

UNIVERSIDADE ESTADUAL DO NORTE FLUMINENSE
LABORATÓRIO DE ENGENHARIA E EXPLORAÇÃO DE PETRÓLEO

PROJETO ENGENHARIA
DESENVOLVIMENTO DO SOFTWARE
MODELOS DE DESLOCAMENTO IMISCÍVEL PARA RECUPERAÇÃO
SECUNDÁRIA DE PETRÓLEO
TRABALHO DA DISCIPLINA PROGRAMAÇÃO PRÁTICA

DAVID HENRIQUE LIMA DIAS
JULIA RANGEL RIBEIRO
MARCOS VINÍCIUS DE PAULA CHAIBEN
Prof. André Duarte Bueno

MACAÉ - RJ
DEZEMBRO - 2022

Sumário

1	Introdução	1
1.1	Escopo do problema	1
1.2	Objetivos	1
2	Especificação	3
2.1	Especificação do Software - Requisitos	3
2.1.1	Nome do sistema/produto	3
2.1.2	Especificação	4
2.1.3	Requisitos funcionais	4
2.1.4	Requisitos não funcionais	5
2.2	Casos de uso do software	5
2.2.1	Diagrama de caso de uso geral	6
2.2.2	Diagrama de caso de uso específico	6
3	Elaboração	8
3.1	Análise de domínio	8
3.2	Conceitos Fundamentais	10
3.2.1	Conceitos de Propriedades de Rochas e Fluidos	10
3.2.2	Conceitos Teóricos:	14
3.3	Formulações Matemáticas	20
3.3.1	Modelo de Fluxo Bifásido (1D)	20
3.3.2	Modelo de Fluxo Bifásido Areal (2D)	21
3.3.3	Modelo de Fluxo Bifásido em Sistemas Estratificados (3D)	27
3.4	Diagrama de Pacotes – assuntos	29
4	AOO – Análise Orientada a Objeto	31
4.1	Diagramas de classes	31
4.1.1	Dicionário de classes	33
4.2	Diagrama de seqüência – eventos e mensagens	34
4.2.1	Diagrama de sequência geral	34
4.3	Diagrama de comunicação – colaboração	35
4.4	Diagrama de máquina de estado	36

4.5	Diagrama de atividades	37
5	Projeto	41
5.1	Projeto do sistema	41
5.2	Projeto orientado a objeto – POO	42
5.3	Diagrama de componentes	43
5.4	Diagrama de implantação	44
6	Implementação	46
6.1	Código fonte	46
7	Teste	107
7.1	Teste: Geral	107
7.2	Teste: Modelo Styles com configuração de poços - 1 injetor e 1 produtor . .	108
7.3	Teste: Modelo Styles com configuração de poços - 1 produtor e 2 injetores	109
7.4	Teste: Modelo Styles com configuração de poços - 2 produtores e 2 injetor	110
7.5	Teste: Modelo Dijkstra com configuração de poços - 1 produtor e 1 injetor .	111
7.6	Teste: Modelo Dijkstra com configuração de poços - 1 produtor e 2 injetores	112
7.7	Teste: Modelo Dykstra com configuração de poços - 2 produtores e 1 injetor	113
8	Documentação	115
8.1	Documentação do usuário	115
8.1.1	Como instalar o software	115
8.1.2	Como rodar o software	115
8.2	Documentação para desenvolvedor	115
8.2.1	Dependências	116
8.2.2	Como gerar a documentação usando doxygen	116
9	Referências	121

Capítulo 1

Introdução

Impulsionado pela importância que o petróleo tem sobre toda a humanidade, sendo ainda hoje uma das maiores fonte de energia em uso pelo ser humano, o presente trabalho de engenharia, desenvolve-se um projeto computacional em linguagem orientada a objeto C++ que tem como principal objetivo o gerenciamento de informações e realização de cálculos para estudo da recuperação de óleo resultante do deslocamento por um fluido imiscível.

1.1 Escopo do problema

No início de sua descoberta, os reservatórios de óleo e gás possuem uma certa quantidade de energia denominada energia primária. Com o avanço da vida produtiva, ocorre uma dissipação dessa energia primária resultando em um esgotamento da energia natural e uma queda no diferencial de pressão entre os limites do reservatório e os poços produtores. Com isso, o reservatório estaria destinado a uma baixa taxa de produção (ROSA ET AL.,2006).

Para contornar tal problema são usadas operações de manutenção de pressão, como a recuperação secundária. Este método consiste na recuperação por injeção de fluidos, como água e/ou gás, principalmente para fins de manutenção de pressão e eficiência de varredura volumétrica (SHENG, 2011). A eficiência deste método pode ser superior a 60%, embora o valor mais frequente seja de 30 a 50%, para os métodos convencionais (ROSA ET AL.,2006).

1.2 Objetivos

Os objetivos deste projeto de engenharia são:

- Objetivo geral:

- Desenvolver um software na área da engenharia de petróleo, mais especificamente, engenharia de reservatório;
 - Propor a solução para aplicação do método de recuperação secundária, a partir de água como fluido injetado;
 - Realizar análise para previsão de escoamento bifásico imiscível num reservatório;
- Objetivos específicos:
 - Solucionar o problema de permeabilidade relativa a partir do Modelo de Corey-Brooks;
 - Desenvolver a curva de fluxo fracionário para um reservatório bifásico;
 - Cálcular a área invadida pela injeção;
 - Calcular e analisar o comportamento das pressões;
 - Aplicar o Modelo de Dykstra-Parsosn (1950) e Stiles (1949) para:
 - * Cálcular da frente de avanço da camada em cada breakthrough (BT);
 - * Cálcular da eficiência vertical em cada BT;
 - * Cálcular do volume de óleo recuperado total;
 - * Cálcular do tempo necessário do BT na última camada (todo óleo recuperável possível por esse método de injeção);

Capítulo 2

Especificação

O desenvolvimento de um projeto de engenharia é constituído por várias etapas, e a primeira delas se trata da especificação/concepção. Neste capítulo serão definidos os requisitos a serem satisfeitos e as especificações do sistema como a descrição do objeto, o que se espera do projeto e o contexto da aplicação para o estudo dos processos de recuperação secundária de óleo.

2.1 Especificação do Software - Requisitos

Nesta seção são descritas as principais características, além dos requisitos para a utilização do software desenvolvido.

2.1.1 Nome do sistema/produto

Na Tabela 2.1, apresenta-se as características do software.

Software	Componentes Principais	Missão
 Modelo de deslocamento Imiscíveis Bifásico usados no Processo de Recuperação Secundária	 Fluxo Bifásico Areal para cálculo do comportamento das propriedades de reservatório homogêneo em esquemas de injeção em malhas Fluxo Bifásico em Sistema Estratificado com o modelo de DykstraParsons (1950) e Stiles (1949).	 Calcular Permeabilidade Relativa  Calcular pressão em reservatório  Calcular “breakthrough”  Calcular Eficiência Vertical  Calcular volume de óleo total  Calcular tempo para o “breakthrough”

Figura 2.1: Características gerais do Software de simulação de reservatórios

2.1.2 Especificação

O projeto a ser desenvolvido consiste em um programa que calculará características de um reservatório homogêneo a partir de um fluxo bifásico areal, preverá o desempenho no processo de recuperação secundária do óleo a partir de um sistema estratificado com fluxo bifásico.

A presente construção do sistema será utilizado em âmbito acadêmico como software livre, a partir do uso da Programação Orientada a Objeto em C++ e software Gnuplot, para que esteja disponível de fácil acesso a todos. A interface selecionada para o programa é em modo texto, o usuário irá se relacionar a partir do uso do teclado, mouse e monitor em conjunto com a interface do sistema construído. Os dados de entrada, propriedades do reservatório, serão fornecidos em modo .xlsx, na qual poderá ser modificado pelo usuário com base nas informações do reservatório em questão, enquanto que os dados de saída serão em modo arquivo de texto .txt e imagem .png com base nos diferentes modelos de deslocamento possíveis do software.

- **Dados/Atributos relativos ao reservatório:**

- Porosidade;
- Diferencial de Pressão [Pa];
- Permeabilidade [mD];
- Dimensões [m];

- **Dados/Atributos relativos aos fluidos:**

- Saturação de água irredutível;
- Saturação de óleo residual;
- Viscosidade da água [Pa.s];
- Viscosidade do óleo [Pa.s];
- Mobilidade [Kg.m³].

- **Dados/Atributos relativos ao teste de injeção:**

- Vazão de injeção [m³/s];
- Esquemas de injeção;
- Volume de óleo produzido [m³];

2.1.3 Requisitos funcionais

Apresenta-se a seguir os requisitos funcionais.

RF-01	O usuário deverá ser capaz de incluir valores de parâmetros de reservatório e propriedades do fluido;
--------------	---

RF-02	O usuário deverá ter liberdade para carregar dados a partir de um arquivo de disco criado pelo mesmo;
RF-03	Os resultados deverão ser exportados como textos e/ou gráficos;
RF-04	O usuário poderá plotar seus resultados em um gráfico. O gráfico poderá ser salvo como imagem ou ter seus dados exportados como texto.
RF-05	O usuário deve ter tal liberdade para escolher os modelos disponíveis para cálculo;

2.1.4 Requisitos não funcionais

Apresenta-se a seguir os requisitos não-funcionais.

RNF-01	Os cálculos devem ser feitos utilizando-se formulações matemáticas conhecidas da literatura;.
RNF-02	O programa deverá ser multi-plataforma, podendo ser executado em <i>Windows</i> , <i>GNU/Linux</i> ou <i>Mac</i> .

2.2 Casos de uso do software

A tabela 2.1 apresenta um caso de uso do sistema, bem como os diagramas de caso de uso.

Tabela 2.1: Caso de uso geral do sistema.

Nome do caso de uso:	Modelagem de Fluxo Bifásico Imiscível em Reservatório
Resumo/descrição:	Cálculo do desempenho no processo de recuperação secundária Calculo do fluxo fracionário
Etapas:	<ol style="list-style-type: none"> 1. Importar dados de entrada; 2. Definir método para avaliação do Breakthrough; 3. Calcular linhas equipotenciais; 4. Importar dados de localização dos poços; 5. Definir a configuração da malha dos poços injetores e produtores; 6. Gerar curvas de permeabilidade relativa; 7. Gerar curva do fluxo fracionário;
Cenários alternativos:	Inserir modelos, esquemas ou dados incompatíveis com a ordem de grandeza do problema.

2.2.1 Diagrama de caso de uso geral

O diagrama de caso uso geral da Figura 2.2 exibe o usuário interagindo com o software para obter o fluxo bifásico, características do reservatório e previsão do desempenho durante um processo de injeção. Neste caso de uso geral, o usuário insere os dados de entrada .dat, define o método para avaliação do Breakthrough, calcula linhas equipotenciais, importa dados de localização dos poços, define a configuração da malha dos poços injetores e produtores, gera os gráficos de pressão, permeabilidade relativa, fluxo fracionário e dados de posição da frente de avanço da água injetada, vazões de injeção e produção, volume de óleo produzido, tempo de produção e área invadida,. O usuário pode então fazer a análise dos resultados obtidos.

2.2.2 Diagrama de caso de uso específico

O diagrama de caso de uso específico da Figura 2.3 é um detalhamento do caso de uso para os cálculos que serão realizados, ele mostra a interação do usuário com o software para realizar os cálculo descritos anteriormente usando os modelos de deslocamento imiscível bifásico.

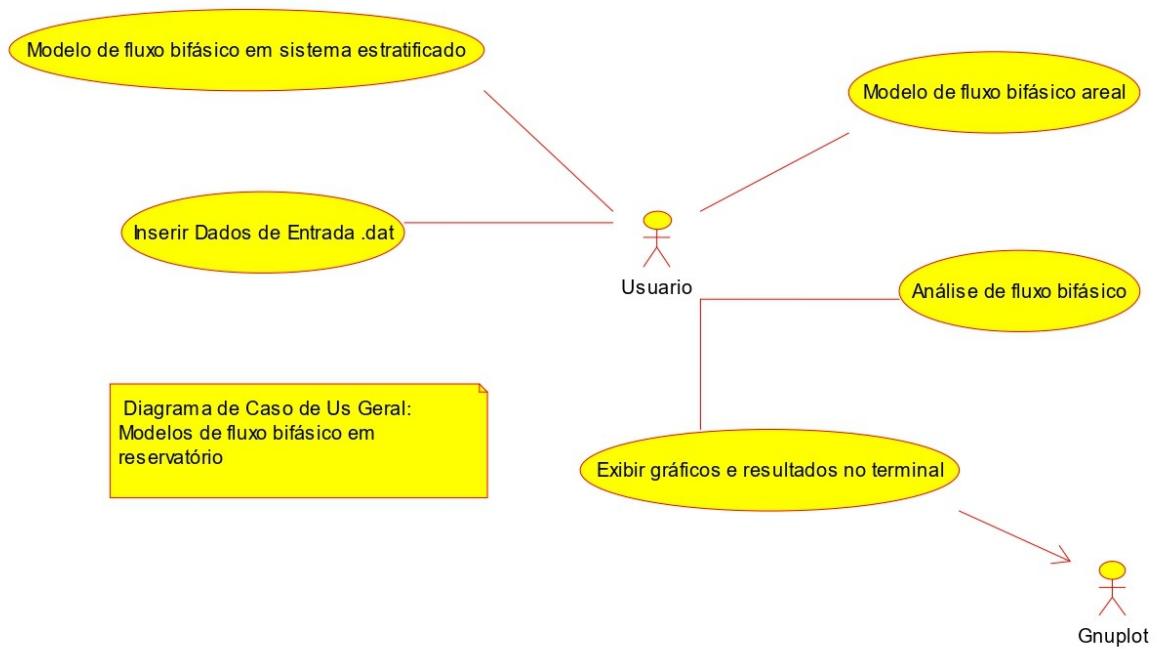


Figura 2.2: Diagrama de caso de uso geral – Modelos de fluxo bifásico em reservatórios

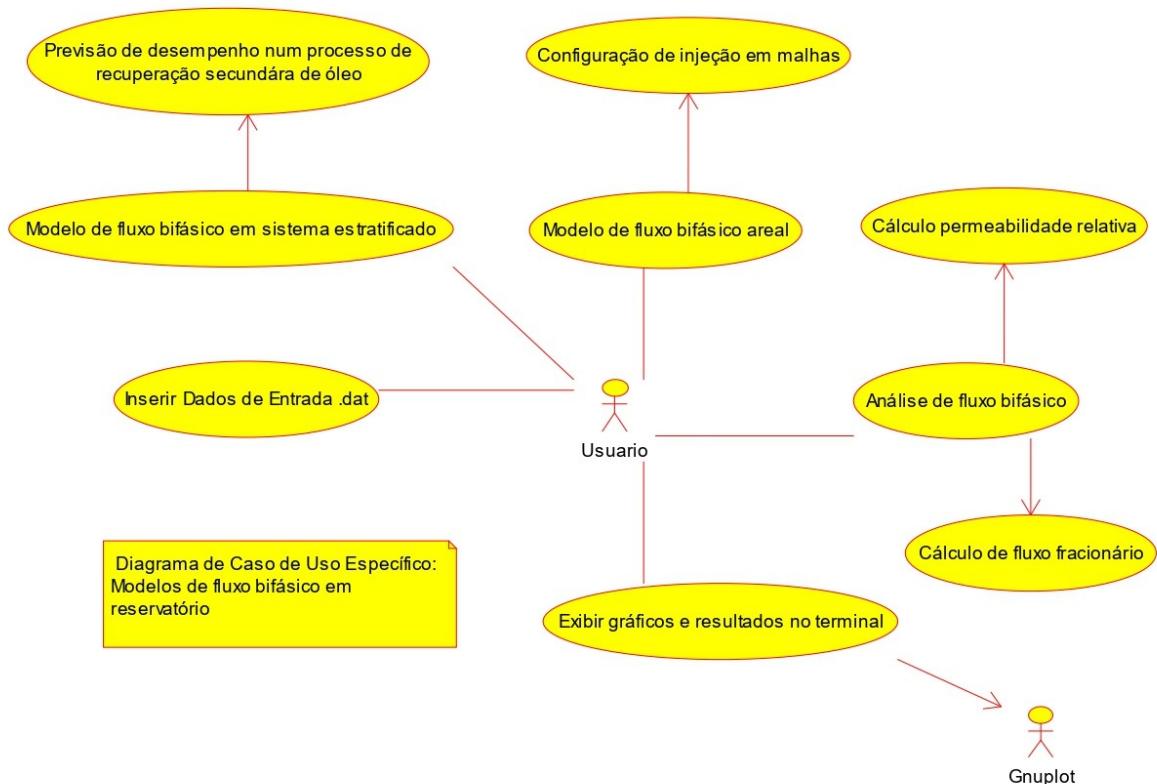


Figura 2.3: Diagrama de caso de uso específico – Modelos de fluxo bifásico em reservatórios

Capítulo 3

Elaboração

Depois da definição dos objetivos, da especificação do software e da montagem dos primeiros diagramas de caso de uso, a etapa do projeto de engenharia envolve a identificação e o estudo dos conceitos relacionados ao sistema a ser desenvolvido, isto é, a análise de domínio e a identificação de pacotes. Na elaboração fazemos uma análise dos requisitos, ajustando os requisitos iniciais de forma a desenvolver um sistema útil e adequado, que atenda às necessidades do usuário além de permitir seu reuso e futura extensão.

3.1 Análise de domínio

As acumulações de petróleo possuem certa quantidade de energia, denominada energia primária. A grandeza dessa energia é determinada pelo volume e pela natureza dos fluidos existentes na acumulação, bem como pelos níveis de pressão e de temperatura reinantes no reservatório. No processo de produção há uma dissipação da energia primária, causada pela descompressão dos fluidos do reservatório e pelas resistências encontradas por eles ao fluírem em direção aos poços de produção. Essas resistências são devidas, ou associadas, às forças viscosas e capilares presentes no meio poroso. O consumo de energia primária reflete-se principalmente no decréscimo da pressão do reservatório durante a sua vida produtiva, e consequente redução da produtividade dos poços (ROSA ET AL., 2006).

Há duas linhas gerais de ação para aprimorar os efeitos nocivos da dissipação da energia primária dos reservatórios de petróleo: Suplementando a com energia secundária através da injeção ou reduzindo as resistências viscosas e/ou capilares por meio de métodos especiais, como por exemplo o aquecimento da jazida (ROSA ET AL., 2006).

A quantidade de óleo que pode ser retirada de um reservatório unicamente às expensas de suas energias naturais é chamada de recuperação primária. Por outro lado, recuperação secundária é a quantidade adicional de óleo obtida por suplementação da energia primária com energia secundária, artificialmente transferida para a jazida, ou por meios que tendem a tornar a energia primária mais eficiente. Os objetivos práticos básicos dos métodos de recuperação secundária são o aumento da eficiência de recuperação e a ace-

leração da produção (ROSA ET AL., 2006). De acordo com Coelho (1991) um processo de exploração e produção de uma reserva de hidrocarbonetos, suas características petrofísicas, geológicas, geofísicas e geoquímicas são fundamentais de entendimento para a eficiente recuperação. A qualidade de um reservatório esta diretamente ligado ao ambiente deposicional do mesmo, bem como aos processos diagenéticos que lhe deram origem.

Em um projeto de injeção de fluidos a escolha do esquema de injeção - distribuição dos poços de injeção e produção - é fundamental, pois o sucesso aumenta à medida que certas linhas básicas de procedimento são adotadas ao se fazer essa escolha. Como o objetivo primordial da injeção é o aumento da recuperação de petróleo, deve-se tentar produzir esse volume adicional desejado utilizando-se esquemas em que os volumes de fluidos injetados sejam os menores possíveis. Devem ser buscadas situações em que a maior quantidade de fluido injetado permaneça no interior do reservatório, ou seja, a produção do fluido injetado seja a menor possível. As relações entre pressões e vazões e as relações destas últimas com o tempo do projeto são da maior importância e, portanto, devem ser encaradas como aspectos fundamentais a serem levados em conta no projeto. Finalmente, devem ser observadas as características particulares do reservatório em estudo, tais como a existência de falhas, variações de permeabilidade, estratificações, barreiras etc. Além disso, o aspecto econômico é decisivo (ROSA ET AL., 2006).

A teoria do avanço frontal é usada para calcular a vazão dos poços em esquemas de injeção, e em algumas de suas simplificações assumem que o fluxo entre os poços de injeção e produção é linear (todos os caminhos de fluxo são linhas retas) e que 100% do volume do poro do reservatório é contatado por água Injetada. Embora este comportamento possa ser aproximado em alguns reservatórios alongados, o fluxo linear ideal seria possível apenas se os fluidos pudessem ser injetados e produzidos a partir de toda a seção transversal do reservatório, ao invés de ser através da área limitada de um poço. Este problema é ainda mais complicado pelo fato da maioria dos campos serem desenvolvidos e a injeção de água ser feita utilizando algum padrão regular de poço (SMITH, COBB , 1997).

Quando se trata de um reservatório heterogêneo as taxas energéticas de sedimentação influenciam diretamente na seleção dos grãos. Em geral, ambientes de deposição com alta energia, geram reservatórios com boas características permoporosas, visto que os grãos foram melhor retrabalhados ao longo do curso até a compactação. Um exemplo interessante são os reservatórios originados em paleo-deltas, que são portadores da maior parte do óleo existente em reservatórios areníticos no planeta, (COELHO, 1991). Contudo, a taxa pode variar num mesmo ambiente, gerando reservatórios com variações verticais de permeabilidade, ou reservatórios heterogêneos, que ainda podem se dividir em dois grupos, aqueles que possuem camadas sem cruzamento de fluxo.

A heterogeneidade do reservatório provavelmente tem mais influência do que qualquer outro fator no desempenho de uma injeção de fluido, ao passo que se torna a variável mais difícil de determinar (SMITH, COBB, 1997). É necessário entender como as variações de permeabilidade vertical e areal podem ser determinadas, a fim de obter uma melhor

previsão de desempenho de eficiência do método de recuperação por injeção

Neste trabalho serão discutidos conceitos de razão mobilidade, condutividade, fluxo fracionário, esquemas de injeções em malhas, linhas de fluxo e de pressão, a área invadida pela água, bem como a eficiência do varrido horizontal e vertical em reservatórios homogêneos e heterogêneos

3.2 Conceitos Fundamentais

O conhecimento das propriedades das rochas e fluidos é essencial para o desenvolvimento de qualquer metodologia para aumentar o fator de recuperação do petróleo. Assim, será mostrado definições de porosidade absoluta e relativa, molhabilidade, permeabilidade absoluta, fator de recuperação, mobilidade, razão mobilidade, entre outros.

3.2.1 Conceitos de Propriedades de Rochas e Fluidos

- **Porosidade:**

A porosidade de uma rocha é definida como a razão entre o volume poroso, capaz de armazenar fluidos, e o volume total da rocha, que é dado pela soma do volume poroso e do volume da parte sólida da rocha. A porosidade mede o volume dos espaços vazios em um meio poroso, independente de estarem ou não interligados. Portanto, a porosidade é um parâmetro petrofísico de grande importância visto que se consegue medir a capacidade de armazenagem de fluidos em um corpo poroso [Rosa et al., 2006].

Assim, a porosidade (ϕ), expresso em porcentagem, pode ser definida como a razão entre o volume poroso (V_p) e o volume total (V_t) da amostra (Equação 3.1), onde o volume total é dado pela soma do espaço poroso e da fase sólida (Equação 3.2).

$$\phi = \frac{V_p}{V_t} \quad (3.1)$$

$$V_t = V_p + V_{sólida} \quad (3.2)$$

- **Conservação de Massas:**

A equação da continuidade é desenvolvida efetuando-se um balanço de massas sobre um elemento de volume $\Delta x \Delta y \Delta z$, fixo no espaço, através do qual um fluido está escoando (Figura 3.1). Pelo Princípio da Conservação das Massas, pode-se dizer que ao longo de um determinado intervalo de tempo, a massa de água que entra num determinado sistema subtraído da massa que sai, será igual ao acúmulo de massa dentro do sistema. (BIRD;LIFHFOOT, STEWART, 1987)

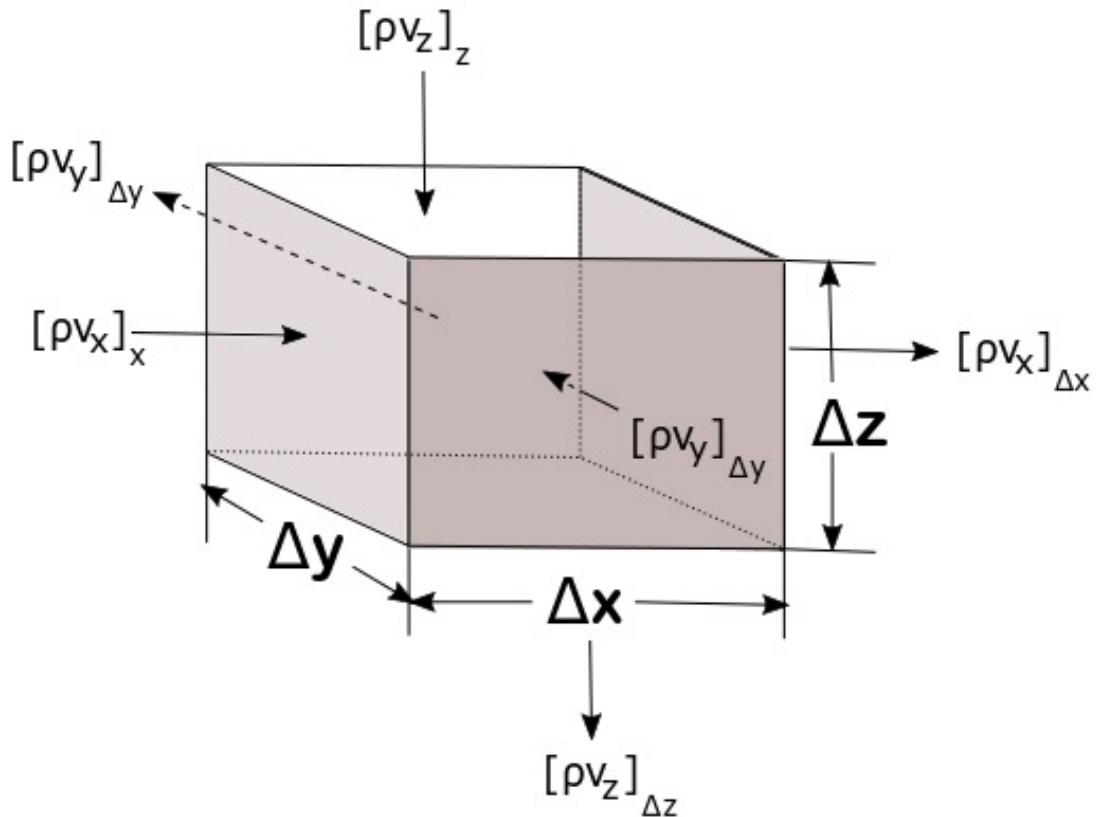


Figura 3.1: Balanço de Massas (Autor)

$$\frac{\partial(\rho_j S_j \phi)}{\partial t} = -\nabla \cdot (\rho_j \mathbf{u}) \quad (3.3)$$

sendo:

$$\nabla \cdot (\rho \mathbf{u}) = \frac{\partial (\rho u_x)}{\partial x} + \frac{\partial (\rho u_y)}{\partial y} + \frac{\partial (\rho u_z)}{\partial z} \quad (3.4)$$

- **Molhabilidade:**

O deslocamento de petróleo por meio dos poros da rocha reservatório também é influenciado por outros parâmetros petrofísicos como a molhabilidade, a qual pode ser definida como a capacidade de um fluido de se espalhar em uma superfície sólida na presença de outros fluidos, ou seja, é a tendência da superfície de ter mais afinidade por um fluido em detrimento de outro, também presente no meio poroso. Isto significa que em um fluxo multifásico, um fluido tem mais afinidade com o meio poroso que outros fluidos presentes [Rosa et al., 2006, Dandekar, 2013]. Desta forma, a molhabilidade quantifica a afinidade que a superfície da rocha apresenta para cada fluido na presença de outros, estando relacionada com as forças intermoleculares que atuam entre a superfície e as moléculas dos líquidos presentes. Em reservatórios de petróleo, encontram-se basicamente duas fases líquidas, formadas pelo óleo e a água [Rosa et al., 2006].

- **Permeabilidade:**

A permeabilidade é uma das características petrofísicas mais importantes de um reservatório, sendo a capacidade da rocha de permitir o escoamento de fluidos. Uma rocha pode ter alta porosidade e apresentar baixa permeabilidade, caso os poros não sejam bem conectados, ou seja, para que o reservatório seja produtivo não basta um alto valor de porosidade, a rocha deve possuir a capacidade de permitir o deslocamento de fluidos através dela [Rosa et al., 2006].

Este parâmetro é um dos que tem mais influência na determinação da capacidade de produção de hidrocarbonetos acumulados. A permeabilidade (k) é uma propriedade dinâmica, definida como a capacidade de um dado meio poroso se deixar atravessar por um fluido [Rosa et al., 2006]. Ela é uma função da posição e pressão, e varia fortemente com o tamanho dos poros e sua distribuição em determinado local [Lake et al., 1989].

O conceito de permeabilidade aparece na lei que governa o deslocamento dos fluidos através de meios porosos, conhecida como a Lei de Darcy, sendo medida em milidarcy (md). Existem dois tipos de permeabilidades, a permeabilidade absoluta, quando o reservatório está saturado com um único fluido, e a permeabilidade efetiva, quando existem dois ou mais fluidos coexistindo dentro de uma mesma rocha [Albuquerque et al., 2007, Dandekar, 2013]. O desenvolvimento da expressão que permite encontrar a permeabilidade absoluta de um meio poroso é utilizado até os dias atuais na indústria do petróleo. O experimento original de Darcy investigou o fluxo de água através da areia e concluiu que um fluxo linear com vazão de injeção (q) é função da condutividade hidráulica (k), da área da seção transversal (A), do diferencial de pressão da entrada para a saída (ΔP) e do comprimento do meio poroso (L), conforme mostra a Equação [Rosa et al., 2006, Lake et al., 1989].

$$q = \frac{kA\Delta P}{\mu L} \quad (3.5)$$

A permeabilidade absoluta pode ser calculada isolando k na equação de Darcy conforme Equação 3.6.

$$k = \frac{q\mu L}{A\Delta P} \quad (3.6)$$

Sendo:

- k = Permeabilidade
- A = Área da seção transversal
- μ = Viscosidade
- L = Comprimento do meio poroso

Desta forma, durante um fluxo, se todas as variáveis são conhecidas, menos a permeabilidade, torna-se possível encontrá-la. A forma para a Equação 3.6 é utilizada para um fluxo linear. Em poços, com fluxo radial, modifica-se a geometria para definir a permeabilidade levando em consideração o raio externo do reservatório, o raio do poço, a pressão externa do reservatório, e a pressão medida no poço em uma determinada altura do reservatório, como será visto na fundamentação teórica dos métodos utilizados neste trabalho [Rosa et al., 2006]. A permeabilidade efetiva (k_e), quando dois ou mais fluidos saturam o meio poroso [Rosa et al., 2006], sempre apresentará valores menores do que o valor da permeabilidade absoluta da rocha. O cálculo das permeabilidades efetivas à água e ao óleo (k_w e k_o) também pode ser realizado usando a Lei de Darcy conforme Equações 3.7 e 3.8 [Rosa et al., 2006].

$$k_{rw} = \frac{k_w L q_w}{A \Delta P} \quad (3.7)$$

$$k_{ro} = \frac{k_o L q_o}{A \Delta P} \quad (3.8)$$

A razão entre a permeabilidade efetiva de determinado fluido no meio poroso e a permeabilidade absoluta é denominada permeabilidade relativa (k_r), a qual pode ser representada pelas Equações 3.9 e 3.10 para um sistema bifásico óleo-água.

$$k_{rw} = \frac{k_w}{k} \quad (3.9)$$

$$k_{ro} = \frac{k_o}{k} \quad (3.10)$$

Este parâmetro sofre efeitos da variação da saturação dos fluidos, da molhabilidade da rocha, da estrutura dos poros da rocha, da tensão de confinamento, do teor de argila da rocha, da migração de finos, da temperatura, além das variações de tensão interfacial, viscosidade e velocidade do fluxo [Dandekar, 2013].

