UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE TECNOLOGIA E GEOCIÊNCIAS
DEPARTAMENTO DE ENGENHARIA CIVIL
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA CIVIL

ALEXANDRE DE SOUZA JUNIOR

APLICAÇÃO DE MÉTODOS DE ORDEM REDUZIDA EM SIMULAÇÃO DE RESERVATÓRIOS DE PETRÓLEO USANDO A PLATAFORMA MRST

Recife
2021

ALEXANDRE DE SOUZA JUNIOR

APLICAÇÃO DE MÉTODOS DE ORDEM REDUZIDA EM SIMULAÇÃO DE RESERVATÓRIOS DE PETRÓLEO USANDO A PLATAFORMA MRST

Dissertação apresentada ao Programa de Pós-graduação em Engenharia Civil da Universidade Federal de Pernambuco, Centro de Tecnologia e Geociências, como requisito parcial para obtenção do título de Mestre em Engenharia Civil. Área de concentração: Simulação e Gerenciamento de Reservatórios de Petróleo.

Orientador: Prof. Dr. Bernardo Horowitz

Recife

2021

Catalogação na fonte:
Sandra Maria Neri Santiago, CRB-4 / 1267

S729a Souza Junior, Alexandre de.
Aplicação de métodos de ordem reduzida em simulação de reservatórios de petróleo usando a Plataforma MRST / Alexandre de Souza Junior. – 2021.
145 f.: figs., tabs.

Orientador: Prof. Dr. Bernardo Horowitz.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CTG. Programa de Pós-Graduação em Engenharia Civil, Recife, 2021.
Inclui referências.

1. Engenharia civil. 2. Otimização. 3. Simulação de reservatórios. 4. Método de ordem reduzida. 5. MRST. 6. Derivação automática. 7. Programação orientada a objetos. I. Horowitz, Bernardo (Orientador). III. Título.

UFPE

624 CDD (22. ed.) BCTG / 2021 - 188

ALEXANDRE DE SOUZA JUNIOR

APLICAÇÃO DE MÉTODOS DE ORDEM REDUZIDA EM SIMULAÇÃO DE RESERVATÓRIOS DE PETRÓLEO USANDO A PLATAFORMA MRST

Dissertação apresentada ao Programa de Pós-graduação em Engenharia Civil da Universidade Federal de Pernambuco, Centro de Tecnologia e Geociências, como requisito parcial para obtenção do título de Mestre em Engenharia Civil. Área de concentração: Simulação e Gerenciamento de Reservatórios de Petróleo.

Aprovada em: 02 / 03 / 2021

BANCA EXAMINADORA

Participação por videoconferência

Prof.ª Dra. Silvana Maria Bastos Afonso da Silva (Examinadora Interna)

Universidade Federal de Pernambuco

Participação por videoconferência

Dr. Juan Alberto Rojas Tueros (Examinador Externo)

Universidade Federal de Pernambuco

Participação por videoconferência

Prof. Dr. Leonardo Correia de Oliveira (Examinador Externo)

Universidade Federal de Pernambuco

AGRADECIMENTOS

Agradeço a Deus por ter me dado saúde e sabedoria para superar as dificuldades e desafios, e conseguir a realização deste trabalho. Aos meus pais e minhas duas irmãs, pelo amor, carinho, paciência e seus ensinamentos, e pelo constante apoio e confiança em todos os momentos da minha vida, e por não medirem esforços para que eu pudesse levar meus estudos sempre adiante.

À UFPE e a FACEPE (Fundação de Amparo à Ciência do Estado de Pernambuco), agradeço por me proporcionar a oportunidade de realizar esta pesquisa e pelo apoio financeiro. Em especial, quero agradecer ao meu orientador Prof. Bernardo Horowitz, que, há quatro anos, sempre desempenhou um papel fundamental no meu desenvolvimento acadêmico e profissional. Aos demais professores: Prof.ª. Silvana Bastos, Prof. Leonardo Guimarães, Prof. Igor Gomes, Prof. Darlan Karlo e Prof. Antônio Barbosa, sou grato pelos ensinamentos oferecidos durante o curso das suas respectivas disciplinas. Agradeço também aos professores Leonardo Oliveira e Liliane Fonseca, que, desde a graduação, me ofereceram apoio ao meu desenvolvimento no meio acadêmico.

Agradeço ao Prof. Jonathan Teixeira pela disposição e pelos ensinamentos relativos ao MATLAB *Reservoir Simulation Toolbox* (MRST), essencial para o desenvolvimento deste trabalho; ao Dr. Juan Alberto Rojas, que sempre se mostrou disponível para tirar dúvidas e transmitir seus conhecimentos; e também ao Prof. Eduardo Gildin e ao Eng. Marcelo Dall'Aqua, ambos da Universidade Texas A&M, pelo trabalho conjunto que levou à publicação do artigo "*Nonlinear State Constraints Handling in Waterflooding Optimization Through Reduced Order Models"*, no XVII *European Conference on the Mathematics of Oil Recovery* (ECMOR). Por fim, agradeço também aos meus colegas pelos bons momentos de descontração no laboratório e pelo companheirismo nas disciplinas que cursamos juntos.

RESUMO

O gerenciamento ótimo de reservatórios de petróleo requer a formulação de um problema de otimização, onde as variáveis de projeto são os controles dos poços injetores e produtores. Devido ao elevado esforço computacional envolvido na avaliação da função objetivo e no tratamento das restrições impostas durante o processo de otimização, torna-se necessário a utilização de modelos substitutos de ordem reduzida. Sendo assim, duas estratégias são comparadas aqui: *Trajectory Piecewise Linearization* em combinação com *Proper Orthogonal Decomposition* (TPWL/POD) e *Dynamic Mode Decomposition* (DMD). Ambos os métodos são linearizações baseadas em *snapshots*, mas são implementados de formas distintas. As técnicas TPWL/POD reduzem a complexidade do problema linearizando as equações governantes em torno dos estados convergidos e armazenados durante uma simulação de treinamento, e a redução de dimensionalidade é obtida projetando estados em subespaços menores via POD, obtendo um ganho de tempo considerável com sua utilização. Este método requer acesso ao código do simulador e, portanto, é um método intrusivo. Para isso, utilizou-se o MATLAB *Reservoir Simulation Toolbox* (MRST), um *software* gratuito e de código aberto. No MRST, foram executadas as simulações, e algumas de suas rotinas foram modificadas a fim de possibilitar o armazenamento dos estados e das matrizes de derivadas, obtidas através da técnica de Derivação Automática (AD), em conjunto ao paradigma de Programação Orientada a Objetos (POO). O DMD, por sua vez, não requer acesso ao código do simulador e, portanto, não é intrusivo, sendo um método puramente orientado a dados, que adota uma perspectiva livre de equações. Esses dados permitem a extração de uma estrutura dinâmica coerente do problema por meio da suposição de que existe um mapeamento linear conectando a evolução temporal do sistema, que pode ser calculada sem executar novas simulações. Ambas foram comparadas através de análises quanto a acurácia e estabilidade do processo, utilizando pressão e saturação de água como variáveis primárias. Buscou-se a aplicação desses métodos em processos de otimização, onde a função objetivo neste trabalho é a maximização da produção de

óleo sujeita a restrições de estado não-lineares relativas às vazões de produção de líquido e de água no campo, em virtude das limitações operacionais comuns nos campos de produção, comprometendo a eficiência do sistema separador óleo-água diante de grandes produções de líquido.

Palavras-Chave: otimização; simulação de reservatórios; método de ordem reduzida; MRST; derivação automática; programação orientada a objetos.

ABSTRACT

Optimal management of oil reservoirs requires the formulation of an optimization problem, where the design variables are the controls of the injector and producer wells. Due to the high computational effort involved in the evaluation of the objective function and in the treatment of restrictions imposed during the optimization process, it is necessary to use reduced order substitute models. Thus, two strategies are compared here: Trajectory Piecewise Linearization in combination with Proper Orthogonal Decomposition (TPWL/POD) and Dynamic Mode Decomposition (DMD). Both methods are snapshot-based linearization, but they are implemented in different ways. The TPWL/POD techniques reduce the complexity of the problem by linearizing the governing equations around the converged and stored states during a training simulation, and the dimensionality reduction is obtained by projecting states in smaller subspaces via POD, obtaining a considerable time saving with its use. This method requires access to simulator code and is therefore an intrusive method. For this, we used the MATLAB Reservoir Simulation Toolbox (MRST), a free and open-source software. In the MRST, simulations were performed, and some of its routines were modified in order to allow the storage of states and matrices of derivatives, obtained through the Automatic Derivation (AD) technique, together with the Object-Oriented Programming paradigm (POO). DMD, in turn, does not require access to the simulator's code and, therefore, is not intrusive, being a purely data-oriented method that adopts an equation-free perspective. These data allow the extraction of a coherent dynamic structure of the problem through the assumption that there is a linear mapping connecting the temporal evolution of the system, which can be calculated without running new simulations. Both were compared through analysis for accuracy and stability of the process, using pressure and water saturation as primary variables. The application of these methods in optimization processes was sought, where the objective function in this work is the maximization of oil production subject to non-linear state restrictions related to liquid and water production flows in the field, due to limitations

common operations in production fields, compromising the efficiency of the oil-water separator system in the face of large liquid productions.

Keywords: optimization; reservoir simulation; reduced order method; MRST; automatic derivation; object-oriented programming.

SUMÁRIO

1 INTRODUÇÃO E REVISÃO BIBLIOGRÁFICA

A simulação de reservatórios tem papel preponderante nas tomadas de decisão sobre o gerenciamento de campos de petróleo. Usada para o estudo do fluxo dinâmico em subsuperfície e dos processos de recuperação de hidrocarbonetos, ela ainda prevê o comportamento do reservatório quando submetido a diferentes condições de operação. O aumento da capacidade de processamento dos computadores e de recursos praticamente ilimitados de armazenamento e transferência de dados têm permitido a simulação de modelos de reservatórios cada vez mais detalhados, com maior número de células e malhas mais complexas, e, portanto, sistemas de equações maiores e mais complexos. A natureza iterativa dos principais *solvers* utilizados na solução desses sistemas geram assim grande esforço computacional.

Trabalhos que envolvem a otimização dos controles de poços, por exemplo, requerem que um número significativo de simulações seja feito. Em algoritmos de otimização que utilizam gradientes obtidos por diferenças finitas, esse número pode chegar na casa de centenas, sendo ainda maior em algoritmos evolucionários, chegando à casa dos milhares. O esforço computacional requerido pode ser substancial em apenas uma única simulação de um determinado modelo. Sendo assim, tem se disseminado o uso de modelos simplificados (de baixa fidelidade), que aproximem o modelo original de reservatórios (de alta fidelidade) com boa acurácia e que apresentem redução no tempo de simulação.

Atendendo tal demanda, este trabalho tem como objetivo a utilização de dois Métodos de Ordem Reduzida (*Reduced Order Methods*, ROM) a fim de mitigar o esforço computacional do processo de simulação de reservatórios. Embora o modelo substituto possua um custo para seu treinamento, ele apresenta ganho de tempo substancial e boa acurácia, podendo ser retreinado quando for necessário. O ROM deve, portanto, ter uma dimensão que envolve um número mínimo de parâmetros necessários que expliquem a dinâmica do sistema.

O primeiro método a ser analisado neste estudo consiste num método híbrido, isto é, resultante da combinação de duas técnicas: *Trajectory Piecewise Linearization* (TPWL), que representa a física do problema e que elimina a necessidade de se

resolver o sistema não-linear de equações de forma iterativa, pois é feita a linearização das equações de fluxo em torno de estados previamente convergidos e armazenados durante uma simulação de treinamento, resultando, portanto, numa redução da complexidade numérica do problema; e o *Proper Orthogonal Decomposition* (POD), também conhecido em outras áreas de estudo como *Principal Component Analysis* (PCA), que, por sua vez, é responsável pela redução da dimensão do problema através da projeção dos estados, isto é, os mapas de pressões e saturações obtidos a cada passo de tempo durante a simulação de treinamento, em um subespaço gerado a partir das direções principais, ou seja, direções de maior variabilidade de um conjunto de dados, baseando-se na existência de correlação entre os dados analisados. Sendo assim, o número de equações a ser resolvido é reduzido. Tal projeção dos estados pode ser feita através da projeção de Bubnov-Galerkin ou a de Petrov-Galerkin, cuja análise de estabilidade será feita neste trabalho.

No caso do TPWL/POD, além de serem armazenados os estados, também é necessário o armazenamento de matrizes de derivadas presentes na equação do TPWL. Visto que essas matrizes não são obtidas diretamente em simuladores comerciais, utilizou-se a plataforma MATLAB *Reservoir Simulation Toolbox* (MRST), um *software* de código aberto desenvolvido pelo grupo SINTEF Digital, publicado *on-line* de forma gratuita para modelagem e simulação de reservatórios sob a Licença Pública Geral (GNU), desde 2009. O MRST não é essencialmente um simulador, mas consiste numa caixa de ferramentas para prototipagem rápida de modelos e demonstração de novos métodos de simulação e conceitos de modelagem. Diferentes módulos fornecem *solvers* diferentes para fluidos incompressíveis ou compressíveis com várias possibilidades de discretização (*Two-Point Flux Approximation* (TPFA), *Multi-Point Flux Approximation* (MPFA), *mimetic* e outros), permitindo o uso de técnicas de transferência de escala (*upscaling*) ou modelos especiais, em problemas geomecânicos ou com presença de fraturas.

Por ser de código aberto, foi possível a modificação de seu código-fonte para a exportação das matrizes de derivadas, que, por sua vez, são obtidas via Derivação Automática (*Automatic Differentiation*, AD), técnica presente no MRST. Na Derivação Automática, a ideia principal é se obter as quantidades e suas derivadas

simultaneamente, ou seja, toda vez que uma operação é aplicada a uma quantidade, a operação diferencial correspondente é aplicada a sua derivada (Lie K.-A. , 2016). Assim, a ferramenta AD permite que a equação de pressão não-linear, usando formulação totalmente implícita, calcule automaticamente a matriz Jacobiana exata para as equações de óleo e água, utilizando operadores discretos. Deste modo, o uso do MRST permite o acesso direto à matriz Jacobiana e, portanto, a implementação de técnicas de modelagem de ordem reduzida para modelos complexos de fluxo em subsuperfície.

O método TPWL/POD, portanto, consiste numa abordagem intrusiva, devido à necessidade de alteração do código original do *software* de simulação escolhido para a obtenção das derivadas. Como será demonstrado ao longo deste trabalho, a alteração do código-fonte do MRST exigiu pleno entendimento da estrutura de seu código-fonte, que, por sua vez, utiliza o paradigma da Programação Orientada a Objetos (POO), cujos conceitos serão abordados mais adiante.

Assim como a maioria dos simuladores comerciais baseados em uma formulação *Black-Oil*, o MRST fornece um simulador totalmente implícito, que, combinado à técnica AD, oferece estabilidade incondicional para uma ampla gama de regimes de fluxo e heterogeneidades de reservatório. Usando rotinas numéricas e vetorização do MATLAB combinadas com operadores discretos diferenciais do MRST, as equações do modelo podem ser implementadas de forma muito compacta e muito próximas à formulação matemática. O MATLAB é uma ferramenta que se mostra bastante eficiente se comparado a demais linguagens compiladas. Contudo, testes em modelos bifásicos e trifásicos da ordem de dez a cem mil células mostram que os simuladores MRST baseados na Diferenciação Automática são entre duas e dez vezes mais lentos que simuladores comerciais totalmente otimizados (Bao et al., 2017).

Por sua vez, o gerenciamento da injeção de água em reservatórios de petróleo pode ser formulado como um problema de otimização, onde as variáveis de projeto são os controles dos poços injetores e produtores. Esses controles podem ser do tipo vazão, pressões do fundo do poço (BHP) ou ainda aberturas da válvula. A função objetivo pode ser o Valor Presente Líquido (VPL) ou a produção acumulada de óleo, por exemplo. As restrições podem envolver limites nos próprios controles, funções

lineares que envolvem as variáveis de projeto, mas muitas vezes restrições não-lineares, que envolvem variáveisde estado, também devem ser impostas. Como exemplo, tem-se o BHP mínimo nos poços produtores ou BHP máximo nos poços injetores, sujeitos a controles de vazão ou vice-versa. Devido à capacidade do separador no campo, é frequentemente necessário limitar a produção líquida no campo, o que consiste, portanto, numa restrição não-linear que, a princípio, deve ser imposta em todos os passos de tempo. Restrições não-lineares impostas a variáveis de estado são de interesse prático na otimização do desempenho da produção de reservatórios, mas são difíceis de lidar numericamente. A aplicação dessas restrições envolve o cálculo repetido das variáveis de estado e, possivelmente, de suas derivadas, não apenas no final dos ciclos de controle, mas também em vários instantes intermediários. Ambos os cálculos requerem esforço computacional e, portanto, propõe-se o uso do método de ordem reduzida TPWL/POD para diminuir esse esforço numérico.

Por fim, o segundo método de ordem reduzida aqui apresentado reside no *Dynamic Mode Decomposition* (DMD), uma técnica de decomposição de matrizes altamente versátil e baseia-se no poder da Decomposição em Valores Singulares (*Singular Value Decomposition*, SVD). O DMD corresponde a um dos vários métodos existentes que são puramente orientados a dados (*Data-Driven Methods*), onde estruturas dominantes de menor dimensionalidade são extraídas, podendo ser aplicada a uma variedade de disciplinas. O método adota uma perspectiva livre de equações, diferentemente do TPWL/POD, sendo capaz de fornecer uma decomposição precisa de um sistema complexo em estruturas espaço-temporais coerentes que podem ser utilizadas para a realização de previsões e controles futuros em curto prazo.

A descoberta de métodos puramente orientados a dados atualmente está revolucionando a maneira como sistemas complexos são modelados e utilizados para previsão e controle. Seu estudo se dá na área de Análise de Dados, também conhecida por *Data Analytics*, na qual dados passados e atuais são utilizados a fim de investigar a causa de um problema e fazer previsões de cenários prováveis que podem acontecer, sendo frequentemente usada de forma intercambiável com outros termos, como mineração de dados, aprendizado estatístico, aprendizado de máquina (*machine learning*), inteligência artificial etc. (Sankaran, 2019)

O *Dynamic Mode Decomposition with Control* (DMDc), por sua vez, consiste numa extensão do método DMD que incorpora o efeito de controles, abordando todas as vantagens do DMD original e fornece inovação adicional de ser capaz de isolar a dinâmica subjacente dos efeitos de atuação externa, resultando em modelos precisos de entrada e saída. Por ser uma técnica puramente orientada a dados, não há necessidade de modificações no código-fonte do simulador, e, portanto, este método apresenta uma abordagem não-intrusiva, ao contrário do TPWL/POD. Para se demonstrar a eficácia deste método em engenharia de reservatórios, foi proposto um modelo sintético de reservatório, do qual dados foram gerados via simulação, utilizando novamente o MRST, e, a partir deles, extrair padrões recorrentes e dominantes, de menor dimensão, a serem utilizados para a construção de um modelo linear que melhor se ajusta a esse conjunto de dados. Além da recuperação dos estados salvos e utilizados para a construção do modelo linear, demonstra-se que, dentro de certos limites, também é possível fazer previsões quanto a estados futuros do sistema, para qualquer instante de tempo.

1.1 Revisão Bibliográfica

O uso de métodos de ordem reduzida em simulação de reservatórios é relativamente recente, onde um dos principais problemas de interesse é, por exemplo, a otimização de controles de poços, visto que a estratégia mais convencional e tradicionalmente mais utilizada na recuperação de hidrocarbonetos nos campos de petróleo do Brasil e do mundo é a injeção de água.

A abordagem utilizada na construção do modelo TPWL/POD foi inicialmente aplicada ao problema de simulação de reservatórios em (Cardoso, 2009) e aperfeiçoado em (He, 2010). Também foi utilizada por (Fragoso, 2014), que aplicou ROM em otimização multifidelidade. Problemas quanto à estabilidade do POD são reportados em (He, 2010) e (Gildin et al., 2013), sendo resolvidos ao trocar-se a projeção de Bubnov-Galerkin pela projeção de Petrov-Galerkin, conforme relatado em (Carlberg et al., 2009) e (He & Durlosfky, 2013).

Os desenvolvimentos do modelo TPWL/POD em (Cardoso, 2009) e (He, 2010) foram aplicados ao problema de simulação bifásica óleo-água com formulação baseada em saturações de água, a mesma utilizada neste trabalho, enquanto que, em (Fragoso, 2014), aplicou-se a técnica de redução a uma formulação baseada em frações mássicas.

A implementação de discretizações de malhas complexas utilizando a ferramenta MRST é apresentada por (Lie et al., 2011), enquanto que o uso da técnica de Derivação Automática é demonstrada em (Krogstad et al., 2015), aplicando-a em modelos *Black-Oil* e validando seus resultados em *benchmarks* do SPE *Comparative Solution Projects*.

Por adotar uma perspectiva livre de equações, a técnica DMD é capaz de fornecer uma decomposição precisa de um sistema complexo em estruturas espaço-temporais coerentes que podem ser utilizadas para a realização de previsões e controles futuros em curto prazo, podendo ser aplicada a uma variedade de disciplinas, para previsão e estimativa de estados e controles de sistemas complexos, como turbulência, neurologia, clima, epidemiologia, finanças, robótica e automação. Maiores detalhes sobre sua implementação e exemplos de aplicação são encontrados em (Kutz et al., 2016) e (Kutz J. , 2013).

1.2 Objetivos do Trabalho

Esta dissertação não se propõe ao desenvolvimento de uma nova técnica para concepção de modelos substitutos, mas para comparar duas abordagens, uma intrusiva e outra não-intrusiva, na construção de um modelo de ordem reduzida a fim de aplicá-lo em simulação de reservatórios. Como visto anteriormente, a abordagem intrusiva consiste na combinação de duas técnicas bastante conhecidas na literatura: o TPWL e o POD, como redutores de complexidade numérica e de dimensão, respectivamente, ao passo que a abordagem não-intrusiva reside no método DMD, puramente baseado na análise de dados para estimativa e previsão de estados futuros.

Portanto, são considerados objetivos específicos deste trabalho:

- Estudar a formulação de fluxo em subsuperfície e o código-fonte do MRST, desenvolvido pelo grupo SINTEF Digital, a fim de utilizá-lo em algoritmos intrusivos para obtenção dos mapas de estados e suas derivadas, usando, para este fim, a técnica de Derivação Automática, em conjunto ao paradigma da Orientação a objetos, utilizado na implementação do MRST;
- Comparar resultados obtidos em simulações no MRST com os já obtidos em simuladores comerciais, como o IMEX, a fim de validar o uso desta caixa de ferramentas para fins de simulação de reservatórios;
- Estudar em detalhes a técnica e desenvolver um código para o método de ordem reduzida TPWL/POD, que possa ser acoplado ao MRST e em algoritmos de otimização;
- Aplicar o modelo de ordem reduzida TPWL/POD em processos de otimização, impondo restrições de estado não-lineares, a fim de mitigar o esforço computacional requerido durante o cálculo das variáveis de estado e de suas derivadas;
- Comparar e avaliar a utilização da projeção de Petrov-Galerkin em substituição da projeção de Bubnov-Galerkin, durante as simulações do modelo TPWL/POD;
- Estudar o método DMD e sua versão estendida (DMDc), que leva em consideração efeitos de atuação externa (controles), e desenvolver um código a fim de aplicar esta técnica em simulação de reservatórios;
- Comparar e avaliar resultados obtidos utilizando ambos os métodos de ordem reduzida apresentados.

1.3 Organização da Dissertação

O principal objetivo desta dissertação é desenvolver e aplicar métodos de ordem reduzida para problemas de fluxo em subsuperfície. Apesar da aplicação ser em simulação de reservatórios, o estudo aqui proposto pode ser aplicado em outras subáreas da engenharia de reservatórios, tais como gerenciamento de aquíferos ou

ainda em sequestro de dióxido de carbono, ou ainda em outras disciplinas, tais como neurologia e epidemiologia.

No Capítulo 2, são descritos conceitos de simulação de reservatórios, na qual são apresentadas as equações diferenciais que regem o fluxo multifásico em meio poroso e sua discretização por diferenças finitas, incluindo a erudição quanto às propriedades dos fluidos e da rocha. Além disso, também é apresentado o modelo de poço utilizado, para o cálculo de vazões nos poços.

O Capítulo 3 detalha a estrutura geral do MRST, *software* escolhido para a realização de simulações e para o armazenamento dos dados a serem aplicados nos métodos de ordem reduzida TPWL/POD e DMD, e também são apresentados conceitos de Orientação a Objetos e Derivação Automática, ferramentas necessárias para o pleno entendimento do código-fonte do MRST. Por fim, é feita uma comparação dos resultados obtidos pelo MRST frente aos obtidos no IMEX, simulador comercial escolhido, para fins de validação do uso da caixa de ferramentas em simulação de modelos de reservatórios.

No Capítulo 4, é apresentado o primeiro método de ordem reduzida a ser aplicado neste trabalho, que consiste na combinação das técnicas de redução da complexidade numérica e de redução da dimensionalidade do problema, ou seja, as técnicas TPWL e POD, respectivamente. É detalhado como se deu a exportação dos mapas de estados e matrizes de derivadas a serem aplicadas na equação do TPWL. Também são ilustrados os resultados obtidos em três casos de simulação de um modelo de reservatório, com diferentes pressões de fundo (BHP) aplicadas aos poços, utilizando os estados e matrizes armazenados durante uma única simulação de treinamento. Foram comparados os erros relativos médios e o ganho de tempo computacional (*speed-up's*) em relação ao modelo de alta fidelidade, e também comparativos quanto à utilização da projeção de Petrov-Galerkin frente a projeção de Bubnov-Galerkin.

No Capítulo 5, é formulado um problema de otimização cuja função objetivo é a produção acumulada de óleo, sendo impostas restrições de estado não-lineares. Também é descrito o algoritmo de Programação Quadrática Sucessiva (*Sequential Quadratic Programming,* SQP), a ser utilizado durante a otimização. Em seguida, o

modelo de ordem reduzida TPWL/POD é testado e aplicado a um modelo de reservatório sintético, cujos controles, definidos como as variáveis de projeto, são as pressões de fundo dos poços (*Bottom Hole Pressure*, BHP).

O Capítulo 6 apresenta o segundo método de ordem reduzida a ser aplicado neste trabalho, o DMD e sua versão estendida, o DMDc, que leva em consideração o efeito de forças externas, que, para o caso de simulação de reservatórios, corresponde aos controles dos poços. É demonstrado sua eficácia e suas limitações ao longo de exemplos ilustrados ao longo desse capítulo.

Por fim, no Capítulo 7, diante dos resultados obtidos para ambos os métodos, são apresentadas as conclusões e as sugestões de trabalhos futuros.

2 CONCEITOS DE SIMULAÇÃO DE RESERVATÓRIOS

O crescente aumento da demanda global por hidrocarbonetos levou à necessidade de se realizar estudos sobre regiões cada vez mais complexas de serem exploradas. Somado a isso, há necessidade de modelos cada vez mais detalhados, motivando o desenvolvimento de simuladores de reservatório mais eficientes e de maior acurácia, além de computadores de maior capacidade de processamento.

O objetivo principal da simulação de reservatórios é obter, através das equações de fluxo, as vazões de fluidos que devem sair de cada poço, dados os controles de injeção e produção aplicados ao longo do tempo de concessão do campo. Para realizar previsões quanto à produção de hidrocarbonetos e estimativas de reserva, foram desenvolvidos modelos físicos que simulam a ocorrência de fluxo multifásico em meios porosos encontrados em subsuperfície. O modelo mais geral é composto por três fluidos distintos: água, óleo e gás. Contudo, a composição destas três fases pode se tornar bastante complexa.

Neste capítulo será abordado o modelo matemático para fluxo bifásico em meio poroso utilizado na simulação de reservatórios de petróleo, considerando o fluxo de apenas duas fases (aquosa e oleosa) e dois pseudocomponentes (água e óleo), conforme modelo *Black-Oil*. Soma-se a tais considerações, por simplicidade, a razão de solubilidade de gás como sendo igual a zero.

Inicialmente, o capítulo tratará da formulação matemática do problema, deduzindo brevemente na primeira seção as equações diferenciais de fluxo para o modelo bifásico óleo-água. Na segunda seção, descreve-se a discretização em diferenças finitas do sistema de equações diferenciais, utilizado para a solução numérica realizada internamente no simulador. Por fim, a terceira seção apresenta o modelo de poço utilizado neste trabalho e que se encontra implementado no MRST.

2.1 Equações de Fluxo

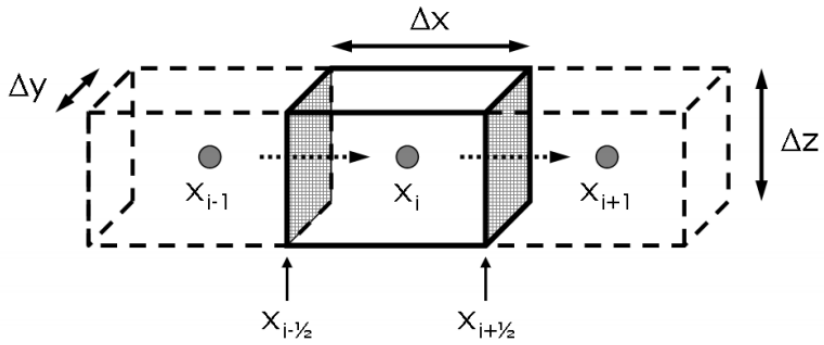
O objetivo de um simulador de reservatórios é resolver numericamente um sistema de equações diferenciais que modela o fluxo em um meio poroso. De modo

geral, no modelo *Black-Oil*, há três fases – aquosa, oleosa e gasosa – e três pseudocomponentes – água, óleo e gás, podendo este último estar livre ou dissolvido na fase oleosa. Neste trabalho, só serão considerados duas fases: aquosa e oleosa, subtendendo-se que os pseudocomponentes envolvidos sejam apenas a água e o óleo.

A seguir serão deduzidas as equações de fluxo para o modelo bifásico, sendo possível a sua extensão para o caso trifásico, para o qual basta adicionar mais uma equação de balanço, referente à fase gás, e levar em consideração a solubilidade do componente gás na fase óleo. Por simplicidade, será considerado, para fins de dedução, o fluxo unidirecional de apenas um fluido, na direção do eixo x, fazendo-se logo em seguida a extensão para o caso tridimensional e bifásico.

Seja um volume controle x_i e seus vizinhos como mostrado na Figura 1.

Figura 1 – Volume de Controle e vizinhança.



Fonte: (Cardoso, 2009)

Fazendo-se o balanço de massa no interior desse volume de controle, tem-se a Equação (2.1), onde M_{in} é a massa que entra no volume de controle, M_{out} é a massa que sai do volume de controle, M_w é a massa injetada/produzida a partir de um poço completado no volume de controle e M_{acc} é a massa acumulada no volume de controle.

$$M_{in} - M_{out} + M_w = M_{acc} \tag{2.1}$$

A vazão mássica que entra ou sai do volume de controle é calculada pelo produto da velocidade aparente do fluido v_x por sua densidade ρ e pela área da interface $A_x = \Delta y \cdot \Delta z$, expressa pela Equação (2.2). A massa total M é dada pela integral de $\dot{m}$ com o tempo.

$$\dot{m} = v_x \cdot \rho \cdot A_x \tag{2.2}$$

A massa acumulada é, por sua vez, obtida pela Equação (2.3), calculada pela diferença entre a massa contida no interior do volume de controle no instante $t + \Delta t$ e a

contida no instante anterior t, função do volume poroso dado pelo produto entre a porosidade ϕ e o volume de controle $V_c = \Delta x \cdot \Delta y \cdot \Delta z$.

$$M_{acc} = (\rho \cdot \phi \cdot V_c)_{t+\Delta t} - (\rho \cdot \phi \cdot V_c)_t \tag{2.3}$$

Substitui-se então as Equações (2.1) e (2.2) na Equação (2.3), resultando na Equação (2.4), que ao ser dividida por $V_c \cdot \Delta t$, se obtém a Equação (2.5).

$$(A_x \cdot \rho \cdot v_{x,i-\frac{1}{2}} - A_x \cdot \rho \cdot v_{x,i+\frac{1}{2}} + \dot{m}_w) \cdot \Delta t = (\rho \cdot \phi \cdot V_c)_{t+\Delta t} - (\rho \cdot \phi \cdot V_c)_t \tag{2.4}$$

$$\frac{\rho \cdot v_{x,i-\frac{1}{2}} - \rho \cdot v_{x,i+\frac{1}{2}}}{\Delta x} + \frac{\dot{m}_w}{V_c} = \frac{(\rho \cdot \phi)_{t+\Delta t} - (\rho \cdot \phi)_t}{\Delta t} \tag{2.5}$$

Admitindo que o volume de controle e o intervalo de tempo considerados sejam infinitesimais para uma aproximação contínua, pode-se reescrever a Equação (2.5) em termos de equação diferencial dada pela Equação (2.6).

$$\frac{\partial(\rho \cdot v_x)}{\partial x} + \frac{\dot{m}_w}{V_c} = \frac{\partial(\rho \cdot \phi)}{\partial t} \tag{2.6}$$

A velocidade aparente v_x é dada pela lei de Darcy, função do potencial hidráulico Φ, sendo expressa pela Equação (2.7), onde k é a permeabilidade intrínseca (ou absoluta) da rocha ao fluido e μ é a viscosidade do fluido. Para o caso bifásico, o termo de permeabilidade é dado pelo produto da permeabilidade absoluta k pela relativa $k_{r\alpha}$, isto é, a permeabilidade efetiva do fluido k_α.

$$v_x = \frac{k}{\mu}\frac{\partial \Phi}{\partial x} \tag{2.7}$$

Substituindo a Equação (2.7) na equação de balanço dada pela Equação (2.6), obtém-se na Equação (2.8).

$$\frac{\partial}{\partial x}\left(\rho \cdot \frac{k}{\mu}\frac{\partial \Phi}{\partial x}\right) + \frac{\dot{m}_w}{V_c} = \frac{\partial(\rho \cdot \phi)}{\partial t} \tag{2.8}$$

A Equação (2.8), por sua vez, pode ser estendida para três dimensões ao serem somadas as contribuições das três direções, utilizando assim o operador divergente para a velocidade e o operador gradiente para o gradiente hidráulico, respectivamente, conforme expresso na Equação (2.9).

$$\nabla \cdot (\lambda \cdot \nabla \Phi) + \tilde{q_j^w} = \frac{\partial(\rho \cdot \phi)}{\partial t} \tag{2.9}$$

Considerando agora a contribuição de cada componente do fluido, para o caso bifásico, o modelo deve possuir duas equações de balanço (uma para cada fase), e assim quatro incógnitas (p_o, p_w, S_o, S_w). Na formulação aqui considerada, adota-se a pressão de óleo p_o e a saturação de água S_w como variáveis primárias, em relação às quais o problema será resolvido.

A equação diferencial que expressa o fluxo de fluidos em subsuperfície é resultante da combinação da Equação de Conservação da Massa com a versão multifásica da Lei de Darcy. Para o caso bifásico aqui considerado, admite-se que não há transferência de massa entre as fases óleo e água, isto é, o componente óleo reside apenas na fase oleosa e a componente água, na fase aquosa. Portanto, para cada fase j, onde $j = o$ para a fase óleo e $j = w$ para a fase água, tem-se a Equação (2.10).

$$\nabla \cdot \left[\lambda_j \cdot k \cdot \left(\nabla p_j - \rho_j \cdot g \cdot \nabla z\right)\right] + \tilde{q_j^w} = \frac{\partial}{\partial t}\left(\phi \cdot \frac{S_j}{B_j}\right) \tag{2.10}$$

Na Equação (2.10), a mobilidade da fase é dada por $\lambda_j = k_{rj}/(\mu_j B_j)$, onde μ_j é a viscosidade e B_j é o Fator Volume de Formação da fase em questão; p_j é a pressão da fase; ρ_j é a massa específica da fase; g é a aceleração da gravidade; z é a profundidade em relação a uma altura de referência H_{ref}; S_j é a saturação da fase e $\tilde{q_j^w}$ é o termo de fonte/sumidouro, isto é, a vazão mássica injetada/produzida da fase j por unidade de volume.

O fator volume de formação B_j é definido como a razão entre o volume da fase j nas condições de reservatório e o volume da fase nas condições padrão de superfície (*standard conditions*, std). Seja a densidade de referência da fase j igual a $\rho_{j,std}$, têm-se então que $B_j = \rho_{j,std}/\rho_j(p)$. Para fluidos incompressíveis, a densidade do fluido não varia com a pressão e, portanto, $B_j = 1$.

Completa-se a formulação do modelo óleo-água com as Equações (2.11) e (2.12).

$$S_w + S_o = 1 \tag{2.11}$$

$$p_{cow}(S_w) = p_o - p_w \tag{2.12}$$

Neste trabalho, desprezou-se a pressão capilar entre as fases. Logo, $p_o = p_w$. Uma vez obtidos os valores de p_o e de S_w, p_w e S_o podem ser calculados diretamente pelas Equações (2.11) e (2.12).

As saturações são utilizadas na determinação da permeabilidade relativa k_{rj}, através de curvas, expressas em função de tais variáveis, podendo ser fornecidas por meio de tabelas cujo pontos são interpolados, ou ainda por meio do modelo de Corey, dado pelas Equações (2.13) e (2.14) que representam, respectivamente, as permeabilidades relativas do óleo k_{ro} e da água k_{rw}, ambas em função da saturação da água S_w.

$$k_{ro} = {k_{ro}}^0 \left(\frac{1 - S_w - S_{or}}{1 - S_{wc} - S_{or}}\right)^{\alpha} \tag{2.13}$$

$$k_{rw} = {k_{rw}}^0 \left(\frac{S_w - S_{wc}}{1 - S_{wc} - S_{or}}\right)^{b} \tag{2.14}$$

Tais curvas de permeabilidade relativa no modelo de Corey são obtidas em laboratório através dos seguintes parâmetros determinados a partir de ensaios: S_{or} é a saturação de óleo residual que representa a mínima saturação de óleo que se pode obter; S_{wc} é a saturação de água conata que representa a mínima saturação de água que normalmente existe no início antes da injeção de água; ${k_{ro}}^0$ é o ponto terminal para permeabilidade relativa do óleo, medido na saturação S_{wc}; ${k_{rw}}^0$ é o ponto terminal para permeabilidade relativa da água, medido na saturação $1 - S_{or}$; α e b são expoentes de Corey para as curvas de permeabilidade relativa de óleo e água, respectivamente.

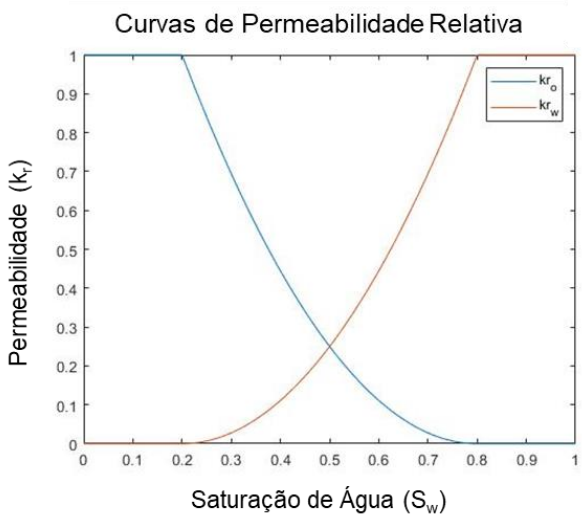
Na Figura 2, é ilustrado um exemplo onde curvas de permeabilidade relativa para água k_{rw} e óleo k_{ro}, ambas funções da saturação de água S_w, são definidas através da entrada dos parâmetros que formulam o Modelo de Corey.

Para cálculo das mobilidades das fases λ_j, é necessário determinar suas densidades e viscosidades em função da pressão. As densidades podem ser calculadas utilizando-se os fatores volume de formação B_j, fornecidos em tabelas PVT conforme a Equação (2.15).

$$\rho_j(p) = \frac{1}{B_j}\rho_{j,std} \tag{2.15}$$

Onde $\rho_{j,std}$ é a densidade da fase em condições-padrão.

Figura 2 – Curvas de permeabilidade relativa para água e óleo, definidas no MRST, segundo Modelo de Corey.



Fonte: O Autor (2019).

As viscosidades μ_j são calculadas pela Equação (2.16), onde $\mu_{j,b}$ é a viscosidade medida na pressão de bolha, $\partial\mu_j/\partial p$ é a viscosibilidade, p_o é a pressão do óleo e p_b é a pressão de bolha.

$$\mu_j = \mu_{j,b} + \frac{\partial \mu_j}{\partial p}(p_o - p_b) \tag{2.16}$$

Por fim, a porosidade ϕ na Equação (2.10) pode ser calculada considerando-se o modelo pouco compressível, admitindo a compressibilidade da rocha constante. Partindo dessa consideração, pode-se calcular a porosidade através da Equação (2.17), onde ϕ_{ref} é a porosidade tomada na pressão de referência, c_r é a compressibilidade da rocha, p_o é a pressão do óleo e p_{ref} é a pressão de referência para o cálculo da porosidade.

$$\phi = \phi_{ref} + c_r\left(p_o - p_{ref}\right) \tag{2.17}$$

2.2 Discretização das Equações de Fluxo

Para a solução numérica do sistema de equações composto pela Equação (2.10) para cada fase, é necessário discretizá-las. Tradicionalmente, para problemas de fluxo, utiliza-se o Método das Diferenças Finitas (MDF), que aproxima as derivadas em relação ao espaço e ao tempo por diferenças divididas.

Quando se utiliza o MDF, é mais comum utilizarem-se malhas estruturadas (Ertekin et al., 2011), nas quais é possível numerar os nós considerando a sua posição em relação aos vizinhos. Para discretização do espaço, utiliza-se uma pequena variação no método, uma vez que, ao discretizar-se a Lei de Darcy, as propriedades são consideradas nas interfaces de cada célula.

Como visto no seção 2.1, o fluxo em subsuperfície é descrito pela Equação (2.10) ao combinar-se a Equação de Conservação da Massa com a versão multifásica da Lei de Darcy. A pressão p_o e a saturação de água S_w foram definidas como as variáveis primárias do problema. Viu-se também que, por simplificação, a pressão capilar é desprezada (logo, $p_o = p_w$). Considerando apenas a direção horizontal x (logo, $\nabla z = 0$) e assumindo que as dimensões das células da malha ($\Delta x, \Delta y, \Delta z$) sejam constantes, a primeira parcela do membro esquerdo da Equação (2.10) representa os termos de fluxo, cuja forma discreta é dada pela Equação (2.18).

$$\frac{\partial}{\partial x}\left[k\lambda_j\left(\frac{\partial p_j}{\partial x}\right)\right] \approx \left\{\left(T_j\right)_{i-1/2}^{n+1}\left[p_{i-1}^{n+1} - p_i^{n+1}\right] + \left(T_j\right)_{i+1/2}^{n+1}\left[p_{i+1}^{n+1} - p_i^{n+1}\right]\right\}\frac{1}{V} \tag{2.18}$$

Onde o subíndice j indica a fase (água ou óleo) e i especifica o índice da célula. O índice $n+1$ expressa o próximo passo de tempo e $V_c = \Delta x \cdot \Delta y \cdot \Delta z$ é o volume da célula i. A transmissibilidade $\left(T_j\right)_{i-1/2}^{n+1}$, dada pela Equação (2.19), relaciona o fluxo da fase j à diferença de pressão entre as células $i-1$ e i.

$$\left(T_j\right)_{i-1/2}^{n+1} = \left(\frac{kA}{\Delta x}\right)_{i-1/2}\left(\frac{k_{rj}}{B_j\mu_j}\right)_{i-1/2}^{n+1} \tag{2.19}$$

Onde $A = \Delta y \cdot \Delta z$ é a área da face comum às células $i-1$ e i, para o fluxo horizontal x. Analogamente, define-se então a transmissibilidade $\left(T_j\right)_{i+1/2}^{n+1}$. Os termos de fluxo introduzem não-linearidade ao sistema, visto que $\left(T_j\right)_{i\pm1/2}^{n+1}$ apresenta termos

que são funções da pressão p_o e da saturação de água S_w, variáveis primárias do problema. Na Equação (2.19), a permeabilidade $k_{i-1/2}$ é calculada como a média harmônica entre k_{i-1} e k_i, e $\left(k_{rj}/B_j\mu_j\right)_{i-1/2}^{n+1}$ depende da direção do fluxo da fase j. A formulação, para este último caso, recebe o nome de *upwind*, isto é, a mobilidade é calculada tomando-se a célula de maior potencial seguindo a direção do fluxo.

A segunda parcela do membro esquerdo da Equação (2.10) refere-se aos termos de fonte/sumidouro, detalhados na seção seguinte. Em simulação de reservatórios, esses termos correspondem aos poços injetores/produtores, ou seja, para o modelo bifásico óleo-água, eles representam a vazão de injeção ou de produção.

O membro direito da Equação (2.10) representa os termos de acumulação de massa, sendo expresso na sua forma discreta pela Equação (2.20).

$$\frac{\partial}{\partial t}\left(\phi\frac{S_j}{B_j}\right) \approx \frac{1}{\Delta t}\left[\left(\phi\frac{S_j}{B_j}\right)^{n+1} - \left(\phi\frac{S_j}{B_j}\right)^{n}\right] \tag{2.20}$$

Onde Δt é o intervalo de tempo. Para sistema incompressíveis, tanto a compressibilidade da rocha quando a do fluido são constantes e desprezíveis, e o Fator Volume de Formação de cada fase $B_j = 1$.

Note que nas Equações (2.19) e (2.20), todos os termos são tomados no passo de tempo $n+1$ com exceção de uma parcela do termo de acumulação, o que caracteriza uma formulação totalmente implícita (*Fully Implicit Discretization)* e demanda a solução de um sistema de equações em cada passo de tempo.

2.3 Modelo de Poço

O objetivo final do estudo de simulação de reservatórios é prever as vazões obtidas em cada poço e/ou pressões de fundo de poço com precisão, e estimar as distribuições de pressão e saturação no reservatório. Em geral, o tratamento de poços nos simuladores apresenta dificuldades que requerem considerações especiais.

Os poços são considerados condições de contorno internas, devendo ser especificadas a fim de se ter um problema bem-posto. Essa condição de contorno interna basicamente pode assumir duas formas: por *vazão especificada* (condição de

Neumann), onde é fixa a vazão total do poço, isto é, a soma da vazão de todas as completações, e, portanto, transforma a pressão em cada completação em variáveis, adicionando-se ao sistema de equações uma nova equação baseada na Equação (2.21) para cada uma das células onde há completação; ou por *pressão especificada* (condição de Dirichlet), onde é fixada a pressão na completação mais profunda p_i^w e calcula, a partir do gradiente hidrostático no interior do poço, a pressão nas completações superiores. Sendo assim, não há necessidade de adição de equações e não há variáveis adicionais. A vazão é então calculada pela Equação (2.21) onde a única variável advinda do sistema de equações de fluxo é a pressão na célula de cada completação. Neste trabalho, os poços foram controlados especificando suas pressões de fundo (*Bottom Hole Pressure*, BHP).

Como visto anteriormente, o membro direito da Equação (2.10) refere-se aos termos de fonte/sumidouro, que correspondem aos poços injetores/produtores. Para o cálculo das vazões de cada poço, é necessário estabelecer uma relação entre o BHP, a pressão da célula onde há completação e a vazão injetada/produzida de cada fase. Além disso, como se admite múltiplas completações, também é necessário relacionar o BHP à pressão no interior do poço em cada uma das completações acima.

O modelo de poço aqui considerado é descrito em (Peaceman, 1983), expresso pela Equação (2.21). Esse modelo baseia-se na premissa de que a pressão calculada para uma célula do poço é igual à pressão em um raio equivalente, r_o (Ertekin et al., 2011). A definição de r_o pode ser usada para relacionar a pressão da célula do poço, p_i^w, com a vazão q_j^w, por meio da pressão da célula do *grid*, p_i^{n+1}.

$$\left(q_j^w\right)_i^{n+1} = \left(\tilde{q_j^w}\right)_i^{n+1} V_c = W_i \left(\lambda_j\right)_i^{n+1} \left(p_i^{n+1} - p_i^w\right) \tag{2.21}$$

Onde $\left(q_j^w\right)_i^{n+1}$ é a vazão volumétrica da fase j da célula i até o poço (ou vice-versa) no passo de tempo $n+1$, p_i^{n+1} é a pressão da célula i no tempo $n+1$, p_i^w é a pressão no poço w na célula i. Na literatura, a diferença $p_i^{n+1} - p_i^w$ é denominada *drawdown*. O termo W_i, por sua vez, representa um fator de proporcionalidade conhecido como Índice de Produtividade (ou Índice do Poço, *Well Index*). Para um poço vertical perfurando a célula i, W_i é dado pela Equação (2.22).

$$W_i = \left[\frac{2\pi k \Delta z}{\ln(r_o/r_w)}\right]_i \tag{2.22}$$

Onde r_w é o raio do poço e $r_o \approx 0{,}2 \cdot \Delta x$. Esta última aproximação refere-se ao modelo de poço de Peaceman, admitindo que a permeabilidade no plano horizontal seja isotrópica, isto é, $k_x = k_y$. Note que, se o poço está sendo controlado por BHP, esse controle é expresso no modelo especificando p_i^w na Equação (2.21).

