



Universidade do Estado do Rio de Janeiro
Centro de Tecnologia e Ciências
Instituto Politécnico IPRJ

Leonardo Figueira Werneck

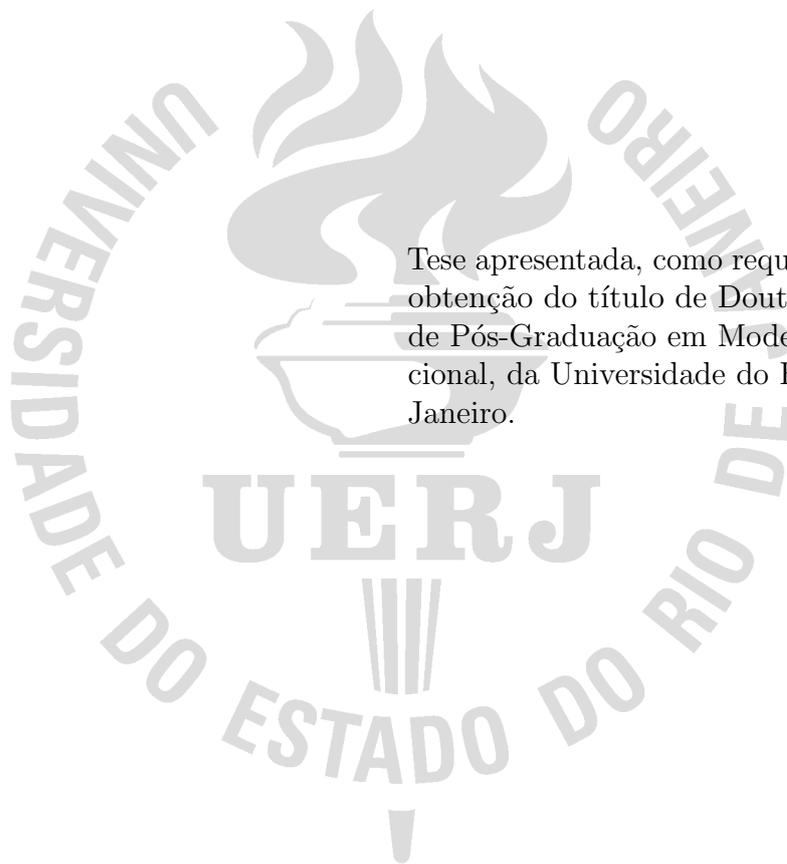
**Simulação numérica de escoamentos não-isotérmicos em
reservatórios de óleo empregando um coprocessador e a API
OpenMP**

Nova Friburgo

2021

Leonardo Figueira Werneck

**Simulação numérica de escoamentos não-isotérmicos em reservatórios de óleo
empregando um coprocessador e a API OpenMP**



Tese apresentada, como requisito parcial para obtenção do título de Doutor, ao Programa de Pós-Graduação em Modelagem Computacional, da Universidade do Estado do Rio de Janeiro.

Orientador: Prof. Dr. Helio Pedro Amaral Souto

Nova Friburgo

2021

CATALOGAÇÃO NA FONTE
UERJ/REDE SIRIUS/BIBLIOTECA CTC/E

W491 Werneck, Leonardo Figueira.
Simulação numérica de escoamentos não-isotérmicos em reservatórios de óleo empregando um coprocessador e a API OpenMP / Leonardo Figueira Werneck. – 2021.
92 f. : il.

Orientador: Helio Pedro Amaral Souto.
Tese (doutorado) - Universidade do Estado do Rio de Janeiro, Instituto Politécnico.

1. Materiais porosos – Escoamento – Métodos de simulação - Teses. 2. Petróleo – Escoamento – Métodos de simulação - Teses. 3. Engenharia de reservatório de óleo - Teses. 4. Métodos de decomposição de operadores - Teses. I. Souto, Helio Pedro Amaral. II. Universidade do Estado do Rio de Janeiro. Instituto Politécnico. III. Título.

CDU 532.546:519.872

Bibliotecária Cleide Sancho CRB7/5843

Autorizo, apenas para fins acadêmicos e científicos, a reprodução total ou parcial desta tese, desde que citada a fonte.


Assinatura

25/10/2021

Data

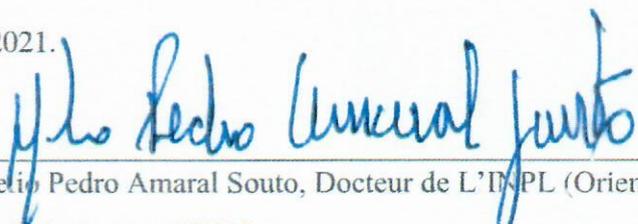
Leonardo Figueira Werneck

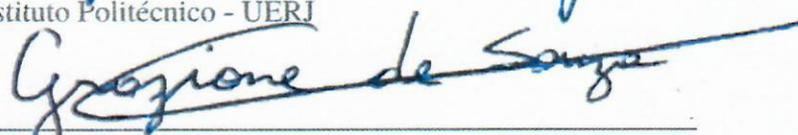
**Simulação Numérica de Escoamentos Não-isotérmicos em Reservatórios de Óleo
Empregando um Coprocessador e a API OpenMP**

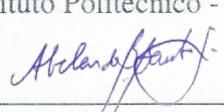
Tese apresentada como requisito parcial para
obtenção do título de Doutor ao Programa de Pós-
Graduação em Modelagem Computacional do
Instituto Politécnico, da Universidade do Estado
do Rio de Janeiro.

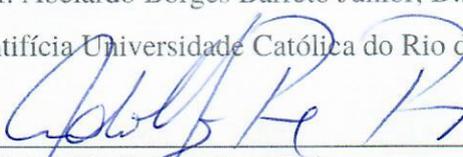
Aprovada em 26 de julho de 2021.

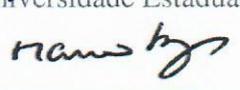
Banca examinadora:


Prof. Hélio Pedro Amaral Souto, Docteur de L'INPL (Orientador)
Instituto Politécnico - UERJ


Prof. Graziane de Souza Boy, D.Sc.
Instituto Politécnico - UERJ


Prof. Abelardo Borges Barreto Junior, D.Sc.
Pontifícia Universidade Católica do Rio de Janeiro


Prof. Adolfo Paine Pires, D.Sc.
Universidade Estadual do Norte Fluminense Darcy Ribeiro


Prof. Márcio Rentes Borges, D.Sc.
Laboratório Nacional de Computação Científica

Nova Friburgo

2021

DEDICATÓRIA

A todos que me ajudaram de alguma forma a chegar até aqui.

AGRADECIMENTOS

Primeiramente, agradeço a Deus por sua infinita fidelidade comigo, ao conceder tudo o que me foi necessário para a conclusão deste Doutorado em Modelagem Computacional. Agradeço imensamente aos meus pais, Itamar Lessa Werneck e Maria Lizete Siqueira Figueira, os quais nunca mediram esforços para me apoiar e me incentivar, além de sempre acreditarem nas minhas escolhas. Aos meus irmãos Ronaldo, Tânia, Jaqueline e Juliana e ao meu sobrinho, Gabriel, pelo apoio em todos os momentos. À minha amada esposa, Mariana Martins Martinez, que muito me apoiou nos momentos difíceis, dando-me força para continuar. Agradeço imensamente ao meu filho Bernardo, que mesmo na barriga da mãe me fazia tirar forças de onde parecia não ter mais, na reta final.

Muitos amigos contribuíram de forma direta ou indireta para que essa conquista fosse alcançada. Agradeço aos meus amigos que, sem dúvidas, contribuíram de maneira ímpar para que eu pudesse vencer todos os obstáculos ao longo desses anos. Agradeço também aos colegas de pós-graduação, que contribuíram com o meu aprendizado durante todo o curso. Um agradecimento especial a todos os integrantes da turma 63 do Curso de Formação de Oficiais do CBMERJ, os quais me apoiaram e entenderam a minha ausência em muitos momentos em que estava me dedicando para a conclusão desta tese de Doutorado.

Ao meu orientador, Dr. Helio Pedro Amaral Souto, por me conceder a oportunidade de fazer parte do LABTRAN e pela sua orientação, amizade, suporte e paciência. Ao professor, Dr. Grazione de Souza, por todo conhecimento compartilhado, incentivo e dedicação para que este trabalho pudesse ser concluído. Também deixo meus sinceros agradecimentos aos técnicos do laboratório, Hilton Guaraldi da Silva Jr. e Paulo de Tarço Honório Jr., os quais tiveram uma grande contribuição neste trabalho. Agradeço ao Instituto Politécnico (IPRJ) da Universidade do Estado do Rio de Janeiro (UERJ) pela infraestrutura oferecida, em especial pelas instalações do Laboratório de Modelagem Multiescala e Transporte de Partículas (LABTRAN). O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

RESUMO

WERNECK, L.F. *Simulação numérica de escoamentos não-isotérmicos em reservatórios de óleo empregando um coprocessador e a API OpenMP*. 2021. 92 f. Tese (Doutorado em Modelagem Computacional) – Instituto Politécnico, Universidade do Estado do Rio de Janeiro, Nova Friburgo, 2021.

A simulação numérica do escoamento em reservatórios de petróleo se tornou, ao longo de décadas, uma ferramenta padrão aplicada pela indústria de óleo e gás para prever o comportamento de um campo produtor de hidrocarbonetos. Tendo em vista a redução do custo computacional dessas simulações, as quais, em geral, demandam mais tempo conforme o caso estudado se torna mais realístico, técnicas de computação de alto desempenho também tem sido utilizadas na simulação de reservatórios. Neste contexto, esta tese tem como contribuição a proposta de uma estratégia para a simulação numérica do escoamento não-isotérmico em reservatórios de óleo utilizando uma decomposição de operadores, a API OpenMP, um coprocessador e um modelo a uma equação para a temperatura sem a necessidade de considerar equilíbrio térmico local. Ao longo do desenvolvimento desta tese, utilizou-se a modelagem do escoamento não-isotérmico em meios porosos, o método de diferenças finitas para a discretização e uma linearização das equações algébricas não-lineares pelo método de Picard. Estudou-se os efeitos da variação do número de *threads* e da malha computacional, tendo sido obtidos *speedups* superiores a 25 em alguns casos testados.

Palavras-chave: Simulação numérica de reservatórios. API OpenMP. Coprocessador.

Escoamento não-isotérmico. Decomposição de operadores.

ABSTRACT

WERNECK, L.F. *Numerical simulation of non-isothermal flows in oil reservoirs using a coprocessor and the OpenMP API*. 2021. 92 f. Tese (Doutorado em Modelagem Computacional) – Instituto Politécnico, Universidade do Estado do Rio de Janeiro, Nova Friburgo, 2021.

The numerical simulation of the flow in oil reservoirs has become, over decades, a standard tool applied by the oil and gas industry to forecast the behavior of a hydrocarbon producing field. In order to reduce the computational effort of these simulations, which in general demand more time as the case studied becomes more realistic, high performance computing techniques have also been applied in reservoir simulation. In this context, this thesis has as a contribution the proposal of a strategy for the numerical simulation of non-isothermal flow in oil reservoirs using an operator splitting, the OpenMP API, a coprocessor and an one-equation model for temperature without the need to consider local thermal equilibrium. Throughout the development of this thesis, non-isothermal flow modeling in porous media, the finite difference method for discretization and a linearization of the nonlinear algebraic equations by Picard's method were used. The effects of varying the number of threads and the computational grid were studied, and speedups greater than 25 were obtained in some cases tested.

Keywords: Numerical reservoir simulation. OpenMP API. Coprocessor. Non-isothermal flow. Operator splitting.

LISTA DE FIGURAS

Figura 1 - Linha do tempo da evolução do OpenMP	18
Figura 2 - Representação de um domínio bidimensional discretizado	34
Figura 3 - Malha unidimensional.	36
Figura 4 - Fluxograma para um passo de tempo	44
Figura 5 - Representação no formato CSR	46
Figura 6 - Modelo de programação OpenMP	50
Figura 7 - Fluxograma para um passo de tempo paralelizado	54
Figura 8 - Perfis de pressão no poço para diferentes malhas	60
Figura 9 - Perfis de pressão no poço - Malha 1 e versões serial e paralelizada . . .	61
Figura 10 - Perfis de pressão no poço - Malha 5 e versões serial e paralelizada . . .	61
Figura 11 - <i>Speedup</i> pelo número de <i>threads</i> para a Malha 4	65
Figura 12 - <i>Speedup</i> pelo número de <i>threads</i> para a Malha 5	66
Figura 13 - <i>Speedup</i> pelo número de <i>threads</i> para a Malha 6	66
Figura 14 - <i>Speedup</i> pelo número de <i>threads</i> para a Malha 7	67
Figura 15 - Análise do <i>speedup</i> para um número fixo de 80 <i>threads</i>	67
Figura 16 - Análise do <i>speedup</i> para um número fixo de 240 <i>threads</i>	68
Figura 17 - Variação da pressão: Códigos 1, 2 e 3	70
Figura 18 - Análise de sensibilidade para a permeabilidade	71
Figura 19 - Análise de sensibilidade para a viscosidade	73
Figura 20 - Análise de sensibilidade para a porosidade	74
Figura 21 - Análise de sensibilidade para a taxa de aquecimento	76
Figura 22 - Análise de sensibilidade para 0, 2 e 4 aquecedores	77
Figura 23 - Variação da pressão em função da vazão	80
Figura 24 - Temperatura na região próxima ao poço produtor - Malha 5	81
Figura 25 - Pressão na região próxima ao poço produtor - Malha 5	81
Figura 26 - Temperatura na região próxima ao poço produtor - Maior difusividade hidráulica	82
Figura 27 - Pressão na região próxima ao poço produtor - Maior difusividade hidráulica	82
Figura 28 - Temperatura na região próxima ao poço produtor - Quatro aquecedores	83
Figura 29 - Pressão na região próxima ao poço produtor - Quatro aquecedores . . .	84

LISTA DE TABELAS

Tabela 1 - Parâmetros para o caso padrão	58
Tabela 2 - Refinamento de malha	59
Tabela 3 - Tempo de execução e <i>speedup</i> para as Malhas 0 e 1	62
Tabela 4 - Tempo de execução e <i>speedup</i> para as Malhas 2 e 3	63
Tabela 5 - Tempo de execução e <i>speedup</i> para as Malhas 4 e 5	63
Tabela 6 - Tempo de execução e <i>speedup</i> para as Malhas 6 e 7	64
Tabela 7 - Tempo de execução e <i>speedup</i> para a Malha 5: diferentes códigos	69
Tabela 8 - Tempo de execução e <i>speedup</i> para a Malha 6: diferentes códigos	69
Tabela 9 - Análise de sensibilidade para a permeabilidade	71
Tabela 10 - Análise de sensibilidade para a viscosidade	72
Tabela 11 - Análise de sensibilidade para a porosidade	74
Tabela 12 - Análise de sensibilidade para a taxa de aquecimento	75
Tabela 13 - Análise de sensibilidade para o número de aquecedores	77
Tabela 14 - Análise de sensibilidade para o Conjunto Otimizado	78
Tabela 15 - Análise de sensibilidade para a vazão	79

SUMÁRIO

	INTRODUÇÃO	11
1	ESCOAMENTO NÃO-ISOTÉRMICO	21
1.1	Propriedades do Fluido e da Rocha	22
1.1.1	<u>Porosidade</u>	22
1.1.2	<u>Permeabilidade</u>	23
1.1.3	<u>Massa específica</u>	23
1.1.4	<u>Viscosidade</u>	24
1.1.5	<u>Fator-Volume-Formação</u>	24
1.1.6	<u>Coefficiente de expansão térmica</u>	25
1.1.7	<u>Compressibilidade</u>	25
1.1.8	<u>Aspectos da transferência de calor</u>	26
1.2	Hipóteses assumidas para o escoamento	27
1.3	Equação governante para a pressão	28
1.3.1	<u>Acoplamento poço-reservatório</u>	31
1.4	Equação governante para a temperatura	31
2	METODOLOGIA NUMÉRICO-COMPUTACIONAL	34
2.1	Discretização da EDP para a pressão	34
2.2	Discretização da EDP para a temperatura	37
2.3	Acoplamento poço-reservatório	38
2.4	Decomposição de operadores e linearização	39
2.5	Métodos de resolução de sistemas lineares	42
3	PARALELIZAÇÃO	45
3.1	Técnicas de compressão de matrizes	45
3.2	Processamento em paralelo com o OpenMP	47
3.2.1	<u>A interface OpenMP</u>	48
3.2.2	<u>Histórico do OpenMP</u>	49
3.2.3	<u>Modelo de execução do OpenMP</u>	50
3.2.4	<u>Coprocessador Intel Xeon Phi</u>	55
4	RESULTADOS	57
4.1	Características gerais da simulação	57
4.2	Refinamento de malha	59
4.3	Eficiência computacional	61
4.4	Análise de sensibilidade	70
4.4.1	<u>Permeabilidade</u>	70
4.4.2	<u>Viscosidade</u>	72
4.4.3	<u>Porosidade</u>	73

4.4.4	<u>Taxa de aquecimento</u>	75
4.4.5	<u>Número de aquecedores</u>	76
4.4.6	<u>Vazão</u>	78
4.5	Campos de pressão e temperatura do reservatório	80
	CONCLUSÕES E PERSPECTIVAS	85
	REFERÊNCIAS	88

INTRODUÇÃO

Simulações numéricas do escoamento multifásico não-isotérmico, na presença de poços, em reservatórios de petróleo, devem levar em consideração a necessidade de resolução das equações governantes, que são equações diferenciais parciais não-lineares. O uso dos métodos de malha tradicionais substitui a busca pela solução dessas equações pela de sistemas de equações algébricas não-lineares oriundas da discretização. Em tais simulações típicas, os reservatórios podem possuir quilômetros nas direções x e y e dezenas de metros na direção z e são, em geral, heterogêneos e o tempo de produção pode ultrapassar décadas. Portanto, a redução do custo computacional é uma demanda importante nesse contexto, visto que simulações realizadas em menos tempo permitem que engenheiros e demais profissionais, envolvidos no gerenciamento de reservatórios, possam rapidamente tomar decisões sobre a melhor estratégia de recuperação de hidrocarbonetos.

Simulação numérica de reservatórios

Os problemas envolvendo o escoamento de fluidos em meios porosos são importantes para diversas áreas do conhecimento científico e tecnológico. Como exemplos, pode-se citar a indústria química, o meio ambiente, a geologia, a engenharia mecânica, a medicina, a indústria de petróleo e gás etc. (SCHULZ, 2009). Desde a década de 1960, devido à pertinência para a indústria petrolífera, a simulação numérica de reservatórios de hidrocarbonetos tem recebido uma grande atenção. Ela demanda conhecimentos da Física, da Matemática Aplicada, da Engenharia de Reservatórios e da Engenharia de *Software*. O seu objetivo principal é o desenvolvimento de ferramentas computacionais que devem ser utilizadas na predição do escoamento em um reservatório, no escopo de diferentes planos de desenvolvimento (ERTEKIN; ABOU-KASSEM; KING, 2001; ISLAM et al., 2010).

Do ponto de vista econômico, é importante estudar o escoamento durante a fase da exploração, fornecendo dados que venham a auxiliar na avaliação das características de produção, tais como os padrões de escoamento, a pressão no poço, o fator de recuperação e o tempo de produção. Reforçando, a simulação de reservatórios é atualmente amplamente utilizada e se tornou a principal ferramenta da Engenharia de Reservatórios, sendo empregada quando da tomada de decisões envolvendo grandes recursos financeiros, na estimativa das reservas contidas no subsolo e para avaliar a eficiência da recuperação e a viabilidade econômica de novas perfurações de poços de petróleo (DUMKWU; AKAND; CARLSON, 2012; REDONDO; RUBIO; VALERO, 2018).

A modelagem físico-matemática do escoamento de fluidos em meios porosos consiste na utilização de um conjunto de equações diferenciais parciais (EDPs), que descrevem

o escoamento dos fluidos no interior dos reservatórios de petróleo, no fornecimento das condições de contorno e inicial apropriadas e das equações constitutivas pertinentes (CHEN; HUAN; MA, 2006). Quanto mais as propriedades físicas e do reservatório forem dependentes das variáveis de interesse (por exemplo, pressão, saturação e temperatura), maior será a complexidade das equações que governam o escoamento podendo originar, assim, não-linearidades no sistema de equações. Por conseguinte, a menos de casos simplificados, não é comum poder se obter soluções analíticas para tais equações. As simplificações físico-matemáticas implicam na obtenção de soluções que diferem significativamente das reais. Para que casos mais realísticos sejam tratados, soluções aproximadas e acuradas devem ser obtidas mediante o uso de métodos numéricos voltados para a obtenção das soluções das equações diferenciais parciais governantes (AZIZ; SETTARI, 1990; ABOU-KASSEM; ALI; ISLAM, 2006).

Recuperação de hidrocarbonetos

Os reservatórios, nos quais os mecanismos de recuperação são pouco eficientes e que por consequência retêm grandes quantidades de hidrocarbonetos após a exaustão da sua energia natural, são fortes candidatos ao emprego de uma série de processos que visam a obtenção de uma recuperação adicional. Os processos, ou os chamados Métodos de Recuperação, são, de maneira geral, os mecanismos que levam ao aumento da eficiência da produção dos fluidos contidos no reservatório. Quase tão antigos quanto a indústria do petróleo, os métodos de recuperação podem ser naturais ou os que foram desenvolvidos para se obter uma maior produção do que aquela que se obteria caso, apenas, a diferença de pressão entre o reservatório e o exterior fosse utilizada. Existem três tipos de recuperação de hidrocarbonetos: primária, secundária e terciária (DAKE, 2001).

A natureza dos fluidos (DANDEKAR, 2013) que preenchem os espaços porosos dos reservatórios de petróleo depende fortemente do estágio do processo de recuperação. No início da recuperação, o reservatório, geralmente, contém essencialmente uma ou mais fases, gás e/ou óleo, com a presença de água conata (EZEKWE, 2010). Neste momento, a produção de óleo e/ou gás é feita pela descompressão natural dos fluidos sem a necessidade de uma suplementação de energia por uma fonte externa. Esse estágio, também conhecido como recuperação primária, termina quando a pressão do reservatório é reduzida, de forma a não mais ser possível conduzir o fluido ao poço produtor. A recuperação primária geralmente deixa de 70 a 85% dos hidrocarbonetos no reservatório (CHEN; HUAN; MA, 2006).

Uma das maneiras de se recuperar parte do óleo, que ainda se encontra no reservatório, é através da injeção de um fluido imiscível (água, por exemplo) através de poços de injeção enquanto os hidrocarbonetos são produzidos nos poços de produção.

Esse procedimento serve para preservar o nível de pressão e viabilizar o escoamento no reservatório. Ele é chamado de recuperação secundária e, caso a pressão do meio estiver acima da pressão do ponto de bolha da fase óleo, no caso imiscível existirão duas fases: a água e o óleo. Porém, se ela se encontrar abaixo da pressão do ponto de bolha, uma terceira fase, gasosa, também aparecerá e estará em equilíbrio termodinâmico com a fase óleo. Nesse caso, muitas vezes considera-se que não há transferência de massa entre a água e as fases óleo e gás, porém, ocorrerá a troca entre as fases óleo e gás. O processo de recuperação secundária pode não ser muito efetivo e, findo esse estágio, 50% ou mais dos hidrocarbonetos ainda permanecem no reservatório (CHEN; HUAN; MA, 2006).

O método de recuperação secundária deixa de ser economicamente viável quando a taxa de produção de água ultrapassa certos valores limites e, então, mesmo que exista ainda significativa quantidade de óleo no reservatório, é hora de se interromper a sua utilização (FRIAS, 1999). A situação é agravada em reservatórios de óleo pesado e viscoso, onde a água é extremamente móvel (CHEN; HUAN; MA, 2006). Consequentemente, a água não consegue impelir de maneira eficaz o óleo residente, devido à formação de canais ou caminhos preferenciais de escoamento (*viscous fingers*), indesejáveis, através dos quais o fluido injetado escoar mais facilmente sem efetuar, de forma satisfatória, a função prevista de varredura dos fluidos residentes (SOUZA, 2008). Para recuperar parte desses hidrocarbonetos, vários outros métodos podem ser aplicados, tais como a injeção de fluidos miscíveis, de fluidos aquecidos, de polímeros e, até mesmo, de agentes biológicos (ROSA; CARVALHO; XAVIER, 2006; EZEKWE, 2010). A aplicação de um desses métodos é conhecida como a recuperação terciária ou métodos de recuperação avançada (*Enhanced Oil Recovery* (EOR)) (LAKE, 1989).

Atualmente, na produção de hidrocarbonetos é utilizada uma prática, conhecida como de manutenção de pressão, que consiste na injeção de água ou gás ainda no início da vida produtiva do reservatório. Ela tem por objetivo manter a pressão em níveis elevados, preservando as características dos fluidos e do escoamento, não sendo necessário esperar o declínio da produção de óleo para iniciar a injeção de fluidos (THOMAS, 2001). Embora se tenha na literatura a distinção entre os métodos de recuperação vale ressaltar que, com o passar do tempo, as denominações secundária e terciária perderam o seu sentido cronológico e passaram a indicar o tipo de processo executado. Assim sendo, a recuperação secundária passou a compreender a injeção de água ou de gás no caso imiscível, enquanto que a recuperação terciária é constituída pelos demais processos (THOMAS, 2001).

Métodos térmicos de recuperação

Ao se injetar água em um reservatório de petróleo, esse fluido desloca um determinado volume de óleo no interior do reservatório. Já no método de recuperação térmica,

por exemplo, mediante os efeitos combinados da transferência de calor por condução e advecção, a injeção de um fluido aquecido pode aumentar a temperatura de grandes regiões do reservatório, causando a redução da viscosidade e a expansão térmica do óleo, além de modificar a molhabilidade da rocha reservatório, melhorando a fluidez e facilitando o escoamento do óleo através do meio poroso (KOVSCZEK, 2012; AFSAR; AKIN, 2016).

Os métodos térmicos são apropriados para serem aplicados em reservatórios contendo óleo variando entre 8° e 22° API e viscosidade entre 0,1 e 10 Pa·s, ou para reservatórios com a permeabilidade muito baixa (AFSAR; AKIN, 2016). Para uma permeabilidade absoluta na faixa de 0,01 mD, a taxa de transferência de calor por advecção pode ser até 10 vezes maior do que a taxa de transferência por condução (KOVSCZEK, 2012). Apesar disso, devido ao fenômeno da condução de calor, os métodos térmicos de recuperação têm a capacidade de atingir poros não conectados aos poços injetores.

Os métodos térmicos de recuperação podem ser divididos em convencionais e não-convencionais. De acordo com Bera e Babadagli (2015), os métodos convencionais preveem:

1. A injeção de água aquecida (*hot water flooding*);
2. A injeção de vapor (*steam injection*);
3. A combustão *in-situ*.

Os três métodos mais comuns de injeção de vapor são: a injeção cíclica, a injeção contínua (*steam flooding*) e a técnica SAGD (*Steam Assisted Gravity Drainage*).

A injeção cíclica de vapor ocorre em três etapas: 1) a injeção do vapor, em um dado intervalo de tempo, através de um poço de produção; 2) o período de imersão; e 3) a posterior produção de óleo. Já no processo de injeção contínua, o vapor é injetado através de poços de injeção, aquecendo e deslocando o óleo em direção ao poço produtor, de modo semelhante ao que ocorre na injeção de água (*water flooding*). No processo de injeção contínua é necessária uma maior quantidade de vapor do que no processo de injeção cíclica e ele é, normalmente, mais eficaz na recuperação de hidrocarbonetos. No caso do SAGD, vapor a alta pressão é continuamente injetado em um poço horizontal para aquecer o óleo e reduzir a sua viscosidade. O óleo aquecido é, então, drenado para uma localização inferior devido aos efeitos gravitacionais, por onde ele é produzido (BERA; BABADAGLI, 2015).

O método de combustão *in-situ* é o mais utilizado na recuperação de óleo pesado e betume. Nesse método, injeta-se ar no reservatório e, via um dispositivo de ignição, inflama-se parte do óleo presente no reservatório ocasionando, conseqüentemente, uma transferência de calor. Conforme o ar é injetado, a frente de calor avança e os gases gerados na combustão fazem com que ocorra um aumento na recuperação de óleo. Este processo também é chamado de *fire flooding* (BERA; BABADAGLI, 2015).

