle linear, como em

$$\hat{z}^{t+1} = \mathbf{A}_r^t z^t + \mathbf{B}_r^t u^t. \tag{3-1}$$

Após usar tal abordagem, passamos a informação pelo *decoder* e obtemos finalmente o estado previsto $\hat{x}^{t+1}$. Já para obter z^{t+1}, precisamos fazer uso do *input* x^{t+1}, onde o processamos pelo *encoder* e daí obtemos finalmente z^{t+1}. Cabe comentar que de certa forma, podemos dizer que o *encoder* é equivalente à projeção POD (*Proper Orthogonal Decomposition*), no sentido de que ambos conseguem transformar o estado original x no estado do espaço latente z.

Observemos que a equação (3-1) está situada no espaço latente, logo pode-se associar $\mathbf{A_r^t}$ e $\mathbf{B_r^t}$ com suas versões reduzidas utilizadas na equação $(2-52)$. Cabe frisar que as matrizes $\mathbf{A_r^t}$ e $\mathbf{B_r^t}$ são modificadas para cada estado de entrada distinto x^t, assim como em outros procedimentos de linearização, tais como TPWL (*trajectory piecewise linearization*).

Destaca-se também que outros elementos importantes incluem as três funções de perda ($\mathcal{L}$) que serão utilizadas durante a fase de treinamento. No âmbito desta discussão, a função de perda de reconstrução $(\mathcal{L})_{rec}$ é adicionada com fins de garantir que o *autoencoder* representado pelo conjunto *enconder* e *decoder* possa reconstruir com êxito o estado $\hat{x}^t$ o mais próximo possível de x^t. Tal estado $\hat{x}^t$ é dado pela seguinte composição

$$\hat{x}^t := Decoder\Big(Encoder\Big(x^t\Big)\Big). \tag{3-2}$$

Já a função de perda de reconstrução é dada por

$$(\mathcal{L}_{rec})_i := \left\{ \left\| x^t - \hat{x}^t \right\|_2^2 \right\}_i = \left\{ \sum_{j=1}^{2n} \left| x_j^t - \hat{x}_j^t \right|^2 \right\}_i, \tag{3-3}$$

com i denotando o índice da amostra.

Por estarmos inseridos no contexto de um sistema dinâmico, ao minimizar a perda de predição, estamos garantindo a acurácia da evolução de tal sistema com o passar do tempo. De posse disso, definimos a função de perda de predição pela composição

$$\hat{x}^{t+1} := \mathit{Decoder}\Big(\mathit{Transition}\Big(\mathit{Encoder}\Big(x^t\Big)\Big)\Big), \tag{3-4}$$

em que $\hat{x}^{t+1}$ é a predição do estado de variáveis em $t+1$, de tal sorte que é possível calcular a perda de predição como

$$(\mathcal{L}_{pred})_i := \left\{\left\|x^{t+1} - \hat{x}^{t+1}\right\|_2^2\right\}_i. \tag{3-5}$$

Por fim, a terceira função de perda é conhecida como função de perda de transição, que visa garantir a capacidade do modelo em evoluir o estado reduzido z com o passar do tempo. Isso se assemelha a ideia do MOR (*model order reduction*), sob a ótica do desenvolvimento de um *framework* de projeção em que a evolução do estado reduzido no tempo é baseada no Jacobiano reduzido. Primeiramente, vamos calcular

$$\hat{z}^{t+1} := \mathit{Transition}\Big(\mathit{Encoder}\Big(x^t\Big)\Big), \tag{3-6}$$

de tal forma que

$$z^{t+1} := \mathit{Encoder}\Big(x^{t+1}\Big). \tag{3-7}$$

Vejamos que com base nas equações (3-6) e (3-7), podemos calcular a função perda de transição da seguinte forma:

$$(\mathcal{L}_{trans})_i := \left\{\left\|z^{t+1} - \hat{z}^{t+1}\right\|_2^2\right\}_i. \tag{3-8}$$

Vamos expor na próxima subseção, a ideia de (JIN; LIU; DURLOFSKY, 2020) que melhora o E2C adicionando uma função de perda física especificamente projetada para a classe de problemas de simulação de reservatório.

3.1.2 *Embed to Control* acoplado com função de perda física

Fundamentando-se no E2C, (JIN; LIU; DURLOFSKY, 2020) criaram um novo *framework* de modelagem de ordem reduzida (ROM) baseada em *deep learning* com fins de aplicá-la em simulação de fluxo subterrâneo. O ROM inclui um *auto-encoder*, que realiza a projeção do sistema para um subespaço de dimensão baixa, e um modelo de transição linear se encarrega de aproximar a evolução dos estados do sistema em baixa dimensão. Os autores introduziram ainda uma função de perda baseada em física (equação 3-9)

$$
\begin{aligned}
(\mathcal{L}_p)_i = & \overbrace{\left\{ \left\| k \cdot \left[\left(\nabla p^t - \nabla \hat{p}^t \right)_{recon} + \left(\nabla p^{t+1} - \nabla \hat{p}^{t+1} \right)_{pred} \right] \right\|_2^2 \right\}_i}^{\text{função de perda de fluxo}} \\
& + \underbrace{\gamma \left\{ \left\| \left(q^{w,t} - \hat{q}^{w,t} \right)_{recon} + \left(q^{w,t+1} - \hat{q}^{w,t+1} \right)_{pred} \right\|_2^2 \right\}_i}_{\text{incompatibilidade na taxa de fluxo dos poços produtores}},
\end{aligned}
\tag{3-9}
$$

que penaliza previsões que são inconsistentes com as equações de fluxo governantes, e modificaram tal função com fins de enfatizar a precisão, por exemplo das taxas de produção de fluidos.

A Tabela 3.2, exibe a descrição dos elementos presentes na equação (3-9).

Elemento(s)	**Descrição**
n_b	número de blocos da malha
n_w	número total de poços
γ	define os pesos para a perda de dados de poços em $\mathcal{L}_p$
$p^t, p^{t+1} \in \mathbb{R}^{n_b}$	campos de pressão em t e $t+1$
$\hat{p}^t, \hat{p}^{t+1} \in \mathbb{R}^{n_b}$	reconstrução da pressão do ROM
$q^{w,t}, q^{w,t+1} \in \mathbb{R}^{n_w}$	número de poços dos dados de treinamento (trein.)
$\hat{q}^{w,t}, \hat{q}^{w,t+1} \in \mathbb{R}^{n_w}$	número de poços reconstruídos e preditos pelo ROM

Tabela 3.2: Descrição dos elementos do diagrama do E2C com função de perda física. Fonte: Adaptado de (JIN; LIU; DURLOFSKY, 2020).

Cabe expressar que na equação (3-9), t e $t+1$ representam passos de tempo, e que p^t e p^{t+1} que são os campos de pressão dos dados de treinamento (componentes das variáveis de estado x^t e x^{t+1}). Em $\hat{p}^t$ e $\hat{p}^{t+1}$ a reconstrução da pressão do ROM decorre em t, conforme definido após a equação (3-3). Claramente em $\hat{q}^{w,t}$, a quantidade de poços reconstruídos ocorre em t, e em $\hat{q}^{w,t+1}$, a quantidade de poços preditos acontece em $t+1$.

Uma explicação mais detalhada da importância de introduzir a equação (3-9) reside no fato de que era necessário abordar o problema inserido no seguinte contexto: o ROM é um modelo puramente orientado a dados, ou seja, objetiva-se minimizar a diferença *pixel* a *pixel* entre a saída E2C e a solução de referência "verdadeira" e/ou *high-fidelity solution*. Nesse sentido, (JIN; LIU; DURLOFSKY, 2020) declaram que o E2C prevê, até certo ponto, o comportamento físico via *snapshots* de pressão e saturação de entrada, todavia esse comportamento não é explicitamente imposto. Caso o ROM seja treinado utilizando a função de perda (3-10),

$$\mathcal{L}_d = \frac{1}{N_t}\sum_{i=1}^{N_t}\left(\left(\mathcal{L}_{\mathrm{R}}\right)_i + \left(\mathcal{L}_{\mathrm{PD}}\right)_i + \lambda\left(\mathcal{L}_{\mathrm{T}}\right)_i\right) \tag{3-10}$$
$$= \frac{1}{N_t}\sum_{i=1}^{N_t}\left(\|\mathbf{x}^t - \hat{\mathbf{x}}^t\|_2^2 + \|\mathbf{x}^{t+1} - \hat{\mathbf{x}}^{t+1}\|_2^2 + \lambda\|\mathbf{z}^{t+1} - \hat{\mathbf{z}}^{t+1}\|_2^2\right)_i,$$

pode ocorrer que efeitos não físicos sejam observados. Esse comportamento é exposto na Figura (3.2), em que se observam previsões para o campo de pressão num momento específico.

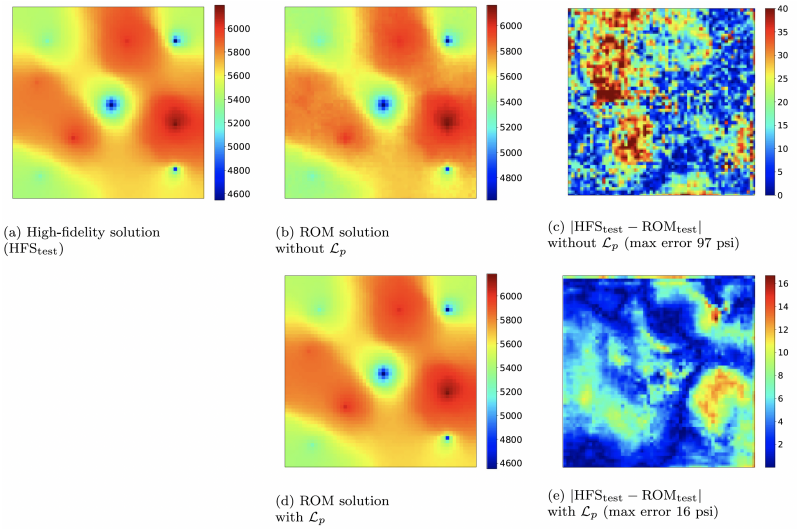

Figura 3.2: Predição de campos de pressão com e sem $\mathcal{L}_p$ (todas as unidades de barras coloridas estão em psi). Fonte: (JIN; LIU; DURLOFSKY, 2020).

Na Tabela 3.3, temos a descrição dos elementos da equação (3-10).

Elemento	Descrição
$i = 1, \ldots, N_t$	ponto de dados de treinamento (trein.)
$N_t = N_s - n_{train}$	n.º de dados (pontos) gerados nas simulações de trein.
N_s	n.º total de *snapshots* nas execuções de treinamento
n_{train}	n.º de simulações de treinamento executadas
$\mathbf{x}^t$	variável de estado em t numa simulação de trein.
$\hat{\mathbf{x}}^t$	estados reconstruídos pelo *encoder* e *decoder*
$\mathbf{x}^{t+1}$	variável de estado em $t+1$ das simulações de trein.
$\hat{\mathbf{x}}^{t+1}$	variável de estado de ordem completa prevista via ROM
$\mathbf{z}^{t+1}$	variável latente codificada em $t+1$
$\hat{\mathbf{z}}^{t+1}$	variável latente prevista pelo modelo de transição linear
λ	termo de peso

Tabela 3.3: Descrição dos elementos da equação (3-10). Fonte: Adaptado de (JIN; LIU; DURLOFSKY, 2020).

Vale comentar que para obter o formato da equação (3-10), precisamos adaptar as notações de (JIN; LIU; DURLOFSKY, 2020) às de (COUTINHO,

2022), isto é, consideramos que $(\mathcal{L}_R)_i = (\mathcal{L}_{rec})_i$, $(\mathcal{L}_{\text{PD}})_i = (\mathcal{L}_{pred})_i$ e $(\mathcal{L}_T)_i = (\mathcal{L}_{trans})_i$. Ainda nesta equação, notemos que N_t e N_s são distintos, pois para uma simulação de treinamento contendo N_{tr} *snapshots*, apenas $N_{tr} - 1$ pontos de dados podem ser coletados, já que pares de estados, em passos de tempo sequenciais são necessários.

Dado o exposto acima, em seu trabalho (JIN; LIU; DURLOFSKY, 2020) relatam que apesar dos dois resultados da predição de campos de pressão aparentarem ser visualmente similares, o mapa de diferenças (Figura 3.2 - item c) mostra que o E2C não é suficientemente suave, e erros relativamente expressivos são notados em alguns locais espaciais. Os autores reforçam que isso pode impactar de forma significativa nas previsões de taxa de poço, que são uma saída crucial do ROM, e justifica que tal impacto significativo nas previsões de taxa de poço os levaram a combinar a perda por incompatibilidade de dados com uma função de perda que considera informações envolvendo a física do fluxo. Com isso, buscou-se minimizar a inconsistência no fluxo entre cada par de blocos de malha adjacentes. Também foi feita a adição de peso extra à quantidades essenciais do poço. Dessa forma, foi considerado tanto a reconstrução, no passo de tempo t quanto a previsão no passo de tempo $(t+1)$.

Para finalizar a discussão em torno da equação (3-9), (COUTINHO, 2022) declara que a parte referente a função de perda de fluxo visa garantir que o fluxo de fluido entre blocos de malha adjacentes no modelo previsto seja o mais fiel possível do modelo verdadeiro. Já na parte que lida com a incompatibilidade na taxa de fluxo dos poços, (COUTINHO, 2022) comenta que (JIN; LIU; DURLOFSKY, 2020) utilizaram essa parte da função de perda objetivando garantir que a pressão do bloco de malha dos produtores nos estados previstos e reconstruídos estivessem o mais contíguo possível de seu equivalente no modelo verdadeiro, e que nessa referida parte foi adicionado na rede neural mais peso à pressão do bloco de malha dos poços produtores durante a etapa de treinamento, ao invés de implementar mais física na mesma. Em sua pesquisa, (COUTINHO, 2022) também relata que (JIN; LIU; DURLOFSKY, 2020) utilizaram somente a pressão do bloco de malha dos poços produtores, ao invés da taxa de fluxo em sua função de perda física.

Para ilustrar melhor os passos relevantes no entendimento dos procedimentos empregados para treinar o E2C com perda de fluxo, consideremos o diagrama da Figura 3.3.

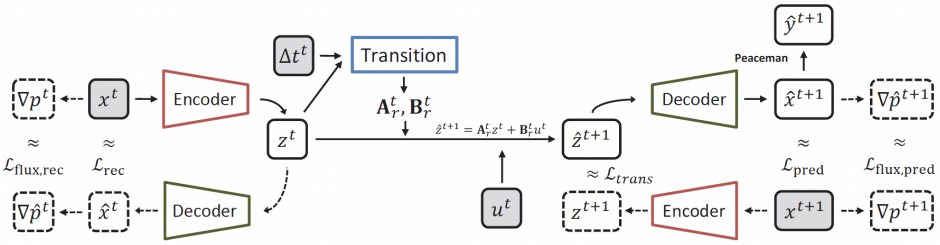

Figura 3.3: Diagrama para treinamento do E2C com perda de fluxo. Fonte: (COUTINHO, 2022).

Como apontado por (COUTINHO, 2022), quando se compara o diagrama da Figura 3.3 com o da Figura 3.1, nota-se no canto superior direito a adição do cálculo do gradiente em ambos os lados, assim como o cálculo das funções de perda de fluxo $(\mathcal{L}_{flux})_i$ nas partes de reconstrução e predição, que é dado por

$$\begin{aligned}(\mathcal{L}_{flux})_i &= (\mathcal{L}_{flux,rec})_i + (\mathcal{L}_{flux,pred})_i \\ &= k\left\{\left\|\nabla p^t - \nabla \hat{p}^t\right\|_2^2 + \left\|\nabla p^{t+1} - \nabla \hat{p}^{t+1}\right\|_2^2\right\}_i,\end{aligned} \tag{3-11}$$

em que k representa a permeabilidade.

Por fim, para completar esta subseção, (JIN; LIU; DURLOFSKY, 2020) pontuam que a função de perda baseada na física é calculada tomando a média sobre todos os pontos de dados. Em simbologia matemática, temos:

$$\mathcal{L}_p = \frac{1}{N_t}\sum_{i=1}^{N_t}(\mathcal{L}_p)_i. \tag{3-12}$$

Combinando a perda por incompatibilidade de dados com a dita perda fundamentada na física, temos que a função de perda total é dada por

$$\mathcal{L} = \mathcal{L}_d + \alpha\mathcal{L}_p, \tag{3-13}$$

em que α refere-se ao termo de peso. Através de experimentos numéricos, (JIN; LIU; DURLOFSKY, 2020) encontraram, por exemplo, o melhor α como sendo $\alpha = 0.033$, e concluíram que houve uma expressiva melhoria da previsão de ROM ao adicionar $\mathcal{L}_p$ na função de perda, já que o erro máximo de pressão passou de 97 psi para 16 psi, conforme descrito na Figura 3.2, o que consequentemente acarreta num campo de pressão mais físico, sob a ótica dele ser mais suave, além de exibir a vantagem de incluir perdas baseadas em física no *E2C ROM*.

Em seguida, exibiremos a nova forma encontrada por (COUTINHO, 2022) para lidar com os dados do poço (BHP - *Bottom-hole pressure* e taxas de fluxo de óleo e água).

3.1.3
Embed to control acoplado com dados de poço e funções de perda de fluxo bifásico

A motivação para o desenvolvimento desta subseção partiu do esforço de (COUTINHO, 2022), no sentido de reconstruir as saídas (BHP e taxas de fluxo de óleo e água) e os estados (pressão e saturação), ou seja, ele buscou dar ênfase à previsão de dados de poço, não apenas aos estados.

Com o objetivo de realizar tal propósito, (COUTINHO, 2022) propôs uma função de perda de dados de poço denominada $\mathcal{L}_{well\_data2}$. Na Figura 3.4, é possível notar a inclusão de tal função no canto superior direito.

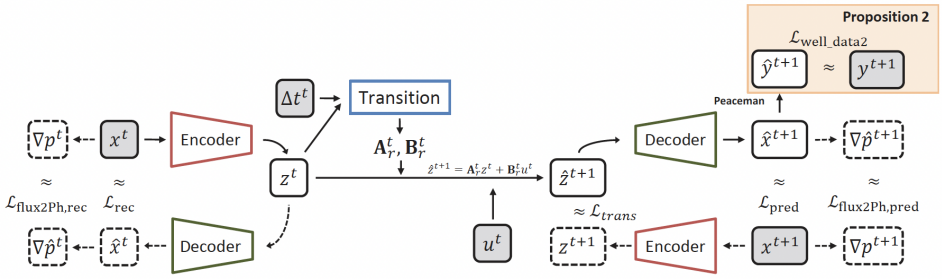

Figura 3.4: Diagrama de treinamento para E2C com fluxo bifásico e funções de perda de dados de poço (proposição 2). Fonte: (COUTINHO, 2022).

A adição da função de perda de dados é de suma importância segundo o autor, pois tornará os dados de poço previstos pelo modelo ($\hat{y}^{t+1}$) o mais contíguo possível do valor considerado verdadeiro (y^{t+1}). Além disso, vale destacar que os vetores de saída y^t e $\hat{y}^t$ têm por componentes as vazões de óleo e água e o BHP para cada passo de tempo t, conforme já visto, por exemplo, na equação (2-44).

Considera-se que a entrada empregada no treinamento do modelo são referentes aos dados de poço calculados por uma simulação de reservatório (provenientes do simulador comercial IMEX - (CMG, 2024)). De forma similar às já vistas anteriormente, (COUTINHO, 2022) construiu a função de perda de dados, de tal sorte que satisfaz a equação abaixo,

$$\left(\mathcal{L}_{well\_data2}\right)_i = \left\{ \left\| y^{t+1} - \hat{y}^{t+1} \right\|_2^2 \right\}_i . \tag{3-14}$$

O autor introduziu ainda uma função de perda de fluxo bifásica (equação 3-15)

$$
\begin{aligned}
\left(\mathcal{L}_{\textit{flux 2Ph}}\right)_i &= \left(\mathcal{L}_{\textit{flux 2Ph, rec }}\right)_i + \left(\mathcal{L}_{\textit{flux 2Ph, pred }}\right)_i \\
&= k\Big\{ \left\| k_{r,o}\left(S_w^t\right)\nabla p^t - k_{r,o}\left(\hat{S}_w^t\right)\nabla \hat{p}^t \right\|_2^2 + \\
&\quad + \left\| k_{r,w}\left(S_w^t\right)\nabla p^t - k_{r,w}\left(\hat{S}_w^t\right)\nabla \hat{p}^t \right\|_2^2 + \\
&\quad + \left\| k_{r,o}\left(S_w^{t+1}\right)\nabla p^{t+1} - k_{r,o}\left(\hat{S}_w^{t+1}\right)\nabla \hat{p}^{t+1} \right\|_2^2 + \\
&\quad + \left\| k_{r,w}\left(S_w^{t+1}\right)\nabla p^{t+1} - k_{r,w}\left(\hat{S}_w^{t+1}\right)\nabla \hat{p}^{t+1} \right\|_2^2 \Big\}_i ,
\end{aligned}
\tag{3-15}
$$

com vistas a obter uma previsão de estado mais transparente, e por conseguinte levando a uma melhor estimativa de dados de poço. Destacamos que k, $k_{r,j}$ e S_w denotam a permeabilidade absoluta, a permeabilidade relativa a fase j (óleo ou água), e a saturação de água, respectivamente.

Por fim, o autor explica que ao adicionar $\mathcal{L}_{well\_data2}$, espera-se uma melhoria na capacidade do modelo de prever os dados do poço, posto que tais informações foram incluídas no processo de treinamento da rede neural. Mais uma vez, isso difere do que (JIN; LIU; DURLOFSKY, 2020) propuseram, sob a ótica de que os dados de poço não faziam parte do treinamento, ou seja, foram calculados nas etapas de previsão embasados puramente nos estados (pressão e saturação).

Finalmente, após uma extensa revisão do estado da arte, passaremos a explorar a teoria do principal tópico teórico desta dissertação, denominado *Embed to Control and Observe*.

3.2 *Embed to Control and Observe*

Baseando-se no modelo E2C, (COUTINHO, 2022) estendeu essa ideia para considerar a saída dos dados, e nomeou a nova técnica como *Embed to control and observe*. A inovação de tal método foi introduzir uma nova rede de transição, dita saída de transição (*transition output*) que recebe dados do estado no espaço latente (z_t), de tal sorte que é possível gerar as matrizes $\mathbf{C}_r^t$ e $\mathbf{D}_r^t$, com fins de estimar os dados do poço $\hat{y}^{t+1}$, em que

$$
\hat{y}^{t+1} = \mathbf{C}_r^t \hat{z}^{t+1} + \mathbf{D}_r^t u^t. \tag{3-16}
$$

Isso foi necessário, pois no modelo E2C de autoria de (WATTER et al., 2015), uma rede de transição (cuja entrada é o estado no espaço reduzido) foi utilizada para gerar as matrizes $\mathbf{A}_r^t$ e $\mathbf{B}_r^t$, que são úteis na evolução do estado no espaço latente, baseado na equação (3-1). Veremos que de fato, a criação da tal nova rede de transição mencionada no primeiro parágrafo desta seção promoveu melhorias na previsão dos dados $\hat{y}^{t+1}$.

Agora, naturalmente, diante do contexto visto nas subseções anteriores, fica claro que o autor precisou produzir uma nova função de perda, cujo formato de escrita matemática já é conhecido, ou seja,

$$(\mathcal{L}_{well\_data3})_i = \left\{ \left\| y^{t+1} - \hat{y}^{t+1} \right\|_2^2 \right\}_i . \tag{3-17}$$

Um ponto importante a destacar, é que a função de perda acima concede dados para treinar os parâmetros de saída de transição da rede neural.

Na Figura 3.5, temos o diagrama do novo método.

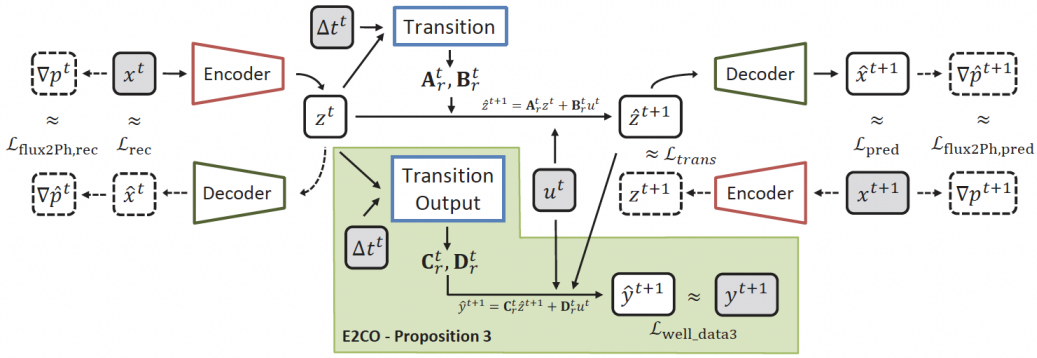

Figura 3.5: Diagrama de treinamento para E2CO com funções de perda de dados de fluxo e poço. Fonte: (COUTINHO, 2022).

O autor destaca, na Figura acima, a rede de saída de transição, e o termo de fluxo bifásico $\mathcal{L}_{flux2PH}$. Além disso, comenta que o E2CO pode ser utilizado com ou sem função de perda de fluxo, o que o torna uma técnica geral, ampliando suas aplicações para quaisquer sistemas compostos por *snapshots* de estado e controles.

Em seu trabalho, (COUTINHO, 2022) conclui frisando que o destaque do E2CO em detrimento das técnicas dispostas anteriormente, reside no fato do mesmo fazer uso do estado no espaço reduzido para gerar as matrizes $\mathbf{C}_r^t$ e $\mathbf{D}_r^t$, empregando ainda uma abordagem de Sistema de Controle para prever os dados de saída. Esta ponderação do autor corrobora com as informações descritas nesta dissertação, acerca do livro de (JANSEN, 2013), no sentido de que tais abordagens são similares à identificação do sistema no âmbito da comunidade de Sistemas de Controle. Sucintamente, o E2CO independe das variáveis de estado no espaço de origem para predizer dados de poço.