No caso de poços com múltiplas completações, faz-se necessário calcular, a partir da pressão de fundo de poço, medida a partir de uma altura de referência H_{ref}, as pressões no interior do poço nas demais completações. Por simplificação, somente serão considerados neste trabalho poços verticais, embora o exposto a seguir pode ser aplicado a qualquer tipo de poço. Logo, seja p_{ref}^w a pressão de fundo de poço (primeiro bloco completado de baixo para cima) e p_k^w a pressão no interior da completação na camada k. A pressão no interior do poço nas completações acima pode ser obtida adicionando-se o peso de fluido como definido na Equação (2.23).

$$p_k^w = p_{ref}^w + \sum_k \gamma_{wb}\left(H_k - H_{ref}\right) \tag{2.23}$$

Onde γ_{wb} é a média dos pesos específicos dos fluidos que adentram o poço desde o fundo até a célula considerada. Note que, apesar de a formulação aqui considerada tratar a dependência temporal de forma totalmente implícita, para cálculo da pressão no interior do poço em cada completação, como na Equação (2.23), necessita-se do peso específico médio dos fluidos no interior do poço, utilizando, para este fim, propriedades calculadas no passo de tempo anterior o que, de certa forma, altera o esquema de cálculo totalmente implícito.

No caso de poços produtores, as densidades das fases são ponderadas pelo fluxo fracionário de cada fase no interior do poço. Define-se então o fluxo fracionário como a razão entre a vazão volumétrica de cada fase e a vazão volumétrica total considerando todo o volume que adentrou o poço desde o fundo até a célula considerada. Entretanto, visto que as vazões volumétricas são diretamente proporcionais às mobilidades de cada fase, o fluxo fracionário pode ser matematicamente expresso pela Equação (2.24).

$$f_j^n = \frac{\sum_k {\lambda_j}_k^n}{\sum_k {\lambda_w}_k^n + \sum_k {\lambda_o}_k^n} \tag{2.24}$$

Em modelos discretos, utilizando o sistema Cartesiano, onde os índices i, j, k (que representam as direções x, y e z, respectivamente) seguem uma ordem lógica, o número total de células é dado por N_c. A fim de se definir alguns conceitos a serem utilizados mais adiante, considere agora o vetor de estados $\boldsymbol{x}$ que contém as duas variáveis primárias, representado pela Equação (2.25).

$$\boldsymbol{x} = \left[(\boldsymbol{p_o})_1, (\boldsymbol{p_o})_2, \dots, (\boldsymbol{p_o})_{N_c-1}, (\boldsymbol{p_o})_{N_c}, (\boldsymbol{S_w})_1, (\boldsymbol{S_w})_2, \dots, (\boldsymbol{S_w})_{N_c-1}, (\boldsymbol{S_w})_{N_c}\right]^{\boldsymbol{T}} \tag{2.25}$$

Introduzidas as discretizações apresentadas ao longo das Equações (2.18) a (2.22), a versão discreta da Equação (2.10) na sua formulação totalmente implícita pode ser expressa na forma residual da Equação (2.26).

$$\boldsymbol{g}(\boldsymbol{x}^{n+1}, \boldsymbol{x}^n, \boldsymbol{u}^{n+1}) = \boldsymbol{A}(\boldsymbol{x}^{n+1}, \boldsymbol{x}^n) + \boldsymbol{F}(\boldsymbol{x}^{n+1}) + \boldsymbol{Q}(\boldsymbol{x}^{n+1}, \boldsymbol{u}^{n+1}) = 0 \tag{2.26}$$

Onde $\boldsymbol{g}$ é o vetor de resíduos; $\boldsymbol{F}$ é uma matriz pentadiagonal para malhas bidimensionais, ou heptadiagonal para malhas tridimensionais, $\boldsymbol{A}$ é uma matriz diagonal, e $\boldsymbol{Q}$ representa os termos de fonte/sumidouro. O passo de tempo é descrito pelos índices n e $n+1$. O termo $\boldsymbol{F}(\boldsymbol{x}^{n+1})$ representa os termos de fluxo enquanto que $\boldsymbol{A}(\boldsymbol{x}^{n+1}, \boldsymbol{x}^n)$ representa os termos de acumulação. As matrizes $\boldsymbol{A}$, $\boldsymbol{F}$ e $\boldsymbol{Q}$ dependem de $\boldsymbol{x}$ e devem ser atualizadas a cada iteração de cada passo de tempo. As matrizes convergidas a cada passo de tempo serão as matrizes exportadas para a aplicação na equação do TPWL, como será visto mais adiante.

A Equação (2.26) representa um sistema não-linear de equações que deve ser resolvido aplicando o Método de Newton a fim de que o resíduo $\boldsymbol{g}$ tenda a zero, através da Equação (2.27).

$$\boldsymbol{J}\boldsymbol{\delta} = -\boldsymbol{g} \tag{2.27}$$

Onde $\boldsymbol{J}$ é a matriz Jacobiana do sistema dada por $J_{ij} = \partial g_i / \partial x_j$ e $\delta_i = x_i^{n+1,v+1} - x_i^{n+1,v}$, com $v+1$ e v indicando o nível da iteração.

3 MATLAB RESERVOIR SIMULATION TOOLBOX

O MATLAB *Reservoir Simulation Toolbox* (MRST) é um *software* de código aberto desenvolvido pelo grupo SINTEF Digital, que consiste numa caixa de ferramentas contendo estruturas de dados e rotinas básicas necessárias a fim de configurar, resolver e visualizar modelos de reservatórios de petróleo. A parte básica do MRST contém um conjunto abrangente de estruturas de dados e rotinas para representar e manipular os parâmetros que compõem um modelo de simulação de um meio poroso: *grids* representando a geometria do domínio, propriedades petrofísicas da rocha e outros termos, como gravidade, condições de contorno e modelos de poço. O MRST ainda fornece um simulador totalmente implícito, oferecendo estabilidade incondicional para uma ampla gama de regimes de fluxo e heterogeneidades de reservatório.

Neste capítulo, será abordada uma breve descrição do MRST e um de seus aspectos de grande importância para este trabalho que consiste na técnica de Derivação Automática (AD), que faz do MRST uma ferramenta poderosa para prototipagem rápida e que permite desenvolver códigos compactos e autoexplicativos, adequados para fins pedagógicos. Presente em grande parte das suas rotinas, a ideia principal da técnica de Derivação Automática é obter as grandezas e suas derivadas simultaneamente, permitindo então que a equação de pressão não-linear calcule automaticamente a matriz Jacobiana exata para as equações de óleo e água. Para isso, é preciso antes descrever o paradigma da Programação Orientada a Objetos (POO), que permite dividir a implementação em diferentes contextos numéricos, ocultar detalhes desnecessários e apenas expor os detalhes realmente necessários para cada contexto específico.

Tecnicamente, o MRST combina três recursos principais: uma linguagem altamente vetorizada que permite ao usuário trabalhar com objetos matemáticos de alto nível e que continue desenvolvendo o programa durante sua execução, uma estrutura de malhas flexível que permite a construção simples de operadores diferenciais discretos, e diferenciação automática que garante que nenhuma derivada analítica seja

programada explicitamente desde que as equações de fluxo discretas e as relações constitutivas sejam implementadas como uma sequência de operações algébricas.

No final desse capítulo, ainda é proposto um teste de validação do MRST frente a simuladores comerciais, comparando os resultados obtidos da simulação de um modelo de reservatório com o MRST e com o IMEX, desenvolvido pelo grupo *Computer Modeling Group* (CMG).

3.1 Estrutura Geral

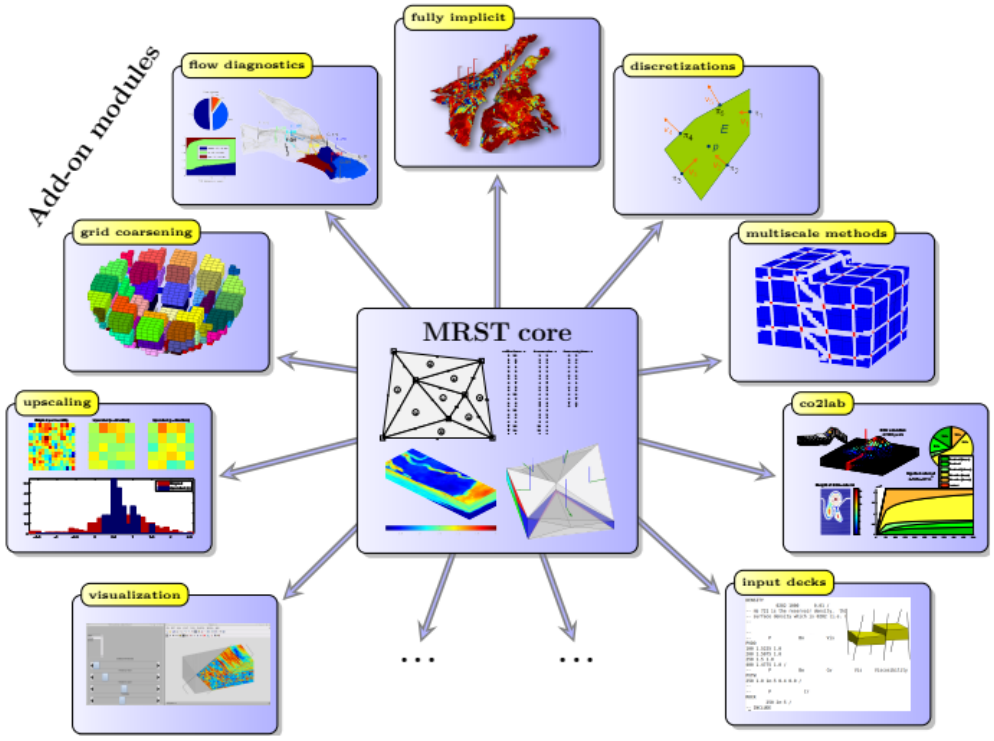
Atualmente, um dos simuladores de código aberto mais conhecidos e amplamente utilizados é o MATLAB *Reservoir Simulation Toolbox* (MRST). O MRST é um *software* de código aberto desenvolvido pelo grupo (SINTEF Digital, 2020), publicado *on-line* de forma gratuita para modelagem e simulação de reservatórios sob a Licença Pública Geral (GNU), desde 2009. O MRST não é essencialmente um simulador, mas consiste numa caixa de ferramentas para prototipagem rápida de modelos e demonstração de novos métodos de simulação e conceitos de modelagem.

O uso de uma linguagem como o MATLAB geralmente introduz um esforço computacional, que pode ser bastante significativo para pequenos sistemas. No entanto, a falta de eficiência computacional é de longe superada por um processo de desenvolvimento mais eficiente, independente da escolha do sistema operacional. Para sistemas maiores, a maior parte do tempo computacional gasto em um simulador bem projetado deve-se ao processamento de números de ponto flutuante, para o qual o MATLAB é bastante eficiente e comparável com linguagens compiladas. Contudo, testes em modelos bifásicos e trifásicos da ordem de dez a cem mil células mostram que os simuladores MRST baseados na diferenciação automática são entre duas e dez vezes mais lentos do que os simuladores comerciais totalmente otimizados (Bao et al., 2017).

A estrutura do MRST basicamente é composta por duas partes, conforme é ilustrado na Figura 3. O núcleo do *software*, denominado MRST-*core*, contém estruturas de dados e rotinas básicas necessárias a fim de configurar, resolver e visualizar

modelos monofásicos, bifásicos ou multifásicos, incompressíveis ou compressíveis, em malhas estruturadas ou não-estruturadas.

Figura 3 – Organização do MRST, que consiste em um módulo principal que fornece estrutura de dados e *solvers* simplificados, e um conjunto de módulos complementares que, por sua vez, oferecem visualizadores e outras ferramentas.



Fonte: (Krogstad et al., 2015).

A segunda parte do *software* consiste em um conjunto de módulos complementares que estendem, complementam e substituem os recursos existentes do núcleo, normalmente na forma de *solvers* específicos. Outros módulos presentes no pacote oferecem funcionalidade conveniente, como leitura e processamento de *decks* de entrada de dados de outros simuladores comerciais, como o ECLIPSE.

Para entender a capacidade de prototipagem rápida de simuladores totalmente implícitos, é preciso entender a funcionalidade básica para geração de malhas e discretizações implementadas pelo MRST. Uma introdução muito mais detalhada pode ser encontrada em (Lie et al., 2011).

Sabe-se que, pelo exposto no capítulo anterior, para descrever matematicamente o fluxo de fluidos em subsuperfície, são necessários dois tipos de modelos: um modelo matemático que descreva o fluxo em um meio poroso, dado como um conjunto de

equações diferenciais parciais que descrevem a conservação da massa das fases fluidas, acompanhado por um conjunto adequado de relações constitutivas; e um modelo geológico que descreva a formação rochosa porosa, isto é, o modelo de reservatório, construído através de uma malha composta por propriedades petrofísicas ou hidrológicas, usadas como dados de entrada para o modelo de fluxo e que, juntos, compõem o modelo de simulação de reservatório. O modelo geológico deve também descrever a geometria da rocha do reservatório e, em particular, os horizontes geológicos do modelo e as principais falhas. Isso requer malhas flexíveis em relação à geometria e topologia (Lie et al., 2011).

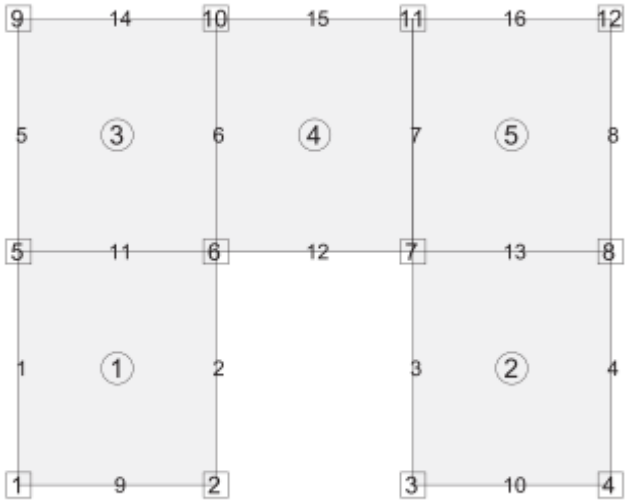
No MRST, seus desenvolvedores optaram por manter a malha e suas propriedades petrofísicas como entidades fundamentais e isoladas, presentes durante toda a simulação. Em particular, a malha (ou *grid*) serve como dado de entrada para quase todos os *solvers* e rotinas de visualização, a fim de garantir a interoperabilidade entre uma ampla variedade de diferentes tipos de malhas e métodos computacionais, mesmo que assim diminua a eficiência do processo de simulação. As malhas então consistem em um conjunto de células poligonais, representadas por três propriedades: células, faces e nós. Cada uma das n_c células corresponde a um conjunto de n_f faces, e cada face a um conjunto de arestas, onde cada uma, por sua vez, corresponde a um par de nós. Para isso, utiliza-se dois tipos de mapeamentos, descritos a seguir.

O primeiro mapeamento é dado por $F: \{1, \dots, n_c\} \rightarrow \{0,1\}^{n_f}$, que mapeia uma célula a um conjunto de faces que a delimita. A estrutura que compõe a malha no MRST é dada por `G`, e o mapeamento descrito anteriormente é representado pelo *array* `G.cells.faces`, onde na primeira coluna estão os índices de cada célula. Esses índices não são armazenados por serem redundantes, mas podem ser obtidos através do comando `gridCellNo(G)`. Na segunda e terceira colunas, são indicadas, respectivamente, as faces que constituem a célula, e as suas direções.

No segundo mapeamento, dada uma face, obtém-se as duas células vizinhas, $N_1, N_2: \{1, \dots, n_f\} \rightarrow \{0, \dots, n_c\}$, onde 0 corresponde ao domínio externo. Em `G`, N_1 é dado por `G.faces.neighbors(:,1)` e N_2 é dado por `G.faces.neighbors(:,2)`. Tanto as células como as faces contêm propriedades geométricas, tais como centroides, volumes, áreas e vetores normais. A Figura 4 ilustra os dois tipos de mapeamento.

A combinação dos mapeamentos, entre as células e as faces, descritos acima, conceitualmente pode parecer mais complicado do que escrever laços de repetição do tipo *for*. No entanto, o esforço extra necessário para entender essas abstrações é de longe superado pela vantagem de poder abranger vários tipos e formatos de malha.

Figura 4 – Ilustração dos campos `cells` e `faces` da malha: os índices das células estão marcados por círculos, os dos nós, por quadrados, e as faces não possuem marcação.




```
cells.faces =                 faces.nodes =       faces.neighbors =
 1    1    1 (West)              1    1              0    1
 1    9    3 (South)             1    5              1    0
 1    2    2 (East)              2    2              0    2
 1   11    4 (North)             2    6              2    0
 2    3    1 (West)              3    3              0    3
 2   10    3 (South)             3    7              3    4
 2    4    2 (East)              4    4              4    5
 2   13    4 (North)             4    8              5    0
 3    5    1 (West)              5    5              0    1
 3   11    3 (South)             5    9              0    2
 3    6    2 (East)              6    6              1    3
 3   14    4 (North)             6   10              0    4
 4    6    1 (West)              7    7              2    5
 4   12    3 (South)             7   11              3    0
 4    7    2 (East)              8    8              4    0
 4   15    4 (North)             8   12              5    0
 5    7    1 (West)              9    2
 5   13    3 (South)             9    1
 5    8    2 (East)              :    :
 5   16    4 (North)             :    :
```

Fonte: (Lie K.-A. , 2016).

Considere agora o problema de fluxo monofásico incompressível, descrito pela Equação (3.1), a ser discretizado através de uma aproximação de volumes finitos por dois pontos (*Two-Point Finite-volume Flux Approximation*, TPFA), onde $\vec{v}$ é a velocidade de Darcy, k é a permeabilidade absoluta, p é a pressão do fluido e q é o termo de fonte/sumidouro. Para fins de simplificação, assume-se que k seja isotrópico e que não haja fluxo no contorno do domínio.

$$\nabla \cdot \vec{v} = q, \qquad \vec{v} = -k \cdot \nabla p \tag{3.1}$$

Os operadores discretos `div` e `grad` para os operadores divergente e gradiente, respectivamente, permitem escrever as equações discretas de maneira semelhante às equações contínuas, sendo representados por matrizes esparsas no MATLAB. Uma matriz é dita esparsa quando possui uma grande quantidade de elementos de valor igual a zero (ou não presentes, ou não necessários). A matriz esparsa é implementada através de um conjunto de listas ligadas que apontam para elementos diferentes de

zero, de forma que os elementos que possuem valor zero não são armazenados e, portanto, não ocupam memória em disco.

O operador `div` é um mapeamento linear de faces para células. Seja $\boldsymbol{v} \in \mathbb{R}^{n_f}$ o fluxo discreto e $\boldsymbol{v}[f]$ restrito a face f com orientação de $N_1(f)$ a $N_2(f)$. O divergente do fluxo restrito à célula c é dado como descrito na Equação (3.2).

$$div(\boldsymbol{v})[c] = \sum_{f \in F(c)} v[f]\mathbf{1}_{\{c=N_1(f)\}} - \sum_{f \in F(c)} v[f]\mathbf{1}_{\{c=N_2(f)\}} \tag{3.2}$$

Onde $\mathbf{1}_{\{expr\}}$ é igual a “um” quando $expr$ for verdadeiro e “zero”, caso contrário. Da mesma forma, o operador `grad` mapeia pares de células a faces. Logo, se $\boldsymbol{p}$ denota um vetor de pressões de uma célula, o gradiente da pressão restrito a face f é definido pela Equação (3.28), para todo $\boldsymbol{p} \in \mathbb{R}^{n_c}$.

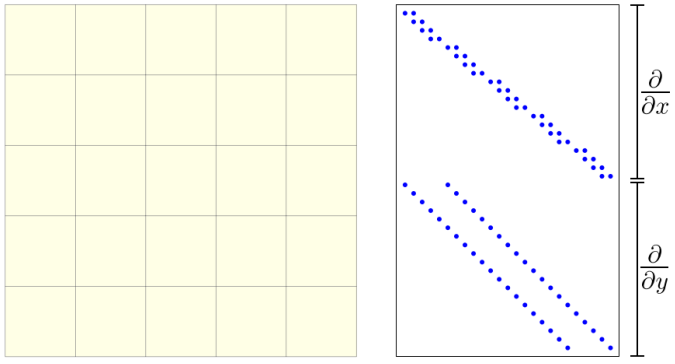
$$grad(\boldsymbol{p})[f] = \boldsymbol{p}[N_2(f)] - \boldsymbol{p}[N_1(f)] \tag{3.28}$$

Assumindo que não haja fluxo no contorno das faces externas, o operador gradiente discreto é o adjunto do operador divergente assim como no caso contínuo (Bao et al., 2017). Essa relação é expressa pela Equação (3.4).

$$\sum_{c} div(\boldsymbol{v})[c]\ \boldsymbol{p}[c] + \sum_{f} grad(\boldsymbol{p})[f]\, \boldsymbol{v}[f] = 0 \tag{3.4}$$

Visto que `div` e `grad` são operadores lineares, eles podem ser representados por uma matriz esparsa $\boldsymbol{D}$, e, portanto, $grad(\boldsymbol{x}) = \boldsymbol{D} \cdot \boldsymbol{x}$ e $div(\boldsymbol{x}) = -\boldsymbol{D}^T \cdot \boldsymbol{x}$. A Figura 5 mostra a estrutura da matriz $\boldsymbol{D}$ para um *grid* Cartesiano 2D.

Figura 5 – Estrutura da matriz esparsa $\boldsymbol{D}$ usada na definição dos operadores diferenciais discretos `div` e `grad` para um *grid* Cartesiano 2D.



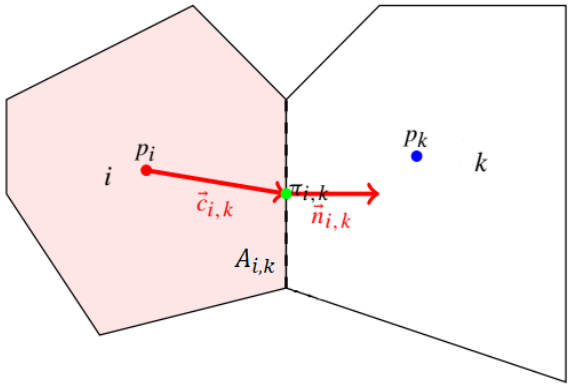
Fonte: (Bao et al., 2017).

Seja $A_{i,k}$ a área de uma “meia-face”, por estar associada apenas à célula de índice i e a um vetor normal $\vec{n}_{i,k}$. Para o caso de malhas com faces em comum, cada meia-face $A_{i,k}$ terá uma meia-face semelhante $A_{k,i}$ de mesma área mas de vetores normais opostos, isto é, $\vec{n}_{k,i} = -\vec{n}_{i,k}$. Logo, considerando o gradiente de pressão como a diferença entre a pressão $\pi_{i,k}$ no centroide da face f e a pressão média p_i dentro da célula, o fluxo $v_{i,k}$ do centroide da célula i até o centroide da meia-face $A_{i,k}$ pode ser expresso pela Equação (3.5).

$$v_{i,k} = A_{i,k} \cdot \boldsymbol{K}_i \cdot \frac{(p_i - \pi_{i,k}) \cdot \vec{c}_{i,k} \cdot \vec{n}_{i,k}}{\left|\vec{c}_{i,k}\right|^2} = T_{i,k} \cdot (p_i - \pi_{i,k}) \tag{3.5}$$

Onde $T_{i,k}$ é definido como “meia-transmissibilidade” por estar associado a uma meia-face, e $\vec{c}_{i,k}$ o vetor do centroide da célula i até o centroide da face f. A Figura 6 esquematiza a aproximação de volumes finitos por dois pontos entre células adjacentes.

Figura 6 – Esquema de duas células utilizadas para definir a discretização de um sistema através de uma aproximação de volumes finitos por dois pontos.



Fonte: (Lie K.-A. , 2016).

A transmissibilidade $T[f]$ associada à conexão entre duas células adjacentes, dadas por $i = N_1(f)$ e $k = N_2(f)$, pode ser então definida como a média harmônica entre as meias-transmissibilidades, conforme expresso na Equação (3.6).

$$T[f] = \left[{T_{i,k}}^{-1} + {T_{k,i}}^{-1}\right]^{-1}, \qquad T_{i,k} = A_{i,k} \cdot \boldsymbol{K}_i \cdot \frac{\vec{c}_{i,k} \cdot \vec{n}_{i,k}}{\left|\vec{c}_{i,k}\right|^2} \tag{3.6}$$

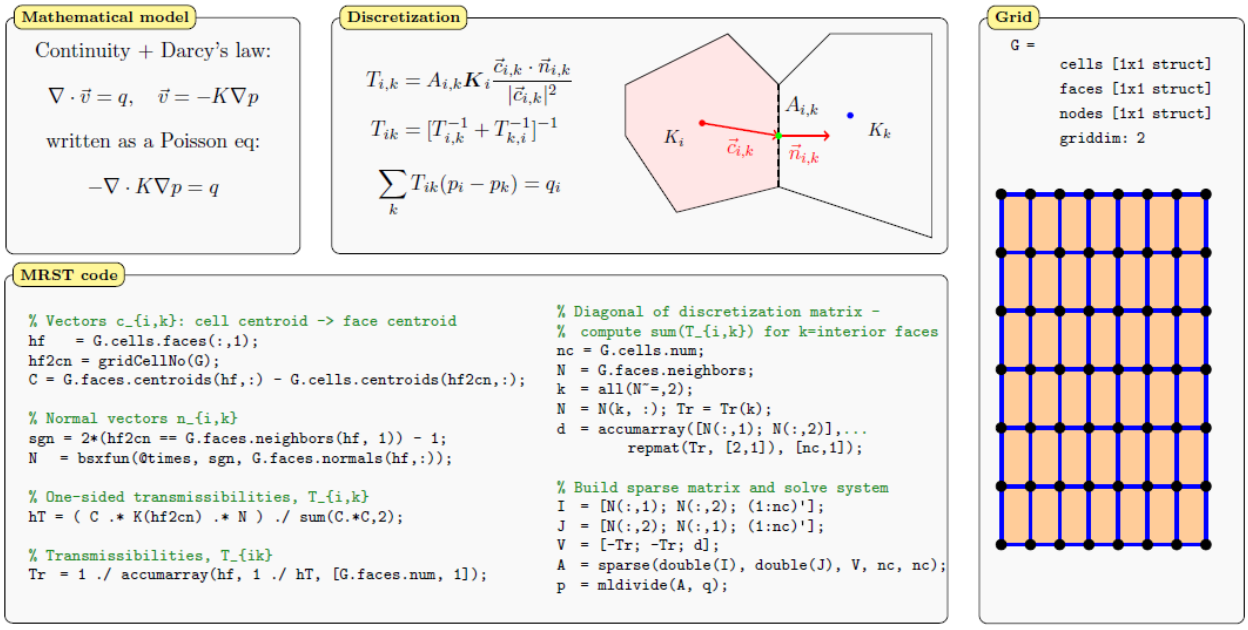
Onde $\boldsymbol{K}_i$ é o tensor de permeabilidade na célula i com os eixos principais alinhados aos eixos da malha.

Por fim, utilizando os operados discretos `div` e `grad` definidos anteriormente, a Equação (3.1) pode ser reescrita conforme a Equação (3.7).

$$div(\boldsymbol{v}) = q, \qquad \boldsymbol{v} = -T \cdot grad(\boldsymbol{p}) \tag{3.7}$$

A Figura 7 ilustra a implementação de um esquema geral de dois pontos para fluxo monofásico incompressível. O *array* `G.cells.faces` mapeia células para meias-faces, enquanto que o *array* `G.cells.neighbors` mapeia cada face a dois índices de célula (um dos dois será zero caso corresponda a uma face externa).

Figura 7 – Implementação de um esquema de dois pontos para fluxo monofásico incompressível.

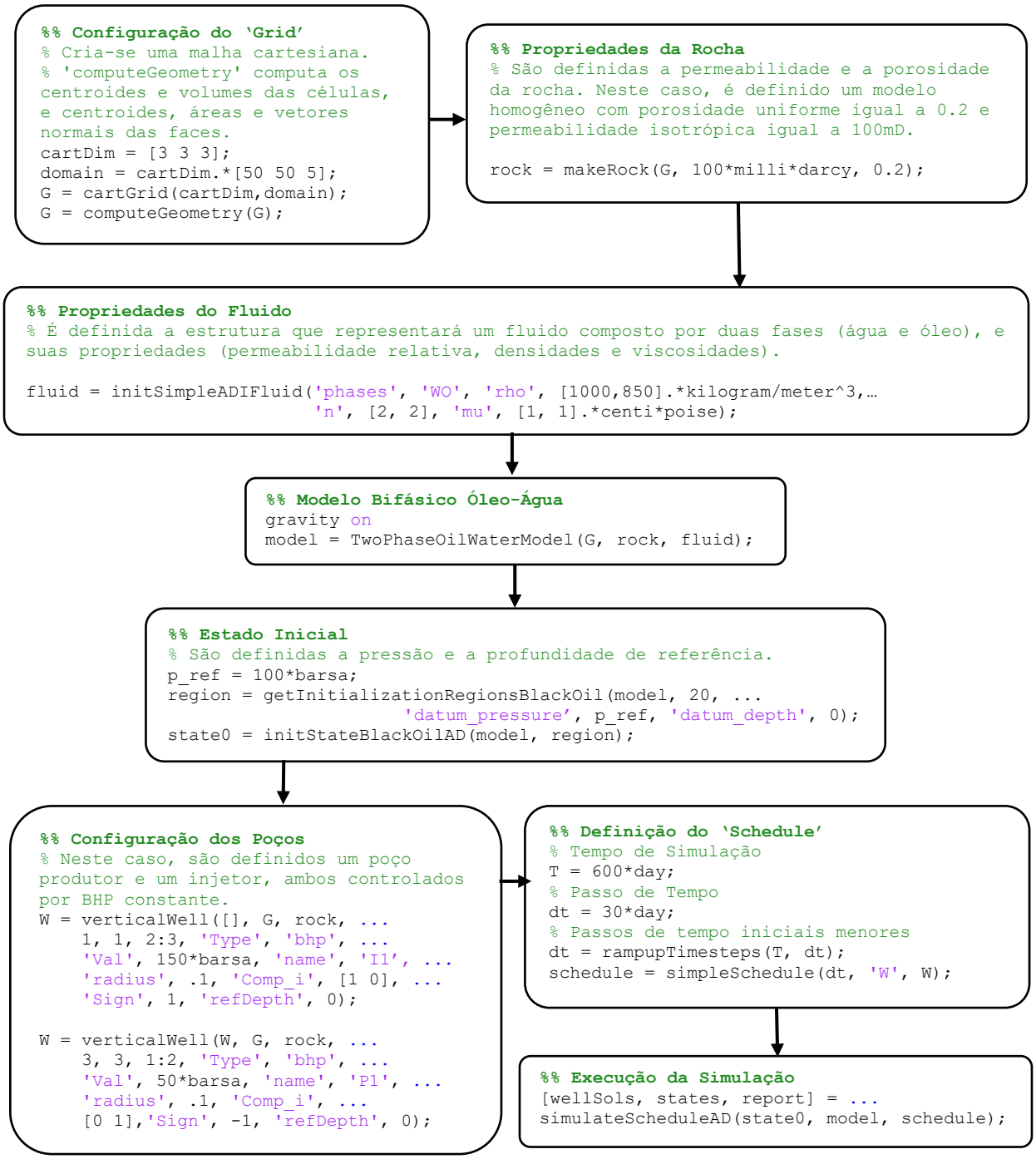


Fonte: (Krogstad et al., 2015).

Além do grid `G`, o MRST apresenta outras estruturas que fazem parte do modelo de simulação. Na estrutura `rock`, são definidas propriedades petrofísicas, como porosidade e permeabilidade, além de fatores multiplicadores que reduzem (ou expandem) o fluxo entre células adjacentes (fator *net-to-gross*, por exemplo). A estrutura `fluid`, por sua vez, representa o modelo de fluido, que implementa um conjunto de funções para obtenção de propriedades como densidades e viscosidades das fases, e também curvas de permeabilidade relativa, fator volume de formação etc. Todas essas estruturas são incorporadas à estrutura `model`, que também implementa funções utilitárias para acessar o comportamento do modelo.

A Figura 8 exemplifica a implementação de um modelo Black-Oil simplificado, composto por apenas vinte e sete células e dois poços, um injetor e outro produtor.

Figura 8 – Exemplo de implementação de um simulador *Black-Oil* no MRST.



Fonte: O Autor (2019).

Definida uma pressão e altura de referência, faz-se a inicialização do modelo *Black-Oil*, estabelecendo um mapa de pressões e saturações inicial do reservatórios, que é armazenado no parâmetro `state0`. Na estrutura `W` são definidos os poços e suas características, como raio, locação, tipo de controle e zona de completação, enquanto que no parâmetro `schedule` são definidos o conjunto de controles a serem aplicados em cada poço. Definidos os parâmetros `state0`, `model` e `schedule`, realiza-se então a simulação do reservatório, cujos mapas de pressão e saturação resultantes são armazenados na estrutura `states`, e cujas curvas de injeção e produção são armazenados na estrutura `wellSols`.

3.2 Programação Orientada a Objetos

O desenvolvimento de *softwares* geralmente envolve o projeto dos dados do aplicativo e a implementação das operações executadas nesses dados. Programas processuais (tradicionalmente chamados de procedurais ou de programação estruturada) passam dados para determinadas funções, que executam as operações necessárias nesses dados. A implementação de s*oftwares* orientada a objetos, por sua vez, encapsula os dados e as operações em objetos e classes que interagem uns com os outros.

Hoje, muitas linguagens de programação são orientadas a objetos como Java, C#, Python e C++. A maioria delas adota as ideias parcialmente, dando espaço para o antigo modelo procedural de programação, como acontece no C++, por exemplo, onde temos a possibilidade de usar programação orientada a objetos, mas a linguagem não força o programador a adotar este paradigma de programação, sendo ainda possível programar da forma procedural tradicional, muito utilizada há alguns anos. A linguagem MATLAB permite que você crie programas usando técnicas orientadas a objetos e procedimentos, e use objetos e funções comuns em seus programas.

Em programas processuais tradicionais, são implementadas operações como funções que tomam as variáveis como argumentos de entrada. Os programas geralmente fazem a chamada de uma sequência de funções, onde, para cada uma, são passados dados de entrada e, em seguida, retornam dados modificados.

A POO foi criada para tentar aproximar o mundo real do mundo virtual: a ideia fundamental é tentar simular o mundo real dentro do computador. Na POO, o programador é responsável por moldar o mundo dos objetos, e explicar para estes objetos como eles devem interagir entre si. Os objetos “conversam” entre si através do envio de mensagens, e o papel principal do programador é especificar quais serão as mensagens que cada objeto pode receber, e também qual a ação que aquele objeto deve realizar ao receber aquela mensagem em específico.

Um programa orientado a objetos envolve:

- Identificar os componentes do aplicativo que você deseja construir;
- Analisar e identificar padrões para determinar quais componentes são usados repetidamente ou que possuem características em comum;
- Classificar componentes com base em semelhanças e diferenças.

Feita essa análise, pode-se definir as *Classes* que definem os objetos que o programa utilizará durante a sua execução. Uma classe descreve um conjunto de objetos com características em comum. *Objetos* são instâncias específicas de uma classe. Os valores contidos nas propriedades (ou atributos) de um objeto são o que tornam um objeto diferente de outros objetos de uma mesma classe. As funções definidas dentro de uma classe, os *Métodos*, implementam comportamentos que são comuns a todos os objetos de uma mesma classe. Os objetos geralmente possuem métodos que executam “*overload*” de funções do MATLAB, isto é, fazem a chamada de funções já existentes no MATLAB redefinidas para a classe específica do objeto. Contudo, objetos devem suportar apenas os métodos que fazem sentido para si. Um método especial é o *construtor*: ele serve para inicializar os atributos e é executado automaticamente sempre que você cria um novo objeto.

Implementar tarefas de programação simples é mais conveniente através de funções simples. No entanto, à medida que as funções se tornam muito complexas, o gerenciamento dos dados transmitidos às funções torna-se difícil e propenso a erros e, portanto, os benefícios de um projeto orientado a objetos tornam-se mais visíveis.

Dentre as vantagens quanto ao uso de POO, está a possibilidade de reutilização do código criado, reduzindo o tempo de desenvolvimento, redundância e complexidade do problema, e consequentemente o número de linhas de código. Um projeto orientado

a objetos pode resumir em um código comum, que é chamado de *classe-base*. A classe-base (superclasse ou “classe-mãe”) define o algoritmo usado e implementa o que for comum para todos os casos que usam esse código, deixando a implementação especializada para as classes derivadas dessa classe-base, ou seja, para as “classes-filhas” (ou subclasses). O código na classe-base não é copiado ou modificado. Classes derivadas da classe-base *herdam* esse código. A *herança* consiste em outra vantagem da POO, visto que reduz a quantidade de código a ser testado e isola seu programa das alterações para o procedimento básico.

O objetivo básico de uma classe é definir um objeto que *encapsula* os seus dados e as operações executadas nesses dados. A Figura 9 ilustra um exemplo de definição de uma classe no MATLAB (The MathWorks Inc., 2018).

Figura 9 – Exemplo de definição de uma classe usando Orientação a Objetos no MATLAB.

```
classdef BasicClass
    properties
        Value
    end
    methods
        function obj = BasicClass(val)
            if nargin == 1
                if isnumeric(val)
                    obj.Value = val;
                else
                    error('Value must be numeric');
                end
            end
        end
        function r = roundoff(obj)
            r = round([obj.Value]);
        end
        function r = multiplyBy(obj,n)
            r = [obj.Value]*n;
        end
        function r = plus(obj1,obj2)
            r = [obj1.Value] + [obj2.Value];
        end
    end
end
```

Fonte: (The MathWorks Inc., 2018) - Adaptado pelo Autor.

Como é possível notar, `BasicClass` armazena o dado (que deve ser numérico) no objeto pertencente à classe através da propriedade `Value`. As funções `roundoff` e `multiplyBy` são métodos definidos que, respectivamente, arredonda o valor da

propriedade para duas casas decimais e multiplica o valor da propriedade pelo número “n” especificado. A função `BasicClass()` é um método especial para criar objetos, chamado *construtor*, que permitem passar argumentos, os quais são validados e valores de propriedade podem ser atribuídos a eles. A função `plus` consiste, por sua vez, num método com o mesmo nome da função MATLAB existente, implementando funcionalidades existentes, como a adição dos valores atribuídos a dois objetos da classe `BasicClass`. Trata-se, portanto, de um “*overload*” da função `plus` do MATLAB.

Ao longo das Equações (3.8) a (3.13), seguem alguns exemplos para a criação e manipulação de classes e objetos no MATLAB, com suas respectivas propriedades.

• Salvar a definição de classe em um arquivo `.m` com o mesmo nome da classe:

```
classdef BasicClass
```
(3.8)

• Criar um objeto da classe `BasicClass`:

```
a = BasicClass
a =
    BasicClass with properties:
    Value: []
```
(3.9)

• Atribuir dados às propriedades do objeto recém-criado:

```
a.Value = pi/3;
```
(3.10)

• Acessar as propriedades do objeto:

```
a.Value
ans =
     1.0472
```
(3.11)

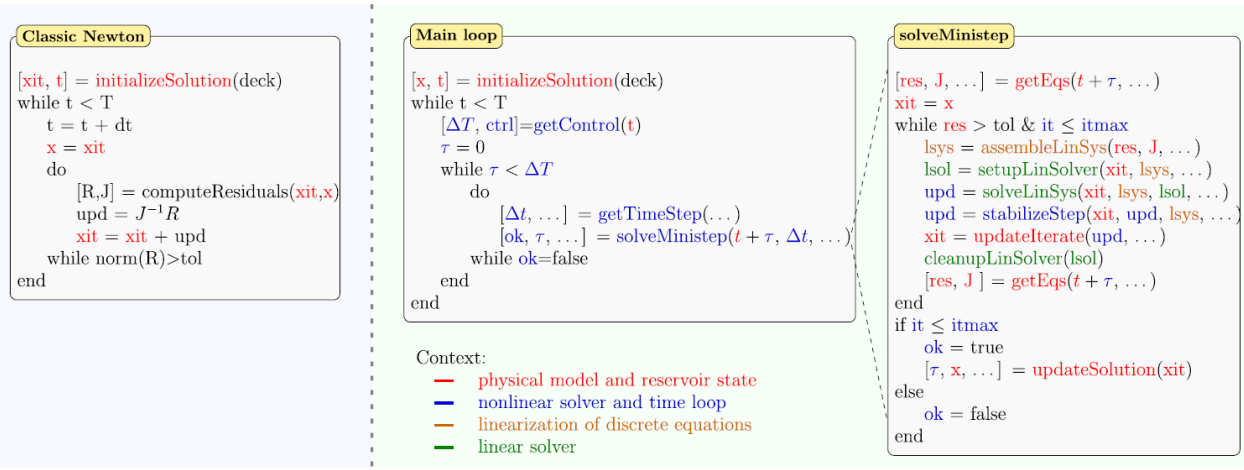
• Chamar métodos para executar a operação nos dados, podendo existir mais de uma notação possível para a chamada da função:

```
roundoff(a)
ans =
     1.0500
```
(3.12)

```
a.multiplyBy(3)
ans =
     3.1416
```
(3.13)

Uma maneira de projetar um código transparente e bem organizado é dividir o *loop* de simulação em diferentes contextos numéricos, por exemplo, conforme descrito na Figura 10, e apenas expor os detalhes necessários em cada um desses contextos (Bao et al., 2017).

Figura 10 – Comparação de um *time loop* tradicional composto pelo Método de Newton e uma abordagem mais sofisticada utilizada no módulo `ad-core`.



Fonte: (Bao et al., 2017).

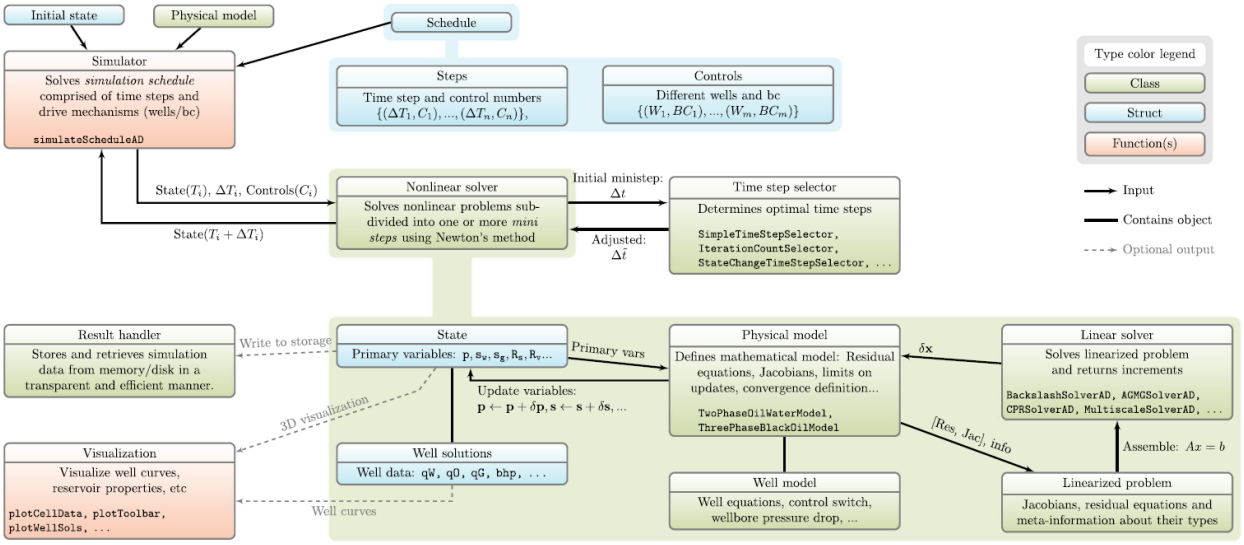
O módulo `ad-core` no MRST oferece uma estrutura orientada a objetos que permite ao usuário separar modelos físicos e estados do reservatório, *solvers* não-lineares e *loops* de tempo, equações e linearizações discretas de fluxo e *solvers* lineares. Tem-se, por exemplo, o método `solveMinistep` que subdivide os ciclos de controle dos poços em intervalos de tempo menores, definido pelo próprio usuário ou como medida mitigadora de erros.

Essa estrutura que compõe o `ad-core` foi feita a fim de oferecer suporte à prototipagem rápida de simuladores de reservatórios com base em formulações totalmente implícitas, além de funcionalidades mais específicas. A Figura 11 descreve como várias classes, estruturas e funções podem ser organizadas para formular um simulador *Black-Oil* de modo eficiente. Os diferentes componentes estão coloridos pelo tipo ao qual foram definidos, isto é, classe, estrutura ou função.

A Derivação Automática (AD), apresentada na próxima seção, é uma classe composta por objetos presentes no MRST, cuja ideia-chave é acompanhar as quantidades e suas derivadas simultaneamente, ou seja, toda vez que uma operação é

aplicada a uma quantidade, a correspondente operação diferencial é aplicada a sua derivada, utilizando ainda o "*overload*" de operadores já existentes no MATLAB.

Figura 11 – Visão geral de como os componentes na estrutura orientada a objetos são organizados para implementar um simulador Black-Oil no MRST.



Fonte: (Bao et al., 2017).

3.3 Derivação Automática

Assim como a maioria dos simuladores comerciais baseados em uma formulação *Black-Oil*, o MRST fornece um simulador totalmente implícito, oferecendo estabilidade incondicional para uma ampla gama de regimes de fluxo e heterogeneidades de reservatório. Além disso, a formulação totalmente implícita é combinada com a técnica de Derivação Automática (AD), garantindo a extensão simples dos modelos de fluxo básicos. Usando rotinas numéricas e vetorização do MATLAB combinadas com operadores discretos diferenciais do MRST, as equações do modelo podem ser implementadas de forma muito compacta e bastante semelhantes à formulação matemática.

Na Derivação Automática, a ideia principal é se obter as quantidades e suas derivadas simultaneamente, ou seja, toda vez que uma operação é aplicada a uma variável, a operação diferencial correspondente é aplicada a sua derivada. Assim, o módulo MRST-AD permite que a equação de pressão não-linear, usando formulação

totalmente implícita, calcule automaticamente a matriz Jacobiana exata para as equações de óleo e água, usando operadores discretos.

Esta é uma técnica que explora o fato de que qualquer código computacional, independentemente de sua complexidade, pode ser dividido em um conjunto limitado de operações aritméticas e de chamadas de funções simples.

Considere uma variável primária escalar x e uma função $f = f(x)$. Suas representações do tipo AD são os pares $\langle x, 1\rangle$ e $\langle f, f_x\rangle$ onde 1 é a derivada ${dx}/{dx}$ e f_x é a derivada ${df}/{dx}$. Por conseguinte, a ação das operações e funções elementares deve ser definida para esses pares, por exemplo, $\langle f, f_x\rangle + \langle g, g_x\rangle = \langle f + g, f_x + g_x\rangle$ ou $\langle f, f_x\rangle * \langle g, g_x\rangle = \langle f * g, f * g_x + f_x * g\rangle$.

Além disso, pode-se usar a regra da cadeia para derivadas de funções compostas, ou seja, se $f(x) = g(h(x))$, então $f_x(x) = {dg}/{dh} * h_x(x)$. Isso resume a ideia principal por trás da diferenciação automática: escrevendo as fórmulas e especificando as variáveis independentes, o *software* calcula as derivadas correspondentes ou jacobianas.

Qualquer nova variável definida em função de x se tornará automaticamente uma variável AD, cujo valor e derivadas são acessadas nas propriedades `val` e `jac` do objeto instanciado, respectivamente.

Exemplo 1: Considere $z = 3e^{-xy}$ e suas derivadas parciais ${dz}/{dx}$ e ${dz}/{dy}$.

```
[x,y] = initVariablesADI(1,2);
```
(3.14)

```
z = 3*exp(-x*y);
```
(3.15)

A Equação (3.14) inicializa x e y como variáveis ADI, isto é, x e y são declarados como objetos da classe ADI do MRST, cujas propriedades correspondem ao seu valor e suas derivadas com relação a todas as variáveis declaradas em conjunto, como detalhado ao longo das Equações (3.16) a (3.18). Note que valores iniciais definidos para x e y são 1 e 2, respectivamente.