A aplicação de métodos térmicos na recuperação avançada de petróleo corresponde a, aproximadamente, 50% das atividades do tipo no mercado (AFSAR; AKIN, 2016). O método convencional de injeção de vapor enfrenta desafios devido aos altos custos de instalação e manutenção. Por outro lado, na aplicação do método da combustão *in-situ* produz-se gases que são liberados na atmosfera. Esses e outros fatores levaram ao desenvolvimento de alternativas que reduzissem o impacto ambiental de algumas dessas técnicas, dando origem ao surgimento de uma nova classe de métodos térmicos, denominados de não-convencionais.

Como exemplo de um método não-convencional é possível citar a recuperação utilizando a energia solar, onde o vapor injetado é produzido gerando uma menor contaminação do ar atmosférico. O desafio encontrado na aplicação prática dessa técnica é devido à localização em que se encontram os campos de petróleo pois, ao contrário do que se encontra em projetos convencionais envolvendo a energia solar onde os painéis solares podem ser instalados em áreas abertas, no método EOR solar eles podem ter que ser instalados em ambientes agressivos e com condições de radiação solar não ideais (AFSAR; AKIN, 2016).

Em contrapartida, existe a possibilidade do uso do aquecimento eletromagnético que, dependendo da frequência da corrente elétrica, pode ser classificado em três categorias:

1. Baixa corrente, aquecimento resistivo;
2. Média frequência, aquecimento indutivo;
3. Alta frequência, radiofrequência ou micro-ondas.

No primeiro, também conhecido como o aquecimento por efeito Joule, uma diferença de potencial é aplicada entre dois eletrodos (na prática, pode-se utilizar dois poços no reservatório, onde um atua como ânodo e o outro como cátodo), fazendo com que uma corrente elétrica passe através da água de formação devido à sua condutividade elétrica.

O aquecimento indutivo é uma técnica na qual materiais eletricamente condutores são colocados na presença de um campo magnético variável gerado por indução. A corrente elétrica produzida nesse processo é também chamada de corrente de Eddy. Devido ao efeito de Joule, a corrente elétrica dissipa calor no material. Os fatores que regem o processo de aquecimento são a frequência da corrente induzida e o calor específico, a permeabilidade e a resistência elétrica do material considerado (BERA; BABADAGLI, 2015).

O uso de radiofrequência, ou de micro-ondas, pode ser uma boa alternativa para o aquecimento *in situ* de reservas não-convencionais. Nele, um tipo de antena é colocada no fundo de um poço, reduzindo a influência da geologia da formação, e ele é capaz de distribuir o calor sobre um maior volume do reservatório mediante a propagação de energia eletromagnética. Outras vantagens desse método, no processo de geração de energia, são a utilização de equipamentos compactos, de alta eficiência, além da possibilidade de

concentrar a energia em locais específicos contendo os hidrocarbonetos (DAVLETBAEV; KOVALEVA; BABADAGLI, 2011; BIENTINESI et al., 2013).

OpenMP e coprocessadores

Diversos problemas modelados na área de simulação de reservatórios de hidrocarbonetos requerem, para a sua solução, a resolução de sistemas lineares contendo um grande número de equações e cujas matrizes dos coeficientes associadas são esparsas. Em geral, estes sistemas são gerados a partir da discretização das equações diferenciais parciais não-lineares governantes. Os métodos iterativos, tipicamente utilizados para a resolução destes sistemas implicam, geralmente, um alto custo computacional: uso de uma quantidade elevada de recursos em termos de espaço em memória e tempo de processamento. Por essas razões, existem simuladores que fazem uso da computação paralela distribuída em núcleos de processadores e em múltiplas máquinas, usando o padrão MPI (*Message Passing Interface*), ou a paralelização empregando múltiplos núcleos (*threads*) em uma única máquina, empregando o padrão OpenMP (*Open Multi-Processing*) (de la Cruz; MONSIVAIS, 2014).

Desde que os computadores começaram a ser amplamente comercializados, as empresas de tecnologia de todo o mundo têm voltado seus esforços para melhor atender aos consumidores, tendo sido impulsionadas a introduzir melhorias em seus produtos. A alta competitividade levou as grandes fabricantes a investirem em tecnologias de *software* e *hardware* de alto desempenho. Portanto, não poderia ser diferente com relação ao desenvolvimento dos processadores e das placas gráficas.

Cada vez mais, as desenvolvedoras têm buscado entregar novos componentes de processamento extremamente velozes (CARVALHO et al., 2020). No entanto, para se obter um desempenho computacional significativo, não basta apenas possuir um *hardware* potente. Na verdade, o uso de recursos físicos necessita do emprego de *softwares* otimizados para que seja possível o desenvolvimento da computação de alto desempenho, ou seja, que tem por finalidade reduzir o tempo empregado na execução dos códigos. Na área de simulação numérica de fenômenos físicos, em problemas de engenharia, tais como na simulação numérica de reservatórios, à medida que se estuda problemas cada vez mais complexos, a resolução numérica requer o uso intenso de recursos de memória e/ou processamento demandando um grande esforço computacional (GASPARINI et al., 2021).

Neste contexto, encontra-se a motivação para a implementação de técnicas de computação de alto desempenho. O objetivo desta aplicação é a de possibilitar a simulação de modelos matemáticos que requerem a resolução de sistemas de equações com um grande número de incógnitas ou, até mesmo, tornar viável as simulações numéricas de problemas físicos que, muitas vezes, não seriam possíveis de serem realizadas utilizando outras

metodologias (WERNECK, 2016). Para tanto, utiliza-se, por exemplo, o processamento em paralelo (uma das técnicas da computação científica de alto desempenho), cuja principal meta é dividir uma determinada tarefa em várias menores e executar cada uma delas em diferentes núcleos de processamento ou computadores.

Como já mencionado, um exemplo de técnica de computação de alto desempenho é o MPI. Neste caso, diferentes processadores/computadores são utilizados na realização de uma dada tarefa empregando o acesso à memória distribuída (CARPEN-AMARIE; HUNOLD; TRÄFF, 2017). Uma alternativa ao MPI seria o processamento usando GPUs (*Graphics Processing Unit*) em conjunto com a API (*Application Programming Interface*) CUDA ou OpenACC. Também é possível o uso da API OpenMP, a qual pode ser utilizada em computadores com arquiteturas de memória compartilhada, sendo as *threads* (linhas de execução) empregadas na execução de tarefas em paralelo (DIETRICH et al., 2017). Comparada com a arquitetura de memória distribuída, a arquitetura de memória compartilhada exige muito menos recursos e, ao mesmo tempo, evita muitos problemas associados ao processamento de mensagens entre os processadores (WU; YE, 2019).

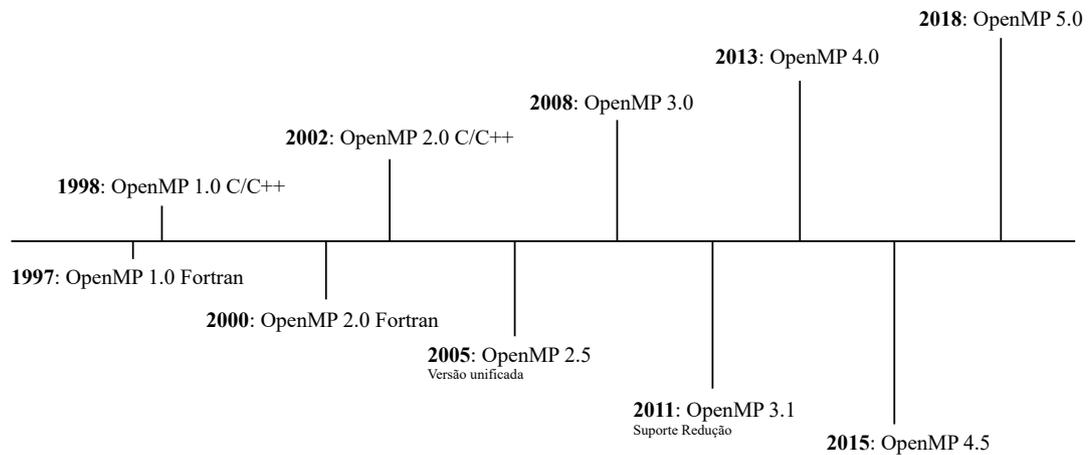
Sistemas híbridos também podem ser pensados, os quais utilizam em conjunto o acesso às memórias distribuída e compartilhada como, por exemplo, códigos escritos com comandos do MPI e do OpenMP (YAKUBOV et al., 2013; CALLE, 2017). É importante salientar, também, que os ganhos atingidos com essas tecnologias são limitados. Ainda não se conhece um tipo de técnica de computação de alto desempenho que obtenha um retorno linear de economia de tempo, em função do aumento do número de *threads* e/ou de processadores, para todos os problemas (HALSEY, 2016).

OpenMP é uma API que permite a paralelização de processos através do uso de memória compartilhada (CHAPMAN; JOST; van der PAS, 2008), de maneira mais simples quando comparada, por exemplo, a uma implementação utilizando a biblioteca CUDA ou mesmo a API MPI, já que é uma aplicação que visa a organizar tarefas de baixo nível através de instruções de alto nível. Por exemplo, é possível definir as regiões a serem paralelizadas, o número de *threads* a serem utilizadas, as variáveis e as regiões de memória a serem compartilhadas e, como o processador e suas *threads* lidarão com um trecho específico do código numérico, a paralelização com apenas uma linha de código contendo algumas diretrizes.

Uma *thread* representa um conjunto de instruções de um código que é executado em um determinado tempo de maneira independente. Essencialmente, o sistema operacional aloca os recursos necessários para a execução do processo em questão e, se múltiplas *threads* participam desta execução, elas compartilham, portanto, das mesmas regiões de memória e recursos. A biblioteca do OpenMP (`omp.h`) é a responsável por essa organização, de maneira a se evitar as complexidades inerentes às implementações e aos erros (uma implementação de mais baixo nível poderia acarretar em falhas e em uma complexidade crescente, de acordo com a evolução do nível de dificuldade do problema a ser tratado

computacionalmente) (CHAPMAN; JOST; van der PAS, 2008). A Figura 1 apresenta a linha do tempo com o histórico da evolução da API OpenMP.

Figura 1 - Linha do tempo da evolução do OpenMP



Fonte: O autor, 2021.

Apesar da possibilidade de se executar um código inteiro em paralelo, podem existir algumas limitações que a tornam inviável. O OpenMP permite paralelizar laços que são finitos e com término previamente conhecido, não podendo sofrer alterações de tamanho durante a sua execução. Um laço só poderá ser paralelizado se as suas iterações forem independentes, conforme discutido em Chapman, Jost e van der Pas (2008). Vale a pena destacar que a biblioteca do OpenMP está disponível para as linguagens de programação C, C++ e Fortran e, mais recentemente, para o Python. Diversas áreas de pesquisa e desenvolvimento, além de aplicações industriais, têm tirado proveito da aplicação de técnicas tais como o OpenMP. Por exemplo, tem-se a simulação numérica de reservatórios de óleo e/ou gás (WERNECK, 2016; CALLE, 2017; WERNECK et al., 2019; CARVALHO et al., 2020).

Os (co)processadores Intel Xeon Phi (ZHU et al., 2016; DIETRICH et al., 2017; MALY et al., 2018) são frequentemente usados para aumentar o desempenho dos supercomputadores modernos. A arquitetura Intel MIC foi projetada, especialmente, para a computação de alto desempenho. Os modelos de programação para o coprocessador Xeon Phi incluem o chamado modo *offload*. Neste caso, a aplicação principal é executada na CPU (*Central Processing Unit*) hospedeira e apenas uma parte do código designado pelo programador é executada no coprocessador (MALY et al., 2018).

Dentre as vantagens da arquitetura MIC é possível citar a sua simplicidade de programação e a facilidade em se usar as ferramentas existentes. O código-fonte do aplicativo pode ser modificado mais facilmente na MIC do que na GPU, devido à sua estrutura semelhante à da CPU. Atualmente, os sistemas baseados na arquitetura Intel

MIC têm sido amplamente usados em muitos campos, como na exploração sísmica e na análise do sequenciamento genético na biologia (ZHU *et al.*, 2016).

Objetivo

Tendo em vista o quadro apresentado, tem-se como objetivo a resolução do problema de escoamento não-isotérmico em reservatórios portadores de óleo, considerando-se a aplicação de um método térmico não-convencional, com a introdução de aquecedores estáticos no interior do reservatório, e um modelo a uma equação para descrever a transferência de energia. A estratégia proposta, principal contribuição deste trabalho, utiliza a computação de alto desempenho e engloba, fundamentalmente, a combinação do uso de uma decomposição de operadores, da API OpenMP e de um coprocessador Intel Xeon Phi.

Organização do trabalho

A Introdução foi dedicada a apresentar o contexto no qual esta tese está inserida, em termos da simulação numérica de reservatórios, da recuperação de óleo por meio da aplicação de métodos térmicos, da computação de alto desempenho e do uso de coprocessadores. Destacando que a principal contribuição do trabalho diz respeito à paralelização de partes do simulador numérico. A seguir, passa-se a uma breve explanação sobre como este trabalho está organizado.

O Capítulo 1 trata da modelagem do escoamento não-isotérmico em reservatórios de óleo. Neste capítulo, são apresentadas as propriedades de fluido e de rocha necessárias na construção do modelo físico-matemático, além das equações que expressam os balanços de massa, de quantidade de movimento e de energia. Na sequência, obtém-se as equações diferenciais parciais não-lineares que governam o escoamento, escritas em termos da pressão do óleo e da temperatura média, sem se considerar a hipótese de equilíbrio térmico local. O modelo é completado com a proposição das condições iniciais e de contorno.

Apresenta-se, no Capítulo 2, a metodologia numérica na resolução das equações governantes. Para tanto, utiliza-se o Método das Diferenças Finitas, uma malha de blocos centrados, uma técnica de acoplamento poço-reservatório, uma linearização via o método de Picard e uma decomposição de operadores. Também revisa-se, brevemente, o método dos Gradientes Conjugados que é empregado na solução dos subsistemas de equações algébricas linearizadas, de modo a determinarmos a pressão do óleo e a temperatura média no reservatório, assim como a pressão no poço produtor. Ele contém ainda um fluxograma com os passos executados em um passo de tempo.

No Capítulo 3, foca-se na API OpenMP e na metodologia de paralelização utilizada. Após se discretizar as equações diferenciais parciais que governam o escoamento, obtém-se os subsistemas de equações algébricas cujas matrizes dos coeficientes são esparsas. Como apenas os valores não nulos dos seus elementos são relevantes no processo de obtenção de solução, utiliza-se e introduz-se a técnica *Compressed Sparse Row* (CSR), ou linha esparsa comprimida, para otimizar o uso de memória disponível para o armazenamento da matriz dos coeficientes.

Os resultados obtidos são apresentados e discutidos no Capítulo 4. Fundamentalmente, no que diz respeito ao desempenho computacional, realizou-se testes envolvendo o refinamento da malha computacional e a variação do número de *threads*, para se verificar como eles afetam a aceleração da versão do código numérico paralelizada. Também foram realizadas simulações variando-se algumas propriedades físicas e verificou-se como diferentes alternativas de paralelização influenciam no desempenho computacional.

Finalmente, no último capítulo, são expostas as principais conclusões e propostas novas perspectivas para trabalhos futuros que prosseguirão nesta linha de pesquisa.

1 ESCOAMENTO NÃO-ISOTÉRMICO

Neste capítulo apresenta-se o modelo físico-matemático utilizado para descrever o escoamento bidimensional monofásico não-isotérmico de óleo em um reservatório. Um modelo físico-matemático é uma combinação de um conjunto de equações diferenciais parciais (EDPs) e de suas respectivas condições de contorno e inicial que, quando resolvidas, fornecem as variáveis dependentes, a pressão e a temperatura média do reservatório. No modelo, deve-se levar em consideração que algumas propriedades de fluido e de rocha podem depender das variáveis primitivas, que são a pressão e a temperaturas média. Tais dependências tornam as equações governantes não lineares.

Um reservatório de hidrocarbonetos é um meio poroso e, portanto, consiste de um arcabouço sólido contendo poros interconectados. A parte estrutural do meio é chamada de matriz sólida. A existência dos poros permite que o fluido escoe pelo meio (LAMPE, 2013). Um meio poroso pode conter regiões muito irregulares e outras com um padrão regular. O escoamento em um meio poroso, considerando a escala microscópica, é governado pelas equações conhecidas da Mecânica do Contínuo e da Dinâmica dos Fluidos. Por outro lado, como na prática é quase impossível resolver o problema diretamente na escala microscópica, métodos foram desenvolvidos com a finalidade de se obter as equações válidas na escala de laboratório, ou escala de Darcy. Estes métodos utilizam o conceito de Volume Elementar Representativo (VER) (WHITAKER, 1999) (assumindo-se a sua existência e a separação de escalas) e um processo de homogeneização é aplicado partindo-se da tomada da média volumétrica das equações governantes, na escala microscópica, e utilizando-se os teoremas da média volumétrica (WHITAKER, 1999).

Aqui, primeiramente, aborda-se a questão das propriedades físicas necessárias para a modelagem do escoamento em um meio poroso e isso é feito na Seção 1.1, que trata de uma breve revisão das propriedades físicas que aparecem nas equações governantes.

No decorrer do capítulo, a partir das equações de conservação da massa, de quantidade de movimento e de energia, obtém-se nas Seções 1.3 e 1.4, respectivamente, as EDPs governantes cujas variáveis dependentes são a pressão e a temperatura média (Seção 1.2) do meio. É importante salientar que as equações de energia não consideram o equilíbrio térmico local, e isso significa que a rocha e o fluido tem temperaturas médias diferentes. Finalizando, são apresentadas as condições iniciais e de contorno que devem ser empregadas.

1.1 Propriedades do Fluido e da Rocha

Seção dedicada à introdução das propriedades essenciais para que se possa simular o escoamento no interior do meio poroso, levando-se em consideração, quando for o caso, a dependência delas em função da variação da pressão e da temperatura.

1.1.1 Porosidade

Segundo Rosa, Carvalho e Xavier (2006), a porosidade é uma das mais importantes propriedades das rochas na Engenharia de Reservatórios. Ela é definida como sendo a relação entre o volume de vazios de uma rocha e o volume total da mesma, ou seja, a porosidade determina a capacidade de armazenamento de fluidos na rocha reservatório (DANDEKAR, 2013),

$$\phi = \frac{V_p}{V_t},$$

onde ϕ é a porosidade, V_p o volume de vazios (ou volume poroso) e V_t o volume total que inclui o sólido e os poros (ROSA; CARVALHO; XAVIER, 2006).

A porosidade geralmente varia espacialmente em uma formação rochosa e, nesse caso, o reservatório é dito ser heterogêneo. Existem dois tipos de porosidade: a absoluta e a efetiva. A primeira se refere à relação entre o volume total de vazios da rocha e o volume total do reservatório. Já a porosidade efetiva é a relação entre os espaços vazios interconectados de uma rocha e o volume total do reservatório. Neste texto, a porosidade será entendida como sendo a porosidade efetiva, que é a que importa em termos da produção dos hidrocarbonetos. A porosidade também pode variar em função da pressão do óleo, p , e da temperatura da rocha, T_ϕ , de modo que (ROUSSET, 2010)

$$\phi = [1 + c_\phi (p - p^0) - c_{\phi T} (T_\phi - T^0)] \phi^0,$$

onde ϕ^0 é a porosidade de referência (determinada na pressão de referência, p^0 , e na temperatura de referência, T^0), c_ϕ é o coeficiente de compressibilidade da rocha e $c_{\phi T}$ representa o coeficiente de expansividade térmica da rocha.

1.1.2 Permeabilidade

Outra propriedade de rocha, fundamental, é a permeabilidade absoluta. Ela é uma propriedade tensorial e é definida pelo tensor \mathbf{k} , que representa uma medida da capacidade de um meio poroso em permitir o escoamento de fluidos através de seus poros interconectados. A permeabilidade é uma característica que depende do número, da geometria e do grau de conectividade dos poros. Ela comumente varia de um ponto ao outro e também pode variar em função da direção do escoamento (ERTEKIN; ABOU-KASSEM; KING, 2001). Em muitas aplicações práticas é razoável considerar que a permeabilidade possa ser representada por um tensor diagonal, cujos elementos da diagonal principal são k_x , k_y e k_z e eles correspondem aos valores que a permeabilidade assume segundo as três direções principais, x , y e z , respectivamente (CHEN; HUAN; MA, 2006),

$$\mathbf{k} = \begin{bmatrix} k_x & 0 & 0 \\ 0 & k_y & 0 \\ 0 & 0 & k_z \end{bmatrix},$$

e, bem entendido, os seus valores serão diferentes em um meio anisotrópico. Entretanto, em se tratando de um meio poroso isotrópico, os valores das permeabilidades serão iguais nas diferentes direções para todo o reservatório ($k_x=k_y=k_z$).

1.1.3 Massa específica

A massa específica de um fluido é dada pela razão da sua massa pelo volume por ela ocupado. No caso de um fluido ligeiramente compressível, o seu volume pode ser alterado em função variação da pressão e da contração ou expansão térmica, que depende da temperatura do fluido, T_f . Então ela deve ser calculada na forma

$$\rho_f = [1 + c_o (p - p^0) - c_{oT} (T_f - T^0)] \rho_f^0,$$

onde ρ_f^0 é a massa específica do fluido nas condições de referência e c_{oT} é o coeficiente de expansão térmica do óleo.

1.1.4 Viscosidade

A viscosidade, μ , é a propriedade física que caracteriza a resistência que um fluido apresenta ao escoamento quando uma tensão de cisalhamento é aplicada sobre ele, ela é decorrente da existência do atrito entre as camadas do fluido que escoam. Na realidade, a viscosidade é função tanto da pressão como da temperatura. Porém, aqui, considerar-se-á que a mesma é uma função somente da temperatura (ROUSSET, 2010),

$$\mu = a \exp\left(\frac{b}{T_f - T_{visc}}\right), \quad (1)$$

onde a e b são coeficientes específicos determinados para um dado tipo de óleo e T_{visc} é a temperatura de referência usada na determinação da viscosidade.

É importante destacar, embora seja do conhecimento geral, que os óleos mais viscosos, os óleos pesados, apresentam uma maior resistência ao escoamento e, portanto, a sua extração da jazida requer o fornecimento de uma maior quantidade de energia. Portanto, o uso de uma técnica de recuperação térmica, como a aqui utilizada, pode ajudar nesse quesito.

1.1.5 Fator-Volume-Formação

Sabe-se que uma determinada massa fixa do fluido pode ocupar diferentes volumes dependendo das condições de pressão e temperatura às quais ela está submetida que, certamente, são diferentes no interior e fora do reservatório. Assim sendo, o Fator-Volume-Formação (FVF), B , é usado a fim de converter os volumes, a pressão e temperatura do reservatório, para os seus respectivos valores equivalentes em condições padrão (subscrito sc) (ERTEKIN; ABOU-KASSEM; KING, 2001),

$$B = \frac{V}{V_{sc}},$$

onde V e V_{sc} são, respectivamente, o volume do fluido no reservatório e o volume do fluido nas condições padrão. Uma outra forma de se calculá-lo é a partir da relação entre as massas específicas nas condições de pressão e temperatura padrão, ρ_{fsc} e de reservatório, ρ_f ,

$$B = \frac{\rho_{fsc}}{\rho_f}.$$

1.1.6 Coefficiente de expansão térmica

Tendo em vista que se está interessado no escoamento não-isotérmico em um meio poroso, é importante considerar-se o efeito que a variação de temperatura provoca tanto na matriz porosa quanto no fluido. Para tanto, introduz-se o coeficiente de expansão térmica que fornece uma medida da variação volumétrica em função da temperatura

$$c_{\alpha T} = \frac{1}{V} \frac{\partial V}{\partial T} = \frac{1}{\rho} \frac{\partial \rho}{\partial T},$$

onde $\alpha = \phi$ ou o , indicando que a expansão está relacionada à rocha ou ao óleo, respectivamente. Apesar desse coeficiente ser dependente de outras propriedades, ele é considerado como sendo constante nas simulações.

1.1.7 Compressibilidade

Frequentemente, os fluidos encontrados nos reservatórios de petróleo podem ser classificados como incompressíveis (compressibilidade nula), ligeiramente compressíveis (compressibilidade pequena e que pode ser considerada constante) ou compressíveis (compressibilidade variável e superior a de um fluido ligeiramente compressível), de acordo com a variação da massa específica em relação à pressão ou, de forma equivalente, à sua compressibilidade (ERTEKIN; ABOU-KASSEM; KING, 2001). Assim sendo, a compressibilidade do fluido, ou da rocha, é definida como sendo a variação volumétrica relativa a uma variação de pressão a uma temperatura constante (ERTEKIN; ABOU-KASSEM; KING, 2001). Para o óleo, tem-se

$$c_o = - \frac{1}{V} \frac{\partial V}{\partial p} \Bigg|_{T_f} = \frac{1}{\rho} \frac{\partial \rho}{\partial p} \Bigg|_{T_f},$$

onde V é o volume do óleo e, para a rocha,

$$c_\phi = \frac{1}{V_p} \frac{\partial V_p}{\partial p} \Bigg|_{T_\phi} = \frac{1}{\phi} \frac{\partial \phi}{\partial p} \Bigg|_{T_\phi}.$$

Considerando que o fluido e a rocha são assumidos serem ligeiramente compressíveis, os valores das suas compressibilidades são tomados como sendo constantes.

1.1.8 Aspectos da transferência de calor

Completando esta seção, revisa-se alguns conceitos ligados à transferência de calor. Considera-se, aqui, que a energia pode ser transportada por condução e advecção. A transferência de calor por condução, na escala microscópica, está associada à transferência de energia através da colisão das moléculas, sendo que as mais energéticas transferem energia para as menos energéticas. Logo, a transferência de energia segue no sentido contrário ao do vetor gradiente de temperatura (INCROPERA et al., 2006). O fluxo de calor não pode ser medido diretamente, mas o seu conceito tem um significado físico pois está relacionado com uma grandeza escalar que pode ser medida, chamada de temperatura (OZISIK, 1994). O coeficiente de proporcionalidade κ , definido como a condutividade térmica, define a proporcionalidade entre o fluxo de calor transferido, por um determinado material, e o gradiente de temperatura nesse corpo. Portanto, as condutividades térmicas da rocha e do fluido devem ser conhecidas para que possa-se determinar os fluxos difusivos transferidos por ambos os meios.

A advecção microscópica é um modo eficiente de transferência de calor, proveniente do movimento conjunto das moléculas de um fluido. Tal mecanismo facilita a transferência de calor quando há um gradiente de temperatura. Essa transferência de energia por advecção, entre meios diferentes, também pode ser caracterizada pela introdução de um coeficiente de transferência de calor. Uma grande discussão, acerca do cálculo do coeficiente de transferência de calor, tem sido feita por diversos grupos de pesquisa. O valor do coeficiente de transferência de calor por advecção para um líquido pode variar de 100 a 20.000 W/m²K (CENGEL, 2006). Tal coeficiente não é uma propriedade física como a condutividade térmica, dependendo, de fato, de algumas propriedades físicas do fluido e do tipo de escoamento, além de geometria da superfície sobre a qual o fluido escoar.

Na escala macroscópica, também distingue-se dois tipos de mecanismos de transferência de calor, que são os seus equivalentes aos termos de advecção e de difusão (condução) de calor presentes nas equações microscópicas. Entretanto, agora, em função da aplicação das técnicas de mudança de escalas (MOYNE et al., 2000), ao invés do coeficiente de condução de calor, no termo difusivo, temos o tensor efetivo de dispersão térmica. As componentes desse tensor dependem diretamente da condutividade térmica, da tortuosidade do meio poroso e da dispersão hidrodinâmica (MOYNE et al., 2000). Porém, neste trabalho desconsiderou-se essas duas últimas contribuições.

Em Nield e Bejan (2006) é mencionado que à medida que o valor do coeficiente de advecção macroscópico tende ao infinito, o problema de transferência de calor se aproxima do equilíbrio térmico local, uma vez que os valores mais elevados do coeficiente de transferência de calor implicam uma maior troca térmica entre a rocha e o fluido, e se tem uma transferência de calor “perfeita” e as temperaturas da rocha e do fluido tendem a um mesmo valor devido ao equilíbrio térmico. Entretanto, dependendo das propriedades

térmicas do reservatório e dos fluidos, este equilíbrio térmico pode não ser alcançado e as duas temperaturas médias serão diferentes. Não obstante, ainda é possível, sob certas hipóteses, descrever a transferência de calor macroscópica empregando-se um modelo a uma equação e definindo-se uma temperatura média representativa para ambas as fases (fluido e rocha). Entretanto, no caso mais geral, o indicado seria o uso de um modelo a duas equações, uma para a descrição da transferência de calor em cada fase, sendo que a energia trocada entre elas é contabilizada por um termo fonte que aparece nas equações macroscópicas e que depende diretamente de um coeficiente de troca de calor efetivo e da diferença de temperatura entre as fases (HERINGER et al., 2019).

1.2 Hipóteses assumidas para o escoamento

Discutidas as propriedades de fluido e rocha passa-se, agora, à apresentação das equações de balanço de massa, de quantidade de movimento e de energia, tendo-se em vista a obtenção das EDPs que governam o escoamento e a transferência de calor. Antes disso, porém, enuncia-se as hipóteses que aqui são consideradas:

1. O escoamento ocorre a baixas velocidades;
2. A geometria bidimensional é descrita em coordenadas cartesianas;
3. As condutividades térmicas da rocha e do fluido constantes;
4. Não ocorrem reações químicas;
5. O reservatório é homogêneo e isotrópico no que diz respeito à permeabilidade;
6. A compressibilidade da rocha pequena e constante;
7. O fluido é newtoniano;
8. A transferência de calor por radiação é desprezível;
9. A ausência de estocagem no poço;
10. Não ocorre o equilíbrio térmico local;
11. O escoamento é monofásico e não-isotérmico;
12. Os efeitos gravitacionais são desconsiderados.

Em função de não se assumir o equilíbrio térmico local, introduz-se uma temperatura média, para representar a temperatura do reservatório (fluido + rocha), definida por (MOYNE et al., 2000)

$$(\rho c_p)T = \phi \rho_o c_{p_o} T_f + (1 - \phi) \rho_r c_{p_r} T_\phi,$$

onde T a temperatura média intrínseca e,

$$(\rho c_p) = \phi \rho_o c_{p_f} + (1 - \phi) \rho_r c_{p_\phi},$$

representa a capacidade térmica média do meio poroso. Essas definições, da temperatura e capacidade térmica médias, serão necessárias quando da obtenção do modelo a uma equação que descreve a transferência de energia na escala macroscópica.

1.3 Equação governante para a pressão

Ao se introduzir o fator-volume-formação

$$B = \frac{\rho_{osc}}{\rho_o},$$

a equação de conservação da massa, para o escoamento monofásico em um meio poroso, pode ser escrita na forma alternativa (ERTEKIN; ABOU-KASSEM; KING, 2001)

$$\frac{\partial}{\partial t} \left(\frac{\phi}{B} \right) + \nabla \cdot \left(\frac{\mathbf{v}_o}{B} \right) - \frac{q_{mo}}{V_b \rho_{osc}} = 0, \quad (2)$$

onde \mathbf{v}_o é a velocidade superficial do fluido (óleo), q_{mo} é o termo fonte (unidade de massa por unidade de tempo), V_b é o volume do volume de controle (rocha mais poros) e ρ_{osc} representa a massa específica do óleo em condições padrão de pressão e temperatura (p_{sc} e T_{sc}), respectivamente.

Para escoamentos a baixas velocidades, a equação resultante da conservação da quantidade de movimento para o escoamento de fluidos em meios porosos é a lei de Darcy (DARCY, 1856),

$$\mathbf{v}_o = -\frac{\mathbf{k}}{\mu_o} (\nabla p - \rho_o g \nabla D), \quad (3)$$

onde \mathbf{k} é o tensor de permeabilidade absoluta, μ_o é a viscosidade do fluido, ρ_o é a sua massa específica, g é a magnitude da aceleração da gravidade e D é a profundidade (ROSA;

CARVALHO; XAVIER, 2006). Deve-se ressaltar que a velocidade e a pressão nessa equação são grandezas macroscópicas, ou seja, são médias volumétricas obtidas de seus valores microscópicos (WHITAKER, 1999). Além disso, o tensor de permeabilidade representa uma propriedade efetiva do meio na escala macroscópica (de laboratório ou de Darcy) (WHITAKER, 1999).

Em seguida, substituindo-se a Eq. (3) na Eq. (2), obtém-se

$$\frac{\partial}{\partial t} \left(\frac{\phi}{B} \right) - \nabla \cdot \left[\frac{\mathbf{k}}{B\mu_o} (\nabla p - \rho_o g \nabla D) \right] - \frac{q_{mo}}{V_b \rho_{osc}} = 0. \quad (4)$$

Agora, caso os efeitos gravitacionais sejam desconsiderados e usando-se a relação $q_{mo} = q_{osc} \rho_{osc}$, a Eq. (4) pode ser reescrita como

$$\frac{\partial}{\partial t} \left(\frac{\phi}{B} \right) - \nabla \cdot \left(\frac{\mathbf{k}}{B\mu_o} \nabla p \right) - \frac{q_{osc}}{V_b} = 0. \quad (5)$$

O termo de acúmulo, $\partial(\phi/B)/\partial t$, pode ser reescrito, usando-se as propriedades do fluido e da rocha, de forma a obter-se explicitamente as derivadas parciais da pressão e da temperatura em relação ao tempo (DYRDAHL, 2014). Alcança-se este resultado utilizando-se a regra da cadeia, de forma que

$$\begin{aligned} \frac{\partial}{\partial t} \left(\frac{\phi}{B} \right) &= \frac{1}{B} \frac{\partial \phi}{\partial t} + \phi \frac{\partial}{\partial t} \left(\frac{1}{B} \right) \\ &= \frac{1}{B} \frac{\partial \phi}{\partial p} \frac{\partial p}{\partial t} + \phi \frac{\partial}{\partial p} \left(\frac{1}{B} \right) \frac{\partial p}{\partial t} + \frac{1}{B} \frac{\partial \phi}{\partial T} \frac{\partial T}{\partial t} + \phi \frac{\partial}{\partial T} \left(\frac{1}{B} \right) \frac{\partial T}{\partial t} \\ &= \left[\frac{1}{B} \frac{\partial \phi}{\partial p} + \phi \frac{\partial}{\partial p} \left(\frac{1}{B} \right) \right] \frac{\partial p}{\partial t} + \left[\frac{1}{B} \frac{\partial \phi}{\partial T} + \phi \frac{\partial}{\partial T} \left(\frac{1}{B} \right) \right] \frac{\partial T}{\partial t}. \end{aligned} \quad (6)$$

Neste trabalho, considera-se a dependência de B em relação à pressão e à temperatura

$$B = \frac{B^0}{1 + c_o(p - p^0) - c_{oT}(T - T^0)}, \quad (7)$$

e já viu-se que

$$\phi = [1 + c_\phi(p - p^0) - c_{\phi T}(T - T^0)] \phi^0 \quad (8)$$

onde B^0 e ϕ^0 são, respectivamente, o fator-volume-formação e a porosidade nas condições de referência: p^0 e T^0 . Os termos c_o e c_{oT} representam, respectivamente, os coeficientes de compressibilidade e de expansão térmica do óleo, enquanto que os termos c_ϕ e $c_{\phi T}$ são os coeficientes de compressibilidade e de expansão térmica da rocha, respectivamente.

Usando-se as Eqs. (7) e (8) na Eq. (6) resulta em

$$\frac{\partial}{\partial t} \left(\frac{\phi}{B} \right) = \left(\frac{\phi c_o}{B^0} + \frac{\phi^0 c_\phi}{B} \right) \frac{\partial p}{\partial t} - \left(\frac{\phi c_{oT}}{B^0} + \frac{\phi^0 c_{\phi T}}{B} \right) \frac{\partial T}{\partial t}$$

e, a partir desse resultado, é possível reescrever a Eq. (5) como

$$\Gamma_p \frac{\partial p}{\partial t} - \Gamma_T \frac{\partial T}{\partial t} - V_b \nabla \cdot \left(\frac{\mathbf{k}}{B\mu} \nabla p \right) - q_{sc} = 0, \quad (9)$$

onde os coeficientes dos dois primeiros termos são dados por

$$\Gamma_p = V_b \left(\frac{\phi c_o}{B^0} + \frac{\phi^0 c_\phi}{B} \right)$$

e

$$\Gamma_T = V_b \left(\frac{\phi c_{oT}}{B^0} + \frac{\phi^0 c_{\phi T}}{B} \right)$$

e eles levam em conta os efeitos compressíveis provenientes das variações de pressão e temperatura, respectivamente.

A Eq. (9) é uma equação diferencial parcial não-linear que será utilizada para a determinação da variável dependente pressão do óleo. Para resolver-se a Eq. (9), além de se considerar o seu acoplamento com a equação que fornecerá a temperatura (T), são necessárias que sejam fornecidas as condições auxiliares apropriadas: a condição inicial e as condições de contorno apropriadas.

Como condição inicial utiliza-se

$$p(x, y, t = 0) = p_{ini}(x, y) = p_{inic},$$

onde a pressão inicial, p_{inic} , representa as condições do reservatório antes dele ser perturbado pela produção/injeção. Por outro lado, as condições de contorno externas são as do tipo fluxo nulo nas fronteiras

$$\left(\frac{\partial p}{\partial x}\right)_{x=0,L_x} = \left(\frac{\partial p}{\partial y}\right)_{y=0,L_y} = 0,$$

onde L_x e L_y são os respectivos comprimentos do reservatório nas direções x e y .

1.3.1 Acoplamento poço-reservatório

Sabe-se que a produção do reservatório é contabilizada via o termo de fonte q_{sc} . Não obstante, mais informação sobre a dinâmica do escoamento pode ser obtida se for utilizada uma técnica do tipo acoplamento poço-reservatório (PEACEMAN, 1978; PEACEMAN, 1983).

Deste modo, pode-se relacionar a pressão no poço produtor, p_{wf} , com a vazão de produção, q_{sc} , por intermédio da equação

$$q_{sc} = -J_w (p - p_{wf}), \quad (10)$$

onde J_w é conhecido como sendo o índice de produtividade (PEACEMAN, 1978).

Em função do conhecimento do valor da pressão do poço produtor (produção a pressão constante) ou da sua vazão (produção a vazão constante), a Eq. (10) permitirá o cálculo da pressão no poço caso a vazão seja prescrita e vice-versa. Neste trabalho, optou-se por se estipular uma condição de vazão de produção prescrita, que é mantida constante durante as simulações.

Maiores detalhes sobre como se determina o índice de produtividade serão providos no próximo capítulo, que trata da resolução numérica das equações governantes.

1.4 **Equação governante para a temperatura**

No caso do balanço de energia, deixou-se claro que existem três possibilidades para se descrever a transferência de energia. A primeira assume que as temperaturas médias das fases fluido e sólida são as mesmas, equilíbrio térmico local, e chega-se a uma única equação efetiva (QUINTARD; KAVIANY; WHITAKER, 1997). Na segunda, ainda descreve-se o fenômeno empregando-se uma única equação mas, agora, a hipótese do equilíbrio térmico local não é assumida e a temperatura média é representativa de ambas as fases. Esse modelo é menos restritivo que o primeiro e a forma das equações são semelhantes, sendo que elas são diferenciadas pelo cálculo das propriedades efetivas, ou seja, o tensor efetivo

de dispersão térmica via os problemas de fechamento (MOYNE et al., 2000). A terceira e última alternativa é o modelo a duas equações, que é o menos restritivo e que preconiza a utilização de uma equação macroscópica para descrever o processo de transferência de calor para cada fase. A transferência de energia entre as fases é computada através de termos fontes, em cada equação, que dependem diretamente do produto do coeficiente de transferência de calor efetivo e a diferença de temperatura entre elas (QUINTARD; KAVIANY; WHITAKER, 1997). No modelo a uma equação, sem a hipótese do equilíbrio termodinâmico local, esses termos se cancelam uma vez que, agora, a temperatura é uma média ponderada de ambas as fases e a troca de calor passa a ser interna.

Utilizando o Método da Média Volumétrica (WHITAKER, 1999) ou a técnica de homogeneização por expansões assintóticas (MEI; AURIAULT; NG, 1996), o modelo a uma equação, sem a hipótese do equilíbrio térmico local, pode ser obtido conforme descrito em Moyne et al. (2000). Portanto, a equação efetiva válida na escala de laboratório (ou de Darcy), contabilizando a energia transferida à jazida devido à presença dos poços estáticos de aquecimento (ROUSSET, 2010), assume a forma final

$$\frac{\partial}{\partial t} [(\rho c_p)T] + \nabla \cdot (\rho_o h_o \mathbf{v}_o) - \nabla \cdot (\boldsymbol{\kappa} \nabla T) - \frac{q_H}{V_b} - \frac{\rho_o c_{p_o} T q_{osc}}{V_b} = 0, \quad (11)$$

onde T é a temperatura média macroscópica, h_o é a entalpia do fluido, q_H é o termo fonte que computa a energia transferida pelos aquecedores estáticos e $\boldsymbol{\kappa}$ é o tensor efetivo de dispersão térmica,

$$\boldsymbol{\kappa} = [\phi \kappa_o + (1 - \phi) \kappa_r] \mathbf{I} + \phi \kappa_o \boldsymbol{\tau}_o + (1 - \phi) \kappa_r \boldsymbol{\tau}_r + \phi (\rho c_p)_o \mathbf{D}_o,$$

onde $\boldsymbol{\tau}_o$ e $\boldsymbol{\tau}_r$ representam os tensores de tortuosidade associados ao fluido e a rocha, \mathbf{D}_o é o tensor de dispersão hidrodinâmica (MOYNE et al., 2000) e as seguintes decomposições espaciais foram introduzidas

$$T_f = T + \tilde{T}_f$$

e

$$T_\phi = T + \tilde{T}_\phi,$$

onde \tilde{T}_f e \tilde{T}_ϕ representando as flutuações espaciais da temperatura em torno da temperatura média do meio (T).

Seguindo as ideias de [Rousset \(2010\)](#), a entalpia do fluido, considerado ideal, é calculada por

$$h_o = c_{p_o} T,$$

embora se saiba que a temperatura média, agora, não corresponde a uma igualdade entre as temperaturas do óleo e da rocha.

Desprezando-se os efeitos devidos à tortuosidade e à dispersão hidrodinâmica ([HERINGER et al., 2019](#)) tem-se que

$$\boldsymbol{\kappa} = [\phi \kappa_o + (1 - \phi) \kappa_r] \mathbf{I},$$

onde κ_o e κ_r são, respectivamente, as condutividades térmicas do fluido e da rocha. Quando despreza-se essas contribuições, o presente modelo a uma equação passa a ter o mesmo tensor efetivo de dispersão térmica daquele do modelo a uma equação considerando-se o equilíbrio local e, portanto, o cálculo da entalpia não acarretará um erro adicional.

A Eq. (11) é uma equação diferencial parcial não-linear que será utilizada para a determinação da variável dependente temperatura. Para resolver-se a Eq. (11), além de se considerar a dependência com a pressão p (por meio da velocidade de Darcy), são necessárias que sejam fornecidas as condições inicial e de contorno apropriadas.

Como condição inicial propõe-se

$$T(x, y, t = 0) = T_{ini}(x, y) = T_{inic},$$

onde T_{inic} é a temperatura inicial na qual se encontra o meio poroso, no instante inicial, antes que o processo de produção de óleo tenha começado.

No que diz respeito às condições de contorno externas, assume-se que não há fluxo de energia através das fronteiras do reservatório, ou seja,

$$\left(\frac{\partial T}{\partial x} \right)_{x=0, L_x} = \left(\frac{\partial T}{\partial y} \right)_{y=0, L_y} = 0.$$

2 METODOLOGIA NUMÉRICO-COMPUTACIONAL

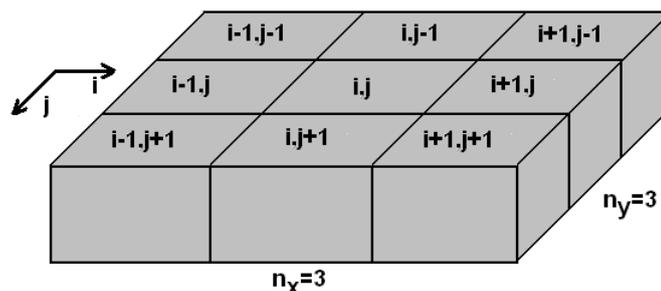
O sistema de equações diferenciais parciais introduzidas no Capítulo 1 não possui uma solução analítica na forma como estão sendo consideradas, sem que hipóteses simplificadoras sejam adotadas. Portanto, necessita-se empregar um método numérico de resolução de EDPs para que se possa obter as soluções aproximadas dessas equações, como o CVFD (*Control-Volume Finite-Difference*) (ERTEKIN; ABOU-KASSEM; KING, 2001). Basicamente, nos métodos que fazem uso de uma malha computacional, o processo de resolução consiste em um particionamento do domínio de resolução e na discretização das equações diferenciais parciais, quando elas são convertidas em equações algébricas.

Nas Seções 2.1 e 2.2, respectivamente, aborda-se a discretização das equações diferenciais parciais, cujas variáveis dependentes são a pressão e a temperatura, usando o Método das Diferenças Finitas e uma malha de bloco centrado. O acoplamento poço-reservatório é visto na Seção 2.3. A utilização de uma decomposição de operadores e de uma técnica de linearização, como estratégias para se resolver os sistemas de equações algébricas, são introduzidas e discutidas na Seção 2.4. Por fim, na Seção 2.5 discorre-se de maneira sucinta sobre o uso do método dos Gradientes Conjugados.

2.1 Discretização da EDP para a pressão

Neste trabalho, empregou-se uma malha de blocos centrados (ERTEKIN; ABOU-KASSEM; KING, 2001; ABOU-KASSEM; ALI; ISLAM, 2006; CHEN; HUAN; MA, 2006), cuja representação esquemática de um domínio bidimensional discretizado pode ser vista na Figura 2, para o sistema de coordenadas cartesianas. A solução numérica é obtida nos nós da malha computacional, localizados nos centros das células (blocos), sendo que n_x e n_y representam o número de células utilizadas nas direções x e y , respectivamente. Os índices inteiros i e j representam as numerações das células nas respectivas direções x e y .

Figura 2 - Representação de um domínio bidimensional discretizado



Fonte: Adaptada de SOUZA, 2013.

Para o problema do escoamento bidimensional, desconsiderando-se os efeitos gravitacionais, pode-se escrever a forma discretizada da equação para a pressão, na célula i, j e no nível temporal $n + 1$ (pressão desconhecida), como

$$\left[\frac{\partial}{\partial x} \left(\mathbb{T}'_x \frac{\partial p}{\partial x} \right) \Delta x + \frac{\partial}{\partial y} \left(\mathbb{T}'_y \frac{\partial p}{\partial y} \right) \Delta y \right]_{i,j}^{n+1} = \left[\Gamma_p \frac{\partial p}{\partial t} + \Gamma_T \frac{\partial T}{\partial t} + q_{sc} \right]_{i,j}^{n+1}, \quad (12)$$

sendo que $(V_b)_{i,j} = (\Delta x \Delta y)_{i,j} L_z$ e introduziu-se as novas variáveis

$$\mathbb{T}'_x \equiv \frac{A_x k_x}{\mu B} \quad \text{e} \quad \mathbb{T}'_y \equiv \frac{A_y k_y}{\mu B},$$

onde $(A_x)_{i,j} = \Delta y_{i,j} L_z$, $(A_y)_{i,j} = \Delta x_{i,j} L_z$, $\Delta x_{i,j}$ e $\Delta y_{i,j}$ são, respectivamente, os espaçamentos da malha nas direções x e y na célula i, j e L_z o comprimento do reservatório na direção z .

Considerando-se um arranjo de malha computacional como o apresentado na Figura 2 e empregando-se uma aproximação do tipo diferença centrada (ERTEKIN; ABOU-KASSEM; KING, 2001), temos que,

$$\frac{\partial}{\partial x} \left(\mathbb{T}'_x \frac{\partial p}{\partial x} \right)_{i,j}^{n+1} \cong \frac{1}{\Delta x_{i,j}} \left[\left(\mathbb{T}'_x \frac{\partial p}{\partial x} \right)_{i+\frac{1}{2},j} - \left(\mathbb{T}'_x \frac{\partial p}{\partial x} \right)_{i-\frac{1}{2},j} \right]^{n+1}. \quad (13)$$

No intuito de aproximar-se as derivadas espaciais da Eq. (13) considera-se, inicialmente, o problema da discretização na direção x do espaço (Figura 3) na qual os subscritos $i - 1/2$ e $i + 1/2$ indicam as faces laterais da célula e o nó i o centro da célula. Escreve-se, então, para aproximações de segunda ordem,

$$\left(\frac{\partial p}{\partial x} \right)_{i+\frac{1}{2},j}^{n+1} \cong \frac{p_{i+1,j}^{n+1} - p_{i,j}^{n+1}}{\Delta x_{i+\frac{1}{2},j}}$$

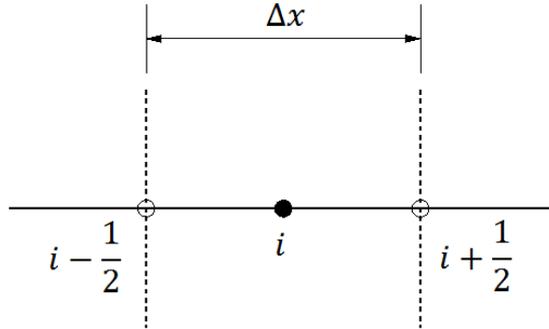
e

$$\left(\frac{\partial p}{\partial x} \right)_{i-\frac{1}{2},j}^{n+1} \cong \frac{p_{i,j}^{n+1} - p_{i-1,j}^{n+1}}{\Delta x_{i-\frac{1}{2},j}},$$

onde $\Delta x_{i\pm 1/2,j}$ é a distância entre os centros das células i e $i \pm 1$. Similarmente, é possível obter-se as aproximações para as derivadas na direção y .

Introduz-se, em seguida, a transmissibilidade na direção x como sendo dada por

Figura 3 - Malha unidimensional.



Fonte: O autor, 2021.

$$\mathbb{T}_{x,i\pm\frac{1}{2},j}^{n+1} = \left(\frac{A_x k_x}{\mu B \Delta x} \right)_{i\pm\frac{1}{2},j}^{n+1},$$

onde é utilizada uma média harmônica para a determinação da área e da permeabilidade na interface $i \pm 1/2, j$, a partir dos seus valores conhecidos em i, j e $i \pm 1, j$. Em se tratando das propriedades de fluido uma média aritmética é aplicada (ERTEKIN; ABOU-KASSEM; KING, 2001). Sem maiores dificuldades, uma expressão similar pode ser obtida para a transmissibilidade na direção y .

Prosseguindo, utilizando-se uma formulação totalmente implícita no tempo, é possível obter-se a forma final discretizada da Eq. (12),

$$\begin{aligned} & \mathbb{T}_x \Big|_{i+1/2,j}^{n+1} (p_{i+1,j}^{n+1} - p_{i,j}^{n+1}) - \mathbb{T}_x \Big|_{i-1/2,j}^{n+1} (p_{i,j}^{n+1} - p_{i-1,j}^{n+1}) \\ & + \mathbb{T}_y \Big|_{i,j+1/2}^{n+1} (p_{i,j+1}^{n+1} - p_{i,j}^{n+1}) - \mathbb{T}_y \Big|_{i,j-1/2}^{n+1} (p_{i,j}^{n+1} - p_{i,j-1}^{n+1}) \\ & = \frac{(\Gamma_p)_{i,j}^{n+1}}{\Delta t} (p_{i,j}^{n+1} - p_{i,j}^n) + \frac{(\Gamma_T)_{i,j}^{n+1}}{\Delta t} (T_{i,j}^{n+1} - T_{i,j}^n) + (q_{sc})_{i,j}^{n+1} \end{aligned} \quad (14)$$

onde adotou-se aproximações de Euler recuadas no tempo,

$$\left(\frac{\partial p}{\partial t} \right)_{i,j}^{n+1} \cong \frac{p_{i,j}^{n+1} - p_{i,j}^n}{\Delta t}$$

e

$$\left(\frac{\partial T}{\partial t}\right)_{i,j}^{n+1} \cong \frac{T_{i,j}^{n+1} - T_{i,j}^n}{\Delta t},$$

nas quais o sobrescrito n indica o nível de tempo no qual a pressão e a temperatura média são conhecidas.

2.2 Discretização da EDP para a temperatura

As etapas da discretização da equação para a temperatura são realizadas de modo análogo ao desenvolvimento mostrado para o caso da equação para a pressão. Então, o ponto de partida é a Eq. (11)

$$\frac{\partial}{\partial t} [(\rho c_p)T] - \nabla \cdot (\boldsymbol{\kappa} \nabla T) = \frac{q_H}{V_b} + \frac{\rho_o h_o q_{osc}}{V_b} - \nabla \cdot (\rho_o h_o \mathbf{v}_o) \quad (15)$$

onde $h_o = c_p T$.

Assim como no caso da equação para a pressão, assume-se um escoamento bidimensional, no plano xy , e sem os efeitos gravitacionais. A partir da Eq. (15), considerando-se que a equação deve ser avaliada na célula i, j e no nível de tempo $n + 1$, no qual as temperaturas são desconhecidas, tem-se

$$\begin{aligned} \left[\frac{\partial}{\partial x} \left(\kappa_x \frac{\partial T}{\partial x} \right) + \frac{\partial}{\partial y} \left(\kappa_y \frac{\partial T}{\partial y} \right) \right]_{i,j}^{n+1} &= \left[\frac{\partial}{\partial t} [(\rho c_p)T] - \frac{q_H}{V_b} - \frac{\rho_o h_o q_{osc}}{V_b} \right]_{i,j}^{n+1} \\ &+ \left[\frac{\partial}{\partial x} (\rho_o u_o h_o) + \frac{\partial}{\partial y} (\rho_o v_o h_o) \right]_{i,j}^{n+1}, \end{aligned}$$

que possui uma forma semelhante à da Eq. (12). Então, empregando-se uma metodologia análoga à utilizada na discretização da Eq. (12) e as componentes do vetor velocidade dadas pela lei de Darcy:

$$\begin{aligned} &\mathbb{K}_x \Big|_{i+1/2,j}^{n+1} (T_{i+1,j} - T_{i,j})^{n+1} - \mathbb{K}_x \Big|_{i-1/2,j}^{n+1} (T_{i,j} - T_{i-1,j})^{n+1} \\ &+ \mathbb{K}_y \Big|_{i,j+1/2}^{n+1} (T_{i,j+1} - T_{i,j})^{n+1} - \mathbb{K}_y \Big|_{i,j-1/2}^{n+1} (T_{i,j} - T_{i,j-1})^{n+1} \end{aligned}$$

$$= V_b \left[\frac{(\rho C_p)}{\Delta t} \right]_{i,j}^{n+1} (T_{i,j}^{n+1} - T_{i,j}^n) + (\rho_o h_o q_{sc})_{i,j}^{n+1} + \Phi_{i,j}^{n+1} + (q_H)_{i,j}^{n+1} \quad (16)$$

onde

$$\mathbb{K}_x \Big|_{i\pm 1/2,j}^{n+1} = \left(\frac{A_x \kappa_x}{\Delta x} \right)_{i\pm 1/2,j}^{n+1}, \quad \mathbb{K}_y \Big|_{i,j\pm 1/2}^{n+1} = \left(\frac{A_y \kappa_y}{\Delta y} \right)_{i,j\pm 1/2}^{n+1},$$

e

$$\begin{aligned} \Phi_{i,j}^{n+1} = & \rho_{osc} [(h_o \mathbb{T}_x)_{i-1/2,j}^{n+1} (p_{i,j} - p_{i-1,j})^{n+1} - (h_o \mathbb{T}_x)_{i+1/2,j}^{n+1} (p_{i+1,j} - p_{i,j})^{n+1}] \\ & + \rho_{osc} [(h_o \mathbb{T}_y)_{i,j-1/2}^{n+1} (p_{i,j-1} - p_{i,j})^{n+1} - (h_o \mathbb{T}_y)_{i,j+1/2}^{n+1} (p_{i,j+1} - p_{i,j})^{n+1}], \end{aligned}$$

com os termos tendo sido reescritos de forma a utilizar-se as transmissibilidades na definição de $\Phi_{i,j}^{n+1}$. Essa equação servirá de base para a estratégia de resolução numérica usando uma decomposição de operadores.