O aumento da saturação de um fluido molhante no meio poroso em relação a outro fluido chama-se de embebição, e, por outro lado, quando existe uma redução de saturação do fluido que molha preferencialmente a rocha em relação a outro fluido, tem-se uma drenagem [Donaldson et al., 1985]. Assim, no processo de embebição, é necessário que haja uma determinada saturação da fase molhante no inicio do fluxo, chamada saturação de água conata ou saturação irredutível (Swi). Da mesma forma ocorre no processo de drenagem, e essa saturação é denominada saturação de óleo residual (Sor). A saturação da fase não molhante atinge seu valor máximo a saturações menores que 100%, o que indica que nem todo o meio poroso interligado irá contribuir ao fluxo desta fase [Núñez, 2011].

- **Mobilidade e Razão Mobilidade:**

Na lei de Darcy, há um fator de proporcionalidade relacionado à velocidade de um fluido e ao gradiente de pressão. Este fator de proporcionalidade, denominado mobilidade do fluido, é a permeabilidade efetiva da rocha à esse fluido, dividida pela viscosidade do mesmo (CRAIG, F.F., 1971).

Se três fluidos (óleo, água e gás) estiverem presentes no meio poroso as suas mobilidades serão definidas, respectivamente, por:

$$\lambda_w = \frac{k_w}{\mu_w}, \quad (3.11)$$

$$\lambda_o = \frac{k_o}{\mu_o}, \quad (3.12)$$

$$\lambda_g = \frac{k_g}{\mu_g}, \quad (3.13)$$

Muskat (1937) discutiu pela primeira vez o termo que ficou conhecido como razão de mobilidade. Posteriormente, foi usado para relacionar a mobilidade da água na porção de uma injeção no contato água com a mobilidade do óleo no banco de óleo. Ele apresentou as distribuições de pressão em regime permanente para uma série de arranjos de poços de produção de injeção, isto é, sob condições de uma razão de mobilidade unitária. A razão de mobilidades (M) é a relação entre a mobilidade do fluido deslocante (λ_d) atrás da frente de avanço do mesmo e a mobilidade do fluido deslocado no banco deste fluido. Por exemplo, no caso do fluido deslocado ser o óleo a razão de mobilidades é dada por :

$$M = \frac{k_d \mu_o}{\mu_d k_o} = \frac{\lambda_d}{\lambda_o}, \quad (3.14)$$

onde o subscrito d denota a fase de deslocamento. Na terminologia de injeção de água, isso se torna,

$$M = \frac{k_w \mu_o}{\mu_w k_o} = \frac{k_w \mu_o}{\mu_w k_{ro}}, \quad (3.15)$$

como a permeabilidade efetiva é função da saturação, a mobilidade também é.

É importante notar que as permeabilidades relativas à água e óleo na Eq.3.15 são definidas em dois pontos separados no reservatório, ou seja, k_w é a permeabilidade relativa à água na parte do reservatório em contato com a água (na parte invadida pela água) e k_o é a permeabilidade relativa ao óleo no banco de óleo (parte não invadida do reservatório) (COBB; SMITH, 1997).

3.2.2 Conceitos Teóricos:

- **Fluxo Fracionário:**

O fluxo fracionário de um fluido é interpretado como o quociente entre a taxa de fluxo

desse fluido e a taxa total de fluxo. Assim, o fluxo fracionário da água, f_w , do óleo, f_o e o total, f_t são definidos, respectivamente, pelas Equações 3.16 , 3.17 e 3.18. (ROSA, 2006).

$$f_w = \frac{u_w}{ut} \quad (3.16)$$

$$f_o = \frac{u_o}{ut} \quad (3.17)$$

$$f_t = f_w + f_o \quad (3.18)$$

- **Eficiência do Verrido Vertical (Ev):**

Definida pela razão entre o volume invadido pela água e o volume total da malha:

$$Ev = \frac{\text{Volume.invadido.pela.água}}{\text{Volume.total.da.malha}} \quad (3.19)$$

Como a própria definição mostrou, quantifica o volume invadido pela água injetada no reservatório, sendo significativamente afetada pela estratificação devido ao movimento preferencial de fluidos nas zonas mais permeáveis. Sofre ainda influência de outros parâmetros como razão de mobilidade, fluxo entre camadas, força da gravidade e forças capilares.

- **Eficiência do Verrido Horizontal (Ev):**

Em qualquer projeto, independentemente do esquema escolhido, existe uma área total definida que está sujeita à influência da injeção. Por exemplo, em um esquema “five-spot” essa área total é a área da malha base, ou seja, um quadrado. Já no modelo “seven-spot” essa área é um hexágono. Em um reservatório como o da Figura ?? a área total pode ser vista em planta, delimitada pelo contato óleo/água. Observe que esta área é sempre medida em planta (ROSA ET AL., 2006).

Se não existissem fatores que interferem no desempenho do processo e se o tempo de atuação fosse infinito, a área da malha ou do reservatório seria integralmente varrida pelo fluido injetado, e a recuperação de petróleo seria proveniente de toda essa área. Em projetos reais, devem ser efetuados cálculos para estimar que percentuais dessa área total foram invadidos em diferentes tempos e diferentes condições, uma vez que o fluido injetado invade apenas uma parte da área total (ROSA ET AL., 2006).

Define-se eficiência de varrido horizontal, E_A , como a relação entre a área invadida pelo fluido injetado e a área total do meio poroso, ambas medidas em planta. Assim:

$$E_A = A_{inv}/A_t \quad (3.20)$$

onde A_{inv} é a área invadida pelo fluido e A_t a área total.

- **Campo potencial e linhas de fluxo**

Para cada distribuição de poços de injeção e de produção que se implanta em um reservatório, e a cada instante, existe um campo potencial que é resultado não só das posições desses poços como também das suas vazões e pressões. Para uma formação horizontal e de pequena espessura, o potencial pode ser substituído pela pressão (ROSA ET AL., 2006).

Os pontos de maior potencial são os poços de injeção e os de menor potencial são os poços de produção, e entre esses pontos existem valores intermediários espalhados por todo o reservatório. Esse campo potencial pode ser representado em planta por meio de linhas equipotenciais. No caso de um único poço situado no centro de um reservatório cilíndrico, por exemplo, as linhas equipotenciais são circunferências que têm o poço como centro como mostrado a Figura 3.2 (ROSA ET AL., 2006).

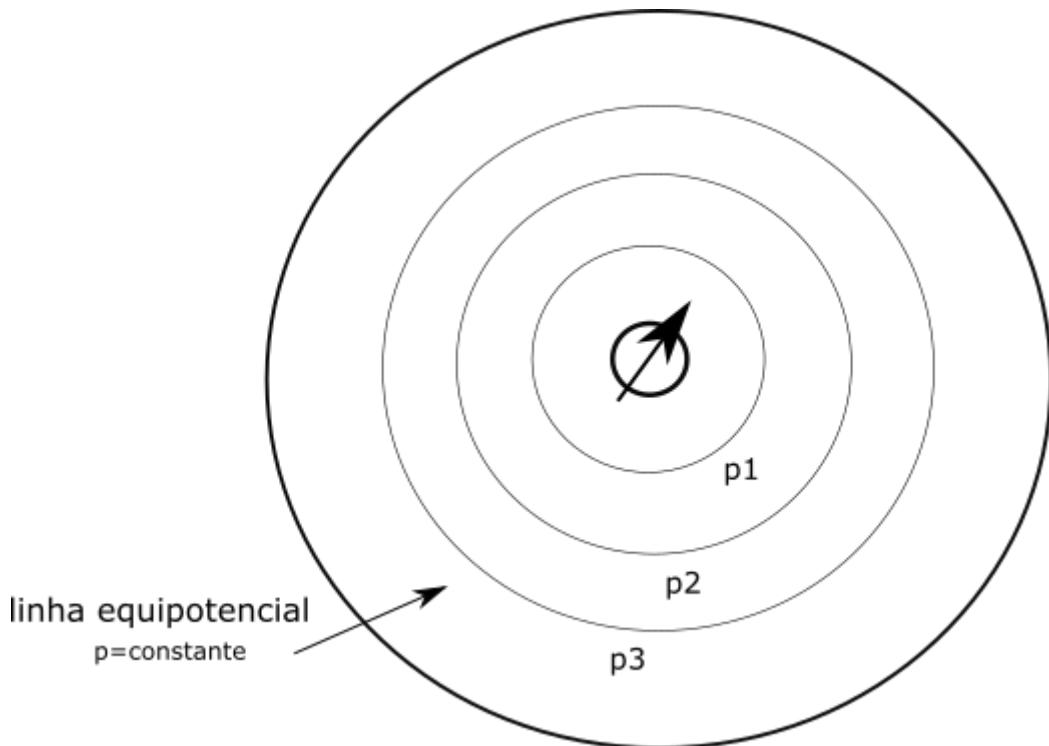


Figura 3.2: Linhas equipotenciais concêntricas em um reservatório infinito.

Perpendiculares às linhas equipotenciais se localizam as linhas de fluxo, que começam nos poços de injeção e se estendem até os poços de produção. Como o próprio nome já indica, o fluxo ocorre ao longo dessas linhas. Se o sistema está em regime permanente, tanto o campo potencial como a localização das linhas de fluxo não se alteram com o tempo.

A Figura 3.3 apresenta uma malha de injeção em linha direta com algumas das suas linhas de fluxo. Nas vizinhanças dos poços as equipotenciais são circunferências concêntricas aos mesmos. Como as linhas de fluxo são perpendiculares às equipotenciais, nessas regiões o fluxo é radial (ROSA ET AL., 2006).

Como pode ser observado na Figura 3.3, as linhas de fluxo entre dois poços têm comprimentos diferentes. Como a diferença de pressão entre o poço de injeção e o de

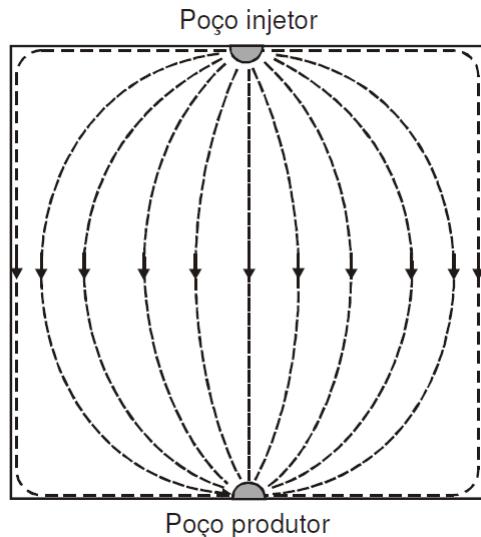


Figura 3.3: Linhas de Fluxo (ROSA ET AL., 2006).

produção é a mesma ao longo de qualquer linha, cada uma tem um gradiente médio de pressão diferente. As linhas de menor comprimento são as de maior gradiente médio (ROSA ET AL., 2006).

Ao penetrarem no meio poroso, as partículas de fluido que se deslocarem ao longo da linha de fluxo mais curta terão maior velocidade que as partículas que percorrerem outras linhas quaisquer. Isso quer dizer que em um determinado instante cada linha de fluxo terá sido varrida de uma maneira diferente das outras. Deve ser observado que a velocidade varia não só de uma linha para outra como ao longo da própria linha. A Figura 3.4 mostra como o fluido injetado penetra no meio poroso e a forma que a região invadida vai tomando em função das diferenças de gradiente médio de pressão entre as linhas de fluxo (ROSA ET AL., 2006).

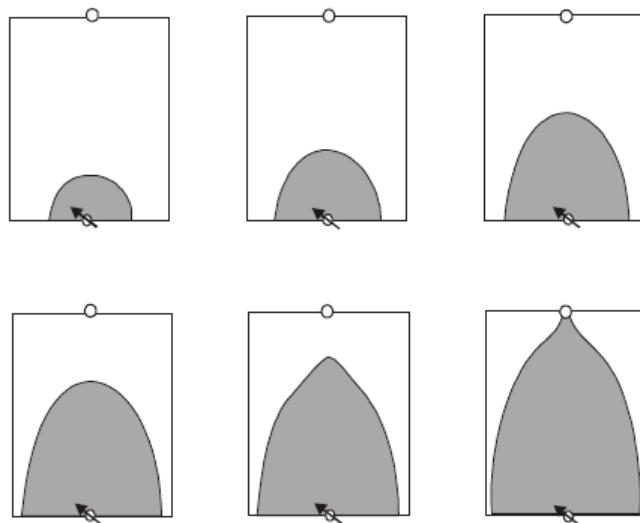


Figura 3.4: Evolução da área invadida em uma malha em linha direta (ROSA ET AL., 2006).

Inicialmente o fluido injetado se propaga radialmente porque nas proximidades do poço de injeção o gradiente de pressão em todas as linhas é praticamente o mesmo. Quando vista em planta, a área invadida pelo fluido tem uma forma também praticamente circular. À medida que o fluido avança em cada linha, como o seu gradiente de pressão vai se alterando, a sua velocidade também vai se alterando, de tal maneira que a região invadida, que inicialmente era circular, vai adquirindo outra forma. No instante em que a primeira partícula do fluido injetado alcança o poço de produção, teoricamente só a linha de fluxo mais curta foi inteiramente varrida, restando partes do reservatório que ainda não foram contatadas. A região invadida pelo fluido injetado vai se alterando não só em forma como também em dimensão, à medida que mais e mais fluido vai penetrando no meio poroso (ROSA ET AL., 2006).

Conforme será discutido na próxima seção, usando as expressões analíticas que descrevem o comportamento da pressão em reservatórios homogêneos infinitos é possível estimar a área de varrido, bem como a distribuição de pressão e o comportamento das linhas de fluxo, em reservatórios submetidos à injeção de água (ROSA ET AL., 2006).

A dimensão da área invadida e, consequentemente, a eficiência de varrido horizontal dependem da geometria de injeção, do volume de fluido injetado e da razão entre a mobilidade do fluido injetado e a mobilidade do fluido deslocado. Para se entender um pouco mais sobre a formação dessas áreas invadidas é necessário um pequeno estudo a respeito de campos potenciais e linhas de fluxo, que será mostrado a seguir (ROSA ET AL., 2006).

- **Modelo de Corey-Brooks:**

A Figura 3.5 mostra os valores de permeabilidade relativa para todo o intervalo de valores de saturação de água. É possível observar que à medida que a saturação de água diminui, a sua permeabilidade efetiva cai de forma sensível no início. Considerando que o meio poroso em estudo é molhável a água, essa situação já era esperada, visto que o óleo irá ocupar inicialmente uma região de maior diâmetro no centro dos capilares. A tendência natural é que a saturação de óleo cresça até atingir a saturação crítica e começar a fluir e em consequência, a saturação de água começará a diminuir de forma mais significativa(Rosa et al, 2016).

O crescimento da saturação de óleo é diretamente proporcional ao da sua permeabilidade relativa. Enquanto isso, a permeabilidade relativa da água descrece até que seja atingido o ponto de saturação irredutível de água(S_{wi}), em que ela parará de fluir , logo , sua permeabilidade relativa será nula.

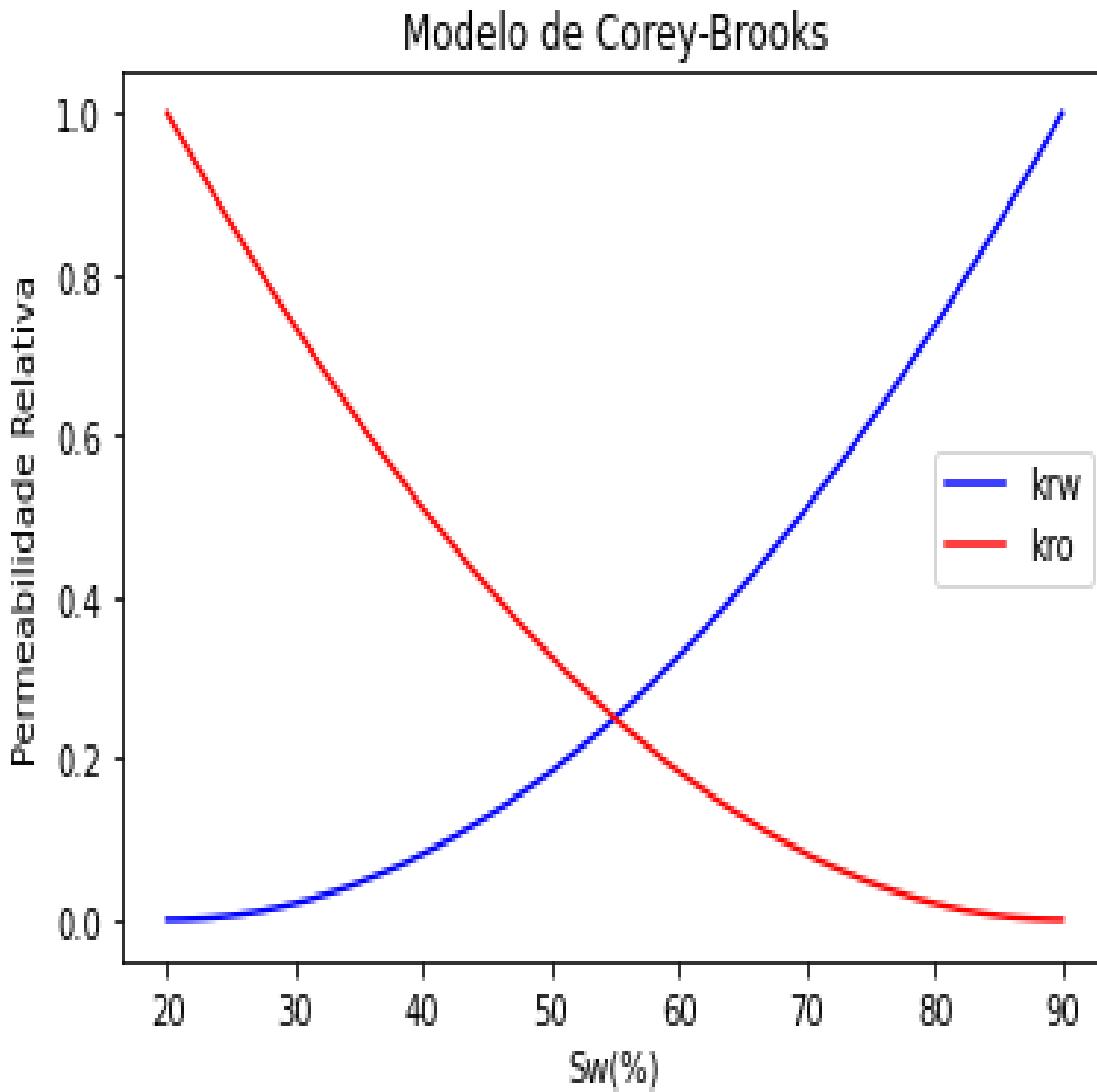


Figura 3.5: Modelo de Corey-Brooks para permeabilidade relativa(Autor).

A partir do modelo de Corey-Brooks para permeabilidade relativa e pressão capilar, temos as seguintes relações:

$$k_{rw}(S_w) = (k_{rw})_{s_{orw}} \left(\frac{S_w - S_{wi}}{1 - S_{wi} - S_{orw}} \right)^{ew} \quad (3.21)$$

$$k_{ro}(S_w) = (k_{ro})_{s_{wi}} \left(\frac{1 - S_w - S_{orw}}{1 - S_{wi} - S_{orw}} \right)^{eow} \quad (3.22)$$

$$P_c(S_w) = (P_c)_{s_{wi}} \left(\frac{1 - S_w - S_{orw}}{1 - S_{wi} - S_{orw}} \right)^{epcow} \quad (3.23)$$

sendo k_{rw} =permeabilidade relativa na água, k_{ro} =permeabilidade relativa na óleo, $(k_{rw})_{s_{orw}}$ =permeabilidade relativa na água na saturação de óleo residual , $(k_{rw})_{s_{wi}}$ =permeabilidade relativa da água na saturação de agua irreductivel , S_w =Saturação de água, S_{wi} =Saturação de água irreductível, S_{orw} =Saturação de óleo residual , P_c =pressão capilar , ew,eow e

epcow= constantes experimentais de Corey-Brooks .

3.3 Formulações Matemáticas

3.3.1 Modelo de Fluxo Bifásido (1D)

Posteriormente as suposições básicas, o assunto é introduzido da maneira convencional, descrevendo a equação fluxo e a equação de Buckley-Leverett. Por ser unidimensional, sua aplicação direta, no cálculo da recuperação de óleo, ficaria restrita à distribuição da saturação de água uniforme em relação ao comprimento. Mediante ao fato de que há uma distribuição de saturação não uniforme, utilizou-se o modelo de Corey-Brooks para permeabilidades relativas, que são funções da saturação de água e foi obtida uma solução para o problema de Riemann e de Goursat-Riemann para uso em conjunto com a teoria de Buckley-Leverett(DAKE,1978).

- Equação de Buckey Leverett

$$\phi \frac{\partial(S_j)}{\partial t} + \frac{\partial(u_{jx})}{\partial x} = 0 \quad (3.24)$$

- Lei de Darcy

$$u_\pi = -k \frac{k_{r\pi}}{\mu_\pi} \left(\frac{\partial P_\pi}{\partial x} - \rho_\pi g \operatorname{sen}\alpha \right) \quad (3.25)$$

- Velocidade de deslocamento do óleo

$$u_o = -k \frac{k_{ro}}{\mu_o} \left(\frac{\partial P_o}{\partial x} - \rho_o g \operatorname{sen}\alpha \right) \quad (3.26)$$

- Velocidade de deslocamento da água

$$u_w = -k \frac{k_{rw}}{\mu_w} \left(\frac{\partial P_w}{\partial x} - \rho_w g \operatorname{sen}\alpha \right) \quad (3.27)$$

- Função Fluxo

$$u_t = -k \frac{k_{rw}}{\mu_w} \left(\frac{\partial P_w}{\partial x} - \rho_w g \operatorname{sen}\alpha \right) - k \frac{k_{ro}}{\mu_o} \left(\frac{\partial P_o}{\partial x} - \rho_o g \operatorname{sen}\alpha \right) \quad (3.28)$$

- Pressão Capilar

$$P_c = P_o - P_w \quad (3.29)$$

- Derivada da Função Fluxo

$$\frac{du_w}{dS_w} = u_t \frac{d}{dS_w} \left(\frac{\lambda_w}{\lambda_t} \right) + g \operatorname{sen}\alpha (\rho_w - \rho_o) \frac{d}{dS_w} \left(\frac{\lambda_w \lambda_o}{\lambda_t} \right) + \frac{d}{dS_w} \left(\frac{\lambda_w \lambda_o}{\lambda_t} \frac{\partial P_c}{\partial x} \right) \quad (3.30)$$

3.3.2 Modelo de Fluxo Bifásido Areal (2D)

- **Método aproximado de Deppe para análise da injetividade relativa contra o avanço da frente de injeção:**

Em um projeto de injeção de água é necessário o conhecimento dos valores, pelo menos aproximados, das vazões e das pressões de injeção. Valores muito altos de pressões de injeção podem acarretar fraturas na formação e prejudicar seriamente o deslocamento do óleo pela água. Por outro lado, é necessária uma boa injetividade para se obter uma boa produtividade. Os valores de vazão e de pressão de injeção são necessários também para o dimensionamento dos equipamentos de superfície a serem utilizados no projeto de injeção (ROSA ET AL., 2006).

Quando se estuda a distribuição de pressão no meio poroso (dentro de uma determinada malha), observa-se que uma grande parcela da queda de pressão entre os poços de injeção e de produção ocorre exatamente nas proximidades dos poços, onde o fluxo comporta-se como sendo radial. Em alguma região entre os poços o fluxo é aproximadamente linear, de modo que a injetividade na malha deve ser calculada fazendo-se a combinação dos fluxos que ocorrem na malha. Diversos estudos foram feitos, principalmente por Deppe J. (1961) e Muskat (1946), sobre injetividade para os vários tipos de geometria de injeção, entre os quais podem ser destacadas as equações para os modelos de linha direta, linha esconsa, “five-spot”, “seven-spot” e “nine-spot” invertido. Essas equações foram deduzidas admitindo-se razão de mobilidades igual a 1, saturação de gás inicial igual a zero e regime permanente.

Quando as mobilidades de fluido nas regiões varridas e não-varridas são iguais, a injetividade não mudará conforme a frente de inundação avança. Para padrões regulares, pode ser calculado por fórmulas matemáticas.

Quando as mobilidades de fluido nas regiões varridas e não-varridas não são iguais, a injetividade aumentará ou diminuirá conforme a frente de inundação avança. Nesse caso, a injetividade não foi calculada por métodos analíticos para nenhum padrão prático de poço e, além disso, os resultados do modelo em escala e analógico foram publicados apenas para o padrão de cinco pontos (DEPPE J., 1961).

O objetivo é apresentar um método aproximado de cálculo da injetividade para o caso de mobilidades desiguais sendo aplicado a padrões regulares. Antes que o método aproximado de Deppe J. seja discutido, as fórmulas analíticas para mobilidades iguais serão resumidas, e a solução analítica para fluxo radial com mobilidades diferentes será usada para mostrar como a injetividade muda conforme a injeção avança.

Considere um sistema radial com um poço de injeção central de raio r_w e imagine que o fluido é produzido uniformemente a partir de cada ponto em um círculo de raio r_e . Um fluxo puramente radial resultará, e a frente entre os fluidos injetados e originais será um círculo cujo raio será denominado r_f como mostram as Figuras 3.6 e 3.7.

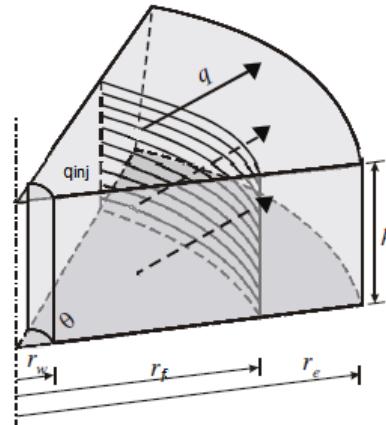


Figura 3.6: Sistema radial em poços de injeção (Adaptado de ROSA ET AL., 2006).

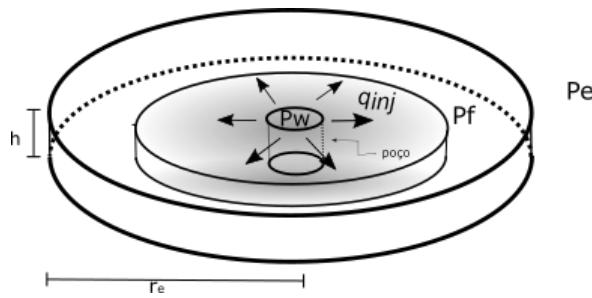


Figura 3.7: Sistema radial em poços de injeção.

Partido na equação da Lei de Darcy, e fazendo substituições para adequar-se ao esquema mostrado na Figura 3.6 encontramos a formulação para a injetividade (Eq. 3.31). Na maioria das aplicações, é conveniente expressar a variação da injetividade como o progresso da frente em termos da injetividade inicial. Neste caso, isto é, uma injetividade relativa q_{ir} , é definida como a razão da injetividade em qualquer momento

$$q_{inj} = \frac{C_1 h \lambda_o \Delta P}{\log\left(\frac{r_f}{r_w}\right) \frac{1}{M} + \log\left(\frac{r_e}{r_f}\right)}, \quad (3.31)$$

onde $\Delta P = P_e - P_w$ e $M = \frac{\lambda_d}{\lambda_o}$.

Na maioria das aplicações, é conveniente expressar a variação da injetividade como o progresso da frente em termos da injetividade inicial. Neste caso, isto é, uma injetividade relativa q_{ir} , é definida como a razão da injetividade em qualquer momento, dada pela Eq. 3.31, pela a injetividade inicial, dada pela Eq. 3.31 com $r_f = r_w$, (DEPPE J., 1961). Fazendo a injetividade inicial ($q_{inj,i}$), temos:

$$q_{inj,i} = \frac{C_1 h \lambda_o \Delta (A_{inv})_{BT}, P}{\log\left(\frac{r_e}{r_f}\right)}. \quad (3.32)$$

Então, a razão entre a Eq. 3.31 e Eq. 3.32, resulta :

$$q_{ir} = \frac{\log\left(\frac{r_e}{r_f}\right)}{\log\left(\frac{rf}{rw}\right) \frac{1}{M} + \log\left(\frac{re}{rf}\right)} \quad (3.33)$$

A equação da injetividade relativa calculada através do método aproximado de Deppe (Eq. 3.33) será um dos cálculos realizados pelo Software na análise do comportamento areal. A injetividade relativa começa unitária quando $r_f = r_w$ e termina quando $r_f = r_e$ (correspondente à varredura completa da área e mudança completa da mobilidade do fluido de λ_o para λ_d).

As curvas calculadas a partir desta equação serão traçadas pelo software gnuplot para quaisquer razões de mobilidade. A injetividade relativa é plotada contra a fração da área varrida E_A , ao invés de contra a posição da frente de avanço. Nesse caso, a relação é $\frac{r_f^2}{r_e^2} = E_A$ para $r_w \ll r_e$.

- **Determinação analítica da área de varrido e do comportamento das linhas de fluxo**

Conforme mostrado por Brigham (1981), em algumas situações particulares é possível a determinação analítica da área de varrido, da distribuição de pressão e do comportamento das linhas de fluxo em um reservatório sujeito à injeção de água. Dentre essas situações pode-se considerar o caso de um reservatório de óleo subsaturado, homogêneo e horizontal, sujeito à injeção de água, onde a razão de mobilidades seja unitária. Em outras situações mais complexas, a solução obtida com essas hipóteses simplificadoras fornecerá uma idéia do comportamento real.

Considere, por exemplo, o caso de dois poços, sendo um deles injetor de água, com vazão q_1 , e o outro produtor de óleo, com vazão q_2 , localizados em um reservatório muito extenso, conforme mostrado na Figura 3.8. O reservatório é homogêneo e horizontal, e a razão entre as mobilidades da água e do óleo é unitária. Admita que a espessura do reservatório seja pequena, de modo que o fluxo possa ser considerado como sendo praticamente horizontal. Admita ainda que as vazões sejam medidas em condições de reservatório e que os valores absolutos das vazões de injeção e de produção sejam iguais a q , sendo $q > 0$. Como normalmente convém que a vazão de produção é positiva, então $q_2 = q$ e $q_1 = -q_2 = -q$ (ROSA ET AL., 2006).



