UNIVERSIDADE FEDERAL DE JUIZ DE FORA
PROGRAMA DE PÓS-GRADUAÇÃO EM MODELAGEM COMPUTACIONAL

Carolina Ribeiro Xavier

Comparação de métodos de otimização para o problema de ajuste de histórico em ambientes paralelos

Juiz de Fora
2009

Carolina Ribeiro Xavier

Comparação de métodos de otimização para o problema de ajuste de histórico em ambientes paralelos

Dissertação apresentada ao Programa de Pós-graduação em Modelagem Computacional da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do grau de Mestre.

Orientador: Prof. Dr. Rodrigo Weber dos Santos

Juiz de Fora

2009

Xavier, Carolina Ribeiro.

Comparação de métodos de otimização para o problema de ajuste de histórico em ambientes paralelos / Carolina Ribeiro Xavier. -- 2009.
99 f. : il.

Dissertação (Mestrado em Modelagem Computacional)- Universidade Federal de Juiz de Fora, Juiz de Fora, 2009.

1. Reservatórios de petróleo. 2. Engenharia de petróleo. I. Título.

CDU 66.076

Carolina Ribeiro Xavier

Comparação de métodos de otimização para o problema de ajuste de histórico em ambientes paralelos

Dissertação apresentada ao Programa de Pós-graduação em Modelagem Computacional da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do grau de Mestre.

Aprovada por

BANCA EXAMINADORA

_____

Prof. Dr. Rodrigo Weber dos Santos – Orientador

Universidade Federal de Juiz de Fora

_____

Prof. Ph.D. Flávio Dickstein – Co-orientador

Universidade Federal de Juiz de Fora

_____

Prof. Dr. Marcelo Lobosco – Co-orientador

Universidade Federal de Juiz de Fora

_____

Prof. Ph.D. Paulo Goldfeld

Universidade Federal de Juiz de Fora

_____

Prof. Dr. Eduardo Lúcio Mendes Garcia

Universidade Federal de Juiz de Fora

Agradecimentos

Gostaria de agradecer primeiramente aos meus pais(Lucia e Guilherme), ao Ricardo e às minhas irmãs (Julia e Laura), pois eles sempre são muito presentes e importantes em todas as fases na minha vida: ensinando, guiando, apoiando, incentivando e me dando toda a assistência e amor.

Gostaria de agradecer muito ao meu namorado, amigo e colega de trabalho, Vinícius, pela paciência, ajuda e carinho nos momentos bons e difíceis dessa fase; agradeço também a família dele: Meire e Vítor, que fazem parte de minha família e do meu dia-a-dia.

Agradeço sinceramente aos amigos do FISIOCOMP, da velha e da nova (nem tão nova assim) geração: Fernando, Sachetto, Vinícius(de novo mesmo porque ele merece), Cadim, Caroline, Gustavo, Elisa, Ronan, pela ajuda e companheirismo sempre, fazendo do laboratório nossa segunda casa.

Agradeço a todos os meus amigos, em especial: Mariana, Fred, Poliana, João, Fábio, Salsicha, pelos momentos de diversão, pela amizade e torcida constante.

Gostaria de agradecer em especial aos meus amigos: Márcia, Beto, Rita e Bernardo, que são muito queridos e amigos exemplares.

Agradeço a todos os professores do curso de Ciência da Computação-UFJF e do Mestrado em Modelagem-UFJF pelo conhecimento adquirido.

Agradeço aos meus orientadores: Rodrigo, Marcelo e Flávio. Agradeço especialmente ao Rodrigo, pela paciência, atenção e dedicação a melhoria do meu trabalho.

Agradeço aos membros da banca pela oportunidade de mostrar o meu trabalho à uma banca com membros tão competentes.

Agradeço a Petrobras pelo suporte financeiro.

Resumo

O processo de ajuste histórico tem como objetivo a determinação dos parâmetros de modelos de reservatório de petróleo. Uma vez ajustados, os modelos podem ser utilizados para a previsão do comportamento do reservatório. Este trabalho apresenta uma comparação de diferentes métodos de otimização para a solução deste problema. Métodos baseados em derivadas são comparados com um algoritmo genético. Em particular, compara-se os métodos: Levenberg-Marquardt, Quasi-Newton, Gradiente Conjugado não linear, máxima descida e algoritmo genético. Devido à grande demanda computacional deste problema a computação paralela foi amplamente utilizada. As comparações entre os algoritmos de otimização foram realizadas em um ambiente de computação paralela heterogêneo e os resultados preliminares são apresentados e discutidos.

Palavras-chave: Método IMPES. Escoamento bifásico. Ajuste de Histórico. Computação paralela. Engenharia de Reservatórios.

Abstract

The process of history matching aims on the determination of the models' parameters from a petroleum reservoir. Once adjusted, the models can be used for the prediction of the reservoir behavior. This work presents a comparsion of different optimization methods for this problem's solution. Derivative based methods are compared to a genetic algorithm. In particular, the following methods are compared: Levenberg-Marquadt, Quasi-Newton, Non Linear Conjugate Gradient, steepest descent and genetic algorithm. Due to the great computational demand of this problem, the parallel computing has been widely used. The comparsions among the optimization algorithms were performed in an heterogeneous parallel computing environment and the preliminar results are presented and discussed.

Key-words: IMPES Method. Biphasic flow. History matching. Parallel computation. Reservoir Engineering.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A Engenharia de Reservatórios se ocupa da retirada de óleo do interior das rochas, de modo que ele possa ser conduzidos até à superfície. São estudadas na Engenharia de Reservatórios a caracterização das jazidas, as propriedades das rochas, as propriedades dos fluidos nelas contidos, a maneira como estes fluidos interagem dentro da rocha e as leis físicas que regem o movimento dos fluidos no seu interior, com o objetivo de maximizar a produção de hidrocarbonetos com o menor custo possível.

Nos estudos de um reservatório de petróleo é fundamental o conhecimento de propriedades básicas da rocha e dos fluidos nela contidos. São essas propriedades que determinam a quantidade de fluido existente no meio poroso, a sua distribuição a habilidade do meio fazer escoar o fluido [1].

Em pequena escala, as propriedades de uma rocha podem ser obtidas através de experimentos em laboratório. Porém, na realidade, os reservatórios apresentam grandes extensões, tornando impossível a obtenção de suas propriedades globalmente. O estudo das propriedades através de amostragem também é inviável, devido à grande heterogeneidade do reservatório. Por isso, a crescente demanda pelo conhecimento das propriedades de reservatórios impulsionou o desenvolvimento de técnicas alternativas para a obtenção dessas informações.

A simulação de reservatório é uma das mais poderosas técnicas atualmente disponíveis para a engenharia de reservatórios. Basicamente, um simulador de reservatório tem como objetivo reproduzir *in-silico* o reservatório e as leis físicas que regem o escoamento de fluidos, por exemplo, água e óleo. Dessa forma, um simulador de reservatório possibilita que diferentes técnicas de recuperação de petróleo

sejam avaliadas a um custo baixo [2]. Porém, novamente, para que o simulador possa reproduzir o objeto de estudo de forma fiel, as propriedades físicas do reservatório devem ser conhecidas.

Um processo muito conhecido na área de engenharia de reservatório que auxilia na estimativa dessas propriedades do reservatório é o denominado ajuste de histórico. Em um reservatório em operação há alguns anos, informações sobre fluxos e pressões nos poços são conhecidas. Este conjunto de dados é chamado de histórico do reservatório. O ajuste de histórico tem como principal objetivo calibrar modelos numéricos de reservatórios de petróleo, a fim de que os resultados simulados sejam coerentes com o histórico de produção existente, para que os modelos possam ser usados na previsão de produção com a confiabilidade desejada. Durante o processo de ajuste de histórico, alguns parâmetros do modelo de reservatório são alterados, visando reproduzir numericamente da melhor forma possível os dados do histórico do reservatório. Tipicamente, um modelo de reservatórios depende de diversos parâmetros físicos. Os parâmetros usuais que são determinados por um processo de ajuste histórico são, os campos de porosidade ϕ e de permeabilidade absoluta K.

1.1 Trabalho proposto

Neste trabalho, foi implementado um simulador de reservatórios bifásico (água/óleo) e bidimensional. Este simulador foi utilizado para comparação de diversos métodos de otimização para a solução de um problema de ajuste de histórico que visa a determinação do mapa de permeabilidades do reservatório. Mais especificamente, deseja-se determinar o campo de permeabilidade do reservatório K, dado que é conhecida a produção de óleo $\bar{O}$ nos poços produtores do reservatório. Esta estimativa de K será realizada através da resolução de um problema de otimização.

Neste trabalho, implementamos o método de busca e otimização, conhecido como algoritmo genético para resolver este problema de ajuste histórico e comparamos o seu desempenho com algumas implementações de métodos clássicos baseados em derivadas, como o Levenberg-Marquardt, Quasi-Newton, Gradientes Conjugados não linear e Máxima Descida, ou Método do Gradiente. O algoritmo genético é uma

método estocástico que implementa uma heurística baseada no fenômeno de evolução das espécies e apresenta como algumas vantagens: a) não necessitar da informação de derivadas; e b) evitar alguns mínimos locais. Essas características motivaram o seu uso neste trabalho, visto que o cálculo da derivada em nosso problema de otimização é uma operação custosa, além de ser bem conhecido que o problema de ajuste de histórico possui múltiplos mínimos, locais e globais. Diversos testes foram realizados e tiveram como objetivo comparar os algoritmos de otimização quanto ao tempo necessário para convergência, melhor ajuste encontrado, média e desvio padrão dos melhores ajustes.

Tanto o simulador quanto os métodos de otimização utilizados neste trabalho são computacionalmente custosos. Para reduzir os tempos de execução de ambos foram implementadas versões paralelas dos algoritmos. A avaliação destas implementações paralelas foi realizada em um *cluster* de computadores multiprocessados.

Na última década, muitos trabalhos analisaram algoritmos científicos paralelos e plataformas de alto desempenho baseadas em cluster de computadores. Atualmente, novas tecnologias e arquiteturas estão surgindo no cenário de computação de alto-desempenho, como as FPGAs [3], *grids* computacionais [4], processadores multicores homogêneos e heterogêneos (como o *Cell processor*) e *GPUs*. Para cada uma dessas tecnologias, podemos encontrar na literatura exemplos de sucesso, isto é, aplicações que tiveram seus tempos de execução acelerados. Porém, nem toda aplicação tem sua execução acelerada quando implementada em uma dessas novas tecnologias. Cada uma dessas novas arquiteturas atende melhor a um conjunto específico de aplicações que compartilham certas características. Quando uma aplicação não possui certos pré-requisitos, sua execução pode ser até prejudicada. Devido à complexidade das aplicações computacionais de hoje e à complexidade das novas arquiteturas de alto-desempenho em estágio ainda recente de desenvolvimento, o mapeamento ótimo entre as classes de aplicação e as novas tecnologias ainda não é claro. Neste trabalho, avaliamos como algumas aplicações de modelagem computacional de reservatórios de petróleo se comportam em um cluster de computadores multiprocessados, isto é, com múltiplos núcleos de processamento (*multicores*).

1.2 Organização do texto

O texto desta dissertação possui a seguinte organização: O Capítulo 2 apresenta os conceitos necessários para o entendimento da modelagem em meios porosos, além de mostrar o modelo utilizado para a implementação do simulador. O Capítulo 3 apresenta o problema de ajuste ao histórico que se deseja resolver neste trabalho. O Capítulo 4 descreve os método utilizados neste trabalho e mostra, em linhas gerais, como eles funcionam. O Capítulo 5 introduz alguns conceitos de computação paralela e explica de forma mais detalhada os algoritmos genéticos paralelos. O Capítulo 6 detalha cada implementação feita neste trabalho e os esquemas numéricos utilizados. O Capítulo 7 apresenta a metodologia utilizada para validar os experimentos. O Capítulo 8 apresenta os resultados, que são discutidos na Capítulo 9. Por fim o Capítulo 10 conclui este trabalho e discute possíveis trabalhos futuros. Finalmente, incluimos um apêndice descrevendo o esquema numérico usado na resolução do escoamento bifásico.

Capítulo 2

Conceitos de meios porosos e Modelo utilizado

Neste capítulo, serão apresentados alguns dos conceitos mais importantes para o entendimento de um modelo de escoamento em meios porosos, como é o caso do modelo implementado e utilizado neste trabalho.

Após a introdução dos conceitos necessários, será mostrado o conjunto de equações que forma o modelo utilizado ao longo deste trabalho.

2.1 Conceitos fundamentais em meios porosos

Um meio poroso é um sólido que possui espaços vazios (poros), como na Figura 2.1, sejam estes conectados ou não, distribuídos uniformemente ou aleatoriamente. Esses poros podem conter vários fluidos, como ar, água e óleo. Se os poros representam uma boa parte do volume total, pode ser formado um caminho que seja capaz de transportar fluidos [5].

Alguns exemplos de meios porosos listados em [5] são:

- Na Ciência do Solo:

  O meio poroso (solo) contém e transporta água e nutrientes para as plantas.

- Na Engenharia de Petróleo:

  O meio poroso é um reservatório rochoso de óleo bruto e gás natural.

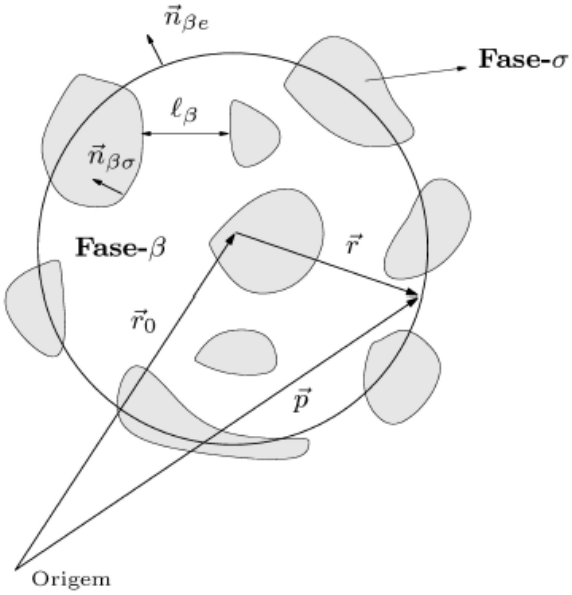

Figura 2.1: Representação do meio poroso

2.1.1 Porosidade

A porosidade de um meio poroso é definida como a porção do volume que os poros ocupam do volume total do sólido. Então,

$$V_t = V_s + V_p \tag{2.1}$$

onde:

- V_t é o volume total
- V_p é o volume dos poros
- V_s é o volume do material sólido

A porosidade ϕ pode ser expressa por:

$$\phi = \frac{V_p}{V_t} = \frac{V_t - V_s}{V_t} \tag{2.2}$$

Podemos distinguir dois tipos de porosidade:

- Porosidade total, que considera todo espaço vazio, incluindo os poros isolados, como acabamos de definir.

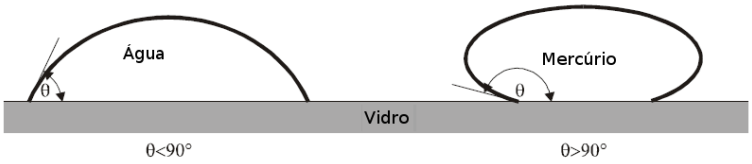

Figura 2.2: Comparação entre fluidos molhantes e não molhantes

- Porosidade efetiva, que considera apenas a parte porosa capaz de transportar fluidos.

A segunda é muito importante para se conhecer a capacidade de escoamento de um reservatório. Sua determinação é bastante complexa. Para uma amostra é possível fazê-la por meio de um experimento em laboratório. Em um reservatório isto é inviável, pois o mesmo não está acessível.

2.1.2 Propriedades Capilares

Saturação

O espaço poroso pode conter várias fases. A saturação de uma certa fase é definida por:

$$S_i = \frac{\text{Volume da fase } i \text{ no meio poroso}}{\text{Volume Poroso Efetivo do meio}} \tag{2.3}$$

Então a soma das saturações das fases resulta em:

$$\sum S_i = 1 \tag{2.4}$$

Molhabilidade

Molhabilidade é a capacidade de um fluido, na presença de outro, de aderir preferencialmente a superfície do sólido. Esta propriedade pode ser determinada pelo ângulo de contato entre o fluido e o sólido.

Um fluido sobre uma superfície plana pode ter vários formatos, que dependerá de sua molhabilidade em relação a superfície. A Figura 2.2 ilustra a propriedade. No caso da água e do ar, a água é a fase molhante. Para o ar e o mercúrio, o ar é a fase molhante.

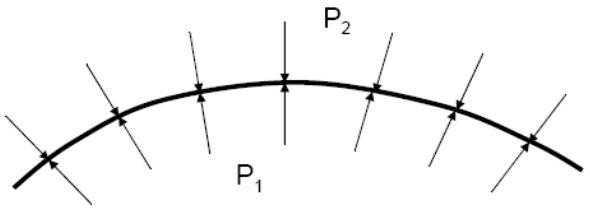

Figura 2.3: Representação da pressão capilar

O ângulo de contato é usado como medida de molhabilidade. No caso do fluido molhante, o ângulo de contato é menor do que 90°, caso contrário, o fluido é considerado não molhante.

Pressão Capilar

Para quaisquer dois fluidos imiscíveis (como água é óleo), a pressão exercida por cada um deles não é igual. Existe uma diferença entre essas pressões na interface entre os fluidos. Esta diferença de pressão é chamada pressão capilar, representada na Figura 2.3, que varia com a saturação:

$$P_c(S_a) = P_o - P_a \tag{2.5}$$

onde:

- P_c é a pressão capilar
- P_o e P_a são as pressões do óleo e da água respectivamente

Os efeitos e propriedades da pressão capilar podem ser obtidos com mais detalhes em [6].

2.1.3 Permeabilidade

A permeabilidade absoluta K é a medida da capacidade de um material (tipicamente uma rocha) para transmitir fluídos. Esta propriedade é de grande importância na determinação das características de fluxo dos hidrocarbonetos em reservatórios de petróleo e gás. A permeabilidade é usada para calcular taxas de fluxo através da lei de Darcy[7].

Viscosidade

Para se entender a lei de Darcy, que será descrita a seguir, é necessário descrever a viscosidade, uma importante propriedade de um fluido.

Viscosidade é a propriedade associada à resistência que o fluido oferece à deformação por cisalhamento. De outra maneira, pode-se dizer que a viscosidade corresponde ao atrito interno nos fluidos devido basicamente a interações intermoleculares, sendo em geral função da temperatura.

É comumente percebida como a "espessura", ou resistência ao despejamento. A viscosidade descreve a resistência interna para fluir de um fluido, e deve ser pensada como a medida do atrito do fluido. Assim, a água é "fina", tendo uma baixa viscosidade, enquanto óleo é "espesso"ou "grosso", tendo uma alta viscosidade.

Lei de Darcy

A teoria do escoamento laminar e lento através de um meio poroso homogêneo é baseada num experimento clássico originalmente desenvolvido por Henry Darcy em 1856. Um desenho esquemático do experimento é mostrado na Figura 2.4. Um filtro homogêneo de altura h é limitado por seções planas de mesma área superficial A. O filtro é preenchido com um líquido incompressível. Manômetros abertos são colocados para se medir a pressão nos pontos inferior e superior do filtro, fornecendo as alturas h_1 e h_2, respectivamente. Pela variação das várias quantidades envolvidas, Darcy deduziu que a vazão q é proporcional à $\frac{S}{L} \cdot (h_2 - h_1)$, onde S é a seção transversal do pacote de areia; L é o tamanho do pacote de areia; h_1 e h_2 as alturas indicadas na Figura 2.4

Outra forma de escrever a lei de Darcy pode ser vista na Equação 2.6. Mais detalhes podem ser obtidos em [8].

$$v = -\frac{k_f}{\mu}\nabla p \tag{2.6}$$

onde k_f é a permeabilidade relativa do meio e μ a viscosidade do fluido e ∇p é o gradiente de pressão.

Uma unidade de permeabilidade usual é o Darcy (D) ou, mais habitualmente, o mili-darcy ou md ($1d = 10^{-12}m^2$). Um meio poroso tem permeabilidade de $1D$ se

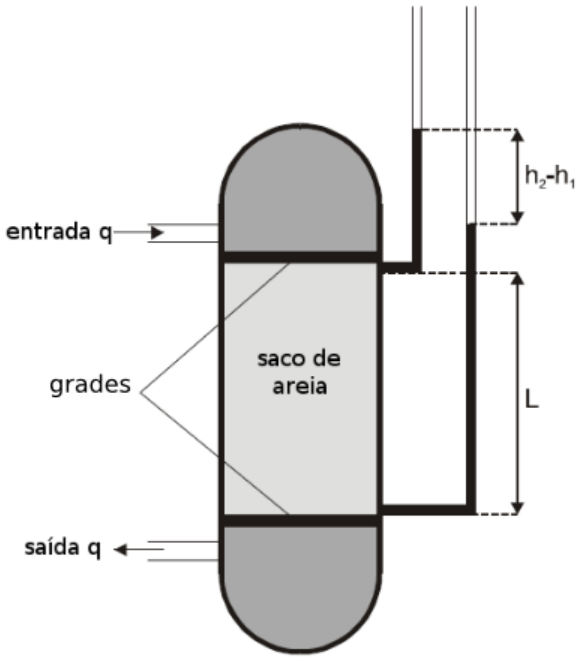

Figura 2.4: Esquema do experimento de filtragem de Darcy

em $1cm^2$ da secção transversal um fluido de viscosidade $1cP$ atravessa o meio com velocidade de de $1cm^3/s$ causando a queda de pressão de $1atm/cm$:

$$k = \frac{\mu}{A} / \frac{\delta P}{L} = \frac{[cP][cm^3/s]}{cm^2} / \frac{atm}{cm} = 1\ Darcy \tag{2.7}$$

Usando SI:

$$k = \frac{[Nsm^{-2}][m^3 s^{-1}]}{[m^2]} / \frac{[Nm^{-2}]}{[m]} = [m^2] \tag{2.8}$$

Fatores que interferem na permeabilidade

Existem alguns fatores que interferem diretamente no valor da permeabilidade de um meio:

- Tamanho, arranjo e forma dos grãos.
- Estado do solo.
- Grau de saturação
- Estrutura;

- Anisotropia;
- Densidade, viscosidade e temperatura do fluido que o atravessa.

2.1.4 Permeabilidade Relativa

Para definir o que é permeabilidade relativa (k_{rf}) é necessário definir antes o que é permeabilidade efetiva (k_f, onde f é o fluido em questão). A permeabilidade efetiva está relacionada não somente à interação entre o fluido e a rocha, mas também ao impacto relativo que a presença de um fluido causa ao escoamento de outro.

A permeabilidade relativa nada mais é do que a permeabilidade efetiva normalizada. Considerando dois fluidos, água e óleo, por exemplo, obtém-se que:

$$k_{ra} = \frac{k_a}{K} \quad e \quad k_{ro} = \frac{k_o}{K}, \ onde \ 0 <= k_{ra}, k_{ro} <= 1 \tag{2.9}$$

Onde k_a e k_{ra} são os valores de permeabilidade efetiva e relativa da água respectivamente; k_o e k_{ro} do óleo; e K é um valor característico de permeabilidade, em geral a permeabilidade absoluta.

As permeabilidades relativas são funções das saturações como mostra (2.10) e ilustra a Figura 2.5.

$$k_{ra} = k_{ra}(S_a) \quad e \quad k_{ro}(S_o) = k_{ro}(S_o) \tag{2.10}$$

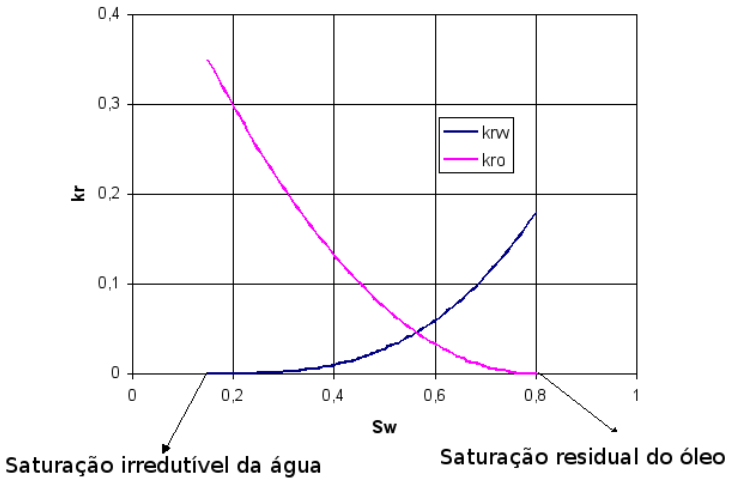

Figura 2.5: Curvas de permeabilidade

A Figura 2.5 apresenta dois novos conceitos, o primeiro é a saturação irredutível da água e o outro é a saturação residual do óleo. Quando a saturação de água diminuir a um valor tal em que ela pára de fluir, sua saturação é chamada de saturação irredutível de água (S_{ai}), e conseqüentemente, suas permeabilidades efetiva e relativa são nulas. Da memsa forma, quando a saturação de óleo vai decrescendo até atingir a chamada saturação de óleo residual (S_{or}) o óleo deixar de fluir.