Para conhecer alguns avanços recentes, é válido inteirar-se sobre o trabalho de (DEOTA, 2023) que estendeu a ideia do E2CO para o caso de modelos de reservatórios tridimensionais, modificando-o para um sistema de controle baseado em *autoencoder* convolucional tridimensional, servindo como um *proxy* para simuladores numéricos tridimensionais, tornando o referido modelo robusto e confiável para reservatórios complexos.

Também vale a pena observar o trabalho de (SATHUJODA; SHETH, 2023) que propuseram um método inovador de Aprendizado Localizado, informado por física e ciente das condições de contorno, que estende os modelos E2C e E2CO para aprender representações locais de variáveis de estado globais num sistema de Reação-Advecção-Difusão. Ao treinar o modelo com dados de simulação de reservatórios, eles exibiram que este é capaz de prever estados futuros do sistema, para um determinado conjunto de controles, com grande precisão, fazendo uso de apenas uma fração dos dados disponíveis.

Na seção abaixo, descrevemos em detalhes como (COUTINHO, 2022) fez para implementar o E2CO, e comentaremos brevemente sobre a teoria por trás do estimador de James-Stein, o qual pretendemos fazer aplicações nos dados de poço com vistas a elevar a confiabilidade dos mesmos.

3.3 Implementação do E2CO

Discorreremos a respeito do treinamento e predição de dados, estruturação das redes neurais (*encoder*, *decoder*, rede de transição, rede de saída de transição), implementação das redes neurais (não-determinismo, normalização), aplicação das camadas convolucionais, geração de dados (simulação numérica, conjunto de dados), análise de erro relativo das predições (cálculo de erro relativo, aplicação do estimador de James-Stein nos dados de poço, sensibilidade ao tamanho de treinamento).

3.3.1 Treinamento e predição de dados

Vimos que as Figuras 3.1, 3.3, 3.4 e 3.5 denotam os diagramas que ilustram os procedimentos de treinamento do E2C, E2C com perda de fluxo, E2C com perda de fluxo bifásico e E2CO, respectivamente. Pelo o que já foi dito, temos ciência do papel relevante das funções de perda para os procedimentos de treinamento, e (COUTINHO, 2022) relembra que elas servem para fornecer dados relativos à dinâmica do sistema e assegurar a física presente na simulação do reservatório.

O autor detalha que ao concluir o treinamento, e encontrar os parâmetros da rede neural (pesos, *bias* e filtros) de modo otimizado, faz-se uso dos mesmos para realizar a predição de comportamento do sistema dinâmico.

Na Figura 3.6, temos o diagrama de predição para o E2C com função de perda de fluxo, e E2C acoplado com dados de poço e funções de perda de fluxo bifásicas.

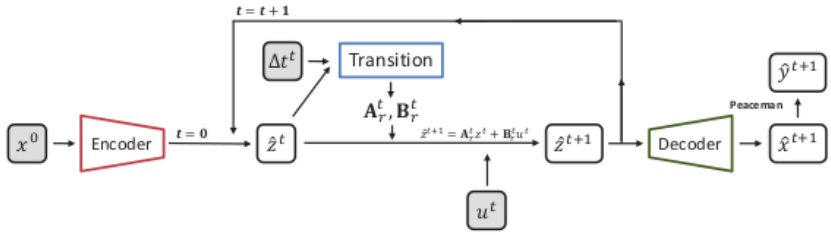

Figura 3.6: Diagrama de predição do E2C com função de perda de fluxo, e E2C acoplado com dados de poço e funções de perda de fluxo bifásicas. Fonte: (COUTINHO, 2022).

Na Figura acima, (COUTINHO, 2022) chama a atenção para o fato da evolução temporal está situada no espaço de dimensão reduzida (z), o que corrobora para um expressivo menor custo computacional na predição, o que é claramente vantajoso. Todavia, há uma grande desvantagem, pois o autor explica que nesta etapa, para prever os dados de poço, é preciso ter o estado no espaço original (x) e passar $\hat{z}^{t+1}$ pelo decodificador, para calcular $\hat{x}^{t+1}$, aplicando em seguida a Equação de Peaceman que fornece os dados de poço $\hat{y}^{t+1}$, e isso pode acarretar numa elevação do custo computacional. Cabe comentar que todo o processo descrito acima ocorre em cada passo de tempo.

Já o E2CO é mais vantajoso, pois os dados do poço são gerados no espaço latente via rede de saída de transição, ou seja, não há decodificação do espaço latente. Consideremos a Figura 3.7:

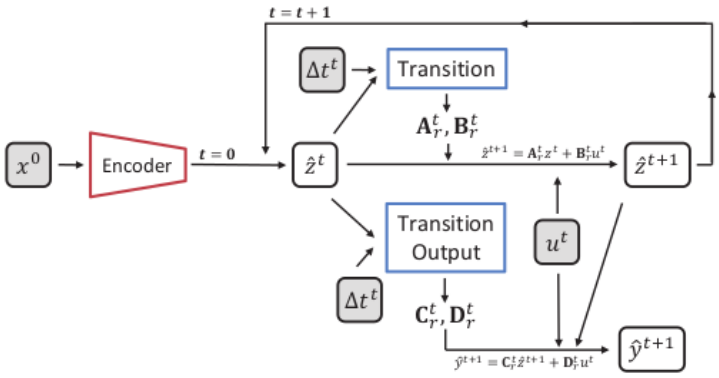

Figura 3.7: Diagrama de predição do E2CO. Fonte: (COUTINHO, 2022).

Nela, podemos ver que tanto a evolução de estado no tempo quanto a estimativa de dados de poço sucedem-se no espaço z (COUTINHO, 2022). No E2CO, o emprego do *decoder* é restrito aos casos em que é desejável predizer o estado no espaço original x.

A partir deste ponto, descrevemos em detalhes todos os aspectos computacionais necessários para a previsão dos dados de poço. Do ponto de vista de implementação dos códigos, esta é a parte mais importante da dissertação.

3.3.2
Estruturação das Redes Neurais

Através de inspiração no artigo de (JIN; LIU; DURLOFSKY, 2020), para parte de sua tese de doutorado, (COUTINHO, 2022) construiu uma estrutura de redes neurais significativamente símil à proposta pelos autores mencionados. Os principais itens a serem analisados seguem nas subsubseções abaixo, e no Apêndice B, pode-se consultar alguns conceitos teóricos relacionados aos mesmos.

3.3.2.1
Encoder

Em síntese, dizemos que o *encoder* e/ou codificador transforma as variáveis de estado (pressão e saturação) do espaço original x no espaço latente z. Isso é feito via construção de uma série de blocos codificados, seguidos por blocos convolucionais residuais e uma camada densa totalmente conectada. A Figura 3.8 ilustra tal processo.

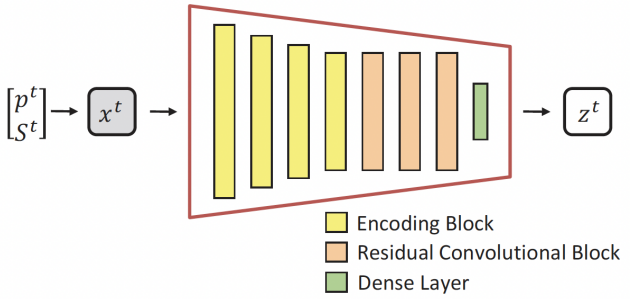

Figura 3.8: Estrutura do *Encoder*. Fonte: (COUTINHO, 2022).

Claramente, após essa transformação das variáveis de estado do espaço x para o espaço z, teremos que $dim\ l_x > dim\ l_z$. A partir deste ponto, vamos considerar que o modelo numérico estudado será dotado de $l_x = 60 \times 60 \times 2 = 7200$ estados e $l_z = 50$. Tais dimensões são referentes ao caso em que aplicaremos os métodos. Durante a aplicação ficará mais evidente os seus respectivos usos.

Analisando a composição bloco a bloco de codificação, temos que cada um possui uma camada convolucional *2D*, sucedida por um *batch normalization* (consultar Apêndice B.6) e função de ativação do tipo *ReLU* (Unidade linear retificada), conforme apontado na parte superior da Figura 3.9.

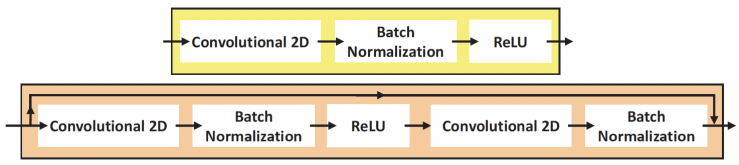

Figura 3.9: Estrutura do bloco de codificação e do bloco residual. Fonte: (COUTINHO, 2022).

Embora a estrutura de cada bloco de codificação seja igual, (COUTINHO, 2022) ressalta que a dimensão de cada bloco será alterada para viabilizar a redução de dimensionalidade da entrada do *decoder*, consoante a Figura 3.8. Ademais, as dimensões de cada componente empregado serão expostas subsequentemente no formato de Tabelas.

Na parte inferior da Figura 3.9, podemos ver que os blocos convolucionais residuais detém a mesma estrutura dos blocos de *encoder*, a menos do incremento de outra camada convolucional $2D$ acompanhada por *batch normalization*.

A seguir, iremos exibir como foi construído o decodificador.

3.3.2.2 *Decoder*

O papel do *decoder* e/ou decodificador é converter as variáveis de estado do espaço de dimensão reduzida (z) de volta ao espaço de dimensão original (x), ou seja, claramente temos um *encoder* reverso (COUTINHO, 2022). Sua estrutura é exibida na Figura 3.10.

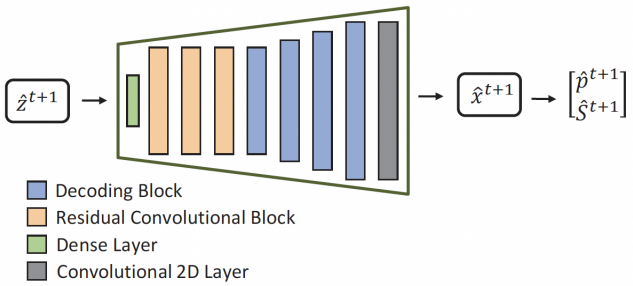

Figura 3.10: Estrutura do *Decoder*. Fonte: (COUTINHO, 2022).

Na Figura acima, nota-se que a composição do *decoder* é de basicamente uma camada densa completamente conectada, acompanhada por três blocos convolucionais residuais, quatro blocos de decodificação e uma camada convolucional *2D*.

O autor ressalta que os blocos convolucionais residuais empregados possuem a mesma estrutura que foi exibida para o *encoder*. O bloco de *decoder*

apresentado na Figura 3.11 possui uma camada convolucional *2D* transposta, *batch normalization* e função *ReLU*.

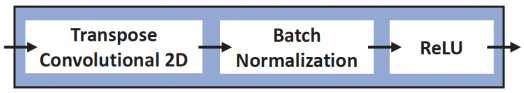

Figura 3.11: Bloco de *Decoder*. Fonte: (COUTINHO, 2022).

Em seguida, vamos discutir a respeito da importante rede de transição.

3.3.2.3 Rede de Transição

Em seu trabalho, (COUTINHO, 2022) comenta que a rede de transição (Figura 3.12) possui dois blocos de transformação, e duas camadas densas diretamente conectadas à saída do último bloco de transformação, respectivamente. Além disso, foi feito um remodelamento da saída destas camadas densas com fins de gerar as matrizes $\mathbf{A}_r^t$ e $\mathbf{B}_r^t$, que são a saída da rede, cujas ordens/dimensões são determinadas pela entrada/estado do sistema.

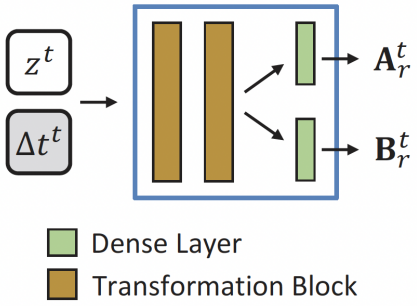

Figura 3.12: Rede de transição. Fonte: (COUTINHO, 2022).

O bloco de transformação (Figura 3.13) é dotado de uma camada densa, com *batch normalization*, acompanhada por uma ReLu, respectivamente.

Figura 3.13: Bloco de transformação. Fonte: (COUTINHO, 2022).

A seguir, iremos conhecer em detalhes a rede de saída de transição desenvolvida por (COUTINHO, 2022) e que permitiu de fato, o êxito do método E2CO como modelo *proxy*.

3.3.2.4 Rede de Saída de Transição

A rede de saída de transição (Figura 3.14) empregada no método E2CO apresenta uma estrutura significativamente semelhante à da rede de transição (COUTINHO, 2022). A diferença entre ambas as redes está relacionada à dimensão da camada densa nos blocos de transformação, sendo natural supor que $\dim \text{camadas}_{\text{rede\_transição}} > \dim \text{camadas}_{\text{rede\_saída\_transição}}$.

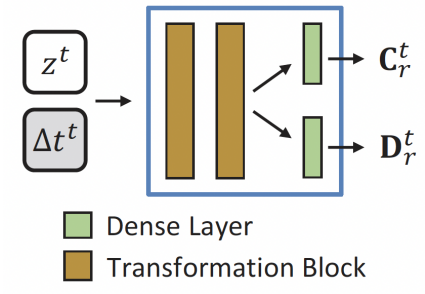

Figura 3.14: Rede de saída de transição. Fonte: (COUTINHO, 2022).

O autor frisa que as dimensões/ordens das matrizes $\mathbf{C}_r^t$ e $\mathbf{D}_r^t$ estão sujeitas às dimensões da entrada/saída do sistema, que, em geral, são menores em relação à entrada/estado apresentada na rede de transição.

Com isso, apresentaremos a seguir os detalhes computacionais da implementação das redes neurais para realizar as simulações. Em particular, explicitaremos os métodos empregados por (COUTINHO, 2022).

3.3.3 Implementação de Redes Neurais

Estamos implementando todos os métodos propostos utilizando (PYTHON, 2024) versão 3.9.19 lançada em 19 de março de 2024, juntamente com o TensorFlow (versão 2.13.0) com Keras API (versão 2.13.1). Em contrapartida, (COUTINHO, 2022) utilizou as versões 3.7.4 e 2.3.1 de Python e TensorFlow, respectivamente. Também adotamos as mesmas dimensões de camadas, número e tamanho de filtros, e tamanho de saída de cada bloco de *encoder* e *decoder* empregadas por (COUTINHO, 2022), e tais dados estão disponíveis para consulta nas Tabelas 3.4 e 3.5.

Layer	*# Filter*	*Filter size*	*Stride*	*Padding*	*Output size*
Input					$(N_x, N_y, 2)$
Encoding Block	16	3×3	2×2	*Same*	$(N_x/2, N_y/2, 16)$
Encoding Block	32	3×3	1×1	*Same*	$(N_x/2, N_y/2, 32)$
Encoding Block	64	3×3	2×2	*Same*	$(N_x/4, N_y/4, 64)$
Encoding Block	128	3×3	1×1	*Same*	$(N_x/4, N_y/4, 128)$
ResConv Block	128	3×3	1×1	*Same*	$(N_x/4, N_y/4, 128)$
ResConv Block	128	3×3	1×1	*Same*	$(N_x/4, N_y/4, 128)$
ResConv Block	128	3×3	1×1	*Same*	$(N_x/4, N_y/4, 128)$
Dense					$(l_z, 1)$

Tabela 3.4: Arquitetura do *Encoder*. Fonte: (COUTINHO, 2022).

Layer	*# Filters*	*Filter size*	*Stride*	*Padding*	*Output size*
Input					$(l_z, 1)$
Dense					$(N_x/4 \times N_y/4 \times 128, 1)$
ResConv Block	128	3×3	1×1	*Same*	$(N_x/4, N_y/4, 128)$
ResConv Block	128	3×3	1×1	*Same*	$(N_x/4, N_y/4, 128)$
ResConv Block	128	3×3	1×1	*Same*	$(N_x/4, N_y/4, 128)$
Decoding Block	128	3×3	1×1	*Same*	$(N_x/4, N_y/4, 128)$
Decoding Block	64	3×3	2×2	*Same*	$(N_x/2, N_y/2, 64)$
Decoding Block	32	3×3	1×1	*Same*	$(N_x/2, N_y/2, 32)$
Decoding Block	16	3×3	2×2	*Same*	$(N_x, N_y, 16)$
Conv2D	2	3×3	1×1	*Same*	$(N_x, N_y, 2)$

Tabela 3.5: Arquitetura do *Decoder*. Fonte: (COUTINHO, 2022).

Adotamos a mesma dimensão para o bloco de transformação utilizada por (COUTINHO, 2022) na rede de Transição, considerando $n_{Trans} = 200$. Além disso, incluímos as últimas camadas densas antes dos cálculos das matrizes, com dimensões l_2^2 para $\mathbf{A}_r^t$ e $l_z(n_i + n_p)$ para $\mathbf{B}_r^t$, em que n_i e n_p correspondem ao número de poços injetores e produtores, respectivamente. Assim como feito pelo autor, a saída das camadas densas foi remodelada para corresponder a cada dimensão da matriz. Considerando a importante Rede de Saída de Transição, seguimos também o autor aplicando $n_{trans\_WD} = 20$ para a dimensão do bloco de transformação, enquanto as últimas camadas densas possuem dimensões $l_z(n_i + 2n_p)$ para $\mathbf{C}_r^t$ e $(n_i + n_p)(n_i + 2n_p)$ para $\mathbf{D}_r^t$.

Assim como descrito por (COUTINHO, 2022), utilizamos o algoritmo de otimização Adam [1] na etapa de treinamento, com uma taxa de aprendizagem de 1×10^{-4}. Vale comentar que (COUTINHO, 2022) empregou um *batch size* de 4, realizou a seleção da amostra de forma aleatória para a composição do *batch* de treinamento, e durante a etapa de treinamento fez uso de 100 épocas.

Os códigos foram executados em uma CPU (*Central Processing Unit*) Intel Xeon E5 de 6 núcleos com 3,5 GHz, acompanhada de 16 GB de

[1] Segundo (KINGMA; BA, 2017), este é um algoritmo para otimização de funções objetivo estocásticas baseado em gradientes de primeira ordem, que faz uso de estimativas adaptativas de momentos de ordem inferior. Tal método é bem adequado para problemas grandes em termos de dados e/ou parâmetros.

memória RAM. O tempo necessário para o treinamento de 100 épocas foi de aproximadamente 35 minutos. Para fins de comparação, também foram realizados alguns testes em uma CPU Intel Core i5 de 4 núcleos com 3,2 GHz e 28 GB de memória RAM DDR3 de 1867 MHz. Nesse ambiente, o treinamento das mesmas 100 épocas foi concluído em cerca de 28 minutos.

3.3.3.1 Não-determinismo

Pode-se definir não-determinismo como uma propriedade de processos em que fornecer as mesmas entradas pode produzir saídas diferentes (COOPER; FRANKLE; SA, 2022). Em sua pesquisa, (COUTINHO, 2022) comenta que algumas das fontes de não-determinismo são, por exemplo, a inicialização dos pesos das camadas e *kernels*, bem como a disposição dos modelos a serem incluídos no *batch* de treinamento. Como o treinamento foi realizado exclusivamente em CPUs, ficamos livres de outras fontes de não-determinismo associadas ao uso de GPUs, diferentemente de (COUTINHO, 2022) que executou os treinamentos em GPUs, ou seja, seus treinamentos estiveram sujeitos a esses fatores de não-determinismo.

Em seu trabalho, (COUTINHO, 2022) também destaca que (RIACH, 2020) discute tais fontes de não-determinismo e apresenta uma solução para a maior parte delas. Desse modo, (COUTINHO, 2022) utilizou tal artifício para garantir a reprodução dos resultados de treinamento, mantendo consistência para os mesmos dados de entrada e parâmetros, de forma que foi evitado o fenômeno em que redes distintas são geradas ao repetir em duplicata o mesmo procedimento de treinamento, na mesma máquina, com os mesmos dados de entrada e parâmetros. Nesta dissertação, a única exceção para ocorrência deste tipo de empecilho foi detectada em uma simulação específica, que será apresentada no capítulo 4. Em outras palavras, as previsões e estimativas de erro geradas foram consistentes entre as repetições.

Na próxima subsubseção discorremos sobre o processo de normalização dos dados, bem como sobre sua importância.

3.3.3.2 Normalização

No trabalho de (JAVAHERI; SEPEHRI; TEIMOURPOUR, 2014), os autores explicam que a normalização dos dados consiste em ajustar os valores dos atributos para que fiquem numericamente no mesmo intervalo ou escala, assegurando a mesma importância relativa entre eles. Os autores destacam a existência de três técnicas de normalização: Normalização Z-*score*, Normaliza-

ção Min-Max (utilizada nesta dissertação) e Normalização por Escalonamento Decimal.

Neste estudo, para lidar com os dados de entrada (advindos do estado), dados de poço e de cálculos das funções de perda, foi adotado um processo de normalização, conforme proposto por (COUTINHO, 2022). Seguindo a metodologia descrita pelo autor, para cada quantidade são definidos os limites de normalização (máximo e mínimo), e esses valores são utilizados tanto no treinamento quanto na predição.

O processo de normalização consiste em calcular o quociente entre a diferença da quantidade a ser normalizada ($\mathcal{Q}$) e o valor mínimo de referência ($\mathcal{Q}_{\min}$), dividido pela diferença entre o valor máximo ($\mathcal{Q}_{\max}$) e o valor mínimo. A formulação matemática desse procedimento é apresentada em (COUTINHO, 2022) como

$$\mathcal{Q}_{\text{NORM}} = \frac{\mathcal{Q} - \mathcal{Q}_{\min}}{\mathcal{Q}_{\max} - \mathcal{Q}_{\min}}. \tag{3-18}$$

Em certo sentido, podemos pensar também que esse processo de normalização se assemelha à ideia de normalizar vetores que permite escalar um vetor para que ele se torne um vetor unitário.

Os valores máximo e mínimo para cada quantidade/grandeza foram selecionados via estado de entrada (quantidades de pressão e saturação), dados de poço (quantidades de BHP e vazão de óleo e água), e dos controles de poço (quantidades de vazão de injeção de água e BHP).

Como já mencionado, foi realizada a normalização de valores para calcular as funções de perda. Em particular, aplicando as equações

$$(\mathcal{L}_{rec})_i = \left\{ \left\| x^t - \hat{x}^t \right\|_2^2 \right\}_i$$

e

$$(\mathcal{L}_{pred})_i = \left\{ \left\| x^{t+1} - \hat{x}^{t+1} \right\|_2^2 \right\}_i,$$

estamos fazendo uso de valores normalizados (pressões e saturações do estado). Com o mesmo valendo para

$$(\mathcal{L}_{well\_data2})_i = \left\{ \left\| y^{t+1} - \hat{y}^{t+1} \right\|_2^2 \right\}_i,$$

em que empregamos valores normalizados para os dados de poço.

Em sua texto, (COUTINHO, 2022) pondera que introduziu a adoção de uma função de perda de fluxo normalizada. Primeiramente, calculando os fluxos máximo e mínimo via estado de entrada e os empregando para

normalizar a função de perda de fluxo. Isso se encaixa nos casos das funções de perda monofásica

$$\begin{aligned}\left(\mathcal{L}_{flux}\right)_i &= \left(\mathcal{L}_{flux,rec}\right)_i + \left(\mathcal{L}_{flux,pred}\right)_i \\ &= k\left\{\left\|\nabla p^t - \nabla \hat{p}^t\right\|_2^2 + \left\|\nabla p^{t+1} - \nabla \hat{p}^{t+1}\right\|_2^2\right\}_i,\end{aligned}$$

e bifásica

$$\begin{aligned}\left(\mathcal{L}_{flux\text{2Ph}}\right)_i =& \left(\mathcal{L}_{\text{flux2Ph, }rec}\right)_i + \left(\mathcal{L}_{flux\text{2Ph, }pred}\right)_i \\ =& k\Big\{\left\|k_{r,o}\left(S_w^t\right)\nabla p^t - k_{r,o}\left(\hat{S}_w^t\right)\nabla \hat{p}^t\right\|_2^2 + \\ &+ \left\|k_{r,w}\left(S_w^t\right)\nabla p^t - k_{r,w}\left(\hat{S}_w^t\right)\nabla \hat{p}^t\right\|_2^2 + \\ &+ \left\|k_{r,o}\left(S_w^{t+1}\right)\nabla p^{t+1} - k_{r,o}\left(\hat{S}_w^{t+1}\right)\nabla \hat{p}^{t+1}\right\|_2^2 + \\ &+ \left\|k_{r,w}\left(S_w^{t+1}\right)\nabla p^{t+1} - k_{r,w}\left(\hat{S}_w^{t+1}\right)\nabla \hat{p}^{t+1}\right\|_2^2\Big\}_i.\end{aligned}$$

3.3.4 Aplicação de camadas convolucionais parciais

Um dos principais entraves do E2CO é a exigência de ter todos os blocos de malha ativos no modelo de simulação (COUTINHO, 2022). Em sua pesquisa, (COUTINHO, 2022) também explica que a operação matemática associada a uma camada convolucional bidimensional é dada por

$$x' = \mathbf{W}^T\mathbf{X} + b, \tag{3-19}$$

em que $\mathbf{X}$ é o vetor de entrada 2D, $\mathbf{W}$ é um *kernel* 2D, b é o termo de *bias*, e x' é o vetor de saída 2D. É realizada a repetição dessa operação com o *kernel* se movendo sobre o vetor de entrada. Em seu trabalho, (COUTINHO, 2022) não considerou a presença de valores inativos no vetor de entrada.

