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
- Versão 1:
Prof. André Duarte Bueno

MACAÉ - RJ
DEZEMBRO - 2021

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 3

2.1.3 Requisitos funcionais 5

2.1.4 Requisitos não funcionais 6

2.2 Casos de uso do software 6

2.2.1 Diagrama de caso de uso geral 6

2.2.2 Diagrama de caso de uso específico 7

3 Elaboração **9**

3.1 Análise de domínio 9

3.2 Conceitos Fundamentais 11

3.2.1 Conceitos de Propriedades de Rochas e Fluidos 11

3.2.2 Conceitos Teóricos: 15

3.3 Formulações Matemáticas 21

3.3.1 Modelo de Fluxo Bifásido (1D) 21

3.3.2 Modelo de Fluxo Bifásido Areal (2D) 22

3.3.3 Modelo de Fluxo Bifásido em Sistemas Estratificados (3D) 28

3.4 Diagrama de Pacotes – assuntos 30

4 AOO – Análise Orientada a Objeto **32**

4.1 Diagramas de classes 32

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 36

5 Projeto **38**
5.1 Projeto do sistema 38
5.2 Projeto orientado a objeto – POO 39
5.3 Diagrama de componentes 40
5.4 Diagrama de implantação 41

6 Implementação **43**
6.1 Código fonte 43

7 Teste **213**
7.1 Teste: Teste modelo de Styles e Linhas Equipotencias com Configuração de 2 poços 213

8 Documentação **218**
8.1 Documentação do usuário 218
8.1.1 Como instalar o software 218
8.1.2 Como rodar o software 218
8.2 Documentação para desenvolvedor 218
8.2.1 Dependências 219
8.2.2 Como gerar a documentação usando doxygen 219

9 Referências **224**

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;
  - Propro 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 depermeabilidade relativa a partir do Modelo de Corey-Brooks;
  - Desenvolver a curva de fluxo fracionário para um reservatório bifasico;
  - 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.

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

Tabela 2.1: Característica do software

Nome	Modelos de Deslocamentos Imiscíveis Bifásico Usados no Processo de Recuperação Secundária de Petróleo
Componentes principais	**Fluxo Bifásico Areal** para cálculo do comportamento das propriedades de reservatório homogênio em esquemas de injeção em malhas **Fluxo Bifásico em Sistema Estratificado** para previsão de desempenho num processo de recuperação secundária do óleo de acordo com o modelo de Dykstra-Parsons (1950) e Stiles (1949).
Missão	Calcular a permeabilidade relativa a partir do modelo de Corey-Brooks; Calcular comportamento da pressão em reservatório homogênio em esquemas de injeção em malhas; Calcular área invadida pela injeção de água no instante do "breakthrough"; Cálculo da frente de avanço no intante do "breakthrough" (BT); Cálculo da eficiência vertical no BT; Cálculo do volume de óleo recuperado total; Cálculo do tempo necessário do BT (todo óleo recuperável possível por esse método de injeção) .

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 irredutivel;
  - Saturação de óleo residual;
  - Viscosidade da água [Pa.s];
  - Viscosidade do óleo [Pa.s];
  - Mobilidade [Kg.m^3].
- **Dados/Atributos relativos ao teste de injeção:**
  - Vazão de injeção [m^3/s];
  - Esquemas de injeção;
  - Volume de óleo produzido [m^3];

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 *Windows*, *GNU/Linux* ou *Mac*.

2.2 Casos de uso do software

A tabela 2.2 apresenta um caso de uso do sistema, bem como os diagramas de caso de uso.

Tabela 2.2: Caso de uso geral do sistema.

Nome do caso de uso:	Modelagem de Fluxo Bifásico Imiscísvel em Reservatório
Resumo/descrição:	Cálculo do desempenho no processo de recuperação secundária Calculo do fluxo fracionário
Etapas:	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 2.1 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.

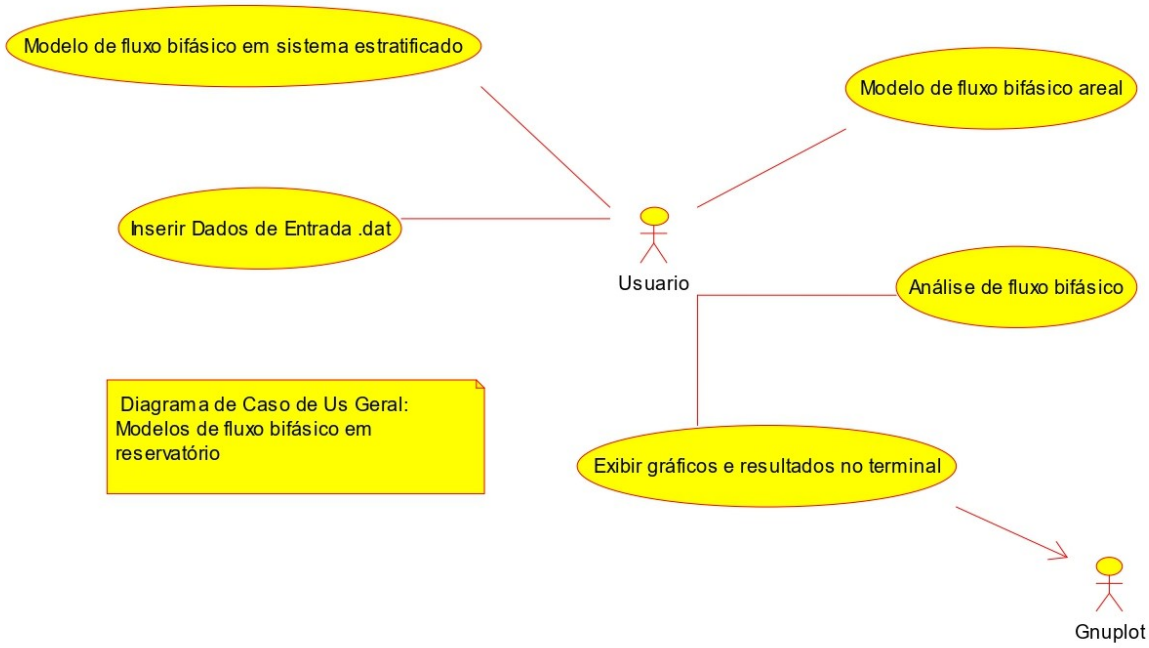


Figura 2.1: Diagrama de caso de uso geral – Modelos de fluxo bifásico em reservatórios

2.2.2 Diagrama de caso de uso específico

O diagrama de caso de uso específico da Figura 2.2 é um detalhamento do caso de uso para os calculos 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.

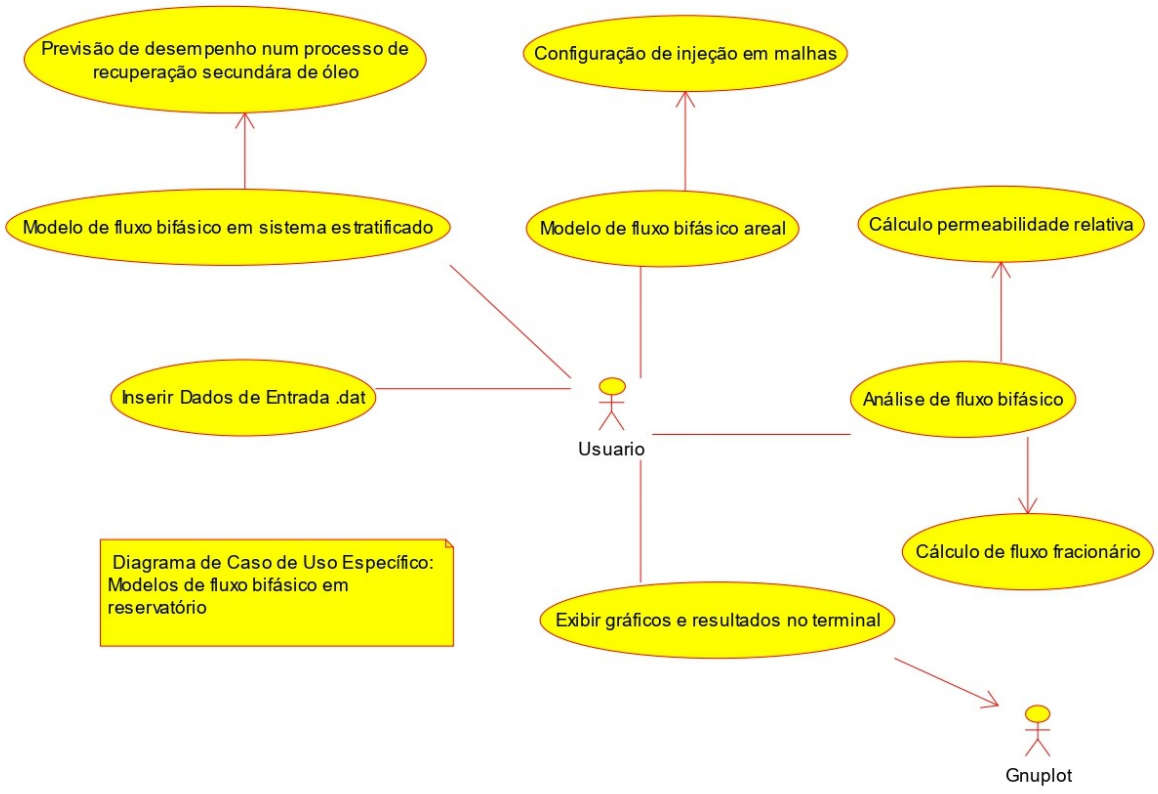


Figura 2.2: Diagrama de caso de uso específico – Modelos de fluxo bifásico em reservatórios

Capítulo 3

Elaboração

Depois da denição dos objetivos, da especicação do software e da montagem dos primeiros diagramas de caso de uso, a etapa do projeto de engenharia envolve a identicação e o estudo dos conceitos relacionados ao sistema a ser desenvolvido, isto é, a análise de domínio e a identicação 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 primaria dos reservatórios de petróleo: Suplementando a com energia secundária através da injeção ou reduzindo as resistências viscosas e/ou capilares por meio de metodos 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 fuidos é essencial para o desenvolvimento de qualquer metodologia para aumentar o fator de recuperação do petróleo. Assim, será mostrado denições de porosidade absoluta e relativa, molhabilidade, permeabilidade absoluta, fator de recuperação, mobilidade, razão mobilodade, entre outros.

3.2.1 Conceitos de Propriedades de Rochas e Fluidos

- **Porosidade:**

  A porosidade de uma rocha é defnida 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 denida como a razão entre o volume poroso (Vp) e o volume total (Vt) da amostra (Equação 3.1), onde o volume total é dado pela soma do espaço poroso e da fase sólida (Equação 3.2).

$$\phi = \frac{V_p}{V_t} \tag{3.1}$$

$$V_t = V_p + V_{sólida} \tag{3.2}$$

- **Conservação de Massas:**

  A equação da continuidade é desenvolvida efetuand-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)

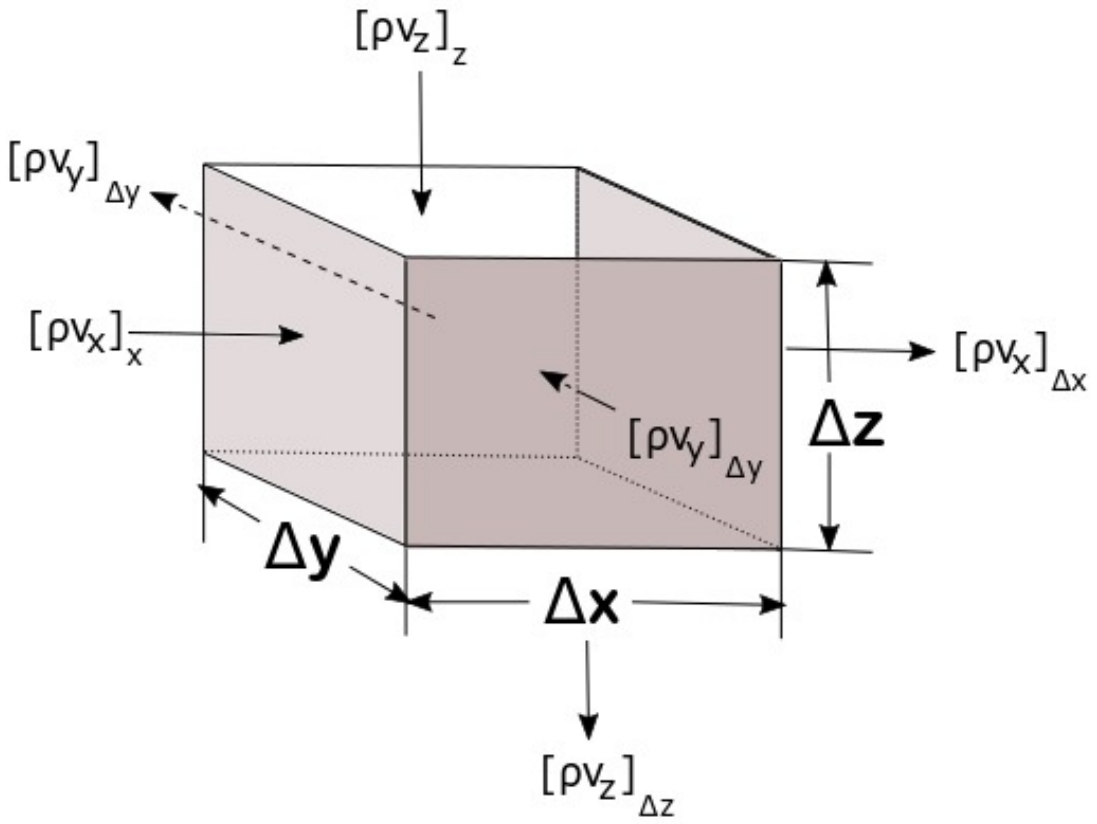


Figura 3.1: Balanço de Massas (Autor)

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

sendo:

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

- **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 signica que em um fluxo.multifásico, um fludo tem mais anidade com o meio poroso que outros uidos 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 petr óleo. 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 ($\triangle P$) e do comprimento do meio poroso (L), conforme mostra a Equação [Rosa et al., 2006, Lake et al., 1989].

$$q = \frac{kA\triangle P}{\mu L} \tag{3.5}$$

A permeabilidade absoluta pode ser calculada isolando k na equação de Darcy conforme Equação3.6.

$$k = \frac{q\mu L}{A\triangle P} \tag{3.6}$$

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 (ke), 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 (kw e ko) também pode ser realizado usando a Lei de Darcy conforme Equações 3.7 e 3.8 [Rosa et al., 2006].

$$k_{rw} = \frac{k_w L q_w}{A \triangle P} \tag{3.7}$$

$$k_{ro} = \frac{k_o L q_o}{A \triangle P} \tag{3.8}$$

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

$$k_{rw} = \frac{k_w}{k} \tag{3.9}$$

$$k_{ro} = \frac{k_o}{k} \tag{3.10}$$

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 uxo 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:

$$\lambda_w = \frac{k_w}{\mu_w}, \tag{3.11}$$

$$\lambda_o = \frac{k_o}{\mu_o}, \tag{3.12}$$

$$\lambda_g = \frac{k_g}{\mu_g}, \tag{3.13}$$

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 :

$$M = \frac{k_d}{\mu_d}\frac{\mu_o}{k_o} = \frac{\lambda_d}{\lambda_o}, \tag{3.14}$$

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

$$M = \frac{k_w}{\mu_w}\frac{\mu_o}{k_o} = \frac{k_w}{\mu_w}\frac{\mu_o}{k_{ro}}, \tag{3.15}$$

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_oe o total, $f_t$são definidos, respectivamente, pelas Equações 3.16 , 3.17 e 3.18. (ROSA, 2006).

$$f_w = \frac{u_w}{ut} \tag{3.16}$$

$$f_o = \frac{u_o}{ut} \tag{3.17}$$

$$f_t = f_w + f_o \tag{3.18}$$

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

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

$$Ev = \frac{Volume.invadido.pela.água}{Volume.total.da.malha} \tag{3.19}$$

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 Varrido 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:

$$E_A = A_{inv}/A_t \tag{3.20}$$

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).

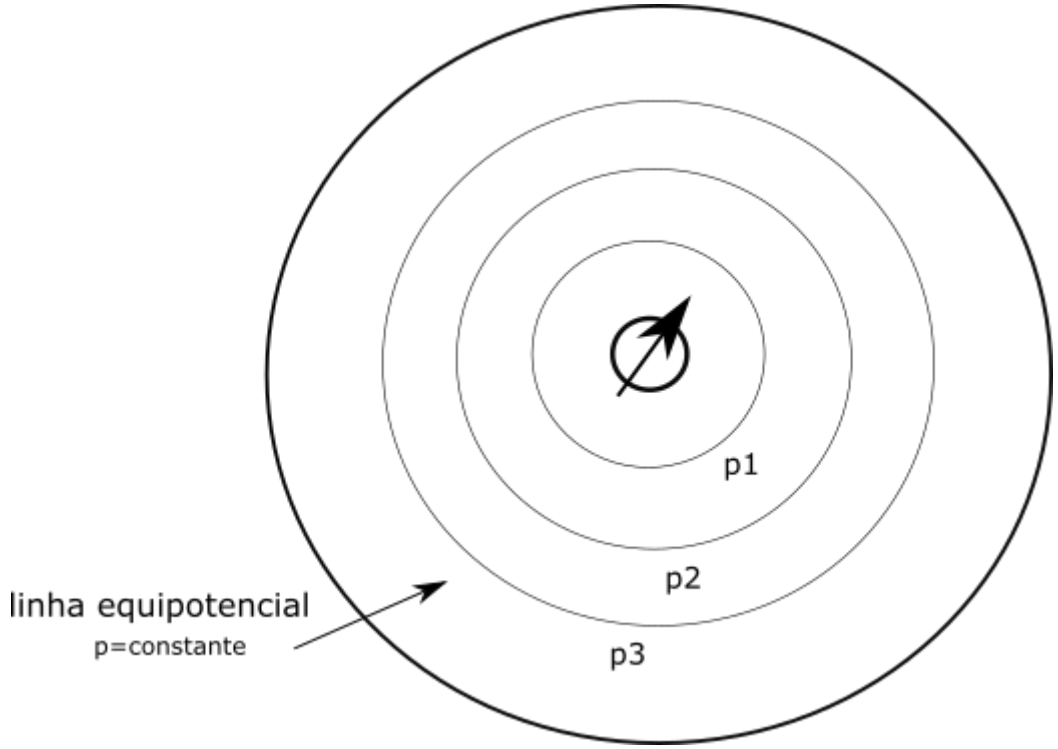


Figura 3.2: Linhas equipotenciais concêntricas em um reservatório infinito.

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

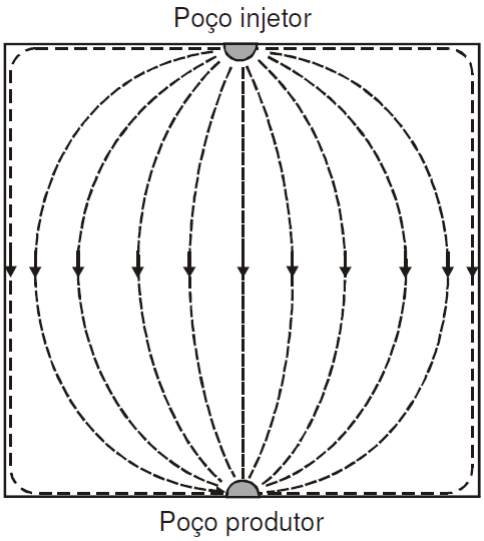


Figura 3.3: Linhas de Fluxo (ROSA ET AL., 2006).

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).

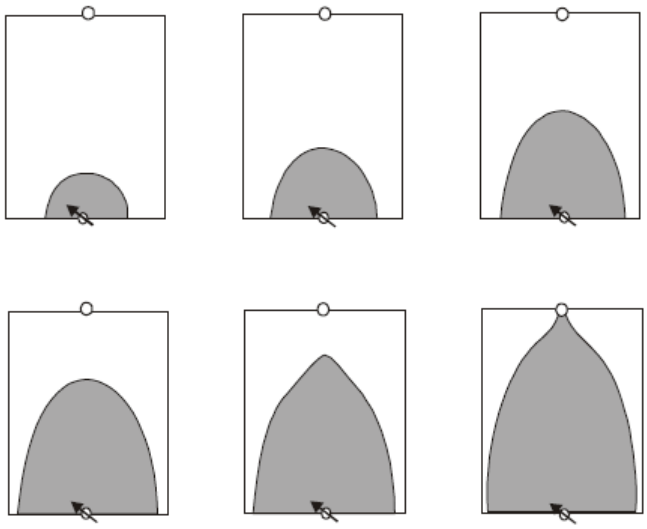


Figura 3.4: Evolução da área invadida em uma malha em linha direta (ROSA ET AL., 2006).

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.

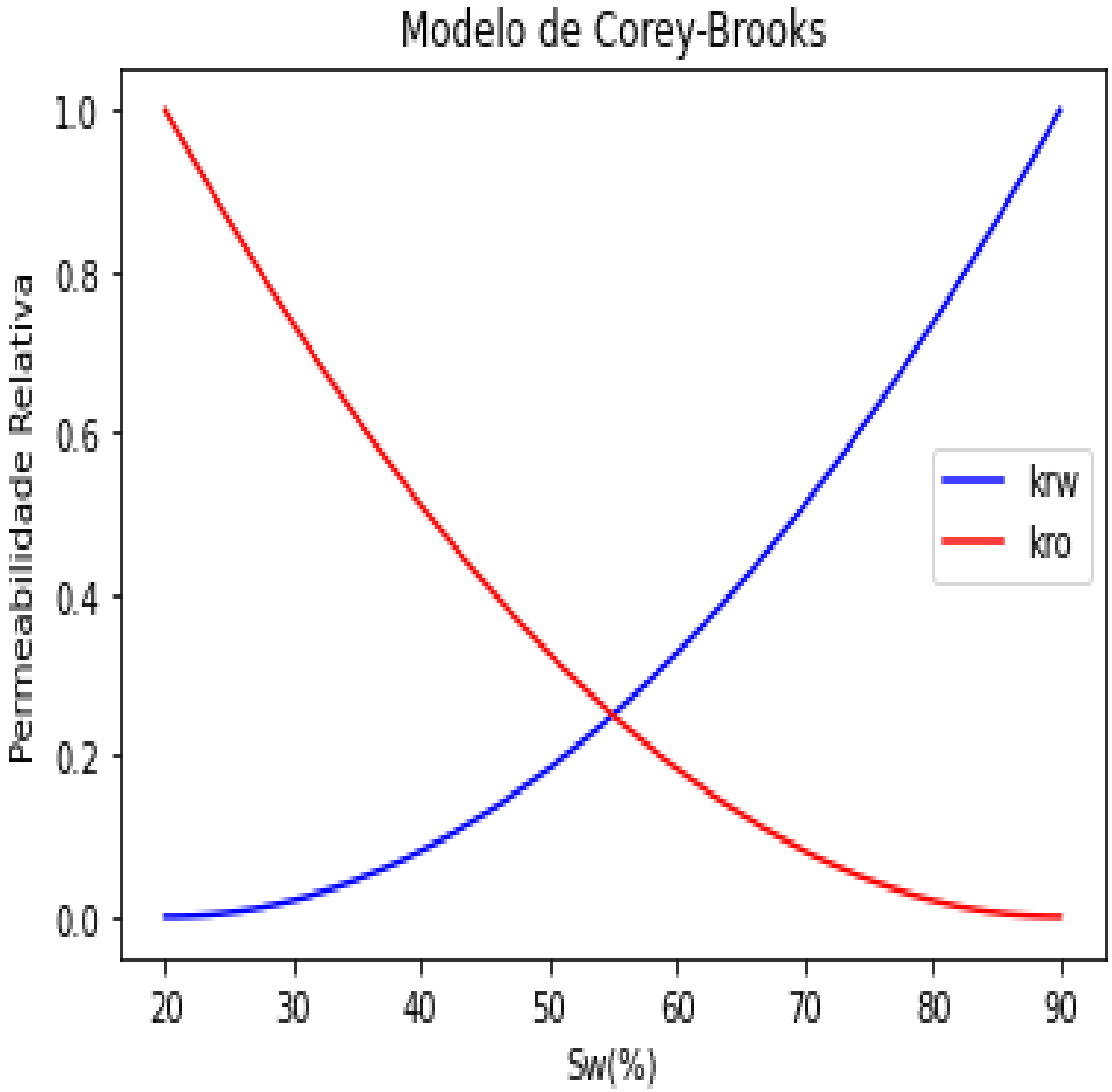


Figura 3.5: Modelo de Corey-Brooks para permeabilidade relativa(Autor).

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

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

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

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

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 irredutivel , S_w =Saturação de água, S_{wi} =Saturação de água irredutível, 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

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

- Lei de Darcy

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

- Velocidade de deslocamento do óleo

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

- Velocidade de deslocamento da água

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

- Função Fluxo

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

- Pressão Capilar

$$P_c=P_o-P_w \tag{3.29}$$

- Derivada da Função Fluxo

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

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.

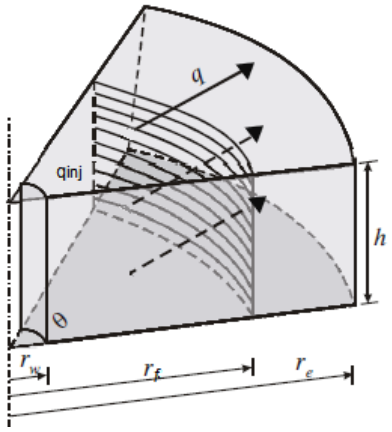


Figura 3.6: Sistema radial em poços de injeção (Adaptado de ROSA ET AL., 2006).

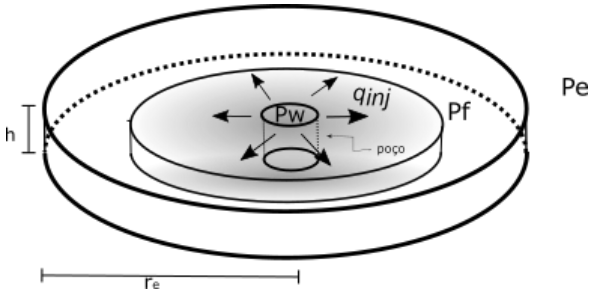


Figura 3.7: Sistema radial em poços de injeção.

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

$$q_{inj} = \frac{C_1 h \lambda_o \triangle P}{log\left(\frac{rf}{rw}\right)\frac{1}{M} + log\left(\frac{re}{rf}\right)}, \tag{3.31}$$

onde $\triangle P = P_e - P_w$ 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. 3.31com $r_f = r_{w.}$, (DEPPE J.,1961).Fazendo a injetividade inicial ($q_{inj,i}$), temos:

$$q_{inj,i} = \frac{C_1 h \lambda_o \triangle (A_{inv})_{BT}, P}{log\left(\frac{re}{rf}\right)}. \tag{3.32}$$

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

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

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 quaise 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 convenciona-se 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).

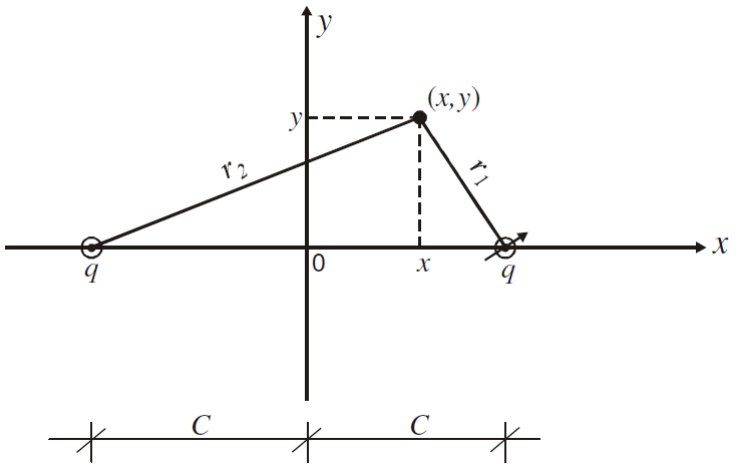
Utilizando a aproximação logarítmica para representar a solução do modelo da fonte

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:

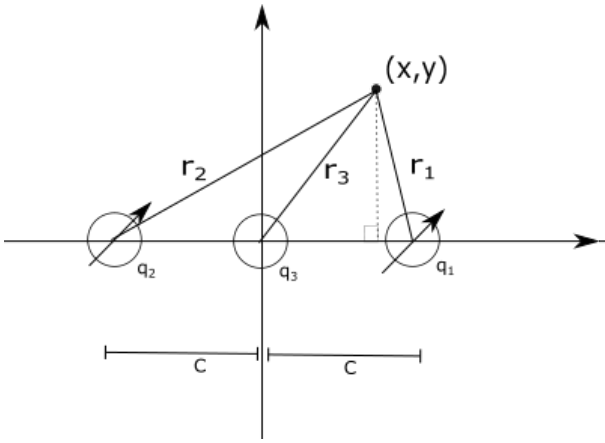
$$p_D(r_D, t_D) \equiv \frac{2\pi kh[p_i - p(r,t)]}{q\mu} = \frac{1}{2}[ln(t_D/r_D^2) + 0,80907] \tag{3.34}$$

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_2e 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.