Em geral as permeabilidades relativas dos fluidos são dadas por curvas parametrizadas que aproximam os resultados empíricos mostrados nas curvas da Figura 2.5.

Mobilidade

Define-se mobilidade de um fluido como sendo o produto de sua permeabilidade efetiva e sua viscosidade. Por exemplo, a mobilidade do óleo é dada por $\lambda_o = k_o\mu_o$ e a da água $\lambda_a = k_a\mu_a$. Assim como as permeabilidades efetivas, as mobilidades também dependem da saturação.

2.1.5 Equação de balanço de massa

Além das propriedades intrínsecas às rochas e aos fluidos, uma relação muito importante para o entendimento da modelagem em meios porosos é a equação de balanço de massa, também denominada lei de balanço de massa.

O princípio básico desta lei é: "em um volume de controle fixo no espaço, a variação de massa em um intervalo de tempo é igual à massa que entrou menos a que saiu através das fronteiras e por ação de fontes ou sumidouros", ou seja,

$$\text{Acumulação} = \text{Entrada - Saída} \tag{2.11}$$

Para o caso do escoamento bifásico incompressível água e óleo, em um meio poroso, a Equação (2.11) toma a seguinte forma:

$$\phi\rho\partial_t s = -\nabla.(\rho v) + Q \tag{2.12}$$

onde o lado esquerdo da equação representa o termo de acumulação de uma das fases: ρ, ϕ, s são respectivamente, a densidade, a porosidade e a saturação de um

dos fluidos, água ou óleo. O lado direito representa os termos de entrada e saída: Q representa injeção ou recuperação do fluido em um poço; o $\nabla.(\rho v)$ representa a diferença entre os fluxos de entrada e saída do fluido, onde v é a velocidade do fluido.

2.2 Modelo utilizado

O modelo utilizado neste trabalho tem como objetivo resolver um problema de escoamento bifásico água e óleo, incompressível e bidimensional. Este é obtido por um sistema de equações diferenciais parciais (EDPs) não lineares.

No escoamento bifásico, as fases em geral, possuem velocidades distintas. Dessa forma, usamos a lei de Darcy para cada fase, o que gera as duas primeiras equações do sistema, que estão representadas por (2.13) e (2.14).

$$v_a = -\frac{K_a}{\mu_a}\nabla p_a, \tag{2.13}$$

$$v_o = -\frac{K_o}{\mu_o}\nabla p_o, \tag{2.14}$$

onde os índices a e o referem-se a água e óleo, respectivamente, e:

- p é a pressão;
- K é a permeabilidade efetiva;
- μ é a viscosidade;

A saturação s é definida para cada fluido e as permeabilidades são escritas como $K_o = Kk_{ro}$, $K_a = Kk_{ra}$, onde K é a permeabilidade absoluta e k_{ro}, k_{ra} são as permeabilidades relativas. São introduzidas as mobilidades:

$$\lambda_o = \frac{k_{ro}}{\mu_o}, \;\; \lambda_a = \frac{k_{ra}}{\mu_a}, \tag{2.15}$$

e as transmissibilidades:

$$T_o = K\lambda_o, \;\; T_a = K\lambda_a. \tag{2.16}$$

Desta forma, $v_o = -T_o \nabla p_o$, $v_a = -T_a \nabla p_a$. As equações de conservação de massa das duas fases são escritas na forma:

$$\begin{cases} \phi \rho_a \partial_t s_a + \nabla.(\rho_a v_a) = Q_a, \\ \phi \rho_o \partial_t s_o + \nabla.(\rho_o v_o) = Q_o, \end{cases} \tag{2.17}$$

onde:

- ϕ é a porosidade do meio;
- ρ é a densidade;
- Q é a vazão mássica nos poços.

A seguir, admite-se que as densidades do óleo e da água são constantes, e que as pressões do óleo e da água são iguais, o que significa que a pressão capilar é nula.

Dividindo a primeira equação (2.17) por ρ_a, a segunda por ρ_o , somando-se as duas e usando que $s_a + s_o = 1$, obtém-se a Equação 2.18.

$$\begin{cases} \phi \partial_t s_a + \nabla.v_a = q_a, \\ \nabla.v_t = q_t, \end{cases} \tag{2.18}$$

onde:

- $v_t = -T_t \nabla p$ é a velocidade total;
- $T_t = T_a + T_o$ é a transmissibilidade total;
- $q_a = \dfrac{Q_a}{\rho_a}$ é a vazão volumétrica de água;
- $q_o = \dfrac{Q_o}{\rho_o}$ é a vazão volumétrica de óleo;
- $q_t = q_a + q_o$ é a vazão volumétrica total;

Escrevendo s para a saturação da água, omitindo subscritos e definindo a mobilidade total $\lambda_t = \lambda_a + \lambda_o$, será introduzido agora o fluxo fracionário

$$f(s) = \frac{T_a}{T_t} = \frac{\lambda_a}{\lambda_t}. \tag{2.19}$$

A Equação (2.18) pode ser reescrita da forma:

$$\begin{cases} \phi \partial_t s - \nabla.(f(s)v_t) = q_a, \\ \nabla.v_t = q_t, \end{cases} \tag{2.20}$$

As permeabilidades relativas $k_{ra}(s)$, $k_{ro}(s)$ serão definidas por curvas como a apresentada na Figura 2.5.

Cabe resaltar que as funções $\phi(x, y)$ e $K(x, y)$ variam espacialmente.

Neste trabalho, vamos resolver o sistema (2.20) assumindo que as vazões totais nos poços, q_t são conhecidas. Além disso, $q_a = q_t$ nos poços injetores e $q_a = f(s)q_t$ nos poços produtores.

Para se obter o sistema completo, ainda é necessário definir a condições iniciais e de contorno.

A condição inicial do reservatório é determinada por um campo de saturação prescrito no reservatório.

Para a definição da condição de contorno, assume-se que o reservatório Ω está isolado, o que significa que v_a, v_o são nulos nos bordos de Ω. Ou seja, não há troca de fluidos entre o bordo do reservatório e o meio externo.

A troca de fluidos entre o reservatório e o exterior é feita somente através dos poços. Além disso, as vazões de entrada e saída de fluidos devem se anular, respeitando a lei de balanço de massa para o sistema.

2.2.1 Exemplo

A seguir, apresentamos um exemplo de simulação numérica do modelo descrito. Este experimento foi realizado com uma configuração de poços típica da engenharia de reservatório, denominada 5-spot, com um poço vertical de um tipo (produtor, por exemplo) rodeado de 4 poços verticais do outro tipo (injetor) distribuídos no reservatório, como apresentado na Figura 2.6.

Para este experimento, foi utilizada uma injeção de $q_t = q_a = 100m^3/dia$ de água em cada poço injetor e a recuperação de $q_t = 400m^3/dia$ pelo poço produtor central (água + óleo). A saturação inicial de água no reservatório é homogênea e igual a 20%, o valor de sua saturação irredutível. A permeabilidade absoluta K é igual

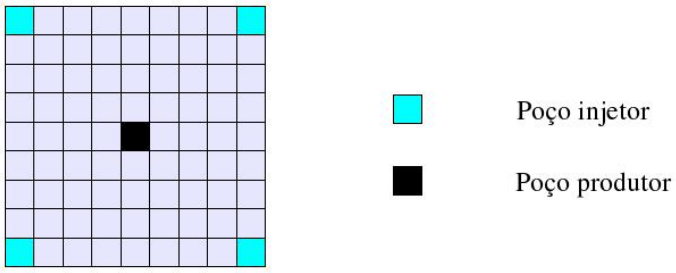

Figura 2.6: Modelo 5-spot

a $100md$. O reservatório tem dimensões 200X200mX20m e a simulação executada correspondeu a 600 dias de produção.

Os níveis de saturação ao longo do tempo são apresentados pelas figuras a seguir. A Figura 2.7 apresenta o campo de saturação depois de 5 dias. As figuras seguintes, para tempos iguais a 100, 300 e 600 dias, respectivamente.

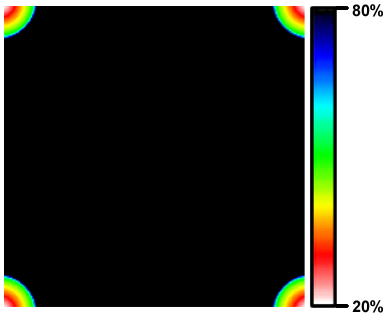

Figura 2.7: Campo de saturação de óleo para o tempo igual a 5 dias

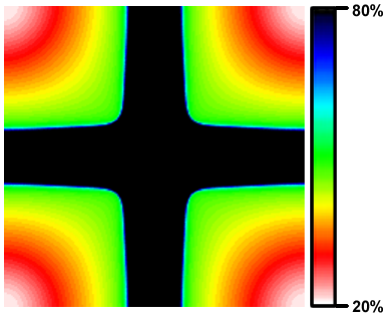

Figura 2.8: Campo de saturação de óleo para o tempo igual a 100 dias

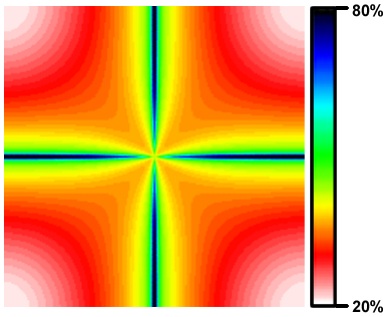

Figura 2.9: Campo de saturação de óleo para o tempo igual a 300 dias

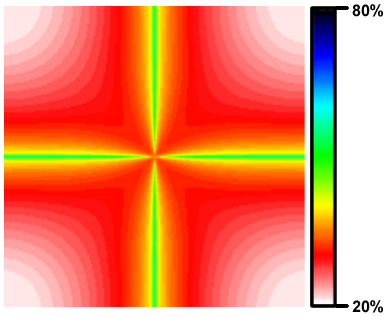

Figura 2.10: Campo de saturação de óleo para o tempo igual a 600 dias

A Figura 2.11 apresenta a vazão de óleo ao longo do tempo.

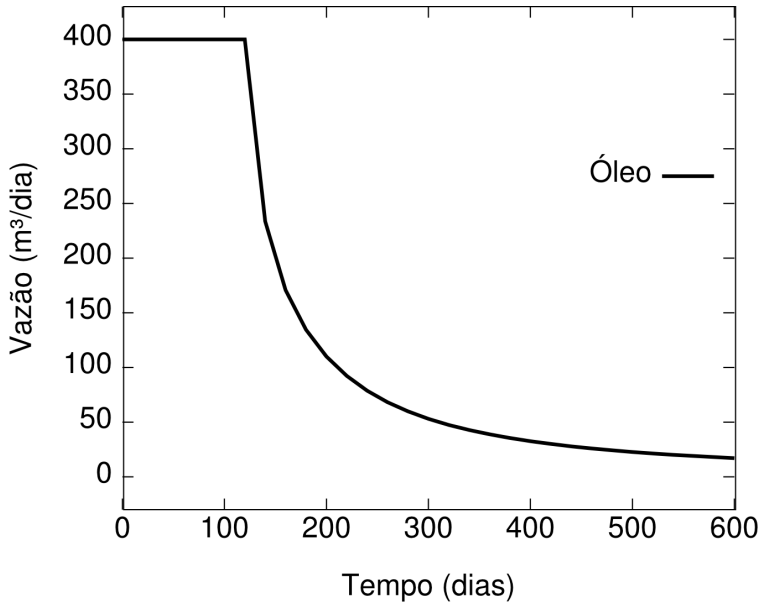

Figura 2.11: Curva de vazão de óleo ao longo do tempo.

Capítulo 3

O Problema Inverso - Ajuste de histórico

Nesta seção, apresentamos a problema de ajuste de histórico que será tratado nesta dissertação. Este problema equivale à solução de um problema inverso: conhecemos resultados do sistema e desejamos estimar características desse sistema. A distinção entre o que seja um problema direto e inverso para um dado fenômeno está relacionada à interpretação do que são causas e efeitos deste fenômeno [9]. É atribuído a Oleg Mikhailovitch Alifanov, proeminente pesquisador russo na área de problemas inversos, a afirmação:

A solução de um problema inverso consiste em determinar causas através da observação de seus efeitos

Seja S um sistema que depende de um conjunto de parâmetros m e tem como resultado final O. Então, o problema dito direto corresponde à obtenção do resultado O, para um dado m conhecido, através de $O = S(m)$. Já no problema inverso temos O e queremos obter o conjunto de parâmetros m tal que $S(m) = O$.

Neste trabalho, o problema inverso em questão consiste no ajuste de um modelo de reservatório aos chamados dados de histórico. Em um reservatório em operação há alguns anos, informações sobre fluxos e pressões nos poços são conhecidas. Este conjunto de dados é chamado o histórico do reservatório. O ajuste de histórico tem como principal objetivo calibrar modelos numéricos de reservatórios de petróleo, a fim de que os resultados simulados sejam coerentes com o histórico de produção. Isto

permite que os modelos possam ser usados na previsão de produção com a confiabilidade desejada. Durante o processo de ajuste de histórico, alguns parâmetros do modelo de reservatório são alterados visando reproduzir numericamente da melhor forma possível os dados do histórico do reservatório. Portanto, o ajuste de histórico é basicamente um processo de estimativa de parâmetros ou de minimização, onde se procura minimizar as discrepâncias entre os resultados do modelo e os dados observados.

Como apresentamos na seção anterior, tipicamente um modelo de reservatórios depende de diversos parâmetros físicos. Alguns parâmetros típicos que podem ser determinados por um processo de ajuste de histórico são, por exemplo, os campos de porosidade ϕ e de permeabilidade absoluta K.

Neste trabalho, deseja-se determinar o campo de permeabilidade do reservatório K, dado que é conhecida a produção de óleo $\bar{O}$ nos poços produtores do reservatório. Esta estimativa de K será realizada através da resolução de um problema de mínimos quadrados. Como K varia em diversas ordens de grandeza, transformamos um problema de otimização com restrição em um problema sem restrição fazendo $m = logK$. O problema de mínimos quadrados aqui proposto é dado por:

$$\min_{m \in \mathbb{R}^{N_m}} f(m) = \|O(m) - \bar{O}\|^2. \tag{3.1}$$

Onde f é chamada função objetivo, $m \in \mathbb{R}^{N_m}$ é o vetor de parâmetros a determinar; $\bar{O} \in \mathbb{R}^{N_o}$ é o conjunto de observações fornecido pelo histórico; $O(m) = O(m, u(m))$ é o histórico obtido pela resolução do problema direto dado pelo sistema (2.20), ou seja $u(m) = (s, p)$, com $u \in \mathbb{R}^{N_u}$.

No caso de ajuste de histórico para a determinação de parâmetros do reservatório, pode-se utilizar vários métodos de otimização [7]. Na próximo capítulo apresentamos alguns esquemas e métodos usados nesta dissertação. Antes, porém, ilustramos alguns exemplos de funções objetivos deste problema de ajuste de histórico.

3.1 Função Objetivo

Para se conhecer melhor o problema, foram testadas algumas configurações de poços tipo 5-spot onde o campo de permeabilidades possui apenas dois valores distintos.

O esquema 5-spot usado e o campo de permeabilidade estão ilustrados na Figura 3.1.

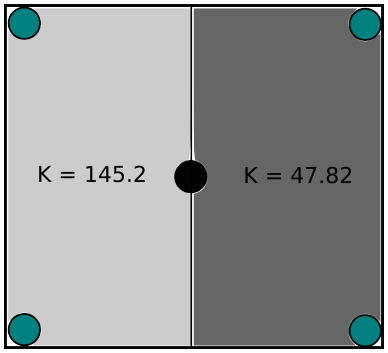

Figura 3.1: 5-spot com dois valores de permeabilidade

Este modelo de reservatório foi simulado para diferentes valores de vazões q_t prescritas nos poços. Cada uma das simulações gerou como resultado vazões de óleo $\bar{O}$ que foram usadas como dados de histórico sintéticos. Vamos considerar a função objetivo como descrita em 3.1:

$$f(m) = \|O(m) - \bar{O}\|^2, \tag{3.2}$$

onde aqui e no restante desta dissertação $O(m)$ representa o resultado de produção de óleo de uma simulação que possui os mesmos parâmetros e configurações da simulação utilizada para gerar a vazão de óleo sintética $\bar{O}$, com exceção dos valores do campo de permeabilidades usado, $K_i = e^{m_i}$. Observe que assumimos que a distribuição espacial é também conhecida. Para o exemplo em questão existem dois blocos de tamanhos iguais. Dessa forma, o que queremos estimar são os valores de permeabilidade de cada bloco.

A seguir, calculamos $f(m)$ para diferentes valores de m, onde cada cálculo envolve a resolução numérica do simulador de reservatório. A Figura 3.2 apresenta a função objetivo para um histórico sintético $\bar{O}$ gerado com 1 injetor no centro ($qt = 400m^3/dia$) e 4 produtores com valores de vazões totais diferentes em cada poço ($qt = -10, -150, -190, -50m^3/dia$).

A Figura 3.3 apresenta a função objetivo para um histórico sintético $\bar{O}$ gerado com 1 produtor no centro ($qt = -400m^3/dia$) e 4 injetores com valores de vazões totais iguais em cada poço injetor. Podemos observar a simetria para este caso,

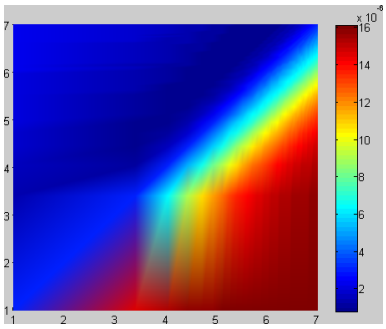

Figura 3.2: Função objetivo para 5-spot vazões distintas

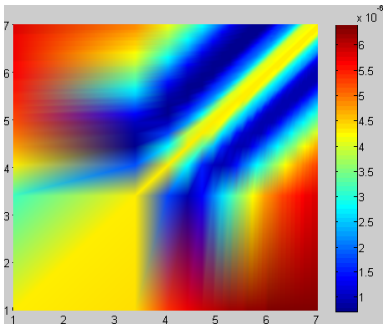

Figura 3.3: Função objetivo para 5-spot vazões iguais

afinal a vazão de óleo é igual para $m = (m_1, m_2)$ e (m_2, m_1).

Uma característica importante do problema de ajuste de histórico é a não-unicidade de soluções, isto é, dada uma resposta, geralmente é possível encontrar diferentes parâmetros que produzem a mesma resposta.[7]. Nota-se pelas Figuras 3.3 e 3.2 que o problema que este trabalho propõe solucionar possui várias soluções. Toda parte em azul escuro do mapa de cores possui valores de $f(x)$ bem pequenos. Esta característica torna o problema ainda mais interessante, pois tem-se vários campos de permeabilidades diferentes que geram o mesmo conjunto de vazões.

Capítulo 4

Métodos de Otimização

Neste capítulo, serão descritos alguns métodos de otimização utilizados neste trabalho.

Um desses métodos é o algoritmo genético, que será descrito de forma mais detalhada, pois foi implementado para este trabalho. Os demais método foram utilizados através de bibliotecas livres, ou seja, foram utilizados como caixa preta. Estes métodos são: gradiente, gradientes conjugados, BFGS(Quasi-Newton) e Levenberg-Marquardt.

4.1 Algoritmos Genéticos

Nesta seção, serão descritos os conceitos e a origem dos algoritmos genéticos, assim como suas técnicas, operações e utilidades no contexto deste trabalho.

Na natureza, os indivíduos competem entre si por recursos como: comida, água e refúgio. Adicionalmente, entre os animais de uma mesma espécie, aqueles que não obtêm êxito tendem a ter um número reduzido de descendentes. Consequentemente, existe uma menor probabilidade de seus genes serem propagados ao longo de sucessivas gerações. De acordo com Charles Darwin:

"*Quanto melhor um indivíduo se adaptar ao seu meio ambiente, melhor será sua chance de sobreviver e gerar descendentes*".

Os Algoritmos Genéticos utilizam uma analogia direta deste fenômeno de evolução, onde cada indivíduo representa uma possível solução para um problema, ou seja, o vetor de parâmetros a ser estimado. Os principais conceitos que serão

abordados nesta seção estão brevemente descritos a seguir:

- Cromossomo - conjunto de parâmetros que descrevem um indivíduo;
- Gene - representação de cada parâmetro de acordo com o alfabeto utilizado (binário, inteiro ou real);
- População - conjunto de indivíduos que formam uma geração;
- Geração - iteração completa do algoritmo genético que determina os indivíduos que irão compor a nova população;
- Aptidão - saída gerada pela função de avaliação para um indivíduo da população.

Para cada indivíduo, é atribuída uma pontuação referente à sua aptidão, que será dado por uma função de avaliação: a função objetivo do problema de otimização. Aos mais adaptados, é dada a oportunidade de reproduzir-se mediante cruzamentos com outros indivíduos da população, produzindo descendentes com características de ambas as partes.

Os algoritmos genéticos diferem-se dos métodos tradicionais de busca e otimização principalmente em três aspectos:

- Trabalham com uma população e não com um único ponto;
- Utilizam informações de custo ou recompensa e não derivadas ou outro conhecimento auxiliar;
- Utilizam regras de transição probabilísticas e não determinísticas.

4.1.1 Esquemas de Reprodução

Existem na literatura vários esquemas de reprodução usados em AG, os dois mais comuns, descritos a seguir, são conhecidos como geracional e *Steady-state*.

Esquema de Reprodução Geracional

Neste esquema de reprodução a população é substituída completamente ao final de cada geração. A desvantagem desse método é a possibilidade de perda de bons indivíduos. Para evitar que isso aconteça, o elitismo pode ser utilizado, ou seja, manter os melhores indivíduos na próxima geração.

Esquema de Reprodução *Steady-state*

Nesse esquema somente um indivíduo é gerado a cada vez. Cada novo indivíduo criado é avaliado e inserido na população de acordo com uma "política de inserção", que pode ser, por exemplo: inserir o novo indivíduo no lugar do pior indivíduo da população; ou inserir o indivíduo na população se sua aptidão for maior que a média das aptidões de toda a população.

4.1.2 Conceitos em AG

Representação do indivíduo

Os indivíduos são análogos a cromossomos e podem ter diversas representações. São sobre esses cromossomos que são realizadas as operações genéticas. A escolha de uma representação é muito importante para o sucesso de um algoritmo genético. Todo cromossomo representa um indivíduo candidato a solução do problema. As representações mais utilizadas são a binária e a real. Neste trabalho, usamos a representação real.

Seleção

Na seleção, os indivíduos mais aptos da geração atual são selecionados. Esses indivíduos são utilizados para gerar uma nova população por cruzamento ou mutação. Cada indivíduo tem uma probabilidade de ser selecionado proporcional à sua aptidão.

Neste trabalho, foi utilizado um método de seleção conhecido como "roleta". Neste os indivíduos que possuem melhores aptidões possuem regiões maiores na "roleta"e, portanto, terão maior probabilidade de serem selecionados para gerar a população seguinte. Mais detalhes podem ser vistos em [10].

Cruzamento

Os indivíduos selecionados na etapa anterior são cruzados da seguinte forma: a lista de indivíduos selecionados é embaralhada, criando-se, desta forma, uma segunda lista, chamada lista de parceiros. Cada indivíduo selecionado é então cruzado com o indivíduo que ocupa a mesma posição na lista de parceiros. Um novo cromossomo é gerado, fazendo-se uma operação denominada *blend* (média ponderada generalizada) entre os dois correspondentes.

Mutação

A operação de mutação é utilizada para garantir uma maior varredura do espaço de busca e evitar a convergência a mínimos locais. A mutação é efetuada alterando-se o valor de um indivíduo sorteado aleatoriamente com uma determinada probabilidade, denominada probabilidade de mutação. Desta forma, vários indivíduos da nova população podem ser alterados de forma aleatória.

Esquema do algoritmo genético

Um algoritmo genético geracional, em linhas gerais, funciona da seguinte forma:

```
1  inicio
2      Inicializa a população;
3      Avalia os indivíduos da população;
4      enquanto critério de parada não satisfeito faça
5          Seleciona indivíduos para reprodução;
6          Aplica o operador de cruzamento;
7          Aplica o operador de mutação;
8          Avalia os indivíduos da população;
9          Seleciona os indivíduos para sobreviver (Elitismo);
10     fim enquanto
11 fim
```

Algoritmo 1: Ciclo Básico de um Algoritmo Genético

Apesar de aparentemente simples, estes algoritmos são capazes de resolver problemas complexos de uma maneira muito elegante.

4.2 Algoritmos Baseados em Derivadas

Neste trabalho os algoritmos de otimização baseados em derivadas não foram implementados, mas usados como caixa preta, através de bibliotecas disponíveis. Nesta seção, esses algoritmos serão brevemente descritos.