O autor pontua que a maior parte dos modelos de reservatório é dotado de blocos inativos, visando a garantia do formato geológico do reservatório, e cita sobre o trabalho de (LIU et al., 2018), no qual é comentado que o emprego de camada convolucional parcial é uma alternativa ao uso de camada convolucional tradicional. Matematicamente, descrevemos a operação de camada convolucional parcial, como

$$x' = \begin{cases} \mathbf{W}^T(\mathbf{X}\odot\mathbf{M})\frac{\text{soma}(\mathbf{1})}{\text{soma}(\mathbf{M})} + b & \text{, se soma}(\mathbf{M}) > 0 \\ 0 & \text{, caso contrário} \end{cases}, \tag{3-20}$$

em que $\mathbf{M}$ é um vetor de máscara, de mesma dimensão do vetor de entrada, e $\odot$ denota a multiplicação elemento a elemento. O vetor de máscara é valorado por 1 nas células ativas, e 0, caso contrário.

3.3.5 Geração de dados

Naturalmente, com base no trabalho de (COUTINHO, 2022), vamos discutir nesta subseção o modelo de simulação empregado para gerar dados de estado (pressão e saturação) e de poço. Em seguida, abordaremos os detalhes de treinamento, validação e geração das amostras de teste.

3.3.5.1 Simulação Numérica

Os dados utilizados neste trabalho foram extraídos a partir da tese de (COUTINHO, 2022), na qual foi empregado o simulador comercial IMEX (CMG, 2024) para a geração de dados necessários às fases de treinamento, validação e teste dos experimentos com o E2CO. Assim, nesta dissertação, fazemos uso direto dos resultados produzidos por (COUTINHO, 2022).

Conforme mencionado por (COUTINHO, 2022), poderia ter sido feito o uso de qualquer simulador de reservatório comercial disponibilizado no mercado para o treinamento do modelo. Visando replicar as condições descritas pelo autor, adotamos uma malha cartesiana composta por $60 \times 60 \times 1$ blocos de malha e um modelo de fluido bifásico (óleo-água).

A Figura 3.15 apresenta o campo de permeabilidade fixo gerado, enquanto a Figura 3.16 exibe as curvas de permeabilidade relativa empregadas na simulação do E2CO.

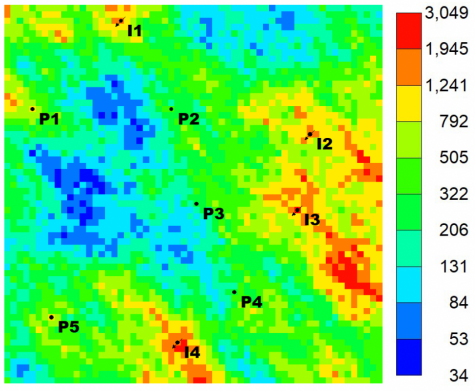

Figura 3.15: Mapa de permeabilidade. Fonte: (COUTINHO, 2022).

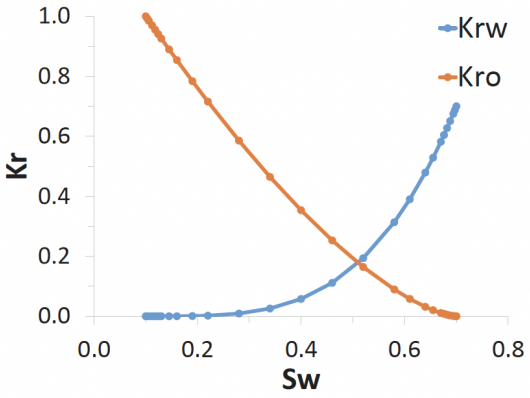

Figura 3.16: Curvas de permeabilidade relativa. Fonte: (COUTINHO, 2022).

Reproduzimos também a quantidade de poços usadas por (COUTINHO, 2022), isto é, simulando predições para cinco poços produtores e quatro poços injetores. O controle dos poços produtores é feito via pressão de fundo do poço (BHP), e o dos injetores pela taxa de injeção de água. Foram feitas predições para 2000 dias, sendo feitas alterações de controles a cada 100 dias, para realmente reproduzir as condições declaradas por (COUTINHO, 2022). A Tabela 3.6 dispõe de algumas propriedades aplicadas no modelo de simulação.

Propriedade	**Valor**	**Unidade**
Dimensão do bloco de grade	$50 \times 50 \times 10$	m
Compressibilidade da rocha	1×10^{-6}	$(\text{kgf/cm}^2)^{-1}$
Fator de volume de formação de óleo	1.0	res-m^3/std-m^3
Fator de volume de formação de água	1.0	res-m^3/std-m^3
Viscosidade do óleo	0.91	cP
Viscosidade da água	0.31	cP
Densidade do óleo	800	kg/cm^3
Densidade da água	1000	kg/cm^3
Porosidade	0.2	adimensional
Pressão inicial	325	kgf/cm^2
Saturação inicial de água	0.1	adimensional
Saturação de água conata (S_{wc})	0.1	adimensional
Saturação residual de óleo (S_{or})	0.3	adimensional
$k_{rw}(1 - S_{or})$	0.7	adimensional
$k_{ro}(S_{wc})$	1.0	adimensional

Tabela 3.6: Propriedades do modelo de reservatório. Fonte: Adaptado de (COUTINHO, 2022).

Vamos discutir na subsubseção a seguir o conjunto de dados.

3.3.5.2 Conjunto de dados

Em seu texto, (COUTINHO, 2022) explica que a construção de um modelo *proxy* rápido insere-se no contexto em que se objetiva integrá-lo a *frameworks* de otimização de controle, nos quais múltiplas *calls* do modelo substituto são empregadas para determinar um conjunto de controles que otimize uma função objetivo. No bojo desta discussão, tem-se a expectativa de que o *proxy* seja capaz de prever dados de poços sob distintos conjuntos de controle, os quais em geral são desconhecidos antes do término da otimização.

Para simular o comportamento de uma configuração de entrada variável, reproduzindo as condições adotadas por (COUTINHO, 2022), realizaremos 300 simulações para treinar o modelo proposto. Cada simulação será executada ao longo de 20 períodos de tempo, correspondendo a predições para 2000 dias, com ajustes nos controles a cada 100 dias.

Como estamos de fato replicando a forma como (COUTINHO, 2022) fez para gerar o conjunto de dados para realizar a etapa de treinamento, a pressão de fundo nos poços produtores foi amostrada numa distribuição uniforme $\mathcal{U}(260, 275)$ kgf/cm^2. Para cada conjunto de controle i, a taxa de injeção base foi amostrada como $q_{\text{inj\_base},i} \sim \mathcal{U}(300, 950)$ m^3/dia. Foi adicionada a amostragem de uma perturbação $q^t_{\text{inj\_pert},i} \sim \mathcal{U}(-80, 90)$ m^3/dia para cada período de tempo t, de modo que podemos escrever a taxa de fluxo de injeção por

$$q_{\text{inj}}(i, t) = q_{\text{inj\_base},i} + q^t_{\text{inj\_pert},i}. \tag{3-21}$$

As medidas declaradas no parágrafo acima, foram assim implementadas por (COUTINHO, 2022) visando garantir a variabilidade do volume de água injetado em todos os conjuntos de controle dos injetores. Na Figura 3.17, podemos ver um exemplo dos controles dos poços injetores, que nos confirma a variabilidade da taxa de injeção.

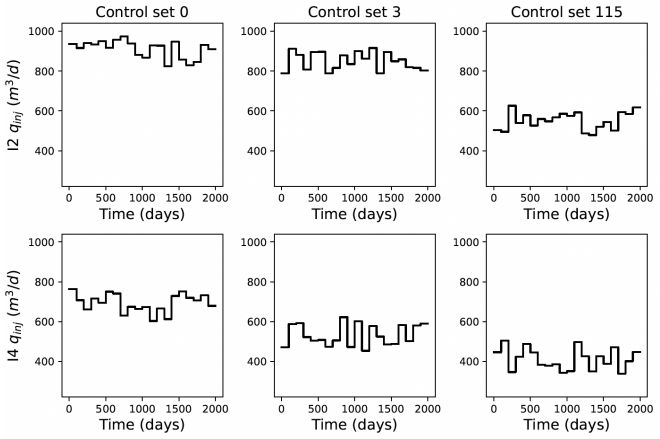

Figura 3.17: Exemplos de controle de vazão de água de injeção para os poços I2 (primeira linha) e I4 (segunda linha), em que cada coluna representa um conjunto de controle. Fonte: (COUTINHO, 2022).

Já a variabilidade nos controles dos produtores pode ser visualizada na Figura 3.18.

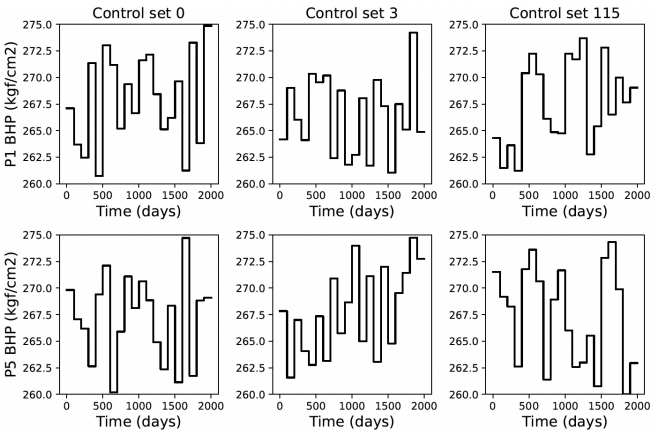

Figura 3.18: Exemplos de controle de BHP para poços produtores P1 (primeira linha) e P5 (segunda linha), em que cada coluna representa um conjunto de controle. Fonte: (COUTINHO, 2022).

Seguindo a metodologia de (COUTINHO, 2022), em cada um dos 300 conjuntos de controle gerados, executamos o simulador de reservatório e registramos/armazenamos o estado (pressão e saturação de água) para cada um dos 20 períodos de tempo. Desse modo, o conjunto de treinamento é dotado de 6000 amostras, cada uma contendo os dados descritos na Tabela 3.7.

Variável	**Descrição**
x^t	estado no instante t
x^{t+1}	estado no instante $t+1$
u^t	controles no instante t
Δt^t	duração do intervalo de tempo t
y^{t+1}	dados do poço no instante $t+1$

Tabela 3.7: Descrição das variáveis do conjunto de treinamento. Fonte: Adaptado de (COUTINHO, 2022).

No cenário descrito, $t = 0, 1, 2, 3, \ldots, 19$, em que $t = 0$ corresponde ao estado inicial, que deve ser armazenado como o primeiro *snapshot* do estado (x_0). Os conjuntos de validação e teste são compostos por 50 conjuntos de controle cada, totalizando 1000 amostras. A geração desses conjuntos segue um processo similar em relação ao conjunto de treinamento, todavia com o emprego de uma semente distinta para o gerador de números aleatórios. Essa abordagem assegura que os dados utilizados para treinamento, validação e teste sejam executados em entradas de controle distintas, o que é considerado uma boa prática em *Machine Learning.*

3.4
Análise de erro relativo das predições

Descreveremos o procedimento de cálculo do erro relativo das predições e apresentaremos brevemente a teoria do estimador de James-Stein, que pretendemos aplicar aos dados de poço, tanto aos fornecidos pela HFS (*high-fidelity solution*) quanto aos preditos pelo E2CO.

3.4.1
Calculando o erro relativo

Apresentaremos as definições de erro relativo empregadas na avaliação da acurácia do E2CO. Naturalmente, seguimos o que foi feito por (COUTINHO, 2022), que, por sua vez, utilizou grande parte de definições similares às de (JIN; LIU; DURLOFSKY, 2020).

Na sua tese de doutorado, (COUTINHO, 2022) definiu que o erro relativo de vazão de água (e_w^p) e óleo (e_o^p) para um único poço produtor (p), é dado de forma geral por:

$$e_j^p = \frac{\int_0^T \left| \hat{q}^{j,p}(t) - q_{\text{HFS}}^{j,p}(t) \right| \, dt}{\int_0^T \left| q_{\text{HFS}}^{j,p}(t) \right| \, dt}, \tag{3-22}$$

em que a variável com chapéu denota a vazão prevista, e j indica a fase em que a predição está associada, podendo ser água (w) ou óleo (o). Notemos que, na equação (3-22), o numerador representa a diferença entre as áreas previstas da fase j e as fornecidas pela HFS, enquanto o denominador corresponde apenas à área da simulação de alta fidelidade. Ou seja, essa equação fornece, de fato, uma métrica que indica o quão distante estamos da solução ideal disponibilizada pelo IMEX para um poço produtor.

O erro para todos os poços produtores é dado pela média sobre a equação (3-22), de tal sorte que

$$E_j = \frac{1}{n_p} \sum_{p=1}^{n_p} e_j^p, \text{ com } j = o, w. \tag{3-23}$$

Podemos estender a ideia apresentada na equação (3-22) para calcular o erro relativo da pressão de fundo de poço em um poço injetor (i). Ou seja, teremos que:

$$e_{\text{BHP}}^i = \frac{\int_0^T \left| \hat{p}^i(t) - p_{\text{HFS}}^i(t) \right| dt}{\int_0^T \left| p_{\text{HFS}}^i(t) \right| dt}. \tag{3-24}$$

O erro para todos os poços injetores é também dado pela média, isto é,

$$E_{\text{BHP}} = \frac{1}{n_i} \sum_{i=1}^{n_i} e^i_{\text{BHP}}, \tag{3-25}$$

com n_i indicando a quantidade de poços injetores.

Podemos definir o erro relativo para as vazões cumulativas como:

$$e^p_{\text{cum},j} = \frac{\left|\hat{Q}^{j,p} - Q^{j,p}_{\text{HFS}}\right|}{\left|Q^{j,p}_{\text{HFS}}\right|}, \text{ com } j = o, w. \tag{3-26}$$

Em que, Q denota a taxa cumulativa. Para todos os produtores, definimos o erro cumulativo como:

$$E_{\text{cum},j} = \frac{1}{n_p} \sum_{p=1}^{n_p} e^p_{\text{cum,j}}. \tag{3-27}$$

Em seu trabalho, (COUTINHO, 2022) também define o erro relativo com respeito à pressão e saturação de água, por:

$$E_v = \frac{\sum_{k=1}^{n_b} \int_0^T \left|\widehat{v}^k(t) - v^k_{\text{HFS}}(t)\right| dt}{\sum_{k=1}^{n_b} \int_0^T \left|v^k_{\text{HFS}}(t)\right| dt}, \tag{3-28}$$

em que v^k indica a variável de estado no bloco da malha k (pressão (p^k) ou saturação (S^k)), e n_b a quantidade de blocos da malha na simulação numérica do reservatório.

Na próxima subseção, abordaremos a análise dos dados de poço utilizando o estimador de James-Stein, amplamente reconhecido na Estatística. Veremos que este estimador possui potencial para aprimorar a avaliação da confiabilidade dos dados de poço.

3.4.2
Aplicação do estimador de James-Stein nos dados de poço

Conforme dito por (GAJO, 2016), estimadores de encolhimento tornaram-se uma das ferramentas básicas na análise de dados de dimensão alta. Um dos estimadores de encolhimento mais conhecidos, é o de James-Stein (JAMES; STEIN, 1961). Objetivamos aplicá-lo no contexto desta dissertação. Como estamos fazendo uma aplicação na área de Engenharia de Reservatórios, este é um ambiente ideal para tentativa de uso deste tipo de estimador. Em particular, queremos aplicar o estimador de James-Stein (JS) tanto nos dados fornecidos pela HFS, quanto nos dados previstos pelo E2CO. Ao decorrer do texto ficará mais evidente a motivação por trás da referida intenção.

De acordo com o trabalho de (HEUMANN, 2011), James e Stein demonstraram que o estimador de mínimos quadrados ordinários (OLS) (que é

equivalente ao estimador de máxima verossimilhança (ML) no caso normal) é superado/dominado pelo estimador de JS, se a dimensão do vetor de parâmetros for maior que dois. Os autores reforçam que o estimador de JS é um estimador não linear, e enviesado (valor esperado do estimador não coincide com o parâmetro), com a superioridade sobre os mínimos quadrados ordinários sendo definida em termos do critério de erro quadrático médio escalar (MSE).

Consideremos uma amostra $X \sim \mathcal{N}_p(\theta, \sigma^2 I_p)$, de tamanho $n = 1$, isto é, $X = \{X_1, \ldots, X_p\}$ denota uma variável aleatória, com distribuição normal $p-$variada com média $\theta = \{\theta_1, \ldots, \theta_p\}$ e matriz de covariância $\sigma^2 I_p$ de ordem $p \times p$. Segundo (FOURDRINIER; STRAWDERMAN; WELLS, 2018) a classe de estimadores de James-Stein é dada por

$$\delta_a^{\mathrm{JS}}(X) = \left(1 - \frac{a\sigma^2}{||X||^2}\right) X. \tag{3-29}$$

Observação 1 (Tipo de variância considerada em James-Stein) *Em seu trabalho, (HEUMANN, 2011) comenta que a variância conhecida σ^2 utilizada no estimador de JS é do tipo homocedástica, ou seja, caracterizada por uma dispersão constante dos dados em todas as partes da distribuição.*

No livro de (FOURDRINIER; STRAWDERMAN; WELLS, 2018), os autores comentam que uma das propriedades básicas de $\delta_a^{\mathrm{JS}}(X)$ é fornecida pelo seguinte Teorema:

Teorema 3.1 *Sob o modelo acima, é válido que*

1. *O risco de $\delta_a^{JS}(X)$ é dado por*

$$\delta_a^{JS}(X) = p\sigma^2 + \sigma^4(a^2 - 2a(p-2))E_\theta\left[\frac{1}{||x||^2}\right] \tag{3-30}$$

   para $p \geq 3$.

2. *$\delta_a^{JS}(X)$ domina $\delta_0(X) = X$ para $0 < a < 2(p-2)$ e é minimax para $0 \leq a \leq 2(p-2)$ para todo $p \geq 3$.*

3. *A escolha uniformemente ótima de a é, $a = p - 2$ para $p \geq 3$.*

4. *O risco em $\theta = 0$ para o estimador James-Stein ótimo $\delta_{p-2}^{JS}(X)$ é $2\sigma^2$ para todo $p \geq 3$.*

Prova. Como o principal caso de interesse é o item 3, focaremos em prová-lo. A prova dos demais itens, também encontra-se em (FOURDRINIER; STRAWDERMAN; WELLS, 2018).

Notemos que para todo θ, o risco de $R(\theta, \delta_a^{\mathrm{JS}})$ é minimizado tomando $a = p - 2$, já que este valor minimiza $a^2 - 2a(p-2)$. ■

De posse disso, podemos definir o estimador de James-Stein como sendo

$$\delta_{p-2}^{\mathrm{JS}}(X) := \left(1 - \sigma^2 \frac{p-2}{||\mathbf{X}||^2}\right) \mathbf{X}, \qquad p \geq 3 \tag{3-31}$$

que é o estimador uniformemente melhor na classe dos estimadores de JS (FOURDRINIER; STRAWDERMAN; WELLS, 2018). Notemos que, para $\theta = 0$, o risco é $2\sigma^2$, para todo $p \geq 3$, o que significa que economias significativas em termos de risco são possíveis numa vizinhança de $\theta = 0$ para grandes p. Além disso, os autores explicam que os estimadores de JS possuem a propriedade de que, quando $||X||^2 < a\sigma^2$, o multiplicador de X torna-se negativo.

A partir desse ponto, iremos expor um caso interessante: a situação em que um valor maior de X para uma coordenada específica pode resultar numa estimativa menor para a média dessa coordenada. Para lidar com esse problema, (FOURDRINIER; STRAWDERMAN; WELLS, 2018) sugerem uma possível solução: modificar o estimador de JS para sua parte positiva, ou seja:

$$\delta_a^{JS+}(X) = \left(\underbrace{1 - \frac{a\sigma^2}{\|X\|^2}}_{=\,t}\right)_+ X. \tag{3-32}$$

em que $(t)_+ = \max(t, 0)$. Um exemplo particular de um estimador do tipo Baranchik (equação 3-33),

$$\delta_{a,r}^{B}(X) = \left(1 - \frac{a\sigma^2 r(||X||^2)}{||X||^2}\right) X, \tag{3-33}$$

é a estimativa da parte positiva. Em que, tipicamente, $r(\cdot)$ é contínua e não decrescente. O formato de $r(||X||^2)$ pode ser consultado em (FOURDRINIER; STRAWDERMAN; WELLS, 2018). Seguindo o material dos autores, tem-se que sob certas condições, os estimadores do tipo Baranchik melhoram sobre X. Mais ainda, que o estimador de parte positiva de JS melhora o estimador de JS em si.

Sob a forma do Teorema 2.3 apresentado em (FOURDRINIER; STRAWDERMAN; WELLS, 2018), os autores exibem outras condições sob as quais um estimador do tipo Baranchik melhora sobre X. Tal Teorema também nos indica que o estimador de parte positiva de JS domina X para $0 < a \leq 2(p-2)$. Como já sabemos que o estimador de parte positiva de JS até melhora o estimador de JS em si, temos que isso é responsável por refletir o fenômeno geral de que um estimador de parte positiva tende a dominar sua contraparte, desde que a densidade subjacente seja simétrica e unimodal. Nesse contexto, há um resultado geral (Teorema 3.2) que formaliza essa observação.

Teorema 3.2 *Suponha que X tenha uma densidade $f(x-\theta)$ em $\mathbb{R}^p$ tal que a função f seja simétrica e unimodal em cada coordenada separadamente para cada valor fixo das outras coordenadas. Então, para qualquer estimador de risco finito de θ da forma*

$$\delta(X) = \left(1 - B(X_1^2, X_2^2, \ldots, X_p^2)\right) X, \tag{3-34}$$

o estimador da parte positiva

$$\delta_+(X) = \left(1 - B(X_1^2, X_2^2, \ldots, X_p^2)\right)_+ X \tag{3-35}$$

domina $\delta(X)$ sob qualquer perda da forma

$$L(\theta, \delta) = \sum_{i=1}^{p} a_i(\delta_i - \theta_i)^2, \quad (a_i > 0 \text{ para todo } i), \tag{3-36}$$

desde que

$$P_\theta[B(X_1^2, X_2^2, \ldots, X_p^2) > 1] > 0. \tag{3-37}$$

Prova. Consultar (FOURDRINIER; STRAWDERMAN; WELLS, 2018). ■

No contexto de nossa aplicação, na equação (3-31), o vetor X possui entradas correspondentes aos dados previstos de BHP, vazão de óleo e água, ou seja, $\dim X = 3$. Neste estudo, pretendemos aplicar o estimador de JS tanto nos *arrays* que contêm dados de HFS quanto nos dados previstos pelo E2CO. Após a aplicação do estimador, esperamos que ele promova o encolhimento das distribuições dos dados de HFS e E2CO, resultando em estimativas mais confiáveis, devido à natureza do estimador (já que, como dito por (HEUMANN, 2011), James e Stein mostraram, que o estimador domina o OLS (ou ML) com relação ao critério MSE, o que significa que sempre apresenta um menor MSE do que o OLS, independentemente do verdadeiro $\boldsymbol{\theta}$, se $p > 2$, isto é, o OLS é inadmissível nesse caso). É importante mencionar que a caracterização do perfil das distribuições dos dados de poço para verificação das hipóteses dos Teoremas 3.1 e 3.2, será analisado através de histogramas.

Baseado nessa teoria, no capítulo seguinte discutiremos a viabilidade de empregar tal estimador com foco no que foi discutido no parágrafo acima.

No que se refere ao desempenho do E2CO e sua relação com o volume de dados, (COUTINHO, 2022) explica que o tamanho original do conjunto de treinamento era de 6000 amostras, mas ele também treinou o modelo com 4000 e 2000 amostras (equivalendo, respectivamente, a 300, 200 e 100 conjuntos de controle). Além disso, o mesmo notou que à medida que o número de amostras de treinamento diminui, o erro de estimativa no conjunto de teste tende a aumentar. Vale ressaltar que, para a taxa de óleo e BHP, o método E2CO

apresenta erro relativo menor quando o tamanho do conjunto de treinamento é compactado.

4
Sensibilidade das Previsões aos Hiperparâmetros

Este capítulo tem como objetivo analisar a sensibilidade na escolha dos hiperparâmetros das redes neurais e apresentar a previsão de dados de poço (pressão de fundo de poço, vazão de água e óleo) utilizando o método E2CO. Os resultados serão comparados com simulações realizadas no simulador comercial IMEX (CMG, 2024), obtidos de (COUTINHO, 2022), avaliando a acurácia das previsões e confrontando-a com os resultados do autor. Além disso, busca-se entender como a escolha dos hiperparâmetros impacta diretamente na qualidade e a confiabilidade das previsões.

Também discutiremos a respeito da viabilidade do uso do estimador de James-Stein (JS), com fins de obter novas predições com base nos dados de poço disponíveis via *High Fidelity Simulation* (HFS) e E2CO.

Nas seções seguintes, apresentaremos a metodologia empregada para estudar a sensibilidade dos hiperparâmetros do modelo E2CO, bem como os resultados observados nesses experimentos e sua relação com as estimativas de dados de poço. Por fim, será exibida a discussão sobre a viabilidade do uso do estimador de JS.

4.1
Metodologia

Para o modelo E2CO conseguir obter boas previsões de dados de poço mencionados acima, é necessário fornecer às redes neurais informações de hiperparâmetros, ou seja, os valores que estamos atribuindo, como os pesos durante o procedimento de treinamento. Conforme destacado por (COUTINHO, 2022), a escolha do conjunto ideal de hiperparâmetros é desafiadora. O autor não estabeleceu um procedimento bem definido para superar essa dificuldade. Também enfatiza que a seleção dos melhores hiperparâmetros exige necessariamente um treinamento completo, seguido da predição das saídas utilizando o conjunto de dados de teste. Esse processo permite a análise do erro associado à escolha dos hiperparâmetros. A estratégia sugerida pelo autor consiste em repetir esse procedimento com diferentes conjuntos de hiperparâmetros, identificando aquele que apresenta o menor erro de previsão. Visando atingir o mesmo objetivo, podemos observar na Figura 4.1 um fluxograma que resume as principais estratégias pensadas nesta dissertação para estudar a sensibilidade dos hiperparâmetros do E2CO.

