UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE TECNOLOGIA E GEOCIÊNCIAS
PROGRAMA DE RECURSOS HUMANOS DA ANP – PRH 26

TESE DE DOUTORAMENTO

Simulação de Escoamento Bifásico Oléo-Água em Reservatórios de Petroléo Usando Computadores Paralelos de Memória Distribuída

Bolsista DSc. Rogério Soares da Silva

Orientador: Prof. Paulo Roberto Maciel Lyra (PhD)
Co-Orientador: Prof. Ramiro Brito Willmersdorf (PhD)

PRH-26

RECIFE – 2008

Livros Grátis

http://www.livrosgratis.com.br

Milhares de livros grátis para download.

Universidade Federal de Pernambuco
Centro de Tecnologia e Geociências
Departamento de Engenharia Civil

Simulação de Escoamento Bifásico Oléo-Água em Reservatórios de Petróleo Usando Computadores Paralelos de Memória Distribuída

Rogério Soares da Silva

TESE SUBMETIDA AO CORPO DOCENTE DO CURSO DE PÓS-GRADUAÇÃO EM ENGENHARIA CIVIL DA UNIVERSIDADE FEDERAL DE PERNAMBUCO COMO PARTE DOS REQUISITOS NECESSÁRIOS À OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS DE ENGENHARIA CIVIL.

ÁREA DE CONCENTRAÇÃO:
ESTRUTURAS (ENGENHARIA DE PETRÓLEO)

Paulo Roberto Maciel Lyra, PhD
(Orientador)

Recife, Pernambuco, Brasil.
©Rogério Soares da Silva, Outubro de 2008.

S586s **Silva, Rogério Soares da.**
Simulação de escoamento bifásico óleo-água em reservatórios de petróleo usando computadores paralelos de memória distribuída / Rogério Soares da Silva. – Recife: O Autor, 2008.
xiv, 144 folhas, il : tabs., grafs.

Tese (Doutorado) – Universidade Federal de Pernambuco. CTG. Programa de Pós-Graduação em Engenharia Civil, 2008.

Inclui Bibliografia e Apêndice.

1. Engenharia civil. 2. Reservatório de petróleo. 3. Método dos volumes finitos. 4. IMPES. 5. Programação orientada a objetos. 6. C++. 7. Computação paralela. I. Título.

UFPE

624 **CDD (22. Ed.)** **BCTG/ 2009-067**

SIMULAÇÃO DE ESCOAMENTO BIFÁSICO ÓLEO-ÁGUA EM RESERVATÓRIOS DE PETRÓLEO USANDO COMPUTADORES PARALELOS DE MEMÓRIA DISTRIBUÍDA

Rogério Soares da Silva

TESE SUBMETIDA AO CORPO DOCENTE DO PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA CIVIL DA UNIVERSIDADE FEDERAL DE PERNAMBUCO COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS À OBTENÇÃO DO GRAU DE DOUTOR EM ENGENHARIA CIVIL

Aprovada por:

Paulo Roberto Maciel Lyra, Ph. D.
(Orientador)

Ramiro Brito Willmersdorf, Ph. D.
(Orientador)

Philippe Remy Bernard Devloo, Ph. D
(Examinador Externo)

Ana Paula de Araújo Costa, D. Sc.
(Examinador Externo)

Darlan Karlo Elisiário de Carvalho, D. Sc.
(Examinador Externo)

Lícia Mouta da Costa, D. Sc.
(Examinador Interno)

Recife, PE – Brasil
Outubro de 2008

Agradecimentos

Ao Programa de Recursos Humanos da Agência Nacional de Petróleo (PRH-26-ANP), pela bolsa concedida durante o período de realização do doutorado.

A todo o pessoal técnico e administrativo do PRH-26, particularmente a Sônia pela dedicação em resolver problemas burocráticos.

Ao coordenador do PRH-26 Prof. Mário Filho pela disposição, alegria e atenção despendida durante os congressos e reuniões.

Aos colegas Daniel Ventura, Alessandro Antunes e D. Eliane, secretária da pós-graduação do DEMEC, pela amizade e descontração durante o intervalo do café das 10h e das 15h.

Também agradeço ao colega Alessandro Antunes pelas idéias trocadas sobre programação paralela.

Agradeço especialmente ao colega Darlan Carvalho, a quem devo imensamente a ajuda prestada em todos os momentos da tese com idéias, sugestões e críticas que foram de grande contribuição para este trabalho.

Aos alunos de iniciação científica do PADMEC: Érika Oliveira e Bruno Pascaretta pelos trabalhos que desenvolvemos juntos.

Ao Professores Paulo Lyra e Ramiro Willmersdorf pela orientação, clareza, conselhos e sugestões durante meu trabalho.

Aos meus pais Arlete e Sérgio pelo apoio que me deram ao longo desses quatro anos de estudo e dedicação.

Agradeço a minha namorada Ana Janaina, uma pessoa formidável e de uma alegria contagiante que me apoiou durante todos os momentos do meu trabalho.

Resumo

A busca por técnicas de alto desempenho por engenheiros e pesquisadores no campo da simulação numérica em reservatórios de petróleo é um caminho que vem sendo percorrido desde o início da década de 1950 com o advento dos primeiros computadores digitais. Prever o comportamento de um reservatório ao longo de sua vida produtiva e extrair a máxima quantidade de hidrocarbonetos economicamente viável é uma tarefa que exige um conhecimento mais detalhado das características dos fluidos (líquidos e/ou gasosos) e do meio poroso por onde escoam. A modelagem matemática da física envolvida no escoamento de fluidos em meios porosos resulta em equações diferenciais parciais (EDPs) não-lineares que apenas apresentam soluções analíticas em casos muito simplificados.

Métodos numéricos como os de diferenças finitas e de volumes finitos, e mais recentemente de elementos finitos, têm sido aplicados neste campo e exigem um tratamento especial quando se deseja empregá-los em simulações com o auxílio de computadores paralelos partindo desde a fase de pré-processamento, passando pela estrutura de dados do simulador, até a resolução das equações discretas.

O uso de computadores paralelos de memória distribuída é apresentado como uma ferramenta de alto desempenho que pode permitir uma redução significativa no tempo total de simulação ou pode viabilizar a simulação de problemas de grande porte, ou seja, problemas que excedem a capacidade de memória de uma única máquina devido a enorme quantidade de dados envolvidos.

Neste trabalho, é apresentado um simulador de reservatórios de petróleo paralelo desenvolvido em C++ e que faz uso de pacotes gratuitos de código aberto voltados à programação paralela como FMDB, ParMetis e Petsc que desempenham tarefas específicas importantes como o gerenciamento de malhas distribuídas, o balanço de carga entre processadores e a utilização de resolvedores (solvers) iterativos paralelos, respectivamente.

Uma formulação de volumes finitos centrada nos vértices e baseadas em uma estrutura de dados por arestas (Carvalho, 2005), aqui chamada de EBFV1 (Edge Based Finite Volume), é utilizada na discretização das equações governantes para simular o escoamento incompressível óleo-água em meios porosos heterogêneos e anisotrópicos tanto em malhas de triângulos quanto de tetraedros. Nesta formulação, a heterogeneidade do meio está associada a subdomínios. O método de integração no tempo IMPES (IMplicit Pressure Explicit Saturation) é adotado nas simulações em conjunto com um procedimento adaptativo no passo de tempo (Hurtado et al., 2006).

Uma segunda formulação de volumes finitos (EBFV2) também com volumes de controle centrados nos vértices e onde a heterogeneidade do meio está associada aos elementos da malha é apresentada e comparada com a formulação EBFV1 para tratar problemas elípticos em malhas 2-D. Exemplos obtidos da literatura e tratados por outras formulações de volumes finitos são usados para avaliar o erro e a taxa de convergência para malhas com diferentes níveis de refinamento. Mostram-se como as duas formulações são capazes de lidar com problemas envolvendo meios porosos com razão de heterogeneidade e anisotropia elevada.

Palavras-chave: Reservatórios de petróleo, Métodos dos volumes finitos, Programação orientada a objetos, C++, Computação paralela.

Abstract

High performance techniques has been applied to petroleum reservoir simulations by engineers and researches since the advent of electronic computers in the 1950's. The requirement to forecast the reservoir behaving along its production life obtaining the maximum of hydrocarbons includes a detailed knowledgement of the fluids (liquids and gases) and the rocks characteristics. Mathematical modeling of fluid flow in porous media results in non-linear partial differential equations that in general cannot be solved by analical means.

Numerical methods like finite difference and finite volume method, and more recently the finite element method, have been applied in petroleum reservoir field. A special attention must be given to such methods when they are used in simulations with parallel computers since the pre-processors stage, passing through the data structure up to the solver of the system of equations.

Parallel computers with distributed memory are presented as a high performance tool that helps to decrease the time of simulations or to allow large scale simulations that could not be possible using a single machine due the large amount of data.

In this work is presented a parallel petroleum reservoir simulator written in C++ and making use of open source parallel libraries such as FMDB (Flexible Distributed Mesh Data Base), ParMetis and PETSc (Portable Extensible Toolkit for Scientific computation) performing specialized tasks like distributed mesh management, mesh partition and solving of system of equations.

A vertex centered finite volume formulation using an edge-based data structure (Carvalho, 2005) called EBFV1 (Edge Based Finite Volume) is used to discretize the incompressible two-phase oil-water flow equations in heterogeneous and anisotropic media using triangle and tetrahedral elements meshes. Heterogeneity is associated to subdomains. A modified IMPES method of time integration is adopted with an adaptive procedure in the time step (Hurtado et al., 2006).

A second vertex centered finite volume formulation (EBFV2) using an edge-based data structure is also presented but associating the porous media heterogeneity to the mesh elements. The formulation is compared to the EBFV1, to others numerical formulations and to the analytical solutions of 2-D elliptic problems taken from literature. Acuracy and convergence ratio for meshes with different level of refinement are evaluated to show how both formulation can deal with porous media with high heterogeneity and anisotropic ratio.

Keywords: Petroleum Reservoir, Finite volume method, Object-oriented programming, C++, Parallel computing.

Sumário

LISTA DE SÍMBOLOS

A_K :Área da superfície de controle

$\vec{C}_{IJ_L}$: Vetor área normal a superfície de controle associada à aresta IJ_L no interior do domínio

$\vec{C}_{IJ_L}^{\Omega_r}$: Vetor área normal da superfície de controle associada à aresta IJ_L para o interior do subdomínio Ω_r

$\vec{D}_{IJ_L}$: Vetor área normal da superfície de controle associada à aresta IJ_L no contorno do domínio

$\vec{D}_{IJ_L}^{\Omega_r}$: Vetor área normal da superfície de controle associada à aresta IJ_L para o contorno do domínio Ω_r

$\vec{D}_{IJ_LJ_H}$: Vetor área normal da superfície de controle triangular no contorno, formada pelos nós I, J_L e J_H

$\|E\|$: Norma do erro assintótico de truncamento

f_i : Fluxo fracional da fase i

$\vec{F}$: Fluxo advectivo exato

$F_{IJ_L}^{\Omega_r}$: Fluxo advectivo aproximado associado a uma aresta IJ_L

g : Módulo da aceleração gravitacional

i : Fase i = óleo (o) ou água (w)

I, J_L : Nós genéricos da malha, através dos quais são definidos os volumes de controle

$\underset{\sim}{I}$: Tensor Identidade

IJ_L : Aresta formada pelos nós I e J_L

K *:* Tensor permeabilidade absoluta do meio poroso

k_{ri} : Permeabilidade relativa da fase i

m, n : Parâmetros empíricos para o modelo de Van Genuchten

$\vec{n}$: Vetor área normal

$\vec{n}_k$: Vetor unitário normal a área $k\,A$

NN : Número de nós vizinhos conectados ao um nó I através de uma aresta IJ_L

p_c : Pressão capilar

p_d : Parâmetro empírico para o modelo de Brooks e Corey

p_i : Pressão da fase *i*

p_m : Pressão média das fases

q_i : Termos de fonte ou sumidouro (i.e. poços)

Q : Vazão volumétrica específica total

Q_i : Vazão volumétrica específica da fase *i*

Q_I : Vazão volumétrica total no poço injetor

Q_P : Vazão volumétrica total no poço produtor

S_i : Saturação da fase i

S_{ro} : Saturação residual de óleo

S_{rw} : Saturação residual de água

$\vec{v}$: Velocidade total do fluido ou velocidade de transporte advectivo

$\vec{v}_i$: Vetor velocidade da fase *i*

V_I : Volume de controle associado a um nó *I*

t : Instante de tempo

u : Variável escalar genérica (*i.e*. temperatura, pressão, concentração, etc.)

u_I : Valor exato da variável escalar *u* num ponto *I*

$\hat{u}_I$: Valor aproximado da variável escalar *u* num ponto *I*

u_{IJ_L} : Valor exato da variável escalar *u* no ponto médio de uma aresta IJ_L

$\hat{u}_{IJ_L}$: Valor aproximado da variável escalar *u* no ponto médio de uma aresta IJ_L

$u_{IJ_LJ_H}$: Valor aproximado da variável escalar *u* associado a uma superfície de controle triangular formada pelos nós *I*, J_L e J_H

$\nabla\hat{u}_I$: Gradiente aproximado da variável escalar *u* no nó *I*

$\nabla\hat{u}_{IJ_L}$: Gradiente aproximado da variável escalar *u* no ponto médio da aresta IJ_L obtido por volumes finitos repetidos

$\nabla u_{IJ_L}^{\Omega_r(N)}$: Componente do gradiente aproximado da variável escalar *u* normal a aresta IJ_L no ponto médio da mesma, obtido por volumes finitos repetidos

$\nabla u_{IJ_L}^{\Omega_r(\mathrm{P})}$: Componente do gradiente aproximado da variável escalar *u* paralelo a aresta IJ_L no ponto médio da mesma, obtido por volumes finitos repetidos

Símbolos Gregos

α : Parâmetro empírico para o modelo de Van Genuchten

ϕ : Porosidade

λ_i : Mobilidade da fase *i*

$\lambda_I^{IJ_L}$: Termo de mobilidade total avaliado no ponto médio da aresta IJ_L

Γ_I : Superfície de controle associada a um volume de controle Ω_I

Γ : Contorno ou fronteira

μ_i : Viscosidade da fase *i*

Ω : Domínio

ω : Parâmetros empíricos para o modelo de Brooks e Corey

ρ_i : Densidade ou massa específica da fase *i*

$\left|\Delta_{IJ_L}\right|$:Comprimento de uma aresta IJ_L

Matrizes de Aresta

$\left[E\right]_{(2x2\text{ndim})}^{IJ_L(\Omega_r)}$: Matriz da aresta IJ_L responsável pela projeção do gradiente numa direção normal a aresta

$\left[F\right]_{(2\text{ndim}x2)}^{IJ_L(\Omega_r)}$: Matriz da aresta IJ_L para o cálculo do gradiente nodal pelo MVFA para meios homogêneos

$\left[G\right]_{(2x2)}^{IJ_L(\Omega_r)}$: Matriz de aresta IJ_L para a discretização do termo elíptico que contém o termo de projeção do gradiente numa direção paralela a aresta

LISTA DE ACRÔNIMOS

API	: Application Programming Interface
ASM	: Additive Schwarz Method
BiCG	: Gradiente Bi-Conjugado
BiCG	: Gradiente Bi-Conjugado Estabilizado
Blas	: Basic Linear Algebra Subprograms
CAF	: Co-Array Fortran
CPU	: Central Processing Unit
CG	: Gradiente Conjugado
CGNR	: Conjugate Gradient Normal Residual
DSM	: Distributed Shared-Memory
EBFV1	: Edge Based Finite Volume 1
EBFV2	: Edge Based Finite Volume 2
FMDB	: Flexible Distributed Mesh Database
GMRES	: Generalized Minimum RESidual
HPC	: High Performance Computing
HPVM	: High Performance Virtual Machines
LAPACK	: Linear Algebra PACKage
Mflops	: Milhões de operações de ponto flutuante por segundo
MPI	: Message Passing Interface
MIMD	: Multiple – Instruction, Multiple – Data stream
MISD	: Multiple-instruction, Single-Data
NOW	: Network of Workstations
PCR	: Preconditioned Conjugate Gradient
PETSc	: Portable, Extensible, Toolkit for Scientific Computation
PRS	: Petroleum Reservoir Simulator
QMR	: Quasi-Minimal Residual
SIMD	: Single – Instruction, Multiple – Data stream
SISD	: Single-Instruction, Single-Data
STL	: Standard Template Library
UPC	: Unified Parallel C

LISTA DE FIGURAS

LISTA DE TABELAS

1. INTRODUÇÃO

1.1 Motivação e Considerações Gerais

O crescimento econômico de qualquer país do mundo está intimamente ligado ao aumento do consumo de energia. Apesar dos constantes apelos de organizações ambientais pela redução da emissão de gases que causam o efeito estufa, o uso de combustíveis fósseis, em particular o petróleo, ainda é a principal fonte de energia de muitos países. A demanda por petróleo e gás natural no mundo se acentuou nas últimas décadas com o crescimento econômico dos chamados países emergentes, entre eles o Brasil.

A produção de hidrocarbonetos exigida pelo mercado depende do conhecimento de como o reservatório de petróleo se comporta ao longo de sua vida produtiva. Um conhecimento mais detalhado das características físicas e geológicas do mesmo implica em uma maior confiabilidade nas previsões do seu comportamento e na otimização das análises de viabilidade econômica.

A simulação numérica de reservatórios aparece como um instrumento de previsão mais detalhada por permitir o estudo de modelos de reservatórios mais complexos envolvendo, por exemplo, falhas geológicas, variações térmicas, reações químicas e a presença de várias fases e de vários componentes. Dessa forma, pode-se definir a melhor estratégia de produção economizando recursos e maximizando os lucros da empresa petrolífera. Outras técnicas, por sua vez, mais simples de serem empregadas são o método baseado na equação de balanço de materiais, restrito a poucos casos, e a análise de curvas de declínio, largamente empregada, por exemplo, em casos onde há uma escassez de dados (Rosa et al., 2006).

Por permitir uma analise mais detalhada do campo em estudo, a simulação de reservatórios, pode trazer diversas vantagens como:

- Fazer estimativas de reservas;
- Minimizar incertezas;
- Avaliar a eficácia de métodos de recuperação secundária e escolher a melhor técnica de produção e a que taxa;
- Otimizar as estratégias de drenagem definindo quantos e onde os poços devem ser posicionados ao longo de sua vida produtiva;

- Analisar se os hidrocarbonetos devem ser recuperados por depleção natural ou por algum processo de recuperação melhorada.

A modelagem matemática de fluxo em meios porosos se baseia, em geral, na equação de conservação da massa e na lei de Darcy cujo resultado é um conjunto de equações diferenciais parciais não-lineares, que na grande maioria das vezes não podem ser resolvidas analiticamente. Um parâmetro de extrema importância presente nas equações de fluxo é a permeabilidade da rocha. Seu conhecimento é importante para qualquer exploração de reservatórios e pode sofrer mudanças abruptas de várias ordens de grandeza em um curto espaço e ao longo do tempo (Crumpton et al., 1995; Verma, 1996; Geiger et al., 2004).

Outra característica que faz da rocha reservatório um meio poroso complexo de ser estudado é a possível presença de várias camadas sedimentares depositadas de maneias diferentes criando direções preferenciais ao fluxo de fluidos, fazendo do reservatório um meio anisotrópico. Além dessas dificuldades, a geometria do reservatório com suas falhas e camadas selantes faz o uso dos métodos numéricos a alternativa mais apropriada para a obtenção de dados que possam determinar de forma aproximada, por exemplo, a melhor distribuição de poços para a produção dos hidrocarbonetos.

A idéia básica de qualquer método aproximado é substituir o problema original por outro mais fácil de ser resolvido e cuja solução é, de alguma forma, próxima da solução do problema original (Aziz and Settari, 1979). O método de diferenças finitas surgiu no início da década de 1950 e logo foi aplicado ao estudo de fluxo de fluidos em meios porosos. Ainda hoje é bastante usado no meio acadêmico e em vários simuladores numéricos comerciais (IMEX e ECLIPSE) e livres (BOAST). Tal interesse se deve em parte ao relativo baixo consumo de memória e utilização de CPU.

O crescimento do poder de processamento dos computadores tem permitido aos engenheiros simular problemas grandes e complexos de reservatórios e aplicar novas técnicas de simulação. Como conseqüência vem tornando-se possível lidar de forma mais acurada com as heterogeneidades, anisotropias e demais características dos processos. Ao mesmo tempo, engenheiros e cientistas procuram meios de reduzir o custo de simulação desenvolvendo *solvers* lineares e pré-condicionadores mais robustos, técnicas adaptativas e estruturas de dados alternativas.

O desenvolvimento da capacidade de processamento dos computadores está concentrado em quatro áreas: velocidade das CPUs, memória, computação paralela e linguagens de programação (Cao, 2002). Cada uma delas contribui para o desenvolvimento da

simulação ao seu modo. A velocidade das CPU cresceu 2000 vezes desde a década de 1970 até meados da década de 90 e continua crescendo, porém em um ritmo mais lento. O aumento na capacidade das memórias saltou de megabyte para gigabytes permitindo realizar simulações maiores e mais completas. A computação de alto desempenho é alcançada pelo uso de múltiplos processadores trabalhando em paralelo na solução do mesmo problema. As linguagens orientadas a objetos tais como C++ facilitam o projeto e o desenvolvimento de programas de computador grandes e complexos. O uso de ferramentas modernas torna os programas mais fáceis de serem mantidos.

O desenvolvimento de computadores paralelos tornou possível conduzir simulações de grande porte em reservatórios de petróleo. Na última década, o número total de blocos usados em simulações típicas de reservatório cresceu de milhares para milhões (Chen et al., 2006). A computação paralela pode reduzir o tempo de computação do código de simulação por um fator proporcional ao número de processadores, entretanto o desenvolvimento do código se torna mais complexo.

Uma das mais populares arquiteturas de computação paralela é a de memória distribuída que pode ser constituída de centenas a milhares de processadores, onde cada um possui uma memória local e o fluxo de dados independe de outros processadores. Na arquitetura de memória compartilhada, o mesmo endereço global de memória está disponível para todos os processadores através de um caminho comum. Como não requer passagem explícita de mensagens entre processadores, o desenvolvimento de aplicações é geralmente mais fácil quando comparada à arquitetura de memória distribuída.

Técnicas de computação paralela para simulação numérica de reservatórios de petróleo têm se tornado uma área de grande interesse de pesquisa e da indústria. Nos últimos anos, companhias petrolíferas, companhias de serviço, universidades e institutos de pesquisa têm empregado técnicas de processamento paralelo para reduzir custos de produção e melhorar a eficiência de trabalhos (Han, 2005).

O uso de clusters como um sistema de computadores paralelos foi primeiro desenvolvido na década de 1960 pela IBM como uma alternativa a interconexão de grandes mainframes para fornecer uma forma de paralelismo a um custo mais atrativo (Buyya, 1999). Esses consistem de um conjunto de computadores interconectados trabalhando com um único sistema integrado (Pfister, 1998).

A computação com clusters ganhou força com o surgimento de microprocessadores de alto desempenho, redes de alta velocidade e ferramentas padronizadas para computação distribuída de alto desempenho. Os recentes avanços nessas tecnologias e suas disponibili-

dades e baixos preços têm feito dos clusters uma solução muito atrativa para a computação paralela. A tendência da computação paralela é se afastar dos tradicionais supercomputadores especializados tais como o Cray/SGI T3E para sistemas mais baratos e de uso geral consistindo de PCs de um ou múltiplos processadores (Yeo et al, 2006), pois permite que se construa uma plataforma para um dado orçamento apropriado a um grande grupo de aplicações.

O surgimento de plataformas de clusters foi conduzido por muitos projetos acadêmicos tais como, Beowulf, Berkeley NOW (Network of Workstations), and HPVM (High Performance Virtual Machines) que apresentam as vantagens dos clusters sobre todas as outras plataformas tradicionais. Essas vantagens incluem obter o alto desempenho de supercomputadores a baixo custo, possibilidade de incorporar novas tecnologias em um sistema de fácil atualização, uso de plataformas de desenvolvimento de código aberto e não restrição a um único vendedor. Hoje, os clusters são largamente usados em pesquisas e desenvolvimento de aplicações em ciências, engenharias, comércio e indústria que demandam computação de alto desempenho.

Os modelos de programação paralela em clusters têm sido tradicionalmente divididos em categorias baseadas na relação de programas com os dados que os programas operam (Flynn, 1972). O modelo SISD (Single-Instruction, Single-Data) que define o tradicional computador de von Neumann, o modelo MIMD (Multiple-Instruction, Multiple-Data) que inclui a maioria dos clusters de hoje. Os modelos SIMD (Single-Instruction, Multiple-Data) e MISD (Multiple-instruction, Single-Data) completam a divisão.

Em simulação numérica com computadores paralelos, o uso de softwares especializados torna o desenvolvimento de aplicações mais rápido e menos propício a erros. Muitos centros de pesquisas têm investido no desenvolvimento de ferramentas de alto desempenho para os mais diversos fins. ParMetis, Jostle e Zoltan aparecem como opções para distribuição e balanço de carga de malhas discretas. Blas (Basic Linear Algebra Subprograms) e LAPACK (Linear Algebra PACKage) fornecem rotinas eficientes para problemas de álgebra linear com vetores e matrizes cheias. O Petsc (Portable, Extensible Toolkit for Scientific Computation), que faz uso dos dois últimos, fornece várias opções de resolvedores (*solvers*), pré-condicionadores e interface de alto nível para a manipulação de matrizes e vetores distribuídos.

Todos esses pacotes trabalham sobre uma API (Application Programming Interface) que se encarrega do trabalho de envio e recebimento de mensagens entre os computa-

dores. As interfaces mais conhecidas são: o PVM (Parallel Virtual Machine) e o MPI (Message Programming Interface), amplamente usado em computação paralela.

1.2 Objetivos e Contribuições do Presente Trabalho

No presente trabalho, foi desenvolvido um simulador de reservatórios de petróleo em linguagem de programação C++ para fluxo bifásico óleo-água em meios porosos heterogêneos e anisotrópicos 2-D/3-D usando computadores paralelos de memória distribuída.

A contribuição deste trabalho se dá como continuação do trabalho efetuado por Carvalho (Carvalho, 2005) onde uma formulação conservativa de volumes finitos foi desenvolvida para resolver as equações diferenciais parciais que surgem da modelagem do escoamento de fluidos em meios porosos. Nesta formulação, baseada em uma estrutura de dados por aresta, os volumes de controle, obtidos pelo método das medianas, são centrados nos vértices da malha primal e outros coeficientes geométricos da formulação estão associados às arestas e às faces (no caso 3-D) da malha primal.

O objetivo principal foi migrar do ambiente MATLAB, onde todo o estudo de Carvalho foi desenvolvido, para um ambiente que permitisse construir um simulador capaz de lidar com problemas de grande porte em computadores paralelos de memória distribuída e que técnicas de alto desempenho pudessem ser aplicadas de uma forma mais eficaz. Tal ambiente foi criado em linguagem C++ seguindo os paradigmas da programação orientada a objetos, tendo assim, um programa mais fácil de ser administrado e modificado.

Como técnicas de alto desempenho adotadas, podem-se citar alguns como o uso da estrutura de dados por aresta que oferece ganhos computacionais significativos em termos de uso de memória e uso de CPU (Löhner, 1994), (Lyra, 1994), (Löhner, 2001), (Rees, 2004), (Lewis e Malan, 2005). Uso de uma linguagem compilada como o C++ que oferece um ganho substancial de uso de CPU comparado a uma linguagem interpretada como o MATLAB. A aplicação do método IMPES (Implicit Pressure Explicit Saturation) adaptativo, explorado no presente trabalho, tende a reduzir o tempo total de simulação reduzindo o número de cálculos a cada passo de tempo. O uso de computadores paralelos com memória distribuída permite dividir o trabalho em dois ou mais sub-trabalhos entre os processadores envolvidos reduzindo ainda mais o tempo de simulação.

Ao longo deste trabalho, também foi desenvolvida outra formulação de volumes finitos, com volumes de controle definidos sobre uma malha dual e que é análoga ao tradi-

cional método de volumes finitos baseado em elementos (CVFEM). Essa formulação foi desenvolvida apenas para casos 2-D e seu código paralelizado para resolver apenas problemas elípticos. Essa formulação possibilita tratar problemas em meios porosos altamente heterogêneos e exige muito menos operações de ponto flutuante para a montagem do sistema de equações.

Uma ferramenta de grande importância que marcou o início deste trabalho foi o desenvolvimento de um pré-processador paralelo 2-D/3-D para malhas não-estruturadas. As informações produzidas por esta ferramenta podem ser armazenadas em arquivos e usadas em sucessivas simulações mudando apenas as variáveis de entrada do problema. O pré-processador paralelo traz a vantagem de economizar tempo de processamento e comunicação entre processadores em casos de simulações de grande porte.

Vale ainda mencionar que, em paralelo ao trabalho aqui desenvolvido, foi também construída, na mesma filosofia do simulador, uma ferramenta para adaptação de malhas 2-D para problemas seriais (um processador apenas) envolvendo outros membros do grupo de trabalho (PADMEC) ao qual pertencem os professores orientadores. Esta ferramenta de adaptação de malhas, cujo trabalho está em andamento, será estendida ainda para problemas envolvendo malhas 3-D em múltiplos processadores.

1.3 Organização Geral da Tese

O presente texto foi dividido em sete capítulos mais a bibliografia e uma lista de três apêndices.

No presente capítulo, uma introdução é feita visando contextualizar os temas abordados no trabalho, mostrando de forma sucinta a importância da simulação numérica e do uso da computação de alto desempenho no estudo da exploração de reservatórios de petróleo, assim como os principais objetivos e contribuições da tese.

No capítulo dois, são apresentadas as equações que descrevem o fluxo bifásico de água e óleo em meios porosos.

No capítulo três, o método dos volumes finitos por aresta (MVFA) em malhas não-estruturadas é descrito resumidamente, incluindo a discretização conservativa de termos elípticos em meios heterogêneos e anisotrópicos. Esta formulação é comparada com outra formulação análoga ao CVFEM e que foi desenvolvida durante o trabalho e descrita em detalhes no apêndice A. Neste capítulo, são comparados acurácia e taxa de convergência

entre ambas e resultados analíticos e numéricos de problemas modelos (benchmarks) encontrados na literatura.

No capítulo quatro, é introduzido o método IMPES e sua variante adaptativa no contexto da computação de alto desempenho. Resultados de acurácia e desempenho são confrontados entre os dois métodos. Também é apresentado, porém de forma muito breve, os métodos Seqüencial Implícito e Totalmente Implícito que não foram explorados neste trabalho.

No capítulo cinco, é descrita em detalhes a paralelização do código do simulador, o uso da programação orientada a objetos, o uso dos pacotes de código aberto pelo programa. Por fim são apresentadas análises de desempenho paralelo.

No capítulo seis, são relatadas as principais conclusões, onde as contribuições do trabalho são listadas e são indicadas as direções para futuras pesquisas.

No apêndice A, são apresentados os passos de discretização da equação de difusão baseado na formulação EBFV2. Os passos mostram, para uma malha 2-D de três elementos, como a matriz global é montada por arestas.

Por fim, é apresentada a bibliografia utilizada durante o desenvolvimento do presente trabalho

2. EQUAÇÕES GOVERNANTES

2.1 Introdução

As equações fundamentais que regem o problema relacionado ao escoamento bifásico óleo-água em meios porosos são abordadas neste capítulo apresentando as formulações mais gerais para o escoamento no interior de rochas.

As equações diferenciais parciais (EDPs) resultantes da modelagem matemática do fluxo de massa podem ser classificadas como: parabólicas, hiperbólicas ou elípticas (Tannehill et al., 1997; Fortuna, 2000). Esta classificação junto com o conhecimento das propriedades físicas e matemáticas das equações tratadas aqui é de grande importância para construir esquemas numéricos eficientes e acurados.

A interpretação física destas equações pode levar a outra classificação: difusão, advecção ou advecção-difusão (Crank, 1973; Fortuna, 2000).

As equações de difusão estão associadas a diversos problemas físicos em que efeitos dissipativos são importantes, tais como, problemas de condução de calor em sólidos e determinação do campo de pressões em escoamentos mono ou multifásicos.

As equações de advecção ou convecção estão associadas a problemas de transporte de ondas idealizados sem amortecimento, característico, por exemplo, da dinâmica dos gases.

As equações de advecção-difusão envolvem o transporte de informação com amortecimento estando presentes no estudo de uma ampla gama de problemas que vão desde problemas da biomecânica computacional ao transporte de contaminantes em meios porosos. De maneira geral, pode-se fazer a seguinte associação:

Tipo de Equação	**Problema Físico**
Elípticas e parabólicas	Difusão
Hiperbólicas	Convecção ou Advecção
Caráter parabólico	Advecção-Difusão (difusivo dominante)
Caráter hiperbólico	Advecção-Difusão (advectivo dominante)

Maiores detalhes sobre a classificação das EDPs podem ser encontrados nos trabalhos de Hirsch (1988), Lyra (1994), Tannehill et al. (1997) e Fortuna (2000).

2.2 Escoamentos Bifásicos e Imiscíveis em Meios Porosos

2.2.1 Conceitos Fundamentais

As equações que modelam os escoamentos bifásicos óleo-água em meios porosos são baseadas, em geral, na equação da continuidade (equação diferencial de conservação da massa) e na equação de momento. A equação de momento empregada no escoamento em meios porosos é conhecida como Lei de Darcy. Essa lei, obtida experimentalmente pela primeira vez por Henry Darcy em 1856, pode ser vista como uma lei experimental, cujo domínio de validade se restringe a escoamentos laminares em meios porosos rígidos (bear, 1972; Helmig, 1997).

O resultado dessa modelagem matemática é um conjunto de equações diferenciais parciais não-lineares devido à dependência que as propriedades físicas presente nas equações como permeabilidade relativa da rocha e viscosidade e densidade dos fluidos têm com a saturação e a pressão das fases.

Essas equações podem ser resolvidas de forma simultânea, onde a Lei de Darcy é substituída diretamente na equação de conservação de massa para cada fase produzindo um conjunto de EDPs parabólicas, ou de forma segregada, onde a equação de pressão, de caráter parabólico quase elíptico, é resolvida separada da equação de saturação, uma equação de advecção-difusão, não-linear, de caráter parabólico quase hiperbólico. O acoplamento entre estas equações é feito através de um termo de velocidade total ou de fluxo total dependendo de como são escritas estas equações.

No presente trabalho, será apresentada detalhadamente a formulação segregada proposta inicialmente por Peaceman (1977) para o escoamento imiscível de óleo e água em reservatórios de petróleo.

Antes de definir as equações, são adotadas algumas hipóteses simplificadoras:

- O meio poroso está totalmente saturado pelas fases líquidas;
- O fluido e a rocha são incompressíveis;

- Escoamento imiscível;
- Escoamento isotérmico;
- Os fluidos obedecem à lei de Darcy generalizada (Helmig, 1997);

Considerando as hipóteses simplificadoras acima, a Lei de Darcy generalizada para uma fase i pode ser escrita conforme a seguir:

$$\vec{v}_i = -\underset{\sim}{K}\lambda_i \left(\nabla p_i - \rho_i g \nabla Z\right). \tag{2.1}$$

Na Eq. (2.1), v_i é a velocidade aparente da fase *i*, $\underset{\sim}{K}$ é o tensor de permeabilidade absoluta do meio, representando uma propriedade apenas da rocha, assim como a porosidade que representa a razão entre o volume de vazios que pode ser ocupado por um fluido e o volume total da rocha, p_i é a pressão da fase *i*, ρ_i é a massa específica ou a "densidade" da fase *i*, *g* é o módulo da aceleração gravitacional, *Z* é o componente do vetor deslocamento orientado para baixo e $\lambda_i = k_{ri}/\mu_i$ é a mobilidade da fase *i*, onde k_{ri} e μ_i são a permeabilidade relativa e a viscosidade da fase *i*, respectivamente. O tensor de permeabilidade absoluta do meio poroso $\underset{\sim}{K}$ é definido em coordenadas cartesianas como:

$$\underset{\sim}{K} = \begin{pmatrix} K_{xx} & K_{xy} & K_{xz} \\ K_{yx} & K_{yy} & K_{yz} \\ K_{zx} & K_{zy} & K_{zz} \end{pmatrix}. \tag{2.2}$$

Por sua vez, a equação de conservação da massa para cada uma das fases, pode ser escrita como (Bear 1972; Helmig, 1997; Kovarik, 2000):

$$-\nabla \cdot \left(\rho_i \vec{v}_i\right) + q_i = \frac{\partial\left(\phi \rho_i S_i\right)}{\partial t}. \tag{2.3}$$

Na Equação (2.3), ϕ é a porosidade, q_i denota termos fonte ou sumidouros (*i.e.*, poços), e S_i é a saturação da fase *i*, que representa a percentagem do volume poroso ocupado por esta fase. Pela primeira hipótese simplificadora adotada, pode-se escrever a equação constitutiva ou de restrição das saturações, como:

$$\sum_{i=o,w} S_i = 1 \text{ ou ainda } S_o + S_w = 1, \tag{2.4}$$

onde *o, w* representam, respectivamente, óleo e água.

A pressão capilar é definida como a diferença entre as pressões das fases não-molhante (óleo) pela molhante (água), ou seja:

$$p_c = p_o - p_w. \tag{2.5}$$

A pressão capilar existe sempre que os poros estão saturados com duas ou mais fases. Ela é uma função da saturação $p_c = p_c\left(S_w\right)$ e do histórico de saturação (drenagem e embebição) para uma dada rocha e fluidos a uma temperatura e composição constante. Dois modelos empíricos clássicos utilizados para relacionar pressão capilar e saturação da água são o modelo de Brooks e Corey, Eq.(2.6), e o modelo de Van Genuchten, Eq. (2.7).

$$p_c\left(S_w\right) = p_d \left(\frac{S_w - S_{wr}}{1 - S_{wr}} \right)^{-\frac{1}{\omega}}, \tag{2.6}$$

$$p_c\left(S_w\right) = \frac{1}{\alpha} \left[\left(\frac{S_w - S_{wr}}{1 - S_{wr}} \right)^{-\frac{1}{m}} - 1 \right]^{\frac{1}{n}}, \tag{2.7}$$

onde S_{wr} é a saturação residual da água. Para maiores detalhes sobre ambos os modelos, e particularmente sobre os coeficientes empíricos p_d, ω, α, m e n, consultar Helmig (1997).

2.2.2 Equação de Pressão

Eliminando as derivadas da saturação da Eq (2.3), obtém-se a equação da pressão de modo a resolver uma equação diferencial parcial com apenas uma variável. Partindo da Eq (2.3), escrita para cada fase *i = o, w* e expandindo as derivadas no tempo fica:

$$-\nabla\cdot\left(\rho_o\vec{v}_o\right)+q_o=\rho_o S_o\frac{\partial\phi}{\partial t}+\phi\rho_o\frac{\partial S_o}{\partial t}, \tag{2.8}$$

$$-\nabla\cdot\left(\rho_w\vec{v}_w\right)+q_w=\rho_w S_w\frac{\partial\phi}{\partial t}+\phi\rho_w\frac{\partial S_w}{\partial t}. \tag{2.9}$$

Onde ρ_o e ρ_w são constantes uma vez que os fluidos foram considerados incompressíveis. Dividindo a Eq. (2.8) por ρ_o, e a Eq. (2.9) por ρ_w, e utilizando a segunda hipótese simplificadora, isto é, fluido e rocha são incompressíveis:

$$-\nabla\cdot\vec{v}_o+Q_o=\phi\frac{\partial S_o}{\partial t}, \tag{2.10}$$

$$-\nabla\cdot\vec{v}_w+Q_w=\phi\frac{\partial S_w}{\partial t}, \tag{2.11}$$

Onde $Q_i=q_i/\rho_i$ é a vazão volumétrica específica da fase *i*. Somando as Eqs. (2.10) e (2.11), obtém-se:

$$-\nabla\cdot\left(\vec{v}_o+\vec{v}_w\right)+\left(Q_o+Q_w\right)=\phi\left(\frac{\partial S_o}{\partial t}+\frac{\partial S_w}{\partial t}\right), \tag{2.12}$$

ou ainda:

$$-\nabla\cdot\left(\vec{v}_o+\vec{v}_w\right)+\left(Q_o+Q_w\right)=\phi\frac{\partial\left(S_o+S_w\right)}{\partial t}. \tag{2.13}$$

Usando a Eq. (2.4), e rearranjando os termos, obtém-se em termos da velocidade total:

$$\nabla\cdot\vec{v}=Q. \tag{2.14}$$

Na Eq. (2.14), $\vec{v}=\vec{v}_o+\vec{v}_w$ é a velocidade total do fluido e $Q=Q_o+Q_w$ é a vazão volumétrica específica total. Agora, a lei de Darcy Eq. (2.1) é introduzida na Eq. 2.14 através do termo de velocidade:

$$\nabla \cdot \left(-\underset{\sim}{K} \lambda_o \left(\nabla p_o - \rho_o g \nabla Z \right) - \underset{\sim}{K} \lambda_w \left(\nabla p_w - \rho_w g \nabla Z \right) \right) = Q. \tag{2.15}$$

Definindo a pressão média das fases como:

$$p_m = p = \frac{p_o + p_w}{2}. \tag{2.16}$$

Da equação anterior e da definição da pressão capilar, Eq. (2.5) resulta ainda:

$$p_o = p + \frac{p_c}{2} \text{ e } p_w = p - \frac{p_c}{2}. \tag{2.17}$$

Reescrevendo a Eq. (2.15) em função da pressão média das fases e da pressão capilar, fica:

$$\nabla \cdot \left(-\underset{\sim}{K} \left(\lambda_o \nabla \left(p + \frac{p_c}{2} \right) + \lambda_w \nabla \left(p - \frac{p_c}{2} \right) - \left(\lambda_o \rho_o + \lambda_w \rho_w \right) g \nabla Z \right) \right) = Q. \tag{2.18}$$

Rearranjando os termos pode-se escrever:

$$\nabla \cdot \left(-\underset{\sim}{K} \left(\left(\lambda_o + \lambda_w \right) \nabla p + \frac{\left(\lambda_o - \lambda_w \right)}{2} \nabla p_c - \left(\lambda_o \rho_o + \lambda_w \rho_w \right) g \nabla Z \right) \right) = Q. \tag{2.19}$$

Como se pode facilmente observar, a equação de pressão, no caso incompressível, é uma equação elíptica ou de difusão (Ewing, 1983). Devido às heterogeneidades ao longo de reservatórios típicos, o coeficiente $\underset{\sim}{K}$ é, em geral, variável e não-isotrópico (Ewing, 1983; Verma, 1996; Verma e Aziz, 1996; Edwards 2000). Portanto, a equação de pressão, no caso incompressível, é uma equação elíptica com coeficientes descontínuos.

A partir da Eq.(2.19), a velocidade total é escrita como:

$$\vec{v} = -\underset{\sim}{K} \left(\left(\lambda_o + \lambda_w \right) \nabla p + \frac{\left(\lambda_o - \lambda_w \right)}{2} \nabla p_c - \left(\lambda_o \rho_o + \lambda_w \rho_w \right) g \nabla Z \right), \tag{2.20}$$

ou ainda:

$$\vec{v} = -(\lambda_o + \lambda_w)\,\underset{\sim}{K}\nabla p - \frac{(\lambda_o - \lambda_w)}{2}\,\underset{\sim}{K}\nabla p_c + (\lambda_o \rho_o + \lambda_w \rho_w)\, g \underset{\sim}{K}\nabla Z. \tag{2.21}$$

Na Equação (2.21), observam-se três termos compondo a velocidade total do escoamento, onde estes estão relacionados ao gradiente da pressão média dos fluidos, ao gradiente da pressão capilar e à força gravitacional, respectivamente.