Em geral, esses algoritmos precisam de um ponto inicial e estabelecem um critério para a escolha de uma direção de busca, um método de busca de mínimo nesta direção (busca linear) e um critério de parada.

A principal diferença entre os algoritmos baseados em derivadas são a forma como as direções são determinadas. Uma vez determinada a direção, os algoritmos procedem da mesma maneira, buscando o ponto mínimo na direção correspondente.

4.2.1 Método da Máxima Descida ou Método do Gradiente

O método da máxima descida, proposto por Cauchy em 1847, é um dos mais fundamentais métodos para minimização de funções diferenciáveis de várias variáveis. Este método é conhecido também como método do gradiente, como será designado neste trabalho, ou método de Cauchy [11]. Para um dado ponto x_i a direção de descida é dada por $d_i = -\nabla f(x_i)$, ou seja, a direção de máxima descida.

```
1  inicio
2      Escolhe x_0;
3      se ∇f(x_0) == 0 então
4          x̄ = x_0;
5          pare;
6      fim
7      enquanto critério de parada não satisfeito faça
8          d_i = −∇f(x_i);
9          Calcula o escalar t_i através de uma busca linear;
10         x_{i+1} = x_i + t_i d_i;
11         i = i + 1;
12     fim enquanto
13 fim
```

Algoritmo 2: Ciclo Básico do Método do Gradiente

O método do gradiente gerou vários outros métodos, e por isso é um dos mais importantes métodos de otimização.

4.2.2 Método de Newton

O método de Newton é um método clássico que busca a direção de descida através de uma aproximação quadrática [11]. Em um problema de minimização, queremos encontrar x que satisfaz $f'(x) = 0$. Partindo de um ponto x_0, podemos usar a aproximação $f'(x_0) + f''(x_0)(x - x_0) \approx f'(x) = 0$. Assim, em uma dimensão as iterações do método de Newton são da forma $x_{i+1} = x_i - f'(x_i)/f''(x_i)$. A generalização para várias dimensões fica da forma: $x_{i+1} = x_i - H^{-1}(x_i)\nabla f(x_i)$, onde H é a matriz Hessiana de f.

```
1  inicio
2      Escolhe x_0 se ∇f(x_0) == 0 então
3          x̄ = x_0;
4          pare;
5      fim
6      enquanto critério de parada não satisfeito faça
7          d_i = H(x_i)^{-1} ∇f(x_i);
8          Calcula o escalar t_i através de uma busca linear;
9          x_{i+1} = x_i + t_i d_i;
10         i = i + 1;
11     fim enquanto
12 fim
```

Algoritmo 3: Ciclo Básico do Método de Newton

A necessidade do cálculo da inversa da Hessiana traz algumas restrições ao método de Newton. O procedimento de inversão da matriz pode ser numericamente instável e muito custoso computacionalmente. Por isso, surgiram muitos algoritmos que usam aproximações de H ou de H^{-1}, como os métodos Quasi-Newton.

4.2.3 Método BFGS

Este método é classificado como um método Quasi-Newton. A idéia por trás dos métodos Quasi-Newton é fazer uma aproximação iterativa da inversa da matriz Hessiana. Para o caso de uma dimensão, $f''(x_{i+1})$ pode ser aproximado por $(f'(x_{i+1}) - f'(x_i))/(x_{i+1} - x_i)$. Ou seja, podemos nos aproximar do método de Newton sem precisar calcular $f''(x)$. Da mesma forma, em dimensões maiores, B é uma aproximação de H se

$$B(x_{i+1} - x_i) = \nabla f(x_{i+1}) - \nabla f(x_i). \tag{4.1}$$

A Equação 4.1 não é suficiente para definir B unicamente. Porém, o método BFGS, assim como outros métodos Quasi-Newton, calcula uma aproximação B_{i+1} que satisfaz 4.1 e utiliza apenas as informações de $\nabla f(x_{i+1})$, $\nabla f(x_i)$, x_{i+1}, x_i e B_i.

```
1  inicio
2      Escolhe x_0;
3      B_0 = I;
4      g_0 = ∇f(x_0);
5      se ∇f(x_0) == 0 então
6          x̄ = x_0;
7          pare;
8      fim
9      enquanto critério de parada não satisfeito faça
10         d_i = B(x_i)^{-1} ∇f(x_i);
11         Calcula o escalar t_i através de uma busca linear;
12         x_{i+1} = x_i + t_i d_i;
13         g_{i+1} = ∇f(x_{i+1});
14         Calcula B(x_{i+1})^{-1} usando x_{i+1}, x_i, g_{i+1}, g_i e B(x_i)^{-1};
15         i = i + 1;
16     fim enquanto
17 fim
```

Algoritmo 4: Ciclo Básico do BFGS

4.2.4 Método Levenberg-Marquardt

O método de Levenberg-Marquardt é um método específico para problemas de mínimos quadrados. Vamos considerar o problema de mínimos quadrados unidimensional: $\min_x \frac{1}{2}f(x)^2 = \frac{1}{2}(y(x) - b)^2$. Aproximamos $y(x)$ por $y(x_0) + y'(x_0)(x - x_0)$. Como procuramos um mínimo, derivamos $\frac{1}{2}(y(x_0) + y'(x_0)(x - x_0) - b)^2$ e igualamos a zero para obter: $y'(x_0)(y(x_0) + y'(x_0)(x - x_0) - b) = 0$. A generalização para um problema multidimensional de mínimos quadrados nos leva a: $J^T(x_0)(y(x_0) + J(x_0)(x - x_0) - b) = 0$, onde J é o jacobiano de y dado por $J_{i,j} = \partial_{x_j} y_i$. Assim obtemos a seguinte iteração para o método denominado Gauss-Newton: $J^T(x_i)J(x_i)(x_{i+1} - x_i) = J^T(x_i)f(x_i)$. É possível provar que este método é uma aproximação do método de Newton para o problema de mínimos quadrados, pois $J^T J$ é uma aproximação de H onde os termos que envolvem segundas derivadas são desprezados e $J^T f$ é o gradiente da função que queremos minimizar.

Porém, como no método de Newton, o cálculo da inversa de $J^T J$ pode ser numericamente instável e a direção calculada d_i pode não ser de descida. Para solucionar este problema perturbamos a aproximação da Hessiana da forma: $J^T J + \lambda D$, onde D é uma matriz diagonal, $D = Diag(J^T J)$ e λ é um escalar que deve ser escolhido a cada passo. Se o algoritmo está convergindo, diminuímos λ a cada iteração e o método se comporta com o de Gauss-Newton. Caso o algoritmo não esteja convergindo, aumentamos λ e o método passa a se comportar como o método do

gradiente [12].

```
1  inicio
2      Escolhe x_0 Escolhe λ_0 se  ∇f(x_0) == 0 então
3          x̄ = x_0;
4          pare;
5      fim
6      enquanto critério de parada não satisfeito faça
7          d_i = (J(x_i)J(x_i)^T + λD)^{-1} J^T(x_i) f(x_i);
8          x_{i+1} = x_i + d_i;
9          Calcula λ_{i+1};
10         i = i + 1;
11     fim enquanto
12 fim
```

Algoritmo 5: Ciclo Básico do Método Levenberg-Marquardt

4.2.5 Método de Gradientes Conjugados Não Linear

O método de gradientes conjugados foi proposto inicialmente por Hestenes e Stiefel em 1952 [13], com o objetivo de obter resoluções de problemas quadráticos sem restrições, mas logo foi estendido para casos mais gerais e para problemas de otimização. Os métodos que utilizam a direção conjugada podem ser considerados métodos intermediários entre o método do gradiente e o método de Newton. Eles são motivados pelo desejo de acelerar a convergência lenta típica do método do gradiente e, ao mesmo tempo, evita a exigência de informações associadas com a avaliação, armazenamento e inversão da Hessiana, cálculos exigidos pelo método de Newton. O método começa seguindo a direção do gradiente: $d_0 = -\nabla f(x_0)$. As próximas iterações são da forma: $d_i = -\nabla f(x_i) + \beta_{i-1} d_{i-1}$, onde o escalar β é escolhido de forma a minimizar esta direção no caso de f ser uma função quadrática. Para o caso de problemas não lineares, diferente do caso linear, existem diferentes alternativas para o cálculo de β. Neste trabalho, β foi calculado seguindo o algoritmo de

Fletcher-Reeves.

inicio
  Escolhe x_0 **se** $\nabla f(x_0) == 0$ **então**
    $\bar{x} = x_0$;
    pare;
  fim
  $d_0 = -\nabla f(x_0)$;
  Obtém t_0 através de uma busca linear;
  $x_1 = x_0 + t_0 d_0$;
  $i = 1$ **enquanto** *critério de parada não satisfeito* **faça**
    $d_i = -\nabla f(x_i) + \frac{\nabla f(x_i)^T(\nabla f(x_i) - \nabla f(x_{i-1}))}{\|\nabla f(x_{i-1}\|^2} d_{i-1}$;
    Obtém t_i através de uma busca linear;
    $x_i + 1 = x_i + t_i d_i$;
    $i = i + 1$;
  fim enquanto
fim

Algoritmo 6: Ciclo Básico do Método dos Gradientes Conjugados

Capítulo 5

Computação Paralela

A principal meta ao usar uma abordagem paralela para um algoritmo é melhorar o tempo de computação, reduzindo o tempo de processamento global. Contudo, a forma de implementar o paralelismo pode variar e, por vezes, depende das soluções em *hardware* disponíveis.

Neste capitulo, abordaremos as principais arquiteturas para computação paralela, bem como os principais mecanismos para desenvolvimento de aplicações paralelas.

5.1 Computação Paralela

5.1.1 Classificação de Flynn

Tomando como base as características de uma arquitetura paralela, é possível classificá-la, utilizando-se diferentes enfoques. A classificação de Flynn, proposta na década de 60, é a mais conhecida, embora apresente alguma dificuldade em abranger todas as arquiteturas atuais. Essa classificação será descrita a seguir.

SISD - *Single Instruction stream/Single Data stream*

(Fluxo único de instruções, fluxo único de dados)

Esta organização representa a maioria dos computadores sequenciais disponíveis atualmente. As instruções são executadas sequencialmente, mas pode existir uma defasagem de estados na execução (*pipeline*). Alguns sistemas monoprocessadores

SISD são estruturados em *pipeline*. A Figura 5.1 apresenta um esquema deste tipo de arquitetura.

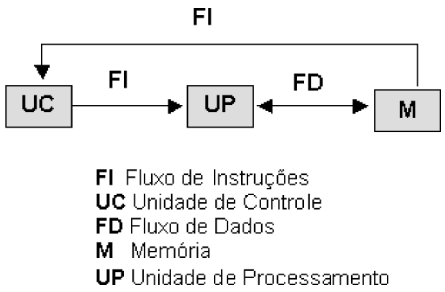

Figura 5.1: Arquitetura SISD

SIMD - *Single Instruction stream/Multiple Data stream*

(Fluxo único de instruções, fluxo múltiplo de dados)

Nesta classe, existem vários elementos de processamento supervisionados por uma unidade de controle. Os processadores recebem as mesmas instruções de uma unidade de controle, mas operam em diferentes conjuntos de dados. Os processadores vetoriais estão normalmente incluídos nesta classificação. A Figura 5.2 apresenta um esquema deste tipo de arquitetura.

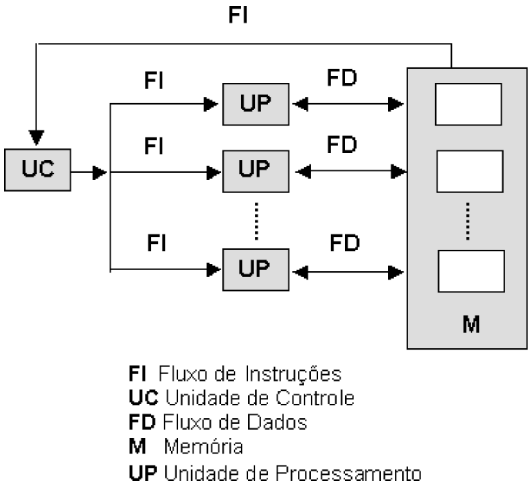

Figura 5.2: Arquitetura SIMD

MISD - *Multiple Instruction stream/Single Data stream*

(Fluxo múltiplo de instruções, fluxo único de dados)

Existem n unidades de processamento, cada uma recebendo diferentes instruções sobre um mesmo conjunto de dados e seus derivados. Alguns autores consideram esta classificação pouco significativa, chegando mesmo a não considerá-la [14]. Em alguns casos, os "*macropipelines*"[15] são encaixados nesta categoria uma vez que as saídas de uma unidade de processamento servem de entrada para outra unidade de processamento, como se pode ver na Figura 5.3.

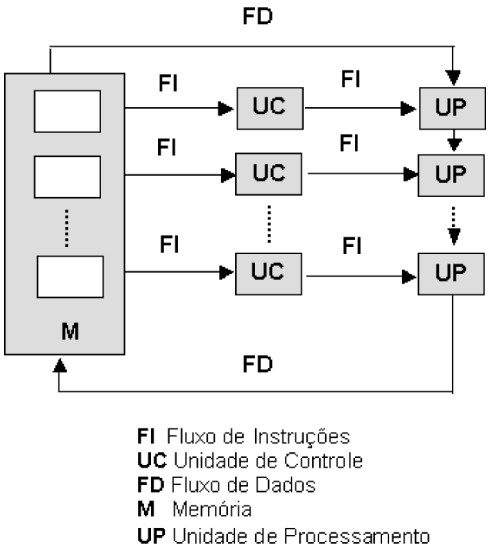

Figura 5.3: Arquitetura MISD

MIMD - *Multiple Instruction stream/Multiple Data stream*

(Fluxo múltiplo de instruções, fluxo múltiplo de dados)

A maioria dos sistemas multiprocessadores estão incluídos nesta categoria. Nesta classe, cada processador é controlado por uma unidade de controle, executando instruções independentemente sobre diferentes fluxos de dados. A Figura 5.4 mostra um esquema deste tipo de arquitetura. Esta arquitetura apresenta uma grande flexibilidade para desenvolvimento de algoritmos paralelos.

5.1.2 Níveis de Paralelismo

O paralelismo nas aplicações pode ser explorado em diferentes níveis, dependendo do problema trabalhado e dos recursos de *hardware* disponíveis. De uma forma geral, a exploração de paralelismo em um algoritmo pode ser feita através do particionamento do problema em respeito a dados e tarefas [16].

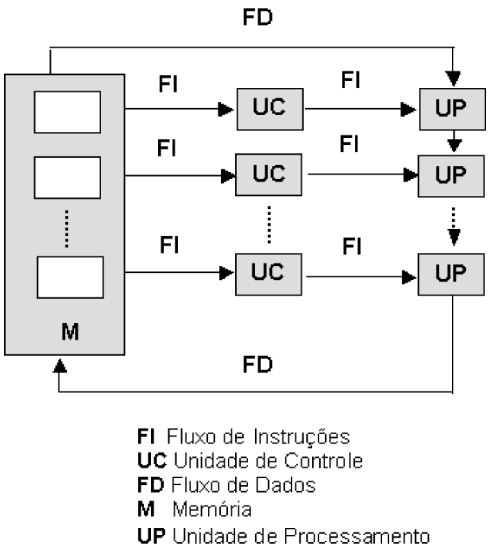

Figura 5.4: Arquitetura MIMD

- Paralelismo de dados: Uma mesma tarefa é alocada aos diversos processadores, sendo que cada um deles trabalha com uma porção de dados diferente do problema. Esta abordagem aplica-se a problemas cuja solução envolve grandes massas de dados. Alguns problemas clássicos desse tipo são a solução de sistemas de equações, a multiplicação de matrizes, a integração numérica e os problemas de autovalores.

- Paralelismo de tarefas: Tarefas distintas são alocadas aos diversos processadores (ou a grupos deles). Esta abordagem aplica-se a problemas onde pode haver uma independência entre blocos de execução, como é o caso dos algoritmos genéticos e do problema do produtor-consumidor.

5.2 Modelos de Programação Paralela

Para a construção de um programa paralelo, deve-se pensar no modo de comunicação entre os processadores envolvidos na sua execução. Existem, basicamente, dois paradigmas de comunicação: memória compartilhada e troca de mensagens.

5.2.1 Memória Compartilhada

O modelo de programação paralela utilizando memória compartilhada baseia-se no mesmo modelo utilizado na programação sequencial. Assim, essencialmente, um

único espaço de endereçamento é utilizado, e esse pertencente ao fluxo de execução do programa. Deste modo, o compartilhamento de memória é um paradigma mais natural e de mais fácil assimilação para programadores não habituados à programação paralela, já que a noção de memória compartilhada faz parte da cultura dos desenvolvedores de aplicações sequenciais [17].

A utilização de memória compartilhada adequa-se a computadores onde uma memória é acessada por vários processadores. Dessa forma, a sincronização entre as tarefas é feita por leitura/escrita na memória compartilhada. Normalmente uma mesma parte da memória não pode ser modificada por um processo enquanto outro a estiver acessando. Isto leva a necessidade de rotinas de sincronização e de exclusão mútua. Mais detalhes sobre esses conceitos podem ser obtidos em [18].

Uma grande vantagem na utilização de memória compartilhada em multiprocessadores é a alta velocidade de comunicação entre as tarefas. Entretanto, a escalabilidade de aplicações desenvolvidas com memória compartilhada é limitada pelo número de barramentos disponíveis entre os processadores e a memória.

No caso de arquiteturas com memória fisicamente distribuída, a programação por compartilhamento de memória exige uma camada de software que forneça a abstração de memória compartilhada. Essa abordagem é denominada memória compartilhada distribuída, ou seja, a memória é fisicamente distribuída, mas logicamente compartilhada.

5.2.2 Troca de Mensagens

Em um programa paralelizado via troca de mensagens deve-se distribuir os dados e/ou tarefas explicitamente entre os processadores. Como os espaços de endereçamento dos processos que compõem o programa paralelo são distintos, utiliza-se a abstração de mensagem, que pode ser enviada de um processo a outro por um canal de comunicação [17].

O paradigma de troca de mensagens tem sido tradicionalmente empregado em sistemas fracamente acoplados, representados pelas arquiteturas baseadas em memória distribuída, em que cada processador tem acesso somente à sua memória local. Por isso, a comunicação, necessariamente, dá-se por meio de uma rede.

A comunicação em aplicações que utilizam troca de mensagens é feita através

de primitivas que informam que um processo pretende enviar uma mensagem a outro, que espera recebê-la. Na prática, existem bibliotecas que fornecem, entre outras funções mais complexas, esse tipo de primitivas de envio e recebimento de mensagens. A maioria dessas bibliotecas implementa o padrão *Message Passing Interface* (MPI), que foi utilizado nesse trabalho e é descrita na próxima seção.

MPI

As bibliotecas que utilizam o padrão MPI foram desenvolvidas para ambientes de memória distribuída, máquinas paralelas, rede de estações de trabalho e redes heterogêneas. MPI define um conjunto de rotinas para facilitar a comunicação (troca de dados e sincronização) entre processos paralelos. As bibliotecas são portáveis para quaisquer arquiteturas e oferecem inúmeras funções para os programadores. Além do mais, também são disponibilizadas ferramentas para análise do desempenho das aplicações [19].

As bibliotecas que implementam o padrão MPI possuem rotinas para programas em Fortran 77 e ANSI C, portanto podem ser utilizadas em Fortran 90 e C++. Os programas são compilados e ligados à essas bibliotecas. Todo o paralelismo é explícito, ou seja, o programador é responsável por identificar o paralelismo e implementar o algoritmo utilizando chamadas às rotinas da biblioteca MPI.

Cada processo, identificado pelo seu IP ou pelo nome da máquina na rede, recebe uma réplica do código de todo o programa paralelo. O trecho de código particular de cada processo (se houver) geralmente é identificado por um comando de teste, onde a condição avaliada envolve o identificador do processo.

5.3 *Clusters* de Computadores Multiprocessados

Com a crescente disponibilidade das redes de alta velocidade e a constante redução no custo dos computadores pessoais, a utilização de grupos de computadores distintos para a execução de uma única aplicação tem sido popularizada e tem recebido bastante atenção nas últimas décadas [20]. O poder de processamento de computadores interligados pode então ser combinado de forma a realizar a computação de diferentes trechos de uma aplicação simultaneamente. A união de um grupo de

computadores conectados através de uma rede local com o objetivo de facilitar a execução paralela de aplicações é denominada *cluster* de computadores.

Ao desenvolver uma aplicação paralela deve-se ter como preocupação a maneira com que as informações serão trocadas entre os processadores. Em um *cluster*, a excessiva transferência de informações entre os processadores pode fazer com que o tempo de envio dos dados reduza significativamente o ganho obtido com a paralelização da aplicação. Isso ocorre devido a limitações no meio físico das redes atualmente disponíveis no mercado, que faz com que o tempo de transferência de um dado de um processador a outro seja muito alto quando comparado ao tempo de acesso a um dado na memória principal de um computador. Além disso, a CPU demanda um certo tempo para a execução de uma comunicação, tanto na entrada de dados quanto na saída. Há ainda a possibilidade de contenção na via de comunicação, ou seja, se algum dado já estiver sendo trafegado, outro dado deve esperar até que a via seja liberada.

Recentemente, tem havido uma forte tendência por parte das empresas de *hardware* na difusão de máquinas pessoais multiprocessadas, ou seja, máquinas que apresentam mais de um núcleo de processamento. Tais máquinas permitem que aplicações paralelas sejam construídas de forma que sua execução não demande transferência de dados entre nós distintos através de uma rede, diminuindo assim a latência ocasionada por essa transferência.

As Figuras 5.5 e 5.6 representam esquematicamente o caminho dos dados da CPU até a memória principal em duas das possíveis arquiteturas multiprocessadas. Na Figura 5.5, cada núcleo de processamento possui sua própria unidade de memória de nível L2, enquanto na Figura 5.6, os núcleos de processamento compartilham a mesma unidade de memória de nível L2.

Em aplicações que utilizam o modelo de paralelismo por compartilhamento de memória, a necessidade de leitura e escrita em memória para acesso aos dados pode fazer com que o número de barramentos entre as unidades de processamento e a memória imponha uma limitação no ganho de desempenho através do paralelismo.

Em aplicações que utilizam o modelo de paralelismo por troca de mensagens, mesmo em computadores multiprocessados, o envio de mensagens ocorre através de trocas explícitas de mensagens entre os processadores. Ou seja, caso um processador

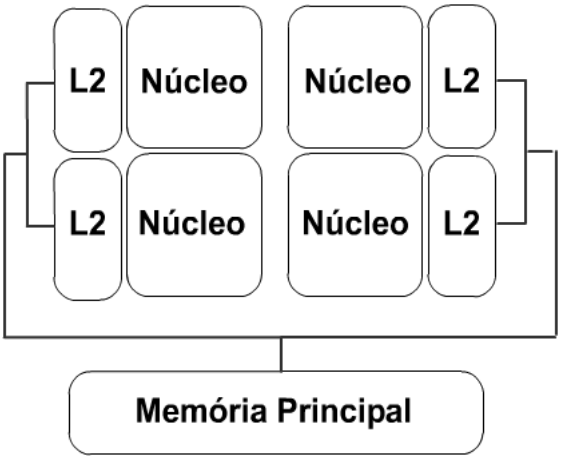

Figura 5.5: Representação de um processador com quatro núcleos e memória L2 privada.

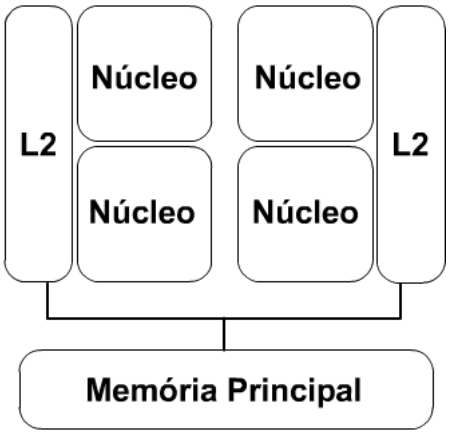

Figura 5.6: Representação de um processador com quatro núcleos e memória L2 compartilhada.

necessite enviar uma mensagem a um outro processador, esta será escrita em um *buffer* de saída do processador de origem. A mensagem será então copiada para um *buffer* de entrada do processador destino que só então fará a leitura desta mensagem. Assim, a contenção imposta pela utilização simultânea de um único barramento por duas unidades de processamento pode ser também um fator limitante no ganho de desempenho em uma aplicação paralela.

De maneira análoga aos *clusters* de computadores monoprocessados pode-se construir *clusters* de computadores multiprocessados. Assim, cada nó do *cluster* será um computador multiprocessado e os diversos nós estarão conectados através de uma rede local. A Figura 5.7 apresenta uma representação desse esquema.