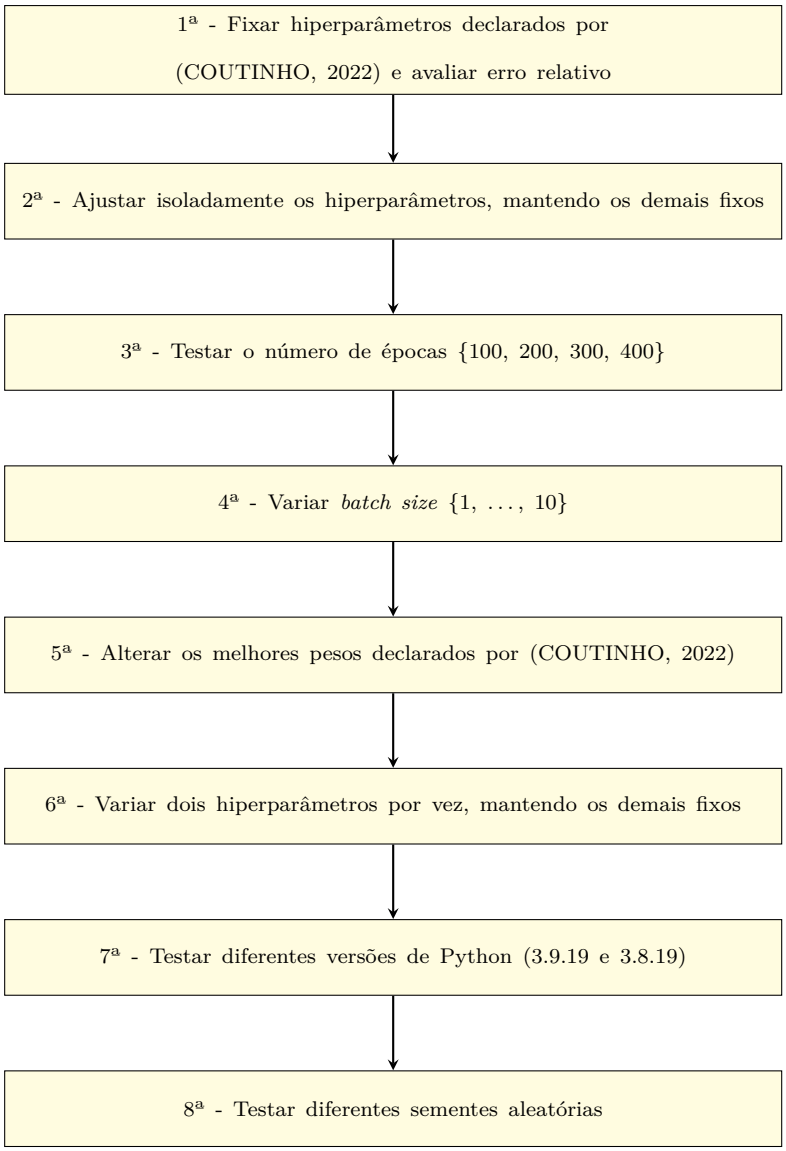

Figura 4.1: Fluxograma com as principais estratégias adotadas para estudar sensibilidade de hiperparâmetros do E2CO.

Com o objetivo de detalhar de forma mais precisa os procedimentos empregados em cada uma das principais estratégias adotadas, apresentamos a lista abaixo:

- **1ª Estratégia:** Observamos os melhores hiperparâmetros declarados por (COUTINHO, 2022) em sua tese, para fixá-los na fase de treinamento inicial. Ao executarmos o treinamento e predição, realizamos a verificação (via gráfico *boxplot*) do erro relativo das predições;
- **2ª Estratégia:** Verificando a existência de um alto erro relativo, procuramos identificar a influência de cada hiperparâmetro por meio de ajustes isolados, assim como feito por (COUTINHO, 2022). Ou seja, mantendo um conjunto de pesos fixos e variando apenas um hiperparâmetro (peso)

por vez, para avaliar sua influência isolada na predição. O valor do hiperparâmetro isolado que promover o menor erro relativo (menor mediana de distribuição) foi considerado como fixo, e passamos a variar outro hiperparâmetro de escolha, mantendo os demais fixos. Repetimos a mesma análise de erro até percorrer todo o conjunto de hiperparâmetros. A faixa de intervalo analisada para identificar o valor do hiperparâmetro que nos forneceu a menor mediana de erro relativo das predições foi de [0, 4] com incrementos de 0.1 de intervalo em cada tentativa;

- **3ª Estratégia:** Testamos o número de épocas para 100, 200, 300 e 400, visando verificar a influência nas predições. A quantidade de épocas que promoveu o menor erro relativo foi fixada em todos os demais procedimentos de treinamento;
- **4ª Estratégia:** Variamos o valor do *batch size*, realizando tentativas no intervalo de [1, 10] com incrementos de 1, até encontrar o *batch size* que promoveu o menor erro relativo. Esse valor foi fixado em todos os demais procedimentos de treinamento;
- **5ª Estratégia:** Alteramos os melhores pesos declarados por (COUTINHO, 2022), isto é, verificamos o que aconteceria nas predições se mudássemos os pesos γ_{flux}, $\gamma_{transition}$, γ_{inj_pres}, γ_{inj_sat}, γ_{prod_pres} e γ_{prod_sat};
- **6ª Estratégia:** Verificamos se a variação de dois hiperparâmetros (pesos) simultaneamente (mantendo os demais fixos) implicaria na redução do erro relativo das predições. Tal estratégia foi tentada, já que supomos que possa ser que haja algum padrão que relacione tais pesos, e isso de algum modo poderia ajudar no êxito das predições. Também procuramos identificar as predições que gerassem menor mediana de erro relativo. Por exemplo, considerando γ_{inj_sat} e γ_{prod_sat}, fizemos variações no intervalo [0, 0.1] em ambos os pesos, com incrementos de 0.01 em cada tentativa, mantendo os demais fixos, e escolhemos o melhor, seguindo o critério de escolha já conhecido. Depois foram feitos mais testes, seguindo essa mesma lógica, mas expandindo o intervalo de busca para [0.2, 1] com incrementos de 0.1;
- **7ª Estratégia:** Fizemos testes em duas versões de Python (3.9.19 e 3.8.19), mas mantendo a versão do TensorFlow em 2.13.0, para verificar qual delas levaria à promoção do menor erro relativo;
- **8ª Estratégia:** Fizemos os treinamentos declarados acima, variando as *seeds* que geram os valores aleatórios, até encontrarmos uma semente que promova menor erro relativo. Esta semente foi fixada em todos os demais procedimentos de treinamento.

O conjunto de hiperparâmetros investigados e suas respectivas utilidades podem ser observados na Tabela 4.1.

Hiperparâmetro	**Função no treinamento**	**Valor**
weight_trans_reg	consultar os Algoritmos 3 e 4	0
weight_trans_reg1	consultar os Algoritmos 3 e 4	0
weight_reg_ABCD	consultar na página 77	1
use_flux_loss	consultar os Algoritmos 3 e 4	1 ph
weight_flux	consultar os Algoritmos 3 e 4	0.1
weight_transition	consultar o Algoritmo 3	1
weight_transition_wd	consultar os Algoritmos 3 e 4	1
weight_inj_gridblock_pres	consultar os Algoritmos 3 e 4	0
weight_inj_gridblock_sat	consultar os Algoritmos 3 e 4	0
weight_prod_gridblock_pres	consultar os Algoritmos 3 e 4	0.01
weight_prod_gridblock_sat	consultar os Algoritmos 3 e 4	0
weight_water_rate	consultar no Algoritmo 3	1
weight_prod_sat_qw	consultar os Algoritmos 3 e 4	0
weight_qw_neg	consultar os Algoritmos 3 e 4	0
n_epochs	consultar Apêndice B.7	100
batch_size	consultar Apêndice B.5	4
SEED	gerar números aleatórios	124

Tabela 4.1: Lista de hiperparâmetros investigados, com suas respectivas funções no treinamento, e valores declarados por (COUTINHO, 2022).

Para observar como estes hiperparâmetros são inseridos na implementação dos códigos, pode-se consultar os Algoritmos 1, 2, 3 e 4 escritos na forma de pseudo-código, em que a maior parte do conjunto de hiperparâmetros da Tabela 4.1 é diretamente mencionada, assim como sua formulação matemática associada, principalmente nos Algoritmos 3 e 4 que são relativos às funções de perda.

Como era de se esperar, o ajuste mais desafiador está relacionado aos pesos aplicados à função de perda total. Assim como em (COUTINHO, 2022), o procedimento de treinamento utilizado depende da combinação das funções de perda, e determinar a ponderação ideal entre elas para construir a função de perda total é um desafio complexo. A função de perda total tem a seguinte formulação matemática:

$$\begin{aligned}\mathcal{L}_{\text{total}} &= (\mathcal{L}_{rec})_i + (\mathcal{L}_{pred})_i + (\mathcal{L}_{trans})_i + \gamma_{flux}(\mathcal{L}_{flux})_i + \gamma_{well\_data3}(\mathcal{L}_{well\_data3})_i \\ &= \left\{\left\|x^t - \hat{x}^t\right\|_2^2\right\}_i + \left\{\left\|x^{t+1} - \hat{x}^{t+1}\right\|_2^2\right\}_i + \left\{\left\|z^{t+1} - \hat{z}^{t+1}\right\|_2^2\right\}_i \\ &\quad + \gamma_{flux}\left(k\left[\left\{\left\|\nabla p^t - \nabla \hat{p}^t\right\|_2^2\right\}_i + \left\{\left\|\nabla p^{t+1} - \nabla \hat{p}^{t+1}\right\|_2^2\right\}_i\right]\right) \\ &\quad + \gamma_{well\_data3}\left(\left\{\left\|y^{t+1} - \hat{y}^{t+1}\right\|_2^2\right\}_i\right),\end{aligned} \tag{4-1}$$

em que i denota o índice da amostra.

De forma semelhante à vista no trabalho de (COUTINHO, 2022), também utilizamos normalização de entradas, saídas e componentes de funções de perda para evitar o ajuste de pesos da função de perda, e testamos variações das funções de perda total, ou seja, verificando o que acontecia nos casos monofásico, bifásico e sem perda de fluxo. Por fim, assim como (COUTINHO, 2022), tentamos fornecer mais peso às variáveis de estado relativas à pressão e saturação nos blocos da malha dos poços (esta análise cabe aos blocos da malha de poços injetores e produtores), e também foi testado diversas combinações de pesos distintas para uma função de perda responsável por gerir a análise de incompatibilidade entre os valores previstos e verdadeiros, das variáveis de estado do bloco da malha de poços. Tais funções de perda e seus respectivos pesos associados estão dispostos na Tabela 4.2.

Função de perda	**Pesos**
$\mathcal{L}_{p\_pres}$	γ_{p_pres}
$\mathcal{L}_{p\_sat}$	γ_{p_sat}
$\mathcal{L}_{i\_pres}$	γ_{i_pres}
$\mathcal{L}_{i\_sat}$	γ_{i_sat}

Tabela 4.2: Tabela com funções de perda e seus respectivos pesos associados. Fonte: Adaptado de (COUTINHO, 2022).

Para escolher o melhor conjunto de hiperpâmetros, verificou-se o erro total de predição de cada conjunto de teste, e assim como feito por (COUTINHO, 2022) foi utilizado a soma da mediana dos erros definidos através das equações (3-23), (3-25), (3-27) e (3-28), que definem o que chamamos de erro total de predição.

Dado os procedimentos metodológicos descritos nesta seção, na próxima iremos discorrer a respeito da análise da sensibilidade aos hiperparâmetros.

4.2
Análise da Sensibilidade aos Hiperparâmetros

Discutiremos o impacto de cada estratégia apresentada no fluxograma da Figura 4.1 na qualidade das previsões dos dados de poço. Nas próximas subseções, analisaremos o histórico das tentativas realizadas e as motivações por trás de cada abordagem.

4.2.1
1ª Estratégia

Os hiperparâmetros declarados na tese de (COUTINHO, 2022) estão expostos na Tabela 4.1. Primeiramente, os mesmos foram inseridos no código para prosseguir com a etapa de treinamento, utilizando as versões 3.8.19 de Python e 2.13.0 de TensorFlow. A Figura 4.2 exibe os resultados obtidos, sendo que em preto são ilustrados os resultados do simulador IMEX, e em azul os resultados do E2CO. Na primeira coluna temos as predições para a pressão de fundo de poço (BHP), na segunda coluna as predições para a vazão de óleo, e por fim na terceira coluna as predições para a vazão de água. Observamos pelos gráficos que o modelo E2CO prevê os dados de poço (BHP, vazão de óleo e água) com perfil constante, ou seja, não reproduz dados similares aos obtidos pelo IMEX via *High Fidelity Simulation* (HFS).

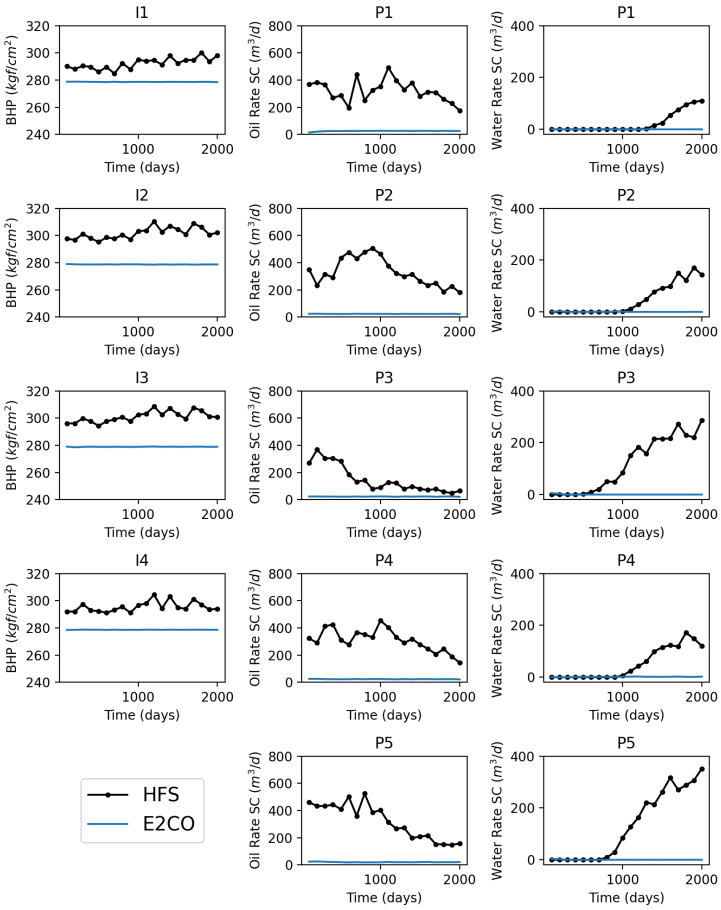

Figura 4.2: Estimativas de dados de poço usando os hiperparâmetros declarados por (COUTINHO, 2022).

Acreditamos que o comportamento de má predição de dados de poço exposto da Figura 4.2 seja atribuído às diferentes versões de Python e do TensorFlow, que possuem algoritmos diferentes e/ou aprimorados promovendo distintas predições.

Nos experimentos realizados se constatou uma grande sensibilidade dos resultados às variações dos hiperparâmetros. O exemplo da Figura 4.3 abaixo ilustra uma alteração no peso *weight_trans_reg* de 2.8 (direita) para 2.801 (esquerda).

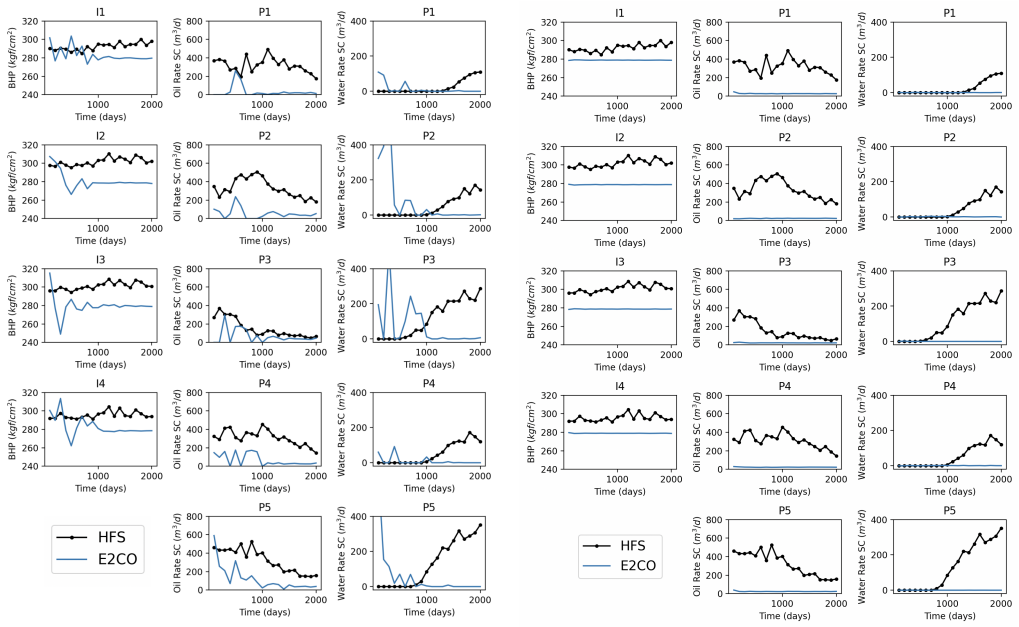

Figura 4.3: Exemplos de estimativas de dados de poço usando os hiperparâmetros declarados por (COUTINHO, 2022).

A Tabela 4.3 exibe alguns dos parâmetros adicionados por (COUTINHO, 2022) para treinar o E2CO. Optamos por mantê-los fixos para os procedimentos de treinamento empregados nas próximas estratégias, pois testar alterações em todos eles seria computacionalmente muito dispendioso.

Parâmetros	**Valores**
$\dim l_z$	50
$\dim n_{Trans}$	200
$\dim n_{trans\_WD}$	20
adam_learning_rate	1×10^{-4}
n.º *Transf_Block_WD*	2
n.º *Residual_Block*	3
activation_layer	ReLU
n.º *training_samples*	300
n.º *validation_samples*	50

Tabela 4.3: Parâmetros declarados por (COUTINHO, 2022).

De posse desta situação, procuramos estabelecer estratégias objetivando estudar a sensibilidade aos hiperparâmetros do E2CO.

4.2.2 2ª Estratégia

Essa estratégia foi pensada da seguinte forma: considerando que no trabalho de (COUTINHO, 2022) os melhores resultados foram obtidos para uma função de perda monofásica, fizemos uso da mesma, e inicialmente

variamos isoladamente o primeiro peso γ_{trans_reg} num intervalo de $[0, 4]$ com incrementos de 0.1, procurando o valor que resultasse no menor erro relativo, enquanto os demais pesos detalhados na Tabela 4.4 declarados na tese de doutorado de (COUTINHO, 2022) permaneceram fixos.

Descrição do Peso	**Valor**
γ_{trans_reg1}	0
γ_{reg_ABCD}	1
$\gamma_{transition}$	1
$\gamma_{transition\_wd}$	1
γ_{flux}	0.1
γ_{inj_pres}	0
γ_{inj_sat}	0
γ_{prod_pres}	0.01
γ_{prod_sat}	0
γ_{water_rate}	1
$\gamma_{prod\_sat\_qw}$	0
γ_{qw_neg}	0

Tabela 4.4: Tabela de pesos com valores iniciais adotados na estratégia na 2ª estratégia.

Para o peso γ_{trans_reg} o melhor valor encontrado foi 2.8 (Figura 4.4), portanto o fixamos para a próxima rodada de testes, e passamos a variar o peso γ_{trans_reg1} no intervalo [0,4] com uma escolha de particionamento similar à declarada para o peso anterior, enquanto os demais pesos permaneciam fixos. Em seguida, encontramos o melhor valor correspondente para γ_{trans_reg1}, o fixamos para próxima rodada de testes, e partimos para o peso γ_{reg_ABCD}. Com o restante do processo seguindo passos análogos até chegarmos em γ_{qw_neg}.

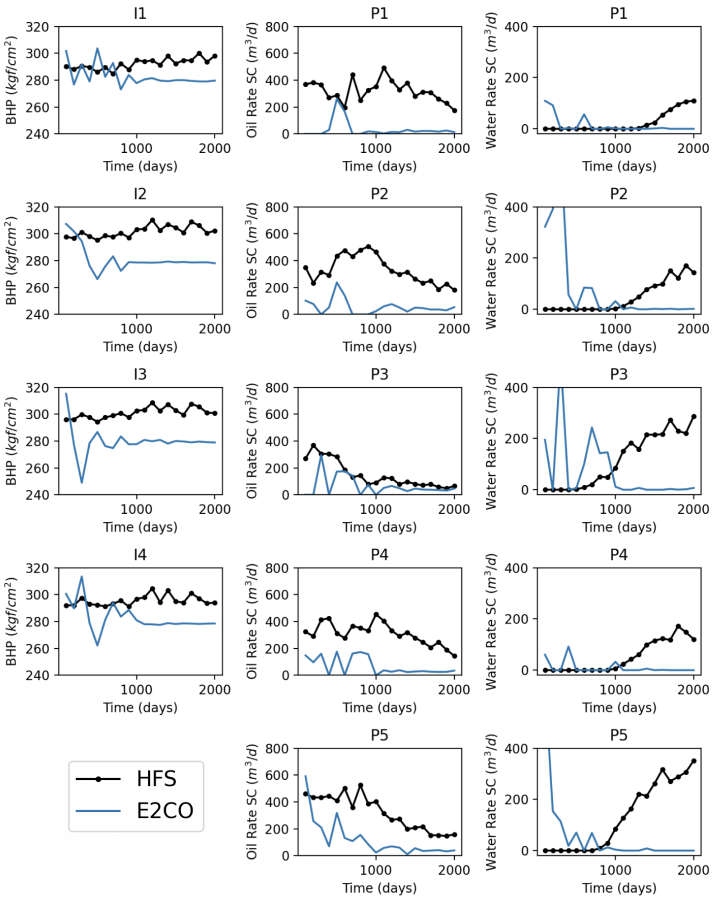

Figura 4.4: Exemplos de predição usando *weight_trans_reg* = 2.8.

Durante o processo, cabe destacar que ao encontrarmos o valor de 0.3 para o peso γ_{reg_ABCD}, observamos uma sensível melhora nas predições para o BHP e a vazão de óleo. Acontece que *weight_reg_ABCD* é importante, pois em seu algoritmo de treinamento (COUTINHO, 2022), fez com que, se *weight_reg_ABCD* > 0, então o código executa o bloco *else*, o que significa que uma variável *activity_regularizer* será definida como *keras.regularizers.L1(weight_reg_ABCD)*. Ou seja, a regularização L_1 será aplicada com um fator igual a *weight_reg_ABCD*. Como visto em (DE; DOOSTAN, 2022), para evitar *overfitting* e melhorar o erro de generalização, a regularização baseada na norma l_1 dos parâmetros, por exemplo, é aplicada.

Na Figura 4.5, podemos visualizar que as predições tornaram-se menos inconsistentes quando comparadas às primeiras observações. Vale frisar que não vamos expor o erro relativo das predições por enquanto, pois claramente este valor ainda é bastante expressivo.

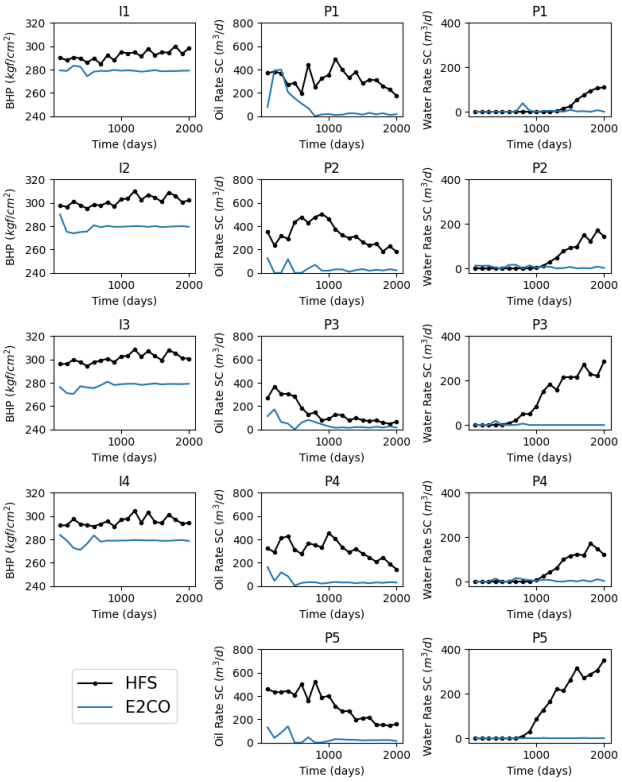

Figura 4.5: Comportamento de predição ao inserir $\gamma_{reg\_ABCD} = 0.3$ no treinamento do E2CO.

A ideia da segunda estratégia mostrou-se promissora, principalmente após o ajuste do peso $\gamma_{transition\_wd}$ para 2.1 (Figura 4.6), pois começamos a ter boas previsões para a pressão de fundo de poço, em que obtivemos uma mediana de erro relativo inferior a 10 % (Figura 4.7).