Figura 3.8: Sistema composto de um poço injetor e de um produtor (ROSA ET AL., 2006).

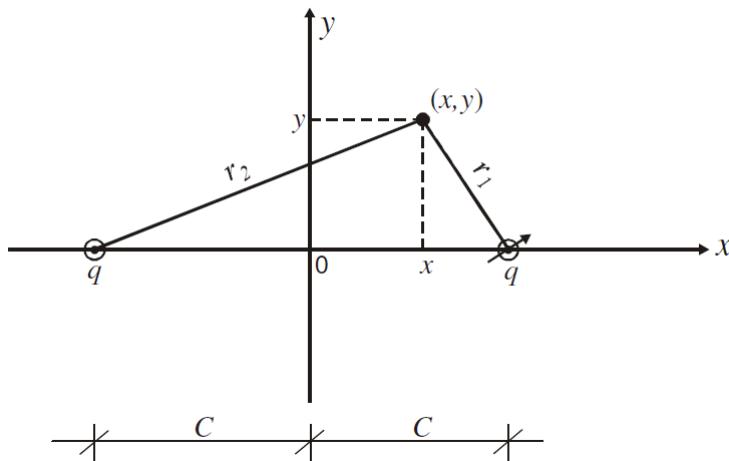
Utilizando a aproximação logarítmica para representar a solução do modelo da fonte

linear, a queda de pressão adimensional em um ponto qualquer de um reservatório infinito, devida à produção de um poço com vazão q , é dada, empregando-se um sistema compatível de unidades, pela expressão:

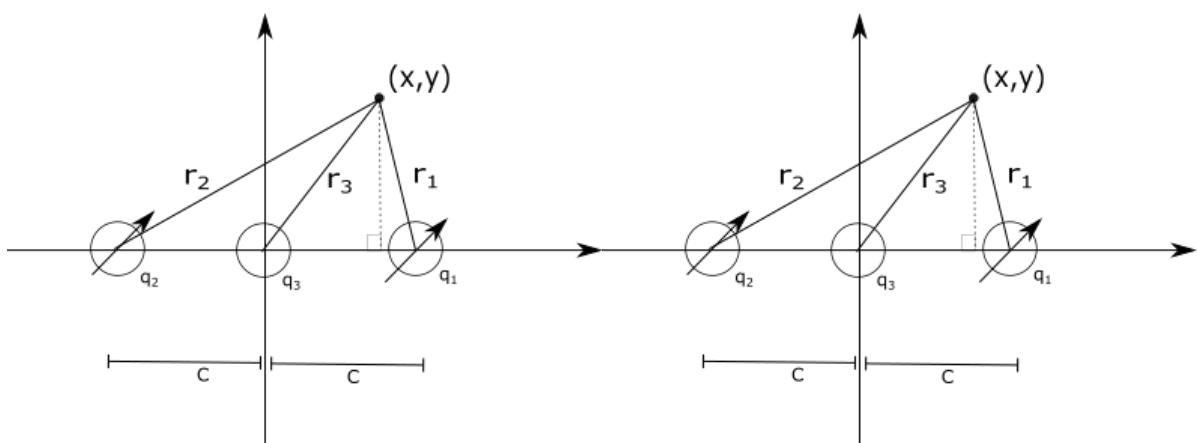
$$p_D(r_D, t_D) \equiv \frac{2\pi kh[p_i - p(r, t)]}{q\mu} = \frac{1}{2}[\ln(t_D/r_D^2) + 0,80907] \quad (3.34)$$

Para facilitar o entendimento da aplicação desse princípio, considere a Figura 3.9, onde está representado um sistema de coordenadas cartesianas para as três situações em que o programa irá fazer os cálculos, sendo C a metade da distância entre os dois poços, (x,y) um ponto qualquer do sistema, r_1, r_2 e r_3 a distância entre os poços e o ponto (x,y) . Após substituir as distâncias na Eq. 3.34 e fazer as manipulações matemáticas necessárias, as Eqs. 3.35, 3.38 e 3.42 permitem calcular a pressão em qualquer ponto do reservatório, em um tempo qualquer t . O Software desenvolvido irá calcular o valor das pressões para um ponto (x,y) qualquer escolhido pelo usuário.

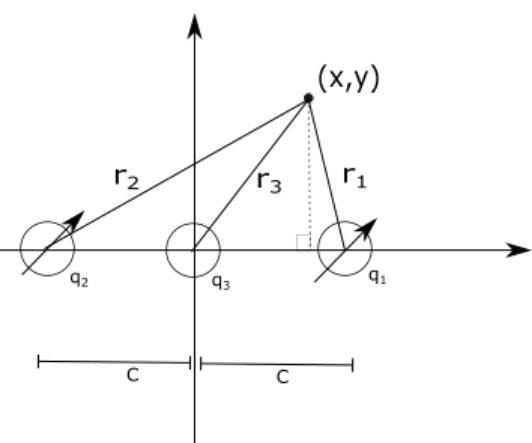
Conforme se observa, usando a aproximação de longo tempo (aproximação logarítmica da solução do modelo da fonte linear) para o comportamento transiente de pressão, Eq. 3.34, e aplicando o princípio da superposição de efeitos, obteve-se uma solução para fluxo permanente, já que não há dependência do tempo no lado direito da equação. Isso ocorre porque os poços injetor e produtor têm a mesma vazão, gerando então no reservatório um estado permanente de fluxo, ou seja, a pressão no reservatório é uma função somente da posição (ROSA ET AL., 2006).



(a) Sistema composto de um poço produtor e um injetor, em um sistema de coordenadas cartesianas.



(b) Sistema composto de um poço produtor e de dois injetores, em um sistema de coordenadas cartesianas.



(c) Sistema composto de um poço injetor e de dois produtores, em um sistema de coordenadas cartesianas.

Figura 3.9: Sistemas de coordenadas cartesianas.

Uma maneira de se analisar o comportamento da pressão (e consequentemente das linhas de fluxo) nos sistemas mostrados na Figura 3.10 é verificar a forma geométrica das linhas de pressão constante, ou seja, das linhas de mesmo potencial (equipotenciais), já que neste caso o potencial de fluxo e a pressão do fluido são iguais, pois o fluxo é horizontal. Para se analisar o comportamento das linhas equipotenciais basta admitir que o lado direito das Eqs. 3.35, 3.38 e 3.42 sejam constantes, isto é, considerar a situação em que o quociente entre as distâncias r_2 , r_1 e r_3 seja constante (ROSA ET AL., 2006). Com isso, após algumas manipulações, obtém-se as Eqs. 3.36, 3.39 e 3.43, que serão origem a gráficos mostrando o comportamento dessas linhas de pressão constante ao redor dos poços de injeção e produção.

Um outro aspecto de interesse é a determinação da área varrida pelo fluido injetado até um determinado instante. Por exemplo, no caso do esquema da Figura 3.10, onde são mostradas as dimensões do sistema, é interessante saber qual seria a área invadida pela água no momento que a água atingisse o poço produtor (“breakthrough”) e, nesse

instante, qual seria a distância percorrida pela água no sentido oposto ao do poço produtor. Para facilitar o desenvolvimento a ser apresentado, admita novamente um sistema de coordenadas cartesianas, em que o eixo horizontal coincide com a linha horizontal que passa pelos dois poços, como ilustrado na Figura 3.10. Para se analisar o comportamento do sistema no instante do “breakthrough”, é conveniente admitir também que o eixo horizontal tem origem no poço injetor, com valores de x crescentes para a direita neste caso, já que o poço injetor encontra-se à esquerda do produtor (ROSA ET AL., 2006). Com isso, obtém-se o valor da área invadida no instante do “breakthrough” (Figura 3.11), , $(A_{inv})_{BT}$, pode ser calculado pelas Eqs. 3.37, 3.41 e 3.44, que serão também dados de saída do programa.

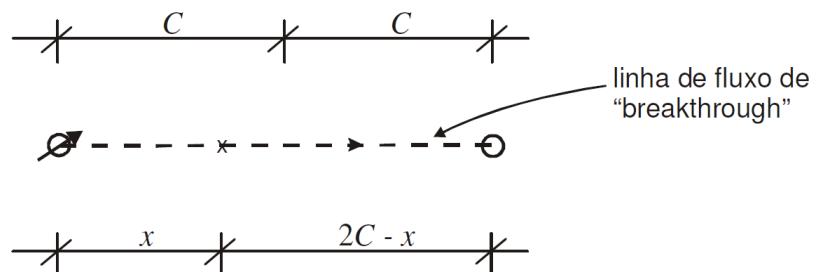


Figura 3.10: Sistema composto de dois poços: injetor e produtor (ROSA ET AL., 2006).

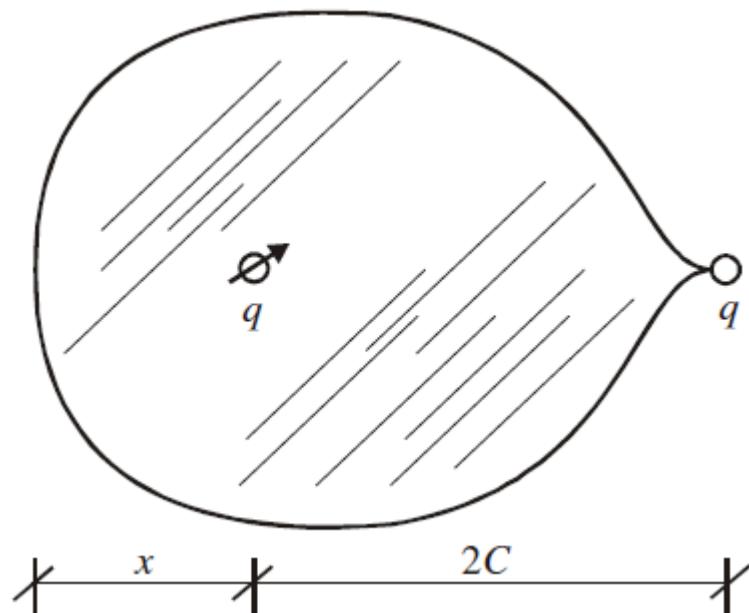


Figura 3.11: Ilustração da área invadida pela água no instante de “breakthrough” em um sistema composto de um injetor e de um produtor (ROSA ET AL., 2006)

- Caso 1 :

$$\frac{2\pi kh[p(x, y, t) - p_i]}{q\mu} = \ln(r_2/r_1), \quad (3.35)$$

$$x^4 + y^4 + 2x^2y^2 + x^2R - y^2R'' = \frac{C^2R''}{2}, \quad (3.36)$$

$$(A_{inv})_{BT} = \frac{\pi C^2}{2}. \quad (3.37)$$

- Caso 2 :

$$\frac{2\pi kh[p_i - p(x, y, t)]}{\mu q} = \ln \left(\frac{r_3^4}{r_1^2 r_2^2} \right)^{\frac{1}{4}}, \quad (3.38)$$

$$(x^2 + y^2)^2 + (y^2 - x^2)R' = -\frac{C^2 R'}{2}, \quad (3.39)$$

Transformando em coordenadas polares,

$$r^4 + r^2 [(-\cos(2\theta))] R' = -\frac{C^2 R'}{2}, \quad (3.40)$$

$$(A_{inv})_{BT} = 2\pi C^2 \quad (3.41)$$

- Caso 3 :

$$\frac{2\pi kh[p_i - p(x, y, t)]}{\mu q} = \ln \left(\frac{r_1^2 r_2^2}{r_3^4} \right)^{\frac{1}{4}}. \quad (3.42)$$

$$x^4 + y^4 + 2x^2y^2 + x^2R - y^2R'' = \frac{C^2 R''}{2}. \quad (3.43)$$

Transformando em coordenadas polares,

$$r^4 - r^2 [(sen^2(\theta) - \cos^2(\theta))] R'' = \frac{C^2 R''}{2}$$

$$(A_{inv})_{BT} = 2\pi C^2 \quad (3.44)$$

onde,

$$R = \frac{r_3^4}{r_1^2 r_2^2}.$$

$$R' = \frac{2RC^2}{R - 1}$$

$$R'' = \frac{2C^2}{R^{-1} - 1}.$$

3.3.3 Modelo de Fluxo Bifásido em Sistemas Estratificados (3D)

Dada uma perspectiva particular de injeção num reservatório heterogêneo (Figura 3.12), pretende-se prever informações como o tempo necessário para o “breakthrough”, recuperação de óleo no “breakthrough”, tempo de produção, desempenho de produção de óleo com a injeção de água, etc. Vários métodos foram propostos para fazer isso, cada um

diferindo na maneira de lidar com a heterogeneidade, cálculos de varredura, desempenho de injeção de água, eficiência do deslocamento e muitas outras variáveis que podem afetar a desempenho de injeção (SMITH; COBB, 1997). Como dito na especificação, será analisada a previsão de desempenho num reservatório com múltiplas camadas, com base nos métodos de Stiles (1949) e Dykstra-Parsons(1950), as formulações para tal são definidas a seguir:

Considerando o Método de Stiles, temos que:

- Posição da frante de avanço da água numa cada i qualquer ($i > j$):

$$X_i = X_j \left(\frac{k_i}{k_j} \right) \quad (3.45)$$

- Vazão de injeção numa camada j:

$$q_j = q_w = q_o = \frac{k_w A_j \Delta p}{\mu_w L} = \frac{k_o A_j \Delta p}{\mu_o L} \quad (3.46)$$

- Volume de óleo produzido por camada em condições padrão:

$$N_{pi} = \frac{V_{pi}(1 - S_w - S_{or})}{Bo} = \frac{W X_i h_i \phi (1 - S_w - S_{or})}{Bo} \quad (3.47)$$

- Volume de óleo produzido em toda a malha em condições padrão:

$$N_p = \frac{V_p(1 - S_w - S_{or})}{Bo} E_v = \frac{W L h_t \phi (1 - S_w - S_{or})}{Bo} \quad (3.48)$$

Considerando o Método de Dykstra-Parsons, temos que:

- Posição da frante de avanço da água numa cada i qualquer ($i > j$):

$$X_i = L \left[\frac{M - \sqrt{M^2 + (1 - M^2) \frac{k_i}{k_j}}}{M - 1} \right] \quad (3.49)$$

- A eficiência do varrido vertical é definida matematicamente como:

$$E_v = \frac{\sum_{i=l}^n X_i h_i}{L h_t} \quad (3.50)$$

- A vazão de injeção em cada camada como sendo dependente da razão de mobilidade M e posição X:

$$(Q_{inj})_i = \frac{k_i k_{rw} A \Delta p}{B_w \mu_w [X_i + M(L - X_i)]} \quad (3.51)$$

- Volume de óleo produzido por camada em condições padrão:

$$N_{pi} = \frac{V_{pi}(1 - S_w - S_{or})}{Bo} = \frac{W X_i h_i \phi (1 - S_w - S_{or})}{Bo} \quad (3.52)$$

- Volume de óleo produzido em toda a malha em condições padrão:

$$N_p = \frac{V_p(1 - S_w - S_{or})}{Bo} E_v = \frac{WLh_t\phi(1 - S_w - S_{or})}{Bo} \quad (3.53)$$

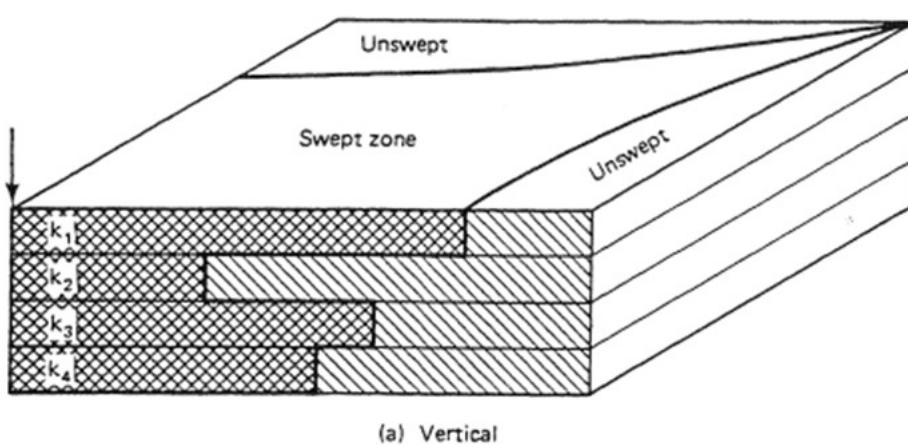


Figura 3.12: Reservatório Estratificado (SMITH, COBB , 1997).

3.4 Diagrama de Pacotes – assuntos

Com base na análise do domínio do software desenvolvido, foram identificados os seguintes pacotes:

- **Propriedades da Rocha e dos fluidos:** é um pacote que possui os dados das propriedades da rocha e dos fluidos, que compõem o meio poroso. Sua função é fornecer estas propriedades para o modelo de recuperação;
- **Métodos de Deslocamento Imiscível 1D, 2D e 3D:** é um pacote que contém diferentes métodos de deslocamentos por fluidos imiscíveis;
- **Recuperação Secundária:** é um pacote que envolve a injeção de água como método de recuperação;
- **Gnuplot:** envolve um utilitário de criação de gráficos orientado por linha de comando multi-plataforma;
- **Engenharia de Reservatório:** é um ramo da engenharia que fornece um estudo específico para fluxo em meios porosos em rochas reservatório.



Figura 3.13: Diagrama de Pacotes

Capítulo 4

AOO – Análise Orientada a Objeto

Apresenta-se neste capítulo a Análise Orientada a Objeto - AOO, as relações entre as classes, os atributos, os métodos e suas associações. A análise consiste em modelos estruturais dos objetos e seus relacionamentos, e modelos dinâmicos, apresentando as modificações do objeto com o tempo. O resultado da análise é um conjunto de diagramas que identificam os objetos e seus relacionamentos.

4.1 Diagramas de classes

O diagrama do software desenvolvido é composto por 15 classes que serão apresentadas em setores separadamente para melhor vizualização (Figura 4.1, 4.2, 4.3) e depois como elas se conectam (Figura 4.4) .

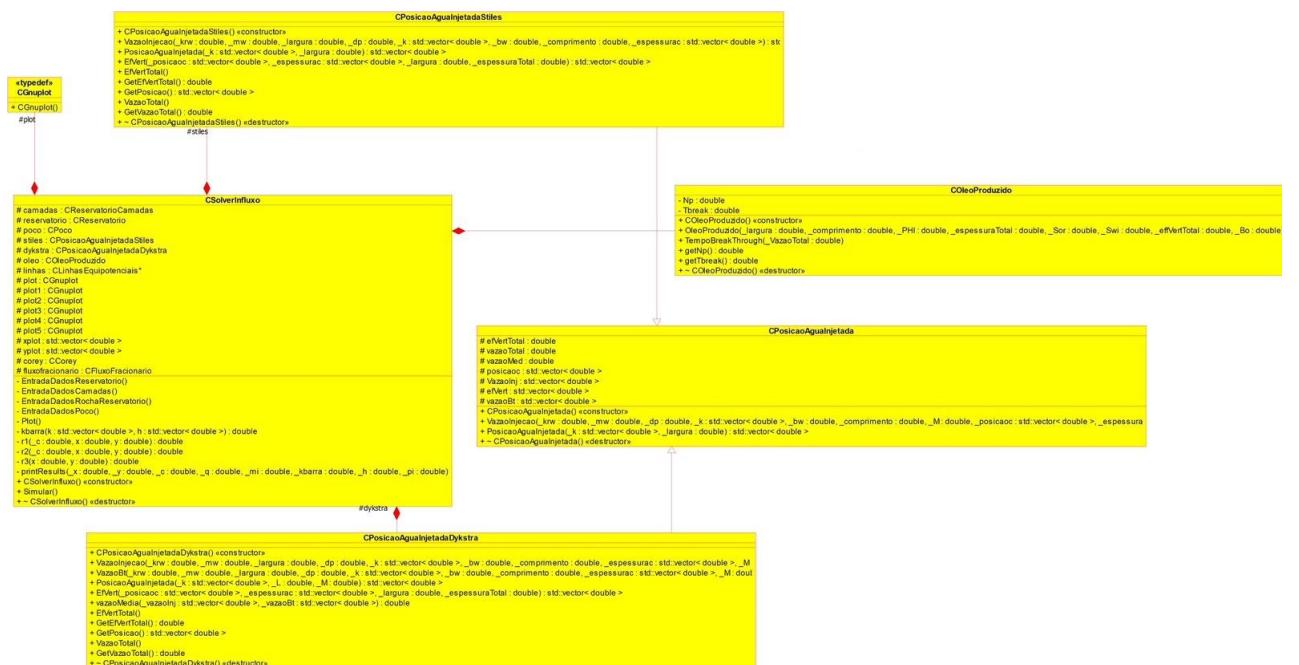


Figura 4.1: Diagrama de classes parte 1



Figura 4.2: Diagrama de classes parte 2

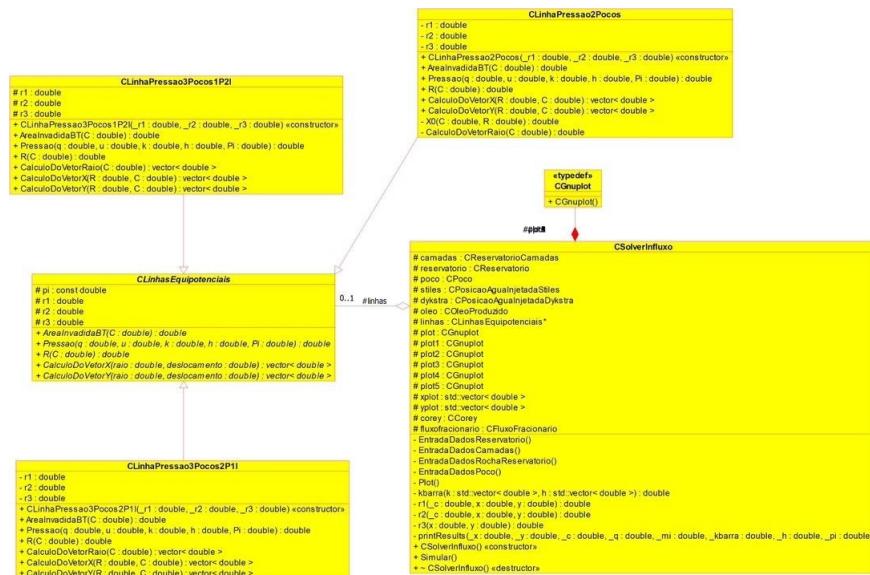


Figura 4.3: Diagrama de classes parte 3

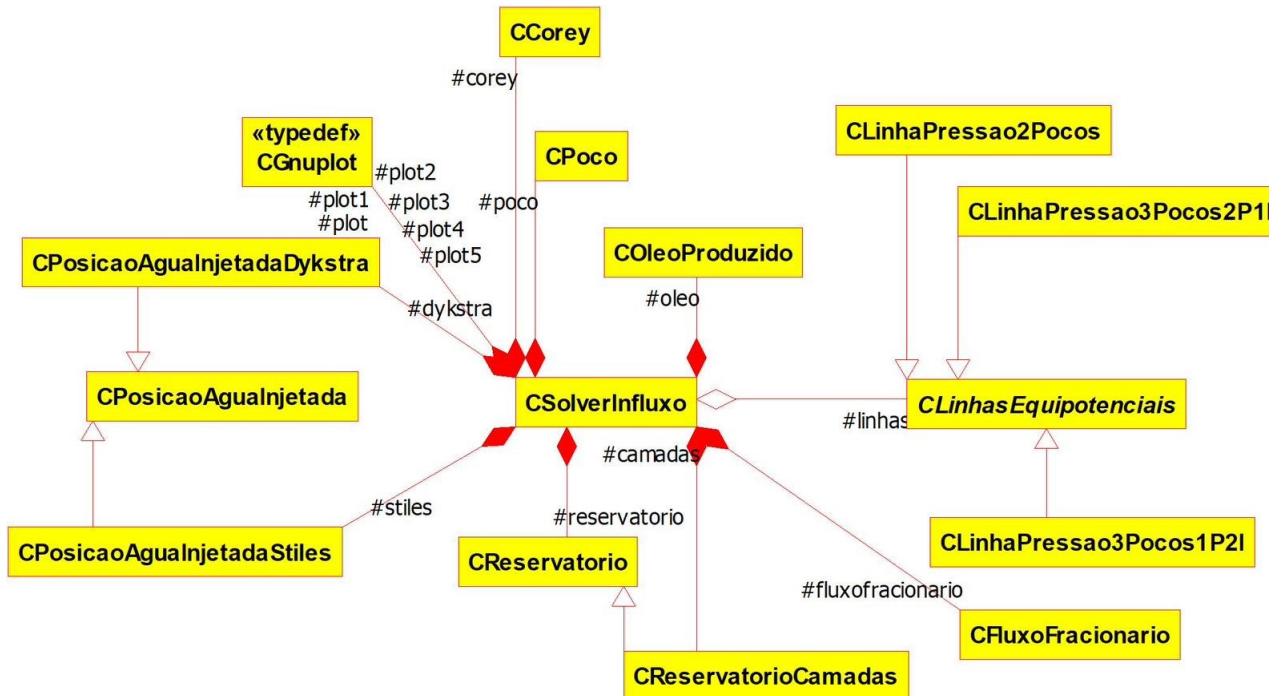


Figura 4.4: Diagrama de classes geral

4.1.1 Dicionário de classes

- Classe CCOREY: classe que representa o método para calcular permeabilidade relativa;
- Classe CFluxoFracionario: classe que representa o método para cálculo do fluxo fracionário;
- Classe CGnuplot: classe que possibilita a geração de gráficos usando o software externo Gnuplot;
- Classe CLinhaPressao2Pocos: classe que representa atributos e métodos referentes ao modelo de injeção com 2 poços, sendo um poço de produção e outra de injeção;
- Classe CLinhaPressao3Pocos1P2I: classe que representa atributos e métodos referentes ao modelo de injeção com 3 poços, sendo um poço de produção e dois de injeção;
- Classe CLinhaPressao3Pocos2P1I: classe que representa atributos e métodos referentes ao modelo de injeção com 3 poços, sendo dois poços de produção e um de injeção;
- Classe CLinhasequipotenciais: classe que representa todos atributos e métodos das configurações de malhas de injeção;
- Classe COleoProduzido: classe que representa os atributos e métodos do óleo;

- Classe CPoco: classe que representa os atributos do poço;
- Classe CPosicaoAguaInjetada: classe que representa os atributos para água injetada;
- Classe CPosicaoAguaInjetadaDykstra: classe que representa os métodos para água injetada utilizando o modelo de Dykstra;
- Classe CPosicaoAguaInjetadaStiles: classe que representa os métodos para água injetada utilizando o modelo de Stiles;
- Classe CReservatorio: classe que representa os atributos das rochas reservatórios;
- Classe CReservatorioCamadas: classe que representa os atributos da sequência de posicional;
- Classe CSolverInfluxo: classe mãe com atributos necessários para a simulação de recuperação secundária nos reservatórios.

4.2 Diagrama de seqüência – eventos e mensagens

O diagrama de seqüência enfatiza a troca de eventos e mensagens e sua ordem temporal. Contém informações sobre o fluxo de controle do software. Costuma ser montado a partir de um diagrama de caso de uso e estabelece o relacionamento dos atores (usuários e sistemas externos) com alguns objetos do sistema.

4.2.1 Diagrama de seqüência geral

Veja o diagrama de seqüência mostrado nas figuras 4.5, 4.6 e 4.7.

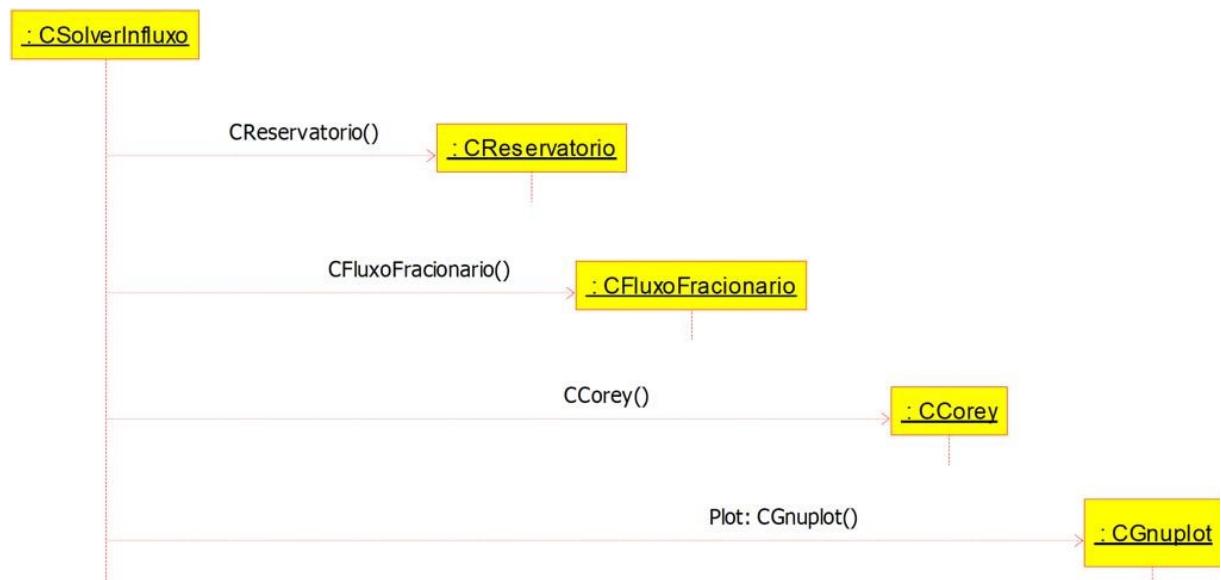


Figura 4.5: Diagrama de seqüência



Figura 4.6: Diagrama de seqüência

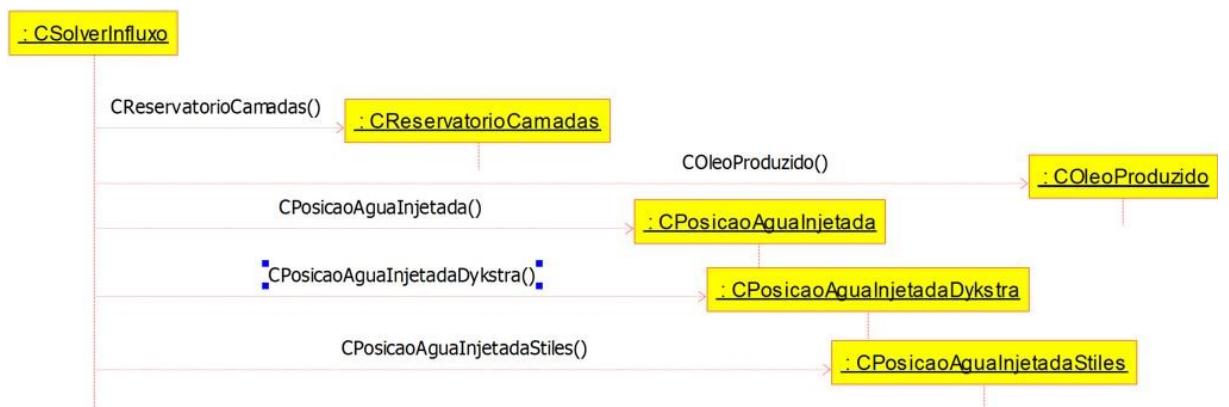


Figura 4.7: Diagrama de seqüência

4.3 Diagrama de comunicação – colaboração

No diagrama de comunicação o foco é a interação e a troca de mensagens e dados entre os objetos.