2.3 Acoplamento poço-reservatório

No modelo proposto, optou-se por utilizar uma técnica de acoplamento poço-reservatório. Então, mais detalhes devem ser fornecidos para que se entenda como a pressão no poço vertical é calculada numericamente.

Viu-se que o termo fonte $(q_{sc})_{i,j}^{n+1}$ é utilizado para incluir, no modelo adotado, a pressão no poço via a Eq. (10). Neste caso, a forma discreta dessa equação é

$$(q_{sc})_{i,j}^{n+1} = - (J_w)_{i,j}^{n+1} \left[p_{i,j}^{n+1} - (p_{wf})_{i,j}^{n+1} \right],$$

sendo que o índice de produtividade, J_w , é dado por (PEACEMAN, 1983)

$$(J_w)_{i,j}^{n+1} = \left[\frac{2\pi \sqrt{k_x k_y} L_z}{B \mu \ln \left(\frac{r_{eq}}{r_w} \right)} \right]_{i,j}^{n+1}$$

onde r_w é o raio do poço e o raio equivalente, r_{eq} , é calculado através da equação (PEACEMAN, 1983)

$$r_{eq} = 0,28 \left[\frac{\sqrt{\sqrt{\frac{k_y}{k_x}} (\Delta x)^2 + \sqrt{\frac{k_x}{k_y}} (\Delta y)^2}}{\sqrt[4]{\frac{k_y}{k_x}} + \sqrt[4]{\frac{k_x}{k_y}}} \right]_{i,j},$$

para um poço vertical.

2.4 Decomposição de operadores e linearização

As Eqs. (14) e (16) formam um sistema de equações algébricas não-lineares que quando resolvido conduz aos valores das variáveis p e T (ISLAM et al., 2010). Para tanto, a velocidade de Darcy na Eq. (16) é escrita em termos da pressão via a Eq. (3). A temperatura também é explicitada a partir da relação $h_o = c_{po}T$. Nos dois casos, a Eq. (16) permanece com uma não-linearidade no termo que contém originariamente a velocidade de Darcy. A resolução desse sistema, para uma malha computacional com um número de células na casa dos milhares, já começa a exigir uma quantidade de memória e de processamento significativas. Consequentemente, a resolução numérica torna-se mais custosa computacionalmente quando é necessário empregar um número ainda maior de células, seja devido ao tamanho do reservatório ou por uma necessidade específica como, por exemplo, a captura das heterogeneidades do reservatório.

Uma outra questão importante é a forma final das equações algébricas, que pode ser diferente em função das escolhas feitas na modelagem físico-matemática e na discretização. Por exemplo, Singh, Goerke e Kolditz (2011) trataram do problema do escoamento não-isotérmico, em meios porosos, reescrevendo o termo de advecção, da equação do balanço de energia, de modo a explicitar o efeito de resfriamento Joule-Thompson. Como é sabido, existem métodos numéricos específicos para resolver diferentes tipos de EDPs (VENNEMO, 2016). Em um mesmo sistema de EDPs é possível, por exemplo, existirem equações parabólicas e hiperbólicas. Portanto, seria desejável que cada equação diferencial pudesse ser resolvida usando o melhor método disponível, tendo em vista as suas características matemáticas.

Na literatura, existem propostas de estratégias de resolução numérica, para o problema do escoamento não-isotérmico em meios porosos, que baseiam-se no uso de uma decomposição de operadores (*operator splitting*), tais como as encontradas nos trabalhos de Rousset (2010), Dyrdaahl (2014) e Vennemo (2016). Neste tipo de solução, o sistema original, escrito em termos de p e T , é dividido em dois subsistemas de equações para cada uma dessas variáveis e prevê a troca de informações entre esses dois subsistemas. No caso

mais geral, isto permite, por exemplo, que diferentes métodos de solução sejam aplicados a cada subsistema, em função da classificação das EDPs correspondentes. Este tipo de abordagem já vem sendo empregada com sucesso há bastante tempo em outros métodos, tais como nos métodos do tipo IMPES (*Implicit Pressure Explicit Saturation*) (ERTEKIN; ABOU-KASSEM; KING, 2001; CHEN; HUAN; LI, 2004) utilizados na resolução de problemas de escoamentos multifásicos em meios porosos.

Um dos objetivos da decomposição de operadores é a de resolver computacionalmente de maneira mais eficiente os dois subsistemas, do que quando o sistema é resolvido com todas as equações acopladas. Segundo Vennemo (2016), os métodos de decomposição de operadores têm sido aplicados a uma ampla gama de problemas, incluindo as simulações de escoamento de água subterrânea, a modelagem da poluição do ar e os problemas envolvendo a combustão. No entanto, ao desacoplar-se as equações introduz-se uma fonte adicional de erro. Por isso, a convergência e a acurácia dos métodos de decomposição de operadores têm sido extensivamente estudada e sabe-se que algumas precauções devem ser tomadas, em função do tipo de equação que se está considerando (MAES et al., 2015). No caso estudado por Vennemo (2016), por exemplo, mostra-se que o uso de iterações internas para determinar p e T devem estar condicionadas ao uso de iterações externas para se verificar a convergência do sistema como um todo. Todos os testes nos quais a convergência foi determinada pelos valores dos erros sendo menores ou iguais a tolerâncias pré-estabelecidas, e não pelo número de iterações prescritas, levaram aos melhores resultados. Assim sendo, procede-se da mesma forma no presente trabalho.

Do ponto de vista da implementação computacional, existe uma vantagem quando os subsistemas de equações são semelhantes, uma vez que as mesmas rotinas de resolução utilizadas em um subsistema podem ser usadas no outro. Por isso, optou-se por mostrar as Eqs. (14) e (16) de modo que em ambas as matrizes dos coeficientes fossem pentadiagonais e que a troca de informações entre os dois subsistemas se desse a partir dos termos fontes. Desta forma, considerando uma decomposição de operadores que leve a uma sequência de resolução primeiramente para a determinação da pressão p e, em seguida, da temperatura T , as Eqs. (14) e (16) são reescritas respectivamente como

$$\begin{aligned} & \mathbb{T}_y \Big|_{i,j-1/2}^{v,n+1} p_{i,j-1}^{v+1,n+1} + \mathbb{T}_x \Big|_{i-1/2,j}^{v,n+1} p_{i-1,j}^{v+1,n+1} \\ & - \left[\mathbb{T}_y \Big|_{i,j-1/2}^{v,n+1} + \mathbb{T}_x \Big|_{i-1/2,j}^{v,n+1} + \frac{(\Gamma_p)_{i,j}^{v,n+1}}{\Delta t} + \mathbb{T}_x \Big|_{i+1/2,j}^{v,n+1} + \mathbb{T}_y \Big|_{i,j+1/2}^{v,n+1} \right] p_{i,j}^{v+1,n+1} \\ & + \mathbb{T}_x \Big|_{i+1/2,j}^{v,n+1} p_{i+1,j}^{v+1,n+1} + \mathbb{T}_y \Big|_{i,j+1/2}^{v,n+1} p_{i,j+1}^{v+1,n+1} \end{aligned}$$

$$= - \frac{(\Gamma_p)_{i,j}^{v,n+1}}{\Delta t} p_{i,j}^n + \frac{(\Gamma_T)_{i,j}^{v,n+1}}{\Delta t} (T_{i,j}^{\bar{w},n+1} - T_{i,j}^n) + (q_{sc})_{i,j}^{n+1} \quad (17)$$

e

$$\begin{aligned} & \mathbb{K}_x \Big|_{i,j-1/2}^{w,n+1} T_{i,j-1}^{w+1,n+1} + \mathbb{K}_x \Big|_{i-1/2,j}^{w,n+1} T_{i-1,j}^{w+1,n+1} \\ & - \left\{ \mathbb{K}_y \Big|_{i,j-1/2}^{w,n+1} + \mathbb{K}_x \Big|_{i-1/2,j}^{w,n+1} + V_b \left[\frac{(\rho C_p)}{\Delta t} \right]_{i,j}^{w,n+1} + \mathbb{K}_x \Big|_{i+1/2,j}^{w,n+1} + \mathbb{K}_y \Big|_{i,j+1/2}^{w,n+1} \right\} T_{i,j}^{w+1,n+1} \\ & + \mathbb{K}_x \Big|_{i+1/2,j}^{w,n+1} T_{i+1,j}^{w+1,n+1} + \mathbb{K}_y \Big|_{i,j+1/2}^{w,n+1} T_{i,j+1}^{w+1,n+1} \\ & = - V_b \left[\frac{(\rho C_p)}{\Delta t} \right]_{i,j}^{w,n+1} T_{i,j}^n + (\rho_0 h_0 q_{sc})_{i,j}^{w,n+1} + \Phi_{i,j}^{\bar{v},n+1} + (q_H)_{i,j}^{n+1}, \end{aligned}$$

onde os índices v e w referem-se, respectivamente, aos níveis iterativos externos para a obtenção de p e T . O sobrescrito \bar{v} indica que o termo $\Phi_{i,j}^{n+1}$ está sendo avaliado com as propriedades determinadas no nível iterativo v , para o qual se obteve as pressões em $v+1, n+1$, também usadas em $\Phi_{i,j}^{n+1}$, obtidas quando da solução da Eq. (17). O mesmo tipo de raciocínio foi usado para o termo $T_{i,j}^{\bar{w},n+1}$.

Assim sendo, considerou-se uma linearização a partir do uso de coeficientes que são determinados em uma iteração externa, onde há um teste de convergência, precedida de iterações internas para a obtenção de p (iterações em v) e para T (iterações em w), seguindo de uma iteração de Picard (ERTEKIN; ABOU-KASSEM; KING, 2001). Em resumo, a ordem de resolução é:

1. Cálculo dos coeficientes e dos termos fontes das equações avaliados nas iterações v e w ;
2. Resolução para a determinação de p^{n+1} considerando o valor de T^{n+1} mais recentemente calculado, empregando um método iterativo de resolução de sistemas de equações algébricas;
3. Teste de convergência para as iterações internas: pressão;
4. Resolução para T^{n+1} empregando o valor mais recente de p^{n+1} , empregando um método iterativo de resolução de sistemas de equações algébricas;
5. Teste de convergência para as iterações internas: temperatura;

6. Teste de convergência para as iterações externas: método de Picard.

Na solução de cada subproblema foi utilizado o método iterativo dos Gradientes Conjugados.

2.5 Métodos de resolução de sistemas lineares

Uma opção para a resolução dos dois subsistemas, após linearização, seria a de se utilizar o Método dos Gradientes Conjugados (*Conjugated Gradient*, CG) (SAAD, 2003). Esse é um método iterativo muito conhecido que é empregado frequentemente para resolver sistemas de N equações lineares com N incógnitas (ERTEKIN; ABOU-KASSEM; KING, 2001). Ele é exemplificado, aqui, em se tratando da resolução de um sistema de equações algébricas escrito na forma matricial:

$$\mathbf{Ax} = \mathbf{d},$$

onde \mathbf{A} é a matriz dos coeficientes (esparsa), positiva-definida e simétrica, \mathbf{x} é o vetor incógnita, cujas componentes se deseja calcular, e \mathbf{d} é o vetor dos termos conhecidos. Resolver esse sistema é equivalente à minimização da função quadrática

$$F(\mathbf{x}^k + a^k \mathbf{v}^k) = \frac{1}{2} [\mathbf{x}^k \cdot \mathbf{Ax}^k - 2\mathbf{d} \cdot \mathbf{x}^k + \mathbf{d} \cdot \mathbf{A}^{-1}\mathbf{d} + a^k(2\mathbf{p}^k \cdot \mathbf{Ax}^k - 2\mathbf{p}^k \cdot \mathbf{d}) + (a^k)^2 \mathbf{p}^k \cdot \mathbf{Ap}^k],$$

onde \mathbf{x}^k é o vetor calculado na iteração k e o seu valor na próxima iteração é dado por

$$\mathbf{x}^{k+1} = \mathbf{x}^k + a^k \mathbf{p}^k,$$

onde a^k é um escalar e \mathbf{p}^k é o vetor direção de busca.

Da derivada da função F , em relação a a^k , sabendo-se que $\mathbf{p}^k \cdot \mathbf{r}^k = \mathbf{r}^k \cdot \mathbf{r}^k$, obtém-se

$$a^k = \frac{\mathbf{p}^k \cdot (\mathbf{d} - \mathbf{Ax}^k)}{\mathbf{p}^k \cdot \mathbf{Ap}^k} = \frac{\mathbf{p}^k \cdot \mathbf{r}^k}{\mathbf{p}^k \cdot \mathbf{Ap}^k} = \frac{\mathbf{r}^k \cdot \mathbf{r}^k}{\mathbf{p}^k \cdot \mathbf{Ap}^k},$$

onde o vetor resíduo é definido, na próxima iteração, por

$$\mathbf{r}^{k+1} = \mathbf{r}^k + a^k \mathbf{Ap}^k,$$

e

$$\mathbf{p}^k = \mathbf{r}^k + b^k \mathbf{p}^{k-1}.$$

O valor do escalar b^k é determinado na forma:

$$\mathbf{b}^k = \frac{\mathbf{r}^{k+1} \cdot \mathbf{r}^{k+1}}{\mathbf{r}^k \cdot \mathbf{r}^k}.$$

O Algoritmo 1 mostra a sequência de operações associadas à implementação do método dos gradientes conjugados na busca pela solução de um sistema linear arbitrário $\mathbf{Ax} = \mathbf{d}$.

Algoritmo 1: Gradientes Conjugados

```

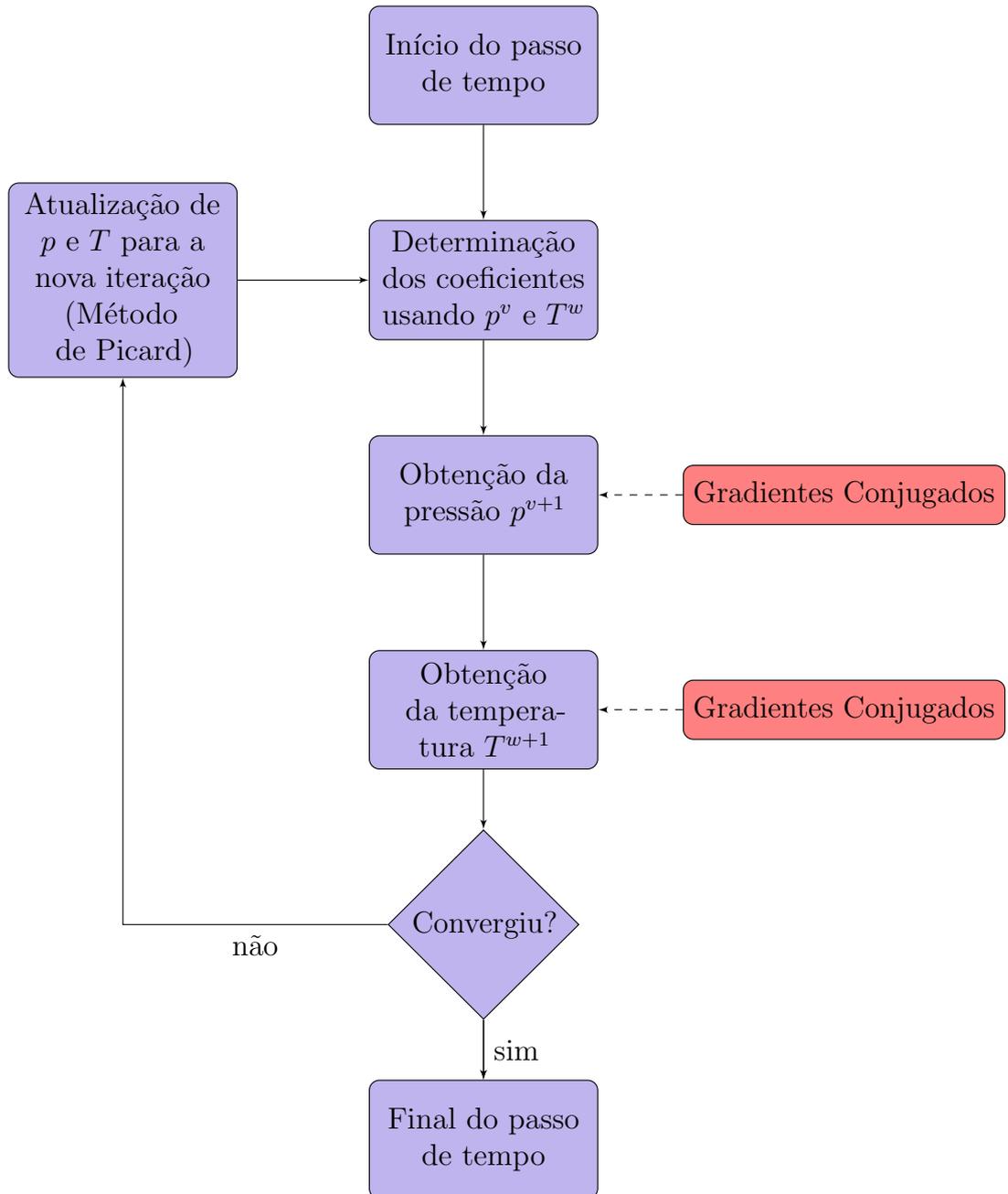
1 Entrada :  $\mathbf{x}^0$ ;  $\mathbf{A}$ ;  $\mathbf{d}$ ;  $\mathbf{r}^0 = \mathbf{p}^0 = \mathbf{d} - \mathbf{Ax}^0$  o número de iterações  $n$  e a tolerância
    $tol$                                      /*  $\mathbf{r}^0$  = vetor de resíduos */
2 while  $|x^{k+1} - x^k| > tol$  do
3    $a^k = \frac{\mathbf{r}^k \cdot \mathbf{r}^k}{\mathbf{p}^k \cdot \mathbf{A}\mathbf{p}^k}$ 
4    $\mathbf{x}^{k+1} = \mathbf{x}^k + a^k \mathbf{p}^k$ 
5    $\mathbf{r}^{k+1} = \mathbf{r}^k + a^k \mathbf{A}\mathbf{p}^k$ 
6    $\mathbf{b}^k = \frac{\mathbf{r}^{k+1} \cdot \mathbf{r}^{k+1}}{\mathbf{r}^k \cdot \mathbf{r}^k}$ 
7    $\mathbf{p}^k = \mathbf{r}^k + b^k \mathbf{p}^{k-1}$ 
8 end
9  $\mathbf{x} = \mathbf{x}^{k+1}$ 

```

Estabeleceu-se como critério de convergência que a diferença absoluta da variação das variáveis dependentes (pressão e temperatura), entre as iterações, seja menor que as tolerâncias pré-estabelecidas: tol_p para a pressão e tol_T para a temperatura. As mesmas verificações são empregadas tanto nas iterações internas (método dos Gradientes Conjugados) quanto externas (método de Picard).

Por último, na Figura 4, é fornecido o fluxograma mostrando os passos executados para a obtenção numérica dos valores de p e T , no simulador desenvolvido nesta tese, já levando em conta a decomposição de operadores, a linearização e o uso do métodos dos gradientes conjugados.

Figura 4 - Fluxograma para um passo de tempo



Fonte: O autor, 2021.

3 PARALELIZAÇÃO

A computação de alto desempenho (*High Performance Computing* - HPC), termo amplamente utilizado na computação científica, tem despertado o interesse de muitos pesquisadores nos últimos anos. Ela contempla qualquer conjunto de técnicas que visem a otimizar, ou até mesmo viabilizar, o processamento eficiente de códigos numéricos. Pode-se citar, como uma destas técnicas, o uso do processamento em paralelo em *clusters* e supercomputadores (HAGER; WELLEIN, 2010). Nesse contexto, este capítulo discorre sobre algumas das técnicas mais eficientes de armazenamento de dados e o processamento em paralelo usando a interface de programação OpenMP.

3.1 Técnicas de compressão de matrizes

No decorrer dos anos, o desenvolvimento científico tem exigido que as simulações numéricas forneçam resultados confiáveis e acurados de problemas cada vez mais complexos. Para poder avançar, a computação científica tem como desafio superar as diversas barreiras que surgem devido à limitação física dos computadores, tais como a demanda por um processamento numérico cada vez mais rápido, por uma maior capacidade de armazenamento de dados e por melhores recursos de visualização gráfica, entre outros.

No que diz respeito à limitação do armazenamento de dados, algumas técnicas de compressão de matrizes foram desenvolvidas na tentativa de atenuar esse problema. Dentre elas, vale destacar (ANTUNES FILHO; XAVIER, 2015):

1. Matriz no Formato Denso ou *Dense Format*;
2. Matriz no Formato Coordenado ou *Coordinate Format* (COO);
3. Matriz no Formato de Linha Comprimida ou *Compressed Sparse Row* (CSR);
4. Matriz no Formato de Coluna Comprimida ou *Compressed Sparse Column* (CSC);
5. *Matrix Market Format*.

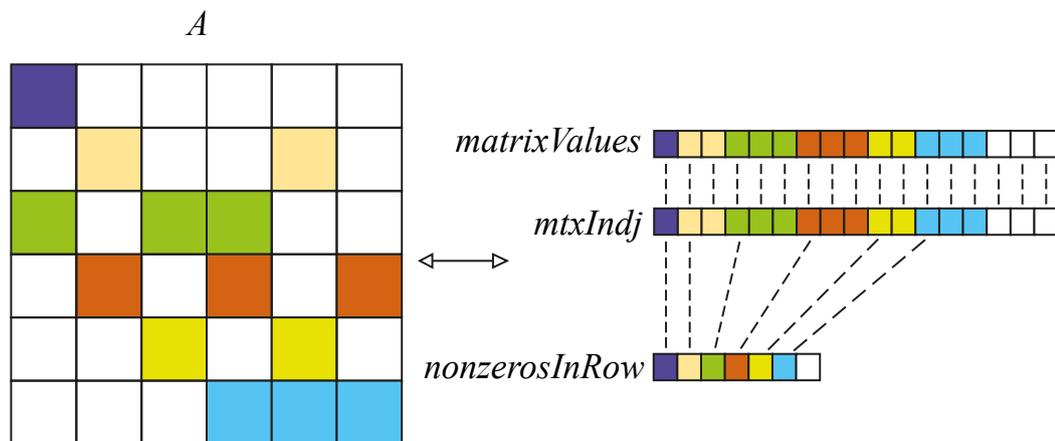
A fim de otimizar o uso da memória disponível, a proposta dessas técnicas, com exceção da primeira, é a de armazenar apenas os elementos relevantes da matriz dos coeficientes associada a um sistema linear, ou seja, os seus elementos não nulos (ANTUNES FILHO; XAVIER, 2015). Os sistemas lineares originários da linearização das equações algébricas não-lineares, discutidos anteriormente no Capítulo 2, têm como característica o fato de que as suas matrizes dos coeficientes são esparsas e possuem uma grande dimensão. Devido ao tamanho destes sistemas, é necessário o uso de muita memória RAM (*Random*

Access Memory) quando da resolução numérica deles. Por isso, o uso de uma técnica de compressão de matrizes torna-se muito útil.

A técnica *Compressed Sparse Row* (CSR), ou Linha Esparsa Comprimida, utiliza três vetores para referenciar e armazenar os valores não nulos da matriz dos coeficientes. Fundamentalmente, o vetor *matrixValues* armazena os valores não nulos da matriz, o vetor *mtxIndj* armazena os índices referentes às colunas em que os elementos não nulos estão posicionados na matriz e o vetor *nonzerosInRow* armazena os índices identificando a posição no vetor *matrixValues* onde se inicia uma nova linha da matriz.

A título de ilustração, a Figura 5 mostra uma representação esquemática da compressão de uma matriz utilizando a técnica CSR, onde os elementos não nulos de cada linha estão representados por quadrados da mesma cor. Cada elemento não nulo da matriz \mathbf{A} é colocado em sequência no vetor *matrixValues*, ao mesmo tempo em que os índices que representam as colunas (da matriz) de cada um destes elementos são armazenados no vetor *mtxIndj*. Por último, o vetor *nonzerosInRow* armazena a posição do primeiro elemento não nulo de cada cor (linha da matriz) no vetor *matrixValues*.

Figura 5 - Ilustração da aplicação da técnica CSR



Fonte: Adaptada de [WERNECK, 2016](#).

Como exemplo de aplicação, considera-se a matriz não simétrica esparsa

$$\mathbf{A} = \begin{pmatrix} 10 & 0 & 0 & 0 & -2 & 0 \\ 3 & 9 & 0 & 0 & 0 & 3 \\ 0 & 7 & 8 & 7 & 0 & 0 \\ 3 & 0 & 8 & 7 & 5 & 0 \\ 0 & 8 & 0 & 9 & 9 & 13 \\ 0 & 4 & 0 & 0 & 2 & -1 \end{pmatrix}$$

que deve ser comprimida via a técnica CSR. Portanto, após a sua utilização os seguintes

vetores são obtidos:

$$\begin{aligned} \text{matrixValues} &= [10 \ -2 \ 3 \ 9 \ 3 \ 7 \ 8 \ 7 \ 3 \ 8 \ 7 \ 5 \ 8 \ 9 \ 9 \ 13 \ 4 \ 2 \ -1]; \\ \text{mtxIndj} &= [0 \ 4 \ 0 \ 1 \ 5 \ 1 \ 2 \ 3 \ 0 \ 2 \ 3 \ 4 \ 1 \ 3 \ 4 \ 5 \ 1 \ 4 \ 5]; \\ \text{nonzerosInRow} &= [0 \ 2 \ 5 \ 8 \ 12 \ 16 \ 19]. \end{aligned}$$

O número 19, armazenado na última posição do vetor *nonzerosInRow*, se refere à posição do primeiro elemento não nulo caso existisse mais uma linha. A razão para tal é que este vetor é utilizado para calcular o número de elementos não nulos, em cada linha da matriz, através da subtração de dois valores consecutivos. Portanto, para se calcular o número de elementos não nulos na última linha, torna-se necessário o armazenamento desse último valor.

Caso a matriz seja simétrica, será necessário armazenar apenas os valores da diagonal principal e dos elementos acima (ou abaixo) da mesma. A técnica CSR foi aqui escolhida para o desenvolvimento dos algoritmos paralelizados, no que diz respeito à resolução dos sistemas de equações algébricas.

3.2 Processamento em paralelo com o OpenMP

De acordo com o que já foi introduzido, a computação de alto desempenho é uma disciplina da computação científica que tem como principal objetivo aumentar a velocidade de processamento ou, até mesmo, viabilizar a realização de simulações numéricas de problemas físicos complexos, difíceis de serem tratados por outras metodologias de resolução. Segundo [Venâncio Júnior e Silva \(2004\)](#), uma das técnicas da computação científica de alto desempenho é o processamento em paralelo, o qual tem como objetivo dividir uma determinada tarefa em várias menores e, também, executar cada uma delas em diferentes processadores ou máquinas. As técnicas de processamento em paralelo mais difundidas atualmente são as de memória distribuída e as de memória compartilhada.

Entende-se por técnicas de memória distribuída aquelas que se aplicam aos computadores contendo vários processadores que possuem seus próprios recursos de memória e que estão interconectados através de uma rede local. Atualmente, o *Message Passing Interface* (MPI) tem sido o mais utilizado nestes casos e o seu funcionamento se dá basicamente via a troca de mensagens e dados entre os processadores. Por sua vez, as técnicas de memória compartilhada são utilizadas em ambientes que possuem vários núcleos de processamento compartilhando o mesmo recurso de memória. Nestes casos, a utilização da interface de programação *Open Multi-processing* (OpenMP) tem crescido bastante nos últimos anos, uma vez que as funcionalidades da mesma facilitam o desenvolvimento de aplicações paralelizadas em ambientes com memória compartilhada ([VENÂNCIO JÚNIOR; SILVA,](#)

2004; CHAPMAN; JOST; van der PAS, 2008).