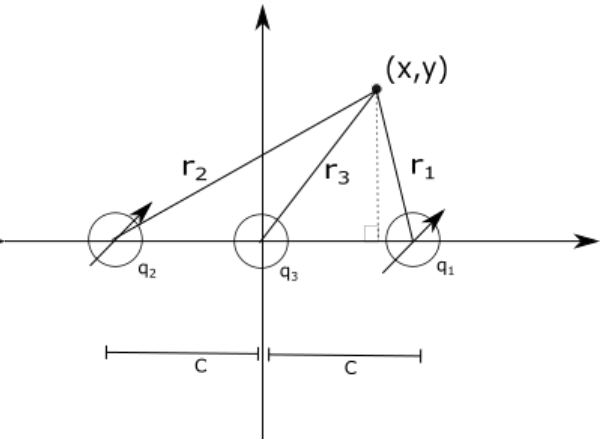
Conforme se observa, usando a aproximação de longo tempo (aproximação logarítmica da solução do modelo da fonte linear) para o comportamento transiente de pressão, Eq. 3.34, e aplicando o princípio da superposição de efeitos, obteve-se uma solução para fluxo permanente, já que não há dependência do tempo 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 injetores, 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_1e r_3 seja constante (ROSA ET AL., 2006). Com isso, após algumas manipulações, obtem-se as Eqs. 3.36, 3.39 e 3.43, que sarão origem a gráficos mostrando o comportamento dessas linhas de pressão consntante 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 "breakthrogh" (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.

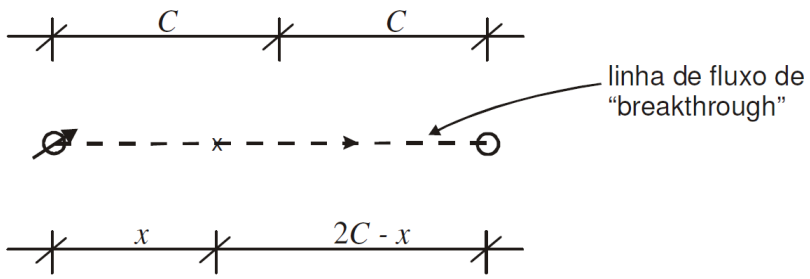


Figura 3.10: Sistema composto de dois poços: injetor e produtor (ROSA ET AL., 2006).

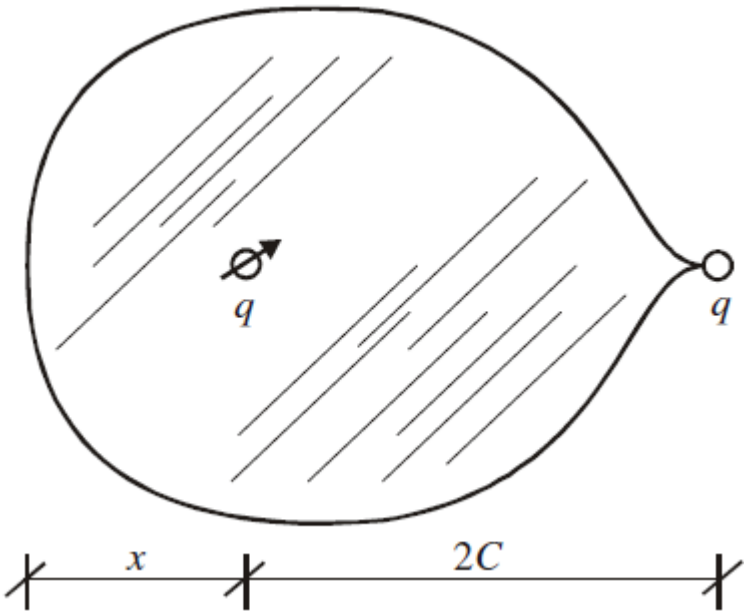


Figura 3.11: Ilustração da área invadida pela água no instante de "breakthrough" em um sistema composto de um injetor e de um produtor (ROSA ET AL., 2006)

- Caso 1 :

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

$$x^4 + y^4 + 2x^2y^2 + x^2R - y^2R" = \frac{C^2R"}{2}, \tag{3.36}$$

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

- Caso 2 :

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

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

Transformando em coordenadas polares,

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

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

- Caso 3 :

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

$$x^4 + y^4 + 2x^2y^2 + x^2R - y^2R" = \frac{C^2R"}{2}. \tag{3.43}$$

Transformando em coordenadas polares,

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

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

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):

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

- Vazão de injeção numa camada j:

$$q_j = q_w = q_o = \frac{k_w A_j \triangle p}{\mu_w L} = \frac{k_o A_j \triangle p}{\mu_o L} \tag{3.46}$$

- Volume de óleo produzido por camada em condições padrão:

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

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

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

Considerando o Método de Dykstra-Parsons, temos que:

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

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

- A eficiência do varrido vertical é definida matematicamente como:

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

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

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

- Volume de óleo produzido por camada em condições padrão:

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

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

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

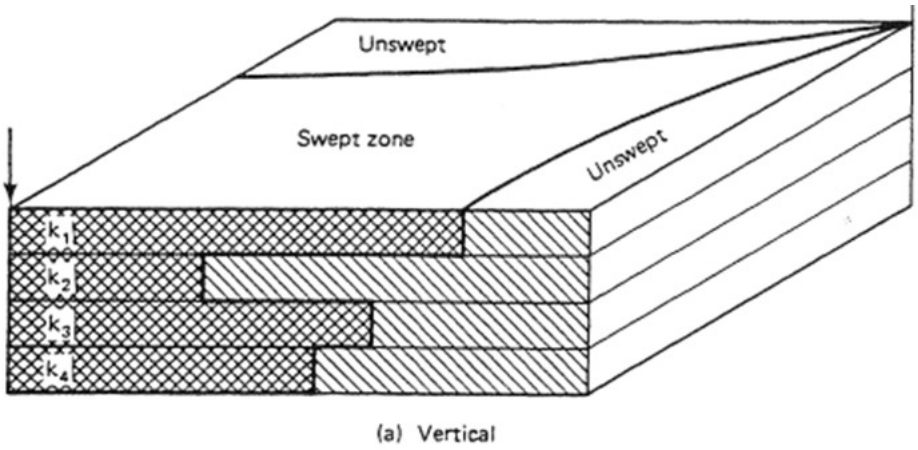


Figura 3.12: Reservatório Estratificado (SMITH, COBB , 1997).

3.4 Diagrama de Pacotes – assuntos

Com base na análise do domínio do software desenvolvivo, foram identificadps ps seguintes pacotes:

- **Propriedades da Rocha e dos fluidos:** é um pacote que possui os dados das propriedades da rocha e dos fuidos, que compõem o meio poroso. Sua função é fornecer estas propriedades para o modelo de recuperação;
- **Métodos de Deslocamento Imiscivel 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 modicações do objeto com o tempo. O resultado da análise é um conjunto de diagramas que identicam os objetos e seus relacionamentos.

4.1 Diagramas de classes

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

Figura 4.1: Diagrama de classes

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 grácos 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 metódos 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 deposicional;
- 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 sequência geral

Veja o diagrama de seqüência na Figura 4.2.

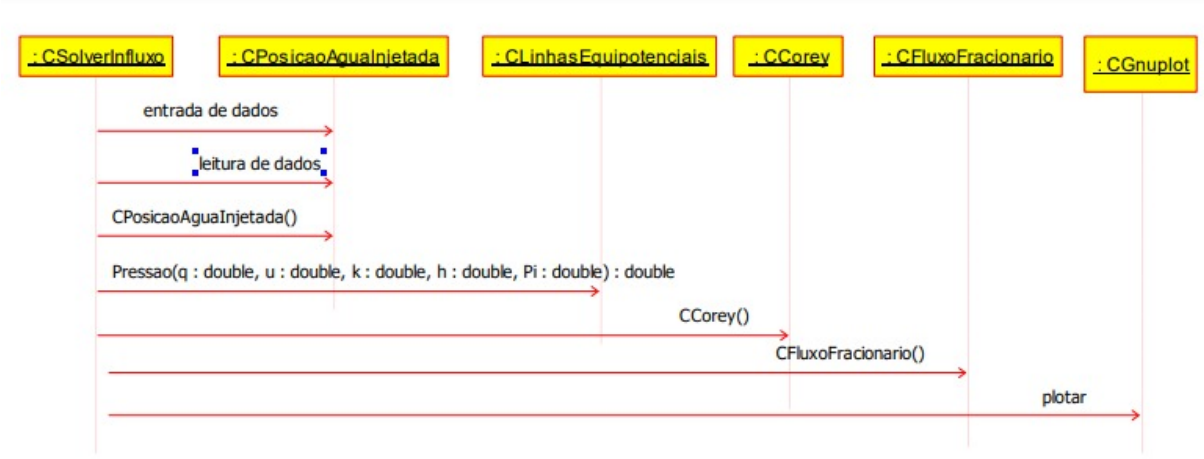


Figura 4.2: Diagrama de seqüência

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.

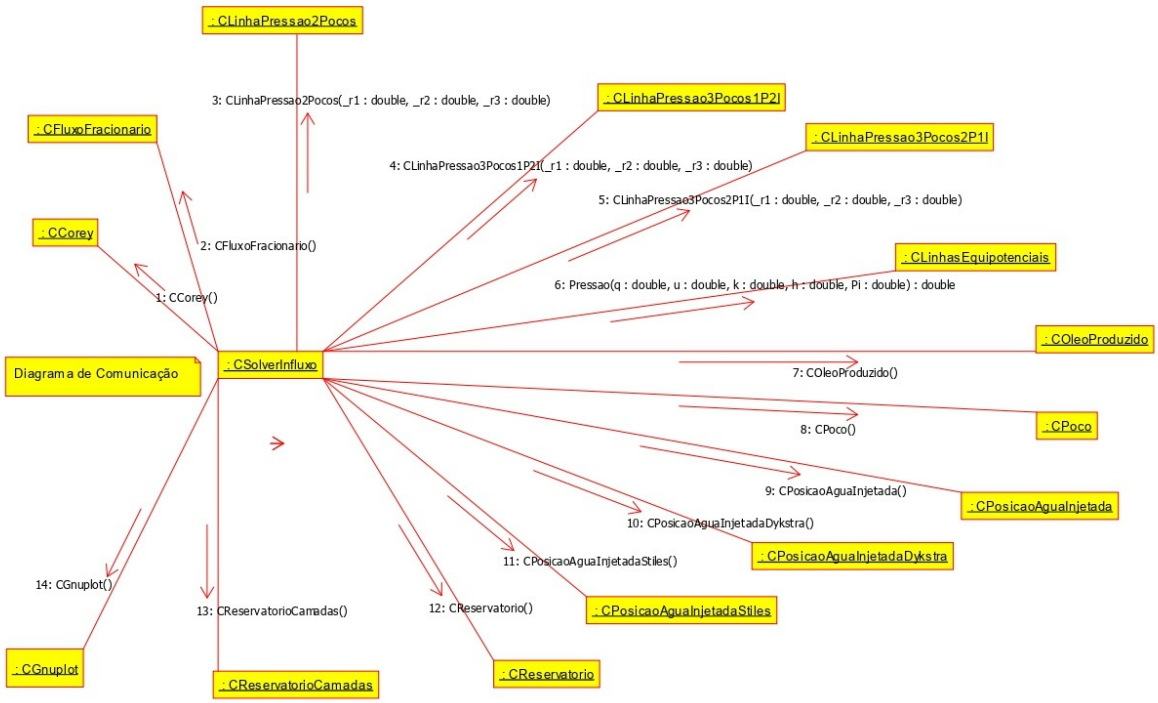


Figura 4.3: Diagrama de comunicação

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 na Figura 4.4.

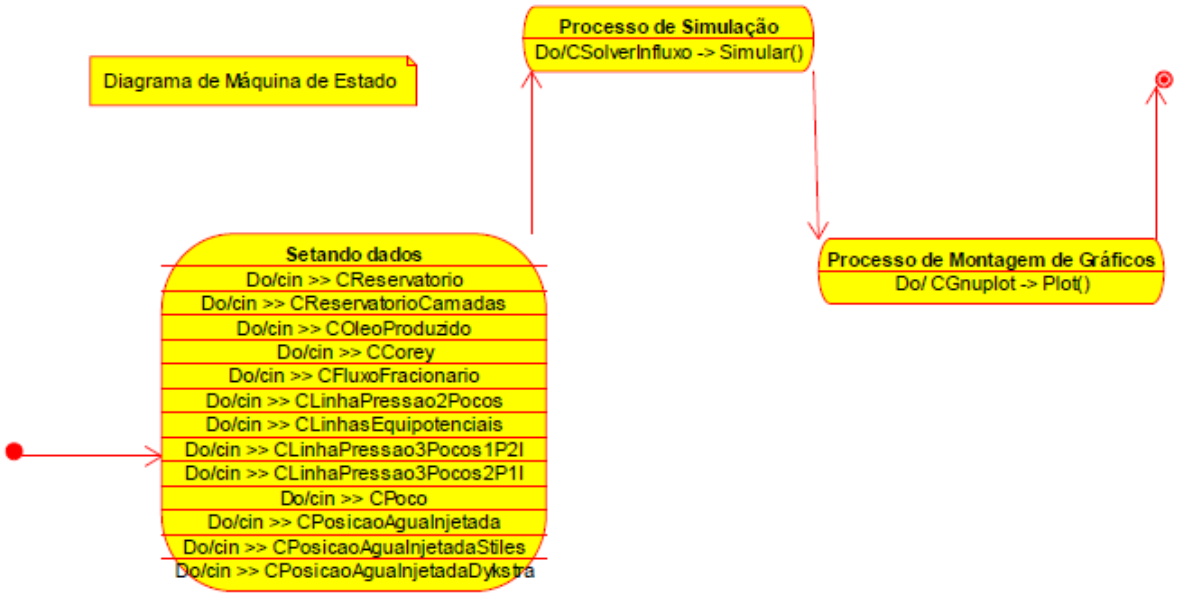


Figura 4.4: Diagrama de máquina de estado.

4.5 Diagrama de atividades

O diagrama de atividades (Figura 4.5) corresponde a uma atividade especíca do diagrama de máquina de estado, onde calcula-se o Fator de Recuperação Geral.

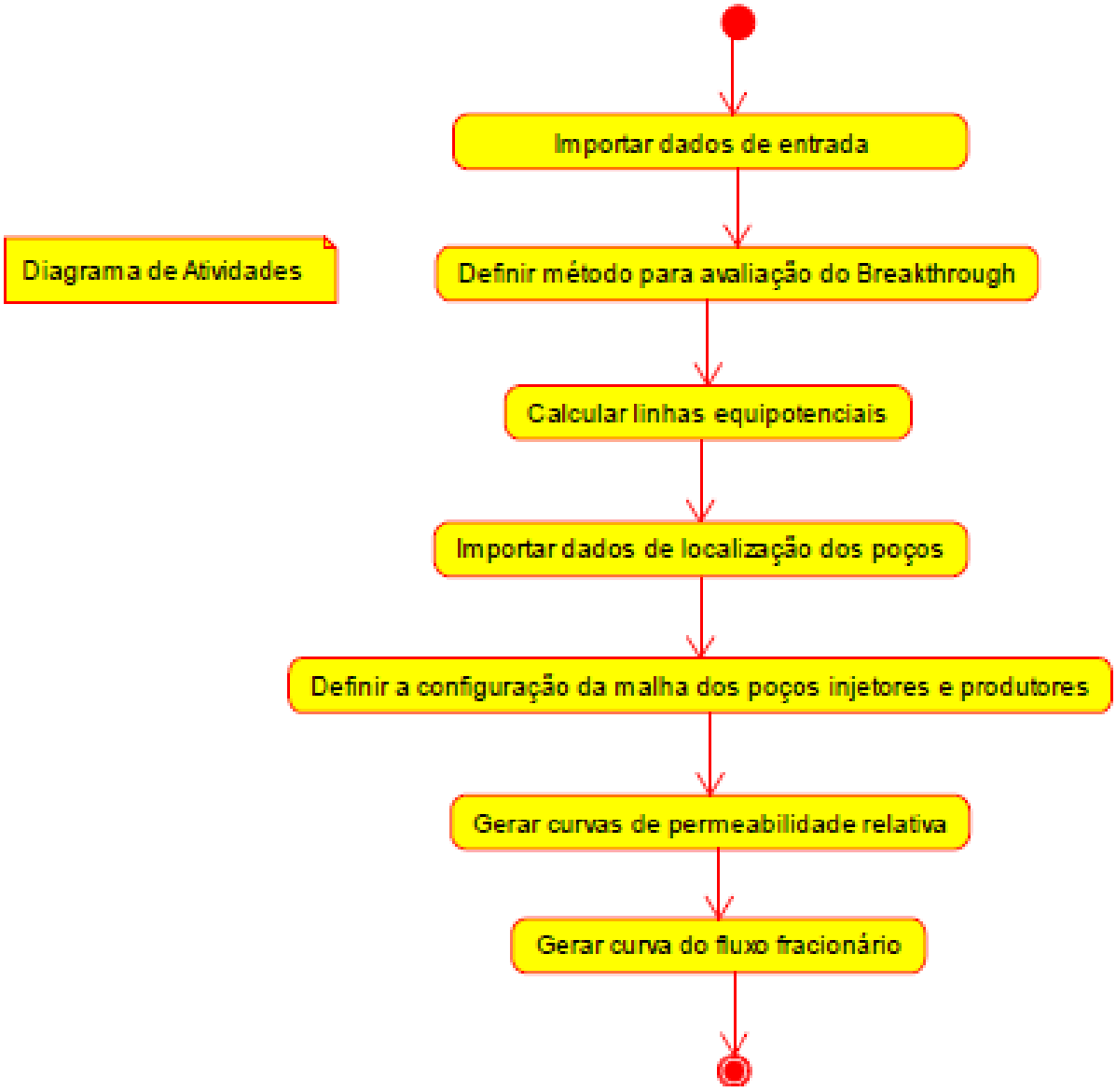


Figura 4.5: Diagrama de atividades

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 entra 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 softwate terá como saída de gráficos em arquivos de extensão .png.

Recursos

- O presente programa precisara 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 espaco 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 sera 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 ítens 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.

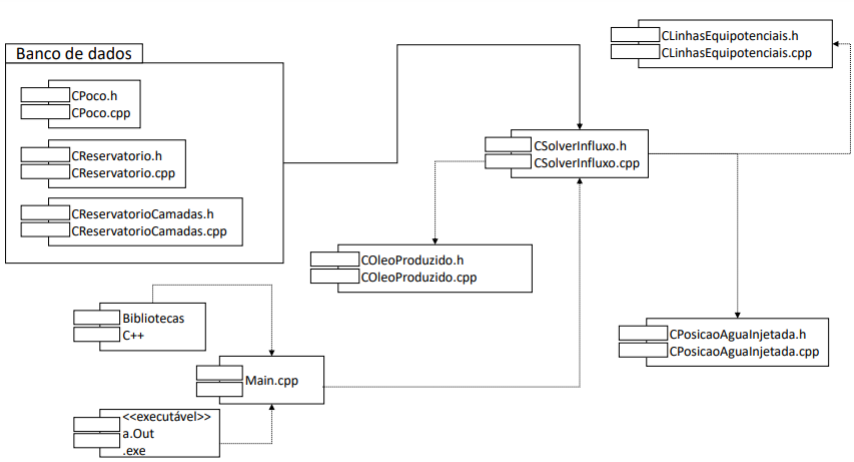


Figura 5.1: Diagrama de componentes

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*).

.

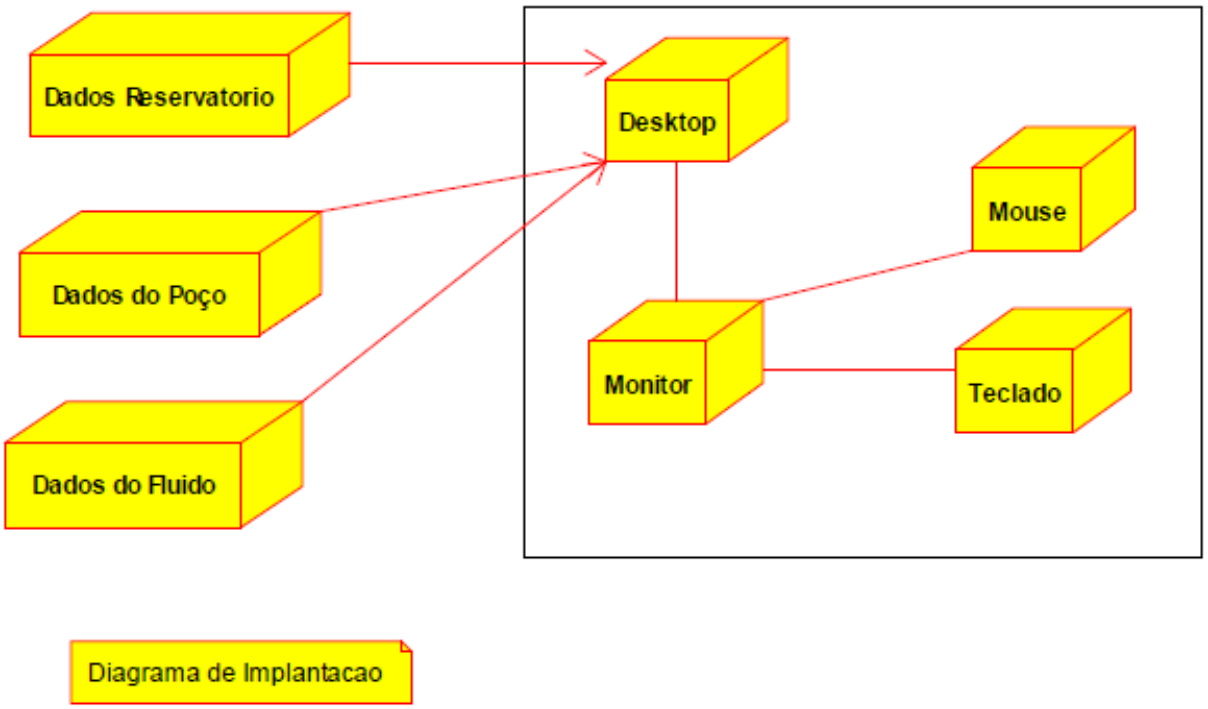


Figura 5.2: Diagrama de implantação

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 K0rw, K0ro, 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
5void 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
14void CCorey::calcKro(std::vector<double> _sw){
15
16       for(double sw:_sw)
17               kro.push_back(K0ro*pow(1 - sw, No));
18
19}
20
21void 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 CCorey::setK0rw(double _K0rw){
29
30         K0rw = _K0rw;
31
32 }
33
34 void CCorey::setK0ro(double _K0ro){
35
36         K0ro = _K0ro;
37
38 }
39
40 void CCorey::setNo(double _No){
41
42         No = _No;
43
44 }
45
46 void CCorey::setNw(double _Nw){
47
48         Nw = _Nw;
49
50 }
51
52 std::vector<double> CCorey::getSw(){
53
54         return sw;
55
56 }
57
58 std::vector<double> CCorey::getKro(){
59
60         return kro;
61
62 }
63
64 std::vector<double> CCorey::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.

```
#ifndef CFLUXOFRACIONARIO_H_
#define CFLUXOFRACIONARIO_H_

#include <vector>

class CFluxoFracionario{

        protected:

                std::vector<double> Frw, Fro;

        public:

                CFluxoFracionario(){};

                void calcFluxoFracionarioAgua(std::vector<double>
                   _krw, std::vector<double> _kro, double miw,
                   double mio);
                void calcFluxoFracionarioOleo(std::vector<double>
                   _krw, std::vector<double> _kro, double miw,
                   double mio);
                std::vector<double> getFluxoFracionarioAgua();
                std::vector<double> getFluxoFracionarioOleo();

                ~CFluxoFracionario(){};

};


#endif
```

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

Listing 6.4: Arquivo de implementação da classe CFluxoFracionario.

```
#include "CFluxoFracionario.h"

void CFluxoFracionario::calcFluxoFracionarioAgua(std::vector<double
    > _krw, std::vector<double> _kro, double miw, double mio){

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

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

Listing 6.5: Arquivo de implementação da classe CGnuplot.

```
1//
    ////////////////////////////////////////////////////////////////////////
2//                Classe de Interface em C++ para o programa gnuplot
    .                  //
3//
    ////////////////////////////////////////////////////////////////////////
4// Esta interface usa pipes e nao ira funcionar em sistemas que nao
     suportam
```

```
5// o padrao POSIX pipe.
6// O mesmo foi testado em sistemas Windows (MinGW e Visual C++) e
    Linux(GCC/G++)
7// Este programa foi originalmente escrito por:
8// Historico de versoes:
9// 0. Interface para linguagem C
10//    por N. Devillard (27/01/03)
11// 1. Interface para C++: tradução direta da versao em C
12//    por Rajarshi Guha (07/03/03)
13// 2. Correcoes para compatibilidadde com Win32
14//    por V. Chyzhdzenka (20/05/03)
15// 3. Novos métodos membros, correcoes para compatibilidade com
    Win32 e Linux
16//    por M. Burgis (10/03/08)
17// 4. Traducao para Portugues, documentacao - javadoc/doxygen,
18//    e modificacoes na interface (adicao de interface alternativa)
19//    por Bueno.A.D. (30/07/08)
20//   Tarefas:
21//   (v1)
22//   Documentar toda classe
23//   Adicionar novos métodos, criando atributos adicionais se
    necessario.
24//   Adotar padrao C++, isto e, usar sobrecarga nas chamadas.
25//   (v2)
26//   Criar classe herdeira CGnuplot, que inclui somente a nova
    interface.
27//   como e herdeira, o usuario vai poder usar nome antigos.
28//   Vantagem: preserva classe original, cria nova interface, fica
    a critério do usuário
29//   qual interface utilizar.
30//
    ////////////////////////////////////////////////////////////////////////

31// Requisitos:
32// - O programa gnuplot deve estar instalado (veja http://www.
    gnuplot.info/download.html)
33// - No Windows: setar a Path do Gnuplot (i.e. C:/program files/
    gnuplot/bin)
34//             ou setar a path usando: Gnuplot::set_GNUPlotPath(
    const std::string &path);
35//             Gnuplot::set_GNUPlotPath("C:/program files/gnuplot/
    bin");
```

```
36// - Para um melhor uso, consulte o manual do gnuplot,
37//   no GNU/Linux digite: man gnuplot ou info gnuplot.
38//
39// - Veja aula em http://www.lenep.uenf.br/~bueno/DisciplinaSL/
40//
41//
    ////////////////////////////////////////////////////////////////////////////////


42
43
44#ifndef CGnuplot_h
45#define CGnuplot_h
46#include <iostream>             // Para teste
47#include <string>
48#include <vector>
49#include <stdexcept>            // Heranca da classe std::
    runtime_error em GnuplotException
50#include <cstdio>               // Para acesso a arquivos FILE
51
52/**
53@brief Erros em tempo de execucao
54@class GnuplotException
55@file  GnuplotException.h
56*/
57class GnuplotException : public std::runtime_error
58{
59public:
60  /// Construtor
61  GnuplotException (const std::string & msg):std::runtime_error (
      msg) {}
62};
63
64/**
65@brief Classe de interface para acesso ao programa gnuplot.
66@class Gnuplot
67@file  gnuplot_i.hpp
68*/
69class Gnuplot
70{
71private:
72  //
      ----------------------------------------------------------------------
```

```
73      Atributos
73  FILE *  gnucmd;       ///< Ponteiro para stream que escreve no
        pipe.
74  bool    valid;        ///< Flag que indica se a sessao do gnuplot
        esta valida.
75  bool    two_dim;      ///< true = verdadeiro = 2d, false = falso
        = 3d.
76  int     nplots;       ///< Numero de graficos (plots) na sessao.
77  std::string pstyle;   ///< Estilo utilizado para visualizacao das
        funcoes e dados.
78  std::string smooth;   ///< interpolate and approximate data in
        defined styles (e.g. spline).
79  std::vector <std::string> tmpfile_list; ///< Lista com nome dos
        arquivos temporarios.
80
81  //
        ------------------------------------------------------------------
        flags
82  bool fgrid;           ///< 0 sem grid,        1 com grid
83  bool fhidden3d;       ///< 0 nao oculta,      1 oculta
84  bool fcontour;        ///< 0 sem contorno,    1 com contorno
85  bool fsurface;        ///< 0 sem superficie,  1 com superficie
86  bool flegend;         ///< 0 sem legendad,    1 com legenda
87  bool ftitle;          ///< 0 sem titulo,      1 com titulo
88  bool fxlogscale;      ///< 0 desativa escala log, 1 ativa escala
        log
89  bool fylogscale;      ///< 0 desativa escala log, 1 ativa escala
        log
90  bool fzlogscale;      ///< 0 desativa escala log, 1 ativa escala
        log
91  bool fsmooth;         ///< 0 desativa,        1 ativa
92
93  //
        ------------------------------------------------------------------