Veja na Figura 4.3 o diagrama de comunicação mostrando a classe CSolverInfluxo.

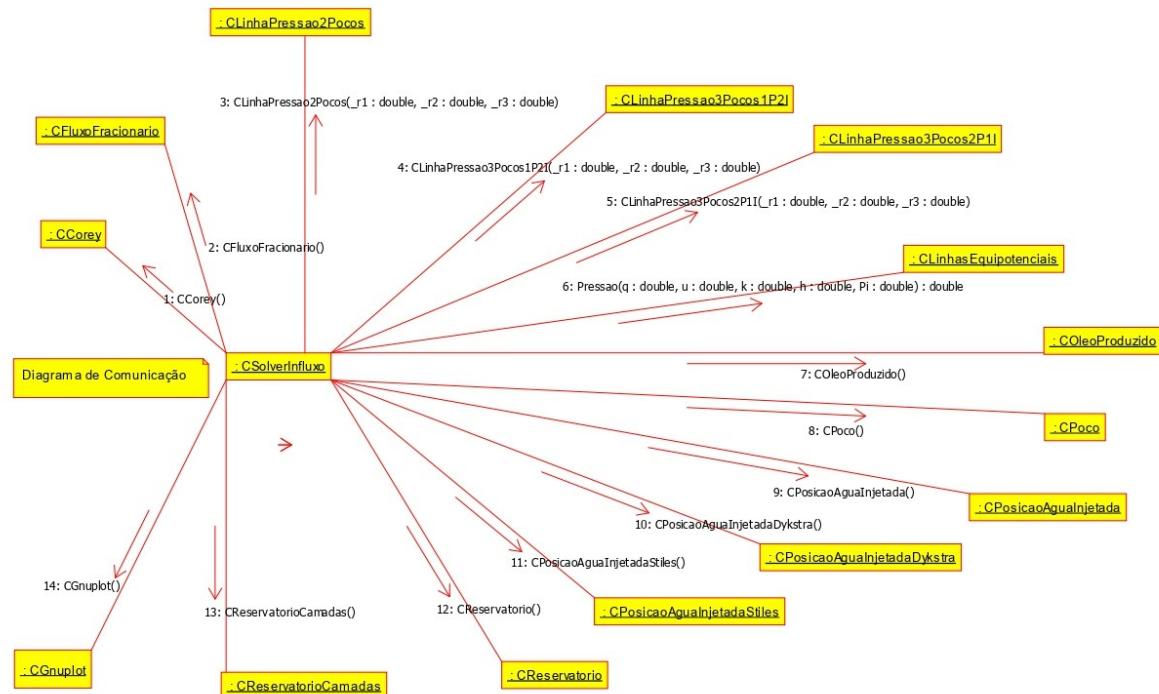


Figura 4.8: Diagrama de comunicação

4.4 Diagrama de máquina de estado

Um diagrama de máquina de estado representa os diversos estados que o objeto assume e os eventos que ocorrem ao longo de sua vida ou mesmo ao longo de um processo (histórico do objeto). É usado para modelar aspectos dinâmicos do objeto como mostrado nas Figuras 4.9, 4.10 e 4.11.

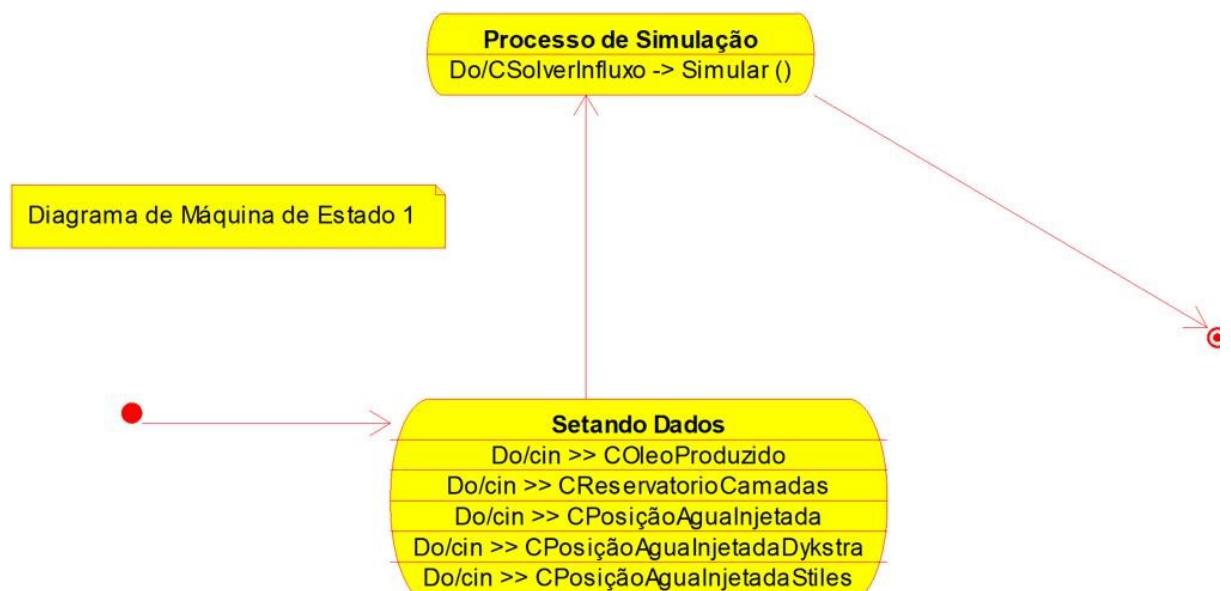


Figura 4.9: Diagrama de máquina de estado 1

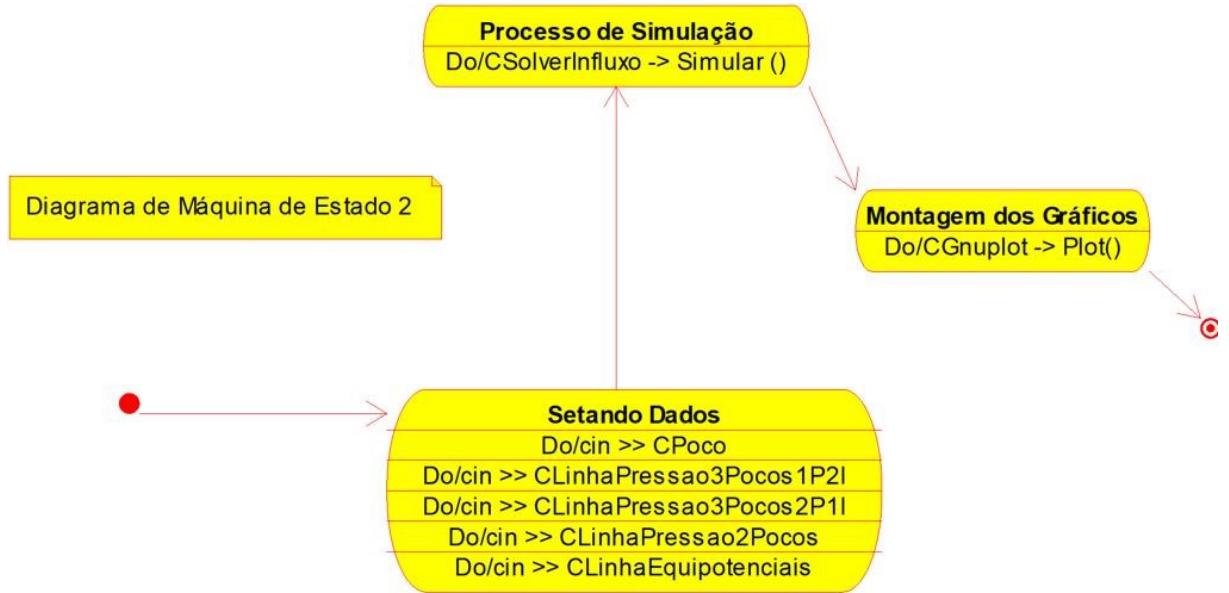


Figura 4.10: Diagrama de máquina de estado 2

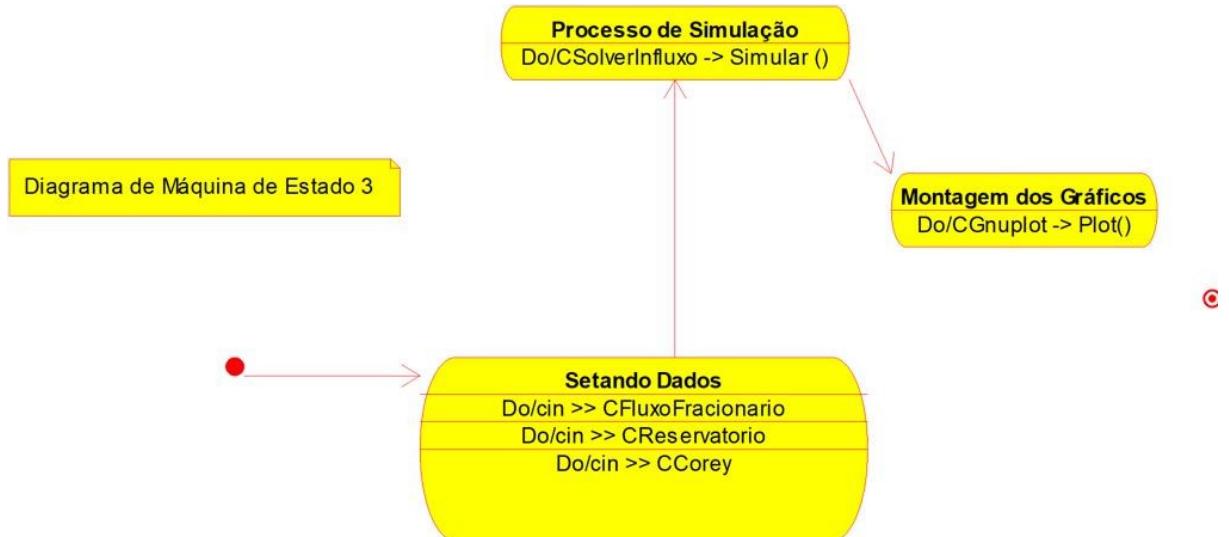


Figura 4.11: Diagrama de máquina de estado 3

4.5 Diagrama de atividades

O diagrama de atividades (Figura 4.5) corresponde a uma atividade específica do diagrama de máquina de estado, onde calcula-se :

1. Calculo do BT pelo metodo de Dykstra e Stiles (Figura 4.12);
2. Calculo do Fluxo Fracionário (Figura 4.13);
3. Calculo das Linhas Equipotencias, Pressão e Área invadida pela injeção de água (Figura 4.14).

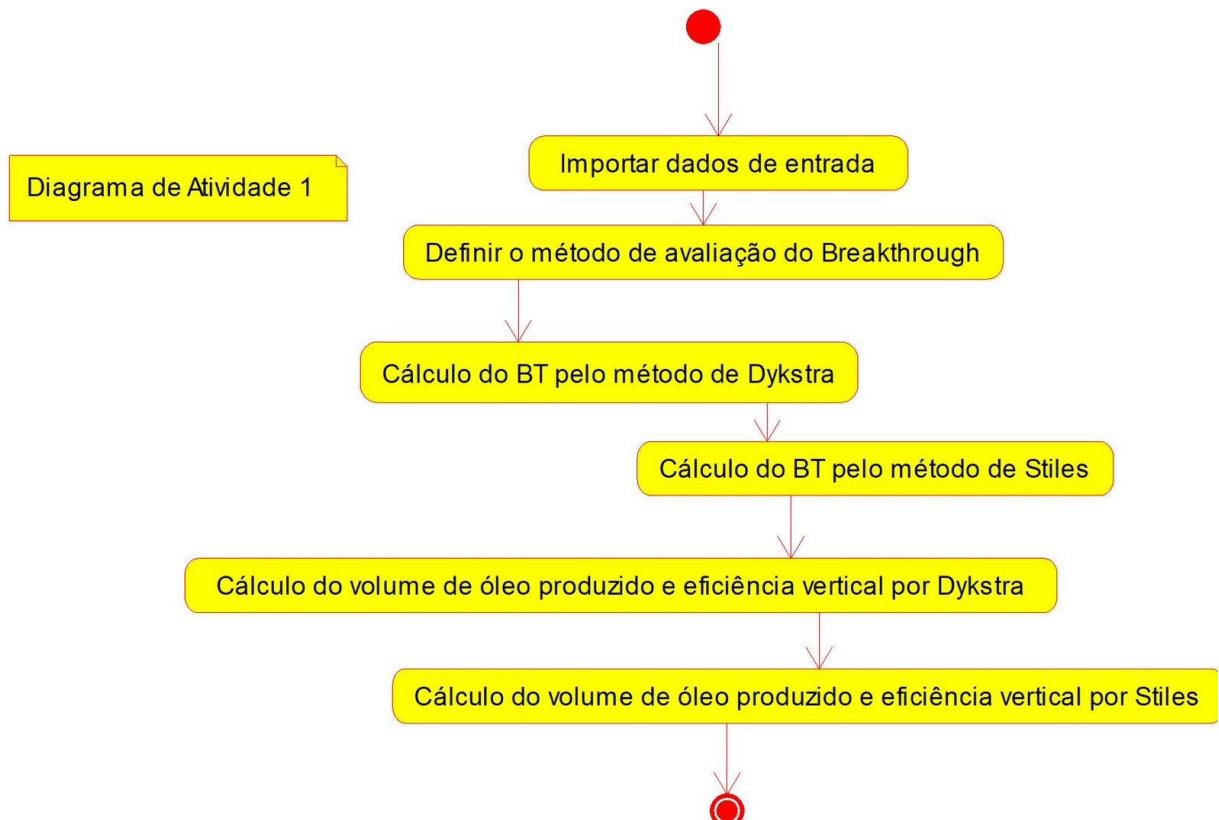


Figura 4.12: Diagrama de atividades 1

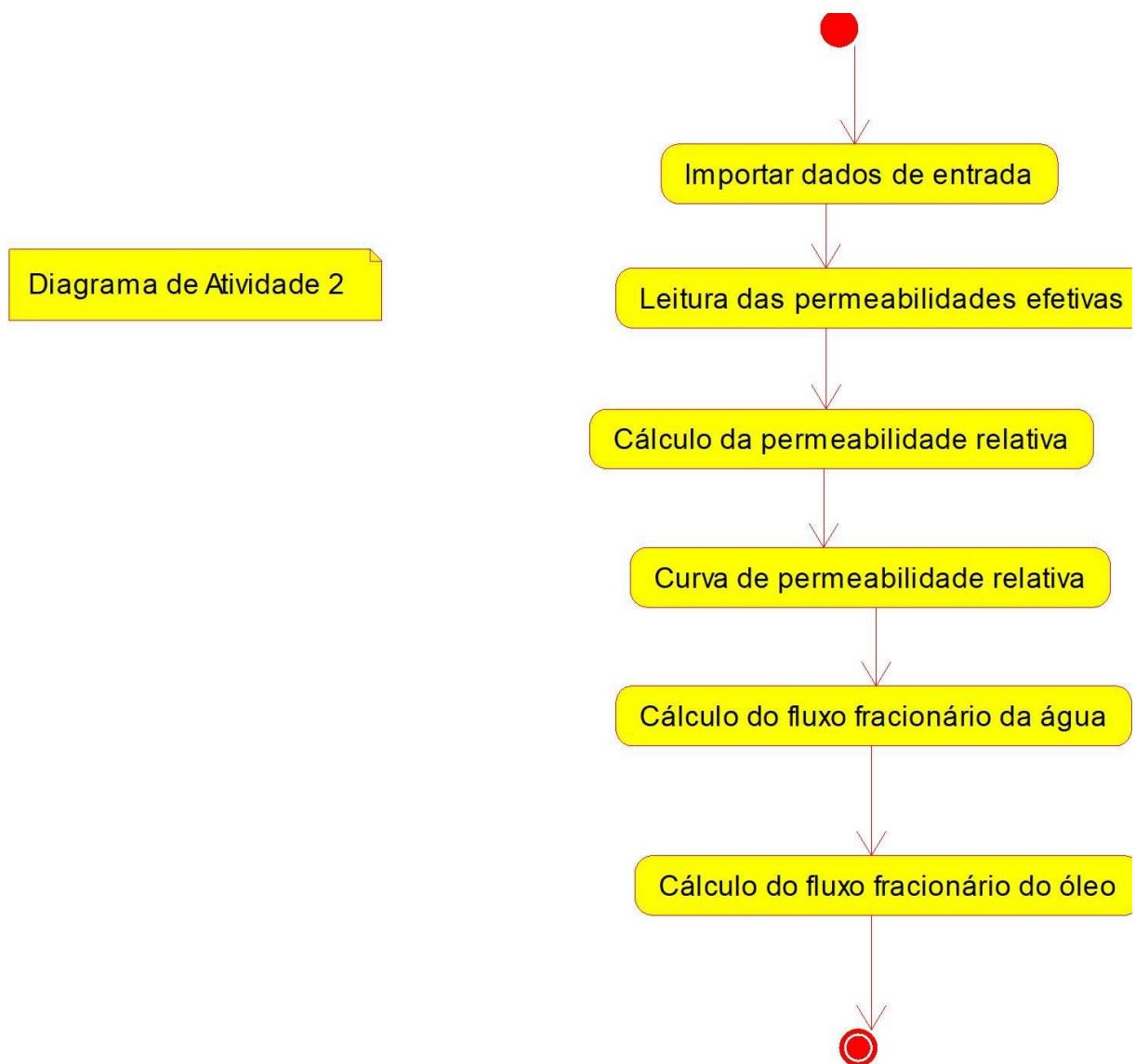


Figura 4.13: Diagrama de atividades 2

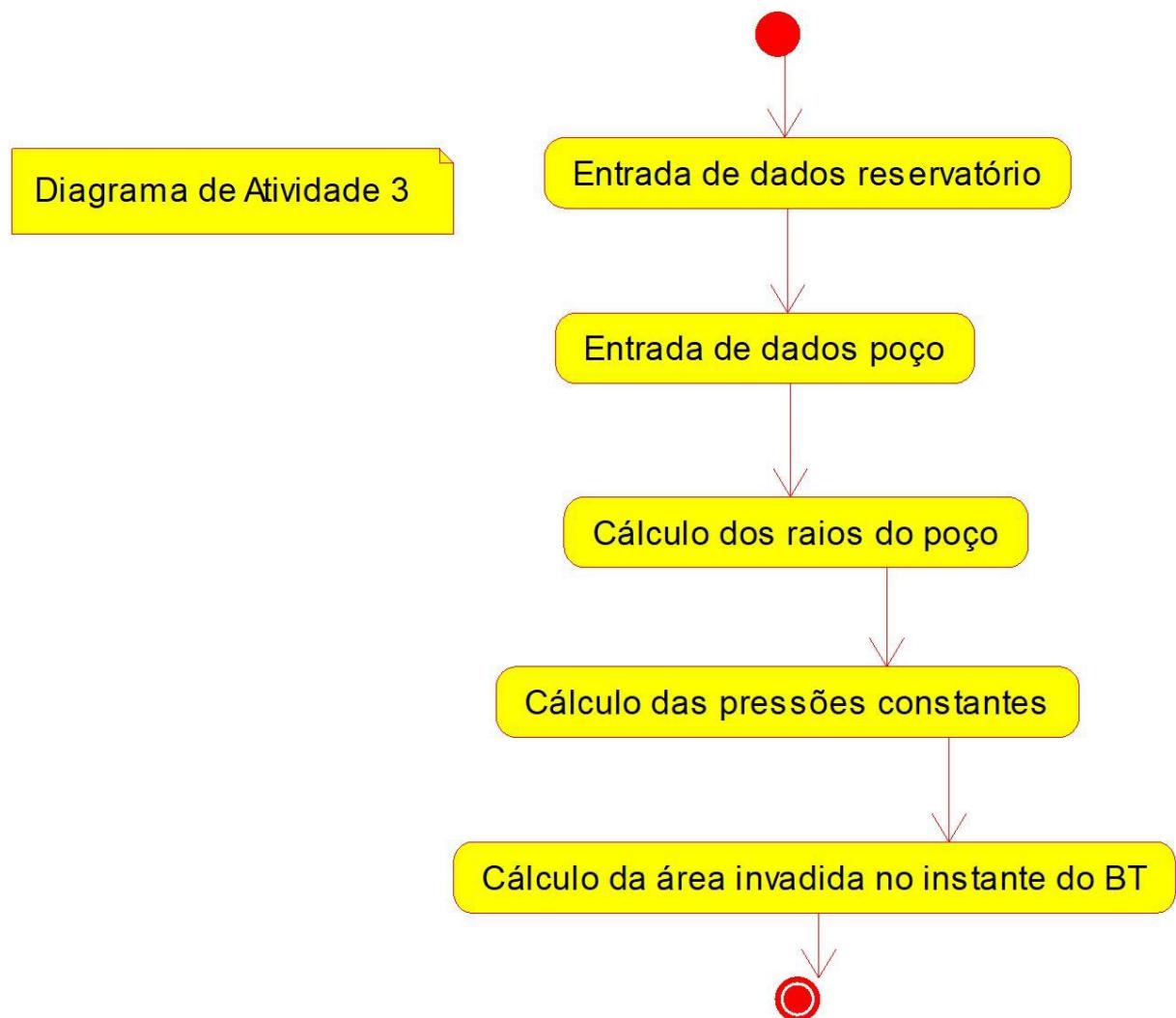


Figura 4.14: Diagrama de atividades 3

Capítulo 5

Projeto

Neste capítulo do projeto de engenharia veremos questões associadas ao projeto do sistema, incluindo protocolos, recursos, plataformas suportadas, implicações nos diagramas feitos anteriormente, diagramas de componentes e implantação. Na segunda parte revisamos os diagramas levando em conta as decisões do projeto do sistema.

5.1 Projeto do sistema

Depois da análise orientada ao objeto desenvolve-se o projeto do sistema, o qual envolve etapas como a definição dos protocolos, da interface API, o uso de recursos, a subdivisão do sistema em subsistemas, a alocação dos subsistemas ao hardware e a seleção das estruturas de controle, a seleção das plataformas do sistema, das bibliotecas externas, dos padrões de projeto, além da tomada de decisões conceituais e políticas que formam a infraestrutura do projeto.

Deve-se definir padrões de documentação, padrões para o nome das classes, padrões de retorno e de parâmetros em métodos, características da interface do usuário e características de desempenho.

Protocolos

- A única intercomunicação será entre o software desenvolvido e o software Gnuplot, que plotará os gráficos desejados pelo usuário;
- O software receberá dados via teclado;
- A interface utilizada será em modo texto;
- O software terá como saída de gráficos em arquivos de extensão .png.

Recursos

- O presente programa precisará utilizar o HD, o processador, o teclado, a tela, o mouse, a memória e demais componentes internos do computador;

Controle

- Não haverá necessidade de grande espaço na memória visto que o programa e seus componentes trabalham com dados relativamente pequenos;
- Neste projeto a maioria dos cálculos necessitam de estruturas de repetição;
- Neste projeto não há necessidade de uso de processos de processamento paralelo, pois os cálculos realizados requerem pouco esforço de processamento;

Plataformas

- Para a geração de gráficos será utilizado o software livre Gnuplot.
- Os ambientes de desenvolvimento serão o Embarcadero DevC++ (Windows) e Kate (Linux);
- O software irá operar nos sistemas operacionais Windows e GNU/Linux, sendo desenvolvido e testado em ambos os sistemas.
- Não haverá necessidade de grandes mudanças para tornar o programa multiplataforma pois a linguagem escolhida, C++, tem suporte em todos estes sistemas operacionais, [Bueno, 2003].

5.2 Projeto orientado a objeto – POO

O projeto orientado a objeto é a etapa posterior ao projeto do sistema. Baseia-se na análise, mas considera as decisões do projeto do sistema. Acrescenta a análise desenvolvida e as características da plataforma escolhida (hardware, sistema operacional e linguagem de programação). Passa pelo maior detalhamento do funcionamento do software, acrescentando atributos e métodos que envolvem a solução de problemas específicos não identificados durante a análise.

Envolve a otimização da estrutura de dados e dos algoritmos, a minimização do tempo de execução, de memória e de custos. Existe um desvio de ênfase para os conceitos da plataforma selecionada.

Como o projeto não alterou os diagramas apresentados na análise orientada a objeto, não houve necessidade de descrever os itens abaixo relacionados:

Efeitos do projeto no modelo estrutural;

Efeitos do projeto no modelo dinâmico;

Efeitos do projeto nos atributos;

Efeitos do projeto nos métodos;

Efeitos do projeto nas heranças;

Efeitos do projeto nas associações;

Efeitos do projeto nas otimizações;

5.3 Diagrama de componentes

O diagrama de componentes mostra a forma como os componentes do software se relacionam, suas dependências. Inclui itens como: componentes, subsistemas, executáveis, nós, associações, dependências, generalizações, restrições e notas. Exemplos de componentes são bibliotecas estáticas, bibliotecas dinâmicas, dlls, componentes Java, executáveis, arquivos de disco, código-fonte.

Veja a Figura 5.1 um exemplo de diagrama de componentes. Observe que este inclui muitas dependências, ilustrando as relações entre os arquivos.

Algumas observações úteis para o diagrama de componentes:

- De posse do diagrama de componentes, temos a lista de todos os arquivos necessários para compilar e rodar o software.
- Observe que um assunto/pacote pode se transformar em uma biblioteca e será incluído no diagrama de componentes.
- A ligação entre componentes pode incluir um estereótipo indicando o tipo de relacionamento ou algum protocolo utilizado.

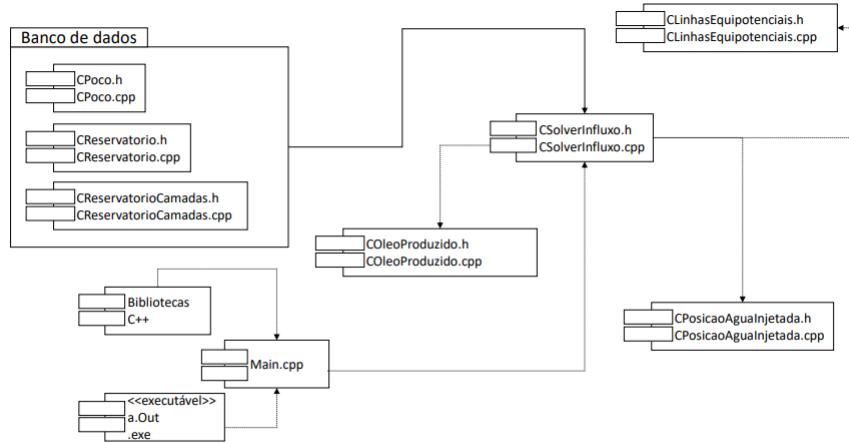


Figura 5.1: Diagrama de componentes

5.4 Diagrama de implantação

O diagrama de implantação é um diagrama de alto nível que inclui relações entre o sistema e o hardware e que se preocupa com os aspectos da arquitetura computacional escolhida. Seu enfoque é o hardware, a configuração dos nós em tempo de execução.

O diagrama de implantação deve incluir os elementos necessários para que o sistema seja colocado em funcionamento: computador, periféricos, processadores, dispositivos, nós, relacionamentos de dependência, associação, componentes, subsistemas, restrições e notas.

Veja na Figura 5.2 um exemplo de diagrama de implantação de um cluster. Observe a presença de um servidor conectado a um switch. Os nós do cluster (ou clientes) também estão conectados ao switch. Os resultados das simulações são armazenados em um servidor de arquivos (*storage*).

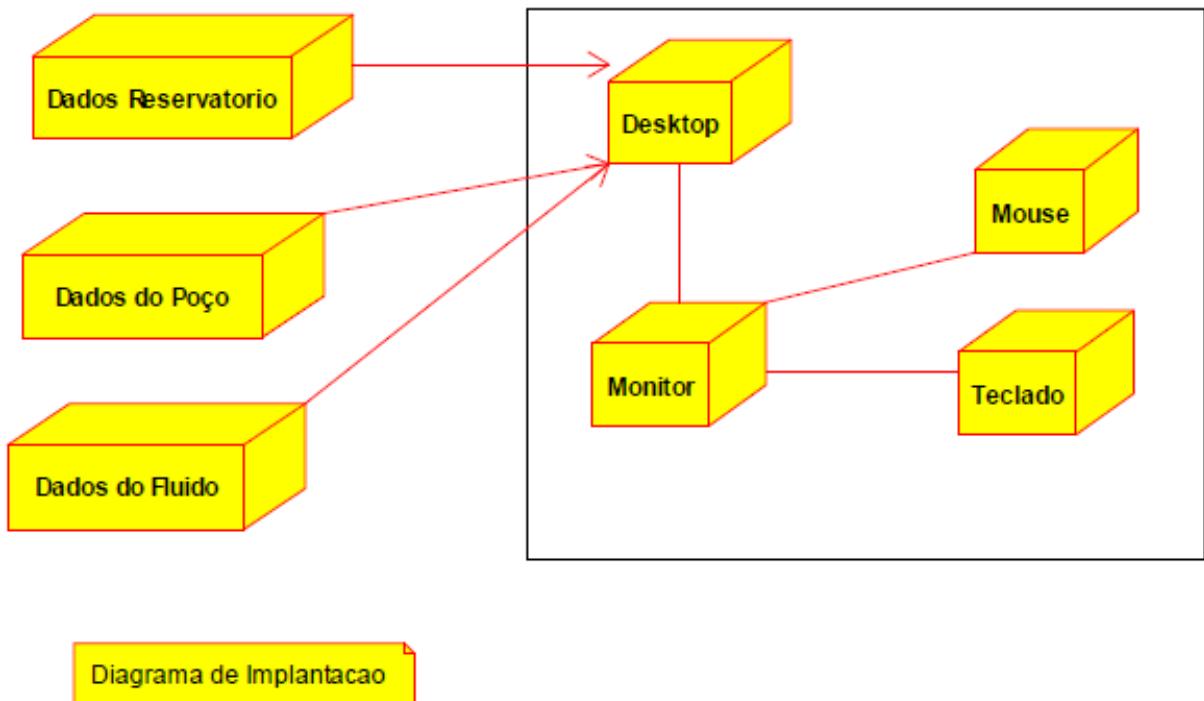


Figura 5.2: Diagrama de implantação

Capítulo 6

Implementação

Neste capítulo do projeto de engenharia apresentamos os códigos fonte que foram desenvolvidos.

6.1 Código fonte

Apresenta-se a seguir um conjunto de classes (arquivos .h e .cpp) além do programa **main** dos softwares.

Apresenta-se na listagem 6.1 o arquivo com código da classe CCorey.