2.2.3 Equação de Saturação

Nesta seção é definida a equação de saturação para a fase água. Utilizando a Eq.(2.1), as velocidades das fases água e óleo podem ser escritas, respectivamente, como:

$$\vec{v}_w = -\underset{\sim}{K}\lambda_w (\nabla p_w - \rho_w g \nabla Z), \tag{2.22}$$

$$\vec{v}_o = -\underset{\sim}{K}\lambda_o (\nabla p_o - \rho_o g \nabla Z). \tag{2.23}$$

Multiplicando a Eq. (2.22) por λ_o e a Eq. (2.23) por λ_w obtem-se:

$$\lambda_o \vec{v}_w = -\underset{\sim}{K}\lambda_w \lambda_o (\nabla p_w - \rho_w g \nabla Z), \tag{2.24}$$

$$\lambda_w \vec{v}_o = -\underset{\sim}{K}\lambda_o \lambda_w (\nabla p_o - \rho_o g \nabla Z). \tag{2.25}$$

Subtraindo a Eq. (2.24) da Eq.(2.25), chega-se a

$$\lambda_w \vec{v}_o - \lambda_o \vec{v}_w = -\underset{\sim}{K}\lambda_o \lambda_w (\nabla p_o - \rho_o g \nabla Z - \nabla p_w + \rho_w g \nabla Z). \tag{2.26}$$

Derivando a Eq. (2.5) e usando-a na Eq. (2.26), tem-se:

$$-\lambda_w \vec{v}_o + \lambda_o \vec{v}_w = +\underset{\sim}{K}\lambda_o \lambda_w \nabla p_c - \underset{\sim}{K}\lambda_o \lambda_w (\rho_o - \rho_w) g \nabla Z. \tag{2.27}$$

Rearranjando os termos e usando ainda que $\vec{v}_o = \vec{v} - \vec{v}_w$, e que $\lambda = \lambda_w + \lambda_o$ pode-se escrever

$$\lambda \vec{v}_w = \lambda_w \vec{v} + \underset{\sim}{K} \lambda_o \lambda_w \left(\nabla p_c + \left(\rho_w - \rho_o \right) g \nabla Z \right). \tag{2.28}$$

Definindo ainda o fluxo fracional de uma fase *i* como $f_i = \lambda_i / \lambda$, e para simplificar a notação definindo $h_w = -\dfrac{(\lambda_o \lambda_w)}{\lambda} \dfrac{dp_c}{dS_w}$, tem-se:

$$\vec{v}_w = f_w \vec{v} - \underset{\sim}{K} h_w \nabla S_w + \underset{\sim}{K} \lambda_o f_w \left(\rho_w - \rho_o \right) g \nabla Z. \tag{2.29}$$

Desta forma conseguimos escrever a velocidade da fase água, $\vec{v}_w$ em função da velocidade total $\vec{v}$. Substituindo a equação anterior na Eq. (2.3), com *i* = *w*, e rearranjando os termos:

$$\frac{\partial \left(\phi \rho_w S_w \right)}{\partial t} = -\nabla \cdot \left(\rho_w \left(f_w \vec{v} - \underset{\sim}{K} h_w \nabla S_w + \underset{\sim}{K} \lambda_o f_w \left(\rho_w - \rho_o \right) g \nabla Z \right) \right) + q_w. \tag{2.30}$$

Utilizando as hipóteses de que o meio poroso é rígido e que os fluidos são incompressíveis pode-se ainda escrever:

$$\phi \rho_w \frac{\partial \left(S_w \right)}{\partial t} = -\rho_w \nabla \cdot \left(\left(f_w \vec{v} - \underset{\sim}{K} h_w \nabla S_w + \underset{\sim}{K} \lambda_o f_w \left(\rho_w - \rho_o \right) g \nabla Z \right) \right) + q_w. \tag{2.31}$$

Finalmente, dividindo por ρ_w e assumindo que $Q_w = q_w / \rho_w$ chega-se a:

$$\phi \frac{\partial S_w}{\partial t} = -\nabla \cdot \left(f_w \vec{v} - \underset{\sim}{K} h_w \nabla S_w + \underset{\sim}{K} \lambda_o f_w \left(\rho_w - \rho_o \right) g \nabla Z \right) + Q_w. \tag{2.32}$$

A Eq. (2.32) é conhecida como a equação de saturação da fase água num meio poroso rígido. Esta equação é similar a uma equação de advecção-difusão-reação não-linear. Em muitos casos, como por exemplo, na vizinhança de poços de injeção ou em partes do reservatório onde o fluido de injeção ainda não chegou, ou de maneira geral, onde a velocidade total for alta, esta equação assume um caráter fortemente advectivo, compartilhando muitas das propriedades das equações hiperbólicas de primeira ordem (Ewing, 1983; Chavent e Jaffre, 1986).

2.2.4 Condições Iniciais e de Contorno

As equações de pressão e saturação apresentadas definem um problema de valor inicial e de contorno cuja solução única exige uma descrição completa do que ocorre nas fronteiras e como o reservatório se apresenta no início da simulação.

A seguir, são apresentadas um conjunto de condições iniciais e de contorno associadas a um reservatório com poços injetores (*I*) e produtores (*P*). Usualmente estes poços são tratados como "fronteiras internas" dos reservatórios e as condições aplicadas a estes poços são chamadas de condições de "contorno internas" (Chavent e Jaffre, 1986; Aziz, 1993; Ertekin et al., 2001).

Possíveis condições de contorno:

- Primeiro tipo (Dirichlet): pressão é especificada nas fronteiras ou nos poços;
- Segundo tipo (Neumann): expressa o fluxo através da fronteira e pode ser usada para especificar taxa de produção em poços ou influxo de um aqüífero,
- Terceiro tipo: combinação dos dois tipos anteriores e podem surgir quando as fronteiras de dois reservatórios coincidem;
- Quarto tipo: conhecida como condição de contorno cíclica e podem surgir em reservatórios em forma de anel (Aziz, 1979).

Para o mesmo reservatório, as condições de contorno e inicial para a equação de saturação podem ser escritas, respectivamente, como:

$$\begin{aligned} &S_w\left(\vec{x},t\right) = \overline{S}_w \qquad \text{em} \quad \Gamma_I \; X\left[0,T\right], \\ &\underset{\sim}{K} h_w \nabla S_w \cdot \vec{n} = 0 \quad \text{em} \quad \Gamma_N \; X\left[0,T\right]. \\ &S_w\left(\vec{x},0\right) = \overline{S}_w^0 \quad \text{em} \quad \Omega. \end{aligned} \tag{2.33}$$

Onde Γ_I e Γ_N representam as fronteiras de valores de saturação e a vazão prescrita, respectivamente, e $\vec{n}$ representa o vetor normal às fronteiras do domínio Ω.

3. MÉTODO DE VOLUMES FINITOS POR ARESTA

3.1 Introdução

A simulação numérica de fluxo de massa em problemas com meio heterogêneo e anisotrópico representa um grande desafio do ponto de vista matemático e numérico. Variações abruptas no campo de permeabilidade (coeficiente de difusão) são comuns em simulação de escoamento de fluidos em reservatórios de petróleo e aqüíferos devido à própria natureza do processo de formação geológica das rochas sedimentares.

Nas últimas décadas, muito esforço foi dedicado em métodos numéricos que fazem uso de malhas não-estruturadas, como os tradicionais métodos de elementos finitos (MEF) e método de volumes finitos (MVF), devido principalmente ao fato de que esses métodos permitem uma melhor modelagem de geometrias complexas e porque permitem uma fácil incorporação de procedimentos adaptativos. Ao lidar com leis de conservação, os métodos de volumes finitos são particularmente úteis, pois conservam massa localmente (volume de controle) e globalmente.

Os convencionais métodos de diferenças finitas 2-D de cinco pontos são incapazes de lidar com tensores cheios ou malhas não-ortogonais. Além disso, esses métodos introduzem erros de primeira ordem na aproximação dos termos de fluxo entre materiais descontínuos (Edwards, 2000), tornando-os, assim, inapropriados para simulação de problemas de difusão em meios heterogêneos e anisotrópicos.

Esquemas localmente conservativos tais como o método de elementos finitos baseado em volumes de controle (CVFEM), o método de volumes finitos de fluxo contínuo (FCFV) também conhecido como Multipoint Flux Approximations (MPFA) e o método de elementos finitos mistos (MFEM) foram extensivamente estudados na literatura (Ewing, 1983; Aavatsmark et al., 1998; Edwards, 2000; Klausen e Eigestad, 2004).

No contexto de fluxo de fluidos em meios porosos, o recém desenvolvido FCFV é definido assumindo pressões constantes (pointwise ou continuidade completa) e fluxo normal através das interfaces dos volumes de controle. Apesar do custo computacional associado a estes métodos, particularmente o MFEM, ambos os métodos são capazes de lidar com equações elípticas com tensores cheios em meios heterogêneos usando malhas estruturadas ou não-estruturadas. Vale ainda mencionar que o método de elementos finitos, que

é globalmente conservativo, mas não localmente (no elemento), requer algum tipo de recuperação de fluxo de forma a garantir formalmente conservação local (Loula et al. 1999).

Neste trabalho, são descritas duas formulações de volumes finitos baseadas em uma estrutura de dados por aresta e centradas nos nós. As acurácias de ambas as formulações são comparada entre si para tratar problemas elípticos em meios heterogêneos e anisotrópicos, assim como problemas com coeficientes de difusão descontínuos.

A formulação EBFV1 (edge-based finite volume 1) foi desenvolvida e validada por Carvalho (Carvalho, 2005) em vários casos extraídos da literatura tratando problemas de escoamento em reservatórios de petróleo e de dispersão de contaminantes em aqüíferos em ambiente MATLAB. Pela eficácia comprovada, esta formulação foi empregada na construção do simulador paralelo 2-D/3-D usando programação orientada a objetos em C++ para tratar problemas de grande porte desenvolvido no presente trabalho. A dedução da discretização das equações será feita de uma forma mais direta.

Como o objetivo deste capítulo está voltado à avaliação da acurácia apenas de problemas elípticos, a discretização do termo advectivo da formulação EBFV1 será apresentada no capítulo 5, onde o simulador paralelo é descrito.

A outra formulação, EBFV2 (edge-based finite volume 2), foi desenvolvida de forma a associar a cada elemento da malha uma propriedade física do meio poroso. Isso permite, por exemplo, tratar problemas que apresentam uma grande variação de permeabilidade e porosidade.

O desenvolvimento da formulação EBFV2, que se apresenta como uma das contribuições do presente trabalho, se deu quando este simulador encontrava-se em um estágio muito avançado de desenvolvimento e sua implementação resumiu-se apenas a resolução de problemas elípticos em computadores seqüencial e paralelo em malhas 2-D de elementos triangulares. Em trabalhos futuros, esta formulação irá tratar problemas multifásicos 2-D e 3-D.

Para ambos os métodos, são usados volumes de controle construídos a partir de uma malha dual (Donald's dual) obtida a partir dos pontos médios da malha primal, embora isso não seja obrigatório. As formulações são baseadas em uma estrutura de dados por aresta onde os coeficientes geométricos são associados às arestas e aos nós da malha primal. Essas formulações são capazes de lidar com meios heterogêneos e anisotrópicos usando malhas estruturadas ou não-estruturadas.

O primeiro algoritmo, EBFV1, consiste em uma modificação da aproximação em dois passos de Crumpton baseada nas arestas. O segundo algoritmo, EBFV2, é análogo a

uma implementação baseada em arestas do método de elementos finitos usando volumes de controle (CVFEM). Ambas as formulações incluem termos de difusão cruzada garantindo conservação local mesmo para malhas não-ortogonais e coeficientes descontínuos, mantendo acurácia de segunda ordem para o campo de pressão, e pelo menos, acurácia de primeira ordem para os fluxos em malhas triangulares e malhas ortogonais quadrilaterais, permitindo assim, serem utilizadas tanto na solução de problemas elíptico-parabólicos quanto hiperbólicos que são característicos das leis que regem o fluxo de fluidos em meios porosos.

Por serem centradas nos nós, as formulações EBFV1 e EBFV2 permitem um menor uso de memória quando comparadas a esquemas centrados nos elementos (Luo et al., 1995, Malan, 2002, Rees, 2004 e Rees et al., 2004). Isso favorece a adoção de uma estrutura de dados por aresta, onde uma malha não-estruturada é representada pelas arestas dos elementos e pelos nós e suas coordenadas espaciais. Para malhas 3-D, dependendo do tipo de integração no contorno, as arestas são substituídas por faces (Carvalho, 2005).

Mostra-se que uma economia significativa de memória e no número de operações de ponto flutuante pode ser alcançada, pois evita que certas redundâncias de armazenamento, operações de coleta e de distribuição de dados e operações de ponto flutuante ocorram (Barth, 1994; Löhner, 1994; Löhner, 2001).

A escolha de uma estrutura de dados por aresta exige que uma fase de pré-processamento seja realizada sobre a malha de elementos obtida através do gerador de malhas. No capítulo 5, quando o simulador paralelo for apresentado, esta etapa será melhor descrita.

3.2 Formulação Matemática

No modelo bidimensional, a equação que define um problema elíptico em um meio heterogêneo e anisotrópico pode ser escrita como:

$$\nabla \cdot \left(-\underset{\sim}{K}\left(\vec{x}\right) \nabla u \right) = f\left(\vec{x}\right), \text{ com } \vec{x} = \left(x, y\right) \in \Omega \subset \mathbb{R}^2 \tag{3.1}$$

onde:

$$\underset{\sim}{K}\left(\vec{x}\right) = \underset{\sim}{K} = \begin{pmatrix} K_{xx} & K_{xy} \\ K_{yx} & K_{yy} \end{pmatrix}. \tag{3.2}$$

é uma matriz simétrica que pode ser descontínua nas fronteiras do domínio Ω. De forma a definir formalmente um problema elíptico (Crumpton, 1995), assume-se inicialmente que:

$$K_{xx}K_{yy} \geq K_{xy}^2. \tag{3.3}$$

Integrando a Eq. (3.1) e aplicando o teorema da divergência no lado esquerdo, implica:

$$\int_{\Gamma}\left(-\underset{\sim}{K}\nabla u\right)\cdot\vec{n}\partial\Gamma = \int_{\Omega} f\,\partial\Omega. \tag{3.4}$$

A Eq. (3.4), que é a forma integral da Eq.(3.1), define, por exemplo, o campo de pressões do fluxo bifásico óleo-água em reservatórios de petróleo (Rees, et al., 2004; Carvalho et al., 2007).

3.3 Formulação Numérica

3.3.1 Método de volumes finitos baseados em aresta 1 - (EBFV1)

A presente abordagem foi originalmente usada por Crumpton et al (1997) para a discretização de termos de difusão nas equações de Navier-Stokes com as modificações propostas por Lyra et al (2004), Carvalho et al (2005) e Carvalho et al (2007). Nesta abordagem, de forma a se obter um sistema de equações discretas final, primeiro são calculados os gradientes nodais como funções do campo escalar discreto e então estes gradientes são usados para calcular os termos elípticos em um segundo passo (Crumpton et al, 1997; Rees et al, 2004; Carvalho et al, 2007). Os gradientes calculados no primeiro passo são usados para calcular os termos de difusão cruzadas que surgem naturalmente em problemas que lidam com tensores cheios $\left(\text{i.e. } K_{xy} = K_{yx} \neq 0.0\right)$ ou quando se usa malhas não-ortogonais, onde ambos os problemas podem ser vistos como equivalentes.

Para um nó I arbitrário da malha, usando uma estrutura de dados baseada em arestas, a Eq. (3.4) pode ser aproximada como:

$$\sum_{L_I \in \Omega} \vec{F}^{\Omega}_{IJ_L} \cdot \vec{C}_{IJ_L} + \sum_{L_I \in \Gamma} \vec{F}^{\Gamma}_{IJ_L} \cdot \vec{D}_{IJ_L} = f_I V_I . \tag{3.5}$$

onde $\vec{F}^{\Omega}_{IJ_L} = -\underset{\sim}{K} \nabla u^{\Omega}_{IJ_L}$ é a função de fluxo definida na superfície de controle para uma aresta IJ_L, V_I é o volume de controle associado ao nó *I*, $\nabla u^{\Omega}_{IJ_L}$ é o gradiente aproximado no ponto médio da aresta IJ_L, o índice Ω representa aproximações que estão associadas a todas as arestas IJ_L da malha primal conectadas ao nó I, Γ refere-se apenas às arestas de contorno conectadas a este nó, o somatório $\sum_{L_I \in \Omega}$ é feito sobre todas as arestas conectadas ao nó I e $\sum_{L_I \in \Gamma}$ é feito apenas sobre as arestas do contorno. Os coeficientes geométricos $\vec{C}_{IJ_L}$ e $\vec{D}_{IJ_L}$ são definidos como:

$$\begin{aligned} \vec{C}_{IJ_L} &= \sum_{k=1}^{NT} A_k \vec{n}_k \\ \vec{D}_{IJ_L} &= A_L \vec{n}_L . \end{aligned} \tag{3.6}$$

Na Equação (3.6), A_K e $\vec{n}_K$ representam, respectivamente, a área e o vetor normal à superfície de controle definida por todos os *NT* elementos que dividem a aresta IJ_L. Para o coeficiente $\vec{D}_{IJ_L}$, A_L e $\vec{n}_L$ possuem o mesmo significado que A_K e $\vec{n}_K$, porém para as arestas de contorno. Para problemas 3-D, dois vetores $\vec{D}^0_{IJ_L}$ e $\vec{D}^1_{IJ_L}$ são usados, pois uma aresta pode estar associada a duas superfícies com condições de contorno distintas.

Os vetores $\vec{C}_{IJ_L}$ e $\vec{D}_{IJ_L}$ são obtidos na fase de pré-processamento e associados às arestas da malha primal. Mais detalhes podem ser encontrados em Carvalho (2005).

3.3.1.1 Cálculo do gradiente

A aproximação do gradiente no ponto médio da aresta $\nabla u_{IJ_L}^{\Omega}$ necessário para resolver a equação (3.5) parte da aplicação do teorema da divergência para integrar o gradiente da variável escalar *u* no nó I, obtendo-se:

$$\int_{\Omega_I} \nabla u_I \, \partial\Omega_I = \int_{\Gamma_I} u_I \vec{n} \, \partial\Gamma. \tag{3.7}$$

Assumindo que o gradiente médio $\overline{\nabla} u_I$ no volume de controle *I* é constante, pode-se defini-lo como:

$$\overline{\nabla} u_I = \left(\int_{\Omega_I} \nabla u_I \partial\Omega_I \right) \Big/ V_I \, . \tag{3.8}$$

Similarmente à Eq. (3.5) e fazendo $\nabla u_I = \overline{\nabla} u_I$, a forma discreta da Eq. (3.7) pode ser escrita como:

$$\nabla u_I^{\Omega} V_I = \left[\sum_{L_I \in \Omega} u_{IJ_L}^{\Omega} \vec{C}_{IJ_L} + \sum_{L_I \in \Gamma} u_{IJ_L}^{\Gamma} \vec{D}_{IJ_L} \right]. \tag{3.9}$$

A principal diferença entre a abordagem de Crumpton e a formulação apresentada em Carvalho et al (2007) é que no último, os gradientes são recuperados para cada subdomínio separadamente a fim de evitar inconsistências físicas.

Neste caso, como mostra a Fig. (3.1), para um problema 2-D envolvendo mais de um domínio, um nó situado na interface possui dois volumes de controle com propriedades físicas distintas.

Assim, as definições dos fluxos sobre as superfícies de controle localizados na interface entre materiais diferentes podem ser ambíguas (Helmig, 1997). Se os gradientes nodais calculados como descrito na Eq. (3.9) forem usados para calcular os fluxos, uma inconsistência física seria obtida ao longo das superfícies de controle adjacentes as descontinuidades dos materiais.

De forma a contornar este problema, os gradientes são recuperados para cada um dos subdomínios.

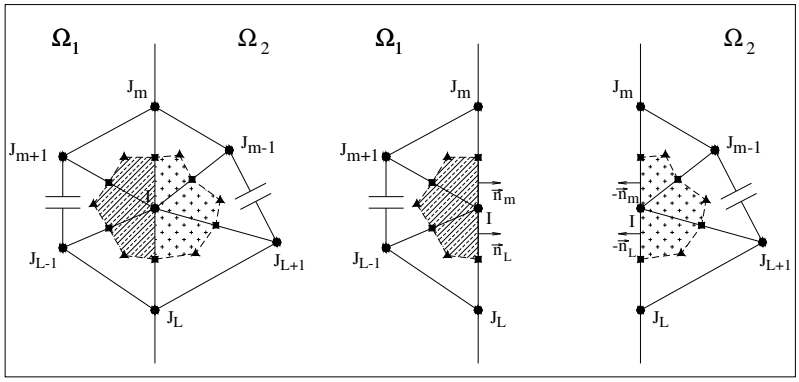


Figura 3.1 – Volume de controle subdividido por dois tipos diferentes de rocha

Para a malha considerada na Fig. (3.1), é necessário incluir novos coeficientes geométricos na interface entre domínios de forma a se definir duas superfícies de controle e obter, assim, um gradiente contínuo por partes.

Dessa forma, para um meio heterogêneo, a Eq. (3.9) pode ser reescrita como:

$$\nabla u_I^{\Omega_r} = \frac{1}{V_I^{\Omega_r}} \left[\sum_{L=1}^{NN^{(\Omega_r)}} u_{IJ_L}^{\Omega_r} \vec{C}_{IJ_L}^{\Omega_r} + \sum_{L=1}^{NN^{(\Gamma_{RI})}} u_{IJ_L}^{\Gamma} \vec{D}_{IJ_L}^{\Omega_r} + \sum_{L=1}^{NN^{(\Gamma_{RE})}} u_{IJ_L}^{\Gamma} \vec{D}_{IJ_L}^{\Omega_r} \right]. \tag{3.10}$$

Na equação (3.10), $\nabla u_I^{\Omega_r}$ e $V_I^{\Omega_R}$ são o gradiente nodal e o volume de controle associados ao nó I referentes ao subdomínio Ω_r. Os coeficientes $\vec{C}_{IJ_L}^{\Omega_r}$ e $\vec{D}_{IJ_L}^{\Omega_r}$ são os coeficientes geométricos da aresta IJ_L referentes ao subdomínio Ω_r. Vale mencionar que na forma de recuperar os gradientes por domínios, $\vec{D}_{IJ_L}^{\Omega_r}$ se refere tanto as arestas externas quanto as das interfaces entre domínios, e Γ_{RE} e Γ_{RI} referem-se, respectivamente, aos laços sobre as arestas de contorno externas e sobre as arestas das interfaces entre domínios.

A Eq. (3.10) acima é construída para cada um dos subdomínios, ou seja, é feito um laço sobre cada subdomínios de forma a garantir que os gradientes nodais e os fluxos sejam aproximados de forma correta para cada material ao longo da interface das arestas. A cada subdomínio pode-se associar uma propriedade física do meio como porosidade e permeabilidade.

Para arestas no interior do domínio, a variável escalar u_{IJ_L} da Eq. (3.10) pode ser aproximada como a média aritmética entre os nós *I* e J_L, que corresponde a uma aproximação de diferenças finitas de segunda ordem.

$$u_{IJ_L}^{\Omega} = \frac{u_I + u_{J_L}}{2}. \tag{3.11}$$

Inspirada no MEF, uma aproximação de segunda ordem da variável u_{IJ_L} nos contornos para malhas 2-D de triângulos e 3-D de tetraedros pode ser obtida, respectivamente, através de:

$$u_{IJ_L}^{\Gamma} = \frac{5u_I + u_{J_L}}{6}, \tag{3.12}$$

$$u_{IJ_L}^{\Gamma} = \frac{6u_I + u_{J_L} + u_{J_K}}{8}. \tag{3.13}$$

Para o caso tridimensional, é necessário ter uma segunda estrutura de dados com as faces dos tetraedros que definem o contorno. Neste caso, o coeficiente $\vec{D}_{IJ_L}$ assume outra notação, $\vec{D}_{IJ_LJ_H}$, que está associado à face do volume de controle sobre o contorno do domínio e é definido como:

$$\vec{D}_{IJ_LJ_H} = \frac{\vec{n}_{IJ_LJ_H} A_{IJ_LJ_H}}{3}. \tag{3.14}$$

onde, $\vec{n}_{IJ_LJ_H}$ e $A_{IJ_LJ_H}$ são o vetor normal e área do triângulo, respectivamente. Dessa forma, o gradiente nodal pode ser calculado para malhas 2-D de triângulos e 3-D de tetraedros como:

$$\nabla u_I^{\Omega_r} = \frac{1}{V_I^{\Omega_r}} \left[\sum_{J_L=1}^{NN^{(\Omega_r)}} \frac{\left(u_I + u_{J_L}\right)}{2} \vec{C}_{IJ_L}^{\Omega_r} + \sum_{J_L=1}^{NN^{(\Gamma_{RE})}} \frac{\left(5u_I + u_{J_L J_H}\right)}{6} \vec{D}_{IJ_L}^{\Omega_r} + \sum_{J_L=1}^{NN^{(\Gamma_{RI})}} \frac{\left(5u_I + u_{J_L J_H}\right)}{6} \vec{D}_{IJ_L}^{\Omega_r} \right]. \tag{3.15}$$

e

$$\nabla u_I^{\Omega_r} = \frac{1}{V_I^{\Omega_r}} \left[\sum_{J_L=1}^{NN^{(\Omega_r)}} \frac{\left(u_I + u_{J_L}\right)}{2} \vec{C}_{IJ_L}^{\Omega_r} + \sum_{J_L=1}^{NN^{(\Gamma_{RE})}} \frac{\left(6u_I + u_{J_L} + u_{J_H}\right)}{8} \vec{D}_{IJ_L J_H}^{\Omega_r} + \sum_{J_L=1}^{NN^{(\Gamma_{RI})}} \frac{\left(6u_I + u_{J_L} + u_{J_H}\right)}{8} \vec{D}_{IJ_L J_H}^{\Omega_r} \right]. \tag{3.16}$$

Definida uma forma de se obter os gradientes nodais e de forma a calcular os fluxos descritos pela Eq. (3.5), utiliza-se um sistema local e ortogonal de coordenadas no qual um eixo está orientado ao longo da direção da aresta (P) e outro eixo em uma direção ortogonal a (P). Assim, o gradiente no ponto médio da aresta é decomposto em duas componentes:

$$\nabla u_{IJ_L}^{\Omega_r} = \nabla u_{IJ_L}^{\Omega_r(N)} + \nabla u_{IJ_L}^{\Omega_r(\mathrm{P})}. \tag{3.17}$$

A componente do gradiente paralela à direção da aresta $\nabla u_{IJ_L}^{\Omega_r(\mathrm{P})}$ pode ser calculada por:

$$\nabla u_{IJ_L}^{\Omega_r(\mathrm{P})} = \left(\nabla u_{IJ_L}^{\Omega_r} \cdot \vec{L}_{IJ_L}\right) \cdot \vec{L}_{IJ_L}. \tag{3.18}$$

e a componente normal $\nabla u_{IJ_L}^{\Omega_r(N)}$ por:

$$\nabla u_{IJ_L}^{\Omega_r(N)} = \nabla u_{IJ_L}^{\Omega_r} - \nabla u_{IJ_L}^{\Omega_r(P)}. \tag{3.19}$$

Aplicando a Eq. (3.18) na Eq. (3.19) e aproximando $\nabla u_{IJ_L}^{\Omega_r}$ pela média aritmética dos gradientes nodais, $\nabla u_{IJ_L}^{\Omega_r(N)}$ pode ser reescrita como:

$$\nabla u_{IJ_L}^{\Omega_r(N)} = \frac{\nabla u_I + \nabla u_{J_L}}{2} - \left(\frac{\nabla u_I + \nabla u_{J_L}}{2} \cdot \vec{L}_{IJ_L} \right) \cdot \vec{L}_{IJ_L} \tag{3.20}$$

Substituindo a componente $\nabla u_{IJ_L}^{\Omega_r(\mathrm{P})}$ por uma aproximação de diferenças centradas, obtém-se:

$$\nabla u_{IJ_L}^{\Omega_r(\mathrm{P})} = \frac{u_{J_L} - u_I}{\Delta_{IJ_L}} \cdot \vec{L}_{IJ_L}. \tag{3.21}$$

$$\nabla u_{IJ_L}^{\Omega_r} = \left(\frac{\nabla u_I^{\Omega_r} + \nabla u_{J_L}^{\Omega_r}}{2} - \left(\frac{\nabla u_I^{\Omega_r} + \nabla u_{J_L}^{\Omega_r}}{2} \cdot \vec{L}_{IJ_L} \right) \cdot \vec{L}_{IJ_L} + \frac{u_{J_L} - u_I}{\Delta_{IJ_L}} \cdot \vec{L}_{IJ_L} \right). \tag{3.22}$$

Assim, chega-se a uma aproximação para o gradiente no ponto médio da aresta que é uma aproximação "híbrida" de volumes finitos para a componente normal e de diferenças finitas centradas para a componente paralela do gradiente, ambas de segunda ordem. A função de fluxo da Eq. (3.5) pode ser escrita como:

$$F_{IJ_L}^{\Omega_r} = -\underset{\sim}{K}^{\Omega_r} \left(\frac{\nabla u_I^{\Omega_r} + \nabla u_{J_L}^{\Omega_r}}{2} - \left(\frac{\nabla u_I^{\Omega_r} + \nabla u_{J_L}^{\Omega_r}}{2} \cdot \vec{L}_{IJ_L} \right) \cdot \vec{L}_{IJ_L} + \frac{u_{J_L} - u_I}{\Delta_{IJ_L}} \cdot \vec{L}_{IJ_L} \right). \tag{3.22}$$

Embora este esquema híbrido pareça ser mais complicado que a aproximação usual de fluxo de dois pontos (TPFA) e a técnica do valor médio do gradiente nodal, esse esquema permite o correto cálculo do termo de difusão cruzada que é o principal ponto fraco do TPFA. Ao mesmo tempo, também evita problemas conhecidos como *checkerboarding* ou a ocorrência de oscilações par-ímpar características de formulações clássicas que utilizam estêncil estendido, onde o gradiente nodal é calculado como uma média aritmética em malha quadrilaterais ortogonais igualmente espaçadas. De fato, pode-se provar que essas duas técnicas usuais são aproximações inconsistentes de volumes finitos para o operador Laplaciano discreto em malhas triangulares (Svärd e Nordström, 2003).

Usando a aproximação de fluxo na superfície do volume de controle dada pela Eq.(3.22), pode-se redefinir a Eq. (3.5) como:

$$\sum_{r=1}^{Ndom}\left(\sum_{L_{I(\Omega_r)}}\vec{F}_{IJ_L}^{\Omega_r *}\cdot\vec{C}_{IJ_L}^{\Omega_r}+\sum_{L_{I(\Gamma_r)}}\vec{F}_{IJ_L}^{\Gamma_r}\cdot\vec{D}_{IJ_L}^{\Omega_r}\right)=\sum_{r=1}^{Ndom}f_I^{\Omega_r}V_I^{\Omega_r}. \tag{3.23}$$

onde *Ndom* refere-se ao número de domínios que cercam o nó I, e o termo $f_I^{\Omega_r}$ está relacionado a termos de fonte/sumidouro associados ao volume $V_I^{\Omega_r}$ que faz parte do subdomínios Ω_r.

3.3.2 Método de volumes finitos baseados em aresta 2 - (EBFV2)

Esta abordagem pode ser obtida considerando-se uma variação linear da variável escalar "*u*" através das arestas da malha primal. Neste caso, os gradientes são considerados constantes sobre os elementos da malha primal e os fluxos podem ser calculados através da malha dual, pois o coeficiente de difusão é constante entre dois volumes de controle adjacentes sobre um mesmo elemento da malha primal.

Esta metodologia é análoga ao tradicional método de elementos finitos com volumes de controle (CVFEM). Na formulação presente, para um nó arbitrário I da malha, o lado esquerdo da Eq. (3.4) pode ser escrita como:

$$-\int_{\Gamma_I}\left(\underset{\sim}{K}\nabla u\right)\cdot\vec{n}\,\partial\Gamma_I=\sum_{e_i}\left(-\underset{\sim}{K}^{e_i}\nabla u^{e_i}\right)\cdot\vec{C}_{IJ}^{e_i}=\sum_{e_i}\vec{F}_{IJ}^{e_i}\cdot\vec{C}_{IJ}^{e_i}. \tag{3.24}$$

Na Eq. (3.24) o sobrescrito "*e*" refere-se à contribuição do fluxo $\vec{F}_{IJ}^{e_i}=-\underset{\sim}{K}^{e_i}\nabla u^{e_i}$ que vem da aresta *IJ* e que está associada ao elemento "*e*" adjacente à aresta *IJ*.

Para um elemento qualquer da malha primal, o gradiente pode ser calculado como:

$$\nabla u^e=\frac{1}{V^e}\sum_{IJ_e}\frac{\left(u_I+u_J\right)}{2}\left(2\vec{D}_{IJ}^e\right). \tag{3.25}$$

onde V^e é o volume (área em 2-D) do elemento, $\vec{D}_{IJ}^e$ são os vetores área normais às arestas e o somatório é feito sobre todas as arestas da malha primal que define este elemento. Nes-

te trabalho, apenas elementos triangulares foram usados. Assim, para um elemento triangular qualquer definido pelos nós 1, 2 e 3, pode-se escrever:

$$\nabla u^e = \frac{1}{V^e}\left[\frac{(u_1+u_2)}{2}\left(2\vec{D}^e_{12}\right)+\frac{(u_2+u_3)}{2}\left(2\vec{D}^e_{23}\right)+\frac{(u_3+u_1)}{2}\left(2\vec{D}^e_{13}\right)\right]. \tag{3.26}$$

Este gradiente é então substituído na Eq. (3.24) de forma a calcular os fluxos através das superfícies de controle. Após algumas manipulações geométricas e algébricas, descrito em mais detalhes no Apêndice A, é possível escrever as contribuições dos fluxos de forma totalmente por arestas. Esta formulação produz um sistema de equações simétricas montado aresta por aresta que pode ser resolvido por um método iterativo tipo gradiente conjugado.

Esta formulação foi desenvolvida e usada apenas para resolver uma equação elíptica com coeficientes descontínuos representada, por exemplo, pelo campo de pressões em escoamento em meios porosos.

O intuito neste trabalho foi apenas em desenvolver e validar uma formulação de volumes finitos por aresta capaz de lidar de uma forma mais simples, conseqüentemente permitindo criar um código de computador mais rápido de ser implementado, com as heterogeneidades do meio poroso. Essa deverá ser explorada no contexto de escoamentos multifásicos em trabalhos futuros.

3.4 Análise de erro

Para avaliar a acurácia de ambos os procedimentos de volumes finitos baseado em arestas para os problemas com coeficientes descontínuos e anisotrópicos, define-se o erro de truncamento assintótico como:

$$\left\|E_h\right\| = Ch^q + O\left(h^{q+1}\right). \tag{3.27}$$

onde h é o espaçamento da malha, q é a ordem do erro, que representa a taxa de convergência, C é uma constante que não depende de h e $\left\|\cdot\right\|$ é alguma norma especificada. Neste caso, a taxa de convergência é estimada como:

$$q \cong \log_2 \frac{\|E_h\|}{\|E_{h/2}\|}. \tag{3.28}$$

com *h/2* representando a malha com espaçamento igual à metade de *h*.

As taxas de convergências foram estimadas usando a norma dos mínimos quadrados médios (RMS - Root Mean Square), $\|E\|_{RMS}$, que é calculada como:

$$\|E\|_{RMS} = \|\hat{u} - u\|_{L_{RMS}} = \left(\sum_{I=1}^{NP} (\hat{u}_I - u_I)^2 \Big/ NP \right)^{1/2}. \tag{3.29}$$

onde u é a solução exata, $\hat{u}$ é a solução aproximada e NP é o numero de nós livre da malha computacional.

3.5 Exemplos

3.5.1 Meio homogêneo e isotrópico

Este problema, que envolve a solução de uma equação elíptica em meio homogêneo e isotrópico, foi apresentado por Chen et al. (2006) que o resolveu usando dois métodos diferentes: o tradicional método de elementos finitos baseados em volumes de controle (CVFEM) e o método de função de aproximação de volumes de controle (CVFA - *control volume function approximation method*) que é um método localmente conservativo no qual são usadas funções de interpolação, tais como *splines*, no processo de aproximação. Na forma compacta, o problema pode ser descrito pelas seguintes equações:

$$\nabla(\underset{\sim}{K}\nabla u) = 2\pi^2 \cos(\pi x)\cos(\pi y) \quad for \quad 0 < x < 1 \quad and \quad 0 < y < 1. \tag{3.30}$$

onde o coeficiente de difusão $\underset{\sim}{K}$ é definido por:

$$\underset{\sim}{K} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}. \tag{3.31}$$

e as condições de contorno são dadas por:

$$\begin{aligned} &\nabla u \cdot \underset{\sim}{n} = 0, \quad para \quad x = 0,\ x = 1 \quad e \quad 0 < y < 1 \\ &u = \cos(\pi x), \quad para \quad 0 < x < 1 \quad e \quad y = 0 \\ &u = -\cos(\pi x), \ para \quad 0 < x < 1 \quad e \quad y = 1 \end{aligned} \tag{3.32}$$

A solução periódica analítica para este problema é $u = \cos(\pi x)\cos(\pi y)$ definida em um domínio quadrado unitário $\Omega = (0,1)^2$. A figura 3.2 mostra o campo escalar *u* obtido usando-se uma malha triangular $(65\text{x}65)$ que está em excelente concordância com a solução analítica.

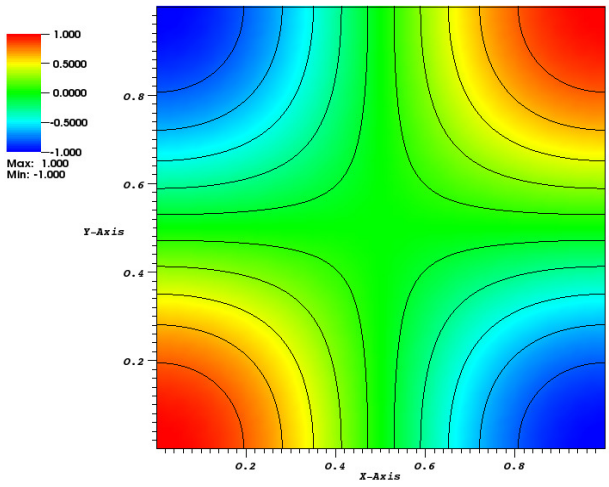


Figura 3.2 – Equação elíptica com coeficientes isotrópicos e contínuos: campo escalar u obtido com o EBFV2 usando uma malha triangular (65x65).

Nas tabelas 3.1, 3.2 e 3.3 estão representados os erros e as taxas de convergência obtida usando os métodos: CVFA, EBFV1 e EBFV2, respectivamente e para malhas com diferentes espaçamentos (9x9), (17x17), (33x33) e (65x65) nós.

Como se pode observar, os três métodos têm acurácia de segunda ordem para a variável escalar u, com resultados muito similares. Pode-se também perceber, para este exemplo em particular, ambos os métodos de volumes finitos baseados em arestas se comparam favoravelmente em relação ao CVFA, com o EBFV2 apresentando erros ligeiramente menores.

Tabela 3.1 – Solução numérica da equação elíptica em um meio homogêneo e isotrópico. Erros e taxa de convergência rates para: a) CVFA; b) EBFV1; c) EBFV2.

N	$\|E\|_{RMS}$	q_{RMS}
9	1.21e-002	----------
17	3.00e-003	2.0199
33	7.45e-004	2.0087
65	1.85e-004	2.0050

(a) CVFA

$\|E\|_{RMS}$	q_{RMS}
6.88e-003	---------
1.41e-003	2.2899
3.21e-004	2.1309
7.70e-005	2.0616

(b) EBFV1

$\|E\|_{RMS}$	q_{RMS}
2.82e-003	----------
6.24e-004	2.1773
1.46e-004	2.0914
3.50e-005	2.0451

(b) EBFV2

3.5.2 Meio homogêneo e anisotrópico

Este problema foi originalmente proposto por Crumpton (1995) que o resolveu usando um método de volumes finitos de fluxo continuo centrado na célula (FCFV). Novamente, os domínios considerados é um quadrado de lado unitário $\Omega = (0,1)^2$, com condições de contorno de Dirichlet extraídas da solução exata que é dada por $u = e^{xy}$. Na forma compacta, este problema pode ser descrito por:

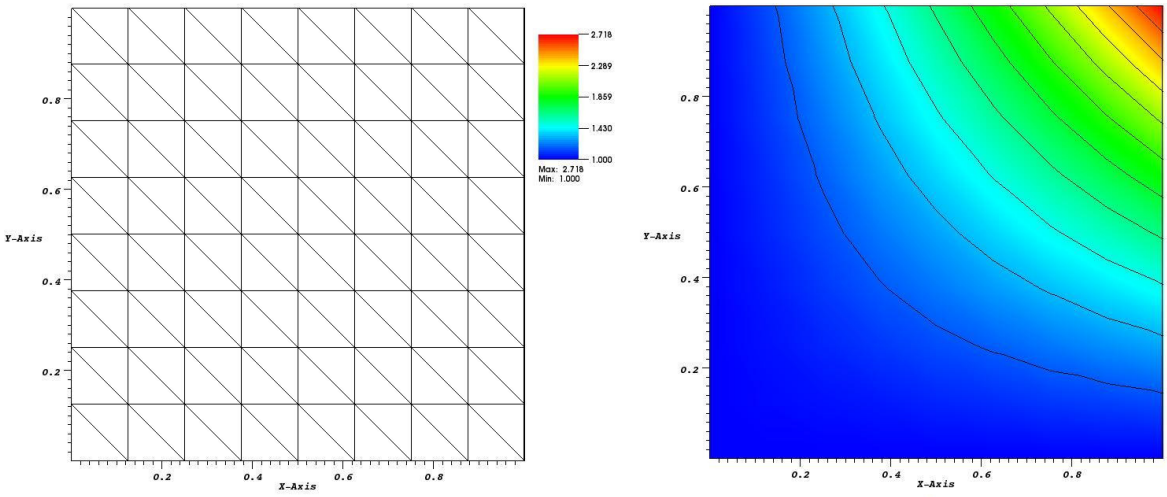
$$\nabla\left(\underset{\sim}{K}\nabla u\right) = -2\left(1 + x^2 + xy + y^2\right)e^{xy}. \tag{3.33}$$

onde o tensor de difusão $\underset{\sim}{K}$ é dado por:

$$\underset{\sim}{K} = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}. \tag{3.34}$$

São mostradas nas Fig. (3.3a) e Fig. (3.3b) uma malha triangular com (9x9) nós e os contornos do campo escalar u, respectivamente, para a mesma.

A tabela 3.2 mostra os erros médios quadrático, $\|E\|_{RMS}$, e as taxas de convergências, q_{RMS}, para os métodos CVFA, EBFV1 e EBFV2, respectivamente. Os resultados obtidos com o FCFV e ambas as formulações de volumes finitos baseadas em arestas são essencialmente de segunda ordem no espaço. Além disso, elas se comparam favoravelmente em relação aos obtidos por Crumpton (1995) usando o FCFV.



(a) (b)

Figura 3.3 – Meio homogêneo e anisotrópico: a) malha triangular (9x9); b) campo escalar u obtido com o EBFV1.

Tabela 3.2– Solução numérica da equação elíptica em um meio homogêneo e anisotrópico. Erros e taxas de convergências para: a) FCFV; b) EBFV1; c) EBFV2.