94  // Atributos estaticos (compartilhados por todos os objetos)
95  static int tmpfile_num;              ///< Numero total de
        arquivos temporarios (numero restrito).
96  static std::string m_sGNUPlotFileName;///< Nome do arquivo
        executavel do gnuplot.
97  static std::string m_sGNUPlotPath;    ///< Caminho para
        executavel do gnuplot.
```

```
98  static std::string terminal_std;      ///< Terminal padrao (
       standart), usado para visualizacoes.
99
100 //
       ------------------------------------------------------------------------
        Metodos
101 // Funcoes membro (métodos membro) (funcoes auxiliares)
102 /// @brief Cria arquivo temporario e retorna seu nome.
103 /// Usa get_program_path(); e popen();
104 void  init ();
105
106 /// @brief Cria arquivo temporario e retorna seu nome.
107 /// Usa get_program_path(); e popen();
108 void Init() { init(); }
109
110 /// @brief Cria arquivo temporario.
111 std::string  create_tmpfile (std::ofstream & tmp);
112
113 /// @brief Cria arquivo temporario.
114 std::string  CreateTmpFile (std::ofstream & tmp) { return
       create_tmpfile(tmp); }
115
116 //
       ------------------------------------------------------------------------

117 // Funcoes estaticas (static functions)
118 /// @brief Retorna verdadeiro se a path esta presente.
119 static bool  get_program_path ();
120
121 /// @brief Retorna verdadeiro se a path esta presente.
122 static bool  Path() { return get_program_path(); }
123
124 /// @brief Checa se o arquivo existe.
125 static bool  file_exists (const std::string & filename, int mode
       = 0);
126
127 /// @brief Checa se o arquivo existe.
128 static bool  FileExists  (const std::string & filename, int mode
       = 0)
129                           { return file_exists( filename, mode );
                              }
130
```

```
131   //
          ------------------------------------------------------------------------

132 public:
133   // Opcional: Seta path do gnuplot manualmente
134   // No windows: a path (caminho) deve ser dada usando '/' e nao
          backslash '\'
135   /// @brief Seta caminho para path do gnuplot.
136   //ex: CGnuplot::set_GNUPlotPath ("\"C:/program files/gnuplot/bin
          /\"");
137
138   static bool set_GNUPlotPath (const std::string & path);
139
140   /// @brief Seta caminho para path do gnuplot.
141   static bool Path(const std::string & path) { return
          set_GNUPlotPath(path); }
142 //
143   /// @brief Opcional: Seta terminal padrao (standart), usado para
          visualizacao dos graficos.
144   /// Valores padroes (default): Windows - win, Linux - x11, Mac -
          aqua
145   static void set_terminal_std (const std::string & type);
146
147   /// @brief Opcional: Seta terminal padrao (standart), usado para
          visualizacao dos graficos.
148   /// Para retornar para terminal janela precisa chamar
          ShowOnScreen().
149   /// Valores padroes (default): Windows - win, Linux - x11 ou wxt
          (fedora9), Mac - aqua
150   static void Terminal (const std::string & type) {
          set_terminal_std(type); }
151
152  //
         ------------------------------------------------------------------------
        Construtores
153   /// @brief Construtor, seta o estilo do grafico na construcao.
154   Gnuplot (const std::string & style = "points");
155
156   /// @brief Construtor, plota um grafico a partir de um vector,
          diretamente na construcao.
157   Gnuplot (const std::vector < double >&x,
158            const std::string & title = "",
```

```
159             const std::string & style = "points",
160             const std::string & labelx = "x",
161             const std::string & labely = "y");
162
163     /// @brief Construtor, plota um grafico do tipo x_y a partir de
        vetores, diretamente na construcao.
164     Gnuplot (const std::vector < double >&x,
165             const std::vector < double >&y,
166             const std::string & title = "",
167             const std::string & style = "points",
168             const std::string & labelx = "x",
169             const std::string & labely = "y");
170
171     /// @brief Construtor, plota um grafico de x_y_z a partir de
        vetores, diretamente na construcao.
172     Gnuplot (const std::vector < double >&x,
173             const std::vector < double >&y,
174             const std::vector < double >&z,
175             const std::string & title = "",
176             const std::string & style = "points",
177             const std::string & labelx = "x",
178             const std::string & labely = "y",
179             const std::string & labelz = "z");
180
181     /// @brief Destrutor, necessario para deletar arquivos
        temporarios.
182      ~Gnuplot ();
183
184     //
        ----------------------------------------------------------------------

185     /// @brief Envia comando para o gnuplot.
186     Gnuplot &  cmd (const std::string & cmdstr);
187
188     /// @brief Envia comando para o gnuplot.
189     Gnuplot &  Cmd (const std::string & cmdstr)     { return cmd(
        cmdstr); }
190
191     /// @brief Envia comando para o gnuplot.
192     Gnuplot &  Command (const std::string & cmdstr) { return cmd(
        cmdstr); }
193
```

```
194 /// @brief Sobrecarga operador <<, funciona como Comando.
195 Gnuplot &  operator<< (const std::string & cmdstr);
196
197 //
    ------------------------------------------------------------------------

198 /// @brief Mostrar na tela ou escrever no arquivo, seta o tipo de
    terminal para terminal_std.
199 Gnuplot &  showonscreen ();           // Janela de saida e setada
    como default (win/x11/aqua)
200
201 /// @brief Mostrar na tela ou escrever no arquivo, seta o tipo de
    terminal para terminal_std.
202 Gnuplot &  ShowOnScreen ()                        { return
    showonscreen();  };
203
204 /// @brief Salva sessao do gnuplot para um arquivo postscript,
    informe o nome do arquivo sem extensao.
205 /// Depois retorna para modo terminal
206 Gnuplot &  savetops (const std::string & filename = "
    gnuplot_output");
207
208 /// @brief Salva sessao do gnuplot para um arquivo postscript,
    informe o nome do arquivo sem extensao
209 /// Depois retorna para modo terminal
210 Gnuplot &  SaveTops (const std::string & filename = "
    gnuplot_output")
211                                                  { return savetops
                                                     (filename); }
212
213 /// @brief Salva sessao do gnuplot para um arquivo png, nome do
    arquivo sem extensao
214 /// Depois retorna para modo terminal
215 Gnuplot &  savetopng (const std::string & filename = "
    gnuplot_output");
216
217 /// @brief Salva sessao do gnuplot para um arquivo png, nome do
    arquivo sem extensao
218 /// Depois retorna para modo terminal
219 Gnuplot &  SaveTopng (const std::string & filename = "
    gnuplot_output")
220                                                  { return
```

```
                                                    savetopng(
                                                    filename); }
221
222 /// @brief Salva sessao do gnuplot para um arquivo jpg, nome do
       arquivo sem extensao
223 /// Depois retorna para modo terminal
224 Gnuplot &  savetojpeg (const std::string & filename = "
       gnuplot_output");
225
226 /// @brief Salva sessao do gnuplot para um arquivo jpg, nome do
       arquivo sem extensao
227 /// Depois retorna para modo terminal
228 Gnuplot &  SaveTojpeg (const std::string & filename = "
       gnuplot_output")
229                                                     { return
                                                       savetojpeg(
                                                       filename); }
230
231 /// @brief Salva sessao do gnuplot para um arquivo filename,
       usando o terminal_type e algum flag adicional
232 /// Ex:
233 /// grafico.SaveTo("pressao_X_temperatura","png", "enhanced size
       1280,960");
234 /// Para melhor uso dos flags adicionais consulte o manual do
       gnuplot (help term)
235 Gnuplot& SaveTo(const std::string &filename,const std::string &
       terminal_type, std::string flags="");
236
237 //
       ------------------------------------------------------------------------
        set e unset
238 /// @brief  Seta estilos de linhas (em alguns casos sao
       necessarias informacoes adicionais).
239 /// lines, points, linespoints, impulses, dots, steps, fsteps,
       histeps,
240 /// boxes, histograms, filledcurves
241 Gnuplot &  set_style (const std::string & stylestr = "points");
242
243 /// @brief  Seta estilos de linhas (em alguns casos sao
       necessarias informacoes adicionais).
244 /// lines, points, linespoints, impulses, dots, steps, fsteps,
       histeps,
```

```
245 /// boxes, histograms, filledcurves
246 Gnuplot &  Style (const std::string & stylestr = "points")
247                                                 { return set_style
                                                       (stylestr); }
248
249 /// @brief  Ativa suavizacao.
250 /// Argumentos para interpolacoes e aproximacoes.
251 /// csplines, bezier, acsplines (para dados com valor > 0),
       sbezier, unique,
252 /// frequency (funciona somente com plot_x, plot_xy, plotfile_x,
253 /// plotfile_xy (se a suavizacao esta ativa, set_style nao tem
       efeito na plotagem dos graficos)
254 Gnuplot &  set_smooth (const std::string & stylestr = "csplines")
       ;
255
256 /// @brief  Desativa suavizacao.
257 Gnuplot &  unset_smooth ();           // A suavizacao nao e
       setada por padrao (default)
258
259 /// @brief  Ativa suavizacao.
260 /// Argumentos para interpolacoes e aproximacoes.
261 /// csplines, bezier, acsplines (para dados com valor > 0),
       sbezier, unique,
262 /// frequency (funciona somente com plot_x, plot_xy, plotfile_x,
263 /// plotfile_xy (se a suavizacao esta ativa, set_style nao tem
       efeito na plotagem dos graficos)
264 Gnuplot &  Smooth(const std::string & stylestr = "csplines")
265                                                 { return set_smooth
                                                       (stylestr); }
266
267 Gnuplot &  Smooth( int _fsmooth )
268                                                 { if( fsmooth =
                                                       _fsmooth )
269                                                       return
                                                           set_contour
                                                           ();
270                                                   else
271                                                       return
                                                           unset_contour
                                                           ();
272                                                 }
273 /// @brief  Desativa suavizacao.
```

```
274 //Gnuplot &  UnsetSmooth()                    { return
       unset_smooth (); }
275
276 /// @brief Escala o tamanho do ponto usado na plotagem.
277 Gnuplot &  set_pointsize (const double pointsize = 1.0);
278
279 /// @brief Escala o tamanho do ponto usado na plotagem.
280 Gnuplot &  PointSize (const double pointsize = 1.0)
281                                                 { return
                                                      set_pointsize(
                                                      pointsize); }
282
283 /// @brief Ativa o grid (padrao = desativado).
284 Gnuplot &  set_grid ();
285
286 /// @brief Desativa o grid (padrao = desativado).
287 Gnuplot &  unset_grid ();
288
289 /// @brief Ativa/Desativa o grid (padrao = desativado).
290 Gnuplot &  Grid(bool _fgrid = 1)
291                                                 { if(fgrid = _fgrid
                                                      )
292                                                       return set_grid
                                                          ();
293                                                   else
294                                                       return
                                                          unset_grid()
                                                          ; }
295
296 /// @brief Seta taxa de amostragem das funcoes, ou dos dados de
       interpolacao.
297 Gnuplot &  set_samples (const int samples = 100);
298
299 /// @brief Seta taxa de amostragem das funcoes, ou dos dados de
       interpolacao.
300 Gnuplot &  Samples(const int samples = 100)   { return
       set_samples(samples); }
301
302 /// @brief Seta densidade de isolinhas para plotagem de funcoes
       como superficies (para plotagen 3d).
303 Gnuplot &  set_isosamples (const int isolines = 10);
304
```

```
305 /// @brief Seta densidade de isolinhas para plotagem de funcoes como superficies (para plotagen 3d).
306 Gnuplot &  IsoSamples (const int isolines = 10){ return set_isosamples(isolines); }
307
308 /// @brief Ativa remocao de linhas ocultas na plotagem de superficies (para plotagen 3d).
309 Gnuplot &  set_hidden3d ();
310
311 /// @brief Desativa remocao de linhas ocultas na plotagem de superficies (para plotagen 3d).
312 Gnuplot &  unset_hidden3d ();          // hidden3d nao e setado por padrao (default)
313
314 /// @brief Ativa/Desativa remocao de linhas ocultas na plotagem de superficies (para plotagen 3d).
315 Gnuplot &  Hidden3d(bool _fhidden3d = 1)
316                                                { if(fhidden3d = _fhidden3d)
317                                                    return set_hidden3d ();
318                                                  else
319                                                    return unset_hidden3d ();
320                                                }
321
322 /// @brief Ativa desenho do contorno em superficies (para plotagen 3d).
323 /// @param base, surface, both.
324 Gnuplot &  set_contour (const std::string & position = "base");
325
326 /// @brief Desativa desenho do contorno em superficies (para plotagen 3d).
327 Gnuplot &  unset_contour ();           // contour nao e setado por default
328
329 /// @brief Ativa/Desativa desenho do contorno em superficies (para plotagen 3d).
330 /// @param base, surface, both.
331 Gnuplot &  Contour(const std::string & position = "base")
```

```
332                                                 { return
                                                      set_contour(
                                                      position); }
333
334   Gnuplot &  Contour( int _fcontour )
335                                                 { if( fcontour =
                                                      _fcontour )
336                                                       return
                                                            set_contour
                                                            ();
337                                                   else
338                                                       return
                                                            unset_contour
                                                            ();
339                                                 }
340   /// @brief  Ativa a visualizacao da superficie (para plotagen 3d)
          .
341   Gnuplot &  set_surface ();             // surface e setado por
          padrao (default)
342
343   /// @brief  Desativa a visualizacao da superficie (para plotagen
          3d).
344   Gnuplot &  unset_surface ();
345
346   /// @brief  Ativa/Desativa a visualizacao da superficie (para
          plotagen 3d).
347   Gnuplot &  Surface( int _fsurface = 1 )
348                                                 { if(fsurface =
                                                      _fsurface)
349                                                       return
                                                            set_surface
                                                            ();
350                                                   else
351                                                       return
                                                            unset_surface
                                                            ();
352                                                 }
353   /// @brief Ativa a legenda (a legenda é setada por padrao).
354   /// Posicao: inside/outside, left/center/right, top/center/bottom
          , nobox/box
355   Gnuplot &  set_legend (const std::string & position = "default");
```

```
356
357 /// @brief Desativa a legenda (a legenda é setada por padrao).
358 Gnuplot &  unset_legend ();
359
360 /// @brief Ativa/Desativa a legenda (a legenda é setada por
       padrao).
361 Gnuplot &  Legend(const std::string & position = "default")
362                                                 { return set_legend
                                                      (position); }
363
364 /// @brief Ativa/Desativa a legenda (a legenda é setada por
       padrao).
365 Gnuplot &  Legend(int _flegend)
366                                                 { if(flegend =
                                                      _flegend)
367                                                     return
                                                         set_legend
                                                         ();
368                                                   else
369                                                     return
                                                         unset_legend
                                                         ();
370                                                 }
371
372 /// @brief Ativa o titulo da secao do gnuplot.
373 Gnuplot &  set_title (const std::string & title = "");
374
375 /// @brief Desativa o titulo da secao do gnuplot.
376 Gnuplot &  unset_title ();            // O title nao e setado por
       padrao (default)
377
378 /// @brief Ativa/Desativa o titulo da secao do gnuplot.
379 Gnuplot &  Title(const std::string & title = "")
380                                                 {
381                                                  return set_title(
                                                      title);
382                                                 }
383 Gnuplot &  Title(int _ftitle)
384                                                 {
385                                                  if(ftitle =
                                                      _ftitle)
386                                                    return set_title
```

```
                                                    ();
387                                               else
388                                                 return
                                                       unset_title()
                                                       ;
389                                               }
390
391   /// @brief Seta o rotulo (nome) do eixo y.
392   Gnuplot &  set_ylabel (const std::string & label = "y");
393
394   /// @brief Seta o rotulo (nome) do eixo y.
395   /// Ex: set ylabel "{/Symbol s}[MPa]" font "Times Italic, 10"
396   Gnuplot &  YLabel(const std::string & label = "y")
397                                                   { return set_ylabel
                                                       (label); }
398
399   /// @brief Seta o rotulo (nome) do eixo x.
400   Gnuplot &  set_xlabel (const std::string & label = "x");
401
402   /// @brief Seta o rotulo (nome) do eixo x.
403   Gnuplot &  XLabel(const std::string & label = "x")
404                                                   { return set_xlabel
                                                       (label); }
405
406   /// @brief Seta o rotulo (nome) do eixo z.
407   Gnuplot &  set_zlabel (const std::string & label = "z");
408
409   /// @brief Seta o rotulo (nome) do eixo z.
410   Gnuplot &  ZLabel(const std::string & label = "z")
411                                                   { return set_zlabel
                                                       (label); }
412
413   /// @brief Seta intervalo do eixo x.
414   Gnuplot &  set_xrange (const int iFrom, const int iTo);
415
416   /// @brief Seta intervalo do eixo x.
417   Gnuplot &  XRange (const int iFrom, const int iTo)
418                                                   { return set_xrange
                                                       (iFrom,iTo); }
419
420   /// @brief Seta intervalo do eixo y.
421   Gnuplot &  set_yrange (const int iFrom, const int iTo);
```

```
422
423  /// @brief Seta intervalo do eixo y.
424  Gnuplot &  YRange (const int iFrom, const int iTo)
425                                                  { return set_yrange
                                                       (iFrom,iTo); }
426
427  /// @brief Seta intervalo do eixo z.
428  Gnuplot &  set_zrange (const int iFrom, const int iTo);
429
430  /// @brief Seta intervalo do eixo z.
431  Gnuplot &  ZRange (const int iFrom, const int iTo)
432                                                  { return set_zrange
                                                       (iFrom,iTo); }
433
434  /// @brief Seta escalonamento automatico do eixo x (default).
435  Gnuplot &  set_xautoscale ();
436
437  /// @brief Seta escalonamento automatico do eixo x (default).
438  Gnuplot &  XAutoscale()                          { return
       set_xautoscale (); }
439
440  /// @brief Seta escalonamento automatico do eixo y (default).
441  Gnuplot &  set_yautoscale ();
442
443  /// @brief Seta escalonamento automatico do eixo y (default).
444  Gnuplot &  YAutoscale()                          { return
       set_yautoscale (); }
445
446  /// @brief Seta escalonamento automatico do eixo z (default).
447  Gnuplot &  set_zautoscale ();
448
449  /// @brief Seta escalonamento automatico do eixo z (default).
450  Gnuplot &  ZAutoscale()                          { return
       set_zautoscale (); }
451
452  /// @brief Ativa escala logaritma do eixo x (logscale nao e
       setado por default).
453  Gnuplot &  set_xlogscale (const double base = 10);
454
455  /// @brief Desativa escala logaritma do eixo x (logscale nao e
       setado por default).
456  Gnuplot &  unset_xlogscale ();
```

```
457
458 /// @brief Ativa escala logaritma do eixo x (logscale nao e
       setado por default).
459 Gnuplot &  XLogscale (const double base = 10) { //if(base)
460                                                     return
                                                        set_xlogscale
                                                         (base);
461                                                     //else
462                                                     //return
                                                        unset_xlogscale
                                                         ();
463                                                 }
464
465 /// @brief Ativa/Desativa escala logaritma do eixo x (logscale
       nao e setado por default).
466 Gnuplot &  XLogscale(bool _fxlogscale)
467                                                 { if(fxlogscale =
                                                     _fxlogscale)
468                                                     return
                                                        set_xlogscale
                                                        ();
469                                                   else
470                                                     return
                                                        unset_xlogscale
                                                        ();
471                                                 }
472
473 /// @brief Ativa escala logaritma do eixo y (logscale nao e
       setado por default).
474 Gnuplot &  set_ylogscale (const double base = 10);
475
476 /// @brief Ativa escala logaritma do eixo y (logscale nao e
       setado por default).
477 Gnuplot &  YLogscale (const double base = 10) { return
       set_ylogscale (base); }
478
479 /// @brief Desativa escala logaritma do eixo y (logscale nao e
       setado por default).
480 Gnuplot &  unset_ylogscale ();
481
482 /// @brief Ativa/Desativa escala logaritma do eixo y (logscale
       nao e setado por default).
```

```
483 Gnuplot &  YLogscale(bool _fylogscale)
484                                                 { if(fylogscale =
                                                       _fylogscale)
485                                                     return
                                                        set_ylogscale
                                                        ();
486                                                   else
487                                                     return
                                                        unset_ylogscale
                                                        ();
488                                                 }
489
490 /// @brief Ativa escala logaritma do eixo y (logscale nao e
       setado por default).
491 Gnuplot &  set_zlogscale (const double base = 10);
492
493 /// @brief Ativa escala logaritma do eixo y (logscale nao e
       setado por default).
494 Gnuplot &  ZLogscale (const double base = 10) { return
       set_zlogscale (base); }
495
496 /// @brief Desativa escala logaritma do eixo z (logscale nao e
       setado por default).
497 Gnuplot &  unset_zlogscale ();
498
499 /// @brief Ativa/Desativa escala logaritma do eixo y (logscale
       nao e setado por default).
500 Gnuplot &  ZLogscale(bool _fzlogscale)
501                                                 { if(fzlogscale =
                                                       _fzlogscale)
502                                                     return
                                                        set_zlogscale
                                                        ();
503                                                   else
504                                                     return
                                                        unset_zlogscale
                                                        ();
505                                                 }
506
507
508 /// @brief Seta intervalo da palette (autoscale por padrao).
509 Gnuplot &  set_cbrange (const int iFrom, const int iTo);
```

```
510
511 /// @brief Seta intervalo da palette (autoscale por padrao).
512 Gnuplot &  CBRange(const int iFrom, const int iTo)
513                                                 { return
                                                      set_cbrange(
                                                      iFrom, iTo); }
514
515 //
    ---------------------------------------------------------------------------

516 /// @brief Plota dados de um arquivo de disco.
517 Gnuplot &  plotfile_x (const std::string & filename,
518              const int column = 1, const std::string & title = "")
                    ;
519
520 /// @brief Plota dados de um arquivo de disco.
521 Gnuplot &  PlotFile (const std::string & filename,
522              const int column = 1, const std::string & title = "")
523                                                 { return plotfile_x
                                                      (filename,
                                                      column, title);
                                                         }
524
525 /// @brief Plota dados de um vector.
526 Gnuplot &  plot_x (const std::vector < double >&x, const std::
    string & title = "");
527
528 /// @brief Plota dados de um vector.
529 Gnuplot &  PlotVector (const std::vector < double >&x, const std
    ::string & title = "")
530                                                 { return plot_x( x,
                                                       title ); }
531
532 /// @brief Plota pares x,y a partir de um arquivo de disco.
533 Gnuplot &  plotfile_xy (const std::string & filename,
534               const int column_x = 1,
535               const int column_y = 2, const std::string & title =
                    "");
536 /// @brief Plota pares x,y a partir de um arquivo de disco.
537 Gnuplot &  PlotFile (const std::string & filename,
538               const int column_x = 1,
539               const int column_y = 2, const std::string & title =
```

```
                "")
540                                                 {
541                                                 return plotfile_xy(
                                                        filename,
                                                        column_x,
                                                        column_y, title
                                                        );
542                                                 }
543
544 /// @brief Plota pares x,y a partir de vetores.
545 Gnuplot &  plot_xy (const std::vector < double >&x,
546           const std::vector < double >&y, const std::string &
                title = "");
547
548 /// @brief Plota pares x,y a partir de vetores.
549 Gnuplot &  PlotVector (const std::vector < double >&x,
550           const std::vector < double >&y, const std::string &
                title = "")
551                                                 { return plot_xy (
                                                        x, y,title ); }
552
553 /// @brief Plota pares x,y com barra de erro dy a partir de um
      arquivo.
554 Gnuplot &  plotfile_xy_err (const std::string & filename,
555                   const int column_x = 1,
556                   const int column_y = 2,
557                   const int column_dy = 3, const std::string &
                        title = "");
558
559 /// @brief Plota pares x,y com barra de erro dy a partir de um
      arquivo.
560 Gnuplot &  PlotFileXYErrorBar(const std::string & filename,
561                   const int column_x = 1,
562                   const int column_y = 2,
563                   const int column_dy = 3, const std::string &
                        title = "")
564                                                   { return
                                                      plotfile_xy_err
                                                      (filename,
565                                                     column_x,
                                                        column_y,
                                                        column_dy,
```

```
                                                        title ); }
566
567 /// @brief Plota pares x,y com barra de erro dy a partir de
        vetores.
568 Gnuplot &  plot_xy_err (const std::vector < double >&x,
569               const std::vector < double >&y,
570               const std::vector < double >&dy,
571               const std::string & title = "");
572
573 /// @brief Plota pares x,y com barra de erro dy a partir de
        vetores.
574 Gnuplot &  PlotVectorXYErrorBar(const std::vector < double >&x,
575               const std::vector < double >&y,
576               const std::vector < double >&dy,
577               const std::string & title = "")
578                                                    { return
                                                       plot_xy_err(
                                                       x, y, dy,
                                                       title); }
579
580 /// @brief  Plota valores de x,y,z a partir de um arquivo de
        disco.
581 Gnuplot &  plotfile_xyz (const std::string & filename,
582                  const int column_x = 1,
583                  const int column_y = 2,
584                  const int column_z = 3, const std::string & title =
                         "");
585 /// @brief  Plota valores de x,y,z a partir de um arquivo de
        disco.
586 Gnuplot &  PlotFile (const std::string & filename,
587                  const int column_x = 1,
588                  const int column_y = 2,
589                  const int column_z = 3, const std::string & title =
                         "")
590                                                    { return
                                                       plotfile_xyz
                                                       (filename,
                                                       column_x,
591                                                     column_y,
                                                        column_z);
                                                         }
592
```

```
593 /// @brief  Plota valores de x,y,z a partir de vetores.
594 Gnuplot &  plot_xyz (const std::vector < double >&x,
595           const std::vector < double >&y,
596           const std::vector < double >&z, const std::string &
              title = "");
597
598 /// @brief  Plota valores de x,y,z a partir de vetores.
599 Gnuplot &  PlotVector(const std::vector < double >&x,
600           const std::vector < double >&y,
601           const std::vector < double >&z, const std::string &
              title = "")
602                                                   { return
                                                      plot_xyz(x,
                                                      y, z, title)
                                                      ; }
603
604 /// @brief Plota uma equacao da forma y = ax + b, voce fornece os
      coeficientes a e b.
605 Gnuplot &  plot_slope (const double a, const double b, const std
    ::string & title = "");
606
607 /// @brief Plota uma equacao da forma y = ax + b, voce fornece os
      coeficientes a e b.
608 Gnuplot &  PlotSlope  (const double a, const double b, const std
    ::string & title = "")
609                                                   { return
                                                      plot_slope(a
                                                      ,b,title); }
610
611 /// @brief Plota uma  equacao fornecida como uma std::string y=f(
    x).
612 /// Escrever somente a funcao f(x) e nao y=
613 /// A variavel independente deve ser x
614 /// Os operadores binarios aceitos sao:
615 /// ** exponenciacao,
616 ///  * multiplicacao,
617 /// / divisao,
618 /// + adicao,
619 /// - subtracao,
620 /// % modulo
621 /// Os operadores unarios aceitos sao:
622 /// - menos,
```

```
623 /// ! fatorial
624 /// Funcoes elementares:
625 /// rand(x), abs(x), sgn(x), ceil(x), floor(x), int(x), imag(x),
       real(x), arg(x),
626 /// sqrt(x), exp(x), log(x), log10(x), sin(x), cos(x), tan(x),
       asin(x), acos(x),
627 /// atan(x), atan2(y,x), sinh(x), cosh(x), tanh(x), asinh(x),
       acosh(x), atanh(x)
628 /// Funcoes especiais:
629 /// erf(x), erfc(x), inverf(x), gamma(x), igamma(a,x), lgamma(x),
        ibeta(p,q,x),
630 /// besj0(x), besj1(x), besy0(x), besy1(x), lambertw(x)
631 /// Funcoes estatisticas:
632 /// norm(x), invnorm(x)
633 Gnuplot &  plot_equation (const std::string & equation,
634                 const std::string & title = "");
635
636 /// @brief Plota uma  equacao fornecida como uma std::string y=f(
       x).
637 /// Escrever somente a funcao f(x) e nao y=
638 /// A variavel independente deve ser x.
639 /// Exemplo: gnuplot->PlotEquation(CFuncao& obj);
640 // Deve receber um CFuncao, que tem cast para string.
641 Gnuplot &  PlotEquation(const std::string & equation,
642                 const std::string & title = "")
643                                           { return
                                              plot_equation(
                                              equation, title );
                                               }
644
645 /// @brief Plota uma equacao fornecida na forma de uma std::
       string z=f(x,y).
646 /// Escrever somente a funcao f(x,y) e nao z=, as variaveis
       independentes sao x e y.
647 Gnuplot &  plot_equation3d (const std::string & equation, const
       std::string & title = "");
648
649 /// @brief Plota uma equacao fornecida na forma de uma std::
       string z=f(x,y).
650 /// Escrever somente a funcao f(x,y) e nao z=, as vaiaveis
       independentes sao x e y.
651 // gnuplot->PlotEquation3d(CPolinomio());
```

```
652  Gnuplot &  PlotEquation3d (const std::string & equation,
653                   const std::string & title = "")
654                                                { return
                                                     plot_equation3d(
                                                     equation, title );
                                                      }
655
656  /// @brief Plota uma imagem.
657  Gnuplot &  plot_image (const unsigned char *ucPicBuf,
658               const int iWidth, const int iHeight, const std::
                    string & title = "");
659
660  /// @brief Plota uma imagem.
661  Gnuplot &  PlotImage (const unsigned char *ucPicBuf,
662                        const int iWidth, const int iHeight, const
                           std::string & title = "")
663                                                { return plot_image (
                                                     ucPicBuf, iWidth,
                                                     iHeight, title); }
664
665  //
       ------------------------------------------------------------------------

666  // Repete o ultimo comando de plotagem, seja plot (2D) ou splot
       (3D)
667  // Usado para visualizar plotagens, após mudar algumas opcoes de
       plotagem
668  // ou quando gerando o mesmo grafico para diferentes dispositivos
        (showonscreen, savetops)
669  Gnuplot &  replot ();
670
671  // Repete o ultimo comando de plotagem, seja plot (2D) ou splot
       (3D)
672  // Usado para visualizar plotagens, após mudar algumas opcoes de
       plotagem
673  // ou quando gerando o mesmo grafico para diferentes dispositivos
        (showonscreen, savetops)
674  Gnuplot &  Replot()                              { return replot();
       }
675
676  // Reseta uma sessao do gnuplot (próxima plotagem apaga
       definicoes previas)
```

```
677   Gnuplot &  reset_plot ();
678
679   // Reseta uma sessao do gnuplot (próxima plotagem apaga
         definicoes previas)
680   Gnuplot &  ResetPlot()                          { return reset_plot
         (); }
681
682   // Chama função reset do gnuplot
683   Gnuplot &  Reset()                        { this->cmd("reset");
         return *this; }
684
685   // Reseta uma sessao do gnuplot e seta todas as variaveis para o
         default
686   Gnuplot &  reset_all ();
687
688   // Reseta uma sessao do gnuplot e seta todas as variaveis para o
         default
689   Gnuplot &  ResetAll ()                          { return reset_all
         (); }
690
691   // Verifica se a sessao esta valida
692   bool  is_valid ();
693
694   // Verifica se a sessao esta valida
695   bool  IsValid ()                                { return is_valid
         (); };
696
697 };
698 typedef Gnuplot CGnuplot;
699 #endif
```

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

Listing 6.6: Arquivo de implementação da classe CGnuplot.