Listing 6.1: Arquivo de cabeçalho da classe CCorey.

```
1 #ifndef CCOREY_H_
2 #define CCOREY_H_
3
4 #include <vector>
5
6 class CCorey {
7
8     protected:
9
10         std::vector<double> sw, kro, krw;
11         double KOrw, KOro, No, Nw;
12
13     public:
14
15         CCorey() {};
16
17         void calcSwn(double _swi, double _sor);
18         void calcKro(std::vector<double> _sw);
19         void calcKrw(std::vector<double> _sw);
```

```

20         void setK0rw(double _K0rw);
21         void setK0ro(double _K0ro);
22         void setNo(double _No);
23         void setNw(double _Nw);
24         std::vector<double> getSw();
25         std::vector<double> getKro();
26         std::vector<double> getKrw();
27
28     ~CCorey() {};
29
30};
31
32#endif

```

Apresenta-se na listagem 6.2 o arquivo de implementação da classe CCorey.

Listing 6.2: Arquivo de implementação da classe CCorey.

```

1 #include "CCorey.h"
2 #include <cmath>
3 #include <iostream>
4
5 void CCorey::calcSwn(double _swi, double _sor){
6     double a;
7     for (double i = _swi; i <=(1.0 - _sor)+.01; i+= .01){
8         a = (i - _swi)/(1.0 - _swi - _sor);
9         sw.push_back(a);
10    }
11
12}
13
14 void CCorey::calcKro(std::vector<double> _sw){
15
16     for(double sw:_sw)
17         kro.push_back(K0ro*pow(1 - sw, No));
18
19}
20
21 void CCorey::calcKrw(std::vector<double> _sw){
22
23     for(double sw:_sw)
24         krw.push_back(K0rw*pow(sw, Nw));
25
26}

```

```
27
28 void CCcorey::setKOrw(double _KOrw){
29
30     KOrw = _KOrw;
31
32 }
33
34 void CCcorey::setKOrO(double _KOrO){
35
36     KOrO = _KOrO;
37
38 }
39
40 void CCcorey::setNo(double _No){
41
42     No = _No;
43
44 }
45
46 void CCcorey::setNw(double _Nw){
47
48     Nw = _Nw;
49
50 }
51
52 std::vector<double> CCcorey::getSw(){
53
54     return sw;
55
56 }
57
58 std::vector<double> CCcorey::getKro(){
59
60     return kro;
61
62 }
63
64 std::vector<double> CCcorey::getKrw(){
65
66     return krw;
67
68 }
```

Apresenta-se na listagem ?? o arquivo de implementação da classe CFluxoFracionario.

Listing 6.3: Arquivo de implementação da classe CFluxoFracionario.

```

1 #ifndef CFLUXOFRACIONARIO_H_
2 #define CFLUXOFRACIONARIO_H_
3
4 #include <vector>
5
6 class CFluxoFracionario{
7
8     protected:
9
10         std::vector<double> Frw, Fro;
11
12     public:
13
14         CFluxoFracionario() {};
15
16         void calcFluxoFracionarioAgua(std::vector<double>
17             _krw, std::vector<double> _kro, double miw,
18             double mio);
19         void calcFluxoFracionarioOleo(std::vector<double>
20             _krw, std::vector<double> _kro, double miw,
21             double mio);
22         std::vector<double> getFluxoFracionarioAgua();
23         std::vector<double> getFluxoFracionarioOleo();
24
25     ~CFluxoFracionario() {};
26 };
27
28#endif

```

Apresenta-se na listagem 6.4 o arquivo de implementação da classe CFluxoFracionario.

Listing 6.4: Arquivo de implementação da classe CFluxoFracionario.

```

1 #include "CFluxoFracionario.h"
2
3 void CFluxoFracionario::calcFluxoFracionarioAgua(std::vector<double>
4     _krw, std::vector<double> _kro, double miw, double mio){
5
6     double a;

```

```

7         for(int i =0; i<_krw.size();i++){
8             a = (_krw[i]/miw)/(( _krw[i]/miw)+(_kro[i]/mio));
9             Frw.push_back(a);
10        }
11    }
12
13 void CFluxoFracionario::calcFluxoFracionarioOleo(std::vector<double>
14 > _krw, std::vector<double> _kro, double miw, double mio){
15
16     double a;
17
18     for(int i =0; i<_krw.size();i++){
19         a = (_kro[i]/mio)/(( _krw[i]/miw)+(_kro[i]/mio));
20         Fro.push_back(a);
21     }
22 }
23
24 std::vector<double> CFluxoFracionario::getFluxoFracionarioAgua(){
25
26     return Frw;
27
28 }
29
30 std::vector<double> CFluxoFracionario::getFluxoFracionarioOleo(){
31
32     return Fro;
33
34 }
```

Apresenta-se na listagem ?? o arquivo de implementação da classe CLinhaPressao2Pocos.

Listing 6.5: Arquivo de implementação da classe CLinhaPressao2Pocos.

```

1 #ifndef CLINHAPRESSAO2POCOS_H
2 #define CLINHAPRESSAO2POCOS_H
3
4 #include <iostream>
5 #include <vector>
6 #include <cmath>
7 #include <string>
8 #include <fstream>
9 #include <iostream>
10
```

```

11 #include "CLinhasequipotenciais.h"
12 using namespace std;
13
14 class CLinhaPressao2Pocos : public CLinhasequipotenciais {
15 public:
16     ///metodos específicos
17     CLinhaPressao2Pocos(double _r1, double _r2, double _r3) :
18         r1{ _r1 }, r2{ _r2 }, r3{ _r3 }{}
19     double AreaInvadidaBT(double C);
20     double Pressao(double q,double u,double k,double h,double
21                     Pi);
22     double R(double C);
23     vector<double> CalculoDoVetorX(double R, double C = 0.0);
24     vector<double> CalculoDoVetorY(double R, double C = 0.0);
25
26 private:
27     double X0(double C, double R);
28     double CalculoDoVetorRaio(double C);
29     double r1, r2, r3;
30 };
31 #endif // CLINHAPRESSAO2POCOS_H

```

Apresenta-se na listagem 6.6 o arquivo de implementação da classe CLinhaPressao2Pocos.

Listing 6.6: Arquivo de implementação da classe CLinhaPressao2Pocos.

```

1 #include "CLinhaPressao2Pocos.h"
2
3 double CLinhaPressao2Pocos::AreaInvadidaBT(double C){
4     return pow((4.0*pi*C),2)/3.0;
5 }
6
7 double CLinhaPressao2Pocos::Pressao(double q,double u,double k,
8                                     double h,double Pi){
9     return ((q*u*log(r2/r1))/(2.0*pi*k*h)) + Pi;
10 }
11 double CLinhaPressao2Pocos::R( double C) {
12     return (r2/r1)*(r2/r1);
13 }
14
15 vector<double> CLinhaPressao2Pocos::CalculoDoVetorX(double R,
16                                                       double C){
17     double x0 = X0(C,R);

```

```

17     int i;
18     vector<double> xplot;
19     for (double theta = 0; theta <= 2 * pi; theta = theta +
20           0.01) {
21         i = theta * 100;
22         xplot.push_back(x0 + (2 * (C)*sqrt(R) / (R - 1)) *
23                           cos(theta));
24     }
25
26 vector<double> CLinhaPressao2Pocos::CalculoDoVetorY(double R,
27   double C){
28     double y0 = 0;
29     int i;
30     vector<double> yplot;
31
32     for (double theta = 0; theta <= 2 * pi; theta = theta +
33           0.01) {
34       i = theta * 100;
35       yplot.push_back(y0 + (2 * (C)*sqrt(R) / (R - 1)) *
36                         sin(theta));
37     }
38     return yplot;
39   }
40
41 ///////////////////////////////////////////////////////////////////
42 double CLinhaPressao2Pocos::X0(double C, double R) {
43   return C * (R + 1.) / (R - 1.);
44 }
45
46 double CLinhaPressao2Pocos::CalculoDoVetorRaio(double C) { /**
47   preciso passar pro R E PRO RAIO ?
48   double r = R(C);
49   return 2 * (C)*sqrt(r) / (r - 1);
50 }
```

Apresenta-se na listagem ?? o arquivo de implementação da classe CLinhaPressao3Pocos1P2I.

Listing 6.7: Arquivo de implementação da classe CLinhaPressao3Pocos1P2I.

1

2 #ifndef CLINHAPRESSAO3POCOS1P2I_H

```

3 #define CLINHAPRESSAO3POCOS1P2I_H
4
5 #include "CLinhasequipotenciais.h"
6 #include <math.h>
7
8 using namespace std;
9
10 class CLinhaPressao3Pocos1P2I : public CLinhasequipotenciais {
11 public:
12     CLinhaPressao3Pocos1P2I(double _r1, double _r2, double _r3)
13         : r1{ _r1 }, r2{ _r2 }, r3{ _r3 }{};
14     double AreaInvadidaBT(double C);
15     double Pressao(double q, double u, double k, double h, double
16                     Pi);
17     double R(double C);
18     vector<double> CalculoDoVetorRaio(double C);
19     vector<double> CalculoDoVetorX(double R, double C = 0.0);
20     vector<double> CalculoDoVetorY(double R, double C = 0.0);
21
22 protected:
23     double r1, r2, r3;
24 };
25 #endif

```

Apresenta-se na listagem 6.8 o arquivo de implementação da classe CLinhaPressao3Pocos1P2I.

Listing 6.8: Arquivo de implementação da classe CLinhaPressao3Pocos1P2I.

```

1 #include "CLinhaPressao3pocos1P2I.h"
2
3 double CLinhaPressao3Pocos1P2I::AreaInvadidaBT(double C){
4     return 2*pi*C*C;
5 }
6
7 double CLinhaPressao3Pocos1P2I::Pressao(double q, double u, double k
8     , double h, double Pi){
9     return -(q*u/4*(log(r1*r1*r2*r2/(r3*r3*r3*r3)))/(2*pi*k*h))
10    + Pi;
11 }
12
13 double CLinhaPressao3Pocos1P2I::R(double C){
14     double r= pow(r1*r2,2)/pow(r3,4);
15     r = 2*C*C /(r-1);
16     return r;

```

```
15 }
16
17 vector<double> CLinhaPressao3Pocos1P2I::CalculoDoVetorRaio( double
18   C) {
19     double r = R( C );
20     vector<double> raio;
21     for (double theta = 0; theta <= 2*pi; theta=theta + 0.01 ){
22       raio.push_back(sqrt((-cos(2*theta)*r+sqrt((r*r*cos(2*theta)
23        )*(r*r*cos(2*theta)) + 2*C*C*r))/2 ));
24     }
25
26   return raio;
27 }
28
29 vector<double> CLinhaPressao3Pocos1P2I::CalculoDoVetorX(double R,
30   double C) {
31   double x0 = 0.0;
32   int i;
33   vector<double> xplot;
34   vector<double> raio = CalculoDoVetorRaio(C);
35
36   for (double theta = 0; theta <= 2 * pi; theta = theta +
37     0.01) {
38     i = theta * 100;
39     xplot.push_back(x0 + raio[i] * cos(theta));
40   }
41   return xplot;
42 }
43
44 vector<double> CLinhaPressao3Pocos1P2I::CalculoDoVetorY(double R,
45   double C) {
46   double y0 = 0.0;
47   int i;
48   vector<double> yplot;
49   vector<double> raio = CalculoDoVetorRaio(C);
50
51   for (double theta = 0; theta <= 2 * pi; theta = theta +
52     0.01) {
53     i = theta * 100;
54     yplot.push_back(y0 + raio[i] * sin(theta));
55   }
56   return yplot;
57 }
```

Apresenta-se na listagem ?? o arquivo de implementação da classe CLinhaPressao3Pocos2P1I.

Listing 6.9: Arquivo de implementação da classe CLinhaPressao3Pocos2P1I.

```

1 #ifndef CLINHAPRESSAO3POCOS2P1I_H
2 #define CLINHAPRESSAO3POCOS2P1I_H
3
4 #include "CLinhasEquipotenciais.h"
5 #include <math.h>
6
7 using namespace std;
8
9 class CLinhaPressao3Pocos2P1I : public CLinhasEquipotenciais {
10 public:
11     CLinhaPressao3Pocos2P1I(double _r1, double _r2, double _r3)
12         : r1{ _r1 }, r2{ _r2 }, r3{ _r3 }{}
13
14     //metodos específicos
15     double AreaInvadidaBT(double C);
16     double Pressao(double q, double u, double k, double h, double
17                     Pi);
18     double R(double C);
19     vector<double> CalculoDoVetorRaio(double C);
20     vector<double> CalculoDoVetorX(double R, double C = 0.0);
21     vector<double> CalculoDoVetorY(double R, double C = 0.0);
22 private:
23     double r1, r2, r3;
24 };
25 #endif

```

Apresenta-se na listagem 6.10 o arquivo de implementação da classe CLinhaPressao3Pocos2P1I.

Listing 6.10: Arquivo de implementação da classe CLinhaPressao3Pocos2P1I.

```

1 #include "CLinhaPressao3pocos2P1I.h"
2
3 double CLinhaPressao3Pocos2P1I::AreaInvadidaBT(double C){
4     return 2*pi*C*C;
5 }
6
7 double CLinhaPressao3Pocos2P1I::Pressao(double q, double u, double k,
8                                         double h, double Pi){
9     return -(q*u*log(pow(r3,4)/(pow(r1,2)*pow(r2,2)))/(8*pi*k*
10               h) + Pi;

```

```
9 }
10
11 double CLinhaPressao3Pocos2P1I::R(double C){
12     double r= (r3*r3*r3*r3)/(r1*r1*r2*r2);
13     r=(2*r* pow(C,2))/(r-1);
14     return r;
15 }
16
17 vector<double> CLinhaPressao3Pocos2P1I::CalculoDoVetorRaio( double
C){
18     double r = R( C);
19     vector<double> raio;
20     for (double theta = 0;theta <= 2*pi; theta=theta + 0.01 ){
21         raio.push_back(sqrt((cos(2*theta)*r + sqrt(pow(r*r*cos(2*
theta),2 ) - (4*C*C*r)/2))/2));
22     }
23     return raio;
24 }
25
26 vector<double> CLinhaPressao3Pocos2P1I::CalculoDoVetorX(double R,
double C) {
27     double x0 = 0;
28     int i;
29     vector<double> xplot;
30     vector<double> raio = CalculoDoVetorRaio(C);
31
32     for (double theta = 0; theta <= 2 * pi; theta = theta +
0.01) {
33         i = theta * 100;
34         xplot.push_back(x0 + raio[i] * cos(theta));
35     }
36     return xplot;
37 }
38
39 vector<double> CLinhaPressao3Pocos2P1I::CalculoDoVetorY(double R,
double C) {
40     double y0 = 0;
41     int i;
42     vector<double> yplot;
43     vector<double> raio = CalculoDoVetorRaio(C);
44
45     for (double theta = 0; theta <= 2 * pi; theta = theta +
```

```

        0.01) {
46            i = theta * 100;
47            yplot.push_back(y0 + raio[i] * sin(theta));
48        }
49        return yplot;
50    }

```

Apresenta-se na listagem ?? o arquivo de implementação da classe CLinhasequipotenciais.

Listing 6.11: Arquivo de implementação da classe CLinhasequipotenciais.

```

1#ifndef CLINHASEQUIPOTENCIAIS_H
2#define CLINHASEQUIPOTENCIAIS_H
3
4#include <vector>
5using namespace std;
6class CLinhasequipotenciais {
7public:
8    // declaração de método virtual puro, o igual a zero indica
9    // que vai ser implementada nas filhas
10   virtual double AreaInvadidaBT(double C) = 0;
11   virtual double Pressao(double q, double u, double k, double h,
12                           double Pi) = 0;
13   virtual double R(double C) = 0;
14   virtual vector<double> CalculoDoVetorX(double raio, double
15                                               deslocamento = 0.0) = 0;
16   virtual vector<double> CalculoDoVetorY(double raio, double
17                                               deslocamento = 0.0) = 0;
18protected:
19   const double pi = 3.14159265;
20   double r1, r2, r3;
21};
22#endif

```

Apresenta-se na listagem 6.12 o arquivo de implementação da classe CLinhasequipotenciais.

Listing 6.12: Arquivo de implementação da classe CLinhasequipotenciais.

```
#include "CLinhasequipotenciais.h"
```

Apresenta-se na listagem ?? o arquivo de implementação da classe COleoProduzido.

Listing 6.13: Arquivo de implementação da classe COleoProduzido.

```
#ifndef COLEOPRODUZIDO_H_

```

```

2 #define COLEOPRODUZIDO_H_
3
4 class COleoProduzido {
5
6     double Np, Tbreak;
7
8 public:
9
10    COleoProduzido();
11
12    void OleoProduzido(double _largura, double _comprimento, double
13                      _PHI, double _espessuraTotal, double _Sor, double _Swi,
14                      double _effVertTotal, double _Bo);
15    void TempoBreakThrough(double _VazaoTotal);
16    double getNp();
17    double getTbreak();
18
19 ~COleoProduzido();
20
21#endif

```

Apresenta-se na listagem 6.14 o arquivo de implementação da classe COleoProduzido.

Listing 6.14: Arquivo de implementação da classe COleoProduzido.

```

1 #include "COleoProduzido.h"
2
3 void COleoProduzido::OleoProduzido(double _largura, double
4                                   _comprimento, double _PHI, double _espessuraTotal, double _Sor,
5                                   double _Swi, double _effVertTotal, double _Bo)
6 {
7
8     Np = (_largura*_comprimento*_PHI*_espessuraTotal*_effVertTotal
9           *(1.0 - _Sor - _Swi))/_Bo;
10
11
12 void COleoProduzido::TempoBreakThrough(double _VazaoTotal)
13 {
14     Tbreak = Np/_VazaoTotal;
15 }
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
287
288
289
289
290
291
292
293
294
295
296
297
297
298
299
299
300
301
302
303
304
305
306
307
307
308
309
309
310
311
312
313
314
315
316
317
317
318
319
319
320
321
322
323
324
325
326
327
327
328
329
329
330
331
332
333
334
335
336
337
337
338
339
339
340
341
342
343
344
345
346
347
347
348
349
349
350
351
352
353
354
355
356
357
357
358
359
359
360
361
362
363
364
365
366
367
367
368
369
369
370
371
372
373
374
375
376
377
377
378
379
379
380
381
382
383
384
385
386
387
387
388
389
389
390
391
392
393
394
395
396
397
397
398
399
399
400
401
402
403
404
405
406
407
407
408
409
409
410
411
412
413
414
415
415
416
417
417
418
419
419
420
421
422
423
424
425
426
426
427
428
428
429
429
430
431
432
433
434
435
436
437
437
438
439
439
440
441
442
443
444
445
446
447
447
448
449
449
450
451
452
453
454
455
456
457
457
458
459
459
460
461
462
463
464
465
466
466
467
468
468
469
469
470
471
472
473
474
475
476
476
477
478
478
479
479
480
481
482
483
484
485
486
487
487
488
489
489
490
491
492
493
494
495
496
497
497
498
499
499
500
501
502
503
504
505
506
507
507
508
509
509
510
511
512
513
514
515
516
516
517
518
518
519
519
520
521
522
523
524
525
526
526
527
528
528
529
529
530
531
532
533
534
535
536
537
537
538
539
539
540
541
542
543
544
545
546
547
547
548
549
549
550
551
552
553
554
555
556
557
557
558
559
559
560
561
562
563
564
565
566
566
567
568
568
569
569
570
571
572
573
574
575
576
576
577
578
578
579
579
580
581
582
583
584
585
586
586
587
588
588
589
589
590
591
592
593
594
595
596
596
597
598
598
599
599
600
601
602
603
604
605
605
606
607
607
608
608
609
609
610
611
612
613
614
614
615
615
616
616
617
617
618
618
619
619
620
621
622
623
624
625
625
626
627
627
628
628
629
629
630
631
632
633
634
635
635
636
637
637
638
638
639
639
640
641
642
643
644
645
645
646
647
647
648
648
649
649
650
651
652
653
654
655
655
656
657
657
658
658
659
659
660
661
662
663
664
665
665
666
667
667
668
668
669
669
670
671
672
673
674
675
675
676
677
677
678
678
679
679
680
681
682
683
684
685
685
686
687
687
688
688
689
689
690
691
692
693
694
694
695
696
696
697
697
698
698
699
699
700
701
702
703
703
704
705
705
706
706
707
707
708
708
709
709
710
711
712
713
713
714
714
715
715
716
716
717
717
718
718
719
719
720
721
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
731
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
741
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
751
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
761
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
771
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
781
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1521
1522
1522
1523
1523
1524
1524
1525
1525
1526
1526
1527
1527
1528
1528
1529
1529
1530
1531
1532
1532
1533
1533
1534
1534
1535
1535
1536
1536
1537
1537
1538
1538
1539
1539
1540
1541
1542
1542
1543
1543
1544
1544
1545
1545
1546
1546
1547
1547
1548
1548
1549
1549
1550
1551
1552
1552
1553
1553
1554
1554
1555
1555
1556
1556
1557
1557
1558
1558
1559
1559
1560
1561
1562
1562
1563
1563
1564
1564
1565
1565
1566
1566
1567
1567
1568
1568
1569
1569
1570
1571
1572
1572
1573
1573
1574
1574
1575
1575
1576
1576
1577
1577
1578
1578
1579
1579
1580
1581
1582
1582
1583
1583
1584
1584
1585
1585
1586
1586
1587
1587
1588
1588
1589
1589
1590
1591
1592
1592
1593
1593
1594
159
```

```
15 double COleoProduzido::getTbreak()
16 {
17     return Tbreak;
18 }
19
20 double COleoProduzido::getNp()
21 {
22     return Np;
23 }
```

Apresenta-se na listagem ?? o arquivo de implementação da classe CPoco.

Listing 6.15: Arquivo de implementação da classe CPoco.

```
1 #ifndef CPOCO_H_
2 #define CPOCO_H_
3
4 class CPoco {
5
6     protected:
7
8         double x, y, c, Pi;
9
10    public:
11
12        CPoco() {};
13
14        CPoco(double _x, double _y, double _c, double _Pi):
15            x(_x), y(_y), c(_c), Pi(_Pi){};
16
17        //metodos set
18        void setX(double _x);
19        void setY(double _y);
20        void setC(double _c);
21        void setPi(double _pi);
22
23        //metodos get
24        double getX();
25        double getY();
26        double getC();
27        double getPi();
28
29        ~CPoco() {};
```

```
30
31 } ;
32
33 #endif
```

Apresenta-se na listagem 6.16 o arquivo de implementação da classe CPoco.

Listing 6.16: Arquivo de implementação da classe CFormaReservatorio.

```
1 #include "CPoco.h"
2
3 void CPoco::setX(double _x){
4
5     x = _x;
6
7 }
8
9 void CPoco::setY(double _y){
10
11    y = _y;
12
13 }
14
15 void CPoco::setC(double _c){
16
17    c = _c;
18
19 }
20
21 void CPoco::setPi(double _pi){
22
23    Pi = _pi;
24
25 }
26
27 double CPoco::getX(){
28
29    return x;
30
31 }
32
33 double CPoco::getY(){
34
35    return y;
```

```

36
37 }
38
39 double CPoco::getC(){
40
41     return c;
42
43 }
44
45 double CPoco::getPi(){
46
47     return Pi;
48
49 }
```

Apresenta-se na listagem ?? o arquivo de implementação da classe CPosicaoAguaInjetada.

Listing 6.17: Arquivo de implementação da classe CPosicaoAguaInjetada.

```

1 #ifndef CPOSICAOAGUAINJETADA_H_
2 #define CPOSICAOAGUAINJETADA_H_
3
4 #include <vector>
5
6 class CPosicaoAguaInjetada
7 {
8
9     protected:
10
11     double efVertTotal, vazaoTotal, vazaoMed;
12             std::vector <double> posicaoc, VazaoInj, efVert,
13             vazaoBt;
14
15     public:
16         CPosicaoAguaInjetada(){};
17
18         std::vector <double> VazaoInjecao(double _krw,
19             double _mw, double _dp, std::vector <double> _k,
20             double _bw, double _comprimento, double _M, std
21             ::vector <double> _posicaoc, std::vector <double
22             > _espessurac);
23         std::vector <double> PosicaoAguaInjetada(std::
```

```

20           vector <double> _k, double _largura);
21
22     ~CPosicaoAguaInjetada() {};
23
24
25 #endif

```

Apresenta-se na listagem 6.18 o arquivo de implementação da classe CPosicaoAguaInjetada.

Listing 6.18: Arquivo de implementação da classe CPosicaoAguaInjetada.

```

1 #include "CPosicaoAguaInjetada.h"
2
3 using namespace std;
4
5 vector <double> CPosicaoAguaInjetada::VazaoInjecao(double _krw,
6   double _mw, double _dp, vector <double> _k, double _bw, double
7   _comprimento, double _M, vector <double> _posicaoc, vector <
8   double> _espessurac)
9 {
10
11
12   return VazaoInj;
13
14
15   return posicaoc;
16
17 }

```

Apresenta-se na listagem ?? o arquivo de implementação da classe CPosicaoAguaInjetadaDykstra.

Listing 6.19: Arquivo de implementação da classe CPosicaoAguaInjetadaDykstra.

```

1 ifndef CPOSICAOAGUAINJETADADYKSTRA_H_
2 define CPOSICAOAGUAINJETADADYKSTRA_H_
3
4 include "CPosicaoAguaInjetada.h"
5
6 class CPosicaoAguaInjetadaDykstra : CPosicaoAguaInjetada {

```

```

7
8     public:
9
10    CPosicaoAguaInjetadaDykstra() {};
11
12    std::vector <double> VazaoInjecao(double _krw,
13                                       double _mw, double _largura, double _dp, std::
14                                       vector <double> _k, double _bw, double
15                                       _comprimento, std::vector <double> _espessurac,
16                                       double _M);
17
18    std::vector <double> VazaoBt(double _krw, double _mw
19                               , double _largura, double _dp, std::vector <
20                               double> _k, double _bw, double _comprimento, std
21                               ::vector <double> _espessurac, double _M, std::
22                               vector<double> posicao);
23
24    std::vector <double> PosicaoAguaInjetada(std::
25                           vector <double> _k, double _L, double _M);
26
27    std::vector <double> EfVert(std::vector<double>
28                               _posicaoc, std::vector<double> _espessurac,
29                               double _largura, double _espessuraTotal);
30
31
32    double vazaoMedia(std::vector<double> _vazaoInj,
33                      std::vector<double> _vazaoBt);
34
35
36    void EfVertTotal();
37    double GetEfVertTotal();
38    std::vector<double> GetPosicao();
39    void VazaoTotal();
40    double GetVazaoTotal();
41
42
43    ~CPosicaoAguaInjetadaDykstra() {};
44
45
46};

47
48#endif

```

Apresenta-se na listagem 6.20 o arquivo de implementação da classe CPosicaoAguaInjetadaDykstra.

Listing 6.20: Arquivo de implementação da classe CPosicaoAguaInjetadaDykstra.

```

1 #include "CPosicaoAguaInjetadaDykstra.h"
2
3 using namespace std;

```

```
4
5 #include <cmath>
6
7
8 vector <double> CPosicaoAguaInjetadaDykstra::VazaoInjecao(double
9   _krw, double _mw, double _largura, double _dp, std::vector <double>
10  _k, double _bw, double _comprimento, std::vector <double>
11  _espessurac, double _M)
12 {
13
14     double A;
15
16     for (int i = 0; i < _k.size(); i++)
17     {
18         A = (_krw*_comprimento*_dp*86400.00*_k[i]*
19               _espessurac[i])/(_bw*_mw*_largura*_M);
20         VazaoInj.push_back(A);
21     }
22
23
24     return VazaoInj;
25
26
27
28
29
30
31
32
33
34
35
36
```

21 }
22
23 vector <double> CPosicaoAguaInjetadaDykstra::VazaoBt(double _krw,
24 double _mw, double _largura, double _dp, std::vector <double> _k,
25 double _bw, double _comprimento, std::vector <double>
26 _espessurac, double _M, vector<double> posicao)
27 {
28 double A;
29
30 for (int i = 0; i < _k.size(); i++)
31 {
32 A = (_krw*_comprimento*_dp*86400.00*_k[i]*
33 _espessurac[i])/(_bw*_mw*(posicao[i] + _M*
34 _largura - posicao[i]));
35 vazaoBt.push_back(A);
36 }

```
37 vector <double> CPosicaoAguaInjetadaDykstra::PosicaoAguaInjetada(38     vector<double> _k, double _L, double _M){39     double A;4041     for (int i = 0; i < _k.size() ; i++)42     {43         A = _L*((_M - sqrt(pow(_M, 2) + (1- pow(_M, 2))*(_k44             [i]/_k[0])))/(_M-1));45         posicaoc.push_back(A);46     }4748     return posicaoc;4950 }51525354 std::vector<double> CPosicaoAguaInjetadaDykstra::EfVert(std::vector55     <double> _posicaoc, std::vector<double> _espessurac, double56     _largura, double _espessuraTotal)57 {58     double A;59     for(int i = 0; i < _espessurac.size(); i++)60     {61         A = (_posicaoc[i]*_espessurac[i])/(_largura*_espessuraTotal62             );63         efVert.push_back(A);64     }6566     return efVert;6768 }6970 double CPosicaoAguaInjetadaDykstra::vazaoMedia(std::vector<double>71     _vazaoInj, std::vector<double> _vazaoBt)72 {
```

```
73     double A = 0;
74     for(int i = 0; i < _vazaoInj.size(); i++)
75     {
76
77         A += (_vazaoInj[i] + _vazaoBt[i]);
78     }
79
80     vazaoMed = A/2.0;
81     return vazaoMed;
82
83 }
84
85 void CPosicaoAguaInjetadaDykstra::EfVertTotal()
86 {
87     efVertTotal = 0;
88     for(double eff: efVert)
89     {
90         efVertTotal += eff;
91     }
92 }
93
94 void CPosicaoAguaInjetadaDykstra::VazaoTotal()
95 {
96     vazaoTotal = 0;
97     for(double vazao: VazaoInj)
98     {
99         vazaoTotal += vazao;
100    }
101 }
102
103 double CPosicaoAguaInjetadaDykstra::GetVazaoTotal()
104 {
105
106     return vazaoTotal;
107
108 }
109
110
111 double CPosicaoAguaInjetadaDykstra::GetEfVertTotal()
112 {
113
114     return efVertTotal;
```

```

115
116 }
117
118 std::vector<double> CPosicaoAguaInjetadaDykstra::GetPosicao(){
119     return posicaoc;
120 }
```

Apresenta-se na listagem ?? o arquivo de implementação da classe CPosicaoAguaInjetadaStiles.