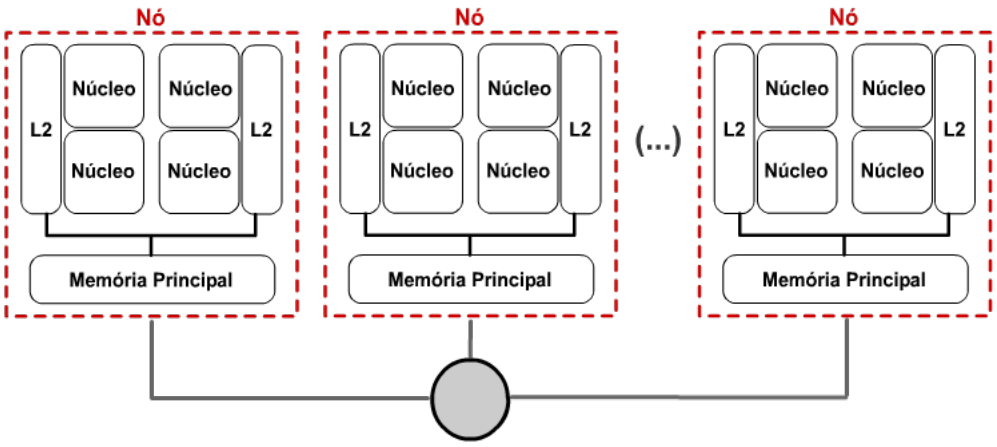

Figura 5.7: Representação de um *cluster* de computadores multiprocessados.

Em *clusters* de computadores multiprocessados a troca de mensagens não é feita

da mesma forma para todos os processadores. Caso as duas unidades de processamento envolvidas na comunicação estejam em nós distintos, a mensagem é copiada para um *buffer* de saída do processador de origem, trafega pela rede e então é copiada para o *buffer* de entrada do processador destino. Caso as duas unidades de processamento envolvidas na comunicação estejam em um mesmo nó, a mensagem não trafega pela rede, mas é passada de um processador a outro através de escrita e leitura de dados em memória, fazendo com que haja novamente a possibilidade de contenção nos barramentos entre os processadores e a memória.

5.4 Algoritmos Genéticos Paralelos

Algumas características dos algoritmos genéticos permitem que a sua paralelização se dê de forma bastante natural [21]. Entre essas características podem-se destacar:

- O valor de aptidão de cada indivíduo numa população é independente de fatores externos a esse indivíduo;
- A aplicação dos operadores genéticos pode ser feita em qualquer ordem a qualquer indivíduo da população.

A estratégia de divisão de tarefas utilizada pode ser aplicada em algoritmos genéticos de várias maneiras. Alguns métodos de paralelização utilizam uma única população, enquanto outros dividem a população em diversas subpopulações relativamente isoladas [22] [23]. Nesse trabalho, adota-se a classificação apresentada por Cantú Paz em [24], que divide a população dos algoritmos genéticos em: (1) população única e global, (2) população única de granularidade fina, (3) população múltipla de granularidade grossa e (4) modelos híbridos. Esses modelos são melhor apresentados a seguir.

5.4.1 Algoritmo Genético Mestre-Escravo

Nos algoritmos genéticos paralelos do tipo mestre-escravo, uma única população é utilizada e a avaliação de indivíduos e/ou aplicação dos operadores genéticos é feita em paralelo. Como em um AG sequencial, cada indivíduo deve competir e ser

operado com qualquer outro indivíduo da população, e por isso algoritmos desse tipo são também conhecidos como algoritmos genéticos paralelos globais.

Como a aptidão de um indivíduo é independente do resto da população, essa é a operação mais comumente paralelizada em algoritmos genéticos mestre-escravo. A avaliação dos indivíduos é paralelizada pela atribuição de uma fração da população a cada um dos processadores disponíveis e a comunicação acontece apenas quando cada processador recebe seu subconjunto de indivíduos para avaliar e quando retorna seus valores de aptidão.

Assim, nesse tipo de algoritmo genético paralelo, há um processo central (mestre) que armazena todos os indivíduos da população e é responsável pela distribuição dos indivíduos aos outros processos (escravos), que são responsáveis por calcular o valor de aptidão de cada indivíduo. Geralmente é tarefa do processo mestre a aplicação dos operadores de seleção, mutação e cruzamento à população. Esse esquema pode ser melhor visualizado na Figura 5.8.

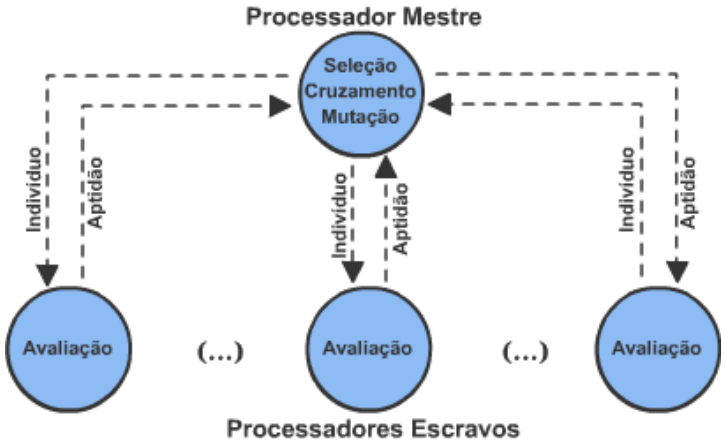

Figura 5.8: Algoritmo genético paralelo mestre-escravo

Se a cada geração o AG aguarda o recebimento dos valores de aptidão de toda população antes de prosseguir, então o algoritmo é síncrono e tem exatamente as mesmas propriedades de um AG sequencial, diferenciado apenas pelo tempo de execução. Entretanto, é possível que se implemente algoritmos genéticos mestre-escravo de maneira assíncrona, onde o algoritmo não aguarda o valor de aptidão de todos os processos, mas esse tipo de abordagem não funciona exatamente como um AG sequencial.

O modelo de paralelização global não assume nada sobre a arquitetura do compu-

tador paralelo em que é executado, e pode ser implementado eficientemente tanto em computadores com memória compartilhada quanto em computadores com memória distribuída. Em computadores com memória distribuída, o processo mestre deve ser responsável por a) enviar explicitamente os indivíduos aos processos escravos para avaliação, e b) coletar os resultados.

Nesse modelo de paralelização deve-se ter como preocupação a manutenção do balanceamento de carga computacional entre os processos através da utilização de esquemas de escalonamento dinâmico.

Outro aspecto dos algoritmos genéticos mestre-escravo que pode ser paralelizado é a aplicação dos operadores genéticos. Cruzamento e mutação podem ser paralelizados utilizando a mesma idéia do particionamento da população e da distribuição do trabalho entre os vários processos. Entretanto, esses operadores são tão simples que geralmente o tempo requerido para o envio dos indivíduos entre os processos pode eliminar qualquer ganho de desempenho.

5.4.2 Granularidade Fina

Algoritmos genéticos paralelos de granularidade fina utilizam apenas uma população, mas há uma estrutura espacial que limita a interação entre os indivíduos: um indivíduo pode apenas competir e ser operado com seus vizinhos. A sobreposição das vizinhanças permite que boas soluções possam ser disseminadas por toda a população. A Figura 5.9 apresenta uma visão esquemática de um modelo de granularidade fina definido por uma topologia de grade, onde cada nó representa um processo. O caso ideal nesse modelo é que cada processo corresponda a um indivíduo.

É comum distribuir os indivíduos de um algoritmo genético paralelo em uma grade bidimensional, já que em muitos computadores massivamente paralelos os elementos de processamento são conectados com essa topologia.

5.4.3 Granularidade Grossa

Modelos de algoritmos genéticos paralelos com múltiplas populações de granularidade grossa consistem na divisão da população em diversas subpopulações, que evoluem independentemente e trocam indivíduos ocasionalmente. Essa troca de indivíduos é denominada migração e é controlada por diversos parâmetros. Algo-

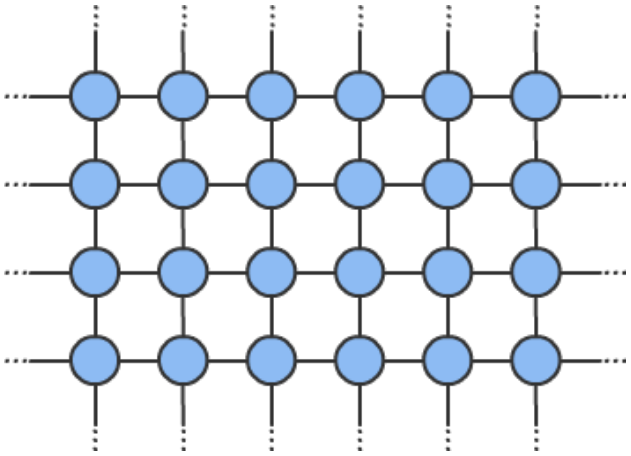

Figura 5.9: Algoritmo genético paralelo de granularidade fina.

ritmos genéticos paralelos com múltiplas populações são conhecidos na literatura por diferentes nomes, como algoritmos genéticos distribuídos ou modelo de ilha. A Figura 5.10 mostra uma representação esquemática desse tipo de modelo.

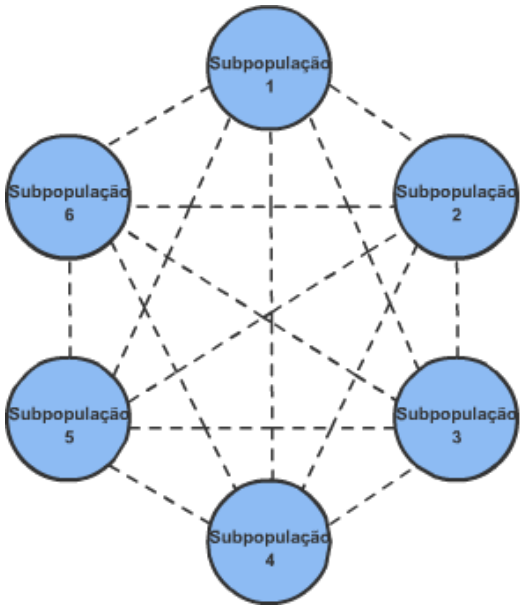

Figura 5.10: Algoritmo genético paralelo de granularidade grossa.

Como o tamanho das populações nesse modelo são menores do que em algoritmos genéticos sequenciais, espera-se que convirjam mais rapidamente. Entretanto, quando compara-se o desempenho do modelo de granularidade grossa com algoritmos sequenciais deve-se também levar em conta que a qualidade da solução em cada ilha tende a ser mais pobre.

O modelo de ilhas tem como inspiração principal a evolução das espécies na na-

tureza. A observação das espécies que se desenvolvem em ambientes isolados mostra que muitas delas desenvolvem uma adaptação muito grande a particularidades de seus ambientes.

Os algoritmos genéticos de granularidade grossa têm uma sequência de execução muito parecida com algoritmos genéticos sequenciais. Cada subpopulação evolui da maneira tradicional, diferenciando-se apenas pelo operador de migração dos elementos entre as diferentes ilhas.

A operação de migração ocorre numa frequência não muito grande, fazendo com que a sobrecarga causada pela comunicação entre os processos seja aceitável. Assim, esse tipo de modelo é muito bem aplicada em sistemas paralelos onde a rede não é muito rápida.

5.4.4 Modelos Híbridos

Uma outra alternativa para modelagem de algoritmos genéticos paralelos é a combinação de dois modelos de paralelização, produzindo algoritmos genéticos paralelos hierárquicos. Quando dois métodos de paralelização de algoritmos genéticos são combinados eles formam uma hierarquia, na qual, no nível mais alto, estão os algoritmos de múltiplas populações.

O nível inferior pode ter um AG de granularidade fina, aproveitando-se o poder de computadores massivamente paralelos, como é esquematizado na Figura 5.11.

Uma outra abordagem é utilizar o modelo mestre-escravo no nível inferior, como é esquematizado na Figura 5.12. Assim, cada ilha corresponde a um conjunto de processadores.

Pode-se também utilizar modelos com múltiplas populações em ambas as camadas, como é esquematizado na Figura 5.13. Assim, pode-se forçar a mistura dos indivíduos no nível inferior com uma taxa de migração alta, enquanto uma taxa de migração baixa é utilizada no nível mais baixo.

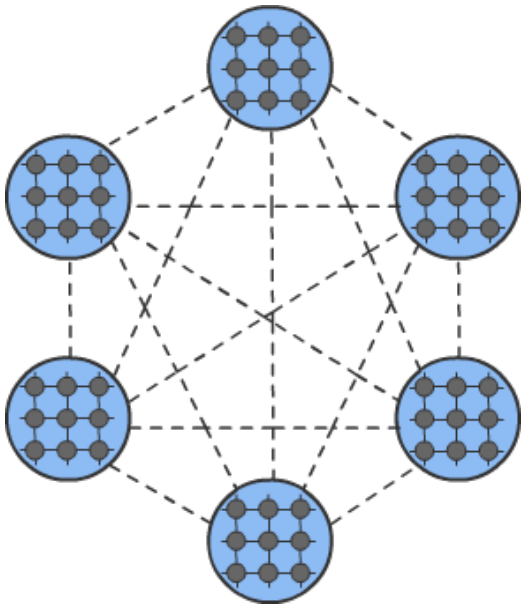

Figura 5.11: Algoritmo genético paralelo híbrido com granularidade fina.

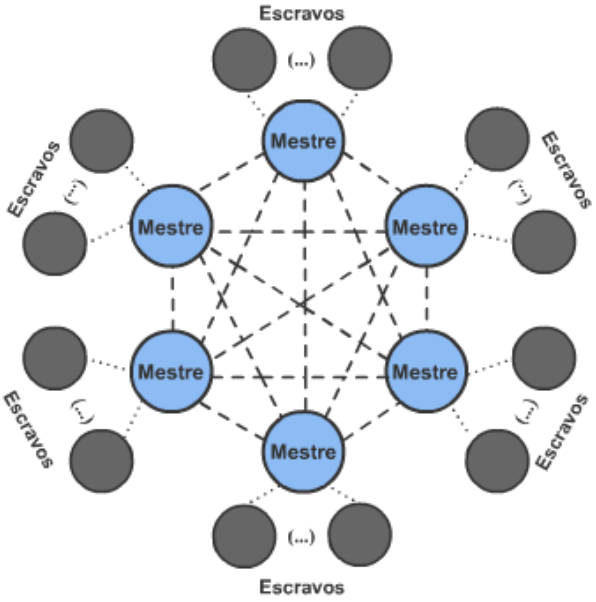

Figura 5.12: Algoritmo genético paralelo híbrido com mestre-escravo.

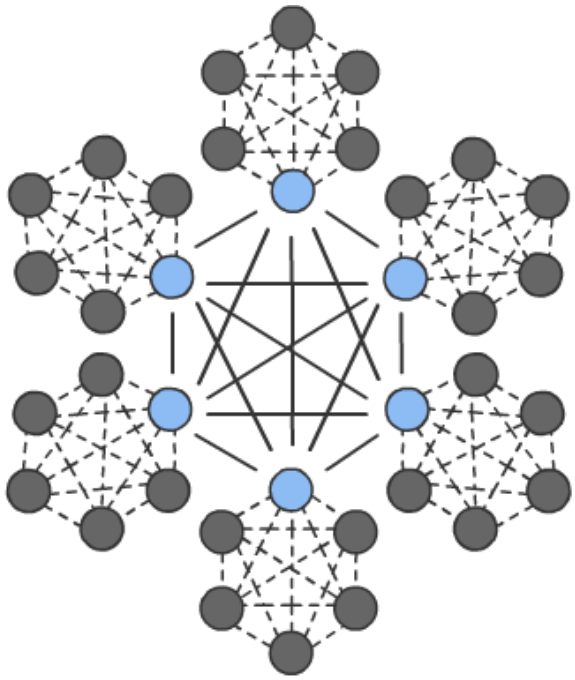

Figura 5.13: Algoritmo genético paralelo híbrido com granularidade grossa.

Capítulo 6

Implementações

6.1 Implementação do Simulador

O modelo implementado neste trabalho é um simulador de reservatórios simplificado. Trata-se de um problema de escoamento bifásico, bidimensional, com fluidos incompressíveis.

Nesta seção, será apresentado o esquema numérico de resolução utilizado e, em seguida alguns detalhes da implementação.

6.1.1 Esquema de resolução numérica

Esta seção apresenta o esquema numérico desenvolvido para o sistema (2.20). Especificações do modelo e da implementação podem ser encontradas ainda mais detalhadas no Apêndice A.

O Esquema IMPES

O esquema *Implicit-Pressure-Explicit-Saturation* (IMPES), como o nome sugere, obtém uma solução de forma implícita para a pressão seguido por uma solução explícita para a saturação.

A pressão é calculada para cada passo de tempo $n = 0, 1, \ldots, N$. Nesta implementação, o tamanho desses passos Δt_p é variável, mudando de acordo com a variação de pressão. Em cada um desses passos, são feitas as iterações para o cálculo da saturação.

A saturação é calculada do instante de tempo da pressão atual n até o próximo instante que a pressão será calculada $n + \Delta t_p$. Neste intervalo o Δt_s da saturação é calculado de forma a respeitar as condições de CFL.

O esquema IMPES consiste nas seguintes etapas.

inicio
  Para $n = 0, 1, \ldots, N$;
  Dado s^n, obtenha p^n como solução de;
  $\nabla_h.(T_t(s^n)\nabla_h p^n) = q_t^n.$;
  Faça $v^n = T_t(s^n)\nabla_h p^n$;
  Conhecidos p^n e $s^n = s^{n,0}$ determine $s^{n,\ell}$ para $\ell = 0, 1, \ldots, L-1$
  resolvendo $\phi\delta_t s^{n,\ell} = \nabla_h.(f(s^{n,\ell})v^n) + q_a^{n,\ell}.$;
  Faça $s^{n+1} = s^{n,L}$;
fim

Algoritmo 7: Pseudo-código de esquema IMPES

Discretização Temporal

A resolução numérica será feita usando dois passos de tempo, $(\Delta t)_n$ para a pressão e $(\Delta t)_{n,\ell}$ para a saturação. Um vetor de pressão p^n corresponde ao instante $t^n = \sum_{1\leq i\leq n}(\Delta t)_n$, enquanto que a saturação $s^{n,\ell}$ corresponde ao instante $t^{n,\ell} = t^n + \sum_{j\leq\ell}(\Delta t)_{n,j}$, para $n = 0, \ldots, N-1$, $\ell = 1, \ldots, L$. Define-se $s^{n,0} = s^{n-1,L}$ e t^0 corresponde a $t = 0$. Aproxima-se de $\partial_t s$ em $t^{n,\ell}$ por

$$\delta_t s^{n,\ell} = \frac{s^{n,\ell+1} - s^{n,\ell}}{(\Delta t)_{n,\ell}}, \quad \ell = 0, 1, \ldots L-1. \tag{6.1}$$

Discretização Espacial

Considera-se o reservatório Ω como um retângulo de dimensões $M_x \times M_y$. Dividem-se os lados de Ω em N_x e N_y partes, respectivamente. Sejam $\Delta x = M_x/N_x$, $\Delta y = M_y/N_y$. Define-se retângulos V_{ij} centrados em $C_{ij} = (x_i, y_j)$ onde $x_i = (i+1/2)\Delta x$ e $y_j = (j+1/2)\Delta y$, $0 \leq i < N_x$, $0 \leq j < N_y$. Denota-se ainda $x_{i+1/2} = (x_i + x_{i+1})/2$, e define-se analogamente $y_{j+1/2}$. Assim, inteiros i, j dizem respeito aos centros V_{ij} ao passo que índices $i + 1/2$ e $j + 1/2$ estão relacionados às interfaces, como apresenta a Figura 6.1

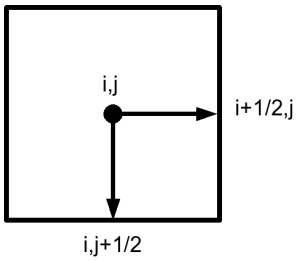

Figura 6.1: Representação dos retângulos

Seja F uma função definida nas interfaces $(i+1/2, j)$ e $(i, j+1/2)$. Introduz-se o divergente discreto ∇_h. dado por

$$\nabla_h.F_{i,j} = \frac{F_{i+1/2,j} - F_{i-1/2,j}}{\Delta x} + \frac{F_{i,j+1/2} - F_{i,j-1/2}}{\Delta y}. \tag{6.2}$$

Para f definida nos centros (i, j) dos retângulos, define-se o gradiente discreto ∇_h como uma função de interface dada por

$$\nabla_h f_{i+1/2,j} = \frac{f_{i+1,j} - f_{i-1,j}}{\Delta x}, \qquad \nabla_h f_{i,j+1/2} = \frac{f_{i,j+1} - f_{i,j-1}}{\Delta y}. \tag{6.3}$$

Cálculo das transmissibilidades

Observe que é preciso calcular as transmissibilidades T em interfaces de V_{ij}. Assume-se que as viscosidades são constantes. Além disso, estima-se a permeabilidade absoluta usando médias harmônicas, como na Equação 6.4.

$$K_{i+1/2,j} = \frac{2K_{i,j}K_{i+1,j}}{K_{i,j} + K_{i+1,j}}, \qquad K_{i,j+1/2} = \frac{2K_{i,j}K_{i,j+1}}{K_{i,j} + K_{i,j+1}}. \tag{6.4}$$

Por fim, será utilizado o esquema a montante (*upwind*) para as permeabilidades relativas: se $p_{i+1,j} > p_{i,j}$ então $(k_r)_{i+1/2,j} = k_r(s_{i+1,j})$. Tratamento análogo é feito nas outras interfaces.

Condição de Fronteira

Como foi dito no Capítulo 2 o reservatório está isolado e portanto $v_a = 0$, $v_o = 0$. Para isto basta anular $T_{i+1/2,j}$ e $T_{i,j+1/2}$ nas interfaces que pertençam aos bordos.

Tratamento dos Poços

Nos poços injetores, a vazão total q_t é igual à vazão de água q_a que é prescrita. Para se obter densidades de vazões, divide-se a vazão dada pelo volume do bloco $\Delta x \Delta y h$, onde h é a altura do reservatório.

Nos poços produtores, q_t é conhecido. Calcula-se então q_a como uma fração da vazão total.

$$(q_a)_{i,j}^{n,\ell} = f(s^{n,\ell}) q_t^n, \tag{6.5}$$

veja (2.19).

Solução do sistema de pressão

A solução do sistema implícito da pressão $Ax = b$ foi implementada de acordo com os seguintes passos: montagem da matriz A de coeficientes; prescrição das vazões nos poços para formar o vetor b; resolução do sistema linear para encontrar o vetor de pressões x.

Como o modelo possui condições de contorno de Neumann homogênea [6], o sistema possui solução a menos de uma constante. Para contornar esse problema, um ponto da malha teve a sua pressão prescrita.

Para a solução foi utilizado o método de gradientes conjugados com precondicionador multigrid algébrico, ambos fornecidos pela biblioteca PETSc, que será descrita mais adiante.

Solução da equação da saturação

A equação da saturação é explícita, portanto sua solução se dá de forma direta. O tratamento nas interfaces usa esquema *upwind* e seus passos de tempo obedecem às condições de CFL.

6.2 Implementação do algoritmo genético

O algoritmo genético implementado neste trabalho utilizou o modelo geracional com representação real, o cruzamento tipo *blend* e seleção por roleta. A população inicial

foi gerada aleatoriamente.

A função de avaliação corresponde a uma chamada ao simulador, onde o indivíduo representa uma solução candidata do campo de permeabilidades; o valor de aptidão é dado pela Equação 3.2.

Os critérios de parada utilizados foram: número máximo de gerações, $f(x)$ estagnado por mais de 5 gerações ou $f(x) < e$, onde e é uma tolerância prescrita.

6.3 Implementação dos métodos baseados em derivadas

Apesar dos algoritmos baseados em derivadas não terem sido implementados, foi necessária a implementação de algumas funções para o seu funcionamento.

Uma delas é a função objetivo dada por (3.2). Outra função necessária foi a do cálculo da derivada do sistema, que foi feito por diferenças finitas.

6.3.1 Diferenças Finitas

Os métodos do gradiente, gradientes conjugado e Quasi-Newton-BFGS necessitam da avaliação de ∇f a cada iteração, onde f á a função objetivo (3.2).

$$\nabla f = (\frac{\partial f}{\partial m_1}, \frac{\partial f}{\partial m_2}, \ldots, \frac{\partial f}{\partial m_{N_o}}); \tag{6.6}$$

Para o método de LM, precisamos avaliar a matriz Jacobiana J em cada iteração:

$$\begin{bmatrix} \frac{\partial O_1}{\partial m_1} & \frac{\partial O_1}{\partial m_2} & \cdots & \frac{\partial O_1}{\partial m_{N_m}} \\ \frac{\partial O_2}{\partial m_1} & \frac{\partial O_2}{\partial m_2} & \cdots & \frac{\partial O_2}{\partial m_{N_m}} \\ \frac{\partial O_3}{\partial m_1} & \frac{\partial O_3}{\partial m_2} & \cdots & \frac{\partial O_3}{\partial m_{N_m}} \\ \cdots & \cdots & \ddots & \cdots \\ \frac{\partial O_{N_o}}{\partial m_1} & \frac{\partial O_{N_o}}{\partial m_2} & \cdots & \frac{\partial O_{N_o}}{\partial m_{N_m}} \end{bmatrix} \tag{6.7}$$

onde O é o resultado do nosso histórico sintético. Observamos que a i-ésima coluna de J é dada por $\frac{\partial O}{\partial m_i}$.