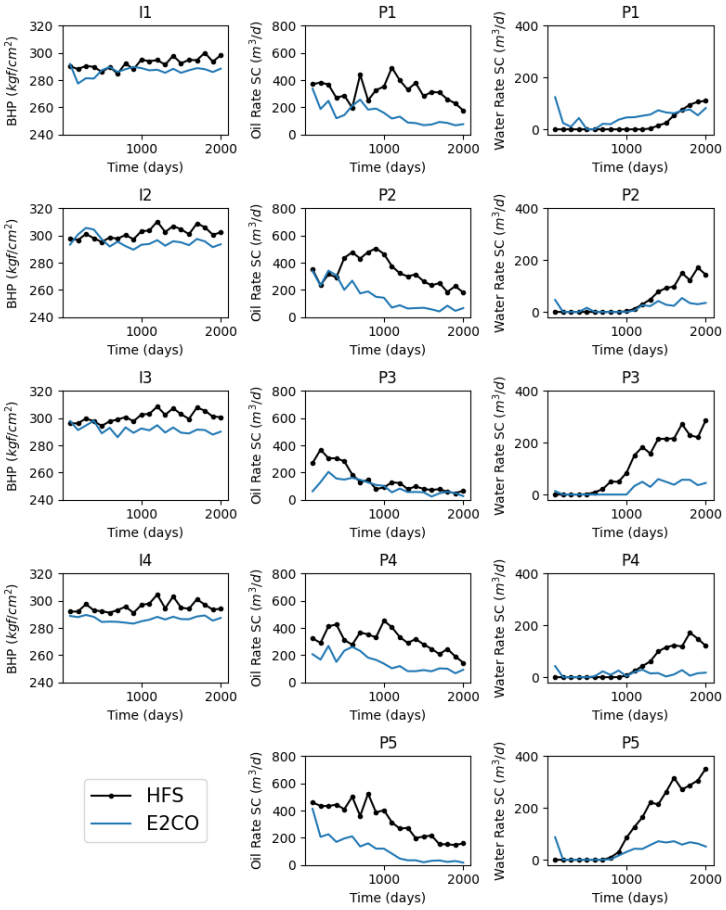

Figura 4.6: Comportamento de predição ao inserir $\gamma_{transition\_wd} = 2.1$ no treinamento do E2CO.

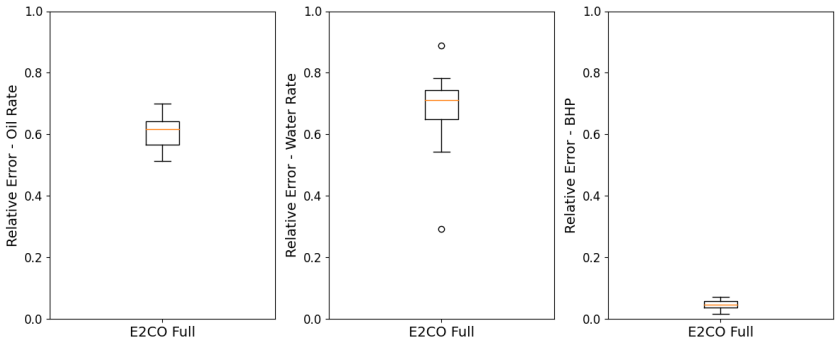

Figura 4.7: Erro relativo de predição de dados de poço ao inserir $\gamma_{transition\_wd} = 2.1$ no treinamento do E2CO.

As predições obtidas na Figura 4.6 são provenientes da Tabela 4.5.

Descrição do Peso	**Valor**
γ_{trans_reg}	2.8
γ_{trans_reg1}	0
γ_{reg_ABCD}	0.3
$\gamma_{transition}$	1
$\gamma_{transition\_wd}$	2.1
γ_{flux}	0.1
γ_{inj_pres}	0
γ_{inj_sat}	0
γ_{prod_pres}	0.01
γ_{prod_sat}	0
γ_{water_rate}	1
$\gamma_{prod\_sat\_qw}$	0
γ_{qw_neg}	0

Tabela 4.5: Tabela de pesos com valores adotados para obter a Figura 4.6.

De fato, ao observarmos os Algoritmos 3 e 4, vemos que este peso $\gamma_{transition\_wd}$ é mencionado entre as linhas 19 e 22 do Algoritmo 3, e nas linhas 12 e 13 do Algoritmo 4.

Posteriormente, ao fazer testes isolados nos pesos γ_{water_rate}, $\gamma_{prod\_sat\_qw}$ e γ_{qw_neg}, respectivamente, mantendo a estratégia já estipulada, na busca no intervalo [0,4], obtivemos dois melhores resultados para cada peso, e estes são: 0.2 e 1 para γ_{water_rate}, 0 e 0.2 para $\gamma_{prod\_sat\_qw}$, e 0 e 0.1 para γ_{qw_neg}. Ao verificar o erro relativo, consideramos que nossas melhores previsões até este momento estão dispostas na Tabela 4.5, que estão atreladas à Figura 4.6, com exceção do peso γ_{water_rate} cujo melhor valor encontrado foi 0.2.

Em seguida, decidimos verificar o que aconteceria se variássemos isoladamente o valor do *batch size* no conjunto $\{1, 2, 3, 5, 6, 7, 8, 9\}$ mantendo os demais hiperparâmetros fixos. Percebemos que não obtivemos bons resultados (por exemplo, Figura 4.8), e permanecemos adotando o *batch size* de 4, como havia sido implementado por (COUTINHO, 2022).

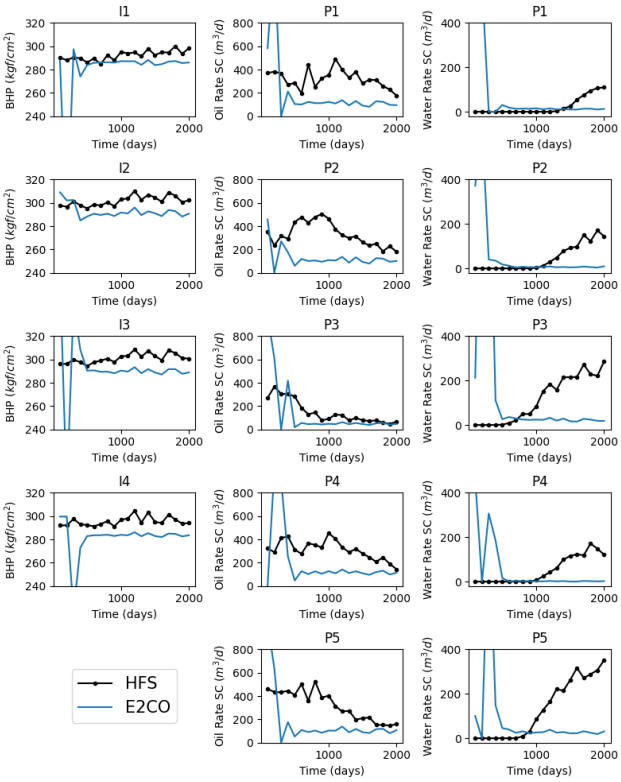

Figura 4.8: Exemplo de tentativa de uso de *batch size* = 6 no treinamento do E2CO.

Ao notarmos que o peso $\gamma_{transition\_wd}$ poderia ser muito relevante devido à sua posição nos Algoritmos já mencionados, procuramos fazer outros testes, mas tomando $\gamma_{transition\_wd} = 3.5$ que produziu o segundo melhor erro relativo. Pelo mesmo motivo, escolhemos $\gamma_{water\_rate} = 1$.

Até este ponto, estamos fazendo uso dos dados da Tabela 4.6.

Descrição do Peso	**Valor**
γ_{trans_reg}	2.8
γ_{trans_reg1}	0
γ_{reg_ABCD}	0.3
$\gamma_{transition}$	1
$\gamma_{transition\_wd}$	3.5
γ_{flux}	0.1
γ_{inj_pres}	0
γ_{inj_sat}	0
γ_{prod_pres}	0.01
γ_{prod_sat}	0
γ_{water_rate}	1
$\gamma_{prod\_sat\_qw}$	0
γ_{qw_neg}	0

Tabela 4.6: Tabela de pesos parcial.

4.2.3
3ª Estratégia

Pelas razões já apontadas acima, fizemos uso de $\gamma_{transition\_wd} = 3.5$, e decidimos variar isoladamente o número de épocas no conjunto $\{200, 300, 400\}$ com fins de verificar se havia alguma alteração nas predições. Percebemos que todas as predições pioraram drasticamente (por exemplo, Figura 4.9), logo para as próximas etapas permaneceremos empregando 100 épocas. Além de péssimas previsões, obtivemos um aumento significativo no tempo de treinamento e no custo computacional.

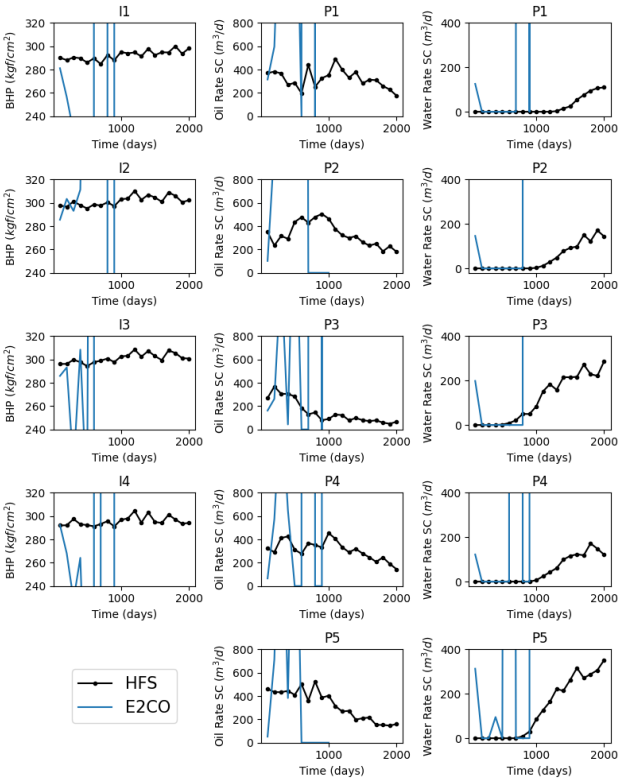

Figura 4.9: Comportamento de predição para treinamento com 200 épocas.

Não encontramos uma justificativa para tal ocorrência, mas (COUTINHO, 2022) experimentou problemas semelhantes, em que ele comenta que observou que a rede que produziu o menor erro de previsão dos dados de poço não é obtida após a época 100, como se esperava. Para se aprofundar sobre o quesito épocas, é válido a consulta ao Apêndice B.7.

4.2.4
4ª Estratégia

De posse dessa situação exposta na estratégia anterior, decidimos tentar alterar novamente o valor do *batch size*, mas escolhendo o conjunto de teste

$\{1, 2, 3, 5, 6, 7, 8, 9, 10\}$, ou seja, estamos empregando a Tabela 4.6 e alterando somente o *batch size*. Obtivemos uma melhoria nos resultados para *batch size* de 6 (Figura 4.10), devido ao seu relativo erro dos dados (Figura 4.11).

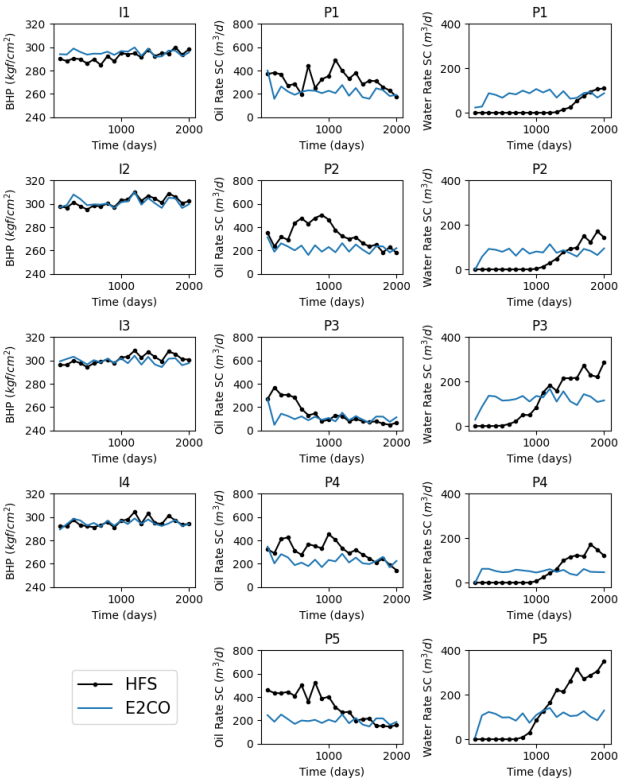

Figura 4.10: Comportamento de predição para treinamento com *batch size* $= 6$ e $\gamma_{transition\_wd} = 3.5$.

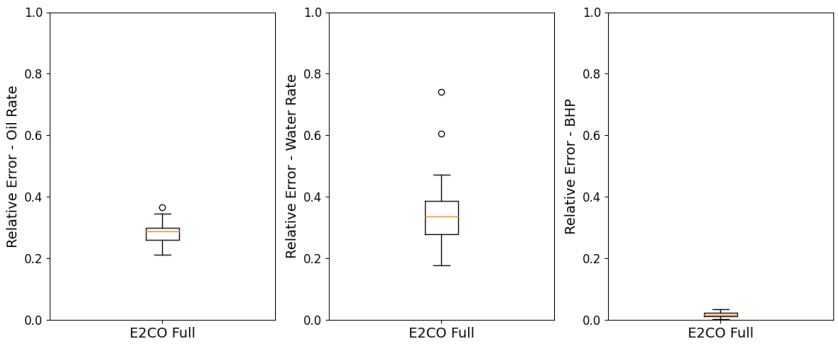

Figura 4.11: Erro de predição referente à Figura 4.10.

Visualmente, podemos perceber que a vazão de água possui com certeza um erro relativo de predição muito maior do que o apontado pela Figura 4.11, posto que identifica-se um valor inferior a 40 % para a mediana de distribuição. Isso acontece, por conta do cálculo do erro relativo ser feito com base na área sobre a predição do E2CO, e o modelo resulta numa área que compensa a

vazão nula de água nos primeiros 600 dias de predição, mascarando a acurácia para este dado de poço.

Para conhecer um pouco mais sobre a temática *batch size*, é interessante a consulta ao Apêndice B.5, mas previamente podemos dizer que em geral o tamanho de lote ideal será menor que 32, como dito por (GÉRON, 2019), e que o mesmo pode impactar significativamente no desempenho do modelo e no tempo de treinamento.

4.2.5
5ª Estratégia

Verificando que não obtivemos um progresso minimamente aceitável para a predição dos dados de poço, decidimos, adotar os segundos melhores valores previstos para $\gamma_{prod\_sat\_qw}$, e γ_{qw_neg}, ou seja, tomando os valores 0.2 e 0.1, respectivamente. Lembremos que tais valores já foram mencionados no parágrafo posterior à Tabela 4.5. Os resultados destas novas predições estão ilustrados nas Figuras 4.12 e 4.13.

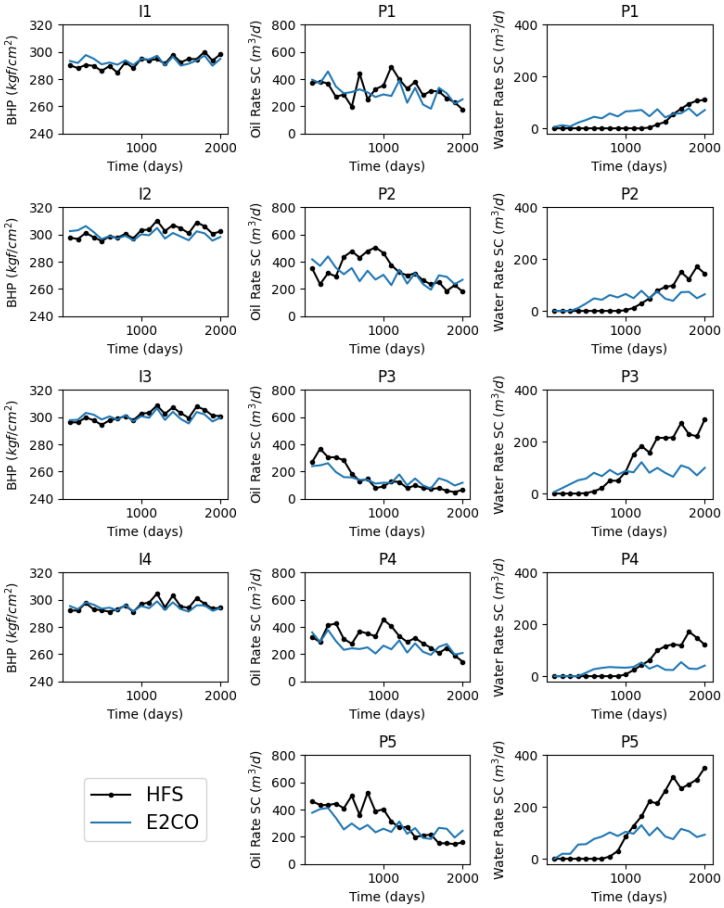

Figura 4.12: Predição dos dados de treinamento tomando $\gamma_{prod\_sat\_qw} = 0.2$ e $\gamma_{qw\_neg} = 0.1$.

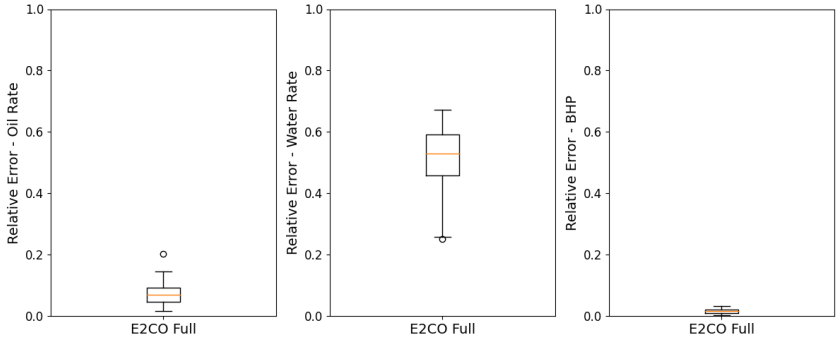

Figura 4.13: Erro de predição referente à Figura 4.12.

Como o erro de vazão de água foi muito elevado, pensamos em alterar os melhores pesos declarados por (COUTINHO, 2022) na Tabela 4.7, com fins de observar o que aconteceria se os mudássemos. Vale frisar que os melhores valores de pesos que já foram submetidos aos processos de otimização das Estratégias anteriores também permaneceram fixos no treinamento.

Peso	**Valor**
γ_{flux}	0.1
$\gamma_{transition}$	1
γ_{inj_pres}	0
γ_{inj_sat}	0
γ_{prod_pres}	0.01
γ_{prod_sat}	0

Tabela 4.7: Pesos declarados por (COUTINHO, 2022).

Seguimos novamente a mesma lógica estabelecida na subseção 4.2.2 (2ª Estratégia) de fazer uso de uma função de perda de fluxo monofásica, e de variar somente um dos pesos, mantendo os demais fixos, observando os erros relativos associados, encontrando o menor deles, e fixando-o para a próxima variação de peso. Por exemplo, encontrando o menor erro relativo para o peso γ_{flux}, o fixamos e partimos para variar somente o peso $\gamma_{transition}$, enquanto os demais pesos da Tabela 4.7 mantiveram-se fixos. Também fizemos esse processo avaliando as predições individuais de cada peso num intervalo de [0,4] com incremento de 0.1 a cada nova simulação. Com base na otimização individual de cada um dos pesos, obtivemos por fim, a Tabela 4.8.

Peso	**Valor**
γ_{flux}	0.1
$\gamma_{transition}$	0.4
γ_{inj_pres}	1.3
γ_{inj_sat}	0
γ_{prod_pres}	0.7
γ_{prod_sat}	0

Tabela 4.8: Novos melhores pesos substitutos para a Tabela 4.7.

A Figura 4.14 expõe as predições relativas à Tabela 4.8.

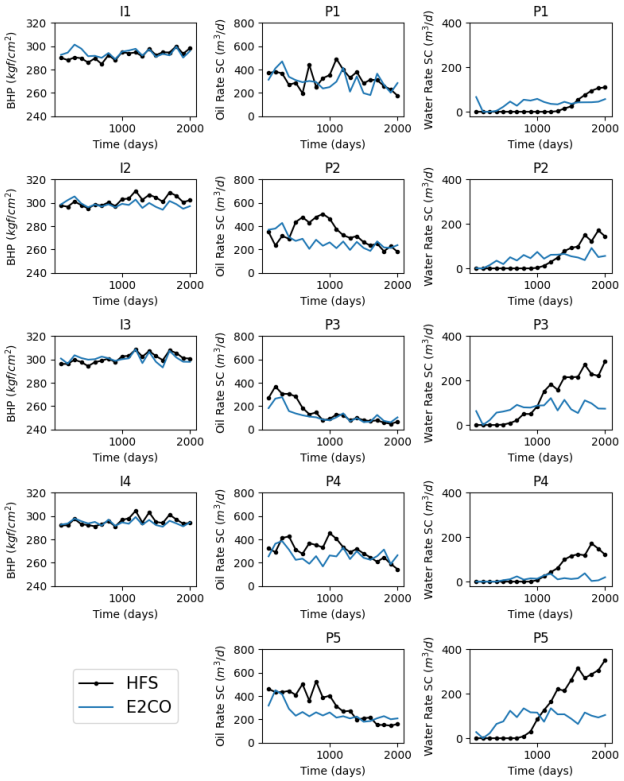

Figura 4.14: Predição dos dados de predição para treinamento da Tabela 4.8.

A Figura 4.15 ilustra os erros relativos associados das predições da Figura 4.14.

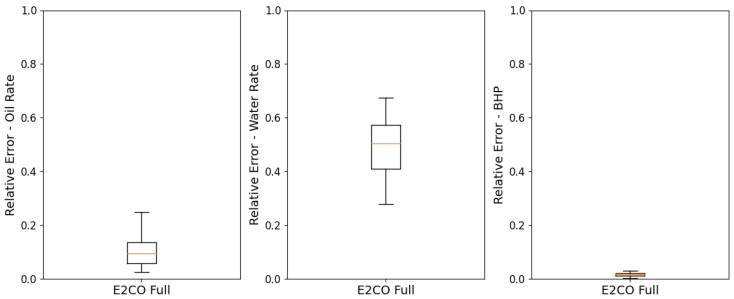

Figura 4.15: Erro de predição referente à Figura 4.14.

Como novamente não houve melhorias para prever a vazão de água, então decidimos tentar implementar uma nova estratégia: procurar pesos que podem estar correlacionados e buscar variá-los dois a dois.

4.2.6
6ª Estratégia

Procurando verificar se ao variar dois pesos simultaneamente haveria a existência de algum padrão que os relacionasse, fizemos alguns testes variando os pesos γ_{inj_pres} e γ_{prod_pres} da forma disposta na Tabela 4.9, enquanto os demais permaneceram fixos.

γ_{inj_pres}	γ_{prod_pres}
0.01	0.01
0.02	0.02
0.03	0.03
0.04	0.04
0.05	0.05
0.06	0.06
0.07	0.07
0.08	0.08
0.09	0.09
0.10	0.10
0.20	0.20
0.30	0.30
0.40	0.40
0.50	0.50
0.60	0.60
0.70	0.70
0.80	0.80
0.90	0.90
1.00	1.00

Tabela 4.9: Testando correlações entre γ_{inj_pres} e γ_{prod_pres}.

Não resultando em bons erros relativos, replicamos a mesma ideia (com os mesmos intervalos) para os pesos γ_{inj_sat}, γ_{prod_sat}, e também não obtivemos resultados tão melhores (Figuras 4.16 e 4.17) quanto aos dispostos nas Figuras 4.10 e 4.11.

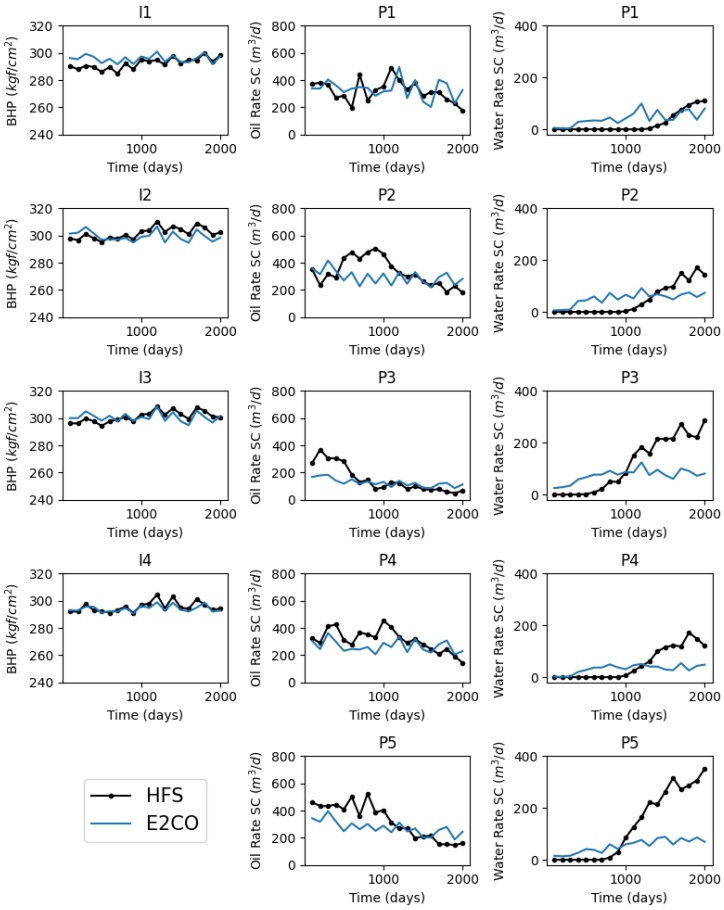

Figura 4.16: Predição dos dados de poço para treinamento escolhendo γ_{inj_sat} e γ_{prod_sat} iguais a 0.1.

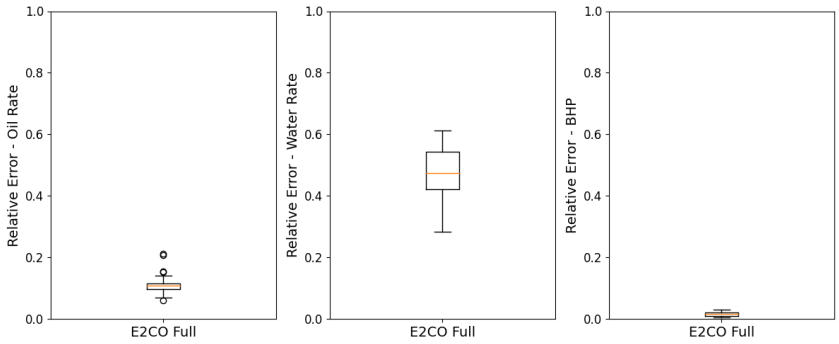

Figura 4.17: Erros relativos do treinamento exposto na Figura 4.16.