```
1 ////////////////////////////////////////////////////////////
2 //
3 // A C++ interface to gnuplot.
4 //
5 // This is a direct translation from the C interface
6 // written by Nicolas Devillard (ndevilla@free.fr) which
7 // is available from http://ndevilla.free.fr/gnuplot/.
8 //
9 // As in the C interface this uses pipes and so wont
```

```
10// run on a system that doesn't have POSIX pipe support
11//
12// Rajarshi Guha
13// e-mail: rguha@indiana.edu, rajarshi@presidency.com
14// http://cheminfo.informatics.indiana.edu/~rguha/code/cc++/
15//
16// 07/03/03
17//
18////////////////////////////////////////////////////////////
19//
20// A little correction for Win32 compatibility
21// and MS VC 6.0 done by V.Chyzhdzenka
22//
23// Notes:
24// 1. Added private method Gnuplot::init().
25// 2. Temporary file is created in the current
26//    folder but not in /tmp.
27// 3. Added #ifdef WIN32 e.t.c. where is needed.
28// 4. Added private member m_sGNUPlotFileName is
29//    a name of executed GNUPlot file.
30//
31// Viktor Chyzhdzenka
32// e-mail: chyzhdzenka@mail.ru
33//
34// 20/05/03
35//
36////////////////////////////////////////////////////////////
37//
38// corrections for Win32 and Linux compatibility
39//
40// some member functions added:
41//  set_GNUPlotPath, set_terminal_std,
42//  create_tmpfile, get_program_path, file_exists,
43//  operator<<, replot, reset_all, savetops, showonscreen,
44//  plotfile_*, plot_xy_err, plot_equation3d
45// set, unset: pointsize, grid, *logscale, *autoscale,
46//  smooth, title, legend, samples, isosamples,
47//  hidden3d, cbrange, contour
48//
49// Markus Burgis
50// e-mail: mail@burgis.info
51//
```

```
52 // 10/03/08
53 //
54 ////////////////////////////////////////////////////////////
55 //
56 // Modificacoes:
57 // Traducao para o portugues
58 // Adicao de novos nomes para os metodos(funcoes)
59 // Uso de documentacao no formato javadoc/doxygen
60 // Bueno A.D.
61 // e-mail: bueno@lenep.uenf.br
62 // 20/07/08
63 //
64 ////////////////////////////////////////////////////////////
65 #include <fstream>              // for std::ifstream
66 #include <sstream>              // for std::ostringstream
67 #include <list>                 // for std::list
68 #include <cstdio>               // for FILE, fputs(), fflush(),
    popen()
69 #include <cstdlib>              // for getenv()
70 #include "CGnuplot.h"
71
72 // Se estamos no windows // defined for 32 and 64-bit environments
73 #if defined(WIN32) || defined(_WIN32) || defined(__WIN32__) ||
    defined(__TOS_WIN__)
74  #include <io.h>                // for _access(), _mktemp()
75  #define GP_MAX_TMP_FILES  27   // 27 temporary files it's
     Microsoft restriction
76 // Se estamos no unix, GNU/Linux, Mac Os X
77 #elif defined(unix) || defined(__unix) || defined(__unix__) ||
    defined(__APPLE__) //all UNIX-like OSs (Linux, *BSD, MacOSX,
    Solaris, ...)
78  #include <unistd.h>            // for access(), mkstemp()
79  #define GP_MAX_TMP_FILES  64
80 #else
81  #error unsupported or unknown operating system
82 #endif
83
84 //
    ------------------------------------------------------------------------

85 //
86 // initialize static data
```

```
87 //
88 int Gnuplot::tmpfile_num = 0;
89
90 // Se estamos no windows
91 #if defined(WIN32) || defined(_WIN32) || defined(__WIN32__) ||
       defined(__TOS_WIN__)
92 std::string Gnuplot::m_sGNUPlotFileName = "gnuplot.exe";
93 std::string Gnuplot::m_sGNUPlotPath = "C:/gnuplot/bin/";
94 // Se estamos no unix, GNU/Linux, Mac Os X
95 #elif defined(unix) || defined(__unix) || defined(__unix__) ||
       defined(__APPLE__)
96 std::string Gnuplot::m_sGNUPlotFileName = "gnuplot";
97 std::string Gnuplot::m_sGNUPlotPath = "/usr/bin/";
98 #endif
99
100 // Se estamos no windows
101 #if defined(WIN32) || defined(_WIN32) || defined(__WIN32__) ||
       defined(__TOS_WIN__)
102 std::string Gnuplot::terminal_std = "windows";
103 // Se estamos no unix, GNU/Linux
104 #elif ( defined(unix) || defined(__unix) || defined(__unix__) ) &&
       !defined(__APPLE__)
105 std::string Gnuplot::terminal_std = "x11";
106 // Se estamos Mac Os X
107 #elif defined(__APPLE__)
108 std::string Gnuplot::terminal_std = "aqua";
109 #endif
110
111 //
       ----------------------------------------------------------------------------

112 //
113 // define static member function: set Gnuplot path manual
114 //   for windows: path with slash '/' not backslash '\'
115 //
116 bool Gnuplot::set_GNUPlotPath(const std::string &path)
117 {
118     std::string tmp = path + "/" + Gnuplot::m_sGNUPlotFileName;
119 #if defined(WIN32) || defined(_WIN32) || defined(__WIN32__) ||
       defined(__TOS_WIN__)
120     if ( Gnuplot::file_exists(tmp,0) ) // check existence
121 #elif defined(unix) || defined(__unix) || defined(__unix__) ||
```

```
    defined(__APPLE__)
122     if ( Gnuplot::file_exists(tmp,1) ) // check existence and
            execution permission
123 #endif
124     {
125         Gnuplot::m_sGNUPlotPath = path;
126         return true;
127     }
128     else
129     {
130         Gnuplot::m_sGNUPlotPath.clear();
131         return false;
132     }
133 }
134
135 //
    ------------------------------------------------------------------------

136 // define static member function: set standart terminal, used by
    showonscreen
137 //  defaults: Windows - win, Linux - x11, Mac - aqua
138 void Gnuplot::set_terminal_std(const std::string &type)
139 {
140 #if defined(unix) || defined(__unix) || defined(__unix__) ||
    defined(__APPLE__)
141     if (type.find("x11") != std::string::npos && getenv("DISPLAY")
        == NULL)
142     {
143         throw GnuplotException("Can't␣find␣DISPLAY␣variable");
144     }
145 #endif
146
147
148     Gnuplot::terminal_std = type;
149     return;
150 }
151
152
153 //
    ------------------------------------------------------------------------

154 // A string tokenizer taken from http://www.sunsite.ualberta.ca/
```

```
    Documentation/
155// /Gnu/libstdc++-2.90.8/html/21_strings/stringtok_std_h.txt
156template <typename Container>
157void stringtok (Container &container,
158                std::string const &in,
159                const char * const delimiters = "␣\t\n")
160{
161    const std::string::size_type len = in.length();
162          std::string::size_type i = 0;
163
164    while ( i < len )
165    {
166        // eat leading whitespace
167        i = in.find_first_not_of (delimiters, i);
168
169        if (i == std::string::npos)
170            return;   // nothing left but white space
171
172        // find the end of the token
173        std::string::size_type j = in.find_first_of (delimiters, i)
               ;
174
175        // push token
176        if (j == std::string::npos)
177        {
178            container.push_back (in.substr(i));
179            return;
180        }
181        else
182            container.push_back (in.substr(i, j-i));
183
184        // set up for next loop
185        i = j + 1;
186    }
187
188    return;
189}
190
191//
    ------------------------------------------------------------------------


192//
```

```
193 // constructor: set a style during construction
194 //
195 Gnuplot::Gnuplot(const std::string &style)
196 {
197     this->init();
198     this->set_style(style);
199 }
200
201 //
    ----------------------------------------------------------------------

202 // constructor: open a new session, plot a signal (x)
203 Gnuplot::Gnuplot(const std::vector<double> &x,
204                  const std::string &title,
205                  const std::string &style,
206                  const std::string &labelx,
207                  const std::string &labely)
208 {
209     this->init();
210
211     this->set_style(style);
212     this->set_xlabel(labelx);
213     this->set_ylabel(labely);
214
215     this->plot_x(x,title);
216 }
217
218 //
    ----------------------------------------------------------------------

219 // constructor: open a new session, plot a signal (x,y)
220 Gnuplot::Gnuplot(const std::vector<double> &x,
221                  const std::vector<double> &y,
222                  const std::string &title,
223                  const std::string &style,
224                  const std::string &labelx,
225                  const std::string &labely)
226 {
227     this->init();
228
229     this->set_style(style);
230     this->set_xlabel(labelx);
```

```
231     this->set_ylabel(labely);
232
233     this->plot_xy(x,y,title);
234 }
235
236 //
      ------------------------------------------------------------------------

237 // constructor: open a new session, plot a signal (x,y,z)
238 Gnuplot::Gnuplot(const std::vector<double> &x,
239                  const std::vector<double> &y,
240                  const std::vector<double> &z,
241                  const std::string &title,
242                  const std::string &style,
243                  const std::string &labelx,
244                  const std::string &labely,
245                  const std::string &labelz)
246 {
247     this->init();
248
249     this->set_style(style);
250     this->set_xlabel(labelx);
251     this->set_ylabel(labely);
252     this->set_zlabel(labelz);
253
254     this->plot_xyz(x,y,z,title);
255 }
256
257 //
      ------------------------------------------------------------------------

258 // Destructor: needed to delete temporary files
259 Gnuplot::~Gnuplot()
260 {
261     if ((this->tmpfile_list).size() > 0)
262     {
263         for (unsigned int i = 0; i < this->tmpfile_list.size(); i
              ++)
264             remove( this->tmpfile_list[i].c_str() );
265
266         Gnuplot::tmpfile_num -= this->tmpfile_list.size();
267     }
```

```
268
269     // A stream opened by popen() should be closed by pclose()
270 #if defined(WIN32) || defined(_WIN32) || defined(__WIN32__) ||
      defined(__TOS_WIN__)
271     if (_pclose(this->gnucmd) == -1)
272 #elif defined(unix) || defined(__unix) || defined(__unix__) ||
      defined(__APPLE__)
273     if (pclose(this->gnucmd) == -1)
274 #endif
275         true;//throw GnuplotException("Problem closing
              communication to gnuplot");
276 }
277
278 //
      ---------------------------------------------------------------------------

279 // Resets a gnuplot session (next plot will erase previous ones)
280 Gnuplot& Gnuplot::reset_plot()
281 {
282     if (this->tmpfile_list.size() > 0)
283     {
284         for (unsigned int i = 0; i < this->tmpfile_list.size(); i
              ++)
285             remove(this->tmpfile_list[i].c_str());
286
287         Gnuplot::tmpfile_num -= this->tmpfile_list.size();
288         this->tmpfile_list.clear();
289     }
290
291     this->nplots = 0;
292
293     return *this;
294 }
295
296 //
      ---------------------------------------------------------------------------

297 // resets a gnuplot session and sets all varibles to default
298 Gnuplot& Gnuplot::reset_all()
299 {
300     if (this->tmpfile_list.size() > 0)
301     {
```

```
302         for (unsigned int i = 0; i < this->tmpfile_list.size(); i
                ++)
303             remove(this->tmpfile_list[i].c_str());
304 
305         Gnuplot::tmpfile_num -= this->tmpfile_list.size();
306         this->tmpfile_list.clear();
307     }
308 
309     this->nplots = 0;
310     this->cmd("reset");
311     this->cmd("clear");
312     this->pstyle = "points";
313     this->smooth = "";
314     this->showonscreen();
315 
316     return *this;
317 }
318 
319 //
      ----------------------------------------------------------------------------

320 // Find out if valid is true
321 bool Gnuplot::is_valid()
322 {
323     return(this->valid);
324 }
325 
326 //
      ----------------------------------------------------------------------------

327 // replot repeats the last plot or splot command
328 Gnuplot& Gnuplot::replot()
329 {
330     if (this->nplots > 0)
331     {
332         this->cmd("replot");
333     }
334 
335     return *this;
336 }
337 
338 
```

```
339 //
    ------------------------------------------------------------------------

340 // Change the plotting style of a gnuplot session
341 Gnuplot& Gnuplot::set_style(const std::string &stylestr)
342 {
343     if (stylestr.find("lines")          == std::string::npos  &&
344         stylestr.find("points")         == std::string::npos  &&
345         stylestr.find("linespoints")    == std::string::npos  &&
346         stylestr.find("impulses")       == std::string::npos  &&
347         stylestr.find("dots")           == std::string::npos  &&
348         stylestr.find("steps")          == std::string::npos  &&
349         stylestr.find("fsteps")         == std::string::npos  &&
350         stylestr.find("histeps")        == std::string::npos  &&
351         stylestr.find("boxes")          == std::string::npos  &&
              // 1-4 columns of data are required
352         stylestr.find("filledcurves")   == std::string::npos  &&
353         stylestr.find("histograms")     == std::string::npos  )
             //only for one data column
354 //        stylestr.find("labels")         == std::string::npos  &&
      // 3 columns of data are required
355 //        stylestr.find("xerrorbars")     == std::string::npos  &&
      // 3-4 columns of data are required
356 //        stylestr.find("xerrorlines")    == std::string::npos  &&
      // 3-4 columns of data are required
357 //        stylestr.find("errorbars")      == std::string::npos  &&
      // 3-4 columns of data are required
358 //        stylestr.find("errorlines")     == std::string::npos  &&
      // 3-4 columns of data are required
359 //        stylestr.find("yerrorbars")     == std::string::npos  &&
      // 3-4 columns of data are required
360 //        stylestr.find("yerrorlines")    == std::string::npos  &&
      // 3-4 columns of data are required
361 //        stylestr.find("boxerrorbars")   == std::string::npos  &&
      // 3-5 columns of data are required
362 //        stylestr.find("xyerrorbars")    == std::string::npos  &&
      // 4,6,7 columns of data are required
363 //        stylestr.find("xyerrorlines")   == std::string::npos  &&
      // 4,6,7 columns of data are required
364 //        stylestr.find("boxxyerrorbars") == std::string::npos  &&
      // 4,6,7 columns of data are required
365 //        stylestr.find("financebars")    == std::string::npos  &&
```

```
    // 5 columns of data are required
//          stylestr.find("candlesticks")   == std::string::npos  &&
    // 5 columns of data are required
//          stylestr.find("vectors")        == std::string::npos  &&
//          stylestr.find("image")          == std::string::npos  &&
//          stylestr.find("rgbimage")       == std::string::npos  &&
//          stylestr.find("pm3d")           == std::string::npos  )
    {
        this->pstyle = std::string("points");
    }
    else
    {
        this->pstyle = stylestr;
    }

    return *this;
}

//
    ------------------------------------------------------------------------


// smooth: interpolation and approximation of data
Gnuplot& Gnuplot::set_smooth(const std::string &stylestr)
{
    if (stylestr.find("unique")     == std::string::npos  &&
        stylestr.find("frequency") == std::string::npos  &&
        stylestr.find("csplines")  == std::string::npos  &&
        stylestr.find("acsplines") == std::string::npos  &&
        stylestr.find("bezier")    == std::string::npos  &&
        stylestr.find("sbezier")   == std::string::npos  )
    {
        this->smooth = "";
    }
    else
    {
        this->smooth = stylestr;
    }

    return *this;
}

//
```

```
404 // unset smooth
405 Gnuplot& Gnuplot::unset_smooth()
406 {
407     this->smooth = "";
408 
409     return *this;
410 }
411 
412 //
    ----------------------------------------------------------------------

413 // sets terminal type to windows / x11
414 Gnuplot& Gnuplot::showonscreen()
415 {
416     this->cmd("set␣output");
417     this->cmd("set␣terminal␣" + Gnuplot::terminal_std);
418 
419     return *this;
420 }
421 
422 //
    ----------------------------------------------------------------------

423 // saves a gnuplot session to a postscript file
424 Gnuplot& Gnuplot::savetops(const std::string &filename)
425 {
426 //     this->cmd("set terminal postscript color");          // Tipo
    de terminal (tipo de arquivo)
427 //
428 //     std::ostringstream cmdstr;                           // Muda
    o nome do arquivo
429 //     cmdstr << "set output \"" << filename << ".ps\"";    // Nome
    do arquivo
430 //     this->cmd(cmdstr.str());
431 //     this->replot();                                      //
    Replota o grï¿œfico, agora salvando o arquivo ps
432 //
433 //     ShowOnScreen ();                                     // Volta
     para terminal modo janela
434 
```

```
435     this->cmd("set␣terminal␣png␣size␣1800,1200");
436
437    // std::ostringstream cmdstr;
438     this->cmd("␣␣set␣output␣\"./out/"+filename + ".png\"");
439     //this->cmd(cmdstr.str());
440     this->Replot();
441
442     return *this;
443 }
444 //
      ---------------------------------------------------------------------------

445 // saves a gnuplot session to a png file and return do on screen
      terminal
446 Gnuplot& Gnuplot::savetopng(const std::string &filename)
447 {
448 //                                                          // Muda
      o terminal
449 //      this->cmd("set term png enhanced size 1280,960");   // Tipo
      de terminal (tipo de arquivo)
450 //
451 //      std::ostringstream cmdstr;                          // Muda
      o nome do arquivo
452 //      cmdstr << "set output \"" << filename << ".png\"";  // Nome
      do arquivo
453 //      this->cmd(cmdstr.str());
454 //      this->replot();                                     //
      Replota o grï¿œfico, agora salvando o arquivo ps
455 //                                                          //
      Retorna o terminal para o padrï¿œo janela
456 //      ShowOnScreen ();                                    // Volta
       para terminal modo janela
457 //      this->replot();                                     //
      Replota o grï¿œfico, agora na tela
458     SaveTo("./out/"+filename,"png", "enhanced␣size␣1280,960");
459
460     return *this;
461 }
462
463 //
      ---------------------------------------------------------------------------
```

```
464// saves a gnuplot session to a jpeg file and return do on screen
     terminal
465Gnuplot& Gnuplot::savetojpeg(const std::string &filename)
466{
467                                                          // Muda o
                                                             terminal
468//     this->cmd("set term jpeg enhanced size 1280,960");   // Tipo
      de terminal (tipo de arquivo)
469//
470//     std::ostringstream cmdstr;                           // Muda
     o nome do arquivo
471//     cmdstr << "set output \"" << filename << ".jpeg\"";  // Nome
      do arquivo
472//     this->cmd(cmdstr.str());
473//     this->replot();                                      //
     Replota o grï¿œfico, agora salvando o arquivo ps
474//                                                          //
     Retorna o terminal para o padrï¿œo janela
475//     ShowOnScreen ();                                     // Volta
      para terminal modo janela
476//     this->replot();                                      //
     Replota o grï¿œfico, agora na tela
477     SaveTo(filename,"jpeg", "enhanced␣size␣1280,960");
478
479     return *this;
480}
481
482
483//
     ------------------------------------------------------------------------


484// saves a gnuplot session to spectific terminal and output file
485// @filename: name of disc file
486// @terminal_type: type of terminal
487// @flags: aditional information specitif to terminal type
488// Ex:
489// grafico.SaveTo("pressao_X_temperatura","png", "enhanced size
     1280,960");
490// grafico.TerminalType("png").SaveFile(pressao_X_temperatura);
     pense nisso?
491Gnuplot& Gnuplot::SaveTo(const std::string &filename,const std::
     string &terminal_type, std::string flags)
```

```
492 {                                                          // Muda o
    terminal
493     this->cmd("set␣term␣" + terminal_type + "␣" + flags);    //
        Tipo de terminal (tipo de arquivo) e flags adicionais
494     std::ostringstream cmdstr;                             // Muda o
        nome do arquivo
495     cmdstr << "set␣output␣\"" << filename << "." << terminal_type
        << "\"";
496     this->cmd(cmdstr.str());
497     this->replot();                                        // Replota
        o grï¿œfico, agora salvando o arquivo ps
498                                                            // Retorna
                                                               o
                                                               terminal
                                                                para o
                                                               padrï¿œo
                                                                janela
499     ShowOnScreen ();
500     this->replot();                                        // Replota
        o grï¿œfico, agora na tela
501
502     return *this;
503 }
504
505 //
    ---------------------------------------------------------------------

506 // Switches legend on
507 Gnuplot& Gnuplot::set_legend(const std::string &position)
508 {
509     std::ostringstream cmdstr;
510     cmdstr << "set␣key␣" << position;
511
512     this->cmd(cmdstr.str());
513
514     return *this;
515 }
516
517 //
    ---------------------------------------------------------------------

518 // Switches legend off
```

```
519 Gnuplot& Gnuplot::unset_legend()
520 {
521     this->cmd("unset␣key");
522 
523     return *this;
524 }
525 
526 //
      ------------------------------------------------------------------------

527 // Turns grid on
528 Gnuplot& Gnuplot::set_grid()
529 {
530     this->cmd("set␣grid");
531 
532     return *this;
533 }
534 
535 //
      ------------------------------------------------------------------------

536 // Turns grid off
537 Gnuplot& Gnuplot::unset_grid()
538 {
539     this->cmd("unset␣grid");
540 
541     return *this;
542 }
543 
544 //
      ------------------------------------------------------------------------

545 // turns on log scaling for the x axis
546 Gnuplot& Gnuplot::set_xlogscale(const double base)
547 {
548     std::ostringstream cmdstr;
549 
550     cmdstr << "set␣logscale␣x␣" << base;
551     this->cmd(cmdstr.str());
552 
553     return *this;
554 }
```

```
555
556 //
    ------------------------------------------------------------------------

557 // turns on log scaling for the y axis
558 Gnuplot& Gnuplot::set_ylogscale(const double base)
559 {
560     std::ostringstream cmdstr;
561
562     cmdstr << "set␣logscale␣y␣" << base;
563     this->cmd(cmdstr.str());
564
565     return *this;
566 }
567
568 //
    ------------------------------------------------------------------------

569 // turns on log scaling for the z axis
570 Gnuplot& Gnuplot::set_zlogscale(const double base)
571 {
572     std::ostringstream cmdstr;
573
574     cmdstr << "set␣logscale␣z␣" << base;
575     this->cmd(cmdstr.str());
576
577     return *this;
578 }
579
580 //
    ------------------------------------------------------------------------

581 // turns off log scaling for the x axis
582 Gnuplot& Gnuplot::unset_xlogscale()
583 {
584     this->cmd("unset␣logscale␣x");
585     return *this;
586 }
587
588 //
    ------------------------------------------------------------------------
```

```
589 // turns off log scaling for the y axis
590 Gnuplot& Gnuplot::unset_ylogscale()
591 {
592     this->cmd("unset␣logscale␣y");
593     return *this;
594 }
595
596 //
    ------------------------------------------------------------------------

597 // turns off log scaling for the z axis
598 Gnuplot& Gnuplot::unset_zlogscale()
599 {
600     this->cmd("unset␣logscale␣z");
601     return *this;
602 }
603
604
605 //
    ------------------------------------------------------------------------

606 // scales the size of the points used in plots
607 Gnuplot& Gnuplot::set_pointsize(const double pointsize)
608 {
609     std::ostringstream cmdstr;
610     cmdstr << "set␣pointsize␣" << pointsize;
611     this->cmd(cmdstr.str());
612
613     return *this;
614 }
615
616 //
    ------------------------------------------------------------------------

617 // set isoline density (grid) for plotting functions as surfaces
618 Gnuplot& Gnuplot::set_samples(const int samples)
619 {
620     std::ostringstream cmdstr;
621     cmdstr << "set␣samples␣" << samples;
622     this->cmd(cmdstr.str());
623
624     return *this;
```

```
625 }
626
627 //
      ------------------------------------------------------------------------
628 // set isoline density (grid) for plotting functions as surfaces
629 Gnuplot& Gnuplot::set_isosamples(const int isolines)
630 {
631     std::ostringstream cmdstr;
632     cmdstr << "set␣isosamples␣" << isolines;
633     this->cmd(cmdstr.str());
634
635     return *this;
636 }
637
638 //
      ------------------------------------------------------------------------
639 // enables hidden line removal for surface plotting
640 Gnuplot& Gnuplot::set_hidden3d()
641 {
642     this->cmd("set␣hidden3d");
643
644     return *this;
645 }
646
647 //
      ------------------------------------------------------------------------
648 // disables hidden line removal for surface plotting
649 Gnuplot& Gnuplot::unset_hidden3d()
650 {
651     this->cmd("unset␣hidden3d");
652
653     return *this;
654 }
655
656 //
      ------------------------------------------------------------------------
657 // enables contour drawing for surfaces set contour {base | surface
      | both}
```

```
658 Gnuplot& Gnuplot::set_contour(const std::string &position)
659 {
660     if (position.find("base")    == std::string::npos  &&
661         position.find("surface") == std::string::npos  &&
662         position.find("both")    == std::string::npos  )
663     {
664         this->cmd("set␣contour␣base");
665     }
666     else
667     {
668         this->cmd("set␣contour␣" + position);
669     }
670
671     return *this;
672 }
673
674 //
      ------------------------------------------------------------------------

675 // disables contour drawing for surfaces
676 Gnuplot& Gnuplot::unset_contour()
677 {
678     this->cmd("unset␣contour");
679
680     return *this;
681 }
682
683 //
      ------------------------------------------------------------------------

684 // enables the display of surfaces (for 3d plot)
685 Gnuplot& Gnuplot::set_surface()
686 {
687     this->cmd("set␣surface");
688
689     return *this;
690 }
691
692 //
      ------------------------------------------------------------------------

693 // disables the display of surfaces (for 3d plot)
```

```
694 Gnuplot& Gnuplot::unset_surface()
695 {
696     this->cmd("unset␣surface");
697 
698     return *this;
699 }
700 
701 //
    ------------------------------------------------------------------------

702 // Sets the title of a gnuplot session
703 Gnuplot& Gnuplot::set_title(const std::string &title)
704 {
705     std::ostringstream cmdstr;
706 
707     cmdstr << "set␣title␣\"" << title << "\"";
708     this->cmd(cmdstr.str());
709 
710     return *this;
711 }
712 
713 //
    ------------------------------------------------------------------------

714 // Clears the title of a gnuplot session
715 Gnuplot& Gnuplot::unset_title()
716 {
717     this->set_title("");
718 
719     return *this;
720 }
721 
722 //
    ------------------------------------------------------------------------