Todos esses cálculos podem ser feitos através de diferenças finitas. O método das diferenças finitas é um método de resolução de equações diferenciais que se baseia na

aproximação de derivadas por diferenças finitas. A fórmula de aproximação obtém-se da série de Taylor da função derivada. O operador de diferenças finitas para derivada pode ser obtido a partir da série de Taylor, vista na Equação 6.8

$$f(x+h) = f(x) + f'(x)h + O(h^2) \tag{6.8}$$

portanto a derivada pode ser escrita como uma diferença mais um termo de erro, como expresso na Equação 6.9

$$f'(x) = \frac{f(x+h) - f(x)}{h} + O(h) \tag{6.9}$$

Desconsiderando-se o termo de $O(h)$ tem-se o operador de diferenças finitas para cada derivada parcial que precisamos:

$$\frac{\partial f}{\partial m_i} = \frac{f(m_1, \ldots, m_i + h, \ldots, m_{N_m}) - f(m_1, \ldots, m_i, \ldots, m_{N_m})}{h} \tag{6.10}$$

$$\frac{\partial O}{\partial m_i} = \frac{O(m_1, \ldots, m_i + h, \ldots, m_{N_m}) - O(m_1, \ldots, m_i, \ldots, m_{N_m})}{h} \tag{6.11}$$

Dessa forma, para se obter o ∇f ou J serão necessários $N_m + 1$ cálculos do simulador, onde N_m é o número de parâmetros a serem estimados.

6.3.2 Critério de parada

Os critérios de parada utilizados foram: número máximo de iterações, $f(x)$ estagnado por mais de 5 iterações, $f(x) < e$, ou $\nabla f < tol$, onde e e tol são tolerâncias prescritas.

6.4 Implementação paralela do simulador

O modelo de paralelização utilizado para o simulador foi o de divisão de dados.

A paralelização do simulador, assim como a implementação sequencial, foi feita em duas etapas. A primeira etapa foi a paralelização da solução do sistema de

equações da pressão, e a segunda o cálculo da equação da saturação.

Na solução da equação da saturação, cada processador atualiza a saturação de suas linhas correpondentes, mas como se utiliza o *stencil* de quatro vizinhos, após cada atualização é necessária a troca dos vizinhos, chamados *ghostpoint*. Para isso cada processador possui, além de suas linhas, mais a última linha do vizinho superior e a primeira linha do vizinho inferior, como apresenta a Figura 6.2.

Este esquema de paralelização é chamado de decomposição de domínio. O lado esquerdo da Figura 6.2 mostra a malha de discretização do reservatório; à direita é mostrado como a distribuição da malha é feita entre os processos.

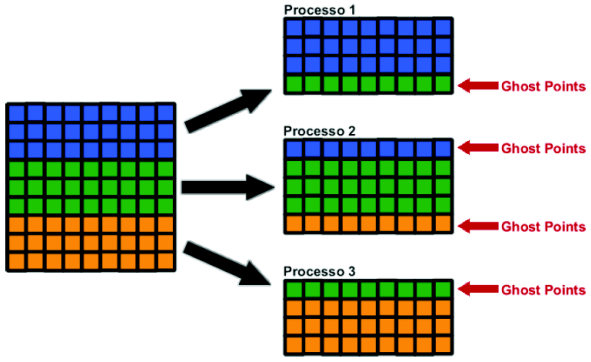

Figura 6.2: Troca das linhas *Ghost Points*

Como a cada passo de tempo a saturação muda, é necessário que os vizinhos troquem as linhas necessárias, para que não se faça nenhum cálculo com valores anteriores.

A solução do sistema de equações foi feita através de rotinas da biblioteca PETSc, que possui opção de resoluções paralelas com diversos métodos.

6.5 Implementação paralela do algoritmo genético

A implementação paralela do AG foi feita segundo a abordagem mestre-escravo, apresentada na Seção 5.4.1.

Porém esta implementação difere da implementação tradicional mestre-escravo, pois os escravos podem estar mapeados em um grupo de processadores, e esse grupo de processadores será agora o responsável por calcular a aptidão dos indivíduos.

O MPI fornece um comunicador predefinido, denominado MPI_COMM_WORLD, que identifica todos os processos envolvidos na computação através de identificadores globais. O PETSc utiliza como comunicador o PETSC_COMM_WORLD, que por padrão é o MPI_COMM_WORLD. Como nesta implementação é necessário fazer várias chamadas a funções do PETSc, foi necessária a redefinição dos comunicadores do PETSc com identicadores locais.

Esta implementação foi possível devido aos conceitos de grupos e comunicadores providos pelo MPI. Um comunicador permite a definição de módulos que encapsulam operações de comunicação dentro de um grupo. O comunicador identifica o grupo de processos e o contexto ao qual uma determinada operação deve ser aplicada.

Esta divisão dos comunicadores pode ser ilustrada pela Figura 6.3.

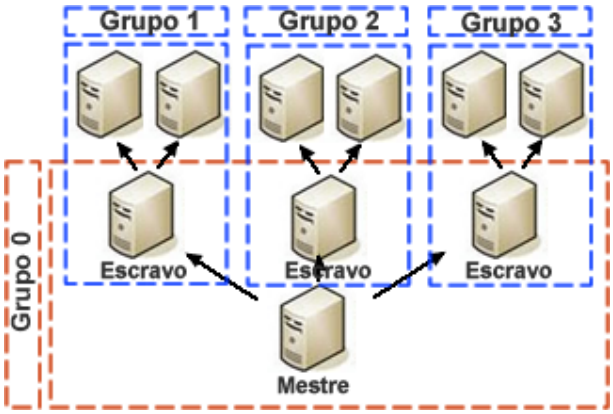

Figura 6.3: Implementação do algoritmo genético em dois níveis

Como a população inicial é aleatoria, um indivíduo pode ser muito diferente do outro; e o tempo de computação entre eles pode variar muito. Esta variação pode causar uma espera nos grupos ou processadores que terminaram sua tarefa antes. Para contornar este problema a implementação do AG paralelo foi feita de forma assincrona. Primeiramente o mestre envia um indivíduo para cada grupo, e em seguida ele aguarda o término de qualquer um dos grupos, sendo que o grupo que terminou recebe imediatamente um novo indivíduo para calcular.

Ao término de uma geração o mestre é quem realiza as operações genéticas, como seleção, cruzamento, mutação e elitismo.

6.6 Implementação paralela dos algoritmos baseados em derivadas

Como vimos na Seção 6.3.1, é necessário a execução do simulador $n + 1$ vezes para o cálculo de ∇f ou J.

A implementação paralela dos métodos baseados em derivadas se resume à paralelização do cálculo do $f(x_i)$ necessário para o cálculo das derivadas, e a abordagem utilizada também foi o mestre-escravo.

Neste cálculo, cada um dos grupos de escravos calcula uma função necessária para o cálculo da derivada parcial em relação a algum das variáveis. Caso haja mais variáveis do que grupos, alguns grupos podem calcular mais de uma dessas funções.

Após o cálculo de todos os $f(x_i)$ necessários, o mestre reúne informações para obter o gradiente ou Jacobiana do sistema; e assim prossegue com o algoritmo, caso nenhum dos critérios de parada tenham sido satisfeitos.

6.7 Linguagens e Bibliotecas

O modelo descrito no Capítulo 2, assim como o AG, foi implementado em C. O simulador utilizou a biblioteca *Portable, Extensible Toolkit for Scientific Computation*(PETSc) e para a implementação paralela foi utilizada a biblioteca Open MPI, que implementa o MPI.

A PETSc é uma biblioteca *open source* e gratuita, ela consiste em conjunto de estruturas de dados e rotinas que podem ser utilizadas em aplicações paralelas. É uma execelente solução para aplicações científicas modeladas por equações diferenciais parciais. Esta biblioteca emprega o padrão MPI para paralelismo.

O Open MPI é um projeto *open source* e gratuíto, que implementa o padrão MPI, o projeto é desenvolvido e mantido por um consórcio de ensino, de investigação, indústria e parceiros.

As funções requeridas pelos algoritmos baseados em derivadas também foram implementadas em C, o método do gradiente, o BFGS e o gradiente conjugado são implementações da biblioteca GSL, e o Levemberg-Marquardt é uma implementação de um grupo de pesquisa disponível em [25].

A biblioteca *GNU Scientific Library*(GSL) é uma biblioteca *open source* e gratuita, para o uso de desenvolvedores C/C++, a biblioteca oferece uma ampla variedade de rotinas matemáticas [26].

Capítulo 7

Metodologia

7.1 Validação do Simulador

Para validar o simulador descrito no Capítulo 6 foram realizados três testes que serão descritos na seções seguintes.

7.1.1 Buckley-Leverett

No problema de Buckley-Leverett considera-se em um reservatório unidimensional com dois poços: um injetor e um produtor, dispostos como mostra a Figura 7.1.

A teoria básica do deslocamento de fluidos imiscíveis apresentada por Buckley-Leverett apresenta as seguintes condições físicas:

- Deslocamento unidimensional
- Meio poroso homogêneo
- Fluidos imiscíveis

Para este problema, a solução analítica para a saturação é conhecida. Portanto, pode-se comparar a solução do simulador implementado com o resultado analítico. A demonstração deste resultado analítico pode ser obtida em [6].

Os parâmetros utilizados para gerar a solução numérica foram os seguintes:

$$k_{ra}(s) = s \ , \ k_{ro}(s) = 1 - s \ , \ s_{ai} = s_{or} = 0.2 \ , \ \rho_a = 1.0cp, \ \rho_o = 5.0cp \ , \ \mu_a = \mu_o = 1.0,$$
$$Q_i = 5m^3, \ Q_p = -5m^3 \ , \ h = 5.0 \ , \ K = 100.0mD \text{ homogêneo e constante,}$$
$$Lx = 50.0 \ , \ Ly = 4.0$$

Figura 7.1: Reservatório Buckley Leverett

Análise de erros

O erro relativo e_r considereado foi:

$$e_r = \frac{\sqrt{\sum_i (s_i - \bar{s}_i)^2 dxdy}}{\sqrt{\sum_i (\bar{s}_i)^2 dxdy}} \tag{7.1}$$

onde s é a saturação numérica e $\bar{s}$ a saturação analítica.

7.1.2 Problema 2-spot homogêneo

O problema 2-spot consiste em um reservatório de dimensões (Lx, Ly), com dois poços: um injetor na posição (0, 0) e um produtor na posição (Lx, Ly).

Para este problema, a solução analítica do campo de pressões é conhecida, podendo então ser comparada à solução numérica para validação.

Os parâmetros utilizados para gerar a solução numérica foram os seguintes:

$$k_{ra}(s) = s,\ k_{ro}(s) = 1 - s\ ,\ s_{ai} = s_{or} = 0.2\ ,\ \rho_a = \rho_o\ ,\ \mu_a = \mu_o = 1.0,\ Q = 25.0\ ,$$
$$h = 20.0m\ ,\ K = 100.0mD \text{ homogêneo e constante }, Lx = Ly = 100.0m$$

Análise de erros

O erro relativo e_r considerado para este problema, calculado como na Equação:

$$e_r = \frac{\sqrt{\sum_{i,j} (p_{ij} - \bar{p}_{ij})^2 dxdy}}{\sqrt{\sum_{i,j} (\bar{p}_{ij})^2 dxdy}}$$

onde p é a pressão numérica e $\bar{p}$ a pressão analítica.

7.1.3 Problema 5-spot homogêneo

O problema 5-spot consiste em um reservatório de dimensões (Lx, Ly) com cinco poços, quatro injetores e um produtor, dispostos como mostra a Figura 7.2.

Para este problema, a solução analítica do campo de pressões é conhecida, podendo então ser comparada a solução numérica para validação.

Os parâmetros utilizados nas simulações do problema 5-spot foram os seguintes

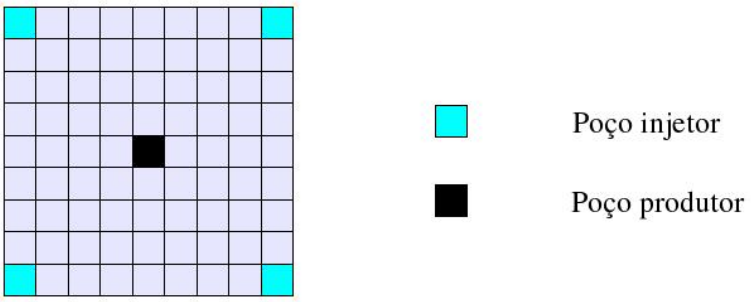

Figura 7.2: 5-spot

$$k_{rw}(s) = \left(\frac{k_{rw0}*(s-s_{wi})}{1-s_{or}-s_{wi}}\right)^{\alpha_w}$$
$$k_{ro}(s) = \left(\frac{k_{ro0}*(1-s_{or}-s)}{1-s_{or}-s_{wi}}\right)^{\alpha_o}$$
$$\alpha_w = \alpha_o = 2$$
$$k_{ra0} = 0.4 \ , \ k_{ro0} = 0.8 \ , \ s_{ai} = s_{or} = 0.2 \ , \ \mu_a = 1.0cp \ , \ \mu_o = 5.0cp \ , \ \rho_a = 1.0 \ ,$$
$$\rho_o = 0.8 \ , \ Q_{inj} = 100.0m^3/dia \ , \ Q_{prod} = -400.0m^3/dia \ , \ K = 100.0mD \ , \ \phi = 0.2 \ ,$$
$$Lx = Ly = 200.0m, h = 20.0m, \ tempo = 600 \text{ dias}$$

Análise de erros

Por não possuir solução analítica para a saturação, o problema 5-spot foi simulado com uma malha 401x401. A solução gerada por esta simulação foi utilizada como solução analítica do problema para a análise do erro. O erro relativo foi calculado da seguinte forma:

$$e_r = \frac{\sqrt{\sum_{i,j}(x_{ij}-\bar{x}_{ij})^2}}{\sqrt{\sum_{i,j}(\bar{x}_{ij})^2}}$$

onde x foi substituído pela saturação e pressão numéricas; e $\bar{x}$ foi substituído pela saturação e pressão usando a malha mais fina.

7.2 Comparação dos métodos de otimização

Para o problema inverso, onde queremos descobrir o campo de permeabilidade, K representa a distribuição de permeabilidades no reservatório.

Primeiramente, foram feitos testes exaustivos para um melhor conhecimento do problema e para o ajuste dos parâmetros dos algoritmos utilizados.

No AG, foi necessário o ajuste das taxas de mutação e de combinação, assim como o número de gerações.

Nos demais algoritmos, alguns parâmetros como o h no cálculo das diferenças finitas e o erro e aceitável também foram ajustados.

Para a comparação dos métodos de otimização foram gerados três problemas sintéticos, com 2, 4 e 9 permeabilidades a serem estimadas.

Esses problemas tem as seguintes características:

$$k_{rw0} = 0.4,\ k_{ro0} = 0.8\ ,\ s_{wi} = s_{or} = 0.2\ ,\ \mu_w = 1.0cp\ ,\ \mu_o = 5.0cp\ ,\ \rho_w = 1.0\ ,$$
$$\rho_o = 0.8\ ,\ Q_{inj} = 100.0m^3/dia\ ,\ Q_{prod} = -400.0m^3/dia\ ,\ \phi = 0.2\ ,$$
$$Lx = Ly = 200.0m\ ,\ h = 20.0m\ ,\ tempo = 350 \text{ dias}$$

Por fim, a única variável que difere entre os problemas é a permeabilidade efetiva K, que se deve estimar. Para 2 parâmetros, os valores objetivos são $K = 47.82; 145.2$, para 4 parâmetros $K = 39.12; 67.99; 52.85; 267.78$. Para 9 parâmetros $K = 43.22; 38.21; 55.76; 56.39; 95.61; 148.45; 36.24; 135.84; 261.57$. Os testes feitos com 4 parâmetros divide a malha em 4 grandes blocos com diferentes valores de permeabilidade, como mostra a Figura 7.3.

Os testes feitos com 9 parâmetros é análogo aos de 4, e é representado pela Figura 7.4.

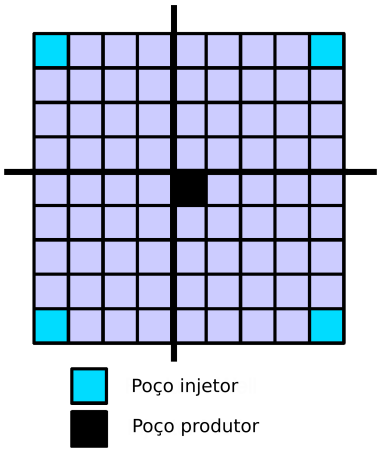

Figura 7.3: Representação da divisão da Malha - 4 blocos

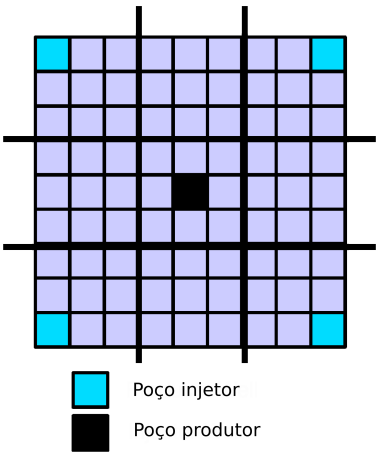

Figura 7.4: Representação da divisão da Malha - 9 blocos

7.2.1 Parâmetros do AG

O algoritmo genético utilizado na comparação dos métodos foi executado com os seguintes parâmetros: A taxa de probabilidade de mutação foi de 2.5%; a taxa de

probabilidade de cruzamento foi de 85%; o critério de parada utilizado foi de 50 gerações, 5 gerações com o $f(x)$ estagnado ou $f(x) < 10^{-6}$. A população inicial foi de 100 indivíduos, que foram gerados aleatoriamente. O elitismo desta população foi de um indivíduo.

7.2.2 Parâmetros dos métodos baseados em derivadas

Os parâmetros ajustados para os algoritmos baseados em derivadas foram $h = 10^{-4}$, tolerâncias do gradiente $tol = 10^{-16}$, e o erro da função objetivo $e = 10^{-6}$, o número máximo de iterações igual a 100.

7.2.3 Pontos Iniciais

Os indivíduos da primeira população do AG foram gerados aleatoriamente. Antes de cada indivíduo ser inserido, ele foi avaliado e se sua aptidão fosse maior do que um $f(x)$ determinado, este indivíduo era escolhido para entrar na população inicial.

Para cada teste, com 2, 4 e 9 parâmetros, cada método foi executado 10 vezes para cada um dos limites de $f(x)$ escolhidos. Para cada uma dessas execuções, o melhor indivíduo da população inicial do AG foi utilizado como ponto inicial dos demais algoritmos baseados em derivadas. Todos os testes foram executados em paralelo.

7.3 Testes das implementações paralelas

Uma medida muito utilizada para avaliar o desempenho de aplicações paralelas é o *speedup*, definido em [27] como $S(n,p) = \frac{T_(n,1)}{T_(n,p)}$, onde n é o tamanho do problema e p o número de processos.

Esta medida deve ser maior do que 1 para que se considere que algoritmo obteve algum ganho, caso seja contrário esta taxa é denominada *slowdown*. O *speedup* ideal é $S(n,p) = p$, mas isto nem sempre é possível, devido à adição de algum *overhead* de comunicação. Neste trabalho, a medida de avaliação do desempenho do algoritmo paralelo foi *speedup*, o qual foi avaliado para diferentes tamanhos de malhas de simulação.

7.4 Ambiente de execução paralelo

Como ambiente para execução das aplicações paralelas, foi utilizado um *cluster* de computadores multiprocessados localizado no FISIOCOMP, Laboratório de Fisiologia Computacional e Computação de Alto Desempenho.

O *cluster* é composto de 4 máquinas idênticas com processadores Core2Quad®, com 4 Gb de memória RAM, 160 de disco rígido e um computador *front-end* Atlhon®XP, com 2 GB de de memória RAM, 160 Gb de disco rígido. Todas as máquinas são interligadas por um *switch* Ethernet 1Gb/s.

7.4.1 Simulador Paralelo

Para testar a corretude e a eficiência da paralelização do simulador foram necessárias várias execuções. Primeiro, para validar a corretude, os resultados de saída foram comparados com a versão sequencial, já validada.

O problema utilizado nos testes paralelos é um 5-spot semelhante ao citado anteriormente, que simula 50 dias. Foram utilizados diferentes tamanhos de malha: 101x101, 401x401 e 1001x1001.

Em seguida o simulador foi executado em 1, 2 ,4, 8 e 16 processos para diferentes tipos de malha, a fim de verificar a escalabilidade do mesmo. Além da verificação da escalabilidade do método, foram executados testes com diferentes mapeamentos entre tarefas e núcleos computacionais. Cada uma das configurações foi executada 5 vezes, os resultados apresentam a média dos tempos de execução.

- 1_1: 1 processo executando em 1 máquina
- 2_1: 2 processos executando em 1 máquina
- 2_2: 2 processos executando em 2 máquinas
- 4_1: 4 processos executando em 1 máquina
- 4_2: 4 processos executando em 2 máquinas
- 4_4: 4 processos executando em 4 máquinas
- 8_2: 8 processos executando em 2 máquinas

- 8_4: 8 processos executando em 4 máquinas
- 16_4: 16 processos executando em 4 máquinas

As Figuras 7.8, 7.9 e 7.10 ilustram as diferentes geometrias para as execuções com 4 processos. Os núcleos em vermelho são os núcleos ocupados com o cálculo de seu domínio da malha do simulador.

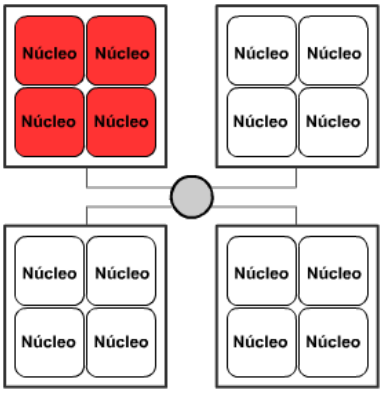

Figura 7.5: Divisão dos processos IMPES 4_1 - 4 processos em 1 máquina

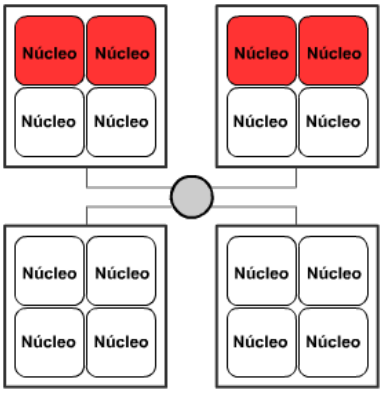

Figura 7.6: Divisão dos processos IMPES 4_2 - 4 processos em 2 máquinas

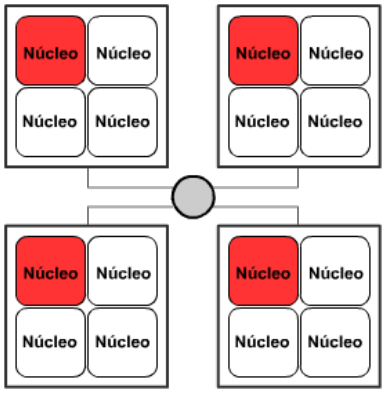

Figura 7.7: Divisão dos processos IMPES 4_4 - 4 processos em 4 máquinas

7.4.2 Algoritmos genéticos paralelos

Os testes dos algoritmos genéticos paralelos foram executados com diversos tamanhos de malha e diferentes distribuições de processos nas máquinas para a escolha do caso que seria mais adequado.

Primeiramente foram utilizadas as mesmas malhas dos testes do simulador paralelo. Para cada uma dessas malhas foram feitos testes com o AG em 17 processadores

dispostos em 4 *quadcores* e 1 um computador monoprocessado. O computador monoprocessado foi tomado sempre como o mestre e os demais *quadcores* como escravos, que apenas calculam o IMPES, que é necessário para a função de avaliação do AG.

Temos 4 máquinas, cada uma com 4 processadores. Usando os 16 processadores além do mestre, os seguintes testes foram realizados com diferentes distribuições de tarefas nos núcleos.

- 1_1: Cada avaliação executa em 1 núcleo.
- 2_1: Cada avaliação executa em 2 núcleos da mesma máquina.
- 2_2: Cada avaliação executa em 2 núcleos em duas máquinas.
- 4_1: Cada avaliação executa em 4 núcleos da mesma máquina.
- 4_2: Cada avaliação executa em 4 núcleo em duas máquinas.
- 4_4: Cada avaliação executa em 4 núcleos em quatro máquinas.
- 8_2: Cada avaliação executa em 8 núcleos em duas máquinas.
- 8_4: Cada avaliação executa em 8 núcleos em quatro máquinas.
- 16_4: Cada avaliação executa em 16 núcleos em quatro máquinas.

No algoritmo genético todos os núcleos estão sempre ocupados, e a distribuição dos processos nas máquinas no AG pode ser ilustrado pelas seguintes figuras 7.8, 7.9 e 7.10.

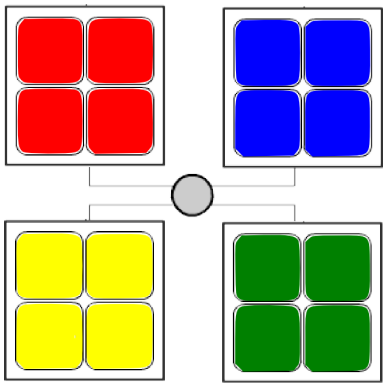

Figura 7.8: Divisão dos processos AG 4_1 - 4 processos em 1 máquina

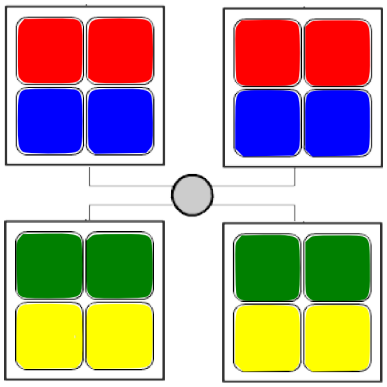

Figura 7.9: Divisão dos processos AG 4_2 - 4 processos em 2 máquinas

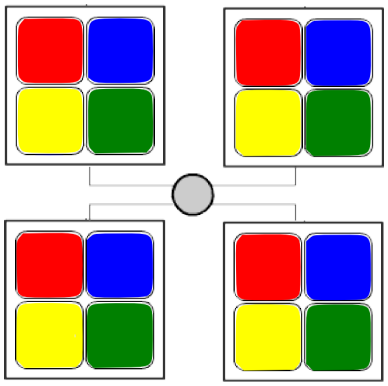

Figura 7.10: Divisão dos processos AG 4_4 - 4 processos em 4 máquinas

Todos os resultados destes testes aqui apresentados serão vistos no Capítulo 8 e a discussão no Capítulo 9.

Capítulo 8

Resultados

8.1 Resultados IMPES

8.1.1 Resultados Buckley Leverett

Com os parâmetros definidos na Seção 7.1.1, foram executadas diversas malhas. As comparações dos campos de saturação gerados pelo simulador são feitas com os valores dados pela solução analítica. Os resultados serão apresentados a seguir, e todos tratam dos valores da saturação ao longo do tempo.

Na malha mais grosseira, $N_x = 100$, as evoluções dos perfis de saturação analítica e numérico foram relativamente próximas. Porém, a solução numérica, como era de se esperar, se mostra mais suave, como ilustrado na Figura 8.1, que compara o perfil de saturação no caso analítico com o caso numérico.

A malha mais refinada utilizada para teste foi a de $N_x = 1600$, que gerou o resultado representado na Figura 8.2, com erros bem menores.

Os erros de todas as malhas simuladas estão na Tabela 8.1:

Tabela 8.1: Análise de erros - Buckley Leverett

Malha	Erro relativo
100	0.021308853 %
200	0.016287871 %
400	0.013178739 %
800	0.009747218 %
1600	0.005778373 %

Os resultados comprovam a convergência de primeira ordem para a equação da

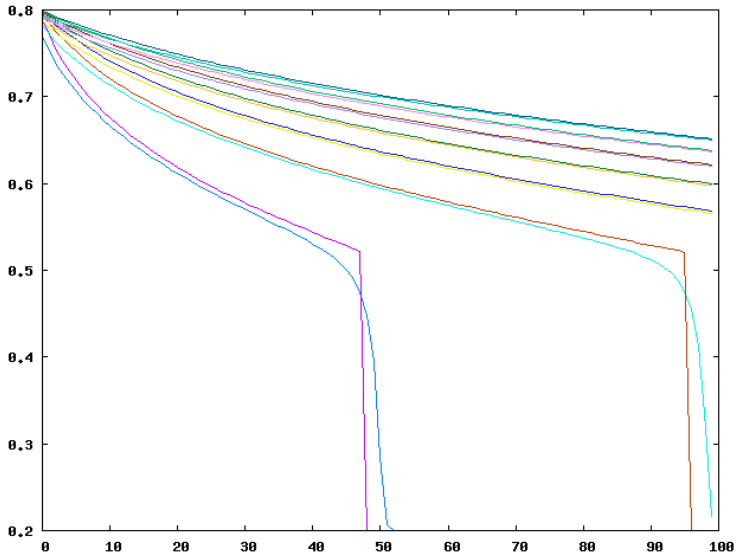

Figura 8.1: Evolução do perfil de saturação (analítico X numérico) malha grosseira

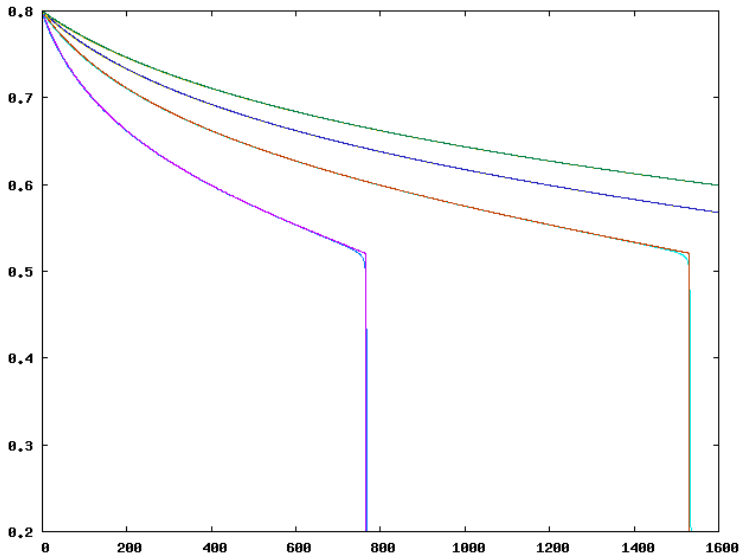

Figura 8.2: Evolução do perfil de saturação (analítico X numérico) malha refinada

saturação.

8.1.2 Resultados 2-Spot

Para validar a solução numérica do problem descrito na Seção 7.1.2, foi necessário subtrair do campo de pressões numérico a sua média, e então compará-lo ao analítico. Isto porque o campo de pressões analítico está definido a menos de uma constante.

A Figura 8.3 representa a malha simulada.

A Figura 8.4 é referente à área hachurada da Figura 8.3 e à simulação de uma

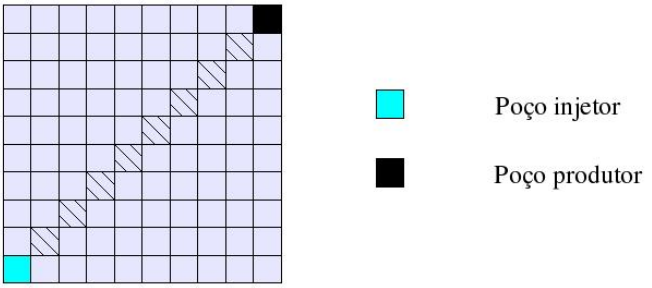

Figura 8.3: Malha 2-spot

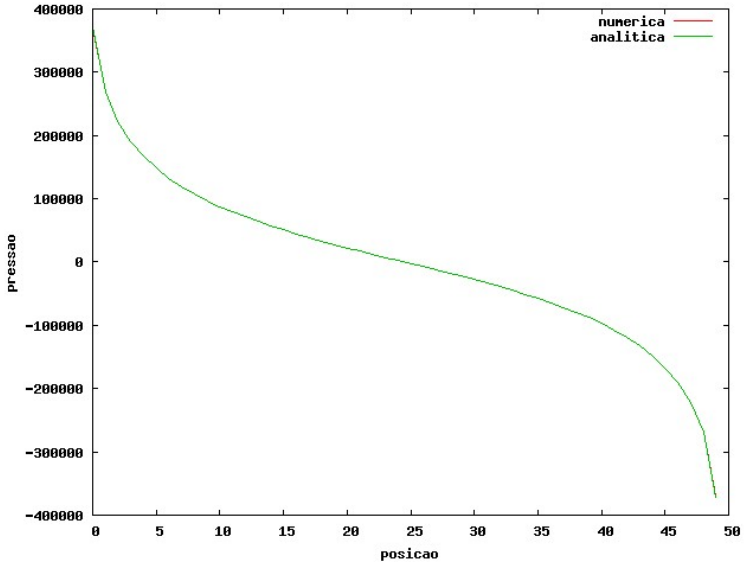

Figura 8.4: Pressões numérica x analítica

malha 50x50. Ela apresenta as pressões numérica e analítica.

Pode-se perceber que, visualmente, o campo de pressões numérico ficou muito próximo do analítico. A seguir será apresentada uma análise dos erros da solução numérica.

Foram feitas simulações com malhas 25x25, 50x50, 100x100, 200x200 e 400x400. Os resultados para cada uma das malhas são apresentados na Tabela 8.2

Tabela 8.2: Análise de erros - 2-spot

Malha	Erro relativo
25x25	2.098208e-03
50x50	8.601082e-04
100x100	2.447825e-04
200x200	6.293299e-05
400x400	1.583606e-05

Pode-se concluir que o erro diminui de forma quadrática para a equação da pressão.

8.1.3 Resultados 5-Spot

As Figuras 8.5, 8.6, 8.7, 8.8 apresentam a saturação, para uma malha de discretização 201x201, nos dias 5, 100, 300 e 600, respectivamente. Pode-se notar que o campo de saturação mantém uma simetria.

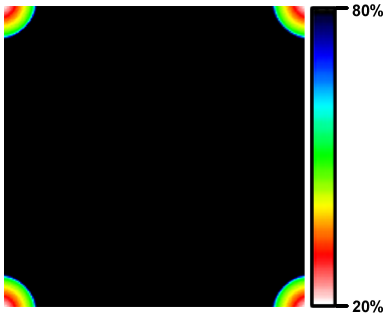

Figura 8.5: 5 dias

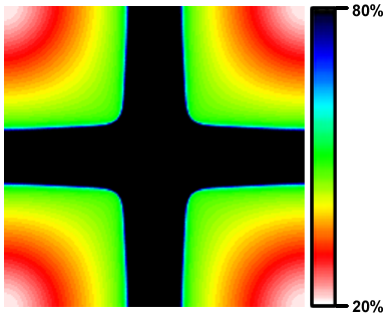

Figura 8.6: 100 dias

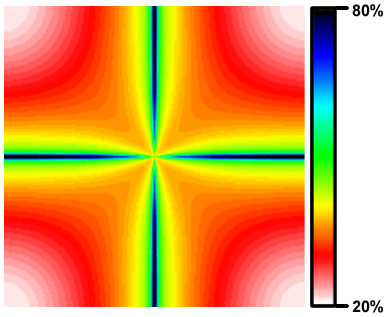

Figura 8.7: 300 dias

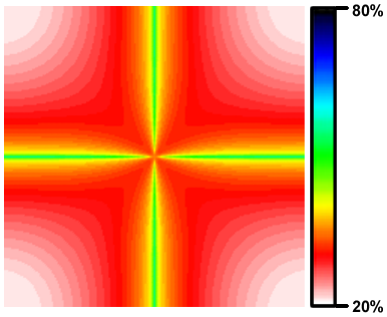

Figura 8.8: 600 dias

As Tabela 8.3 e 8.4 apresentam os erros relativos calculados para a saturação e pressão, respectivamente.

Tabela 8.3: Análise de erros da saturação

Malha	Erro relativo
25x25	1.112241e-01
51x51	6.604010e-02
101x101	3.865579e-02
201x201	2.263281e-02

Tabela 8.4: Análise de erros da pressão

Malha	Erro relativo
25x25	4.110158e-01
51x51	3.972870e-01
101x101	2.653558e-01
201x201	1.338608e-01

8.2 Resultados do Ajuste de Histórico

Nesta seção, serão apresentados os resultados obtidos com os métodos de otimização que foram testados neste trabalho.

A Tabela 8.5 apresenta o número de convergências de cada algoritmo, no caso de 2 parâmetros, para diferentes valores iniciais. Todos os algoritmos foram executados 10 vezes. Os métodos apresentados na tabela são algoritmo genético (AG), gradienta conjugado (CG), Quase-Newton (BFGS), gradiente (GRAD) e Levenberg-Marquardt (LM).

Tabela 8.5: Convergência para 2 parâmetros

Ponto inicial (m)	AG	GC	BFGS	GRAD	LM
$f(m) \geq 2.5\,10^{-3}$	10	3	3	2	2
$f(m) \geq 2.0\,10^{-3}$	10	4	4	4	4
$f(m) \geq 1.5\,10^{-3}$	10	6	6	3	5
$f(m) \geq 1.0\,10^{-3}$	10	7	7	4	6

Na Tabela 8.5 pode-se observar a superioridade do AG para os testes de 2 parâmetros e o desempenho inferior do método do gradiente.

Nenhum método baseado em derivadas obteve uma taxa de sucesso maior do que 70%, somente o algoritmo genético. A taxa de convergência desses métodos baseados em derivadas diminuiu à medida que os pontos iniciais se afastam do objetivo.

Na Tabela 8.6 são mostrados os resultados relativos aos dados iniciais que satisfazem $f(x) \geq 1.5 \cdot 10^{-3}$. A tabela apresenta o melhor ajuste obtido, a média e o desvio padrão dos resultados, e o total de avaliações do método.

Tabela 8.6: Estatísticas para 2 parâmetros - ponto inicial m - $f(m) \geq 1.5\,10^{-3}$

	Melhor ajuste	Média	Desvio Padrão	# Total de avaliações
AG	1.587e-09	1.488e-07	2.01e-07	3236
GC	2.869e-16	6.641e-05	9.960e-05	486
BFGS	3.401e-17	6.641e-05	9.960e-05	462
GRAD	5.446e-07	1.350e-04	1.011e-04	700
LM	3.442e-09	9.866e-05	9.832e-04	312

Na Tabela 8.6 observa-se que os melhores ajustes foram do método do gradiente conjugado e do Quasi-Newton (BFGS). No entanto a média e o desvio padrão desses

algoritmos não são tão bons. O método mais robusto neste caso foi o AG, pois é ele que possui melhor médias e desvio padrão. Em contrapartida, é o método que necessitou de maior número de avaliações.

A Tabela 8.7 apresenta os resultados obtidos com 4 parâmetros para cada algoritmo de otimização. Nas Figuras 8.9, 8.10 e 8.11 que seguem, pode-se observar graficamente a convergência desses métodos.

Tabela 8.7: Estatísticas para 4 parâmetros - $f(x) \geq 1.0 \cdot 10^{-2}$

	Melhor ajuste	Média	Desvio Padrão	# Total de avaliações
AG	1.259e-07	4.775e-06	5.478e-06	18489
GC	3.001e-07	1.702e-04	2.791e-04	2989
LM	3.935e-08	5.042e-05	4.366e-05	670

Na Tabela 8.7 observa-se que o algoritmo que conseguiu o melhor ajuste foi o LM, este método conseguiu também o menor número de avaliações. No entanto, o AG foi mais robusto para 4 parâmetros, pois sua média e seu desvio são muito melhores que as medidas dos demais.

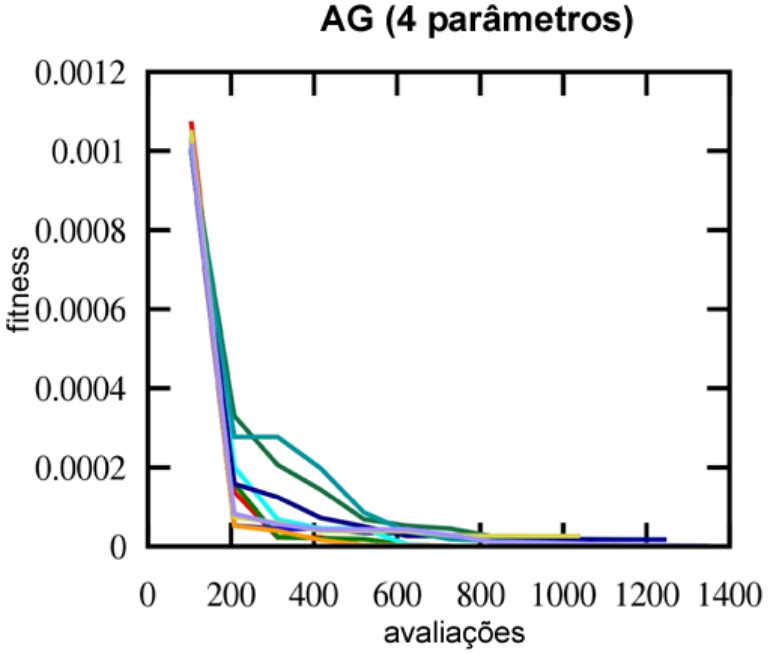

Figura 8.9: AG desempenho para 4 parâmetros

Pode-se perceber através da Figura 8.9 que realmente todas as execuções do AG convergiram, mostrando uma maior queda próximo de 200 avaliações.

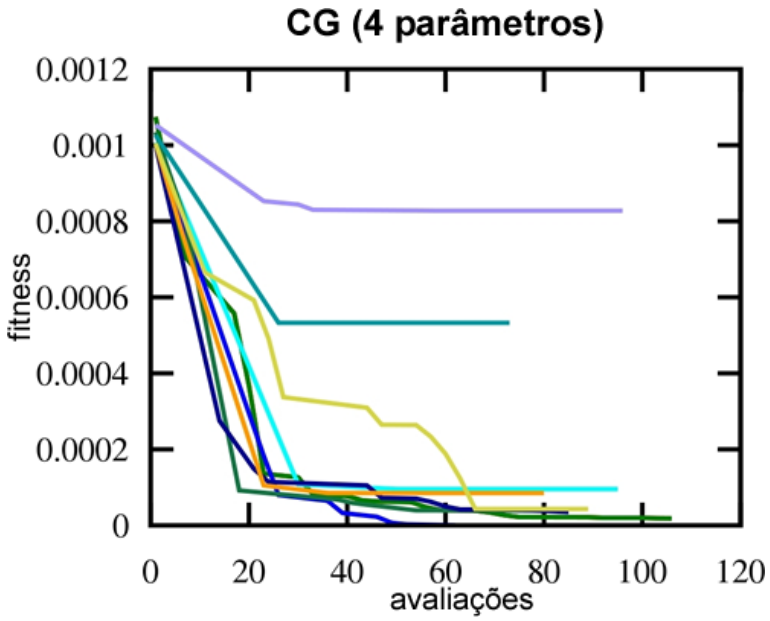

Figura 8.10: GC desempenho para 4 parâmetros

Na Figura 8.10 é facilmente notado que algumas execuções simplesmente não convergiram, algumas curvas indicam que o algoritmo parou em algum mínimo local.

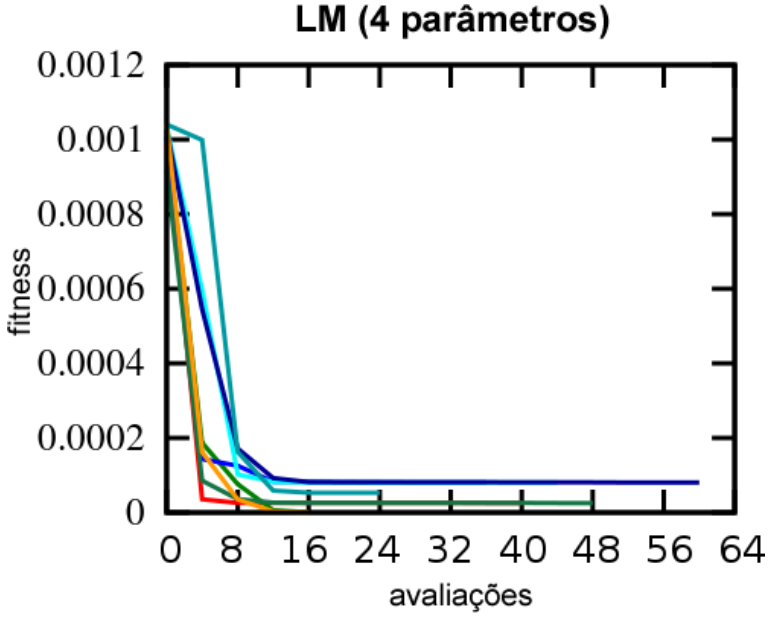

Figura 8.11: LM desempenho para 4 parâmetros

O método LM em alguns casos parece que encontrou um mínimo local, observando a Figura 8.11 podemos observar que o mínimo local encontrado pelo algoritmo esta relativamente próximo do critério de convergência.

Na Tabela 8.8 temos os resultados obtidos com 9 parâmetros para cada algoritmo de otimização. Nas Figuras 8.12, 8.13, 8.14 pode-se observar graficamente a

convergência desses métodos.

Tabela 8.8: Estatísticas para 9 parâmetros

	Melhor ajuste	Média	Desvio Padrão	# Total de avaliações
AG	9.479e-07	3.872e-06	3.053e-06	50960
GC	9.197e-07	6.135e-05	8.261e-05	3940
LM	1.865e-07	1.082e-06	1.396e-06	855

Nas estatísticas apresentadas na Tabela 8.8 pode-se notar um melhor desempenho dos métodos LM e AG, com uma vantagem do LM quanto ao número de avaliações, que foi 60 vezes menor do que do AG.

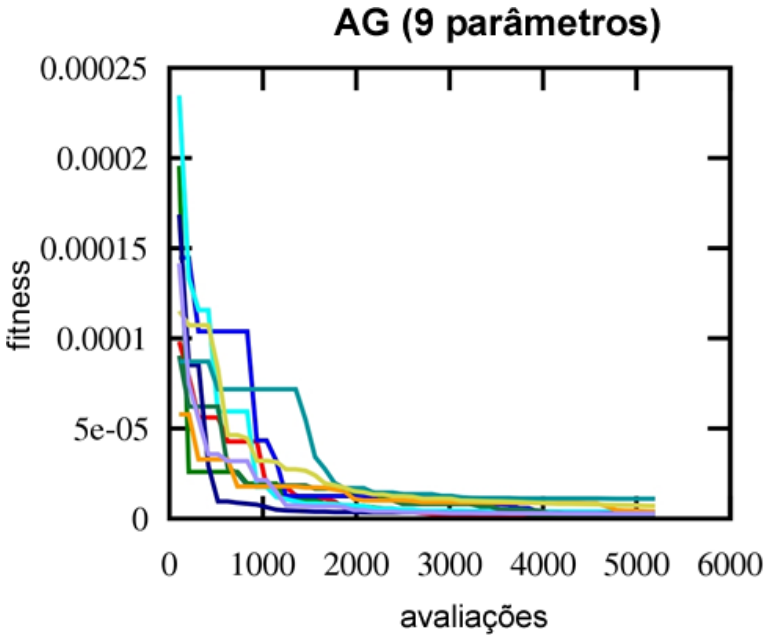

Figura 8.12: AG desempenho para 9 parâmetros

O gráfico da Figura 8.12 mostra uma boa convergência do AG novamente, no entanto existem algumas curvas que sugerem que o algoritmo se prendeu em algum mínimo local. Neste caso o melhor resultado foi do AG, convergindo com 1000 avaliações.

O GC apresentou uma convergência muito ruim para 9 parâmetros. Na Figura 8.13 vemos várias execuções que encontraram um mínimo local ou estagnaram logo no começo da execução, prejudicando o desempenho do método de um modo geral.

O LM apresentou uma boa convergência e seu melhor resultado convergiu com menos de 100 avaliações.