De posse disso, por não obtivermos ganhos significativos na predição de vazão de água fazendo alterações de pesos dois a dois simultaneamente variáveis, decidimos abandonar essa estratégia. Até este ponto nossas melhores predições estão expostas na Tabela 4.10.

Descrição do Peso	Valor
γ_{trans_reg}	2.8
γ_{trans_reg1}	0
γ_{reg_ABCD}	0.3
$\gamma_{transition}$	0.4
$\gamma_{transition\_wd}$	3.5
γ_{flux}	0.1
γ_{inj_pres}	1.3
γ_{inj_sat}	0
γ_{prod_pres}	0.7
γ_{prod_sat}	0
γ_{water_rate}	1
$\gamma_{prod\_sat\_qw}$	0.2
γ_{qw_neg}	0.1

Tabela 4.10: Nova Tabela de pesos parcial.

Até este momento, todos os testes foram realizados empregando a versão 3.8.19 do Python, mas como penúltima tentativa de melhorar as predições, decidimos treinar o modelo E2CO numa nova versão de Python, a 3.9.19 para verificar o que aconteceria. Faremos isso utilizando os mesmos hiperparâmetros, considerando as mesmas bibliotecas instaladas, a menos da versão de Python.

4.2.7
7ª Estratégia

Como já dito na subseção 3.3.3, executamos o treinamento em duas CPUs distintas. A Tabela 4.11 descreve as situações.

CPU	Versão do Python	Tempo de treinamento (min.)
Intel Xeon E5	3.9.19	35
Intel Core i5	3.8.19	28

Tabela 4.11: Versões do Python empregadas para treinamento.

Para verificar se há diferença de predições por conta das versões distintas de Python, consideremos o exemplo da Figura 4.14 originada por um treino na versão 3.8.19 do Python. Ao compararmos com a Figura 4.18, advinda de treino sob as mesmas condições de simulação e bibliotecas, a menos da versão do Python, que foi a 3.9.19, vemos que de fato, as predições são absolutamente diferentes. Por conta disso, acreditamos que a distinção das versões de Python é um fator altamente importante para a mudança nas predições dos dados de poço.

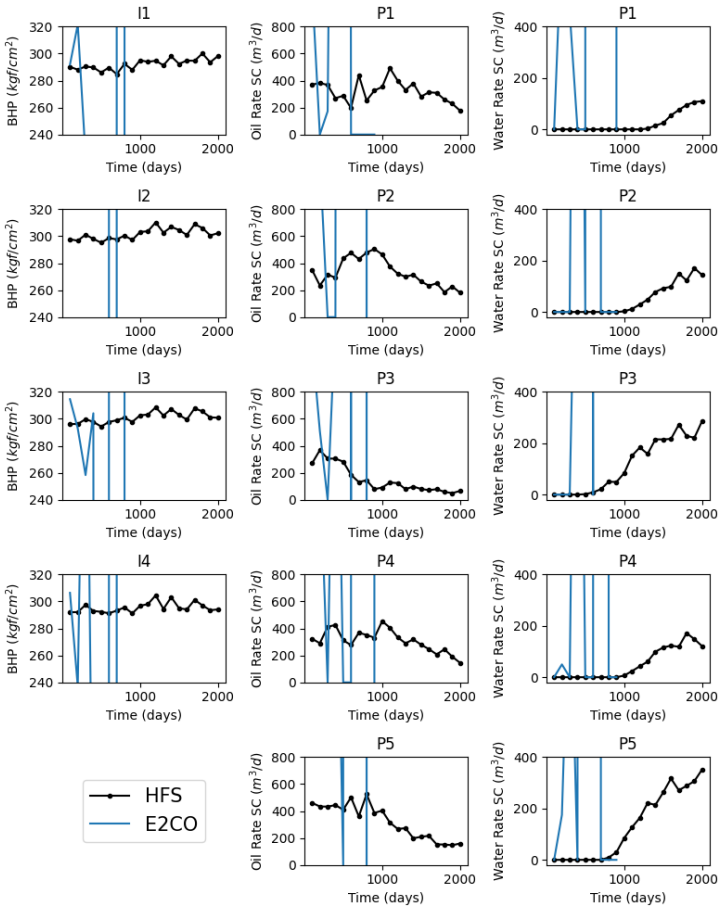

Figura 4.18: Treinando o E2CO com os dados da Tabela 4.10, empregando a versão do Python 3.9.19.

Como última tentativa de aprimorar os dados de poço, e considerando que já estávamos testando esta versão do Python, decidimos explorar uma variação da *seed* nesta mesma versão. A seguir, discutiremos a ideia de modificar a semente responsável pela geração de números aleatórios, que inicializam a etapa de treinamento. Nosso objetivo é avaliar se essa alteração pode resultar em predições mais precisas, especialmente no que se refere à vazão de água.

4.2.8
8ª Estratégia

No contexto de *Machine Learning*, (DUTTA; ARUNACHALAM; MISAILOVIC, 2022) destacam que muitos algoritmos possuem uma natureza inerentemente aleatória, isto é, múltiplas execuções do mesmo algoritmo com os mesmos dados de entrada e configurações podem gerar resultados distintos. Para lidar com essa problemática, os autores apontam que costuma-se definir sementes para os geradores de números aleatórios utilizados no código em teste. Essa prática pode tornar a execução dos testes determinística, reduzindo

a influência da aleatoriedade.

Nos testes realizados até este momento, fizemos uso somente da semente 124. Assim, decidimos empiricamente tentar alterar o valor da semente para 1234, usando a versão 3.9.19 do Python. Para fazer os testes, repetimos os procedimentos detalhados nas subseções 4.2.2 e 4.2.5, todavia empregando essa nova semente, a fim de determinar os novos valores da Tabela 4.7 para essa configuração. Ao finalizar os testes, baseados nos mesmos procedimentos de acurácia já conhecidos, chegamos a valores interessantes, que estão expostos na Tabela 4.12.

Descrição do Peso	**Valor**
γ_{trans_reg}	2.8
γ_{trans_reg1}	0
γ_{reg_ABCD}	0.3
$\gamma_{transition}$	3.6
$\gamma_{transition\_wd}$	3.5
γ_{flux}	3.1
γ_{inj_pres}	0
γ_{inj_sat}	0
γ_{prod_pres}	3.7
γ_{prod_sat}	0
γ_{water_rate}	1
$\gamma_{prod\_sat\_qw}$	0.2
γ_{qw_neg}	0.1

Tabela 4.12: Tabela de pesos e seus respectivos melhores valores encontrados com a *seed* 1234.

Também reproduzimos os testes com a semente 1234, empregando o método comentado na subseção 4.2.6, ou seja, variando dois pesos ao mesmo tempo, e mantendo os demais fixos. Não houveram resultados exitosos. Adicionalmente, fizemos testes envolvendo uma função de perda de fluxo bifásica, e não obtivemos sucesso.

Observamos que diferentes valores de *seed* promovem distintas previsões, o que consequentemente implica que devemos "recalibrar" os pesos para que possam ser ajustados a essa nova semente. Por exemplo, ao treinar o E2CO com os dados da Tabela 4.12, adotando a *seed* 1245, tivemos uma piora nas predições (Figuras 4.19 e 4.20).

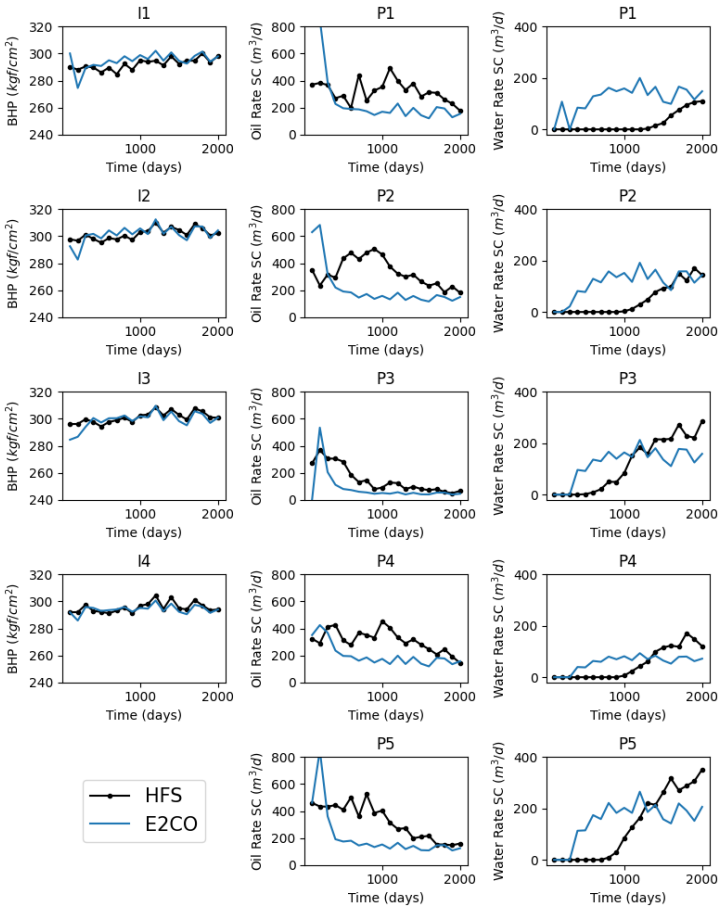

Figura 4.19: Treinando o E2CO com os dados da Tabela 4.12, empregando a *seed* 1245.

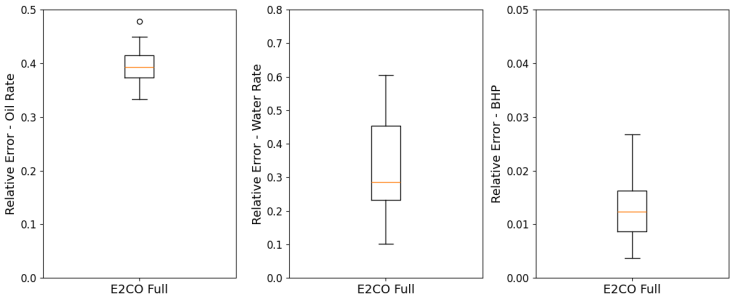

Figura 4.20: Erro relativo referente às predições da Figura 4.19.

A piora nas predições com os dados treinados usando a *seed* 1245, baseada na Tabela 4.12, ficará evidente na Seção 4.3.

4.3 Previsão de dados de poço

Como resultado das estratégias anteriores, os melhores hiperparâmetros aferidos estão exibidos na Tabela 4.12 (a nova tabela com os melhores valores).

Aplicando estes hiperparâmetros discutiremos os resultados obtidos relativos à pressão de fundo de poço (BHP), vazão de óleo e água, assim como apresentaremos alguns Algoritmos utilizados na pesquisa de (COUTINHO, 2022) para treinar o E2CO. Na última seção será exposta uma breve discussão da tentativa de aplicar o estimador de James-Stein.

4.3.1
Previsão da pressão de fundo de poço dos poços injetores

Notemos que na Figura 4.21 temos que uma boa aproximação da previsão de pressão de fundo de poço (BHP) obtida através do método E2CO para os quatro poços injetores. Isso é evidenciado visualmente, pois nota-se que as curvas do gráfico azul, que denotam a previsão do E2CO se aproximam da simulação de alta fidelidade (HFS). Quando comparado à Figura 2.25 de (COUTINHO, 2022), pode-se notar um perfil similar de aproximação.

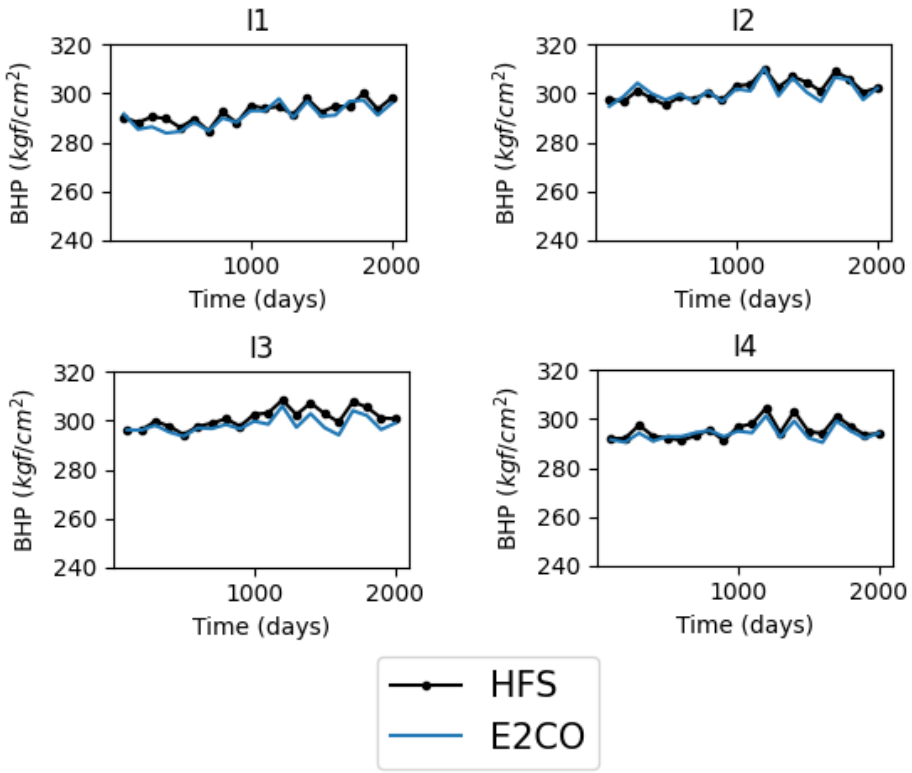

Figura 4.21: Previsão de dados de pressão de fundo de poço para uma amostra de validação.

Com fins de verificar o erro relativo na previsão de BHP, consideremos a Figura abaixo.

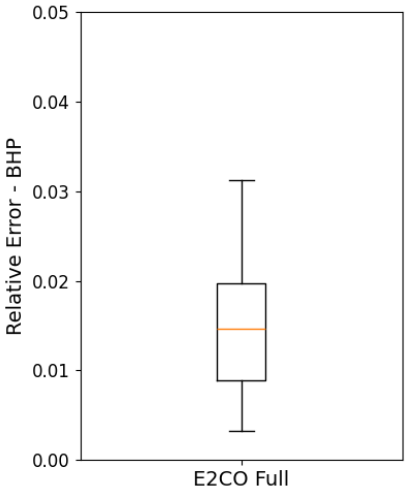

Figura 4.22: Erro do conjunto de validação do E2CO para pressão de fundo de poço.

Percebe-se que a mediana de distribuição, simbolicamente representada pela linha laranja dentro da caixa apresenta um erro relativo em torno de 1,5 %, o que indica que o modelo E2CO realizou uma previsão confiável para o BHP. Utilizando os conjuntos de hiperparâmetros declarados em sua tese, (COUTINHO, 2022) encontrou uma mediana de distribuição apontando um erro relativo de 0.2 % para BHP, consoante ilustrado na Figura 2.21 de seu trabalho. Os resultados alcançados são diferentes, e podemos notar que neste trabalho foram utilizadas versões distintas da biblioteca TensorFlow e do Python empregadas por (COUTINHO, 2022), isto é, o autor utilizou a versão 2.3.1 do TensorFlow e a versão 3.7.0 do Python, e aqui fez-se o uso das versões 2.13.0 e 3.9.19 do Tensorflow e do Python, respectivamente. Além disso, foram encontrados outros pesos ótimos distintos dos de (COUTINHO, 2022).

Ainda quanto à Figura 4.22, nota-se que as extremidades da caixa ilustram os quartis superior e inferior que apresentam erros relativos da ordem de 2 a 1 %, respectivamente. As linhas extremas (*whiskers*) indicam os valores máximo e mínimo da distribuição de erro relativo dos dados, em particular, eles são dotados da ordem de 3 e 0.3 % de erro aproximadamente, respectivamente. Ademais, não houve a presença de *outliers*. Já no trabalho de (COUTINHO, 2022), os *whiskers* foram em torno de 0.3 % e 0.1 %, e houve presença de *outliers* apontando para valores entre 0.3 e 0.4 % de erro relativo.

Adicionalmente, durante as simulações, ao testar variações nos pesos apresentados na Tabela 4.12, especialmente aplicando a estratégia de alterar apenas um peso enquanto os demais permaneciam fixos, constatou-se que as mudanças na predição, e sobretudo no erro relativo foram mínimas. Esse

resultado é positivo, pois indica que as funções de perda associadas estão desempenhando corretamente o seu papel.

4.3.2 Previsão da vazão de óleo dos poços produtores

No que tange a vazão/taxa de óleo, podemos observar que o método E2CO tentou aproximar-se de HFS, consoante ao exposto na Figura 4.23.

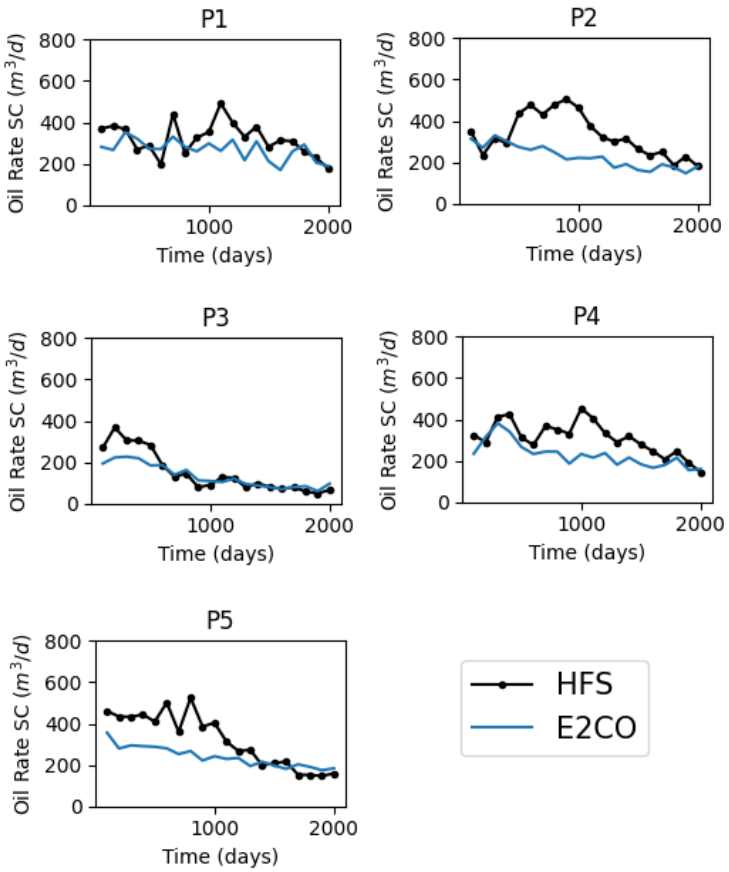

Figura 4.23: Previsão de dados de vazão de óleo para uma amostra de validação.

A Figura 4.24 destaca que a mediana do erro relativo para esta previsão é da ordem de 32 % o que evidencia uma razoável aproximação. Já em (COUTINHO, 2022), o autor conseguiu obter uma mediana em torno de 7 % de erro relativo, conforme destacado na sua tese. É válido frisar que os quartis inferior e superior são da ordem de 22 e 35 %, respectivamente, e que os (*whiskers* - valores mínino e máximo) são de aproximadamente 8 % e 44 %, respectivamente. Assim como no caso da percentagem de erro relativo de BHP, também não há a presença de *outliers* para a taxa de óleo. Por outro lado, (COUTINHO, 2022) encontrou *whisker* minímo em torno de 4 %

e *whisker* máximo por volta de 11 %, além um *outlier* por volta desta mesma percentagem.

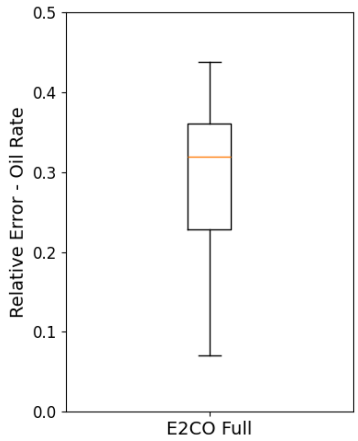

Figura 4.24: Erro do conjunto de validação do E2CO para taxa de óleo.

Na Figura 4.25 temos a produção acumulada de óleo. Nessa Figura vale pontuar que a linha reta pontilhada indica a produção acumulada de óleo dada pela HFS em 1000 m^3, e os pontos em cor verde ilustram a produção acumulada de óleo fornecidas pelo método E2CO. A relativa distância dos pontos de cor verde à reta, de fato corroboram com o alto valor de erro relativo da predição de vazão de óleo (Figura 4.24).

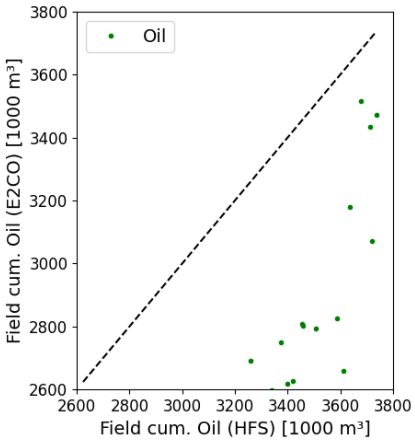

Figura 4.25: Produção acumulada de óleo.

Claramente pelo perfil do *boxplot* (Figura 4.26), temos que a produção de óleo acumulada apresenta uma mediana de distribuição relativamente alta, por volta de 30 %, o que faz sentido dado os comentários e observações realizadas em torno da Figura 4.24.

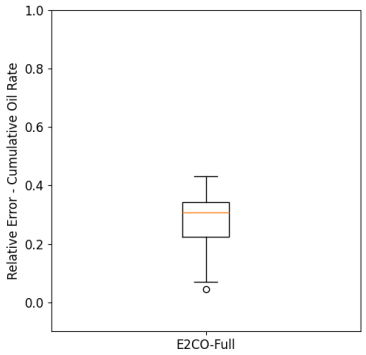

Figura 4.26: Erro de Produção acumulada de óleo.

Como pode-se observar, na Figura acima a previsão da taxa de óleo prevista via E2CO não é tão precisa e isso já era esperado, dado a discussão acima referente às respostas de cunho estatístico destacadas principalmente em torno da Figura 4.24. Por outro lado, (COUTINHO, 2022) informa na Figura 2.23 de seu trabalho, que sua produção cumulativa de óleo está mais bem distribuída em torno da reta que representa a produção acumulada de óleo por meio da HFS.

Posto isso, é notório destacar a necessidade de refinar os processos de otimização que levaram a escolha dos melhores hiperparâmetros presentes no sistema dinâmico, com fins de obter resultados mais precisos, e consequentemente levando a nuvem de pontos de cor verde a se aproximar da reta que fornece a produção acumulada de óleo via HFS.

4.3.3
Previsão da vazão de água dos poços produtores

Notemos que na Figura 4.27, especificamente obtemos razoáveis previsões de vazão de água. Em particular, visualmente P2 e P3 são os poços em que há melhor aproximação da simulação de alta fidelidade (HFS).

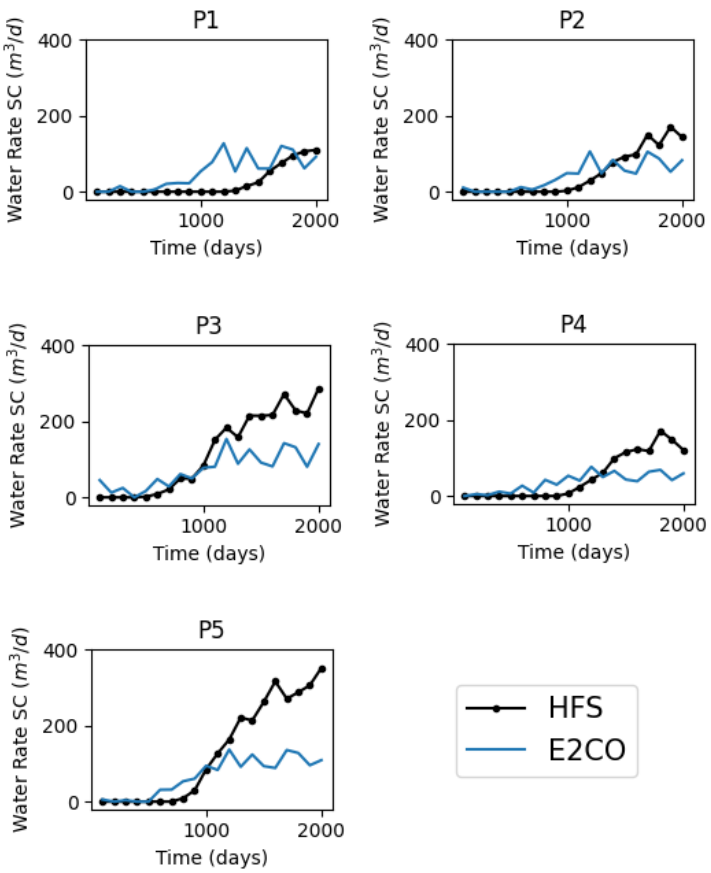

Figura 4.27: Previsão de dados de vazão de água para uma amostra de validação.

A discussão elencada no parágrafo acima se apresenta nos resultados dos erros relativos para a previsão de água (Figura 4.28), no sentido de que encontrou-se um erro na mediana de distribuição em torno de 33 % com quartis inferior e superior da ordem de 27 a 47 % de erro relativo, respectivamente. Também nota-se a presença de *whiskers* mínino e máximo em torno de 12 e 69 % de erro relativo, respectivamente. Na sua tese, na Figura 2.21 (COUTINHO, 2022) encontrou mediana de distribuição em torno de 11 %, com quartis minímo e máximo de aproximadamente 8 e 15 %, respectivamente, além de *whiskers* mínimo e máximo em torno de 5 e 25 %, respectivamente.