N	$\|E\|_{RMS}$	q_{RMS}
9	1.16e-003	-------
17	2.89e-004	2.0005
33	7.28e-005	1.9891
65	1.83e-005	1.9921

(a) CVFA

$\|E\|_{RMS}$	q_{RMS}
3.09e-003	-------
7.14e-004	2.1106
1.72e-004	2.0545
4.20e-005	2.0280

(b) EBFV1

$\|E\|_{RMS}$	q_{RMS}
5.35e-004	-------
1.12e-004	2.1122
2.60e-005	2.0758
6.00e-006	1.9278

(b) EBFV2

3.5.3 Meio heterogêneo e anisotrópico

O exemplo seguinte foi proposto originalmente por Crumpton (1997) e consiste em um quadrado de lado unitário formado por dois materiais diferentes. As condições de contorno de Dirichlet são dadas pela solução exata. Crumpton (1997) resolveu este problema usando o FCFV com malha estruturada quadrilateral ortogonal. O problema é definido por:

$$\nabla\left(\underset{\sim}{K}\nabla u\right) = f\left(x, y\right). \tag{2.35}$$

onde os termos de fonte e o coeficiente de difusão, ambos descontínuos, são dados, respectivamente, por:

$$f(x,y)=\begin{cases}\left[-2sin(y)-\cos(y)\right]\alpha x-sin(y) & \text{para} \quad x\le 0\\ 2\alpha\exp(x)\cos(y) & \text{para} \quad x>0\end{cases}. \tag{3.36}$$

e

$$\underset{\sim}{K}=\begin{cases}\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix} & \text{para} \quad x<0\\ \alpha\begin{pmatrix}2 & 1\\ 1 & 2\end{pmatrix} & \text{para} \quad x>0\end{cases}. \tag{3.37}$$

onde α controla a intensidade da descontinuidade, ou seja, o salto na propriedade do material entre as duas regiões distintas. A solução exata para este problema, encontrada em Crumpton (1997) é apresentada pela Eq. (3.30).

$$u(x,y)=\begin{cases}\left[2sin(y)+\cos(y)\right]\alpha x+sin(y) & \text{para} \quad x\le 0\\ \exp(x)\cos(y) & \text{para} \quad x>0\end{cases}. \tag{3.38}$$

Neste trabalho, o problema foi resolvido usando uma seqüência uniforme de malhas estruturadas triangulares e quadrilaterais ortogonais com $N=(9\text{x}9)$, $(17\text{x}17)$, $(33\text{x}33)$ e $(65\text{x}65)$ nós. São mostradas as linhas de contorno da função escalar $u(x,y)$ obtidas para os métodos EBFV1 e EBFV2 para uma malha triangular de $(65\text{x}65)$ nós associadas a $\alpha=1.0$, e $\alpha=1000.0$ nas Fig. (3.4a) a (3.4d), respectivamente.

As tabelas 3.3 a 3.6 representam os erros médios quadráticos $\|E\|_{RMS}$ e as taxas de convergência q_{RMS} para quatro valores diferentes de α obtidos com o método FCFV de Crumpton (1997) usando malhas quadrilaterais ortogonais e os esquemas EBFV1 e EBFV2 usando malhas triangulares estruturadas.

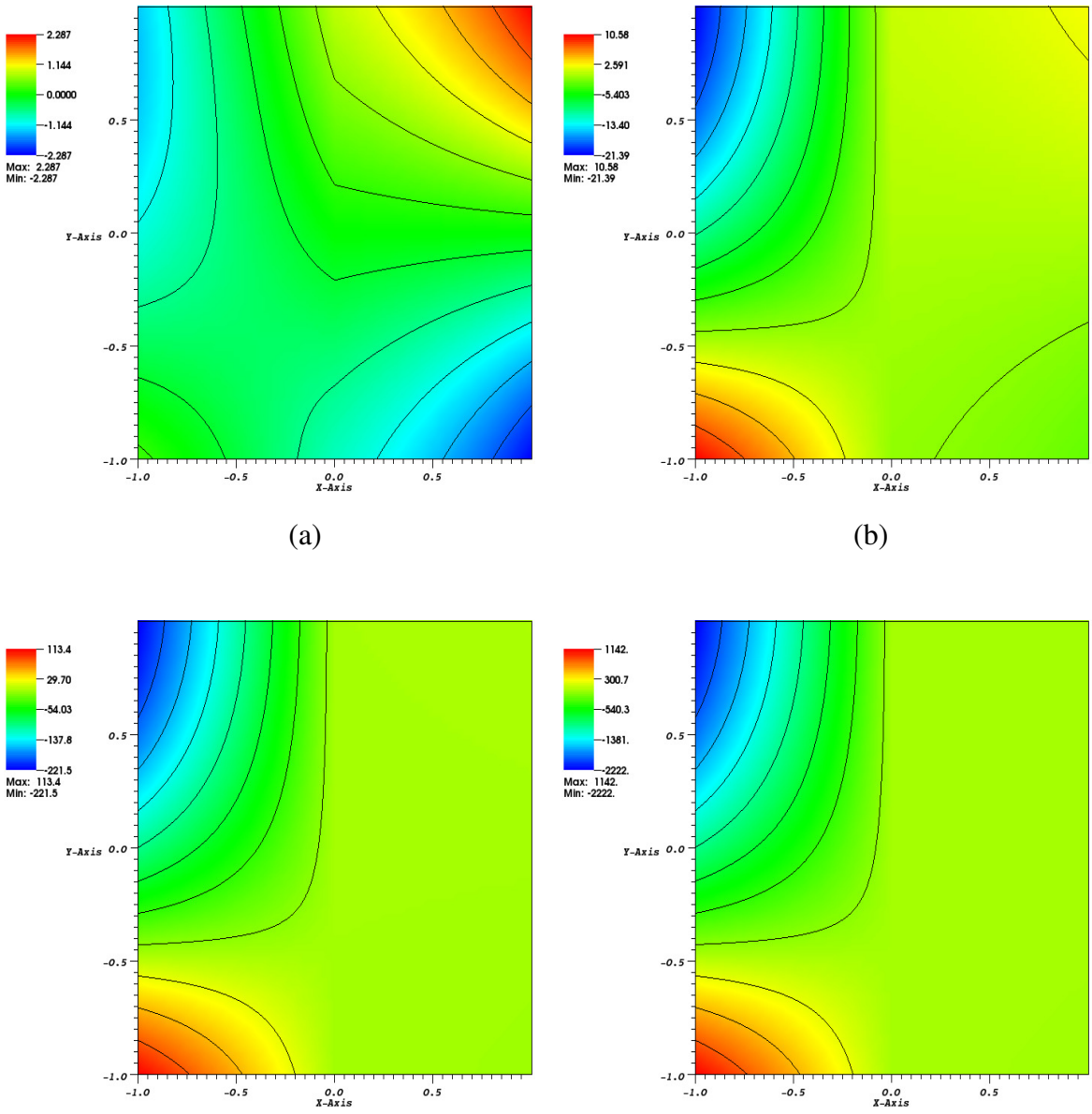


Figura 3.4 – Linhas de contorno da função escalar $u(x, y)$ obtidas com o EBFV1 usando uma malha estruturada triangular (65x65) para diferentes valores de α: (a) $\alpha = 1.0$; (b); $\alpha = 10.0$ (c) $\alpha = 100.0$; (d) $\alpha = 1000.0$

Como se pode observar nas tabelas 3.3 a 3.6, para este problema particular de benchmark, apesar do fato do aumento da intensidade da descontinuidade também aumentar a magnitude do erro para os três métodos, estes mostram acurácia de segunda ordem no espaço para todos os valores de α.

Pode-se também notar que para todos os casos testados, os esquemas EBFV1 e EBFV2 são bastante similares ao método FCFV. Vale mencionar que os experimentos nu-

méricos (não apresentados aqui) também mostram que o EBFV1 gera resultados um pouco melhores que o FCFV quando malhas estruturadas quadrilaterais são usadas.

Tabela 3.3 – Solução numérica da equação elíptica em um meio anisotrópico com coeficiente de difusão descontínuo $\alpha = 1.0$. Erros e taxas de convergências para: a) FCFV; b) EBFV1; c) EBFV2.

N	$\|E\|_{RMS}$	q_{RMS}
9	3.33e-003	----------
17	9.37e-004	1.8294
33	2.45e-004	1.9353
65	6.25e-005	1.9709

(a) CVFA

$\|E\|_{RMS}$	q_{RMS}
6.02e-003	----------
1.50e-003	1.9989
3.77e-004	2.0012
9.40e-005	2.0014

(b) EBFV1

$\|E\|_{RMS}$	q_{RMS}
1.49E-003	----------
3.92E-004	1.9300
1.00E-004	1.9590
2.56E-005	1.9780

(b) EBFV2

Tabela 3.4– Solução numérica da equação elíptica em um meio anisotrópico com coeficiente de difusão descontínuo $\alpha = 10.0$. Erros e taxas de convergências para: a) FCFV; b) EBFV1; c) EBFV2.

N	$\|E\|_{RMS}$	q_{RMS}
9	1.62e-002	----------
17	4.35e-003	1.9146
33	1.11e-003	1.9705
65	2.81e-004	1.9819

(a) CVFA

$\|E\|_{RMS}$	q_{RMS}
4.57e-002	----------
1.17e-002	1.9569
2.98e-003	1.9829
7.49e-004	1.9925

(b) EBFV1

$\|E\|_{RMS}$	q_{RMS}
2.18E-003	----------
5.72E-004	1.9361
1.47E-004	1.9601
3.73E-005	1,9780

(b) EBFV2

Tabela 3.5– Solução numérica da equação elíptica em um meio anisotrópico com coeficiente de difusão descontínuo $\alpha = 100.0$. Erros e taxas de convergências para: a) FCFV; b) EBFV1; c) EBFV2.

N	$\|E\|_{RMS}$	q_{RMS}
9	1.81e-001	----------
17	4.74e-002	1.9330
33	1.21e-002	1.9699
65	3.04e-003	1.9929

(a) CVFA

$\|E\|_{RMS}$	q_{RMS}
4.40e-001	----------
1.13e-001	1.9538
2.88e-002	1.9813
7.24e-003	1.9916

(b) EBFV1

$\|E\|_{RMS}$	q_{RMS}
1.09E-002	----------
2.97E-003	1.8845
7.69E-004	1.9497
1.95E-004	1.9762

(b) EBFV2

Tabela 3.6 – Solução numérica da equação elíptica em um meio anisotrópico com coeficiente de difusão descontínuo $\alpha = 1000.0$. Erros e taxas de convergências para: a) FCFV; b) EBFV1; c) EBFV2.

N	$\|E\|_{RMS}$	q_{RMS}
9	4.39e-000	----------
17	1.13e-000	1.9536
33	2.87e-001	1.9812
65	7.22e-002	1.9915

(a) CVFA

$\|E\|_{RMS}$	q_{RMS}
4.39e-000	----------
1.13e-000	1.9536
2.87e-001	1.9812
7.22e-002	1.9915

(b) EBFV1

$\|E\|_{RMS}$	q_{RMS}
1.14E-001	----------
1.14E-001	1.8812
3.10E-002	1.9489
8.04E-003	1.9760

(b) EBFV2

3.5.4 Meio homogêneo e anisotrópico com solução não-suave

Este último exemplo foi resolvido por Lipnikov et al. (2007) usando um esquema de volumes finitos monótono não-linear centrado na célula (NLMFV) desenvolvido especificamente para produzir soluções monotônicas, ou seja, que preserva a positividade da solução na presença de materiais altamente heterogêneos e anisotrópicos. O domínio computacional consiste de um quadrado de lado unitário que um furo quadrado em seu centro $\Omega = (0,1)^2 / [4/9, 5/9]^2$ como é mostrado na Fig. (3.5). Este problema pode ser brevemente descrito com:

$$\nabla \left(\underset{\sim}{K} \nabla u \right) = f(x, y). \tag{3.39}$$

onde:

$$\underset{\sim}{K} = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} k_1 & 0 \\ 0 & k_2 \end{pmatrix} \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}. \tag{3.40}$$

Com $f(x, y) = 0$ e condições e contorno dadas por $u = 0$ nas fronteiras externas e $u = 2$ nas fronteiras internas, isto é:

$$
\begin{aligned}
u = 0, \quad & para \quad \begin{cases} 0 < x < 1 & e \quad (y = 0 \quad ou \quad y = 1) \\ 0 < y < 1 & e \quad (x = 0 \quad ou \quad x = 1) \end{cases}. \\
u = 2, \quad & para \quad \begin{cases} 4/9 < x < 5/9 & e \quad (y = 4/9 \quad ou \quad y = 5/9) \\ 4/9 < y < 5/9 & e \quad (x = 4/9 \quad ou \quad x = 5/9) \end{cases}.
\end{aligned}
\tag{3.41}
$$

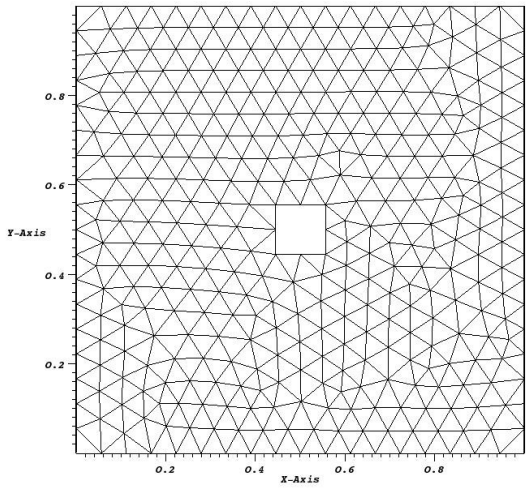


Figura 3.5 – Geometria e malha triangular não-estruturada usada para resolver o problema homogêneo e anisotrópico com solução não suave.

Este problema foi resolvido usando dois valores diferentes para o ângulo $\theta = \pi/6$ e $\theta = 5\pi/6$ e a razão $k_1/k_2 = 1000$. Não há solução analítica para este problema, mas os resultados devem claramente ficar entre os valores das fronteiras no interior do domínio, isto é, $0 \le u(x, y) \le 2$, sem "*overshoots*" ($u(x, y) > 2$) ou "*undershoots*" ($u(x, y) < 0$).

Foi usada uma malha triangular não-estruturada com um espaçamento médio de $h \approx 1/18$. As Fig. 3.6 e 3.7 mostram isolinhas da variável escalar *u* para ambos os métodos e ambos os valores de rotação. Pode-se perceber nestas figuras que os resultados obtidos são qualitativamente bastantes similares um com o outro e muito próximos dos obtidos com o NLMFV de Lipnikov et al. (2007).

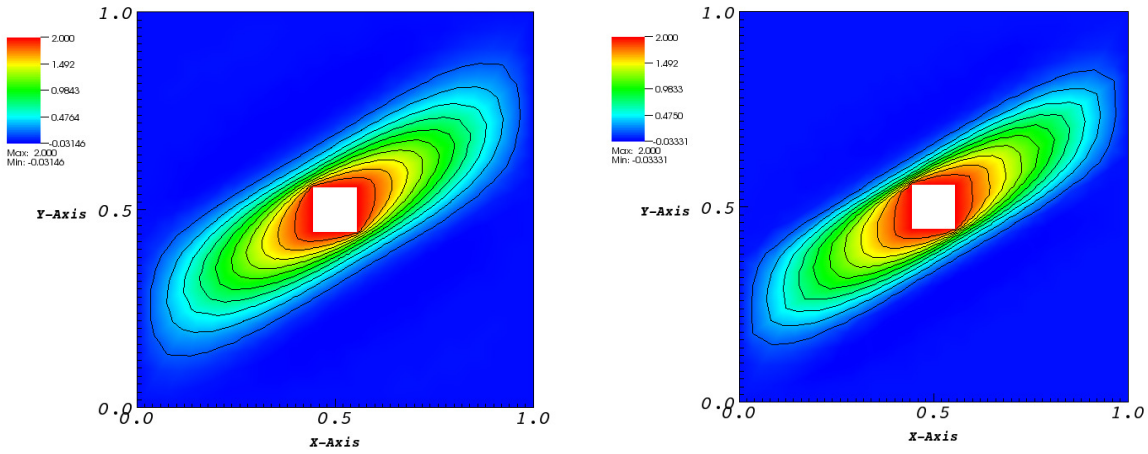
Embora ambos os métodos EBFV1 e EBFV2 não apresentem "*overshoots*", estes produziram uma quantidade razoável de "*undershoots*" com mais de 30% dos nós livres enquanto o NLMFV produziu alguns "*overshoots*".

Para a malha não-estruturada usada aqui, os valores mínimos de *u* foram: $u = -0.03146$ para $\theta = \pi/6$ e $u = -0.03024$ para $\theta = 5\pi/6$, para o EBFV1 e

$u = -0.03331$ para $\theta = \pi/6$ e $u = -0.02734$ para $\theta = 5\pi/6$, para o EBFV2, respectivamente.

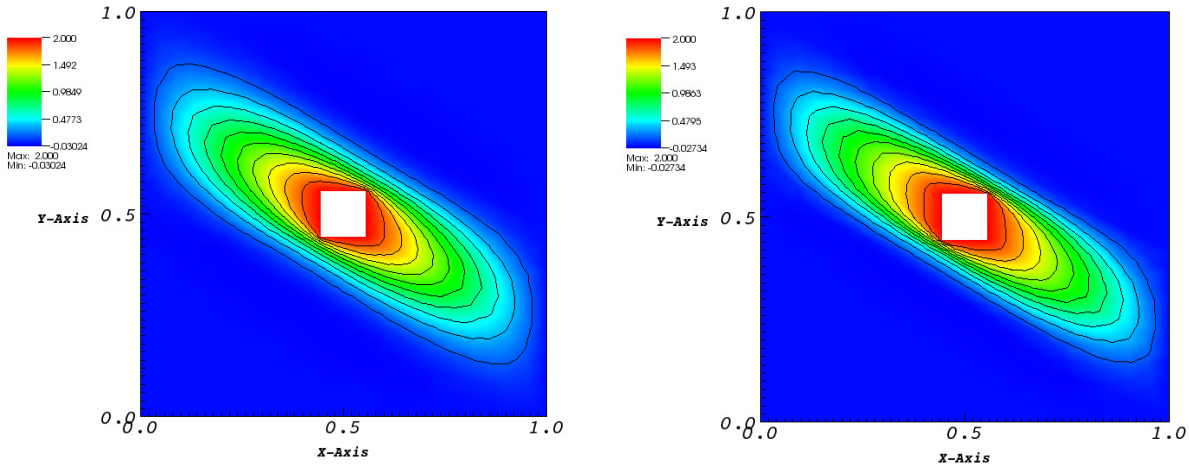
O comportamento da variável *u* ao longo de uma reta paralela ao eixo x passando pelo centro da figura para um dos casos ($\theta = 5\pi/6$, EBFV2) pode ser visto na Fig. (3.8) onde se verifica a presença de "*undershoots*".

A fim de melhorar a monotonicidade da solução, em Lipnikov et al. (2007), autores sugerem que se use a estratégia diferente (*inverse distance weighting*) para interpolar a variável escalar *u* pelo domínio ao invés de usar as técnicas mais tradicionais de interpolação.



(a) (b)

Figura 3.6 – Solução numérica para o problema homogêneo e anisotrópico com solução não suave: ângulo de rotação $\theta = \pi/6$; a) EBFV1; b) EBFV2.



(a) (b)

Figura 3.7 – Solução numérica para o problema homogêneo e anisotrópico com solução não suave: ângulo de rotação $\theta = 5\pi/6$; a) EBFV1; b) EBFV2.

Além disso, eles também estabeleceram que "*overshoots*" não são observados com o NLMFV quando os gradientes acentuados estão alinhados com as arestas da malha. Este exemplo mostra claramente que outros melhoramentos podem ser feitos para ambos os EBFVs caso se deseje garantir monotonicidade nos resultados em problemas mais complexos.

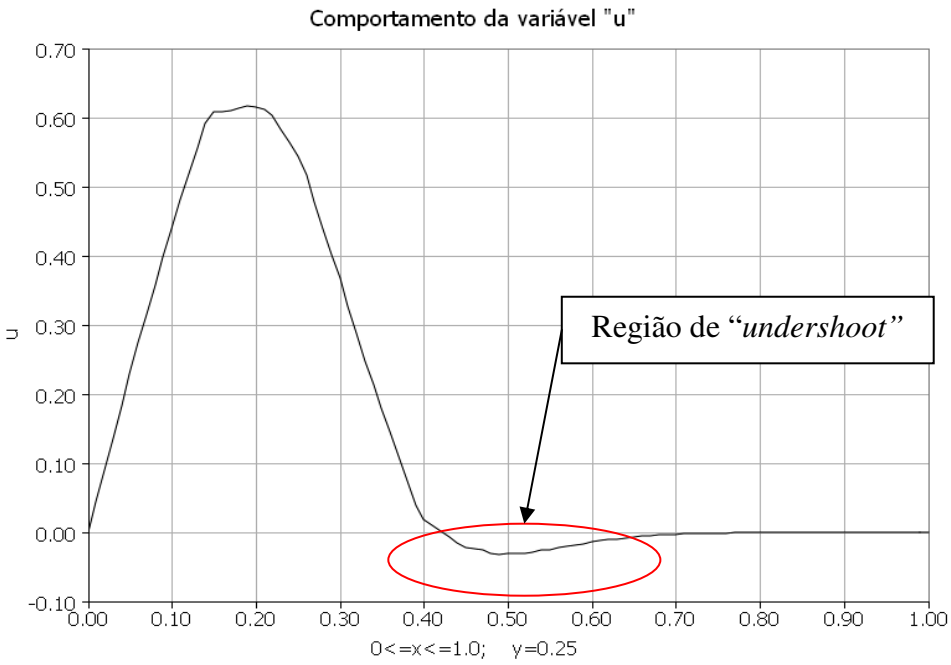


Figura 3.8 – Comportamento da variável u ao longo de uma reta paralela ao eixo x passando pelo centro do domínio. Para um problema com valores prescritos mínimo e máximo de 0.0 e 2.0, respectivamente, esperava-se que os resultados ficassem confinados dentro desse intervalo. Devido a forte anisotropia, verifica-se a presença de "*undershoots*", ou seja, valores calculados abaixo do mínimo esperado.

4. MÉTODOS DE INTEGRAÇÃO NO TEMPO

4.1 Introdução

As equações de fluxo multifásico em reservatórios de petróleo envolvem um conjunto de equações diferenciais parciais não-lineares, acopladas e dependentes do tempo. Um importante problema de simulação numérica em reservatórios de petróleo é desenvolver esquemas estáveis, robustos e acurados para resolver estas equações acopladas (Li *et al*, 2004). Essencialmente, existem quatro esquemas de solução importantes: IMPES (IMplicit Pressure Explicit Saturation), seqüencial implícito, totalmente implícito e o adaptativo implícito-explícito. Neste trabalho, será dada ênfase ao método IMPES por ser o método de integração no tempo adotado na construção do simulador bifásico óleo-água e em particular será apresentado uma estratégia do IMPES modificado baseado no trabalho de Hurtado *et al*. (2006).

4.2 IMPES

O método IMPES surgiu no início das simulações de reservatórios e tornou-se bastante popular na resolução das equações de fluxo multifásico. Para o caso bifásico, ele se caracteriza por ser um método segregado que resolve de forma parcialmente desacoplada as variáveis de pressão e saturação. Este desacoplamento parcial resulta essencialmente na resolução de duas equações, uma na forma implícita (pressão), através de um sistema de equações lineares, e outra na forma explícita (saturação).

Este método foi escolhido devido à sua simplicidade de implementação comparada a outros métodos e pela relativa baixa demanda computacional para certas classes de problemas e dimensões mínimas dos elementos da malha.

Uma grande desvantagem deste método surge devido à aproximação explícita da saturação que impõe uma severa restrição de estabilidade quanto ao tamanho do passo de tempo, o que pode representar um custo de CPU extremamente elevado tornando-o proibitivo para problemas de grande porte com malhas muito refinadas.

Para problemas de fluxo bifásico incompressíveis, o método IMPES produz bons resultados. Entretanto, em problemas com forte não-linearidade como nos casos *black-oil* e composicional, este método não é eficiente e robusto (Li *et al, 2003*).

Para ambos os casos, envolvendo malhas muito refinadas ou apresentando forte não-linearidade, métodos adaptativos são usados para melhorar a eficiência do método IMPES. Geralmente, estes métodos adaptativos, são baseados na condição Courant–Friedrichs–Lewy (CFL), de forma a determinar quando os blocos da malha são tratados implicitamente ou quando são tratados explicitamente (Li *et al*, 2003).

4.3 Descrição do método IMPES

Partindo de um campo de saturações inicial, $S_w(\Omega, t=0) = S_{w_0}$, monta-se um sistema de equações para o cálculo do campo de pressões através da Eq. (4.1):

$$-\nabla.\left(K\lambda(S_w)\nabla p\right) = q. \tag{4.1}$$

onde K, $\lambda(S_w)$ e q representam a permeabilidade absoluta da rocha, a mobilidade da fase água, que é função da saturação, e um termo fonte/sumidouro, respectivamente. Por questão de simplificação, os termos de gravidade e capilaridade, inicialmente apresentados na Eq. (2.19), foram removidos.

O novo campo de saturações é obtido através da Eq. (4.3), uma simplificação da Eq. (2.32):

$$\phi\frac{\partial S_w}{\partial t} = -\nabla\cdot\left(f_w\vec{v}\right) + Q_w. \tag{4.3}$$

Como se percebe na Eq. (4.3), o campo de saturações depende do termo de fluxo, função do campo de velocidades. Logo, três passos precisam ser efetuados antes na seguinte ordem: recuperação dos gradientes de pressão, cálculo do campo de velocidades e cálculo dos termos de fluxo. Os gradientes de pressão e os termos de fluxo estão associados aos volumes de controle, enquanto o campo de velocidades está associado às arestas. Esse último é obtido por meio da lei de Darcy, Eq. (2.1), cuja equação discreta avalia a variável no ponto médio da aresta, sobre a superfície de controle. O gradiente, calculado de forma nodal, é aproximado conforme a Eq. (3.22), o que implica:

$$\vec{v}_{IJ_L}^{\Omega_r} = -\left(\underset{\sim}{K}^{\Omega_r}\lambda_T^{IJ_L}\right)\left(\frac{\left(\nabla\hat{p}_I^{\Omega_r}+\nabla\hat{p}_{J_L}^{\Omega_r}\right)}{2}-\left(\frac{\left(\nabla\hat{p}_I^{\Omega_r}+\nabla\hat{p}_{J_L}^{\Omega_r}\right)}{2}\cdot\vec{L}_{IJ_L}\right)\vec{L}_{IJ_L}+\frac{\left(\hat{p}_{J_L}-\hat{p}_I\right)}{\left|\Delta_{IJ_L}\right|}\vec{L}_{IJ_L}\right). \tag{4.4}$$

O índice Ω_r indica que, para problemas com meios heterogêneos, o campo de velocidades deve ser calculado domínio a domínio onde os nós na interface entre materiais acumulam a contribuição de cada campo. O fluxo de massa em cada volume de controle pode, então, ser obtido por:

$$F_{IJ_L}^{\Omega_r}\cdot\vec{C}_{IJ_L}^{\Omega_r} = \left(\frac{f_I+f_{J_L}}{2}\right)\vec{v}_{IJ_L}^{\Omega_r}\cdot\vec{C}_{IJ_L}^{\Omega_r} + AD. \tag{4.5}$$

onde AD corresponde a um de termo de difusão numérica. Dessa forma, utilizando o método Euler explícito, o novo campo de saturações pode ser obtido através de sua forma discreta por:

$$\hat{S}_w^{n+1} = \hat{S}_w^n - \sum_{r=1}^{Ndom}\frac{\Delta t}{\phi^{\Omega_r}V_I^{\Omega_r}}\left(\sum_{L=1}^{NN^{(\Omega_r)}}\vec{F}_{IJ_L(w)}^{\Omega_r}\cdot\vec{C}_{IJ_L}^{\Omega_r}+\sum_{L=1}^{NN^{(\Gamma_r)}}\vec{F}_{IJ_L(w)}^{\Gamma_r}\cdot\vec{D}_{IJ_L}^{\Omega_r}+Q_I^{\Omega_r}V_I^{\Omega_r}\right)^n. \tag{4.6}$$

onde ϕ^{Ω_r} representa a porosidade do domínio Ω_r e Δt corresponde ao avanço no tempo. Este avanço corresponde a uma discretização temporal de primeira ordem que satisfaz um rígido critério de estabilidade (CFL).

O método IMPES implementado neste trabalho pode, então, ser descrito pelo algoritmo 4.1. A resolução do sistema de equações para o cálculo do campo de pressões representa mais de 80% do tempo de CPU consumido para cada iteração do método e sua constante atualização torna-se uma barreira para problemas maiores ou quando se leva em conta, por exemplo, a presença de efeitos de capilaridade.

Um método incondicionalmente estável como o totalmente implícito (*fully implicit*), onde as mesmas variáveis são resolvidas de forma acoplada em um único sistema de equações, permite que passos de tempo maiores sejam empregados, porém trazendo a desvantagem de exigir mais memória e tempo de CPU para cada iteração.

```
Sw(Ω,t=0)
Enquanto (tempo_simulação < tempo_total){

        Laço sobre os domínios{
                Laço sobre as aresta e faces do contorno (se 3-D) para:
                Cálculo do campo de pressões por meio de um sistema de equações;
        }

        Laço sobre os domínios{
                Laço sobre as aresta e faces do contorno (se 3-D) para:
                Recuperação dos gradientes nodais;

                Laço sobre as arestas para:
                Cálculo do campo de velocidades;

                Laço sobre as arestas para:
                Cálculo dos termos de fluxo nodal e determinação de Δt^Ωr ;
        {
        Δt = min(Δt^Ωr ) ;

        Laço sobre os nós para:
        Cálculo do campo de saturações;

        tempo_simulação = tempo_simulação + Δt ;
}
```

Algoritmo 4.1 – Método IMPES.

A escolha de que método usar é fortemente influenciada pelo tipo de problema e isto pode trazer um ganho de tempo de simulação significativo. Embora exista uma tendência em se adotar métodos mais robustos como o totalmente implícito para simulação de reservatórios, ainda existe um interesse considerável no uso e aperfeiçoamento do método IMPES (Chen *et al*., 2004, Hurtado *et al*., 2006). Neste trabalho, é apresentado um procedimento adaptativo do método IMPES que visa reduzir o tempo total de simulação para problemas de fluxo bifásico óleo-água.

4.3 IMPES modificado

No presente trabalho, é implementada uma metodologia IMPES modificada seguindo essencialmente o trabalho de Hurtado *et al*. (2006), cuja implementação por elemento é bastante direta sendo bem mais simples e mais econômica computacionalmente que a estratégia proposta em Chen *et al*. (2004). A diferença básica entre as duas metodo-

logias se concentra essencialmente na forma de se controlar o intervalo de tempo em que o campo de pressões e de velocidades devem ser atualizados.

Na metodologia proposta por Chen *et al*. (2004), o controle do passo de tempo se baseia no controle da variação local máxima do campo de saturações, enquanto que na técnica proposta por Hurtado *et al*. (2006) este controle se dá pela variação temporal do campo de velocidades. A implementação por arestas adotada, além de ser bastante direta, pode ser utilizada indistintamente para problemas 2-D e 3-D.

A estratégia por trás do IMPES modificado adotado aqui consiste em permitir que a equação de pressão seja resolvida um número menor de vezes atribuindo passos de tempo mais longos, porém limitados, de forma a não comprometer significativamente a acurácia da solução numérica obtida. O cálculo do campo de saturações continua sendo regido pela condição de CFL. Os passos a serem seguidos podem ser melhor compreendidos através do algoritmo 4.2.

O algoritmo para o IMPES modificado faz uso de duas variáveis booleanas para o controle do avanço implícito e explícito através de $avanço\_imp$ e $avanço\_exp$ respectivamente. Durante o laço sobre as arestas para o cálculo do campo de velocidades, também é calculada uma norma $\left|\Delta\vec{v}_T\right|^n$ para esta variável e que será usada para determinar o avanço de tempo implícito Δt_P^{n+1}. Na prática, este avanço representa o intervalo de tempo em que os campos de pressões e velocidades são atualizados e é calculado como:

$$\Delta t_P^{n+1} = \frac{DVTOL}{\left|\Delta\vec{v}_T\right|^n}\Delta t_P^n \tag{4.7}$$

onde *DVTOL* é uma tolerância prescrita e $\left|\Delta\vec{v}_T\right|^n$ é uma norma para a variação do campo de velocidades longo do intervalo de tempo Δt_P^n. No início da simulação, assume-se $\Delta t_P^n = \Delta t_{CFL}$, ou seja, o mesmo passo de tempo regido pela condição de CFL. A partir de uma pequena modificação do trabalho de Hurtado et al. (2006), $\left|\Delta\vec{v}_T\right|^n$ é calculada como uma norma L^2 por arestas:

$$\left|\Delta\vec{v}_T\right|^n = \sqrt{\sum_{IJ_{\Omega_r}}\left(\frac{\left|\left(\vec{v}_{IJ}^{\,\Omega_r,n} - \vec{v}_{IJ}^{\,\Omega_r,n-1}\right)\cdot\vec{C}_{IJ}^{\,\Omega_r}\right|}{\left|\vec{C}_{IJ}^{\,\Omega_r}\right|}\right)^2}. \tag{4.8}$$

Sw(t=0)
Enquanto (tempo_simulação < tempo_total){
Laço sobre os domínios{
Laço sobre as aresta e faces do contorno (se 3-D) para:
Cálculo do campo de pressões por meio de um sistema de equações;
}

$avanço\_imp = verdadeiro$
$avanço\_exp = verdadeiro$
Se $(avanço\_exp = verdadeiro)$*{*
Laço sobre os domínios{
Se $(avanço\_imp = verdadeiro)$*{*
Laço sobre as aresta e faces do contorno (se 3-D) para:
Recuperação dos gradientes nodais;

Laço sobre as arestas para:
Cálculo do campo de velocidades e

$$\left|\Delta\vec{v}_T\right|^n = \sqrt{\sum_{IJ_{\Omega_r}}\left(\frac{\left|\left(\vec{v}_{IJ}^{\,\Omega_r,n} - \vec{v}_{IJ}^{\,\Omega_r,n-1}\right)\cdot\vec{C}_{IJ}^{\,\Omega_r}\right|}{\left|\vec{C}_{IJ}^{\,\Omega_r}\right|}\right)^2};$$

}
Laço sobre as arestas para:
Cálculo dos termos de fluxo nodal e determinação de Δt^{Ω_r} *;*
}
$avanço\_imp = falso$
$\Delta t = min(\Delta t^{\Omega_r})$
$$\Delta t_P^{n+1} = \frac{DVTOL}{\left|\Delta\vec{v}_T\right|^n}\Delta t_P^n$$
$sum\_exp = sum\_exp + \Delta t$
Se $(sum\_exp \geq \Delta t_P^{n+1})$
$avanço\_exp = falso$
Senão
Laço sobre os nós para:
Cálculo do campo de saturações;
}
$tempo\_simulação = tempo\_simulação + \Delta t_P^{n+1}$
}

Algoritmo 4.2 – Método IMPES modificado.

Esta norma representa a variação do campo de velocidades durante o intervalo de tempo Δt_P^n, obtida a partir da variação média das vazões (fluxos normais) ao longo das superfícies de controle para todos os volumes de controle da malha, onde $\sum_{IJ}$ indica que o somatório é realizado para todas as arestas da malha. No algoritmo 4.2, sum_exp acumula

os avanços de tempo do campo de saturações dentro do intervalo de tempo Δt_P^{n+1}. Enquanto o somatório dos passos de tempo da equação de saturação sum_exp for inferior a Δt_P^{n+1}, os campos de pressões e velocidades permanecem inalterados.

Com o objetivo de evitar uma variação (crescimento ou decrescimento) muito rápida no intervalo de tempo para o campo de pressões, utiliza-se uma restrição da forma:

$$0.75 \leq \frac{\Delta t_P^{n+1}}{\Delta t_P^n} \leq 1.25 \tag{4.9}$$

Portanto, se Δt_P^{n+1} calculado pela Eq. (4.7) estiver fora dos limites fornecidos pela Eq. (4.9) adota-se $\Delta t_P^{n+1} = 0.75 \Delta t_P^n$ ou $\Delta t_P^{n+1} = 1.25 \Delta t_P^n$.

A fim de utilizar esta metodologia de modo seguro, é necessário que se verifique o efeito que a mesma tem na acurácia das soluções numéricas obtidas. Inicialmente, deve-se determinar quais os valores da tolerância *DVTOL* fornecem o melhor compromisso entre acurácia e custo de CPU.

A seguir, são apresentados testes realizados com malhas de tetraedros para um problema de ¼ de cinco poços com meio homogêneo e heterogêneo. Nestes testes, procura-se mostrar para vários *DVTOL* a relação entre ganho de tempo de CPU e acurácia nos resultados de óleo recuperado e óleo acumulado para dois tipos de malhas com refinamentos de uma ordem de grandeza de diferença.

4.4 Resultados

4.4.1 Problema de ¼ de f*ive-spot* invertido com meio homogêneo e razão de viscosidade moderadamente adversa

Este exemplo é uma versão adimensionalizada do clássico problema de ¼ de *five-spot* invertido, no qual um poço injetor de água está rodeado por quatro poços produtores de óleo. Para este problema, adaptado de Durlofsky (1993) e Carvalho (2005), as saturações residuais de água e óleo são $S_{rw} = S_{ro} = 0.0$ e o meio poroso é considerado homogêneo e isotrópico com $K = \underset{\sim}{I}$ através de todo o domínio.

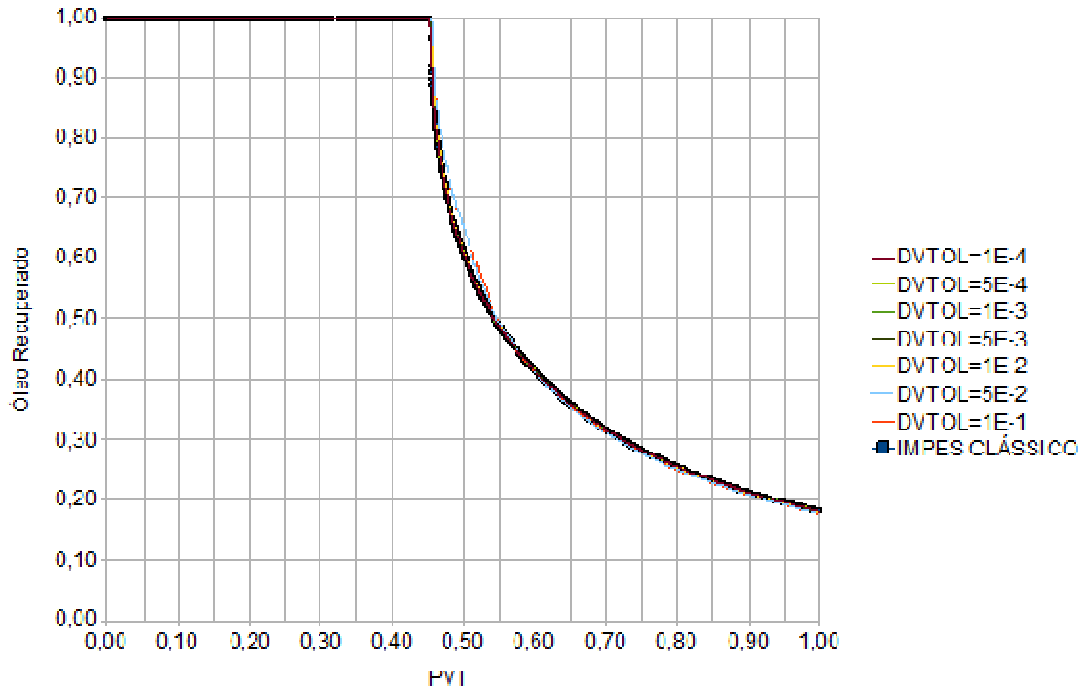
Considera-se ainda que a porosidade é constante, mas seu valor real não é relevante, já que a mesma é utilizada apenas para adimensionalizar o tempo. As viscosidades da água e do óleo são, respectivamente, $\mu_w = 1.0$ e $\mu_o = 4.0,$ de modo que a razão entre as viscosidades é $M = (\mu_o / \mu_w) = 4.0.$ As condições de contorno são $S_i = 1.0$ no poço injetor e pressões nos cantos diagonais superior esquerdo e inferior direito $P_{se} = P_{id} = 0.0.$ Admite-se ainda fluxo nulo através das fronteiras do reservatório.

Durlofsky (1993) resolveu este problema, que apresenta um efeito de orientação de malhas moderado devido à razão entre as viscosidades $M > 1.0,$ utilizando uma formulação híbrida, em que o problema pressão-velocidade foi resolvido a partir do método dos elementos finitos mistos e a equação de saturação foi resolvida utilizando um método de volumes finitos de alta ordem.

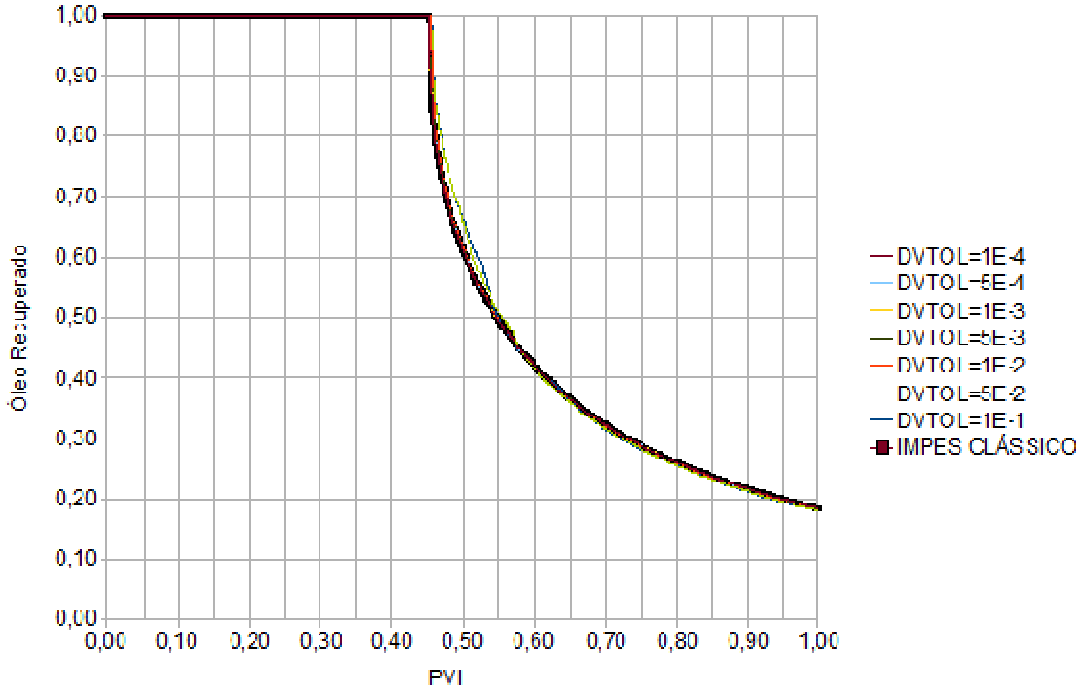
Para avaliar o efeito de orientação de malhas, Durlofsky (1993) utilizou duas malhas estruturadas e uniformes com 400 elementos triangulares, uma cujos elementos são orientados segundo a direção do escoamento (malha alinhada) e outra, cujos elementos são orientados transversalmente ao escoamento.

No presente trabalho, são utilizadas duas malhas tetraédricas não-estruturadas: uma com 2.004 nós e 7.641 elementos e outra com 10.201 nós e 45.127 elementos (malha B). Nas Figuras 4.1 (a) e (b) são apresentados os valores do óleo recuperado e nas Figuras 4.2 (a) e (b) os valores de óleo acumulado, ambos obtidos pelo método IMPES convencional e pelo método IMPES modificado para os seguintes valores de DVTOL: 10^{-4}, $5x10^{-4}$, 10^{-3}, $5x10^{-3}$, 10^{-2}, $5x10^{-2}$, 10^{-1}.

O volume poroso injetado (*Pore volume injected* – PVI) é o parâmetro usado para determinar a duração das simulações e representa o tempo adimensionalizado e calculado por $\int \frac{Qdt}{V_p}$, onde V_p é o volume total de poros do sistema e Q é a vazão total Aqui, todas as simulações são realizadas para *PVI=1.0*. Uma ampliação é feita sobre as curvas de óleo recuperado, Figs. 4.3a e 4.3b e de óleo acumulado, Fig. 4.4a e 4.4b, próxima a região de irrupção de água mostrando a ocorrência de uma forte discrepância nas diferentes curvas obtidas para diferentes valores de DVTOL.