723 // set labels
724 // set the xlabel
725 Gnuplot& Gnuplot::set_xlabel(const std::string &label)
726 {
727     std::ostringstream cmdstr;
728 
729     cmdstr << "set␣xlabel␣\"" << label << "\"";
```

```
730     this->cmd(cmdstr.str());
731 
732     return *this;
733 }
734 
735 //
    ------------------------------------------------------------------------

736 // set the ylabel
737 Gnuplot& Gnuplot::set_ylabel(const std::string &label)
738 {
739     std::ostringstream cmdstr;
740 
741     cmdstr << "set␣ylabel␣\"" << label << "\"";
742     this->cmd(cmdstr.str());
743 
744     return *this;
745 }
746 
747 //
    ------------------------------------------------------------------------

748 // set the zlabel
749 Gnuplot& Gnuplot::set_zlabel(const std::string &label)
750 {
751     std::ostringstream cmdstr;
752 
753     cmdstr << "set␣zlabel␣\"" << label << "\"";
754     this->cmd(cmdstr.str());
755 
756     return *this;
757 }
758 
759 //
    ------------------------------------------------------------------------

760 // set range
761 // set the xrange
762 Gnuplot& Gnuplot::set_xrange(const int iFrom,
763                              const int iTo)
764 {
765     std::ostringstream cmdstr;
```

```
766
767     cmdstr << "set␣xrange[" << iFrom << ":" << iTo << "]";
768     this->cmd(cmdstr.str());
769
770     return *this;
771 }
772
773 //
      ----------------------------------------------------------------------------
774 // set autoscale x
775 Gnuplot& Gnuplot::set_xautoscale()
776 {
777     this->cmd("set␣xrange␣restore");
778     this->cmd("set␣autoscale␣x");
779
780     return *this;
781 }
782
783 //
      ----------------------------------------------------------------------------
784 // set the yrange
785 Gnuplot& Gnuplot::set_yrange(const int iFrom, const int iTo)
786 {
787     std::ostringstream cmdstr;
788
789     cmdstr << "set␣yrange[" << iFrom << ":" << iTo << "]";
790     this->cmd(cmdstr.str());
791
792     return *this;
793 }
794
795 //
      ----------------------------------------------------------------------------
796 // set autoscale y
797 Gnuplot& Gnuplot::set_yautoscale()
798 {
799     this->cmd("set␣yrange␣restore");
800     this->cmd("set␣autoscale␣y");
801
```

```
802     return *this;
803 }
804
805 //
    ------------------------------------------------------------------------

806 // set the zrange
807 Gnuplot& Gnuplot::set_zrange(const int iFrom,
808                              const int iTo)
809 {
810     std::ostringstream cmdstr;
811
812     cmdstr << "set␣zrange[" << iFrom << ":" << iTo << "]";
813     this->cmd(cmdstr.str());
814
815     return *this;
816 }
817
818 //
    ------------------------------------------------------------------------

819 // set autoscale z
820 Gnuplot& Gnuplot::set_zautoscale()
821 {
822     this->cmd("set␣zrange␣restore");
823     this->cmd("set␣autoscale␣z");
824
825     return *this;
826 }
827
828 //
    ------------------------------------------------------------------------

829 // set the palette range
830 Gnuplot& Gnuplot::set_cbrange(const int iFrom,
831                               const int iTo)
832 {
833     std::ostringstream cmdstr;
834
835     cmdstr << "set␣cbrange[" << iFrom << ":" << iTo << "]";
836     this->cmd(cmdstr.str());
837
```

```
838     return *this;
839 }
840 
841 //
    ------------------------------------------------------------------------

842 // Plots a linear equation y=ax+b (where you supply the
843 // slope a and intercept b)
844 Gnuplot& Gnuplot::plot_slope(const double a,
845                              const double b,
846                              const std::string &title)
847 {
848     std::ostringstream cmdstr;
849 
850     // command to be sent to gnuplot
851     if (this->nplots > 0   &&   this->two_dim == true)
852         cmdstr << "replot␣";
853     else
854         cmdstr << "plot␣";
855 
856     cmdstr << a << "␣*␣x␣+␣" << b << "␣title␣\"";
857 
858     if (title == "")
859         cmdstr << "f(x)␣=␣" << a << "␣*␣x␣+␣" << b;
860     else
861         cmdstr << title;
862 
863     cmdstr << "\"␣with␣" << this->pstyle;
864 
865     // Do the actual plot
866     this->cmd(cmdstr.str());
867 
868     return *this;
869 }
870 
871 //
    ------------------------------------------------------------------------

872 // Plot an equation which is supplied as a std::string y=f(x) (only
      f(x) expected)
873 Gnuplot& Gnuplot::plot_equation(const std::string &equation,
874                                 const std::string &title)
```

```
875 {
876     std::ostringstream cmdstr;
877 
878     // command to be sent to gnuplot
879     if (this->nplots > 0   &&   this->two_dim == true)
880         cmdstr << "replot␣";
881     else
882         cmdstr << "plot␣";
883 
884     cmdstr << equation << "␣title␣\"";
885 
886     if (title == "")
887         cmdstr << "f(x)␣=␣" << equation;
888     else
889         cmdstr << title;
890 
891     cmdstr << "\"␣with␣" << this->pstyle;
892 
893     // Do the actual plot
894     this->cmd(cmdstr.str());
895 
896     return *this;
897 }
898 
899 //
    ------------------------------------------------------------------------

900 // plot an equation supplied as a std::string y=(x)
901 Gnuplot& Gnuplot::plot_equation3d(const std::string &equation,
902                                   const std::string &title)
903 {
904     std::ostringstream cmdstr;
905 
906     // command to be sent to gnuplot
907     if (this->nplots > 0   &&   this->two_dim == false)
908         cmdstr << "replot␣";
909     else
910         cmdstr << "splot␣";
911 
912     cmdstr << equation << "␣title␣\"";
913 
914     if (title == "")
```

```
915         cmdstr << "f(x,y) = " << equation;
916     else
917         cmdstr << title;
918 
919     cmdstr << "\" with " << this->pstyle;
920 
921     // Do the actual plot
922     this->cmd(cmdstr.str());
923 
924     return *this;
925 }
926 
927 //
    ------------------------------------------------------------------------

928 // Plots a 2d graph from a list of doubles (x) saved in a file
929 Gnuplot& Gnuplot::plotfile_x(const std::string &filename,
930                              const int column,
931                              const std::string &title)
932 {
933     // check if file exists
934     if( !(Gnuplot::file_exists(filename,4)) ) // check existence
           and read permission
935     {
936         std::ostringstream except;
937         if( !(Gnuplot::file_exists(filename,0)) ) // check
               existence
938             except << "File \"" << filename << "\" does not exist";
939         else
940             except << "No read permission for File \"" << filename
                   << "\"";
941         throw GnuplotException( except.str() );
942         return *this;
943     }
944 
945     std::ostringstream cmdstr;
946 
947     // command to be sent to gnuplot
948     if (this->nplots > 0  &&  this->two_dim == true)
949         cmdstr << "replot ";
950     else
951         cmdstr << "plot ";
```

```
952
953     cmdstr << "\"" << filename << "\" using " << column;
954
955     if (title == "")
956         cmdstr << " notitle ";
957     else
958         cmdstr << " title \"" << title << "\" ";
959
960     if(smooth == "")
961         cmdstr << "with " << this->pstyle;
962     else
963         cmdstr << "smooth " << this->smooth;
964
965     // Do the actual plot
966     this->cmd(cmdstr.str()); //nplots++; two_dim = true;  already
        in this->cmd();
967
968     return *this;
969 }
970
971 //
    ------------------------------------------------------------------------
972 // Plots a 2d graph from a list of doubles: x
973 Gnuplot& Gnuplot::plot_x(const std::vector<double> &x,
974                          const std::string &title)
975 {
976     if (x.size() == 0)
977     {
978         throw GnuplotException("std::vector too small");
979         return *this;
980     }
981
982     std::ofstream tmp;
983     std::string name = create_tmpfile(tmp);
984     if (name == "")
985         return *this;
986
987     // write the data to file
988     for (unsigned int i = 0; i < x.size(); i++)
989         tmp << x[i] << std::endl;
990
```

```
991     tmp.flush();
992     tmp.close();
993 
994     this->plotfile_x(name, 1, title);
995 
996     return *this;
997 }
998 
999 //
      ---------------------------------------------------------------------------

1000 // Plots a 2d graph from a list of doubles (x y) saved in a file
1001 Gnuplot& Gnuplot::plotfile_xy(const std::string &filename,
1002                               const int column_x,
1003                               const int column_y,
1004                               const std::string &title)
1005 {
1006     // check if file exists
1007     if( !(Gnuplot::file_exists(filename,4)) ) // check existence
           and read permission
1008     {
1009         std::ostringstream except;
1010         if( !(Gnuplot::file_exists(filename,0)) ) // check
              existence
1011             except << "File␣\"" << filename << "\"␣does␣not␣exist";
1012         else
1013             except << "No␣read␣permission␣for␣File␣\"" << filename
                  << "\"";
1014         throw GnuplotException( except.str() );
1015         return *this;
1016     }
1017 
1018     std::ostringstream cmdstr;
1019 
1020     // command to be sent to gnuplot
1021     if (this->nplots > 0  &&  this->two_dim == true)
1022         cmdstr << "replot␣";
1023     else
1024         cmdstr << "plot␣";
1025 
1026     cmdstr << "\"" << filename << "\"␣using␣" << column_x << ":" <<
          column_y;
```

```
1027
1028     if (title == "")
1029         cmdstr << "␣notitle␣";
1030     else
1031         cmdstr << "␣title␣\"" << title << "\"␣";
1032
1033     if(smooth == "")
1034         cmdstr << "with␣" << this->pstyle;
1035     else
1036         cmdstr << "smooth␣" << this->smooth;
1037
1038     // Do the actual plot
1039     this->cmd(cmdstr.str());
1040
1041     return *this;
1042 }
1043
1044 //
      ---------------------------------------------------------------------------

1045 // Plots a 2d graph from a list of doubles: x y
1046 Gnuplot& Gnuplot::plot_xy(const std::vector<double> &x,
1047                           const std::vector<double> &y,
1048                           const std::string &title)
1049 {
1050     if (x.size() == 0 || y.size() == 0)
1051     {
1052         throw GnuplotException("std::vectors␣too␣small");
1053         return *this;
1054     }
1055
1056     if (x.size() != y.size())
1057     {
1058         throw GnuplotException("Length␣of␣the␣std::vectors␣differs"
                ");
1059         return *this;
1060     }
1061
1062     std::ofstream tmp;
1063     std::string name = create_tmpfile(tmp);
1064     if (name == "")
1065         return *this;
```

```
1066
1067     // write the data to file
1068     for (unsigned int i = 0; i < x.size(); i++)
1069         tmp << x[i] << " " << y[i] << std::endl;
1070
1071     tmp.flush();
1072     tmp.close();
1073
1074     this->plotfile_xy(name, 1, 2, title);
1075
1076     return *this;
1077 }
1078
1079 //
      ---------------------------------------------------------------------------
1080 // Plots a 2d graph with errorbars from a list of doubles (x y dy)
      saved in a file
1081 Gnuplot& Gnuplot::plotfile_xy_err(const std::string &filename,
1082                                   const int column_x,
1083                                   const int column_y,
1084                                   const int column_dy,
1085                                   const std::string &title)
1086 {
1087     // check if file exists
1088     if( !(Gnuplot::file_exists(filename,4)) ) // check existence
           and read permission
1089     {
1090         std::ostringstream except;
1091         if( !(Gnuplot::file_exists(filename,0)) ) // check
              existence
1092             except << "File \"" << filename << "\" does not exist";
1093         else
1094             except << "No read permission for File \"" << filename
                  << "\"";
1095         throw GnuplotException( except.str() );
1096         return *this;
1097     }
1098
1099     std::ostringstream cmdstr;
1100
1101     // command to be sent to gnuplot
```

```
1102     if (this->nplots > 0   &&   this->two_dim == true)
1103         cmdstr << "replot ";
1104     else
1105         cmdstr << "plot ";
1106 
1107     cmdstr << "\"" << filename << "\" using " << column_x << ":" <<
         column_y;
1108 
1109     if (title == "")
1110         cmdstr << " notitle ";
1111     else
1112         cmdstr << " title \"" << title << "\" ";
1113 
1114     cmdstr << "with " << this->pstyle << ", \"" << filename << "\"
        using "
1115             << column_x << ":" << column_y << ":" << column_dy << "
                notitle with errorbars";
1116 
1117     // Do the actual plot
1118     this->cmd(cmdstr.str());
1119 
1120     return *this;
1121 }
1122 
1123 //
      ------------------------------------------------------------------------


1124 // plot x,y pairs with dy errorbars
1125 Gnuplot& Gnuplot::plot_xy_err(const std::vector<double> &x,
1126                               const std::vector<double> &y,
1127                               const std::vector<double> &dy,
1128                               const std::string &title)
1129 {
1130     if (x.size() == 0 || y.size() == 0 || dy.size() == 0)
1131     {
1132         throw GnuplotException("std::vectors too small");
1133         return *this;
1134     }
1135 
1136     if (x.size() != y.size() || y.size() != dy.size())
1137     {
1138         throw GnuplotException("Length of the std::vectors differs"
```

```
            );
1139        return *this;
1140    }
1141
1142    std::ofstream tmp;
1143    std::string name = create_tmpfile(tmp);
1144    if (name == "")
1145        return *this;
1146
1147    // write the data to file
1148    for (unsigned int i = 0; i < x.size(); i++)
1149        tmp << x[i] << " " << y[i] << " " << dy[i] << std::endl;
1150
1151    tmp.flush();
1152    tmp.close();
1153
1154    // Do the actual plot
1155    this->plotfile_xy_err(name, 1, 2, 3, title);
1156
1157    return *this;
1158}
1159
1160//
    ------------------------------------------------------------------------

1161// Plots a 3d graph from a list of doubles (x y z) saved in a file
1162Gnuplot& Gnuplot::plotfile_xyz(const std::string &filename,
1163                               const int column_x,
1164                               const int column_y,
1165                               const int column_z,
1166                               const std::string &title)
1167{
1168
1169    // check if file exists
1170    if( !(Gnuplot::file_exists(filename,4)) ) // check existence
        and read permission
1171    {
1172        std::ostringstream except;
1173        if( !(Gnuplot::file_exists(filename,0)) ) // check
            existence
1174            except << "File " << filename << " does not exist";
1175        else
```

```
1176            except << "No␣read␣permission␣for␣File␣\"" << filename
                   << "\"";
1177        throw GnuplotException( except.str() );
1178        return *this;
1179    }
1180
1181    std::ostringstream cmdstr;
1182
1183    // command to be sent to gnuplot
1184    if (this->nplots > 0  &&  this->two_dim == false)
1185        cmdstr << "replot␣";
1186    else
1187        cmdstr << "splot␣";
1188
1189    cmdstr << "\"" << filename << "\"␣using␣" << column_x << ":" <<
        column_y << ":" << column_z;
1190
1191    if (title == "")
1192        cmdstr << "␣notitle␣with␣" << this->pstyle;
1193    else
1194        cmdstr << "␣title␣\"" << title << "\"␣with␣" << this->
           pstyle;
1195
1196    // Do the actual plot
1197    this->cmd(cmdstr.str());
1198
1199    return *this;
1200}
1201
1202//
    ------------------------------------------------------------------------

1203// Plots a 3d graph from a list of doubles: x y z
1204Gnuplot& Gnuplot::plot_xyz(const std::vector<double> &x,
1205                           const std::vector<double> &y,
1206                           const std::vector<double> &z,
1207                           const std::string &title)
1208{
1209    if (x.size() == 0 || y.size() == 0 || z.size() == 0)
1210    {
1211        throw GnuplotException("std::vectors␣too␣small");
1212        return *this;
```

```
1213     }
1214
1215     if (x.size() != y.size() || x.size() != z.size())
1216     {
1217         throw GnuplotException("Length␣of␣the␣std::vectors␣differs"
                );
1218         return *this;
1219     }
1220
1221
1222     std::ofstream tmp;
1223     std::string name = create_tmpfile(tmp);
1224     if (name == "")
1225         return *this;
1226
1227     // write the data to file
1228     for (unsigned int i = 0; i < x.size(); i++)
1229     {
1230         tmp << x[i] << "␣" << y[i] << "␣" << z[i] <<std::endl;
1231     }
1232
1233     tmp.flush();
1234     tmp.close();
1235
1236
1237     this->plotfile_xyz(name, 1, 2, 3, title);
1238
1239     return *this;
1240 }
1241
1242
1243
1244 //
       ------------------------------------------------------------------------

1245 /// *  note that this function is not valid for versions of GNUPlot
       below 4.2
1246 Gnuplot& Gnuplot::plot_image(const unsigned char * ucPicBuf,
1247                              const int iWidth,
1248                              const int iHeight,
1249                              const std::string &title)
1250 {
```

```
1251     std::ofstream tmp;
1252     std::string name = create_tmpfile(tmp);
1253     if (name == "")
1254         return *this;
1255 
1256     // write the data to file
1257     int iIndex = 0;
1258     for(int iRow = 0; iRow < iHeight; iRow++)
1259     {
1260         for(int iColumn = 0; iColumn < iWidth; iColumn++)
1261         {
1262             tmp << iColumn << " " << iRow  << " " << static_cast<
                    float>(ucPicBuf[iIndex++]) << std::endl;
1263         }
1264     }
1265 
1266     tmp.flush();
1267     tmp.close();
1268 
1269 
1270     std::ostringstream cmdstr;
1271 
1272     // command to be sent to gnuplot
1273     if (this->nplots > 0   &&   this->two_dim == true)
1274         cmdstr << "replot ";
1275     else
1276         cmdstr << "plot ";
1277 
1278     if (title == "")
1279         cmdstr << "\"" << name << "\" with image";
1280     else
1281         cmdstr << "\"" << name << "\" title \"" << title << "\" 
                with image";
1282 
1283     // Do the actual plot
1284     this->cmd(cmdstr.str());
1285 
1286     return *this;
1287 }
1288 
1289 //
```

```
1290// Sends a command to an active gnuplot session
1291Gnuplot& Gnuplot::cmd(const std::string &cmdstr)
1292{
1293    if( !(this->valid) )
1294    {
1295        return *this;
1296    }
1297
1298    // int fputs ( const char * str, FILE * stream );
1299    // writes the string str to the stream.
1300    // The function begins copying from the address specified (str)
         until it reaches the
1301    // terminating null character ('\0'). This final null-character
         is not copied to the stream.
1302    fputs( (cmdstr+"\n").c_str(), this->gnucmd );
1303
1304    // int fflush ( FILE * stream );
1305    // If the given stream was open for writing and the last i/o
       operation was an output operation,
1306    // any unwritten data in the output buffer is written to the
       file.
1307    // If the argument is a null pointer, all open files are
       flushed.
1308    // The stream remains open after this call.
1309    fflush(this->gnucmd);
1310
1311
1312    if( cmdstr.find("replot") != std::string::npos )
1313    {
1314        return *this;
1315    }
1316    else if( cmdstr.find("splot") != std::string::npos )
1317    {
1318        this->two_dim = false;
1319        this->nplots++;
1320    }
1321    else if( cmdstr.find("plot") != std::string::npos )
1322    {
1323        this->two_dim = true;
1324        this->nplots++;
1325    }
```

```
1326     return *this;
1327 }
1328
1329 //
     ------------------------------------------------------------------------

1330 // Sends a command to an active gnuplot session, identical to cmd()
1331 Gnuplot& Gnuplot::operator<<(const std::string &cmdstr)
1332 {
1333     this->cmd(cmdstr);
1334     return *this;
1335 }
1336
1337 //
     ------------------------------------------------------------------------

1338 // Opens up a gnuplot session, ready to receive commands
1339 void Gnuplot::init()
1340 {
1341     // char * getenv ( const char * name );  get value of an
          environment variable
1342     // Retrieves a C string containing the value of the environment
           variable whose
1343     // name is specified as argument.
1344     // If the requested variable is not part of the environment
          list, the function returns a NULL pointer.
1345 #if ( defined(unix) || defined(__unix) || defined(__unix__) ) && !
     defined(__APPLE__)
1346     if (getenv("DISPLAY") == NULL)
1347     {
1348         this->valid = false;
1349         throw GnuplotException("Can't␣find␣DISPLAY␣variable");
1350     }
1351 #endif
1352
1353     // if gnuplot not available
1354     if (!Gnuplot::get_program_path())
1355     {
1356         this->valid = false;
1357         throw GnuplotException("Can't␣find␣gnuplot");
1358     }
1359
```

```
1360     // open pipe
1361     std::string tmp = Gnuplot::m_sGNUPlotPath + "/" + Gnuplot::
             m_sGNUPlotFileName;
1362
1363     // FILE *popen(const char *command, const char *mode);
1364     // The popen() function shall execute the command specified by
             the string command,
1365     // create a pipe between the calling program and the executed
             command, and
1366     // return a pointer to a stream that can be used to either read
              from or write to the pipe.
1367#if defined(WIN32) || defined(_WIN32) || defined(__WIN32__) ||
      defined(__TOS_WIN__)
1368     this->gnucmd = _popen(tmp.c_str(),"w");
1369#elif defined(unix) || defined(__unix) || defined(__unix__) ||
      defined(__APPLE__)
1370     this->gnucmd = popen(tmp.c_str(),"w");
1371#endif
1372
1373     // popen() shall return a pointer to an open stream that can be
              used to read or write to the pipe.
1374     // Otherwise, it shall return a null pointer and may set errno
             to indicate the error.
1375     if (!this->gnucmd)
1376     {
1377         this->valid = false;
1378         throw GnuplotException("Couldn't␣open␣connection␣to␣gnuplot
                ");
1379     }
1380
1381     this->nplots = 0;
1382     this->valid = true;
1383     this->smooth = "";
1384
1385     //set terminal type
1386     this->showonscreen();
1387
1388     return;
1389}
1390
1391//
      ---------------------------------------------------------------------------
```

```
1392// Find out if a command lives in m_sGNUPlotPath or in PATH
1393bool Gnuplot::get_program_path()
1394{
1395    // first look in m_sGNUPlotPath for Gnuplot
1396    std::string tmp = Gnuplot::m_sGNUPlotPath + "/" + Gnuplot::
        m_sGNUPlotFileName;
1397
1398#if defined(WIN32) || defined(_WIN32) || defined(__WIN32__) ||
    defined(__TOS_WIN__)
1399     if ( Gnuplot::file_exists(tmp,0) ) // check existence
1400#elif defined(unix) || defined(__unix) || defined(__unix__) ||
    defined(__APPLE__)
1401     if ( Gnuplot::file_exists(tmp,1) ) // check existence and
        execution permission
1402#endif
1403    {
1404        return true;
1405    }
1406
1407    // second look in PATH for Gnuplot
1408    char *path;
1409    // Retrieves a C string containing the value of the environment
        variable PATH
1410    path = getenv("PATH");
1411
1412    if (path == NULL)
1413    {
1414        throw GnuplotException("Path is not set");
1415        return false;
1416    }
1417    else
1418    {
1419        std::list<std::string> ls;
1420        //split path (one long string) into list ls of strings
1421#if defined(WIN32) || defined(_WIN32) || defined(__WIN32__) ||
    defined(__TOS_WIN__)
1422        stringtok(ls,path,";");
1423#elif defined(unix) || defined(__unix) || defined(__unix__) ||
    defined(__APPLE__)
1424        stringtok(ls,path,":");
1425#endif
```

```
1426         // scan list for Gnuplot program files
1427         for (std::list<std::string>::const_iterator i = ls.begin();
                 i != ls.end(); ++i)
1428         {
1429             tmp = (*i) + "/" + Gnuplot::m_sGNUPlotFileName;
1430#if defined(WIN32) || defined(_WIN32) || defined(__WIN32__) ||
     defined(__TOS_WIN__)
1431             if ( Gnuplot::file_exists(tmp,0) ) // check existence
1432#elif defined(unix) || defined(__unix) || defined(__unix__) ||
     defined(__APPLE__)
1433             if ( Gnuplot::file_exists(tmp,1) ) // check existence
                    and execution permission
1434#endif
1435             {
1436                 Gnuplot::m_sGNUPlotPath = *i; // set m_sGNUPlotPath
1437                 return true;
1438             }
1439         }
1440
1441         tmp = "Can't find gnuplot neither in PATH nor in \"" +
                Gnuplot::m_sGNUPlotPath + "\"";
1442         throw GnuplotException(tmp);
1443
1444         Gnuplot::m_sGNUPlotPath = "";
1445         return false;
1446     }
1447}
1448
1449//
     ------------------------------------------------------------------------


1450// check if file exists
1451bool Gnuplot::file_exists(const std::string &filename, int mode)
1452{
1453     if ( mode < 0 || mode > 7)
1454     {
1455         throw std::runtime_error("In function \"Gnuplot::
                file_exists\": mode has to be an integer between 0 and 7
                ");
1456         return false;
1457     }
1458
```

```
1459     // int _access(const char *path, int mode);
1460     //  returns 0 if the file has the given mode,
1461     //  it returns -1 if the named file does not exist or is not
            accessible in the given mode
1462     // mode = 0 (F_OK) (default): checks file for existence only
1463     // mode = 1 (X_OK): execution permission
1464     // mode = 2 (W_OK): write permission
1465     // mode = 4 (R_OK): read permission
1466     // mode = 6       : read and write permission
1467     // mode = 7       : read, write and execution permission
1468 #if defined(WIN32) || defined(_WIN32) || defined(__WIN32__) ||
      defined(__TOS_WIN__)
1469     if (_access(filename.c_str(), mode) == 0)
1470 #elif defined(unix) || defined(__unix) || defined(__unix__) ||
      defined(__APPLE__)
1471     if (access(filename.c_str(), mode) == 0)
1472 #endif
1473     {
1474         return true;
1475     }
1476     else
1477     {
1478         return false;
1479     }
1480
1481 }
1482
1483 //
      ---------------------------------------------------------------------------


1484 // Opens a temporary file
1485 std::string Gnuplot::create_tmpfile(std::ofstream &tmp)
1486 {
1487 #if defined(WIN32) || defined(_WIN32) || defined(__WIN32__) ||
      defined(__TOS_WIN__)
1488     char name[] = "gnuplotiXXXXXX"; //tmp file in working directory
1489 #elif defined(unix) || defined(__unix) || defined(__unix__) ||
      defined(__APPLE__)
1490     char name[] = "/tmp/gnuplotiXXXXXX"; // tmp file in /tmp
1491 #endif
1492
1493     // check if maximum number of temporary files reached
```

```
1494     if (Gnuplot::tmpfile_num == GP_MAX_TMP_FILES - 1)
1495     {
1496         std::ostringstream except;
1497         except << "Maximum␣number␣of␣temporary␣files␣reached␣(" <<
             GP_MAX_TMP_FILES
1498             << "):␣cannot␣open␣more␣files" << std::endl;
1499 
1500         throw GnuplotException( except.str() );
1501         return "";
1502     }
1503 
1504     // int mkstemp(char *name);
1505     // shall replace the contents of the string pointed to by "name
         " by a unique filename,
1506     // and return a file descriptor for the file open for reading
         and writing.
1507     // Otherwise, -1 shall be returned if no suitable file could be
          created.
1508     // The string in template should look like a filename with six
         trailing 'X' s;
1509     // mkstemp() replaces each 'X' with a character from the
         portable filename character set.
1510     // The characters are chosen such that the resulting name does
         not duplicate the name of an existing file at the time of a
         call to mkstemp()
1511 
1512 
1513     // open temporary files for output
1514 #if defined(WIN32) || defined(_WIN32) || defined(__WIN32__) ||
     defined(__TOS_WIN__)
1515     if (_mktemp(name) == NULL)
1516 #elif defined(unix) || defined(__unix) || defined(__unix__) ||
     defined(__APPLE__)
1517     if (mkstemp(name) == -1)
1518 #endif
1519     {
1520         std::ostringstream except;
1521         except << "Cannot␣create␣temporary␣file␣\"" << name << "\""
             ;
1522         throw GnuplotException(except.str());
1523         return "";
1524     }
```

```
1525
1526     tmp.open(name);
1527     if (tmp.bad())
1528     {
1529         std::ostringstream except;
1530         except << "Cannot␣create␣temporary␣file␣\"" << name << "\""
                ;
1531         throw GnuplotException(except.str());
1532         return "";
1533     }
1534
1535     // Save the temporary filename
1536     this->tmpfile_list.push_back(name);
1537     Gnuplot::tmpfile_num++;
1538
1539     return name;
1540 }
```

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

Listing 6.7: 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 especificos
17         CLinhaPressao2Pocos(double _r1, double _r2, double _r3) :
              r1{ _r1 }, r2{ _r2 }, r3{ _r3 }{}
18         double AreaInvadidaBT(double C);
19         double Pressao(double q,double u,double k,double h,double
              Pi);
20         double R(double C);
```

```
21          vector<double> CalculoDoVetorX(double R, double C = 0.0);
22          vector<double> CalculoDoVetorY(double R, double C = 0.0);
23
24  private:
25          double X0(double C, double R);
26          double CalculoDoVetorRaio(double C);
27          double r1, r2, r3;
28  };
29  #endif // CLINHAPRESSAO2POCOS_H
```

Apresenta-se na listagem 6.8 o arquivo de implementação da classe CLinhaPressao2Pocos.