A Equação (3.15), por sua vez, expressa z em função de x e y. Na Derivação Automática, toda e qualquer nova variável expressa em função de outras variáveis já

declaradas como ADI, automaticamente também é uma variável ADI. Logo, as derivadas de z em relação a x e y serão calculadas como expresso a seguir.

```
x =
     ADI with properties:
     val: 1
     jac: {[1]  [0]}
```
(3.16)

```
y =
     ADI with properties:
     val: 2
     jac: {[0]  [1]}
```
(3.17)

```
z =
     ADI with properties:
     val: 0.4060
     jac: {[-0.8120]  [-0.4060]}
```
(3.18)

O principal objetivo ao se utilizar objetos da classe ADI no MRST é linearizar e construir (grandes) sistemas de equações discretas. Para usar a Derivação Automática a fim de construir e resolver um sistema linear do tipo $Ax = b$, primeiro escreve-se o sistema na sua forma residual, conforme expresso na Equação (3.19).

$$r(x) = Ax - b = 0 \therefore {}^{\partial r(x)}/_{dx} = A \tag{3.19}$$

Exemplo 2: Considere o sistema de equações linear definido na Equação (3.20).

$$\begin{bmatrix} 3 & 2 & -4 \\ 1 & -4 & 2 \\ -2 & -2 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -5 \\ -1 \\ 6 \end{bmatrix} \tag{3.20}$$

Inicializando a variável x como sendo objeto da classe ADI, e concatenando as equações escritas, cada uma, na sua forma residual, dadas pelas Equações (3.21) a (3.25), pode-se observar que ${}^{dr(x)}/_{dx} = A$, isto é, a derivada do resíduo é o próprio Jacobiano do sistema, dado pela Equação (3.26).

```
x = initVariablesADI(zeros(3,1));
```
(3.21)

```
eq1 = [3, 2, -4]*x + 5;
```
(3.22)

```
eq2 = [1, -4, 2]*x + 1;
```
(3.23)

```
eq3 = [-2, -2, 4]*x - 6;
```
(3.24)

```
eq = cat(eq1, eq2, eq3);
```
(3.25)

```
eq.jac{1} =
     3     2    -4
     1    -4     2
    -2    -2     4
```
(3.26)

É claro que, para a solução de sistemas lineares como exposto no Exemplo 2, o uso de Derivação Automática não é vantajoso. Contudo, para problemas de otimização baseado em gradientes, onde restrições não-lineares são impostas, como ilustrado no Exemplo 3, torna-se de grande valia o uso de tal técnica.

Exemplo 3: Seja o problema de otimização, definido pela Equação (3.27), sujeito a restrições não-lineares.

$$Minimize\ f(x) = 2\alpha L {x_1}^2 + L {x_2}^2$$

$$Subject\ to:$$

$$\begin{aligned} g_1(x) &= \frac{qL}{2{x_1}^2} + \frac{3qL^1 x_1}{6{x_1}^4 + 4\alpha {x_2}^2} - \sigma_{adm} \leq 0 \\ g_2(x) &= \frac{{x_1}^4 (x_2 + 6\alpha L)}{{x_2}^3 (6{x_1}^4 + 4\alpha {x_2}^4)} \frac{qL}{2\alpha} - \sigma_{adm} \leq 0 \\ g_3(x) &= \frac{{x_1}^4 x_2 + \alpha L (3{x_1}^4 + 6\alpha {x_2}^4)}{{x_2}^3 (6{x_1}^4 + 4\alpha {x_2}^4)} \frac{qL}{2\alpha} - \sigma_{adm} \leq 0 \end{aligned} \tag{3.27}$$

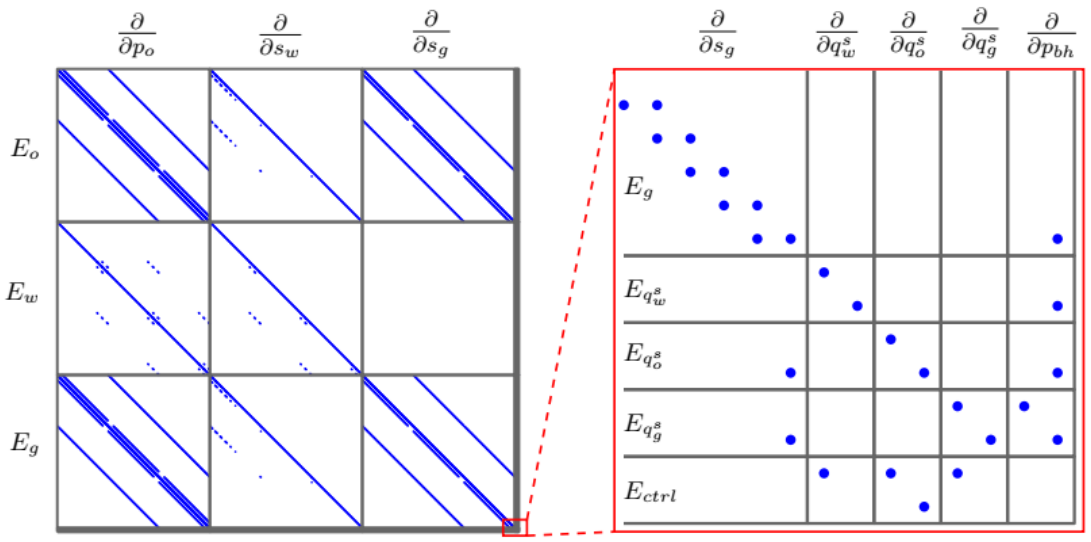
$$x_1, x_2 \geq 0$$

A classe ADI difere da maioria das outras bibliotecas pois, ao invés de representar o Jacobiano em relação a múltiplas variáveis como uma única matriz, os desenvolvedores do MRST optaram por permitir que `jac` seja uma lista de matrizes esparsas, cada uma representando as derivadas em relação a uma única variável primária. Dessa forma, mantendo os sub-Jacobianos separados, evita-se manipular subconjuntos de grandes matrizes esparsas, que apresenta baixo desempenho no MATLAB. Além disso, essa abordagem permite aos usuários manipular blocos das matrizes que representam sub-equações específicas no Jacobiano de um sistema de equações.

A diferenciação automática no MRST, portanto, usa uma lista de matrizes que representam as derivadas com relação a diferentes variáveis que constituirão sub-

blocos da Jacobiana do sistema completo, ao invés de trabalhar com uma única matriz Jacobiana para o sistema discreto completo (Krogstad et al., 2015). Essa representação em “blocos” do Jacobiano permite uma implementação eficiente e compacta. Na Figura 12, é ilustrada a interface do Jacobiano para um sistema *Black-Oil* trifásico contendo dois poços, composto por um sistema de sete equações: três equações do reservatório, três equações definindo as vazões do poço na superfície igual à soma das contribuições de cada perfuração, e a equação de controle garantindo que cada poço opere sob o controle especificado (BHP e/ou vazão). Da mesma forma, existem sete variáveis primárias: pressão de óleo, saturação de água, saturação de gás, vazões de superfície para cada um dos três componentes e a pressão de fundo de cada poço. Assim, por exemplo, o bloco na posição $(3,2)$ da Figura 12 (à esquerda) é a derivada parcial da equação de gás em relação à saturação da água, isto é, $\partial E_g / \partial S_w$.

Figura 12 – Ilustração da estrutura das equações do modelo *Black-Oil* linearizadas e de todos os sub-Jacobianos que compõem o sistema linear geral para um problema com dois poços.



Fonte: (Krogstad et al., 2015).

3.4 Comparação com Simulador Comercial

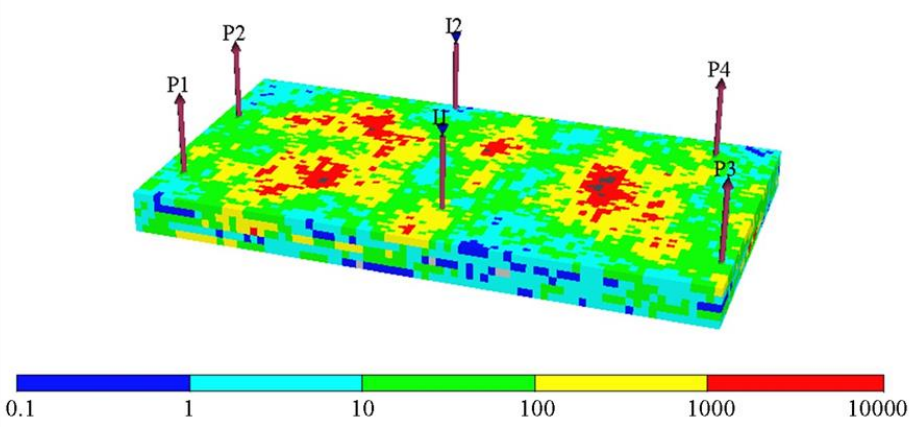
Como uma tentativa de validação do MRST frente ao uso de simuladores comerciais, buscou-se comparar os resultados da simulação de um modelo de

reservatórios obtida com o uso do MRST ao resultado obtido com o IMEX do grupo CMG, simulador já consagrado pela indústria e por ser mais comumente utilizado na Petrobras. Utilizou-se durante toda a realização deste trabalho a versão 2018b do MRST.

O teste proposto para essa validação é uma porção modificada de um reservatório sintético paralelepípedo (“caixa de sapato”) com a existência de canais, proposto inicialmente em (Christie et al., 2001), também utilizado em (Cardoso, 2009) e (Fragoso, 2014) como um dos modelos para aplicação do modelo de ordem reduzida TPWL/POD.

O modelo é tridimensional de dimensões 60 × 80 × 5 células, isto é, possui 24000 células, cada um com dimensões de 50 × 70 × 20ft. Há 4 poços produtores (nomeados P1, P2, P3 e P4) e dois injetores (I1 e I2). O valor médio para a permeabilidade na direção x é $k_x = 418mD$. A Figura 13 ilustra o mapa de permeabilidades na direção x, k_x com os poços posicionados. A permeabilidade vertical é definida como $k_z = 0{,}3k_x$ nos canais e $k_z = 10^{-3}k_x$ na porção restante. As saturações iniciais de óleo e água são respectivamente $S_{oi} = 0{,}2$ e $S_{wc} = 0{,}8$ e as residuais, $S_{or} = S_{wr} = 0{,}2$. Para o óleo tem-se $\rho_o = 45lb/ft^3$ e $\mu_o = 3cp$ e para a água $\rho_w = 60lb/ft^3$ e $\mu_w = 0{,}3cp$. O sistema é pouco compressível, com $c_o = c_w = c_r = 10^{-6}psi^{-1}$ (compressibilidades dos ambos os fluidos e da rocha iguais), e efeito da pressão capilar é desprezado.

Figura 13 – Mapa de permeabilidades k_x do modelo SPE10 (Modificado).



Fonte: (Fragoso, 2014).

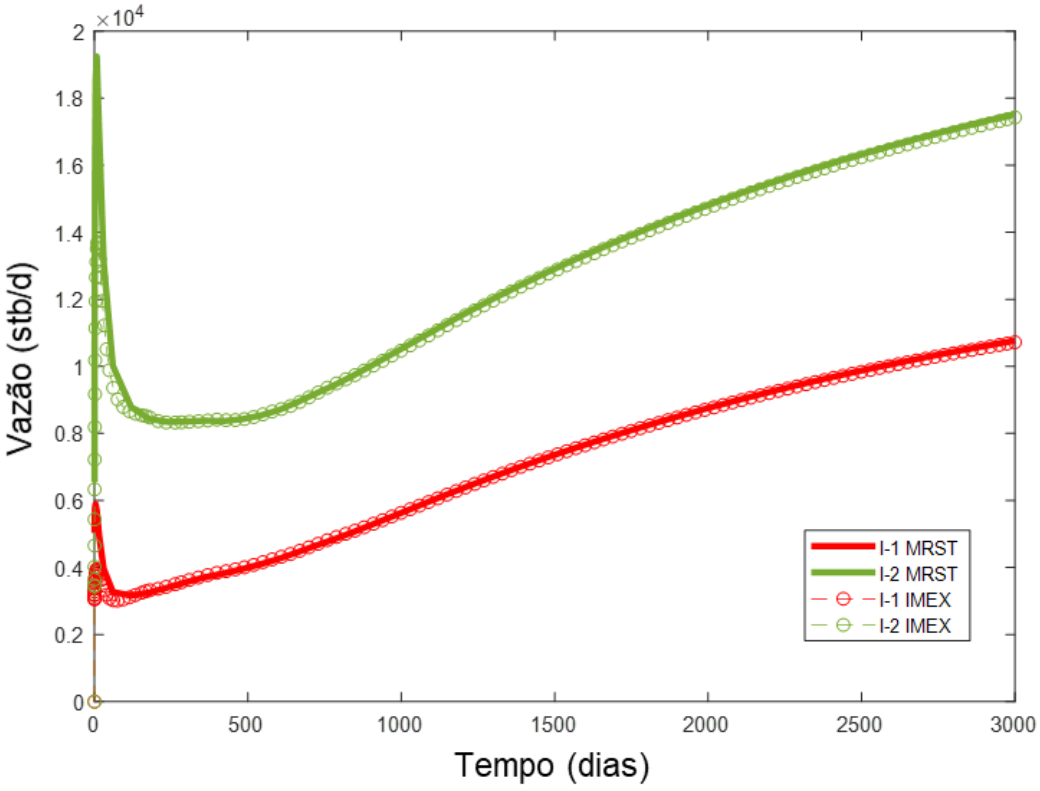
As permeabilidades relativas, por sua vez, são calculadas através do modelo de Corey, dadas pelas Equações (2.13) e (2.14) com os seguintes parâmetros: $k_{ro}^{0} = k_{rw}^{0} = 1$ e $a = b = 2$.

Neste teste comparativo, todos os poços foram controlados por pressão de fundo especificada (BHP), sendo para os produtores $p_{prod}^{w} = 4000\text{psi}$ e para os injetores $p_{inj}^{w} = 11000\text{psi}$. O tempo total de simulação foi de 3000 dias.

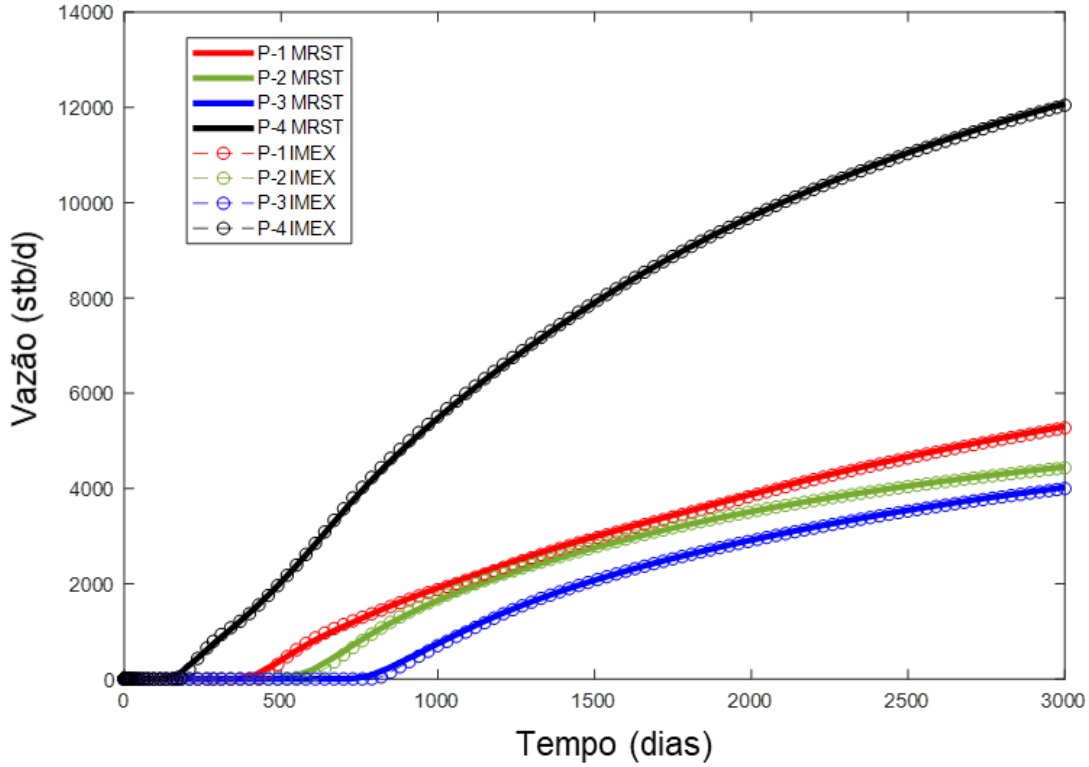
Os resultados obtidos para este modelo SPE10 (Modificado) estão ilustrados Figura 14. A comparação foi feita considerando a vazão de água injetada (Figura 14a), a vazão de água produzida (Figura 14b), e a vazão de óleo produzido (Figura 14c), para cada respectivo poço. Observando os gráficos, percebe-se boa aderência entre as vazões calculadas por ambos os simuladores para todo o tempo de simulação.

Figura 14 – Comparação de resultados obtidos para o Modelo SPE10, via MRST e IMEX.

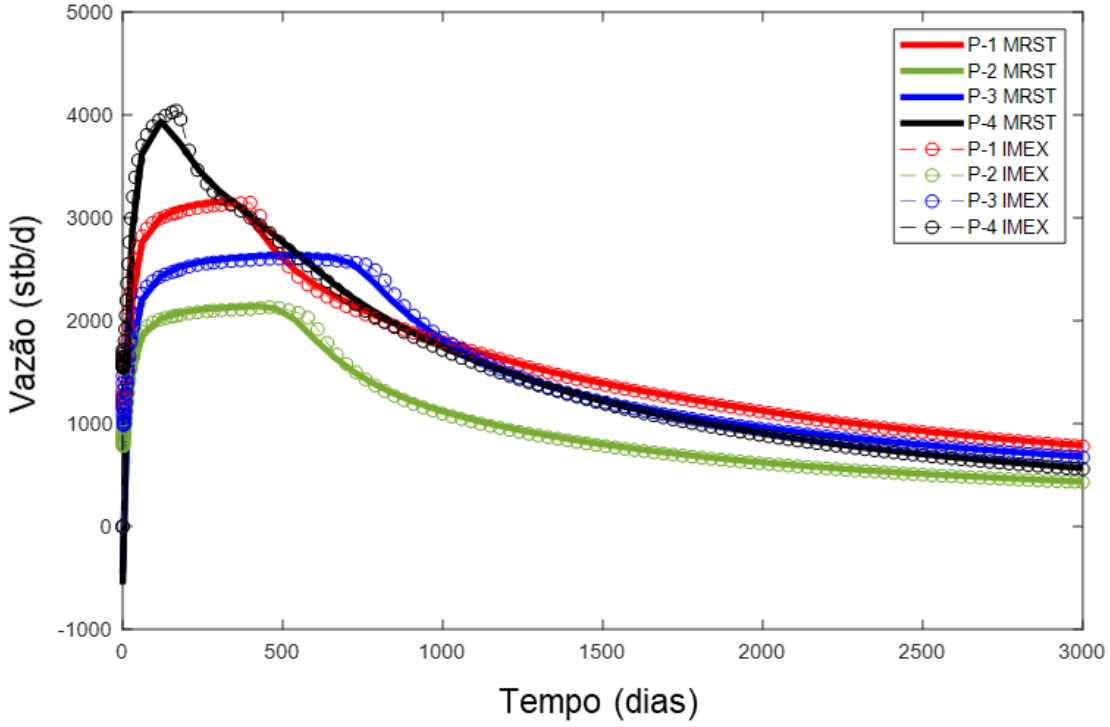
(a) Vazão de Água Injetada (stb/d)



(b) Vazão de Água Produzida (stb/d)



(c) Vazão de Óleo Produzido (stb/d)



Fonte: O Autor (2019).

As curvas obtidas no IMEX foram geradas a partir de dados fornecidos por (Fragoso, 2014), que utilizou um computador com processador Intel Core 2 Duo 3GHz com 4GB de RAM e sistema operacional Windows 7, cujo tempo computacional requerido durante a simulação foi de 194s. A simulação no MRST, por sua vez, utilizou um computador com processador Intel Core i7 3.4GHz com 16GB de RAM e sistema operacional também Windows 7, cujo tempo computacional requerido foi de 2052s.

Portanto, nota-se que, mesmo com um computador de melhores configurações, o MRST ainda apresenta um desempenho muito abaixo de simuladores comerciais, como o IMEX. Conforme já mencionando anteriormente, (Bao et al., 2017) afirma que testes em modelos bifásicos e trifásicos da ordem de dez a cem mil células mostram que os simuladores MRST baseados na Diferenciação Automática são entre duas e dez vezes mais lentos que o simulador comercial totalmente otimizado.

4 MÉTODO DE ORDEM REDUZIDA: TPWL/POD

Neste capítulo, são introduzidos os aspectos teóricos acerca das estratégias utilizadas para a criação do modelo substituto de ordem reduzida a ser aplicado em processos de otimização mais adiante. A técnica escolhida para a redução da complexidade numérica denomina-se *Trajectory Piecewise Linearization* (TPWL) e consiste na linearização da equação de fluxo na sua forma residual em torno de estados convergidos e armazenados durante uma simulação previamente executada, denominada simulação de treinamento. Para a redução da dimensionalidade do problema, a técnica escolhida é o *Proper Orthogonal Decomposition* (POD). A combinação de ambas as técnicas foi aplicada inicialmente ao contexto de simulação de reservatórios em (Cardoso, 2009).

Contudo, há um grande número de trabalhos que mencionam problema de estabilidade durante a aplicação desta técnica, como em (He, 2010) e (Gildin et al., 2013). Estudos como os de (Carlberg et al., 2009) e (He et al., 2013) demostram que a técnica pode ser estabilizada pela substituição do tipo de projeção utilizada no POD, isto é, substituindo a projeção de Bubnov-Galerkin pela projeção de Petrov-Galerkin. Neste capítulo, serão feitas análises comparando a utilização de ambos os tipos de projeção.

Na primeira seção deste capítulo, é descrito o funcionamento do algoritmo do TPWL, apresentando detalhes quanto à linearização aplicada à equação de fluxo. Em seguida, são apresentados detalhes de como foram exportados os estados, conhecidos também por *snapshots*, e as matrizes de derivadas necessárias à equação do TWPL, a partir de modificações realizadas em classes e funções que compõem a estrutura do MRST. Em seguida, é detalhado o método de redução da dimensionalidade do problema através do POD, com a inclusão do esquema *Local Resolution*, que mantém de forma integral as informações relativas às células que apresentam perfurações e completações de poços. Por fim, são apresentados os resultados obtidos do modelo de ordem reduzida utilizando os estados e as matrizes salvas durante uma única simulação de treinamento, tomando como controles as pressões de fundo dos poços (BHP), e comparando tais resultados aos obtidos pela simulação de alta fidelidade.

4.1 *Trajectory Piecewise Linearization*

No Capítulo 2, viu-se que a equação de fluxo na sua forma contínua, dada pela Equação (2.10), ao ser discretizada, pode ser reescrita dividindo-a em três parcelas, conforme a Equação (4.1), referentes: aos *termos de acumulação*, que representam toda a massa acumulada na célula, em função da compressibilidade dos fluidos e da rocha; aos *termos de fluxo*, que representam a massa que flui de uma célula para outra, função das transmissibilidades nas interfaces; e aos *termos de fonte/sumidouro*, que representam a massa injetada ou produzida em células onde há completação, função do modelo de poço adotado. A soma dessas três parcelas resulta no resíduo $\boldsymbol{g}$, que é igual a zero pois representa o resíduo previamente convergido durante a simulação de treinamento.

$$\boldsymbol{g}\left(\boldsymbol{x}^{i+1}, \boldsymbol{x}^{\boldsymbol{i}}, \boldsymbol{u}^{i+1}\right) = \boldsymbol{A}\left(\boldsymbol{x}^{i+1}, \boldsymbol{x}^{\boldsymbol{i}}\right) + \boldsymbol{F}\left(\boldsymbol{x}^{i+1}\right) + \boldsymbol{Q}\left(\boldsymbol{x}^{i+1}, \boldsymbol{u}^{i+1}\right) = 0 \qquad (4.1)$$

Como a Equação (4.1) é não-linear, utiliza-se o Método de Newton-Raphson para a solução do sistema, onde os índices i e $i+1$ designam o passo de tempo da simulação. As matrizes $\boldsymbol{A}$, $\boldsymbol{F}$ e $\boldsymbol{Q}$ representam os termos de acumulação, fluxo e fonte/sumidouro, respectivamente. O vetor $\boldsymbol{x}$, por sua vez, representa o vetor de estados, conhecido também por *snapshot*, composto pelo mapa de pressão e de saturação de água em um dado passo de tempo da simulação de um modelo de reservatório.

A técnica conhecida como *Trajectory Piecewise Linearization* (TPWL) se baseia na linearização da equação de fluxo na sua forma residual em torno de estados salvos durante uma simulação previamente executada, também chamada de simulação de treinamento. A cada passo de tempo, lineariza-se o modelo em torno de um ponto de alguma trajetória de treinamento, cujos estados foram previamente convergidos e armazenados (Cardoso, 2009).

Observa-se que, na Equação (4.1), o resíduo $\boldsymbol{g}$ é função do vetor de estados do passo de tempo atual $\boldsymbol{x}^{i+1}$, do vetor de estados do passo anterior $\boldsymbol{x}^{i}$ e do vetor de controles dos poços $\boldsymbol{u}^{i+1}$. Pode-se então aproximar o resíduo de uma nova simulação

$\boldsymbol{g}^{n+1}$, utilizando expansão em Série de Taylor, a partir do vetor de estados calculado no passo de tempo anterior $\boldsymbol{x}^{n}$, utilizando os estados previamente salvos no treinamento.

$$\boldsymbol{g}^{n+1} = \boldsymbol{g}^{i+1} + \frac{\partial \boldsymbol{g}^{i+1}}{\partial \boldsymbol{x}^{i+1}}\left(\boldsymbol{x}^{n+1} - \boldsymbol{x}^{i+1}\right) + \frac{\partial \boldsymbol{g}^{i+1}}{\partial \boldsymbol{x}^{i}}\left(\boldsymbol{x}^{n} - \boldsymbol{x}^{i}\right) + \frac{\partial \boldsymbol{g}^{i+1}}{\partial \boldsymbol{u}^{i+1}}\left(\boldsymbol{u}^{n+1} - \boldsymbol{u}^{i+1}\right) \tag{4.2}$$

Note que na Equação (4.2), o sobrescrito i representa um passo de tempo de uma simulação de treinamento, onde as derivadas, estados e controles utilizados foram convergidos e armazenados, enquanto que o sobrescrito n representa os passos de tempo de uma nova simulação, considerando o mesmo estado inicial, porém com um novo conjunto de controles.

Para fins de simplificação, serão adotadas a partir de agora as notações $\boldsymbol{g}^{n+1} = \boldsymbol{g}(\boldsymbol{x}^{n+1}, \boldsymbol{x}^{n}, \boldsymbol{u}^{n+1})$ e $\boldsymbol{g}^{i+1} = \left(\boldsymbol{x}^{i+1}, \boldsymbol{x}^{i}, \boldsymbol{u}^{i+1}\right)$. De maneira análoga ao resíduo $\boldsymbol{g}^{i+1}$ da Equação (4.1), $\boldsymbol{g}^{n+1}$ é igual a zero pois representa o resíduo previamente convergido durante a nova simulação.

Por definição, a matriz Jacobiana é a derivada do resíduo em relação ao vetor de estados. Assim, a primeira derivada parcial presente na Equação (4.2) é definida conforme a Equação (4.3).

$$\frac{\partial \boldsymbol{g}^{i+1}}{\partial \boldsymbol{x}^{i+1}} = \boldsymbol{J}^{i+1} \tag{4.3}$$

Visto que o resíduo é resultado da soma de fatores que não dependem todos da tripla $\boldsymbol{x}^{i+1}, \boldsymbol{x}^{i}, \boldsymbol{u}^{i+1}$, as outras duas derivadas também podem ser reescritas. Como somente o termo de acumulação possui dependência em relação ao estado no passo de tempo anterior, tem-se que a derivada do resíduo $\boldsymbol{g}^{i+1}$ em relação ao vetor de estados $\boldsymbol{x}^{i}$ representa a derivada dos termos de acumulação em relação a $\boldsymbol{x}^{i}$, conforme expresso na Equação (4.4).

$$\frac{\partial \boldsymbol{g}^{i+1}}{\partial \boldsymbol{x}^{i}} = \frac{\partial}{\partial \boldsymbol{x}^{i}}\left[\boldsymbol{A}(\boldsymbol{x}^{n+1}, \boldsymbol{x}^{n}) + \boldsymbol{F}(\boldsymbol{x}^{n+1}) + \boldsymbol{Q}(\boldsymbol{x}^{n+1}, \boldsymbol{u}^{n+1})\right] = \frac{\partial \boldsymbol{A}^{i+1}}{\partial \boldsymbol{x}^{i}} \tag{4.4}$$

Tendo em vista que a derivada da Equação (4.4) não é obtida facilmente no simulador, utiliza-se a relação expressa na Equação (4.5), dada em (Fragoso, 2014).

$$\frac{\partial \boldsymbol{A}^{i+1}}{\partial \boldsymbol{x}^{i}} = -\frac{\Delta t^{i}}{\Delta t^{i+1}}\frac{\partial \boldsymbol{A}^{i}}{\partial \boldsymbol{x}^{i}} \tag{4.5}$$

Analogamente, para a derivada do resíduo em relação ao vetor de controles $\boldsymbol{u}^{n+1}$, como somente o termo de fonte/sumidouro possui dependência em relação aos controles, então a derivada $\boldsymbol{\partial g}^{i+1}\big/_{\boldsymbol{\partial u}^{i+1}}$ é dada pela Equação (4.6).

$$\frac{\boldsymbol{\partial g}^{i+1}}{\boldsymbol{\partial u}^{i+1}} = \frac{\boldsymbol{\partial Q}^{i+1}}{\boldsymbol{\partial u}^{i+1}} \tag{4.6}$$

Como o resíduo $\boldsymbol{g}^{n+1} = 0$, a Equação (4.2) pode ser reescrita na forma da Equação (4.7), que representa a equação do TPWL expressa em (He, 2010).

$$\boldsymbol{J}^{i+1}\left(\boldsymbol{x}^{n+1} - \boldsymbol{x}^{i+1}\right) = -\left[\frac{\boldsymbol{\partial A}^{i+1}}{\boldsymbol{\partial x}^{i}}\left(\boldsymbol{x}^{n} - \boldsymbol{x}^{i}\right) + \frac{\boldsymbol{\partial Q}^{i+1}}{\boldsymbol{\partial u}^{i+1}}\left(\boldsymbol{u}^{n+1} - \boldsymbol{u}^{i+1}\right)\right] \tag{4.7}$$

Observa-se que a Equação (4.7) relaciona o desvio entre o vetor de estados simulado pelo TPWL e o vetor salvo durante a simulação de treinamento $\left(\boldsymbol{x}^{n+1} - \boldsymbol{x}^{i+1}\right)$ com os desvios entre os vetores de estados simulados e gravados para o passo de tempo anterior $\left(\boldsymbol{x}^{n} - \boldsymbol{x}^{i}\right)$ e os desvios entre os vetores dos controles aplicados na simulação TPWL e na simulação de treinamento $\left(\boldsymbol{u}^{n+1} - \boldsymbol{u}^{i+1}\right)$. Note também que o TPWL elimina a necessidade de um processo iterativo para a solução do sistema. Logo, explicitando o vetor de estados $\boldsymbol{x}^{n+1}$ da Equação (4.7), obtém-se a Equação (4.8).

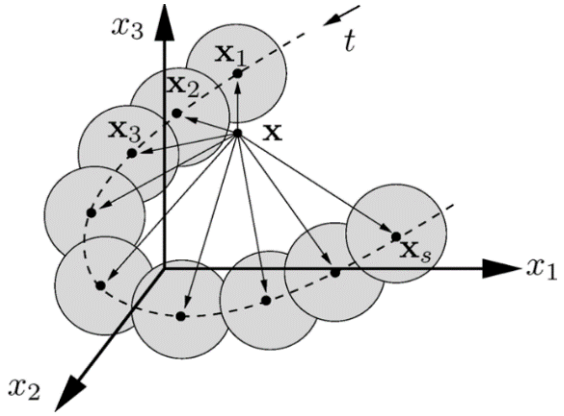
$$\boldsymbol{x}^{n+1} = \boldsymbol{x}^{i+1} - \left(\boldsymbol{J}^{i+1}\right)^{-1}\left[\frac{\boldsymbol{\partial A}^{i+1}}{\boldsymbol{\partial x}^{i}}\left(\boldsymbol{x}^{n} - \boldsymbol{x}^{i}\right) + \frac{\boldsymbol{\partial Q}^{i+1}}{\boldsymbol{\partial u}^{i+1}}\left(\boldsymbol{u}^{n+1} - \boldsymbol{u}^{i+1}\right)\right] \tag{4.8}$$

Assim, dada a solução no passo de tempo n $(\boldsymbol{x}^{n})$, representa-se $\boldsymbol{x}^{n+1}$ na forma da Equação (4.8). Por definição, o estado mais próximo de $\boldsymbol{x}^{n}$ é $\boldsymbol{x}^{i}$, segundo a norma Euclidiana, considerando apenas as saturações de água, pois a saturação varia mais significativamente que a pressão, visto que a primeira é descrita por uma equação de conservação hiperbólica ao passo que esta última é descrita por uma equação elíptica, para casos incompressíveis, e, portanto, a acurácia da solução pelo TPWL é mais sensível às saturações utilizadas durante a linearização (Cardoso, 2009). A partir de $\boldsymbol{x}^{i}$, determina-se o estado consecutivo na simulação de treinamento que será $\boldsymbol{x}^{i+1}$ e suas derivadas correspondentes.

Quanto mais próximo for o vetor de estados simulado do vetor de estados gravado mais próximo, melhor será a linearização, que, por sua vez, é dependente da trajetória, ou seja, dos estados pelos quais passou anteriormente. Logo, haverá, em

torno de cada *snapshot* da trajetória de treinamento, uma região de validade para o modelo linearizado, dentro da qual a aproximação é boa. A Figura 15 ilustra a região de validade do TPWL em torno da trajetória de treinamento.

Figura 15 – Região de validade do TPWL em torno da trajetória de treinamento.



Fonte: (Fragoso, 2014).

Seja então uma sequência de controles (neste caso, pressões de fundo de poço), dado por $\{\boldsymbol{BHP}^t\}_{t=1}^{n_{cc}}$, onde n_{cc} é o número de ciclos de controle ao longo do tempo de simulação T para os quais se deseja calcular as vazões nos poços, usando uma simulação do modelo TPWL. Realiza-se uma ou mais simulações de treinamento, que passa por N_t passos de tempo. Em seguida, são exportados e armazenados os incrementos de tempo de simulação $\{\Delta t^i\}_{i=1}^{N_t}$, os controles utilizados $\{\boldsymbol{u}^i\}_{i=1}^{N_t}$, os estados convergidos $\{\boldsymbol{x}^i\}_{i=1}^{N_t}$ e, por fim, as matrizes de derivadas convergidas, dadas por $\{\boldsymbol{J}^{i+1}\}_{i=1}^{N_t}$, $\left\{\boldsymbol{\partial A}^{i+1}/\boldsymbol{\partial x}^i\right\}_{i=1}^{N_t}$ e $\left\{\boldsymbol{\partial Q}^{i+1}/\boldsymbol{\partial u}^{i+1}\right\}_{i=1}^{N_t}$.

Em resumo, o algoritmo do TPWL consiste na execução dos seguintes passos:

1. Faz-se $n = i = 1$, $t^n = 0$ e $x^n = x^i$;

2. Determina-se o vetor de controle do passo de tempo $\boldsymbol{u}^{n+1}$ a partir da lista de controles a serem simulados $\{\boldsymbol{BHP}^t\}_{t=1}^{n_{cc}}$. No caso de poços com mais de uma completação, faz-se necessário calcular a pressão nas células superiores à de fundo pelo gradiente hidrostático dado pela Equação (2.23);

3. Busca-se entre todos os estados gravados $\{\boldsymbol{x}^i\}_{i=1}^{N_t}$ qual o mais próximo de $\boldsymbol{x}^n$, considerando apenas as saturações de água. Determinado o índice i, seleciona-

se os estados gravados $\boldsymbol{x}^i$ e $\boldsymbol{x}^{i+1}$, e o intervalo de tempo Δt^i correspondente entre os passos de tempo i e $i+1$;

4. Selecionam-se as derivadas $\boldsymbol{J}^{i+1}$, $\boldsymbol{\partial A}^{i+1}/_{\boldsymbol{\partial x}^i}$ e $\boldsymbol{\partial Q}^{i+1}/_{\boldsymbol{\partial u}^{i+1}}$;

5. Calcula-se o vetor de estados $\boldsymbol{x}^{n+1}$ através da Equação (4.8) com os dados selecionados e o estado anterior $\boldsymbol{x}^n$;

6. Determina-se o tempo transcorrido até então na simulação por $t^{n+1} = t^n + \Delta t^i$ e atualiza-se o vetor de estados $\boldsymbol{x}^n \leftarrow \boldsymbol{x}^{n+1}$;

7. Calculam-se as vazões nos poços aplicando-se as pressões das células onde há completação, calculadas em $\boldsymbol{x}^{n+1}$, pelo modelo de poço da Equação (2.21);

8. Caso $t^{n+1} < T$, retorna-se ao passo 2. Caso contrário, a simulação se encerra.

Note que, até então, o problema possui a mesma dimensão da simulação original. A redução da dimensionalidade é realizada com a aplicação da técnica *Proper Orthogonal Decomposition*, descrita na Seção 4.3.

4.2 Exportação de Mapas de Estados e suas Derivadas

Na seção anterior, viu-se que o TPWL transforma a solução de um sistema de equações não-lineares que modela o problema de escoamento em uma sequência de sistemas lineares, a partir do armazenamento dos mapas de pressão e saturação de água em cada passo de tempo da simulação de treinamento, que juntos constituirão os vetores de estado $\boldsymbol{x}^i$, e das matrizes de derivadas convergidas em todos os passos de tempo, $\boldsymbol{J}^{i+1}$, $\boldsymbol{\partial A}^{i+1}/_{\boldsymbol{\partial x}^i}$ e $\boldsymbol{\partial Q}^{i+1}/_{\boldsymbol{\partial u}^{i+1}}$.

Para a exportação e armazenamento dos estados e das matrizes de derivadas, foi necessário realizar modificações em partes do código-fonte do MRST e criar algumas rotinas específicas, de modo que tais elementos pudessem ser carregados posteriormente durante a simulação TPWL. Devido a essa necessidade de serem alteradas classes e funções que compõem a estrutura do MRST, a construção do modelo TPWL/POD consiste numa abordagem intrusiva. Nada referente ao cálculo do simulador foi modificado, embora um estudo detalhado da sua implementação foi necessário a fim de que todas as variáveis necessárias pudessem ser acessadas e

exportadas. Os mapas de pressão e saturação são calculados a cada iteração pelo Método de Newton-Raphson, sendo armazenados dentro da estrutura `states` na saída da função `simulateScheduleAD` no MRST. Definiu-se assim o vetor $\boldsymbol{x}$, cujas primeiras linhas correspondem as pressões das células enquanto que as linhas subsequentes referem-se às saturações de água, conforme descrito na Equação (2.25).

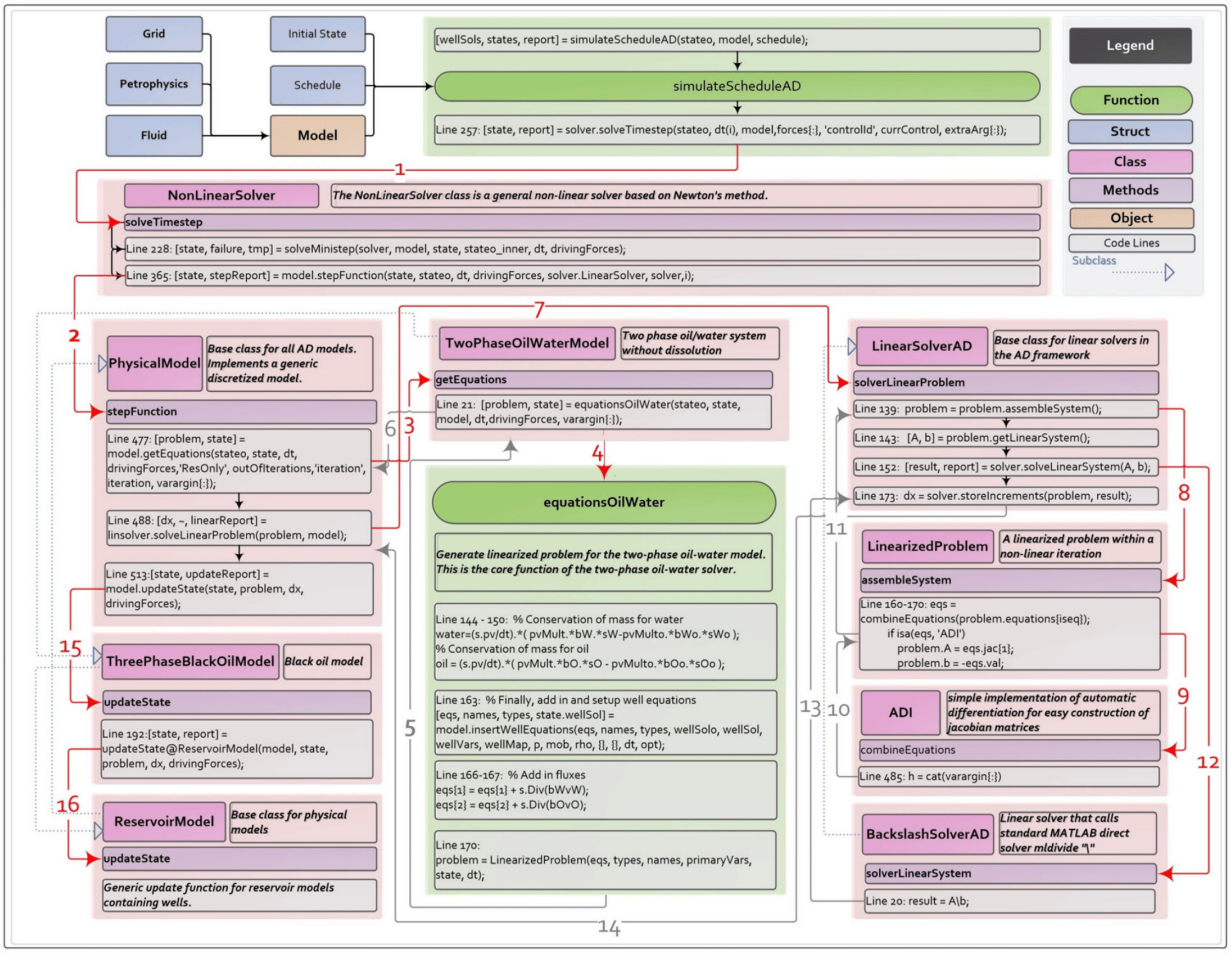
Analogamente, faz-se a exportação dos mapas de controle, que, no caso deste trabalho, foram definidos com as pressões de fundo especificadas em cada poço, armazenados dentro da estrutura `wellSols`, que também consiste num parâmetro de saída da função `simulateScheduleAD` no MRST. Vale ressaltar que o peso da coluna de fluido pode ser obtido diretamente pelo simulador, estando armazenado na variável `wellSols.cdp`, e, portanto, pode ser utilizado para se obter as pressões em todas as células onde há perfuração.

Já para a exportação das matrizes de derivadas, foi necessário acrescentar linhas no código de algumas classes e funções do MRST. Todo o trabalho de alteração do código do MRST foi desenvolvido em (Castro, 2020) e exigiu pleno conhecimento da implementação do código-fonte original e do entendimento da sequência de execução de funções ao longo da simulação, como encontra-se esquematizado na Figura 16.

No total, duas classes e uma função que compõem o código-fonte do MRST foram modificadas a fim de que as matrizes de derivadas, a serem aplicadas na Equação (4.8), pudessem ser obtidas diretamente via simulador, e, consequentemente, armazenadas. Para isso, na classe `PhysicalModel`, o campo `problem` foi incorporado, como destacado em vermelho na Figura 17, na saída da estrutura `report`, terceiro *output* da função `simulateScheduleAD`. O acesso às matrizes se dá então pela variável `problem.matrices` definida como propriedade da classe `LinearizedProblem`.

Através da função `EquationsOilWater`, são salvas as matrizes de derivadas para cada termo referente à Equação (4.2), isto é, para os termos de acumulação, de fluxo e de fonte/sumidouro, incorporando-as na estrutura `problem`.

Figura 16 – Esquema referente ao processo interno de chamada de funções e classes durante a realização de uma simulação utilizando o MRST.



Fonte: (Castro, 2020)

A matriz A, declarada na variável `problem.A` da Figura 18, representa a matriz Jacobiana total do sistema de equações definido durante uma simulação no MRST, gerada por Derivação Automática, conforme já ilustrado na Figura 12, para o caso trifásico. A Figura 19, por sua vez, detalha a matriz A, para o caso bifásico, indicando as derivadas correspondentes para cada “bloco” do Jacobiano.

A matriz Jacobiana $\boldsymbol{J}^{i+1}$ da Equação (4.3), por exemplo, corresponde à parcela Reservatório/Reservatório (R/R) da matriz da Figura 19, e, por definição, consiste na soma das derivadas dos termos de fluxo, de acumulação e de poço em relação às variáveis pressão e saturação de água.

Figura 17 – Métodos da classe `PhysicalModel`, nos quais é incorporada a estrutura `problem` na saída do `report`, *output* da função `simulateScheduleAD`.

```
function [state, report] = stepFunction(model, state, state0, ...
dt, drivingForces, linsolver, nonlinsolver, iteration, varargin)

report = model.makeStepReport('LinearSolver', linearReport, ...
                     'UpdateState', updateReport, ...
                       'StabilizeReport', stabilizeReport, ...
                       'ResidualsConverged', convergence, ...
                       'problem', problem.assembleSystem());
```

```
function report = makeStepReport(model, varargin)

report = struct('LinearSolver', [], ...

                'UpdateState', [], ...

                'Failure', false, ...

                ...

                'StabilizeReport', [], ...

                'ResidualsConverged', [], ...

                'problem', []);
```

Fonte: O Autor (2019).

Figura 18 – Campos da estrutura `problem` definida na classe `LinearizedProblem`.

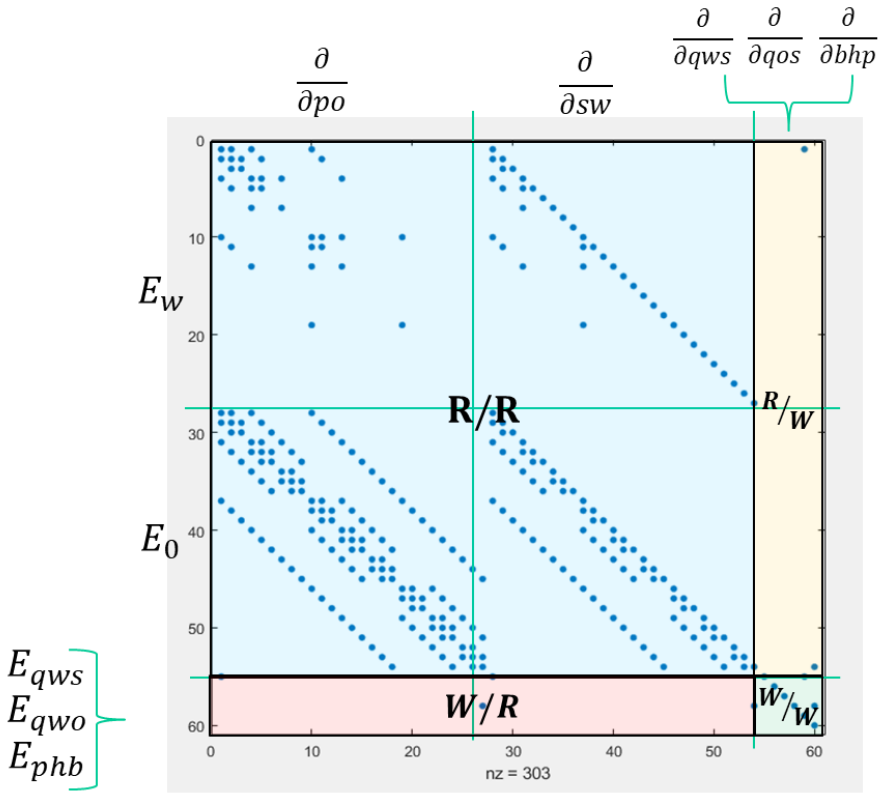
```
eqs = combineEquations(problem.equations{iseq});

if isa(eqs, 'ADI')

               problem.A = eqs.jac{1};

               problem.b = -eqs.val;

           else

             ...
```

Fonte: O Autor (2019).

É importante ressaltar que devido à natureza do método de diferenças finitas e por se tratar de uma malha estruturada, esta matriz, no caso tridimensional, é heptadiagonal por blocos. A exceção ocorre quando há conexão entre células não-vizinhas, que acontece em casos de *pinch-out* em que uma camada desaparece, havendo a necessidade de conectar duas camadas não-consecutivas ou no caso de falhas, onde parte do reservatório se desloca para cima ou para baixo.

Por ser uma matriz esparsa, não há a necessidade de exportar todos os seus elementos. Além disso, como no problema aqui em questão, só se admite controle dos poços por BHP, esta matriz não possui as linhas e as colunas correspondentes às equações de poço, e seus elementos não-nulos são sempre os mesmos ao longo do tempo. Identificadas as matrizes, faz-se então a exportação e o seu armazenamento, a fim de aplicá-las posteriormente na Equação (4.8).