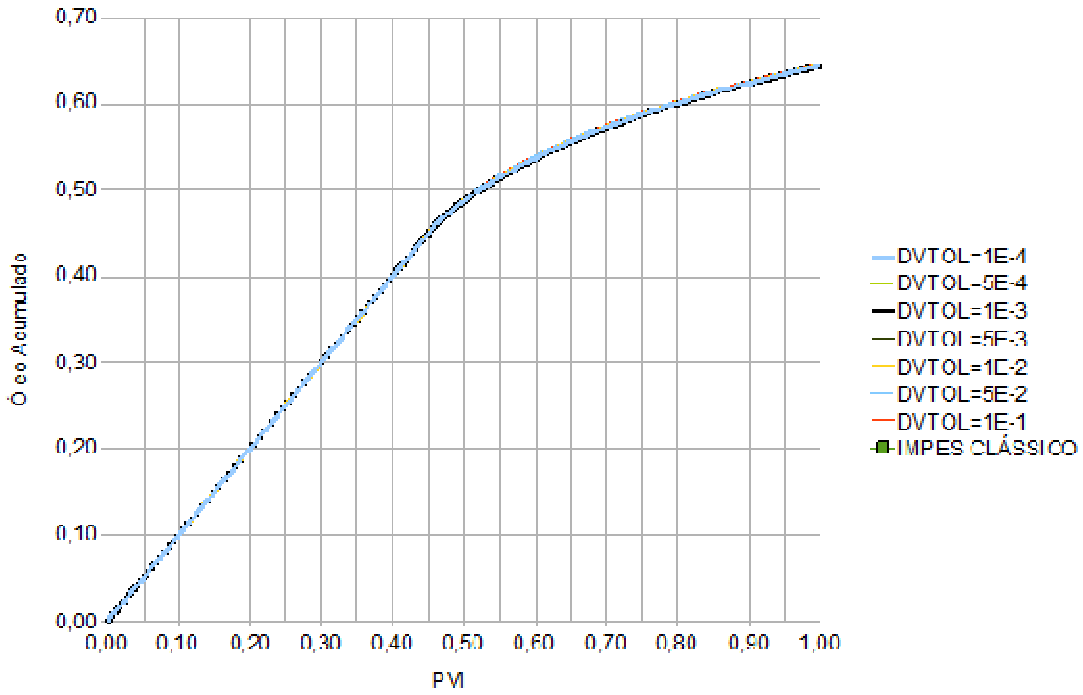


(a)

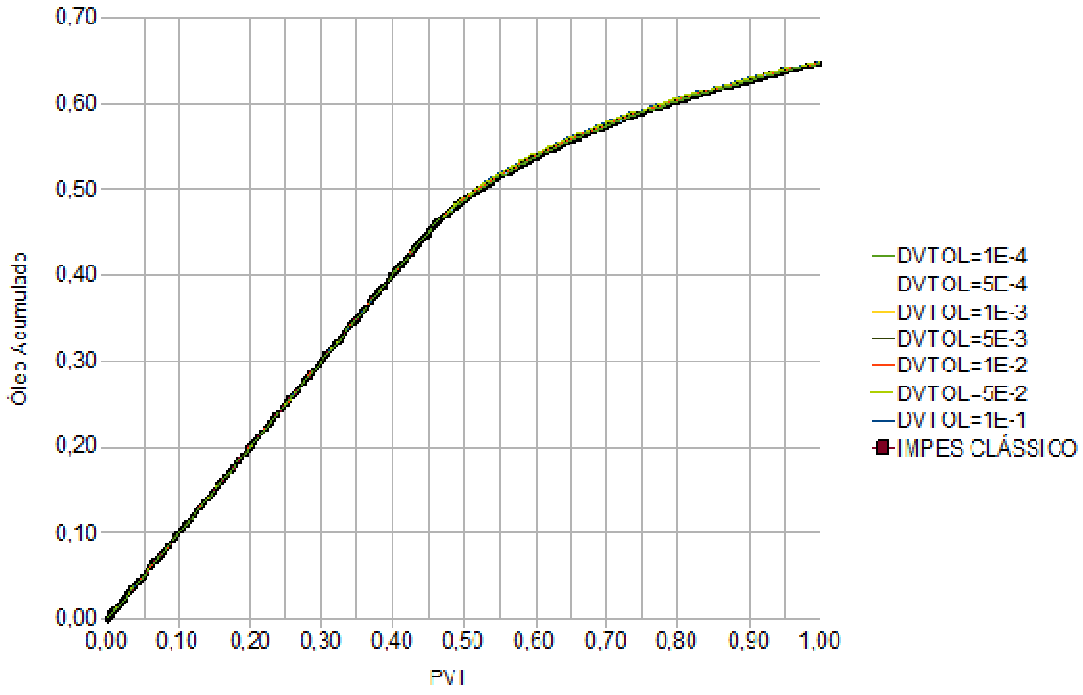


(b)

Figura 4.1 – Curvas de óleo recuperado. Vários DVTOL do IMPES modificado confrontados com o IMPES clássico para malhas de (a) 2004 e (b) 10.201 nós.

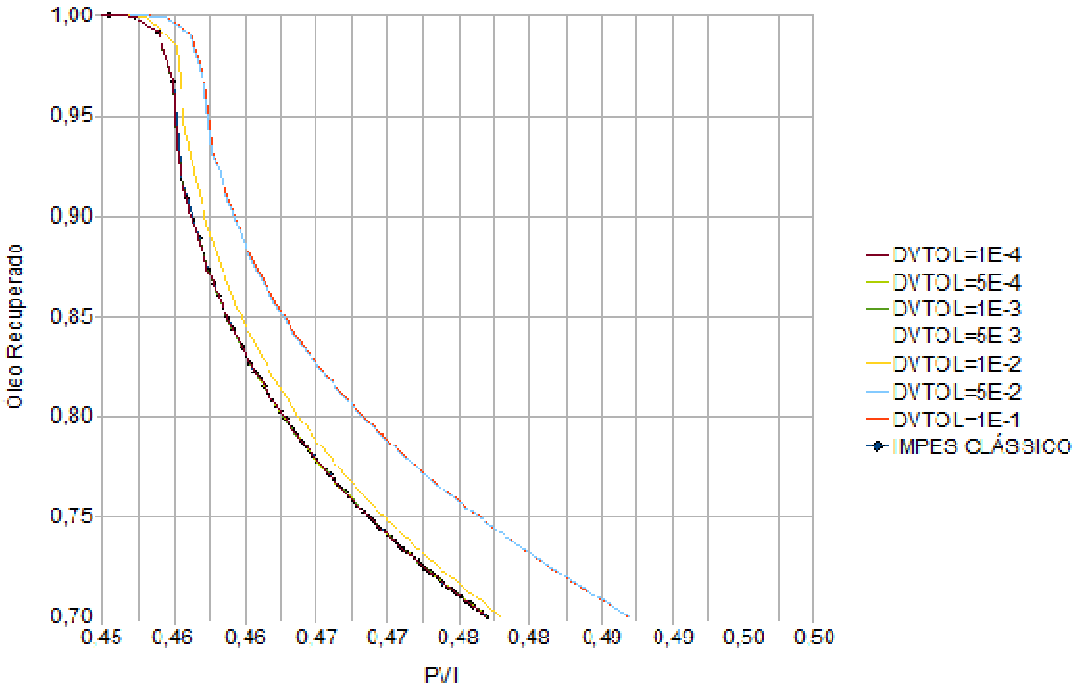


(a)

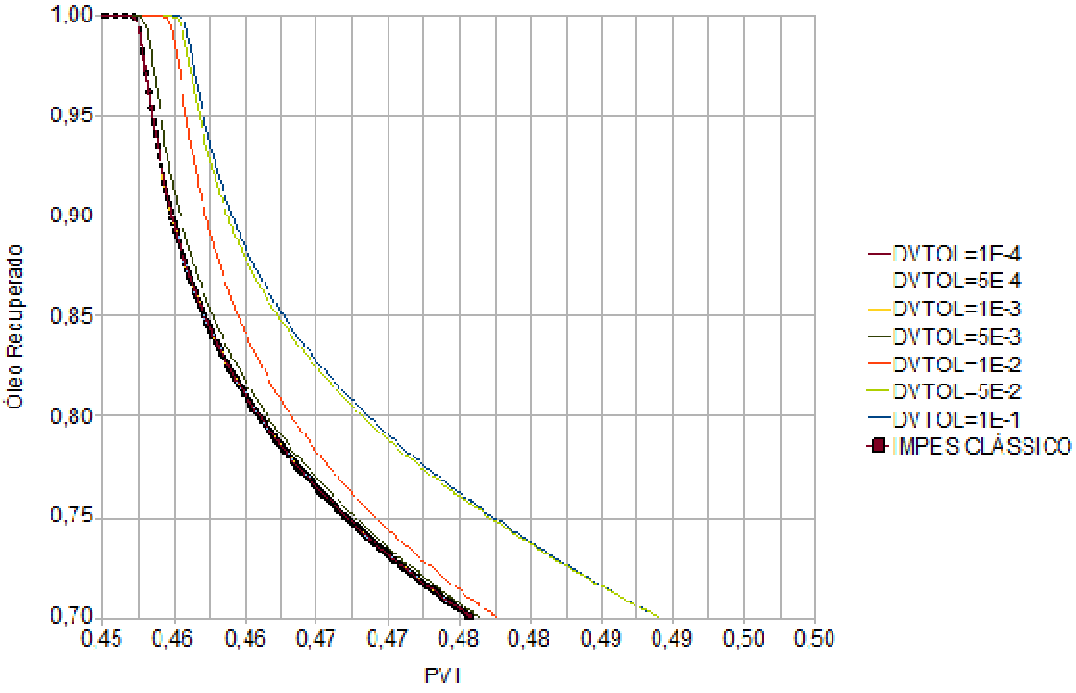


(b)

Figura 4.2 – Curvas de óleo acumulado. Vários DVTOL do IMPES modificado confrontados com o IMPES clássico para as malhas de 2004 (a) e 10.201 B (b) nós.

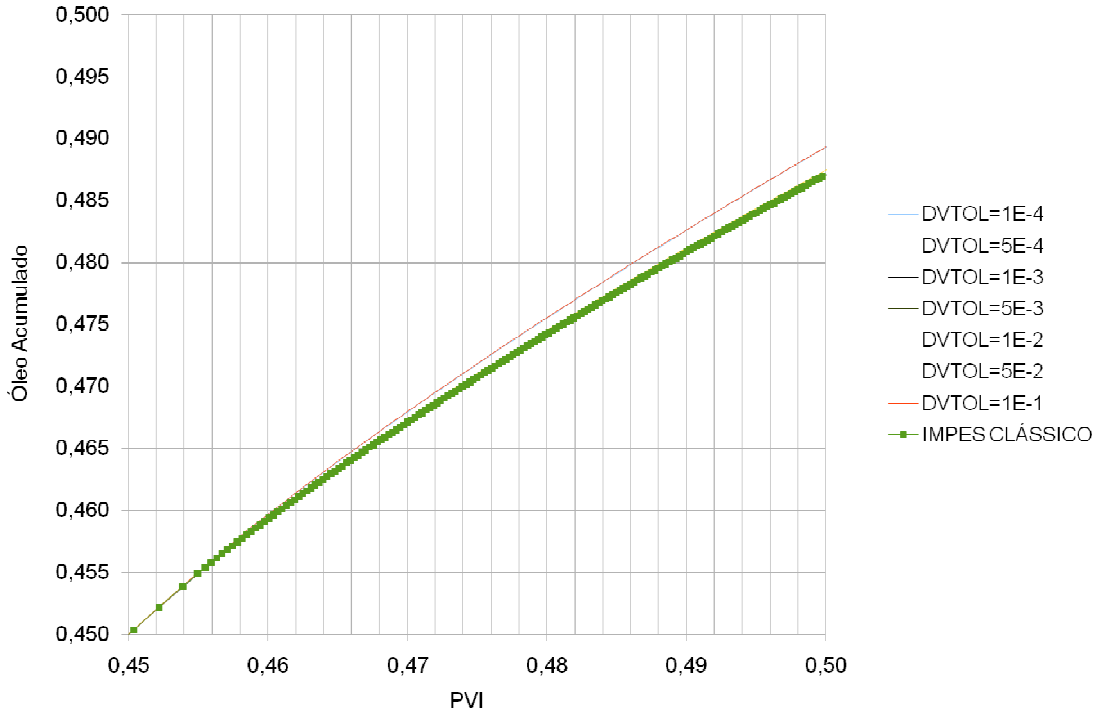


(a)

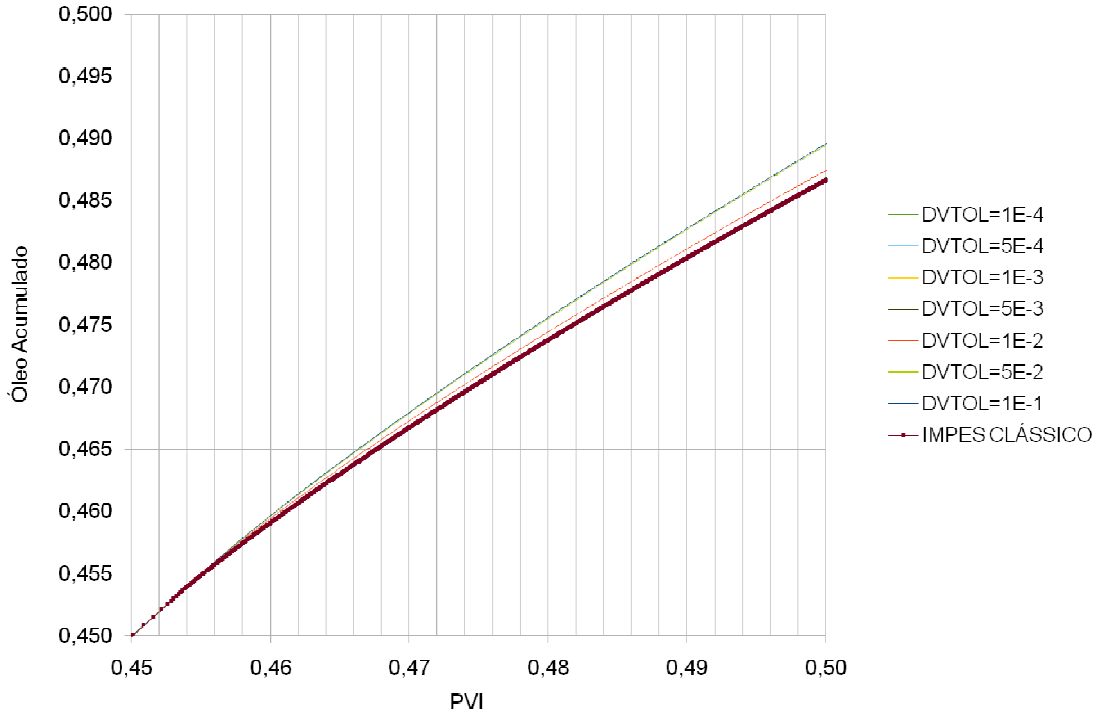


(b)

Figura 4.3 – Ampliação das curvas de óleo recuperado na região onde a água atinge o poço produtor (*breakthrough*) para as malhas de 2004 (a) e 10.201 B (b) nós.



(a)



(b)

Figura 4.4 – Ampliação das curvas de óleo acumulado destacando a discrepância entre os resultados para vários DVTOL e o IMPES tradicional para as malhas de 2004 (a) e 10.201 B (b) nós.

A partir das Figs.4.3 e 4.4 e tomando como referência o método IMPES convencional observa-se claramente que os DVTOLS que melhor reproduzem tanto a curva de óleo recuperado quanto a curva de óleo acumulado são aqueles variando entre $1.10^{-4} \leq DVTOL \leq 5.10^{-3}$. Fica claro também que, à medida que se aumenta a tolerância para valores maiores $\left(DVTOL \geq 1.10^{-2}\right)$, tanto os valores de óleo produzido quanto os valores de óleo acumulado vão se distanciando do valor referencial, havendo ainda um retardamento do tempo de irrupção de água (*breakthrough*).

Outro fator importante a ser considerado, e que na verdade é a motivação inicial para a utilização da versão modificada do método IMPES, é a questão do ganho computacional obtido com a utilização do procedimento e sua relação com a variação da acurácia do método.

Na Figura 4.5, são apresentados, no mesmo gráfico, os tempos de computação (normalizados pelo tempo gasto pelo método IMPES tradicional), bem como os erros relativos na produção acumulada de óleo, comparando-se o método IMPES tradicional com o método IMPES modificado utilizando-se os diferentes valores da tolerância DVTOL.

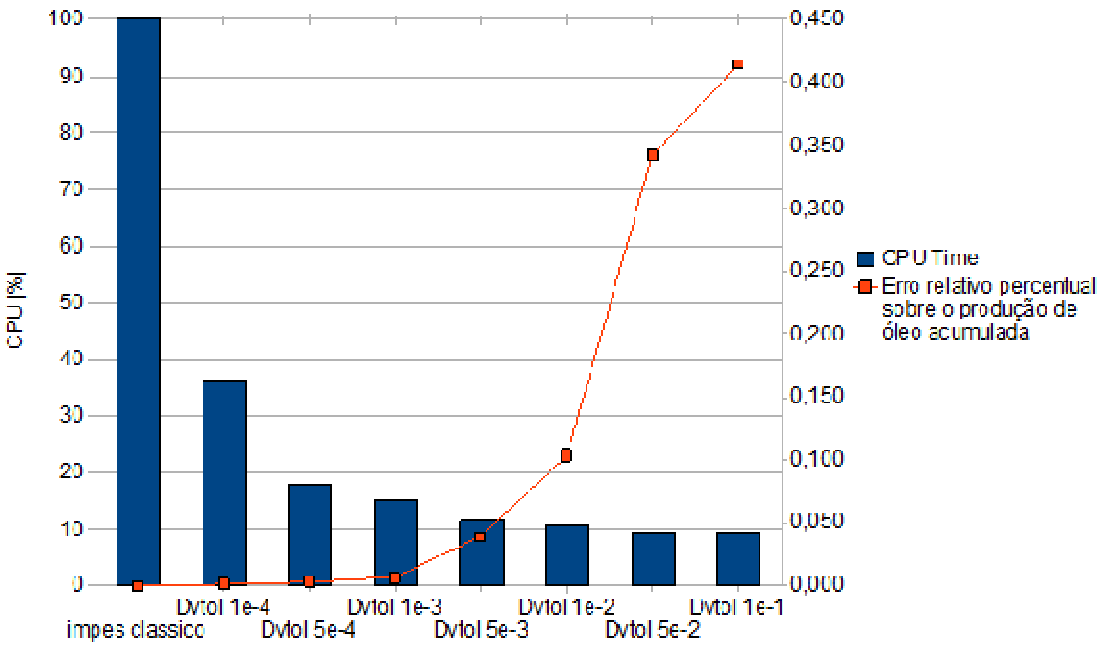


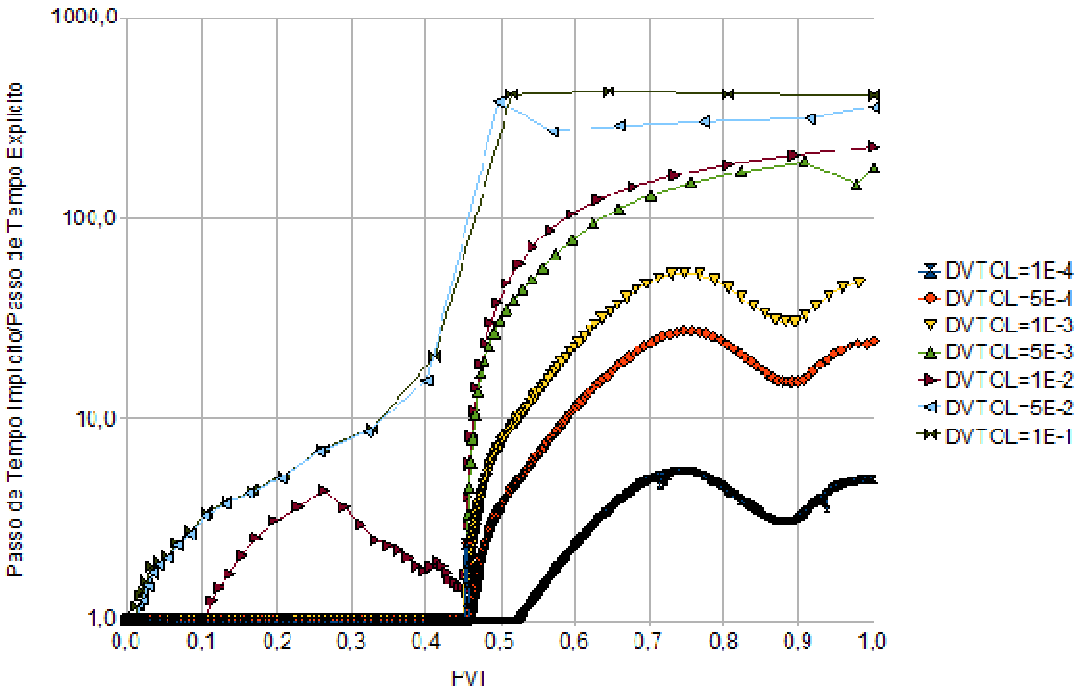
Figura 4.5 – Relação entre o tempo total de CPU e a acurácia dos resultados para vários DVTOL. É tomada como referência a curva de óleo acumulado do método IMPES tradicional. Apenas para a malha de 10.201 nós com meio homogêneo.

Novamente, fica claro nesta figura, que os valores de DVTOL que apresentam melhor concordância com o método IMPES tradicional são aqueles compreendidos no intervalo $1.10^{-4} \leq DVTOL \leq 5.10^{-3}$. Na verdade, para os valores de $DVTOL = 1.10^{-4}$ e $DVTOL = 5.10^{-4}$ os valores para o erro percentual da produção acumulada são menores que 0,004% enquanto que para $DVTOL = 1.10^{-1}$ este erro foi de 0,44% ao final de 1,0 PVI. Por outro lado, o tempo de análise decresce com o aumento do DVTOL, atingindo um valor "mínimo" para $DVTOL = 1.10^{-1}$ no intervalo considerado, como pode claramente ser visto na Fig. 4.5.

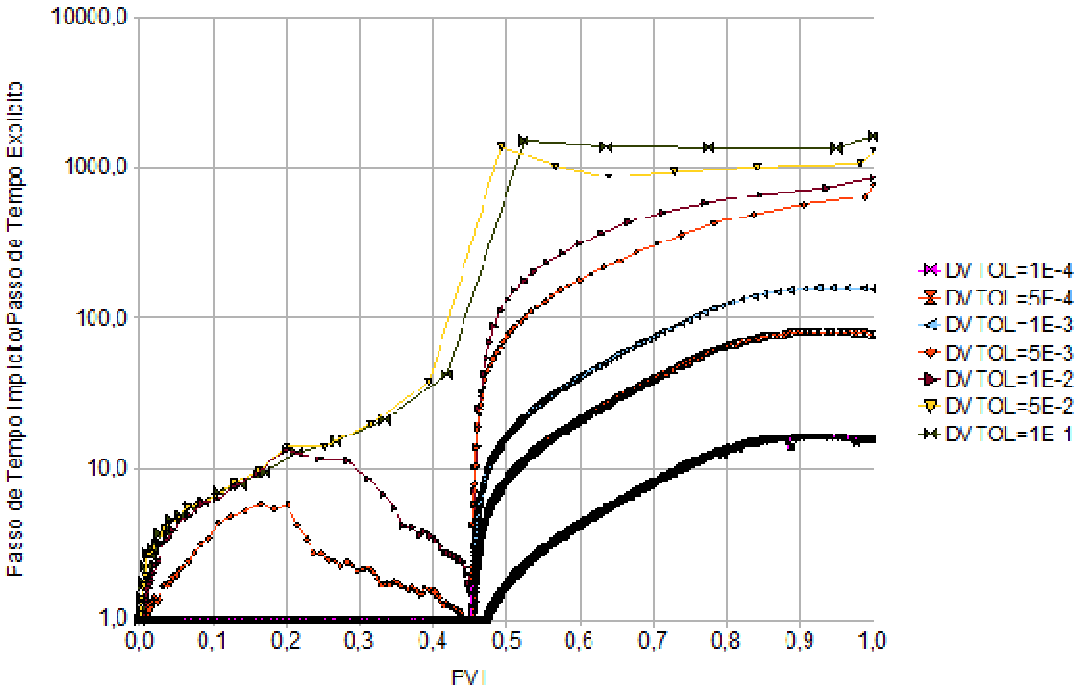
A partir da figura 4.5, pode-se ainda concluir que um bom compromisso entre acurácia e tempo de computação é obtido no intervalo $1.10^{-3} \leq DVTOL \leq 5.10^{-3}$, em que o erro relativo percentual para a produção acumulada de óleo varia entre $0,00619\% \leq Err \leq 0,039\%$ e o tempo relativo percentual de computação entre as metodologias IMPES e IMPES modificado varia entre $15,05\% \leq TR \leq 11,65\%$.

As figuras 4.6a e 4.6b mostram a razão entre o tamanho do passo de tempo implícito (avanço do campo de pressões e de velocidades) pelo tamanho do passo de tempo explícito (avanço do campo de saturações) para as malhas de 2.004 e 10.201 nós, respectivamente. Percebe-se que para a malha menor e para a faixa de DVTOL que apresentam resultados mais próximos do IMPES tradicional, o método só começa a acelerar a partir da chegada da água ao poço produtor $(PVI \cong 0.45)$.

Com a malha de 10.201 nós, a curva com $DVTOL = 5 \cdot 10^{-3}$, por exemplo, cujos resultados não se distanciam demasiadamente do IMPES tradicional, permite que se ganhe algum tempo de simulação antes do *breakthrough.*



(a)



(b)

Figura 4.6 – Relação entre o tamanho do passo de tempo implícito pelo passo de tempo explicito para as malhas de 2004 (a) e 10.201 B (b) nós.

4.4.2 Problema de ¼ de Cinco Poços Heterogêneo com Razão de Viscosidade Moderadamente Adversa.

Neste exemplo, todas as condições iniciais e de contorno são idênticas as do problema anterior, onde a única modificação feita foi a introdução de uma barreira quadrada no interior do domínio com $K = 10^{-2} \cdot \underset{\sim}{I}$ a fim de avaliar se este tipo de configuração poderia causar alguma modificação importante na determinação do DVTOL ótimo de simulação.

Novamente, foram utilizadas duas malhas tetraédricas com as 2021 e 10.448 nós. Nas Figuras 4.7 (a) e (b) e 4.8 (a) e (b) são apresentadas, respectivamente, os valores do óleo recuperado e do óleo acumulado, ambos obtidos pelo método IMPES convencional e pelo método IMPES modificado para os mesmos valores de DVTOL: 10^{-4}, $5x10^{-4}$, 10^{-3}, $5x10^{-3}$, 10^{-2}, $5x10^{-2}$, 10^{-1} utilizados na aplicação anterior.

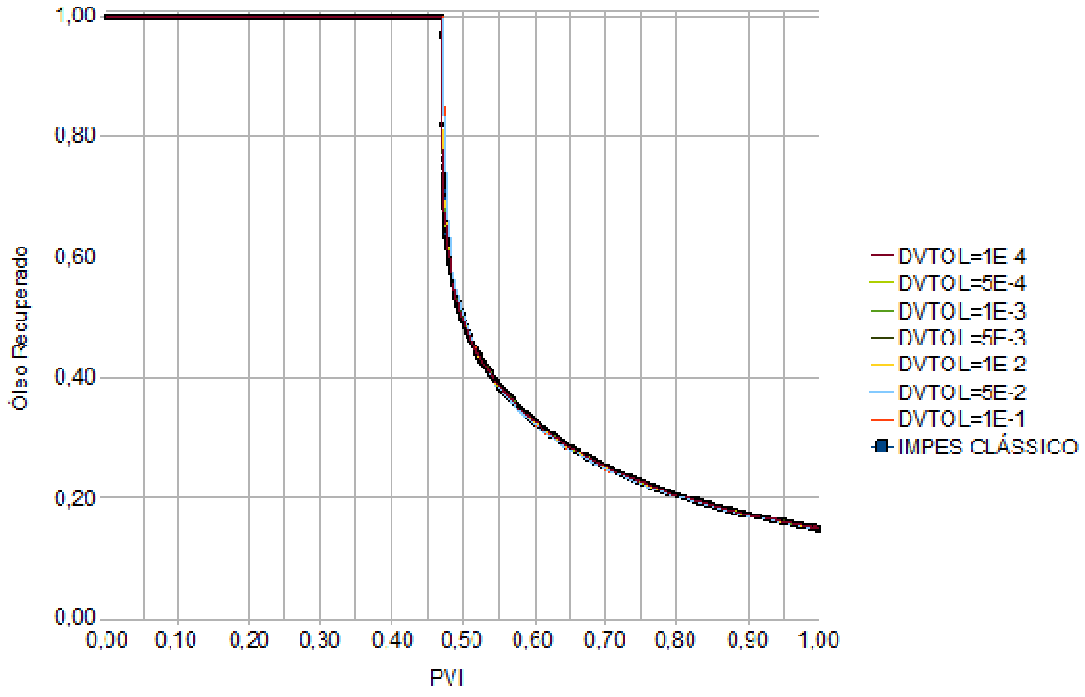
As Figuras 4.9 (a) e (b) apresentam uma ampliação para a curva do óleo recuperado na região próxima a irrupção de água e nas figuras 4.10 (a) e (b) apresenta-se outra ampliação para as curvas de óleo acumulado numa região onde ocorre uma discrepância nas diferentes curvas obtidas para diferentes valores de DVTOL.

A partir das figuras 4.9 e 4.10 e tomando como referência o método IMPES convencional, observa-se claramente que os DVTOLS que melhor reproduzem, tanto a curva de óleo recuperado, quanto a curva de óleo acumulado são aqueles variando entre $1.10^{-4} \leq DVTOL \leq 5.10^{-3}$. Fica claro também que, à medida que aumentamos a tolerância para valores maiores $\left(DVTOL \geq 1.10^{-2}\right)$, tanto os valores de óleo produzido quanto os valores para o óleo acumulado vão se distanciando do valor referencial, havendo ainda um forte retardamento do tempo de irrupção de água.

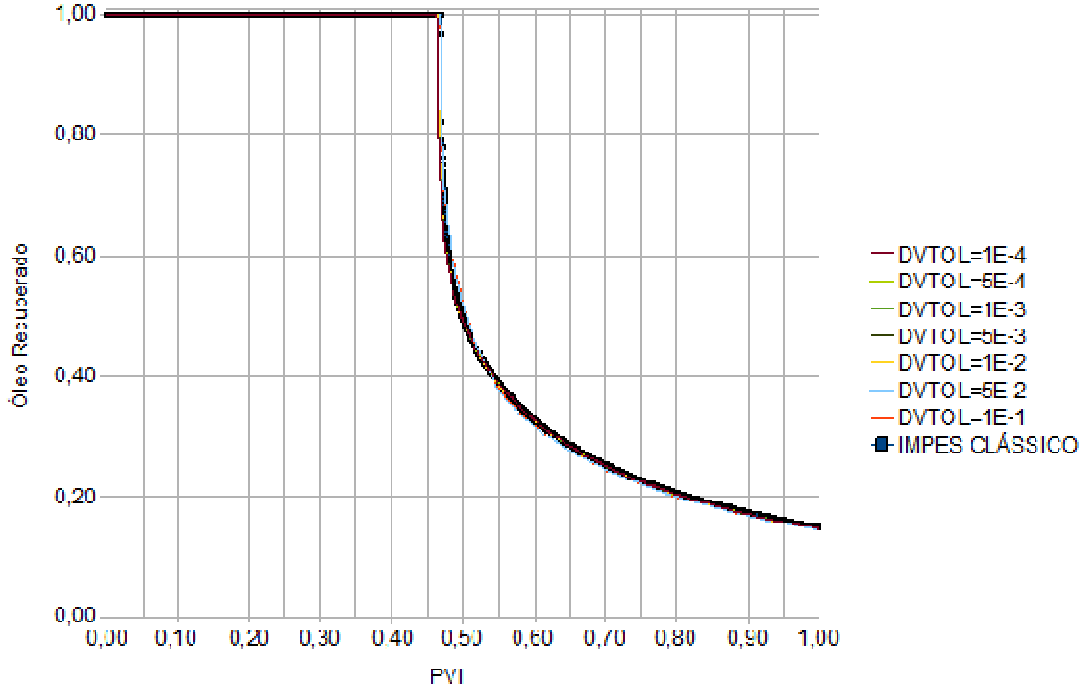
Na Figura 4.11 são apresentados no mesmo gráfico os tempos de computação (normalizados pelo tempo gasto pelo método IMPES clássico), bem como os erros relativos na produção acumulada comprando-se com o método IMPES tradicional como o método IMPES modificado utilizando-se os diferentes valores de DVTOL.

Novamente, fica claro nesta figura, que os valores de DVTOL que apresentam melhor concordância com o método IMPES clássico são aqueles compreendidos no intervalo $1.10^{-4} \leq DVTOL \leq 5.10^{-3}$. Na verdade, para os valores de $DVTOL = 1.10^{-4}$ e

$DVTOL = 5.10^{-4}$ os valores para o erro percentual da produção acumulada são menores que 0,004% enquanto que para $DVTOL = 1.10^{-1}$ este erro foi de 0,44% ao final de 1,0 PVI.

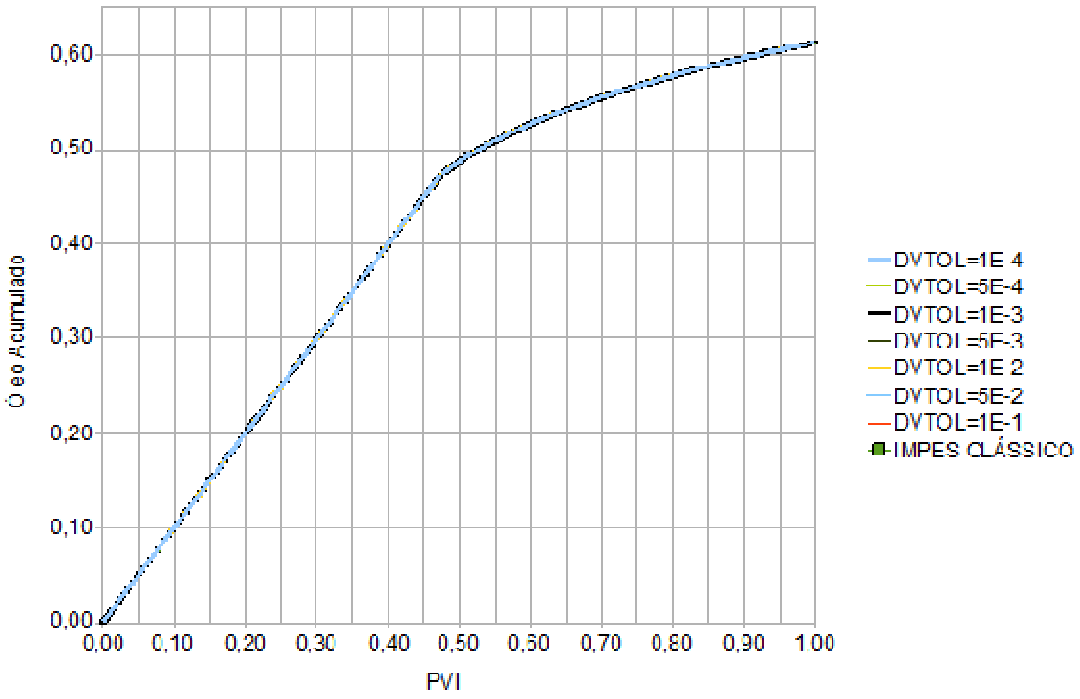


(a)

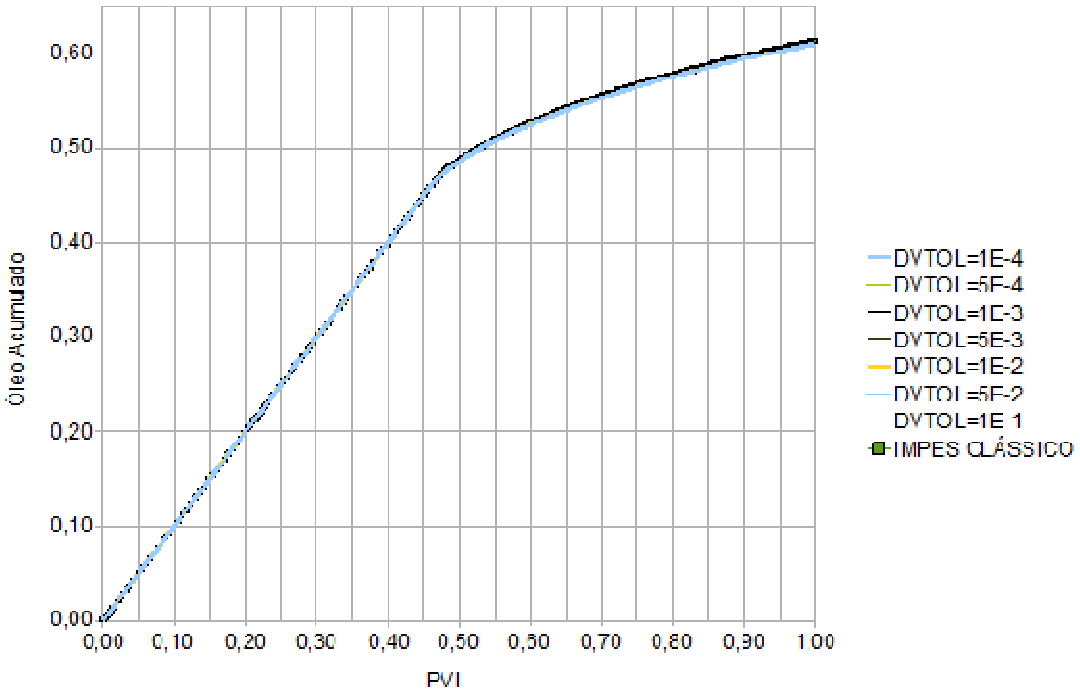


(b)

Figura 4.7 – Curvas de óleo recuperado para malhas de 2021 (a) e 10.448 (b) nós.

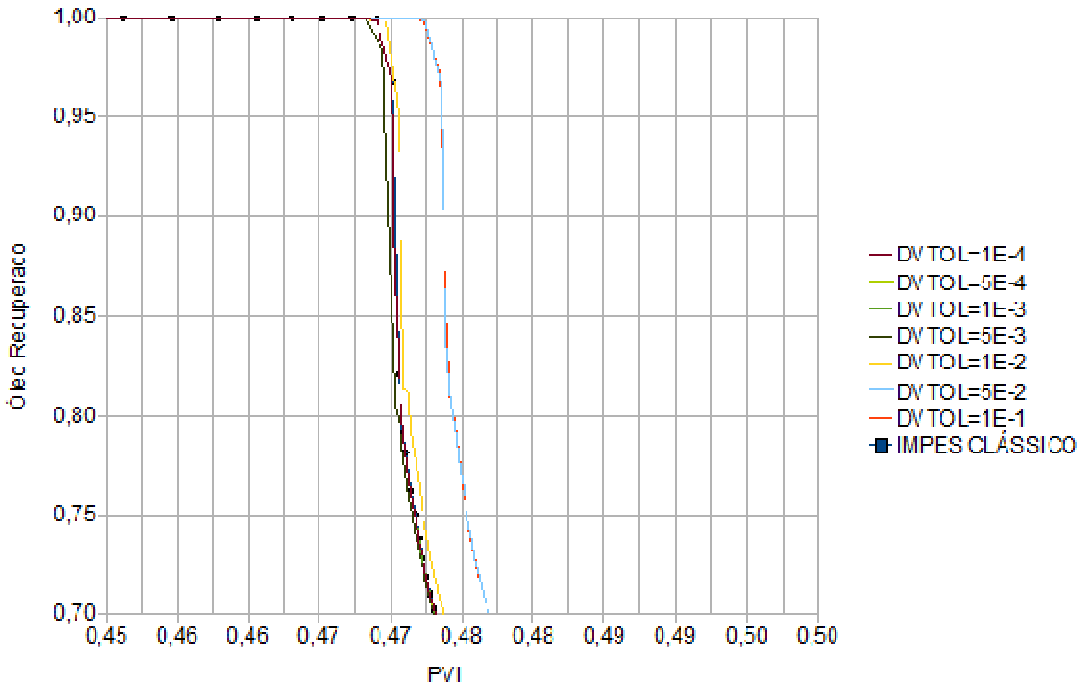


(a)

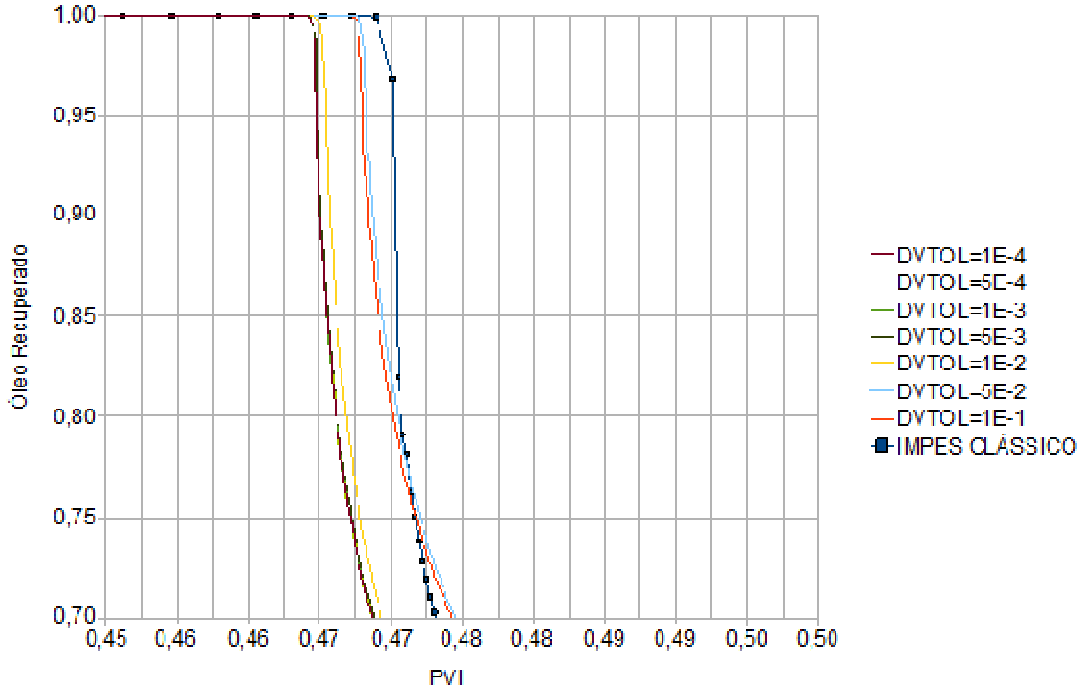


(b)

Figura 4.8 – Curvas de produção acumulada para malhas de 2021 (a) e 10.448 (b) nós.