Listing 6.21: Arquivo de implementação da classe CPosicaoAguaInjetadaStiles.

```

1 #ifndef CPOSICAOAGUAINJETADASTILES_H_
2 #define CPOSICAOAGUAINJETADASTILES_H_
3
4 #include <vector>
5
6 #include "CPosicaoAguaInjetada.h"
7
8 class CPosicaoAguaInjetadaStiles : CPosicaoAguaInjetada{
9
10    public:
11
12        CPosicaoAguaInjetadaStiles(){};
13
14        std::vector <double> VazaoInjecao(double _krw,
15                                         double _mw, double _largura, double _dp, std::vector <double> _k, double _bw, double _comprimento, std::vector <double> _espessurac);
16        std::vector <double> PosicaoAguaInjetada(std::vector <double> _k, double _largura);
17        std::vector <double> EfVert(std::vector<double> _posicaoc,
18                                   std::vector<double> _espessurac, double _largura, double _espessuraTotal);
19        void EfVertTotal();
20        double GetEfVertTotal();
21        std::vector<double> GetPosicao();
22        void VazaoTotal();
23        double GetVazaoTotal();
24
25    };
26
```

27 #endif

Apresenta-se na listagem 6.22 o arquivo de implementação da classe CPosicaoAguaInjetadaStiles.

Listing 6.22: Arquivo de implementação da classe CPosicaoAguaInjetadaStiles.

```

1 #include "CPosicaoAguaInjetadaStiles.h"
2
3 #include <iostream>
4
5 using namespace std;
6
7 std::vector <double> CPosicaoAguaInjetadaStiles::VazaoInjecao(
8     double _krw, double _mw, double _largura, double _dp, std::vector <
9         double> _k, double _bw, double _comprimento, std::vector <double> _espessurac)
10 {
11
12     double A;
13
14     for (int i = 0; i < _k.size(); i++)
15     {
16         A = (_krw*_comprimento*_dp*86400.00*_k[i]*
17               _espessurac[i])/(_bw*_mw*_largura);
18         VazaoInj.push_back(A);
19     }
20
21     return VazaoInj;
22 }
23
24 std::vector <double> CPosicaoAguaInjetadaStiles::
25 PosicaoAguaInjetada(std::vector <double> _k, double _largura)
26 {
27     double A;
28     for (int i = 0; i < _k.size(); i++)
29     {
30         A = (_largura*_k[i]/(_k[0]));
31         posicaoc.push_back(A);
32     }
33
34     return posicaoc;

```

```
33 }
34
35 std::vector<double> CPosicaoAguaInjetadaStiles::EfVert(std::vector<
36     double> _posicaoc, std::vector<double> _espessurac, double
37     _largura, double _espessuraTotal)
38 {
39     double A;
40     for(int i = 0; i < _espessurac.size(); i++)
41     {
42         A = (_posicaoc[i]*_espessurac[i])/(_largura*_espessuraTotal
43             );
44         efVert.push_back(A);
45     }
46
47     return efVert;
48 }
49 }

50
51 void CPosicaoAguaInjetadaStiles::EfVertTotal()
52 {
53     efVertTotal = 0;
54     for(double eff: efVert)
55     {
56         efVertTotal += eff;
57     }
58 }

59
60 void CPosicaoAguaInjetadaStiles::VazaoTotal()
61 {
62     vazaoTotal = 0;
63     for(double vazao: VazaoInj)
64     {
65         vazaoTotal += vazao;
66     }
67 }

68
69 double CPosicaoAguaInjetadaStiles::GetVazaoTotal()
70 {
71 }
```

```

72     return vazaoTotal;
73
74 }
75
76 double CPosicaoAguaInjetadaStiles::GetEfVertTotal()
77 {
78
79     return efVertTotal;
80
81 }
82
83 std::vector<double> CPosicaoAguaInjetadaStiles::GetPosicao(){
84     return posicaoc;
85 }
```

Apresenta-se na listagem ?? o arquivo de implementação da classe CReservatorio.

Listing 6.23: Arquivo de implementação da classe CReservatorio.

```

1 #ifndef CReservatorio_H_
2 #define CReservatorio_H_
3
4 class CReservatorio {
5
6     protected:
7
8         double kro, krw, sor, phi, swi, M;
9
10    public:
11
12        CReservatorio(){};
13
14        CReservatorio(double _kro, double _krw, double _sor, double
15                      _phi, double _swi, double _M): kro(_kro), krw(_krw),
16                      sor(_sor), phi(_phi), swi(_swi), M(_M){};
17
18        void SetKro(double _kro);
19        void SetKrw(double _krw);
20        void SetSor(double _sor);
21        void SetPHI(double _phi);
22        void SetSwi(double _swi);
```

```

23             void SetM(double _M);
24
25             // metodos para adquirir os valores das variaveis
26             // da classe
27
28             double GetKro();
29             double GetKrw();
30             double GetSor();
31             double GetPHI();
32             double GetSwi();
33             double GetM();
34
35             ~CReservatorio() {};
36 };
37
38 #endif

```

Apresenta-se na listagem 6.24 o arquivo de implementação da classe CReservatorio.

Listing 6.24: Arquivo de implementação da classe CReservatorio.

```

1 #include "CReservatorio.h"
2
3 void CReservatorio::SetKro(double _kro)
4 {
5
6     this->kro = _kro;
7
8 }
9
10 void CReservatorio::SetKrw(double _krw)
11 {
12
13     krw = _krw;
14
15 }
16
17 void CReservatorio::SetSor(double _sor)
18 {
19
20     sor = _sor;
21
22 }

```

```
23
24 void CReservatorio::SetPHI(double _phi)
25 {
26
27     phi = _phi;
28
29 }
30
31 void CReservatorio::SetSwi(double _swi)
32 {
33
34     swi = _swi;
35
36 }
37
38 void CReservatorio::SetM(double _M)
39 {
40
41     M = _M;
42
43 }
44
45 double CReservatorio::GetKro()
46 {
47
48     return kro;
49
50 }
51
52 double CReservatorio::GetKrw()
53 {
54
55     return krw;
56
57 }
58
59 double CReservatorio::GetSor()
60 {
61
62     return sor;
63
64 }
```

```
65
66 double CReservatorio::GetPHI()
67 {
68
69     return phi;
70
71 }
72
73 double CReservatorio::GetSwi()
74 {
75
76     return swi;
77
78 }
79
80 double CReservatorio::GetM()
81 {
82
83     return M;
84
85 }
```

Apresenta-se na listagem ?? o arquivo de implementação da classe CReservatorioCamadas.

Listing 6.25: Arquivo de implementação da classe CReservatorioCamadas.

```
1 #ifndef CRESERVATORIOCAMADAS_H_
2 #define CRESERVATORIOCAMADAS_H_
3
4 #include <vector>
5
6 #include "CReservatorio.h"
7
8 class CReservatorioCamadas : CReservatorio
9 {
10
11     protected:
12
13         double bo, bw, dp, mo, mw, sg, M, largura,
14             comprimento, espessuraTotal;
15
16         std::vector <double> espessurac, k;
```

```
17
18
19     public:
20
21         CReservatorioCamadas(){};
22
23         CReservatorioCamadas(double _bo, double _bw, double
24             _dp, double _mo, double _mw, double _sg, double
25             _largura, double _comprimento, CReservatorio
26             reservatorio):bo(_bo), bw(_bw), dp(_dp), mo(_mo)
27             , mw(_mw), sg(_sg), largura(_largura),
28             comprimento(_comprimento){ M = (reservatorio.
29             GetKrw()/mw)/(reservatorio.GetKro()/mo);};
30
31         //entrada de dados
32
33         void SetBo(double _bo);
34         void SetBw(double _bw);
35         void SetDp(double _dp);
36         void SetMo(double _mo);
37         void SetMw(double _mw);
38         void SetSg(double _sg);
39         void SetLargura(double _largura);
40         void SetComprimento(double _comprimento);
41         void SetEspec(std::vector<double> _espessurac);
42         void SetK(std::vector<double> _k);
43
44         //metodo
45
46         void CalcEspessT();
47         void CalcM();
48
49         //Saida de dados
50
51         double GetBo();
52         double GetBw();
```

```
53         double GetComprimento();
54     double GetEspessuraTotal();
55             std::vector <double> GetEspec();
56             std::vector <double> GetK();
57
58     ~CReservatorioCamadas() {};
59
60 };
61
62 #endif
```

Apresenta-se na listagem 6.26 o arquivo de implementação da classe CReservatorioCamadas.

Listing 6.26: Arquivo de implementação da classe CReservatorioCamadas.

```
1 #include "CReservatorioCamadas.h"
2
3 using namespace std;
4
5 void CReservatorioCamadas::SetBo(double _bo)
6 {
7
8     bo = _bo;
9
10 }
11
12 void CReservatorioCamadas::SetBw(double _bw)
13 {
14
15     bw = _bw;
16
17 }
18
19 void CReservatorioCamadas::SetDp(double _dp)
20 {
21
22     dp = _dp;
23
24 }
25
26 void CReservatorioCamadas::SetMo(double _mo)
27 {
28
```

```
29         mo = _mo;
30
31 }
32
33 void CReservatorioCamadas::SetMw(double _mw)
34 {
35
36     mw = _mw;
37
38 }
39
40 void CReservatorioCamadas::SetSg(double _sg)
41 {
42
43     sg = _sg;
44
45 }
46
47 void CReservatorioCamadas::SetLargura(double _largura)
48 {
49
50     largura = _largura;
51
52 }
53
54 void CReservatorioCamadas::SetComprimento(double _comprimento)
55 {
56
57     comprimento = _comprimento;
58
59 }
60
61 void CReservatorioCamadas::SetEspec(std::vector<double> _espessurac
62 )
63
64     espessurac = _espessurac;
65
66 }
67
68 void CReservatorioCamadas::SetK(std::vector<double> _k)
69 {
```

```
70
71         k = _k;
72
73 }
74
75 void CReservatorioCamadas::CalcEspessT()
76 {
77     espessuraTotal = 0;
78     for(double espessura:espessurac)
79     {
80         espessuraTotal += espessura;
81     }
82 }
83
84 void CReservatorioCamadas::CalcM(){
85
86     M = (CReservatorio::GetKrw()/mw)/(CReservatorio::GetKro()/
87     mo);
88 }
89
90 double CReservatorioCamadas::GetBo()
91 {
92
93     return bo;
94
95 }
96
97 double CReservatorioCamadas::GetBw()
98 {
99
100    return bw;
101
102 }
103
104 double CReservatorioCamadas::GetDp()
105 {
106
107    return dp;
108
109 }
110
```

```
111 double CReservatorioCamadas::GetMo()
112 {
113
114     return mo;
115
116 }
117
118 double CReservatorioCamadas::GetMw()
119 {
120
121     return mw;
122
123 }
124
125 double CReservatorioCamadas::GetSg()
126 {
127
128     return sg;
129
130 }
131
132 double CReservatorioCamadas::GetM()
133 {
134
135     return M;
136
137 }
138
139 double CReservatorioCamadas::GetLargura()
140 {
141
142     return largura;
143
144 }
145
146 double CReservatorioCamadas::GetComprimento()
147 {
148
149     return comprimento;
150
151 }
152
```

```

153 vector <double> CReservatorioCamadas::GetEspec()
154 {
155
156     return espessurac;
157
158 }
159
160 vector <double> CReservatorioCamadas::GetK()
161 {
162
163     return k;
164
165 }
166
167 double CReservatorioCamadas::GetEspessuraTotal()
168 {
169
170     return espessuraTotal;
171
172 }
```

Apresenta-se na listagem ?? o arquivo de implementação da classe CSolverInfluxo.

Listing 6.27: Arquivo de implementação da classe CSolverInfluxo.

```

1 #ifndef CSOLVERINFLUXO_H
2 #define CSOLVERINFLUXO_H
3
4 #include <vector>
5 #include <cmath>
6
7 #include "CReservatorioCamadas.h"
8 #include "CReservatorio.h"
9 #include "CGnuplot.h"
10 #include "CPosicaoAguaInjetadaStiles.h"
11 #include "CPosicaoAguaInjetadaDykstra.h"
12 #include "COleoProduzido.h"
13 #include "CLinhasequipotenciais.h"
14 #include "CPoco.h"
15 #include "CLinhaPressao2Pocos.h"
16 #include "CLinhaPressao3Pocos1P2I.h"
17 #include "CLinhaPressao3Pocos2P1I.h"
18 #include "CCorey.h"
19 #include "CFluxoFracionario.h"
```

```
20
21 class CSolverInfluxo {
22
23 protected:
24
25     CReservatorioCamadas camadas;
26     CReservatorio reservatorio;
27     CPoco poco;
28     CPosicaoAguaInjetadaStiles stiles;
29     CPosicaoAguaInjetadaDykstra dykstra;
30     COleoProduzido oleo;
31     CLinhasequipotenciais* linhas;
32     CGnuplot plot, plot1, plot2, plot3, plot4, plot5;
33     std::vector<double> xplot, yplot;
34     CCorey corey;
35     CFluxoFracionario fluxofracionario;
36
37 private:
38
39     void EntradaDadosReservatorio();
40     void EntradaDadosCamadas();
41     void EntradaDadosRochaReservatorio();
42     void EntradaDadosPoco();
43     void Plot();
44     double kbarra(std::vector<double> k, std::vector<double> h);
45     double r1(double _c, double x, double y);
46         double r2(double _c, double x, double y);
47         double r3(double x, double y);
48         void printResults(double _x, double _y, double _c, double
49                         _q, double _mi, double _kbarra, double _h, double _pi);
50
51 public:
52
53     CSolverInfluxo() {};
54
55     void Simular();
56
57     ~CSolverInfluxo() {};
58 };
59
60 #endif
```

Apresenta-se na listagem 6.28 o arquivo de implementação da classe CSolverInfluxo.

Listing 6.28: Arquivo de implementação da classe CSolverInfluxo.

```
1 #include <string>
2 #include <filesystem>
3 #include <iostream>
4 #include <fstream>
5 #include <vector>
6
7 #include "CSolverInfluxo.h"
8
9 using namespace std;
10
11 void CSolverInfluxo::Simular()
12 {
13
14     bool rodar = true;
15     bool teste = false;
16
17     while (!teste)
18     {
19
20         do
21         {
22             cout << endl;
23             cout << "Deseja executar o código? (1 - sim, 2 - não): ";
24             char c;
25             cin >> c;
26             cin.get();
27
28         if (c == '1' or c == 's')
29         {
30
31             cout << endl;
32
33             cout << "#"
34             ##### "#"
35             cout << "#Projeto de programação prática"
36             cout << "#Professor: André Duarte Bueno"
```

```
    #####\n"
42 cout << endl << endl;\n"
43\n"
44 cout << "Aperte ENTER para continuar com carregamento de dados..." \n"
45 cin.get();\n"
46\n"
47 EntradaDadosReservatorio();\n"
48\n"
49 cout << "Aperte ENTER para continuar com carregamento de dados..." \n"
50 cin.get();\n"
51\n"
52 EntradaDadosRochaReservatorio();\n"
53\n"
54 cout << "Aperte ENTER para continuar com carregamento de dados..." \n"
55 cin.get();\n"
56\n"
57 EntradaDadosCamadas();\n"
58 cout << "#####\n"
59 ;\n"
60 cout << "# Todos arquivos foram carregados com sucesso!!\n";
61 cout << "#####\n"
62 cout << "# qual metodo deseja utilizar?" << endl << endl;\n"
63 cout << "# 1 - Stiles" << endl;\n"
64 cout << "# 2 - Dijkstra" << endl << endl;\n"
65 cout << "#\n"
66
```

```
67 char handler;
68
69 cin >> handler;
70 cin.get();
71
72 camadas.CalcM();
73 camadas.CalcEspessT();
74
75 bool tes = false;
76
77 do
78 {
79
80 switch (handler){
81
82 case '1':
83 {
84
85 vector<double> vazao;
86 vazao = stiles.VazaoInjecao(reservatorio.GetKrw(), camadas.GetMw(),
camadas.GetLargura(), camadas.GetDp(), camadas.GetK(), camadas.
GetBw(), camadas.GetComprimento(), camadas.GetEspec());
87
88 vector<double> perm = camadas.GetK();
89
90 for (int i = 0 ; i< vazao.size() ; i++)
91 {
92 cout <<"Vazao\u2022Stiles\u2022" << vazao[i] << "\tPermeabilidade\u2022" <<
perm[i]<< endl;
93 }
94 cout << endl;
95
96 stiles.VazaoTotal();
97 double vazaototal;
98 vazaototal = stiles.GetVazaoTotal();
99
100 cout << "Vazao\u2022total\u2022" << vazaototal << endl;
101
102 cout << endl;
103
104 vector<double> posicao;
105 posicao = stiles.PosicaoAguaInjetada(camadas.GetK(), camadas.
```

```
GetLargura());  
106  
107 for (double item: posicao)  
108 cout <<"Posicao_frente_de_avanco_de_agua=" << item << endl;  
109  
110 cout << endl;  
111  
112 double eff;  
113 vector<double> effVet;  
114  
115 effVet = stiles.EfVert(stiles.GetPosicao(), camadas.GetEspec(),  
    camadas.GetLargura(), camadas.GetEspessuraTotal());  
116 stiles.EfVertTotal();  
117 eff = stiles.GetEfVertTotal();  
118  
119 for (double item: effVet)  
120 cout <<"Eficiencia_vertical=" << item << endl;  
121  
122 cout << endl;  
123 cout << "Eficiencia_vertical_total=" << eff << endl;  
124  
125 double oleoProduzidoTotal;  
126 oleo.OleoProduzido(camadas.GetLargura(), camadas.GetComprimento(),  
    reservatorio.GetPHI(), camadas.GetEspessuraTotal(), reservatorio  
    .GetSor(), reservatorio.GetSwi(), stiles.GetEfVertTotal(),  
    camadas.GetBo());  
127 oleoProduzidoTotal = oleo.getNp();  
128  
129 cout << endl;  
130 cout << "Np=" << oleoProduzidoTotal << endl;  
131  
132 double Bt;  
133 oleo.TempoBreakThrough(stiles.GetVazaoTotal());  
134 Bt = oleo.getTbreak();  
135  
136 cout << endl;  
137 cout << "O_tempo_de_break_through_foi:" << Bt << endl;  
138  
139 bool ecpot = false;  
140  
141 do  
142 {
```

```
143
144 cout << "\nGostaria de exibir os graficos de linhas equipotenciais?
145   (1 - sim, 2 - nao)" << endl;
146
147 char d;
148 cin >> d;
149 cin.get();
150 if(d == '1')
151 {
152 cout << "Aperte ENTER para continuar com carregamento de dados..." 
153   << endl;
154 cin.get();
155 EntradaDadosPoco();
156
157 double kb = kbarra(perm, camadas.GetEspec());
158
159 cout << "# Qual configuracao de pozos deseja calcular?" << endl;
160 cout << "# 1 - 1 poco produtor e 1 injetor" << endl;
161 cout << "# 2 - 2 pozos injetores e 1 produtor" << endl;
162 cout << "# 3 - 1 poco injetor e 2 produtores" << endl << endl;
163
164 char ans;
165 bool ans1 = false;
166 cin >> ans;
167 cin.get();
168
169 do{
170 switch(ans){
171 case '1':
172 {
173 ans1 = true;
174 double _r1 = r1(poco.getC(), poco.getX(), poco.getY());
175 double _r2 = r2(poco.getC(), poco.getX(), poco.getY());
176 linhas = new CLinhaPressao2Pocos(_r1, _r2, 0.0);
177 printResults(poco.getX(), poco.getY(), poco.getC(), stiles.
178   GetVazaoTotal(), camadas.GetMo(), kb, camadas.GetEspessuraTotal
179   (), poco.getPi());
180 cout << "R=" << linhas->R(poco.getC()) << "C=" << poco.getC()
181   << endl;
182 xplot = linhas->CalculoDoVetorX(linhas->R(poco.getC()), poco.getC())
183
```

```
    );  
180 yplot = linhas->CalculoDoVetorY(linhas->R(poco.getC()), poco.getC()  
    );  
  
181  
182 Gnuplot::Terminal("qt");  
183 plot.set_style("lines");  
184 plot.Title("Curvas de equipotenciais.");  
185 plot.unset_legend();  
186 plot.set_xlabel("x");  
187 plot.set_ylabel("y");  
188 plot.ShowOnScreen();  
189 plot.Grid();  
190 plot.PlotVector(xplot, yplot);  
191 cin.get();  
  
192  
193 plot.savetops("dois_pocos_stiles");  
194  
195  
196  
197 break;  
198 }  
199 case '2':  
200 {  
201  
202 ans1 = true;  
203 double _r1 = r1(poco.getC(), poco.getX(), poco.getY());  
204 double _r2 = r2(poco.getC(), poco.getX(), poco.getY());  
205 double _r3 = r3(poco.getX(), poco.getY());  
206 linhas = new CLinhaPressao3Pocos1P2I(_r1, _r2, _r3);  
207 printResults(poco.getX(), poco.getY(), poco.getC(), stiles.  
    GetVazaoTotal(), camadas.GetMo(), kb, camadas.GetEspessuraTotal  
    (), poco.getPi());  
208 cout << "R=" << linhas->R(poco.getC()) << "C=" << poco.getC()  
    << endl;  
209 xplot = linhas->CalculoDoVetorX(linhas->R(poco.getC()), poco.getC()  
    );  
210 yplot = linhas->CalculoDoVetorY(linhas->R(poco.getC()), poco.getC()  
    );  
  
211  
212 Gnuplot::Terminal("qt");  
213 plot1.set_style("lines");  
214 plot1.Title("Curvas de equipotenciais.");
```

```
215 plot1.unset_legend();
216 plot1.set_xlabel("u");
217 plot1.set_ylabel("u");
218 plot1.ShowOnScreen();
219 plot1.Grid();
220 plot1.PlotVector(xplot, yplot);
221 cin.get();
222
223 plot1.savetops("dois-pocos-injetores-um-poco-produtor-stiles");
224
225
226 break;
227 }
228 case '3':
229 {
230
231 ans1 = true;
232 double _r1 = r1(poco.getC(), poco.getX(), poco.getY());
233 double _r2 = r2(poco.getC(), poco.getX(), poco.getY());
234 double _r3 = r3(poco.getX(), poco.getY());
235 linhas = new CLinhaPressao3Pocos2P1I(_r1, _r2, _r3);
236 printResults(poco.getX(), poco.getY(), poco.getC(), stiles.
    GetVazaoTotal(), camadas.GetMo(), kb, camadas.GetEspessuraTotal
    (), poco.getPi());
237 cout << "R=" << linhas->R(poco.getC()) << "C=" << poco.getC()
    << endl;
238 xplot = linhas->CalculoDoVetorX(linhas->R(poco.getC()), poco.getC()
    );
239 yplot = linhas->CalculoDoVetorY(linhas->R(poco.getC()), poco.getC()
    );
240
241 Gnuplot::Terminal("qt");
242 plot2.set_style("lines");
243 plot2.Title("Curvas equipotenciais.");
244 plot2.unset_legend();
245 plot2.set_xlabel("u");
246 plot2.set_ylabel("u");
247 plot2.ShowOnScreen();
248 plot2.Grid();
249 plot2.PlotVector(xplot, yplot);
250 cin.get();
251
```

```
252 plot2.savetops("um-poco-injetor-dois-pocos-produtores-stiles");  
253  
254  
255 break;  
256 }  
257 default:  
258 cout << "Opção invalidade!!!" << endl << endl;  
259 cout << "#Qual configuração de poços deseja calcular?" << endl;  
260 cout << "#1-1poco produtor e 1injetor" << endl;  
261 cout << "#2-2pocos injetores e 1produtor" << endl;  
262 cout << "#3-1poco injetor e 2produtores" << endl << endl;  
263 cin >> ans;  
264 cin.get();  
265 }  
266 }while(!ans1);  
267  
268 ecpot = true;  
269 }else if( d == '2'){  
270 ecpot = true;  
271 }else {  
272 cout << "opção invalida!!!" << endl;  
273 }  
274  
275 }while(!ecpot);  
276 tes = true;  
277 break;  
278 }  
279 case '2':  
280 {  
281  
282 vector<double> vazao;  
283 vazao = dykstra.VazaoInjecao(reservatorio.GetKrw(), camadas.GetMw()  
, camadas.GetLargura(), camadas.GetDp(), camadas.GetK(), camadas  
.GetBw(), camadas.GetComprimento(), camadas.GetEspec(),  
reservatorio.GetM());  
284  
285 vector<double> perm = camadas.GetK();  
286  
287 for (int i = 0 ; i< vazao.size() ; i++)  
288 {  
289 cout <<"Vazao Dykstra = " << vazao[i] << "\tPermeabilidade = "  
perm[i]<<endl;
```

```
290 }
291 cout << endl;
292
293 dykstra.Vazaototal();
294 double vazaototal;
295 vazaototal = dykstra.GetVazaototal();
296
297 cout << "Vazaototal=" << vazaototal << endl;
298
299 cout << endl;
300
301 vector<double> posicao;
302 posicao = dykstra.PosicaoAguaInjetada(camadas.GetK(), camadas.
    GetLargura(), reservatorio.GetM());
303
304 for (double item: posicao)
305 cout <<"Posicao_frente_de_avanco_de_agua=" << item << endl;
306
307 cout << endl;
308
309 double eff;
310 vector<double> effVet;
311
312 effVet = dykstra.EfVert(dykstra.GetPosicao(), camadas.GetEspec(),
    camadas.GetLargura(), camadas.GetEspessuraTotal());
313 dykstra.EfVertTotal();
314 eff = dykstra.GetEfVertTotal();
315
316 for (double item: effVet)
317 cout <<"Eficiencia_vertical=" << item << endl;
318
319 cout << endl;
320 cout << "Eficiencia_vertical_total=" << eff << endl;
321
322 double oleoProduzidoTotal;
323 oleo.OleoProduzido(camadas.GetLargura(), camadas.GetComprimento(),
    reservatorio.GetPHI(), camadas.GetEspessuraTotal(), reservatorio
    .GetSor(), reservatorio.GetSwi(), dykstra.GetEfVertTotal(),
    camadas.GetBo());
324 oleoProduzidoTotal = oleo.getNp();
325
326 cout << endl;
```

```
327 cout << "Nº de poços" << oleoProduzidoTotal << endl;
328
329 double Bt;
330 oleo.TempoBreakThrough(dykstra.GetVazaoTotal());
331 Bt = oleo.getTbreak();
332
333 cout << endl;
334 cout << "O tempo de break-through foi:" << Bt << endl;
335
336 bool ecpot = false;
337
338 do
339 {
340
341 cout << "\nGostaria de exibir os graficos de linhas equipotenciais?
342     (1 - sim, 2 - não)" << endl;
343
344 char d;
345 cin >> d;
346 cin.get();
347 if(d == '1' or d == 's')
348 {
349 cout << "Aperte ENTER para continuar com carregamento de dados..." 
350     << endl;
351 cin.get();
352 EntradaDadosPoco();
353
354 double kb = kbarra(perm, camadas.GetEspec());
355
356 cout << "# Qual configuração de poços deseja calcular?" << endl;
357 cout << "# 1 - 1 poço produtor e 1 injetor" << endl;
358 cout << "# 2 - 2 poços injetores e 1 produtor" << endl;
359 cout << "# 3 - 1 poço injetor e 2 produtores" << endl << endl;
360
361 char ans;
362 bool ans1 = false;
363 cin >> ans;
364 cin.get();
365
366 do {
```

```
367 switch(ans){  
368 case '1':  
369 {  
370 ans1 = true;  
371 double _r1 = r1(poco.getC(), poco.getX(), poco.getY());  
372 double _r2 = r2(poco.getC(), poco.getX(), poco.getY());  
373 linhas = new CLinhaPressao2Pocos(_r1, _r2, 0.0);  
374 printResults(poco.getX(), poco.getY(), poco.getC(), dykstra.  
    GetVazaoTotal(), camadas.GetMo(), kb, camadas.GetEspessuraTotal  
    (), poco.getPi());  
375 cout << "R=" << linhas->R(poco.getC()) << "C=" << poco.getC()  
    << endl;  
376 xplot = linhas->CalculoDoVetorX(linhas->R(poco.getC()), poco.getC()  
    );  
377 yplot = linhas->CalculoDoVetorY(linhas->R(poco.getC()), poco.getC()  
    );  
378  
379 Gnuplot::Terminal("qt");  
380 plot.set_style("lines");  
381 plot.Title("Curvas de equipotenciais.");  
382 plot.unset_legend();  
383 plot.set_xlabel("x");  
384 plot.set_ylabel("y");  
385 plot.ShowOnScreen();  
386 plot.Grid();  
387 plot.PlotVector(xplot, yplot);  
388 cin.get();  
389  
390 plot.savetops("dois-pocos-dykstra");  
391  
392 break;  
393 }  
394 case '2':  
395 {  
396  
397 ans1 = true;  
398 double _r1 = r1(poco.getC(), poco.getX(), poco.getY());  
399 double _r2 = r2(poco.getC(), poco.getX(), poco.getY());  
400 double _r3 = r3(poco.getX(), poco.getY());  
401 linhas = new CLinhaPressao3Pocos1P2I(_r1, _r2, _r3);  
402 printResults(poco.getX(), poco.getY(), poco.getC(), dykstra.  
    GetVazaoTotal(), camadas.GetMo(), kb, camadas.GetEspessuraTotal
```

```

        () , poco.getPi());
403 cout << "R=" << linhas->R(poco.getC()) << "C=" << poco.getC()
     << endl;
404 xplot = linhas->CalculoDoVetorX(linhas->R(poco.getC()), poco.getC()
     );
405 yplot = linhas->CalculoDoVetorY(linhas->R(poco.getC()), poco.getC()
     );
406
407 Gnuplot::Terminal("qt");
408 plot1.set_style("lines");
409 plot1.Title("Curvas equipotenciais.");
410 plot1.unset_legend();
411 plot1.set_xlabel("x");
412 plot1.set_ylabel("y");
413 plot1.ShowOnScreen();
414 plot1.Grid();
415 plot1.PlotVector(xplot, yplot);
416 cin.get();
417
418 plot1.savetops("dois-pocos-injetores-um-poco-produtor-dykstra");
419
420 break;
421 }
422 case '3':
423 {
424
425 ans1 = true;
426 double _r1 = r1(poco.getC(), poco.getX(), poco.getY());
427 double _r2 = r2(poco.getC(), poco.getX(), poco.getY());
428 double _r3 = r3(poco.getX(), poco.getY());
429 linhas = new CLinhaPressao3Pocos2P1I(_r1, _r2, _r3);
430 printResults(poco.getX(), poco.getY(), poco.getC(), dykstra.
     GetVazaoTotal(), camadas.GetMo(), kb, camadas.GetEspessuraTotal
     (), poco.getPi());
431 cout << "R=" << linhas->R(poco.getC()) << "C=" << poco.getC()
     << endl;
432 xplot = linhas->CalculoDoVetorX(linhas->R(poco.getC()), poco.getC()
     );
433 yplot = linhas->CalculoDoVetorY(linhas->R(poco.getC()), poco.getC()
     );
434
435 Gnuplot::Terminal("qt");

```

```
436 plot2.set_style("lines");
437 plot2.Title("Curvas de equipotenciais.");
438 plot2.unset_legend();
439 plot2.set_xlabel("u");
440 plot2.set_ylabel("u");
441 plot2.ShowOnScreen();
442 plot2.Grid();
443 plot2.PlotVector(xplot, yplot);
444 cin.get();
445
446 plot2.savetops("um-poco-injetor-dois-pocos-produtores-dykstra");
447
448 break;
449 }
450 default:
451 cout << "Opção inválida!!!" << endl << endl;
452 cout << "# Qual configuração de poços deseja calcular?" << endl;
453 cout << "# 1 - 1 poço produtor e 1 injetor" << endl;
454 cout << "# 2 - 2 poços injetores e 1 produtor" << endl;
455 cout << "# 3 - 1 poço injetor e 2 produtores" << endl << endl;
456 cin >> ans;
457 cin.get();
458 }
459 }while(!ans1);
460
461 ecpot = true;
462 }else if( d == '2'){
463 ecpot = true;
464 }else {
465 cout << "opção inválida!!!" << endl;
466 }
467
468 }while(!ecpot);
469
470 tes = true;
471 break;
472 }
473 default:
474 {
475 cout << "Opção inválida!!!" << endl << endl;
476 cout << "# qual método deseja utilizar?" << endl << endl;
477 cout << "# 1 - Stiles" << endl;
```

```
478 cout << "#_2_-_Dikstra" << endl << endl;
479
480 cin >> handler;
481 cin.get();
482
483 }
484 }
485 }while(!tes);
486
487 corey.setK0rw(reservatorio.GetKrw());
488 corey.setK0ro(reservatorio.GetKro());
489 corey.setNo(2.0);
490 corey.setNw(2.0);
491
492 corey.calcSwn(reservatorio.GetSwi(), reservatorio.GetSor());
493 corey.calcKro(corey.getSw());
494 corey.calcKrw(corey.getSw());
495
496 cout << "#_Gostaria_de_plotar_os_graficos_de_permeabilidade_
        relativa_de_corey-brooks?" << endl;
497 cout << "#_1_-_sim,_2_-_nao_" << endl << endl;
498
499 char e;
500
501 cin >> e;
502 cin.get();
503
504 bool tes1 = false;
505
506 do
507 {
508
509 if (e == '1' or e == 's'){
510
511 vector<double> _sw, _krw, _kro;
512
513 _sw = corey.getSw();
514 _krw = corey.getKrw();
515 _kro = corey.getKro();
516
517 Gnuplot::Terminal("qt");
518 plot3.set_style("lines");
```

```
519 plot3.Title("Modelo de permeabilidade relativa de Corey-Brooks.");
520 plot3.set_legend("inside_center_top_box");
521 plot3.set_xlabel("Sw (saturação normalizada)");
522 plot3.set_ylabel("Kr");
523 plot3.ShowOnScreen();
524 plot3.Grid();
525 plot3.PlotVector(_sw, _krw, "Krw");
526 plot3.set_style("points");
527 plot3.PlotVector(_sw, _kro, "Kro");
528
529 cin.get();
530
531 plot3.savetops("Permeabilidade-relativa-Corey-Brooks");
532
533 tes1 = true;
534 }else
535 if(e == '2'){
536 tes1 = true;
537 } else{
538 cout << "Opção invalida!!!" << endl << endl;
539 cout << "#Gostaria de plotar os graficos de permeabilidade
      relativa de corey-brooks?" << endl;
540 cout << "#1 - sim, 2 - não" << endl << endl;
541
542 cin >> e;
543 cin.get();
544
545 }
546
547 }while(!tes1);
548
549 cout << "#Gostaria de plotar as curvas de fluxo fracionário?" <<
      endl;
550 cout << "#1 - sim, 2 - não" << endl << endl;
551
552 char f;
553
554 cin >> f;
555 cin.get();
556
557 bool tes2 = false;
558
```

```
559 do
560 {
561
562 if (f == '1'){
563
564 fluxofracionario.calcFluxoFracionarioAgua(corey.getKrw(), corey.
      getKro(), camadas.GetMw(), camadas.GetMo());
565 fluxofracionario.calcFluxoFracionarioOleo(corey.getKrw(), corey.
      getKro(), camadas.GetMw(), camadas.GetMo());
566
567 vector<double> _fro, _frw, _sw;
568
569 _sw = corey.getSw();
570 _fro = fluxofracionario.getFluxoFracionarioOleo();
571 _frw = fluxofracionario.getFluxoFracionarioAgua();
572
573 Gnuplot::Terminal("qt");
574 plot4.set_style("lines");
575 plot4.Title("Fluxo_fracionario_de_agua.");
576 plot4.set_legend("inside_center_top_box");
577 plot4.set_xlabel("Swn(saturação_normalizada)");
578 plot4.set_ylabel("Frw");
579 plot4.ShowOnScreen();
580 plot4.Grid();
581 plot4.PlotVector(_sw, _frw, "Krw");
582 cin.get();
583
584 plot4.savetops("Fluxo_fracionario_de_agua");
585
586 Gnuplot::Terminal("qt");
587 plot5.set_style("lines");
588 plot5.Title("Fluxo_fracionario_de_oleo");
589 plot5.set_legend("inside_center_top_box");
590 plot5.set_xlabel("Swn(saturação_normalizada)");
591 plot5.set_ylabel("Fro");
592 plot5.ShowOnScreen();
593 plot5.Grid();
594 plot5.PlotVector(_sw, _fro, "Kro");
595 cin.get();
596
597 plot5.savetops("Fluxo-fracionario-de-oleo");
598
```

```
599
600 tes2 = true;
601 }else
602 if(f == '2'){
603 tes2 = true;
604 } else{
605 cout << "Opcao invalida!!!" << endl << endl;
606 cout << "# Gostaria de plotar os graficos de permeabilidade
607      relativa de corey-brooks?" << endl;
608 cout << "# 1 - sim, 2 - nao" << endl << endl;
609
610 cin >> e;
611 cin.get();
612 }
613
614 }while(!tes2);
615
616 } else if(c == '2'){
617 cout << endl;
618 cout << "Codigo encerrado!!" << endl << endl;
619 rodar = false;
620 teste = true;
621 } else {
622 cout << endl;
623 cout << "Opcao invalida!" << endl;
624 }
625 } while (rodar);
626
627 }
628
629
630 }
631
632 void CSolverInfluxo::EntradaDadosRochaReservatorio(){
633
634 cout << endl;
635 cout << "##### Arquivos de dados da rocha #####" << endl
636 cout << "# Arquivos de dados da rocha#" << endl
637 cout << "##### Arquivos de dados da rocha #####" << endl
```

```
<< endl;

638
639 string path = "./in";
640 for (const auto & entry : filesystem::directory_iterator(path))
641 cout << "#\t" << entry.path() << std::endl;
642
643 cout << endl;
644
645 bool sucess = false;
646
647 do
648 {
649
650 cout << "Entre com o nome do arquivo com dados da rocha reservatorio:
651 ";
652 string tmp;
653 getline(cin, tmp);
654 cout << endl;
655 ifstream in;
656
657 in.open(tmp);
658
659 double Kro, Krw, Sor, phi, swi;
660
661 in >> Kro >> Krw >> Sor >> phi >> swi;
662
663 cout << "Kro=" << Kro << "Krw=" << Krw << "Sor=" << Sor <<
664 "phi=" << phi << "swi=" << swi << endl;
665 cout << endl;
666
667 bool test = false;
668
669 while (!test)
670 {
671 char t;
672 cout << "O carregamento foi correto? (1-sim, 2-nao): ";
673 cin >> t;
674 cin.get();
675
676 if (t == '1' or t == 's')
```

```
677 {  
678     test = true;  
679     sucess = true;  
680  
681     reservatorio.SetKro(Kro);  
682     reservatorio.SetKrw(Krw);  
683     reservatorio.SetSor(Sor);  
684     reservatorio.SetPHI(phi);  
685     reservatorio.SetSwi(swi);  
686  
687  
688     cout << endl << endl;  
689  
690     cout << "##### Dados salvos! #####" << endl;  
691     cout << "#uuuuuuDados salvo!uuuuuuuu#" << endl;  
692     cout << "##### Arquivos de dados do reservatorio #####" << endl << endl;  
693  
694  
695 } else  
696 {  
697     if (t == '2' or t == 'n')  
698         test = true;  
699     else  
700         cout << "valor invalido!!" << endl;  
701 }  
702 }  
703  
704     cout << endl;  
705  
706     in.close();  
707  
708 } while (!sucess);  
709 }  
710  
711 void CSolverInfluxo::EntradaDadosReservatorio()  
712 {  
713  
714     cout << endl;  
715     cout << "##### Arquivos de dados do reservatorio #####" << endl  
    ;  
716     cout << "#uuuuuuArquivos de dados do reservatoriouuuuuu#" << endl  
    ;
```

```
717 cout << "#####\n" << endl;
    << endl;

718
719 string path = "./in";
720 for (const auto & entry : filesystem::directory_iterator(path))
721 cout << "#\t" << entry.path() << std::endl;
722
723 cout << endl;
724
725 bool sucess = false;
726
727 do
728 {
729
730 cout << "Entre com o caminho do arquivo com dados do reservatorio:\n";
731 string tmp;
732
733 getline(cin, tmp);
734 cout << endl;
735 ifstream in;
736
737
738 in.open(tmp);
739
740 double L, C, Bo, Bw, Dp, Mo, Mw, Sg, M;
741
742 in >> L >> C >> Bo >> Bw >> Dp >> Mo >> Mw >> Sg >> M ;
743
744 cout << "Largura=\n" << L << "Comprimento=\n" << C << "Bo=\n" <<
    Bo << "Bw=\n" << Bw << "deltaP=\n" << Dp << "Mo=\n" << Mo
    << "Mi_w=\n" << Mw << "Sg=\n" << Sg << "M=\n" << M << endl;
745
746 cout << endl;
747
748 bool test = false;
749
750 while(!test)
751 {
752 char t;
753 cout << "O carregamento foi correto? (1-sim, 2-nao):\n";
754 cin >> t;
```

```
755 cin.get();  
756  
757 if (t == '1')  
758 {  
759 test = true;  
760 sucess = true;  
761  
762 camadas.SetLargura(L);  
763 camadas.SetComprimento(C);  
764 camadas.SetBo(Bo);  
765 camadas.SetBw(Bw);  
766 camadas.SetDp(Dp);  
767 camadas.SetMo(Mo);  
768 camadas.SetMw(Mw);  
769 camadas.SetSg(Sg);  
770 reservatorio.SetM(M);  
771  
772  
773 cout << endl << endl;  
774  
775 cout << "##### Dados salvos! #####" << endl;  
776 cout << "#uuuuuuDados salvo!uuuuuuuu#" << endl;  
777 cout << "##### Dados salvos! #####" << endl << endl;  
778  
779  
780 } else  
781 {  
782 if (t == '2')  
783 test = true;  
784 else  
785 cout << "valor invalido!!" << endl;  
786 }  
787 }  
788  
789 cout << endl;  
790  
791 in.close();  
792  
793 } while (!sucess);  
794  
795 }  
796
```

```
797 void CSolverInfluxo::EntradaDadosCamadas()
798 {
799
800 cout << endl;
801 cout << "#####Arquivos de entradas de dados#####" << endl
802 ;
803 cout << "#uuuuuuuuArquivos de entradas de dados uuuuuuuuuu#" << endl
804 ;
805 cout << "#####Arquivos de entradas de dados#####" << endl
806 << endl;
807
808 string path = "./in";
809 for (const auto & entry : filesystem::directory_iterator(path))
810 cout << "#\t" << entry.path() << std::endl;
811
812 cout << endl;
813
814 bool sucess = false;
815
816 do
817 {
818
819 cout << "Entre com o caminho do arquivo com dados da camada:" ;
820 string tmp;
821
822 getline(cin, tmp);
823 cout << endl;
824
825 ifstream in;
826
827 in.open(tmp);
828
829 double inK, inH;
830 vector<double> K, H;
831
832 while (!in.eof())
833 {
834 char c = in.peek();
835 if (c == '#' || c == '\n') {
836 in.ignore(256, '\n');
837 continue;
838 }
839 }
```

```
836 in >> inK >> inH;
837 cout << "Permeabilidade:" << inK << "Espessura:" << inH << endl;
838 K.push_back(inK);
839 H.push_back(inH);
840 }
841
842 cout << endl;
843
844 bool test = false;
845
846 while (!test)
847 {
848 char t;
849 cout << "O carregamento foi correto? (1-sim, 2-nao): ";
850 cin >> t;
851 cin.get();
852
853 if (t == '1')
854 {
855 test = true;
856 sucess = true;
857
858 camadas.SetK(K);
859 camadas.SetEspec(H);
860
861 cout << endl << endl;
862
863 cout << "#####" << endl;
864 cout << "# Dados salvos!" << endl;
865 cout << "#####" << endl << endl;
866
867
868 } else
869 {
870 if (t == '2')
871 test = true;
872 else
873 cout << "valor invalido!!" << endl;
874 }
875 }
876
877 cout << endl;
```

```
878
879 in.close();
880
881 } while (!sucess);
882
883 }
884
885 void CSolverInfluxo::EntradaDadosPoco()
886 {
887
888 cout << endl;
889 cout << "#####Arquivos de dados do poco#####" << endl
890 ;
891 cout << "#\t" << endl;
892 cout << endl;
893 string path = "./poco";
894 for (const auto & entry : filesystem::directory_iterator(path))
895 cout << "#\t" << entry.path() << std::endl;
896
897 cout << endl;
898
899 bool sucess = false;
900
901 do
902 {
903
904 cout << "Entre com o caminho do arquivo com dados do poco:" ;
905 string tmp;
906
907 getline(cin, tmp);
908 cout << endl;
909 ifstream in;
910
911 in.open(tmp);
912
913 double X, Y, C, Pi;
914
915 in >> X >> Y >> C >> Pi;
916
```

```
917 cout << "X=" << X << "Y=" << Y << "C=" << C << "Pi=" <<
    Pi << endl;

918
919 cout << endl;
920
921 bool test = false;
922
923 while (!test)
924 {
925     char t;
926     cout << "O carregamento foi correto? (1-sim, 2-nao): ";
927     cin >> t;
928     cin.get();
929
930     if (t == '1')
931     {
932         test = true;
933         sucess = true;
934
935         poco.setX(X);
936         poco.setY(Y);
937         poco.setC(C);
938         poco.setPi(Pi);
939
940
941     cout << endl << endl;
942
943     cout << "#####" << endl;
944     cout << "# Dados salvos!" << endl;
945     cout << "#####" << endl << endl;
946
947
948 } else
949 {
950     if (t == '2')
951         test = true;
952     else
953         cout << "valor invalido!!" << endl;
954 }
955 }
956
957 cout << endl;
```

```
958
959     in.close();
960
961 } while (!sucess);
962
963 }
964
965 double CSolverInfluxo::kbarra(vector<double> k, vector<double> h){
966
967     double kh = 0;
968     double sumh = 0;
969
970     for (int i = 0; i < k.size(); i++) {
971         kh += k[i]*h[i];
972         sumh += h[i];
973     }
974
975     return kh/sumh;
976
977 }
978
979 void CSolverInfluxo::printResults(double _x, double _y, double _c,
980                                     double _q, double _mi, double _kbarra, double _h, double _pi) {
980     cout << "ÁREA INVADIDA PELA ÁGUA NO INSTANTE DO BREAKTHROUGH"
981     << linhas->AreaInvadidaBT(_c) << "\n";
982     cout << "P(" << _x << "," << _y << ")=" << linhas->Pressao(_q, _mi,
983     _kbarra, _h, _pi) << endl;
984 }
985
986 double CSolverInfluxo::r1(double _c, double x, double y) {
987     return sqrt((_c - x) * (_c - x) + y * y);
988 }
989
990 double CSolverInfluxo::r2(double _c, double x, double y) {
991     return sqrt((_c + x) * (_c + x) + y * y);
992 }
993
994 }
```

Capítulo 7

Teste

Todo projeto de engenharia passa por uma etapa de testes. Neste capítulo apresentamos alguns testes do software desenvolvido. Estes testes devem dar resposta aos diagramas de caso de uso inicialmente apresentados (diagramas de caso de uso geral e específicos).

7.1 Teste: Geral

Foi testado todas as possíveis maneiras para se utilizar o software desenvolvido. A seguir está de maneira geral os passos a serem seguidos para utilizar o software:

- Primeiro passo é entrar no Arquivo Simulador para rodar o código. O código é executado no prompt de comando. Em seguida, é perguntado se deseja executar o programa.
- A próxima etapa é o carregamento dos dados do reservatório, das camadas. O programa exige a conferência dos dados informados solicitando sua confirmação .
- Depois de inserir os dados, deve-se escolher qual modelo de fluxo deseja utilizar , Dikstra ou Styles.
- Uma vez resolvido o método escolhido, é solicitado a escolha da configuração dos poços que deseja - 1 injetores e 1 produtor, 1 produtor e 2 injetores ou 2 produtores e 1 injetor.
- Logo em seguida, é pergutado se deseja plotar os graficos de permeabilidade relativa e depois do fluxo fracionário. Assim, gera-se um gráfico dessas propriedades.
- Para finalizar, o código pergunta se deseja executar novamente o código.

```
Deseja executar o codigo? (1 - sim , 2 - nao): 1
#####
# Projeto de programacao pratica
# Professorr: Andre Duarte Bueno
#
# Alunos: David Henrique Lima Dias
#          Julia Rangel Ribeiro
#          Marcos Vinicius de Paula Chaiben
#####
Aperte ENTER para continuar com carregamento de dados...
#####
# Arquivos de dados do reservatorio      #
#####
#     "./in\dadosCamadas.dat"
#     "./in\dadosReservatorio.dat"
#     "./in\dadosRochaReservatorio.dat"

Entre com nome do arquivo com dados do reservatorio: dadosReservatorio.dat
Largura = 400 Comprimento = 200 Bo = 1.3 Bw = 1 deltaP = 4.05e+06 Mi_o = 0.00107 Mi_w = 0.0005 Sg = 0 M = 1.5
0 carregamento foi correto? (1 - sim , 2- nao): 1
#####
# Dados salvos!                         #
#####
Aperte ENTER para continuar com carregamento de dados...
#####
# Arquivos de dados da rocha            #
#####
#     "./in\dadosCamadas.dat"
#     "./in\dadosReservatorio.dat"
#     "./in\dadosRochaReservatorio.dat"

Entre com nome do arquivo com dados da rocha reservatorio: dadosRochaReservatorio.dat
# Carregamento Inicial de Dados
# A etapa de carregamento inicial se repete para todos os modelos
1

#     "./in\dadosReservatorio.dat"
#     "./in\dadosRochaReservatorio.dat"

Entre com nome do arquivo com dados da rocha reservatorio: dadosRochaReservatorio.dat
Kro = 1 Krw = 0.7 Sor = 0.3 phi = 0.1 swi = 0.1
0 carregamento foi correto? (1 - sim , 2- nao): 1
#####
# Dados salvos!                         #
#####
Aperte ENTER para continuar com carregamento de dados...
#####
# Arquivos de entradas de dados         #
#####
#     "./in\dadosCamadas.dat"
#     "./in\dadosReservatorio.dat"
#     "./in\dadosRochaReservatorio.dat"

Entre com nome do arquivo com dados da camada: dadosCamadas.dat
Permeabilidade: 2.96e-13 Espessura: 5
Permeabilidade: 1.97e-13 Espessura: 4
Permeabilidade: 9.87e-14 Espessura: 7
Permeabilidade: 9.87e-15 Espessura: 2
0 carregamento foi correto? (1 - sim , 2- nao): 1
#####
# Dados salvos!                         #
#####
# Todos arquivos foram carregados com sucesso! #
#####
# qual metodo deseja utilizar?
# 1 - Stiles
# 2 - Dijkstra
2
```

Figura 7.1: Carregamento inicial de dados

7.2 Teste: Modelo Styles com configuração de poços - 1 injetor e 1 produtor

```
#####
# Dados salvos!                         #
#####
# Todos arquivos foram carregados com sucesso! #
#####
# qual metodo deseja utilizar?
# 1 - Stiles
# 2 - Dikstra
3

#####
# Arquivos de dados do poco             #
#####
#     "./poco\dadosPoco.dat"

Entre com nome do arquivo com dados do poco: dadosPoco.dat
X = 300 Y = 100 C = 300 Pi = 500
0 carregamento foi correto? (1 - sim , 2- nao): 1
#####
# Dados salvos!                         #
#####
# Qual configuracao de poços deseja calcular?
# 1 - 1 poco produtor e 1 injetor
# 2 - 2 pozos injetores e 1 produtor
# 3 - 1 poco injetor e 2 produtores
4

1
AREA INVADIDA PELA AGUA NO INSTANTE DO BREAKTHROUGH = 4.73741e+06
P(300,100)=7.5311e+10
R = 37 C = 300
5

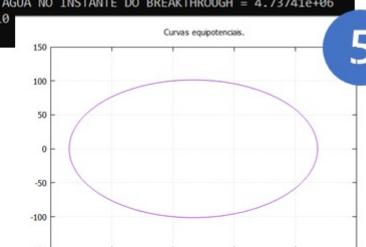
Curvas equipotenciais geradas

```

Figura 7.2: Modelo de Stiles para configuração de 3 poços - 1 injetor e 1 produtor

```
# Qual configuração de poços deseja calcular?
# 1 - 1 poço produtor e 1 injetor
# 2 - 2 poços injetores e 1 produtor
# 3 - 1 poço injetor e 2 produtores

1
AREA INVADIDA PELA AGUA NO INSTANTE DO BREAKTHROUGH = 4.73741e+06
P(300,100)=7.5311e+10
R = 37 C = 300

# Gostaria de plotar os graficos de permeabilidade relativa de corey-brooks?
# 1 - sim, 2 - não

1

# Gostaria de plotar as curvas de fluxo fracionario?
# 1 - sim, 2 - não

1
```

6

Dados sobre área invadida
e Breakthrough

7

Gráficos de permeabilidade relativa e fluxo fracionário gerados

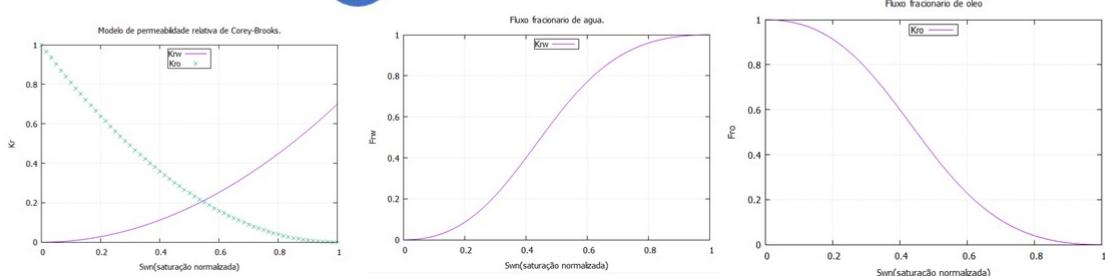


Figura 7.3: Modelo de Stiles para configuração de 3 poços - 1 injetor e 1 produtor

7.3 Teste: Modelo Styles com configuração de poços - 1 produtor e 2 injetores

```
#####
# Dados salvos! #
#####

#####
# Todos arquivos foram carregados com sucesso! #
#####

# qual metodo deseja utilizar?
# 1 - Stiles
# 2 - Dijkstra

1
Vazao Stiles = 362.517 Permeabilidade = 2.96e-13
Vazao Stiles = 193.016 Permeabilidade = 1.97e-13
Vazao Stiles = 169.232 Permeabilidade = 9.87e-14
Vazao Stiles = 4.83519 Permeabilidade = 9.87e-15

Vazao total = 729.6

Posicao frente de avanco de agua = 400
Posicao frente de avanco de agua = 266.216
Posicao frente de avanco de agua = 133.378
Posicao frente de avanco de agua = 13.3378

Eficiencia vertical = 0.277778
Eficiencia vertical = 0.147898
Eficiencia vertical = 0.129673
Eficiencia vertical = 0.00370495

Eficiencia vertical total = 0.559054

Np = 37155.6

O tempo de break through foi : 50.926

Gostaria de exibir os graficos de linhas equipotenciais? (1 - sim, 2 - nao)
```

3

```
#####
# Arquivos de dados do poço #
#####

#     "./poco\dadosPoco.dat"

Entre com nome do arquivo com dados do poço: dadosPoco.dat

X = 300 Y = 100 C = 300 Pi = 500

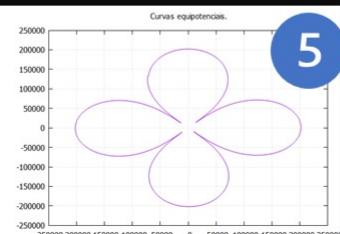
O carregamento foi correto? (1 - sim , 2- nao): 1

#####
# Dados salvos! #
#####

# Qual configuração de poços deseja calcular?
# 1 - 1 poço produtor e 1 injetor
# 2 - 2 poços injetores e 1 produtor
# 3 - 1 poço injetor e 2 produtores

2
AREA INVADIDA PELA AGUA NO INSTANTE DO BREAKTHROUGH = 565487
P(300,100)=2.07366e+10
R = -285714 C = 300
```

4

Curvas
equipotenciais
geradas

5

Figura 7.4: Modelo de Stiles para configuração de 3 poços - 2 injetores e 1 produtor

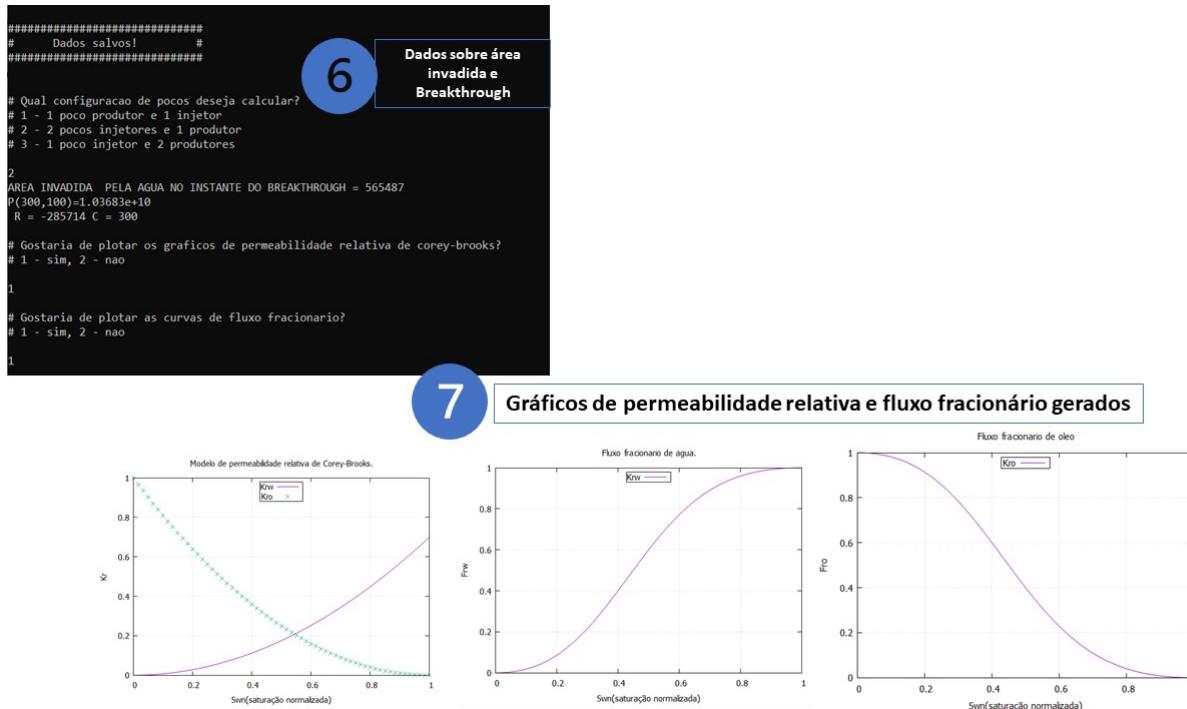


Figura 7.5: Modelo de Stiles para configuração de 3 poços - 2 injetores e 2 produtor

7.4 Teste: Modelo Styles com configuração de poços - 2 produtores e 2 injetor

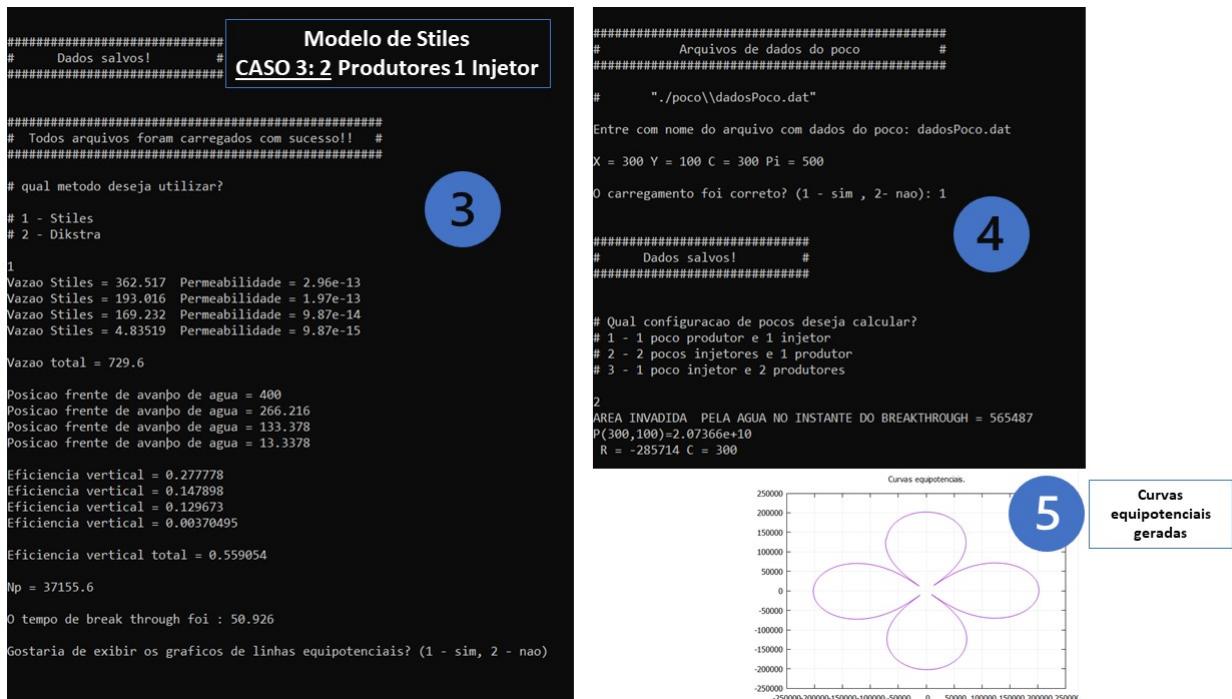


Figura 7.6: Modelo de Stiles para configuração de 3 poços - 1 injetor e 2 produtores