Figura 19 – Esquema da matriz A, do sistema $Ax = b$.



Fonte: O Autor (2019).

4.3 *Proper Orthogonal Decomposition*

Como visto na seção anterior, o TPWL reduz a complexidade numérica do problema a partir da sua linearização em torno de estados previamente convergidos e armazenados durante a simulação de treinamento, o que elimina a necessidade de um processo iterativo como o Método de Newton para a solução das equações em cada passo de tempo, e, portanto, transforma o problema de simulação em uma sequência de sistemas lineares.

A técnica *Proper Orthogonal Decomposition* (POD), também conhecida em outras áreas de estudo como *Principal Component Analysis* (PCA), é responsável por promover a redução da dimensionalidade do problema, isto é, diminui a quantidade de variáveis e de equações, criando assim um modelo de ordem reduzida com a consequente redução no esforço computacional requerido.

Neste trabalho, combinou-se a técnica TPWL ao POD a fim de serem reduzidas tanto a complexidade numérica quanto a dimensionalidade do problema a ser resolvido. O primeiro trabalho a aplicar esse esquema à simulação de reservatórios foi (Cardoso, 2009).

O POD tira proveito da correlação existente entre elementos de um mesmo conjunto de dados, podendo ser aplicado em diversos ramos, não só da engenharia, como também em problemas de reconhecimento automático de faces humanas, que apesar de todas diferirem entre si, há características comuns a todas elas, ou seja, dois olhos, orelhas, testa, boca etc. Logo, essas imagens possuem correlação entre si, o que sugere que o espaço gerado por elas, admitindo-se uma certa tolerância, tem dimensão inferior ao número total de imagens consideradas.

Analogamente, mapas de pressão e saturação obtidos por simulação de reservatórios apresentam correlações similares, pois ambas as variáveis seguem uma tendência fortemente dependente das características do meio simulado. Para isso, é necessário descobrir as direções principais do espaço gerado por esses mapas, considerando apenas as direções mais significativas com base em um critério de energia. Vale ressaltar que o procedimento descrito a seguir foi aplicado

separadamente para os estados de pressão e de saturação de água, por estas grandezas apresentarem naturezas bastante distintas.

A premissa básica consiste na análise espectral da matriz de covariância, cujos autovalores indicam a variabilidade em cada uma das direções principais, que, por sua vez, são seus autovetores. Contudo, devido ao custo inerente ao cálculo da matriz de covariância e de seus autovalores e autovetores, será utilizada uma técnica de fatoração de matrizes bastante comum na Álgebra Linear, conhecido por Decomposição em Valores Singulares (*Singular Value Decomposition,* SVD).

4.3.1 Formulação do POD

Seja um sistema de equação de dimensão $2N_c$ a ser resolvido pelo simulador e descrito pela Equação (4.1), onde N_c é o número de células da malha, e um conjunto de $\mathcal{S}$ soluções desse sistema, ou seja, vetores de dimensão $2N_c$, $\{x_i\}_{i=1}^{\mathcal{S}}$, denominados *snapshots*, gerados a partir de simulação numérica. Entende-se por *snapshot* como um par de vetores $\mathbf{x}_p$ e $\mathbf{x}_s$, contendo os valores de pressão e saturação de água de todas as N_c células do modelo, para um dado passo de tempo.

Considere então $\mathbf{X}_p$ e $\mathbf{X}_s$ como sendo as matrizes que contém em suas colunas todos os *snapshots* obtidos ao longo dos passos de tempo, definidas na Equação (4.9).

$$\mathbf{X}_p = \left[\mathbf{x}_p^1, \mathbf{x}_p^2, \mathbf{x}_p^3, \dots, \mathbf{x}_p^{\mathcal{S}}\right], \qquad \mathbf{X}_s = \left[\mathbf{x}_s^1, \mathbf{x}_s^2, \mathbf{x}_s^3, \dots, \mathbf{x}_s^{\mathcal{S}}\right] \tag{4.9}$$

Devido à diferença de magnitude entre os valores de pressão e saturação, os estados dever ser normalizados separadamente, conforme a Equação (4.10).

$$x_{p_i} = \frac{(p_i - p_{min})}{(p_{max} - p_{min})}, \qquad x_{s_i} = \frac{(s_i - s_{min})}{(s_{max} - s_{min})} \tag{4.10}$$

A redução da dimensionalidade é aplicada separadamente para os estados de pressão e saturação de água, visto que essas duas variáveis possuem naturezas bastante distintas. Logo, o procedimento descrito a seguir é feito tanto para $\mathbf{X}_p$ como para $\mathbf{X}_s$, denotados a partir de agora como matriz $\mathbf{X}$, representada pela Equação (4.11) de dimensão N_c x $\mathcal{S}$, cujas colunas representam cada um dos $\mathcal{S}$ *snapshots*.

$$\mathbf{X} = \left[\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \dots, \mathbf{x}_{\mathcal{S}}\right] \tag{4.11}$$

O objetivo do POD é determinar v vetores próprios ortonormais, $\{\phi_i\}_{i=1}^{v}$, de tal modo que a distância entre os *snapshots* e sua projeção no subespaço gerado por esses vetores ortonormais, expressa na Equação (4.12), seja mínima para um dado v.

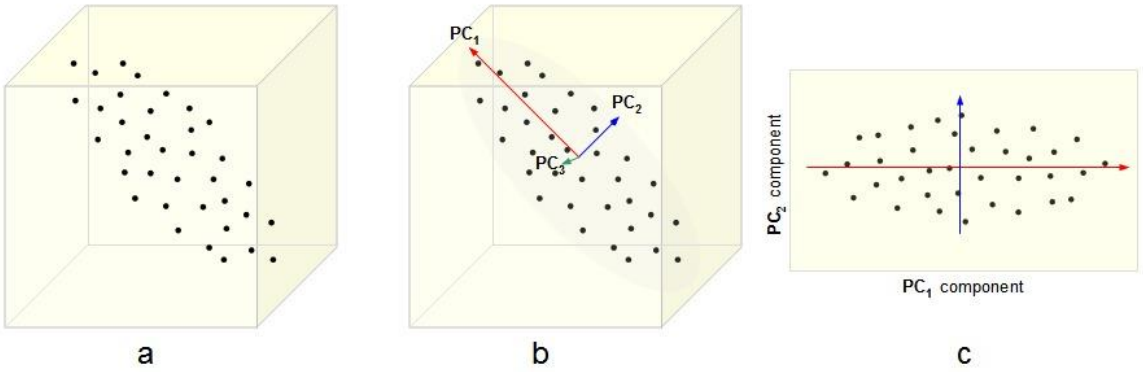
$$\mathrm{Q}(\varphi) = \frac{1}{\mathcal{S}}\sum_{\mathrm{i}=1}^{\mathcal{S}}\left\|\mathbf{x}_{\mathrm{i}} - \varphi\varphi^{\mathrm{T}}\mathbf{x}_{\mathrm{i}}\right\| \tag{4.12}$$

A solução deste problema de minimização é dada por $\{\phi_i\}_{i=1}^{v}$, considerando v autovetores da matriz N_c x N_c dada pela Equação (4.13), que representa a covariância entre os *snapshots*. Tais autovetores correspondem aos v maiores autovalores $\{\lambda_i\}_{i=1}^{v}$.

$$\mathbf{R} = \frac{1}{\mathcal{S}}\mathbf{X}\cdot\mathbf{X}^{\mathrm{T}} \tag{4.13}$$

Determinar os autovetores da matriz de covariância significa descobrir as direções ortogonais em que os dados apresentam maior variabilidade, desprezando as direções em que apresentem menor variabilidade.

Figura 20 – Esquema ilustrativo referente à técnica POD.



Fonte: (Fragoso, 2014)

A fim de ilustrar o funcionamento do POD, tem-se a Figura 20. Considere que os estados de um sistema possam ser representados, num espaço tridimensional, dentro de um elipsoide, como ilustrado na Figura 20a. As três direções principais ortogonais para esse conjunto dados são os autovetores da matriz de covariância e estão representados na Figura 20b. Como o módulo dos autovetores é proporcional aos seus respectivos autovalores, tem-se que as direções de maior variabilidade são aquelas cujo vetor diretor possui maior módulo. Desprezando-se a direção com menor autovalor, ou seja, de menor variabilidade, os pontos do conjunto de dados podem ser

representados de maneira aproximada em um plano no interior de uma elipse, a partir de sua projeção sobre o espaço gerado por esse plano de menor dimensão, conforme ilustrado na Figura 20c.

Entretanto, somente o cálculo da matriz de covariância dos dados por si só é bastante custoso, e junto a isso, a determinação dos autovetores dessa matriz é uma operação que exige grande esforço computacional. Logo, utiliza-se a Decomposição em Valores Singulares (SVD) para a determinação dos autovetores e autovalores da matriz de covariância.

Seja a matriz N_c x $\mathcal{S}$ de *snapshots* de simulação, na qual será aplicada o POD. A sua decomposição em valores singulares consiste na fatoração em um produto de três matrizes, $\mathbf{U}$, $\mathbf{\Sigma}$ e $\mathbf{V}$, representado na Equação (4.14).

$$\mathbf{X} = \mathbf{U} \cdot \mathbf{\Sigma} \cdot \mathbf{V}^{\mathrm{T}} \tag{4.14}$$

$\mathbf{U}$ e $\mathbf{V}$ são ortonormais, de dimensão N_c x N_c e $\mathcal{S}$ x $\mathcal{S}$, respectivamente, enquanto que $\mathbf{\Sigma}$ é uma matriz diagonal N_c x $\mathcal{S}$, cujos elementos são os valores singulares não-negativos σ_j , $j = 1, \dots, \min(N_c, \mathcal{S})$, dispostos em ordem decrescente ao longo da diagonal.

As colunas de $\mathbf{U}$ são os autovalores de $\mathbf{X} \cdot \mathbf{X}^{\mathrm{T}}$ e as colunas de $\mathbf{V}$ são os autovalores de $\mathbf{X}^{\mathrm{T}} \cdot \mathbf{X}$. Além disso, tanto os autovalores de $\mathbf{X} \cdot \mathbf{X}^{\mathrm{T}}$ quanto os de $\mathbf{X}^{\mathrm{T}} \cdot \mathbf{X}$ são iguais aos quadrados dos valores singulares de $\mathbf{X}$.

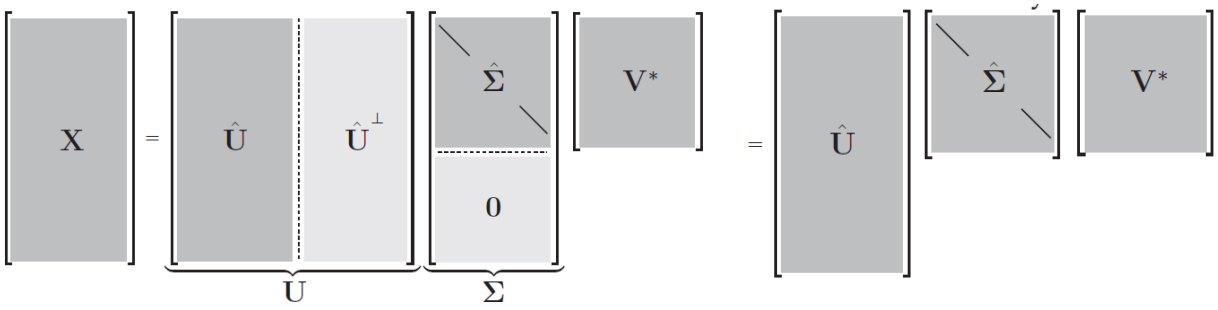
O posto de $\mathbf{X}$ é, por sua vez, igual ao número de valores singulares não-nulos, e portanto, se $rank(\mathbf{X}) = r$, pode-se reescrever a SVD na sua forma reduzida ("econômica"), representada na Equação (4.15) e ilustrada na Figura 21, onde $\widehat{\mathbf{U}}$ corresponde às primeiras r colunas de $\mathbf{U}$, e $\widehat{\mathbf{\Sigma}}$ é uma matriz diagonal de dimensão r x r. Essa forma de representação reduz o espaço alocado em memória ao longo da operação.

$$\mathbf{X} = \mathbf{U\Sigma V}^{*} = [\widehat{\mathbf{U}} \quad \widehat{\mathbf{U}}^{\perp}] \begin{bmatrix} \widehat{\mathbf{\Sigma}} \\ \mathbf{0} \end{bmatrix} \mathbf{V}^{\mathbf{T}} \tag{4.15}$$

Como mencionado anteriormente, e utilizando-se a SVD descrita na Equação (4.14), a matriz $\widehat{\mathbf{U}}$ é a matriz de todos os autovalores de $\mathbf{X} \cdot \mathbf{X}^{\mathrm{T}}$. Sendo v, o número de direções principais utilizadas na aproximação do POD, onde $v < \mathcal{S} \ll N_c$, monta-se então a matriz $\widetilde{\mathbf{U}}$, considerando somente as v primeiras colunas de $\widehat{\mathbf{U}}$. As colunas,

ordenadas em ordem decrescente dos autovalores associados, correspondem aos maiores autovalores, e, portanto, às direções de maior variabilidade (ou de maior energia).

Figura 21 – Esquema de matrizes na SVD completa e econômica.

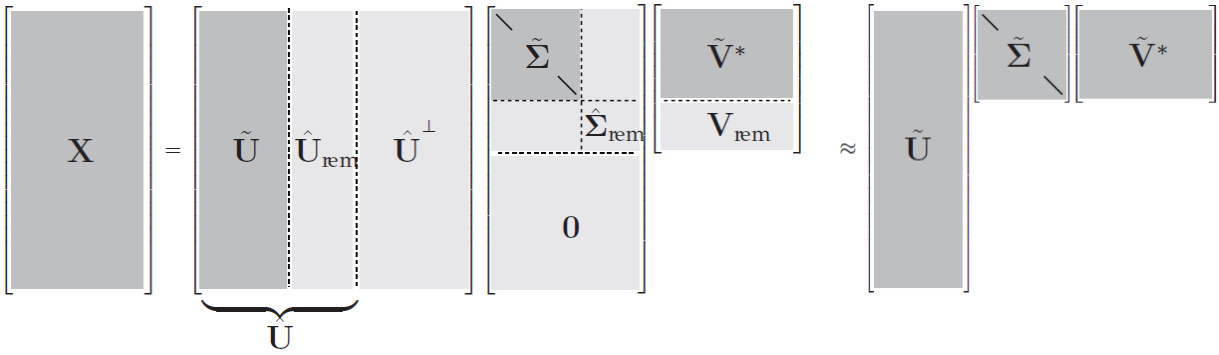


Fonte: BRUNTON, Steven L. (2019)

Assim, $\tilde{\mathbf{U}}$ constitui a base do POD, ou seja, consiste no subespaço gerado pelos autovetores $\{\phi_i\}_{i=1}^{v}$ definido pela matriz $\boldsymbol{\phi}$, de dimensão $2N_c$ x v, considerando agora as duas variáveis primárias, onde, $v = v_p + v_s$. A matriz $\boldsymbol{\phi}$ encontra-se representada na Equação (4.16), e na Figura 22, ilustra-se a SVD de modo que o subscrito rem indica o restante (*remainder*) de $\hat{\mathbf{U}}$, $\hat{\mathbf{\Sigma}}$ ou $\mathbf{V}$ após o truncamento.

$$\mathbf{U}_v = \boldsymbol{\phi} = [\phi_1 \quad \phi_2 \quad \ldots \quad \phi_v] \tag{4.16}$$

Figura 22 – Diagrama esquemático da SVD, após o truncamento.



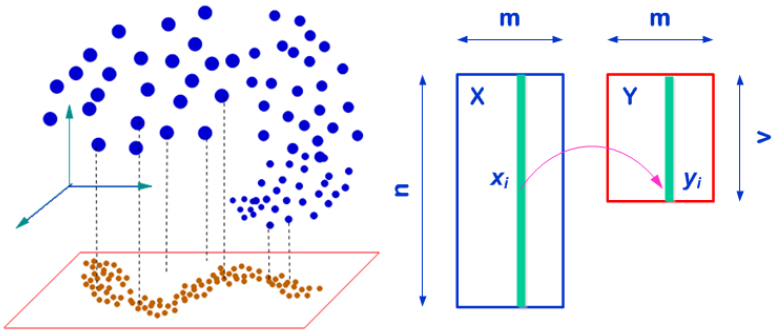
Fonte: BRUNTON, Steven L. (2019)

Visto que são realizadas duas decomposições em separado com os mapas normalizados de pressão e os de saturação de água, obtém-se duas matrizes $\boldsymbol{\phi}_p$ e $\boldsymbol{\phi}_s$, o que permite a escolha de um número diferente de autovalores para cada grandeza.

Combinando as matrizes $\boldsymbol{\phi}_p$ e $\boldsymbol{\phi}_s$, obtém-se a base do POD que é utilizada para redução da dimensão dos mapas de pressão e saturação de água como projeção dos estados no espaço reduzido. Tal projeção é representada pela matriz $\boldsymbol{\phi}$ e gera estados reduzidos $\{z_i\}_{i=1}^{v}$ correspondentes a $\{x_i\}_{i=1}^{v}$, definido na Equação (4.17) e ilustrado na Figura 23.

$$\mathbf{z} = \boldsymbol{\phi}^{\mathrm{T}} \cdot \mathbf{x} \tag{4.17}$$

Figura 23 – Projeção de um conjunto de dados de um espaço X, de dimensão $m \times n$, em um subespaço Y, de dimensão $m \times v$.



Fonte: Prof. Bernardo Horowitz.

Enfim, a redução da dimensionalidade através do POD pode ser aplicada diretamente à Equação (4.8), resultando na Equação (4.24), vista mais adiante, cuja relação entre os *snapshots* armazenados e os estados reduzidos é definida nas Equações (4.18) e (4.19), respectivamente.

$$\begin{bmatrix} x_{p_1} \\ x_{p_2} \\ \vdots \\ x_{p_{N_c}} \\ x_{s_1} \\ x_{s_2} \\ \vdots \\ x_{s_{N_c}} \end{bmatrix} \approx \begin{bmatrix} \phi_{p_1}^{1} & \cdots & \phi_{p_1}^{v_p} & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & 0 & \cdots & 0 \\ \phi_{p_{N_c}}^{1} & \cdots & \phi_{p_{N_c}}^{v_p} & 0 & \cdots & 0 \\ 0 & \cdots & 0 & \phi_{s_1}^{1} & \cdots & \phi_{s_1}^{v_s} \\ 0 & \cdots & 0 & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \phi_{s_{N_c}}^{1} & \cdots & \phi_{s_{N_c}}^{v_s} \end{bmatrix} \cdot \begin{bmatrix} z_{p_1} \\ \vdots \\ z_{p_{v_p}} \\ z_{s_1} \\ \vdots \\ z_{s_{v_s}} \end{bmatrix} \tag{4.18}$$

$$\mathbf{x} = \boldsymbol{\phi} \cdot \mathbf{z} \tag{4.19}$$

4.3.2 Esquema *Local Resolution*

O esquema *Local Resolution* (LR) foi desenvolvido com objetivo de melhorar a acurácia do modelo TPWL/POD referente a células de importância significativa dentro

do reservatório. Uma nova base $\boldsymbol{\phi}_{LR}$ é criada a fim de reter toda a informação original das células onde há completação nos poços e, assim, minimizar o erro. A Equação (4.18), portanto, terá a forma da Equação (4.20).

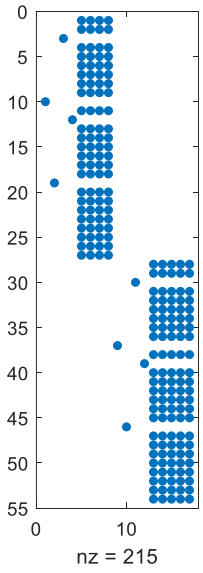
$$\boldsymbol{x} = \mathbf{P}\begin{bmatrix}\boldsymbol{x}_{LR}\\ \boldsymbol{x}_{G}\end{bmatrix} \approx \mathbf{P}\begin{bmatrix}\boldsymbol{\phi}_{LR} & 0\\ 0 & \boldsymbol{\phi}_{G}\end{bmatrix}\begin{bmatrix}\boldsymbol{z}_{LR}\\ \boldsymbol{z}_{G}\end{bmatrix} \tag{4.20}$$

Onde $\mathbf{P}$ é a matriz de permutação construída a fim de isolar as células dos poços onde há completação, $\boldsymbol{x}_{LR}$ é o vetor de estados para essas células, $\boldsymbol{x}_{G}$ é o vetor de estados para o restante das células do reservatório, $\boldsymbol{\phi}_{LR}$ e $\boldsymbol{\phi}_{G}$ são suas respectivas bases, e $\boldsymbol{z}_{LR}$ e $\boldsymbol{z}_{G}$ são os respectivos vetores de estados reduzidos. A matriz de base modificada $\boldsymbol{\phi}^{*}$ é então definida pela Equação (4.21).

$$\boldsymbol{\phi}^{*} = \mathbf{P}\begin{bmatrix}\boldsymbol{\phi}_{LR} & 0\\ 0 & \boldsymbol{\phi}_{G}\end{bmatrix} \tag{4.21}$$

A Figura 24 ilustra a base geral do POD para o exemplo de modelo de reservatório implementado no Capítulo 3, ilustrado na Figura 8, composto apenas por 27 células, contendo os blocos referentes à pressão, no canto superior esquerdo, e à saturação de água, no canto inferior direito. Na implementação deste exemplo, foram definidos dois poços, cada um com duas completações, resultando no total de quatro células onde há completação. Suas informações então são isoladas das demais células do modelo, para cada variável primária, como se pode observar na Figura 24.

Figura 24 – Base do POD, para um sistema composto por apenas 27 células.



Fonte: O Autor (2019).

4.4 Acoplamento TPWL/POD

A redução de dimensionalidade pode ser então aplicada ao TPWL, ao introduzir os estados reduzidos $\mathbf{z}$ nas equações de fluxo na sua forma discreta, expressa pela Equação (4.2). Isso permite a solução de um sistema com v incógnitas ao invés de $2N_c$ incógnitas. Inserindo a Equação (4.19) na Equação (4.2), obtém-se a Equação (4.22).

$$\mathbf{g} = \mathbf{F}^{i+1}\left(\boldsymbol{\phi}\cdot\mathbf{z}^{i+1}\right) + \mathbf{A}^{i+1}\boldsymbol{\phi}\cdot\left(\mathbf{z}^{i+1}-\mathbf{z}^{i}\right) + \mathbf{Q}^{i+1} \tag{4.22}$$

Pré-multiplicando a Equação (4.22) por $\boldsymbol{\phi}^{\mathrm{T}}$, tem-se a Equação (4.23).

$$\mathbf{g}_r = \boldsymbol{\phi}^{\mathbf{T}}\cdot\mathbf{g} = \boldsymbol{\phi}^{\mathbf{T}}\cdot\mathbf{F}^{i+1}\left(\boldsymbol{\phi}\cdot\mathbf{z}^{i+1}\right) + \boldsymbol{\phi}^{\mathbf{T}}\cdot\mathbf{A}^{i+1}\boldsymbol{\phi}\cdot\left(\mathbf{z}^{i+1}-\mathbf{z}^{i}\right) + \boldsymbol{\phi}^{\mathbf{T}}\cdot\mathbf{Q}^{i+1} \tag{4.23}$$

A fim de isolar o termo $\boldsymbol{z^{n+1}}$, procede-se da mesma forma descrita na Seção 4.1, resultando na Equação (4.24), análoga à expressão do TPWL, dada anteriormente pela Equação (4.2), após a projeção dos estados em um subespaço de menor dimensão.

$$\boldsymbol{z^{n+1}} = \boldsymbol{z^{i+1}} - (\boldsymbol{J}^{i+1})_r^{-1}\left[\left(\frac{\partial \boldsymbol{A}^{i+1}}{\partial \boldsymbol{x}^{i}}\right)_r\left(\boldsymbol{z^{n}}-\boldsymbol{z^{i}}\right) + \left(\frac{\partial \boldsymbol{Q}^{i+1}}{\partial \boldsymbol{u}^{i+1}}\right)_r\left(\boldsymbol{u^{n+1}}-\boldsymbol{u^{i+1}}\right)\right] \tag{4.24}$$

Contudo, há a necessidade de serem calculadas as derivadas com dimensões reduzidas $(\boldsymbol{J}^{i+1})_r$, $\left(\partial \boldsymbol{A}^{i+1}/_{\partial \boldsymbol{x}^{i}}\right)_r$ e $\left(\partial \boldsymbol{Q}^{i+1}/_{\partial \boldsymbol{u}^{i+1}}\right)_r$, definidas na Equação (4.25).

$$\begin{cases} (\boldsymbol{J}^{i+1})_r = \boldsymbol{\phi}^{\mathrm{T}}\cdot\mathbf{J}^{i+1}\cdot\boldsymbol{\phi} \\ \left(\dfrac{\partial \boldsymbol{A}^{i+1}}{\partial \boldsymbol{x}^{i}}\right)_r = \boldsymbol{\phi}^{\mathrm{T}}\cdot\left(\dfrac{\partial \boldsymbol{A}^{i+1}}{\partial \boldsymbol{x}^{i}}\right)\cdot\boldsymbol{\phi} \\ \left(\dfrac{\partial \boldsymbol{Q}^{i+1}}{\partial \boldsymbol{u}^{i+1}}\right)_r = \boldsymbol{\phi}^{\mathrm{T}}\cdot\left(\dfrac{\partial \boldsymbol{Q}^{i+1}}{\partial \boldsymbol{u}^{i+1}}\right) \end{cases} \tag{4.25}$$

O procedimento acima descrito é conhecido como projeção de Bubnov-Galerkin e representa uma projeção ortogonal no espaço de estados reduzido, aplicada em (Cardoso, 2009) e (He, 2010). Esse tipo de projeção consiste em pré-multiplicar as derivadas pela base transposta $\boldsymbol{\phi}^{\mathrm{T}}$, e, portanto, é uma projeção ortogonal no espaço reduzido. Entretanto, problemas relativos à estabilidade do método são relatados em (He & Durlosfky, 2013) e (Gildin et al., 2013), levando à aplicação neste trabalho de um outro tipo de projeção: a de Petrov-Galerkin.

A projeção de Petrov-Galerkin foi relatada como a responsável pela estabilização do método em (Carlberg et al., 2009) e (He & Durlosfky, 2013), sendo equivalente a resolver o sistema por mínimos quadrados, pré-multiplicando as derivadas por $\boldsymbol{\phi}^{\mathrm{T}} \cdot \left(\mathbf{J}^{i+1}\right)^{\mathrm{T}}$. Dessa forma, trata-se de uma projeção oblíqua no espaço de dimensão reduzida. A Equação (4.26) apresenta as três matrizes de derivadas com dimensões reduzidas a serem utilizadas na Equação (4.24).

$$\begin{cases} (\boldsymbol{J}^{i+1})_r = \boldsymbol{\phi}^{\mathrm{T}} \cdot \left(\mathbf{J}^{i+1}\right)^{\mathrm{T}} \cdot \mathbf{J}^{i+1} \cdot \boldsymbol{\phi} \\ \left(\dfrac{\partial \boldsymbol{A}^{i+1}}{\partial \boldsymbol{x}^{i}}\right)_r = \boldsymbol{\phi}^{\mathrm{T}} \cdot \left(\mathbf{J}^{i+1}\right)^{\mathrm{T}} \cdot \left(\dfrac{\partial \boldsymbol{A}^{i+1}}{\partial \boldsymbol{x}^{i}}\right) \cdot \boldsymbol{\phi} \\ \left(\dfrac{\partial \boldsymbol{Q}^{i+1}}{\partial \boldsymbol{u}^{i+1}}\right)_r = \boldsymbol{\phi}^{\mathrm{T}} \cdot \left(\mathbf{J}^{i+1}\right)^{\mathrm{T}} \cdot \left(\dfrac{\partial \boldsymbol{Q}^{i+1}}{\partial \boldsymbol{u}^{i+1}}\right) \end{cases} \tag{4.26}$$

É importante destacar que para o cálculo das matrizes reduzidas, é necessário antes normalizar as matrizes $\boldsymbol{J}^{i+1}$, ${\partial \boldsymbol{A}^{i+1}}/{\partial \boldsymbol{x}^{i}}$ e ${\partial \boldsymbol{Q}^{i+1}}/{\partial \boldsymbol{u}^{i+1}}$, como demonstrado a seguir.

Seja $\boldsymbol{x}$ uma variável de estado, onde $x_{max} \leq x_i \leq x_{max}, \forall i \in \mathbb{N}$. A variável $\boldsymbol{y}$ resultante da normalização de $\boldsymbol{x}$ é calculada conforme a Equação (4.27).

$$y_i = \frac{x_i - x_{min}}{x_{max} - x_{min}}, 0 \leq y_i \leq 1 \tag{4.27}$$

Analogamente, considere $\boldsymbol{u}$ o vetor de controles, onde $u_{max} \leq u_i \leq u_{max}, \forall i \in \mathbb{N}$. A variável $\boldsymbol{b}$ resultante da normalização de $\boldsymbol{u}$ é calculada conforme a Equação (4.28).

$$b_i = \frac{u_i - u_{min}}{u_{max} - u_{min}}, 0 \leq u_i \leq 1 \tag{4.28}$$

Seja agora uma matriz $\boldsymbol{M}$, cuja derivada em relação a $\boldsymbol{x}$ deseja-se calcular. Tem-se então que ${\partial \boldsymbol{M}}/{\partial \boldsymbol{x}}$ é dada pela Equação (4.29). Analogamente, para a derivada em relação a $\boldsymbol{u}$, tem-se a Equação (4.30).

$$\frac{\partial \boldsymbol{M}}{\partial \boldsymbol{x}} = \frac{\partial \boldsymbol{M}}{\partial \boldsymbol{y}} \cdot \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}} = \frac{\partial \boldsymbol{M}}{\partial \boldsymbol{y}} \cdot (x_{max} - x_{min})^{-1} \therefore \frac{\partial \boldsymbol{M}}{\partial \boldsymbol{y}} = (x_{max} - x_{min}) \cdot \frac{\partial \boldsymbol{M}}{\partial \boldsymbol{x}} \tag{4.29}$$

$$\frac{\partial \boldsymbol{M}}{\partial \boldsymbol{u}} = \frac{\partial \boldsymbol{M}}{\partial \boldsymbol{b}} \cdot \frac{\partial \boldsymbol{b}}{\partial \boldsymbol{u}} = \frac{\partial \boldsymbol{M}}{\partial \boldsymbol{b}} \cdot (u_{max} - u_{min})^{-1} \therefore \frac{\partial \boldsymbol{M}}{\partial \boldsymbol{b}} = (u_{max} - u_{min}) \cdot \frac{\partial \boldsymbol{M}}{\partial \boldsymbol{u}} \tag{4.30}$$

Sendo assim, para a normalização das matrizes $\boldsymbol{J}^{i+1}$ e $\partial \boldsymbol{A}^{i+1}/\partial \boldsymbol{x}^{i}$, basta aplicar a Equação (4.29), enquanto que para normalizar a matriz $\partial \boldsymbol{Q}^{i+1}/\partial \boldsymbol{u}^{i+1}$, procede-se conforme a Equação (4.30).

O processo de treinamento do TPWL/POD reúne, portanto, a execução de uma ou mais simulações no MRST, armazenando os mapas dos estados e suas derivadas convergidos, seguida pelo cálculo da base do POD, dos mapas de estados reduzidos, e, por fim, o cálculo das derivadas reduzidas.

Neste trabalho, o treinamento foi realizado com uma única simulação no MRST com controles de pressão de fundo (BHP) em ciclos pré-definidos, cujos valores variam aleatoriamente entre os limites máximo e mínimo estabelecidos. A análise do erro da aproximação se dará a partir da mudança desses controles em diferentes casos que serão mais detalhados brevemente. A quantidade de autovalores v_p e v_z considerados na redução da dimensão foi definida empiricamente através de um simples estudo da qualidade da aproximação. A sequência a seguir descreve o algoritmo implementado na realização do TPWL/POD.

1. Aplica-se uma sequência aleatória de controles de pressão de fundo (BHP) a serem aplicadas na simulação de treinamento, a ser executada logo em seguida;

2. Armazenam-se os mapas, as derivadas e os controles utilizados;

3. Leem-se os mapas e montam-se as matrizes normalizadas de *snapshots* de pressão $\mathbf{X}_p$ e saturação de água $\mathbf{X}_p$, usando o esquema *Local Resolution* para as células onde há completação nos poços;

4. Calculam-se as bases $\boldsymbol{\phi}_p$ e $\boldsymbol{\phi}_S$ através da SVD considerando as quantidades de autovalores v_p e v_z, respectivamente. Multiplica-se cada uma dessas bases pela matriz de permutação $\mathbf{P}$, obtendo-se então a base $\boldsymbol{\phi}$;

5. Calculam-se os mapas reduzidos $\boldsymbol{z}^{\boldsymbol{i}}$ através da Equação (4.24);

6. Leem-se as derivadas $\boldsymbol{J}^{i+1}$, $\partial \boldsymbol{A}^{i+1}/\partial \boldsymbol{x}^{i}$ e $\partial \boldsymbol{Q}^{i+1}/\partial \boldsymbol{u}^{i+1}$ exportadas durante a simulação de treinamento;

7. Calculam-se as derivadas reduzidas $(\boldsymbol{J}^{i+1})_r$, $\left(\partial \boldsymbol{A}^{i+1}/\partial \boldsymbol{x}^{i}\right)_r$ e $\left(\partial \boldsymbol{Q}^{i+1}/\partial \boldsymbol{u}^{i+1}\right)_r$ conforme Equação (4.26);

8. Realiza-se a simulação TPWL/POD utilizando-se a Equação (4.8) e o algoritmo descrito na Seção 4.1 com novos controles, ou ainda, caso se queira avaliar a qualidade da aproximação, os mesmos controles utilizados para o treinamento.

4.5 Exemplo de Aplicação: Modelo SPE10

Para o modelo SPE10 proposto e descrito no Capítulo 3, foram realizadas três novas análises para fins de comparação e validação das técnicas até então apresentadas: simulação de alta fidelidade (MRST), simulação TPWL (com apenas redução da complexidade numérica) e a simulação TPWL/POD (com inclusão da redução da dimensionalidade do problema). Designou-se como Caso 1, Caso 2 e Caso 3 tais análises, onde foram aplicadas diferentes sequências aleatórias de BHP's nos poços produtores.

Para a exportação das matrizes e dos estados utilizados na linearização das equações governantes no TPWL, realizou-se apenas uma única simulação de treinamento, composta por 10 ciclos de controle de valores aleatórios aplicados nos poços produtores, no intervalo de 1000 e 4000 psi. O tempo total de simulação foi de 3000 dias. Em todas as simulações, especificou-se para os poços injetores um BHP constante e igual a 8000 psi.

Na simulação do Caso 1, os controles variam entre 2000 e 3000 psi, isto é, um intervalo dentro do especificado na simulação de treinamento, enquanto que, no Caso 2, os controles variam entre 500 e 4500 psi, isto é, fora do intervalo especificado no treinamento. Em ambos os casos, as compressibilidades da rocha e do fluido foram consideradas desprezíveis, e, portanto, considera-se todo o sistema incompressível. No Caso 3, as pressões de fundo nos poços produtores variam entre 1000 e 4000 psi, ou seja, um intervalo igual ao especificado na simulação de treinamento, porém com novos valores estabelecidos na sequência de controles. Neste último caso, admitiu-se uma compressibilidade da rocha e do fluido constantes e iguais a $10^{-6}\ psi^{-1}$, e, portanto, considera-se o sistema pouco compressível.

Os erros entre as curvas de produção de óleo e as curvas de injeção e produção de água, obtidas nas simulações de alta fidelidade e de ordem reduzida, podem ser mensurados usando, respectivamente, as Equações (4.31), (4.32) e (4.33), onde q é a vazão e os índices hf e rom correspondem às simulações de alta fidelidade e de ordem reduzida, respectivamente.

$$E_o = \sum_{n=1}^{N} \left(\frac{\left| q_{o,n}^{hf} - q_{o,n}^{rom} \right|}{q_{o,n}^{hf}} \right) \tag{4.31}$$

$$E_{w,inj} = \sum_{n=1}^{N} \left(\frac{\left| q_{wi,n}^{hf} - q_{wi,n}^{rom} \right|}{q_{wi,n}^{hf}} \right) \tag{4.32}$$

$$E_{w,prod} = \sum_{n=1}^{N} \left(\frac{\left| q_{wp,n}^{hf} - q_{wp,n}^{rom} \right|}{q_{wp,n}^{hf}} \right) \tag{4.33}$$

Em (Fragoso, 2014), foi desenvolvida uma maneira de medir a qualidade da aproximação realizada pelo modelo de ordem reduzida baseada nos estados calculados na simulação. Observa-se, na Equação (4.24), que o modelo linear que constitui o TPWL/POD é baseado na distância entre os estados e controles armazenados previamente durante a simulação de treinamento, e os estados e controles calculados numa nova simulação. O estado reduzido armazenado $\boldsymbol{z^i}$, a ser utilizado no passo de tempo n, é escolhido de forma a minimizar a distância ao último estado $\boldsymbol{z^n}$, calculado na simulação TPWL/POD. Esta distância considerada diz respeito somente a parte do vetor de estados correspondentes às saturações de água. Sendo assim, a medida do erro médio E_{rom} na estimativa de estados realizada pelo modelo de ordem reduzida é definida como o valor médio normalizado dessa distância ao longo do tempo de simulação T, conforme a Equação (4.34). Contudo, a estimativa de erro adotada nas comparações a serem apresentadas a seguir seguirá conforme as Equações (4.31), (4.32) e (4.33), ficando a Equação (4.34) como um mecanismo para avaliar a necessidade de retreinamento do modelo TPWL/POD em trabalhos futuros.

$$E_{rom} = \frac{1}{T \times \sqrt{v_z}} \sum_{n=1}^{N} \left(\left\| \boldsymbol{z^n} - \boldsymbol{z^i} \right\| \times \Delta t_n \right) \tag{4.34}$$

Nas simulações realizadas, obteve-se s*peed-up's* da ordem de 2000, entre o tempo computacional requerido no modelo de alta fidelidade e no modelo TPWL/POD, utilizando um critério de energia mínima de 0,9999, para o qual 17 de 116 autovalores/autovetores foram incluídos durante o processo, para o modelo SPE10. A magnitude dos s*peed-up's* se deve, como já mencionado anteriormente, ao fato de que os simuladores MRST baseados em Derivação Automática são entre duas e dez vezes mais lentos que simuladores comerciais totalmente otimizados (Bao et al., 2017).

Ao contrário do que reporta (He et al., 2013) e (Gildin et al., 2013), nas simulações aqui realizadas, não houve problemas relativos à estabilidade do método TPWL/POD, utilizando a projeção de Bubnov-Galerkin. Além disso, viu-se que os erros nas curvas de injeção e produção resultantes da simulação TPWL/POD em relação ao modelo de alta-fidelidade, utilizando este tipo de projeção, apresentaram erro menor que as simulações que utilizaram a projeção de Petrov-Galerkin. Por esse motivo, serão ilustrados a seguir apenas os resultados obtidos utilizando a projeção de Bubnov-Galerkin.

A seguir, são apresentadas as sequências de controles aplicadas às simulações de alta fidelidade, TPWL e TPWL/POD, para os três casos apresentados anteriormente. Também são comparados os erros relativos às curvas de vazão de injeção e de produção obtidas nos três tipos de simulação, e o tempo computacional requerido para cada caso.

4.5.1 Modelo SPE10: Caso 1

Nas simulações do Caso 1, os controles dos poços produtores variam entre 2000 e 3000 psi, isto é, um intervalo dentro do especificado na simulação de treinamento, distribuído em 10 ciclos de controle, ilustrados na Figura 25. O tempo total de simulação foi de 3000 dias. Em todas as simulações, aplicou-se para os poços injetores um BHP constante e igual a 8000 psi. Além disso, trata-se de um caso incompressível, visto que

as compressibilidades da rocha e do fluido foram consideradas desprezíveis. Os resultados obtidos para este caso são ilustrados ao longo das Figuras 26 a 28.

Figura 25 – Trajetória de BHP aplicada aos poços produtores do modelo SPE10, para o Caso 1.

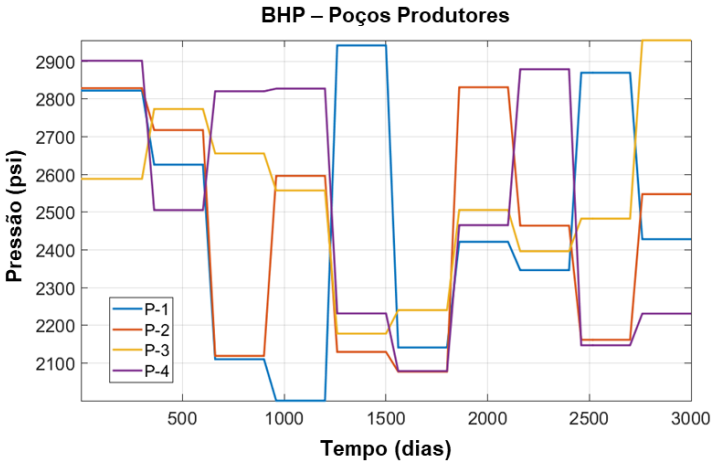
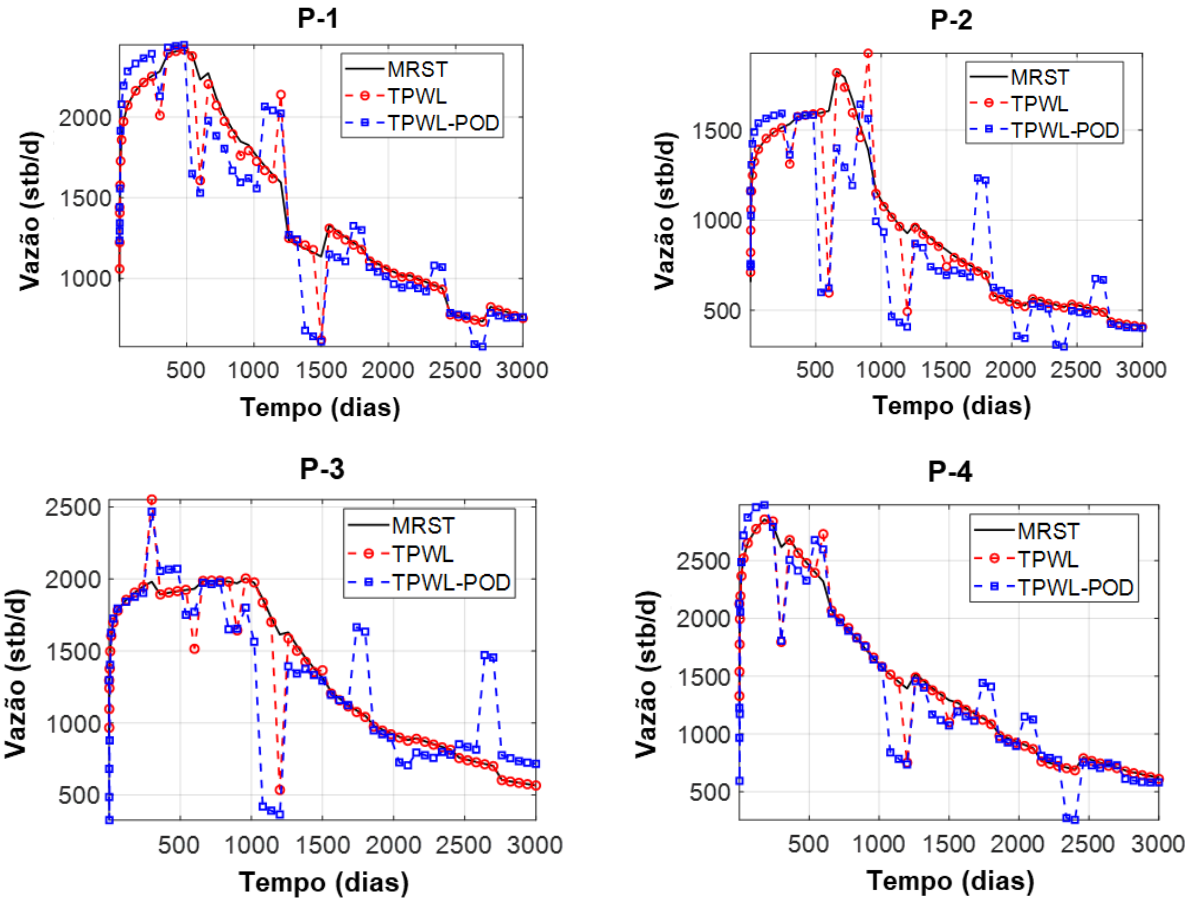


Figura 26 – Curvas de produção de óleo dos poços do modelo SPE10, para o Caso 1.



Fonte: O Autor (2019).

Figura 27 – Curvas de produção de água dos poços do modelo SPE10, para o Caso 1.

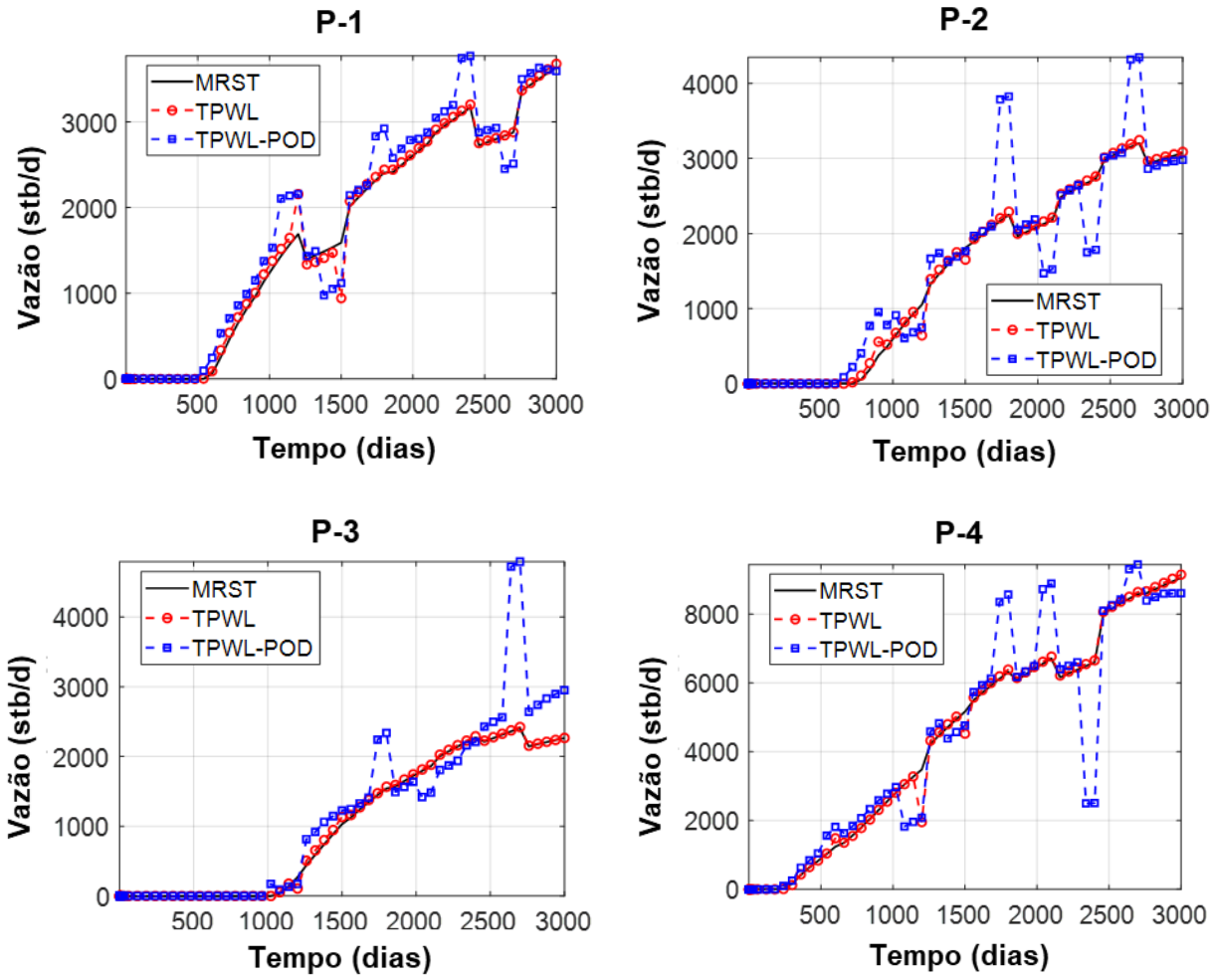
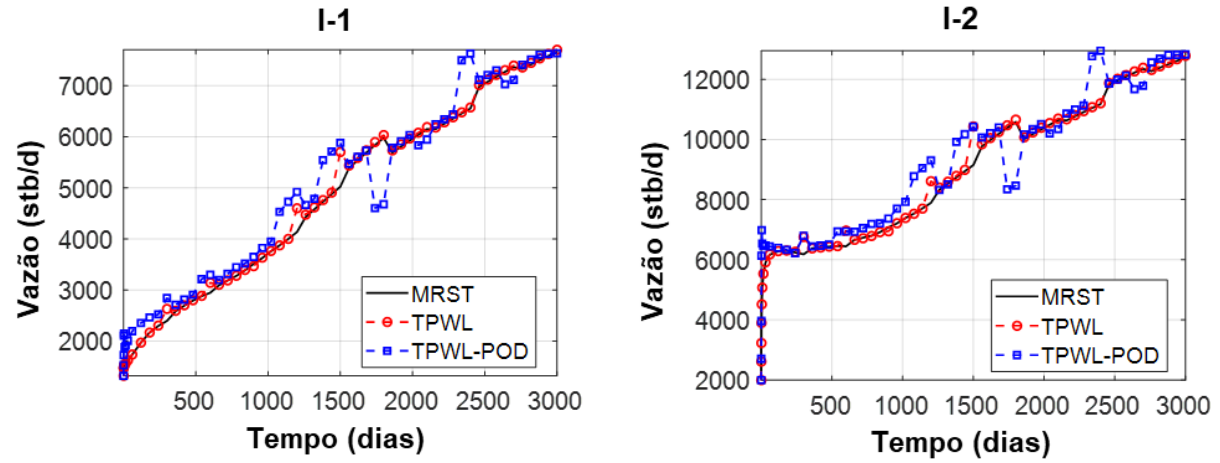


Figura 28 – Curvas de injeção de água dos poços do modelo SPE10, para o Caso 1.