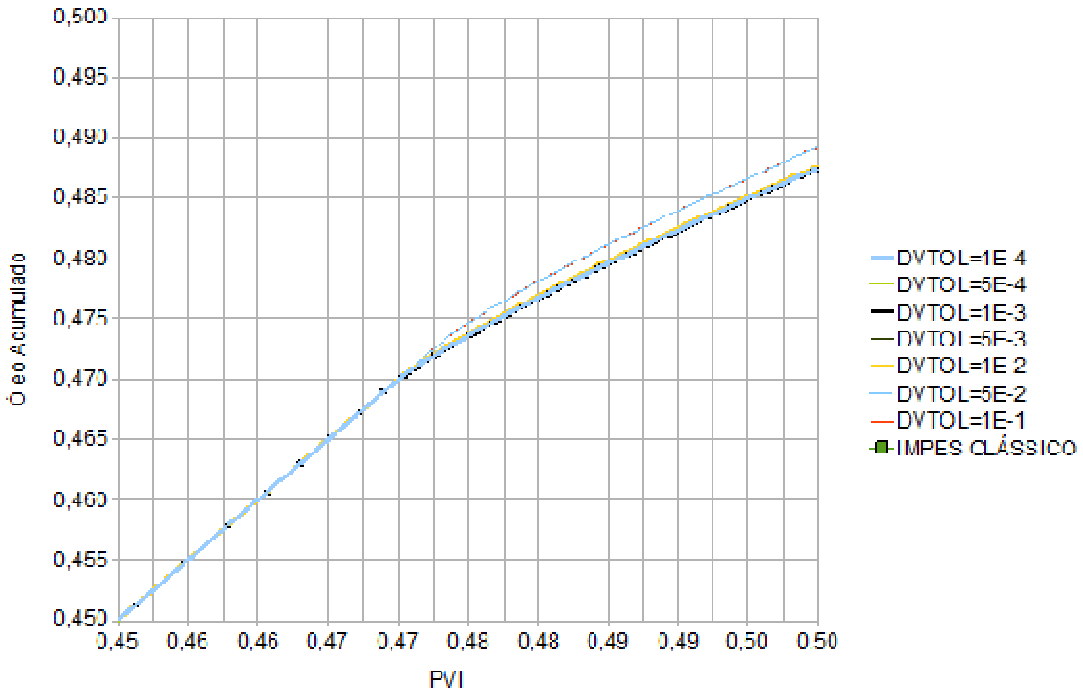


(a)

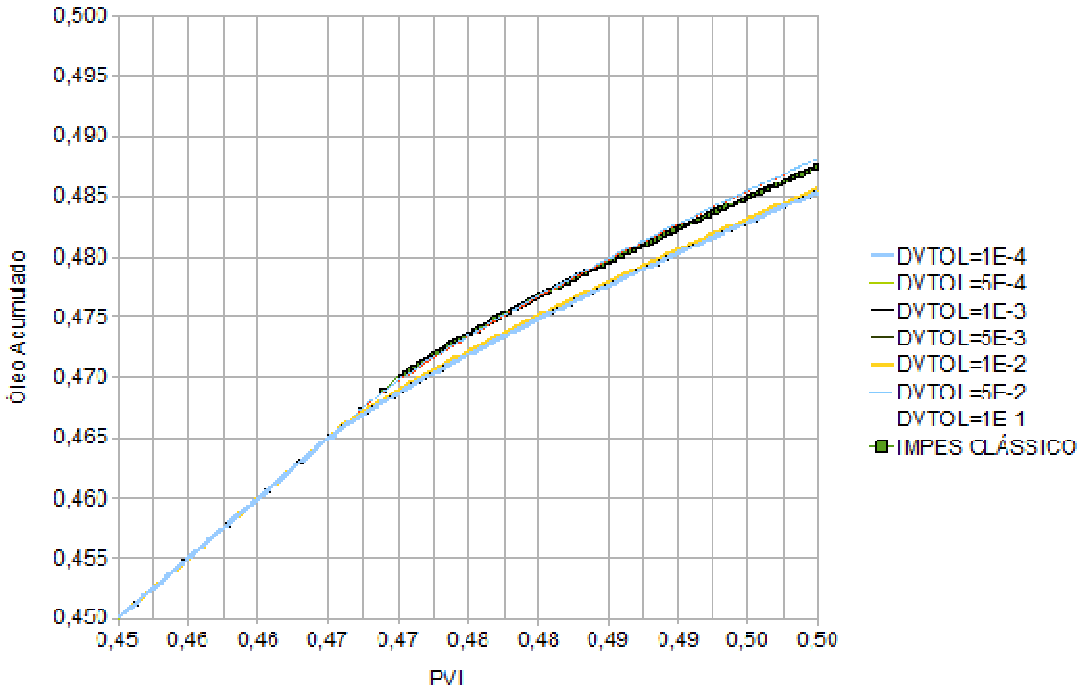


(b)

Figura 4.9 – Ampliação das curvas de óleo recuperado para o instante onde a água chega ao poço produtor.

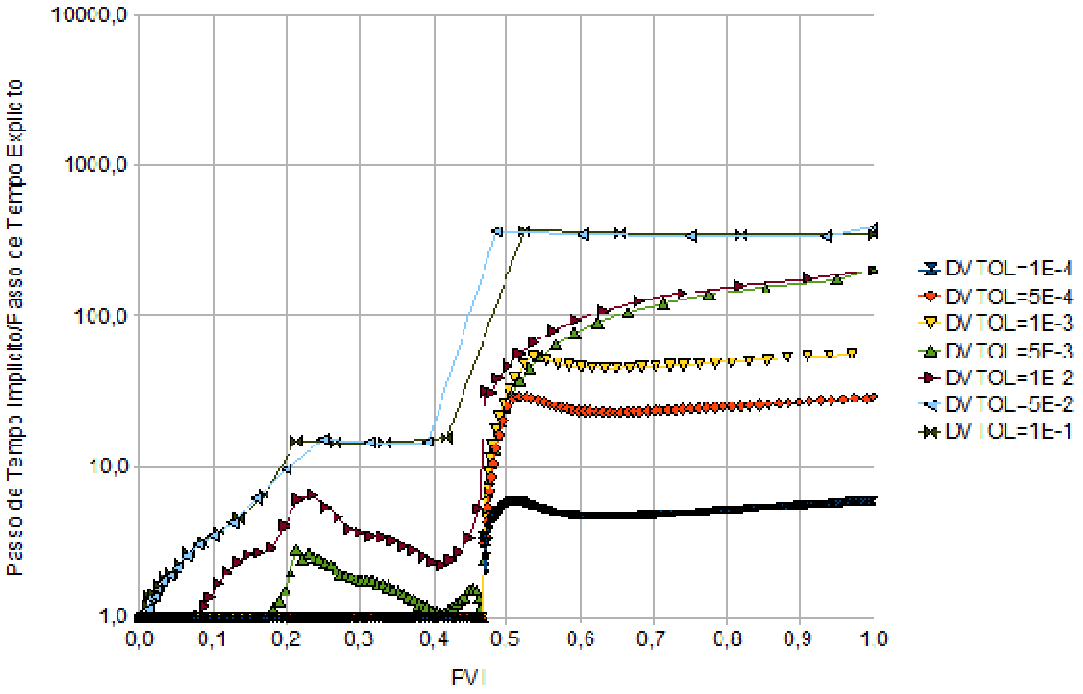


(a)

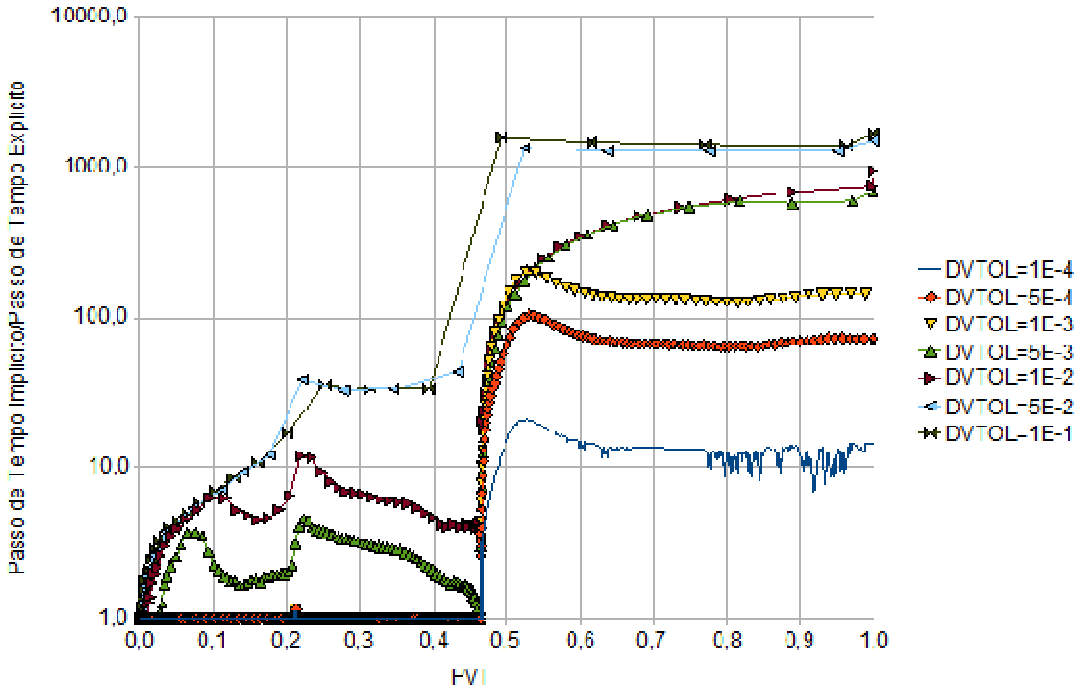


(b)

Figura 4.10 – Ampliação das curvas de óleo acumulado.



(a)



(b)

Figura 4.11 – Relação entre o tamanho do passo de tempo implícito pelo passo de tempo explícito para uma malha de 2021 (a) e 10.448 (b) nós respectivamente.

As figuras 4.12 (a) e (b) mostram a relação entre o tamanho do passo de tempo implícito (avanço do campo de pressões e de velocidades) pelo tamanho do passo de tempo explícito (avanço do campo de saturações) para as malhas de 2.000 e 10.000 nós. Nos dois exemplos, todos os DVTOLS que oferecem os melhores resultados, apenas aceleram a simulação a partir do ponto de irrupção de água com exceção do $DVTOL = 5 \cdot 10^{-3}$ como aconteceu no caso homogêneo. Isso mostra que é possível já obter um ganho computacional antes do instante de $(PVI \cong 0.45)$.

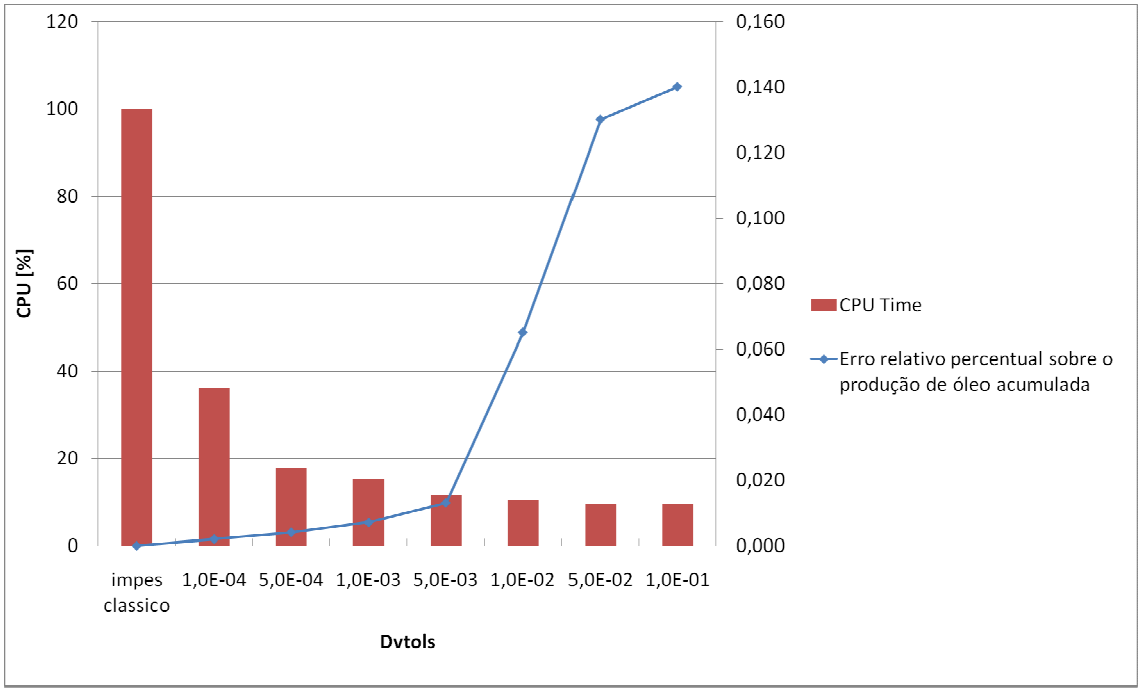


Figura 4.12 – Relação entre o tempo total de CPU e a acurácia dos resultados para vários DVTOL. É tomada como referência a curva de óleo acumulado do método IMPES tradicional. Apenas para a malha de 10.448 nós com meio heterogêneo

4.4 Outros métodos: Seqüencial Implícito, Totalmente Implícito e Adaptativo Implícito-Explícito

O método seqüencial implícito e o método totalmente implícito (*fully implicit*), também conhecido como método de solução simultânea são métodos mais robustos do que o IMPES e empregados em problemas com maior grau de acoplamento implicando em uma não-linearidade mais forte das equações. O método seqüencial implícito aparece como

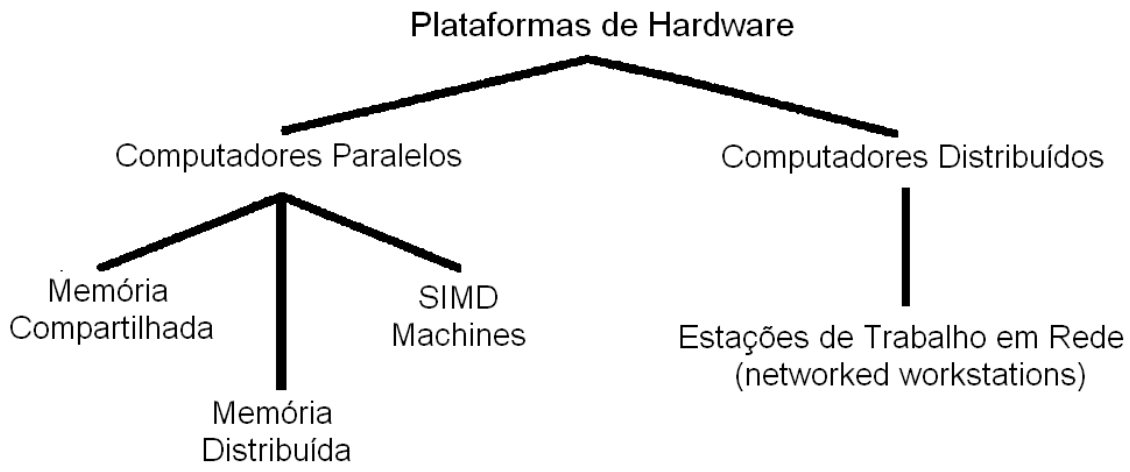
um método intermediário em termos de eficiência computacional e robustez quando comparado ao IMPES e ao totalmente implícito. Já este último aparece no extremo de máxima robustez e máximo uso de CPU e memória sendo o mais indicado na resolução de problemas utilizando o modelo *black-oil*. Para problemas de grande porte, o método de solução simultânea pode se tornar um grande desafio em termos de custo computacional. O método adaptativo implícito-explícito tenta aliar a economia do método IMPES e a robustez do método totalmente implícito (Chen *et al*. 2006).

Vale a pena enfatizar que, neste trabalho, o propósito de construir um simulador de reservatórios em computadores paralelos já traz uma série de desafios a serem suplantados e a escolha do método IMPES (modificado) como método de integração no tempo foi feito apenas devido a simplicidade de implementação do método.

5. COMPUTAÇÃO PARALELA

5.1 Introdução

Aplicações de computador podem ser geralmente classificadas como aplicações seriais, distribuídas ou paralelas. Aplicações seriais, onde uma instância de uma aplicação é executada em um único processador, representa a grande maioria dos softwares usados pelas pessoas em seus computadores pessoais. Aplicações distribuídas fazem uso de múltiplos computadores que contem tarefas independentes que não interagem entre si. Um tipo de tarefa distribuída pode ser visto nas simulações de Monte Carlo, onde tipicamente o mesmo algoritmo é executado várias e várias vezes com diferentes parâmetros de entrada. As aplicações em paralelo, o foco deste trabalho, também fazem uso de vários computadores, mas contem tarefas interdependentes que trocam dados enquanto estão sendo executadas. A figura 5.1 apresenta de forma geral a taxonomia de importantes classes de computadores paralelos e distribuídos.



Fonte: Fujimoto, M. R, Parallel And Distributed Simulation Systems, A Wiley-Interscience publication, 2000.

Figura 5.1 – Taxonomia de importantes classes de computadores paralelos e distribuídos.

Dentro da classe de computadores paralelos, onde se insere o presente trabalho, os computadores de memória compartilhada e de memória distribuída, conhecidos como computadores MIMD (*Multiple – Instruction, Multiple – Data stream*), têm em comum o fato dos processadores executarem instruções de forma completamente independentes. Já os computadores de arquitetura SIMD (*Single – Instruction, Multiple – Data stream*), todos

os processadores completam a execução de uma instrução antes de prosseguir para outra instrução.

Máquinas paralelas com arquitetura SIMD se diferenciam das duas anteriores por usarem processadores mais especializados (processadores vetoriais) e exigirem *hardware* especificamente projetado para usá-los (Flynn, 1996) (Fujimoto, 2000).
Embora os *clusters*, uma coleção de Wokstations/PCs conectados em rede, tenham prevalecido como a arquitetura mais usada em aplicações de computação alto desempenho, (HPC) *High Performance Computing*, (Strohmaier et al, 2005), computadores vetoriais como o *Earth Simulator* mostram que muitas aplicações científicas podem se beneficiar dos computadores vetoriais (Strohmaier et al, 2005).

Os primeiros testes de simulação realizados neste trabalho foram feitos usando um *cluster* (zumbi) adquirido pelo grupo de processamento de alto desempenho (PADMEC) da UFPE que contem 48 nós, onde cada nó possui dois processadores de 32bits, 2GB de RAM, HD 40GB. A rede é controlada por 3 *switches* 3 Com *Fast Ethernet* e o sistema operacional é o *Linux*.

Os resultados de desempenho e escalabilidade foram obtidos usando um *cluster* mais moderno comparado ao zumbi e que faz parte do Núcleo de Atendimento em Computação de Alto Desempenho (NACAD) da COPPE/UFRJ. O *cluster* do NACAD é um SGI Altix ICE 8200 com as seguintes configurações:

- 38 CPUs Quad Core Intel Xeon: 152 cores – 64 bits
- Memória: 304 Gbytes RAM (distribuída)
- Armazenamento em disco: SGI Infinite Storage NAS (32 TBytes)
- Sistema operacional: Suse Linux Enterprise Server + SGI ProPack
- Compiladores: Intel e GNU (Fortran-90 e C/C++) com suporte OpenMP e MPI
- Rede: Inifiniband e Gigabit

Embora todo o desenvolvimento do simulador tenha sido pensado para funcionar em computadores de memória distribuída, vale mencionar o surgimento de um novo paradigma em computação paralela com o uso de máquinas de memória distribuída-compartilhada, *distributed shared-memory* (DSM), nas comunidades científicas. O interesse é fazer o melhor uso destes dois sistemas onde a combinação destas duas metodologias

de paralelização pode fornecer uma forma mais efetiva para explorar os modernos sistemas com memória distribuída e compartilhada (Hughes, 2003)

Simulações de grande porte que requerem computadores de alto desempenho são importantes em muitas áreas da engenharia. Problemas que envolvem dinâmica dos fluidos, eletromagnetismo, acústica, entre outros, geralmente são definidos por equações diferenciais parciais não-lineares dependentes do tempo que consomem um grande tempo de computação.

A simulação em reservatórios de petróleo representa outra área da engenharia onde o uso da computação paralela é imprescindível. Além do grande número de blocos usados na discretização do reservatório, que pode conter de centenas de milhares a milhões, dependendo do modelo usado (*black-oil*, composicional ou térmico, por exemplo), o problema pode conter várias incógnitas como pressão, saturação das fases, composição e temperatura (Chen, 2004).

A seguir, são descritos os procedimentos e conseqüentes esclarecimentos a respeito da paralelização do simulador de fluxo óleo-água em reservatórios de petróleo abordando a filosofia paralela adotada, o tipo de linguagem de programação e das ferramentas auxiliares usadas e uma descrição do simulador e de seu pré-processador.

5.2 Filosofia adotada na implementação

A computação paralela tem como principal filosofia a divisão de uma tarefa em outras menores onde cada uma destas é chamada de processos. Uma das premissas para se atingir o máximo desempenho em simulações paralelas é atribuir um processo a uma única unidade de processamento (CPU).

Projetar um programa para ser executado em paralelo requer uma forma diferente de pensar daquela usada para se construir um programa puramente seqüencial. Um programa paralelo é construído para ser executado com se fosse um único programa onde cada processador executa uma cópia deste e processa apenas uma fração do problema total. A simulação de reservatórios de petróleo, por exemplo, faz uso de uma malha que representa o reservatório discretizado a ser simulado. Todos os processadores executam o mesmo programa, porém lidando com partes distintas do problema.

Em computadores de memória distribuída, cada processo atua de forma assíncrona, ou seja, por possuírem uma CPU e uma memória própria (acesso local), estes trabalham de

forma independe uns dos outros. Eventualmente, há necessidade de troca de informações entre os processos (acesso remoto), pois lidam com as partes de um todo.

O programa paralelo deve ser projetado de tal modo que todos os processos envolvidos troquem entre si o mínimo de informações possíveis. O tempo de acesso a uma informação remota é muito maior do que a uma informação armazenada localmente podendo ser superior uma ordem de grandeza ou mais (Fujimoto, 2000).

O tempo de troca de uma mensagem de tamanho L entre processadores $\tau_C(L)$ é fortemente dependente do *hardware* utilizado e é comumente associado a dois parâmetros que exercem uma relação linear com o tempo total demonstrado pela seguinte expressão:

$$\tau_C(L) = \tau_S + \beta L. \tag{5.1}$$

onde τ_S é a latência (ou atraso) que representa o tempo requerido para enviar um pacote de mensagem da fonte para o destino. A largura de banda, β, expressa geralmente em bits por segundo, refere-se à quantidade de dados que podem ser transferidos por segundo.

A distribuição do trabalho entre os processadores tem um grande impacto no desempenho geral da simulação. O item 5.4.3 apresenta mais detalhes sobre como a distribuição de carga afeta a simulação paralela.

5.3 Decomposição de domínios

Técnicas de decomposição de domínios são largamente usadas em computação paralela. Este método divide o domínio computacional em *n* subdomínios tal que um processador trabalha sobre apenas um subdomínio. Como o mesmo problema é dividido entre *n* processadores, espera-se um ganho significativo de tempo (*speed-up*), pois o tempo total de processamento *T* é reduzido a *T/n*.

Em geral, as técnicas de decomposição de domínios se dividem entre decomposição com e sem sobreposição de domínios. Na primeira, também conhecida como *overlapping*, os nós da região sobreposta pertencem a mais de um subproblema e na outra (*non-overlapping*) apenas os nós comuns da fronteira entre subdomínios pertencem a mais de um subproblema. Esta ultima, é a técnica adotada neste trabalho e tem como vantagem a redução computacional dos subproblemas.

5.4 Programação Orientada a Objetos

Uma introdução sobre programação orientada a objetos dá uma idéia de como os programas que são projetados podem se beneficiar desse paradigma.

A definição mais aceita nas comunidades de programadores sobre uma linguagem orientada a objetos é que esta deve suportar diretamente:

- Abstração.
- Hierarquia.
- Polimorfismo.

o que inclui as seguintes linguagens: Ada95, Beta, C++, CLOS, Eiffel, Simula, Smalltalk entre outras.

Por abstração, que não significa "vago" ou "impreciso", entende-se a capacidade de representar conceitos de forma direta e independente por meio de classes. Na construção de um gerenciador de malhas, o conceito de nó, por exemplo, é bastante claro, pois representa uma entidade com características bem definidas como uma numeração global (*ID*) e três coordenadas espaciais (*x,y* e *z*). É por meio das classes que são criados os objetos. Para este exemplo, valeria a seguinte linha de código:

```
CNode node;
```

onde *CNode* é a classe ou o conceito por trás de um nó, e o objeto *node* é o nó de fato, e que pode acessar e modificar dados a este associados por meio de funções definidas em *CNode*.

A hierarquia está associada à reutilização de classes, onde outras novas são criadas baseadas nas existentes e que possuem componentes em comum. Em problemas de adaptação, por exemplo, deseja-se que uma entidade possa acessar seus filhos ou pais. Assim como *node* é um objeto, um objeto aresta (*edge*), que deu origem a duas outras arestas em um processo de refinamento, deve ser capaz de acessar suas entidades filhos.

Uma classe C*Entity* pode fornecer o meio de obter estes filhos não apenas para o objeto aresta, mas como também para qualquer outro objeto que represente uma entidade da malha (triângulo, quadrilátero, tetraedro, prisma, pirâmide, etc) que tenha sido refinado. O trecho de código abaixo exemplifica tal relação:

```
// definição de um objeto edge
CEdge edge;

// acesso às entidades filhos por meio de uma função escrita na classe
// CEntity da qual CEdge é derivada
edge.getLeaves();
```

Se houvessem outros objetos representando outros tipos, a função *getLeaves* poderia ser usada para cada um destes objetos sem a necessidade de escrever novas linhas códigos.

Por fim, o conceito de polimorfismo fecha as definições do que se espera que uma linguagem orientada a objetos tenha. O polimorfismo permite que membros de classes possam ser (re)definidos em classes derivadas. Uma função hipotética *getType()* definida em *CEntity* pode assumir várias formas (implementações distintas) em cada classe derivada informando, por exemplo, que tipo de entidade ela é. Usando a sintaxe de C++, um possível trecho de código seria:

```
virtual CType getType() const=0;
```

A palavra virtual indica que getType pode servir de interface para outras classes que venham a ser derivada desta. Aqui, getType não faz nada. É puramente virtual. Dentro da definição da classe CEdge, que herda CEntity, getType passa a ter um propósito claro e que está relacionado com a classe CEdge retornando um objeto hipotético do tipo CType.

```
getType CType getType(){
      return edgeType;
}
```

Na descrição do simulador no item 5.5.2 funções virtuais são usadas para permitir que uma equação possa ser resolvida por mais de uma formulação numérica.

5.4.1 A linguagem C++

Uma linguagem usada essencialmente em todos os domínios de programação, desde *drivers* para dispositivos às aplicações industriais e acadêmicas de grande porte, de aplicações para os mais modestos computadores aos mais avançados supercomputadores, C++ tem sido amplamente adotada por pesquisadores, projetistas e desenvolvedores profissionais ao redor do mundo.

Criada por Bjarne Stroustrup na década de 1980, a linguagem de programação orientada a objetos C++ é baseada principalmente na linguagem C, definida pelo próprio inventor como um superconjunto desta, e na linguagem Simula67, da qual herdou os conceitos de classes, herança e funções virtuais (Stroustrup, 1997). Tomando ainda como inspiração o conceito de sobrecarga de operadores da linguagem Algol68, C++ se tornou uma das linguagens mais populares no meio científico.

Como C, C++ não foi especificamente projetada para computação científica. No entanto, muitas computações numéricas, científicas e de engenharia foram feitas em C++ (Stroustrup, 2000). Uma importante razão para isto é que o trabalho numérico tradicional freqüentemente deve ser combinado com gráficos e com computações baseadas em estruturas de dados que não se ajustam à forma de Fortran (Stroustrup, 2000). O principal poder de C++ está na sua capacidade de ser eficazmente usada para aplicações que abrangem diversas áreas de aplicação.

A tarefa de projetar um programa passa basicamente por três estágios: compreender de forma clara o problema (análise), identificar os principais conceitos envolvidos na solução (projeto) e expressar a solução em um programa (programação). Geralmente, os dois primeiros estágios só se tornam realmente claros e bem definidos quando se tenta expressá-los e executá-los em um programa de forma aceitável. Neste momento é que a escolha da linguagem de programação faz diferença.

As principais razões que fazem C++ ser uma linguagem usada em aplicações de diversas naturezas são:

- Há compiladores C++ para essencialmente todos os tipos de máquinas, desde computadores pessoais (PC), passando por w*orkstations* e *mainframes*, até super computadores.
- A padronização da linguagem torna-a portável entre diferentes plataformas.

- C++ foi desenvolvida para ter uma eficiência equiparável a C, uma linguagem escolhida para sistemas operacionais e outros sistemas que exigem alto desempenho.
- C++ é uma linguagem de múltiplos propósitos e não está voltada para um campo particular da programação.
- C++ permite que se construam interfaces que possam acessar bibliotecas construídas em FORTRAN, beneficiando-se, assim, de rotinas já implementadas e testadas.

Quando a linguagem C++ foi padronizada por um comitê internacional ISSO/ANSI no final de1997, uma biblioteca padrão foi incluída ao desenvolvimento de C++. O STL (*Standard Template Library)* é uma biblioteca genérica que oferece soluções de gerenciamento de coleções de dados junto com o uso de algoritmos modernos e eficientes. Todos os componentes da STL são classes *templates*, o que significa que estas podem ser usadas para tipos arbitrários de elementos.

O STL dispensa o programador de criar *arrays* dinâmicos, listas encadeadas, árvores binárias ou algoritmos de busca. Basta que se escolha o tipo apropriado de container e se use as funções membros e algoritmos para processar os dados.

O uso desta biblioteca é constante tanto no simulador, item 5.5.2, quanto no código do pacote FMDB, item 5.4.4, onde toda a estrutura de dados para gerenciar a malha se beneficia das estruturas primitivas do STL.

5.5 Ferramentas computacionais paralelas

Como dito anteriormente, projetar um programa para ser executado em paralelo requer uma forma diferente de pensar quando comparado a um programa serial. Uma série de desafios surge à medida que se aumenta a complexidade do programa, seja na distribuição da malha entre os processadores ou na resolução de um sistema de equações em paralelo. Todas são tarefas que exigem um certo grau de experiência em programação paralela e muito tempo de desenvolvimento. Assim, a tentativa de produzir códigos para tratar tarefas complexas em paralelo demanda um longo tempo de trabalho desviando, assim, do foco da construção do simulador. Isto pode resultar em programas pouco robustos ou ineficientes.

Uma série de pacotes de código aberto desenvolvidos por terceiros permite que não se reinvente a roda a cada momento que se deseje realizar algum trabalho que envolva tro-

ca de mensagens entre processos. Esses pacotes são bibliotecas que foram incorporados ao simulador de forma a poupar tempo e obter resultados mais eficientes e acurados. São eles:

- AutoPack
- FMDB
- ParMetis
- PETSc

Estes quatro pacotes de código aberto, que serão detalhados mais adiante, são todos escritos em linguagem C com exceção do FMDB que é escrito em C++. Esta linguagem foi escolhida para o desenvolvimento do simulador por ser uma linguagem orientada a objetos – um paradigma para escrever "bons" programas para um conjunto de problemas (Stroustrup, 2000) e que será apresentada logo adiante no item 5.4.1.

O envio e recebimento de mensagens entre processos é uma tarefa complexa e que é exigida pelos quatro pacotes citados. Este tipo de tarefa é realizado pelo MPI, uma biblioteca que se tornou um padrão na passagem de mensagens em computadores paralelos de memória distribuída. Todos estes pacotes são utilizados na construção do pré-processador e do simulador paralelo e onde ambos são detalhados nos itens 5.5.1 e 5.5.2, respectivamente.

5.5.1 Interface paralela de mensagens (MPI)

Atualmente, o padrão predominante de programação paralela de memória distribuída é baseado na passagem de mensagens usando MPI (*Message Passing Interface*). Uma tecnologia com mais de 15 anos, o MPI é uma biblioteca de rotinas para escrever programas paralelos por passagem de mensagem de forma eficiente e portátil. MPI não é uma linguagem, mas uma especificação de rotinas que podem ser chamadas de um programa (El-Rewini, 2005). Embora a portabilidade dos programas baseados no MPI seja garantida, o desenvolvimento destes tem um alto custo (Reed, 2003).

Como alternativas para tornar mais fácil o projeto de programas paralelos, pode-se mencionar: o *Co-Array Fortran* (CAF) e o *Unified Parallel C* (UPC) (Reed, 2003). Ambas são extensões das linguagens Fortran 95 e C, respectivamente, e são voltadas para o desenvolvimento em computação de alto desempenho em máquinas paralelas de grande porte.

Com suporte a hardwares de memória distribuída e compartilhada estes dois sistemas de programação podem oferecer grandes vantagens em termo de portabilidade, desempenho e produtividade.

5.5.2 Particionamento de malha e balanço de carga (ParMetis)

A primeira preocupação que surge quando se faz simulação paralela usando malhas não-estruturadas é como distribuí-las entre os processadores. O particionamento de malhas em computadores de memória distribuída exige que algumas medidas sejam tomadas a fim de não comprometer o desempenho geral da simulação, pois caso contrário, uma simulação serial poderia ser mais rápida do que com vários processadores. Primeiro, deve-se garantir que cada processador receba a mesma quantidade de nós, se possível, de forma a evitar um desbalanço de trabalho onde um processador pode ficar sobrecarregado enquanto outro permanece ocioso. Segundo, o número de nós com cópias remotas deve ser mínimo de forma a reduzir a numéro de troca de informações entre processadores durante a simulação.

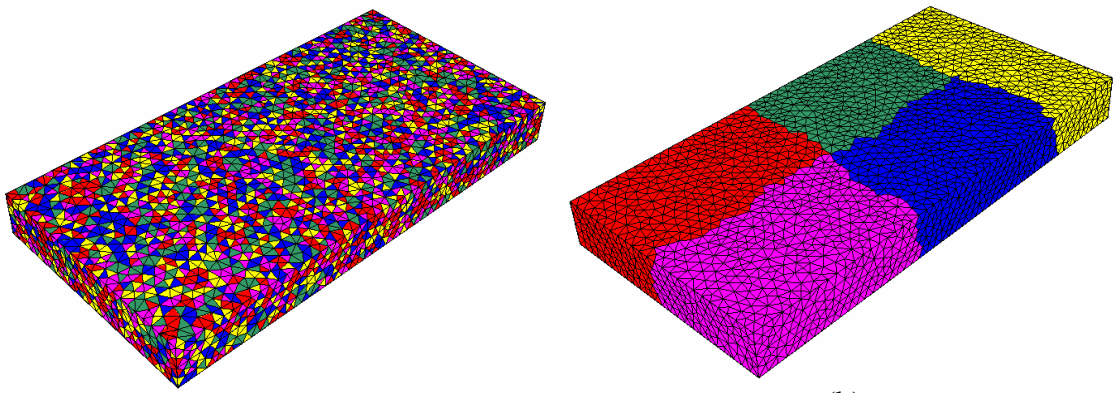
Diversos pacotes disponíveis na internet foram desenvolvidos com o intuito de otimizar a interface das partições. Alguns exemplos podem ser citados como: ParMetis, Zoltan , Jostle e Scotch. Neste trabalho, o ParMetis foi adotado, pois é usado internamente pelo gerenciador de malhas FMDB.

A Fig. 5.2 mostra um exemplo de um particionamento paralelo de uma malha 3-D de tetraedros que é distribuída entre cinco processos de duas formas diferentes. Cada processo, que é representado por uma cor, recebeu a mesma quantidade de elementos garantindo um equilíbrio aproximado de carga de trabalho. Na Fig. 5.2a, a malha é distribuída seguindo a ordem em que as conectividades dos elementos aparecem no arquivo produzido pelo gerador de malhas, ou seja, para uma malha com mil elementos, por exemplo, os duzentos primeiros elementos da lista presentes no arquivo são destinados ao processo 0, os duzentos seguintes para o processo 1 e assim por diante. Dessa forma, o número de nós com cópias remotas se torna extremamente alto e por isso indesejado, pois os processos passam a consumir mais tempo trocando mensagens entre si do que realizando os cálculos da simulação.

Na Fig. 5.2b, o ParMetis, baseado no particionamento da Fig. 5.2a, consegue determinar uma distribuição ótima onde não apenas o balanço de carga é garantido como

também o número de nós nas fronteiras das partições é reduzido drasticamente, o que reflete diretamente no tempo total gasto no envio e recebimento de dados.

O ParMetis apenas informa qual será a distribuição dos elementos, i.e, informa para qual processador cada elemento deve ser enviado. Este trabalho de envio, que é realizado uma única vez, fica a cargo do usuário que deve implementar algum algoritmo de distribuição de dados. No presente trabalho, a biblioteca FMDB, que será apresentada a seguir no item 5.4.4, foi usada com esse intuito.



(a) (b)

Figura 5.2 – Exemplo de uma malha 3-D de tetraedros particionada entre 5 processadores. À esquerda (a), o particionamento considera apenas o balanço em termos do número de elementos, ou seja, cada processador tem basicamente a mesma quantidade de elementos. A direita (b), particionamento otimizado baseado no ParMetis onde o número de entidades com cópias remotas é reduzido significativamente.

5.5.3 Estrutura de dados distribuída (FMDB)

O ponto inicial para a construção de qualquer simulador que lida com malhas não-estruturadas é definir que tipo de estrutura de dados será usado para acessar os elementos (nós, aresta, faces e volumes) da malha ao longo da simulação. Se tratando de computação paralela, outra preocupação que surge na ausência de um gerador de malhas paralelo é como distribuir a malha entre os processadores e obter informações entre elementos nas fronteiras das partições.

A construção de rotinas eficientes para esses tipos de tarefas é complexa e requer muito tempo e experiência do programador. Por esses motivos e tendo de concentrar esforços na construção de um simulador que demanda muito trabalho foi pesquisado uma biblioteca que fosse capaz, por exemplo, de construir uma estrutura de dados por aresta com malhas 2-D/3-D distribuídas a partir de uma malha de elementos finitos.

O FMDB (*Flexible Distributed Mesh Database*) (Seol, 2005) é um pacote com uma série de rotinas escritas em C++ que se apresenta como uma boa opção no gerenciamento de malhas seriais ou distribuídas em computadores paralelos. Fazendo uso constante da biblioteca padrão do C++ (STL), o FMDB também utiliza o ParMetis e o Autopack na distribuição da malha entre processadores, tendo o MPI como base para todas as rotinas de envio e recebimento de dados remotos.

A flexibilidade que o próprio nome FMDB passa vem do fato de que é possível criar estruturas de dados de acordo com as necessidades do usuário, ou seja, pode-se criar uma estrutura com entidades que geralmente não são fornecidas pelo gerador de malhas como as arestas.

As estruturas de dados obtidas através do FMDB podem auxiliar, por exemplo, em problemas envolvendo *multigriding* onde é necessário conhecer as entidades que circundam um dado elemento.

Outro ponto interessante deste pacote é a possibilidade de trabalhar com adaptação de malhas onde elementos filhos são criados a partir dos pontos médios das arestas que compõe cada elemento pai. Dessa forma, um elemento triangular dá origem a quatro novos triângulos e um tetraedro dá origem a oito novos tetraedros. Embora não tenha sido explorado, o FMDB se propõe a tratar o desbalanceamento de carga gerado quando a malha distribuída é refinada ou desrefinada.

A construção da estrutura de dados do simulador começa com a definição de um objeto do tipo *mMesh* como mostra a seguinte linha de código:

```
mMesh *theMesh;
```

A classe *mMesh* representa uma interface para manipular todos os elementos da malha que pode também ser distribuída. Por meio desta classe são criadas as estruturas de nós e de elementos baseadas nas conectividades dos mesmos e que são oriundos do gerador de malhas. A construção de um elemento tetraédrico, por exemplo, pode ser realizado da seguinte forma:

```
// primeiro são criados os nós do tetraedro
mVertex* v1 = theMesh->createVertex(x,y,z,ID1);
mVertex* v2 = theMesh->createVertex(x,y,z,ID2);
mVertex* v3 = theMesh->createVertex(x,y,z,ID3);
mVertex* v4 = theMesh->createVertex(x,y,z,ID4);
```

```
// o tetraedro é criado em seguida
mRegion* tet = theMesh->createTetWithVertices(v1,v2,v3,v4);
```

Inicialmente, os vértices do tetraedro são definidos através das coordenadas espaciais e de suas respectivas numerações globais por meio da função *createVertice*, e em seguida é criado o elemento tetraédrico por meio da função *createTetWithVertices*.

Criando uma rotina de automatização, todos os nós e elementos do arquivo de malha podem ser passados para a estrutura criada pelo FMDB à medida que novos elementos são inseridos. No caso de malhas distribuídas, o FMDB fornece uma função que lê alguns tipos de arquivos de malha entre eles os do gerador Gmsh. A tarefa de leitura e distribuição da malha é completamente automatizada por meio da chamada da função *M_load* como mostrado a seguir:

```
M_load(theMesh,fileName);
```

Onde *fileName* é um ponteiro do tipo *char** e refere-se ao nome do arquivo a ser aberto. Esta função realiza de forma serial a leitura do arquivo de malha para montar uma estrutura de nós e de elementos. Em seguida, uma estrutura de arestas é criada para, com o auxílio do ParMetis, distribuí-la entre todos os processos.

A redistribuição de uma malha com as características apresentadas pela Fig. 5.2(a), onde a leitura da malha foi feita de forma paralela e cada processo recebeu uma fração dos elementos da malha, para atingir a configuração da Fig. 5.2(b) será alvo de trabalhos futuros. Isso permite que malhas que representam problemas de grande porte possam ser carregadas na memória.

Embora a forma como a função *M_load* lida com o carregamento da malha possa parecer um empecilho para malhas com milhões de nós isso não é um problema grave aqui, pois o FMDB permite que, após a distribuição da mesma entre os processos, os elementos de cada partição possam ser subdivididos até que um nível de refinamento seja atingido. Essa característica também será tema de trabalhos futuros.

O membro *modifyState*, definido na classe *mMesh*, é responsável pela criação das estruturas de dados adicionais como a de arestas. Esta, criada em função dos tetraedros, pode ser construída por meio de uma única chamada:

```
theMesh->modifyState(3,1);
```

As entidades da malha (vértices, arestas, faces e tetraedros) possuem dimensões 0, 1, 2 e 3, respectivamente, e é por meio destes valores que uma entidade é diferenciada da outra. A função *modifyState* permite criar qualquer tipo de estrutura de vizinhanças para uma entidade como, por exemplo, arestas ao redor de arestas, tetraedros ao redor de nós, faces ao redor de arestas, etc.

No caso do simulador aqui desenvolvido, a formulação de volumes finitos empregada necessita apenas dos nós e das arestas da malha tornando, assim, o uso da função *modifyState* necessária apenas na fase de pré-processamento onde a estrutura de arestas é criada e os coeficiente geométricos calculados. No simulador, as arestas podem ser criadas de forma semelhante à criação dos tetraedros como mostrado a seguir:

```
// criam-se primeiro os nós
mVertex* v1 = theMesh->createVertex(x,y,z,ID1);
mVertex* v2 = theMesh->createVertex(x,y,z,ID2);

// em seguida as arestas
mEdge* edge = theMesh->createEdge(v1,v2);
```

Vale lembrar ainda que a classe *mMesh* fornece funções do tipo *get* para obter um entidade qualquer como, por exemplo, *getEdge(v1,v2)*

```
mEdge *edge = theMesh->getEdge(v1,v2);
```

e uma função para remoção da entidade da estrutura de dados *DEL(entity)*

```
theMesh->DEL(edge);
```

útil em problemas de adaptação.

Cada entidade da malha é definida por uma classe (*mVertex, mEdge, mFace* e *mRegion*). Então, um nó ou tetraedro é um objeto do tipo *mVertex* e *mRegion*, respectivamente. Todas estas classes são derivadas de uma classe base chamada *mEntity* como mostra o diagrama de herança na Fig. 5.3.

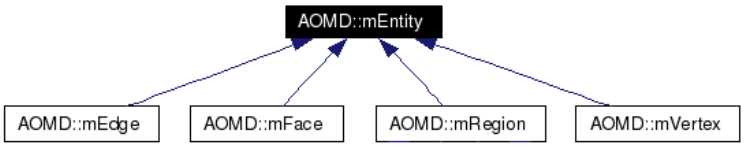


Figura 5.3 – Diagrama de classes para *mEntity*.

Todas as classes são agrupadas dentro do *namespace* AOMD (*Algorithm Oriented Mesh Database*) do qual o FMDB herdou, aperfeiçoou e adicionou recursos para o gerenciamento de malhas distribuídas.