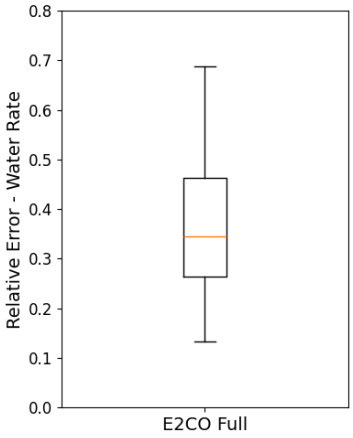

Figura 4.28: Erro do conjunto de validação do E2CO para taxa de água.

Ainda quanto à Figura 4.28, não há a presença de *outliers*. Todavia, (COUTINHO, 2022) encontrou três *outliers* com aproximadamente 26, 30 e 37 % de erro relativo.

Nossa produção acumulada de água claramente está aquém do ideal (Figura 4.29). Observa-se que boa parte dos pontos da distribuição de água acumulada encontra-se a uma distância considerável da reta que representa o campo acumulado de água estimado pela HFS.

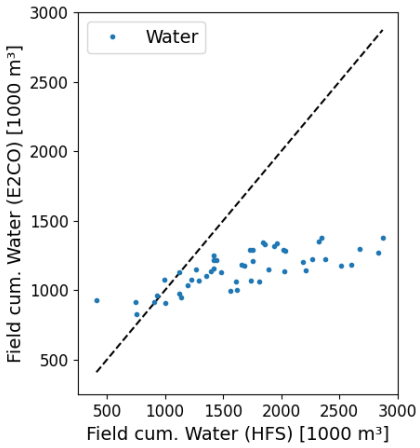

Figura 4.29: Produção acumulada de água.

Pelo perfil do *boxplot* (Figura 4.30), temos que a produção de água acumulada, assim como a do óleo apresenta uma mediana de distribuição de erro relativo ainda relativamente alta, por volta de 30 %. Isso faz sentido dado os comentários e observações realizadas em torno da Figura 4.28.

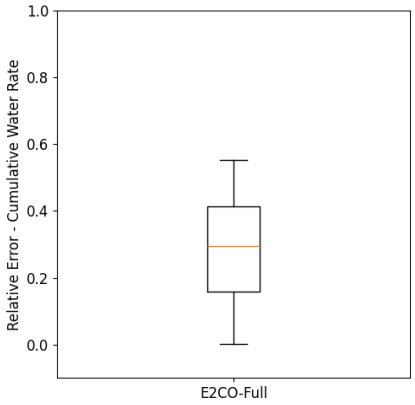

Figura 4.30: Erro de Produção acumulada de água.

Já (COUTINHO, 2022) expõe na Figura 2.23 de sua tese que sua produção acumulada de água se aproximou de forma significativa da reta que indica a produção ideal de água acumulada (dada pela HFS), o que de fato corrobora com melhores dados de erro relativo citados na discussão centrada em torno de sua Figura 2.21. Vale comentar também que o autor explica que realmente há problemas na estimativa de vazão de água, sobretudo antes da chegada de água no poço produtor, e aponta que isso poderia ser solucionado restringindo seus valores para apenas valores positivos.

Em seguida, faremos comentários gerais sobre a percepção das predições de forma conjunta.

4.4 Percepções gerais das previsões de dados de poço via E2CO

Durante as simulações de execução, percebeu-se alguns padrões de comportamento que serão descritos a seguir. Por exemplo, diversos experimentos mostraram boas previsões para BHP e vazão de óleo. Todavia, a previsão da vazão de água foi seriamente comprometida, conforme visualizado na terceira coluna da Figura 4.31.

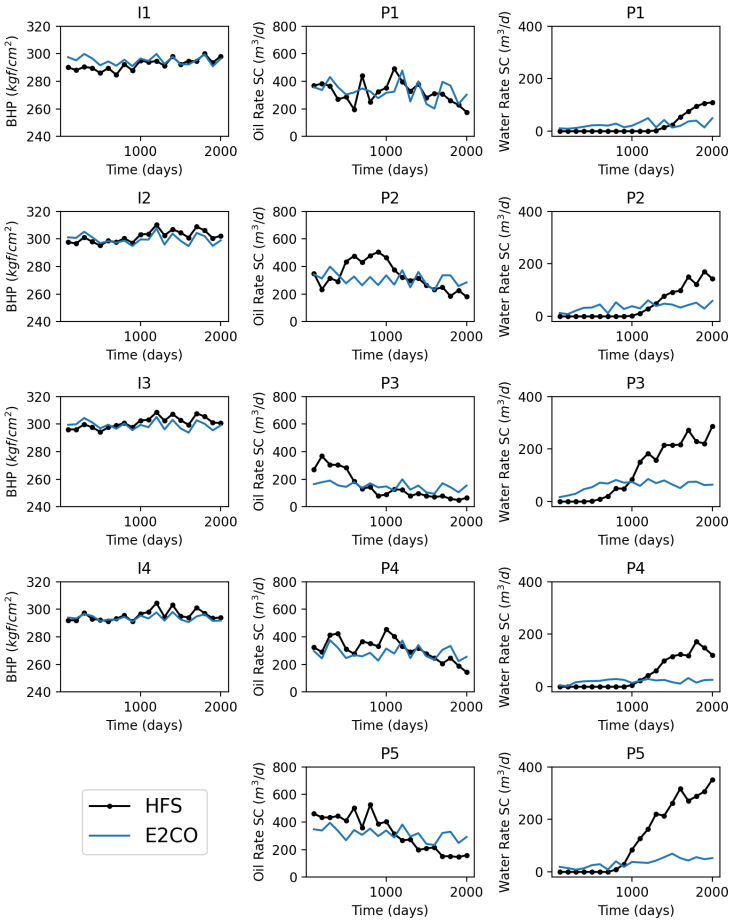

Figura 4.31: Previsão de dados de poço para uma amostra de validação.

A discussão do parágrafo acima é endossada na Figura 4.32, no sentido de que tem-se uma mediana de erro relativo muito alta para a vazão de água (superior a 60 %) e, ao mesmo tempo, uma boa previsão para a vazão de óleo (mediana de distribuição menor do que 10 %). Quanto ao erro relativo de BHP, novamente, é claro que é relativamente baixo.

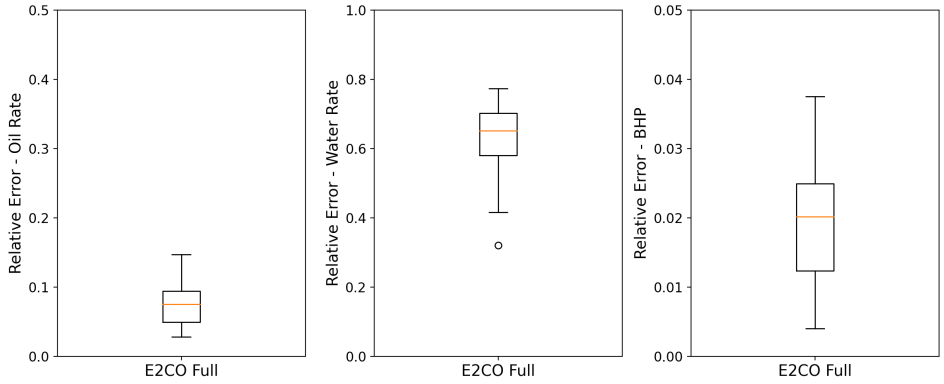

Figura 4.32: Erro relativo das previsões.

Sob a mesma ótica já discutida anteriormente, tem-se que a produção

acumulada de óleo e água são bem discrepantes, devido aos pontos de distribuição de óleo acumulado se distribuírem bem em torno da reta que indica a produção ideal de óleo acumulado (dada pela HFS), reforçando os resultados supracitados, com o contrário valendo para a produção acumulada de água.

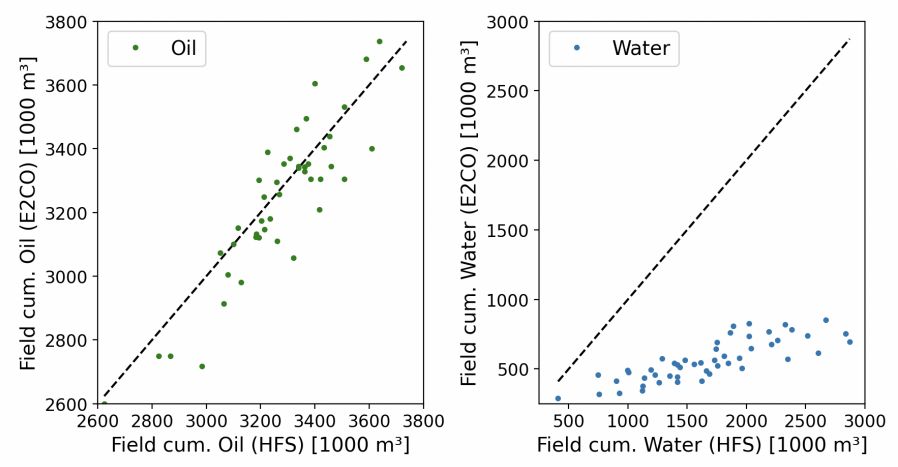

Figura 4.33: Produção acumulada de óleo e água.

Dado o contexto acima, como já era esperado, o *boxplot* do erro relativo de predição de óleo cumulativo é bem baixo para o caso do óleo, e significativamente alto para o caso da água (Figura 4.34).

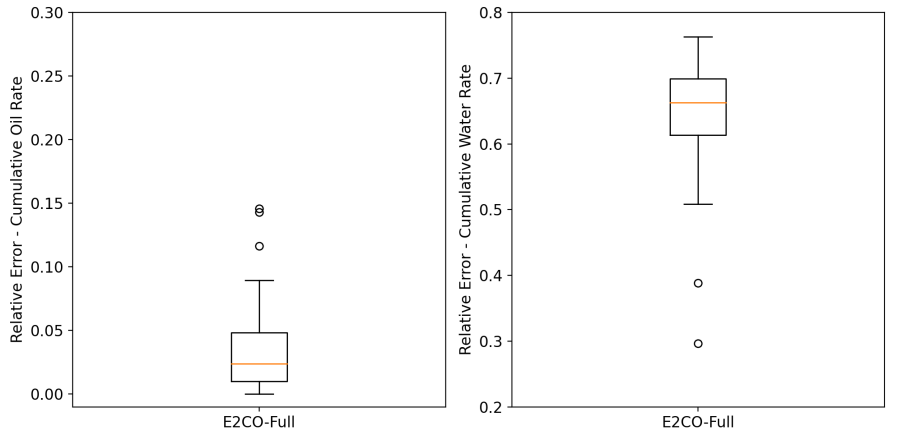

Figura 4.34: Erro relativo da produção acumulada de óleo e água.

Os resultados acima foram encontrados fazendo uso da Tabela 4.13, e alcançados ao testar uma diferente versão de Python (3.8.19), mas mantendo a versão do TensorFlow em 2.13.0.

Descrição do Peso	**Valor**
γ_{trans_reg}	2.8
γ_{trans_reg1}	0
γ_{reg_ABCD}	0.3
$\gamma_{transition}$	0.4
$\gamma_{transition\_wd}$	3.5
γ_{flux}	0.1
γ_{inj_pres}	1.3
γ_{inj_sat}	0
γ_{prod_pres}	0.01
γ_{prod_sat}	0
γ_{water_rate}	1
$\gamma_{prod\_sat\_qw}$	0.2
γ_{qw_neg}	0.1

Tabela 4.13: Exemplo de Tabela de pesos para evidenciar a discrepância entre os erros relativos de vazão de óleo e água.

Quanto ao padrão recorrente observado de que ao diminuir o erro relativo da vazão de óleo, o erro relativo da vazão de água aumenta significativamente, conseguimos resolvê-lo parcialmente alterando a distribuição de pesos, conforme destacado na Tabela 4.12. Dessa forma, promovemos uma melhor distribuição de erro relativo entre a vazão de óleo e água.

Vamos discutir a importância de analisar o erro relativo.

Importância do erro relativo na acurácia das predições

Consideremos o novo conjunto de pesos descrito na Tabela 4.14.

Descrição do Peso	**Valor**
γ_{trans_reg}	2.8
γ_{trans_reg1}	0
γ_{reg_ABCD}	0.3
$\gamma_{transition}$	3.6
$\gamma_{transition\_wd}$	3.5
γ_{flux}	3.1
γ_{inj_pres}	0
γ_{inj_sat}	0
γ_{prod_pres}	0.01
γ_{prod_sat}	0
γ_{water_rate}	1
$\gamma_{prod\_sat\_qw}$	0.2
γ_{qw_neg}	0.1

Tabela 4.14: Tabela de pesos e seus respectivos valores.

A Figura 4.35 refere-se às predições geradas através da atribuição dos pesos declarados na Tabela acima. Aqui, voltamos a usar a versão do Python

3.9.19, e manteremos a mesma versão do TensorFlow.

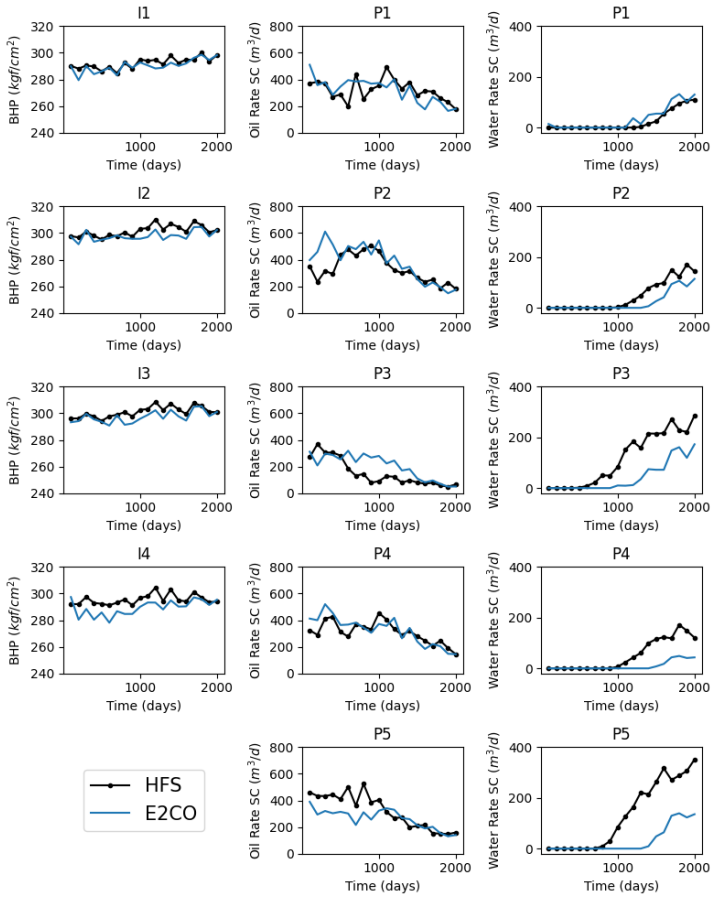

Figura 4.35: Outro exemplo de predição de dados.

Embora visualmente a predição de água da Figura 4.35 aparente ser melhor do que as demais previamente apresentadas, devido, por exemplo, a uma boa aproximação com a HFS nos primeiros 1000 dias de predição e a um perfil semelhante ao da HFS, sobretudo nos poços P1, P2 e P3, notamos na Figura 4.36 que a mediana de erro relativo foi superior (cerca de 38 %) ao apresentado na Figura 4.28 (em torno de 32 %).

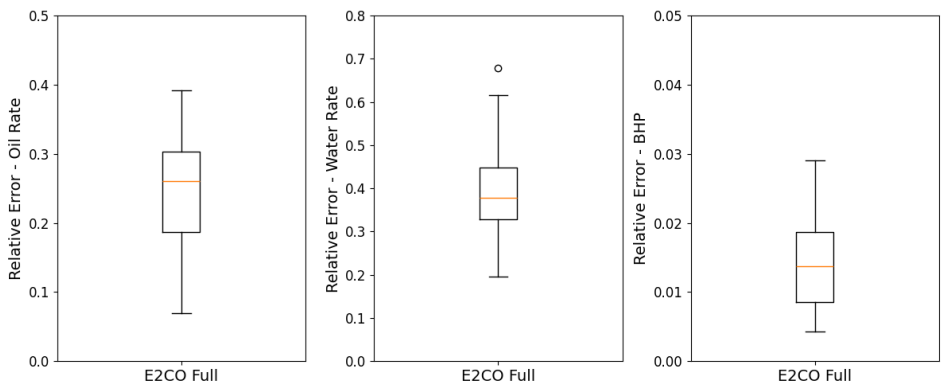

Figura 4.36: Erro relativo para outro conjunto de hiperparâmetros.

Vamos relatar a seguir, o único caso em que houve inconsistência de previsões e estimativas de erro, isto é, o acontecimento comentado no segundo parágrafo da subsubseção 3.3.3.1.

Caso em que ocorreu problema de consistência entre as repetições

O único caso em que ocorreu o problema de consistência de previsões e estimativas de erro entre as repetições é referente a um procedimento específico de treinamento que surgiu ao abordar a estratégia apresentada na subseção 4.2.6.

De fato, como já discutido anteriormente uma possível causa para isso pode estar relacionado à inicialização dos pesos das camadas e *kernels*, bem como à ordem em que os modelos são processados durante o treinamento do *batch*.

Vamos explorar com mais detalhes as funções de perda e os pesos utilizados, já que é evidente que esses elementos desempenham um papel importante nos resultados das predições. Para facilitar essa análise, utilizaremos pseudo-códigos. Primeiramente, consideremos o Algoritmo 1, que auxilia na identificação das entradas definidas para as redes neurais abordadas nesta dissertação.

Algoritmo 1: Definição das Entradas da Rede

Entrada: `nX`, `nY`, `nC`, `nW`, `nObs`, `way_well_data`

▷ em que esses parâmetros indicam dimensão da malha em x e y, número de estados/coordenadas (pressão e saturação), número de poços e identificação do modelo empregado, respectivamente

Define as entradas para a rede neural usando o framework Keras, com foco em modelagem de dados relacionados a estados no tempo (t e t+1), campos de permeabilidade, máscaras, controles de poço e dados de produção observados.

`AE_x_t` ← *Estado no passo de t*

`AE_x_tp1` ← *Estado no passo de tempo* $t+1$

`AE_mask` ← *Máscara*

`AE_delta_t` ← *Comprimento do passo de tempo entre* t *e* $t+1$

`AE_u` ← *Controles de poço entre* t *e* $t+1$

`AE_permi` ← *Campo de permeabilidade em i*

`AE_permj` ← *Campo de permeabilidade em j*

se `way_well_data` $= 3$ **então**

    ▷ Indica o modelo testado é o E2CO

    `AE_y_tp1` ← *Dados observados no instante* $t+1$

Já no Algoritmo 2, temos a construção do modelo baseado na biblioteca Keras, empregado na etapa de treinamento. A título informativo, as saídas declaradas são relativas à predição, como era de se esperar.

Algoritmo 2: Construção do Modelo Keras

Entrada: `AE_x_t`, `AE_x_tp1`, `AE_delta_t`, `AE_u`, `AE_permi`, `AE_permj`, `AE_mask`

▷ A entrada é um array com todas essas variáveis.

Saída: `AE_x_t_hat, AE_x_tp1_hat`

▷ A saída é um array com essas variáveis.

se *way_well_data* = *3* **então**

    Adiciona os dados de produção observados no instante $t+1$ *(AE_y_tp1) à lista de entradas do modelo.*

    Inclui a estimativa do modelo para os dados de produção no instante $t+1$ *(AE_y_tp1_hat) como uma saída do modelo.*

autoencoder ← *O autoencoder é criado no momento em que o modelo Keras é efetivamente definido, associando as entradas e saídas previamente especificadas.*

Finalmente no Algoritmo 3 podemos definir as funções de perda utilizadas no treinamento. As variáveis *AE_z_t*, *AE_z_tp*1*_hat* e *AE_x_tp*1*_hat* são relativas ao *encoder, transition* e *decoder*, respectivamente na etapa de predição. Já *AE_x_t_hat* é atribuída ao *decoder* na etapa de reconstrução, e *AE_z_tp*1 é usada na etapa de *encoder* para calcular z_{tp1}. Finalmente, *AE_y_tp*1*_hat* é usado para prever os dados observados com base no novo modelo de dados de transição de poço (*transition output*).

Algoritmo 3: Definindo funções de perda

Entrada: `AE_x_t`, `AE_x_tp1`, `AE_x_t_hat`, `AE_x_tp1_hat`, `AE_z_tp1`, `AE_z_tp1_hat`, `AE_z_t`, `AE_y_tp1`, `AE_y_tp1_hat`, `AE_permi`, `AE_permj`, `AE_mask`, `ij_inj`, `ij_prod`, `myKrel`, `norm_state_pres`, `norm_state_sat`, `norm_flux`, `norm_flux_o`, `norm_flux_w`, `norm_obs_water_rate`, `indices_y_bhp`, `indices_y_qo`, `indices_y_qw`, `use_qw_norm`, `use_flux_loss`, `is_pconv`, `way_well_data`, `weights...`

```
se is_pconv então
    loss_reconstruction ← euclidean_loss_mask(AE_x_t, AE_x_t_hat, AE_mask)
    loss_prediction ← euclidean_loss_mask(AE_x_tp1, AE_x_tp1_hat, AE_mask)
senão
    loss_reconstruction ← euclidean_loss(AE_x_t, AE_x_t_hat)
    loss_prediction ← euclidean_loss(AE_x_tp1, AE_x_tp1_hat)
loss_trans ← weight_transition · euclidean_loss(AE_z_tp1, AE_z_tp1_hat)
loss_trans_reg ← weight_trans_reg · l1_reg_loss(AE_z_t)
loss_trans_reg1 ← weight_trans_reg1 · (l1_reg_loss(AE_z_tp1_hat) + l1_reg_loss(AE_z_tp1))/2.0
se use_flux_loss = '1ph' então
    loss_physics_grad ← weight_flux · loss_flux_singlePhase(...)
senão se use_flux_loss = '2ph' então
    loss_physics_grad ← weight_flux · loss_flux_twoPhase(...)
loss_inj_gridblock_pres ← weight_inj_gridblock_pres · loss_wells_gridblock(..., 'pressure')
loss_inj_gridblock_sat ← weight_inj_gridblock_sat · loss_wells_gridblock(..., 'saturation')
loss_prod_gridblock_pres ← weight_prod_gridblock_pres · loss_wells_gridblock(..., 'pressure')
loss_prod_gridblock_sat ← weight_prod_gridblock_sat · loss_wells_gridblock(..., 'saturation')
se way_well_data = 3 então
    se weight_water_rate = 1 então
        loss_well_data ← weight_transition_wd · euclidean_loss(AE_y_tp1, AE_y_tp1_hat)
    senão
        loss_well_data ← weight_transition_wd · well_data_loss(...)
    loss_prod_sat_qw ← weight_prod_sat_qw · loss_prod_sat_qw(...)
    loss_qw_neg ← weight_qw_neg · loss_qw_neg(...)
```

Analisando a estrutura do Algoritmo 3, observa-se que os pesos mais relevantes são *weight_flux*, *weight_transition_wd*, *weight_prod_sat_qw* e *weight_qw_neg*. Durante as simulações, constatou-se que, ao adotar a estra-

tégia previamente definida de variar esses pesos isoladamente, mantendo os demais fixos, foi possível identificar bons valores para esses hiperparâmetros. Como resultado, as previsões relacionadas à pressão de fundo de poço (BHP) e à vazão de óleo apresentaram melhorias substanciais. Além disso, verificou-se que alterações nos outros pesos não impactaram significativamente o erro relativo de BHP, que permaneceu inferior a 3 %.

No que se refere à melhoria na predição de água, foram observados avanços ao realizar testes com diferentes valores de *seed*. Isso ocorre porque a alteração da semente gera diferentes valores aleatórios, que influenciam a inicialização de todo o procedimento de treinamento. Dessa forma, o uso de algumas sementes resultou em predições superiores a outras. Entretanto, é importante destacar que devido ao elevado custo computacional, não é viável variar excessivamente esse hiperparâmetro, pois observou-se a necessidade de recalibrar os pesos previamente ajustados sempre que a semente era alterada.

Por fim, apresentamos o Algoritmo 4, que nos mostra o que já era esperado, que os pesos somente são ativados ao serem atribuídos valores positivos aos mesmos.

Algoritmo 4: Adicionando as funções de perda definidas ao modelo Keras

```
Adicionar as seguintes perdas:
- Perda para reconstrução, que mede o erro entre as entradas originais e as
  reconstruídas pelo autoencoder;
- Perda para previsão;
- Perda relacionada à transição.
se weight_trans_reg > 0 então
    Adicionar uma perda adicional loss_trans_reg.
se weight_trans_reg1 > 0 então
    Adicionar uma perda adicional loss_trans_reg1.
se weight_flux > 0 e use_flux_loss ∈ {'1ph', '2ph'} então
    Adicionar a perda loss_physics_grad.
se way_well_data = 3 então
    se weight_transition_wd > 0 então
        Adicionar a perda loss_well_data.
    se weight_prod_sat_qw > 0 então
        Adicionar a perda loss_prod_sat_qw_.
    se weight_qw_neg > 0 então
        Adicionar a perda loss_qw_neg_.
se weight_inj_gridblock_pres > 0 então
    Adicionar a perda loss_inj_gridblock_pres.
se weight_inj_gridblock_sat > 0 então
    Adicionar a perda loss_inj_gridblock_sat.
se weight_prod_gridblock_pres > 0 então
    Adicionar a perda loss_prod_gridblock_pres.
se weight_prod_gridblock_sat > 0 então
    Adicionar a perda loss_prod_gridblock_sat.
```

Embora não tenhamos conseguido apresentar uma estratégia bem definida sobre como escolher o melhor conjunto de hiperparâmetros, fizemos uma análise completa de todas as tentativas realizadas no estudo e suas implicações nas predições.

A seguir, discutiremos a respeito da viabilidade do uso do estimador de James-Stein, com fins de que possamos obter resultados de predições mais confiáveis, devido às características inerentes deste estimador.