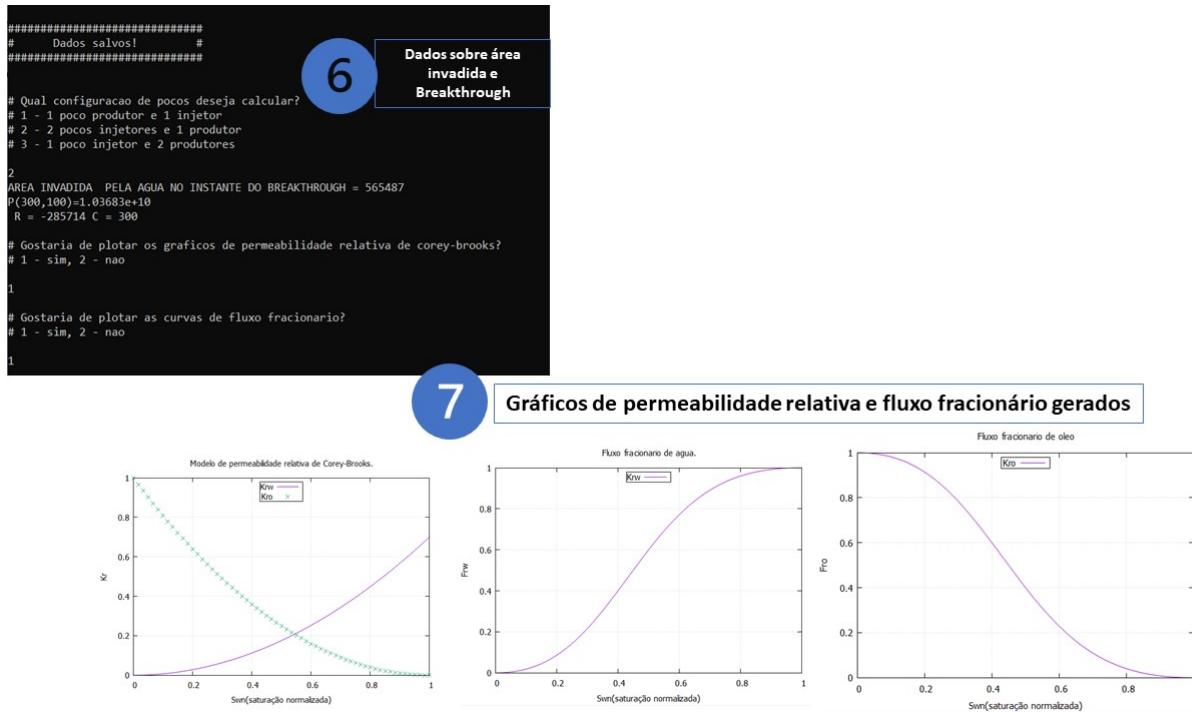


Figura 7.7: Modelo de Stiles para configuração de 3 poços - 1 injetor e 2 produtores

7.5 Teste: Modelo Dijkstra com configuração de poços - 1 produtor e 1 injetor

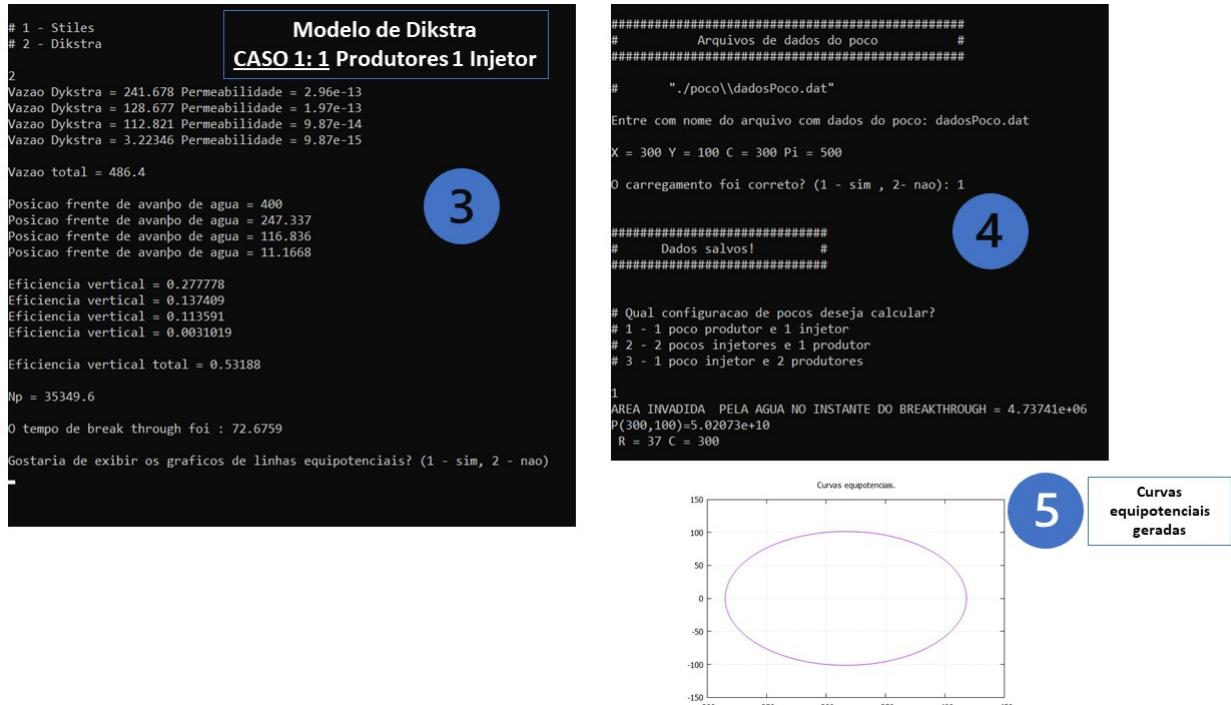


Figura 7.8: Modelo de Dijkstra para configuração de 3 poços - 1 injetor e 1 produtor

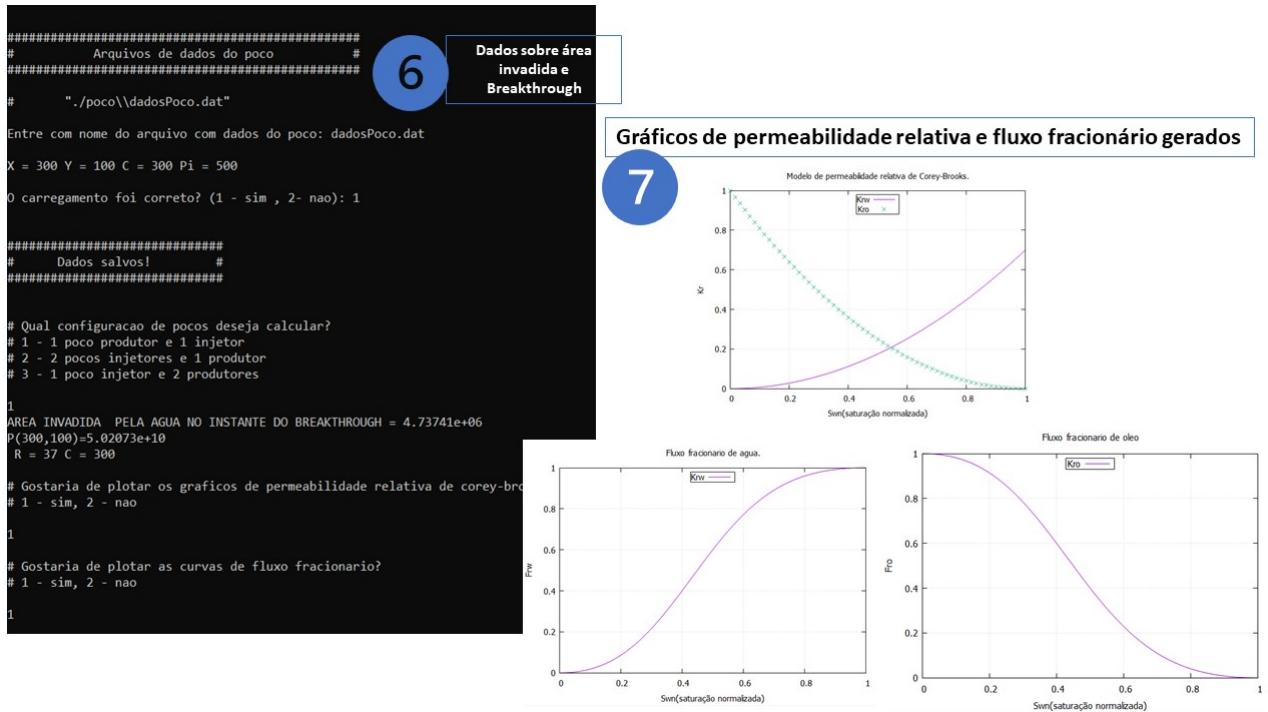


Figura 7.9: Modelo de Dikstra para configuração de 3 poços - 1 injetor e 1 produtor

7.6 Teste: Modelo Dikstra com configuração de poços - 1 produtor e 2 injetores

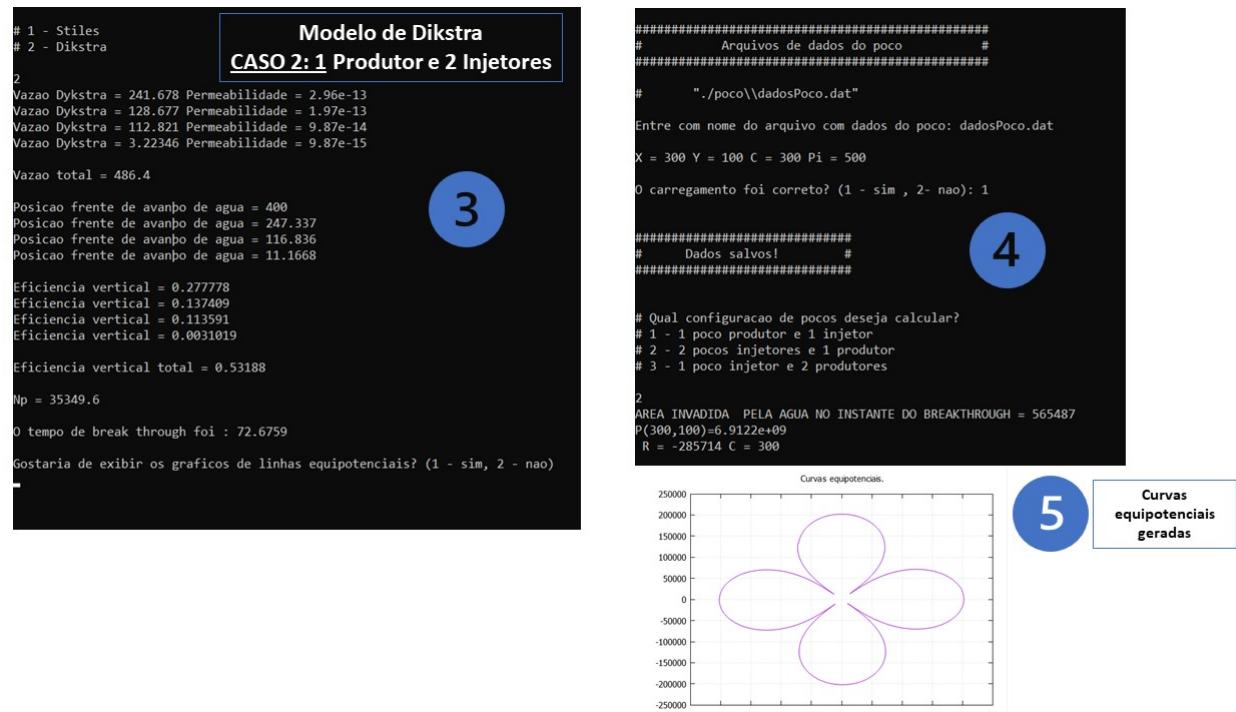


Figura 7.10: Modelo de Dikstra para configuração de 3 poços - 2 injetores e 1 produtor

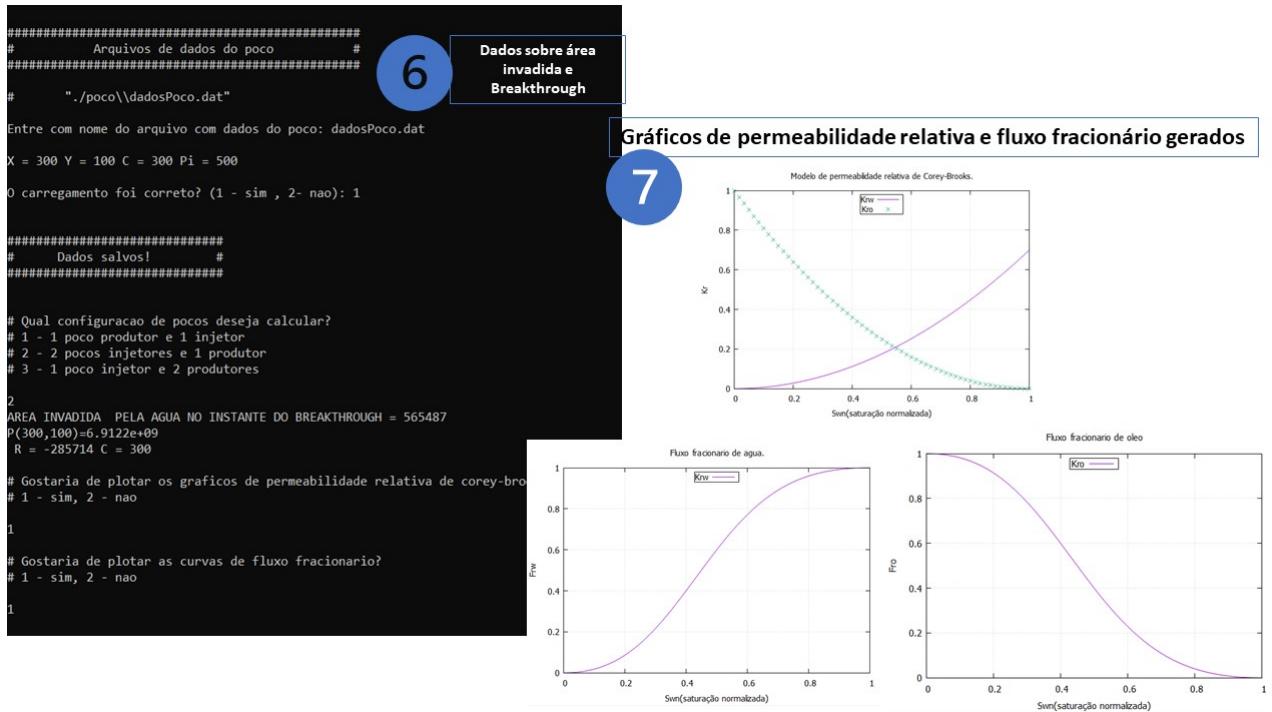


Figura 7.11: Modelo de Dikstra para configuração de 3 poços - 2 injetores e 1 produtor

7.7 Teste: Modelo Dykstra com configuração de poços - 2 produtores e 1 injetor

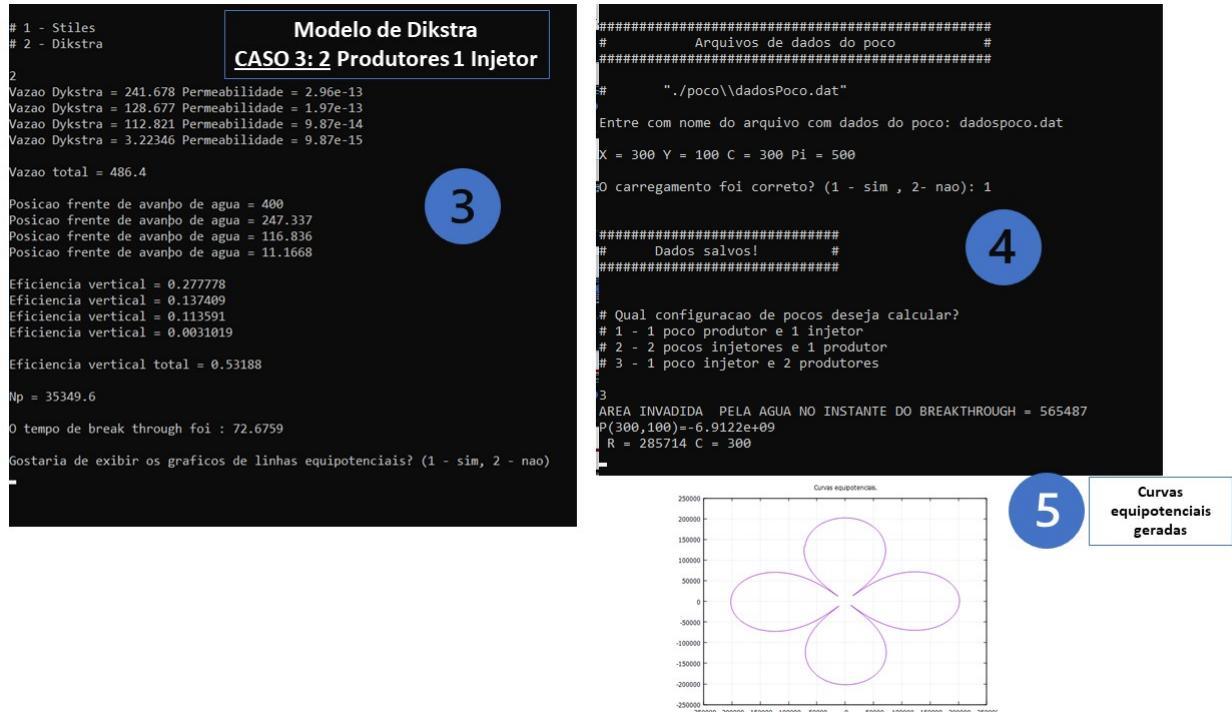


Figura 7.12: Modelo de Dikstra para configuração de 3 poços - 1 injetor e 2 produtores

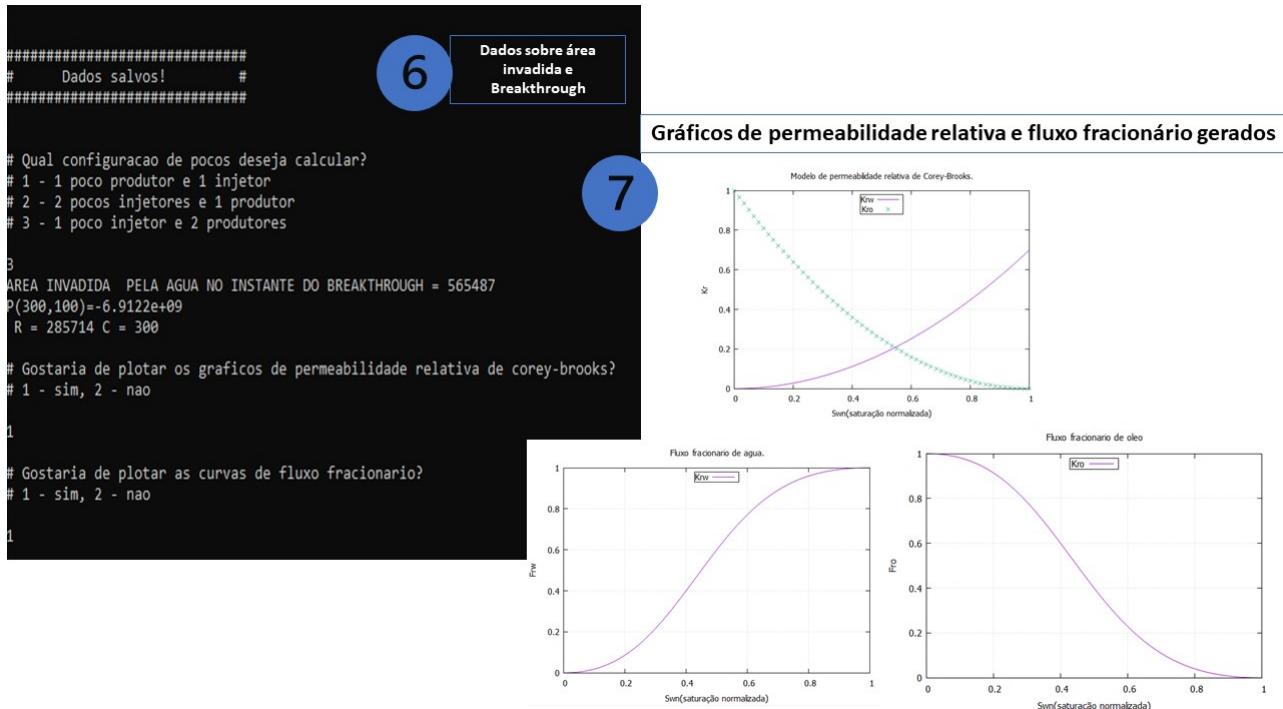


Figura 7.13: Modelo de Dijkstra para configuração de 3 poços - 1 injetor e 2 produtores

Capítulo 8

Documentação

Todo projeto de engenharia precisa ser bem documentado. Neste sentido, apresenta-se neste capítulo a documentação de uso do "software XXXX". Esta documentação tem o formato de uma apostila que explica passo a passo como usar o software.

8.1 Documentação do usuário

Descreve-se aqui o manual do usuário, um guia que explica, passo a passo a forma de instalação e uso do software desenvolvido.

8.1.1 Como instalar o software

Para instalar os softwares, execute o seguinte passo a passo:

- Em Windows: Faça o download de um compilador, como por exemplo o g++ (por linhas de comando utilizando o MinGw); e o Dev C++, disponível em <https://devc.softonic.com.br/>. Compile o simulador e execute-o.
- Em Linux: Abra o terminal, vá para o diretório onde está o simulador, faça a compilação, e em seguida a execução.

8.1.2 Como rodar o software

No capítulo de teste têm todas as informações necessárias para que se possa rodar os softwares.

8.2 Documentação para desenvolvedor

Apresenta-se nesta seção a documentação para o desenvolvedor, isto é, informações para usuários que querem modificar, aperfeiçoar ou ampliar este software.

8.2.1 Dependências

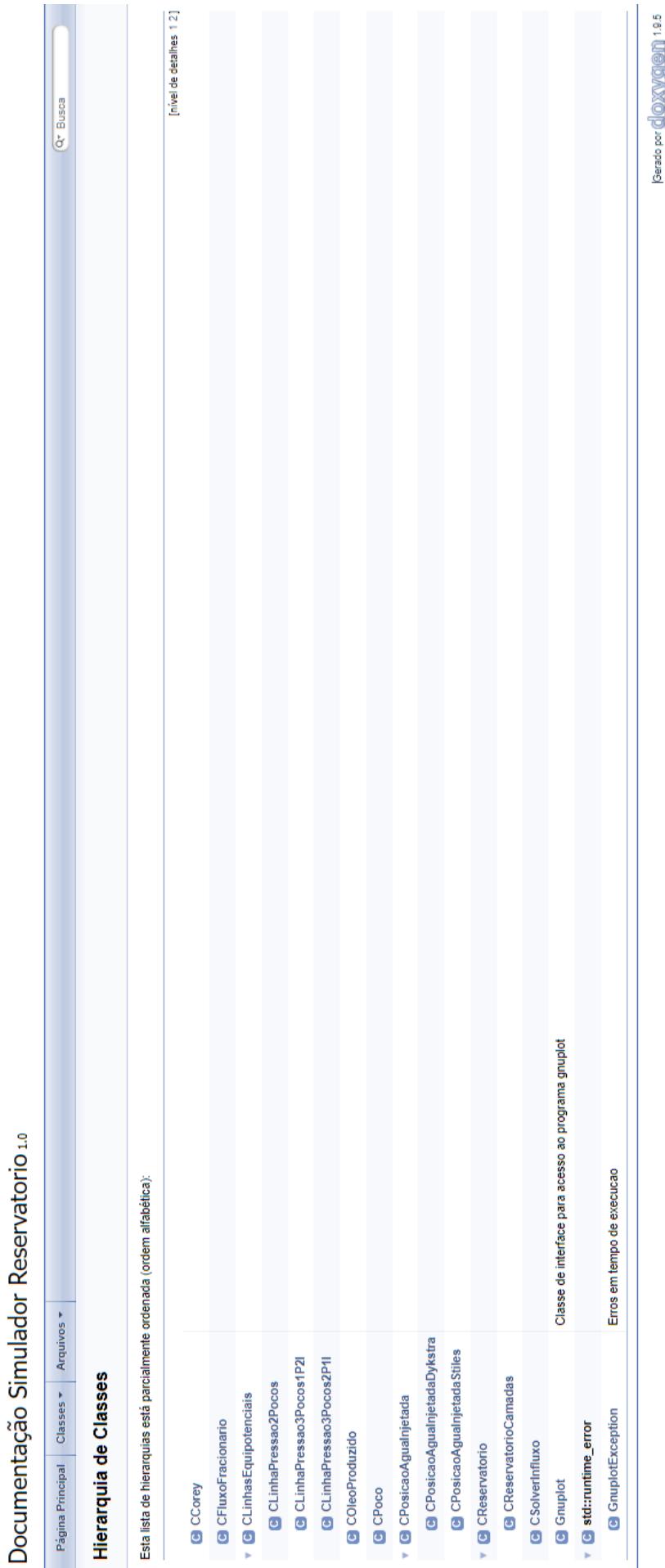
Para compilar o software é necessário atender as seguintes dependências:

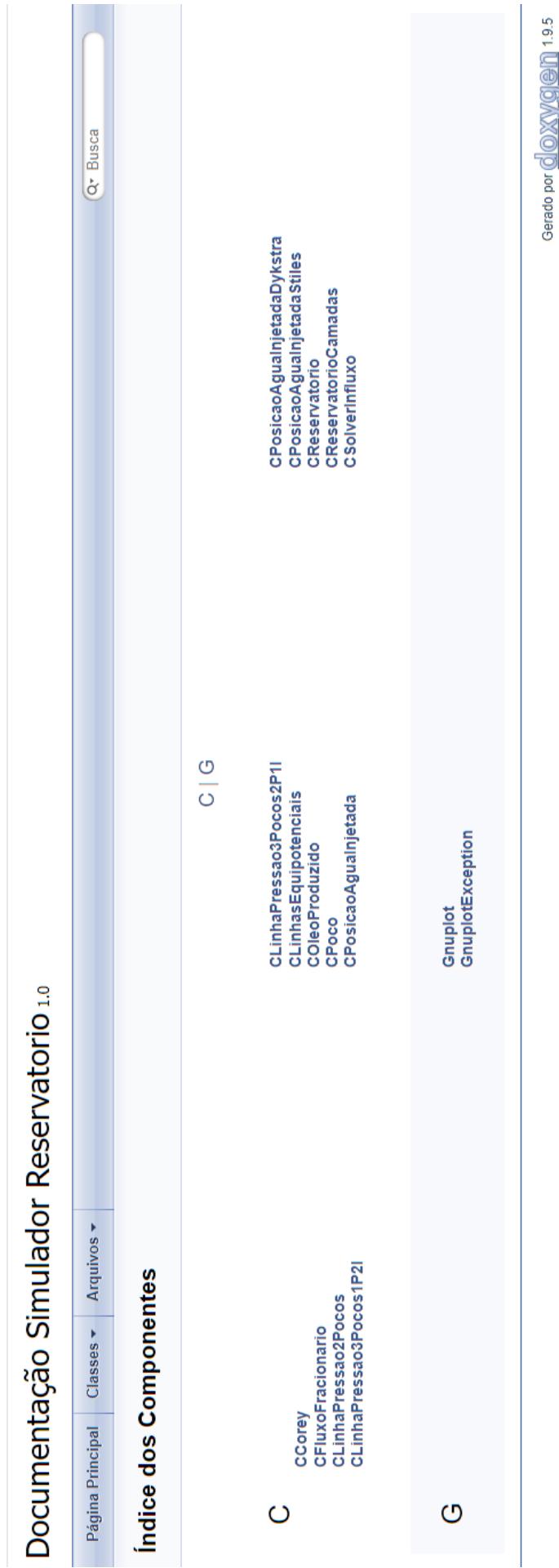
- Instalar o compilador g++ da GNU disponível em <http://gcc.gnu.org>. Para instalar no GNU/Linux use o comando `sudo yum install gcc`.
- Biblioteca CGnuplot; os arquivos para acesso a biblioteca CGnuplot devem estar no diretório com os códigos do software;
- O software `gnuplot`, disponível no endereço <http://www.gnuplot.info/>, deve estar instalado. É possível que haja necessidade de setar o caminho para execução do `gnuplot`.
- Os arquivos com dados de reservatório, poços e fluidos podem ser mudados desde que haja coerência com os mesmos, ou seja, os parâmetros de um reservatório e suas unidades de medida precisam ser respeitadas. Se essas alterações forem realizadas, é preciso que as mesmas sejam feitas diretamente no código e nos arquivos de dados de reservatório com extensão .txt.

8.2.2 Como gerar a documentação usando doxygen

A documentação do código do software deve ser feita usando o padrão JAVADOC, conforme apresentada no Capítulo - Documentação, do livro texto da disciplina. Depois de documentar o código, use o software `doxygen` para gerar a documentação do desenvolvedor no formato html. O software `doxygen` lê os arquivos com os códigos (*.h e *.cpp) e gera uma documentação muito útil e de fácil navegação no formato html.

Página Principal	Classes	Arquivos
<p>(Q) Busca</p>		
<h2>Listas de Classes</h2>		
	Aqui estão as classes, estruturas, uniones e interfaces e suas respectivas descrições:	
C CCorey		
C CFluxoFracionario		
C CLinhaPressaoPocos		
C CLinhaPressao3PocosIP2I		
C CLinhaPressao3PocosZP1I		
C CLinhaseEquipotenciais		
C ColeoProduzido		
C CPOCO		
C CPosicaoAQualInjetada		
C CPosicaoAQualInjetadaDijkstra		
C CPosicaoAQualInjetadaStiles		
C CReservatorio		
C CReservatorioCamadas		
C CSolverInfluxo		
C GnuPlot	Classe de interface para acesso ao programa gnuplot	
C GnuPlotException	Eros em tempo de execução	





Capítulo 9

Referências

1. BIRD, R; STEWART, W.; LIGHTFOOT, E. Transport Phenomena. Wiley, 1987. (Wiley International edition).
2. DAKE, L. Fundamentals of reservoir engineering. [S.I.]: ELSEVIER SCIENCE B.V., 1978.
3. ROSA, A.; CARVALHO, R. de S.; XAVIER, J. Engenharia de Reservatórios de Petróleo. Interciência, 2006.
4. SHENG, J. Modern Chemical Enhanced Oil Recovery Theory and Practice .ELSE VIER , 2011.
5. PICO, C. Introdução a Engenharia de Reservatório. Curso de Engenharia de Exploração de Petróleo e Gás. Data completa 2019. Notas de Aula. Universidade Estadual do Norte Fluminense.
6. QUEIROZ, W. Elementos de Matemática Aplicada. Curso de Engenharia de Exploração de Petróleo e Gás. Data completa 2013. Notas de Aula. Universidade Estadual do Norte Fluminense.
7. CAUDLE, B. H., Fundamentals of Reservoir Engineering. Dallas, Texas, USA, SPE of AIME, 1968.
8. CRAIG, F.F., 1971. The Reservoir Engineering Aspects of Waterflooding, SPE Monograph, Dallas.
9. DEPPE, J. C., Injection Rates – The Effect of Mobility Ratio, Area Swept, and Pattern. SPEJ, june 1961.
10. MUSKAT, M., Physical Principles of Oil Production. New York, NY, USA, McGraw-Hill Book Company, Inc., 1949. ROSA, A.;
11. CARVALHO, R.; XAVIER, D.; Engenharia de reservatórios de petróleo, Editora Interciência Ltda: Rio de Janeiro, 2006.

12. SMITH, J. T. & COBB, W. M.; Waterflooding. Midwest Office of the Petroleum Technology Transfer Council (U.S.).1997.
13. WILLHITE, G. P., Waterflooding, Society of petroleum,1986.