Um objeto do tipo *mEntity* pode ter qualquer conjunto de adjacências, ou seja, pode ter ponteiros para todas as outras entidades a ele conectado. Uma entidade pode também ser dividida recursivamente e permite que a entidade pai e seus filhos possam ser acessados.

Tanto no pré-processador quanto no simulador se faz necessário associar às entidades da malha informações (escalar ou vetor) como coeficientes geométricos ou dados físicos como pressão, velocidade e saturação. As três funções abaixo permitem anexar um valor inteiro, um valor de ponto flutuante e um ponteiro to tipo *void**, respectivamente, a qualquer ponteiro *mEntity* por meio de um identificador *pMeshDataId*. Este identificador é útil quando se deseja que mais de um dado seja associado a uma entidade.

```
void EN_attachDataInt(mEntity *, pMeshDataId, int);
void EN_attachDataDbl(mEntity *, pMeshDataId, double);
void EN_attachDataPtr(mEntity *, pMeshDataId, void *);
```

A última função é especialmente interessante, pois abre a possibilidade de associar a entidade um tipo definido pelo usuário que pode conter mais de um dado. Cada uma destas funções possui uma equivalente para extrair os dados antes anexados.

```
int EN_getDataInt(mEntity *, pMeshDataId, int *);
int EN_getDataDbl(mEntity *, pMeshDataId, double *);
int EN_getDataPtr(mEntity *, pMeshDataId, void **);
```

Uma das principais fontes de *overhead*, i.e, a quantidade de tempo requerida para coordenar tarefas paralelas que não se traduz como trabalho útil, em sistemas de memória distribuída é a comunicação entre processadores. A distribuição da malha e o consecutivo balanço de carga exige a troca de muitas mensagens pequenas podendo ocasionar um custo

excessivo de tempo (Seol, 2005). Oculto ao usuário, o FMDB consegue fazer este trabalho internamente de forma eficiente através da biblioteca *Autopack*. Este consegue agrupar várias mensagens de tamanhos pequenos e as envia em um único pacote. O *Autopack* tem como propósito reduzir o número de troca de fragmentos de mensagens entre partições. Suas funções, assim como as do ParMetis, ficam a cargo do FMDB. Nenhuma delas precisa ser usada explicitamente pelo usuário a menos que deseje algo que não seja fornecido pelo FMDB.

5.5.4 Resolução de sistemas de equações (PETSc)

Em simulação numérica, sistemas de equações lineares ou não-lineares comumente surgem e seus tamanhos podem variar entre milhares e milhões de graus de liberdade. A resolução destes sistemas, que geralmente envolvem matrizes esparsas, exige métodos robustos e eficientes para resolvê-los de forma acurada. O PETSc (*Portable, Extensible, Toolkit for Scientific Computation*) contém uma série de métodos iterativos baseados nos subespaços de Krylov para a resolução de sistemas de equações de forma serial ou paralela.

Para acelerar a taxa de convergência destes métodos ou até mesmo para permitir que os mesmos convirjam, uma série de pré-condicionadores está à disposição. A tabela 5.1 mostra algumas opções de métodos iterativos e de pré-condicionadores disponíveis no PETSc.

O PETSc contém um conjunto de rotinas, entre muitas outras, que fornecem várias interfaces de alto nível para a manipulação de matrizes e de vetores. Por interface de alto nível, entende-se uma interface simples onde tarefas trabalhosas ou complexas ficam ocultas.

Tabela 5.1 – Exemplos de *solvers* e pré-condicionadores disponíveis no PETSc.

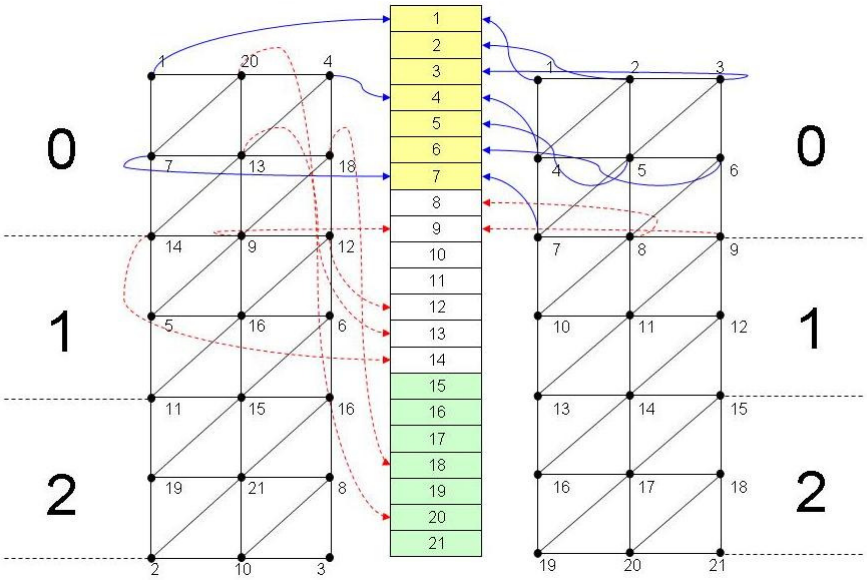
Solvers	Pré-condicionadores
Gradiente Conjugado (CG)	ILU
Gradiente Bi-Conjugado (BiCG)	LU
Gradiente Bi-Conjugado Estabilizado (BiCG)	Jacobi
Generalized Minimum RESidual (GMRES)	ASM (additive Schwarz method)

Uma dessas rotinas de alto nível, chamada de *applicationOrdering*, é utilizada no mapeamento da numeração de uma malha distribuída. A figura 5.4 mostra uma malha 2-D de triângulos com 21 nós particionada entre três processadores e onde cada partição teve sua interface minimizada como descrito no item 5.3.4.

Supondo que todos os nós sejam livres, a matriz A do sistema de equações terá 21 linhas que devem ser distribuídas de forma balanceada entre os processos a fim de que todos tenham a mesma carga de trabalho.

Logo, A será distribuída e cada um dos três processos terá sete linhas. A numeração das linhas segue uma seqüência global e ordenada de 1 a 21 onde ao processo de *rank* 0 são atribuídas as linhas 1 a 7, ao *rank* 1 as linhas de 8 a 14 e ao *rank* 3 as linhas de 15 a 21. O número de colunas de A é sempre o mesmo para todos os processos.

Montar as equações de cada nó em uma linha correspondente a sua numeração global ocasiona um excesso de comunicação entre os processadores devido ao grande número de equações montadas remotamente. Estas operações são representadas pelas linhas tracejadas na Fig. 5.5 (a) e relativas apenas ao processo de *rank* 0. Percebe-se que o número de operações paralelas supera o número de operações locais.



(a) (b)

Figura 5.4 – Malha 2-D distribuída entre três processadores. A malha com a numeração global inicial dos nós (a) gera uma troca maior de informações entre processadores. A malha com os nós renumerados (b) permite que a maior parte das equações referente a cada nó seja montada localmente, aumentando assim, a eficiência na montagem da matriz.

Atribuindo agora uma nova numeração por meio do *applicationOrdering* aos nós da malha global, Fig. 5.4 (b), apenas as equações referentes aos nós nas fronteiras entre partições são montadas remotamente. Dessa forma, a situação se inverte e se passa a ter mais operações locais que paralelas.

O *applicationOrdering* cria um objeto que permite acessar de forma simples e eficiente a nova numeração atribuída aos nós e, assim, montar as equações de forma mais local possível.

Na prática, alguns nós da malha tem valores prescritos (condição de Dirichlet) para que o problema esteja bem posto e logo, não devem ser renumerados. A numeração é atribuída apenas aos nós livres (as incógnitas do problema).

No presente trabalho, o termo elíptico apresentado pela Eq. (3.22) pode ser escrito na seguinte forma matricial:

$$Ax = \left(EFx + Gx\right) \tag{5.2}$$

onde $\left[G\right]$ representa a projeção do fluxo paralelo à direção da aresta e o produto $\left[E\right]\left[F\right]$ representa a projeção normal do mesmo à aresta.

As matrizes $\left[E\right]$, $\left[F\right]$ e $\left[G\right]$ são montadas por arestas e por faces (se 3-D) pelas Eq. (5.3), (5.4) e (5.5) respectivamente:

$$\left[E\right]^{IJ_L(\Omega_r)}_{(2\,x\,2\mathrm{ndim})} = \frac{-\underset{\sim}{K}^{\Omega_r}\lambda_T^{IJ_L}}{2}\begin{pmatrix}\left(\underset{\sim}{I} - \vec{L}_{IJ_L}\cdot\vec{L}_{IJ_L}\right) & \left(\underset{\sim}{I} - \vec{L}_{IJ_L}\cdot\vec{L}_{IJ_L}\right)\\ \left(\vec{L}_{IJ_L}\cdot\vec{L}_{IJ_L} - \underset{\sim}{I}\right) & \left(\vec{L}_{IJ_L}\cdot\vec{L}_{IJ_L} - \underset{\sim}{I}\right)\end{pmatrix}\cdot\vec{C}^{\Omega_r}_{IJ_L}, \tag{5.3}$$

$$\left[F\right]^{IJ_L(\Omega_r)}_{(2\mathrm{ndim}\,x\,2)} = \frac{\vec{C}^{\Omega_r}_{IJ_L}}{2}\begin{pmatrix}\dfrac{1}{V_I} & \dfrac{1}{V_I}\\ -\dfrac{1}{V_{J_L}} & \dfrac{1}{V_{J_L}}\end{pmatrix}. \tag{5.4}$$

$$\left[G\right]^{IJ_L(\Omega_r)}_{(2\mathrm{ndim}\,x\,2\mathrm{ndim})} = -\underset{\sim}{K}^{\Omega_r}\lambda_T^{IJ_L}\frac{\vec{L}_{IJ_L}\cdot\vec{C}^{\Omega_r}_{IJ_L}}{\left|\Delta_{IJ_L}\right|}\begin{pmatrix}1 & -1\\ -1 & 1\end{pmatrix}, \tag{5.5}$$

Nas Eq. (5.3), (5.4) e (5.5), *ndim* refere-se ao número de dimensões do domínio (2-D ou 3-D), $\underset{\sim}{I}$ é a matriz identidade, $\vec{L}_{IJ}$ o vetor paralelo à aresta, $\underset{\sim}{K}$ o tensor de permeabilidade, $\lambda_T^{IJ_L}$ é a média aritmética da mobilidade total definida nos nós *I* e *J* e $\left|\Delta_{IJ_L}\right|$ o comprimento da aresta *IJ*.

Todas as matrizes $[E]$, $[F]$ e $[G]$ são esparsas e são definidas como tais no código a fim de poupar memória. Operações de multiplicação e adição de matrizes com perfis de esparsidade diferentes representam um custo extremamente elevado de uso de CPU tornado, assim, impraticável este tipo de tarefa para problemas de grande porte.

Por esse motivo, um esquema livre de matriz (*matrix-free*) suportado pelo PETSc foi usado de forma a não realizar operações entre matrizes. De fato, todas são montadas e apenas operações matriz-vetor são realizadas. A matriz final A não é montada. O PETSc permite que sistemas de equações lineares possam ser resolvidos por métodos iterativos como o GMRES ou o Gradiente Conjugado usando apenas o produto Ax.

O PETSc permite criar uma matriz sem que suas entradas sejam de fato armazenadas na memória através da chamada *MatCreateShell*. A esta matriz deve-se associar alguma operação fornecida pelo usuário, neste caso uma operação matriz-vetor, de forma a ter o produto desejado. Isto é feito através da chamada *MatShellSetOperation*. Quando a chamada responsável pela resolução do sistema de equações do PETSc é executada (*KSPSolver*), a operação definida pelo usuário é invocada, internamente, para retornar o produto Ax a cada iteração do método escolhido. Esta operação se resume em três passos apresentados pelas Eqs. (5.6), (5.7) e (5.8) respectivamente.

$$y = Gx \tag{5.6}$$

$$z = Fx \tag{5.7}$$

$$y = y + Ez \tag{5.8}$$

Partindo de uma solução inicial x_0, primeiro é calculado o produto Gx da Eq. (5.5) atribuindo o resultado a um vetor *y* como mostra a Eq. (5.6). Em seguida, um segundo vetor *z* calcula o produto Fx como na Eq. (5.7). Por último, o vetor *y* conclui as operações matriz-vetor da Eq. (5.5) representando o produto Ax.

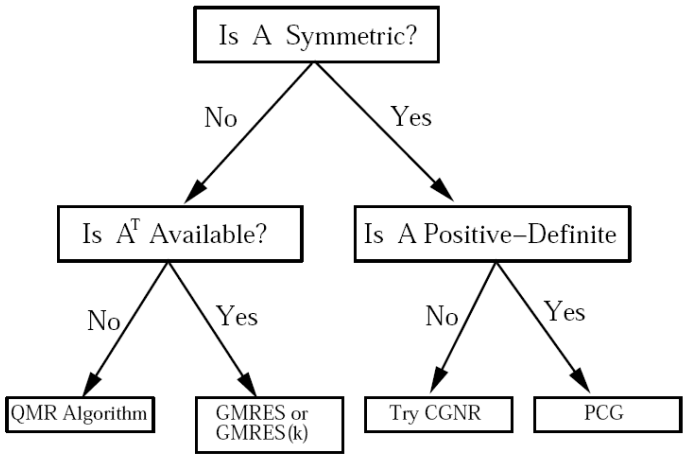
É possível fazer este tipo de operação sem realmente montar matrizes. Entretanto, para resolver problemas envolvendo EDPs não-lineares o uso de uma matriz pré-condicionadora pode trazer um ganho computacional significativo em termos de uso de CPU reduzindo o número de iterações necessárias para a convergência do sistema linear, podendo até mesmo ser imprescindível para que a solução do sistema de equações convirja, i.e, garante estabilidade.

Ao se definir o método de resolução para o sistema $Ax = b$, pode-se associar uma matriz pré-condicionadora à matriz de elementos. Como neste trabalho todas as matrizes são montadas, $[G]$ é usada como tal. A matriz $[G]$ é o resultado de uma aproximação de diferenças finitas que gera uma matriz esparsa, simétrica e positiva-definida. As matrizes $[E]$ e $[F]$ não podem ser usadas porque não são quadradas.

Saber que tipo de método iterativo e de pré-condicionador usar é uma tarefa complexa e que em muitos casos pode haver mais de uma boa opção. O que obviamente pode ser uma boa solução para simulações seriais pode não ser em paralelo. Certas medidas devem ser tomadas na escolha do melhor método a ser usado e baseado nas propriedades da matriz A deve-se examinar se:

- A é bem ou mal-condicionada
- A é simétrica ou não-simétrica
- A é positivo-definida
- A é esparsa

Apesar da matriz A não ser realmente montada, esta é esparsa, mal-condicionada e não-simétrica devido à introdução dos termos de difusão cruzada produzido pelo produto das matrizes $[E]$ e $[F]$. Karniadakis (2003) sugere uma possível árvore de decisões Fig. 5.5 na escolha do método iterativo mais apropriado para resolver problemas de grande porte.



Fonte: Karniadakis, G. E. e Kirby II, R. M., "Parallel Scientific Computing in C++ and MPI: A Seamless Approach to Parallel Algorithms and their Implementation", Cambridge University Press, 2003

Figura 5.5 – Possível árvore de decisões na escolha do método iterativo para problemas de grande porte e não deve ser tratada como uma regra única.

Na Fig 5.5, QMR, CGNR e PCR referem-se aos algoritmos *Quasi-Minimal Residual* (uma versão mais estável do gradiente bi-conjugado)*, Conjugate Gradient Normal Residual, Preconditioned Conjugate Gradient* respectivamente. Para a classe de problemas abordados neste trabalho, o *solver* e o pré-condicionador adotado nas simulações foram GMRES e ASM (*Additive Schwartz Method*) respectivamente.

5.6 Implementação paralela

Nos itens 5.5.1 e 5.5.2 seguintes, são descritos os aplicativos desenvolvidos no presente trabalho. Estes consistem de um pré-processador 2D/3D seqüencial e paralelo e de um simulador de escoamento bifásico óleo-água em meios porosos também seqüencial e paralelo para malhas de triângulos e tetraedros. Estes aplicativos foram desenvolvidos usando todo o ferramental apresentado nos itens anteriores e escritos em C++.

5.6.1 Pré-processador

A formulação de volumes finitos baseada em arestas utilizada na discretização das equações de escoamento do presente trabalho e que foi detalhada no capítulo 3 exige que uma fase de pré-processamento seja realizada sobre a malha de elementos. Este pré-processamento consiste na montagem da estrutura de dados de arestas e no cálculo de coe-

ficientes que, para esta formulação, representam os vetores áreas das superfícies de controle associados ($\vec{C}_{IJ}$ e $\vec{D}_{IJ}$) e o volume dos volumes de controle (V_I). Os coeficientes de área são associados às arestas da malha e os volumes aos nós da mesma. No caso 3-D, o coeficiente $\vec{D}_{IJK}$, pode ser associado às faces do contorno. Cada coeficiente representa a contribuição de todos os elementos que usam a mesma aresta ou nó.

Antes do cálculo destes coeficientes, é preciso criar uma estrutura de arestas a partir da malha de elementos. A biblioteca FMDB permite criar tal estrutura a partir de uma malha de triângulos ou de tetraedros. Por possuir uma interface de leitura para malhas do gerador Gmsh, a obtenção de uma estrutura de arestas se torna uma tarefa muito simples tanto para malhas seqüenciais quanto distribuídas. O Gmsh é um gerador de malhas seqüenciais de código aberto com uma interface gráfica amigável e que permite produzir malhas 2-D ou 3-D a partir de modelos geométricos onde podem ser definidas as condições de contorno do problema.

Após a geração da malha e criação da estrutura de dados de arestas, as etapas executadas pelo pré-processador consistem em varrer os elementos (triângulos ou tetraedro) extraindo as informações necessárias e associando-as às arestas e aos nós do elemento de forma acumulativa, ou seja, ao final da varredura, uma aresta ou nó terá a contribuição de todos os elementos que os compartilham. No caso de malhas 3-D, devido ao tipo de integração adotada, os elementos de face dos contornos são usados e logo, deve-se ter uma estrutura de dados para armazená-los.

Quando se usa malhas distribuídas, os coeficientes associados às arestas e aos nós presentes nas fronteiras das partições ficam incompletos, pois parte dos elementos que compartilham estas entidades estão presentes em outros processos. Neste caso, o procedimento adotado aqui foi fazer com que uma aresta ou um nó que possui uma ou mais cópias remotas tenha os mesmos coeficientes caso a mesma malha fosse pré-processada de forma seqüencial. Isto evita certas complicações e inconsistências, por exemplo, no momento da montagem de matrizes distribuídas para o cálculo de sistemas de equações. Dessa forma, para problemas em paralelo, mais uma etapa se faz necessária ao pré-processamento. Apenas os coeficientes associados às arestas e aos nós presentes nas fronteiras das partições são somados. Os passos do pré-processamento podem ser entendidos através do algoritmo 5.1.

Na etapa paralela, onde as entidades com cópias remotas são varridas, p_i refere-se a todos os processos que contêm a entidade em questão. Ao final da fase de pré-

processamento, uma lista de nós com suas respectivas numerações globais, coordenadas espaciais e volume dos volumes de controle associados a este nó e mais uma lista de arestas e faces de contorno com seus respectivos coeficientes são necessários para iniciar a simulação, descartando, assim, a malha de elementos finitos original.

// Etapa seqüencial
Laço sobre os elementos{
    Laço sobre as arestas do elemento{
        $\vec{C}_{IJ} = \vec{C}_{IJ} +$ *contribuição do elemento à aresta IJ*
    }
    Laço sobre os nós do elemento{
        $V_I = V_I +$ *contribuição do elemento ao nó I*
    }
}
Laço sobre os elementos da face do contorno (se 3-D){
    $\vec{D}_{IJK} =$ *contribuição do elemento à face IJK*
}
// Etapa paralela
Laço sobre as arestas com cópias remotas{
    $\vec{C}_{IJ} = \sum_i \vec{C}_{IJ}^{p_i}$
}
Laço sobre os nós com cópias remotas{
    $V_I = \sum_i V_I^{p_i}$
}

Algoritmo 5.1

5.6.2 Simulador

O principal foco deste trabalho consiste na construção de um simulador para resolver problemas de escoamento óleo-água em reservatório de petróleo com malhas 2-D de triângulos ou 3-D de tetraedros usando computadores paralelos de memória distribuída. O uso da programação orientada a objetos em C++, assim como da biblioteca STL (*Standard Template Library*) foram usados intensivamente no desenvolvimento deste simulador de forma a definir bem alguns conceitos por meio de classes. Isto trás a possibilidade de reusá-las em outros simuladores com o mínimo de alterações.

De forma a proteger as classes existentes da incorporação de outras definidas por terceiros ao simulador que possam ter nomes coincidentes, foi escolhido um *namespace*

chamado PRS (*Petroleum Reservoir Simulator*) para melhor caracterizar a finalidade do programa aqui desenvolvido.

Este item apresenta as principais classes com trechos de códigos e algoritmos que tentam passar uma idéia clara de como o simulador funciona e como se dá o uso dessas classes no programa.

Como dito anteriormente, a formulação IMPES empregada envolve o cálculo de uma equação elíptica e outra hiperbólica, onde ambas são resolvidas pelo método EBFV1. Visando possibilitar a utilização da estrutura do simulador junto com suas classes em outras formulações que podem vir a ser incorporadas ao programa, o conceito de polimorfismo do C++ foi aplicado tendo-se um objeto referente à resolução da equação de difusão e outro referente à resolução da equação hiperbólica. Na principal classe do simulador, *SIMULATION_core*, são definidos dois ponteiros, um para cada tipo de equação:

```
Elliptic_equation* pElliptic_eq;
Hyperbolic_equation* pHyperbolic_eq;
```

A inicialização destes ponteiros é baseada na escolha feita pelo usuário que resulta em:

```
pElliptic_eq = new EBFV1_elliptic;
pHyperbolic_eq = new EBFV1_hyperbolic;
```

Neste caso, os ponteiros *pElliptic_eq* e *pHyperbolic_eq* são inicializados de forma a resolver suas respectivas equações por meio da formulação EBFV1. Como outras formulações podem ser incorporadas, estes ponteiros podem estar associados à formulações diferentes como a EBFV2 discutida no capítulo três.

O trecho de código abaixo mostra o método *tsSolver* (*time step Solver*) responsável pelo processo de avanço no tempo da simulação. O critério de parada fica a cargo do objeto *pSimPar* da classe *SimulationParameters* que é reponsável por todos os dados físicos, numéricos e de simulação fornecidos pelo usuário.

```
int SIMULATION_core::tsSolver(){
      double timeStep, simTime = .0;
      while ( !pSimPar->finishSimulation(simTime) ){

            // solve elliptic equation
```

```
            pElliptic_eq->solver();

            // solve hiperbolic equation
            pHyperbolic_eq->solver(timeStep);

            simTime += timeStep;
        }
}
```

O uso de polimorfismo permite que não haja alterações ao membro *SIMULATION_core::tsSolver()* poupando o desenvolvedor de tarefas adicionais. Para isso, o uso de métodos virtuais é necessário de forma que o ponteiro *pElliptic_eq*, que é definido como um *Elliptic_equation*, possa acessar os membros da classe *EBFV1_elliptic* como é o caso da função *solver*. Esta é declarada como uma função puramente virtual

```
virtual solver() const=0;
```

pois permite que em cada classe derivada ela possa ser implementada de acordo com a formulação desejada. O mesmo princípio é utilizado para as classes da equação hiperbólica.

Classes auxiliares complementam o trabalho de cálculo das equações apresentadas de forma a fornecer as informações de uma forma bem definida e clara. Um ponto interessante da programação orientada a objetos está em permitir que outros desenvolvedores possam criar novas funcionalidades sobre algo que já existe sem necessidade de conhecer como esta base funciona. Um exemplo neste trabalho está no uso da função do FMDB

```
void EN_attachDataPtr(mEntity * ,pMeshDataId, void *);
```

que permite associar um ponteiro (*void**) a uma entidade (*mEntity**) da malha por meio de um identificador (*pMeshDataId*).

A classe *ManagerData* foi criada de forma a servir de base para outras classes desenvolvidas com o propósito de inserir e extrair dados às entidades da malha sem se ater aos detalhes do funcionamento das funções *EN_attachDataPtr* e *EN_getDataPtr*.
Além disso, esta classe permite que se construam códigos mais enxutos, pois evita a repetição de linhas de código toda vez que se deseje acessar dados de uma entidade.

O código abaixo mostra os principais membros (*getDataPointer* e *setDataPointer*) da classe *ManagerData* onde o uso de funções *templates* permite a parametrização dos dados a serem associados a uma entidade da malha tornando-as, assim, genéricas para qualquer tipo de ponteiro.

```
template <class T>
T* getDataPointer(pEntity ent){
      void *p; p=0;
      EN_getDataPtr(ent,dataAttached_id,&p);
      return (p) ? (T*)p : new T;
}

template <class T>
void setDataPointer(pEntity ent, T* ptr){
      void *p; p=0;
      p = (T*)ptr;
      EN_attachDataPtr(ent,dataAttached_id,p);
}
```

Duas classes que herdam *ManagerData* são *GeometricCoefficientsData* e *PhysicalPropertiesData* como mostra o diagrama de hierarquia na Fig. (5.6).

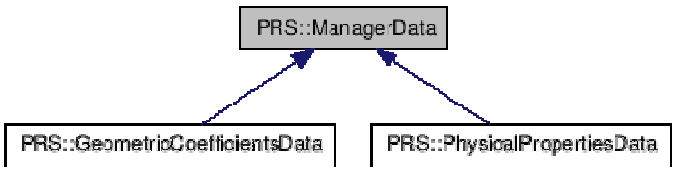


Figura 5.6 – Diagrama de classes *ManagerData.*

Ao inicializar um objeto de uma destas classes, o construtor da classe base (*ManagerData*) é chamado e este se encarrega de gerar um identificador (*pMeshDataId*). Estas classes foram criadas para gerenciar os dados geométricos da formulação de volumes finitos e dados físicos do problema como pressão, gradiente de pressão, velocidade, fluxo e saturação. Para associar estas informações a uma entidade, estas são agrupadas em uma estrutura (*struct).* No caso da classe *PhysicalPropertiesData* uma estrutura agrupa todas as propriedades físicas nodais como pressão, gradiente de pressão, saturação de água e o termo de fluxo como mostra o código abaixo.

```
struct NodePhysicalProperties{
      double p;                     // pressure
      std::vector<double> p_Grad;   // pressure gradient
      double Sw;                    // water saturation
      double flux;                  // flux term
};
```

A classe *PhysicalPropertiesData* fornece apenas as funções de inserção (*set*) e obtenção (*get*) de dados. Para cada nó da malha, um novo ponteiro (*NodePhysicalProperties**) é alocado para esta entidade. Como exemplo, a variável de saturação tem seu valor alterado em um nó por meio do membro *setSaturation (pEntity, double)* onde um ponteiro para a estrutura *NodePhysicalProperties* é definido e inicializado com o valor de retorno da função *getDataPointer* que é um membro da classe base *ManagerData*. O trecho de código a seguir mostra como funciona a função para mudar a saturação de um nó.

```
void PhysicalPropertiesData::setSaturation(pEntity node, double Sw){

        // Define um ponteiro que agrupa dados.
        NodePhysicalProperties* pNode;

        // getDataPointer retorna um ponteiro do tipo NodePhysicalPro-
        // perties associado a node.
        pNode = getDataPointer<NodePhysicalProperties>(node);

        // acessa a variável Sw referente à saturação atribuindo um no-
        // novo valor.
        pNode->Sw = Sw;
}
```

A variável velocidade não está associada aos nós, mas sim às arestas. Dessa forma, uma estrutura *EdgePhysicalProperty* é definida e um novo ponteiro é associado a cada aresta da malha.

As classes *GeometricCoefficientsData* e *PhysicalPropertiesData* apresentadas não realizam nenhuma tarefa relacionada a troca de mensagens entre processos. Essas tarefas, por sua vez, são atribuídas à classe *MeshData* que serve de base ao envio e recebimento de dados. Informações sobre nós livres e nós de valores prescritos local e global e o mapeamento de nós usando a rotina *applicationOrdering* do PETSc discutido no item 5.4.5 são

implementadas de forma a auxiliar principalmente a montagem do sistema de equações para a resolução do campo de pressões.

Para que cada processo possa exportar em arquivo os dados para posterior visualização como o campo de saturações mostrado na Fig. 5.7, as propriedades físicas sobre os nós nas fronteiras das partições devem ser idênticas às das suas cópias remotas.

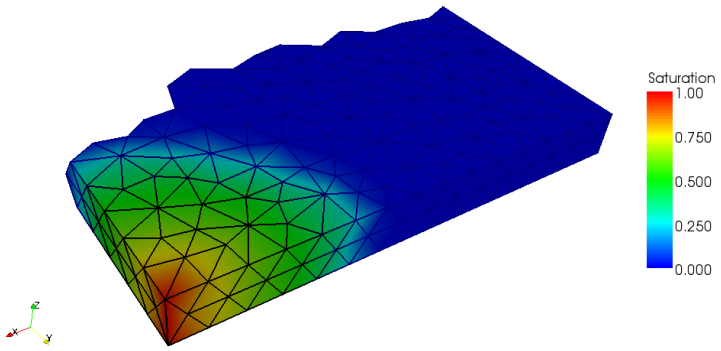


Figura 5.7 – Partição de uma malha mostrando o campo de saturação de água em um dado instante da simulação.

Um membro da classe *MeshData* é responsável por coletar todos os valores de uma certa propriedade relacionados aos nós nas fronteiras das partições somando-os e em seguida distribuindo os resultados.

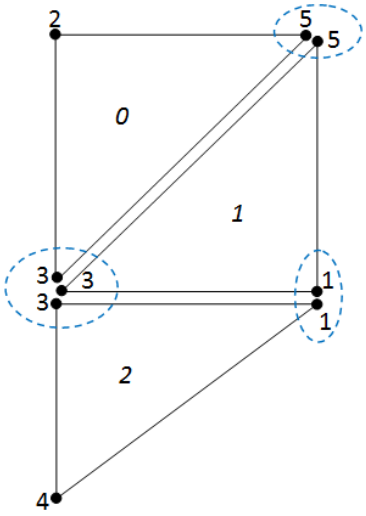


Figura 5.8 – Malha de triângulos distribuídos entre três processos destacando pelas linhas tracejadas os nós com cópias remotas.

A figura 5.8 mostra uma malha formada por três triângulos distribuída entre três processos. Nesta malha, os nós 1, 3 e 5 possuem cópias remotas cuja propriedade física

nodal é obtida pela contribuição das arestas locais conectadas a estes. Um exemplo seria o fluxo.

As etapas para o cálculo do fluxo compreendem inicialmente uma fase seqüencial seguida de uma fase paralela. A tarefa de coleta e distribuição é dada pela função *updateScalarData* da classe *MeshData*.

Laço sobre as arestas{

Cálculo local dos termos de fluxo nodal

}

// Coleta e distribuição dos fluxos sobre nós com cópias remotas

```
pMData->updateScalarData(theMesh,&PhysicalPropertiesData::nonvisc);
```

Na chamada de *updateScalarData*, a malha e um ponteiro de função são passados como argumentos. As estruturas de vetores e matrizes distribuídas do PETSc são usadas para realizar as tarefas de coleta e distribuição dos dados onde um mapeamento semelhante ao discutido no item 5.4.5 é usado para nós com cópias remotas. Para este exemplo, um vetor de três posições recebe e soma os valores dos três nós como mostra a Fig. 5.9.

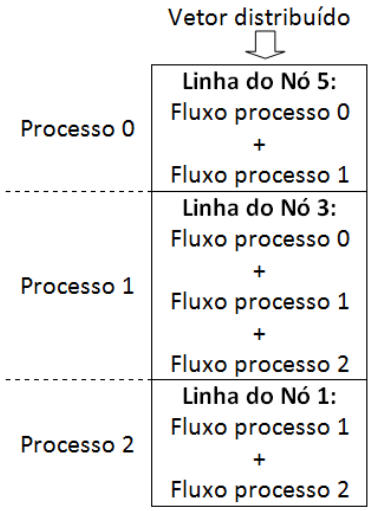


Figura 5.9 – Vetor distribuído entre três processos acumulando a contribuição local de uma propriedade física associada aos nós remotos de uma malha de triângulos.

Outra propriedade que também passa pelo mesmo procedimento é o gradiente de pressão, que como o termo de fluxo, também está associado aos nós e é obtido pela contribuição das arestas. Neste caso, a propriedade se trata de um vetor com coordenadas x, y e

z. Seguindo o mesmo procedimento da Fig. 5.9, cada nó, com uma ou mais cópias remotas, envia seus dados a uma matriz de três colunas, e não um vetor, distribuída entre os processos.

A distribuição dos resultados coletados e somados segue um caminho um pouco diferente. Vetores e matrizes do PETSc permitem que um valor seja inserido remotamente, ou seja, permite que um processo insira um valor em uma posição do vetor ou em uma linha da matriz que é local a outro processo. Essa tarefa é feita por meio da função:

```
MatSetValue(matrix,&row,&col,value,ADD_VALUES);
```

Porém, para um processo obter dados de linhas que são remotas a ele, usa-se uma segunda matriz, que também é distribuída, cujas linhas locais contêm os dados requeridos por este processo. Esta coleta de dados é feita pela função do PETSc:

```
MatGetSubMatrixRaw(A,
                   pMData->get_A_nrows(),
                   pMData->get_A_rows_idx(),
                   pMData->get_ncols(),
                   pMData->get_A_cols_idx(),
                   PETSC_DECIDE,MAT_INITIAL_MATRIX,
                   &B);
```

A matriz *A* é de onde se deseja extrair as linhas e *B* é a matriz que recebe (coleta) os dados requeridos. O ponteiro *pMData* é do tipo *MeshData* e é usado para informar as linhas e colunas de *A* que devem ser importadas.

A construção de classes e a relação entre elas quando preciso demanda um bom tempo de trabalho, pois devem ser projetadas de forma a representar bem os conceitos que pretendem moldar, facilitando, assim, a construção do programa. As classes que compõem o simulador e que foram aqui apresentadas tentam expressar da forma mais clara possível seus propósitos permitindo que estas possam ser estendidas a outros programas.

5.7 Análises de desempenho

Geralmente, a análise de desempenho de um programa paralelo refere-se a como este se compara a um programa seqüencial. Se ambos levam o mesmo tempo para resolver

um mesmo problema, então o primeiro é completamente inútil, pois se espera que o tempo final de execução de um programa paralelo seja reduzido significativamente à medida que se aumenta o número de processadores.

Duas formas de medidas comumente usadas para avaliar o desempenho de programas paralelos são: o *speed-up* e a eficiência. O *speed-up* compara o tempo de execução do programa paralelo com o do programa seqüencial. A primeira vista, pode-se imaginar que se um programa seqüencial leva um tempo $t_{seq} = t$, logo o tempo de execução do programa paralelo resolvendo o mesmo problema usando, por exemplo, dois processadores seria $t_{par} = t_{seq} / 2$, com três processadores $t_{par} = t_{seq} / 3$ e assim por diante. Entretanto, o que acontece é diferente. Programas paralelos trocam informações entre si durante a execução e esta tarefa consume um tempo que não existe em programas seqüenciais e esse custo tende a afastar o tempo de execução paralelo real do ideal. Há programas paralelos que conseguem uma aproximação muito boa do ideal, mas como já foi dito esses resultados são conseqüência de anos de aprimoramento do código.

O *speed-up*, que representa a razão do tempo de execução do programa seqüencial pelo tempo de execução em paralelo para um mesmo problema.

$$S(n, p) = \frac{T_{seq}(n)}{T_{par}(n, p)} \tag{5.9}$$

Onde n representa o tamanho do problema e p o número de processadores.

Com os dados de *speed-up* traça-se um gráfico que mostra como o programa paralelo se comporta com o aumento do número de processadores. Neste trabalho, um problema de ¼ de "*five-spot*" foi analisado com duas malhas representando a mesma geometria e com as mesmas condições de contorno, porém com número de nós diferentes sendo uma com 150.725 nós e 846.855 tetraedros e a outra com 589.487 nós e 3.299.170 tetraedros. Esta última foi a maior que pode ser executada por um único processador.

Em ambas as análises foram usados: 1, 2, 4, 8,16 e 32 processadores. Como a cada passo de tempo do problema as operações de ponto flutuante e o número de envio e recebimento de mensagens se repetem, foi obtida uma média do tempo de execução para vinte passos de tempo. As tabelas 5.2 e 5.3 mostram os resultados obtidos para as duas análises, onde NP refere-se ao número de processadores e o tempo acumulado refere-se ao tempo de execução consumido após 20 passos de tempo.

Tabela 5.2 – Malha com 150.725 nós.

	Tempo [s]		*Speed-up*		
NP	Acumulado	Médio	Obtido	Ideal	Eficiência
1	6133	360,76	1	1	1
2	3668	215,76	1,67	2	0,83
4	2602	153,05	2,35	4	0,58
8	1482	87,17	4,13	8	0,51
16	1098	64,58	5,58	16	0,34
32	412	24,23	14,88	32	0,46

Tabela 5.3 – Malha com 589.487 nós.

	Tempo [s]		*Speed-up*		
NP	Acumulado	Médio	Obtido	Ideal	Eficiência
1	89988	4499,4	1	1	1
2	66106	3305,3	1,36	2	0,68
4	28979	1448,95	3,10	4	0,77
8	19855	992,75	4,53	8	0,56
16	12416	620,8	7,24	16	0,45
32	4093	204,65	21,98	32	0,68

O gráfico de *speed-up* da figura 5.10 mostra que apesar de haver um ganho de tempo com o aumento do número de processadores este vai sendo reduzido e tendendo a se estabilizar como se pode observar na figura 5.11.

As figuras 5.10 e 5.11 mostram as curvas de *speed-up* e a queda de tempo de execução para o problema com a malha de 150.725 nós.

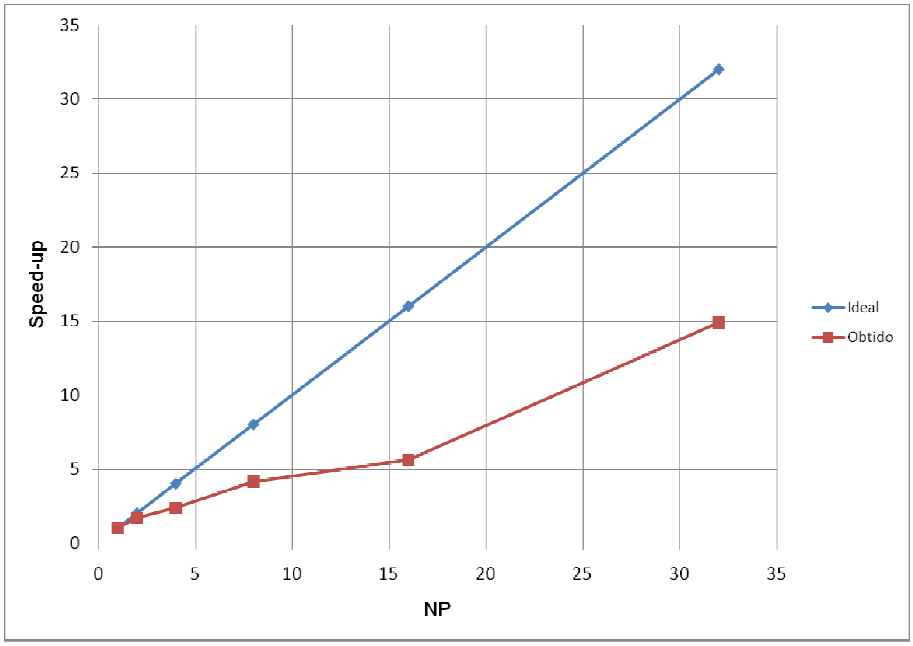


Figura 5.10 – Comparação entre as curvas de *speed-up* ideal e real para a malha de 150.725 nós.

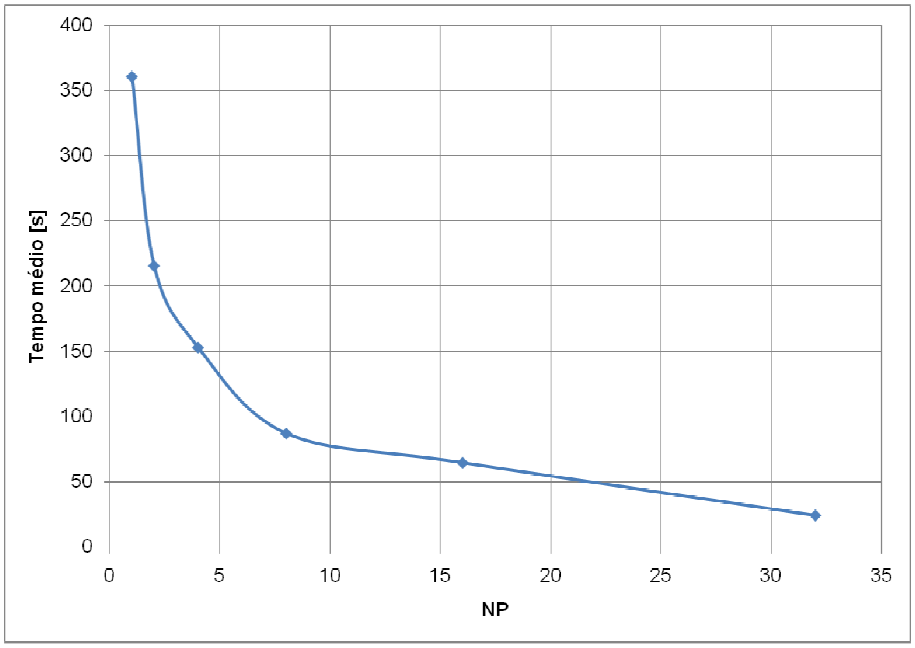


Figura 5.11 – Redução do tempo médio de execução por passo de tempo para a malha de 150.725 nós.

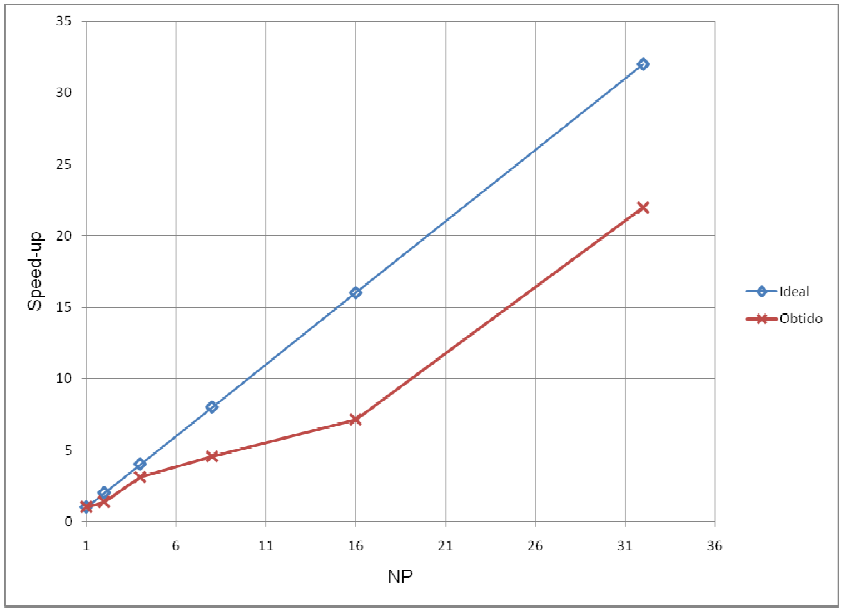


Figura 5.12 – Comparação entre as curvas de *speed-up* ideal e real para a malha de 589.487 nós.

Em outras palavras, dedicar mais processadores à execução do programa não implicará em ganho de tempo. Pelo contrário, a partir de um certo número de processadores a tendência é que a execução em paralelo passe a consumir mais tempo devido, principalmente, ao custo de comunicação entre processadores.