Uma das razões que vem contribuindo para um intenso crescimento no campo da computação de alto desempenho é a evolução da capacidade computacional apresentada pelas GPUs (*Graphics Processing Unit*), Coprocessadores e CPUs (*Central Processing Unit*). Paralelamente, o surgimento de algumas APIs (*Application Programming Interface*) com suporte ao paralelismo possibilitaram a utilização da real capacidade computacional desses dispositivos. Dentre elas, destacam-se o OpenMP e o MPI, as quais são utilizadas para paralelizar códigos (ou trechos deles) para a execução em CPUs, bem como da API CUDA (*Compute Unified Architecture*), o OpenCL (*Open Computing Language*) e o OpenACC (*Open Accelerator*) para a execução paralelizada em unidades gráficas de processamento. Como exemplos de alguns trabalhos, que utilizaram a programação paralelizada, vale ressaltar os de Meister e Bader (2015), DeVries et al. (2013) e Gillberg, Sourouri e Cai (2012), os quais utilizaram uma programação híbrida (com o objetivo de unir dispositivos de computação com diferentes arquiteturas trabalhando em conjunto) e os trabalhos de Werneck et al. (2019), Liu, Tafti e Li (2014) e Terboven et al. (2008) que utilizaram o OpenMP, foco deste trabalho.

Em função de como é feita a paralelização, a implementação do processamento em paralelo pode ser classificada em:

1. Explícita: quando o programador implementa o paralelismo;
2. Implícita: quando o compilador implementa o paralelismo;
3. Semi-implícita: quando o programador passa as diretivas para o compilador realizar o paralelismo.

sendo que a API OpenMP se encaixa na última classificação.

3.2.1 A interface OpenMP

O OpenMP é uma interface de programação que é desenvolvida e mantida pelo grupo OpenMP *Architecture Review Board* (ARB). Tem como objetivo padronizar diretivas de paralelização de alto nível, tornando-as mais produtivas e portáteis. Sua concepção é baseada no modelo de programação paralela de memória compartilhada para arquiteturas de múltiplos processadores (CHAPMAN; JOST; van der PAS, 2008). O OpenMP é composto por três componentes básicos:

1. Diretivas de Compilação;
2. Biblioteca de Execução;

3. Variáveis de Ambiente.

O OpenMP caracteriza-se pela sua simplicidade de implementação, pois o programador informa, através de diretivas de compilação, o “tipo” das variáveis e o trecho do código numérico a ser paralelizado. Cabe ao compilador implementar o paralelismo e à biblioteca de execução a gestão dos núcleos de processamento. Ele caracteriza-se, como já destacado, como sendo um modelo semi-implícito.

O OpenMP está disponível para uso com os compiladores Fortran, C, C++ e Python, podendo ser executado em ambientes Unix, GNU/Linux e Windows (Sistemas Multithreads) (BRUNSCHEN; BRORSSON, 2000).

3.2.2 Histórico do OpenMP

No início da década de 90, os fabricantes de multiprocessadores com memória compartilhada e similar forneciam extensões, para a linguagem de programação Fortran, com o objetivo de prover aos desenvolvedores meios de paralelizar laços em programas sequenciais (CHANDRA, 2001). Estas extensões traziam diretivas¹, que delegavam ao compilador a tarefa de paralelizar automaticamente tais laços. No entanto, havia muitas implementações divergentes para funcionalidades similares. Não existia, portanto, um padrão.

O primeiro padrão, chamado de ANSI X3H5, foi lançado em 1994 com a finalidade de estabelecer um conjunto de extensões padronizadas para auxiliar na implementação de programas paralelos em Fortran. Esse padrão foi amplamente adotado para compilar programas paralelizados, com o objetivo de serem executados em arquiteturas com memória distribuída, em função da popularidade que estas máquinas tinham à época (CHANDRA, 2001; CHAPMAN; JOST; van der PAS, 2008). Assim, foi aberto o caminho para o estabelecimento de um padrão a ser utilizado por compiladores para as arquiteturas com memória compartilhada e, em 28 de outubro de 1997, foi divulgado e disponibilizado a API OpenMP Fortran e, no final de 1998, a API OpenMP C/C++ (SILVA, 2011).

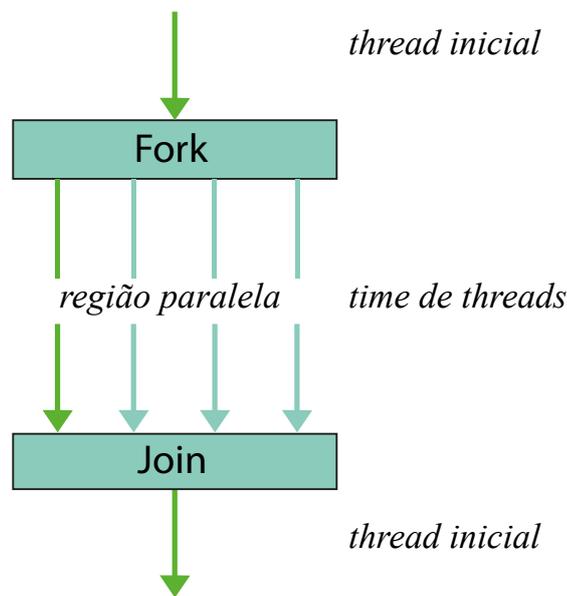
O OpenMP não é uma linguagem de programação. A técnica representa um padrão que define como os compiladores devem gerar códigos paralelizados por meio da incorporação, nos programas sequenciais, de diretivas que indicam como o trabalho será dividido entre os processadores ou *cores*. Dessa forma, muitas aplicações podem tirar proveito desse padrão a partir de pequenas alterações realizadas nos códigos pré-existentes.

¹ Consiste em uma linha de código com significado “especial” para o compilador. Por exemplo, nas linguagens C e C++ as diretivas são identificadas por `#pragma omp` (TORELLI; BRUNO, 2004), enquanto que na linguagem Fortran as diretivas são identificadas por `!$omp`.

3.2.3 Modelo de execução do OpenMP

Os programas desenvolvidos com o OpenMP utilizam um modelo de execução conhecido na literatura como *fork-join*, o qual se encontra esquematizado na Figura 6. Múltiplas unidades de execução realizam tarefas que são definidas implicitamente ou explicitamente por diretivas OpenMP (CHAPMAN; JOST; van der PAS, 2008).

Figura 6 - Modelo de programação do OpenMP



Fonte: Adaptada de CHAPMAN; JOST; van der PAS, 2008.

Um programa que utiliza a interface de programação OpenMP inicia com uma única unidade de execução, chamada *thread* inicial ou *thread* mestre, a qual executa sozinha as instruções, sequencialmente, até encontrar uma região paralelizada, identificada pela diretiva ***parallel*** (HEMSTAD, 2013). Essa diretiva é responsável por demarcar as regiões do código que serão executadas por múltiplas *threads* (regiões paralelizadas). Cada *thread* possui sua própria pilha de execução, porém, compartilha o mesmo endereço de memória com as demais *threads* do mesmo processo, enquanto que cada processo possui o seu próprio espaço de memória.

No momento em que uma *thread* encontra a diretiva ***parallel***, ela cria um grupo de *threads* (*fork*) (TORELLI; BRUNO, 2004). Para cada *thread* é delegada a execução das instruções contidas na região paralela. Ao concluir a execução em paralelo, o grupo de *threads* sincroniza-se em uma barreira² implícita à *thread* mestre. Neste momento, o

² Uma barreira ao final de uma região significa que nenhuma *thread* pode prosseguir até que todas as outras *threads* do grupo alcancem este ponto no programa (CHAPMAN; JOST; van der PAS, 2008).

grupo de *threads* termina sua execução e a *thread* mestre continua sua execução sequencial (*join*) até que seja encontrada uma nova região paralelizada ou até que o programa termine (CHAPMAN; JOST; van der PAS, 2008). Se um time de *threads*, que estão sendo executadas em uma região paralelizada, encontrar outra diretiva *parallel*, cada *thread* desse time criará um novo grupo de *threads* e a mesma se tornará também uma *thread* mestre.

A programação *multithread* com o OpenMP tem vantagens em relação a outras formas de implementação do paralelismo em arquiteturas de memória compartilhada. Dentre elas, pode-se citar (CHAPMAN; JOST; van der PAS, 2008):

1. Normalmente, são feitas poucas alterações no código serial existente (facilidade de conversão);
2. Possui uma robusta estrutura de suporte à programação paralela;
3. Fácil compreensão e uso das diretivas;
4. Suporte ao paralelismo aninhado;
5. Possibilita o ajuste dinâmico do número de *threads* empregadas na paralelização.

Uma das desvantagens do OpenMP é o fato de que não existe um controle refinado do código efetivamente implementado, o que pode ser realizado quando um programador implementa manualmente os algoritmos de paralelização. No entanto, um detalhamento ainda mais refinado é possível quando as estruturas de dados e os algoritmos são implementados, desde o início, considerando os modelos e as técnicas de processamento em paralelo e concorrente.

No que diz respeito ao trabalho aqui realizado, algumas partes do código numérico foram paralelizadas, tomando-se por base a estimativa do custo computacional das mesmas quando da execução em série. Portanto, paralelizou-se as seguintes operações ou tarefas:

1. O produto matriz-vetor (Algoritmo 2);
2. O produto escalar entre vetores (Algoritmo 3);
3. A operação de atualização de vetores: formada pela soma de dois vetores na qual uma das parcelas é uma multiplicação de um escalar por um vetor (Algoritmo 4);
4. Os laços *for* dentro da implementação do método dos Gradientes Conjugados (Algoritmo 5);
5. Na determinação das transmissibilidades (Algoritmo 6), na iteração externa de Picard;

6. No cálculo dos valores dos erros absolutos para os testes de convergência interna (Gradientes Conjugados) e externa (método de Picard) (Algoritmo 7).

Algoritmo 2: Produto Matriz-Vetor.

Entrada: forneça o número máximo de células na direção x e y (nxy) e os vetores obtidos aplicando a técnica CSR.

```

1 #pragma omp parallel for private (i,j) reduction (+:soma)
2 para  $i = 0 : nxy$  faça
3    $soma = 0;$ 
4   para  $j = Nonzerosinrow[i] : Nonzerosinrow[i + 1];$  faça
5      $soma+ = Matrixvalues[j] \times x[Mtxindj[j] + 1];$ 
6    $b[i + 1] = soma;$ 

```

Algoritmo 3: Produto Escalar.

Entrada: forneça os vetores a e b , cujo número de elementos é N .

```

1  $soma = 0;$ 
2 #pragma omp parallel for reduction (+:soma)
3 para  $i = 1 : N$  faça
4    $soma+ = a[i] \times b[i];$ 

```

Algoritmo 4: Atualização de Vetores.

Entrada: forneça os vetores b e c , cujo número de elementos é N , e o escalar d .

```

1 #pragma omp parallel for private (i)
2 para  $i = 1 : N$  faça
3    $a[i] = b[i] + d \times c[i];$ 

```

No intuito de se atualizar o fluxograma mostrado na Figura 4, do Capítulo 2, o mesmo foi feito de modo a que fossem indicadas as etapas onde ocorreram as paralelizações do código numérico, vide a Figura 7.

Algoritmo 5: Método do Gradiente Conjugado (CG).

Entrada: forneça o número máximo de iterações ($nmax$) e a tolerância (tol);

Calcule $r_0 = b - Ax_0$ para um valor inicial arbitrário x_0 ; Faça

$p_0 = r_0$.

```

1 Calcule  $\|r_0\|$ 
2 para  $k = 0 : nmax$  faça
3   Calcule  $Ap_k$                                 // #pragma omp parallel for private
                                                (i,j,sum)
4   Calcule  $\langle p_k, Ap_k \rangle$                 // #pragma omp parallel for reduction
                                                (+:pAp)
5   Calcule  $\langle r_k, r_k \rangle$                 // #pragma omp parallel for reduction
                                                (+:r_old_norm)
6    $\alpha_k = \frac{\langle r_k, r_k \rangle}{\langle p_k, Ap_k \rangle}$ 
7    $x_{k+1} = x_k + \alpha_k p_k$                 // #pragma omp parallel for
8    $r_{k+1} = r_k - \alpha_k Ap_k$             // #pragma omp parallel for
9   Calcule  $\langle r_{k+1}, r_{k+1} \rangle$         // #pragma omp parallel for reduction
                                                (+:r_new_norm)
10   $\beta_k = \frac{\langle r_{k+1}, r_{k+1} \rangle}{\langle r_k, r_k \rangle}$ 
11   $p_{k+1} = r_{k+1} + \beta_k p_k$             // #pragma omp parallel for
12  Calcule  $\|r_{k+1}\|$ 
13  se  $\|r_{k+1}\|/\|r_0\| < tol$  então
14     $x = x_{k+1}$ 
15  senão
16    Se  $k = nmax$ , não houve convergência.
```

Algoritmo 6: Cálculo da transmissibilidade.

Entrada: forneça o número máximo de células nas direções x (nx) e y (ny).

```

1 #pragma omp parallel for private (j,i,n)
2 para  $j = 1 : ny$  faça
3   para  $i = 1 : nx$  faça
4      $n = ((j - 1) \times nx) + i$ ;
5     Calcule a transmissibilidade
```

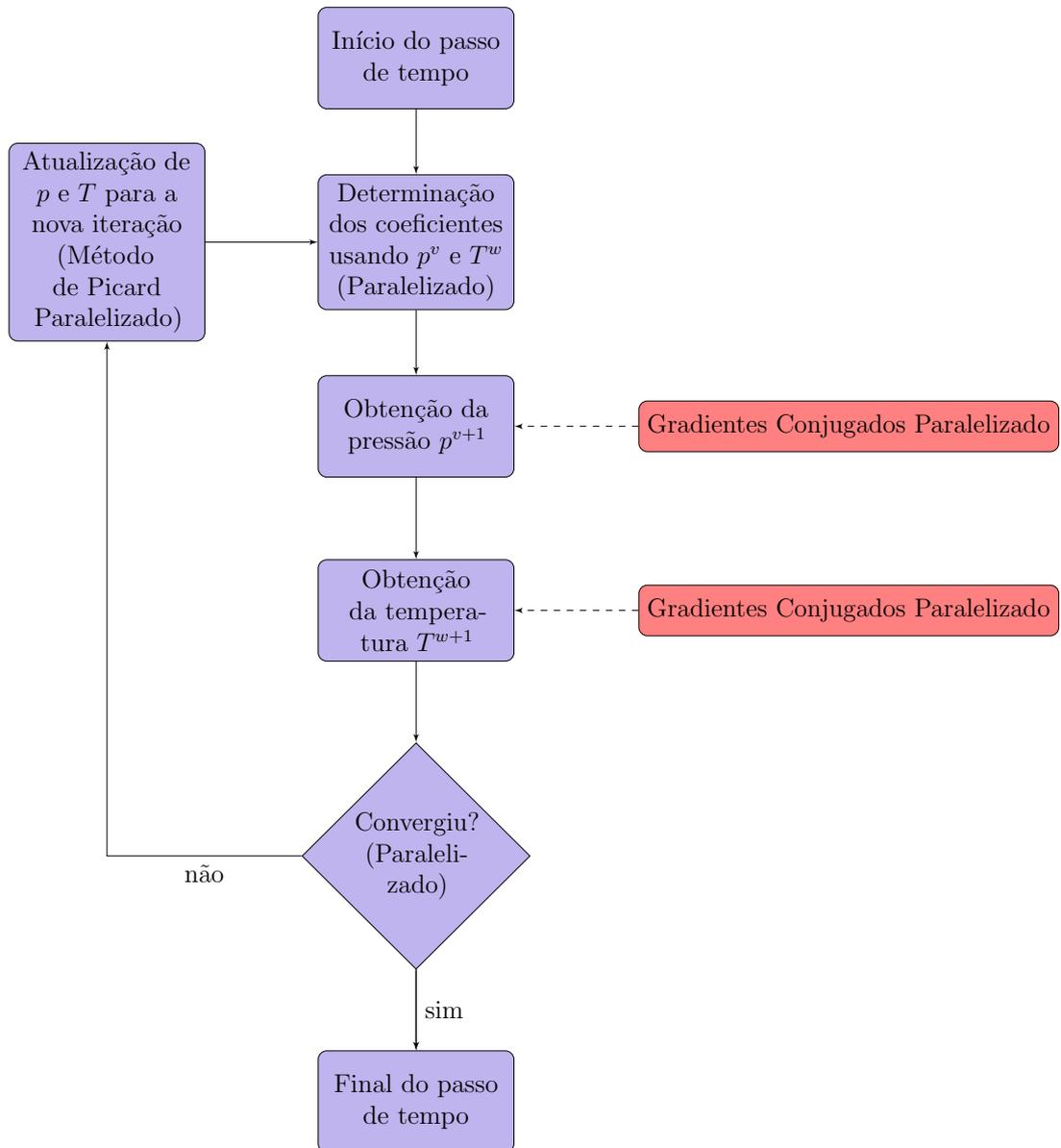
Algoritmo 7: Cálculo do erro.

Entrada: forneça o número máximo de células na direção x e y (nxy) e a tolerância (tol).

```

1 #pragma omp parallel for private (n) reduction (max:erro)
2 para  $n = 1 : nxy$  faça
3   Calcule o erro
4   se  $erro < tol$  então
5     Houve convergência.
  
```

Figura 7 - Fluxograma para um passo de tempo paralelizado



3.2.4 Coprocessador Intel Xeon Phi

A arquitetura Intel MIC (*Many Integrated Core Architecture*), comercializada sob a marca XEON Phi, era voltada especificamente para a computação de alto desempenho, que demanda a utilização de uma grande quantidade de dados quando do processamento em paralelo, em uma variedade de áreas como a Física Computacional, as Engenharias, a Química, a Biologia e a área financeira (RAHMAN, 2013). Hoje em dia, estas necessidades são supridas mediante o uso de grandes *clusters* que integram muitos computadores. Além do *hardware*, o uso do coprocessador é suportado por um ambiente de desenvolvimento que inclui diversos produtos, tais como compiladores, bibliotecas, ferramentas de *tuning* e depuradores.

A Intel lançou o código de suporte à arquitetura MIC em 2013 e continuou a aprimorá-lo por um período de tempo. Entretanto, infelizmente, o MIC/Xeon Phi foi abandonado pela companhia, em favor de seus esforços voltados para o desenvolvimento de CPUs e GPUs, e o código Intel MIC está sendo removido a partir do Linux 5.10. A justificativa para a descontinuação dos dispositivos e não manutenção dos códigos foi a de que o *hardware* nunca atingiu uma grande escala de vendas.

Nas simulações utilizadas nesta tese, o modelo de coprocessador Intel Xeon Phi utilizado foi o 7120P. Esse modelo é um *Knights Corner* do tipo *server* com litografia de 22 nm e que possui 61 núcleos físicos. Cada *core* é capaz de executar quatro *threads* simultâneas, então, isto significa que ele disponibiliza um total de 244 threads. Além disso, os processadores possuem uma frequência de 1,24 GHz, ou de 1,33 GHz em modo turbo max, e 16 GB de memória total do tipo GDDR5.

O coprocessador Intel Xeon Phi está instalado em um nó, pertencente ao LABTRAN, do *cluster* do Laboratório de Computação de Alto Desempenho (LCAD) e encontra-se ligado ao processador Intel Xeon do nó “hospedeiro” (HEBENSTREIT, 2013). Uma vez que o coprocessador Intel Xeon Phi executa uma distribuição do sistema operacional Linux, o usuário pode acessá-lo como um sistema independente. Assim, qualquer usuário pode conectar-se ao coprocessador por meio de uma interface iterativa *shell* segura e executar diretamente os códigos individuais ou submetê-los em lotes (*batch*). Fisicamente, ele é instalado em um *slot* do tipo PCI Express 225 W/300 W *High Power Card Electromechanical Specification 1.0*.

O coprocessador também suporta aplicações heterogêneas, nas quais parte do código é executada no nó hospedeiro, enquanto outra é executada no coprocessador. Vários coprocessadores Intel Xeon Phi podem ser instalados em um único nó servidor (hospedeiro). Dentro de um único sistema, os coprocessadores podem se comunicar uns com os outros através das interconexões PCI e sem qualquer intervenção do servidor. Da mesma forma, os coprocessadores também podem se comunicar através da rede local.

Pode-se dividir os modelos de programação paralela do coprocessador em dois

grupos. O primeiro grupo permite a execução em modo explícito, usando a memória virtual compartilhada e, basicamente, as diretivas do OpenMP e a *runtime offload libraries* existentes na *Manycore Platform Software Stack* (MPSS), e o modelo de programação é o *offload*. A MPSS é necessária para que seja possível as execuções empregando o coprocessador Intel Xeon Phi e ela é dependente dos *kernels* Linux 2.6.34 ou posteriores, tendo sido testada em versões específicas de sistemas operacionais de 64 *bits* para distribuições do Red Hat Enterprise, SuSE Linux Enterprise Server e Microsoft Windows. Já no segundo grupo a execução se dá no modo *standalone* ou distribuída via processos empregando a API MPI e utilizando conexões *ssh* e os protocolos TCP/IP para estabelecer a comunicação entre o *host* e os coprocessadores Intel Xeon Phi (GONÇALVES, 2018).

4 RESULTADOS

Este capítulo apresenta e discute os resultados numéricos obtidos com o código paralelizado, utilizado na simulação do escoamento monofásico não isotérmico em um reservatório contendo poços aquecedores. Vale ressaltar que o simulador inicial foi escrito em linguagem de programação C e, antes de ser paralelizado, foi testado e validado, conforme apresentado no trabalho de [Lopes, Souza e Souto \(2018\)](#).

Para tanto, na Seção 4.1 é estabelecido o conjunto básico de dados de entrada, padrão para todas as simulações, e discorre-se sobre algumas características gerais das simulações.

Em seguida, passa-se ao estudo do refinamento de malha e à verificação da convergência numérica, na Seção 4.2. Ao todo, foram empregadas 7 diferentes malhas computacionais.

Além das simulações realizadas variando-se o número de células da malha computacional, foram realizados também testes variando o número de *threads* e diferentes abordagens de paralelização de partes do simulador (Seção 4.3).

Na última seção, faz-se uma análise de sensibilidade a partir da variação de alguns dos parâmetros numéricos e físicos, a fim de se verificar a influência deles no desempenho computacional do simulador paralelizado (Seção 4.4). Também são fornecidos, para alguns casos, os tempos de execução computacional para as simulações realizadas com as versões serial (onde as diretivas do OpenMP são ignoradas) e paralelizada.

4.1 Características gerais da simulação

Para as simulações numéricas definiu-se um conjunto padrão de parâmetros extraídos, em sua maioria, do trabalho de [Heringer et al. \(2018\)](#). No que diz respeito ao problema padrão considerado, ele é composto em um arranjo composto de um poço produtor vertical, centralizado no plano xy , e de dois poços estáticos de aquecimento, ambos posicionados a uma distância de 256 m do poço produtor. A vazão de produção é mantida constante (salvo menção em contrário) e os demais parâmetros utilizados no caso base (padrão) são apresentados na Tabela 1.

Os resultados são apresentados através de gráficos e tabelas, sendo que o *speedup* é calculado como sendo a razão entre o tempo de simulação empregando-se as versões serial e paralelizada.

A análise dos resultados é favorecida em função das condições de simetria do problema, decorrentes da distribuição dos aquecedores no reservatório, das características da formação rochosa, da geometria e dos espaçamentos ($\Delta x = \Delta y$) da malha computacional

nas duas direções espaciais.

Tabela 1 - Parâmetros para o caso padrão

Parâmetro	Valor	Unidade
a	$0,6 \times 10^{-4}$	Pa·s
b	300	K
μ_o	0,0146	Pa·s
B_0	1	m ³ /std m ³
c_o	$7,25 \times 10^{-7}$	kPa ⁻¹
c_p	2100	J/(kg K)
$c_{p\phi}$	1.200	J/(kg K)
c_{T_o}	$9,2 \times 10^{-4}$	K ⁻¹
c_v	1.800	J/(kg K)
c_ϕ	5×10^{-7}	kPa ⁻¹
$c_{\phi T}$	$1,8 \times 10^{-3}$	K ⁻¹
$F_{\Delta t}$	1,1	–
$k_x = k_y = k$	$0,025 \times 10^{-12}$	m ²
$L_x = L_y$	6.400	m
L_z	50	m
L_{wf}	50	m
$n_x = n_y$	769	–
$p_{inic} = p_0$	69.000	kPa
q_{sc}	-100	std m ³ /dia
Q_H	2.000	W
t_{max}	1.000	dia
tol_p	1×10^{-6}	kPa
tol_T	1×10^{-6}	K
$T_i = T_0$	330	K
T_{visc}	500	K
κ_o	0,45	W/(m K)
κ_r	3,5	W/(m K)
ρ_r	2.500	kg/m ³
ρ_{sc}	2.500	kg/m ³
Δt_i	0,1	dia
$\Delta t_{f_{max}}$	1	dia
$\phi_i = \phi_0$	0,25	–

Fonte: O autor, 2021

Nas simulações de reservatórios, muitas vezes utiliza-se um passo de tempo pequeno para uma melhor captura e visualização dos efeitos que se desenrolam nos instantes iniciais da produção. Entretanto, o incremento de tempo aumenta de valor ao longo da simulação, atingido um valor limite pré-estabelecido que é usado, então, no restante dos cálculos. Neste trabalho, optou-se por iniciar as simulações com um passo de tempo inicial, Δt_i , com uma razão de crescimento $F_{\Delta t}$, tal que para os passos consecutivos $\Delta t^{n+1} = F_{\Delta t} \Delta t^n$, até atingir-se o seu valor máximo, $\Delta t_{f_{max}}$. A simulação transcorre até que o tempo máximo de produção, t_{max} , seja atingido. Além disso, destaca-se que o poço vertical tem comprimento aberto ao fluxo dado por L_{wf} , igual à espessura da formação produtora, L_z .

4.2 Refinamento de malha

A Tabela 2 mostra o número de células (n_x e n_y) das várias malhas computacionais empregadas no estudo do refinamento da malha. O número de células na direção z (n_z) é mantido constante e igual a 1 para todas as simulações, uma vez que estudou-se somente o escoamento bidimensional.

Tabela 2 - Refinamento de malha

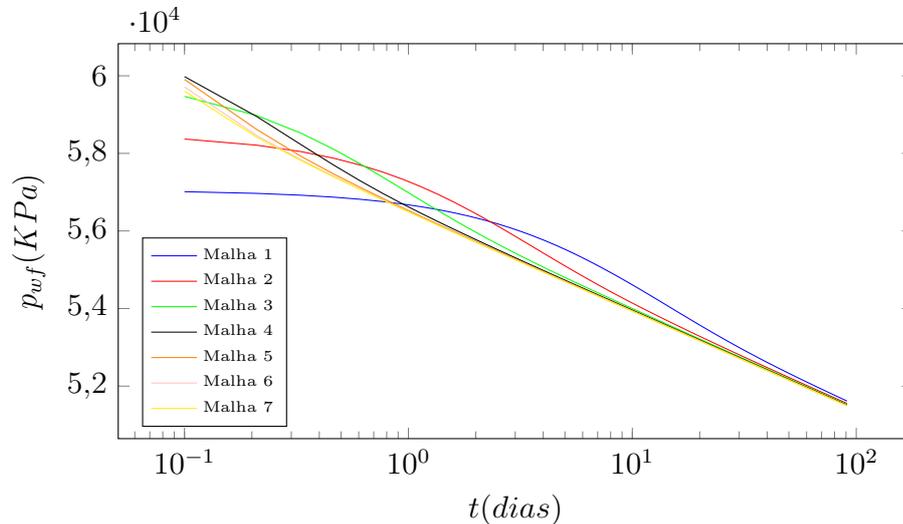
Malha	n_x	n_y
0	25	25
1	49	49
2	97	97
3	193	193
4	385	385
5	769	769
6	1537	1537
7	3073	3073

Fonte: O autor, 2021

A Figura 8 traz os resultados, para a pressão no poço produtor, quando do refinamento da malha computacional empregando os sete conjuntos vistos na Tabela 2. Todas as simulações foram efetuadas com a versão serial do código numérico. É possível observar, para os instantes de tempo iniciais, a existência de um patamar no trecho inicial correspondente das curvas de pressão. A sua existência é devida ao artefato numérico, associado à escolha da técnica de acoplamento poço-reservatório adotada (PEACEMAN, 1978), que assume que o escoamento se dá em regime permanente nas proximidades do poço. Entretanto, esse efeito diminui com o refinamento da malha computacional (SOUZA, 2013), como pode ser constatado no gráfico apresentado. O efeito do artefato termina quando surge a linha decrescente, característica do regime transiente em meio poroso. Na

literatura, pode-se encontrar algumas técnicas de mitigação deste efeito, porém, a sua eliminação não se encontra no escopo deste trabalho.