Fonte: O Autor (2019).

Na Tabela 1, encontram-se resumidos os erros relativos das curvas de injeção de água e de produção de água e de óleo, definidos ao longo das Equações (4.31) a (4.33), e o tempo computacional requerido para os três tipos de simulação do Caso 1.

Tabela 1 – Erros e tempo computacional requerido, para o Caso 1 (Modelo SPE 10).

Tipo de Simulação	**Erro**		**Tempo Decorrido**
MRST	-		783,449s
TPWL	E_o	2,89%	72,208s
	$E_{w,inj}$	1,10%	
	$E_{w,prod}$	1,94%	
TPWL/POD (BG)	E_o	12,1%	0,366s
	$E_{w,inj}$	7,0%	
	$E_{w,prod}$	11,26%	
TPWL/POD (PG)	E_o	15,43%	0,369s
	$E_{w,inj}$	8,76%	
	$E_{w,prod}$	20,37%	

Fonte: O Autor (2019).

4.5.2 Modelo SPE10: Caso 2

Nas simulações do Caso 2, os controles dos poços produtores variam entre 500 e 4500 psi, fora do intervalo especificado no treinamento, distribuído em 10 ciclos de controle, ilustrados na Figura 29. O tempo total de simulação também foi de 3000 dias, e o BHP dos poços injetores foi constante e igual a 8000 psi. Trata-se também de um caso incompressível, visto que as compressibilidades da rocha e do fluido foram consideradas desprezíveis. Os resultados obtidos para este caso são ilustrados ao longo das Figuras 30 a 32.

Figura 29 – Trajetória de BHP aplicada aos poços produtores do modelo SPE10, para o Caso 2.

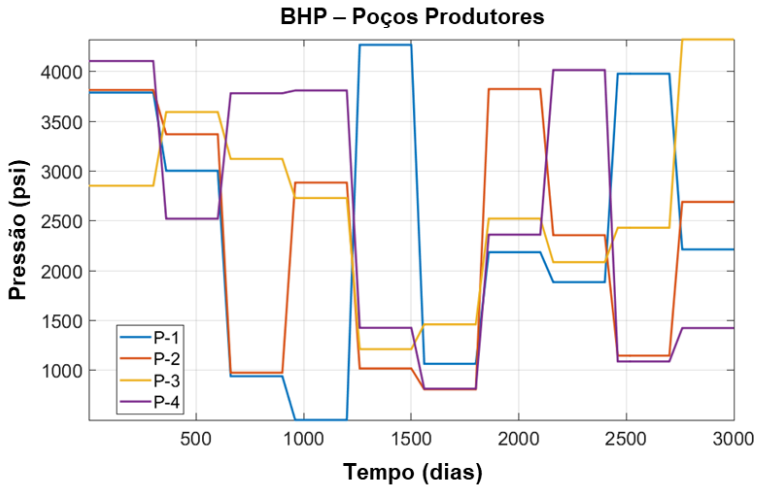
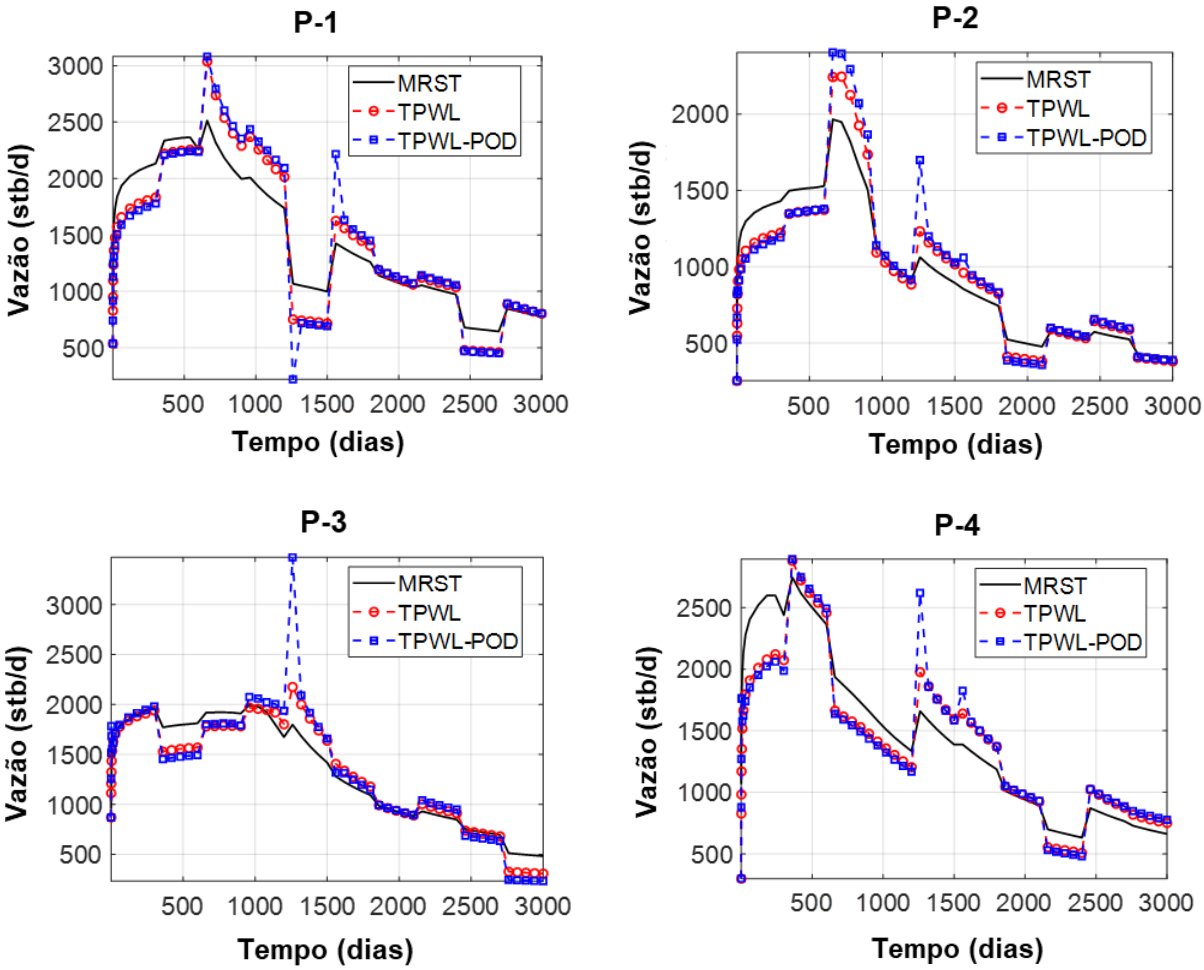


Figura 30 – Curvas de produção de óleo dos poços do modelo SPE10, para o Caso 2.



Fonte: O Autor (2019).

Figura 31 – Curvas de produção de água dos poços do modelo SPE10, para o Caso 2.

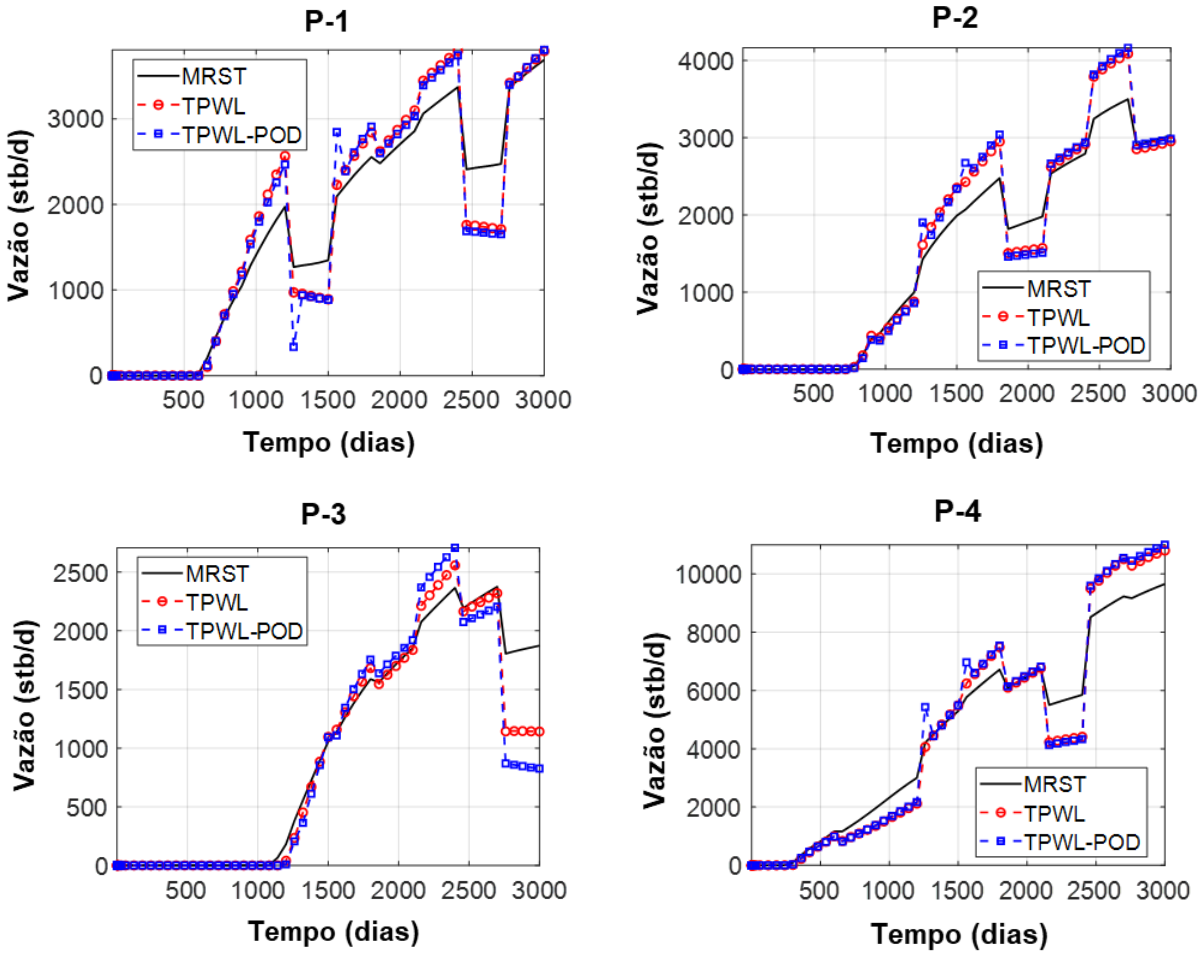
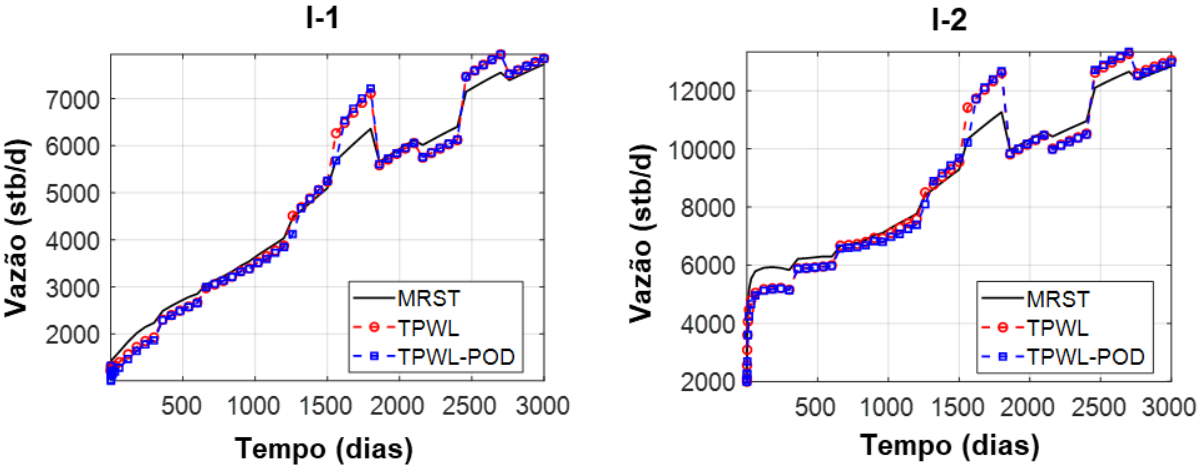


Figura 32 – Curvas de injeção de água dos poços do modelo SPE10, para o Caso 2.



Fonte: O Autor (2019).

Na Tabela 2**Erro! Fonte de referência não encontrada.**, encontram-se r esumidos os erros relativos das curvas de injeção de água e de produção de água e de óleo, definidos ao longo das Equações (4.31) a (4.33), e o tempo computacional requerido para os três tipos de simulação do Caso 2.

Tabela 2 – Erros e tempo computacional requerido, para o Caso 2 (Modelo SPE 10).

Tipo de Simulação	**Erro**		**Tempo Decorrido**
MRST	-		880,126s
TPWL	E_o	7,88%	72,350s
	$E_{w,inj}$	7,28%	
	$E_{w,prod}$	1,27%	
TPWL/POD (BG)	E_o	10,0%	0,383s
	$E_{w,inj}$	8,7%	
	$E_{w,prod}$	4,02%	
TPWL/POD (PG)	E_o	12,56%	0,393s
	$E_{w,inj}$	4,09%	
	$E_{w,prod}$	10,51%	

Fonte: O Autor (2019).

4.5.3 Modelo SPE10: Caso 3

Neste caso, os valores de BHP nos poços produtores variam entre 1000 e 4000 psi, ou seja, um intervalo igual ao especificado na simulação de treinamento, porém com novos valores estabelecidos na sequência de controles, ilustrados na Figura 33. O tempo total de simulação também foi de 3000 dias, e os valores de BHP dos poços injetores foi constante e igual a 8000 psi. Trata-se de um caso pouco compressível, pois foram admitidas as compressibilidades da rocha e do fluido constantes e iguais a

$10^{-6}\ psi^{-1}$. Os resultados obtidos para este caso são ilustrados ao longo das Figuras 34 a 36.

Figura 33 – Trajetória de BHP aplicada aos poços produtores do modelo SPE10, para o Caso 3.

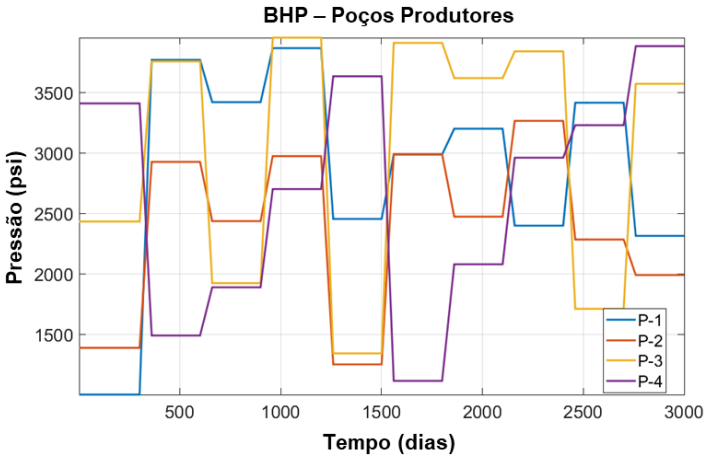
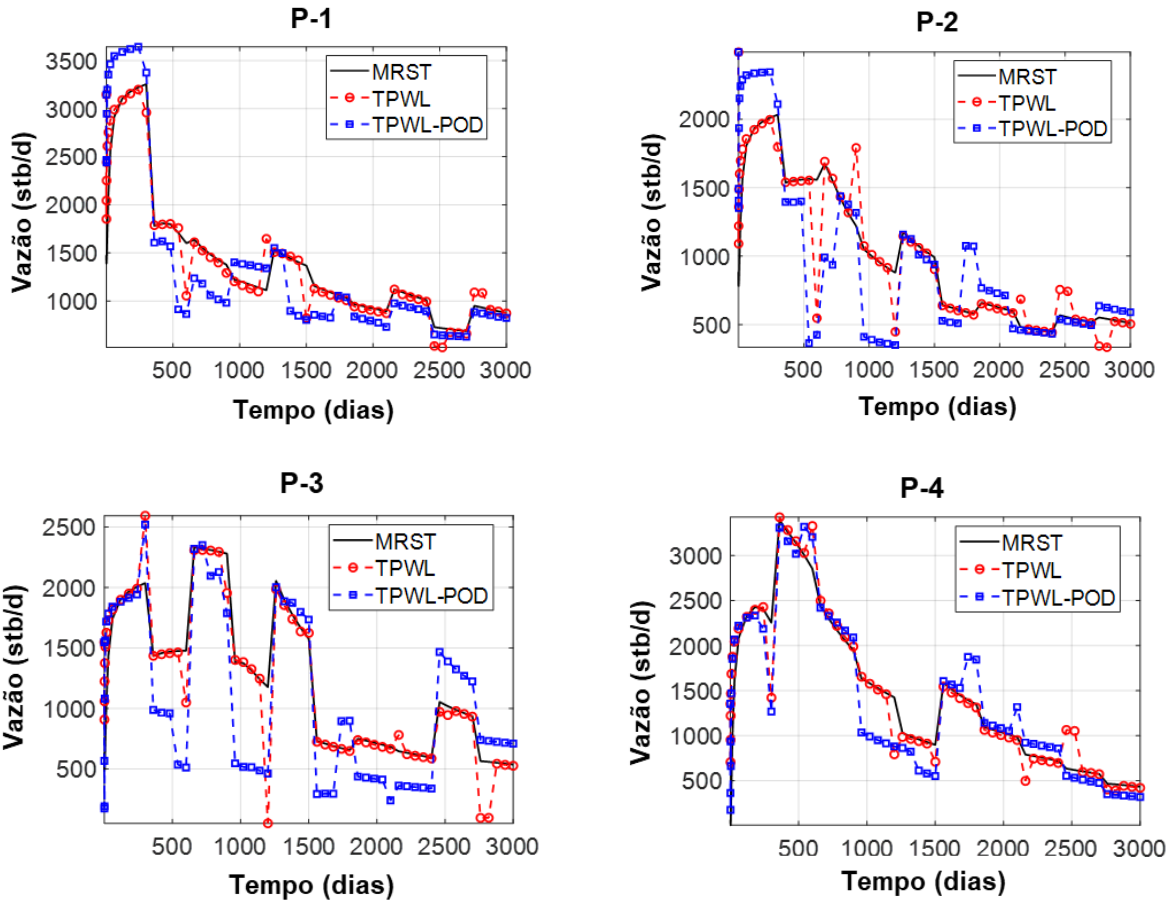


Figura 34 – Curvas de produção de óleo dos poços do modelo SPE10, para o Caso 3.



Fonte: O Autor (2019).

Figura 35 – Curvas de produção de água dos poços do modelo SPE10, para o Caso 3.

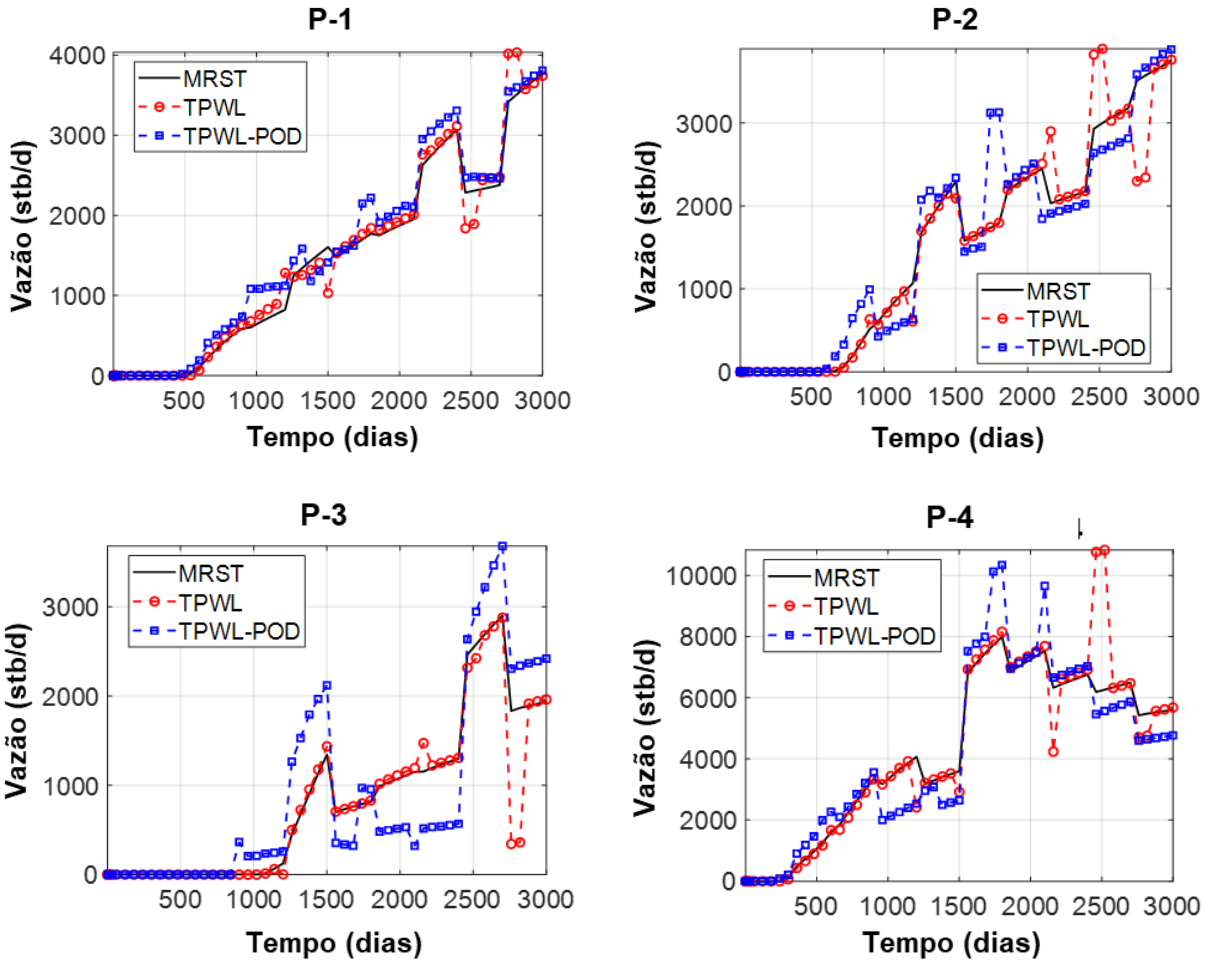
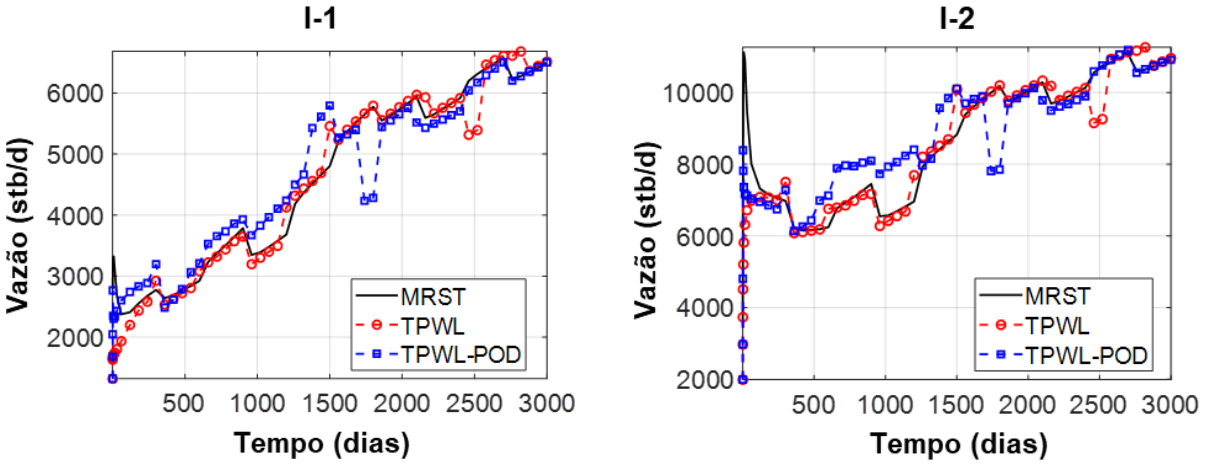


Figura 36 – Curvas de injeção de água dos poços do modelo SPE10, para o Caso 3.



Fonte: O Autor (2019).

Na Tabela 3, encontram-se resumidos os erros relativos das curvas de injeção de água e de produção de água e de óleo, definidos ao longo das Equações (4.31) a (4.33), e o tempo computacional requerido para os três tipos de simulação do Caso 3.

Tabela 3 – Erros e tempo computacional requerido, para o Caso 3 (Modelo SPE 10).

Tipo de Simulação	**Erro**		**Tempo Decorrido**
MRST	-		5571,865s
TPWL	E_o	3,89%	503,721s
	$E_{w,inj}$	2,32%	
	$E_{w,prod}$	6,58%	
TPWL/POD (BG)	E_o	15,66%	0,474s
	$E_{w,inj}$	8,65%	
	$E_{w,prod}$	18,13%	
TPWL/POD (PG)	E_o	28,47%	0,644s
	$E_{w,inj}$	16,22%	
	$E_{w,prod}$	23,95%	

Fonte: O Autor (2019).

Pode-se observar, portanto, que os erros relativos para os casos incompressíveis (Casos 1 e 2), onde os controles encontram-se dentro e fora do intervalo de treinamento, respectivamente, possuem magnitude semelhante, mostrando a capacidade do TPWL/POD de capturar o comportamento do modelo de alta fidelidade para ambas circunstâncias. Para o Caso 3, nota-se que a magnitude do erro é maior que a dos casos anteriores, mesmo quando o intervalo estabelecido é igual ao do treinamento, o que evidencia a influência da compressibilidade do sistema na acurácia do modelo de ordem reduzida, além de exigir um maior esforço computacional, em comparação aos tempos requeridos nos Casos 1 e 2. Em todas simulações, utilizou-se

um computador com processador Intel Core i7 3.4GHz com 16GB de RAM e sistema operacional Windows 7.

5 APLICAÇÃO EM OTIMIZAÇÃO: MODELO BRUGGE

Na área de gerenciamento de reservatórios, estudos de otimização tornam-se indispensáveis, seja na determinação da locação de novos poços, seja na determinação da trajetória ótima de controles a serem aplicados aos poços. O gerenciamento de reservatórios de petróleo submetidos à injeção de água pode ser formulado como um problema de otimização onde a função objetivo pode ser o VPL (Valor Presente Líquido) ou a produção acumulada de óleo, enquanto que as variáveis de projeto são os controles dos poços, injetores e produtores. Esses controles podem ser vazão, pressão de fundo de poço (BHP), abertura de válvulas, ou uma combinação deles. Restrições, por sua vez, podem ser impostas como limites (*bounds*) nos próprios controles, como funções lineares envolvendo as variáveis de projeto, ou ainda, como funções não-lineares envolvendo variáveis de estado.

Contudo, como já mencionado nas seções anteriores, simulações de modelos de alta fidelidade apresentam elevado custo computacional, e apesar dos avanços da tecnologia, e consequentemente da capacidade de processamento dos computadores, o tempo de avaliação desses modelos durante o processo de otimização é razoavelmente alto. O tratamento de restrições durante o processo de otimização requer o cálculo das variáveis de estado repetidas vezes, e também de suas derivadas, não apenas a cada passo de tempo, mas em diversos pontos intermediários. Por esse motivo, faz-se a utilização de modelos substitutos de ordem reduzida, cujos resultados são obtidos de modo aproximado, dentro de uma tolerância estabelecida, sendo executados em um tempo bem menor.

Neste trabalho, o modelo de ordem reduzida TPWL/POD é aplicado na otimização do modelo Brugge, um reservatório sintético controlado por BHP, onde a função objetivo a ser maximizada é a produção de óleo do campo sujeito a restrições de produção líquida e de água do campo. O interesse nesse tipo de problema deve-se às limitações operacionais comuns nos campos de produção, comprometendo a eficiência do sistema separador óleo-água diante de grandes produções de líquido.

O algoritmo a ser empregado durante o processo de otimização é a Programação Quadrática Sucessiva (*Sequential Quadratic Programming*, SQP), o

principal algoritmo utilizado em problemas de injeção de água, que envolve um grande número de variáveis e restrições. Uma aplicação deste algoritmo ao problema de otimização do gerenciamento de reservatórios de petróleo pode ser encontrada em (Pinto et al., 2019).

O algoritmo encontra-se disponível na caixa de ferramentas de otimização no MATLAB (The MathWorks Inc., 2018), e considerou-se como função de alta fidelidade o modelo de reservatórios simulado no MRST versão 2018b, descrito no Capítulo 3, e como função de baixa fidelidade, o modelo de ordem reduzida TPWL/POD, descrito no Capítulo 4. Em ambos os modelos, a função-objetivo utilizada no processo de otimização é a produção acumulada de óleo do campo e os controles especificados são do tipo BHP.

5.1 Formulação do Problema de Injeção de Água

Em geral, problemas de otimização restrita são definidos conforme Equação (5.1).

$$\begin{aligned} &\underset{u \in \mathbb{R}^{N_u}}{Maximize} \quad F(x) \\ &Sujeito\ a: \\ &\quad h(u) = 0 \\ &\quad g(u) \leq 0 \\ &\quad u_i^{low} \leq u_i \leq u_i^{up}; \quad i = 1, \dots, N_u \end{aligned} \tag{5.1}$$

Onde F é a função objetivo, u é o vetor composto pelas variáveis de controle, h é a restrição linear ou não-linear de igualdade, e g são as restrições lineares ou não-lineares de desigualdade. Os índices up e low representam, respectivamente, os limites superior e inferior de u, e N_u é o número total de variáveis de controle.

A função objetivo a ser maximizada é a produção acumulada de óleo do campo, sujeito a restrições não-lineares relativas às vazões de produção de líquido e de água do campo. Sua formulação matemática encontra-se na Equação (5.2), onde: $C_{oil,field}$ é a produção acumulada de óleo do campo; P é o conjunto de poços produtores; I é o conjunto de poços injetores; N_t é o número de ciclos de controle; $q_{w,t}^{l}$ e $q_{w,t}^{wp}$ são,

respectivamente, as vazões de líquido e de água no ciclo de controle t do produtor w; $Q_{liq,max}$ e $Q_{wp,max}$ são, respectivamente, as vazões de líquido e de água máximas permitidas no campo; BHP_w^{low} e BHP_w^{up} são os limites inferior e superior da pressão de fundo de poço, respectivamente; e $\bar{P}_{field}$ é a pressão média inicial do reservatório.

$$
\begin{aligned}
&Maximize \quad C_{oil,field} \\
&Sujeito\ a: \\
&\sum_{w \in P} q_{w,t}^{l} \leq Q_{liq,max}; \ \ t = 1, \dots, N_t \\
&\sum_{w \in P} q_{w,t}^{wp} \leq Q_{wp,max}; \ \ t = 1, \dots, N_t \\
&BHP_w^{low} \leq BHP_{w,t} \leq \bar{P}_{field}; \quad t = 1, \dots, N_t \ e \ w \in P \\
&\bar{P}_{field} \leq BHP_{w,t} \leq BHP_w^{up}; \quad t = 1, \dots, N_t \ e \ w \in I
\end{aligned}
\tag{5.2}
$$

As variáveis de controle são as pressões de fundo de poço em cada válvula, em cada ciclo de controle t, representadas por u_w^t, devendo ser normalizadas para a sua entrada no otimizador, conforme as Equações (5.3) e (5.4).

$$\bar{u}_w^t = \frac{BHP_{w,t} - BHP_w^{low}}{BHP_w^{up} - BHP_w^{low}}, \qquad w \in P \tag{5.3}$$

$$\bar{u}_w^t = \frac{BHP_w^{up} - BHP_{w,t}}{BHP_w^{up} - BHP_w^{low}}, \qquad w \in I \tag{5.4}$$

As restrições de vazão de líquido e de água consistem em restrições de estado e, assim, restrições não-lineares, que são funções das variáveis de controle. As vazões de líquido e de água nas válvulas, por sua vez, também devem ser normalizadas, neste caso, pela respectiva vazão máxima de líquido e de água no campo, como descrito nas Equações (5.5) e (5.6).

$$\bar{q}_{w,t}^{l} = \frac{\sum_{w \in P} q_{w,t}^{l} - Q_{liq,max}}{Q_{liq,max}} \leq 0 \tag{5.5}$$

$$\bar{q}_{w,t}^{wp} = \frac{\sum_{w \in P} q_{w,t}^{wp} - Q_{wp,max}}{Q_{wp,max}} \leq 0 \tag{5.6}$$

Logo, feitas as normalizações, tem-se uma formulação alternativa ao exposto na Equação (5.2), proposta na Equação (5.7), onde ψ é o conjunto formado por todos os poços (injetores e produtores). As restrições de estado, por sua vez, foram impostas durante o processo de otimização.

$$
\begin{aligned}
&Maximize \quad C_{oil,field} \\
&Sujeito\ a: \\
&\quad \bar{q}_{w,t}^{l} \leq 0 \\
&\quad \bar{q}_{w,t}^{wp} \leq 0 \\
&\quad 0 \leq u_{w}^{t} \leq 1;\ t = 1, \dots, N_t\ e\ w \in \psi
\end{aligned}
\tag{5.7}
$$

O cálculo da função objetivo adotada requer uma simulação completa do reservatório, exigindo um grande esforço computacional. O objetivo deste trabalho é, portanto, aplicar o procedimento TPWL/POD a fim de permitir avaliações de função mais rápidas ao longo do processo de otimização.

5.2 Algoritmo SQP

Nesta seção, será apresentado o principal algoritmo utilizado em problemas de injeção de água, que envolve um grande número de variáveis e de restrições. O algoritmo SQP é uma técnica de otimização baseada em gradientes. Estudos como o de (Liu et al., 2018) mostram que o SQP apresenta melhor performance no tratamento dado às restrições em relação ao método Lagrangeano Aumentado (*The Augmented Lagrangian Method*), enquanto que trabalhos como o de (Tueros, 2019) afirmam que o SQP apresenta melhor performance que o método IP (*Interior Point Method*) no tratamento da função objetivo e no número de avaliação da função necessária durante o processo de otimização. Assim, o algoritmo escolhido para este trabalho é o SQP, que se encontra disponível na função `fmincom` do MATLAB.

Seja F a função objetivo, e sejam h e g as restrições de igualdade e desigualdade, respectivamente. A estratégia básica para resolver problemas de otimização é escrever as condições de primeira ordem necessárias e aplicar o Método de Newton para encontrar iterativamente uma solução para sistema de equações não-lineares resultante. Contudo, restrições de desigualdade impõem condições de

complementaridade onde os produtos do valor de cada restrição pelo multiplicador Lagrange correspondente deve ser nulo, o que torna o sistema de equações mal condicionado. Esse problema é corrigido usando a estratégia de "conjunto ativo", onde o algoritmo decide quais restrições de desigualdade estão ativas na solução, transformando-as em restrições de igualdade. O SQP, então, encontra o conjunto ativo resolvendo, em cada iteração k, o subproblema de minimização quadrático com restrições lineares, dado pela Equação (5.8).

$$\begin{aligned} &Minimize \quad \nabla F(x^k) d_x + \frac{1}{2} d_x^T W^k d_x \\ &Sujeito\ a: \\ &\quad h(x^k) + \nabla h(x^k)^T d_x = 0 \\ &\quad g(x^k) + \nabla \mathrm{g}(x^k)^T d_x \leq 0 \end{aligned} \tag{5.8}$$

Onde d_x é o vetor de correção x na iteração k, e W^k é a matriz Hessiana da função Lagrangeana do problema original, e não apenas da função objetivo, o que faz as curvaturas das restrições serem levadas em consideração. Devido ao seu alto custo, a matriz Hessiana é obtida via aproximação usando a técnica BFGS (*Broyden–Fletcher–Goldfarb–Shanno*), e, por ser uma matriz positiva definida, o subproblema quadrático pode ser resolvido em um número finito de iterações (Biegler, 2010).

O algoritmo SQP pode então ser resumido da seguinte maneira:

1. Selecione uma solução inicial x^0, tomando $k = 0$;
2. Estabeleça uma aproximação inicial para a matriz Hessiana dos termos quadráticos da função objetivo;
3. Resolva o subproblema da Equação (5.8) e encontre a direção de busca d_x;
4. Faça uma busca linear para determinar o tamanho do passo α na direção d_x;
5. Atualize a solução para a posição indicada: $x^{k+1} = x^k + \alpha d_x$;
6. Verifique a convergência do algoritmo:
   a. Caso o mínimo local tenha sido encontrado, encerre o processo.
   b. Caso contrário, atualize a matriz Hessiana via BFGS e retorne ao passo 3.

5.3 Modelo Brugge

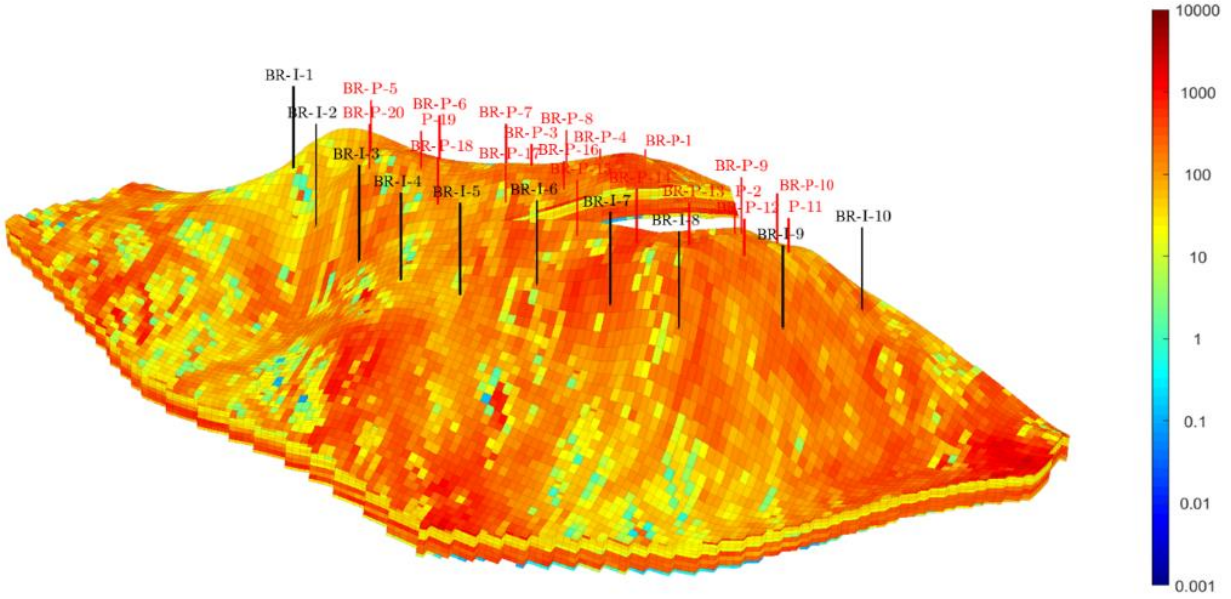
Neste capítulo, será considerado um modelo de maiores dimensões, que mais se assemelha aos modelos encontrados em simulação de reservatórios na prática, a fim de se demonstrar a aplicabilidade do modelo TPWL/POD em processos de otimização. O modelo consiste em quatro zonas, divididas em nove camadas, com o total de 60048 células, dentre as quais 43217 são ativas, e possui 30 poços, onde 20 são produtores e 10 são injetores. Para o caso particular deste trabalho, todos os poços apresentam uma única completação, e foram estabelecidos como limites do BHP o intervalo de 1000 a 2400 psi, para os poços produtores, e de 2400 a 3000 psi, para os poços injetores, visto 2400 psi ser a pressão média inicial do reservatório.

Nesse estudo, usou-se a realização de número 28, dentre as 104 realizações existentes do modelo. A permeabilidade horizontal média é de 182,54mD e a vertical, 12,29mD. A porosidade média, por sua vez, é 0,1916. As compressibilidades da rocha e da água são, respectivamente, $5{,}0763 \times 10^{-10}$ e $4{,}3511 \times 10^{-10}$. As densidades do óleo e da água são aproximadamente, 897kg/m^3 e 1000kg/m^3. As curvas de permeabilidade relativa seguem o modelo de Corey descrito nas Equações (2.13) e (2.14), para uma saturação de água conata igual a 0,25. A Figura 37 ilustra a distribuição dos poços e a permeabilidade k_x do Modelo Brugge.

De forma análoga ao modelo SPE10 do Capítulo 4, para o modelo Brugge, foi realizada apenas uma única simulação de treinamento, especificando o valor médio dos limites estabelecidos para cada poço como seus controles ao longo de todo o tempo de concessão, igual a 21 anos. O passo de tempo adotado no simulador foi de 30 dias.

Armazenados os estados e as matrizes de derivadas, realizou-se então uma nova simulação, especificando uma nova sequência de BHP's nos poços, dividida em 14 ciclos de controles, modificados a cada 1,5 ano, dentro do mesmo intervalo adotado na simulação de treinamento, ilustrada na Figura 38.

Figura 37 – Permeabilidade k_x do Modelo Brugge, com distribuição dos poços.

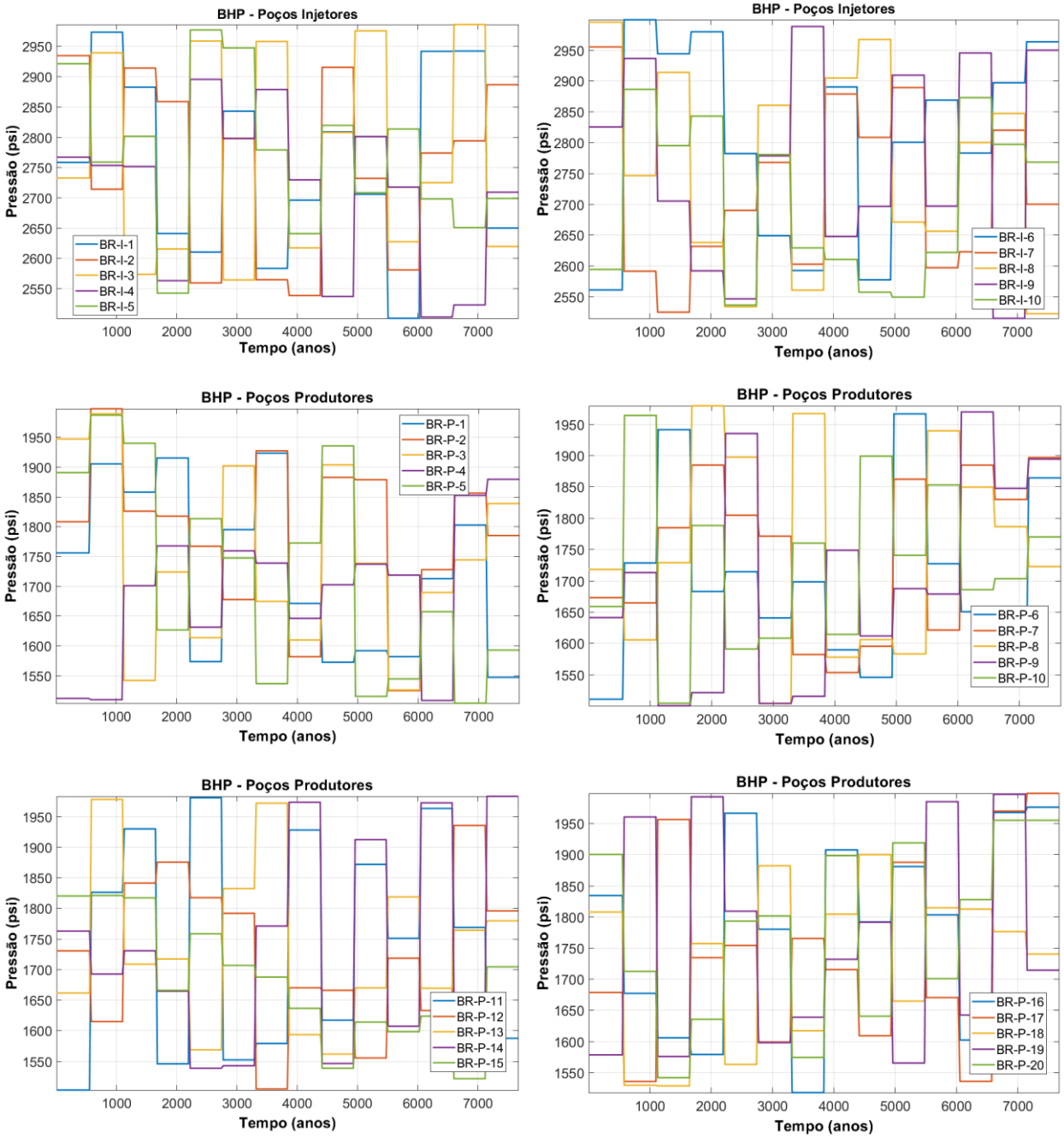


Fonte: O Autor (2019).

Ao longo das Figura 39 a 41, é demonstrado a performance do modelo de ordem reduzida, para o teste descrito anteriormente, comparando as curvas de produção e injeção obtidas para três tipos de simulação, isto é, para os modelos de alta fidelidade (MRST), TPWL e TPWL/POD, ilustrando apenas os resultados obtidos utilizando a projeção de Petrov-Galerkin, cujo erro neste caso demonstrou ser menor que o obtido com a projeção de Bubnov-Galerkin, utilizando um critério de energia mínima igual a 0,9999, onde 6 de 520 autovalores/autovetores foram incluídos.

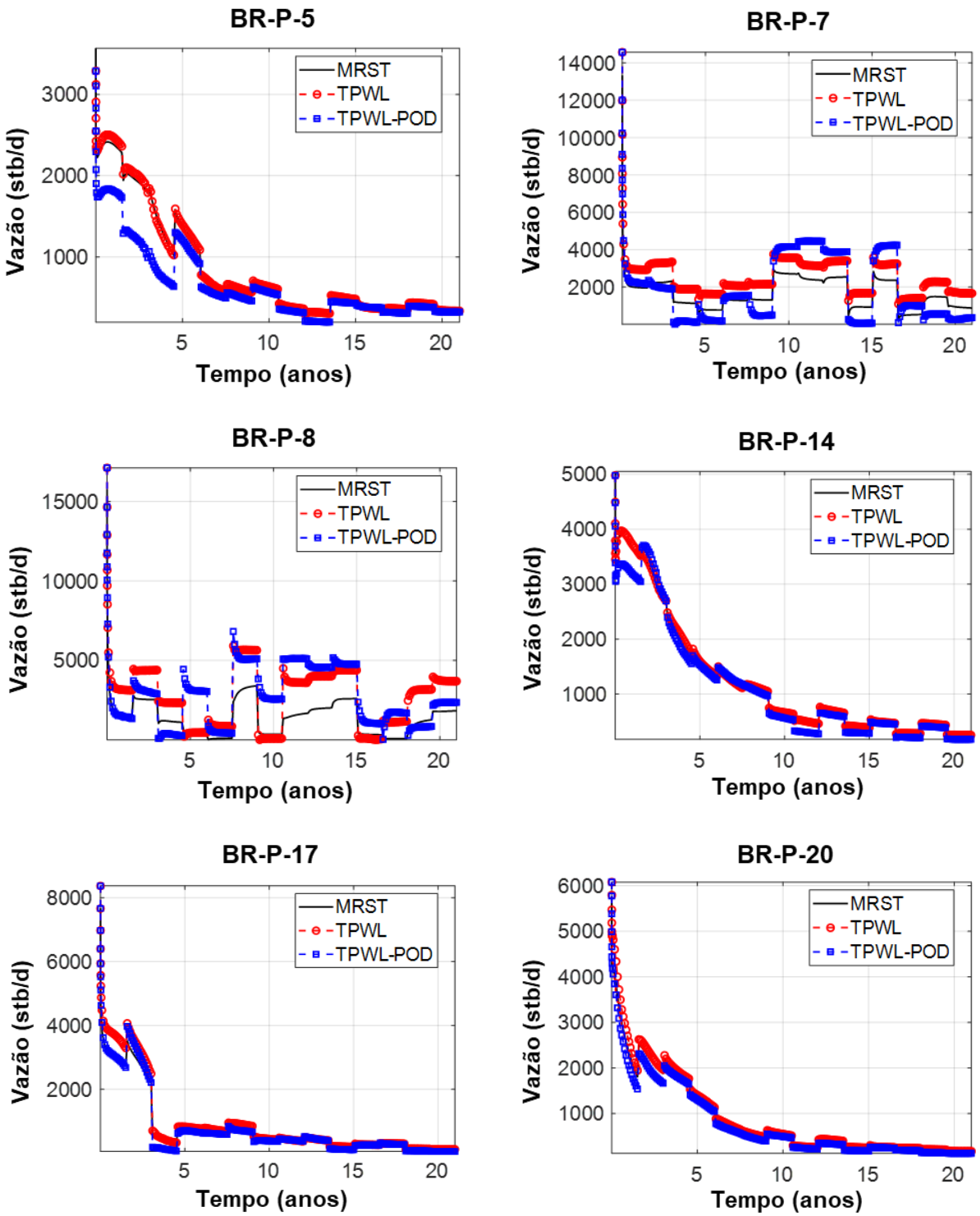
Obteve-se s*peed-up's* da ordem de 800, entre o tempo computacional requerido no modelo de alta fidelidade e TPWL/POD. Para a maioria dos poços, as curvas apresentam boa aderência entre si, para ambos os tipos de projeção. Contudo, anormalidades se mostraram presentes em alguns poços específicos, levando a magnitude do erro ser maior em relação aos erros obtidos no modelo SPE10, apresentados no Capítulo 4.