Listing 6.8: 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,
       double h,double Pi){
8          return ((q*u*log(r2/r1))/(2.0*pi*k*h) )+ Pi;
9  }
10
11 double CLinhaPressao2Pocos::R( double C) {
12     return (r2/r1)*(r2/r1);
13 }
14
15 vector<double> CLinhaPressao2Pocos::CalculoDoVetorX(double R,
       double C){
16          double x0 = X0(C,R);
17          int i;
18          vector<double> xplot;
19          for (double theta = 0; theta <= 2 * pi; theta = theta +
               0.01) {
20                  i = theta * 100;
21                  xplot.push_back(x0 + (2 * (C)*sqrt(R) / (R - 1)) *
                       cos(theta));
22          }
23          return xplot;
24  }
25
26 vector<double> CLinhaPressao2Pocos::CalculoDoVetorY(double R,
```

```
    double C){
27          double y0 = 0;
28          int i;
29          vector<double> yplot;
30
31          for (double theta = 0; theta <= 2 * pi; theta = theta +
                0.01) {
32                  i = theta * 100;
33                  yplot.push_back(y0 + (2 * (C)*sqrt(R) / (R - 1)) *
                        sin(theta));
34          }
35          return yplot;
36}
37
38/// mÃ@todos privados
39
40double CLinhaPressao2Pocos::X0(double C, double R) {
41          return C * (R + 1.) / (R - 1.);
42}
43
44double CLinhaPressao2Pocos::CalculoDoVetorRaio(double C) { ///
    preciso passasr pro R E PRO RAIO ?
45          double r = R(C);
46          return 2 * (C)*sqrt(r) / (r - 1);
47}
```

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

Listing 6.9: 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
8using namespace std;
9
10class CLinhaPressao3Pocos1P2I : public CLinhasEquipotenciais {
11public:
12          CLinhaPressao3Pocos1P2I(double _r1, double _r2, double _r3)
                : r1{ _r1 }, r2{ _r2 }, r3{ _r3 }{};
13          double AreaInvadidaBT(double C);
```

```
14          double Pressao(double q,double u,double k,double h,double
               Pi);
15          double R(double C);
16          vector<double> CalculoDoVetorRaio(double C);
17          vector<double> CalculoDoVetorX(double R, double C = 0.0);
18          vector<double> CalculoDoVetorY(double R, double C = 0.0);
19
20  protected:
21          double r1, r2, r3;
22  };
23  #endif
```

Apresenta-se na listagem 6.10 o arquivo de implementação da classe CLinhaPressao3Pocos1P2I.

Listing 6.10: 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
       ,double h,double Pi){
8          return -(q*u/4*(log(r1*r1*r2*r2/(r3*r3*r3*r3)))/(2*pi*k*h))
               + Pi;
9  }
10
11 double CLinhaPressao3Pocos1P2I::R(double C){
12         double r= pow(r1*r2,2)/pow(r3,4);
13     r = 2*C*C /(r-1);
14     return r;
15 }
16
17 vector<double> CLinhaPressao3Pocos1P2I::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((r*r*cos(2*theta)
               )*(r*r*cos(2*theta)) + 2*C*C*r))/2 ));
22         }
23   return raio;
```

```
24 }
25
26 vector<double> CLinhaPressao3Pocos1P2I::CalculoDoVetorX(double R,
       double C) {
27         double x0 = 0.0;
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> CLinhaPressao3Pocos1P2I::CalculoDoVetorY(double R,
       double C) {
40         double y0 = 0.0;
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 CLinhaPressao3Pocos2P1I.

Listing 6.11: 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)
               : r1{ _r1 }, r2{ _r2 }, r3{ _r3 }{}
12
13 ///metodos especificos
14         double AreaInvadidaBT(double C);
15         double Pressao(double q,double u,double k,double h,double
              Pi);
16         double R(double C);
17         vector<double> CalculoDoVetorRaio(double C);
18         vector<double> CalculoDoVetorX(double R, double C = 0.0);
19         vector<double> CalculoDoVetorY(double R, double C = 0.0);
20 private:
21         double r1, r2, r3;
22 };
23 #endif
```

Apresenta-se na listagem 6.12 o arquivo de implementação da classe CLinhaPressao3Pocos2P1I.

Listing 6.12: 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,
    double h,double Pi){
8         return -(q*u*log(pow(r3,4)/(pow(r1,2)*pow(r2,2))))/(8*pi*k*
              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
26vector<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
39vector<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 6.13: Arquivo de implementação da classe CLinhasEquipotenciais.

```
1#ifndef CLINHASEQUIPOTENCIAIS_H
```

```
#define CLINHASEQUIPOTENCIAIS_H

#include <vector>
using namespace std;
class CLinhasEquipotenciais {
public:
    // declaraÃ§Ã£o de mÃ©todo virtual puro, o igual a zero indica
       que vai ser implementada nas filhas
    virtual double AreaInvadidaBT(double C) = 0;
    virtual double Pressao(double q, double u, double k, double h,
       double Pi) = 0;
    virtual double R(double C) = 0;
    virtual vector<double> CalculoDoVetorX(double raio, double
       deslocamento = 0.0) = 0;
    virtual vector<double>  CalculoDoVetorY(double raio, double
       deslocamento = 0.0) = 0;
protected:
    const double pi = 3.14159265;
    double r1, r2, r3;
};
#endif
```

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

Listing 6.14: Arquivo de implementação da classe CLinhasEquipotenciais.

```
#include "CLinhasEquipotenciais.h"
```

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

Listing 6.15: Arquivo de implementação da classe COleoProduzido.

```
#ifndef COLEOPRODUZIDO_H_
#define COLEOPRODUZIDO_H_

class COleoProduzido {

    double Np, Tbreak;

public:

    COleoProduzido(){};

    void OleoProduzido(double _largura, double _comprimento, double
        _PHI, double _espessuraTotal, double _Sor, double _Swi,
```

```
        double _effVertTotal, double _Bo);
13    void TempoBreakThrough(double _VazaoTotal);
14    double getNp();
15    double getTbreak();
16
17    ~COleoProduzido(){};
18
19};
20
21#endif
```

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

Listing 6.16: Arquivo de implementação da classe COleoProduzido.

```
1#include "COleoProduzido.h"
2
3void COleoProduzido::OleoProduzido(double _largura, double
    _comprimento, double _PHI, double _espessuraTotal, double _Sor,
    double _Swi, double _effVertTotal, double _Bo)
4{
5
6    Np = (_largura*_comprimento*_PHI*_espessuraTotal*_effVertTotal
       *(1.0 - _Sor - _Swi))/_Bo;
7
8}
9
10void COleoProduzido::TempoBreakThrough(double _VazaoTotal)
11{
12    Tbreak = Np/_VazaoTotal;
13}
14
15double COleoProduzido::getTbreak()
16{
17    return Tbreak;
18}
19
20double COleoProduzido::getNp()
21{
22    return Np;
23}
```

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

Listing 6.17: Arquivo de implementação da classe CPoco.

```
#ifndef CPOCO_H_
#define CPOCO_H_

class CPoco {

        protected:

                double x, y, c, Pi;

        public:

                CPoco(){};

                CPoco(double _x, double _y, double _c, double _Pi):
                    x(_x), y(_y), c(_c), Pi(_Pi){};

                //metodos set
                void setX(double _x);
                void setY(double _y);
                void setC(double _c);
                void setPi(double _pi);

                //metodos get

                double getX();
                double getY();
                double getC();
                double getPi();

                ~CPoco(){};

};

#endif
```

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

Listing 6.18: Arquivo de implementação da classe CFormaReservatorio.

```
#include "CPoco.h"

void CPoco::setX(double _x){

```

```
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 6.19: 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,
                       vazaoBt;
13
14         public:
15
16                 CPosicaoAguaInjetada(){};
17
18                 std::vector <double> VazaoInjecao(double _krw,
                       double _mw, double _dp, std::vector <double> _k,
                        double _bw, double _comprimento, double _M, std
                       ::vector <double> _posicaoc, std::vector <double
                       > _espessurac);
19                 std::vector <double> PosicaoAguaInjetada(std::
                       vector <double> _k,double _largura);
20
21                 ~CPosicaoAguaInjetada(){};
22
23 };
24
25 #endif
```

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

Listing 6.20: Arquivo de implementação da classe CPosicaoAguaInjetada.

```
#include "CPosicaoAguaInjetada.h"

using namespace std;

vector <double> CPosicaoAguaInjetada::VazaoInjecao(double _krw,
    double _mw, double _dp, vector <double> _k, double _bw, double
    _comprimento, double _M, vector <double> _posicaoc, vector <
    double> _espessurac)
{

        return VazaoInj;

}

vector <double> CPosicaoAguaInjetada::PosicaoAguaInjetada(vector <
    double> _k,double _largura)
{

        return posicaoc;

}
```

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

Listing 6.21: Arquivo de implementação da classe CPosicaoAguaInjetadaDykstra.

```
#ifndef CPOSICAOAGUAINJETADADYKSTRA_H_
#define CPOSICAOAGUAINJETADADYKSTRA_H_

#include "CPosicaoAguaInjetada.h"

class CPosicaoAguaInjetadaDykstra : CPosicaoAguaInjetada {

        public:

                CPosicaoAguaInjetadaDykstra(){};

                std::vector <double> VazaoInjecao(double _krw,
                   double _mw, double _largura,double _dp, std::
                   vector <double> _k, double _bw, double
                   _comprimento, std::vector <double> _espessurac,
                   double _M);
```

```
13                  std::vector <double> VazaoBt(double _krw,double _mw
                        , double _largura,double _dp, std::vector <
                        double> _k, double _bw, double _comprimento, std
                        ::vector <double> _espessurac, double _M, std::
                        vector<double> posicao);
14                  std::vector <double> PosicaoAguaInjetada(std::
                        vector <double> _k, double _L, double _M);
15                  std::vector <double> EfVert(std::vector<double>
                        _posicaoc, std::vector<double> _espessurac,
                        double _largura, double _espessuraTotal);
16
17                  double vazaoMedia(std::vector<double> _vazaoInj,
                        std::vector<double> _vazaoBt);
18
19                  void EfVertTotal();
20                  double GetEfVertTotal();
21                  std::vector<double> GetPosicao();
22                  void VazaoTotal();
23                  double GetVazaoTotal();
24
25                  ~CPosicaoAguaInjetadaDykstra(){};
26
27};
28
29#endif
```

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

Listing 6.22: Arquivo de implementação da classe CPosicaoAguaInjetadaDykstra.

```
1#include "CPosicaoAguaInjetadaDykstra.h"
2
3using namespace std;
4
5#include <cmath>
6
7
8vector <double> CPosicaoAguaInjetadaDykstra::VazaoInjecao(double
    _krw,double _mw,double _largura,double _dp, std::vector <double>
     _k, double _bw, double _comprimento, std::vector <double>
    _espessurac, double _M)
9{
10
```

```
11          double A;
12
13          for (int i = 0; i < _k.size(); i++)
14          {
15                  A = (_krw*_comprimento*_dp*86400.00*_k[i]*
                        _espessurac[i])/(_bw*_mw*_largura*_M);
16                  VazaoInj.push_back(A);
17          }
18
19          return VazaoInj;
20
21}
22
23vector <double> CPosicaoAguaInjetadaDykstra::VazaoBt(double _krw,
    double _mw,double _largura,double _dp, std::vector <double> _k,
    double _bw, double _comprimento, std::vector <double>
    _espessurac, double _M, vector<double> posicao)
24{
25
26          double A;
27
28                  for (int i = 0; i < _k.size(); i++)
29          {
30                  A = (_krw*_comprimento*_dp*86400.00*_k[i]*
                        _espessurac[i])/(_bw*_mw*(posicao[i] + _M*(
                        _largura - posicao[i] )));
31                  vazaoBt.push_back(A);
32          }
33
34          return vazaoBt;
35}
36
37vector <double> CPosicaoAguaInjetadaDykstra::PosicaoAguaInjetada(
    vector<double> _k, double _L, double _M){
38
39          double A;
40
41          for ( int i = 0; i < _k.size() ; i++)
42          {
43
44                  A = _L*((_M - sqrt(pow(_M, 2) + (1- pow(_M, 2))*(_k
                        [i]/_k[0])))/(_M-1));
```

```
45
46                  posicaoc.push_back(A);
47
48          }
49
50          return posicaoc;
51
52}
53
54std::vector<double> CPosicaoAguaInjetadaDykstra::EfVert(std::vector
    <double> _posicaoc, std::vector<double> _espessurac, double
    _largura, double _espessuraTotal)
55{
56
57     double A;
58     for(int i = 0; i < _espessurac.size(); i++)
59     {
60
61          A = (_posicaoc[i]*_espessurac[i])/(_largura*_espessuraTotal
               );
62          efVert.push_back(A);
63
64     }
65
66     return efVert;
67
68}
69
70double CPosicaoAguaInjetadaDykstra::vazaoMedia(std::vector<double>
    _vazaoInj, std::vector<double> _vazaoBt)
71{
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 6.23: Arquivo de implementação da classe CPosicaoAguaInjetadaStiles.

```
#ifndef CPOSICAOAGUAINJETADASTILES_H_
#define CPOSICAOAGUAINJETADASTILES_H_

#include <vector>

#include "CPosicaoAguaInjetada.h"

class CPosicaoAguaInjetadaStiles : CPosicaoAguaInjetada{

        public:

                CPosicaoAguaInjetadaStiles(){};

                std::vector <double> VazaoInjecao(double _krw,
                   double _mw, double _largura,double _dp, std::
                   vector <double> _k, double _bw, double
                   _comprimento, std::vector <double> _espessurac);
                std::vector <double> PosicaoAguaInjetada(std::
                   vector <double> _k,double _largura);
        std::vector <double> EfVert(std::vector<double> _posicaoc,
           std::vector<double> _espessurac, double _largura, double
            _espessuraTotal);
        void EfVertTotal();
                double GetEfVertTotal();
                std::vector<double> GetPosicao();
                void VazaoTotal();
                double GetVazaoTotal();

                ~CPosicaoAguaInjetadaStiles(){};

};

#endif
```

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

Listing 6.24: Arquivo de implementação da classe CPosicaoAguaInjetadaStiles.

```
#include "CPosicaoAguaInjetadaStiles.h"

#include <iostream>

using namespace std;
```

```
6
7 std::vector <double> CPosicaoAguaInjetadaStiles::VazaoInjecao(
    double _krw,double _mw,double _largura,double _dp, std::vector <
    double> _k, double _bw, double _comprimento, std::vector <double
    > _espessurac)
8 {
9
10         double A;
11
12         for (int i = 0; i < _k.size(); i++)
13         {
14                 A = (_krw*_comprimento*_dp*86400.00*_k[i]*
                      _espessurac[i])/(_bw*_mw*_largura);
15                 VazaoInj.push_back(A);
16         }
17
18         return VazaoInj;
19
20 }
21 std::vector <double> CPosicaoAguaInjetadaStiles::
    PosicaoAguaInjetada(std::vector <double> _k,double _largura)
22 {
23
24         double A;
25         for (int i = 0; i< _k.size(); i++)
26     {
27             A = (_largura*_k[i]/(_k[0]));
28         posicaoc.push_back(A);
29     }
30
31         return posicaoc;
32
33 }
34
35 std::vector<double> CPosicaoAguaInjetadaStiles::EfVert(std::vector<
    double> _posicaoc, std::vector<double> _espessurac, double
    _largura, double _espessuraTotal)
36 {
37
38     double A;
39     for(int i = 0; i < _espessurac.size(); i++)
40     {
```

```
41
42          A = (_posicaoc[i]*_espessurac[i])/(_largura*_espessuraTotal
                );
43          efVert.push_back(A);
44
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 6.25: 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
              _phi, double _swi, double _M): kro(_kro), krw(_krw),
             sor(_sor), phi(_phi), swi(_swi), M(_M){};
15
16                // metodos que mudam os valores das variaveis
                     protegidas
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
                     da classe
26
27                double GetKro();
28                double GetKrw();
29                double GetSor();
30                double GetPHI();
31                double GetSwi();
```

```
32                  double GetM();
33
34                  ~CReservatorio(){};
35
36};
37
38#endif
```

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

Listing 6.26: Arquivo de implementação da classe CReservatorio.

```
1#include "CReservatorio.h"
2
3void CReservatorio::SetKro(double _kro)
4{
5
6         this->kro = _kro;
7
8}
9
10void CReservatorio::SetKrw(double _krw)
11{
12
13         krw = _krw;
14
15}
16
17void CReservatorio::SetSor(double _sor)
18{
19
20         sor = _sor;
21
22}
23
24void CReservatorio::SetPHI(double _phi)
25{
26
27         phi = _phi;
28
29}
30
31void 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 6.27: 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,
                        comprimento, espessuraTotal;
14
15                  std::vector <double> espessurac, k;
16
17
18
19          public:
20
21                  CReservatorioCamadas(){};
22
23                  CReservatorioCamadas(double _bo, double _bw, double
                        _dp, double _mo, double _mw, double _sg, double
                        _largura, double _comprimento, CReservatorio
                       reservatorio):bo(_bo), bw(_bw), dp(_dp), mo(_mo)
```

```
                    , mw(_mw), sg(_sg), largura(_largura),
                    comprimento(_comprimento){ M = (reservatorio.
                    GetKrw()/mw)/(reservatorio.GetKro()/mo);};
24
25                  //entrada de dados
26
27                  void SetBo(double _bo);
28                  void SetBw(double _bw);
29                  void SetDp(double _dp);
30                  void SetMo(double _mo);
31                  void SetMw(double _mw);
32                  void SetSg(double _sg);
33                  void SetLargura(double _largura);
34                  void SetComprimento(double _comprimento);
35                  void SetEspec(std::vector<double> _espessurac);
36                  void SetK(std::vector<double> _k);
37
38          //metodo
39
40          void CalcEspessT();
41                  void CalcM();
42
43                  //Saida de dados
44
45                  double GetBo();
46                  double GetBw();
47                  double GetDp();
48                  double GetMo();
49                  double GetMw();
50                  double GetSg();
51          double GetM();
52                  double GetLargura();
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 6.28 o arquivo de implementação da classe CReservatorioCamadas.

Listing 6.28: 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()/
               mo);
87
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 6.29: 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
               _q, double _mi, double _kbarra, double _h, double _pi);
49 
50 
51 public:
52 
53     CSolverInfluxo(){};
54 
55     void Simular();
56 
57     ~CSolverInfluxo(){};
58 };
59 
60 #endif
```

Apresenta-se na listagem 6.30 o arquivo de implementação da classe CSolverInfluxo.

Listing 6.30: 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 codigo? (1 - sim , 2 - nao): ";
24         char c;
25         cin >> c;
26         cin.get();
27
28         if (c == '1')
29         {
30
31         cout << endl;
32
33         cout << "
               ##################################################################
               " << endl;
34         cout << "#                                                              
               #" << endl;
35         cout << "#                    Projeto de programacao pratica            
               #" << endl;
36         cout << "# Professorr: Andre Duarte Bueno                               
               #" << endl;
37         cout << "#                                                              
               #" << endl;
38         cout << "# Alunos: David Henrique Lima Dias                             
               #" << endl;
39         cout << "#          Julia Rangel Ribeiro                                
               #" << endl;
40         cout << "#          Marcos Vinicius de Paula Chaiben                    
               #" << endl;
41         cout << "#                                                              
```

```
            ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣#" << endl;
42          cout << "
               ##############################################################
               " << endl << endl;
43
44          cout << "Aperte␣ENTER␣para␣continuar␣com␣carregamento␣de␣
               dados..." << endl;
45          cin.get();
46
47          EntradaDadosReservatorio();
48
49          cout << "Aperte␣ENTER␣para␣continuar␣com␣carregamento␣de␣
               dados..." << endl;
50          cin.get();
51
52          EntradaDadosRochaReservatorio();
53
54          cout << "Aperte␣ENTER␣para␣continuar␣com␣carregamento␣de␣
               dados..." << endl;
55          cin.get();
56
57          EntradaDadosCamadas();
58          cout << "
               ##################################################" <<
                endl;
59          cout << "#␣␣Todos␣arquivos␣foram␣carregados␣com␣sucesso!!␣␣
               ␣#" << endl;
60          cout << "
               ##################################################" <<
                endl << endl;
61
62          cout << "#␣qual␣metodo␣deseja␣utilizar?" << endl << endl;
63          cout << "#␣1␣-␣Stiles" << endl;
64          cout << "#␣2␣-␣Dikstra" << endl << endl;
65
66          char handler;
67
68          cin >> handler;
69          cin.get();
70
71          camadas.CalcM();
72              camadas.CalcEspessT();
```

```
73
74          bool tes = false;
75
76          do
77                  {
78
79          switch (handler){
80
81              case '1':
82                  {
83
84                  vector<double> vazao;
85                  vazao = stiles.VazaoInjecao(reservatorio.GetKrw(),
                        camadas.GetMw(), camadas.GetLargura(), camadas.
                        GetDp(), camadas.GetK(), camadas.GetBw(),
                        camadas.GetComprimento(), camadas.GetEspec());
86
87                  vector<double> perm = camadas.GetK();
88
89                  for (int i = 0 ; i< vazao.size() ; i++)
90                          {
91                      cout <<"Vazao Stiles = " << vazao[i] << "\
                            tPermeabilidade = " << perm[i]<<endl;
92                          }
93                  cout << endl;
94
95                  stiles.VazaoTotal();
96                  double vazaototal;
97                  vazaototal = stiles.GetVazaoTotal();
98
99                  cout << "Vazao total = " << vazaototal <<endl;
100
101                 cout << endl;
102
103                         vector<double> posicao;
104                 posicao = stiles.PosicaoAguaInjetada(camadas.GetK()
                        , camadas.GetLargura());
105
106                 for (double item: posicao)
107                     cout <<"Posicao frente de avanço de agua = " <<
                            item <<endl;
108
```

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

```
143                     cout << "\nGostaria de exibir os
                            graficos de linhas
                            equipotenciais? (1 - sim, 2 -
                            nao)" << endl;
144
145                     char d;
146                     cin >> d;
147                     cin.get();
148
149                     if(d == '1')
150                     {
151                             cout << "Aperte ENTER para
                                    continuar com
                                    carregamento de dados...
                                    " << endl;
152                     cin.get();
153
154                             EntradaDadosPoco();
155
156                             double kb = kbarra(perm,
                                    camadas.GetEspec());
157
158                             cout << "# Qual
                                    configuracao de pocos
                                    deseja calcular?" <<
                                    endl;
159                             cout << "# 1 - 1 poco
                                    produtor e 1 injetor" <<
                                     endl;
160                             cout << "# 2 - 2 pocos
                                    injetores e 1 produtor"
                                    << endl;
161                             cout << "# 3 - 1 poco
                                    injetor e 2 produtores"
                                    << endl << endl;
162
163                             char ans;
164                             bool ans1 = false;
165                             cin >> ans;
166                             cin.get();
167
168                             do{
```

```
169                                                 switch(ans){
170                                                         case '1':
171                                                                 {
172                                                                         ans1 = true;
173                                                                         double _r1 = r1(poco.getC(), poco.getX(), poco.getY());
174                                                                         double _r2 =
```

```
                                                                r2
                                                                (
                                                                poco
                                                                .
                                                                getC
                                                                ()
                                                                ,

                                                                poco
                                                                .
                                                                getX
                                                                ()
                                                                ,
                                                                poco
                                                                .
                                                                getY
                                                                ()
                                                                )
                                                                ;
175                                                         linhas

                                                                =

                                                                new

                                                                CLinhaPre
                                                                (
                                                                _r1
                                                                ,

                                                                _r2
                                                                ,

                                                                0.0)
                                                                ;
176                                                         printResults
                                                                (
                                                                poco
                                                                .
                                                                getX
```

```
()
,

poco
.
getY
()
,

poco
.
getC
()
,

stiles
.
GetVazaoT
()
,

camadas
.
GetMo
()
,

kb
,

camadas
.
GetEspess
()
,

poco
.
getPi
()
)
;
```

```
177                                                                 cout << " R = " << linhas -> R ( poco . getC () ) << " C = " << poco . getC ()
```

```
                                                                    <<

                                                                    endl
                                                                    ;
178                                                             xplot

                                                                    =

                                                                    linhas
                                                                    ->
                                                                    CalculoDo
                                                                    (
                                                                    linhas
                                                                    ->
                                                                    R
                                                                    (
                                                                    poco
                                                                    .
                                                                    getC
                                                                    ()
                                                                    )
                                                                    ,

                                                                    poco
                                                                    .
                                                                    getC
                                                                    ()
                                                                    )
                                                                    ;
179                                                             yplot

                                                                    =

                                                                    linhas
                                                                    ->
                                                                    CalculoDo
                                                                    (
                                                                    linhas
                                                                    ->
                                                                    R
```

```
                                                    (
                                                    poco
                                                    .
                                                    getC
                                                    ()
                                                    )
                                                    ,

                                                    poco
                                                    .
                                                    getC
                                                    ()
                                                    )
                                                    ;

180
181                                                 Gnuplot
                                                    ::
                                                    Terminal
                                                    (
                                                    "
                                                    qt
                                                    "
                                                    )
                                                    ;

182                                             plot.
                                                    set_style
                                                    ("lines"
                                                    );
183                                             plot.Title(
                                                    "Curvas␣
                                                    equipotenciais
                                                    .");
184                                             plot.
                                                    unset_legend
                                                    ();
185                                             plot.
                                                    set_xlabel
                                                    ("␣");
186                                             plot.
                                                    set_ylabel
```

```
                                                    ("␣");
187                                             plot.
                                                    ShowOnScreen
                                                    ();
188                                             plot.Grid()
                                                    ;
189                                             plot.
                                                    PlotVector
                                                    (xplot,
                                                    yplot);
190                                             cin.get();
191
192                                             plot.
                                                    savetops
                                                    ("
                                                    dois_pocos_stiles
                                                    ");
193
194
195
196                                                     break
                                                        ;
197                                             }
198                                         case '2':
199                                             {
200
201                                                     ans1
                                                        =
                                                        true
                                                        ;
202                                                     double
                                                        _r1
                                                        =
                                                        r1
                                                        (
```

```
                                                                poco
                                                                .
                                                                getC
                                                                ()
                                                                ,

                                                                poco
                                                                .
                                                                getX
                                                                ()
                                                                ,
                                                                poco
                                                                .
                                                                getY
                                                                ()
                                                                )
                                                                ;
203                                                             double

                                                                _r2

                                                                =

                                                                r2
                                                                (
                                                                poco
                                                                .
                                                                getC
                                                                ()
                                                                ,

                                                                poco
                                                                .
                                                                getX
                                                                ()
                                                                ,
                                                                poco
                                                                .
                                                                getY
                                                                ()
                                                                )
```

```
                ;
204             double _r3 = r3(poco.getX(), poco.getY());
205             linhas = new CLinhaPre(_r1, _r2, _r3);
206             printResults
```

```
(
poco
.
getX
()
,

poco
.
getY
()
,

poco
.
getC
()
,

stiles
.
GetVazaoT
()
,

camadas
.
GetMo
()
,

kb
,

camadas
.
GetEspess
()
,

poco
.
```

```
                                                                getPi
                                                                ()
                                                                )
                                                                ;

207                                                         cout

                                                                <<

                                                                "
                                                                ␣
                                                                R
                                                                ␣
                                                                =
                                                                ␣
                                                                "

                                                                <<

                                                                linhas
                                                                ->
                                                                R
                                                                (
                                                                poco
                                                                .
                                                                getC
                                                                ()
                                                                )

                                                                <<

                                                                "
                                                                ␣
                                                                C
                                                                ␣
                                                                =
                                                                ␣
                                                                "

                                                                <<

                                                                poco
```

```
                                                                .
                                                                getC
                                                                ()

                                                                <<

                                                                endl
                                                                ;
208                                                         xplot

                                                                =

                                                                linhas
                                                                ->
                                                                CalculoDo
                                                                (
                                                                linhas
                                                                ->
                                                                R
                                                                (
                                                                poco
                                                                .
                                                                getC
                                                                ()
                                                                )
                                                                ,

                                                                poco
                                                                .
                                                                getC
                                                                ()
                                                                )
                                                                ;
209                                                         yplot

                                                                =

                                                                linhas
                                                                ->
                                                                CalculoDo
```

```
                                                    (
                                                    linhas
                                                    ->
                                                    R
                                                    (
                                                    poco
                                                    .
                                                    getC
                                                    ()
                                                    )
                                                    ,

                                                    poco
                                                    .
                                                    getC
                                                    ()
                                                    )
                                                    ;