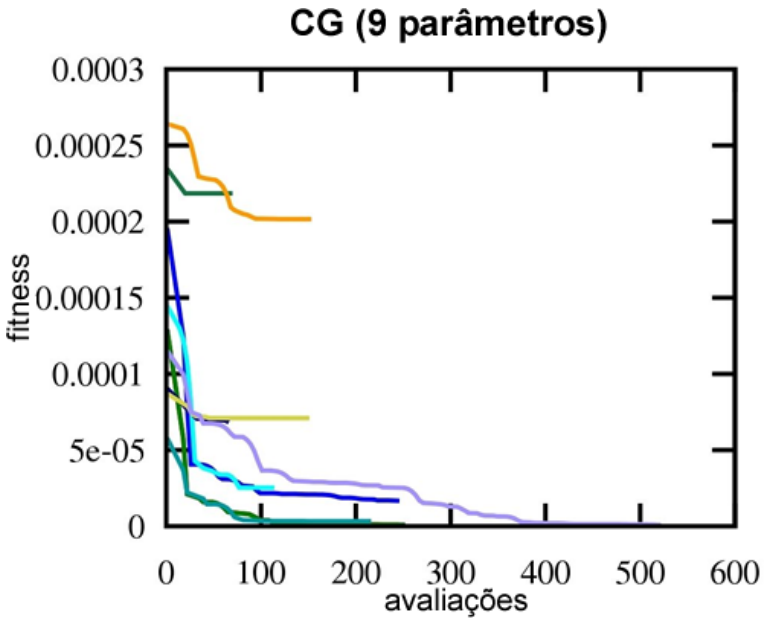

Figura 8.13: GC desempenho para 9 parâmetros

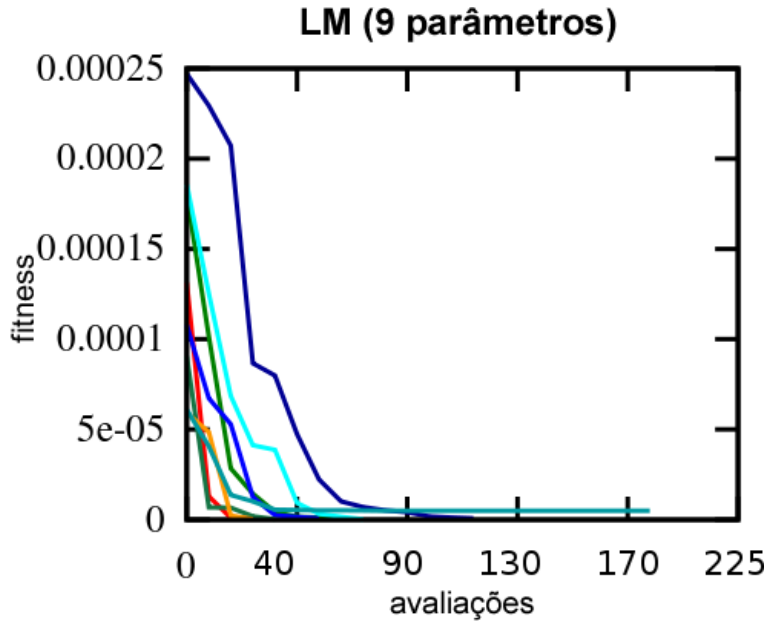

Figura 8.14: LM desempenho para 9 parâmetros

8.3 Resultados Paralelos

8.3.1 Simulador Paralelo

A Tabela 8.9 apresenta os tempo observados para a execução do simulador paralelo para a malha 101x101 em segundos, e a Figura 8.15 apresenta o *speedup* calculado para os tempos de execução obtidos.

Tabela 8.9: Tempo de execução para a malha 101x101 (em segundos)

núcleos_ máquinas	1_1	2_1	2_2	4_1	4_2	4_4	8_2	8_4	16_4
Saturação	0,05	0,02	0,03	0,01	0,02	0,02	0,01	0,02	0,02
Pressão	0,76	0,5	2,71	0,32	1,74	1,03	1,75	1,87	1,85
Total	0,85	0,55	2,77	0,35	1,78	1,07	1,78	1,91	1,89

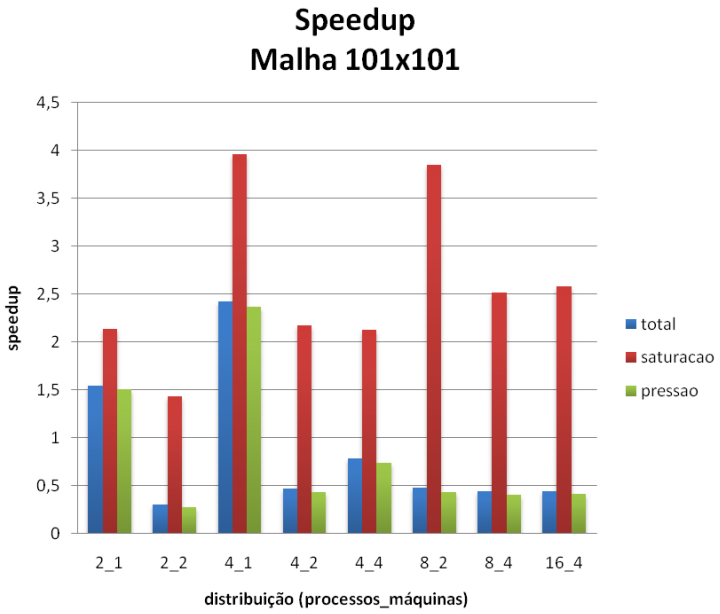

Figura 8.15: Speedup 101x101

Pela Tabela 8.9 e pela Figura 8.15 percebe-se que o algoritmo paralelo não é eficiente para a malha 101x101. Neste caso, só houve *speedup* em dois casos. 2_-1, ou seja, dois processos na mesma máquina e 4_1, o que sugere que a perda de desempenho em malhas se dá devido a um *overhead* de comunicação remota.

O tempo gasto na pressão é muito maior do que o tempo gasto na saturação, e o *speedup* do saturação é muito melhor que o da pressão, o que afeta diretamente no ganho do simulador.

A Tabela 8.10 apresenta os tempo observados para a execução do simulador paralelo para a malha 401x401 em segundos, e a Figura 8.16 apresenta o *speedup* calculado para os tempos de execução obtidos.

Tabela 8.10: Tempo de execução para a malha 401x401 (em segundos)

núcleos_ máquinas	1_1	2_1	2_2	4_1	4_2	4_4	8_2	8_4	16_4
Saturação	4,99	2,5	2,26	1,67	1,31	1,32	0,89	0,7	0,48
Pressão	101,19	69,58	68,85	67,93	43,6	41,77	36,65	24,35	22,39
Total	109,17	73,73	72,62	70,48	45,73	43,9	38,01	25,48	23,12

Pela Tabela 8.10 e pela Figura 8.16 percebe-se que o algoritmo paralelo é eficiente para a malha 400x400. Neste caso, houve *speedup* em todos casos.

Para a malha 400x400 as execuções com 4 processos tiveram melhor ganho quando ocupavam 4 máquinas, seguida pela que ocupava 2 e só então a que ocupava somente 1 das máquinas. O motivo deste resultado pode ser uma possível contenção de memória, que supera o *overhead* de comunicação remota.

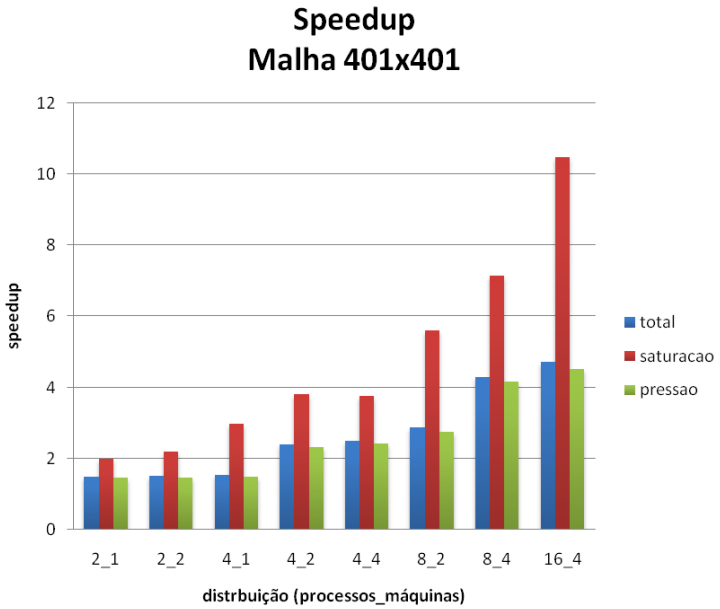

Figura 8.16: Speedup 401x401

A Tabela 8.11 apresenta os tempo observados para a execução do simulador paralelo para a malha 1001x1001 em segundos, e a Figura 8.17 apresenta o *speedup* calculado para os tempos de execução obtidos.

Na malha 1000x1000 de todas as execuções utilizando 4 processos, a melhor

Tabela 8.11: Tempo de execução para a malha 1001x1001 (em segundos)

núcleos_ máquinas	1_1	2_1	2_2	4_1	4_2	4_4	8_2	8_4	16_4
Saturação	50,56	25,67	22,89	15,89	11,88	11,96	8,04	6,11	4,22
Pressão	926,09	699,04	555,22	766,12	412,7	425,94	393,47	218,38	199,59
Total	1002,24	738,4	590,98	791,23	432,51	445,85	406,25	228,53	206,17

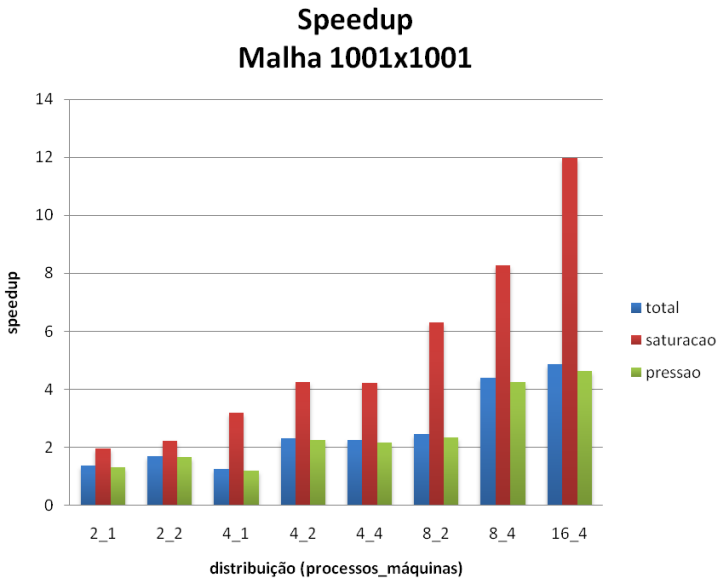

Figura 8.17: Speedup 1001x1001

configuração foi a que usou 2 máquinas, e não 4. Provavelmente, o *ovehead* da comunicação remota utilizando 4 máquinas superou a contenção de memória das execuções que utilizaram 2 máquinas. No entanto, no caso da execução que utilizou uma só máquina, uma possível maior contenção de memória fez com que o desempenho fosse bem inferior.

Para este caso é interessante notar o aumento do *speedup* da saturação, que de 2 a 8 processos apresento um *speedup* super-linear, o que significa que esta medida foi superior ao número de processadores utilizados.

8.4 Testes AG paralelo

A Tabela 8.12 apresenta os resultados do algoritmo genético executado em 17 núcleos,sendo 1 mestre e 16 escravos.

Os dados da Tabela 8.12 mostra que para malhas pequenas, a exemplo da

Tabela 8.12: Tempo em segundos do AG Paralelo com 17 núcleos

núcleos_ máquinas	1_1	2_1	2_2	4_1	4_2	4_4	8_2	8_4	16_4
101x101	3,95	6,76	6,66	11,03	11,19	11,13	24,33	25,25	69,91
401x401	567,19	599,64	599,83	627,03	626,95	627,75	583,12	586,1	720,49

101x101, não se deve utilizar a paralelização do simulador, a melhor execução é obtido com o simulador sequencial.

Para malhas maiores, como a 401x401 os resultados mostraram que algumas configurações em dois níveis são boas, mas o melhor tempo de execução ainda foi da configuração que usou o simulador sequencial. As execuções 8_2 e 8_4 foram as melhores configurações.

Capítulo 9

Discussão

Neste capítulo serão discutidos todos os resultados obtidos neste trabalho.

9.1 Validação do Simulador

Vimos no capítulo anterior que os erros obtidos na comparação dos resultados fornecidos através de métodos analíticos e resultados gerados pelo simulador implementado neste trabalho são muito pequenos, validando o simulador. Esses resultados nos dão a segurança da utilização do simulador como ferramenta para resolver e explorar outros problemas da Engenharia de Reservatórios que exijam um simulador numérico.

No problema de Buckley-Leverett, brevemente descrito na Seção 7.1.1, os gráficos que apresentam as curvas de saturação analítica e numérica ao longo do tempo estão visualmente muito próximas, o que é confirmado pelos erros relativos apresentados na Tabela 8.1, que variam pouco mais de 2% para uma malha mais grosseira, a 0.5% para a malha mais refinada.

No problema 2-spot, podemos observar que a curva de pressões ao longo do tempo da simulação numérica foi muito similar à da solução analítica, o que pode ser confirmado pelos dados da Tabela 8.2, que apresenta erros menores do que 0.3% para as malhas mais grosseiras.

Outro problema também testado para esta validação foi o 5-spot, que não possui solução analítica para a saturação. Foi utilizada como "solução analítica"uma malha mais refinada (401x401), e os resultados de pressão e saturação foram comparados

com os das demais malhas, mais grosseiras.

Todos estes testes para a validação do simulador mostraram, além de uma boa aproximação entre o modelo numérico e o analítico, a importância e necessidade do uso de malhas refinadas.

9.2 Simulador Paralelo

Para a validação do modelo paralelo, foram comparadas a saturação e a pressão ao longo do tempo calculadas no simulador paralelo com os valores obtidos no simulador sequencial validado. Os resultados obtidos, como se esperava, foram idênticos.

Em relação à eficiência do paralelismo do modelo, podemos destacar a grande escalabilidade da resolução da saturação na maioria dos casos, enquanto a resolução da pressão apresenta resultados piores. Os piores resultados de escalabilidade da resolução da pressão se devem à grande dependência de dados existente na solução dos sistemas de equações lineares. Já os bons resultados obtidos na resolução da saturação são explicados pela paralelização direta da equação explícita.

Para malhas menores, como a 101x101, a execução paralela não trouxe melhora de tempo na maioria dos casos. Os únicos casos em que houve *speedup* foram os de 2 processos na mesma máquina e o de 4 processos na mesma máquina. O provável motivo desse desempenho ruim é um grande *overhead* de comunicação, pois os casos que não utilizaram a rede para troca de dados foram os únicos que obtiveram algum ganho com a execução paralela.

Observando somente o cálculo da saturação, pode-se notar que sempre houve algum ganho. No entanto, quando há mais comunicação via rede, ou seja, quando há mais de uma máquina envolvida, o desempenho diminui significativamente.

Para a malha 401x401 podemos observar que o *speedup* aumenta sempre com o aumento do número de processadores. No entanto, as configurações que ocuparam uma só máquina, ao contrário da malha 101x101, obtiveram piores resultados do que as que ocuparam máquinas diferentes.

Pode-se notar ainda que as execuções com 4 processos tiveram melhor ganho quando ocupavam 4 máquinas, seguida pela que ocupava 2 e só então a que ocupava somente 1 das máquinas. O motivo deste resultado pode ser uma possível contenção

de memória, que supera o *overhead* de comunicação remota. Afinal, quanto maior o número de processos na mesma máquina, maior a contenção por memória, pois o processador *quadcore* em questão possui apenas um barramento conectando os 4 núcleos à memória.

O mesmo ocorreu com as execuções com 8 processos, onde o melhor tempo foi obtido com a execução de 8 processos em 4 máquinas.

Nas malhas 1001x1001 o comportamento observado foi diferente. Para execução utilizando 4 processos, a melhor configuração foi a que usou 2 máquinas, e não 4. Provavelmente, o *overhead* da comunicação utilizando 4 máquinas superou a contenção de memória das execuções que utilizaram 2 máquinas. No entanto, no caso da execução que utilizou uma só máquina, uma possível e maior contenção de memória fez com que o desempenho fosse bem inferior.

Na execução com 8 processos, a utilização de 4 máquinas foi visivelmente melhor do que a utilização de 2 máquinas, fazendo que o *speedup* fosse quase o dobro. Isso reforça a suspeita de que a causa do mau desempenho quando se ocupa os quatro núcleos de uma mesma máquina seja a contenção de memória.

Na execução com 16 processos, o ganho foi muito pequeno em todas as execuções estudadas. Novamente, uma possível causa é a contenção de memória, pois nesse caso todos os núcleos das quatro máquinas estão ocupados.

A queda no desempenho do paralelismo causada possivelmente por contenção de memória, como foi observado muitas vezes nos resultados das execuções paralelas, nos motivou a estudar com mais detalhes esse problema. Alguns testes adicionais foram feitos com o objetivo de confirmar a existência da contenção no ambiente paralelo utilizado. Os resultados destes testes serão discutidos posteriormente, ainda neste capítulo.

9.3 Comparação dos métodos de otimização

Na comparação dos métodos de otimização para o ajuste de 2 parâmetros, podemos observar que o algoritmo genético é mais robusto em relação à convergência, mas necessita de mais avaliações da função objetivo do que os demais.

Os métodos GC, LM e o QN (BFGS) obtiveram resultados bastante semelhan-

tes. Estes métodos apresentaram uma dependência da distância entre ponto inicial e o objetivo, funcionando melhor para pontos mais próximos do objetivo, porém o número de avaliações necessárias é significativamente menor, principalmente no método LM, que obteve o menor número de avaliações.

O método do gradiente foi claramente inferior aos demais, obtendo uma taxa de convergência baixa, o que é uma característica do método.

A partir desses resultados, foram excluídos dois métodos, o método do gradiente e o BFGS. O primeiro foi excluído por ter a taxa de convergência muito baixa e o segundo por obter resultados muito similares ao do método do gradiente conjugado.

Nos resultados de 4 parâmetros todos o métodos foram bem parecidos quanto ao melhor ajuste, mas o AG continuou se mostrando mais robusto, com a menor média e desvio. No entanto, o número de avaliações continuou sendo o ponto fraco deste método. O AG também foi o método que mais vezes convergiu, 6 das 10 execuções realizadas, contra apenas 1 do GC e 3 do LM.

O método LM se mostrou ainda mais eficiente quanto ao número de avaliações, realizando 26 vezes menos avaliação do que o AG e 4 vezes menos do que o GC.

Para os testes com 9 parâmetros, o melhor ajuste foi semelhante para os três métodos. Os valores de média e desvio padrão mostram uma superioridade dos métodos AG e LM em relação ao GC. Ao observar as Figuras 8.12, 8.13 e 8.14 pode-se ver que o GC estagnou em várias execuções com valores de $f(x)$ grandes.

Portanto, observamos que o AG e o LM foram os métodos que obtiveram os melhores resultados. O AG se mostrou mais robusto, com uma dependência menor em relação aos pontos iniciais, enquanto o LM se apresentou como o mais rápido, convergindo com poucas avaliações.

9.4 Teste do ambiente

Nesta seção serão apresentados e discutidos alguns testes realizados no ambiente descrito na Seção 7.4 para permitir o melhor entendimento dos resultados nele obtidos.

Para verificar se o uso da memória por vários núcleos concorrentemente prejudica o desempenho da aplicação, o que foi observado como tendência na Seção 9.2, foi

realizado um teste que executa simulações sequenciais do IMPES em um mesmo computador ao mesmo tempo.

Primeiramente, uma única instância do IMPES foi executada; em seguida 2 simultaneamente; após, 3 e 4 execuções foram feitas, também simultaneamente.

A Tabela 9.1 mostra as medidas de tempo dessas execuções concorrentes.

Tabela 9.1: Execução simultânea - IMPES

Número de instâncias	1	2	3	4
101x101	0,29	0,29	0,36	0,43
401x401	32	43	63	86
1001x1001	246	315	438	616

Os números da Tabela 9.1 sugerem que, de fato, se todos os núcleos do processador estiverem executando tarefas simultaneamente, o desempenho da aplicação pode ser afetado.

Para descartar a hipótese de que exista algum problema na implementação do simulador, foram implementados e executados dois testes sequenciais, ambos bem simples.

O primeiro está representado pelo algoritmo 8, e este código, assim como foi feito com o IMPES, foi executado uma única instância na máquina, com 2 instâncias simultaneamente, 3 instâncias simultaneamente e 4 instâncias simultaneamente.

```
1 inicio
2     double k;
3     for (i = 0; i < 1000000; i++) do
4         for (j = 0; j < 1000000; j++) do
5             k = 1/i + 1/j;
6         end
7     end
8 fim
```

Algoritmo 8: Algoritmo sem acesso à memória

Este algoritmo não faz acesso à memória; e como pode ser notado na Tabela 9.2 não há nenhuma perda significativa de desempenho com o aumento de instâncias por máquina.

Tabela 9.2: Execução simultânea - *loop* sem acesso à memória

Número de instâncias	1	2	3	4
loop	65,46	65,47	65,50	65,49

O segundo algoritmo está representado pelo código 9, executado da maneira como foi feito com o IMPES e com o teste anterior.

```
1 inicio
2     float k[1000][1000];
3     for (i = 0; i < 1000; i++) do
4         for (j = 0; j < 1000; j++) do
5             k[i * j/10][j] = (float)(i + 1.3)/(j + 1.5) + (i + 1.7) * (j + 1.9);
6         end
7     end
8 fim
```

Algoritmo 9: Algoritmo com acesso à memória

Este código, diferentemente do 8 faz acesso à memória a cada passagem pelo *loop*. Como pode ser notado na Tabela 9.3, os tempos de execução aumentam à medida em que mais execuções ocorrem simultaneamente, confirmando assim nossas suspeitas de que a contenção de memória é a principal responsável pela perda de desempenho de nosso simulador.

Tabela 9.3: Execução simultânea - *loop* com acesso à memória

Número de instâncias	1	2	3	4
loop	32,95	47,82	70,35	105,05

Como o simulador é uma aplicação que faz constantes acessos à memória, de acordo com os números vistos nos *benchmarks*, o ambiente utilizado não favoreceu os testes feitos com o simulador.

9.5 AG Paralelo

Pelos resultados obtidos com a malha 101x101, observa-se à primeira vista que para malhas de pequeno porte a implementação do AG que utiliza o simulador paralelizado não apresentou resultados satisfatórios. Isto é um resultado que já podíamos esperar, pois o simulador paralelo foi avaliado na Seção 9.2 não apresentou *speedup* significativos.

Os resultados do AG para a malha 401x401 mostraram também resultados melhores para a avaliação do simulador sequencial, no entanto os números das execuções com 8 máquinas apontam uma possível melhora para malha maiores.

Para malhas que não cabem na memória principal e teriam que utilizar a região de *swap* para executar a abordagem que utiliza o simulador paralelo é uma boa alternativa.

Após a investigação do ambiente realizamos um teste com o AG usando 13 núcleos, o mestre e 12 escravos. Em cada máquina escrava foi deixado um núcleo livre.

O resultado foi bastante interessante, a execução com 12 processos para o simulador sequencial executou em $525,52s$ enquanto esta mesma execução com 17 processos executou em $667,19s$ o que confirma a suspeita de que a contenção de memória nos processadores utilizados faz com que as aplicações fiquem mais lentas com um maior número de processadores se todos eles estiverem ocupados.

Alguns estudos apresentam análises sobre a arquitetura utilizada. Estes estudos podem ser encontrados em [28], [29] [30].

9.6 Trabalhos Futuros

Como trabalhos futuros, propomos algumas modificações no AG para diminuir seu número total de avaliações. Uma dessas modificações é a utilização de outro esquema de reprodução, como o por exemplo o *steady-state*, que se mostrou mais eficiente em [31].

Para os métodos baseados em derivadas podemos utilizar o cálculo da derivada através da adjunta, o que diminuiria sensivelmente o número de vezes que o simulador é calculado, já que a adjunta exige apenas duas chamadas ao simulador [42].

Outro trabalho que podemos realizar para a tentativa de melhorar os resultados do ajuste de histórico é a implementação de um modelo híbrido, que utilizaria AG para encontrar um ponto inicial para os métodos baseados em derivadas, semelhantes aos trabalho apresentados em [32] e em [33].

Em relação às aplicações paralelas, podemos propor a continuação da análise aqui

apresentada para diferentes tipos de arquiteturas multiprocessadas, como as que são baseadas nos novos processadores que possuem um barramento exclusivo para cada núcleo de processamento [34] [35]. Esses novos processadores potencialmente reduzirão os problemas de contenção de memória observados neste trabalho.

Capítulo 10

Conclusão

Neste trabalho foi implementado um simulador bifásico incompressível (água/óleo), bidimensional. Resultados de testes executados para a validação deste simulador foram apresentados, e mostraram que o modelo foi corretamente implementado.

Devido à grande demanda computacional deste simulador, uma versão paralela foi implementada. Os resultados dos testes para esta implementação foram apresentados no Capítulo 8. Esses resultados indicaram que para malhas pequenas a paralelização não obteve resultados que justificassem sua paralelização.

Com o aumento do tamanho das malhas, pode-se perceber uma melhora significativa nos valores de *speedup*, o que indica que para essas malhas a paralelização foi útil e reduz o tempo de simulação.

Ao analisar esses resultados, vimos que estava ocorrendo uma possível contenção de memória quando se ocupa mais de um núcleo da mesma máquina. Este problema afetou o desempenho da aplicação, o que nos levou a realizar mais testes para obter mais informações sobre o ambiente de execução.

Em malhas grandes, pudemos observar um excelente escalabilidade da saturação, obtendo inclusive, *speedups* super lineares.