4.5 Estimador de James-Stein

Conforme visto nos Teoremas 3.1 e 3.2, para aplicar o estimador de James-Stein (JS), precisamos que as distribuições associadas sejam normais e/ou com densidade simétrica e unimodal. Além disso, quando olhamos para a equação (3-31), lembremos que a dimensão do vetor X tem que ser maior ou igual a 3. Em particular, no nosso caso $\dim X = 3$, e tal vetor possui como entradas os dados de predição de BHP, vazão de óleo e água.

Para se ter uma ideia da viabilidade de uso do estimador, consideremos as distribuições em formato de histograma dos dados previstos do poço injetor I1, e do poço produtor P1, dispostos na Figura 4.37, por exemplo.

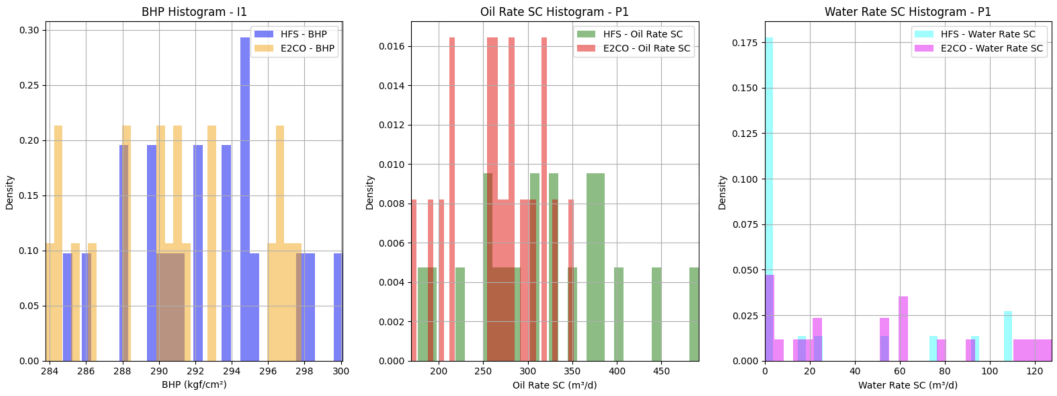

Figura 4.37: Histogramas com as distribuições de dados de BHP do poço injetor I1, e do poço produtor de óleo e água P1, respectivamente.

Ao analisar a Figura 4.37, observa-se que o perfil dos histogramas de vazão de água é bastante semelhante para os demais poços produtores. Isso pode ser constatado pela predição de dados gerada pela HFS, apresentada na Figura 4.27. Em particular, nota-se que, nos primeiros 600 dias, todos os poços exibem uma vazão constante de água próxima a 0 m^3/dia. Esse comportamento é claramente evidenciado pelo pico de densidade correspondente no histograma da Figura 4.37. Essa informação já indica que não é possível ajustar uma distribuição normal aos dados de predição de vazão de água, tampouco uma distribuição simétrica.

O pico de densidade mencionado já era esperado, conforme a discussão elencada no penúltimo parágrafo da seção 4.3.3 trazida à tona por (COUTINHO, 2022). Naquela análise, foram destacados os desafios relacionados à estimativa da vazão de água, especialmente no período anterior à chegada de água ao poço produtor. Durante essa fase, a vazão de água é nula, o que gera variações na curva de água, já que o sistema de treinamento da rede neural é

exposto a um longo intervalo de tempo sem fluxo, consoante visto no perfil de HFS da vazão de água na Figura 4.27.

Além disso, ao observar os histogramas apresentados, conclui-se que as variáveis analisadas não possuem perfis compatíveis com distribuições normais, não atendendo, portanto, às condições do Teorema 3.1. Ademais, as variáveis também não satisfazem versões mais flexíveis do estimador de James-Stein (Teorema 3.2), já que a densidade da variável "vazão de água" claramente não apresenta simetria. Há ainda uma dificuldade adicional para se empregar o JS, devido termos que garantir que a variância dos dados seja do tipo homocedástica, consoante Observação 1.

Em síntese, considerando que estamos utilizando redes neurais para prever o comportamento dos poços e tomando como referência os dados fornecidos pela HFS como controle da qualidade das predições, a aplicação do estimador de James-Stein em nosso problema fica restrita à natureza das distribuições fornecidas pela HFS e à variância homocedástica dos dados.

Por fim, embora a aplicação do estimador de James-Stein não tenha sido viável neste estudo, é importante ressaltar que outros trabalhos baseados em novos dados de HFS talvez apresentem histogramas com distribuições que atendam às condições exigidas pelos teoremas mencionados. Em particular, a presença de distribuições unimodais e simétricas possibilitaria a utilização do estimador de James-Stein nesses contextos.

5
Conclusões e trabalhos futuros

Recordamos que o objetivo desta dissertação é avaliar a viabilidade de construir um modelo *proxy* para simulação de reservatórios de petróleo, utilizando o E2CO, capaz de prever pressão de fundo de poço e vazões de óleo e água por 2000 dias, com resultados comparáveis aos de (COUTINHO, 2022). O principal desafio foi estudar como a sensibilidade aos hiperparâmetros afeta a acurácia do modelo. Além disso, foi investigado o uso do estimador de James-Stein (JS) para melhorar a precisão das estimativas, aplicando-o tanto nos dados da simulação de alta fidelidade (HFS) quanto nos obtidos pelo E2CO.

O capítulo 4 fez um relato de várias simulações realizadas, dos padrões observados durante essas simulações e da influência dos pesos e das funções de perda nas predições de dados de poço. Essas discussões buscam contribuir para um entendimento mais claro do tema *Embed to Control and Observe* (E2CO).

Quanto à questão do estimador de James-Stein, havia-se de fato uma expectativa de que sua implementação melhorasse a confiabilidade dos dados de poço. Verificamos que não se adapta ao modelo trabalhado.

Quanto aos trabalhos futuros, sugere-se que seja realizado um novo treino do E2CO utilizando a versão 3.7.4 do Python e 2.3.1 do TensorFlow empregadas nos experimentos de (COUTINHO, 2022). Devido à natureza do estimador de James-Stein, recomenda-se aplicá-lo em uma modelagem trifásica via E2CO, considerando, por exemplo, um sistema que prevê os seguintes dados de poço: pressão no fundo do poço (BHP), vazão de óleo, vazão de água e vazão de gás, ou seja, $\dim X = 4$. Assim, dependendo do perfil de distribuição dos histogramas dos dados presentes na HFS deste estudo, pode-se cumprir as hipóteses dos teoremas apresentados.

Em paralelo, sugere-se investigar o comportamento das distribuições de dados de poço com o uso de estimadores de encolhimento que considerem distribuições não normais.

Além disso, acreditamos que um problema de pesquisa relevante seja resolver a questão da chegada de água aos poços produtores, conhecida como *water breakthrough*. Essa questão é especialmente importante porque o simulador IMEX, que gera a HFS, busca replicar o comportamento físico do poço, refletindo as condições reais do sistema injetor-produtor, nas quais o fenômeno da chegada de água é intrínseco. Diante disso, surge uma questão central: é possível considerar apenas os dados a partir do momento em que a água começa a chegar ao poço produtor? Gostaríamos de implementar a abordagem proposta

por (COUTINHO, 2022) para solucionar esse problema, restringindo os valores apenas aos positivos (dados a partir do *water breakthrough*), e analisar suas implicações nas predições.

6
Referências bibliográficas

ATANGANA, A. Chapter 2 - principle of groundwater flow. In: ATANGANA, A. (Ed.). **Fractional Operators with Constant and Variable Order with Application to Geo-Hydrology**. [S.l.]: Academic Press, 2018. p. 15–47.

BAHRAMI, P.; MOGHADDAM, F. S.; JAMES, L. A. A review of proxy modeling highlighting applications for reservoir engineering. **Energies**, MDPI, v. 15, n. 14, p. 5247, 2022.

CMG. **IMEX, Black Oil & Unconventional Reservoir Simulator**. 2024. Disponível em: <https://www.cmgl.ca/solutions/software/imex/>. Acesso em: 20 out. 2024.

COOPER, A. F.; FRANKLE, J.; SA, C. D. Non-determinism and the lawlessness of machine learning code. In: **Proceedings of the 2022 Symposium on Computer Science and Law**. [S.l.]: ACM, 2022. (CSLAW '22), p. 1–8.

COUTINHO, E. J. R. **Embedding Reservoir Physics into Machine Learning**. Tese (Doctor of Philosophy) — Texas A&M University, 2022.

DE, S.; DOOSTAN, A. Neural network training using l_1-regularization and bi-fidelity data. **Journal of Computational Physics**, v. 458, p. 111010, 2022.

DEOTA, U. P. **Physics-Aware Deep Learning Based Proxy For 3D Petroleum Reservoir Simulation**. Dissertação (Mestrado) — Texas AM University, 2023.

DUTTA, S.; ARUNACHALAM, A.; MISAILOVIC, S. To seed or not to seed? an empirical analysis of usage of seeds for testing in machine learning projects. In: **2022 IEEE Conference on Software Testing, Verification and Validation (ICST)**. [S.l.: s.n.], 2022. p. 151–161.

FOURDRINIER, D.; STRAWDERMAN, W. E.; WELLS, M. T. **Shrinkage estimation**. [S.l.]: Springer, 2018.

GAJO, C. A. **Propriedades e aspectos geométricos de estimadores tipo James-Stein e do estimador de Hartigan**. 156 p. Tese (Doutorado em Estatística e Experimentação Agropecuária) — Universidade Federal de Lavras, Lavras, 2016.

GÉRON, A. **Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems**. [S.l.]: "O'Reilly Media, Inc.", 2019.

GU, W.; BAI, S.; KONG, L. A review on 2d instance segmentation based on deep neural networks. **Image and Vision Computing**, v. 120, p. 104401, 2022. ISSN 0262-8856.

HE, K. et al. Deep residual learning for image recognition. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2016. p. 770–778.

HEUMANN, C. James–stein estimator. In: _____. **International Encyclopedia of Statistical Science**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. p. 699–701. ISBN 978-3-642-04898-2.

HWANG, J.-S. et al. Determination of optimal batch size of deep learning models with time series data. **Sustainability**, MDPI, v. 16, n. 14, p. 5936, 2024.

JAMES, W.; STEIN, C. Estimation with quadratic loss. **Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability**, University of California Press, v. 1, p. 361–380, 1961.

JANSEN, J. D. **A Systems Description of Flow Through Porous Media**. [S.l.]: Springer, 2013.

JAVAHERI, S. H.; SEPEHRI, M. M.; TEIMOURPOUR, B. Chapter 6 - response modeling in direct marketing: A data mining-based approach for target selection. In: ZHAO, Y.; CEN, Y. (Ed.). **Data Mining Applications with R**. Boston: Academic Press, 2014. p. 153–180. ISBN 978-0-12-411511-8.

JIN, Z.; LIU, Y.; DURLOFSKY, L. Deep-learning-based surrogate model for reservoir simulation with time-varying well controls. **Journal of Petroleum Science and Engineering**, Elsevier B.V, v. 192, p. 107273, 2020.

KINGMA, D. P.; BA, J. **Adam: A Method for Stochastic Optimization**. 2017. Disponível em: <https://arxiv.org/abs/1412.6980>. Acesso em: 20 dez. 2024.

LIU, G. et al. Partial convolution based padding. **arXiv preprint arXiv:1811.11718**, Nov. 2018.

MOTAEI, E.; GANAT, T. Smart proxy models art and future directions in the oil and gas industry: A review. **Geoenergy Science and Engineering**, Elsevier B.V, v. 227, 2023.

NISE, N. S. **Control systems engineering**. [S.l.]: John Wiley & Sons, 2014.

O'SHEA, K.; NASH, R. An introduction to convolutional neural networks. **arXiv preprint arXiv:1511.08458**, 2015.

PEACEMAN, D. W. **Fundamentals of Numerical Reservoir Simulation**. [S.l.]: Elsevier, 1977. v. 6.

PEACEMAN, D. W. Interpretation of well-block pressures in numerical reservoir simulation (includes associated paper 6988). **Society of Petroleum Engineers Journal**, Society of Petroleum Engineers (SPE), v. 18, 1978.

PYTHON. **PYTHON Software Foundation. Downloads.** 2024. Disponível em: <https://www.python.org/downloads/>. Acesso em: 29 nov. 2024.

RIACH, D. **GitHub - NVIDIA/framework-determinism: Providing determinism in deep learning frameworks**. 2020. Disponível em: <https://github.com/NVIDIA/framework-determinism>. Acesso em: 30 nov. 2024.

ROSA, A.; CARVALHO, R.; XAVIER, J. **Engenharia de Reservatórios de Petróleo**. [S.l.]: Interciência, 2006.

SATHUJODA, S. T.; SHETH, S. M. **Physics-Informed Localized Learning for Advection-Diffusion-Reaction Systems**. 2023. Disponível em: <https://arxiv.org/abs/2305.03774>.

SHARMA, O. Deep challenges associated with deep learning. In: **2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)**. [S.l.: s.n.], 2019. p. 72–75.

VIANA, M.; ESPINAR, J. M. **Differential equations: a dynamical systems approach to theory and practice**. [S.l.]: American Mathematical Society, 2021. v. 212.

WATTER, M. et al. Embed to control: A locally linear latent dynamics model for control from raw images. **Arxiv**, 2015.

A
Lei de Darcy

No livro de (ROSA; CARVALHO; XAVIER, 2006), os autores relatam que, em 1856, Darcy analisou os resultados de experimentos cujo objetivo era a purificação da água por meio de filtros de areia. A partir dessas observações, ele concluiu que existia uma relação direta entre a vazão que atravessa o leito de areia e a diferença de carga associada a essa vazão. Além disso, verificou que as dimensões do meio poroso influenciavam os resultados obtidos e apresentou uma relação matemática fundamental para a compreensão do escoamento de fluidos em meios porosos. Assim, pode-se expressar matematicamente essa relação como:

$$q = KA\frac{h_1 - h_2}{L} \tag{A-1}$$

A Tabela A.1 nos informa as variáveis envolvidas na equação A-1.

Símbolo	**Descrição**
q	Vazão volumétrica através do meio poroso
K	Constante de proporcionalidade do meio poroso
A	Área transversal do meio poroso
L	Comprimento do leito poroso
$h_1 - h_2$	Diferença de carga d'água associada à vazão obtida

Tabela A.1: Descrição das variáveis da equação A-1. Fonte: Adaptado de (ROSA; CARVALHO; XAVIER, 2006).

Vale ressaltar que h_1 e h_2 são medidos através de um nível de referência comum e ilustram, respectivamente, as cargas hidráulicas na entrada e na saída do meio poroso. Dessa forma, o escoamento através do meio poroso ocorre com uma perda de carga, denotada pela diferença de alturas (h_1 e h_2). Em seu trabalho, (ATANGANA, 2018) nos fornece a Figura A.1, que nos ajuda a entender as variáveis envolvidas no processo descrito.

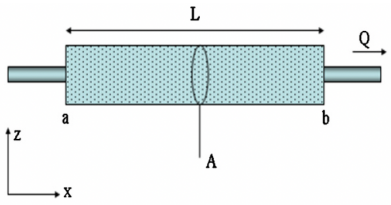

Figura A.1: Diagrama ilustrando a Lei de Darcy. Fonte: (ATANGANA, 2018).

Ao estender essa análise para outros fluidos, verificou-se que a constante de proporcionalidade K também depende da viscosidade μ e do peso específico γ do fluido, levando à seguinte expressão:

$$K = \frac{k}{\mu}\gamma. \tag{A-2}$$

Para um mesmo meio poroso, a nova constante de proporcionalidade k independe do fluido em escoamento, desde que o meio esteja completamente saturado (ROSA; CARVALHO; XAVIER, 2006). Dessa forma, k caracteriza exclusivamente o meio poroso, sendo denominada sua permeabilidade. Assim, a Lei de Darcy pode ser expressa como:

$$q = \frac{k}{\mu}A\gamma\frac{h_1 - h_2}{L}. \tag{A-3}$$

B
Alguns conceitos em *Deep Learning*

B.1
Redes Neurais Artificiais

As Redes Neurais Artificiais (ANNs - *Artificial Neural Networks*) podem ser definidas como sistemas de processamento computacional amplamente inspirados no funcionamento dos sistemas nervosos biológicos, como o cérebro humano (O'SHEA; NASH, 2015). Segundo os autores, as ANNs são compostas por um grande número de nós computacionais interconectados, conhecidos como neurônios, que operam de maneira distribuída e interligada objetivando aprender com os dados de entrada e aprimorar/otimizar a saída final.

B.2
Redes Neurais Convolucionais

As Redes Neurais Convolucionais (CNNs) são similares às ANNs tradicionais, já que contêm neurônios que se auto-otimizam via aprendizagem (O'SHEA; NASH, 2015). De acordo com (O'SHEA; NASH, 2015), cada neurônio recebe uma entrada e executa uma operação, como um produto escalar seguido por uma função não linear. Tal operação é o pilar de uma gama de ANNs. Desde os vetores de imagem bruta de entrada até a saída final da pontuação da classe, a rede expressa uma única função de pontuação perceptiva, ou seja, o peso. A última camada é dotada de funções de perda associadas às classes, e as técnicas e estratégias comuns desenvolvidas para ANNs tradicionais ainda são válidas. Por fim, destacam que a única distinção expressiva entre CNNs e ANNs tradicionais é que as CNNs são empregadas principalmente no campo do reconhecimento de padrões em imagens.

B.3
Arquitetura de Redes Neurais Convolucionais

Esta seção será inteiramente baseada na seção 2 do *paper* de (O'SHEA; NASH, 2015).

CNNs possuem três tipos de camadas. Estas são: convolucionais, de *pooling* e as totalmente conectadas. O empilhamento destas, forma a arquitetura de uma CNN. Vamos conhecer um pouco mais sobre cada uma destas camadas nas próximas subseções.

B.3.1
Camada Convolucional (CC)

Esta camada é crucial para o funcionamento de CNNs. Seus parâmetros estão centrados no emprego dos chamados *kernels* aprendíveis. Tais *kernels* cobrem toda a profundidade da entrada, embora sejam dotados de um tamanho pequeno em termos de dimensionalidade espacial.

Quando os dados chegam a uma CC, a camada convolui cada filtro (*filter*) por meio da dimensionalidade espacial da entrada para gerar um mapa de ativação bidimensional. À medida que há avanços na entrada, o produto escalar é calculado para cada valor nesse *kernel*. Com base nisso, a rede vai aprendendo os *kernels* que "disparam" ao ser notado uma *feature* específica numa determinada posição espacial da entrada. Nomeadamente, isso é conhecido como ativações. Vale comentar que cada *kernel* será dotado de um mapa de ativação correspondente, que será empilhado ao longo da dimensão de profundidade para gerar o volume de saída total da camada convolucional.

As CC têm o potencial de amenizar consideravelmente a complexidade do modelo via otimização de sua saída. Tal otimização é baseada nos hiperparâmetros: profundidade (*depth*), passo (*stride*) e configuração de preenchimento de zeros (*zero-padding*). Na Tabela B.1, podemos ver as funcionalidades de cada um.

Hiperparâmetros	**Funcionalidade**
Depth	Representar o número de neurônios dentro da camada para a mesma região da entrada.
Stride	Usado para definir a profundidade em torno da dimensionalidade espacial da entrada para posicionar o campo receptivo.
Zero-padding	Preencher a borda da entrada com zeros. É útil para fornecer maior controle no que tange à dimensionalidade dos volumes de saída.

Tabela B.1: Hiperparâmetros da CC e funcionalidades. Fonte: Adaptado de (O'SHEA; NASH, 2015).

B.3.2
Camada de *Pooling* (CP)

Também conhecida como camadas de agrupamento, estas almejam reduzir gradualmente a dimensionalidade da representação, diminuindo com mais afinco o número de parâmetros e a complexidade computacional do modelo. A CP opera sobre cada mapa de ativação na entrada, e modifica sua dimensionalidade via função "MAX".

B.3.3
Camada totalmente conectada (CTC)

A CTC é caracterizada por conter neurônios que são diretamente conectados aos neurônios nas duas camadas adjacentes, sem serem conectados a nenhuma camada dentro delas. É uma forma similar ao modo como os neurônios são organizados em formas tradicionais de ANN.

B.4
Residual Neural Network (ResNet)

Baseando-se no trabalho de (HE et al., 2016), (GU; BAI; KONG, 2022) afirma que a rede neural residual tem sido uma das mais utilizadas na atualidade. O autor também explica que, por meio de uma estrutura de aprendizado residual profundo, o ResNet consegue enfrentar o problema de degradação, extraindo mais informações dos dados originais. Sua principal contribuição está no emprego do módulo de mapeamento de identidade, que permite o ajuste de um mapeamento residual via conexões de salto ou atalhos para pular essas camadas.

B.5
Batch size

Batch size e/ou tamanho do lote pode ser definido como o conjunto de dados de amostra empregados para atualizar os pesos de um modelo de *deep learning* de uma só vez (HWANG et al., 2024).

O tamanho do *batch* pode impactar significativamente o desempenho do modelo e o tempo de treinamento (GÉRON, 2019). Em geral, o tamanho ideal será inferior a 32. O autor destaca que um *batch size* pequeno garante iterações de treinamento mais rápidas, enquanto um *batch* maior proporciona uma estimativa mais precisa dos gradientes. No entanto, na prática, essa precisão adicional não é tão relevante, pois o processo de otimização é complexo e os gradientes reais nem sempre apontam diretamente para o ótimo. Além disso, (GÉRON, 2019) ressalta que utilizar um tamanho de lote superior a 10 permite aproveitar melhor otimizações de *hardware* e *software*, particularmente em operações de multiplicação de matrizes, acelerando o treinamento. Caso seja utilizada a técnica de *Batch Normalization*, recomenda-se que o *batch size* não seja muito pequeno, geralmente não inferior a 20.

Nos dias atuais, de modo geral, os pesquisadores aplicam um tamanho de lote de forma empírica, variando de 2 a 128, por meio de tentativa e erro (HWANG et al., 2024). Entretanto, é incerto se esses valores recomendados podem realmente otimizar o desempenho preditivo dos modelos de *deep learning*. Adicionalmente,

não há uma explicação clara se o valor escolhido é o *batch size* adequado para os modelos.

B.6 *Batch Normalization*

Embora o uso de qualquer variante de ReLU possa reduzir significativamente os problemas de gradientes que desaparecem ou explodem no início do treinamento, isso não garante que eles não voltem a ocorrer ao longo do processo (GÉRON, 2019).

O autor menciona que, em 2015, foi proposta a técnica *Batch Normalization* (BN) para lidar com esses problemas. Essa abordagem consiste em inserir uma operação no modelo imediatamente antes ou depois da função de ativação de cada camada oculta, centralizando em zero e normalizando cada entrada. Em seguida, o resultado é escalado e deslocado por meio de dois novos vetores de parâmetros por camada: um para a escala e outro para o deslocamento. Dessa forma, o modelo consegue aprender a escala e a média ideais de cada uma das entradas da camada.

Por fim, (GÉRON, 2019) também destaca que de modo geral, ao incluir uma camada de BN como a primeira camada da rede neural, a padronização do conjunto de treinamento torna-se desnecessária, já que a camada de BN a realiza. No entanto, essa normalização ocorre de forma aproximada, pois é baseada em *batches* individuais e também pode reescalar e deslocar cada característica (*feature*) de entrada.

B.7 *Epochs*

Uma época pode ser pensada como um conjunto de dados de entrada completos disponíveis (SHARMA, 2019). O autor explica que, ao final de cada época, os pesos do modelo são ajustados e testados novamente contra o próximo ciclo da mesma simulação de conjunto de dados (denominado próxima época). Durante esse processo, todos os dados de treinamento precisam estar carregados na memória principal. Todavia, ao considerar conjuntos de dados extensos, manter todas as informações simultaneamente na memória principal pode ser inviável. Para contornar essa limitação, o conjunto de dados é segmentado em lotes (*batches*), que são processados sequencialmente. Cada lote é carregado na memória principal, executado e sua contribuição é acumulada (somada) e ilustrada finalmente como uma saída de época.

O autor chama a atenção que quando a descida de gradiente não consegue mais reduzir a função de custo e os valores de precisão permanecem praticamente constantes (quase iguais ou repetidos), não há vantagem em aumentar o número de

épocas. Nesse cenário, pode-se concluir que o modelo já atingiu um nível otimizado de desempenho, isto é, o "caminho" foi percorrido e otimizado.

A otimização das épocas enfrenta dois desafios principais (SHARMA, 2019). O primeiro é o subajuste (*under-fitting*), que ocorre quando o número de épocas é insuficiente para que o modelo alcance sua precisão ideal. O segundo é justamente o caso oposto, ou seja, o sobreajuste (*over-fitting*), que acontece quando, apesar do aumento no número de épocas, não há melhorias significativas no desempenho do modelo. Na Figura B.1, podemos identificar ambos os desafios.

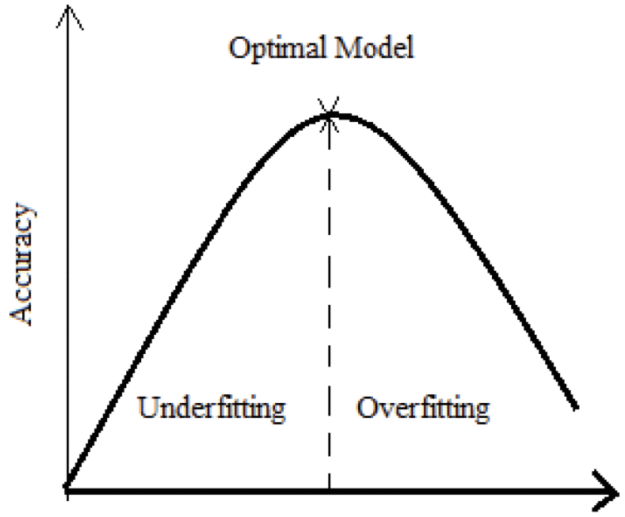

Figura B.1: Relação com o número ótimo de épocas com subajuste (*Under-Fitting*) e sobreajuste (*Over-Fitting*). Fonte: (SHARMA, 2019).