210
211                                                 Gnuplot
                                                    ::
                                                    Terminal
                                                    (
                                                    "
                                                    qt
                                                    "
                                                    )
                                                    ;

212                                             plot1.
                                                    set_style
                                                    ("lines"
                                                    );
213                                             plot1.Title
                                                    ("Curvas
                                                    ␣
                                                    equipotenciais
                                                    .");
214                                             plot1.
                                                    unset_legend
                                                    ();
```

```
215                                                 plot1.
                                                        set_xlabel
                                                        ("␣");
216                                                 plot1.
                                                        set_ylabel
                                                        ("␣");
217                                                 plot1.
                                                        ShowOnScreen
                                                        ();
218                                                 plot1.Grid
                                                        ();
219                                                 plot1.
                                                        PlotVector
                                                        (xplot,
                                                        yplot);
220                                                 cin.get();
221
222                                                 plot1.
                                                        savetops
                                                        ("dois-
                                                        pocos-
                                                        injetores
                                                        -um-poco
                                                        -
                                                        produtor
                                                        -stiles"
                                                        );
223
224
225                                                         break
                                                            ;
226                                                 }
227                                             case '3':
228                                                 {
229
230                                                         ans1
                                                            =
                                                            true
                                                            ;
```

```
231 double _r1 = r1(poco.getC(), poco.getX(), poco.getY());
232 double _r2 = r2(poco.getC(), poco
```

```
                                                                .
                                                                getX
                                                                ()
                                                                ,
                                                                poco
                                                                .
                                                                getY
                                                                ()
                                                                )
                                                                ;
233                                                         double
                                                                _r3
                                                                =
                                                                r3
                                                                (
                                                                poco
                                                                .
                                                                getX
                                                                ()
                                                                ,
                                                                poco
                                                                .
                                                                getY
                                                                ()
                                                                )
                                                                ;
234                                                         linhas
                                                                =
                                                                new
                                                                CLinhaPre
                                                                (
                                                                _r1
                                                                ,
```

```
                                                    _r2
                                                    ,

                                                    _r3
                                                    )
                                                    ;

235                                             printResults
                                                    (
                                                    poco
                                                    .
                                                    getX
                                                    ()
                                                    ,

                                                    poco
                                                    .
                                                    getY
                                                    ()
                                                    ,

                                                    poco
                                                    .
                                                    getC
                                                    ()
                                                    ,

                                                    stiles
                                                    .
                                                    GetVazaoT
                                                    ()
                                                    ,

                                                    camadas
                                                    .
                                                    GetMo
                                                    ()
                                                    ,

                                                    kb
                                                    ,
```

```
                                                                camadas
                                                                .
                                                                GetEspess
                                                                ()
                                                                ,

                                                                poco
                                                                .
                                                                getPi
                                                                ()
                                                                )
                                                                ;

236                                                         cout

                                                                <<

                                                                "
                                                                ␣
                                                                R
                                                                ␣
                                                                =
                                                                ␣
                                                                "

                                                                <<

                                                                linhas
                                                                ->
                                                                R
                                                                (
                                                                poco
                                                                .
                                                                getC
                                                                ()
                                                                )

                                                                <<

                                                                "
                                                                ␣
```

```
                                                                C
                                                                ␣
                                                                =
                                                                ␣
                                                                "
                                                                <<
                                                                poco
                                                                .
                                                                getC
                                                                ()
                                                                <<
                                                                endl
                                                                ;
237                                                         xplot
                                                                =
                                                                linhas
                                                                ->
                                                                CalculoDc
                                                                (
                                                                linhas
                                                                ->
                                                                R
                                                                (
                                                                poco
                                                                .
                                                                getC
                                                                ()
                                                                )
                                                                ,
                                                                poco
                                                                .
                                                                getC
                                                                ()
                                                                )
```

```
                                                                ;
238                                                         yplot
                                                                =
                                                                linhas
                                                                ->
                                                                CalculoDo
                                                                (
                                                                linhas
                                                                ->
                                                                R
                                                                (
                                                                poco
                                                                .
                                                                getC
                                                                ()
                                                                )
                                                                ,
                                                                poco
                                                                .
                                                                getC
                                                                ()
                                                                )
                                                                ;
239
240                                                         Gnuplot
                                                                ::
                                                                Terminal
                                                                (
                                                                "
                                                                qt
                                                                "
                                                                )
                                                                ;
241                                                     plot2.
                                                            set_style
                                                            ("lines"
```

```
                                                    );
242                                                 plot2.Title
                                                    ("Curvas
                                                    ␣
                                                    equipotenciais
                                                    .");
243                                                 plot2.
                                                    unset_legend
                                                    ();
244                                                 plot2.
                                                    set_xlabel
                                                    ("␣");
245                                                 plot2.
                                                    set_ylabel
                                                    ("␣");
246                                                 plot2.
                                                    ShowOnScreen
                                                    ();
247                                                 plot2.Grid
                                                    ();
248                                                 plot2.
                                                    PlotVector
                                                    (xplot,
                                                    yplot);
249                                                 cin.get();
250
251                                                 plot2.
                                                    savetops
                                                    ("um-
                                                    poco-
                                                    injetor-
                                                    dois-
                                                    pocos-
                                                    produtores
                                                    -stiles"
                                                    );
252
253
254                                                     break
                                                        ;
255                                                 }
```

```
256                                                 default:
257                                                         cout << "
                                                            Opcao␣
                                                            invalidade
                                                            !!!" <<
                                                            endl <<
                                                            endl;

258                                                         cout << "#␣
                                                            Qual␣
                                                            configuração
                                                            ␣de␣
                                                            pocos␣
                                                            deseja␣
                                                            calcular
                                                            ?" <<
                                                            endl;
259                                                         cout << "#␣
                                                            1␣-␣1␣
                                                            poco␣
                                                            produtor
                                                            ␣e␣1␣
                                                            injetor"
                                                             << endl
                                                            ;
260                                                         cout << "#␣
                                                            2␣-␣2␣
                                                            pocos␣
                                                            injetores
                                                            ␣e␣1␣
                                                            produtor
                                                            " <<
                                                            endl;
261                                                         cout << "#␣
                                                            3␣-␣1␣
                                                            poco␣
                                                            injetor␣
                                                            e␣2␣
                                                            produtores
                                                            " <<
                                                            endl <<
                                                            endl;
```

```
262                                                     cin >> ans;
263                                                     cin.get();
264                                             }
265                                             }while(!ans1);
266
267                                             ecpot = true;
268                                     }else if( d =='2'){
269                                             ecpot = true;
270                                     }else {
271                                             cout << "opcao invalida!!!"
                                                    << endl;
272                                     }
273
274                             }while(!ecpot);
275                             tes = true;
276                     break;
277             }
278             case '2':
279             {
280
281                     vector<double> vazao;
282                     vazao = dykstra.VazaoInjecao(reservatorio.GetKrw(),
                            camadas.GetMw(), camadas.GetLargura(), camadas.
                           GetDp(), camadas.GetK(), camadas.GetBw(),
                           camadas.GetComprimento(), camadas.GetEspec(),
                           reservatorio.GetM());
283
284                     vector<double> perm = camadas.GetK();
285
286                     for (int i = 0 ; i< vazao.size() ; i++)
287                             {
288                         cout <<"Vazao Dykstra = " << vazao[i] << "\
                               tPermeabilidade = " << perm[i]<<endl;
289                             }
290                     cout << endl;
291
292                     dykstra.VazaoTotal();
293                     double vazaototal;
294                     vazaototal = dykstra.GetVazaoTotal();
295
296                     cout << "Vazao total = " << vazaototal <<endl;
297
```

```
298                 cout << endl;
299 
300                         vector<double> posicao;
301                 posicao = dykstra.PosicaoAguaInjetada(camadas.GetK
                       (), camadas.GetLargura(), reservatorio.GetM());
302 
303                 for (double item: posicao)
304                     cout <<"Posicao frente de avanço de agua = " <<
                           item <<endl;
305 
306                 cout << endl;
307 
308                 double eff;
309                 vector<double> effVet;
310 
311                 effVet = dykstra.EfVert(dykstra.GetPosicao(),
                       camadas.GetEspec(), camadas.GetLargura(),
                       camadas.GetEspessuraTotal());
312                 dykstra.EfVertTotal();
313                 eff = dykstra.GetEfVertTotal();
314 
315                 for (double item: effVet)
316                     cout <<"Eficiencia vertical = " << item <<endl;
317 
318                 cout << endl;
319                 cout << "Eficiencia vertical total = " << eff <<
                       endl;
320 
321                 double oleoProduzidoTotal;
322                 oleo.OleoProduzido(camadas.GetLargura(), camadas.
                       GetComprimento(), reservatorio.GetPHI(), camadas
                       .GetEspessuraTotal(), reservatorio.GetSor(),
                       reservatorio.GetSwi(), dykstra.GetEfVertTotal(),
                        camadas.GetBo());
323                 oleoProduzidoTotal = oleo.getNp();
324 
325                 cout << endl;
326                 cout << "Np = " << oleoProduzidoTotal << endl;
327 
328                 double Bt;
329                 oleo.TempoBreakThrough(dykstra.GetVazaoTotal());
330                 Bt = oleo.getTbreak();
```

```
331
332                 cout << endl;
333                 cout << "O tempo de break through foi : " << Bt <<
                        endl;
334
335                 bool ecpot = false;
336
337                         do
338                         {
339
340                                 cout << "\nGostaria de exibir os
                                        graficos de linhas
                                        equipotenciais? (1 - sim, 2 -
                                        nao)" << endl;
341
342                                 char d;
343                                 cin >> d;
344                                 cin.get();
345
346                                 if(d == '1')
347                                 {
348                                         cout << "Aperte ENTER para
                                                continuar com
                                                carregamento de dados...
                                                " << endl;
349                                 cin.get();
350
351                                         EntradaDadosPoco();
352
353                                         double kb = kbarra(perm,
                                                camadas.GetEspec());
354
355                                         cout << "# Qual
                                                configuracao de pocos
                                                deseja calcular?" <<
                                                endl;
356                                         cout << "# 1 - 1 poco
                                                produtor e 1 injetor" <<
                                                 endl;
357                                         cout << "# 2 - 2 pocos
                                                injetores e 1 produtor"
                                                << endl;
```

```
358                                 cout << "# 3 - 1 poco
                                       injetor e 2 produtores"
                                       << endl << endl;
359
360                                 char ans;
361                                 bool ans1 = false;
362                                 cin >> ans;
363                                 cin.get();
364
365                                 do{
366                                 switch(ans){
367                                         case '1':
368                                                 {
369                                                         ans1 = true;
370                                                         double _r1 = r1(poco.getC(), poco.getX(), poco.getY
```

```
                                                                    ()
                                                                    )
                                                                    ;
371                                                         double
                                                                    _r2
                                                                    =
                                                                    r2
                                                                    (
                                                                    poco
                                                                    .
                                                                    getC
                                                                    ()
                                                                    ,
                                                                    poco
                                                                    .
                                                                    getX
                                                                    ()
                                                                    ,
                                                                    poco
                                                                    .
                                                                    getY
                                                                    ()
                                                                    )
                                                                    ;
372                                                         linhas
                                                                    =
                                                                    new
                                                                    CLinhaPre
                                                                    (
                                                                    _r1
                                                                    ,
                                                                    _r2
```

```
                                                                ,

                                                                0.0)
                                                                ;

373                                                         printResults
                                                                (
                                                                poco
                                                                .
                                                                getX
                                                                ()
                                                                ,

                                                                poco
                                                                .
                                                                getY
                                                                ()
                                                                ,

                                                                poco
                                                                .
                                                                getC
                                                                ()
                                                                ,

                                                                dykstra
                                                                .
                                                                GetVazaoT
                                                                ()
                                                                ,

                                                                camadas
                                                                .
                                                                GetMo
                                                                ()
                                                                ,

                                                                kb
                                                                ,

                                                                camadas
                                                                .
```

```
                                                                GetEspess
                                                                ()
                                                                ,

                                                                poco
                                                                .
                                                                getPi
                                                                ()
                                                                )
                                                                ;

374                                                         cout

                                                                <<

                                                                "
                                                                ␣
                                                                R
                                                                ␣
                                                                =
                                                                ␣
                                                                "

                                                                <<

                                                                linhas
                                                                ->
                                                                R
                                                                (
                                                                poco
                                                                .
                                                                getC
                                                                ()
                                                                )

                                                                <<

                                                                "
                                                                ␣
                                                                C
                                                                ␣
                                                                =
```

```
                                                                    "
                                                                    <<
                                                                    poco
                                                                    .
                                                                    getC
                                                                    ()
                                                                    <<
                                                                    endl
                                                                    ;
375                                                                 xplot
                                                                    =
                                                                    linhas
                                                                    ->
                                                                    CalculoDo
                                                                    (
                                                                    linhas
                                                                    ->
                                                                    R
                                                                    (
                                                                    poco
                                                                    .
                                                                    getC
                                                                    ()
                                                                    )
                                                                    ,
                                                                    poco
                                                                    .
                                                                    getC
                                                                    ()
                                                                    )
                                                                    ;
376                                                                 yplot
```

```
                                                    =
                                                    linhas
                                                    ->
                                                    CalculoDo
                                                    (
                                                    linhas
                                                    ->
                                                    R
                                                    (
                                                    poco
                                                    .
                                                    getC
                                                    ()
                                                    )
                                                    ,
                                                    poco
                                                    .
                                                    getC
                                                    ()
                                                    )
                                                    ;
377
378                                                 Gnuplot
                                                    ::
                                                    Terminal
                                                    (
                                                    "
                                                    qt
                                                    "
                                                    )
                                                    ;
379                                             plot.
                                                    set_style
                                                    ("lines"
                                                    );
380                                             plot.Title(
                                                    "Curvas␣
```

```
                                                    equipotenciais
                                                    .");
381                                             plot.
                                                    unset_legend
                                                    ();
382                                             plot.
                                                    set_xlabel
                                                    ("␣");
383                                             plot.
                                                    set_ylabel
                                                    ("␣");
384                                             plot.
                                                    ShowOnScreen
                                                    ();
385                                             plot.Grid()
                                                    ;
386                                             plot.
                                                    PlotVector
                                                    (xplot,
                                                    yplot);
387                                             cin.get();
388
389                                             plot.
                                                    savetops
                                                    ("dois-
                                                    pocos-
                                                    dykstra"
                                                    );
390
391                                                 break
                                                        ;

392                                             }
393                                         case '2':
394                                             {
395
396                                                 ans1
                                                        =
                                                    true
                                                    ;
```

```
397                                                                 double _r1 = r1(poco.getC(), poco.getX(), poco.getY());
398                                                                 double _r2 = r2(poco.getC(), poco
```

```
                                                                    .
                                                                    getX
                                                                    ()
                                                                    ,
                                                                    poco
                                                                    .
                                                                    getY
                                                                    ()
                                                                    )
                                                                    ;
399                                                         double
                                                                    _r3
                                                                    =
                                                                    r3
                                                                    (
                                                                    poco
                                                                    .
                                                                    getX
                                                                    ()
                                                                    ,
                                                                    poco
                                                                    .
                                                                    getY
                                                                    ()
                                                                    )
                                                                    ;
400                                                         linhas
                                                                    =
                                                                    new
                                                                    CLinhaPre
                                                                    (
                                                                    _r1
                                                                    ,
```

```
                                                    _r2
                                                    ,

                                                    _r3
                                                    )
                                                    ;

401                                             printResults
                                                    (
                                                    poco
                                                    .
                                                    getX
                                                    ()
                                                    ,

                                                    poco
                                                    .
                                                    getY
                                                    ()
                                                    ,

                                                    poco
                                                    .
                                                    getC
                                                    ()
                                                    ,

                                                    dykstra
                                                    .
                                                    GetVazaoT
                                                    ()
                                                    ,

                                                    camadas
                                                    .
                                                    GetMo
                                                    ()
                                                    ,

                                                    kb
                                                    ,
```

```
                camadas
                .
                GetEspess
                ()
                ,

                poco
                .
                getPi
                ()
                )
                ;

402         cout

                <<

                "
                ␣
                R
                ␣
                =
                ␣
                "

                <<

                linhas
                ->
                R
                (
                poco
                .
                getC
                ()
                )

                <<

                "
                ␣
```

```
                                                                C
                                                                ␣
                                                                =
                                                                ␣
                                                                "
                                                                <<
                                                                poco
                                                                .
                                                                getC
                                                                ()
                                                                <<
                                                                endl
                                                                ;
403                                                         xplot
                                                                =
                                                                linhas
                                                                ->
                                                                CalculoDc
                                                                (
                                                                linhas
                                                                ->
                                                                R
                                                                (
                                                                poco
                                                                .
                                                                getC
                                                                ()
                                                                )
                                                                ,
                                                                poco
                                                                .
                                                                getC
                                                                ()
                                                                )
```

```
                                                    ;

404                                             yplot
                                                    =
                                                    linhas
                                                    ->
                                                    CalculoDo
                                                    (
                                                    linhas
                                                    ->
                                                    R
                                                    (
                                                    poco
                                                    .
                                                    getC
                                                    ()
                                                    )
                                                    ,
                                                    poco
                                                    .
                                                    getC
                                                    ()
                                                    )
                                                    ;
405
406                                             Gnuplot
                                                    ::
                                                    Terminal
                                                    (
                                                    "
                                                    qt
                                                    "
                                                    )
                                                    ;
407                                         plot1.
                                                set_style
                                                ("lines"
```

```
                                                    );
408                                                 plot1.Title
                                                    ("Curvas
                                                    ␣
                                                    equipotenciais
                                                    .");
409                                                 plot1.
                                                    unset_legend
                                                    ();
410                                                 plot1.
                                                    set_xlabel
                                                    ("␣");
411                                                 plot1.
                                                    set_ylabel
                                                    ("␣");
412                                                 plot1.
                                                    ShowOnScreen
                                                    ();
413                                                 plot1.Grid
                                                    ();
414                                                 plot1.
                                                    PlotVector
                                                    (xplot,
                                                    yplot);
415                                                 cin.get();
416
417                                                 plot1.
                                                    savetops
                                                    ("dois-
                                                    pocos-
                                                    injetores
                                                    -um-poco
                                                    -
                                                    produtor
                                                    -dykstra
                                                    ");
418
419                                                         break
                                                            ;

420                                                 }
421                                             case '3':
```

```
422                                                                 {
423
424                                                                     ans1 = true;
425                                                                     double _r1 = r1(poco.getC(), poco.getX(), poco.getY());
426                                                                     double _r2 = r2
```

```
                                                                    (
                                                                    poco
                                                                    .
                                                                    getC
                                                                    ()
                                                                    ,
                                                                    poco
                                                                    .
                                                                    getX
                                                                    ()
                                                                    ,
                                                                    poco
                                                                    .
                                                                    getY
                                                                    ()
                                                                    )
                                                                    ;
427                                                                 double
                                                                    _r3
                                                                    =
                                                                    r3
                                                                    (
                                                                    poco
                                                                    .
                                                                    getX
                                                                    ()
                                                                    ,
                                                                    poco
                                                                    .
                                                                    getY
                                                                    ()
                                                                    )
                                                                    ;
428                                                                 linhas
```

```
=
new
CLinhaPre
(
_r1
,
_r2
,
_r3
)
;
printResults
(
poco
.
getX
()
,
poco
.
getY
()
,
poco
.
getC
()
,
dykstra
.
GetVazaoT
()
,
```

```
                                                                camadas
                                                                .
                                                                GetMo
                                                                ()
                                                                ,

                                                                kb
                                                                ,

                                                                camadas
                                                                .
                                                                GetEspess
                                                                ()
                                                                ,

                                                                poco
                                                                .
                                                                getPi
                                                                ()
                                                                )
                                                                ;

430                                                         cout

                                                                <<

                                                                "
                                                                ␣
                                                                R
                                                                ␣
                                                                =
                                                                ␣
                                                                "

                                                                <<

                                                                linhas
                                                                ->
                                                                R
                                                                (
                                                                poco
                                                                .
```

```
                                                getC
                                                ()
                                                )

                                                <<

                                                "
                                                ␣
                                                C
                                                ␣
                                                =
                                                ␣
                                                "

                                                <<

                                                poco
                                                .
                                                getC
                                                ()

                                                <<

                                                endl
                                                ;

431                                         xplot

                                                =

                                                linhas
                                                ->
                                                CalculoDo
                                                (
                                                linhas
                                                ->
                                                R
                                                (
                                                poco
                                                .
                                                getC
                                                ()
```

```
                )
                ,

                poco
                .
                getC
                ()
                )
                ;

432         yplot

                =

                linhas
                ->
                CalculoDo
                (
                linhas
                ->
                R
                (
                poco
                .
                getC
                ()
                )
                ,

                poco
                .
                getC
                ()
                )
                ;

433
434         Gnuplot
                ::
                Terminal
                (
                "
```

```
                                                qt
                                                "
                                                )
                                                ;

435                                     plot2.
                                            set_style
                                            ("lines"
                                            );
436                                     plot2.Title
                                            ("Curvas
                                            ␣
                                            equipotenciais
                                            .");
437                                     plot2.
                                            unset_legend
                                            ();
438                                     plot2.
                                            set_xlabel
                                            ("␣");
439                                     plot2.
                                            set_ylabel
                                            ("␣");
440                                     plot2.
                                            ShowOnScreen
                                            ();
441                                     plot2.Grid
                                            ();
442                                     plot2.
                                            PlotVector
                                            (xplot,
                                            yplot);
443                                     cin.get();
444
445                                     plot2.
                                            savetops
                                            ("um-
                                            poco-
                                            injetor-
                                            dois-
                                            pocos-
                                            produtores
```

```
                                                    -dykstra
                                                    ");
446
447                                                      break
                                                            ;

448                                                 }
449                                             default:
450                                                 cout << "
                                                    Opcao␣
                                                    invalidade
                                                    !!!" <<
                                                    endl <<
                                                    endl;

451                                                 cout << "#␣
                                                    Qual␣
                                                    configuração
                                                    ␣de␣
                                                    pocos␣
                                                    deseja␣
                                                    calcular
                                                    ?" <<
                                                    endl;
452                                                 cout << "#␣
                                                    1␣-␣1␣
                                                    poco␣
                                                    produtor
                                                    ␣e␣1␣
                                                    injetor"
                                                     << endl
                                                    ;
453                                                 cout << "#␣
                                                    2␣-␣2␣
                                                    pocos␣
                                                    injetores
                                                    ␣e␣1␣
                                                    produtor
                                                    " <<
                                                    endl;
454                                                 cout << "#␣
                                                    3␣-␣1␣
```

```
                                                                poco␣
                                                                injetor␣
                                                                e␣2␣
                                                                produtores
                                                                " <<
                                                                endl <<
                                                                endl;
455                                                         cin >> ans;
456                                                         cin.get();
457                                             }
458                                             }while(!ans1);
459
460                                             ecpot = true;
461                                     }else if( d =='2'){
462                                             ecpot = true;
463                                     }else {
464                                             cout << "opcao␣invalida!!!"
                                                    << endl;
465                                     }
466
467                             }while(!ecpot);
468
469                             tes = true;
470                             break;
471                     }
472                     default:
473                             {
474                                     cout << "Opcao␣invalida!!" << endl
                                        << endl;
475                                     cout << "#␣qual␣metodo␣deseja␣
                                        utilizar?" << endl << endl;
476                             cout << "#␣1␣-␣Stiles" << endl;
477                             cout << "#␣2␣-␣Dikstra" << endl << endl;
478
479                             cin >> handler;
480                             cin.get();
481
482                             }
483                     }
484                     }while(!tes);
485
486                     corey.setK0rw(reservatorio.GetKrw());
```

```
487                 corey.setK0ro(reservatorio.GetKro());
488                 corey.setNo(2.0);
489                 corey.setNw(2.0);
490 
491                 corey.calcSwn(reservatorio.GetSwi(), reservatorio.
                       GetSor());
492                 corey.calcKro(corey.getSw());
493                 corey.calcKrw(corey.getSw());
494 
495                 cout << "# Gostaria de plotar os graficos de
                       permeabilidade relativa de corey-brooks?" <<
                       endl;
496                 cout << "# 1 - sim, 2 - nao " << endl << endl;
497 
498                 char e;
499 
500                 cin >> e;
501                 cin.get();
502 
503                 bool tes1 = false;
504 
505                 do
506                 {
507 
508                         if (e == '1'){
509 
510                                 vector<double> _sw, _krw, _kro;
511 
512                                 _sw = corey.getSw();
513                                 _krw = corey.getKrw();
514                                 _kro = corey.getKro();
515 
516                                 Gnuplot::Terminal("qt");
517                                 plot3.set_style("lines");
518                                 plot3.Title("Modelo de
                                       permeabilidade relativa de Corey
                                       -Brooks.");
519                                 plot3.set_legend("inside center top
                                       box");
520                                 plot3.set_xlabel("Swn(saturação
                                       normalizada)");
521                                 plot3.set_ylabel("Kr");
```

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

```
557
558                 do
559                 {
560
561                         if (f == '1'){
562
563                                 fluxofracionario.
                                    calcFluxoFracionarioAgua(corey.
                                    getKrw(), corey.getKro(),
                                    camadas.GetMw(), camadas.GetMo()
                                    );
564                                 fluxofracionario.
                                    calcFluxoFracionarioOleo(corey.
                                    getKrw(), corey.getKro(),
                                    camadas.GetMw(), camadas.GetMo()
                                    );
565
566                                 vector<double> _fro, _frw, _sw;
567
568                                 _sw = corey.getSw();
569                                 _fro = fluxofracionario.
                                    getFluxoFracionarioOleo();
570                                 _frw = fluxofracionario.
                                    getFluxoFracionarioAgua();
571
572                                 Gnuplot::Terminal("qt");
573                                 plot4.set_style("lines");
574                                 plot4.Title("Fluxo␣fracionario␣de␣
                                    agua.");
575                                 plot4.set_legend("inside␣center␣top
                                    ␣box");
576                                 plot4.set_xlabel("Swn(saturação␣
                                    normalizada)");
577                                 plot4.set_ylabel("Frw");
578                                 plot4.ShowOnScreen();
579                                 plot4.Grid();
580                                 plot4.PlotVector(_sw, _frw, "Krw");
581                                 cin.get();
582
583                                 plot4.savetops("
                                    Fluxo_fracionario_de_agua");
584
```

```
585                                 Gnuplot::Terminal("qt");
586                                 plot5.set_style("lines");
587                                 plot5.Title("Fluxo␣fracionario␣de␣
                                        oleo");
588                                 plot5.set_legend("inside␣center␣top
                                        ␣box");
589                                 plot5.set_xlabel("Swn(saturação␣
                                        normalizada)");
590                                 plot5.set_ylabel("Fro");
591                                 plot5.ShowOnScreen();
592                                 plot5.Grid();
593                                 plot5.PlotVector(_sw, _fro, "Kro");
594                                 cin.get();
595
596                                 plot5.savetops("Fluxo-fracionario-
                                        de-oleo");
597
598
599                                 tes2 = true;
600                         }else
601                         if(f == '2'){
602                                 tes2 = true;
603                         } else{
604                                 cout << "Opcao␣invalida!!!" << endl
                                         << endl;
605                                 cout << "#␣Gostaria␣de␣plotar␣os␣
                                        graficos␣de␣permeabilidade␣
                                        relativa␣de␣corey-brooks?" <<
                                        endl;
606                                 cout << "#␣1␣-␣sim,␣2␣-␣nao␣" <<
                                        endl << endl;
607
608                                 cin >> e;
609                                 cin.get();
610
611                         }
612
613                 }while(!tes2);
614
615         } else if(c == '2'){
616             cout << endl;
617             cout << "Codigo␣encerrado!!␣" << endl << endl;
```

```
618             rodar = false;
619             teste = true;
620         } else {
621             cout << endl;
622             cout << "Opcao␣invalida!" << endl;
623         }
624         } while (rodar);
625 
626     }
627 
628 
629 }
630 
631 void CSolverInfluxo::EntradaDadosRochaReservatorio(){
632 
633     cout << endl;
634     cout << "#################################################" <<
           endl;
635     cout << "#␣␣␣␣␣␣␣␣␣␣Arquivos␣de␣dados␣da␣rocha␣␣␣␣␣␣␣␣␣␣#" <<
           endl;
636     cout << "#################################################" <<
           endl << endl;
637 
638     string path = "./in";
639     for (const auto & entry : filesystem::directory_iterator(path))
640     cout << "#␣\t" << entry.path() << std::endl;
641 
642     cout << endl;
643 
644     bool sucess = false;
645 
646     do
647     {
648 
649     cout << "Entre␣com␣nome␣do␣arquivo␣com␣dados␣da␣rocha␣
           reservatorio:␣";
650     string tmp;
651 
652     getline(cin, tmp);
653     cout << endl;
654     ifstream in;
655 
```