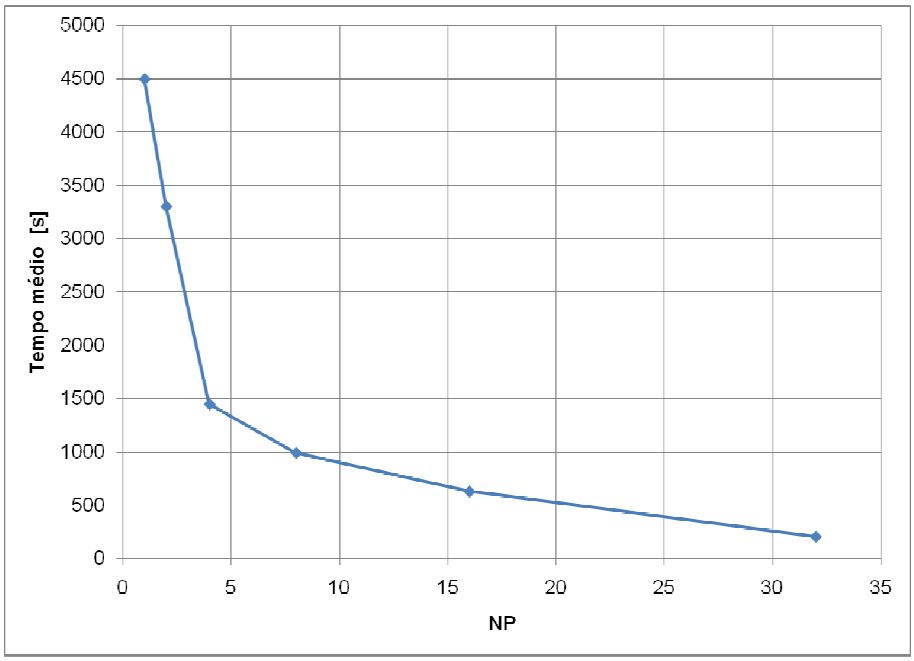


Figura 5.13 – Redução do tempo médio de execução por passo de tempo para a malha de 589.487 nós.

Nas figuras 5.12 e 5.13 são apresentados os gráficos de *speed-up* e de redução de tempo para o problema com a malha de 589.487 nós.

O propósito de usar uma malha maior é mostrar como o tamanho do problema pode influenciar a eficiência do programa paralelo que é um outro parâmetro de medida usado na avaliação de desempenho. A eficiência é definida como a razão do *speed-up* pelo número de processadores usados.

$$E(n,p) = \frac{S(n,p)}{p} \tag{5.10}$$

Para um problema de tamanho fixo, à medida que se aumenta o número de processadores o número de comunicações remotas passa a representar uma parte maior do tempo total de processamento. Quando se trata de problemas maiores, que ocupam a maior parte da memória disponível, esse tempo de comunicação tende a representar uma percentagem menor desse tempo total e, conseqüentemente, a eficiência do programa paralelo pode ser melhor avaliada. A figura 5.14 mostra as curvas de eficiências para as duas malhas usadas.

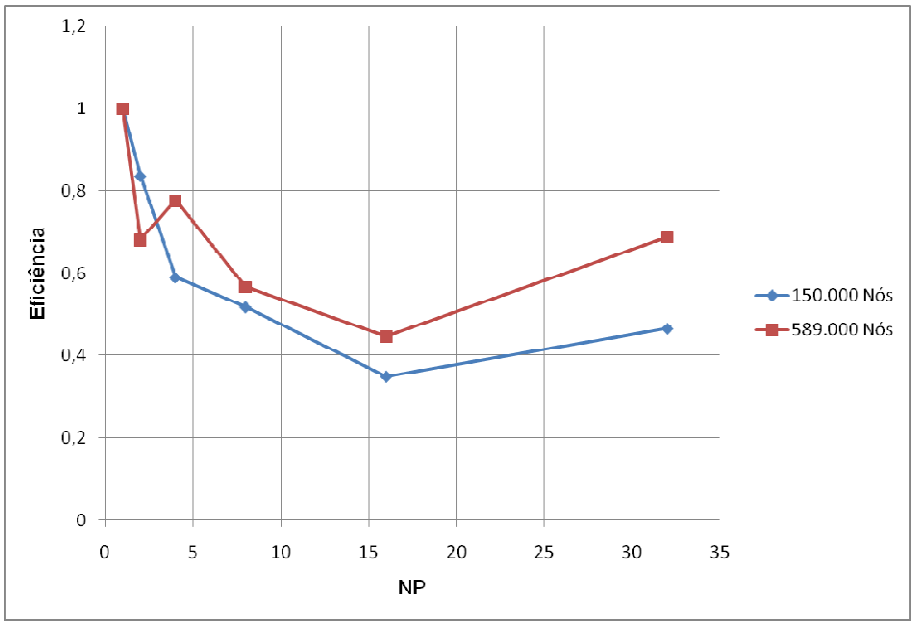


Figura 5.14 – Comparação das curvas de eficiência para duas malhas.

Como era de se esperar, o programa paralelo apresentou uma eficiência maior com a malha de 589.487 nós. A figura 5.15 abaixo também mostra que o *speed-up* para a malha maior também ficou acima da malha de 150.725 nós.

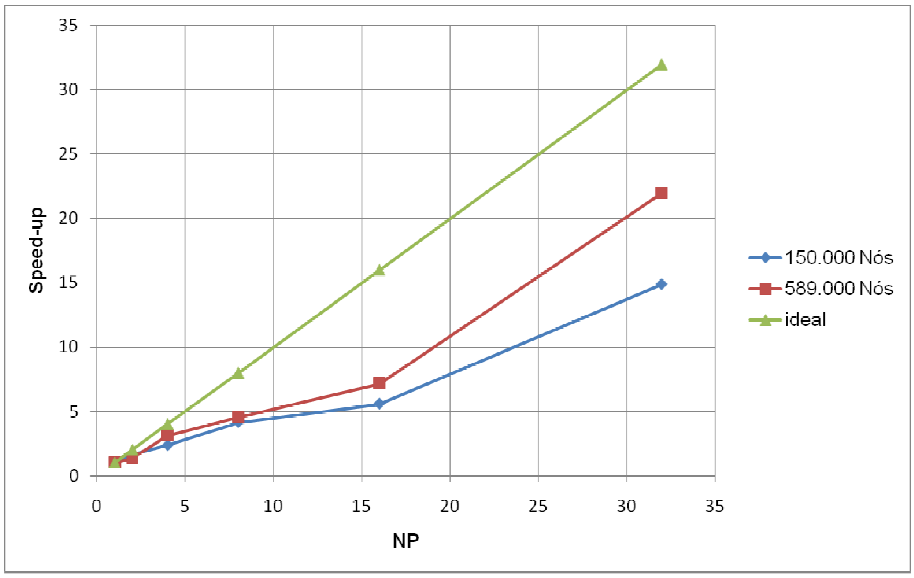


Figura 5.15 – Comparação entre curvas de *speed-up* para o um mesmo problema tratado com malhas de tamanhos diferentes.

As análises de desempenho aqui apresentadas mostram que apesar do programa paralelo ter atingido seu objetivo de reduzir o tempo de simulação com o aumento do número de processadores, vários aprimoramentos precisam ser realizados no código visando um uso mais eficiente da máquina. A parte da simulação que representou o maior custo no uso de CPU (99%) foi o cálculo do campo de pressões efetuado através da biblioteca PETSc.

A tabela 5.4 mostra o tempo, em segundos, utilizado para a montagem das matrizes $[E]$, $[F]$ e $[G]$ e para o cálculo do sistema de equações, assim como o número de iterações necessárias para a sua convergência e o número de operações de ponto flutuante por segundo. Foram mantidas as configurações padrões do PETSc para o *solver* iterativo:

Tolerância absoluta de convergência:	1e-50
Tolerância relativa de convergência:	1e-8
Tolerância de divergência:	10.000
Número máximo de iterações:	10.000
Número de restarts do método GMRES:	30

Pela tabela 5.4, vê-se que o custo computacional da equação hiperbólica, resolvida explicitamente, é extremamente inferior ao da equação elíptica.

Tabela 5.4 – Tempo, em segundos, consumido por vários estágios a cada passo de tempo da simulação para a malha de 589.487 nós

NP	Eq. Elíptica					Eq. Hiperbólica
	t1	t2	t3	Número de iterações	*Mflops	
1	63	4436	4499	4082	154	24
2	449	2856	3305	4937	78	13
4	260	1188	1448	4701	124	7
8	177	815	992	4259	117	3,37
16	86	534	620	7533	136	1,58
32	55	149	204	4213	98	0,81

t1- montagem das matrizes; t2 – resolução do sistema de equações; t3 –total;

*Mflops – Milhões de operações de ponto flutuante por segundo.

6. CONCLUSÕES E TRABALHOS FUTUROS

6.1 Conclusões

Neste trabalho, foram apresentados alguns resultados de acurácia de duas formulações de volumes finitos baseados em arestas (EBFV1 e EBFV2) que podem ser aplicados para resolver equações elípticas com coeficientes altamente descontínuos usando estruturas de dados por arestas. Estas equações elípticas são bastante comuns em inúmeras áreas do conhecimento, particularmente em problemas de condução de calor e escoamento de fluidos em meios porosos, tais como o fluxo bifásico óleo-água em reservatórios de petróleo e transporte de contaminantes em aqüíferos. Tensores cheios (coeficientes de difusão não-diagonal) e malhas não-ortogonais são lidadas naturalmente por ambas as formulações EBFV e os termos de difusão cruzada são corretamente calculados.

De forma a mostrar o potencial dos dois procedimentos de volumes finitos apresentados, foram resolvidos alguns problemas benchmark envolvendo coeficientes de difusão não-diagonais e descontínuos e termos fonte distribuídos descontínuos. Ambos os métodos mostraram acurácia de segunda ordem para a variável escalar, por exemplo pressão, para soluções com anisotropia suave e moderada. Ambos os métodos demandam mais melhoramentos caso se deseje garantir monotonicidade na presença de soluções com alta anisotropia. Os métodos não foram comparados em termos de eficiência computacional. Os resultados são bem promissores e se comparam favoravelmente com outros resultados encontrados na literatura.

A formulação EBFV2 foi desenvolvida com o intuito de se ter uma formulação de volumes finitos capaz de lidar com a heterogeneidade do meio poroso associada aos elementos da malha e não a domínios como originalmente foi desenvolvida a formulação EBFV1.

O uso de um modelo de passo de tempo adaptativo para a formulação IMPES mostrou ser possível obter um ganho de tempo de uso de CPU significativo para problemas de fluxo bifasico óleo-água tanto em meios homogêneos quanto heterogêneos e garantindo resultados muito próximos dos obtidos pelo método IMPES tradicional. A determinação do melhor parâmetro de tolerância para a variação do campo de velocidade DVTOL foi feita após várias análises de acurácia e eficiência computacional para diferentes malhas e diferentes meios porosos. Em um dos problemas, por exemplo, onde uma malha de aproximadamente 10.000 nós num meio heterogêneo foi avaliada, para o $DVTOL = 5 \cdot 10^{-3}$, foi pos-

sível efetuar uma simulação ate PVI=1.0 resolvendo o campo de pressão e velocidade apenas 151 vezes contra 5882 vezes do campo de saturação.

Esse resultado mostra que o problema de estabilidade que restringe severamente o tamanho do passo de tempo do termo hiperbólico, maior entrave ao uso do método IMPES para problemas com malhas muito refinadas, pode ser superado e se conseguir resultados e próximos do IMPES clássico.

O uso de computadores paralelos aparecem como uma importante ferramenta na simulação numérica de alto desempenho em reservatórios de petróleo para problemas de grande porte. O simulador desenvolvido no presente trabalho, escrito em C++ e fazendo uso dos paradigmas da programação orientada a objetos, permite a avaliação de problemas de fluxo óleo-água em meios homogêneos e heterogêneos tanto em computadores seqüenciais (um processador) ou computadores paralelos (múltiplos processadores) de memória distribuída.

Diversos pacotes de código aberto voltados à programação paralela foram estudados e incorporados ao simulador, e estes desempenham diversas tarefas tais como: o gerenciamento de malhas distribuídas (FMDB), o balanço de carga entre processadores (ParMetis) e a resolução de sistemas de equações lineares e manipulação de matrizes e vetores distribuídos (PETSc).

Apesar da ausência de um gerador de malhas paralelo que permitiria produzir malhas de grande porte a ponto de ocupar a maior parte da memória disponível dos processadores envolvidos na simulação, pode-se produzir malhas relativamente grandes e extrair alguns resultados qualitativos e de desempenho. A complexidade do uso dos pacotes empregados exigiu um bom tempo de dedicação no uso de suas rotinas da forma mais eficiente possível.

6.2 Trabalhos Futuros

- Aplicar a formulação EBFV2 em problemas de simulação de reservatórios usando o método IMPES e IMPES modificado.

- Empregar técnicas de aproximação de alta ordem para o termo de saturação para ambas as formulações.

- Acrescentar outras físicas à simulação de reservatórios como pressão capilar e gravidade.

- Estudo de técnicas que garantam a monotonicidade de resultados para problemas com heterogeneidade e anisotropia elevada.

- Estudo e validação de outros métodos de integração no tempo de grande interesse como o seqüencial implícito e o totalmente implícito.

- Estudo e implementação da formulação *black-oil*.

- Estudar novas técnicas para o melhoramento de operações que exigem troca de mensagens entre processadores, em particular a montagem e a resolução de sistemas de equações lineares em paralelo.

- Estudo de técnicas de alto desempenho como adaptação de malhas e *multgrid* aplicados tanto em simulações seqüenciais como em simulações paralelas.

- Na ausência de geradores de malha paralelo, estudar a possibilidade de aplicar procedimentos de refinamento nas partições da malha distribuída de forma a gerar problemas de grande porte e avaliar a capacidade de escalabilidade do simulador.

APÊNDICE A

Equação de pressão para escoamento monofásico em meios porosos totalmente heterogêneo (elemento por elemento) e anisotrópico usando MVFA.

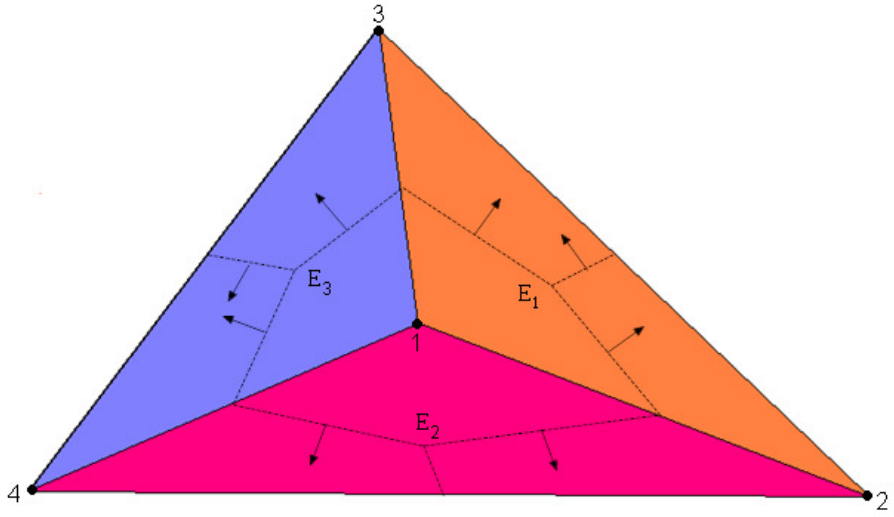
Termo de Fluxo (equação de pressão):

$$\nabla.\left(-K(x)\vec{\nabla}u(x)\right)=0 \tag{A.1}$$

onde:

$$x=(x,y)\in\Omega\subset\Re^2 \text{ e } K(x)=\begin{bmatrix}K_{xx} & K_{xy}\\ K_{yx} & K_{yy}\end{bmatrix}$$

Considerando a malha 2D de elementos triangulares representada pela figura A.1 e onde cada cor representa um material com permeabilidade e porosidade diferentes, são construídos volumes de controle ao redor de cada nó da malha ligando-se os pontos médios das arestas e o baricentro dos elementos. As setas representam os vetores ortogonais às faces do volume de controle conseqüência da aplicação do teorema da divergência ao termo de fluxo mostrada na equação 2.



Aplicado o teorema da divergência na equação 1:

$$\int_{\Omega} \nabla.\left(-K\vec{\nabla}u\right)\partial\Omega = \int\left(-K\vec{\nabla}u\right).\vec{n}\partial\Gamma \tag{A.2}$$

Considerando *K* constante em cada elemento (Cell-Based Scheme) o lado direito da Eq. (A.2) pode ser aproximado para um nó I como:

$$\int_{\Gamma}\left(-K\vec{\nabla}u\right).\vec{n}\partial\Gamma = -\sum_{m}^{NE}\left(K^{e_m}\vec{\nabla}u^{e_m}\right).\vec{C}_I^{e_m} \tag{A.3}$$

onde:

m e NE são o elemento e o número de elementos associado ao nó I, e

$$\vec{C}_I^{e_m} = \vec{C}_{IJ}^{e_m} + \vec{C}_{IK}^{e_m} \tag{A.4}$$

Expandindo o somatório da Eq. (A.3) para o nó 1, fica:

$$\begin{aligned} -\sum_{m}^{NE}\left(K^{e_m}\vec{\nabla}u^{e_m}\right).\vec{C}_1^{e_m} = \Big[&\left(K^{e_1}\vec{\nabla}u^{e_1}\right)\left(\vec{C}_{12}^{e_1} + \vec{C}_{13}^{e_1}\right) + \left(K^{e_2}\vec{\nabla}u^{e_2}\right)\left(\vec{C}_{13}^{e_2} + \vec{C}_{14}^{e_2}\right) + \\ &\left(K^{e_3}\vec{\nabla}u^{e_3}\right)\left(\vec{C}_{12}^{e_3} + \vec{C}_{14}^{e_3}\right)\Big] \end{aligned} \tag{A.5}$$

A Eq. (A.5) precisa ser escrita em termos dos valores nodais de *u*. Dessa forma, uma nova aproximação deve ser feita para os gradientes que são admitidos constantes nos elementos. Aplicando o teorema da divergência ao gradiente de um elemento e:

$$\int_{\Omega_e} \vec{\nabla}u\partial\Omega_e = \int_{\Gamma_e} u.\vec{n}\partial\Gamma_e \tag{A.6}$$

O lado direito da Eq. (A.6) pode ser aproximado como:

$$\int_{\Gamma_e} u.\vec{n}\partial\Gamma_e = -\sum_{\Gamma_{IJ}} u.\vec{D}_{IJ} \tag{A.7}$$

onde:

u_{IJ} é a media aritmética dos valores nodais *I* e *J* e $\vec{D}_{IJ}$ é vetor normal à aresta e externo ao elemento.

Admitindo a seguinte aproximação

$$\int_{\Omega_e} \nabla u \partial\Omega_e = \nabla u^e 3V^e \tag{A.8}$$

o gradiente ∇u^e do elemento pode ser escrito como:

$$\nabla u^e = \frac{1}{3V^e}\left(\sum_{\Gamma_{IJ}^e}\left(\frac{u_I + u_J}{2}\right)2\vec{D}_{IJ}\right) \tag{A.9}$$

onde V^e corresponde a 1/3 da área total do elemento e $\vec{D}_{IJ}$ representa 1/2 da aresta.
Escrevendo a Eq. (A.9) para todos os elementos da malha, fica:

$$\vec{\nabla} u^{e_1} = \frac{1}{3V^e}\left(\frac{u_1+u_2}{2}2.\vec{D}_{12} + \frac{u_2+u_3}{2}2.\vec{D}_{23} + \frac{u_1+u_3}{2}2.\vec{D}_{13}\right) \tag{A.10}$$

$$\vec{\nabla} u^{e_2} = \frac{1}{3V^{e_2}}\left(\frac{u_1+u_3}{2}2.\vec{D}_{13} + \frac{u_3+u_4}{2}2.\vec{D}_{34} + \frac{u_4+u_1}{2}2.\vec{D}_{14}\right) \tag{A.11}$$

$$\vec{\nabla} u^{e_3} = \frac{1}{3V^{e_3}}\left(\frac{u_1+u_4}{2}2.\vec{D}_{14} + \frac{u_4+u_2}{2}2.\vec{D}_{24} + \frac{u_2+u_1}{2}2.\vec{D}_{12}\right) \tag{A.12}$$

Rearranjando as Eq. (A.10) a (A.12), fica:

$$\vec{\nabla} u^{e_1} = \frac{1}{3V^{e_1}}\left[u_1\left(\vec{D}_{12}+\vec{D}_{13}\right) + u_2\left(\vec{D}_{12}+\vec{D}_{23}\right) + u_3\left(\vec{D}_{13}+\vec{D}_{23}\right)\right] \tag{A.13}$$

$$\vec{\nabla}u^{e_2} = \frac{1}{3V^{e_2}}\left[u_1\left(\vec{D}_{13}+\vec{D}_{14}\right)+u_3\left(\vec{D}_{13}+\vec{D}_{34}\right)+u_4\left(\vec{D}_{14}+\vec{D}_{34}\right)\right] \tag{A.14}$$

$$\vec{\nabla}u^{e_3} = \frac{1}{3V^{e_3}}\left[u_1\left(\vec{D}_{12}+\vec{D}_{14}\right)+u_2\left(\vec{D}_{12}+\vec{D}_{24}\right)+u_4\left(\vec{D}_{14}+\vec{D}_{24}\right)\right] \tag{A.15}$$

Sabendo que:

$$\vec{D}_{IJ} = -\vec{D}_{JI} \tag{A.16}$$

$$\vec{D}_{IJ}+\vec{D}_{IK}+\vec{D}_{JK} = 0 \tag{A.17}$$

As Eq. (A.13) a (A.15) podem ser reescritas como:

$$\vec{\nabla}u^{e_1} = \frac{1}{3V^{e_1}}\left[u_1\vec{D}_{23}+u_2\vec{D}_{13}+u_3\vec{D}_{12}\right] \tag{A.18}$$

$$\vec{\nabla}u^{e_2} = \frac{1}{3V^{e_2}}\left[u_1\vec{D}_{34}+u_3\vec{D}_{14}+u_4\vec{D}_{13}\right] \tag{A.19}$$

$$\vec{\nabla}u^{e_3} = \frac{1}{3V^{e_3}}\left[u_1\vec{D}_{24}+u_2\vec{D}_{14}+u_4\vec{D}_{12}\right] \tag{A.20}$$

Com os gradientes dos elementos agora conhecidos, as Eq. (A.18) a (A.20) são introduzidas na Eq. (A.5) que calcula o termo de fluxo para o nó 1. Em seguida são deduzidas as equações para os nós 1, 2, 3 e 4 respectivamente.

Nó 1:

$$-\sum_{m}^{NE}\left(K^{e_m}\vec{\nabla}u^{e_m}\right).\vec{C}_1^{e_m} = -\left[\left(K^{e_1}\vec{\nabla}u^{e_1}\right)\left(\vec{C}_{12}^{e_1}+\vec{C}_{13}^{e_1}\right)+\left(K^{e_2}\vec{\nabla}u^{e_2}\right)\left(\vec{C}_{13}^{e_2}+\vec{C}_{14}^{e_2}\right)+\left(K^{e_3}\vec{\nabla}u^{e_3}\right)\left(\vec{C}_{12}^{e_3}+\vec{C}_{14}^{e_3}\right)\right] \tag{A.21}$$

$$
\begin{aligned}
&= -\left\{ \frac{K^{e_1}}{3V^{e_1}} \left(u_1 \vec{D}_{23} + u_2 \vec{D}_{13} + u_3 \vec{D}_{12} \right) \left(\vec{C}_{12}^{e_1} + \vec{C}_{13}^{e_1} \right) + \right. \\
&\quad \frac{K^{e_2}}{3V^{e_2}} \left(u_1 \vec{D}_{34} + u_3 \vec{D}_{14} + u_4 \vec{D}_{13} \right) \left(\vec{C}_{13}^{e_2} + \vec{C}_{14}^{e_2} \right) + \\
&\quad \left. \frac{K^{e_3}}{3V^{e_3}} \left(u_1 \vec{D}_{24} + u_2 \vec{D}_{14} + u_4 \vec{D}_{12} \right) \left(\vec{C}_{12}^{e_3} + \vec{C}_{14}^{e_3} \right) \right\}
\end{aligned} \tag{A.22}
$$

Rearranjando a Eq. (A.22), fica:

$$
\begin{aligned}
&= -\frac{1}{3} \left\{ u_1 \left[\frac{K^{e_1} \vec{D}_{23}}{V^{e_1}} \left(\vec{C}_{12}^{e_1} + \vec{C}_{13}^{e_1} \right) + \frac{K^{e_2} \vec{D}_{34}}{V^{e_2}} \left(\vec{C}_{13}^{e_2} + \vec{C}_{14}^{e_2} \right) + \frac{K^{e_3} \vec{D}_{24}}{V^{e_3}} \left(\vec{C}_{12}^{e_3} + \vec{C}_{14}^{e_3} \right) \right] + \right. \\
&\quad u_2 \left[\frac{K^{e_1} \vec{D}_{13}}{V^{e_1}} \left(\vec{C}_{12}^{e_1} + \vec{C}_{13}^{e_1} \right) + \frac{K^{e_3} \vec{D}_{14}}{V^{e_3}} \left(\vec{C}_{12}^{e_3} + \vec{C}_{14}^{e_3} \right) \right] + \\
&\quad u_3 \left[\frac{K^{e_1} \vec{D}_{12}}{V^{e_1}} \left(\vec{C}_{12}^{e_1} + \vec{C}_{13}^{e_1} \right) + \frac{K^{e_2} \vec{D}_{14}}{V^{e_2}} \left(\vec{C}_{13}^{e_2} + \vec{C}_{14}^{e_2} \right) \right] + \\
&\quad \left. u_4 \left[\frac{K^{e_2} \vec{D}_{13}}{V^{e_2}} \left(\vec{C}_{13}^{e_2} + \vec{C}_{14}^{e_2} \right) + \frac{K^{e_3} \vec{D}_{12}}{V^{e_3}} \left(\vec{C}_{12}^{e_3} + \vec{C}_{14}^{e_3} \right) \right] \right\}
\end{aligned} \tag{A.23}
$$

Na Eq. (A.23), nota-se que u_2, u_3, u_4 dependem de coeficientes $\vec{C}_{JK}$ que não concorrem com o nó I. Isso inviabiliza a montagem da matriz das equações nodais por aresta. Logo, é preciso escrever o coeficiente $\vec{C}_{JK}$ em função do nó I. Isso é feito através de um simples artifício geométrico. Por exemplo, u_2 depende de $\vec{C}_{13}$ que não concorre com o nó 2. Observando o elemento 1 do malha em questão, tem-se que:

$$
\vec{D}_{13} + \vec{D}_{12} + \vec{C}_{13} + \vec{C}_{12} = 0 \tag{A.24}
$$

Como conseqüência do teorema da divergência. Além disso, do triangulo 1B3 se obtém pelo mesmo princípio que:

$$
2\vec{D}_{13} + \vec{D}_{12} = 3\vec{C}_{12} \tag{A.25}
$$

Onde $\vec{C}_{12} = (1/3)\overline{B3}$ (A.26)

e $\overline{B3}$ é a distância do ponto média da aresta 1-2 ao nó 3.

Introduzindo Eq. (A.26) em (A.25) em função de $\vec{D}_{13}$, fica:

$$\vec{C}_{12} + \vec{C}_{13} = \frac{3\vec{C}_{12} + \vec{D}_{12}}{2} - \vec{D}_{12} \tag{A.27}$$

$$\vec{C}_{12} + \vec{C}_{13} = \frac{3\vec{C}_{12} - \vec{D}_{12}}{2} \tag{A.28}$$

Como se nota na Eq. (A.28), $\vec{C}_{13}$ pode ser escrito em função das arestas que usam o nó 2. A Eq. (A.28) pode então ser escrita na forma genérica:

$$\vec{C}_{IJ} + \vec{C}_{IK} = \frac{3\vec{C}_{1IJ} - \vec{D}_{IJ}}{2} \tag{A.29}$$

Adotando-se este artifício na Eq. (A.23), fica:

$$\begin{aligned}
&-\sum_{m}^{NE}\left(K^{e_m}\vec{\nabla}u^{e_m}\right).\vec{C}_1^{e_m} = \\
&-\frac{1}{3}\left\{u_1\left[\frac{K^{e_1}\vec{D}_{23}}{V^{e_1}}\left(\vec{C}_{12}^{e_1}+\vec{C}_{13}^{e_1}\right)+\frac{K^{e_2}\vec{D}_{34}}{V^{e_2}}\left(\vec{C}_{13}^{e_2}+\vec{C}_{14}^{e_2}\right)+\frac{K^{e_3}\vec{D}_{24}}{V^{e_3}}\left(\vec{C}_{12}^{e_3}+\vec{C}_{14}^{e_3}\right)\right]+\right. \\
&\quad u_2\left[\frac{K^{e_1}\vec{D}_{13}}{V^{e_1}}\left(\vec{C}_{12}^{e_1}+\frac{\vec{C}_{12}^{e_1}-\vec{D}_{12}}{2}\right)+\frac{K^{e_3}\vec{D}_{14}}{V^{e_3}}\left(\vec{C}_{12}^{e_3}+\frac{\vec{C}_{12}^{e_3}-\vec{D}_{12}}{2}\right)\right]+ \\
&\quad u_3\left[\frac{K^{e_1}\vec{D}_{12}}{V^{e_1}}\left(\vec{C}_{13}^{e_1}+\frac{\vec{C}_{13}^{e_1}-\vec{D}_{13}}{2}\right)+\frac{K^{e_2}\vec{D}_{14}}{V^{e_2}}\left(\vec{C}_{13}^{e_2}+\frac{\vec{C}_{13}^{e_2}-\vec{D}_{13}}{2}\right)\right]+ \\
&\quad \left.u_4\left[\frac{K^{e_2}\vec{D}_{13}}{V^{e_2}}\left(\vec{C}_{14}^{e_2}+\frac{\vec{C}_{14}^{e_2}-\vec{D}_{14}}{2}\right)+\frac{K^{e_3}\vec{D}_{12}}{V^{e_3}}\left(\vec{C}_{14}^{e_3}+\frac{\vec{C}_{14}^{e_3}-\vec{D}_{14}}{2}\right)\right]\right\}
\end{aligned} \tag{A.30}$$

$$
\begin{aligned}
&-\sum_{m}^{NE}\left(K^{e_m}\vec{\nabla}u^{e_m}\right).\vec{C}_1^{e_m} = \\
&-\frac{1}{3}\Bigg\{u_1\left[\frac{K^{e_1}\vec{D}_{23}}{V^{e_1}}\left(\vec{C}_{12}^{e_1}+\vec{C}_{13}^{e_1}\right)+\frac{K^{e_2}\vec{D}_{34}}{V^{e_2}}\left(\vec{C}_{13}^{e_2}+\vec{C}_{14}^{e_2}\right)+\frac{K^{e_3}\vec{D}_{24}}{V^{e_3}}\left(\vec{C}_{12}^{e_3}+\vec{C}_{14}^{e_3}\right)\right]+ \\
&\quad u_2\left[\frac{K^{e_1}\vec{D}_{13}}{V^{e_1}}\left(\frac{3\vec{C}_{12}^{e_1}-\vec{D}_{12}}{2}\right)+\frac{K^{e_3}\vec{D}_{14}}{V^{e_3}}\left(\frac{3\vec{C}_{12}^{e_3}-\vec{D}_{12}}{2}\right)\right]+ \\
&\quad u_3\left[\frac{K^{e_1}\vec{D}_{12}}{V^{e_1}}\left(\frac{3\vec{C}_{13}^{e_1}-\vec{D}_{13}}{2}\right)+\frac{K^{e_2}\vec{D}_{14}^{e_2}}{V^{e_2}}\left(\frac{3\vec{C}_{13}^{e_2}-\vec{D}_{13}}{2}\right)\right]+ \\
&\quad u_4\left[\frac{K^{e_2}\vec{D}_{13}}{V^{e_2}}\left(\frac{3\vec{C}_{14}^{e_2}-\vec{D}_{14}}{2}\right)+\frac{K^{e_3}\vec{D}_{12}}{V^{e_3}}\left(\frac{3\vec{C}_{14}^{e_3}-\vec{D}_{14}}{2}\right)\right]\Bigg\}
\end{aligned}
\tag{A.31}
$$

Nó 2:

$$
-\sum_{m}^{NE}\left(K^{e_m}\vec{\nabla}u^{e_m}\right).\vec{C}_2^{e_m} = -\left[\left(K^{e_1}\vec{\nabla}u^{e_1}\right)\left(-\vec{C}_{12}^{e_1}+\vec{C}_{23}^{e_1}\right)+\left(K^{e_3}\vec{\nabla}u^{e_3}\right)\left(-\vec{C}_{12}^{e_3}+\vec{C}_{24}^{e_3}\right)\right] \tag{A.32}
$$

Manipulando apenas o lado direito da Eq. (A.32):

$$
\begin{aligned}
&= -\Bigg[\frac{K^{e_1}}{3V^{e_1}}\left(u_1\vec{D}_{23}+u_2\vec{D}_{13}+u_3\vec{D}_{12}\right)\left(-\vec{C}_{12}^{e_1}+\vec{C}_{23}^{e_1}\right)+ \\
&\quad \frac{K^{e_3}}{3V^{e_3}}\left(u_1\vec{D}_{24}+u_2\vec{D}_{14}+u_4\vec{D}_{12}\right)\left(-\vec{C}_{12}^{e_3}+\vec{C}_{24}^{e_3}\right)\Bigg]
\end{aligned}
\tag{A.32}
$$

$$
\begin{aligned}
&= -\Bigg\{u_1\left[\frac{K^{e_1}\vec{D}_{23}}{3V^{e_1}}\left(-\vec{C}_{12}^{e_1}+\vec{C}_{23}^{e_1}\right)+\frac{K^{e_3}\vec{D}_{24}}{3V^{e_3}}\left(-\vec{C}_{12}^{e_3}+\vec{C}_{24}^{e_3}\right)\right]+ \\
&\quad u_2\left[\frac{K^{e_1}\vec{D}_{13}}{3V^{e_1}}\left(-\vec{C}_{12}^{e_1}+\vec{C}_{23}^{e_1}\right)+\frac{K^{e_3}\vec{D}_{14}}{3V^{e_3}}\left(-\vec{C}_{12}^{e_3}+\vec{C}_{24}^{e_3}\right)\right]+ \\
&\quad u_3\left[\frac{K^{e_1}\vec{D}_{12}}{3V^{e_1}}\left(-\vec{C}_{12}^{e_1}+\vec{C}_{23}^{e_1}\right)\right]+ \\
&\quad u_4\left[\frac{K^{e_3}\vec{D}_{12}}{3V^{e_3}}\left(-\vec{C}_{12}^{e_3}+\vec{C}_{24}^{e_3}\right)\right]\Bigg\}
\end{aligned}
\tag{A.33}
$$

$$
\begin{aligned}
=-\frac{1}{3}\Bigg\{ & u_1\left[\frac{K^{e_1}\vec{D}_{23}}{V^{e_1}}\left(-\vec{C}_{12}^{e_1}+\frac{-\vec{C}_{12}^{e_1}-\vec{D}_{12}}{2}\right)+\frac{K^{e_3}\vec{D}_{24}}{V^{e_3}}\left(-\vec{C}_{12}^{e_3}+\frac{-\vec{C}_{12}^{e_1}+\vec{D}_{12}}{2}\right)\right]+ \\
& u_2\left[\frac{K^{e_1}\vec{D}_{13}}{V^{e_1}}\left(-\vec{C}_{12}^{e_1}+\vec{C}_{23}^{e_1}\right)+\frac{K^{e_3}\vec{D}_{24}}{V^{e_3}}\left(-\vec{C}_{12}^{e_3}+\vec{C}_{24}^{e_3}\right)\right]+ \\
& u_3\left[\frac{K^{e_1}\vec{D}_{12}}{V^{e_1}}\left(\frac{\vec{C}_{23}^{e_1}-\vec{D}_{23}}{2}+\vec{C}_{23}^{e_1}\right)\right]+ \\
& u_4\left[\frac{K^{e_3}\vec{D}_{12}}{V^{e_3}}\left(\frac{\vec{C}_{24}^{e_3}-\vec{D}_{24}}{2}+\vec{C}_{24}^{e_3}\right)\right]\Bigg\}
\end{aligned}
\tag{A.34}
$$

$$
\begin{aligned}
=-\frac{1}{3}\Bigg\{ & u_1\left[\frac{K^{e_1}\vec{D}_{23}}{V^{e_1}}\left(\frac{-3\vec{C}_{12}^{e_1}-\vec{D}_{12}}{2}\right)+\frac{K^{e_3}\vec{D}_{24}}{V^{e_3}}\left(\frac{-3\vec{C}_{12}^{e_1}-\vec{D}_{12}}{2}\right)\right]+ \\
& u_2\left[\frac{K^{e_1}\vec{D}_{13}}{V^{e_1}}\left(-\vec{C}_{12}^{e_1}+\vec{C}_{23}^{e_1}\right)+\frac{K^{e_3}\vec{D}_{24}}{V^{e_3}}\left(-\vec{C}_{12}^{e_3}+\vec{C}_{24}^{e_3}\right)\right]+ \\
& u_3\left[\frac{K^{e_1}\vec{D}_{12}}{V^{e_1}}\left(\frac{3\vec{C}_{23}^{e_1}-\vec{D}_{23}}{2}\right)\right]+ \\
& u_4\left[\frac{K^{e_3}\vec{D}_{12}}{V^{e_3}}\left(\frac{3\vec{C}_{24}^{e_3}-\vec{D}_{24}}{2}\right)\right]\Bigg\}
\end{aligned}
\tag{A.35}
$$

Nó 3:

$$
-\sum_{m}^{NE}\left(K^{e_m}\vec{\nabla}u^{e_m}\right)\vec{C}_3^{e_m}=-\left[\left(K^{e_1}\vec{\nabla}u^{e_1}\right)\left(-\vec{C}_{13}^{e_1}-\vec{C}_{23}^{e_1}\right)+\left(K^{e_2}\vec{\nabla}u^{e_2}\right)\left(-\vec{C}_{13}^{e_2}+\vec{C}_{34}^{e_2}\right)\right] \tag{A.36}
$$

Manipulando apenas o lado direito da Eq. (A.36):

$$
\begin{aligned}
=-\Bigg[& \frac{K^{e_1}}{3V^{e_1}}\left(u_1\vec{D}_{23}+u_2\vec{D}_{13}+u_3\vec{D}_{12}\right)\left(-\vec{C}_{13}^{e_1}-\vec{C}_{23}^{e_1}\right)+ \\
& \frac{K^{e_2}}{3V^{e_2}}\left(u_1\vec{D}_{34}+u_3\vec{D}_{14}+u_4\vec{D}_{13}\right)\left(-\vec{C}_{13}^{e_2}+\vec{C}_{34}^{e_2}\right)\Bigg]
\end{aligned}
\tag{A.37}
$$

$$= -\frac{1}{3}\left\{ u_1\left[\frac{K^{e_1}\vec{D}_{23}}{V^{e_1}}\left(-\vec{C}_{13}^{e_1}-\vec{C}_{23}^{e_1}\right)+\frac{K^{e_2}\vec{D}_{34}}{V^{e_2}}\left(-\vec{C}_{13}^{e_2}+\vec{C}_{34}^{e_2}\right)\right]+ \right.$$
$$u_2\left[\frac{K^{e_1}\vec{D}_{13}}{V^{e_1}}\left(-\vec{C}_{13}^{e_1}-\vec{C}_{23}^{e_1}\right)\right]+$$
$$u_3\left[\frac{K^{e_1}\vec{D}_{12}}{V^{e_1}}\left(-\vec{C}_{13}^{e_1}-\vec{C}_{23}^{e_1}\right)+\frac{K^{e_2}\vec{D}_{14}}{V^{e_2}}\left(-\vec{C}_{13}^{e_2}+\vec{C}_{34}^{e_2}\right)\right]+ \tag{A.38}$$
$$\left. u_4\left[\frac{K^{e_2}\vec{D}_{13}}{V^{e_2}}\left(-\vec{C}_{13}^{e_2}+\vec{C}_{34}^{e_2}\right)\right]\right\}$$

$$= -\frac{1}{3}\left\{ u_1\left[\frac{K^{e_1}\vec{D}_{23}}{V^{e_1}}\left(-\vec{C}_{13}^{e_1}+\frac{-\vec{C}_{13}^{e_1}-\vec{D}_{13}}{2}\right)+\frac{K^{e_2}\vec{D}_{34}}{V^{e_2}}\left(-\vec{C}_{13}^{e_2}+\frac{-\vec{C}_{13}^{e_2}-\vec{D}_{13}}{2}\right)\right]+ \right.$$
$$u_2\left[\frac{K^{e_1}\vec{D}_{13}}{V^{e_1}}\left(\frac{-\vec{C}_{23}^{e_1}-\vec{D}_{23}}{2}-\vec{C}_{23}^{e_1}\right)\right]+$$
$$u_3\left[\frac{K^{e_1}\vec{D}_{12}}{V^{e_1}}\left(-\vec{C}_{13}^{e_1}-\vec{C}_{23}^{e_1}\right)+\frac{K^{e_2}\vec{D}_{14}}{V^{e_2}}\left(-\vec{C}_{13}^{e_2}+\vec{C}_{34}^{e_2}\right)\right]+ \tag{A.39}$$
$$\left. u_4\left[\frac{K^{e_2}\vec{D}_{13}}{V^{e_2}}\left(\frac{\vec{C}_{34}^{e_2}-\vec{D}_{34}}{2}+\vec{C}_{34}^{e_2}\right)\right]\right\}$$

$$= -\frac{1}{3}\left\{ u_1\left[\frac{K^{e_1}\vec{D}_{23}}{V^{e_1}}\left(\frac{-3\vec{C}_{13}^{e_1}-\vec{D}_{13}}{2}\right)+\frac{K^{e_2}\vec{D}_{34}}{V^{e_2}}\left(\frac{-3\vec{C}_{13}^{e_2}-\vec{D}_{13}}{2}\right)\right]+ \right.$$
$$u_2\left[\frac{K^{e_1}\vec{D}_{13}}{V^{e_1}}\left(\frac{-3\vec{C}_{23}^{e_1}-\vec{D}_{23}}{2}\right)\right]+$$
$$u_3\left[\frac{K^{e_1}\vec{D}_{12}}{V^{e_1}}\left(-\vec{C}_{13}^{e_1}-\vec{C}_{23}^{e_1}\right)+\frac{K^{e_2}\vec{D}_{14}}{V^{e_2}}\left(-\vec{C}_{13}^{e_2}+\vec{C}_{34}^{e_2}\right)\right]+ \tag{A.40}$$
$$\left. u_4\left[\frac{K^{e_2}\vec{D}_{13}}{V^{e_2}}\left(\frac{3\vec{C}_{34}^{e_2}-\vec{D}_{34}}{2}\right)\right]\right\}$$

Nó 4:

$$-\sum_{m}^{NE}\left(K^{e_m}\vec{\nabla}u^{e_m}\right).\vec{C}_4^{e_m} = -\left[\left(K^{e_2}\vec{\nabla}u^{e_2}\right)\left(-\vec{C}_{34}^{e_2}-\vec{C}_{14}^{e_2}\right)+\left(K^{e_3}\vec{\nabla}u^{e_3}\right)\left(-\vec{C}_{14}^{e_3}-\vec{C}_{24}^{e_3}\right)\right] \tag{A.41}$$