Figura 38 – Trajetória de BHP aplicada aos poços do modelo Brugge.



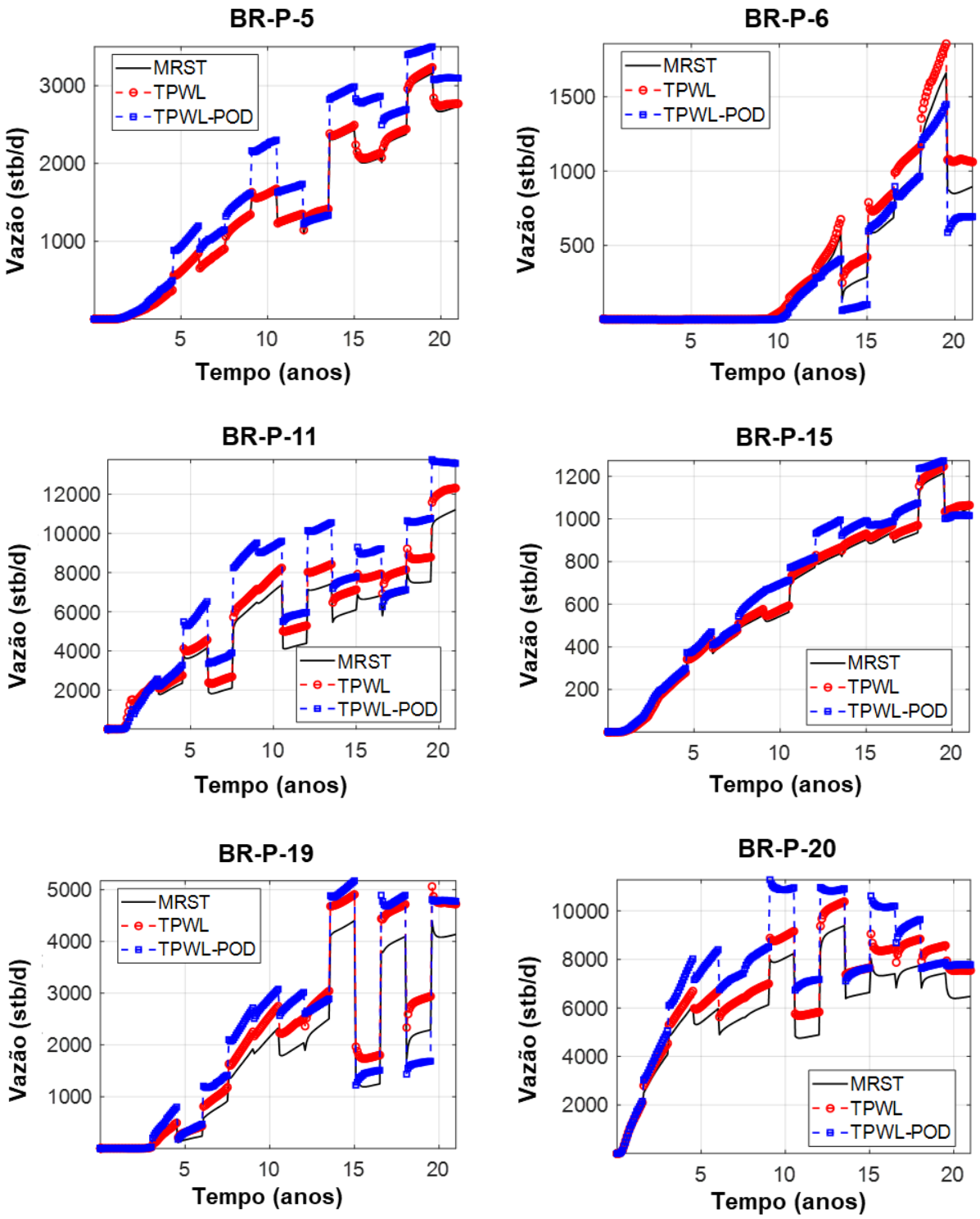
Fonte: O Autor (2019).

Figura 39 – Curvas de produção de óleo dos poços produtores do modelo Brugge.



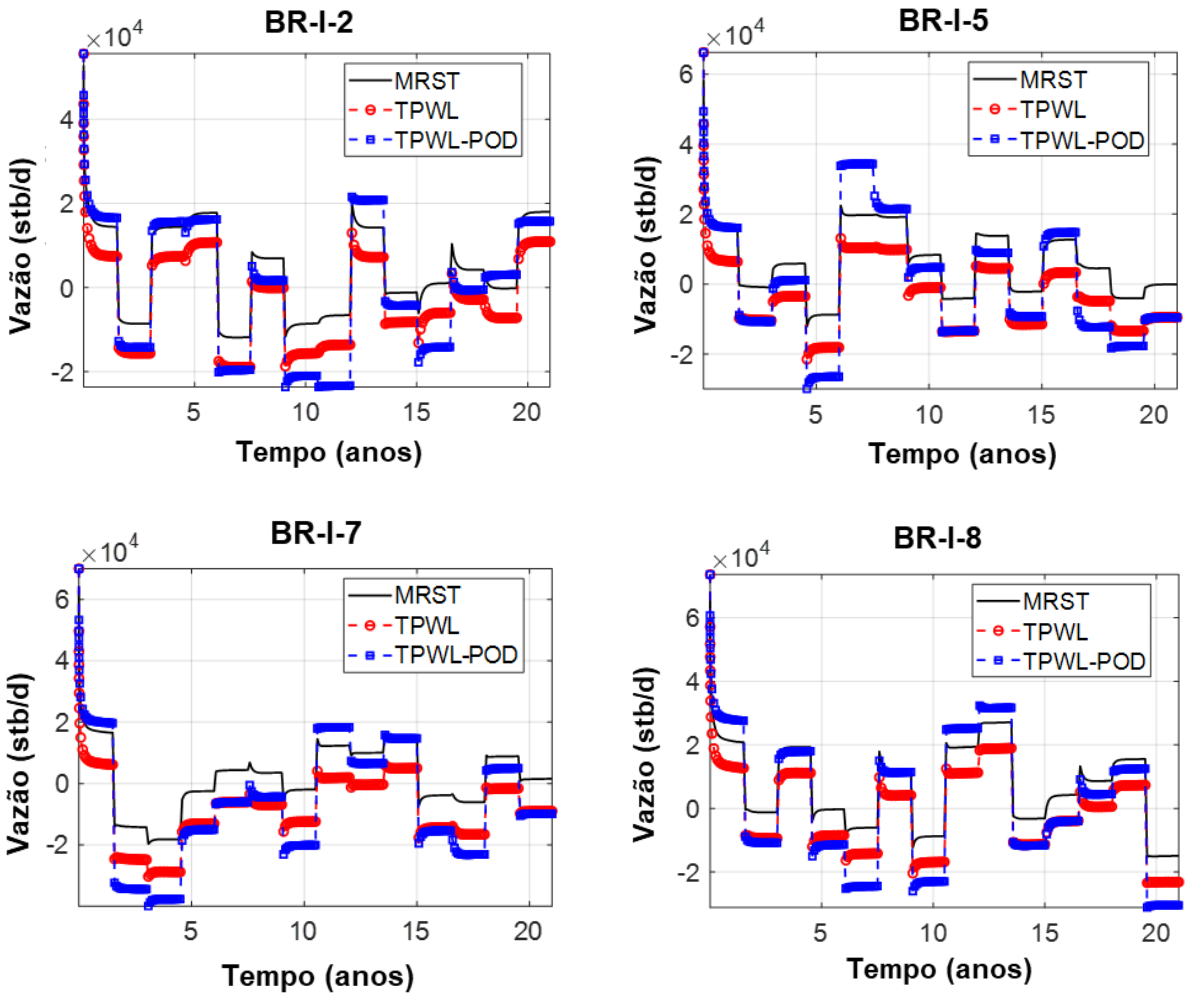
Fonte: O Autor (2019).

Figura 40 – Curvas de produção de água dos poços produtores do modelo Brugge.



Fonte: O Autor (2019).

Figura 41 – Curvas de injeção de água dos poços injetores do modelo Brugge.



Fonte: O Autor (2019).

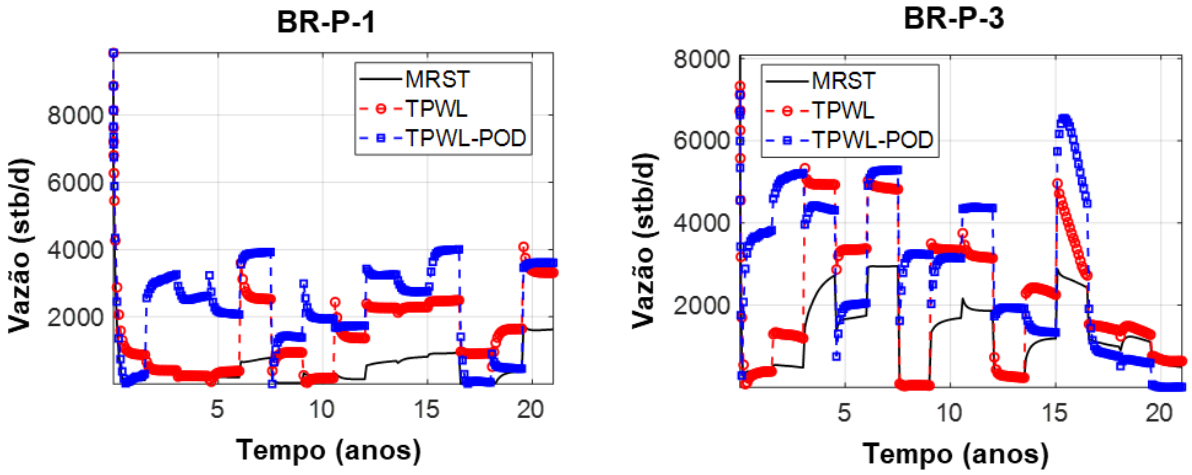
Como mencionado anteriormente, para a maioria dos poços, as curvas apresentam boa aderência entre si, para ambos os tipos de projeção. Contudo, anormalidades se mostraram presentes em alguns poços específicos, e, portanto, a magnitude do erro se mostrou bastante elevada, acima de 100%, levando a uma perda de acurácia e má aderência entre as curvas de vazão de produção e injeção, obtidas nos três tipos de simulação, como se pode observar nas Figura 42 a 44. Uma possível causa dessa magnitude do erro deve-se às mudanças bruscas de BHP impostas e que podem ser observadas na Figura 38. Os erros entre as curvas de injeção e de produção, definidos nas Equações (4.31) a (4.33), e o tempo computacional requerido para os três tipos de simulação, encontram-se resumidos na Tabela 54.

Tabela 4 – Erro médio e tempo computacional, para o modelo Brugge.

Tipo de Simulação	Erro		Tempo Decorrido
MRST	-		3730,149s
TPWL	E_o	24,8%	544,326s
	$E_{w,inj}$	137,93%	
	$E_{w,prod}$	13,18%	
TPWL/POD (BG)	E_o	20,1%	5,037s
	$E_{w,inj}$	107,6%	
	$E_{w,prod}$	28,0%	
TPWL/POD (PG)	E_o	20,4%	4,869s
	$E_{w,inj}$	78,7%	
	$E_{w,prod}$	23,6%	

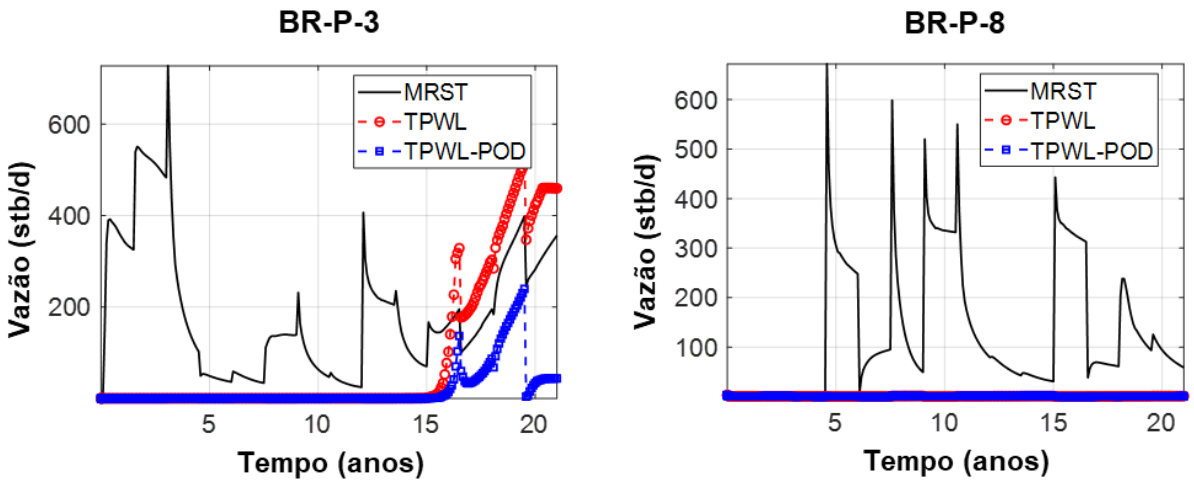
Fonte: O Autor (2019).

Figura 42 – Casos de má aderência entre as curvas de produção de óleo do modelo Brugge.



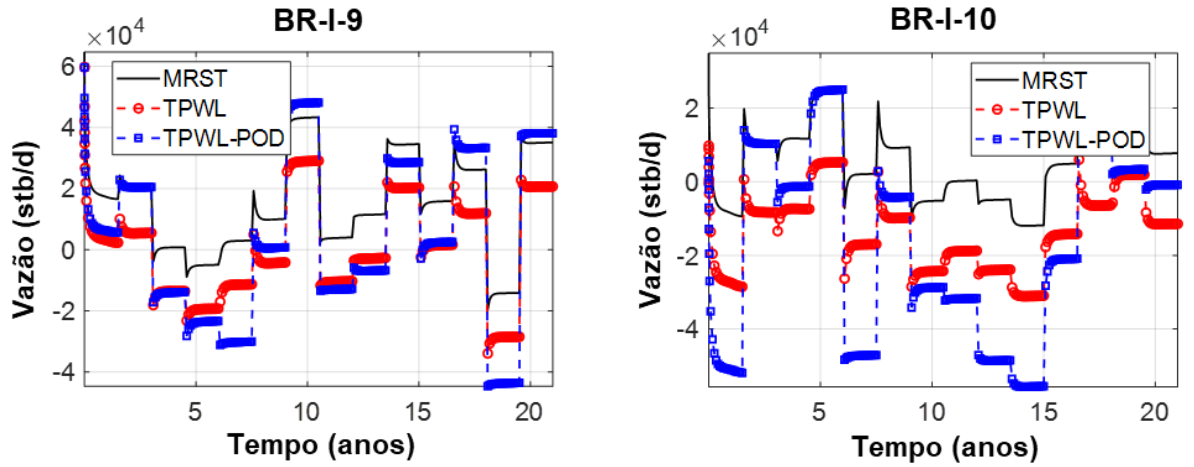
Fonte: O Autor (2019).

Figura 43 – Casos de má aderência entre as curvas de produção de água do modelo Brugge.



Fonte: O Autor (2019).

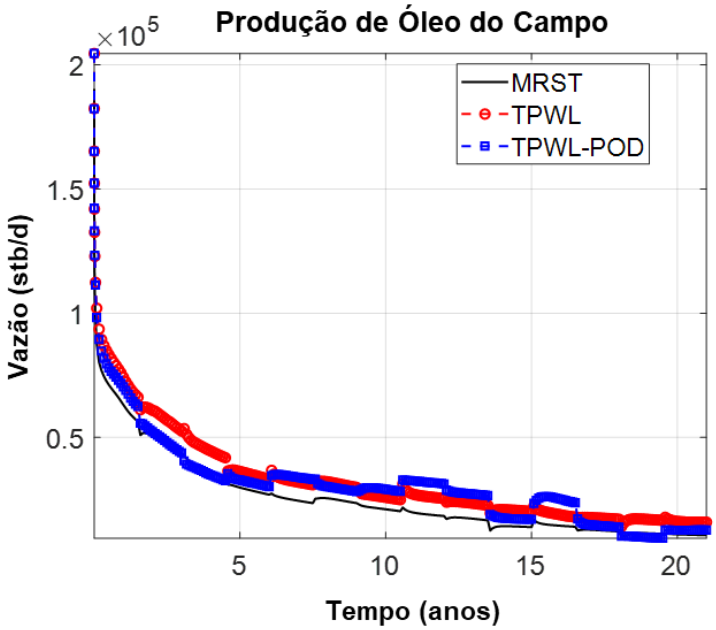
Figura 44 – Casos de má aderência entre as curvas de injeção de água do modelo Brugge.



Fonte: O Autor (2019).

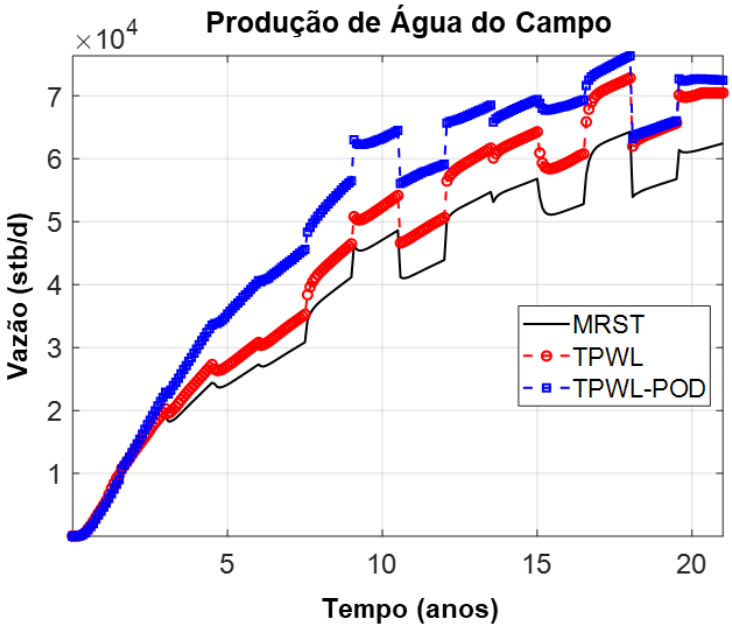
Apesar da elevada magnitude do erro nesses poços, as vazões de produção e injeção obtidas para o campo Brugge apresentam resultados razoáveis para os três tipos de simulação realizados, como se pode notar ao longo das Figura 45 a 47.

Figura 45 – Curva de produção de óleo do campo Brugge.



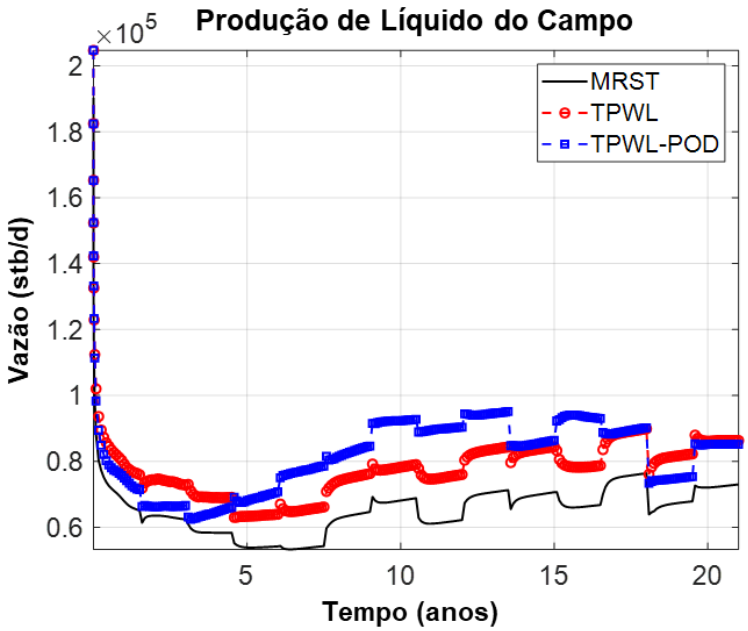
Fonte: O Autor (2019).

Figura 46 – Curva de produção de água do campo Brugge.



Fonte: O Autor (2019).

Figura 47 – Curva de produção de líquido do campo Brugge.



Fonte: O Autor (2019).

5.4 Aplicação em Processo de Otimização

Neste trabalho, o modelo sintético de reservatório Brugge, descrito na seção anterior, foi utilizado a fim de validar o uso de modelos substitutos de ordem reduzida via TPWL/POD em problemas de otimização restrita. Todas as simulações foram feitas usando o MRST, versão 2018b. Durante o processo de otimização, os gradientes foram obtidos através da aproximação por diferenças finitas.

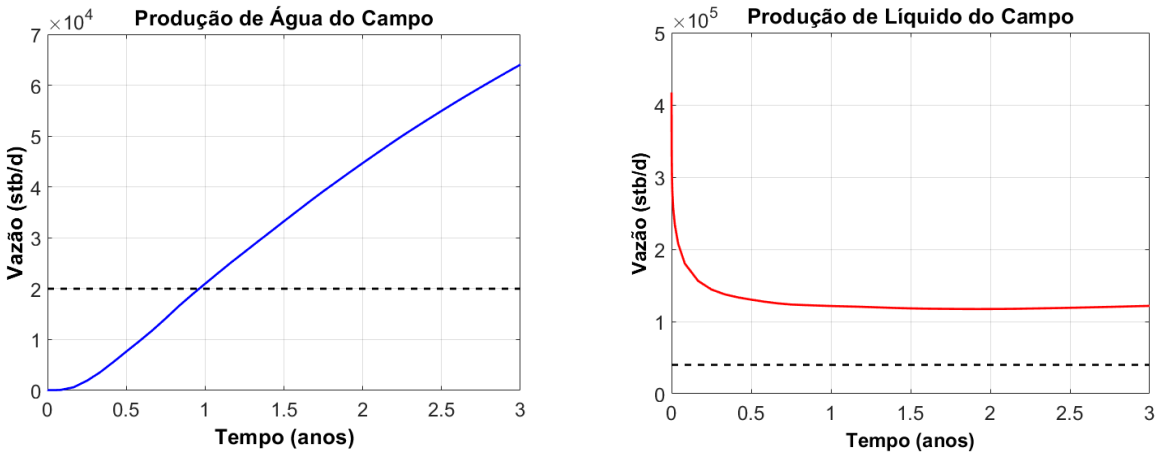
Devido ao grande número de simulações necessárias para o cálculo do gradiente por diferenças finitas, adotou-se um período de 3 anos, composto por 6 ciclos de controle. Os controles especificados foram BHP, tanto para poços injetores quanto para os produtores, modificados a cada mês, resultando num total de 180 variáveis, ou seja, 6 ciclos de controles aplicados individualmente a 30 poços. As restrições relativas ao BHP foram impostas diretamente no otimizador, estabelecendo o intervalo de 1000 a 2400 psi como os limites do BHP para os poços produtores, e de 2400 a 3000 psi, para os poços injetores, visto 2400 psi ser a pressão média inicial do reservatório.

Para a geração do modelo substituto, realizou-se uma única simulação de treinamento, para o tempo total de 3 anos, especificando um BHP constante ao longo do tempo de concessão e igual ao valor médio entre os limites estabelecidos para cada tipo de poço (injetor ou produtor). Logo, realizada a simulação de treinamento no MRST, fez-se a exportação dos estados e das matrizes de derivadas, conforme descrito no Capítulo 4, a fim de serem inseridos na equação do TPWL, dada pela Equação (4.8).

Quanto às restrições de estado, a máxima vazão de produção de água permitida no campo foi de 20.000 stb/d, enquanto que a máxima produção de líquido permitida foi 40.000 stb/d. Adotou-se os limites extremos estabelecidos em cada poço como ponto inicial do processo de otimização. A ideia de usar este ponto é partir de uma solução inviável para uma solução viável, tornando o processo de otimização mais desafiador.

Fez-se então uma simulação do modelo Brugge no MRST, especificando os limites extremos de cada poço como seus respectivos controles, constantes ao longo dos 3 anos de simulação, cujas curvas de produção do campo encontram-se ilustradas na Figura 48. As linhas tracejadas referem-se aos limites a serem impostos às restrições de estado durante o processo de otimização a ser realizado logo em seguida.

Figura 48 – Curvas de produção de água (à esquerda) e de líquido (à direita) do campo Brugge, com controles referentes ao ponto inicial do processo de otimização.



Fonte: O Autor (2019).

5.5 Desempenho e Resultados

O ponto inicial do processo de otimização, para todos os ciclos de controle, foram os limites extremos de cada poço, isto é, BHP igual a 1000 psi, para os poços produtores, e 3000 psi, para os poços injetores.

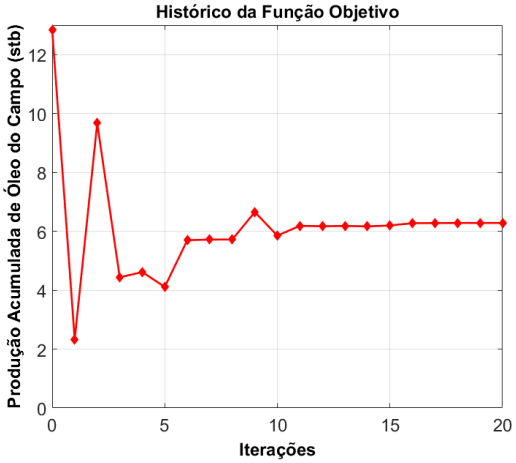
Visando mitigar os erros obtidos nos testes simulados na seção 5.3, ilustrados nas Figura 42 a 44, buscou-se ampliar o critério de energia mínima do modelo TPWL/POD para 0,999999, no qual 11 de 88 autovalores/autovetores foram incluídos.

Nos Casos 1 e 2 apresentados a seguir, o valor estabelecido para a tolerância entre duas iterações consecutivas foi de 10^{-4}, e adotou-se o número máximo de iterações igual a 30. Os gradientes, por sua vez, foram obtidos por diferenças finitas.

5.5.1 Modelo TPWL/POD: Caso 1 – Projeção Bubnov-Galerkin

Os resultados obtidos para o Caso 1 mostram que não houve problema de estabilidade com a utilização da projeção de Bubnov-Galerkin, conforme é mencionado em (Fragoso, 2014). Na Figura 49, ilustra-se o histórico da produção acumulada de óleo, ao longo das 20 iterações decorridas durante o processo de otimização. Observa-se inicialmente oscilações no valor da função objetivo, a fim de que ambas as restrições de estado sejam atendidas, como se pode notar nas Figura 50 e Figura 51.

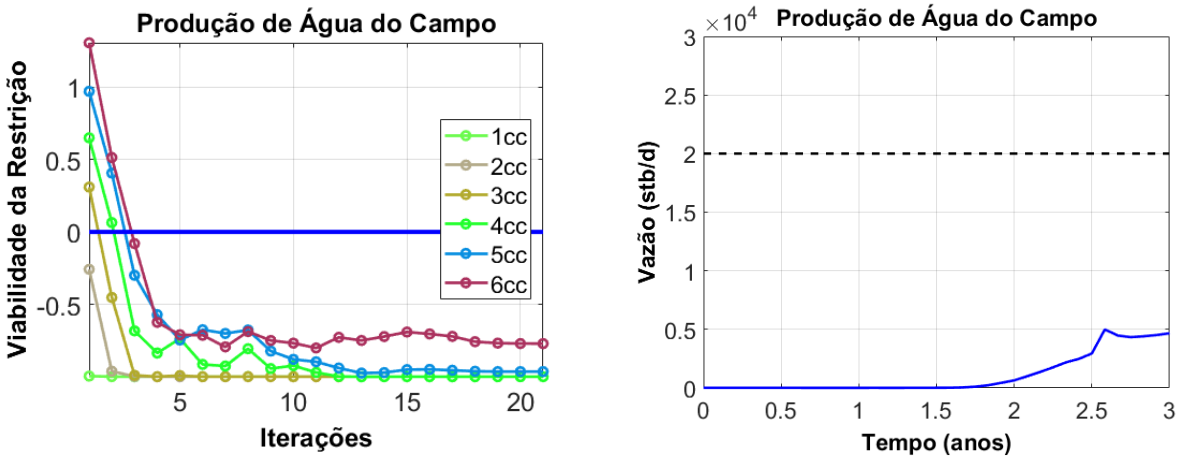
Figura 49 – Histórico da evolução da produção acumulada de óleo, $C_{oil,field}$, obtida no Caso 1.



Fonte: O Autor (2020).

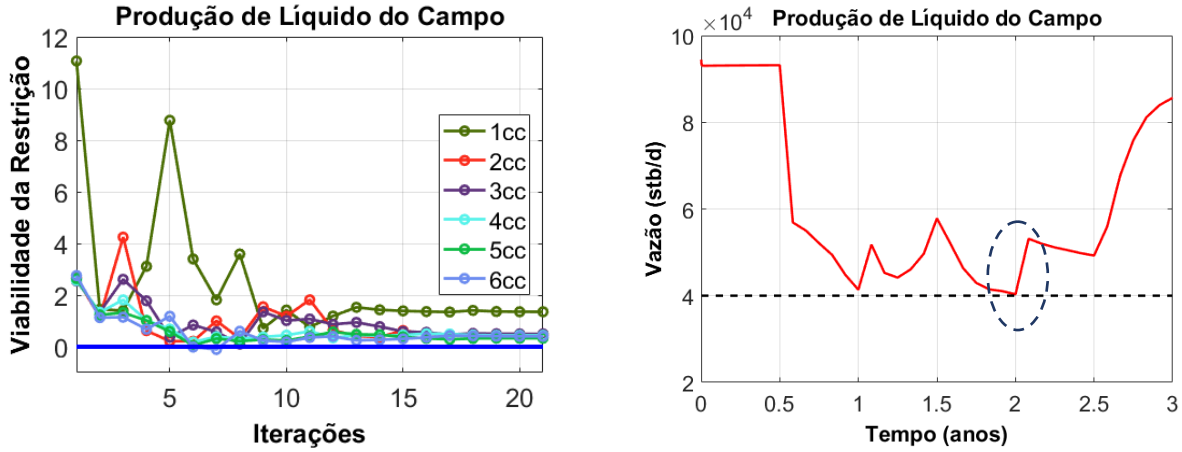
Na Figura 50, observa-se que as restrições quanto à produção de água no campo foram todas satisfeitas após a otimização, seja nos extremos, seja no interior de cada ciclo de controle. Já as restrições de líquido, embora não tenham sido satisfeitas ao fim da otimização na maioria dos instantes em que há mudança nos controles, é possível observar que, mesmo no ciclo de número 5 (ver destaque na Figura 51), onde a restrição é satisfeita, observa-se uma excessiva produção de líquido após a mudança do controle, ou seja, devido à descontinuidade dos controles (BHP) em cada ciclo de controle, as restrições tendem a não serem respeitadas no interior de cada ciclo.

Figura 50 – Viabilidade das restrições de estado (à esquerda) e resultado da otimização via TPWL/POD (à direita) para produção de água do campo, para o Caso 1.



Fonte: O Autor (2020).

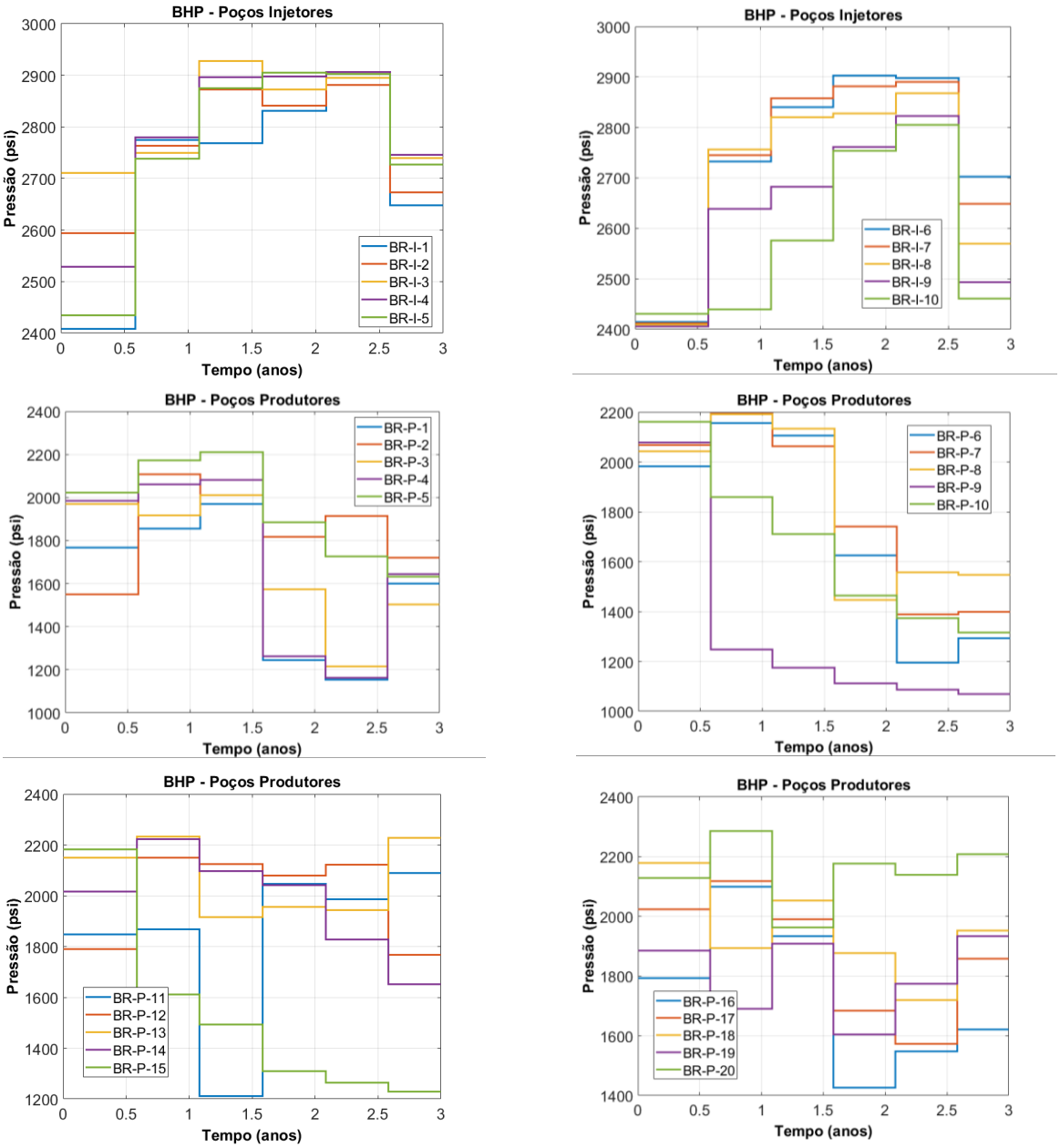
Figura 51 – Viabilidade das restrições de estado (à esquerda) e resultado da otimização via TPWL/POD (à direita) para produção líquida do campo, para o Caso 1.



Fonte: O Autor (2020).

Na Figura 52, é apresentada a trajetória ótima dos BHP's de cada poço injetor e produtor, obtidos após o processo de otimização. Observa-se uma tendência de se fechar alguns dos poços injetores no primeiro ciclo de controle, visto que seus controles nesse instante tendem à pressão média inicial do reservatório (2400 psi), enquanto que as pressões de fundo dos poços produtores tendem ao limite máximo estabelecido (3000 psi), a fim de maximizar a produção de óleo do campo.

Figura 52 – Trajetória ótima de BHP aplicada aos poços do modelo Brugge, para o Caso 1.



Fonte: O Autor (2020).

5.5.2 Modelo TPWL/POD: Caso 2 – Projeção Petrov-Galerkin

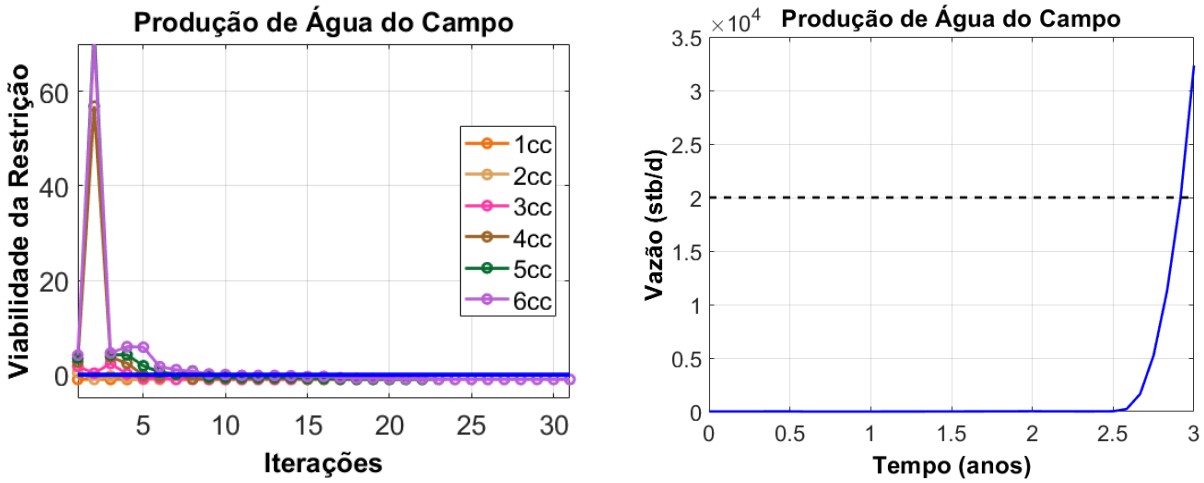
A Figura 53 ilustra o histórico da produção acumulada de óleo durante o processo de otimização, com a utilização da projeção de Petrov-Galerkin, ao longo de 30 iterações, número máximo estabelecido como critério de parada. Semelhante ao Caso 1, observa-se inicialmente oscilações no valor da função objetivo, a fim de que ambas as restrições de estado sejam atendidas, como se pode notar nas Figura 54 e Figura 55.

Figura 53 – Histórico da evolução da produção acumulada de óleo, $C_{oil,field}$, obtida no Caso 2.

Fonte: O Autor (2020).

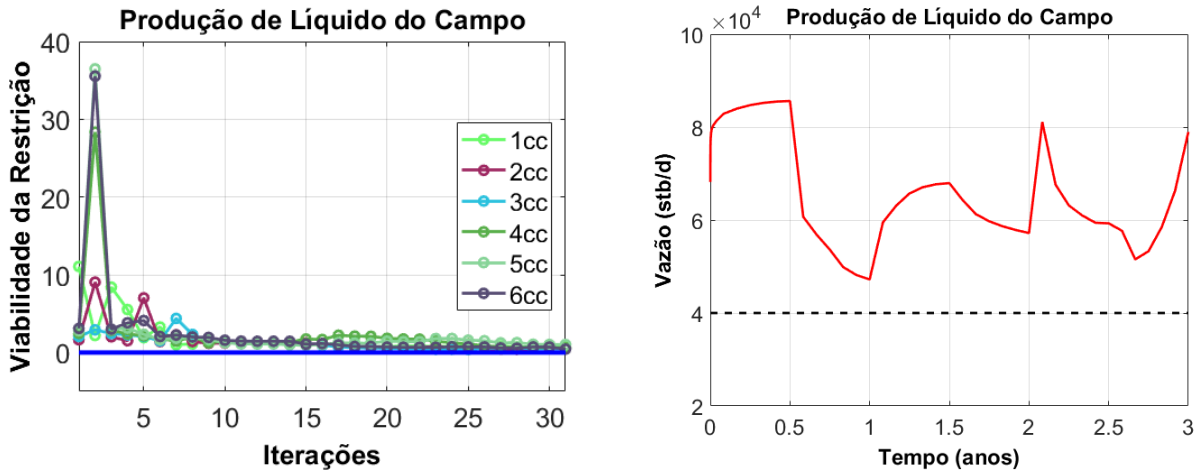
Na Figura 54, observa-se que, após a otimização, as restrições quanto à produção de água no campo foram satisfeitas em todos os pontos onde há mudança nos controles. Já as restrições de líquido, embora não tenham sido satisfeitas após 30 iterações nesses pontos (ver Figura 55), observa-se uma excessiva produção de líquido após a mudança do controle, o que demonstra novamente, de maneira análoga ao Caso 1, a tendência das restrições não serem respeitadas no interior de cada ciclo, devido à descontinuidade dos controles (BHP) nos seus extremos, como é possível notar no lado direito da Figura 54, onde a restrição relativa à produção de água é satisfeita no instante em que há mudança nos controles, porém é violada no seu interior.

Figura 54 – Viabilidade das restrições de estado (à esquerda) e resultado da otimização via TPWL/POD (à direita) para produção de água do campo, para o Caso 2.



Fonte: O Autor (2020).

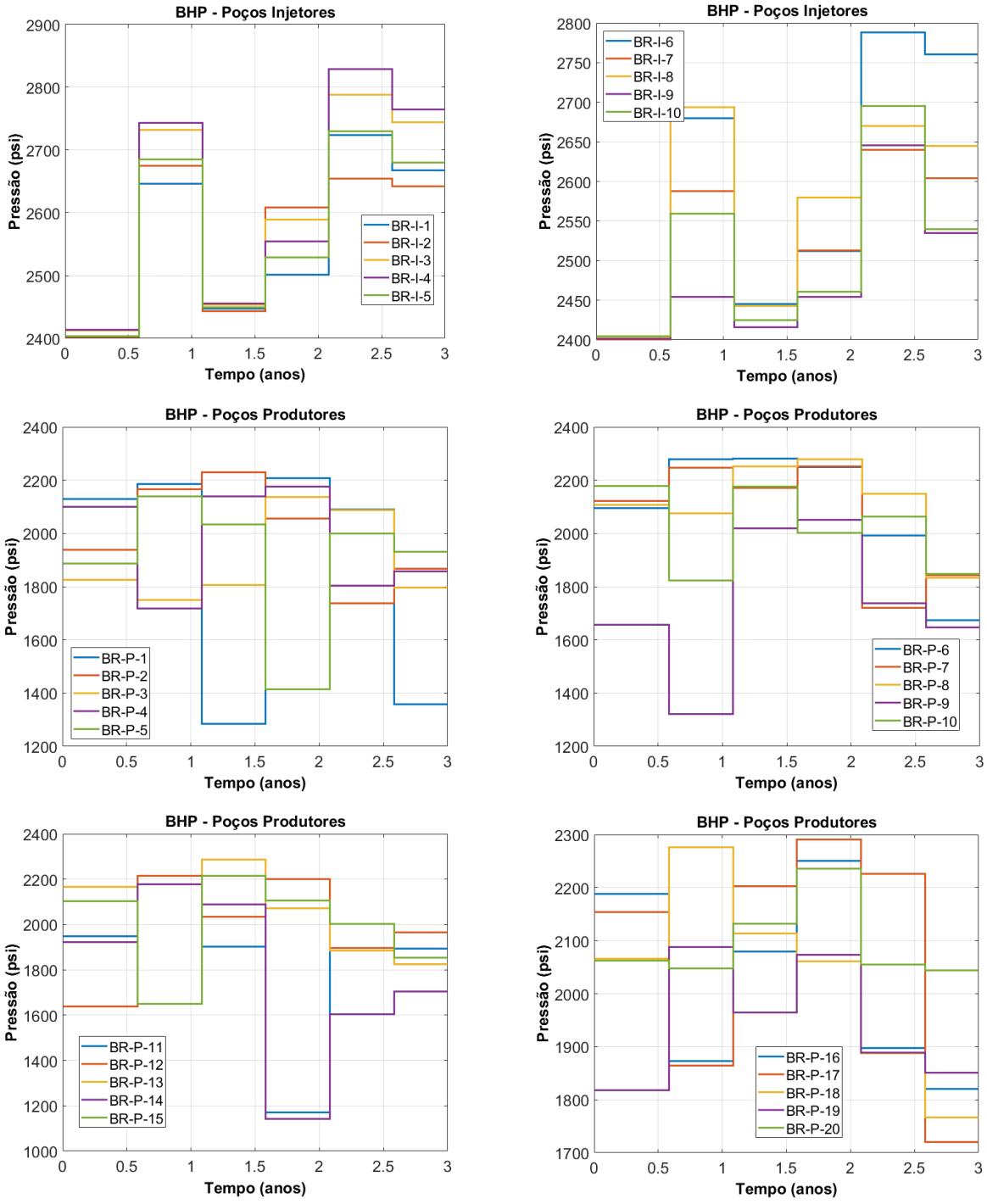
Figura 55 – Viabilidade das restrições de estado (à esquerda) e resultado da otimização via TPWL/POD (à direita) para produção líquida do campo, para o Caso 2.



Fonte: O Autor (2020).

Na Figura 56, é apresentada a trajetória ótima dos BHP's de cada poço injetor e produtor, obtidos após o processo de otimização. Semelhante ao Caso 1, observa-se também uma tendência de fechamento dos poços injetores no primeiro ciclo de controle, visto que seus controles nesse instante tendem à pressão média inicial do reservatório (2400 psi), enquanto que as pressões de fundo dos poços produtores tendem ao limite máximo estabelecido (3000 psi), a fim de maximizar a produção de óleo do campo.

Figura 56 – Trajetória ótima de BHP aplicada aos poços do modelo Brugge, para o Caso 2.



Fonte: O Autor (2020).

Para o Caso 2, possivelmente um maior número de iterações levaria às restrições de estado nos extremos de cada ciclo de controle serem satisfeitas, e a fim de atendê-las no interior de cada ciclo, uma alternativa seria impor tais restrições a cada passo de tempo de simulação, o que demandaria grande esforço computacional. Para isso, trabalhos futuros sugerem a estratégia de impor “pontos de correção”, isto é, restrições igualmente espaçadas no interior dos ciclos de controles, para que os limites estabelecidos sejam atendidos o máximo possível.

Fez-se então a análise do uso das projeções de Bubnov-Galerkin e de Petrov-Galerkin para o mesmo problema de otimização, definidos pelos Casos 1 e 2, respectivamente. Informações quanto ao número de iterações, número de avaliações de função e o tempo requerido para cada caso encontram-se resumidos na Tabela 5. No Caso 1, o fim do processo de otimização ocorreu devido ao passo da iteração ser menor que o valor estabelecido para a tolerância entre duas iterações consecutivas. Já para o Caso 2, o número máximo de iterações estabelecido foi atingido, servindo de critério de parada para o processo.

Tabela 5 – Comparativo dos resultados da otimização obtidos com o modelo TPWL/POD, utilizando as projeções de Bubnov-Galerkin e Petrov-Galerkin.

Modelo TPWL/POD	**Bubnov-Galerkin**	**Petrov-Galerkin**
Número de iterações	20	30
Número de avaliações de função	3775	5641
Tempo decorrido (segundos)	227948,882	390277,373

Fonte: O Autor (2020).

6 MÉTODO DE ORDEM REDUZIDA: DMD

Os avanços nas tecnologias de *hardware*, computação em nuvem, gerenciamento de dados, inovações em algoritmos de ciência de dados e o aumento do número de sensores disponíveis marcaram a nova era da análise de dados (*Data Analytics*) nos últimos anos. A análise de dados, também conhecida por *Data Analytics*, é um processo de coleta, limpeza, transformação e modelagem de dados para descobrir informações úteis que podem ser usadas para, entre outros objetivos, o diagnóstico, isto é, a análise de dados passados e atuais para investigar a causa de um problema e explicar por que aconteceu, e ainda fazer predições de cenários prováveis que podem acontecer.

Em geral, a análise de dados e os métodos orientados a dados (*Data-Driven Methods*) são uma coleção de ferramentas que englobam coleta, validação, agregação, processamento e análise de dados para extrair *insights* do passado, prever desempenho futuro e recomendar ações para decisões ideais com base em cenários possíveis. O processo de descobrir padrões recorrentes, bem como modelos descritivos e preditivos a partir de dados em larga escala, é o cerne dos modelos baseados em dados. Geralmente, eles são construídos usando dados sozinhos e requerem um entendimento de variáveis dependentes e independentes. Correlação não é causalidade, mas muitas vezes pode levar a uma melhor compreensão da causalidade. O reconhecimento de padrões recorrentes pode direcionar uma área para investigação e ajudar a entender o processo de causa e efeito.

O impulsionamento da ciência de dados moderna é dado pela disponibilidade de vastas e crescentes quantidades de dados, possibilitado por inovações notáveis em sensores de baixo custo, o aumento da capacidade de processamento computacional e recursos praticamente ilimitados de armazenamento e transferência de dados. Tais vastas quantidades de dados estão oferecendo aos engenheiros e cientistas de todas as disciplinas novas oportunidades para descobertas baseadas em dados, comumente referida como o quarto paradigma da descoberta científica.

Um método que vem ganhando destaque na comunidade científica é o *Dynamic Mode Decomposition* (DMD). O DMD é uma técnica de decomposição de matrizes

altamente versátil e baseia-se no poder da Decomposição em Valores Singulares (*Singular Value Decomposition*, SVD). As estruturas de menor dimensionalidade extraídas do DMD, no entanto, estão associadas a aspectos temporais e espaciais, considerando o impacto do *Principal Component Analysis* (PCA), visto em capítulos anteriores, para entender a importância de métodos de decomposição de matrizes versáteis e caracterização de séries temporais para análise de dados. A técnica DMD pode ser ainda aplicada a uma variedade de disciplinas, podendo ser usado com sucesso para previsão, estimativa de estados e controles de complexos sistemas.

DMD é, portanto, um método poderoso orientado a dados para analisar sistemas complexos, e por não precisar de modificação do código fonte do simulador utilizado, trata-se de um método não-intrusivo, diferentemente do método TPWL/POD, visto anteriormente. Contudo, ele por si só não leva em consideração uma possível atuação externa no sistema. Surge então o *Dynamic Mode Decomposition with Control* (DMDc), que incorpora ao DMD original a habilidade de analisar dados de medição coletados a partir de sistemas complexos onde forças externas, representadas neste trabalho pelos controles dos poços, são aplicadas durante o período de observação. Maiores detalhes sobre sua implementação e exemplos de aplicação são encontrados em (Kutz et al., 2016) e (Kutz J. , 2013).

Para demonstrar a eficácia do método DMDc na área de engenharia de reservatórios, aplicou-se esta técnica ao modelo Brugge, descrito no Capítulo 5, submetido a novos controles, a partir de dados gerados via simulação, utilizando novamente a caixa de ferramentas MATLAB *Reservoir Simulation Toolbox* (MRST), e a partir deles extrair padrões recorrentes e dominantes de menor dimensão, a serem utilizados para a construção de um modelo linear que melhor se ajusta a esse conjunto de dados. Além da recuperação dos estados salvos e utilizados para a construção do modelo linear, demonstra-se que também é possível fazer previsões quanto a estados futuros do sistema, para qualquer instante de tempo.

6.1 *Dynamic Mode Decomposition*

Muitos dos sistemas de interesse atualmente podem ser caracterizados como sistemas dinâmicos não-lineares de alta dimensionalidade que exibem fenômenos multiescala no espaço e no tempo. Por mais complexos que sejam, a evolução da maioria desses sistemas se dá através de um conjunto de valores numéricos de menor dimensão que podem ser caracterizados por estruturas espaço-temporais coerentes.

O crescente sucesso da técnica *Dynamic Mode Decomposition* (DMD) deriva do fato de ser um método livre de equações e puramente orientado a dados, capaz de fornecer uma decomposição precisa de um sistema complexo em estruturas espaço-temporais coerentes que podem ser utilizadas para a realização de previsões e controles futuros em curto prazo.

O método DMD pode ser pensado como uma combinação ideal de técnicas de redução de dimensionalidade espacial, como, por exemplo, a técnica *Proper Orthogonal Decomposition* (POD), cuja erudição foi feita ao longo do Capítulo 4. Assim, os modos espaciais correlacionados entre si também estão agora associados a uma determinada frequência temporal, possivelmente a uma taxa de crescimento ou decaimento. O DMD é algoritmicamente uma regressão de dados em uma dinâmica linear local, definida pela Equação (6.1).

$$\boldsymbol{x_{k+1}} = \mathbf{A}\boldsymbol{x_k} \tag{6.1}$$

Onde a matriz $\mathbf{A}$ é escolhida a fim de minimizar $\|\boldsymbol{x}_{k+1} - \mathbf{A}\boldsymbol{x}_k\|_2$ ao longo dos $k = 1,\ 2,\ 3, \ldots, m-1$ *snapshots*.

O algoritmo DMD apresenta uma gama de possibilidades de uso e interpretações, mas possui, mais especificamente, três tarefas principais:

- *Diagnóstico*: Em particular, o algoritmo extrai as principais estruturas espaço-temporais de baixa dimensionalidade de muitos sistemas de alta dimensionalidade, permitindo resultados fisicamente interpretáveis em termos de estruturas espaciais e suas respostas temporais associadas, análogas à análise obtida pelo POD;

- *Estimativas de estados e previsões de estados futuros*: o DMD limita-se à construção de um sistema dinâmico linear que melhor se ajusta, por mínimos quadrados, ao sistema dinâmico não-linear que gera os dados. Ao contrário do diagnóstico, o objetivo é prever o estado do sistema em um regime em que nenhuma medida tenha sido realizada. Essa abordagem de modelo generativo pode ser usada para previsões de estados futuros de sistemas dinâmicos e foi usada com sucesso em muitas áreas de aplicação;
- *Controle*: Considerando que um modelo dinâmico linear seja utilizado para prever o futuro de um sistema dinâmico não-linear, é razoável esperar que exista apenas uma janela limitada no futuro, talvez de curto prazo, na qual os dois modelos realmente convergem. Espera-se que essa janela de previsão seja longa o suficiente para permitir uma decisão de controle capaz de influenciar o estado futuro do sistema.

Considere, a partir de então, os dados coletados de um sistema dinâmico, expresso de modo genérico pela Equação (6.2).

$$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{f}(\boldsymbol{x}, t, \mu) \tag{6.2}$$

Onde $\boldsymbol{x}(t) \in \mathbb{R}^n$ é um vetor que representa o estado de nosso sistema dinâmico no instante t (geralmente bastante grande, com a dimensão $n \gg 1$), μ contém parâmetros do sistema e $\boldsymbol{f}(\cdot)$ representa a sua dinâmica.

Geralmente, um sistema dinâmico é representado como um sistema acoplado de equações diferenciais ordinárias que geralmente são não-lineares. Por fim, a dinâmica do sistema em função do tempo na sua forma contínua, dada pela Equação (6.2), pode induzir a uma representação discreta correspondente, na qual são feitas amostras do sistema a cada instante Δt e denota-se o tempo com um subíndice, onde $\boldsymbol{x}_k = \boldsymbol{x}(k \cdot \Delta t)$.

Denota-se o *mapa de fluxo* discretizado no tempo obtido pela evolução dada pela Equação (6.2) para cada Δt por $\mathbf{F}$, expresso pela Equação (6.3).

$$\boldsymbol{x}_{k+1} = \mathbf{F}(\boldsymbol{x}_k) \tag{6.3}$$

As medições do sistema, dadas pela Equação (6.3), são coletadas nos instantes t_k, onde $k = 1, 2, 3, \dots, m-1$, para um total de m medições (*snapshots*).

$$\boldsymbol{y}_k = \mathbf{g}(\boldsymbol{x}_k) \tag{6.4}$$

Em muitas aplicações, as medições são simplesmente os estados, de modo que $\boldsymbol{y}_k = \boldsymbol{x}_k$. A técnica DMD adota uma perspectiva livre de equações, onde a dinâmica $\boldsymbol{f}(\boldsymbol{x}, t, \mu)$ pode ser desconhecida. Assim, apenas as medições de dados do sistema são utilizadas para aproximar a dinâmica do sistema e fazer previsões de estados futuros.

O DMD então idealiza um sistema dinâmico linear local aproximado, definido pela Equação (6.5), com condição inicial $\boldsymbol{x}(0)$ e solução conhecida, dada pela Equação (6.6).

$$\frac{d\boldsymbol{x}}{dt} = \mathbb{A}\boldsymbol{x} \tag{6.5}$$

$$\boldsymbol{x}(t) = \sum_{k=1}^{n} \phi_k \cdot exp\,(\omega_k t) \cdot b_k = \boldsymbol{\Phi}\, exp(\boldsymbol{\Omega} t)\, \mathbf{b} \tag{6.6}$$

Onde ϕ_k e ω_k são os autovetores e autovalores da matriz $\mathbb{A}$, e os coeficientes b_k são as coordenadas de $\boldsymbol{x}(0)$ na base dos autovetores.

Dada a dinâmica contínua como na Equação (6.6), é sempre possível descrever um sistema discretizado no tempo análogo, composto por amostras a cada Δt no tempo, conforme Equação (6.7), onde a matriz $\mathbb{A}$ refere-se à dinâmica contínua no tempo, dada pela Equação (6.8).

$$\boldsymbol{x}_{k+1} = \mathbf{A}\boldsymbol{x}_k, \tag{6.7}$$

$$\mathbf{A} = exp(\mathbb{A}\Delta t). \tag{6.8}$$

A solução para esse sistema pode ser expressa simplesmente em termos dos autovalores λ_j e autovetores ϕ_k do mapa discretizado no tempo, dado pela matriz $\mathbf{A}$, onde $\mathbf{b}$ são os coeficientes da condição inicial $\boldsymbol{x}_1$ na base dos autovetores, de modo que $\boldsymbol{x}_1 = \boldsymbol{\Phi}\mathbf{b}$.

$$\boldsymbol{x}_k = \sum_{j=1}^{r} \phi_j \cdot \lambda_j^k \cdot b_j = \boldsymbol{\Phi} \cdot \boldsymbol{\Lambda}^k \cdot \mathbf{b} \tag{6.9}$$

O algoritmo DMD produz uma decomposição de baixa dimensionalidade, dada pela Equação (6.9), da matriz $\mathbf{A}$ que se ajusta perfeitamente na trajetória das medições $\boldsymbol{x}_k$ para $k = 1,\ 2,\ 3, \ldots, m$, por mínimos quadrados, para que a expressão dada pela Equação (6.10) seja minimizada em todos os pontos, para $k = 1,\ 2,\ 3, \ldots, m-1$.

$$\|\boldsymbol{x}_{k+1} - \mathbf{A}\boldsymbol{x}_k\|_{\,2} \tag{6.10}$$

Para minimizar o erro de aproximação, dado pela Equação (6.10), em todos os *snapshots* de $k = 1,\ 2,\ 3, \ldots, m$, é preciso organizá-los em duas grandes matrizes de dados, definidas pela Equação (6.11).

$$\mathbf{X} = \begin{bmatrix} \vdots & \vdots & \vdots & \vdots \\ \boldsymbol{x}_1 & \boldsymbol{x}_2 & \cdots & \boldsymbol{x}_{m-1} \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} \qquad \mathbf{X}' = \begin{bmatrix} \vdots & \vdots & \vdots & \vdots \\ \boldsymbol{x}_2 & \boldsymbol{x}_3 & \cdots & \boldsymbol{x}_m \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} \tag{6.11}$$

Esses *snapshots* de dados são amostras de um sistema dinâmico não-linear, definido na Equação (6.2), para o qual se deseja encontrar uma aproximação linear local ideal. A aproximação linear local, dada pela Equação (6.7) pode ser escrita em termos dessas matrizes de dados, como expresso na Equação (6.12).

$$\mathbf{X}' \approx \mathbf{A}\mathbf{X} \tag{6.12}$$

A matriz que melhor se ajusta aos dados, por mínimos quadrados, é dada, por sua vez, pela Equação (6.13), onde $\dagger$ é a matriz pseudoinversa de Moore-Penrose. Esta solução minimiza o erro, dado pela Equação (6.14).

$$\mathbf{A} = \mathbf{X}'\mathbf{X}^{\dagger} \tag{6.13}$$

$$\left\|\mathbf{X}' - \mathbf{A}\mathbf{X}^{\dagger}\right\|_{\,F} \tag{6.14}$$

O operador $\|\cdot\|_{\,F}$ consiste na norma de Frobenius, dada pela Equação (6.15).

$$\|\mathbf{X}\|_{\,F} = \sqrt{\sum_{j=1}^{n}\sum_{k=1}^{m} {X_{jk}}^2} \tag{6.15}$$

Como na maioria dos casos a matriz $\mathbf{A}$ é de alta dimensionalidade, isto é, $n \gg m$, pode ser difícil sua representação e/ou decomposição. No entanto, o posto de $\mathbf{A}$ é no máximo $m-1$, visto que ela é construída como uma combinação linear das $m-1$ colunas de $\mathbf{X}'$.

Ao invés de resolver diretamente pela matriz $\mathbf{A}$, primeiro projetam-se os dados em um subespaço de menor dimensão definido por no máximo $m-1$ modos (POD *modes)*, e depois resolve-se por uma matriz $\widetilde{\mathbf{A}}$, de baixa dimensionalidade, que evolui ao longo dos POD *modes*. Usa-se então esse operador de baixa dimensão $\widetilde{\mathbf{A}}$ para reconstruir os principais autovetores e os autovalores diferentes de zero do operador de alta dimensão $\mathbf{A}$, sem nunca computar explicitamente essa matriz.

O método DMD fornece uma decomposição espaço-temporal de dados em um conjunto de modos dinâmicos derivados de *snapshots* ou medições de um determinado sistema ao longo do tempo. O processo de coleta de dados envolve dois parâmetros.

- n é o número de pontos espaciais salvos em cada instante de tempo;
- m é o número de *snapshots* salvos.

Seja um sistema dinâmico dado pela Equação (6.2) e dois conjuntos de dados como definidos pela Equação (6.11), de modo que $\boldsymbol{x}'_k = \mathbf{F}(\boldsymbol{x}_k)$, onde $\mathbf{F}$ é o mapa dado na Equação (6.3) correspondente à evolução da Equação (6.2) para cada instante Δt no tempo. O DMD calcula a decomposição em autovalores e autovetores do operador linear de melhor ajuste $\mathbf{A}$, que relaciona os dados através da relação $\mathbf{X}' \approx \mathbf{A}\mathbf{X}$, e que é obtido pela Equação (6.16).

$$\mathbf{A} = \mathbf{X}'\mathbf{X}^{\dagger} \tag{6.16}$$

Os DMD *modes*, também chamados de modos dinâmicos (*Dynamic Modes*), são os autovetores de $\mathbf{A}$ e cada DMD *mode*, por sua vez, corresponde a um autovalor de $\mathbf{A}$.

Na prática, quando a dimensão de estado n é grande, a matriz $\mathbf{A}$ torna-se intratável para analisar diretamente. Em vez disso, o DMD contorna a decomposição de autovalores e autovetores da matriz $\mathbf{A}$ considerando a sua representação reduzida em termos de uma matriz de menor posto $\widetilde{\mathbf{A}}$, projetada via POD. A seguir, é descrito com maiores detalhes o algoritmo do DMD.

i. Primeiramente, faz-se a SVD da matriz $\mathbf{X}$, conforme Equação (6.17).

$$\mathbf{X} \approx \mathbf{U\Sigma V}^{*} \tag{6.17}$$

Onde * denota a matriz conjugada transposta, $\mathbf{U} \in \mathbb{C}^{n\times r}$, $\mathbf{\Sigma} \in \mathbb{C}^{r\times r}$, e $\mathbf{V} \in \mathbb{C}^{m\times r}$. Aqui r é o posto da aproximação reduzida de $\mathbf{X}$, via SVD. Os chamados *vetores singulares à esquerda* $\mathbf{U}$ são os POD *modes*.

A redução imposta via SVD executa um truncamento de menor posto dos dados para um número limitado de modos dominantes. O posto r da aproximação pode ser obtido pelo algoritmo de Gavish e Donoho (Brunton & Kutz, 2019).

ii. A matriz $\mathbf{A}$ da Equação (6.16) pode ser obtida usando a pseudoinversa de $\mathbf{X}$ obtida via SVD, conforme Equação (6.18).

$$\mathbf{A} = \mathbf{X'V\Sigma^{-1}U}^{*} \tag{6.18}$$

Na prática, é mais computacionalmente eficiente calcular $\widetilde{\mathbf{A}}$, isto é, a projeção $r \times r$ da matriz completa $\mathbf{A}$ no subespaço gerado pelos POD *modes*. Tal projeção é definida pela Equação (6.19).

$$\widetilde{\mathbf{A}} = \mathbf{U}^{*}\mathbf{AU} = \mathbf{U}^{*}\mathbf{X'V\Sigma^{-1}} \tag{6.19}$$

A matriz $\widetilde{\mathbf{A}}$ define um modelo linear de baixa dimensionalidade do sistema dinâmico nas coordenadas do subespaço gerado pelo POD, dado pela Equação (6.20).

$$\widetilde{\boldsymbol{x}}_{k+1} = \widetilde{\mathbf{A}}\widetilde{\boldsymbol{x}}_{k} \tag{6.20}$$

É possível reconstruir os estados do espaço de alta dimensionalidade através da relação $\boldsymbol{x}_k = \mathbf{U}\widetilde{\boldsymbol{x}}_k$.

iii. Calcula-se a decomposição de autovalores e autovetores de $\widetilde{\mathbf{A}}$, conforme Equação (6.21), onde as colunas de $\mathbf{W}$ são os autovetores e $\mathbf{\Lambda}$ é uma matriz diagonal que contém os autovalores correspondentes λ_k.

$$\widetilde{\mathbf{A}}\mathbf{W} = \mathbf{W\Lambda} \tag{6.21}$$

iv. Finalmente, pode-se reconstruir a decomposição de $\mathbf{A}$ a partir de $\mathbf{W}$ e $\mathbf{\Lambda}$. Em particular, os autovalores de $\mathbf{A}$ são dados por $\mathbf{\Lambda}$ e os autovetores de $\mathbf{A}$ (DMD *modes*) são dados pelas colunas de $\mathbf{\Phi}$, definido pela Equação (6.22).

$$\mathbf{\Phi} = \mathbf{X'V\Sigma^{-1}W} \tag{6.22}$$

Os modos definidos na Equação (6.23) são frequentemente chamados de *exact DMD modes*, porque foi comprovado que esses são os autovetores exatos da matriz $\mathbf{A}$.

Com as aproximações de menor posto os autovalores e dos autovetores em mãos, a solução projetada para previsões futuras pode ser construída para todos os tempos futuros. Reescrevendo primeiro, por conveniência, $\omega_k = ln(\lambda_k)/\Delta t$, a solução aproximada em todos os instantes de tempo futuros é dada pela Equação (6.23).

$$\boldsymbol{x}(t) \approx \sum_{k=1}^{n} \phi_k \cdot exp\,(\omega_k t) \cdot b_k = \mathbf{\Phi}\, exp(\mathbf{\Omega} t)\, \mathbf{b} \tag{6.23}$$

Onde b_k é a amplitude inicial de cada modo, $\mathbf{\Phi}$ é a matriz cujas colunas são os autovetores (DMD *modes*) ϕ_k, e $\mathbf{\Omega} = diag(\omega)$ é uma matriz diagonal cujos elementos são os autovalores ω_k. Os autovetores ϕ_k têm o mesmo tamanho que o estado $\boldsymbol{x}$, e $\mathbf{b}$ é o vetor dos coeficientes b_k.

É possível interpretar a Equação (6.23) como um ajuste, ou regressão, por mínimos quadrados de um sistema dinâmico linear dado por $\boldsymbol{x}_{k+1} = \mathbf{A}\boldsymbol{x}_k$ para um conjunto de dados amostrados. A matriz $\mathbf{A}$ é construída de modo que o erro $\|\boldsymbol{x}_{k+1} - \mathbf{A}\boldsymbol{x}_k\|_{\,2}$ seja minimizado ao longo de todos os *snapshots*.

Resta apenas calcular os valores dos coeficientes iniciais b_k. Seja o *snapshot* inicial $\boldsymbol{x}_1$ no momento $t_1 = 0$, a Equação (6.23) fornece $\boldsymbol{x}_1 = \mathbf{\Phi b}$. Em geral, a matriz de autovetores $\mathbf{\Phi}$ não é uma matriz quadrada, de modo que as condições iniciais, obtidas pela Equação (6.24), podem ser encontradas usando a sua pseudoinversa.

$$\mathbf{b} = \mathbf{\Phi}^{\dagger}\boldsymbol{x}_1 \tag{6.24}$$

De fato, $\mathbf{\Phi}^{\dagger}$ denota a pseudoinversa de Moore-Penrose que pode ser acessada no MATLAB através do comando `pinv`. A pseudoinversa é equivalente a encontrar a solução $\mathbf{b}$ mais adequada por mínimos quadrados.

O algoritmo DMD apresentado aqui utiliza como vantagem a baixa dimensionalidade dos dados para fazer uma aproximação de menor posto do mapa linear que melhor se aproxima da dinâmica não-linear dos dados coletados para o sistema. Feito isso, realizar previsões de estados futuros do sistema é possível para quaisquer instantes de tempo futuros.

Diferentemente do método POD-Galerkin, que exige a solução de um conjunto de variáveis dinâmicas de baixa dimensionalidade para prever estados futuros, no DMD, nenhum trabalho adicional é necessário para a previsão de estados futuros, a não ser inserir o instante de tempo desejado na Equação (6.23).

Assim, as vantagens do DMD giram em torno do fato de que: (i) é um procedimento livre de equações; (ii) é possível construir previsões de estados futuros para qualquer instante de tempo t (desde que a aproximação do DMD seja válida); e (iii) é formulado inteiramente em termos de dados de medição. Por esse motivo, pode ser aplicado a uma ampla gama de disciplinas.

6.2 *Dynamic Mode Decomposition with Control*

A modelagem de sistemas dinâmicos complexos e de alta dimensionalidade não se restringe somente a áreas como engenharia e ciências físicas, mas também se encontram em aplicações modernas, como esforços de erradicação de doenças infecciosas, por exemplo. O *Dynamic Mode Decomposition with Control* (DMDc) utiliza não somente as medições do sistema, como também as medições relativas ao controle externo aplicado a fim de extrair a dinâmica subjacente do sistema de maneira livre de equações.

A técnica DMDc herda as características vantajosas do DMD: opera a partir de dados que constituem os *snapshots*, e lida com eficiência com dados de medição de sistemas dinâmicos não-lineares de alta dimensionalidade. Contudo, o DMDc também é capaz de extrair características de entrada e saída do sistema, permitindo a construção de um conjunto de modelos de ordem reduzida (ROMs) que podem ser utilizados para prever e projetar controles desses sistemas complexos.

O método DMDc foi originalmente desenvolvido com a finalidade de analisar os dados coletados de sistemas complexos nos quais forças externas são aplicadas durante o período de observação, caracterizando a dinâmica subjacente e as propriedades de entrada e saída para o desenvolvimento de modelos preditivos e estratégias de controle a fim de conduzir o sistema a um estado específico. O objetivo do DMDc é, portanto, caracterizar a relação entre três medidas: a medida do estado atual $\boldsymbol{x_k}$, do estado futuro $\boldsymbol{x_{k+1}}$ e o controle atual $\boldsymbol{u_k}$. A relação pode ser descrita pelo sistema dinâmico linear discreto definido pela Equação (6.25).

$$\boldsymbol{x_{k+1}} = \mathbf{A}\boldsymbol{x_k} + \mathbf{B}\boldsymbol{u_k} \tag{6.25}$$

Onde $\boldsymbol{x_k} \in \mathbb{R}^n$, $\boldsymbol{u_k} \in \mathbb{R}^q$, $\mathbf{A} \in \mathbb{R}^{n\times n}$ e $\mathbf{B} \in \mathbb{R}^{n\times q}$. O operador $\mathbf{A}$ descreve a dinâmica do sistema na ausência de forças externas, enquanto que o operador $\mathbf{B}$ caracteriza o impacto do controle $\boldsymbol{u_k}$ no estado $\boldsymbol{x_{k+1}}$. O objetivo então é encontrar aproximações de $\mathbf{A}$ e $\mathbf{B}$ a partir de tais medições. As matrizes de *snapshots* $\mathbf{X}$ e $\mathbf{X}'$ são coletadas de modo semelhante ao DMD original, conforme Equação (6.11). Entretanto, uma nova matriz de *snapshots* é construída para os controles, definida na Equação (6.26).

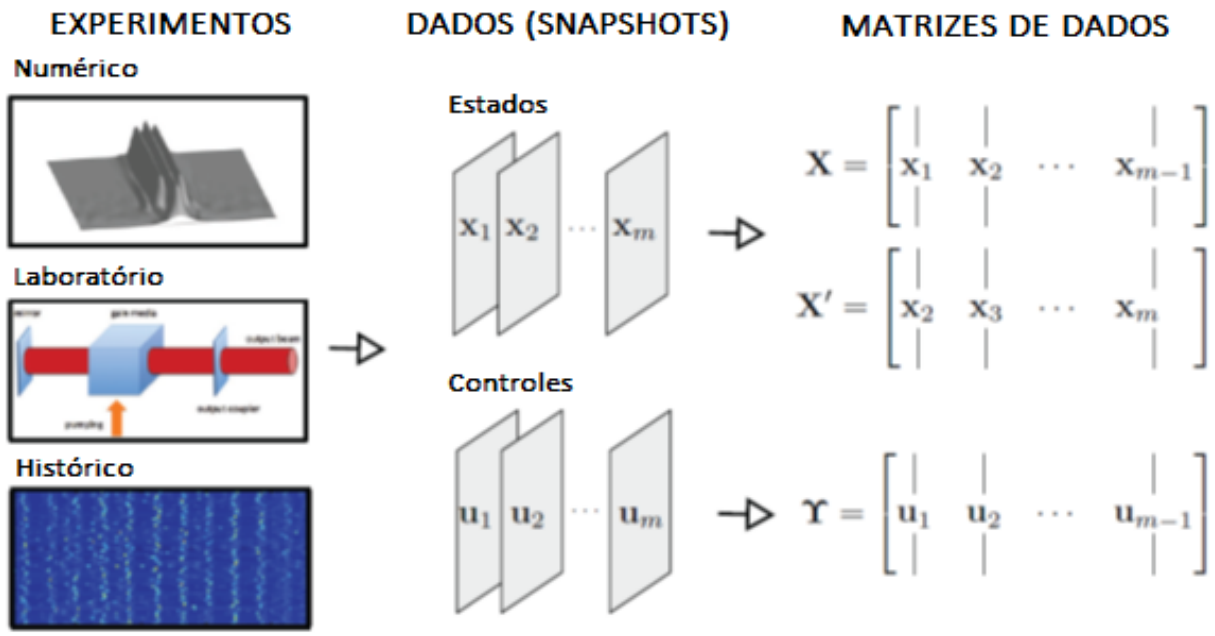
$$\boldsymbol{\Upsilon} = \begin{bmatrix} \vdots & \vdots & \vdots & \vdots \\ \boldsymbol{u}_1 & \boldsymbol{u}_2 & \cdots & \boldsymbol{u}_{m-1} \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} \tag{6.26}$$

Logo, a Equação (6.12) pode ser reescrita conforme Equação (6.27).

$$\mathbf{X}' \approx \mathbf{A}\mathbf{X} + \mathbf{B}\boldsymbol{\Upsilon} \tag{6.27}$$

A Figura 57 ilustra o esquema geral do DMDc, que tem como ponto de partida a coleta de dados de medição de estado e entrada de dados numéricos a partir de simulações, experimentos de laboratório e/ou registros históricos.

Figura 57 – Esquema geral do DMDc.



Fonte: (Brunton & Kutz, 2019) – Adaptador pelo Autor.

O DMDc utiliza as três matrizes de *snapshots* $\mathbf{X}$, $\mathbf{X}'$ e $\boldsymbol{\Upsilon}$ para descobrir os operadores $\mathbf{A}$ e $\mathbf{B}$, que melhor se ajustam por mínimos quadrados a esse conjunto de dados. O operador $\mathbf{B}$, por sua vez, pode ser conhecido para um dado sistema. A suposição de que $\mathbf{B}$ é conhecido ou estimado é uma visão idealista dos sistemas mais complexos, indicando uma quantidade significativa de conhecimento sobre como os controles afetam o sistema.

Logo, neste trabalho, adota-se a hipótese que $\mathbf{B}$ é desconhecido, o que, para um experimentalista ou analista de dados, é muito mais interessante, pois apenas os *snapshots* dos controles e dos estados do sistema são necessários para caracterizar o processo descrito pelo operador $\mathbf{A}$ e como esse processo é afetado pelos controles fornecidos pelo operador $\mathbf{B}$.

A relação aproximada entre as matrizes de dados da Equação (6.27) pode ser reescrita conforme a Equação (6.28).

$$\mathbf{X}' \approx \mathbf{G}\boldsymbol{\Omega} \tag{6.28}$$

Onde $\mathbf{G} = [\mathbf{A}\ \mathbf{B}]$ e $\boldsymbol{\Omega} = \begin{bmatrix}\mathbf{X}\\ \boldsymbol{\Upsilon}\end{bmatrix}$. Assim, semelhante à definição do DMD original, o DMDc do trio de medidas é a decomposição em autovalores e autovetores do operador $\mathbf{A}$ definidas pelas Equações (6.29) e (6.30).

$$\mathbf{G} = \mathbf{X}'\boldsymbol{\Omega}^{\dagger} \tag{6.29}$$

$$[\mathbf{A}\ \mathbf{B}] = \mathbf{X}' \begin{bmatrix}\mathbf{X}\\ \boldsymbol{\Upsilon}\end{bmatrix}^{\dagger} \tag{6.30}$$

Onde $\boldsymbol{\Omega}$ contém ambas as informações referentes aos *snapshots* relativos aos estados e aos controles do sistema. Aqui, semelhante ao DMD original, procura-se o operador $\mathbf{G}$ que melhor se ajusta à dinâmica do sistema, dada pelo operador $\mathbf{A}$, e também à atuação de forças externas no sistema, dado pelo operador $\mathbf{B}$. Além disso, uma solução por mínimos quadrados $\mathbf{G}$ para o problema indeterminado $\mathbf{X}' \approx \mathbf{G}\boldsymbol{\Omega}$ pode ser encontrada minimizando-se a norma Frobenius, $\|\mathbf{X}' - \mathbf{G}\boldsymbol{\Omega}\|_{\boldsymbol{F}}$.

Entretanto, para obter o operador $\mathbf{G}$ de maneira apropriada, isto é, através de uma aproximação dessa matriz, utiliza-se novamente uma decomposição SVD, agora realizado na matriz de dados aumentada (espaço de entrada, ou *input space*), conforme Equação (6.31), com o valor de truncamento definido como p.

$$\boldsymbol{\Omega} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^{*} \approx \tilde{\mathbf{U}}\tilde{\boldsymbol{\Sigma}}\tilde{\mathbf{V}}^{*} \tag{6.31}$$

Observe que o valor de truncamento de $\boldsymbol{\Omega}$ deve ser maior que o de $\mathbf{X}$. A aproximação de $\mathbf{G}$ é dada pela Equação (6.32), onde $\mathbf{G} \in \mathbb{R}^{\boldsymbol{n}\times(\boldsymbol{n}+\boldsymbol{q})}$.

$$\mathbf{G} \approx \bar{\mathbf{G}} = \mathbf{X}'\tilde{\mathbf{V}}\tilde{\boldsymbol{\Sigma}}^{-1}\tilde{\mathbf{U}}^{*} \tag{6.32}$$

Agora é possível obter aproximações das matrizes $\mathbf{A}$ e $\mathbf{B}$, separando o operador linear $\tilde{\mathbf{U}}$ em dois componentes distintos, conforme expresso na Equação (6.33), onde $\tilde{\mathbf{U}}_1^{*} \in \mathbb{R}^{\boldsymbol{n}\times\boldsymbol{p}}$, $\tilde{\mathbf{U}}_2^{*} \in \mathbb{R}^{\boldsymbol{q}\times\boldsymbol{p}}$, e $\tilde{\mathbf{U}}^{*} = [\tilde{\mathbf{U}}_1^{*}\ \ \tilde{\mathbf{U}}_2^{*}]$.

$$\mathbf{G} = [\mathbf{A},\ \mathbf{B}] \approx [\bar{\mathbf{A}},\ \bar{\mathbf{B}}] = [\mathbf{X}'\tilde{\mathbf{V}}\tilde{\boldsymbol{\Sigma}}^{-1}\tilde{\mathbf{U}}_1^{*}, \mathbf{X}'\tilde{\mathbf{V}}\tilde{\boldsymbol{\Sigma}}^{-1}\tilde{\mathbf{U}}_2^{*}] \tag{6.33}$$

Diferentemente do DMD original, os vetores singulares à esquerda $\tilde{\mathbf{U}}$ não podem ser usados para definir o subespaço no qual os estados evoluem, porque os vetores

singulares à esquerda de $\boldsymbol{\Omega}$ definem o espaço de entrada. Logo, para encontrar uma transformação linear para a medição $\boldsymbol{x}$, utiliza-se um subespaço de ordem reduzida do espaço de saída (*output space*). Assim, é necessária uma segunda decomposição SVD, com o valor de truncamento definido como r, dada pela Equação (6.34).

$$\mathbf{X}' \approx \widehat{\mathbf{U}}\widehat{\mathbf{\Sigma}}\widehat{\mathbf{V}}^{*} \tag{6.34}$$

Onde $\widehat{\mathbf{U}} \in \mathbb{R}^{n\times r}$, $\widehat{\mathbf{\Sigma}} \in \mathbb{R}^{r\times r}$, and $\widehat{\mathbf{V}}^{*} \in \mathbb{R}^{r\times m-1}$. Note que as duas decomposições SVD provavelmente terão valores de truncamento diferentes das matrizes de entrada e saída p e r, respectivamente, e, portanto, $p > r$.

Usando a transformação $\boldsymbol{x} = \widehat{\mathbf{U}} \cdot \widetilde{\boldsymbol{x}}$, as seguintes aproximações de ordem reduzida para $\mathbf{A}$ e $\mathbf{B}$ podem ser calculadas, dadas pelas Equações (6.35) e (6.36), onde $\widetilde{\mathbf{A}} \in \mathbb{R}^{r\times r}$ e $\widetilde{\mathbf{B}} \in \mathbb{R}^{r\times q}$.

$$\widetilde{\mathbf{A}} = \widehat{\mathbf{U}}^{*}\overline{\mathbf{A}}\widehat{\mathbf{U}} = \widehat{\mathbf{U}}^{*}\mathbf{X}'\widetilde{\mathbf{V}}\widetilde{\mathbf{\Sigma}}^{-1}\widetilde{\mathbf{U}}_{\mathbf{1}}^{*}\widehat{\mathbf{U}} \tag{6.35}$$

$$\widetilde{\mathbf{B}} = \widehat{\mathbf{U}}^{*}\overline{\mathbf{B}} = \widehat{\mathbf{U}}^{*}\mathbf{X}'\widetilde{\mathbf{V}}\widetilde{\mathbf{\Sigma}}^{-1}\widetilde{\mathbf{U}}_{\mathbf{1}}^{*} \tag{6.36}$$

A relação linear descrita pela Equação (6.25) pode então ser reescrita na sua forma reduzida através da Equação (6.37).

$$\widetilde{\boldsymbol{x}}_{\boldsymbol{k+1}} = \widetilde{\mathbf{A}}\widetilde{\boldsymbol{x}}_{\boldsymbol{k}} + \widetilde{\mathbf{B}}\widetilde{\boldsymbol{u}}_{\boldsymbol{k}} \tag{6.37}$$

Semelhante ao DMD original, os modos dinâmicos de $\mathbf{A}$ podem ser calculados a partir da decomposição em autovalores e autovetores da matriz reduzida $\widetilde{\mathbf{A}}$, conforme Equação (6.38).

$$\widetilde{\mathbf{A}}\mathbf{W} = \mathbf{W}\mathbf{\Lambda} \tag{6.38}$$

A transformação dos autovetores obtidos pela Equação (6.38) em modos dinâmicos de $\mathbf{A}$ é ligeiramente diferente da Equação (6.22) e é dada pela Equação (6.39), onde a relação entre $\mathbf{\Phi}$ e $\mathbf{W}$ é semelhante ao DMD original.

$$\mathbf{\Phi} = \mathbf{X}'\widetilde{\mathbf{V}}\widetilde{\mathbf{\Sigma}}^{-1}\widetilde{\mathbf{U}}_{\mathbf{1}}^{*}\widehat{\mathbf{U}}\mathbf{W} \tag{6.39}$$

6.3 Exemplo de Aplicação: Modelo Brugge

No Capítulo 5, foi apresentado o modelo Brugge, que, devido as suas dimensões, mais se assemelha aos modelos encontrados em simulação de reservatórios na prática. Nesta seção, o objetivo será demonstrar a aplicabilidade do DMDc, uma técnica puramente orientada a dados, em modelos tais como o modelo Brugge, tanto na reconstrução dos estados utilizados para a construção do modelo DMD quanto para a previsão de estados futuros.

A geometria do modelo, a distribuição de poços e as propriedades físicas do modelo foram mantidas inalteradas em relação àquelas apresentadas na Figura 37. Para os casos aqui descritos, todos os poços apresentam uma única completação. De maneira análoga aos exemplos até aqui apresentados, foi realizada apenas uma única simulação de treinamento para cada caso. O tempo de simulação e o passo de tempo adotado durante o treinamento varia conforme os casos de estudo realizados:

- Caso 1: o modelo DMD é construído a partir de 100 *snapshots* obtidos de uma simulação de treinamento, com tempo total e passo de tempo iguais a 3500 dias e 35 dias, respectivamente, a fim de prever novos 100 *snapshots*, mantendo os controles dos poços constantes durante todo o tempo de simulação;
- Caso 2: o modelo DMD é construído a partir de 100 *snapshots* obtidos de uma simulação de treinamento (a mesma do Caso 1), a fim de prever novos 100 *snapshots*. Os controles dos poços na nova simulação variam ao longo de 20 ciclos de controles, com limites iguais a 3800 a 4200 stb/d, para os poços injetores, e 690 a 760 psi, para os poços produtores;
- Caso 3: para fins de comparação com o exemplo apresentado na seção 5.3, um modelo DMD é construído a partir de 180 *snapshots* obtidos de uma simulação de treinamento, com tempo total e passo de tempo iguais a 15 anos e 30 dias, respectivamente, a fim de serem previstos os 6 anos seguintes. Os controles dos poços durante o treinamento permaneceram constantes e iguais ao valor médio dos limites estabelecidos para cada poço, ao passo que, durante a nova simulação, tais controles variam

aleatoriamente ao longo de 14 ciclos de controle, conforme ilustrado na Figura 38.

Os três casos acima apresentados têm por objetivo a análise da acurácia do sistema preditivo de acordo com o número de *snapshots* utilizados durante a obtenção do modelo DMD, utilizando controles constantes e variáveis. Em todos eles, foi utilizado um critério de energia igual a 0,9999 para a redução de dimensionalidade via POD.

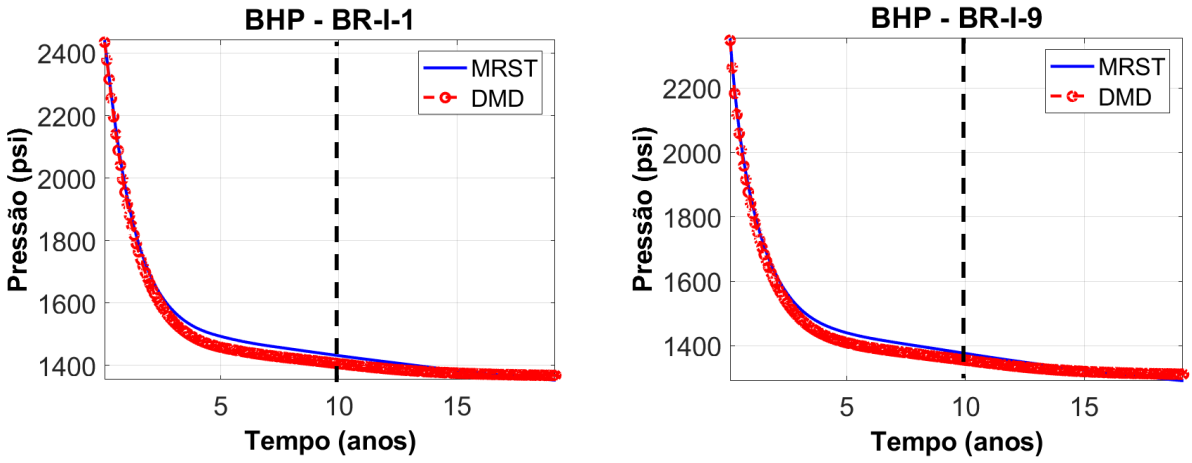
Ao longo das Figuras 58 à 64, é apresentada a performance do modelo DMD, para cada caso listado anteriormente, comparando as curvas de produção e injeção obtidas pela técnica DMD e pela simulação no MRST. Semelhante às simulações realizadas utilizando o TPWL/POD, foram obtidos s*peed-up's* da ordem de 2000, entre o tempo computacional requerido no modelo de alta fidelidade e o modelo DMDc. Os erros relativos, definidos nas Equações (4.31) a (4.33), encontram-se detalhados ao longo das Tabela 6 à 8.

6.3.1 Modelo DMD: Caso 1

Neste caso, o modelo DMD foi construído a partir de 100 *snapshots* obtidos via simulação no MRST, com tempo de simulação igual a 3500 dias e passo de tempo adotado igual a 35 dias. Os controles dos poços, durante o treinamento, foram mantidos constantes e iguais a 4000 stb/d, para os 10 poços injetores, e 725 psi, para os 20 poços produtores. O objetivo deste caso é executar uma nova simulação com tempo total igual a 7000 dias, a fim de se demonstrar a habilidade da técnica em prever estados futuros, submetendo o reservatório a novos controles, constantes e iguais a 4400 stb/d, para os poços injetores, e 655 psi, para os poços produtores.

Na Figura 58, são apresentadas as curvas resultantes para a trajetória dos BHP's apenas dos poços BR-I-1 e BR-I-9, visto que todas as curvas resultantes dos poços injetores são bastante semelhantes. A linha tracejada representa o instante a partir do qual o modelo realiza a previsão de estados futuros.

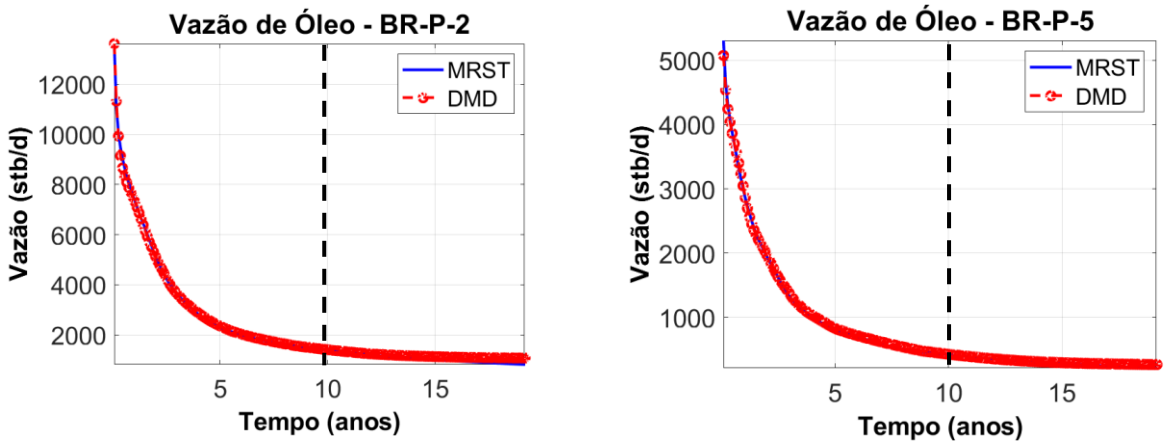
Figura 58 – Trajetórias de BHP dos poços injetores do modelo Brugge, para o Caso 1.

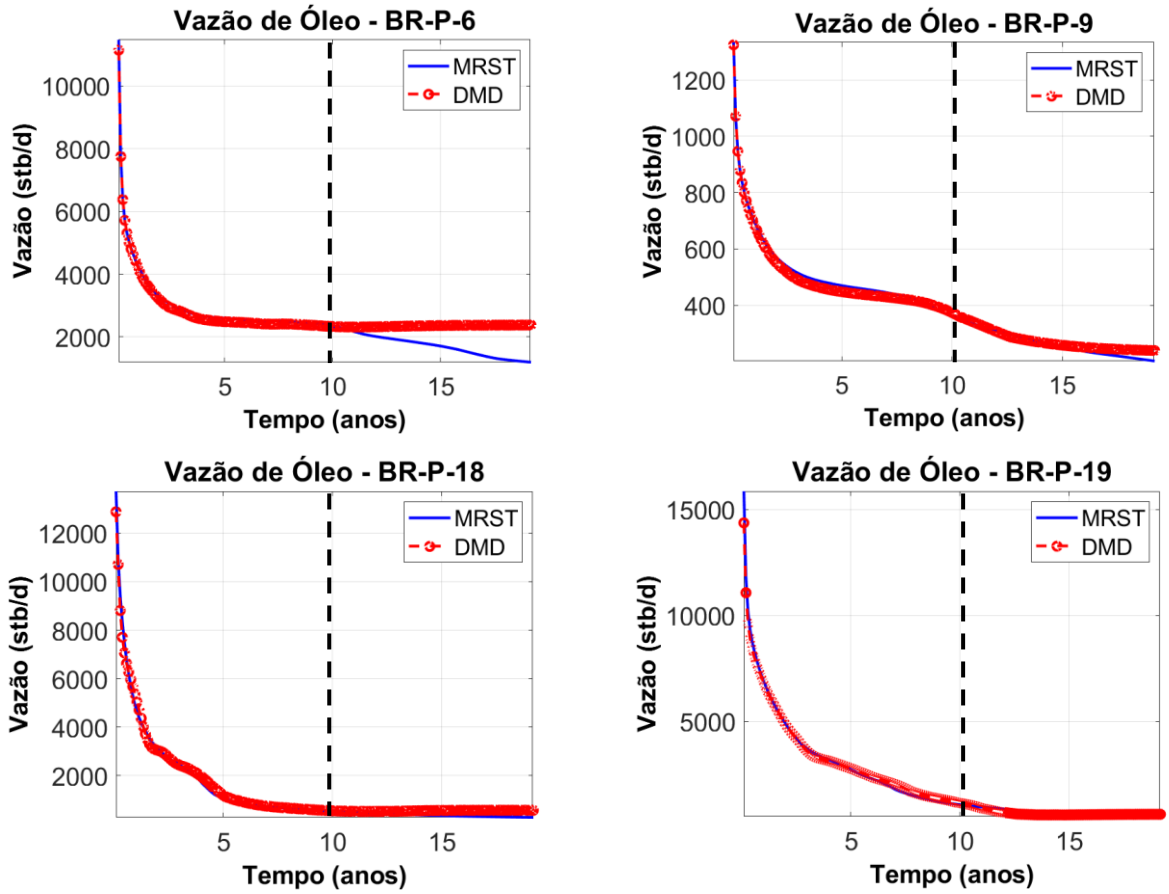


Fonte: O Autor (2020).

Na Figura 59, são apresentadas as curvas resultantes para a vazão de óleo de alguns dos poços produtores, visto que todas as curvas resultantes apresentam comportamento semelhante. Como se pode observar, por exemplo, no poço BR-P-6, o modelo preditivo falha significativamente ao longo das curvas de produção.

Figura 59 – Curvas de produção de óleo dos poços do modelo Brugge, para o Caso 1.