```
656     in.open(".\\in\\"+tmp);
657 
658     double  Kro, Krw, Sor, phi, swi;
659 
660     in >> Kro >> Krw >> Sor >> phi >> swi;
661 
662     cout << "Kro = " << Kro << " Krw = " << Krw << " Sor = " << Sor
            << " phi = " << phi << " swi = " << swi << endl;
663 
664     cout << endl;
665 
666     bool test = false;
667 
668     while(!test)
669     {
670     char t;
671     cout << "O carregamento foi correto? (1 - sim , 2 - nao): ";
672     cin >> t;
673     cin.get();
674 
675     if (t == '1')
676     {
677         test = true;
678         sucess = true;
679 
680         reservatorio.SetKro(Kro);
681         reservatorio.SetKrw(Krw);
682         reservatorio.SetSor(Sor);
683         reservatorio.SetPHI(phi);
684         reservatorio.SetSwi(swi);
685 
686 
687         cout << endl << endl;
688 
689         cout << "################################"<< endl;
690         cout << "#      Dados salvos!         #"<< endl;
691         cout << "################################"<< endl << endl;
692 
693 
694     } else
695          {
696             if (t == '2')
```

```
697                 test = true;
698             else
699                 cout << "valor invalido!!" << endl;
700         }
701     }
702
703     cout << endl;
704
705     in.close();
706
707     } while (!sucess);
708 }
709
710 void CSolverInfluxo::EntradaDadosReservatorio()
711 {
712
713     cout << endl;
714     cout << "#####################################################" <<
            endl;
715     cout << "#       Arquivos de dados do reservatorio         #" <<
            endl;
716     cout << "#####################################################" <<
            endl << endl;
717
718     string path = "./in";
719     for (const auto & entry : filesystem::directory_iterator(path))
720     cout << "# \t" << entry.path() << std::endl;
721
722     cout << endl;
723
724     bool sucess = false;
725
726     do
727     {
728
729     cout << "Entre com nome do arquivo com dados do reservatorio: "
            ;
730     string tmp;
731
732     getline(cin, tmp);
733     cout << endl;
734     ifstream in;
```

```
735
736     in.open(".\\in\\"+tmp);
737
738     double  L, C, Bo, Bw, Dp, Mo, Mw, Sg, M;
739
740     in >> L >> C >> Bo >> Bw >> Dp >> Mo >> Mw >> Sg  >> M  ;
741
742     cout << "Largura = " << L << " Comprimento = " << C << " Bo = "
            << Bo << " Bw = " << Bw << " deltaP = " << Dp << " Mi_o = "
            << Mo << " Mi_w = " << Mw << " Sg = " << Sg << " M = " << M
            << endl;
743
744     cout << endl;
745
746     bool test = false;
747
748     while(!test)
749     {
750     char t;
751     cout << "O carregamento foi correto? (1 - sim , 2 - nao): ";
752     cin >> t;
753     cin.get();
754
755     if (t == '1')
756     {
757         test = true;
758         sucess = true;
759
760         camadas.SetLargura(L);
761         camadas.SetComprimento(C);
762         camadas.SetBo(Bo);
763         camadas.SetBw(Bw);
764         camadas.SetDp(Dp);
765         camadas.SetMo(Mo);
766         camadas.SetMw(Mw);
767         camadas.SetSg(Sg);
768         reservatorio.SetM(M);
769
770
771         cout << endl << endl;
772
773         cout << "################################"<< endl;
```

```
774         cout << "#␣␣␣␣␣␣Dados␣salvos!␣␣␣␣␣␣␣␣␣#"<< endl;
775         cout << "##############################"<< endl << endl;
776 
777 
778     } else
779          {
780             if (t == '2')
781                 test = true;
782             else
783                 cout << "valor␣invalido!!" << endl;
784         }
785     }
786 
787     cout << endl;
788 
789     in.close();
790 
791     } while (!sucess);
792 
793 }
794 
795 void CSolverInfluxo::EntradaDadosCamadas()
796 {
797 
798     cout << endl;
799     cout << "################################################" <<
           endl;
800     cout << "#␣␣␣␣␣␣␣␣Arquivos␣de␣entradas␣de␣dados␣␣␣␣␣␣␣␣␣#" <<
           endl;
801     cout << "################################################" <<
           endl << endl;
802 
803     string path = "./in";
804     for (const auto & entry : filesystem::directory_iterator(path))
805     cout << "#␣\t" << entry.path() << std::endl;
806 
807     cout << endl;
808 
809     bool sucess = false;
810 
811     do
812     {
```

```
813
814     cout << "Entre com nome do arquivo com dados da camada: ";
815     string tmp;
816
817     getline(cin, tmp);
818     cout << endl;
819     ifstream in;
820
821     in.open(".\\in\\"+tmp);
822
823     double inK, inH;
824     vector<double> K, H;
825
826     while(!in.eof())
827     {
828         char c = in.peek();
829         if (c == '#' || c == '\n') {
830             in.ignore(256, '\n');
831             continue;
832         }
833
834         in >> inK >> inH;
835         cout <<" Permeabilidade: " << inK << " Espessura: " << inH
               << endl;
836         K.push_back(inK);
837         H.push_back(inH);
838     }
839
840     cout << endl;
841
842     bool test = false;
843
844     while(!test)
845     {
846     char t;
847     cout << "O carregamento foi correto? (1 - sim , 2- nao): ";
848     cin >> t;
849     cin.get();
850
851     if (t == '1')
852     {
853         test = true;
```

```
854         sucess = true;
855
856         camadas.SetK(K);
857         camadas.SetEspec(H);
858
859         cout << endl << endl;
860
861         cout << "################################"<< endl;
862         cout << "#␣␣␣␣␣␣␣Dados␣salvos!␣␣␣␣␣␣␣␣␣␣#"<< endl;
863         cout << "################################"<< endl << endl;
864
865
866     } else
867           {
868              if (t == '2')
869                  test = true;
870              else
871                  cout << "valor␣invalido!!" << endl;
872         }
873     }
874
875     cout << endl;
876
877     in.close();
878
879     } while (!sucess);
880
881}
882
883void CSolverInfluxo::EntradaDadosPoco()
884{
885
886     cout << endl;
887     cout << "#################################################" <<
          endl;
888     cout << "#␣␣␣␣␣␣␣␣␣␣␣␣Arquivos␣de␣dados␣do␣poco␣␣␣␣␣␣␣␣␣␣␣#" <<
          endl;
889     cout << "#################################################" <<
          endl << endl;
890
891     string path = "./poco";
892     for (const auto & entry : filesystem::directory_iterator(path))
```

```
893     cout << "#␣\t" << entry.path() << std::endl;
894 
895     cout << endl;
896 
897     bool sucess = false;
898 
899     do
900     {
901 
902     cout << "Entre␣com␣nome␣do␣arquivo␣com␣dados␣do␣poco:␣";
903     string tmp;
904 
905     getline(cin, tmp);
906     cout << endl;
907     ifstream in;
908 
909     in.open(".\\poco\\"+tmp);
910 
911     double  X, Y, C, Pi;
912 
913     in >> X >> Y >> C >> Pi;
914 
915     cout << "X␣=␣" << X << "␣Y␣=␣" << Y << "␣C␣=␣" << C << "␣Pi␣=␣"
            << Pi << endl;
916 
917     cout << endl;
918 
919     bool test = false;
920 
921     while(!test)
922     {
923     char t;
924     cout << "O␣carregamento␣foi␣correto?␣(1␣-␣sim␣,␣2-␣nao):␣";
925     cin >> t;
926     cin.get();
927 
928     if (t == '1')
929     {
930         test = true;
931         sucess = true;
932 
933         poco.setX(X);
```

```
934         poco.setY(Y);
935         poco.setC(C);
936         poco.setPi(Pi);
937 
938 
939         cout << endl << endl;
940 
941         cout << "##############################"<< endl;
942         cout << "#␣␣␣␣␣␣Dados␣salvos!␣␣␣␣␣␣␣␣␣#"<< endl;
943         cout << "##############################"<< endl << endl;
944 
945 
946     } else
947          {
948             if (t == '2')
949                 test = true;
950             else
951                 cout << "valor␣invalido!!" << endl;
952         }
953     }
954 
955     cout << endl;
956 
957     in.close();
958 
959     } while (!sucess);
960 
961 }
962 
963 double CSolverInfluxo::kbarra(vector<double> k, vector<double> h){
964 
965         double kh = 0;
966         double sumh = 0;
967 
968         for (int i = 0; i<k.size(); i++){
969                 kh += k[i]*h[i];
970                 sumh += h[i];
971         }
972 
973         return kh/sumh;
974 
975 }
```

```
976
977 void CSolverInfluxo::printResults(double _x, double _y, double _c,
        double _q, double _mi, double _kbarra, double _h, double _pi) {
978         cout << "AREA␣INVADIDA␣␣PELA␣AGUA␣NO␣INSTANTE␣DO␣
                BREAKTHROUGH␣=␣" << linhas->AreaInvadidaBT(_c) << "\n";
979         cout << "P(" << _x << "," << _y << ")=" << linhas->Pressao(
                _q, _mi, _kbarra, _h, _pi) << endl;
980 }
981
982 double CSolverInfluxo::r1(double _c, double x, double y) {
983         return sqrt((_c - x) * (_c - x) + y * y);
984 }
985
986 double CSolverInfluxo::r2(double _c, double x, double y) {
987         return sqrt((_c + x) * (_c + x) + y * y);
988 }
989
990 double CSolverInfluxo::r3(double x, double y) {
991         return sqrt(x * x + y * y);
992 }
```

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: Teste modelo de Styles e Linhas Equipotencias com Configuração de 2 poços

- Primeio passo é entrar no Arquivo Simulador para rodar o código. O codigo é executado no prompt de comando. Em seguida, é perguntado se deseja executar o programa (Figura 7.1);
- A próxima etapa é o carregamento dos dados do reservatório, das camadas. O programa exige a conferência dos dados informados soliticando as sua confurmação (Figura 7.4 e 7.3)
- Depois de inserir os dados, deve-se escolher qual modelo de fluxo deseja utilizar. Neste caso, vamos testar o modelo de styles (Figura 7.5)
- Uma vez resolvido o método de Styles, é solicitado a escolha da configuração dos poços que deseja. Neste teste, iremos escolher a configuração de 2 poços. Isso irá gerar o gráfico das linhas equipotenciais (Figura 7.6)
- 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 (Figura 7.7)
- Para finalizar, o código pergunta se deseja executar novamente o codigo.

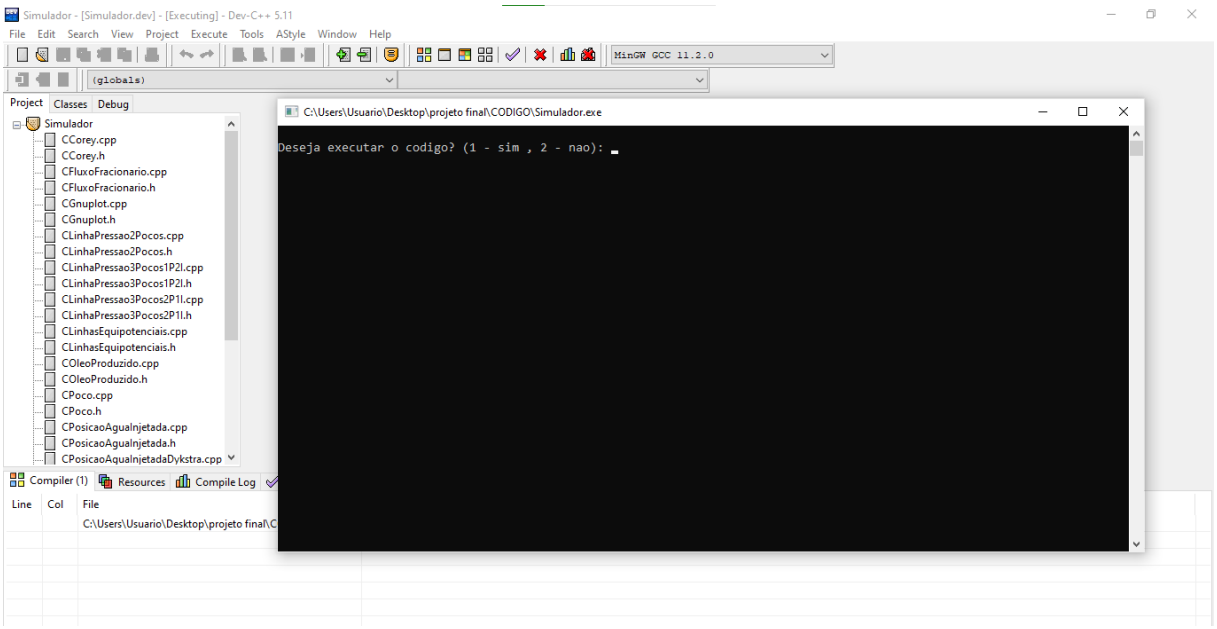


Figura 7.1: Inicialização do programa

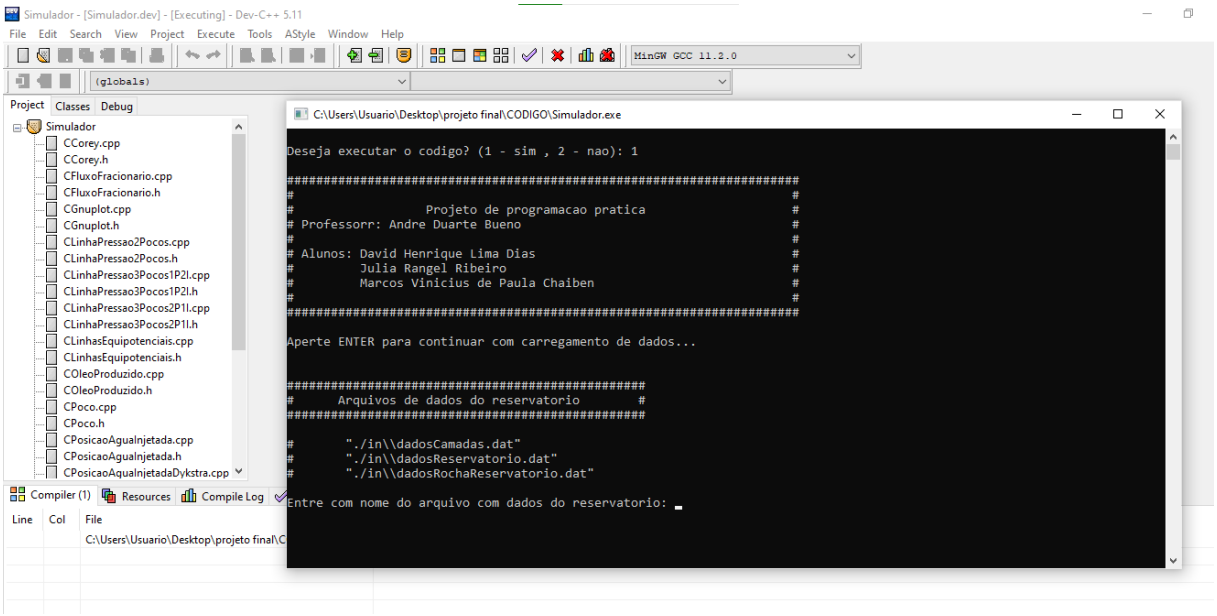


Figura 7.2: Tela do software mostrando o carregamento de dados

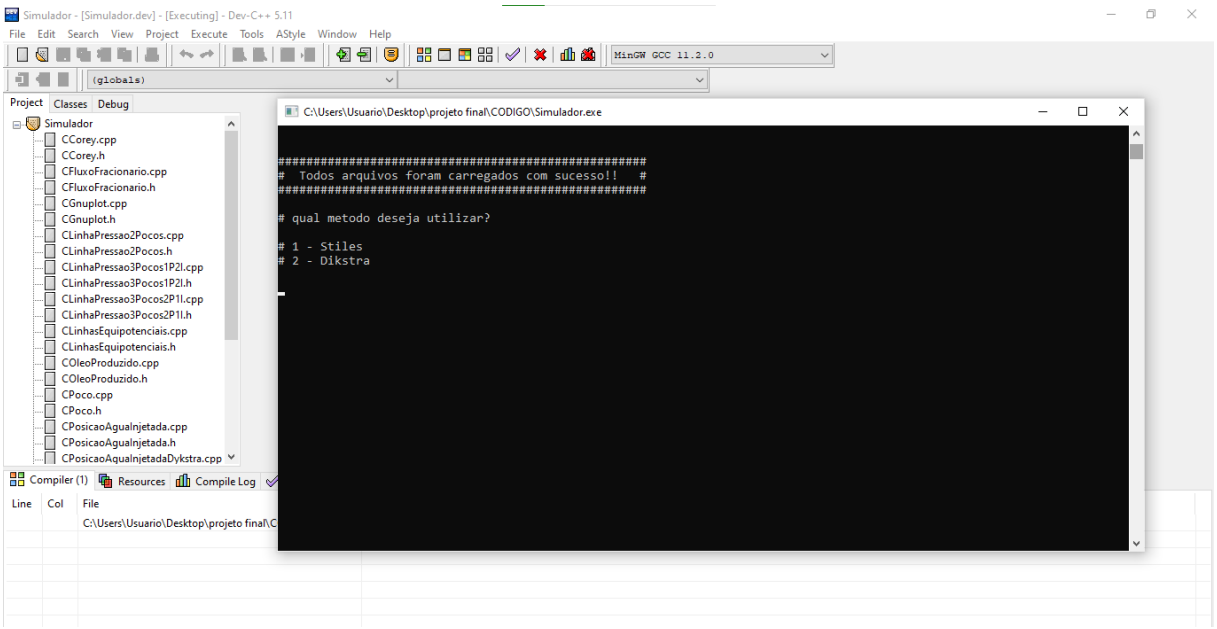


Figura 7.3: Tela do software mostrando a seleção do método

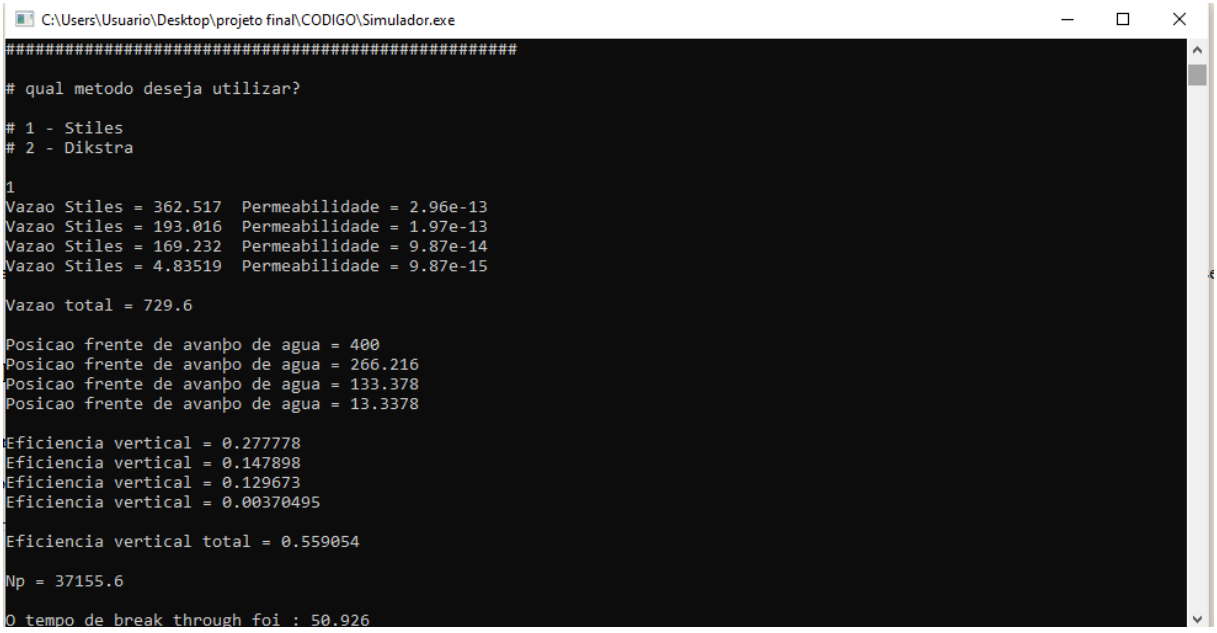


Figura 7.4: Tela do software mostrando os resultados imprimidos

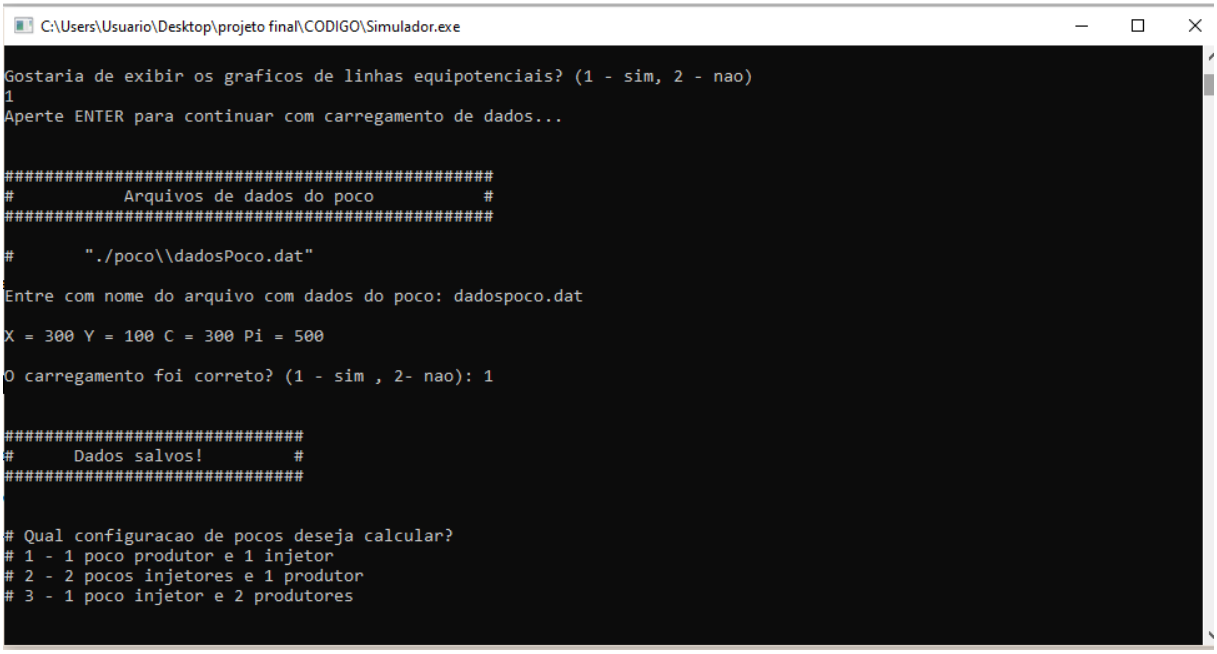


Figura 7.5: Tela do software mostrando a solicitação de qual configuração dos poços deseja utilizar

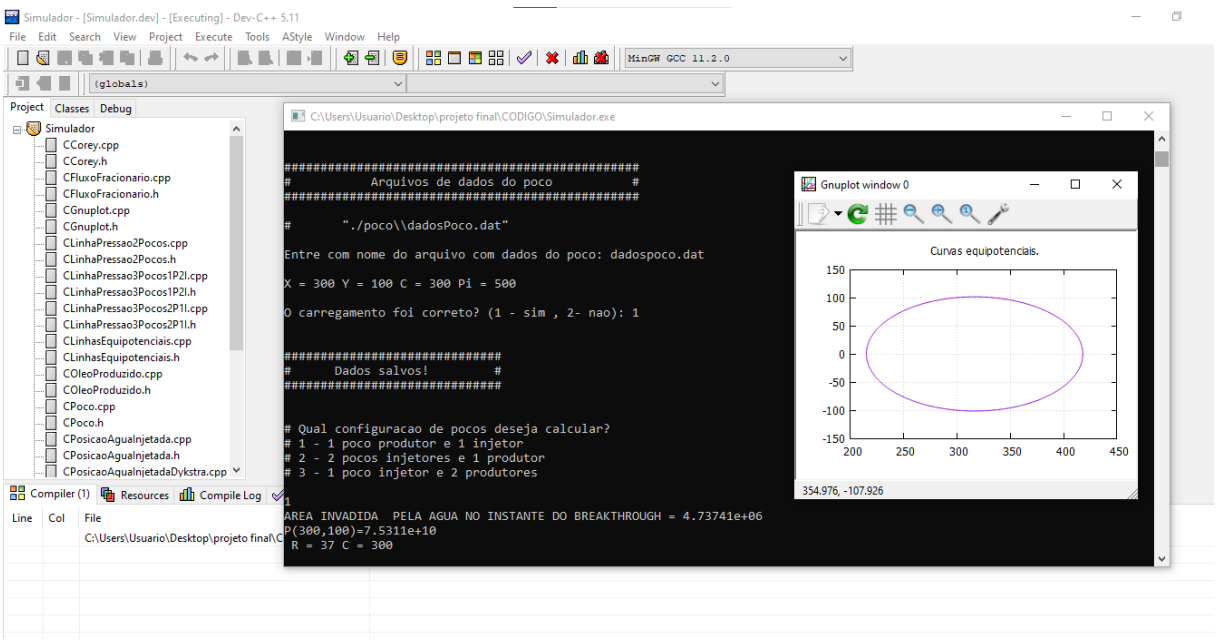


Figura 7.6: Tela do software mostrando os gráficos das linhas equipotenciais e os resultados do valor de pressão e área invatida

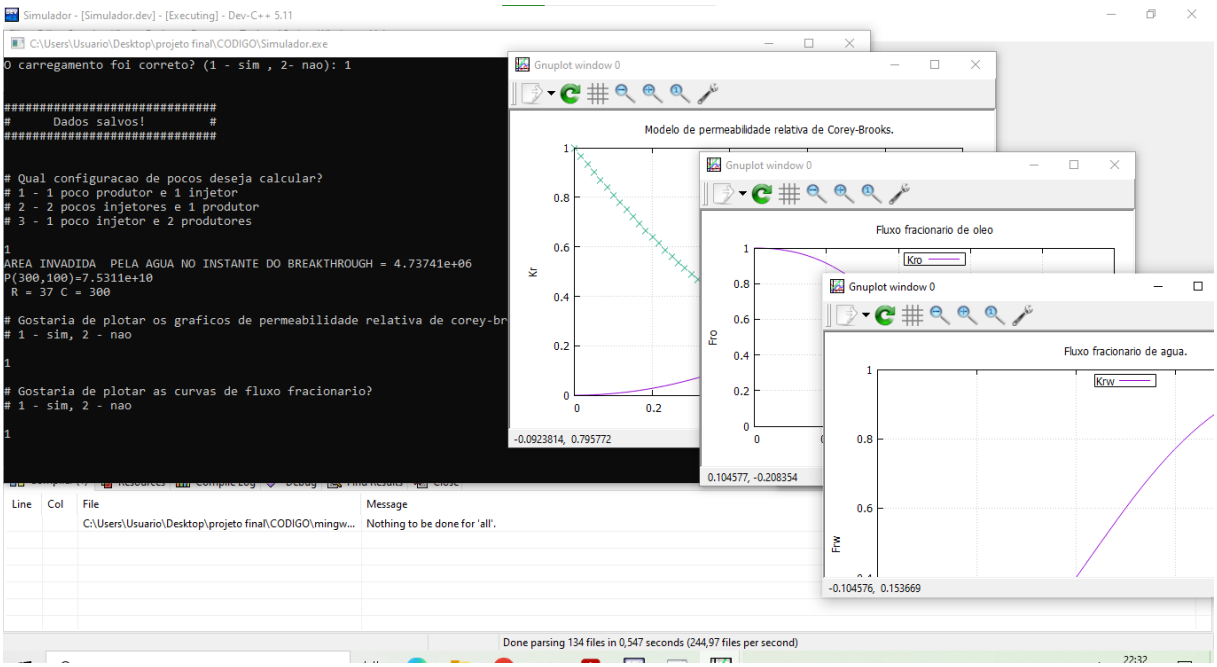


Figura 7.7: Tela do software mostrando os gráficos gerados para o fluxo fracionário e permeabilidade relativa

NOTA : O mesmo teste foi realizado para todos os métodos e todos foram executados com sucesso !!

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 utlizando 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 informções 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 queiram 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 `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 reseservatório e suas unidades de medida precisam ser respeitdas. Se esssas alterações foem 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.

Figura 8.1: Lista de classes Simulador Reservatório - Doxygen

Figura 8.2: Hierarquia de Classes - Doxygen

Figura 8.3: Índice de Componetes - Doxygen

Capítulo 9

Referências

1. BIRD, R; STEWART, W.; LIGHTFOOT, E. Transport Phenomena. Wiley, 1987. (Wi ley 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 Ex ploração de Petróleo e Gás. Data completa 2019. Notas de Aula. Universidade Esta dual do Norte Fluminense.

6. QUEIROZ, W. Elementos de Matemática Aplicada. Curso de Engenharia de Explo ração 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 Mono graph, 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 Techno logy Transfer Council (U.S.).1997.

13. WILLHITE, G. P., Waterflooding, Society of petroleum,1986.