Figura 8 - Perfis de pressão no poço para diferentes malhas

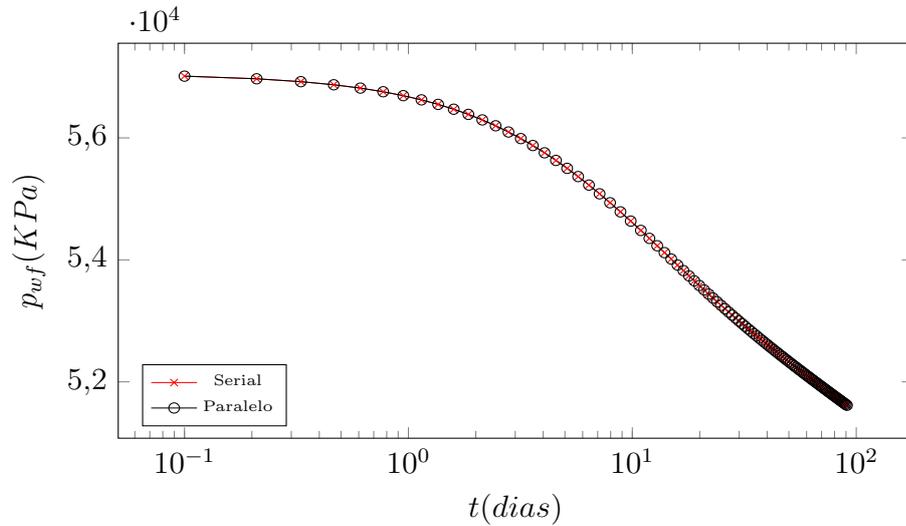


Fonte: O autor, 2021.

O estudo do refinamento de malha mostrou que as curvas de pressão se aproximaram cada vez uma das outras à medida que as malhas eram refinadas, indicando que o método numérico é convergente. Em função da constatação da convergência numérica e da não existência do artefato numérico, para instantes de tempo menores do que 1 dia de produção, a Malha 5 foi considerada ser suficientemente refinada para fornecer resultados acurados e foi adotada como sendo a padrão nas simulações.

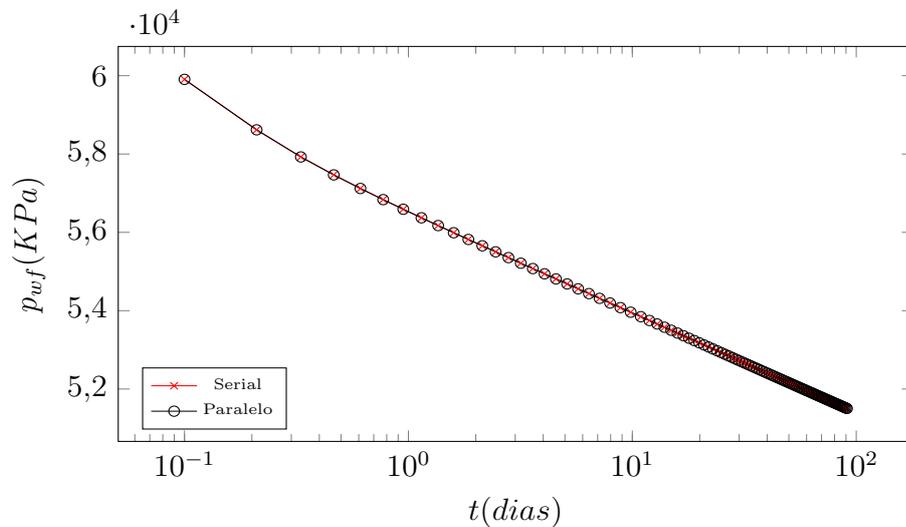
As Figuras 9 e 10 trazem os resultados do estudo comparativo entre as curvas de pressão obtidas, com a Malha 5, via a execução do simulador nas suas versões serial e paralelizada. No último caso, utilizou-se o coprocessador e estipulou-se o uso de um total de 240 *threads*. Apesar de não terem sido mostrados aqui, o mesmo tipo de verificação foi realizada considerando outros valores para o número de *threads*. Assim sendo, em decorrência da sobreposição das curvas tanto no caso da Malha 1 (menos refinada) e Malha 5 (mais refinada), nas figuras mostradas, conclui-se que a paralelização do código numérico não comprometeu a integridade dos resultados numéricos obtidos com ambas as versões.

Figura 9 - Perfis de pressão no poço - Malha 1 e versões serial e paralelizada



Fonte: O autor, 2021.

Figura 10 - Perfis de pressão no poço - Malha 5 e versões serial e paralelizada



Fonte: O autor, 2021.

4.3 Eficiência computacional

Para se analisar o desempenho computacional, do código paralelizado, utilizou-se como parâmetro de medida o *speedup*, S , que representa a razão entre os tempos de uma simulação obtidos com as execuções em série e em paralelo. Trata-se do parâmetro que é usualmente utilizado nestes casos e, com ele, é possível se estudar, por exemplo, o desempenho computacional em função da variação do número de *threads* utilizadas nas

execuções, para uma determinada malha, assim como em função da variação do número de células da malha computacional e do número de *threads*. Portanto, conforme apresentado na Tabela 2, foram empregadas oito diferentes malhas e, em cada tabela, apresenta-se os tempos de execução (em segundos) tanto para as simulações realizadas com a versão serial como com a paralelizada em função da variação de número estipulado de *threads*. Apesar do coprocessador possuir um número total de *threads* igual a 244, decidiu-se fixar os valores como variando de 20 até 240 com um intervalo de vinte.

Baseando-se nos resultados estipulados, anteriormente, para o caso base, são fornecidas nas Tabelas 3, 4, 5 e 6 os valores do tempo de execução e do *speedup* para as oito malhas distintas aqui examinadas. Nelas, o desempenho computacional pode ser aferido por meio das variações do *speedup* (aumento) em função do refinamento da malha e do acréscimo do número de *threads* empregados quando da execução das versões do código.

Tabela 3 - Tempo de execução e *speedup* para as Malhas 0 e 1

<i>Threads</i>	Malha 0		Malha 1	
	Tempo (s)	<i>S</i>	Tempo (s)	<i>S</i>
Serial	1	-	4	-
20	3	0,333	3	1,333
40	2	0,500	3	1,333
60	3	0,333	3	1,333
80	2	0,500	3	1,333
100	3	0,333	4	1,000
120	2	0,500	3	1,333
140	3	0,333	4	1,000
160	2	0,500	4	1,000
180	3	0,333	3	1,333
200	3	0,333	4	1,000
220	2	0,500	3	1,333
240	2	0,500	3	1,333

Fonte: O autor, 2021

Tabela 4 - Tempo de execução e *speedup* para as Malhas 2 e 3

<i>Threads</i>	Malha 2		Malha 3	
	Tempo (s)	<i>S</i>	Tempo (s)	<i>S</i>
Serial	14	-	67	-
20	6	2,333	15	4,467
40	6	2,333	15	4,467
60	5	2,800	15	4,467
80	6	2,333	14	4,786
100	5	2,800	14	4,786
120	5	2,800	14	4,786
140	6	2,333	15	4,467
160	5	2,800	15	4,467
180	6	2,333	14	4,786
200	6	2,333	15	4,467
220	6	2,333	15	4,467
240	6	2,333	15	4,467

Fonte: O autor, 2021

Tabela 5 - Tempo de execução e *speedup* para as Malhas 4 e 5

<i>Threads</i>	Malha 4		Malha 5	
	Tempo (s)	<i>S</i>	Tempo (s)	<i>S</i>
Serial	401	-	2528	-
20	66	6,076	312	8,103
40	53	7,566	241	10,490
60	51	7,863	210	12,038
80	52	7,712	216	11,704
100	50	8,020	206	12,272
120	51	7,863	204	12,392
140	52	7,712	214	11,813
160	51	7,863	205	12,332
180	50	8,020	207	12,213
200	53	7,566	214	11,813
220	53	7,566	210	12,038
240	53	7,566	207	12,213

Fonte: O autor, 2021

Tabela 6 - Tempo de execução e *speedup* para as Malhas 6 e 7

<i>Threads</i>	Malha 6		Malha 7	
	Tempo (s)	<i>S</i>	Tempo (s)	<i>S</i>
Serial	17253	-	117630	-
20	1705	10,119	9673	12,161
40	1170	14,746	6128	19,195
60	998	17,288	4929	23,865
80	1034	16,686	5076	23,174
100	992	17,392	4607	25,533
120	955	18,066	4339	27,110
140	970	17,787	4684	25,113
160	937	18,413	4315	27,261
180	959	17,991	4273	27,529
200	976	17,677	4256	27,639
220	963	17,916	4097	28,711
240	953	18,104	4030	29,189

Fonte: O autor, 2021

Claro está que o aumento do número de células das malhas também acarreta o acréscimo do número de operações a serem realizadas, pois em consequência também cresce a ordem do sistema de equações algébricas a ser resolvido. Então, espera-se que haja uma subsequente amplificação no ganho do desempenho computacional quando utiliza-se a computação em paralelo. Deve-se levar em conta, como já deixado claro, que nem todas as partes do código numérico do simulador foram paralelizadas, de forma que ainda existem sequências de instruções no código que não sofrem nenhuma influência da paralelização e, assim sendo, o esforço computacional despendido nelas não será alterado. Tal fato pode ajudar a explicar a não linearidade do comportamento da variação do *speedup* em função do número de *threads*.

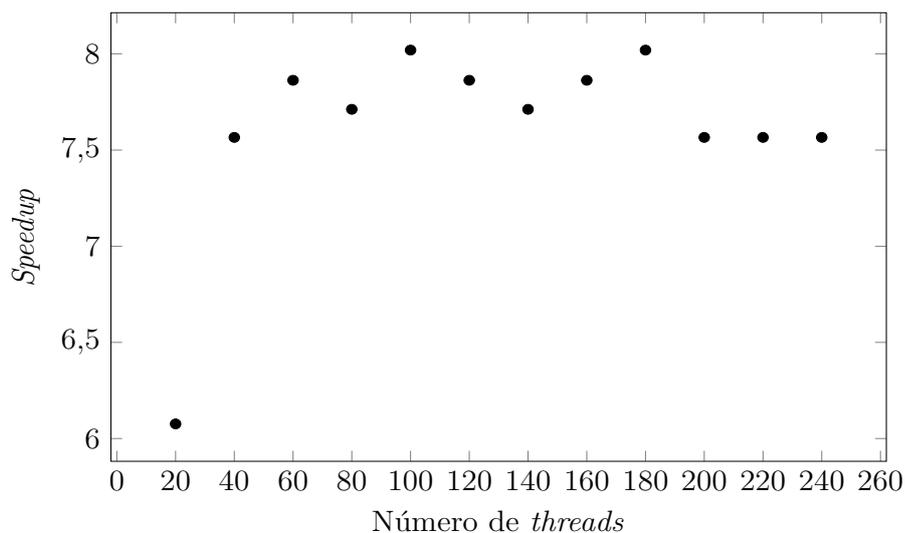
Para as malhas menos refinadas (Malhas 0, 1 e 2), o valor do *speedup* não foi significativo, pois para os melhores resultados eles ficaram abaixo de 3, conforme pode ser observado nas Tabelas 3 e 4. Essa baixa performance pode ser explicada tendo em vista que, para as malhas mais grossas, o esforço computacional para se inicializar os ambientes paralelos no OpenMP não foi compensado, tendo em vista os tempos curtos de simulação. Todavia, conforme aumenta-se o número de células esse esforço será retribuído com os ganhos adicionais provenientes da paralelização. No entanto, vale ressaltar que isso não significa que o desempenho sempre aumentará proporcionalmente ao aumento do número de *threads*, mesmo levando em conta as malhas mais refinadas.

Por outro lado, é possível visualizar nas Figuras 11, 12, 13 e 14 os gráficos que

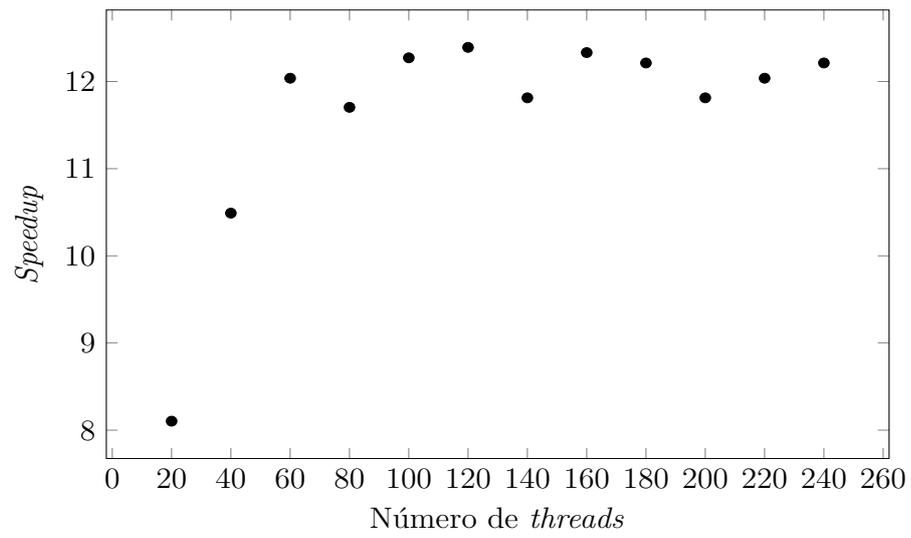
mostram a modificação dos valores do *speedup* em função da variação do número de *threads*, para os quatro casos correspondentes aos maiores números de células, ou seja, às malhas mais refinadas aqui estudadas: Malhas 4, 5, 6 e 7. Examinando-se os resultados contidos nelas verifica-se que o ganho em desempenho computacional não é linear e que, de acordo com o acréscimo do número de *threads*, o *speedup* aumenta até certo ponto, seguindo de uma queda/estagnação para as quatro malhas. Alcança-se essa estagnação mais rapidamente quando as simulações são efetuadas com as malhas menos refinadas. Segundo a literatura (CHAPMAN; JOST; van der PAS, 2008), uma possível explicação para essa estagnação pode ser decorrente do uso de *threads* virtuais, que possuem menor desempenho do que os *cores* físicos. Entretanto, não se está afirmando que essa seja a causa, uma vez que o mesmo não foi verificado em relação à Malha 7 no que diz respeito ao limite de ganho. Para essa malha atingiu-se o maior valor do *speedup* dentre todos os casos testes simulados e ele é igual a 29,189.

Portanto, deve-se ressaltar que a tendência geral, verificada nas demais malhas, não se reproduziu no caso do desempenho relacionado à Malha 7. Nesse caso, constata-se que o tempo de execução vai diminuindo à medida que aumenta-se a quantidade de *threads* utilizadas nas simulações. Provavelmente, quando do uso desta malha, a dinâmica envolvendo os números de células, de *threads* e de *threads* virtuais tenha viabilizado se atingir maiores valores para o *speedup*. Uma vez mais, trata-se de hipóteses aventadas e não pode-se fazer nenhuma afirmação. Estudos mais aprofundados seriam necessários para que tal comportamento fosse elucidado.

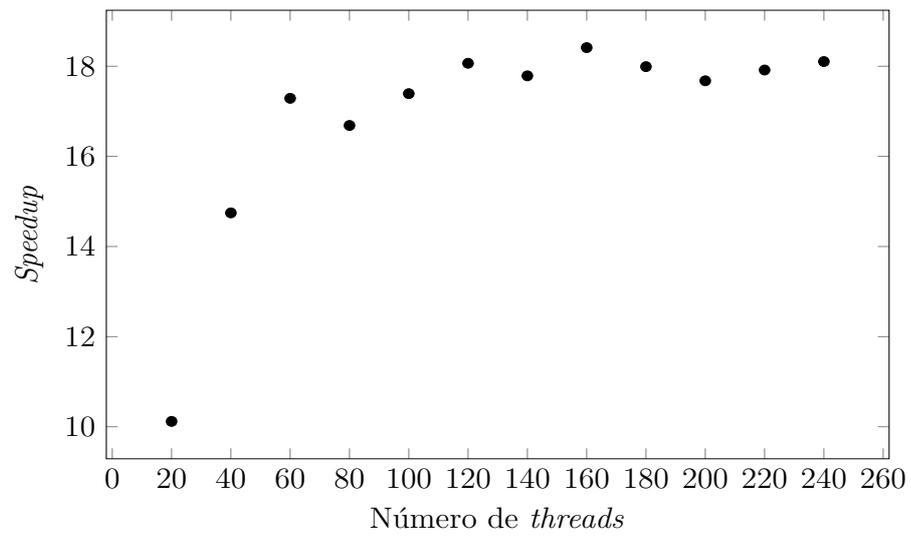
Figura 11 - *Speedup* pelo número de *threads* para a Malha 4



Fonte: O autor, 2021.

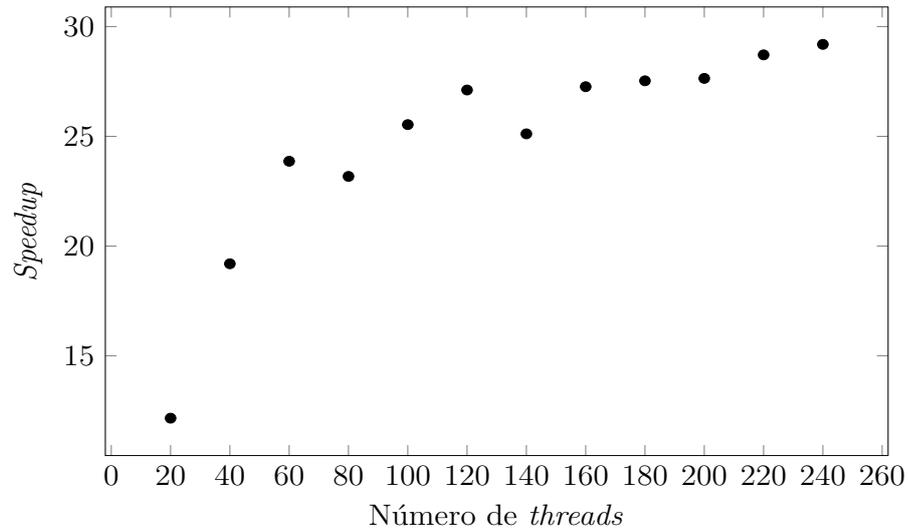
Figura 12 - *Speedup* pelo número de *threads* para a Malha 5

Fonte: O autor, 2021.

Figura 13 - *Speedup* pelo número de *threads* para a Malha 6

Fonte: O autor, 2021.

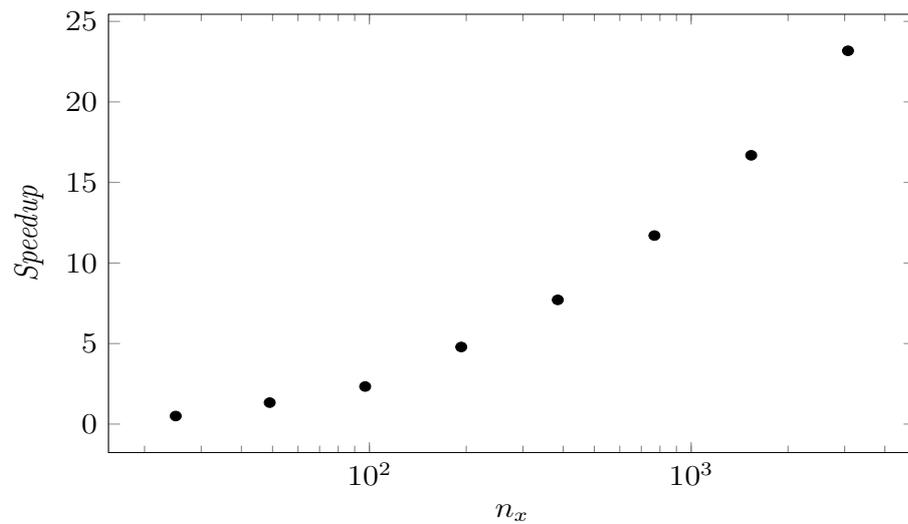
Figura 14 - *Speedup* pelo número de *threads* para a Malha 7



Fonte: O autor, 2021.

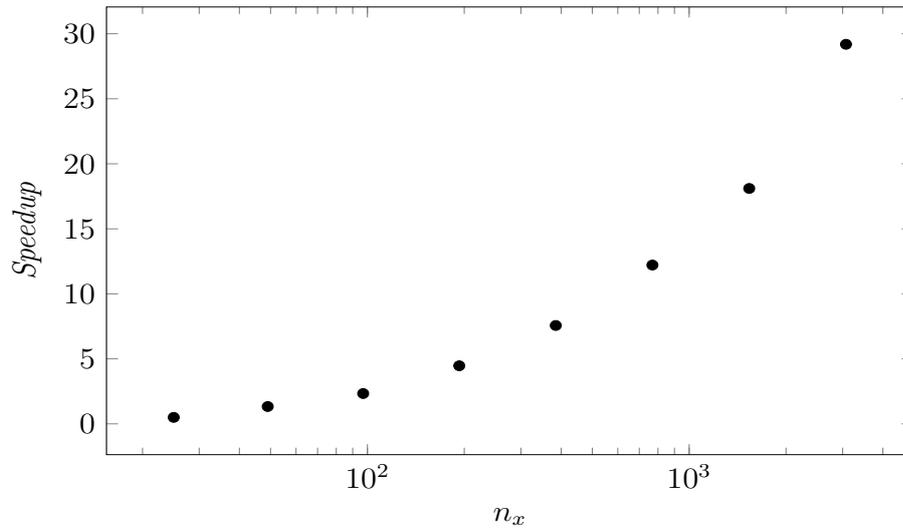
Dando continuidade, as Figuras 15 e 16 trazem agora os resultados para os ganhos em aceleração, em gráficos do tipo semi-log, para valores fixos do número de *threads*, em função do número de células contidas nas malhas computacionais. Na primeira figura, tem-se o caso dos valores obtidos com 80 *threads*, enquanto que na segunda usou-se 240 *threads*. É importante destacar que para os dois casos em questão o comportamento das curvas são semelhantes, ou seja, existe um crescimento contínuo do *speedup* conforme o número de células das malhas cresce.

Figura 15 - Análise do *speedup* para um número fixo de 80 *threads*



Fonte: O autor, 2021.

Figura 16 - Análise do *speedup* para um número fixo de 240 *threads*



Fonte: O autor, 2021.

Uma consequência relevante que pode-se tirar, visto os comportamentos observados, é o de que talvez seja interessante determinar para qual faixa do número de *threads* os valores do *speedup* começam a estabilizar, para a malha que pretende-se usar nas simulações, e escolher um determinado valor e mantê-lo fixo, na prática, em todas as simulações. Por exemplo, ter-se-ia obtido acelerações praticamente equivalentes para quase todas as malhas empregando-se 120 ou 240 *threads*, à exceção da Malha 7: 27,11 contra 29,189.

Para finalizar esta seção, compara-se o desempenho computacional de algumas versões do simulador, onde diferentes trechos do código numérico foram paralelizadas:

1. Código 1: apenas o produto matriz vetor no Método dos Gradientes Conjugados foi paralelizado;
2. Código 2: outras partes do código do Método Método dos Gradientes Conjugados foram paralelizadas, conforme indicado no Algoritmo 5;
3. Código 3: além da paralelização do Código 2, incluiu-se também novas regiões paralelizadas: atualização de vetores (Algoritmo 4), cálculo das transmissibilidades (Algoritmo 6) e cálculo dos erros absolutos (Algoritmo 7).

sabendo-se que o padrão para o simulador, neste trabalho, é o Código 3.

Então, como pode ser observado nas Tabelas 7 e 8, os novos trechos paralelizados resultaram em maiores *speedups* do que os obtidos por [Werneck \(2016\)](#) e [Werneck et al. \(2019\)](#) com o Código 2. Em relação aos ganhos de aceleração, obtidos com os três diferentes códigos, nota-se que para o Código 3 houve um aumento superior a 110% em relação ao Código 1 e de quase 25% em relação ao Código 2 utilizando-se a Malha 5. No que diz respeito à Malha 6, esses aumentos foram de 160% e 20%, respectivamente.

Tabela 7 - Tempo de execução e *speedup* para a Malha 5: diferentes códigos

<i>Threads</i>	Código 1		Código 2		Código 3	
	Tempo (s)	<i>S</i>	Tempo (s)	<i>S</i>	Tempo (s)	<i>S</i>
Serial	2528	-	2528	-	2528	-
20	523	4,834	364	6,945	312	8,103
40	460	5,496	288	8,778	241	10,490
60	434	5,825	260	9,723	210	12,038
80	444	5,694	269	9,398	216	11,704
100	445	5,681	260	9,723	206	12,272
120	442	5,719	255	9,914	204	12,392
140	457	5,532	269	9,398	214	11,813
160	442	5,719	263	9,612	205	12,332
180	440	5,745	261	9,686	207	12,213
200	457	5,532	273	9,260	214	11,813
220	460	5,496	268	9,433	210	12,038
240	459	5,508	267	9,468	207	12,213

Fonte: O autor, 2021

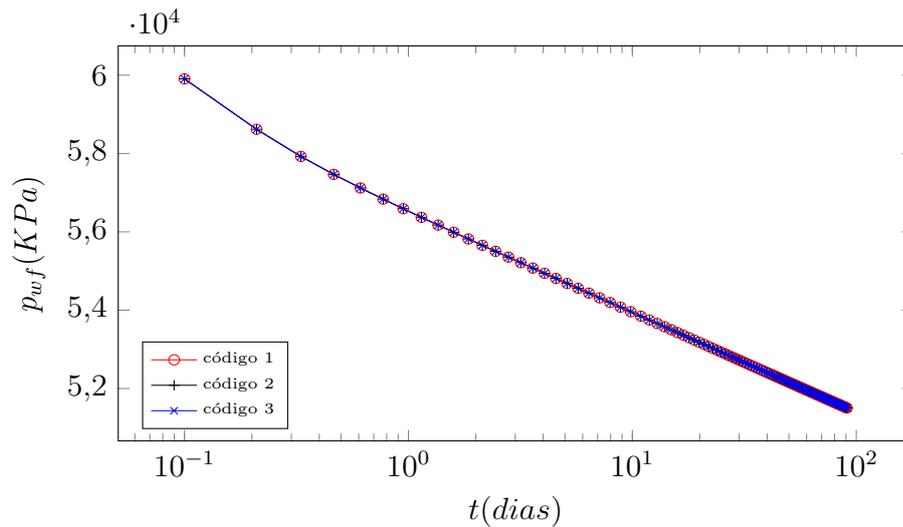
Tabela 8 - Tempo de execução e *speedup* para a Malha 6: diferentes códigos

<i>Threads</i>	Código 1		Código 2		Código 3	
	Tempo (s)	<i>S</i>	Tempo (s)	<i>S</i>	Tempo (s)	<i>S</i>
Serial	17253	-	17253	-	17253	-
20	3127	5,517	1891	9,124	1705	10,119
40	2697	6,397	1383	12,475	1170	14,746
60	2517	6,855	1208	14,282	998	17,288
80	2557	6,747	1233	13,993	1034	16,686
100	2524	6,836	1174	14,696	992	17,392
120	2513	6,865	1158	14,899	955	18,066
140	2507	6,882	1171	14,734	970	17,787
160	2497	6,909	1147	15,042	937	18,413
180	2480	6,957	1133	15,228	959	17,991
200	2509	6,876	1159	14,886	976	17,677
220	2505	6,887	1147	15,042	963	17,916
240	2493	6,921	1127	15,309	953	18,104

Fonte: O autor, 2021

Esta seção é encerrada com um comparativo entre as curvas de pressão no poço produtor, Figura17, determinadas a partir da Malha 5 e das três versões dos códigos paralelizados: Códigos 1, 2 e 3. Verifica-se, então, que as curvas encontram-se sobrepostas, indicando os códigos paralelizados diferentemente não comprometeram a integridade dos resultados.

Figura 17 - Variação da pressão: Códigos 1, 2 e 3



Fonte: O autor, 2021.

4.4 Análise de sensibilidade

Uma análise de sensibilidade foi empreendida com intuito de verificar a influência que teria a modificação de alguns dos parâmetros de simulação no tempo de execução e no *speedup* e, também, no comportamento da pressão no poço produtor. Assim sendo, variou-se os valores da permeabilidade, da viscosidade, da taxa de aquecimento, da porosidade, do número de poços aquecedores e da vazão. Para esse estudo, apenas a Malha 5 ($n_x = n_y = 769$) foi utilizada.

4.4.1 Permeabilidade

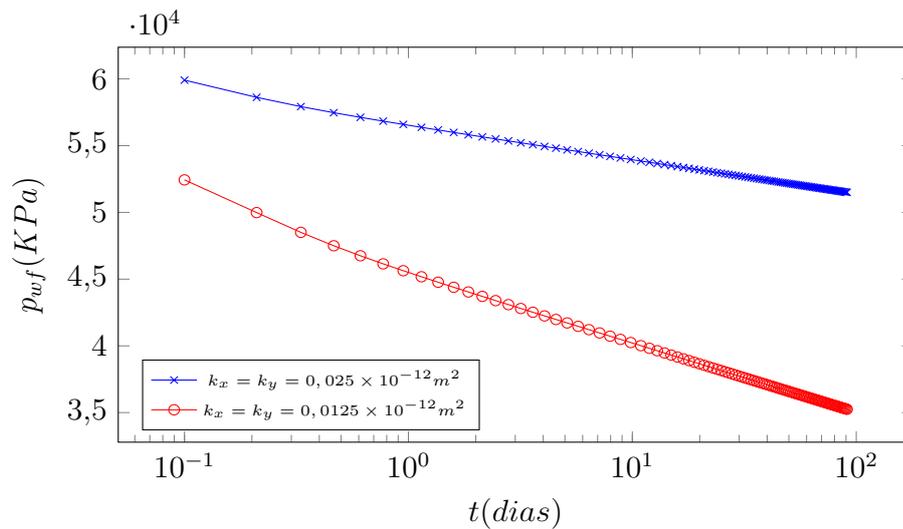
O primeiro estudo abordou a permeabilidade, onde foram feitas simulações utilizando os valores $k_x = k_y = 0,0125 \times 10^{-12} \text{ m}^2$ e o valor do caso base, $k_x = k_y = 0,025 \times 10^{-12} \text{ m}^2$. Os resultados são apresentados na Tabela 9, desempenho computacional, e na Figura 18 para a variação de pressão no poço produtor.

Tabela 9 - Análise de sensibilidade para a permeabilidade

Threads	Permeabilidade (m^2)			
	$k = 0,0125 \times 10^{-12}$		$k = 0,025 \times 10^{-12}$	
	Tempo (s)	S	Tempo (s)	S
Serial	2091	-	2528	-
20	286	7,311	312	8,103
40	221	9,462	241	10,490
60	198	10,561	210	12,038
80	201	10,403	216	11,704
100	197	10,614	206	12,272
120	190	11,005	204	12,392
140	203	10,300	214	11,813
160	197	10,614	205	12,332
180	195	10,723	207	12,213
200	205	10,200	214	11,813
220	203	10,300	210	12,038
240	202	10,351	207	12,213

Fonte: O autor, 2021

Figura 18 - Análise de sensibilidade para a permeabilidade



Fonte: O autor, 2021.

O valor máximo do *speedup* obtido nesta análise foi de 12,392 para um número de *threads* igual a 120 e um correspondente tempo de execução de 204 segundos. Por outro lado, o valor mínimo foi de 7,311, com 20 *threads* e um tempo de execução de 286

segundos, conforme pode ser observado na Tabela 9.

Já na Figura 18, percebe-se que uma maior queda de pressão no poço produtor pode ser observada para um menor valor de permeabilidade, e esta tendência está de acordo com o esperado para esta condição (WERNECK et al., 2019)

4.4.2 Viscosidade

A Tabela 10 e a Figura 19 trazem os resultados obtidos considerando-se a variação de a na correlação para a viscosidade (1). Desta forma, foca-se agora na mudança da viscosidade do óleo que se encontra no reservatório. Esta comparação é muito relevante, uma vez que é justamente a redução da viscosidade do óleo o principal objetivo da técnica de aquecimento, levando em consequência a uma menor queda de pressão no reservatório. Da Eq. (1) vê-se que os maiores valores de a implicam os maiores valores das viscosidades e, conseqüentemente, as maiores quedas de pressão no poço produtor (Figura 19). Assim como a permeabilidade, a viscosidade tem um impacto significativo na dinâmica do escoamento em meios porosos, vide a lei de Darcy (3).

Tabela 10 - Análise de sensibilidade para a viscosidade

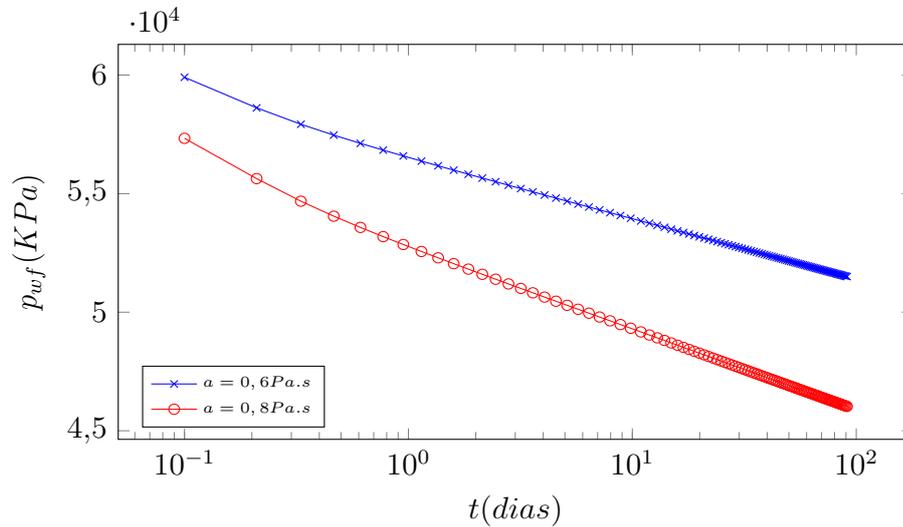
<i>Threads</i>	Parâmetro a (Pa·s)			
	$a = 0,6$		$a = 0,8$	
	Tempo (s)	S	Tempo (s)	S
Serial	2528	-	2334	-
20	312	8,103	304	7,678
40	241	10,490	229	10,192
60	210	12,038	212	11,009
80	216	11,704	209	11,167
100	206	12,272	203	11,498
120	204	12,392	201	11,612
140	214	11,813	220	10,609
160	205	12,332	202	11,554
180	207	12,213	199	11,729
200	214	11,813	211	11,062
220	210	12,038	209	11,167
240	207	12,213	204	11,441

Fonte: O autor, 2021

Na Tabela 10 nota-se que o máximo valor do *speedup* atingido neste caso teste foi

de 12,392 para o uso de 120 *threads*, um tempo de execução de 204 segundos e para para $a=0,6$ Pa.s (caso base). Em contrapartida, alcançou-se um mínimo de 7,678 empregando 20 *threads*, com uma simulação consumindo 286 segundos de execução e para $a=0,8$ Pa.s.

Figura 19 - Análise de sensibilidade para a viscosidade



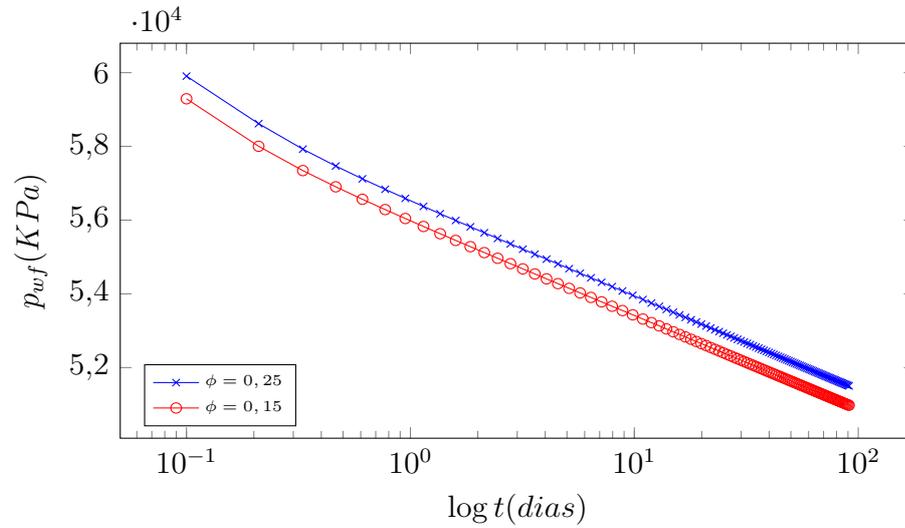
Fonte: O autor, 2021.

4.4.3 Porosidade

Estendeu-se, também, a análise de sensibilidade ao caso da variação da porosidade. Assim como a permeabilidade, esta é uma propriedade que encontra-se presente nas duas equações diferenciais parciais governantes escritas em termos da pressão, Eq. (9), e da temperatura, Eq. (11). A Figura 20 contém os resultados referentes ao caso tratado. De acordo com o que observa-se na Figura 20, o efeito de uma menor porosidade se traduz em uma maior queda de pressão, como esperado devido à menor quantidade de fluido disponível para ser produzida.

Agora, na Tabela 11 é possível atestar que para uma porosidade de 0,15 obtém-se um *speedup* máximo de 14,605 usando 180 *threads* e, nesse caso, o tempo de execução foi igual a 190 segundos. Já no que se refere ao valor mínimo, 8,103 com 20 *threads*, ele foi determinado para $\phi = 0,15$ (caso padrão) e despendeu um tempo de execução de 312 segundos.

Figura 20 - Análise de sensibilidade para a porosidade



Fonte: O autor, 2021.

Tabela 11 - Análise de sensibilidade para a porosidade

Threads	Porosidade inicial			
	$\phi = 0,15$		$\phi = 0,25$	
	Tempo (s)	S	Tempo (s)	S
Serial	2775	-	2528	-
20	316	8,782	312	8,103
40	229	12,118	241	10,490
60	205	13,537	210	12,038
80	207	13,406	216	11,704
100	196	14,158	206	12,272
120	194	14,304	204	12,392
140	205	13,537	214	11,813
160	198	14,015	205	12,332
180	190	14,605	207	12,213
200	203	13,670	214	11,813
220	198	14,015	210	12,038
240	197	14,086	207	12,213

Fonte: O autor, 2021

4.4.4 Taxa de aquecimento

A Tabela 12 e a Figura 21 trazem os resultados obtidos considerando-se a variação da taxa de aquecimento, decorrente da mudança da potência liberada nos poços aquecedores. Como já descrito anteriormente, do ponto de vista físico, à medida que aumenta-se a taxa de aquecimento reduz-se a viscosidade do óleo e, por conseguinte, preserva-se a energia disponível no reservatório passível de fazer o fluido escoar, levando a uma maior manutenção da pressão.

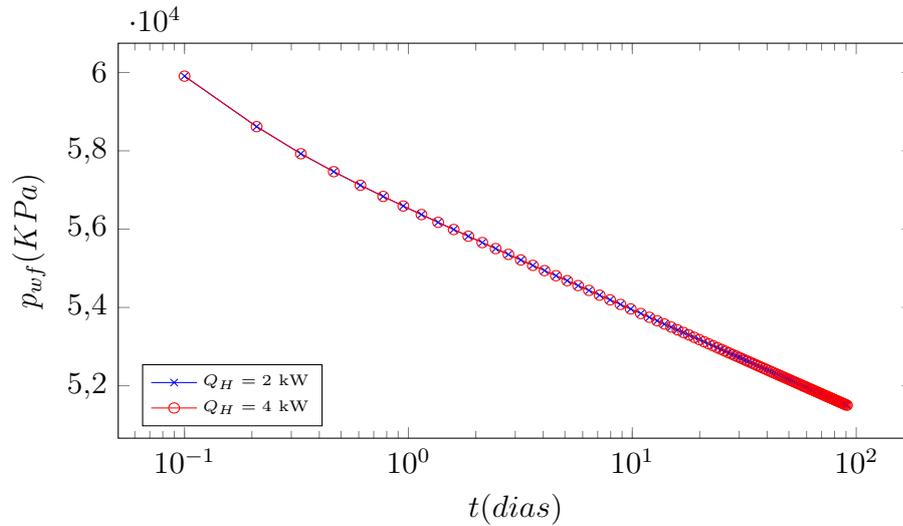
Tabela 12 - Análise de sensibilidade para a taxa de aquecimento

<i>Threads</i>	Aquecimento (kW)			
	$Q_H = 2$		$Q_H = 4$	
	Tempo (s)	S	Tempo (s)	S
Serial	2528	-	2657	-
20	312	8,103	333	7,979
40	241	10,490	246	10,801
60	210	12,038	214	12,416
80	216	11,704	221	12,023
100	206	12,272	217	12,244
120	204	12,392	204	13,025
140	214	11,813	214	12,416
160	205	12,332	212	12,533
180	207	12,213	205	12,961
200	214	11,813	219	12,132
220	210	12,038	222	11,968
240	207	12,213	213	12,474

Fonte: O autor, 2021

Na Tabela 12 verifica-se que o melhor desempenho obtido foi para $Q_H=4$ kW, com 180 *threads*, um tempo de execução de 205 segundos e um *speedup* de 12,961. Agora, para $Q_H=4$ kW e 20 *threads*, alcançou-se somente uma aceleração de 7,979 para um correspondente tempo de execução de 333 segundos.

Figura 21 - Análise de sensibilidade para a taxa de aquecimento



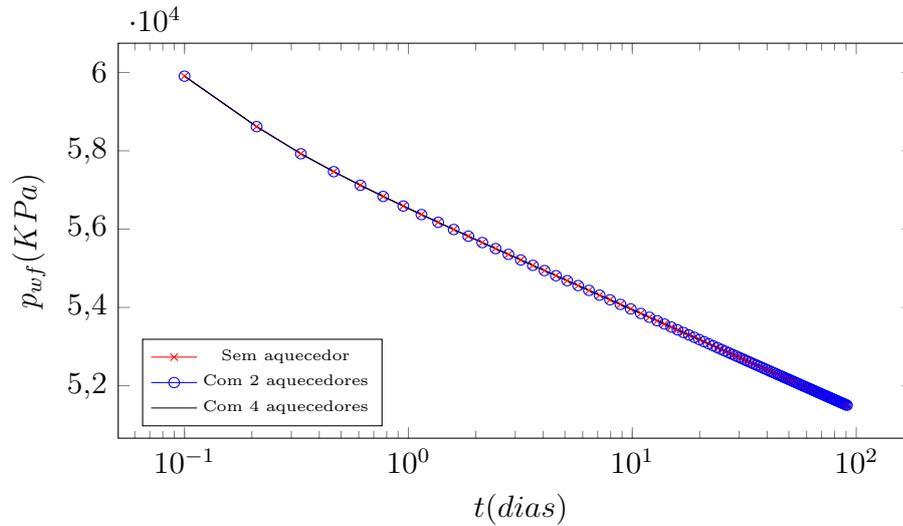
Fonte: O autor, 2021.

Apesar de na Figura 21 as curvas de pressão serem coincidentes, existe uma pequena diferença entre os valores da pressão do poço calculadas para as duas taxas de aquecimento. Porém, para que essa diferença fosse acentuada, seria necessário se realizar mais testes variando outros parâmetros, como a distância entre o poço aquecedor e o poço injetor e, até mesmo, os valores da taxa de aquecimento.

4.4.5 Número de aquecedores

Na Figura 22 e na Tabela 13, são apresentados os resultados para a variação da pressão no poço em função do número de aquecedores. Assim como no caso do aumento da taxa de aquecimento, uma maior quantidade de energia disponível levaria à preservação da pressão no reservatório. Destaca-se, porém, em função dos valores utilizados nas simulações, que as diferenças entre os valores são muito pequenas e não podem ser observadas na figura. Assim como no caso anterior, um conjunto com outros valores para os parâmetros de simulação deveriam ser pensados, de maneira a tornar as curvas de pressão distintas no gráfico da Figura 22.

Figura 22 - Análise de sensibilidade para 0, 2 e 4 aquecedores



Fonte: O autor, 2021.

Tabela 13 - Análise de sensibilidade para o número de aquecedores

Threads	Aquecedores					
	n=0		n=2		n=4	
	Tempo (s)	S	Tempo (s)	S	Tempo (s)	S
Serial	1949	-	2528	-	2645	-
20	263	7,411	312	8,103	316	8,370
40	198	9,843	241	10,490	244	10,840
60	180	10,828	210	12,038	211	12,536
80	181	10,768	216	11,704	235	11,255
100	173	11,266	206	12,272	209	12,656
120	175	11,137	204	12,392	201	13,159
140	187	10,422	214	11,813	216	12,245
160	177	11,011	205	12,332	210	12,595
180	174	11,201	207	12,213	204	12,966
200	185	10,535	214	11,813	217	12,189
220	178	10,949	210	12,038	211	12,536
240	176	11,074	207	12,213	214	12,360

Fonte: O autor, 2021

Os valores máximo, 13,159, e mínimo, 7,411, para o *speedup* forma atingidos, respectivamente, com 120 e 20 threads, $n=4$ e 0 e tempos de execução de 201 e 263 segundos.

Dando prosseguimento à análise de sensibilidade, combinou-se os parâmetros dos

casos anteriores que apresentaram os maiores *speedups*, ou seja, $k=0,025 \times 10^{-12} \text{ m}^2$, $a=0,6$, $\phi=0,15$, $Q_H=4 \text{ kW}$ e $n=4$ e um novo conjunto de dados foi elaborado, chamado de Caso Otimizado. Os resultados obtidos com esses valores foram comparados com os da Caso Base apresentado na Tabela 14. Como esperado, com o novo conjunto de dados foi possível obter os maiores *speedups* para essa análise de sensibilidade. O melhor desempenho, com o Conjunto Otimizado, forneceu um *speedup* de 14,970, para 120 *threads* e um tempo de execução de 199 segundos. Já o pior resultou em um *speedup* de 8,103 com 20 *threads*, um tempo de execução de 312 segundos e o Caso Base.

Tabela 14 - Análise de sensibilidade para o Conjunto Otimizado

<i>Threads</i>	Caso Base		Conjunto Otimizado	
	Tempo (s)	<i>S</i>	Tempo (s)	<i>S</i>
Serial	2528	-	2979	-
20	312	8,103	337	8,840
40	241	10,490	242	12,310
60	210	12,038	212	14,052
80	216	11,704	217	13,728
100	206	12,272	208	14,322
120	204	12,392	199	14,970
140	214	11,813	204	14,603
160	205	12,332	207	14,391
180	207	12,213	204	14,603
200	214	11,813	207	14,391
220	210	12,038	203	14,675
240	207	12,213	209	14,254

Fonte: O autor, 2021

4.4.6 Vazão

Para fechar esta seção, abordou-se a questão da variação da vazão de produção. Para tanto, foram utilizadas vazões de -100, -150 e -200 std m^3/dia . Além disso, empregou-se uma taxa de aquecimento de 8.000 W e 4 poços aquecedores distribuídos simetricamente ao redor do poço produtor. Os resultados alcançados encontram-se apresentados na Tabela 15 e na Figura 23, para a pressão p_{wf} em função do tempo.

Na Tabela 15, como nos casos anteriores, são fornecidos o tempo de execução computacional e os valores do *speedup* para os diferentes números de *threads* considerados. Para os valores testados, percebe-se que os tempos de execução e os *speedups* ficaram

muito próximos uns dos outros.

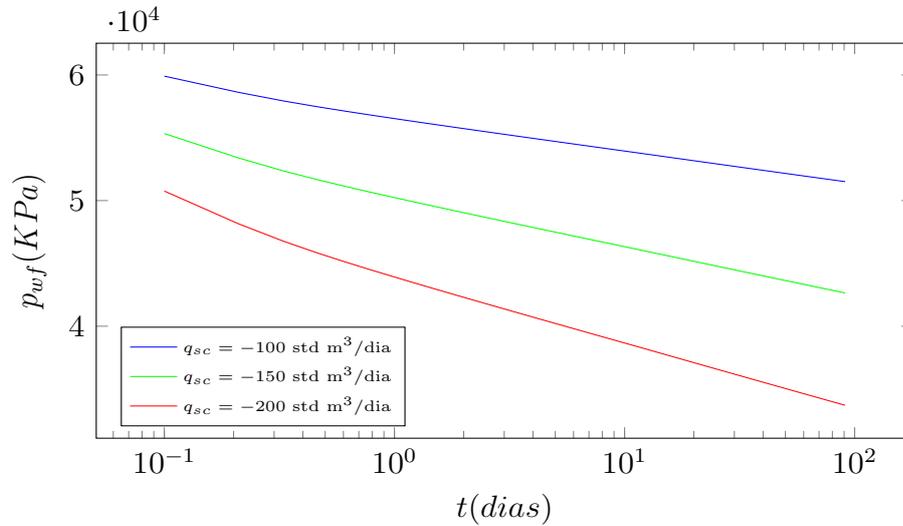
Na Figura 23, verifica-se que as maiores quedas da pressão ocorrem para as maiores vazões. Fisicamente, uma suplementação de energia deve ser fornecida para que o reservatório produza uma maior quantidade de fluido no mesmo intervalo de tempo. Assim sendo, as curvas estão de acordo com o comportamento previsto.

Tabela 15 - Análise de sensibilidade para a vazão

<i>Threads</i>	Vazão (std m ³ /dia)					
	-100		-150		-200	
	Tempo (s)	<i>S</i>	Tempo (s)	<i>S</i>	Tempo (s)	<i>S</i>
Serial	2899	-	2918	-	2919	-
20	343	8,452	335	8,710	345	8,461
40	255	11,369	254	11,488	253	11,538
60	223	13,000	219	13,324	226	12,916
80	227	12,771	233	12,524	234	12,474
100	220	13,177	214	13,636	214	13,640
120	212	13,675	213	13,700	208	14,034
140	221	13,118	220	13,264	221	13,208
160	214	13,547	213	13,700	214	13,640
180	211	13,739	211	13,829	211	13,834
200	221	13,118	224	13,027	219	13,329
220	216	13,421	219	13,324	218	13,390
240	218	13,298	213	13,700	214	13,640

Fonte: O autor, 2021

Figura 23 - Variação da pressão em função da vazão



Fonte: O autor, 2021.

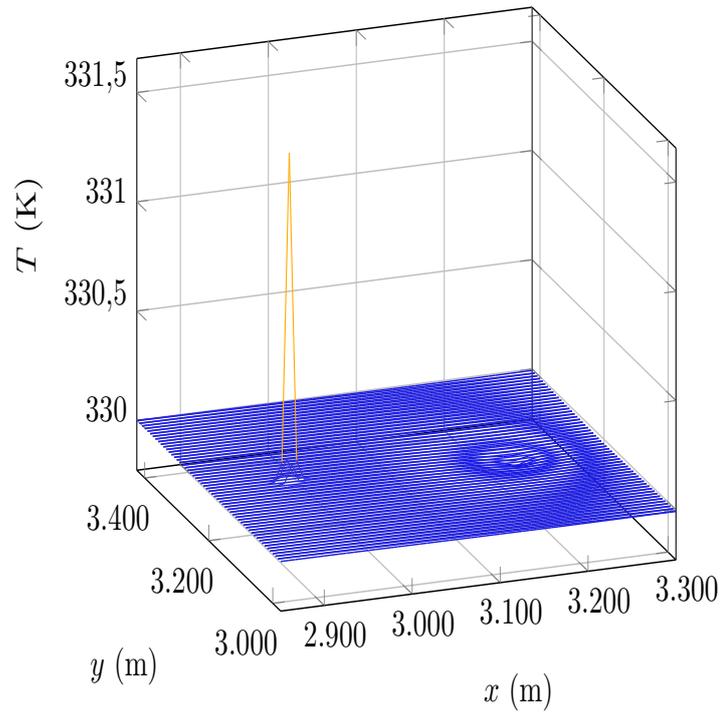
4.5 Campos de pressão e temperatura do reservatório

Finalmente, ainda que não seja o foco deste trabalho, apresentam-se algumas superfícies dos campos de temperatura e de pressão do reservatório, vide as Figuras 24-29. Eles foram todos calculados com a Malha 5, que é a padrão, e retratam os valores visualizados a partir de uma ampliação da região próxima da localização do poço e dos aquecedores estáticos.

Nas Figuras 24 e 25, respectivamente, encontram-se os campos de temperatura e de pressão. Deve-se destacar que foi capturado o aumento de temperatura região onde encontra-se um dos aquecedores e uma ligeira alteração de temperatura na região próxima ao poço produtor. No caso da pressão, como previsto, os menores valores de pressão ocorrem nas células que contêm o poço produtor.

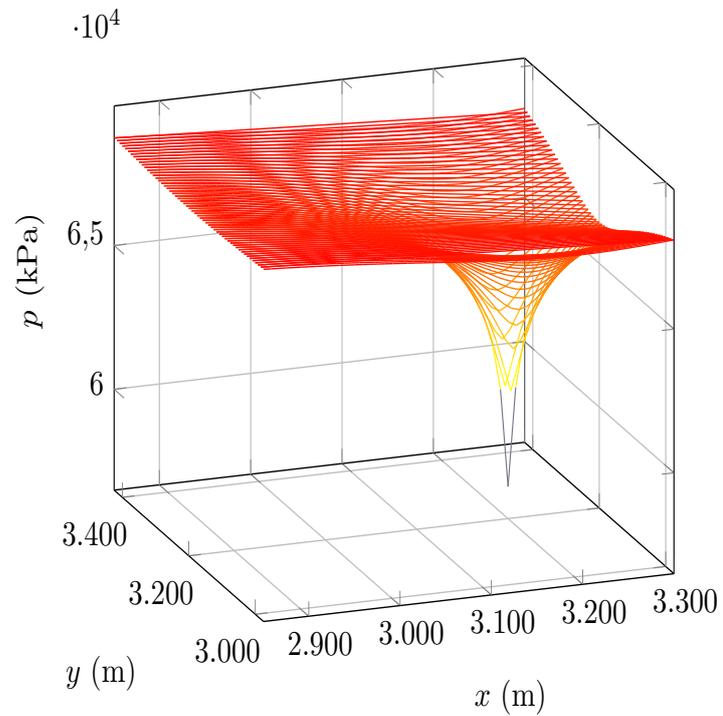
No caso das Figuras 26 e 27 também tem-se, respectivamente, os campos de temperatura e de pressão que foram determinados com a Malha 5. No entanto, agora, as curvas correspondem aos valores computados para o conjunto de propriedades que resulta na maior difusividade hidráulica e que conduziu ao maior valor do *speedup*.

Figura 24 - Temperatura na região próxima ao poço produtor - Malha 5



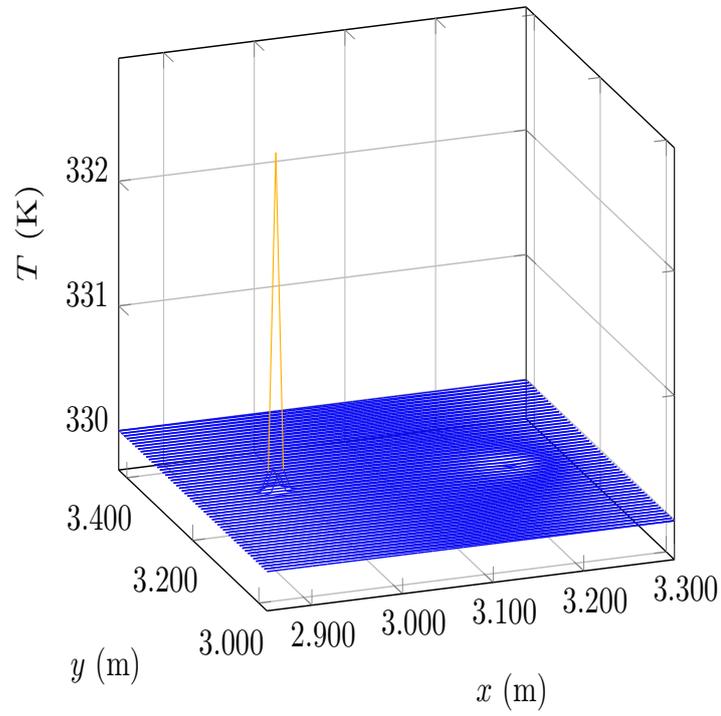
Fonte: O autor, 2021.

Figura 25 - Pressão na região próxima ao poço produtor - Malha 5



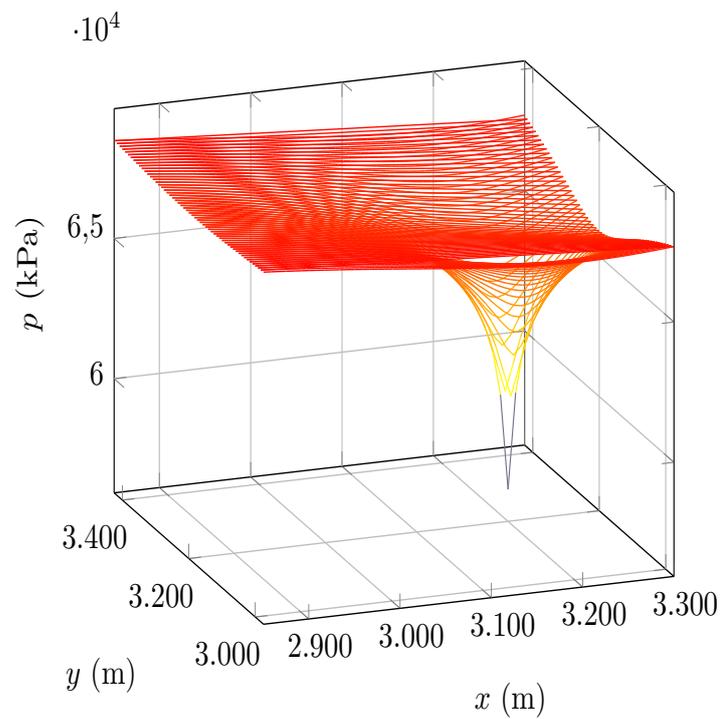
Fonte: O autor, 2021.

Figura 26 - Temperatura na região próxima ao poço produtor - Maior difusividade hidráulica



Fonte: O autor, 2021.

Figura 27 - Pressão na região próxima ao poço produtor - Maior difusividade hidráulica



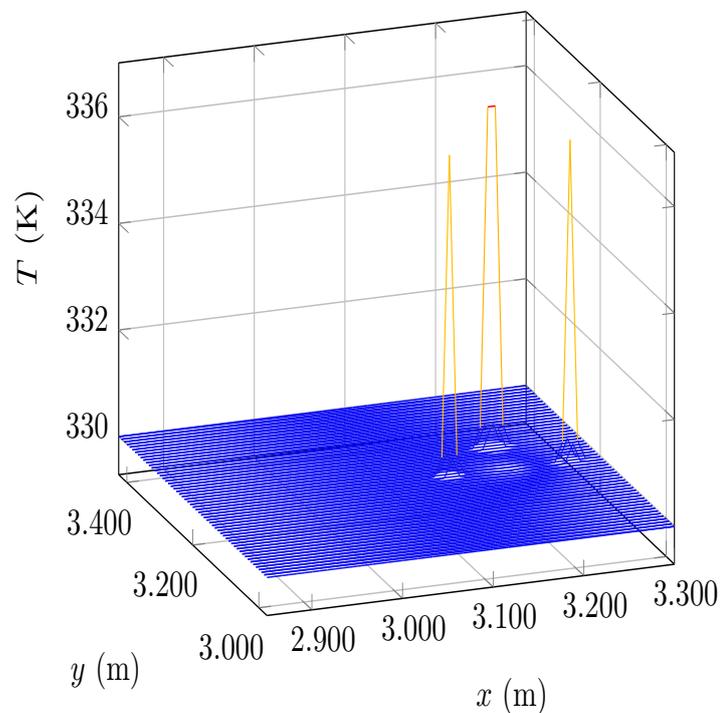
Fonte: O autor, 2021.

Constata-se que o mesmo comportamento qualitativo, análogo aos encontrados nas Figuras 24 e 25, foi observado nas Figuras 26 e 27, ou seja, aumento da temperatura nas regiões próximas ao aquecedor e ao poço produtor, e uma queda de pressão mais pronunciada na região onde encontra-se o poço produtor. Numericamente, ressalta-se que os valores obtidos com a Malha 5, no caso padrão e no de maior *speedup*, são diferentes. Isso pôde ser verificado via a inspeção dos arquivos de saída.

Concluindo, tem-se que as Figuras 28 e 29 contêm, nesta ordem, as superfícies representando os campos de temperatura e de pressão do reservatório, determinados em uma simulação usando a Malha 5, a existência de quatro aquecedores com potência de 8 kW, distribuídos simetricamente ao redor do poço, e todos distantes de 64 m do poço produtor. No caso da temperatura, pode-se afirmar que a presença dos aquecedores e da maior potência liberada foram devidamente capturadas e resultaram nos picos de temperatura vistos no corte da superfície representando o campo de temperatura.

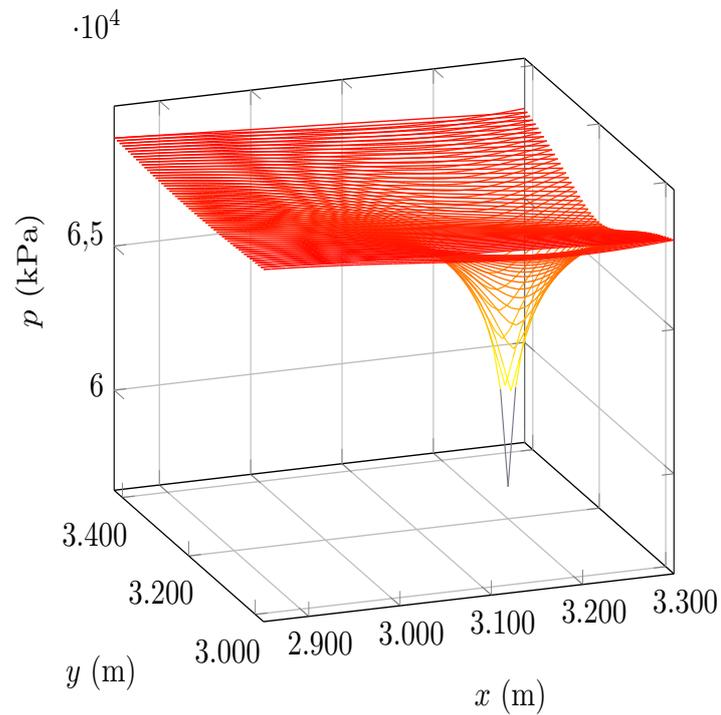
Do ponto de vista da pressão, o seu campo segue apresentando a maior queda de pressão na região ao redor do poço produtor. No entanto, acredita-se que o conjunto de propriedades utilizado e as condições de operação adotadas não foram muito favoráveis, de modo que não foi possível evidenciar, como pretendido, os efeitos do aquecimento nos campos de temperatura e pressão.

Figura 28 - Temperatura na região próxima ao poço produtor - Quatro aquecedores



Fonte: O autor, 2021.

Figura 29 - Pressão na região próxima ao poço produtor - Quatro aquecedores



Fonte: O autor, 2021.

Em particular, alterações no número de poços, no número de aquecedores, na distância entre o poço produtor e os aquecedores, bem como nas dimensões do reservatório poderiam auxiliar nesta direção, tomando por base o que foi realizado, por exemplo, no trabalho de [Heringer et al. \(2019\)](#). Entretanto, destaca-se aqui que o caso padrão foi configurado com o objetivo de discutir a questão do desempenho computacional, de modo a se ter uma malha suficientemente refinada tal que o raio equivalente de Peaceman pudesse ser calculado.

CONCLUSÕES E PERSPECTIVAS

Neste trabalho, utilizou-se técnicas de simulação numérica de reservatórios para resolver o problema da recuperação terciária utilizando um método térmico, considerando um escoamento monofásico e não-isotérmico de óleo. Nele, aquecedores são introduzidos no reservatório de maneira a tornar o óleo menos viscoso e, assim, favorecer a sua recuperação através do poço produtor. Uma série de resultados foram analisados a fim de se verificar o desempenho computacional, quando aplicada a estratégia proposta de utilização conjunta de uma decomposição de operadores, de uma linearização, da API OpenMP e de um coprocessador.

Conclusões

A tese traz como contribuição a proposição de uma combinação de estratégias, para a determinação numérica da pressão e da temperatura no escoamento não-isotérmico em reservatórios de óleo, utilizando a computação em paralelo e um coprocessador. As equações para a pressão e a temperatura foram resolvidas de maneira desacoplada sendo, primeiramente, obtido o campo de pressões seguido do campo de temperatura média representativa do reservatório. Desta forma, um método de decomposição de operadores foi empregado na resolução dos dois subsistemas de equações algébricas não-lineares, um para a determinação da pressão e o outro para a temperatura, após a linearização das mesmas pelo método de Picard. O método de resolução dos sistemas algébricos escolhido foi o dos Gradientes Conjugados, paralelizado com a API OpenMP, que se mostrou eficiente na obtenção das soluções, preservando a integridade dos resultados obtidos com a versão serial do simulador. Os resultados foram obtidos a partir da execução do simulador em um Coprocessador Intel Xeon Phi.

Como foi constatado, o desempenho computacional depende do refinamento da malha utilizada, dos parâmetros físicos, do número de *threads* e da forma como o código é paralelizado. A utilização da API OpenMP e do coprocessador possibilitaram uma redução significativa no tempo de CPU, para as simulações numéricas realizadas neste trabalho usando as Malhas 4, 5, 6 e 7, para as quais o *speedup* sempre foi maior do que 6 (o que ocorre a partir da Malha 4 para os casos aqui estudados) e se aproximaram de 30 nas simulações com 220 e 240 *threads* em se tratando da Malha 7. Foi possível observar que conforme o número total de células cresce o *speedup* obtido também aumenta. No caso do aumento no número de *threads*, notou-se que para a Malha 4 houve uma queda de desempenho para os maiores números de *threads*, enquanto que para as Malhas 5 e 6 há uma tendência de estagnação para os maiores números de *threads*. Para a Malha 7, os

maiores *speedups* foram obtidos justamente para os maiores números de *threads*. Nesse caso, o *speedup* sempre cresceu à medida que o número de *threads* aumentou, ainda que em um ritmo mais suave do que ocorre para a faixa do número de *threads* igual ou menor do que 100.

Finalmente, em função das diferentes formas de implementação da paralelização aqui utilizadas, ficou claro que a versão utilizada na maioria dos testes foi a mais eficiente. Nela, paralelizou-se não somente os produtos escalar e matriz-vetor no método dos gradientes conjugados, além de outros trechos do código numérico. Foram paralelizados a avaliação do erro absoluto (para o estabelecimento da convergência) no método de Picard bem como o cálculo das transmissibilidades. Isso é relevante, visto que o maior esforço computacional neste tipo de problema é demandado, em geral, pela execução do método iterativo de solução dos sistemas de equações sendo, tipicamente, o produto matriz-vetor a operação mais custosa. No entanto, conseguiu-se atingir ganhos expressivos também na etapa de avaliação dos erros (no método dos Gradientes Conjugados e na iteração de Picard). Isso pôde ser visto nas Tabelas 7 e 8, quando foram comparados os Códigos 1, 2 e 3, ou seja, as três opções de paralelização do código numérico.

Perspectivas

Findas as etapas de codificação, obtenção dos resultados e discussão, se é capaz de listar algumas perspectivas para o desenvolvimento de futuros trabalhos que deem prosseguimento a esta pesquisa.

Uma primeira linha de trabalho seria a paralelização do simulador numérico considerando o uso de um modelo a duas equações, em termos das temperaturas de fluido e de rocha, para a descrição do processo de transferência de calor na escala de laboratório, e utilizando também uma decomposição de operadores (HERINGER *et al.*, 2019). Nesse modelo, seguindo a estratégia apresentada em (HERINGER *et al.*, 2019), deve-se resolver três subsistemas de equações algébricas: para a determinação da pressão do óleo, da temperatura do óleo e da temperatura da rocha.

Outras possibilidades envolvem a utilização dos conhecimentos aqui adquiridos na paralelização de outros simuladores do Laboratório de Modelagem Multiescala e Transporte de Partículas - LABTRAN (www.labtran.iprj.uerj.br) como, por exemplo, os destinados à resolução do escoamento bifásico imiscível e isotérmico (FREITAS; SOUZA; SOUTO, 2020), a extensão do trabalho de Werneck *et al.* (2019) para tratar do escoamento monofásico de gás com um poço de produção horizontal e o escoamento de gás incluindo os efeitos não-Darcy (PESSANHA *et al.*, 2020).

Por fim, uma outra frente de trabalho seria voltada para a melhoria na programação das inserções/opções das diretivas do OpenMP, otimizando a paralelização, como o já

realizado em [Carvalho et al. \(2020\)](#).

REFERÊNCIAS

- ABOU-KASSEM, J. H.; ALI, S. M.; ISLAM, M. R. *Petroleum Reservoir Simulation, A Basic Approach*. Houston, USA: Gulf Publishing Company, 2006.
- AFSAR, C.; AKIN, S. Solar generated steam injection in heavy oil reservoirs: A case study. *Renewable Energy*, v. 91, p. 83–89, 2016.
- ANTUNES FILHO, A.; XAVIER, A. C. C. Solução de sistemas lineares esparsos utilizando CUDA: uma comparação de desempenho em sistemas Windows e Linux. *Revista de Ciências Exatas e Tecnologia*, v. 8, n. 8, p. 181–195, 2015.
- AZIZ, M.; SETTARI, A. *Petroleum Reservoir Simulation*. New York, USA: Elsevier Applied Science, 1990.
- BERA, A.; BABADAGLI, T. Status of electromagnetic heating for enhanced heavy oil/bitumen recovery and future prospects: A review. *Applied Energy*, v. 151, p. 206–226, 2015.
- BIENTINESI, M. et al. A radiofrequency/microwave heating method for thermal heavy oil recovery based on a novel tight-shell conceptual design. *Journal of Petroleum Science and Engineering*, v. 107, p. 18–30, 2013.
- BRUNSCHEN, C.; BRORSSON, M. OdinMP/CCp - a portable implementation of OpenMP for C. *Concurrency - Practice and Experience*, v. 12, n. 12, p. 1193–1203, 2000.
- CALLE, C. R. *A fast IMPES multiphase flow solver in porous media for reservoir simulation*. Tese (Doutorado) — Universidad Politécnica de Madrid, 2017.
- CARPEN-AMARIE, A.; HUNOLD, S.; TRÄFF, J. L. On expected and observed communication performance with mpi derived datatypes. *Parallel Computing*, v. 69, p. 98–117, 2017.
- CARVALHO, L. T. M. et al. Uma implementação paralelizada via a API OpenMP para a simulação numérica de reservatórios de gás natural. *Revista Brasileira de Computação Aplicada*, v. 12, p. 103–121, 2020.
- CENGEL, Y. A. *Heat and Mass Transfer, A Practical Approach*. [S.l.]: McGraw-Hill, 2006.
- CHANDRA, R. *Parallel programming in OpenMP*. [S.l.]: Morgan Kaufmann, 2001.
- CHAPMAN, B.; JOST, G.; van der PAS, R. *Using OpenMP: portable shared memory parallel programming*. [S.l.]: MIT press, 2008. v. 10.
- CHEN, Z.; HUAN, G.; LI, B. An improved impes method for two-phase flow in porous media. *Transport in Porous Media*, v. 54, n. 3, p. 361–376, 2004.
- CHEN, Z.; HUAN, G.; MA, Y. *Computational Methods for Multiphase Flows in Porous Media*. Philadelphia, USA: Society of Industrial and Applied Mathematics, 2006.

DAKE, L. P. *The Practice of Reservoir Engineering (Revised Edition)*. Amsterdam, The Netherlands: Elsevier, 2001. Developments in Petroleum Science 36.

DANDEKAR, A. Y. *Petroleum reservoir rock and fluid properties*. [S.l.]: CRC press, 2013.

DARCY, H. *Les fontaines publiques de la ville de Dijon. Exposition et application des principes à suivre et des formules à employer dans les questions de distribution d'eau*. [S.l.]: Victor Dalmont, 1856.

DAVLETBAEV, A.; KOVALEVA, L.; BABADAGLI, T. Mathematical modeling and field application of heavy oil recovery by radio-frequency electromagnetic stimulation. *Journal of Petroleum Science and Engineering*, v. 78, n. 3, p. 646–653, 2011.

de la Cruz, L. M.; MONSIVAIS, D. Parallel numerical simulation of two-phase flow model in porous media using distributed and shared memory architectures. *Geofísica Internacional*, v. 53, n. 1, p. 59–75, 2014.

DEVRIES, B. et al. Parallel implementations of FGMRES for solving large, sparse non-symmetric linear systems. *Procedia Computer Science*, Elsevier, v. 18, p. 491–500, 2013.

DIETRICH, R. et al. Critical-blame analysis for OpenMP 4.0 offloading on Intel Xeon Phi. *Journal of Systems and Software*, v. 125, p. 381–388, 2017.

DUMKWU, F. A.; AKAND, I. W.; CARLSON, E. S. Review of well models and assessment of their impacts on numerical reservoir simulation performance. *Journal of Petroleum Science and Engineering*, p. 174–186, 2012.

DYRDAHL, J. *Thermal flow in fractured porous media and operator splitting*. Dissertação (Mestrado) — Norwegian University of Science and Technology, Trondheim, Norway, 2014.

ERTEKIN, T.; ABOU-KASSEM, J. H.; KING, G. R. *Basic Applied Reservoir Simulation*. Richardson, USA: Society of Petroleum Engineers, 2001.

EZEKWE, N. *Petroleum Reservoir Engineering Practice*. Westford, USA: Prentice Hall, 2010.

FREITAS, M. M.; SOUZA, G.; SOUTO, H. P. A. A Picard-Newton approach for simulating two-phase flow in petroleum reservoirs. *International Journal of Advanced Engineering Research and Science*, v. 7, n. 4, p. 428–457, 2020.

FRIAS, D. *Modelagem e simulação numérica de escoamentos miscíveis compressíveis em meios porosos*. Tese (Doutorado) — Universidade do Estado do Rio de Janeiro, 1999.

GASPARINI, L. et al. Hybrid parallel iterative sparse linear solver framework for reservoir geomechanical and flow simulation. *Journal of Computational Science*, v. 51, p. 101330, 2021.

GILLBERG, T.; SOUROURI, M.; CAI, X. A new parallel 3D front propagation algorithm for fast simulation of geological folds. *Procedia Computer Science*, Elsevier, v. 9, p. 947–955, 2012.

- GONÇALVES, R. R. d. O. *Análise do Desempenho e Consumo de Energia de Coprocessadores Xeon Phi utilizando Diferentes Modelos de Programação*. Dissertação (Mestrado) — Universidade Federal do Pampa, Alegrete, Brasil, 2018.
- HAGER, G.; WELLEIN, G. *Introduction to high performance computing for scientists and engineers*. [S.l.]: CRC Press, 2010.
- HALSEY, T. C. Computational sciences in the upstream oil and gas industry. *Philosophical Transactions of Royal Society*, v. 374, p. 1–12, 2016.
- HEBENSTREIT, M. Configuring intel® xeon phitm coprocessors inside a cluster. 2013.
- HEMSTAD, J. *Modeling a non-uniform memory access architecture for optimizing conjugate gradient performance with sparse matrices*. Dissertação (Dissertação de Mestrado) — College of Saint Benedict and Saint John's University, Minnessota, USA, 2013.
- HERINGER, J. D. d. S. et al. *Simulação numérica de escoamento tridimensional não-isotérmico em reservatórios de petróleo*. Dissertação (Mestrado), 2018.
- HERINGER, J. D. S. et al. Numerical simulation of non-isothermal flow in oil reservoirs using two-equation model. *Coupled Systems Mechanics*, v. 8, n. 2, p. 147–168, 2019.
- INCROPERA, A. et al. *Fundamentals of Heat and Mass Transfer*. Philadelphia, USA: Society of Industrial and Applied Mathematics, 2006.
- ISLAM, M. R. et al. *Advanced Petroleum Reservoir Simulation*. Trondheim, Norway: Wiley, 2010.
- KOVSCZEK, A. R. Emerging challenges and potential futures for thermally enhanced oil recovery. *Journal of Petroleum Science and Engineering*, v. 98-99, p. 130–143, 2012.
- LAKE, W. L. *Enhanced Oil Recovery*. Englewood Cliffs, USA: Prentice-Hall, 1989.
- LAMPE, V. *Modelling Fluid Flow and Heat Transport in Fractured Porous Media*. Dissertação (Mestrado) — University of Bergen, Noruega, 2013.
- LIU, H.; TAFTI, D. K.; LI, T. Hybrid parallelism in MFIX CFD-DEM using OpenMP. *Powder Technology*, Elsevier, v. 259, p. 22–29, 2014.
- LOPES, R. B.; SOUZA, G. de; SOUTO, H. P. A. Simulação numérica de escoamento não isotérmico em reservatório de óleo com poços aquecedores. *CALIBRE-Revista Brasiliense de Engenharia e Física Aplicada*, v. 3, n. 2, p. 9–24, 2018.
- MAES, J. et al. Modelling in-situ upgrading of heavy oil using operator splitting method. *Computational Geoscience*, v. 18, n. 3, p. 183–194, 2015.
- MALY, L. et al. Evaluation of the intel xeon phi offload runtimes for domain decomposition solvers. *Advances in Engineering Software*, v. 125, p. 146–154, 2018.
- MEI, C. C.; AURIAULT, J.-L.; NG, C.-O. Some applications of the homogenization theory. In: HUTCHINSON, J. W.; WU, T. Y. (Ed.). [S.l.]: Elsevier, 1996, (Advances in Applied Mechanics, v. 32). p. 277–348.

- MEISTER, O.; BADER, M. 2D adaptivity for 3D problems: Parallel SPE10 reservoir simulation on dynamically adaptive prism grids. *Journal of Computational Science*, Elsevier, v. 9, p. 101–106, 2015.
- MOYNE, C. et al. Thermal dispersion in porous media: one-equation model. *International Journal of Heat and Mass Transfer*, v. 43, n. 20, p. 3853–3867, 2000.
- NIELD, D. A.; BEJAN, A. *Convection in Porous Media*. United States of America: Springer, 2006.
- OZISIK, M. N. *Finite difference methods in Heat Transfer*. [S.l.]: CRC Press., 1994.
- PEACEMAN, D. W. Interpretation of well-block pressures in numerical reservoir simulation. *Society of Petroleum Engineers Journal*, v. 18, n. 3, p. 183–194, 1978.
- PEACEMAN, D. W. Interpretation of well-block pressures in numerical reservoir simulation with nonsquare grid blocks and anisotropic permeability. *Society of Petroleum Engineers Journal*, v. 23, n. 3, p. 531–543, 1983.
- PESSANHA, M. L. O. et al. Comparative study and sensitivity analysis in simulation of non-darcy flow in shale gas reservoirs. *International Journal of Advanced Engineering Research and Science*, v. 7, n. 11, p. 109–121, 2020.
- QUINTARD, M.; KAVIANY, M.; WHITAKER, S. Two-medium treatment of heat transfer in porous media: numerical results for effective properties. *Advances in Water Resources*, v. 20, n. 2, p. 77–94, 1997.
- RAHMAN, R. *Intel Xeon Phi coprocessor architecture and tools: the guide for application developers*. [S.l.]: Springer Nature, 2013.
- REDONDO, C.; RUBIO, G.; VALERO, E. On the efficiency of the impes method for two phase flow problems in porous media. *Journal of Petroleum Science and Engineering*, v. 164, p. 427–436, 2018.
- ROSA, A. J.; CARVALHO, R. S.; XAVIER, J. A. D. *Engenharia de reservatórios de petróleo*. [S.l.]: Interciência, 2006.
- ROUSSET, M. *Reduced-order modeling for thermal simulation*. Dissertação (Mestrado) — University of Stanford, Stanford, USA, 2010.
- SAAD, Y. *Iterative Methods for Sparse Linear Systems*. 2. ed. Philadelphia: SIAM, 2003.
- SCHULZ, J. A. T. *Simulação numérica do escoamento bifásico em meios porosos heterogêneos empregando uma formulação semi-implícita, limitadores de fluxo e o método dos volumes finitos*. Tese (Doutorado) — Universidade do Estado do Rio de Janeiro, 2009.
- SILVA, M. O. *Controle de granularidade de tarefas em OpenMP*. Porto Alegre, RS, 2011.
- SINGH, A. K.; GOERKE, U. J.; KOLDITZ, O. Numerical simulation of non-isothermal compositional gas flow: application to carbon dioxide injection into gas reservoirs. *Energy*, v. 36, p. 3446–3458, 2011.

SOUZA, G. *Modelagem computacional de escoamentos com duas e três fases em reservatórios petrolíferos heterogêneos*. Dissertação (Mestrado) — Universidade do Estado do Rio de Janeiro, Nova Friburgo, Brasil, 2008.

SOUZA, G. *Acoplamento poço-reservatório na simulação numérica de reservatórios de gás*. Tese (Doutorado) — Universidade Estadual do Norte Fluminense, 2013.

TERBOVEN, C. et al. Experiences with the OpenMP parallelization of DROPS, a Navier-Stokes solver written in C++. In: *OpenMP Shared Memory Parallel Programming*. [S.l.]: Springer, 2008. p. 95–106.

THOMAS, J. E. *Fundamentos de engenharia de petróleo*. Rio de Janeiro: Interciência, 2001.

TORELLI, J. C.; BRUNO, O. M. Programação paralela em SMPs com OpenMP e POSIX Threads: um estudo comparativo. In: *Anais do IV Congresso Brasileiro de Computação (CBCComp)*. [S.l.: s.n.], 2004. v. 1, p. 486–491.

VENÂNCIO JÚNIOR, A. C.; SILVA, R. G. *Computação distribuída de alto desempenho - um estudo de caso de acesso móvel*. Florianópolis, SC, 2004.

VENNEMO, S. B. *Multiscale Simulation of Thermal Flow in Porous Media*. Dissertação (Mestrado) — Norwegian University of Science and Technology, Trondheim, Norway, 2016.

WERNECK, L. F. *Implementação paralelizada de métodos de resolução de sistemas algébricos na simulação de reservatórios de gás*. Dissertação (Mestrado) — Universidade do Estado do Rio de Janeiro, Nova Friburgo, 2016.

WERNECK, L. F. et al. An OpenMP parallel implementation using a coprocessor for numerical simulation of oil reservoirs. *Computational and Applied Mathematics*, v. 38, p. 1–30, 2019.

WHITAKER, S. *The Method of Volume Average. Theory and Applications of Transport in Porous Media*. [S.l.]: Kluwer Academic Publishers, 1999.

WU, Y.; YE, M. A parallel sparse grid construction algorithm based on the shared memory architecture and its application to flash calculations. *Computers & Mathematics with Applications*, v. 77, n. 8, p. 2114–2129, 2019.

YAKUBOV, S. et al. Hybrid MPI/OpenMP parallelization of an euler–lagrange approach to cavitation modelling. *Computers & Fluids*, v. 80, p. 365–371, 2013.

ZHU, H. et al. A parallel non-local means denoising algorithm implementation with OpenMP and OpenCL on Intel Xeon Phi coprocessor. *Journal of Computational Science*, v. 17, p. 591–598, 2016.