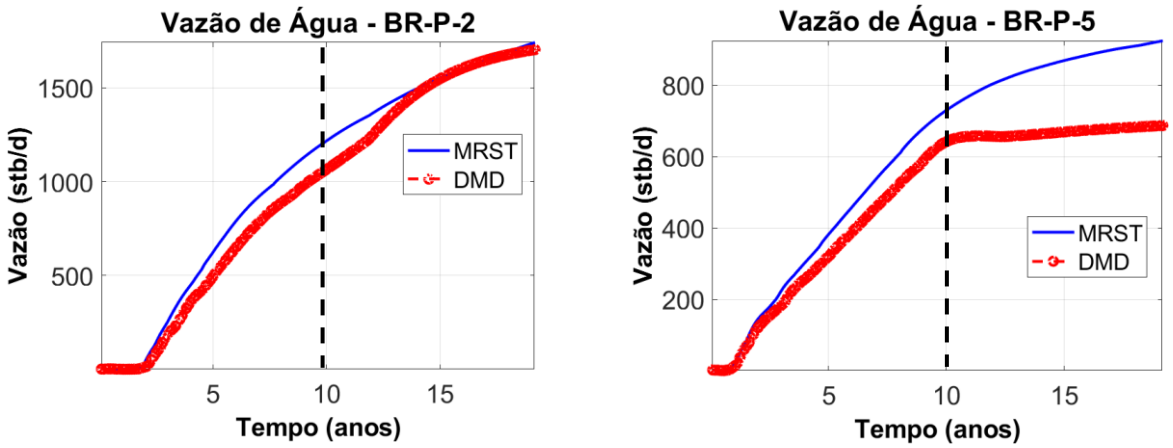


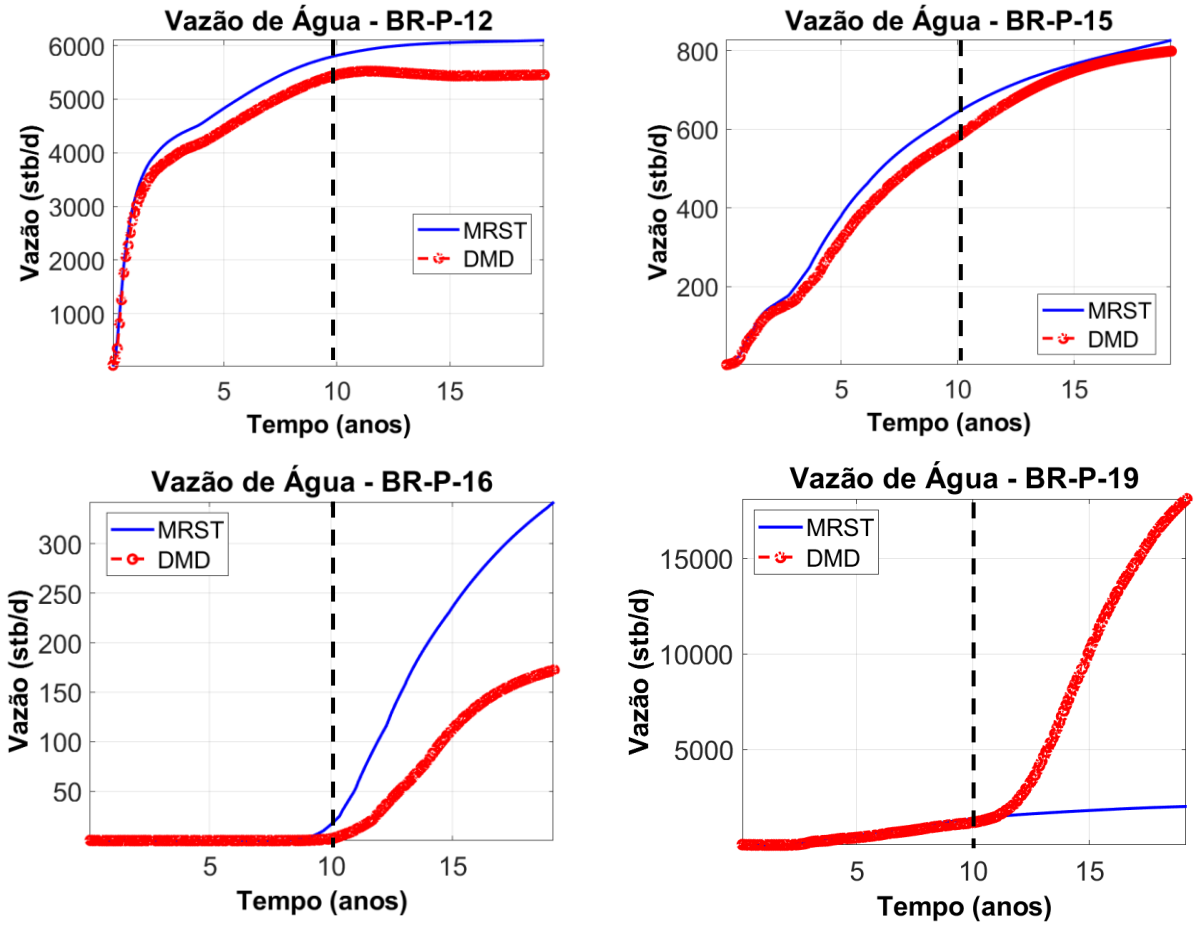


Fonte: O Autor (2020).

Na Figura 60, são apresentadas as curvas resultantes para a vazão de água de alguns dos poços produtores. Como se pode observar, em alguns poços, o modelo preditivo falha significativamente ao longo das curvas de produção.

Figura 60 – Curvas de produção de água dos poços do modelo Brugge, para o Caso 1.





Fonte: O Autor (2020).

Na Tabela 6, são apresentados os erros relativos das curvas apresentadas anteriormente, definidos ao longo das Equações (4.31) a (4.33), e o tempo computacional requerido para os dois tipos de simulação do Caso 1.

Tabela 6 – Erros e tempo computacional requerido, para o Caso 1 (Modelo Brugge).

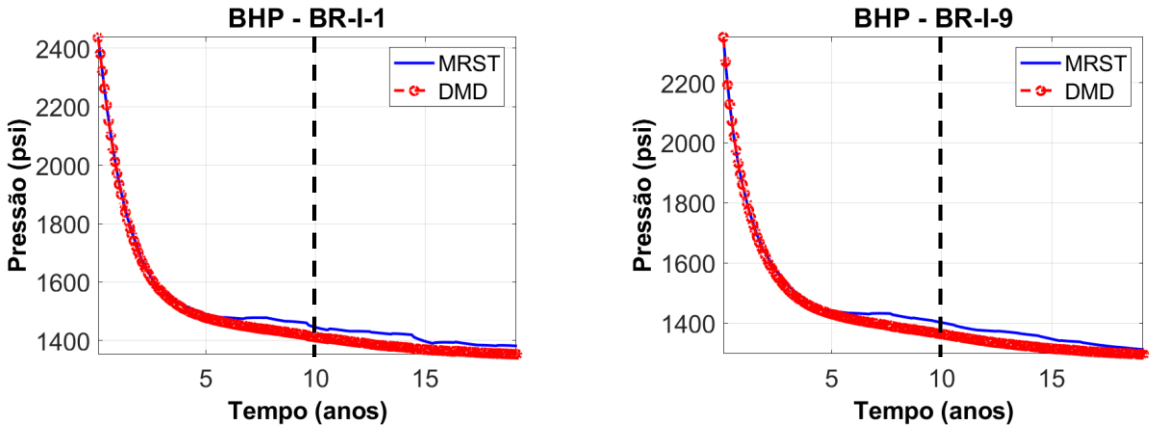
Tipo de Simulação	**Erro**		**Tempo Decorrido**
MRST	-		6849,64s
DMDc	$E_{bhp,inj}$	1,37%	3,4507s
	$E_{o,prod}$	11,93%	
	$E_{w,prod}$	28,64%	

Fonte: O Autor (2019).

6.3.2 Modelo DMD: Caso 2

Utilizando o mesmo modelo DMD do Caso 1, executou-se uma nova simulação de tempo total igual a 7000 dias, submetendo agora o reservatório a 20 ciclos de controle, cujos valores variam entre 3800 a 4200 stb/d, nos poços injetores, e 690 a 760 psi, nos poços produtores. Na Figura 61, são apresentadas as curvas resultantes para a trajetória dos BHP's apenas dos poços BR-I-1 e BR-I-9, visto que todas as curvas resultantes dos poços injetores são bastante semelhantes. A linha tracejada representa o instante a partir do qual o modelo realiza a previsão de estados futuros.

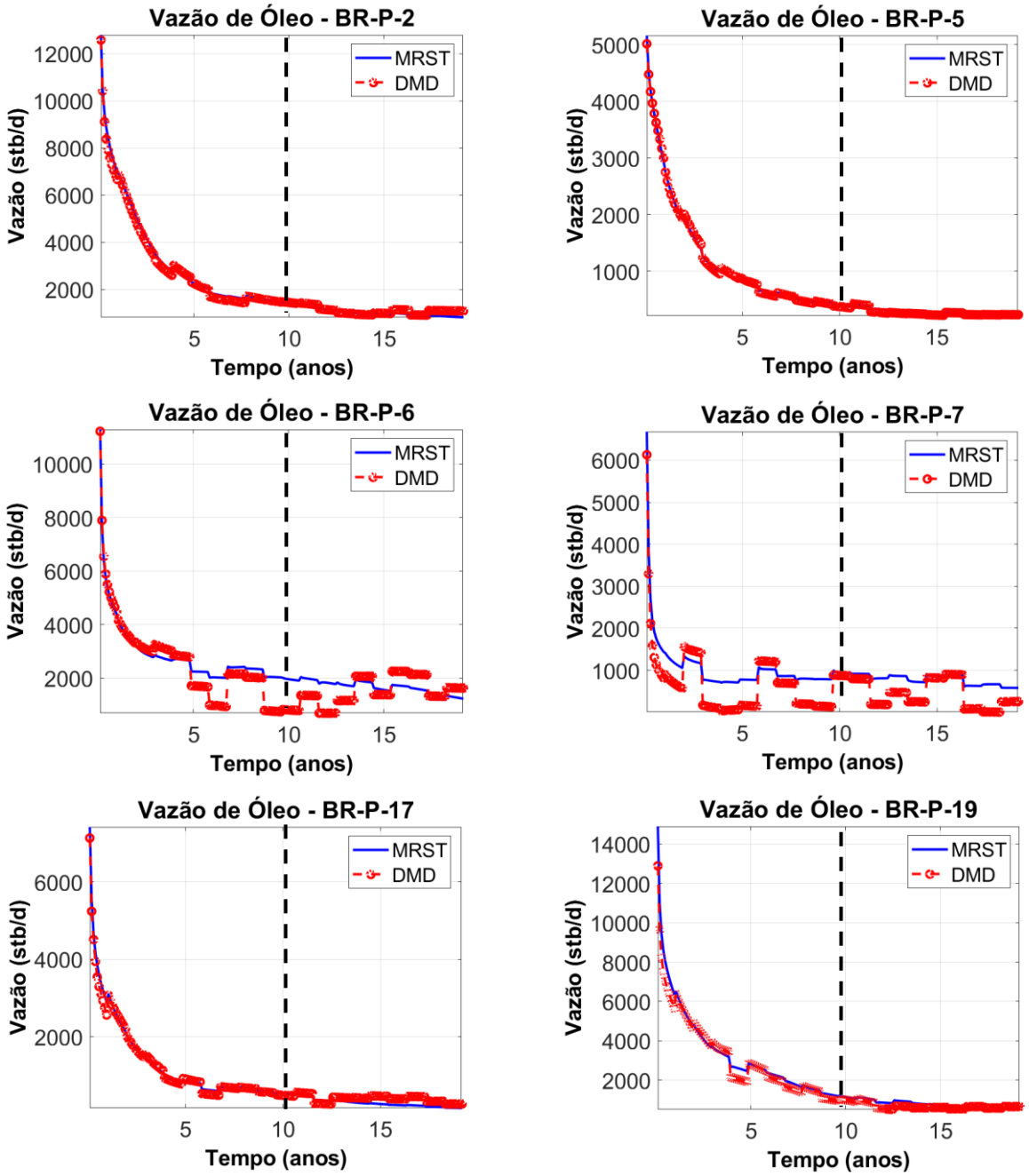
Figura 61 – Trajetórias de BHP dos poços injetores do modelo Brugge, para o Caso 2.



Fonte: O Autor (2020).

A Figura 62 ilustra as curvas resultantes de vazão de óleo de alguns dos poços produtores, visto que todas as curvas resultantes apresentam comportamento semelhante. Nota-se que há convergência entre as curvas para a maioria dos poços, exceto em alguns como o BR-P-6 e o BR-P-7.

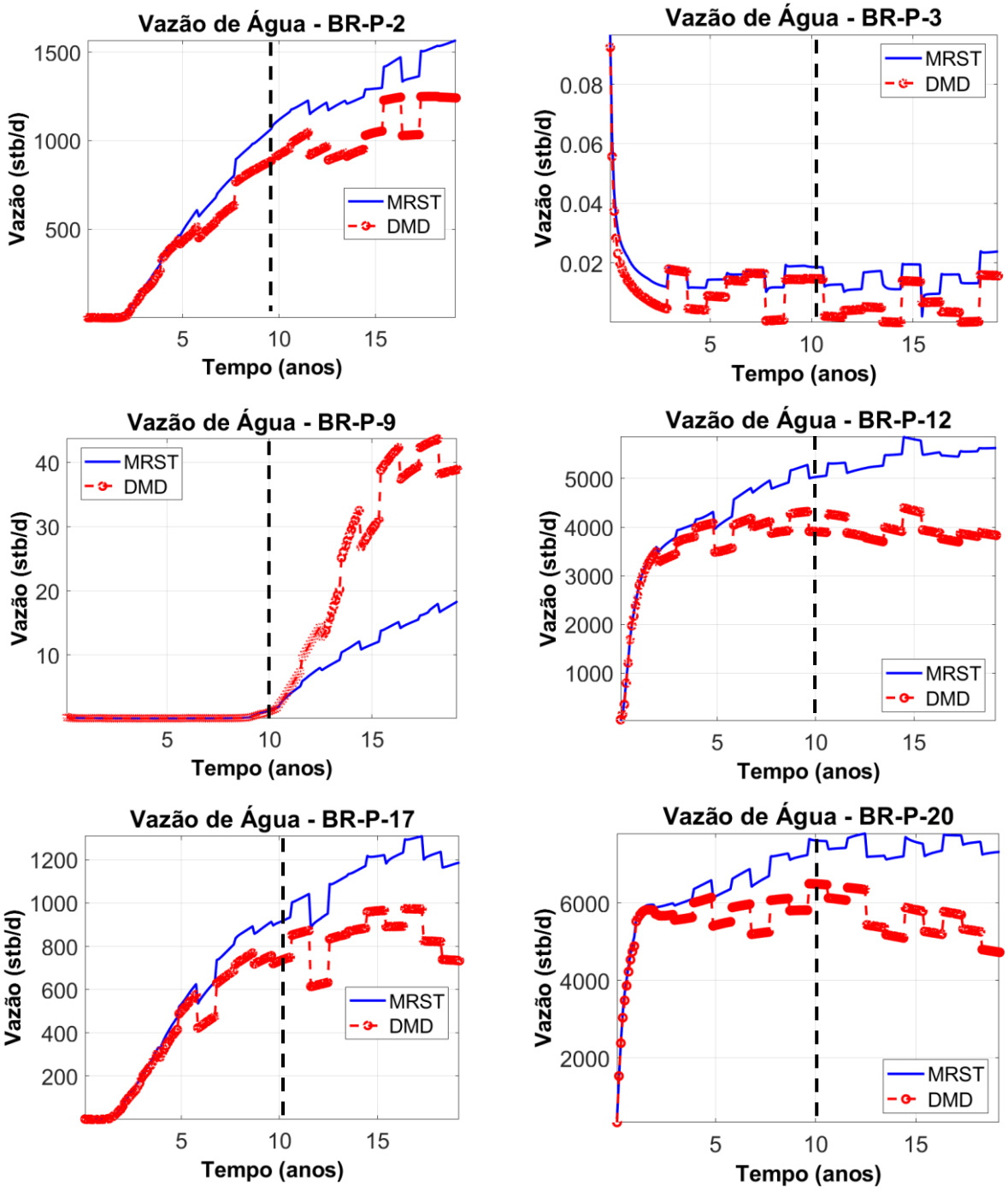
Figura 62 – Curvas de produção de óleo dos poços do modelo Brugge, para o Caso 2.



Fonte: O Autor (2020).

Na Figura 63, são apresentadas as curvas resultantes para a vazão de água de alguns dos poços produtores. Como se pode observar, na maioria dos poços, o modelo preditivo falha significativamente ao longo das curvas de produção.

Figura 63 – Curvas de produção de água dos poços do modelo Brugge, para o Caso 2.



Fonte: O Autor (2020).

Na Tabela 7, são apresentados os erros relativos das curvas apresentadas anteriormente, definidos ao longo das Equações (4.31) a (4.33), e o tempo computacional requerido para os dois tipos de simulação do Caso 2.

Tabela 7 – Erros e tempo computacional requerido, para o Caso 2 (Modelo Brugge).

Tipo de Simulação	**Erro**		**Tempo Decorrido**
MRST	-		7192,12s
DMDc	$E_{bhp,inj}$	1,80%	3,3960s
	$E_{o,prod}$	283,18%	
	$E_{w,prod}$	44,99%	

Fonte: O Autor (2020).

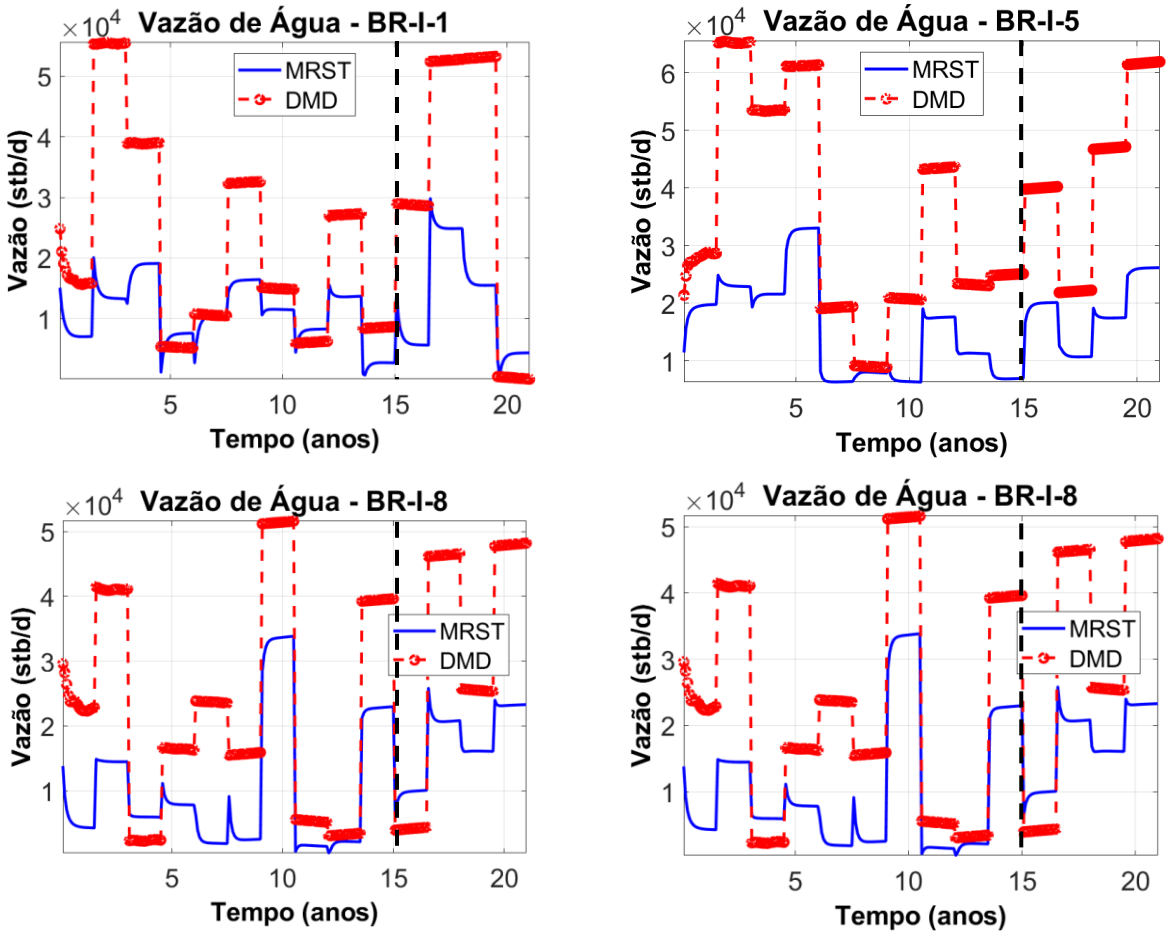
6.3.3 Modelo DMD: Caso 3

Para fins de comparação com o exemplo apresentado na seção 5.3, um modelo DMD foi construído a partir de 180 *snapshots* obtidos a partir de uma simulação de treinamento, com tempo total e passo de tempo iguais a 15 anos e 30 dias, respectivamente, a fim de serem previstos os 6 anos seguintes, totalizando os 21 anos de simulação do exemplo em questão.

Os controles dos poços durante o treinamento permaneceram constantes e iguais ao valor médio dos limites estabelecidos para cada poço, isto é, limites iguais a 2400 a 3000 psi, para os injetores, e 1000 a 2400 psi, para os produtores. Durante a nova simulação, tais controles foram distribuídos ao longo de 14 ciclos de controle, conforme já ilustrado na Figura 38.

Na Figura 64, são apresentadas as curvas resultantes para a vazão de água de alguns dos poços injetores, para fins de ilustração. A linha tracejada representa o instante a partir do qual o modelo realiza a previsão de estados futuros. Nota-se que, para todos os poços, o modelo preditivo falha consideravelmente ao longo do tempo de simulação.

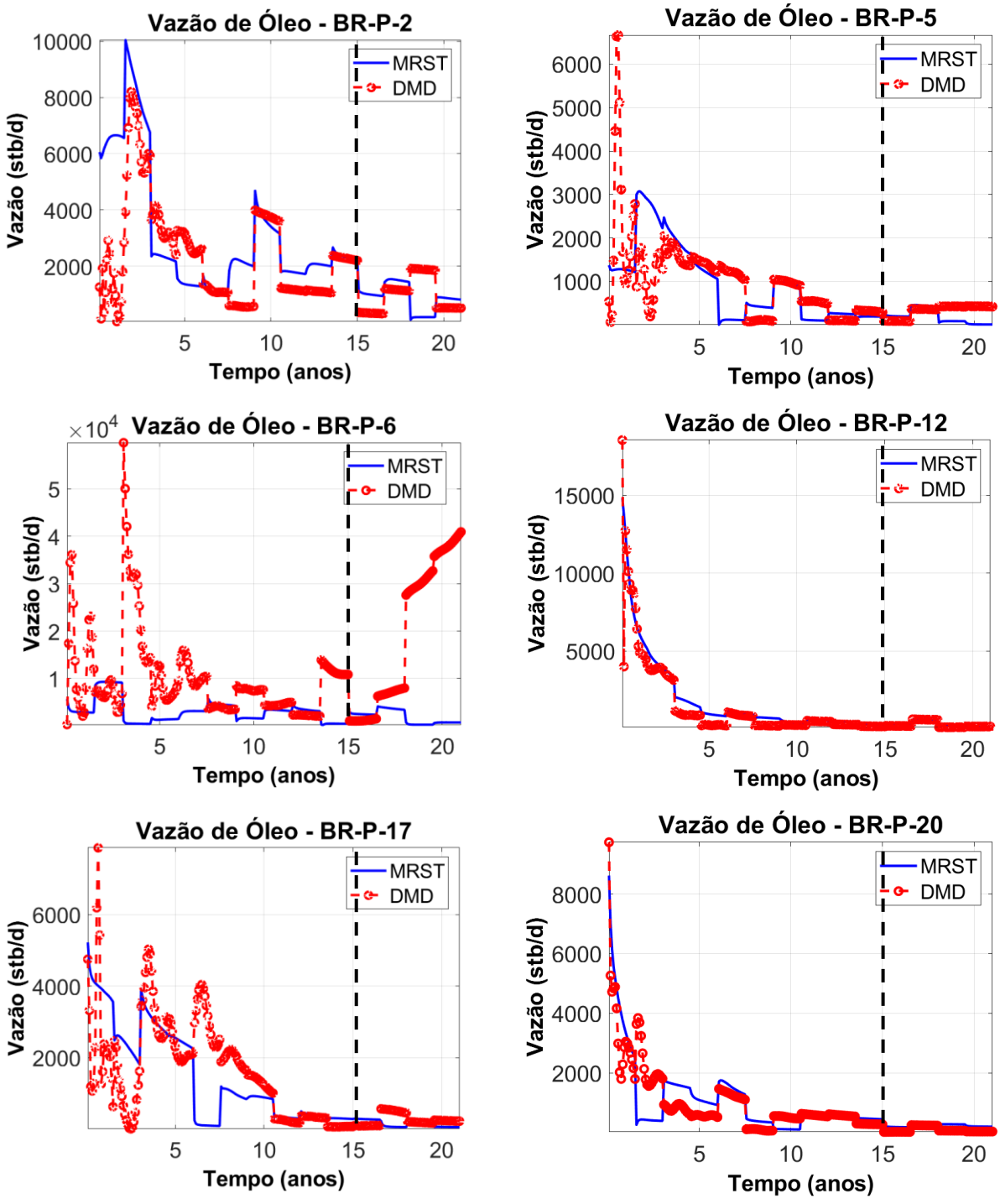
Figura 64 – Curvas de injeção de água dos poços do modelo Brugge, para o Caso 3.



Fonte: O Autor (2020).

Na Figura 65, são apresentadas as curvas resultantes para a vazão de óleo de alguns dos poços produtores, visto que todas as curvas resultantes apresentam o mesmo comportamento. Como se pode observar, para a maioria dos poços, o modelo preditivo falha significativamente ao longo do tempo de simulação.

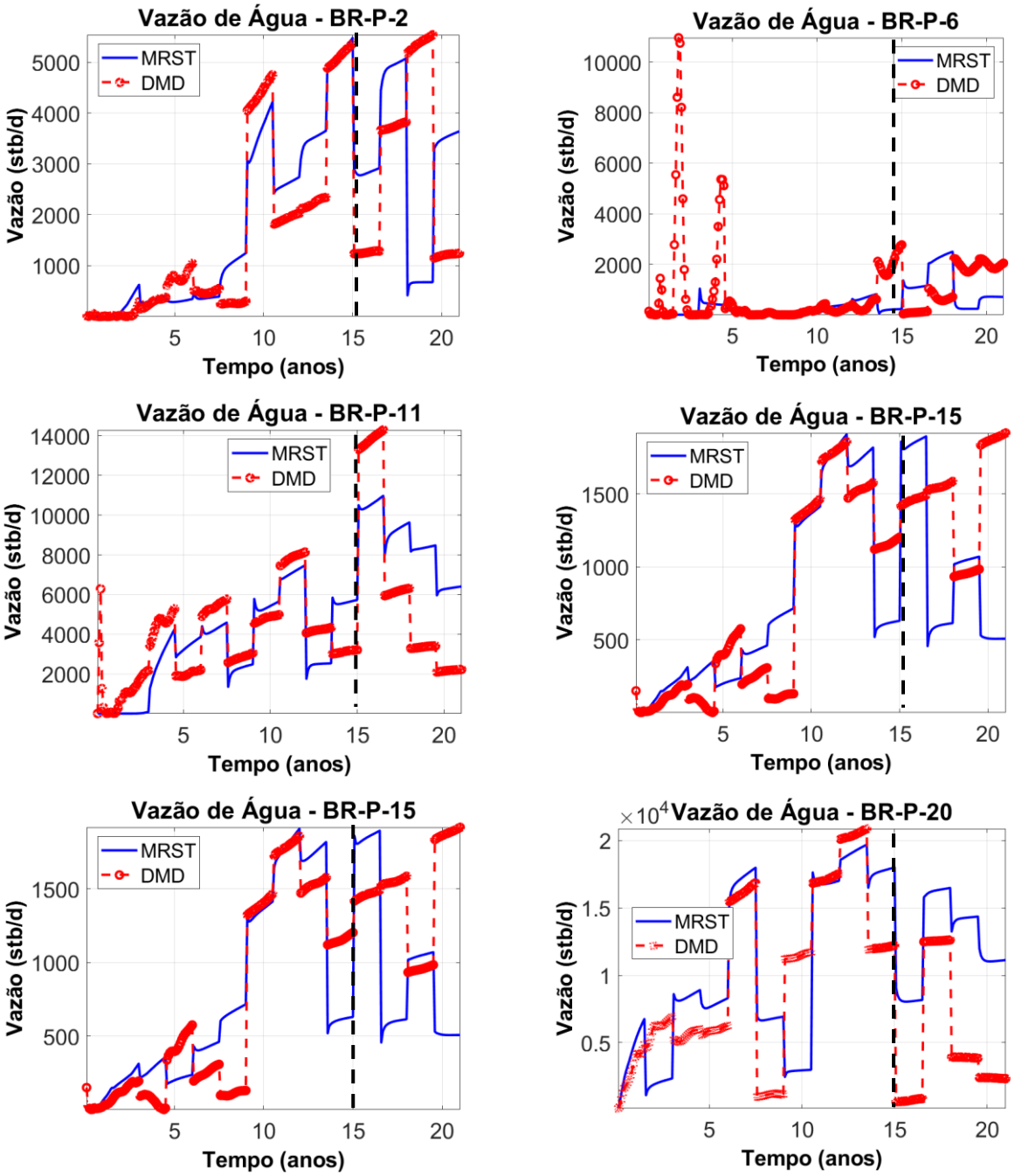
Figura 65 – Curvas de produção de óleo dos poços do modelo Brugge, para o Caso 3.



Fonte: O Autor (2020).

Na Figura 66, são apresentadas as curvas resultantes para a vazão de água de alguns dos poços produtores. Como se pode observar, na maioria dos poços, o modelo preditivo falha significativamente ao longo das curvas de produção.

Figura 66 – Curvas de produção de água dos poços do modelo Brugge, para o Caso 3.



Fonte: O Autor (2020).

Na Tabela 8, são apresentados os erros relativos das curvas apresentadas anteriormente, definidos ao longo das Equações (4.31) a (4.33), e o tempo computacional requerido para os dois tipos de simulação do Caso 3. Como se pode observar, semelhante ao Caso 2, a acurácia do modelo preditivo novamente fica comprometida no caso em que os controles variam durante o tempo de simulação, levando a concluir que a acurácia da simulação via DMDc fica comprometida à medida que se intensifica a variação dos controles aplicados aos poços.

Tabela 8 – Erros e tempo computacional requerido, para o Caso 3 (Modelo Brugge).

Tipo de Simulação	**Erro**		**Tempo Decorrido**
MRST	-		8315,45s
DMDc	$E_{w,inj}$	784,8%	4,1909s
	$E_{o,prod}$	1588,8%	
	$E_{w,prod}$	5407,9%	

Fonte: O Autor (2020).

7 CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho teve como objetivo a utilização de métodos de ordem reduzida a fim de mitigar o esforço computacional do processo de simulação de reservatórios e aplicá-lo no estudo de gerenciamento ótimo de reservatórios, tendo como variáveis de projeto os controles dos poços. As técnicas escolhidas para a construção do modelo substituto foram: o *Trajectory Piecewise Linearization* (TPWL), que lineariza as equações de fluxo em torno de estados previamente convergidos e armazenados durante uma ou mais simulações de treinamento, resultando numa redução da complexidade numérica do problema; e o *Proper Orthogonal Decomposition* (POD), responsável pela redução da dimensionalidade do problema através da projeção dos estados em um subespaço gerado a partir das direções principais (direções de maior variabilidade) de um conjunto de dados, baseando-se na existência de correlação entre os mapas de pressões e saturações previamente salvos. Fez-se ainda a análise de estabilidade quanto ao uso das projeções de Bubnov-Galerkin e de Petrov-Galerkin.

Para o armazenamento das matrizes de derivadas presentes na equação do TPWL, utilizou-se a ferramenta de Derivação Automática (AD) presente na plataforma MATLAB *Reservoir Simulation Toolbox* (MRST), *software* de código aberto desenvolvido pelo grupo SINTEF Digital, composto por uma caixa de ferramentas para prototipagem rápida de modelos e demonstração de novos métodos de simulação e conceitos de modelagem. Na Derivação Automática, são obtidas as quantidades e suas derivadas simultaneamente, o que permite que a equação de pressão não-linear, usando formulação totalmente implícita, calcule automaticamente a matriz Jacobiana exata para as equações de óleo e água, usando operadores discretos. Por ser de código aberto, a utilização do MRST permite o acesso direto à matriz Jacobiana e, portanto, modificações foram realizadas em algumas de suas rotinas a fim de exportar os mapas de estados e suas derivadas, de forma a permitir a construção do modelo TPWL/POD. Para o pleno entendimento do seu código-fonte e para a alteração de algumas de suas rotinas, foi necessário ainda o estudo do paradigma da Programação Orientada a Objetos (POO).

No Capítulo 2, foi abordado o modelo matemático para fluxo bifásico em meio poroso utilizado na simulação de reservatórios de petróleo e os principais aspectos do MRST, mais especificamente o módulo “ad-blackoil”, considerando o fluxo das fases água e óleo, considerando, por simplicidade, a razão de solubilidade de gás igual a zero. Foram deduzidas as equações diferenciais de fluxo para o modelo água-óleo na sua forma contínua e, em seguida, foi feita a discretização por diferenças finitas, utilizado para a solução numérica realizada internamente no simulador, incluindo o tratamento dado aos termos de poço pelo MRST.

Já no Capítulo 3, foi feita uma breve descrição do MRST e da técnica de Derivação Automática, que permite obter as quantidades e suas derivadas simultaneamente, e consequentemente a matriz Jacobiana exata para as equações de óleo e água. Foi preciso antes descrever o paradigma da Programação Orientada a Objetos, que divide a implementação em diferentes contextos numéricos, ocultar detalhes desnecessários e apenas expor os detalhes realmente necessários para cada contexto específico. Foi ainda proposto um teste de validação do MRST frente ao IMEX, um dos simuladores comerciais mais comumente utilizados na indústria, comparando os resultados obtidos por ambas as plataformas.

No Capítulo 4, foram introduzidos os aspectos teóricos acerca das estratégias utilizadas para a criação do primeiro modelo de ordem reduzida, isto é, o TPWL, responsável pela redução da complexidade numérica, e o POD, responsável pela redução da dimensionalidade do problema, incluindo o esquema *Local Resolution*, que mantém de forma integral as informações relativas às células que apresentam perfurações e completações de poços. São apresentados detalhes de como foram exportados os estados, conhecidos também por *snapshots*, e as matrizes de derivadas a partir de modificações realizadas em classes e funções que compõem a estrutura do MRST. Por fim, para o modelo SPE10, foram apresentados os resultados obtidos pelo modelo de ordem reduzida TPWL/POD utilizando os estados e as matrizes armazenadas durante uma única simulação de treinamento, tomando como controles as pressões de fundo dos poços (BHP), e comparando tais resultados aos obtidos na simulação de alta fidelidade.

A aplicação do modelo de ordem reduzida TPWL/POD em processos de otimização é feita no Capítulo 5, a partir da formulação de um problema onde a função objetivo a ser maximizada é a produção acumulada de óleo de um reservatório sintético controlado por BHP (modelo Brugge), sujeito a restrições de produção líquida e de água do campo, que se referem às limitações operacionais comuns nos campos de produção, comprometendo a eficiência do sistema separador óleo-água diante de grandes produções de líquido. Visto que o tratamento de restrições de estado durante o processo de otimização requer o cálculo das variáveis repetidas vezes, e também de suas derivadas, em múltiplos instantes de tempo, faz-se a utilização de modelos substitutos de ordem reduzida, cujos resultados são obtidos de modo aproximado, dentro de uma tolerância estabelecida, sendo executados em um tempo bem menor.

O algoritmo empregado durante o processo de otimização foi o SQP, o principal algoritmo utilizado em problemas de injeção de água, que envolve um grande número de variáveis e restrições. Fez-se então a análise quanto à acurácia e a estabilidade dos resultados obtidos, com o uso das projeções de Bubnov-Galerkin e de Petrov-Galerkin para o mesmo problema de otimização.

Nas simulações de teste realizadas para o modelo SPE10, foi demonstrado a capacidade do TPWL de capturar o comportamento do modelo de alta fidelidade para sequências de controles dentro e fora do intervalo de treinamento, para os quais foram obtidos s*peed-up's* da ordem de 2000, relacionando o tempo computacional requerido no modelo de alta fidelidade e no modelo TPWL/POD. Viu-se que a magnitude desses s*peed-up's* se deve ao fato de que os simuladores MRST baseados em Derivação Automática são entre duas e dez vezes mais lentos que simuladores comerciais. Ao contrário que a literatura reporta, neste trabalho não houve problemas relativos à estabilidade do método TPWL/POD, utilizando a projeção de Bubnov-Galerkin. Além disso, foi demonstrado que a acurácia do modelo de ordem reduzida está relacionada ao efeito da compressibilidade no sistema.

Resultados semelhantes foram obtidos para as simulações de teste do modelo Brugge, no qual obteve-se s*peed-up's* da ordem de 800. Para a maioria dos poços, as curvas apresentam boa convergência entre si, para ambos os tipos de projeção.

Contudo, anormalidades se mostraram presentes em alguns poços específicos, levando a magnitude do erro ser maior em relação aos erros obtidos no modelo SPE10.

Aplicou-se ainda o modelo TPWL/POD para um problema de otimização da produção de óleo do campo Brugge, sujeito a restrições de estado não-lineares, utilizando ambos os tipos de projeção. Para o caso onde a projeção de Bubnov-Galerkin foi utilizada, um menor número de iterações foi necessário para o fim do processo, não havendo problemas quanto à estabilidade do modelo. Em ambos os casos, foi possível observar a tendência de se fechar os poços injetores no primeiro ciclo de controle, visto que seus controles nesse instante tendem à pressão média inicial do reservatório, enquanto que as pressões de fundo dos poços produtores tendem ao limite máximo estabelecido, a fim de maximizar a produção de óleo do campo. Viu-se também que, ao final da otimização, as restrições relativas à produção de líquido não foram inteiramente satisfeitas, seja nos extremos, seja no interior de cada ciclo de controle, embora haja uma excessiva produção de líquido após a mudança do BHP, devido à descontinuidade dos controles aplicados em cada ciclo. Logo, mesmo que as restrições de estado nos extremos de cada ciclo de controle fossem satisfeitas, haveria a necessidade de serem impostos “pontos de correção”, isto é, restrições igualmente espaçadas no interior dos ciclos de controles, a fim de atender tais restrições no seu interior.

Servindo como uma alternativa à técnica TPWL/POD, foi apresentado no Capítulo 6 o método *Dynamic Mode Decomposition* (DMD). O DMD é uma técnica *data-driven*, isto é, puramente orientada a dados, onde estruturas dominantes de menor dimensionalidade são extraídas, podendo ser aplicada a uma variedade de disciplinas. O método adota uma perspectiva livre de equações, sendo capaz de fornecer uma decomposição precisa de um sistema complexo em estruturas espaço-temporais coerentes que podem ser utilizadas para a realização de previsões e controles futuros em curto prazo. Ao contrário do TPWL/POD, sua formulação é não-intrusiva visto que nenhuma modificação no código fonte do simulador foi necessária.

Os modelos orientado a dados, a exemplo do DMD, são uma coleção de ferramentas que englobam coleta, validação, agregação, processamento e análise de dados (*Data Analytics*) para extrair *insights* do passado, fazer previsões futuras e

recomendar decisões com base em cenários possíveis. Seu impulsionamento é dado pela disponibilidade de vastas e crescentes quantidades de dados, possibilitado por inovações notáveis em sensores de baixo custo, os avanços nas tecnologias de hardware, computação em nuvem, gerenciamento de dados e inovações em algoritmos de ciência de dados. Contudo, ainda há desafios importantes para a aplicação bem-sucedida da análise de dados na engenharia de reservatórios e sua adoção, como a escolha de métodos de modelagem, falta de habilidades e de equilíbrio adequado entre entendimento físico e métodos orientados a dados, entre outros.

A técnica DMD baseia-se no poder da Decomposição em Valores Singulares (*Singular Value Decomposition*, SVD), que fornece uma decomposição numericamente estável de uma matriz que pode ser usada para uma variedade de propósitos e é garantida a sua existência. A SVD serve como o algoritmo fundamental do *Proper Orthogonal Decomposition* (POD), também conhecido por *Principal Component Analysis* (PCA), onde um conjunto de dados de alta dimensionalidade é decomposto em seus fatores mais estatisticamente representativos e projetados em um subespaço de menor dimensão.

Foi apresentado ainda uma extensão do método, isto é, o *Dynamic Mode Decomposition with Control* (DMDc), que incorpora o efeito de atuação de dados de entrada, que no caso deste trabalho, são representados pelos controles dos poços. O DMDc, por sua vez, aborda todas as vantagens do DMD e fornece inovação adicional de ser capaz de isolar a dinâmica subjacente dos efeitos de atuação, resultando em modelos precisos de entrada e saída.

Para fins de demonstração da eficácia do método DMDc aplicado em engenharia de reservatórios, realizou-se uma série de casos e testes, utilizando o modelo Brugge, apresentado no Capítulo 5, cujos *snapshots* utilizados na construção do modelo linear foram gerados via simulação, tanto para a pressão quanto para a saturação de água, utilizando novamente o MRST.

Pode-se observar, pelas curvas de evolução e mapas apresentados, que a técnica DMD recupera de maneira satisfatória tanto os estados utilizados para a construção do modelo linear, quanto à previsão de estados futuros, para o caso em que os controles dos poços são constantes ao longo de todo o tempo de simulação.

Entretanto, a acurácia do modelo preditivo mostrou-se bastante comprometida ao submeter o reservatório a ciclos de controles variáveis durante a simulação, o que impossibilitou a sua aplicação em processos de otimização.

Para trabalhos futuros, pode-se incluir a aplicação do método DMD no estudo de fluxo multifásico em formações geológicas mais complexas, a exemplo de *karsts*, que apresentam como característica principal a presença de cavernas, cuja dinâmica é bastante complexa e desafiadora para os principais métodos de simulação atualmente existentes. A ideia é caracterizar sistemas de alta dimensionalidade compostos por *karsts*, através de padrões recorrentes de menor dimensão, obtidos puramente a partir de dados experimentais já existentes, e fazer o seu diagnóstico e previsões futuras.

Além disso, haja a vista que o método híbrido TPWL/POD, apesar dos resultados promissores, requer acesso ao código fonte do simulador (o que, na prática, não é possível na maioria dos simuladores comerciais, como o IMEX), percebe-se a importância do estudo de métodos orientados a dados, como o DMD, e sua aplicação em engenharia de reservatórios. Como apontado anteriormente, essa metodologia não foi validada a fim de permitir sua aplicação em processos de otimização. Trabalhos futuros sugerem que há duas questões principais a serem abordadas.

Primeiro, a dinâmica de um sistema bifásico óleo-água apresenta uma natureza não-linear, o que rompe a suposição de linearidade do DMD, cuja possível solução reside em encontrar espaços alternativos onde exista um mapeamento entre o sistema não-linear e o sistema linearizado, sem usar neste caso expansões por séries de Taylor. Isso pode ser alcançado utilizando, por exemplo, o operador Koopman (Rowley et al., 2009). O segundo problema está relacionado à deterioração do número de condicionamento das matrizes $\mathbf{X}$ e $\mathbf{X}'$, que é relativamente alto, impondo problemas durante o cálculo da matriz pseudoinversa, no qual pequenos erros são amplificados.

7.1 Sugestões de Trabalhos Futuros

Diante dos resultados obtidos e das observações expostas, segue abaixo algumas das sugestões para trabalhos futuros, relacionados à utilização de métodos de ordem reduzida no gerenciamento ótimo de reservatórios de petróleo.

- Inserir critério de retreinamento para o modelo de ordem reduzida baseado no erro obtido entre as simulações de alta fidelidade e TPWL/POD, conforme a Equação (4.34). Neste caso, uma nova simulação de treinamento seria requerida a fim de que novos estados e novas matrizes de derivadas sejam adicionados aos já existentes;
- Definir modelos de reservatórios que sejam controlados por vazão e, assim, aplicar as técnicas de modelo de ordem reduzida de maneira análoga ao que foi apresentado neste trabalho, para pressões de fundo dos poços (BHP);
- Estabelecer estratégias que garantam o respeito às restrições de estado dentro de cada ciclo de controle, durante o processo de otimização, utilizando o TPWL/POD como forma de reduzir o custo computacional envolvido;
- Caracterizar sistemas complexos compostos por *karsts*, através de padrões recorrentes de menor dimensão, obtidos puramente a partir de dados experimentais já existentes, utilizando a técnica DMD a fim de se realizar diagnósticos e previsões futuras;
- Aprimorar a aplicação do método DMD em problemas de engenharia de reservatórios, a partir da abordagem de espaços alternativos que realizem o mapeamento entre sistemas não-lineares e sistemas lineares, como o operador Koopman, preservando o condicionamento das matrizes envolvidas.

REFERÊNCIAS

Alim, M. **Constraint Handling In Life-Cycle Optimization Using Ensemble Gradients.** Delft University of Technology, 2013.

Bao, K., Lie, K.-A., Møyner, O., & Liu, M. (). **Fully Implicit Simulation Of Polymer Flooding with MRST**. Comput. Geosci*,* 2017*.*

Biegler, L. **Nonlinear Programming: Concepts, Algorithms, and Applications to Chemical Processes***.* Philadelphia: SIAM - Society for Industrial and Applied Mathematics, 2010.

Brunton, S., & Kutz, J. **Data-Driven Science and Engineering: Machine Learning, Dynamical Systems and Control***.* Cambridge: Cambridge University Press, 2019.

Cardoso, M. A. **Development and Application of Reduced-Order Modeling Procedures for Reservoir Simulation***.* PhD Thesis, 2009.

Carlberg, K., Bou-Mosleh, C., & Farhat., a. C. **Effcient Nonlinear Model Reduction Via A Leastsquares Petrov-Galerkin Projection And Compressive Tensor Approximations.** International Journal for Numerical Methods in Engineering, 2009.

Castro, A. D. **Implementação do Modelo de Ordem Reduzida TPWL para Simulação de Reservatórios de Petróleo na Plataforma MRST**. Dissertação de Mestrado. Recife - PE, Brasil, 2020.

Christie, M. A., & Blunt, a. M. **Tenth Spe Comparative Solution Project: A Comparison Of Upscaling Techniques.** SPE Reservoir Evaluation and Engineering, 2001.

SINTEF Digital. **MRST - MATLAB Reservoir Simulation Toolbox**. https://www.sintef.no/projectweb/mrst/ Acesso no dia 28/07/2020.

Dornelas, A., Souza Jr., A. d., & Horowitz, B. **Trajectory Piecewise Linearization (Tpwl) Using The Matlab Reservoir**. XL Ibero-Latin-American Congress on Computational Methods in Engineering*,* 2019.

Ertekin, T., Abou-Kassem, J. H., & King, G. R. **Basic Applied Reservoir Simulation.** Society of Petroleum Engineering - SPE*,* 2011.

Fragoso, M. **Aplicações De Um Modelo Substituto De Ordem Reduzida A Estudos De Gerenciamento De Reservatórios**. Dissertação de Mestrado. Recife - PE, Brasil, 2014.

Gildin, E., Ghasemi, M., Protasov, A., & Efendiev, Y. **Nonlinear Complexity Reduction**. In SPE Reservoir Simulation Symposium, 2013.

He, J. **Enhanced Linearized Reduced-Order Models For Subsurface Flow Simulation**. Master's thesis, Stanford University, 2010.

He, J., & Durlofsky, a. L. **Reduced-Order Modelling For Compositional Simulation Using Trajectory Piecewise Linearization**. In Reservoir Simulation Symposium, The Woodlands, Texas, USA, 2013.

Krogstad, S., Lie, K.–A., Møyner, M., Nilsen, H., Raynaud, X., & Skaflestad, B. **MRST-AD – an Open-Source Framework for Rapid Prototyping and Evaluation of Reservoir Simulation Problems**, 2015.

Kutz, J. **Data-Driven Modeling & Scientific Computation: Methods for Complex Systems and Big Data**. Oxford: Orford University Press, 2013.

Kutz, J. N., Brunton, S. L., Brunton, B. W., & Proctor, J. L. **Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems**. Philadelphia: Society for Industrial and Applied Mathematics, 2016.

Lie, K.-A. **An Introduction to Reservoir Simulation Using MATLAB**. Olso, Norway: SINTEF ICT, Departement of Applied Mathematics, 2016.

Lie, K.–A., Krogstad, S., Ligaarden, I., Natvig, J., Nilsen, H., & Skaflestad, B. **Open-Source MATLAB Implementation Of Consistent Discretisations On Complex Grids**. Comput Geosci, 2011.

Liu, Z., Forouzanfar, F., & Zhao, Y. **Comparison Of SQP And AL Algorithms For Deterministic Constrained Production Optimization Of Hydrocarbon Reservoirs.** *J. Pet. Sci,* 2018*. Eng. 171*, 542–557.

Peaceman, D. W. **Interpretation Of Well-Block Pressures In Numerical Reservoir Simulation With Nonsquare Grid Blocks And Anisotropic Permeability**. SPE Journal, 1983.

Pinto, J. W., Tueros, J. A., Horowitz, B., Silva, S. M., Willmersdorf, R. B., & Oliveira, D. F. **Gradient-Free Strategies To Robust Well Control Optimization.** Computational Geosciences*,* 2019.

Rowley, C. W., MeziC, I., Bagheru, S., & Schlatter, P. &. **Spectral Analysis of Nonlinear Flows**. Journal of Fluid Mechanics, Cambridge University Press (CUP), 2009.

Sankaran, S. **Data Analytics in Reservoir Engineering**. SPE Technical Report, 2019.

The MathWorks Inc. **MATLAB Object-Oriented Programming**, 2018*.*

Tueros, J. A. **Refinamento do Método Baseado em Essembles para Controle Ótimo do Problema de Injeção de Água***,* 2019*.*