A seguir, abordamos o problema inverso de ajuste de histórico e comparamos diferentes métodos de otimização: algoritmo genético(AG), gradiente(GRAD), gradiente conjugado(CG), Quasi-Newton(BFGS) e Levenberg-Marquardt(LM).

Na comparação dos algoritmos de otimização, foram apresentados vários resultados interessantes. Para ajuste de 2 parâmetros, o algoritmo genético foi superior aos demais, convergindo em todas as 10 execuções realizadas, para as 4 rodadas de

teste. O ponto fraco deste algoritmo é o excessivo número de avaliações para se obter a convergência. Os métodos LM, BFGS e GC também obtiveram bons resultados e fizeram menos avaliações, mas nenhum deles convergiu mais de 7 vezes para qualquer rodada de testes.

Para os testes de 4 e 9 os algoritmos BFGS e GRAD foram excluídos, o primeiro por apresentar resultados semelhantes ao CG e o segundo por ter resultados inferiores aos demais métodos. Os resultados sugerem que o AG e o LM foram os métodos que obtiveram os melhores resultados. O AG se mostrou mais robusto, com uma dependência menor em relação aos pontos iniciais, enquanto o LM se apresentou como o mais rápido, convergindo com poucas avaliações.

Avaliamos também uma implementação paralela para o problema de ajuste de histórico. A implementação paralela obteve bons resultados de *speedup*. Porém os resultados confirmaram a suspeita de uma grande contenção de memória no ambiente de *cluster* de máquinas multiprocessadas utilizado. Por exemplo, em nosso ambiente de 16 núcleos computacionais, o melhor resultado de desempenho foi obtido utilizando apenas 12 núcleos, isto é, deixando um núcleo ocioso de cada máquina *quadcore*.

Referências Bibliográficas

[1] THOMAS, J. E., *Fundamentos de Engenharia de Petróleo*. 2nd ed. Interciência: Rio de Janeiro, 2001.

[2] KOEDERITZ, F., *Lecture Notes on Applied Reservoir Simulation*. World Scientific Publishing Company, Agosto 2005.

[3] BROWN, S., ROSE, J., "Architecture of FPGAs and CPLDs: A Tutorial", *IEEE Design and Test of Computers*, v. 13, n. 2, pp. 42–57, 1996.

[4] FOX, G., GANNON, D., "Computational Grids", *Computing in Science and Engineering*, v. 3, n. 4, pp. 74–77, 2001.

[5] HEINEMANN, Z. E., *Fluid Flow in Porous Media*, Tech. rep., Petroleum Engineering Department - University of Leoben, Leoben, Outubro 2005.

[6] AZIZ, K., SETTARI, A., *Petroleum Reservoir Simulation*. Applied Science Publishers: London, 1979.

[7] ROSA, A. J., DE SOUZA CARVALHO, R., XAVIER, J. A. D., *Engenharia de Reservatórios de Petróleo*. 1st ed. Interciência, 2006.

[8] LISBOA, É. F. A., *Uma abordagem multi-escala para o cálculo da permeabilidade longitudinal de meios porosos fibrosos randômicos*, Master's Thesis, Universidade Federal do Rio de Janeiro, Rio de Janeiro, RJ, Brasil, 2000.

[9] RAMOS, F. M., VELHO, H. F. C., CARVALHO, J. C., et al., "Novel Approaches on Entropic Regularization", *Inverse Problems*, v. 15, n. 5, pp. 1139–1148, 1999.

[10] MITCHELL, M., *An Introduction to Genetic Algorithms*. The MIT Press, Fevereiro 1998.

[11] BAZARAA, M. S., SHERALI, H. D., SHETTY, C. M., *Nonlinear Programming: Theory and Algorithms*. 3rd ed. Wiley-Interscience, Maio 2006.

[12] FLETCHER, R., *Practical Methods of Optimization*. 2nd ed. Wiley, Maio 2000.

[13] HESTENES, M. R., STIEFEL, E., "Method of Conjugate Gradients for Solving Linear Systems", *Journal of Research of the National Bureau of Standards*, v. 49, n. 6, pp. 409–436, Dezembro 1952.

[14] ALMASI, G. S., GOTTLIEB, A., *Highly Parallel Computing*. The Benjamin/Cummings Publishing Company, Inc.: Redwood City, CA, 1989.

[15] GORLACH, S. P., "Design of macropipeline algorithms and programs", *Cybernetics and Systems Analysis*, pp. 445–448, Fevereiro 2005.

[16] DUNCAN, R., "A Survey of Parallel Computer Architectures", *Computer*, v. 23, n. 2, pp. 5–16, Fevereiro 1990.

[17] FILHO, O. G. L., CARRERA, E. V., LOQUES, O. G., "Integrando Memória Compartilhada, Troca de Mensagens e Configuração". In: *Anais do X Simpósio Brasileiro de Arquitetura de Computadores*, 1998.

[18] TENENBAUM, A. S., *Sistemas Operacionais Modernos*. 3rd ed. Prentice Hall do Brasil: São Paulo, 2000.

[19] MPI-FORUM, "MPI: A Message-Passing Interface Standard", `http://www.mpi-forum.org/`, Último acesso em: 25 de Julho de 2009.

[20] SLOAN, J. D., *High Performance Linux Clusters with OSCAR, Rocks, OpenMosix, and MPI*. O'Reilly, 2004.

[21] TOMASSINI, M., "Parallel and Distributed Evolutionary Algorithms: A Review", *Evolutionary Algorithms in Engineering and Computer Science*, pp. 113–133, 1999.

[22] NOWOSTAWSKI, M., POLI, R., "Parallel Genetic Algorithm Taxonomy", *Knowledge-Based Intelligent Information Engineering Systems, 1999. Third International Conference*, pp. 88–92, Dezembro 1999.

[23] GOLUB, M., JAKOBOVIC, D., "A New Model of Global Parallel Genetic Algorithm", *Information Technology Interfaces, 2000. ITI 2000. Proceedings of the 22nd International Conference on*, pp. 363–368, Junho 2000.

[24] CANTÚ-PAZ, E., "A Survey of Parallel Genetic Algorithms", *Calculateurs Paralleles*, v. 10, 1998.

[25] LOURAKIS, M., "levmar: Levenberg-Marquardt nonlinear least squares algorithms in C/C++", `http://www.ics.forth.gr/~lourakis/levmar/`, Último acesso em: 29 de Julho de 2009.

[26] ET AL, M. G., "GSL - GNU Scientific Library", `http://www.gnu.org/software/gsl/`, Último acesso em: 28 de Julho de 2009.

[27] PACHECO, P., *Parallel Programming with MPI*. 1st ed. Morgan Kaufmann, Outubro 1996.

[28] CURTIS-MAURY, M., DING, X., ANTONOPOULOS, C. D., et al., "An Evaluation of OpenMP on Current and Emerging Multithreaded Processors". In: *Proceedings of the First International Workshop on OpenMP, Lecture Notes in Computer Science*, v. 4315, pp. 133–142, Springer, Julho 2005.

[29] ALAM, S. R., BARRETT, R. F., KUEHN, J. A., et al., "Characterization of Scientific Workloads on Systems with Multi-core Processors". In: *IEEE International Symposium on Workload Characterization*, pp. 225–236.

[30] CHAI, L., GAO, Q., PANDA, D. K., "Understanding the Impact of Multi-Core Architecture in Cluster Computing: A Case Study with Intel Dual-Core System". In: *Cluster Computing and the Grid, 2007. CCGRID 2007. Seventh IEEE International Symposium on*, pp. 471–478, 2007.

[31] VAVAK, F., FOGARTY, T. C., "Comparison of steady state and generational genetic algorithms for use in nonstationary environments". In: *IEEE International Conference on Evolutionary Computation (ICEC)*, pp. 192–195, 1996.

[32] BORGES, C., BARBOSA, H., "On the behavior of a parallel nongenerational genetic algorithm". In: *XXIV CILAMCE, 24th Iberian Latin-American Congress on Computational Methods in Engineering (CD-ROM)*, 2003.

[33] GUDLA, P., GANGULI, R., "An automated hybrid genetic-conjugate gradient algorithm for multimodal optimization problems", *Applied Mathematics and Computation*, v. 2, n. 167, pp. 1457–1474, 2005.

[34] INTEL, "Processador Intel Core i7", `http://www.intel.com/portugues/products/processor/corei7/index.htm`, Último acesso em: 27 de Julho de 2009.

[35] AMD, "Processador AMD Opteron Six-Core", `http://www.amd.com/br/products/server/six-core-opteron/Pages/six-core-opteron.aspx`, Último acesso em: 27 de Julho de 2009.

[36] VAFAI, K., *Handbook Of Porous Media*. Crc Press, Março 2005.

[37] KASSEM, J. H. A., ALI, S. M. F., ISLAM, M. R., *Petroleum Reservoir Simulation: A Basic Approach*. Gulf Publishing Company, Março 2006.

[38] TARANTOLA, A., *Inverse Problem Theory and Methods for Model Parameter Estimation*. SIAM: Society for Industrial and Applied Mathematics, Dezembro 2004.

[39] OLIVER, D. S., REYNOLDS, A. C., LIU, N., *Inverse Theory for Petroleum Reservoir Characterization and History Matching*. 1st ed. Cambridge University Press, Junho 2008.

[40] NOCEDAL, J., WRIGHT, S. J., *Numerical Optimization*. Springer, Abril 2000.

[41] BALAY, S., BUSCHELMAN, K., EIJKHOUT, V., et al., *PETSc Users Manual*, Tech. Rep. ANL-95/11 - Revision 3.0.0, Argonne National Laboratory, 2008.

[42] GONZALEZ-RODRIGUEZ, P., KINDELAN, M., MOSCOSO, M., et al., "History matching problem in reservoir engineering using the propagation–backpropagation method", *Inverse Problems*, v. 21, n. 2, pp. 565–590, 2005.

Anexo A

Bifásico IMPES

Autores: Flávio Dicktein
Paulo Goldfeld
Rodrigo Weber dos Santos

A.1 Introdução

Estamos interessados em resolver o problema do escoamento bifásico (óleo e água), incompressível 2D. Introduzimos as velocidades de escoamento

$$v_o = -\frac{K_o}{\mu_o}\nabla p_o, v_a = -\frac{K_a}{\mu_a}\nabla p_a, \tag{A.1}$$

onde os índices a e o referem-se a água e óleo, respectivamente, e

- p é a pressão;
- K é a permeabilidade efetiva;
- μ é a viscosidade;

Definimos ainda a saturação s de cada fluido e escrevemos $K_o = Kk_{ro}$, $K_a = Kk_{ra}$, onde K é a permeabilidade absoluta e k_{ro}, k_{ra} são as permeabilidades relativas, que dependem da saturação de água e óleo em cada ponto. Introduzimos ainda as mobilidades

$$\lambda_o = \frac{k_{ro}}{\mu_o}, \quad \lambda_a = \frac{k_{ra}}{\mu_a}, \tag{A.2}$$

e as transmilibilidades

$$T_o = K\lambda_o, \;\; T_a = K\lambda_a, \tag{A.3}$$

Desta forma, $v_o = -T_o \nabla p_o$, $v_a = -T_a \nabla p_a$. As equações de conservação de massa das duas fases se escrevem:

$$\begin{cases} \phi \rho_a \partial_t s_a + \nabla.(\rho_a v_a) = Q_a, \\ \phi \rho_o \partial_t s_o + \nabla.(\rho_o v_o) = Q_o, \end{cases} \tag{A.4}$$

onde

- ϕ é a porosidade do meio;
- ρ é a densidade;
- Q é a densidade de fluxo nos poços.

Vamos admitir que as densidades do óleo e da água são constantes, e que as pressões do óleo e da água são iguais. Assim, fazemos $p_a = p_o : p$. Dividimos a primeira equação de (A.4) por ρ_a, a segunda por ρ_o e somamos as duas. Usando que $s_a + s_o = 1$, obtemos

$$\begin{cases} \phi \partial_t s_a + \nabla.v_a = q_a, \\ \nabla.v_t = q_t, \end{cases} \tag{A.5}$$

onde

- $v_t = -T_t \nabla p$ é a velocidade total;
- $T_t = T_a + T_o$ é a transmissibilidade total;
- $q_a = \dfrac{Q_a}{\rho_a}$ é a densidade de vazão de água;
- $q_o = \dfrac{Q_o}{\rho_o}$ é a densidade de vazão de óleo;
- $q_t = q_a + q_o$ é a densidade de vazão total;

Escrevemos s para a saturação da água, omitindo subscritos. Definindo a mobilidade total $\lambda_t = \lambda_a + \lambda_o$, introduzimos o **fluxo fracionário**

$$f(s) = \frac{T_a}{T_t} = \frac{\lambda_a}{\lambda_t}. \tag{A.6}$$

Reescrevemos (A.5) como

$$\begin{cases} \phi\partial_t s - \nabla.(f(s)v) = q_a, \\ \nabla.v_t = q_t, \end{cases} \tag{A.7}$$

Supomos que são conhecidas as funções $k_{ra}(s)$, $k_{ro}(s)$, $\phi(x,y)$ e $K(x,y)$. A resolução de (A.7) será feita pelo método IMPES, que descrevemos a seguir.

A.2 Discretização Temporal

A resolução numérica será feita usando dois passos de tempo, $(\Delta t)_n$ para a pressão e $(\Delta t)_{n,\ell}$ para a saturação. Uma pressão p^n corresponde ao instante $t^n = \sum_{1\leq i\leq n}(\Delta t)_n$, enquanto que a saturação $s^{n,\ell}$ corresponde ao instante $t^{n,\ell} = t^n + \sum_{j\leq\ell}(\Delta t)_{n,j}$, para $n = 0,\ldots,N-1$, $\ell = 1,\ldots,L$. Definimos $s^{n,0} = s^{n-1,L}$ e t^0 corresponde a $t = 0$. Aproximamos de $\partial_t s$ em $t^{n,\ell}$ por

$$\delta_t s^{n,\ell} = \frac{s^{n,\ell+1} - s^{n,\ell}}{(\Delta t)_{n,\ell}}, \quad \ell = 0,1,\ldots L-1. \tag{A.8}$$

A.3 Discretização Espacial

Consideramos o reservatório Ω como um retângulo de dimensões $M_x \times M_y$. Dividimos os lados de Ω em N_x e N_y partes, respectivamente. Sejam $\Delta x = M_x/N_x$, $\Delta y = M_y/N_y$. Definimos retângulos V_{ij} centrados em $C_{ij} = (x_i, y_j)$ onde $x_i = (i+1/2)\Delta x$ e $y_j = (j+1/2)\Delta y$, $0 \leq i < N_x$, $0 \leq j < N_y$. Denotamos ainda $x_{i+1/2} = (x_i + x_{i+1})/2$, e definimos analogamente $y_{j+1/2}$. Assim, inteiros i, j dizem respeito aos centros V_{ij} ao passo que índices $i+1/2$ e $j+1/2$ estão relacionados às interfaces.

Seja F uma função definida nas interfaces $(i+1/2, j)$ e $(i, j+1/2)$. Introduzimos o divergente discreto ∇_h. dado por

$$\nabla_h.F_{i,j} = \frac{F_{i+1/2,j} - F_{i-1/2,j}}{\Delta x} + \frac{F_{i,j+1/2} - F_{i,j-1/2}}{\Delta y}. \tag{A.9}$$

Para f definida nos centros (i,j) dos retângulos, definimos o gradiente discreto

∇_h como uma funçao de interface dada por

$$\nabla_h f_{i+1/2,j} = \frac{f_{i+1,j} - f_{i-1,j}}{\Delta x}, \qquad \nabla_h f_{i,j+1/2} = \frac{f_{i,j+1} - f_{i,j-1}}{\Delta y}. \tag{A.10}$$

A.4 O Esquema IMPES

O esquema IMPES consiste nas seguintes etapas. Para $n = 0, 1, \ldots, N$,

1. Dado s^n, obtenha p^n como solução de

$$-\nabla_h.(T_t(s^n)\nabla_h p^n) = q_t^n. \tag{A.11}$$

2. Faça $v^n = T_t(s^n)\nabla_h p^n$.

3. Conhecidos p^n e $s^n = s^{n,0}$ determine $s^{n,\ell}$ para $\ell = 0, 1, \ldots, L-1$ resolvendo

$$\phi\delta_t s^{n,\ell} = \nabla_h.(f(s^{n,\ell})v^n) + q_a^{n,\ell}. \tag{A.12}$$

4. Faça $s^{n+1} = s^{n,L}$.

Observe que precisamos calcular transmissibilidades T em interfaces de V_{ij}. Assumimos que as viscosidades são constantes. Além disso, estimamos a permeabilidade absoluta usando médias harmônicas,

$$K_{i+1/2,j} = \frac{2K_{i,j}K_{i+1,j}}{K_{i,j} + K_{i+1,j}}, \qquad K_{i,j+1/2} = \frac{2K_{i,j}K_{i,j+1}}{K_{i,j} + K_{i,j+1}}. \tag{A.13}$$

Por fim, usaremos o esquema a montante (upwind) para as permeabilidades relativas: se $p_{i+1,j} > p_{i,j}$ então $(k_r)_{i+1/2,j} = k_r(s_{i+1,j})$. Tratamento análogo é feito nas outras interfaces.

Falta discutir o tratamento da fronteira e como calcular as vazões nos poços.

A.5 Condição de Fronteira

Iremos assumir que o reservatório está isolado, o que significa que v_a, v_o são nulos nos bordos de Ω. Basta, portanto, anular $T_{i+1/2,j}$ e $T_{i,j+1/2}$ nas interfaces que pertençam

aos bordos.

A.6 Tratamento dos Poços

Nos poços injetores, a vazão total q_t é igual à vazão de água q_a, que é prescrita.[1] Nos poços produtores, q_t é conhecido. Calculamos então q_a como uma fração da vazão total.

$$(q_a)_{i,j}^{n,\ell} = f(s^{n,\ell})q_t^n, \tag{A.14}$$

veja (A.6).

A.7 Resolução da Equação da Pressão

Trata-se de um sistema linear simétrico. Usamos um método de gradientes conjugados precondicionado do PETSc.

A.8 Os Passos de Tempo

Escolhemos $(\Delta t)_{n,\ell}$ de modo a respeitar a condição de CFL, que discutimos a seguir. Seja $V_{i,j}$ um bloco da malha de discretização e considere o caso em que a velocidade total v é positiva em todas as interfaces. De acordo com o esquema a montante, a equação (A.12) se escreve

$$s_{i,j}^{n,\ell+1} - s_{i,j}^{n,\ell} + \frac{(\Delta t)_{n,\ell}}{\phi \Delta x}(f(s_{i,j}^{n,\ell})v_{i+1/2,j}^n - f(s_{i-1,j}^{n,\ell})v_{i-1/2,j}^n) +$$

$$\frac{(\Delta t)_{n,\ell}}{\phi \Delta y}(f(s_{i,j}^{n,\ell})v_{i,j+1/2}^n - f(s_{i,j-1}^{n,\ell})v_{i,j-1/2}^n) = \frac{(\Delta t)_{n,\ell}}{\phi} q_a^{n,\ell}.$$

Mas $\nabla_h . v = q_t$, isto é.

$$\frac{1}{\Delta x}(v_{i+1/2,j}^n - v_{i-1/2,j}^n) + \frac{1}{\Delta y}(v_{i,j+1/2}^n - v_{i,j-1/2}^n) = q_t^n.$$

[1] O que o usuário prescreve são vazões. Para se obter densidades, divide-se a vazão dada pelo volume do bloco $\Delta x \Delta y h$, onde h é a altura do reservatório

Deduzimos que

$$s_{i,j}^{n,\ell+1} - s_{i,j}^{n,\ell} + \frac{(\Delta t)_{n,\ell}}{\phi \Delta x}(f(s_{i,j}^{n,\ell}) - f(s_{i-1,j}^{n,\ell}))v_{i-1/2,j}^{n} +$$

$$\frac{(\Delta t)_{n,\ell}}{\phi \Delta y}(f(s_{i,j}^{n,\ell}) - f(s_{i,j-1}^{n,\ell}))v_{i,j-1/2}^{n} = \frac{(\Delta t)_{n,\ell}}{\phi}(q_a^{n,\ell} - f(s_{i,j}^{n,\ell})q_t^n).$$

Começamos por considerar o caso em que o lado direito da equação acima é nulo. Isto ocorre nos blocos sem poços, mas também nos blocos com poços produtores, veja (A.14).

Neste caso, usando o Teorema do Valor Médio, escrevemos

$$s_{i,j}^{n,\ell+1} = s_{i,j}^{n,\ell}(1 - \frac{(\Delta t)_{n,\ell}}{\phi \Delta x} f'(s_1)v_{i-1/2,j}^{n} - \frac{(\Delta t)_{n,\ell}}{\phi \Delta y} f'(s_2)v_{i,j-1/2}^{n}) +$$

$$\frac{(\Delta t)_{n,\ell}}{\phi \Delta x} f'(s_1)v_{i-1/2,j}^{n} s_{i-1,j}^{n,\ell} + \frac{(\Delta t)_{n,\ell}}{\phi \Delta y} f'(s_2)v_{i,j-1/2}^{n} s_{i,j-1}^{n,\ell},$$

para algum s_1, s_2. Assim, se

$$1 - \frac{(\Delta t)_{n,\ell}}{\phi \Delta x} f'(s_1)v_{i-1/2,j}^{n} - \frac{(\Delta t)_{n,\ell}}{\phi \Delta y} f'(s_2)v_{i,j-1/2}^{n} \geq 0, \tag{A.15}$$

garantimos que

$$\min\{s_{i-1,j}^{n,\ell}, s_{i,j}^{n,\ell}, s_{i,j-1}^{n,\ell}\} < s_{i,j}^{n,\ell+1} < \max\{s_{i-1,j}^{n,\ell}, s_{i,j}^{n,\ell}, s_{i,j-1}^{n,\ell}\}.$$

Nos blocos onde as velocidades interfaciais não são todas positivas, a condição (A.15) deve ser modificada para

$$\sum_m \frac{(\Delta t)_{n,\ell}}{\phi \Delta_m} f'(s_m)|v_m^n| \leq 1, \tag{A.16}$$

onde o índice m indica as interfaces com fluxo entrante, Δ_m é igual a Δx em faces verticais, e igual a Δy em faces horizontais. Além disso, s_m está situado entre o menor e o maior valor das saturações no bloco e seus vizinhos, no instante associado a (n, ℓ). Na implementação do esquema, mudamos (A.16) para

$$\max f'(s_m) \sum_m \frac{(\Delta t)_{n,\ell}}{\phi \Delta_m}|v_m^n| \leq \rho_1, \tag{A.17}$$

onde $0 < \rho_1 < 1$ é um parâmetro a escolher e o máximo é tomado entre a menor e a maior saturação de cada bloco e seus vizinhos cuja interface seja de fluxo entrante no bloco.

Falta considerar blocos com poço injetor. É razoável admitir que não haja fluxo entrando nestes blocos. Neste caso, lembrando que $q_a = q_t$, obtemos

$$s_{i,j}^{n,\ell+1} - s_{i,j}^{n,\ell} = \frac{(\Delta t)_{n,\ell}}{\phi} q_a^{n,\ell}(1 - f(s_{i,j}^{n,\ell})).$$

Escolhemos $(\Delta t)_{n,\ell}$ de modo a garantir que

$$s_{i,j}^{n,\ell} + \rho_2 \frac{(\Delta t)_{n,\ell}}{\phi}(q_a^{n,\ell}(1 - f(s_{i,j}^{n,\ell})) < 1 - s_{o,res}, \tag{A.18}$$

onde $\rho_2 > 1$ é um parâmetro a escolher e $s_{o,res}$ é a saturação residual do óleo.

Resumindo, escolhemos Δt de modo a satisfazer (A.17) e (A.18). Deste modo, garantimos que em cada instante a saturação da água e do óleo permanecem maiores que as respectivas saturações mínimas admissíveis (saturações residuais). Isto assegura, em particular, a estabilidade do esquema explícito.

Além disso, controlamos o passo de tempo da pressão $(\Delta t)_n$ da seguinte forma. Calculamos a variação percentual da pressão

$$VP^n = \frac{\|p^{n+1} - p^n\|_\infty}{\|p^n\|_\infty}.$$

Caso VP^n ultrapasse VP_{max}, uma variação máxima estabelecida, diminuímos o passo de tempo da pressão na próxima etapa, fazendo $(\Delta t)_{n+1} = \alpha(\Delta t)_n$, onde $\alpha < 1$ é dado. Da mesma forma, podemos aumentar $(\Delta t)_{n+1} = \beta(\Delta t)_n$, onde $\beta > 1$, caso VP^n seja menor que uma variação mínima estabelecida.

Note que, para compatibilizar os passos de tempo da saturação e da pressão, precisamos que $(\Delta t)_n = (\Delta t)_{n,1} + \cdots + (\Delta t)_{n,L}$. Para isso, basta ajustar o último passo de tempo da saturação $(\Delta t)_{n,L}$.