Manipulando apenas o lado direito da Eq. (A.41):

$$= -\left[\frac{K^{e_1}}{3V^{e_1}} \left(u_1 \vec{D}_{34} + u_3 \vec{D}_{14} + u_4 \vec{D}_{13} \right) \left(-\vec{C}_{34}^{e_2} - \vec{C}_{14}^{e_2} \right) + \right.$$
$$\left. \frac{K^{e_3}}{3V^{e_3}} \left(u_1 \vec{D}_{24} + u_2 \vec{D}_{14} + u_4 \vec{D}_{12} \right) \left(-\vec{C}_{14}^{e_3} - \vec{C}_{24}^{e_3} \right) \right] \tag{A.42}$$

$$= -\frac{1}{3} \left\{ u_1 \left[\frac{K^{e_2} \vec{D}_{34}}{V^{e_2}} \left(-\vec{C}_{34}^{e_2} - \vec{C}_{14}^{e_2} \right) + \frac{K^{e_3} \vec{D}_{24}}{V^{e_3}} \left(-\vec{C}_{14}^{e_3} - \vec{C}_{24}^{e_3} \right) \right] + \right.$$
$$u_2 \left[\frac{K^{e_3} \vec{D}_{14}}{V^{e_3}} \left(-\vec{C}_{14}^{e_3} - \vec{C}_{24}^{e_3} \right) \right] +$$
$$u_3 \left[\frac{K^{e_2} \vec{D}_{14}}{V^{e_2}} \left(-\vec{C}_{34}^{e_2} - \vec{C}_{14}^{e_2} \right) \right] +$$
$$\left. u_4 \left[\frac{K^{e_2} \vec{D}_{13}}{V^{e_2}} \left(-\vec{C}_{34}^{e_2} - \vec{C}_{14}^{e_2} \right) + \frac{K^{e_3} \vec{D}_{12}}{V^{e_3}} \left(-\vec{C}_{14}^{e_3} - \vec{C}_{24}^{e_3} \right) \right] \right\} \tag{A.43}$$

$$= -\frac{1}{3} \left\{ u_1 \left[\frac{K^{e_2} \vec{D}_{34}}{V^{e_2}} \left(-\vec{C}_{14}^{e_2} + \frac{-\vec{C}_{14}^{e_2} - \vec{D}_{14}}{2} \right) + \frac{K^{e_3} \vec{D}_{14}}{V^{e_3}} \left(-\vec{C}_{14}^{e_3} + \frac{-\vec{C}_{14}^{e_3} - \vec{D}_{14}}{2} \right) \right] + \right.$$
$$u_2 \left[\frac{K^{e_3} \vec{D}_{14}}{V^{e_3}} \left(-\vec{C}_{24}^{e_3} + \frac{-\vec{C}_{24}^{e_3} - \vec{D}_{24}}{2} \right) \right] +$$
$$u_3 \left[\frac{K^{e_2} \vec{D}_{14}}{V^{e_2}} \left(-\vec{C}_{34}^{e_2} + \frac{-\vec{C}_{34}^{e_2} - \vec{D}_{34}}{2} \right) \right] +$$
$$\left. u_4 \left[\frac{K^{e_2} \vec{D}_{13}}{V^{e_2}} \left(-\vec{C}_{34}^{e_2} - \vec{C}_{14}^{e_2} \right) + \frac{K^{e_3} \vec{D}_{12}}{V^{e_3}} \left(-\vec{C}_{14}^{e_3} - \vec{C}_{24}^{e_3} \right) \right] \right\} \tag{A.44}$$

$$= -\frac{1}{3} \left\{ u_1 \left[\frac{K^{e_2} \vec{D}_{34}}{V^{e_2}} \left(\frac{-3\vec{C}_{14}^{e_2} - \vec{D}_{14}}{2} \right) + \frac{K^{e_3} \vec{D}_{14}}{V^{e_3}} \left(\frac{-3\vec{C}_{14}^{e_3} - \vec{D}_{14}}{2} \right) \right] + \right.$$
$$u_2 \left[\frac{K^{e_3} \vec{D}_{14}}{V^{e_3}} \left(\frac{-3\vec{C}_{24}^{e_3} - \vec{D}_{24}}{2} \right) \right] +$$
$$u_3 \left[\frac{K^{e_2} \vec{D}_{14}}{V^{e_2}} \left(\frac{-3\vec{C}_{34}^{e_2} - \vec{D}_{34}}{2} \right) \right] +$$
$$\left. u_4 \left[\frac{K^{e_2} \vec{D}_{13}}{V^{e_2}} \left(-\vec{C}_{34}^{e_2} - \vec{C}_{14}^{e_2} \right) + \frac{K^{e_3} \vec{D}_{12}}{V^{e_3}} \left(-\vec{C}_{14}^{e_3} - \vec{C}_{24}^{e_3} \right) \right] \right\} \tag{A.45}$$

Escrevendo as Eq. (A.31), (A.35), (A.40) e (A.45) na forma matricial, fica:

$$-\frac{1}{3}\begin{pmatrix} \frac{K^{e_1}\vec{D}_{23}}{V^{e_1}}\left(\vec{C}_{12}^{e_1}+\vec{C}_{13}^{e_1}\right)+\frac{K^{e_2}\vec{D}_{34}}{V^{e_2}}\left(\vec{C}_{13}^{e_2}+\vec{C}_{14}^{e_2}\right)+\frac{K^{e_3}\vec{D}_{24}}{V^{e_3}}\left(\vec{C}_{12}^{e_3}+\vec{C}_{14}^{e_3}\right) & \frac{K^{e_1}\vec{D}_{13}}{V^{e_1}}\left(\frac{3\vec{C}_{12}^{e_1}-\vec{D}_{12}}{2}\right)+\frac{K^{e_3}\vec{D}_{14}}{V^{e_3}}\left(\frac{3\vec{C}_{12}^{e_3}-\vec{D}_{12}}{2}\right) & \frac{K^{e_1}\vec{D}_{12}}{V^{e_1}}\left(\frac{3\vec{C}_{13}^{e_1}-\vec{D}_{13}}{2}\right)+\frac{K^{e_2}\vec{D}_{14}}{V^{e_2}}\left(\frac{3\vec{C}_{13}^{e_2}-\vec{D}_{13}}{2}\right) & \frac{K^{e_2}\vec{D}_{13}}{V^{e_2}}\left(\frac{3\vec{C}_{14}^{e_2}-\vec{D}_{14}}{2}\right)+\frac{K^{e_3}\vec{D}_{12}}{V^{e_3}}\left(\frac{3\vec{C}_{14}^{e_3}-\vec{D}_{14}}{2}\right) \\ \frac{K^{e_1}\vec{D}_{23}}{V^{e_1}}\left(\frac{-3\vec{C}_{12}^{e_1}-\vec{D}_{12}}{2}\right)+\frac{K^{e_3}\vec{D}_{24}}{V^{e_3}}\left(\frac{-3\vec{C}_{12}^{e_1}-\vec{D}_{12}}{2}\right) & \frac{K^{e_1}\vec{D}_{13}}{V^{e_1}}\left(-\vec{C}_{12}^{e_1}+\vec{C}_{23}^{e_1}\right)+\frac{K^{e_3}\vec{D}_{14}}{V^{e_3}}\left(-\vec{C}_{12}^{e_3}+\vec{C}_{24}^{e_3}\right) & \frac{K^{e_1}\vec{D}_{12}}{V^{e_1}}\left(\frac{3\vec{C}_{23}^{e_1}-\vec{D}_{23}}{2}\right) & \frac{K^{e_3}\vec{D}_{12}}{V^{e_3}}\left(\frac{3\vec{C}_{24}^{e_3}-\vec{D}_{24}}{2}\right) \\ \frac{K^{e_1}\vec{D}_{23}}{V^{e_1}}\left(\frac{-3\vec{C}_{13}^{e_1}-\vec{D}_{13}}{2}\right)+\frac{K^{e_2}\vec{D}_{34}}{V^{e_2}}\left(\frac{-3\vec{C}_{13}^{e_2}-\vec{D}_{13}}{2}\right) & \frac{K^{e_1}\vec{D}_{13}}{V^{e_1}}\left(\frac{-3\vec{C}_{23}^{e_1}-\vec{D}_{23}}{2}\right) & \frac{K^{e_1}\vec{D}_{13}}{V^{e_1}}\left(-\vec{C}_{13}^{e_1}-\vec{C}_{23}^{e_1}\right)+\frac{K^{e_2}\vec{D}_{14}}{V^{e_2}}\left(-\vec{C}_{13}^{e_2}+\vec{C}_{34}^{e_2}\right) & \frac{K^{e_2}\vec{D}_{13}}{V^{e_2}}\left(\frac{3\vec{C}_{34}^{e_2}-\vec{D}_{34}}{2}\right) \\ \frac{K^{e_2}\vec{D}_{34}}{V^{e_2}}\left(\frac{-3\vec{C}_{14}^{e_2}-\vec{D}_{14}}{2}\right)+\frac{K^{e_3}\vec{D}_{14}}{V^{e_3}}\left(\frac{-3\vec{C}_{14}^{e_3}-\vec{D}_{14}}{2}\right) & \frac{K^{e_3}\vec{D}_{14}}{V^{e_3}}\left(\frac{-3\vec{C}_{24}^{e_3}-\vec{D}_{24}}{2}\right) & \frac{K^{e_2}\vec{D}_{14}}{V^{e_2}}\left(\frac{-3\vec{C}_{34}^{e_2}+\vec{D}_{34}}{2}\right) & \frac{K^{e_2}\vec{D}_{13}}{V^{e_2}}\left(-\vec{C}_{34}^{e_2}-\vec{C}_{14}^{e_2}\right)+\frac{K^{e_3}\vec{D}_{12}}{V^{e_3}}\left(-\vec{C}_{14}^{e_3}-\vec{C}_{24}^{e_3}\right) \end{pmatrix} \tag{A.46}$$

Manipulando os sinais dos elementos da parte abaixo da diagonal principal e fazendo $\vec{W}_{LK}^{e_m}=\frac{K^{e_n}\vec{D}_{LK}^{e_n}}{V^{e_n}}$, fica:

$$-\frac{1}{6}\begin{pmatrix} 2\left[\vec{W}_{23}^{e_1}\left(\vec{C}_{12}^{e_1}+\vec{C}_{13}^{e_1}\right)+\vec{W}_{34}^{e_2}\left(\vec{C}_{13}^{e_2}+\vec{C}_{14}^{e_2}\right)+\vec{W}_{24}^{e_3}\left(\vec{C}_{12}^{e_3}+\vec{C}_{14}^{e_3}\right)\right] & W_{13}^{e_1}\left(3\vec{C}_{12}^{e_1}-\vec{D}_{12}\right)+W_{14}^{e_3}\left(3\vec{C}_{12}^{e_3}-\vec{D}_{12}\right) & \vec{W}_{12}^{e_1}\left(3\vec{C}_{13}^{e_1}-\vec{D}_{13}\right)+\vec{W}_{14}^{e_2}\left(3\vec{C}_{13}^{e_2}-\vec{D}_{13}\right) & \vec{W}_{13}^{e_2}\left(3\vec{C}_{14}^{e_2}-\vec{D}_{14}\right)+\vec{W}_{12}^{e_3}\left(3\vec{C}_{14}^{e_3}-\vec{D}_{14}\right) \\ -\left[\vec{W}_{23}^{e_1}\left(3\vec{C}_{12}^{e_1}-\vec{D}_{12}\right)+\vec{W}_{24}^{e_3}\left(3\vec{C}_{12}^{e_1}-\vec{D}_{12}\right)\right] & 2\left[W_{13}^{e_1}\left(-\vec{C}_{12}^{e_1}+\vec{C}_{23}^{e_1}\right)+W_{14}^{e_3}\left(-\vec{C}_{12}^{e_3}+\vec{C}_{24}^{e_3}\right)\right] & \vec{W}_{12}^{e_1}\left(3\vec{C}_{23}^{e_1}-\vec{D}_{23}\right) & \vec{W}_{12}^{e_3}\left(3\vec{C}_{24}^{e_3}-\vec{D}_{24}\right) \\ -\left[\vec{W}_{23}^{e_1}\left(3\vec{C}_{13}^{e_1}-\vec{D}_{13}\right)-\vec{W}_{34}^{e_2}\left(3\vec{C}_{13}^{e_2}-\vec{D}_{13}\right)\right] & -\left[W_{13}^{e_1}\left(3\vec{C}_{23}^{e_1}-\vec{D}_{23}\right)\right] & 2\left[\vec{W}_{12}^{e_1}\left(-\vec{C}_{13}^{e_1}-\vec{C}_{23}^{e_1}\right)+\vec{W}_{14}^{e_2}\left(-\vec{C}_{13}^{e_2}+\vec{C}_{34}^{e_2}\right)\right] & \vec{W}_{13}^{e_2}\left(3\vec{C}_{34}^{e_2}-\vec{D}_{34}\right) \\ -\left[\vec{W}_{34}^{e_2}\left(3\vec{C}_{14}^{e_2}+\vec{D}_{14}\right)-\vec{W}_{14}^{e_3}\left(3\vec{C}_{14}^{e_3}-\vec{D}_{14}\right)\right] & -\left[W_{14}^{e_3}\left(3\vec{C}_{24}^{e_3}-\vec{D}_{24}\right)\right] & -\left[\vec{W}_{14}^{e_2}\left(3\vec{C}_{34}^{e_2}-\vec{D}_{34}\right)\right] & 2\left[\vec{W}_{13}^{e_2}\left(-\vec{C}_{34}^{e_2}-\vec{C}_{14}^{e_2}\right)+\vec{W}_{14}^{e_3}\left(-\vec{C}_{14}^{e_3}-\vec{C}_{24}^{e_3}\right)\right] \end{pmatrix} \tag{A.47}$$

A seguir, são montadas as matrizes para as arestas no interior do domínio.

Aresta 1-2

$$-\frac{1}{6}\begin{pmatrix} 2\left[W_{23}^{e_1}\vec{C}_{12}^{e_1}+W_{24}^{e_3}\vec{C}_{12}^{e_3}\right] & W_{13}^{e_1}\left(3\vec{C}_{12}^{e_1}-\vec{D}_{12}\right)+W_{14}^{e_3}\left(3\vec{C}_{12}^{e_3}-\vec{D}_{12}\right) \\ -\left[W_{23}^{e_1}\left(3\vec{C}_{12}^{e_1}+\vec{D}_{12}\right)+W_{24}^{e_3}\left(3\vec{C}_{12}^{e_1}+\vec{D}_{12}\right)\right] & -2\left[W_{13}^{e_1}\vec{C}_{12}^{e_1}+W_{14}^{e_3}\vec{C}_{12}^{e_3}\right] \end{pmatrix} \tag{A.49}$$

Aresta 1-3

$$-\frac{1}{6}\begin{pmatrix} 2\left[\vec{W}_{23}^{e_1}\vec{C}_{13}^{e_1}+\vec{W}_{34}^{e_2}\vec{C}_{13}^{e_2}\right] & \vec{W}_{12}^{e_1}\left(3\vec{C}_{13}^{e_1}-\vec{D}_{13}\right)+\vec{W}_{14}^{e_2}\left(3\vec{C}_{13}^{e_2}-\vec{D}_{13}\right) \\ -\left[\vec{W}_{23}^{e_1}\left(3\vec{C}_{13}^{e_1}+\vec{D}_{13}\right)+\vec{W}_{34}^{e_2}\left(3\vec{C}_{13}^{e_2}+\vec{D}_{13}\right)\right] & -2\left[\vec{W}_{12}^{e_1}\vec{C}_{13}^{e_1}+\vec{W}_{14}^{e_2}\vec{C}_{13}^{e_2}\right] \end{pmatrix} \tag{A.50}$$

Aresta 1-4

$$-\frac{1}{6}\begin{pmatrix} 2\left[\vec{W}_{34}^{e_2}\vec{C}_{14}^{e_2}+\vec{W}_{24}^{e_3}\vec{C}_{14}^{e_3}\right] & \vec{W}_{13}^{e_2}\left(3\vec{C}_{14}^{e_2}-\vec{D}_{14}\right)+\vec{W}_{12}^{e_3}\left(3\vec{C}_{14}^{e_3}-\vec{D}_{14}\right) \\ -\left[\vec{W}_{34}^{e_2}\left(3\vec{C}_{14}^{e_2}+\vec{D}_{14}\right)+\vec{W}_{14}^{e_3}\left(3\vec{C}_{14}^{e_3}+\vec{D}_{14}\right)\right] & -2\left[\vec{W}_{13}^{e_2}\vec{C}_{14}^{e_2}+\vec{W}_{12}^{e_3}\vec{C}_{14}^{e_3}\right] \end{pmatrix} \tag{A.51}$$

Matrizes para arestas no contorno. As arestas são: 2-3, 3-4, 2-4

Aresta 2-3

$$-\frac{1}{6}\begin{pmatrix} 2\left[\vec{W}_{13}^{e_1}\vec{C}_{23}^{e_1}\right] & \vec{W}_{12}^{e_1}\left(3\vec{C}_{23}^{e_1}-\vec{D}_{23}\right) \\ -\left[\vec{W}_{13}^{e_1}\left(3\vec{C}_{23}^{e_1}+\vec{D}_{23}\right)\right] & -2\left[\vec{W}_{12}^{e_1}\vec{C}_{23}^{e_1}\right] \end{pmatrix} \tag{A.52}$$

Aresta 3-4

$$-\frac{1}{6}\begin{pmatrix} 2\left[\vec{W}_{14}^{e_2}\vec{C}_{34}^{e_2}\right] & \vec{W}_{13}^{e_2}\left(3\vec{C}_{34}^{e_2}-\vec{D}_{34}\right) \\ -\left[\vec{W}_{14}^{e_2}\left(3\vec{C}_{34}^{e_2}+\vec{D}_{34}\right)\right] & -2\left[\vec{W}_{13}^{e_2}\vec{C}_{34}^{e_2}\right] \end{pmatrix} \tag{A.53}$$

Aresta 2-4

$$-\frac{1}{6}\begin{pmatrix} 2\left(\vec{W}_{14}^{e_3}\vec{C}_{24}^{e_3}\right) & \vec{W}_{12}^{e_3}\left(3\vec{C}_{24}^{e_3}-\vec{D}_{24}\right) \\ -\left[\vec{W}_{14}^{e_3}\left(3\vec{C}_{24}^{e_3}+\vec{D}_{24}\right)\right] & -2\left(\vec{W}_{12}^{e_3}\vec{C}_{24}^{e_3}\right) \end{pmatrix} \tag{A.54}$$

BIBLIOGRAFIA

Aavatsmark, I.; Barkeve, T.; Bøe, Ø.; Mannseth, T. *Discretization on Unstructured Grids for Inhomogeneous, Anisotropic Media. Part I: Derivation of The Methods*. SIAM J. Sci. Comput. v. 19, p. 1700-1716, 1998a.

Aavatsmark, I.; Barkeve, T.; Bøe, Ø.; Mannseth, T. *Discretization on Unstructured Grids for Inhomogeneous, Anisotropic Media. Part II: Discussion and Numerical Results*. SI AM J. Sci. Comput. v. 19, p. 1717-1736, 1998b.

Abou-Kassem, J. H.; Farouq Ali, S. M.; Islam, M. R. *Petroleum Reservoir Simulation – A basic Approach.* Gulf Publishing Company, Houston, Texas, 2006.

Ahmed, T. *Reservoir Engineering Handbook.* Gulf Publishing Company, Houston, Texas, 2000.

Aziz, K., In: *Notes for Petroleum Reservoir Simulation*. Stanford University, Stanford, 1993.

Barth, T. J. In: *Computational Fluid Dynamics: Aspects of Unstructured Grids and Finite Volume Solvers for the Euler and Navier-Stokes Equations*. NASA Ames Research Center, Moffet Field, 1994.

Barth, T. J.; Linton, W. S. *An Unstructured Mesh Newton Solver for Compressible Fluid Flow and Its Parallel Implementation*. Technical Report, AIAA-95-0221, p. 1-16, 1995.

Barton, J. J. e Nackman, L. R. *Scientific and Engineering C++: An Introduction with Advanced Techniques and Examples.* Addison-Wesley, 7ª ed., 2001.

Bastian, P. *Numerical Computation of Multiphase Flows in Porous Media.* Habilitationss chrift, Universität Kiel, Heidelberg, June 1999.

Bear, J. *Dynamics of Fluids in Porous Media*., Dover, New York, 1972.

Binning, P.; Celia, M. A. *Pratical Implementation of the Fractional Flow Approach to Multi-Phase Flow Simulation. Advances in Water Resources*, v. 22, p. 461-478, 1999.

0

BOAST, http://www.netl.doe.gov/technologies/oil-gas/software/simulat.html

Buckley, S. E.; Leverett, M. C. *Mechanism of fluid displacement in porous media*. Petroleum Technology, p. 107-116, 1942.

Buyya, R. *High Performance Cluster Computing: Architectures and Systems.* Prentice Hall, vol 1, 1999.

Byer, T. J. *Preconditioned Newton Methods for Simulation of Reservoirs with Surface Facilities.* PhD Thesis, Stanford University, May 2000.

Cao, H. *Development of techniques for general purpose simulators*. Ph.D. Tese de Doutorado,Stanford University, 2002.

Carvalho, D. K. E.; Lyra, P. R. M.; Willmersdorf. *Towards A Numerical Reservoir Simulator Using an Unstructured Finite Volume Formulation with the Aid of High Performance Tools*. Segundo Workshop dos Programas de Formação de Recursos Humanos para o Setor de Petróleo e Gás da UFPE, Recife-PE. p.31-35, 2002b.

Carvalho, D. K. E.; Lyra, P. R. M.; Willmersdorf. *A First Step Towards a Petroleum Reservoir Simulator Using an Edge -Based Unstructured Finite Volume Formulation*. 2o Congresso Brasileiro de P&D em Petróleo e Gás Natural, Rio De Janeiro, 2003.

Carvalho, D. K. E.; Lyra, P. R. M.; Willmersdorf, R. B. *Two and Three Dimensional Simulation of Immiscible Two-Phase Flows in Porous Media Using an Edge-Based Control Volume Formulation*. Congresso Ibero-Latino Americano Em Métodos Computacionais Em Engenharia (CILAMCE 2005), Guarapari. 2005.

Carvalho, D. K. E.; Luna, B. G. B.; Willmersdorf, R. B.; Silva, R. S.; Lyra, P. R. M. *Some Results on the Accuracy of na Edge-based Finite Volume Formulation for the Solution*

of Elliptic Problems in Heterogeneous and Anisotropic Porous Media. 19th International Congress of Mechanical Engineering, 2007.

Carvalho, D. K. E.; Willmersdorf, R. B.; Silva, R. S.; Luna, B. G. B.; Lyra, P. R. M. *A Vertex-Centered Control Volume Procedure for the Solution of Diffusion Type Problems in Heterogeneous and Anisotropic Porous Media.* IV Congresso Nacional de Engenharia Mecânica (CONEM), 2006.

Carvalho, L. M.; Dickstein, F.; Rodrigues, J. R. P.; Santos, R. W. *Um Esquema GMRES Precondicionado para Simulação de Reservatórios.* Tendencias em Matemática Aplicada e Computacional, 3, No 2, p. 63-72, 2002.

Chavent, G.; Jaffre, J. In: *Mathematical Models and Finite Elements for Reservoir Simulation: Single Phase, Multiphase and Multicomponent Flows Through Porous Media*. Amsterdam, North-Holland, 1986.

Chen, Z.; Ewing, R; Espedal, M. *Multiphase Flow Simulation with Various Boundary Conditions. Computational Methods in Water Resources* Kluwer Academic Publishers, Netherlands, pp. 925-932, 1994.

Chen, Z.; Ewing, R. *Comparison Of Various Formulations Of Three-Phase Flow In Por ous Media*. Journal of Computational Physics, v. 132, p. 362-373, 1997.

Chen, Z.; Huan, G.; LI, B.; LI, W.; Espin, D. *Comparison Of Practical Approaches To Reservoir Simulation*. 2nd Meeting on Reservoir Simulation, Buenos Aires, Argentina, 2002 (em CD-ROM).

Chen, Z.; Huan, G. *Numerical Experiments with Various Formulations for Two Phase Flow in Petroleum Reservoir*. Transport in Porous Media, v. 51, p. 89-102, 2003.

Chen, Z.; Huan, G.; LI, B. *An Improved IMPES Method For Two-Phase Flow in Porous Media*. Transport in Porous Media, v. 54, p. 361-376, 2004.

Craft, B. C., Howkins, M., *Applied Petroleum Reservoir Enginnering*, Prentice Hall, Upper Saddle River, NJ, 2ª ed., 1991.

Crank, J., In: *The Mathematics of Diffusion*. New York, Oxford University Press, 1973.

Crumpton, P. I. *Discretization Multigrid Solution of Elliptic Equations with Mixed Deriva tives Terms and Strongly Discontonous Coefficients*. Journal of Computational Physics, v. 116, p. 343-358, 1995.

Crumpton, P. I.; Moinier, P.; Giles, M. B. T. J. *An Unstructured Algorithm for High Reynolds Number Flows on Highly Stretched Grids*. In: C. Taylor and J. T. Cross, 179 (eds.), Numerical Methods In Laminar and Turbulent Flow, Pineridge Press, p. 561-572, 1997.

Culler, D. and Singh, J. P. *Parallel Computer Architecture A Hardware / Software Approach.* Morgan Kaufmann Publishers, 1997.

Devloo, P. R. B. and Longhin, G. C. *Object Oriented Design Philosophy for Scientific Computing*. Mathematical Modelling and Numerical Analysis, Vol. 36, No 5, pp. 793–807, 2002.

Douglas JR. J. *Finite Difference Methods for Two-Phase Incompressible Flow in Porous Media*. Journal of Numerical Analysis (SIAM), v. 20, p. 681-697, 1983.

Durlofsky, L. J. A *Triangle Based Mixed Finite Element-Finite Volume Technique for Modeling Two Phase Flow Through Porous Media*. Journal of Computational Physics, v. 105, p. 252-266, 1993.

Eclipse Reservoir Engineering Software, http://www.slb.com/content/services/software/reseng/index.asp.

Edwards, M. G. *Cross Flow Tensors And Finite Volume Approximations with Deferred Correction*. Computer Methods in Applied Mechanics and Engineering, v. 151, p. 143-161, 1998.

Edwards, M. G.; Rogers C. F. *Finite Volume Discretization with Imposed Flux Continui ty for the General Tensor Pressure Equation*. Computational Geosciences 2, v. 2, p. 259-290, 1998.

Edwards, M. G. *M-Matrix Flux Splitting for General Full Tensor Discretization Operators on Structured and Unstructured Grids*. Journal of Computational Physics, v. 160, p. 1-28, 2000.

El-Rewini, H. and Abd-El-Barr, M., *Advanced Computer Architecture And Parallel Processing*, Wiley-Interscience, 2005.

Silva, E. O.; Silva, R. S.; Luna, B. G. B.; Carvalho, D. K. E.; Lyra, P. R. M. A*daptaçãoTi-po-h Aplicada à Simulação de Reservatórios de Petróleo usando Programação Orientada a Objetos em C++*. Congresso Íbero-Latino Americano de Métodos Computacionais em Engenharia (CILAMCE), 2008.

Ertekin, T.; Abou-Kassem, J. H.; King, G. R. In: *Basic Applied Reservoir Simulation*. Richardson, TX, Society of Petroleum Engineers (SPE), 2001.

Eymard, R. In: R.Herbin and D. Kröner, (eds.). *Finite Volume for Complex Applications III: Problems and Perspectives. Two-Phase Flow in Porous Media and Finite Volume Schemes*. Hermes Penton Science, p. 65-79, 2002.

Ewing, R. E. In: Ewing, R. E. (editor), The *Mathematics of Reservoir Simulatio*n. Philadelphia, SIAM, 1983.

Fanchi, J. R. In: *Principles of Applied Reservoir Simulation*. Boston, Gulf Professional Publishing, 2001.

Fortuna, A. O. In: *Técnicas Computacionais para Dinâmica dos Fluidos: Conceitos Básicos e Aplicações*. São Paulo, EDUSP, 2000.

Flynn, M. J. and Rudd, K. W. *Parallel Architectures.* ACM Computer Surveys, 1996.

Flynn, M. *Some computer organizations and their effectiveness.* IEEE Transactions on Computers, vol. C-21, no. 9, Sep. 1972, pp. 948-960.

Fujimoto, R. M., *Parallel and distributed simulation systems*. Wiley-Interscience Publicati on, 2000.

Fuhrmann, S. *Advanced Concepts in C++: Garbage Collection Techniques in C++ Smart Pointers for automated memory management*. Darmstadt University of Technol ogy, Germany, 2005

Garcia, E. L. M. *Formulações Bi e Tri Dimensionais do Método dos Elementos Finitos para a Simulação Paralela de Escoamentos em Reservatórios de Petróleo*. Tese de 180 Doutorado em Engenharia Civil - Universidade Federal do Rio de Janeiro (UFRJ), Rio de Janeiro-RJ, 1997.

Geiger, S.; Matthai, K.; Roberts S.; Zoppou, C. *Combining Finite Volume and Finite Element Methods to Simulate Fluid Flow in Geologic Media*. Australian & New Zealand Industrial and Applied Mathmatics Journal, v. 44, p. c180-c201, 2003.

Geiger, S.; Roberts S.; Matthai, K.; Zoppou, C.; Burri, A. *Combining Finite Element and Finite Volume for Efficient Multiphase Flow Simulations in Highly Heterogeneous and Structuraly Complex Geologic Media*. Geofluids, v. 4, p. 284- 299, 2004.

Gresho, P. M.; Sani, R. L. In: *Incompressible Flow and the Finite Element Method: Vo lume One, Advection-Diffusion*. New York, Wiley & Sons, 2000.

Guimarães, C. S. C.; Lima, R. C. F.; Lyra, P. R. M.; Carvalho, D. K. E. *Aplicação de Uma Ferramenta Computacional Utilizando o Método dos Volumes Finitos em Malhas Não-Estruturadas na Simulação do Tratamento por Hipertermia em Tumures de Duodeno*. 55ª Reunião Anual da Sociedade Brasileira para o Progresso da Ciência (SBPC, 2003), Recife, 2003.

Harten, A. *High Resolution Schemes for Conservation Laws*. Journal of Computational Physics v. 49, p. 357-393, 1983.

Heimsund, B. O., *Mathematical and Numerical Methods for Reservoir Fluid Flow Simulation*, Doctor Scientiarum thesis, Department of Mathematics, University of Bergen, 2005.

Helmig, R. In: *Multiphase Flow and Transport Processes in the Subsurface (A Contributi on to the Modeling of Hydrosystems*). Springer, Berlin, Germany, 1997.

Helmig, R.; Huber R. *Comparison of Galerkin-Type Discretization Techniques for Two-Phase Flow Problems in Heterogeneous Porous Media*. Advances in Water Resources, v. 21, p. 697-711, 1998.

Hirsch, C. In: *Numerical Computation of Internal and External Flows: Volume I, Fundamentals of Numerical Discretization*. New York, Wiley & Sons, 1988.

Hirsch, C. In: *Numerical Computation of Internal and External Flows: Volume II, Computational Methods for Inviscid and Viscous Flows*. New York, Wiley & Sons, 1990.

Hugues, T. In: *The Finite Element Method: Linear Static and Dynamic Element Analysis*. Dover, Mineola, New York, 2000.

Hughes, C.; Hughes, T. *Parallel and Distributed Programming Using C++*. Publisher: Addison Wesley, 2003

Hubbard, J. R. *Schaum's outline of theory and problems of programming with C++*. Schaum's outline series, 1996

Hurtado, F. S. V.; Maliska, C. R.; Silva, A. F. C.; Cordazzo, J.; Ambrus, J.; Contessi, B. A. *An Element-Based Finite Volume Formulation for Simulating Two-Phase Immiscible Displacements in Core Samples*. Proceedings of the 10th Brazilian Congress of Thermal Sciences and Engineering (ENCIT 2004), Rio de Janeiro, 2004.

Hyman, J.; Shashkov, M.; Steinberg, S. *The Numerical Solution of Diffusion Problems in Heterogeneous Non-Isotropic Materials*. Journal of Computational Physics, v. 132, p. 130-148, 1997.

IMEX, http://www.cmgroup.com/software/imex.htm

Jameson, A.; Schmidt, W.; Turkel, E. *Numerical Simulation of Euler Equations by Finite Volume Methods Unising Runge-Kutta Time Stepping Schemes*. Techinical Report, 81-1259, AIAA, Paper, 1981.

Jenny, P.; Lee, S. H.; Tchelepi, H. A. *Adaptive fully implicit multi-scale finite-volume method for multi-phase flow and transport in heterogeneous porous media.* Journal of Computational Physics, 217, p. 627-641, 2006.

Jostle- – graph partitioning software http://staffweb.cms.gre.ac.uk/~c.walshaw/jostle/

Josuttis, N. M. *The C++ Standart Library*: *A Tutorial and Reference*. Addison Wesley, 11ªed., 2003.

Kelley, C. T. *Iterative Methods for Linear and Nonlinear Equations.* Society for Industrial and applied mathematics (SIAM), 1995

Kovarik, K. In: Numerical Models in Grounwater Pollution. New York, Springer, 2000. Klausen R. A.; Eigestad, G. T. *Multi-Point Flux Approximations and Finite Element Methods: Practical Aspects of Discontinuous Media*. 9th European Conference on the Mathematics of Oil Recovery. Cannes, France, 2004.

Leveque, R. J. In: *Numerical Methods for Conservation Laws*. Berlin, Birkhauser, 1992.

Leveque, R. J. In: *Finite Volume Methods for Hyperbolic Problems*. Cambridge, Cabridge University Press, 2002.

Lewis R. W., Malan, A. G. *Continuum Thermodynamic Modeling of Drying Capillary Particulate Materials via an Edge-Based Algorithm*. Computer Methods in Applied Mechanics v. 194, p. 2043-2057, 2005.

Li, R.; Chen, Z.; Wu, W. In: *Generalized Finite Differences Methods for Differential Equations: Numerical Analysis of Finite Volume Methods*. New York, Marcel Dekker, 2000.

Li, B.; Chen, Z.; Huan, G. *Control Volume Function Approximation Methods and Their Applications to Modeling Porous Media Flow*. Advances in Water Resources, v. 26, p. 435-444, 2003a.

Li, B.; Chen, Z.; Huan, G. *The Sequential Method for Black Oil Reservoir Simulation on Unstructured Grids*. Journal of Computational Physics, v. 192, p. 36-72, 2003b.

Löhner, R. *Edges, stars, superedges and chains*. Computer Methods in Applied Mechaics and Engineering, v. 111, p. 255-263, 1994.

Löhner, R. In: *Applied CFD Techniques: An Introduction Based on Finite Element Methods*. New York, Wiley & Sons, 2001.

Loula, A. F. D.; Garcia, E. L. M.; Coutinho, A. L. G. A. *Miscible Displacement Simultion by Finite Element Methods in Distributed Memory Machines*. Computer Methods in Applied Mechanics and Engineering, v. 174, p. 339-354, 1999.

Lyra, P. R. M. *Unstructured Grid Adaptive Algorithms for Fluid Dynamics and Heat Conduction*. 332 f. Tese de Doutorado em Engenharia Civil, University of Wales, Swansea, 1994.

Lyra, P. R. M.; Morgan K. *A Review and Comparative Study of Upwind Biased Schemes-for Compressible Flow Computation. Part III: Multidimensional Extension on Unstructured Grids*. Arch. Comput. Meth. Engng, v. 9, p. 207-256, 2002.

Lyra, P. R. M.;, Lima, R. C. F. de, Guimarães, C. S. C., Carvalho, D. K. E. *An Edge-Based Unstructured Finite Volume Procedure for the Numerical Analysis of Heat Conduction* Applications. Journal of the Brazilian Society of Mechanical Engineering. Brazil, v., 26, p. 160-169, 2004a.

Lyra, P. R. M.; Lima, R. C. F.; ; Carvalho, D. K. E.; Silva, G. M. L. L. *An Axisymmetric Finite Volume Formulation for the Solution of Heat Conduction Problems Using Unstructured Meshes*. Journal of the Brazilian Society of Mechanical Science,.v. 27, p.1-18, 2005.

Luo, H.; Baum, J. D.; Löhner, R. *An Improved Finite Volume Scheme for Compressible Flows on Unstructured Grids*. Technical Report AIAA-95-0348, p. 1-9, 1995.

McCain, W. D. Jr. *The Properties of Petroleum Fluids.* PennWell Books, Tulsa, Oklohma, 2ª ed., 1989.

Malan, A. G. *Investigation into the Continuum Thermodynamic Modeling of Investment Casting Shell-Mould Drying*. Tese de Doutorado em Engenharia Civil, University of Wales, Swansea, 2002.

Maliska, C. R. In: *Transferência de Calor e Mecânica dos Fluidos Computacional*. Rio de Janeiro, Livros Técnicos e Científicos (LTC), 2004.

Malta, S. C.; Loula, A. F. D.; Garcia, E. L. M. *A Post-Processing Technique to Approximate the Velocity Field in Miscible Displacement Simulations*. Laboratório Nacional de Computação Científica, (LNCC) Relatórios de Pesquisa e Desenvolvimento, No. 24/93, 1993.

Martins, M. A. D. *Solução Iterativa em Paralelo de Sistemas de Equações do Método dos Elementos Finitos Empregando Estruturas de Dados por Arestas*. Dissertação de Mestrado em Engenharia Civil, Universidade Federal do Rio de Janeiro, (UFRJ), Rio de Janeiro - RJ, 1996.

Mattax, C. C.; Dalton, R. L. In: *Reservoir Simulation*. Richardson, TX, Society of Petroleum Engineers (SPE), 1990.

Morton, K. W., In: *Numerical Solution of Convection-Diffusion Problems*. London, Chapman & Hall, 1996.

Pacheco, P. S., *Parallel programming with MPI*. Morgan Kaufmann Publishers Inc, 1997.

ParMetis, Parallel Graph Partitioning and Fill-reducing Matrix Ordering, http://glaros.dtc.umn.edu/gkhome/metis/parmetis/overview.

Patankar, S. V., In: *Numerical Heat Transfer and Fluid Flow*. New York, Hemisphere Publishing Corporation, 1980.

Peaceman, D. W., In: *Fundamentals of Numerical Reservoir Simulation*. New York, Elsevier, 1977.

Peraire J., Peiro, J., Morgan, K *Finite Element Multigrid Solution of Euler Flows Past Installed Aero-Engines.* Journal of Computational Mechanics, v. 11, p. 433–451, 1993.

Petsc – Portable, Extensible Toolkit for Scientific Computation, http://www.unix.mcs.anl.gov/petsc/petsc-as/.

Pfister, G. F., In *Search of Clusters*, 2nd Edition, Prentice Hall, 1998.

Rosa, A. J.; Carvalho, R. de S.; Xavier, J. A. *Engenharia de Reservatórios de Petróleo.* Ed. Interciências, Rio de Janeiro, 2006.

Rosa, A. J.; Carvalho, R. de S. *Previsão de Comportamento de Reservatórios de Petróleo*. Ed. Interciências, Rio de Janeiro, 2002.

Reed, D. A. *Workshop on The Roadmap for the Revitalization of High-End Computing*. 2003.

Rees, I. *Development of an Edge-Based Finite Volume Solver for Porous Media Applcaions*. Tese de Doutorado em Engenharia Civil, University of Wales, Swansea, 2004.

Rees, I.; Masters, I.; Malan, A. G.; Lewis, R. W. *An Edge-Based Finite Volume Scheme for Saturated-Unsaturated Groundwater Flow*. Computer Methods in Applied Mechanics and Engineering, v. 193, p. 4741-4759, 2004.

Reichenberger, V., Jakobs, H.; Bastian, P.; Helmig, R. *A mixed-dimensional finite volume method for two-phase flow in fractured porous media.* Advances in Water Resources 29 p.1020-1036, 2006.

Reis, G. *Data Abstraction And Polymorphism in C++*. Centre de Mathématiques et de Leurs Applications, ACCU Spring Conference, France, 2002.

Roe P. L. *Approximate Rieman Solvers, Parameters Vectors and Difference Schemes*. Journal of Computational Physics, v. 43, p. 357-372, 1981.

Scotch – Software package and libraries for graph, mesh and hypergraph partitioning, static mapping, and parallel and sequential sparse matrix block ordering. http://www.labri.fr/perso/pelegrin/scotch/

Souza, D. A. F.; Coutinho, A. L. G. A.; Alves, J. L. D. *Three Dimensional Finite Element Solutions of Immiscible Two-Phase Flow Through Porous Media*. Congresso Íbero-Latino Americano de Métodos Computacionais em Engenharia (CILAMCE 2004), (Em CD-ROM), 2004.

Sterling, T. *Beowulf Cluster Computing with Linux*. The MIT Press Cambridge, Masschu setts, 2002.

Strohmaier, E., Dongarra, J., Meuer, H., Simon, H. *Recent Trends in the Marketplace of High Performance Computing.* CTWatch Quarterly, Volume 1, Number 1, February 2005.

Stroustrup, B. *The C+ + Programming Language.* Third Edition, AT&T Labs MurrayHill, New Jersey, 1997.

Stroustrup, B. *Why C++ is not just an Object-Oriented Programming Language.* AT&T Bell Laboratories, Murray Hill, New Jersey 07974.

Stroustrup, B. *Learning Standard C++ as a New Language.* The C++ Users Journal, 1999.

Stroustrup, B. *Abstraction and the C++ Machine Model.* Texas A&M University, ICESS'04, 2005.

Svärd M., Nordström, J. *A Stable and Accurate Summation by Parts Finite Volume Formu lation of the Laplacian Operator*. Uppsala Universitet, Institutionen för Informations teknologi, Technical Report 003, 2003.

Tai, C. H.; Zhao, Y.; Liew, K. M. *Parallel Computation of Unsteady Three-Dimensional Incompressible Viscous Flow using an Unstructured Multigrid Method.* Computers and Structures, 82, p. 2425-2436, 2004.

Tannehill, J. C.; Anderson, D. A.; Pletcher, H. R., In: *Computational Fluid Mechanics and Heat Transfer*. Philadelphia, Taylor & Francis, 1997.

Thiele, M. R.; Batycky, R. P.; Iding, M.; Blunt, M. *Extension of Streamline-Based Dual Porosity Flow Simulation to Realistic Geology*. 9th European Congress on the Mathematics of Oil Recovery, 2004.

Wang, H. F.; Anderson, M. P. In: *Introduction to Groundwater Modeling: Finite Differ ences and Finite Elements Methods*. New York, Academic Press, 1983.

Wheeler, M. F.; Russel, T. F. In: Ewing, R. E. (editor), *The Mathematics of Reservoir Simulation*. Philadelphia, SIAM, 1983.

Willmersdorf, R. B.; Lyra, P. R. M.; Carvalho, D. K. E. *Unstructured Control Volume Formulation for Solving Biphasic Flows in Porous Media*. 13th Finite Elements in Flow Problems 2005 (FEF2005), 2005, Swansea, Reino Unido. 2005.

Woll, C.; Correia, A. C.; Carvalho, D. K. E.; Willmersdorf, R. B.; Lyra, P. R. M. *An Object-Oriented Implementation of the Finite Volume Method with an Edge-Based Data Structure for a Parallel Computing Environment*. Congresso Ibero-Latino Americano em Métodos Computacionais em Engenharia (CILAMCE 2004),Recife. 2004.

Vargas, R. S. *Matrix Iterative Analysis*. Prentice-Hall, Englewood Cliffs, 1962.

Verma, S. K.; Aziz, K. *Three Dimensional Flexible Grids for Reservoir Simulation*. Fifth European Conference on the Mathematics of Oil Recovery, Leoben, Austria, p. 1-13, 1996.

Verma, S. K. *Flexible Grids for Reservoir Simulation*, Stanford University, Stanford, USA, 1996.

Yeo, C. S.; Buyya, R.; Pourreza H.; Eskicioglu, R.; Graham, P.; and Sommers, F. *Cluster Computing: High-Performance, High-Availability, and High-Throughput Processing on a Network of Computers, Handbook of Nature-Inspired and Innovative Computting.* pp.521-551 (Chapter 16), Springer Science+Business Media Inc., New York, USA, 2006.

Zhang, K.; Wu, Y. S.; Ding, C.; Pruess, K.; Elmroth, E. *Parallel Computing Techniques for Large-Scale Reservoir Simulation of Multicomponent and Multiphase Fluid Flow. SPE 66343, 2001.*

Zoltan – Parallel Partitioning, Load Balancing and Data-Management Services http://www.cs.sandia.gov/Zoltan/

Zienkiewicz, O. C.; Morgan K. In: *Finite Elements and Approximations*. New York, Wiley & Sons, 1983.

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo