

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

PROJETO ENGENHARIA
DESENVOLVIMENTO DO SOFTWARE
PREVISÃO DE COMPORTAMENTO DE RESERVATÓRIOS DE ÓLEO
COM CAPA DE GÁS
OU GÁS EM SOLUÇÃO E ÓLEO & GÁS COM INFLUXO DE ÁGUA
TRABALHO DA DISCIPLINA PROGRAMAÇÃO PRÁTICA

Versão 1:

Carlos André Martins de Assis e Gabriel Clemente Franklin

Versão 2:

Thiago Couto de Almeida Chaves

Prof. André Duarte Bueno

MACAÉ - RJ

Junho - 2017

Sumário

1	Introdução	1
1.1	Escopo do problema	1
1.2	Objetivos	2
2	Especificação	4
2.1	Especificação do sistema - descrição dos requisitos	4
2.2	Especificação do software - requisitos	5
2.2.1	Nome do sistema/produto	5
2.2.2	Requisitos funcionais	5
2.2.3	Requisitos não funcionais	5
2.3	Casos de uso	5
2.3.1	Diagrama de caso de uso geral	6
2.3.2	Diagrama de caso de uso específico	7
3	Elaboração	8
3.1	Análise de domínio	8
3.2	Formulação teórica	9
3.2.1	Mecanismos de produção de reservatórios	9
3.2.2	Previsão de comportamento dos reservatórios	10
3.2.3	Formulação matemática - reservatórios com influxo de água	11
3.2.4	Formulação Matemática - Reservatórios com Capa de Gás e Gás em Solução	13
3.3	Identificação de pacotes – assuntos	15
3.4	Diagrama de pacotes – assuntos	16
4	AOO – Análise Orientada a Objeto	18
4.1	Diagramas de classes	18
4.1.1	Dicionário de classes	21
4.2	Diagrama de sequência – eventos e mensagens	22
4.2.1	Diagrama de sequência geral	22
4.3	Diagrama de comunicação – colaboração	24
4.4	Diagrama de máquina de estado	25

4.5	Diagrama de atividades	26
5	Projeto	28
5.1	Projeto do sistema	28
5.2	Projeto orientado a objeto – POO	30
5.3	Diagrama de componentes	31
5.4	Diagrama de implantação	34
6	Implementação	35
6.1	Código fonte	35
7	Teste	112
7.1	Teste 1: Tela inicial	112
7.2	Teste 2: Interfaces do programa	112
7.3	Teste 3: Calculando o influxo de água e plotando gráfico associado	112
7.4	Teste 4: Calculando a saturação de óleo, outros parâmetros e plotando gráfico associado	112
7.5	Teste 5: Salvando simulação e saindo do programa	118
8	Documentação	121
8.1	Documentação do usuário	121
8.1.1	Como instalar o software	121
8.1.2	Como rodar o software	121
8.2	Documentação para desenvolvedor	122
8.2.1	Dependências	122
8.2.2	Como gerar a documentação usando doxygen	122

Capítulo 1

Introdução

Este documento contém o projeto e o desenvolvimento de um software orientado a objetos para a *estimativa de parâmetros de reservatórios de óleo com capa de gás (um problema da Engenharia de Reservatórios)*, que irão contribuir para a caracterização e entendimento no estudo deste reservatório. Tais parâmetros ajudam a prever o comportamento futuro deste tipo de reservatório, parâmetros como: saturação de óleo, produção acumulada de HC e influxo de água.

1.1 Escopo do problema

No ambiente da *Engenharia de Reservatórios*, o objeto de estudo é o próprio reservatório de óleo e gás. No entanto, para que esse estudo ocorra de forma eficiente, é necessário que se entenda as características (porosidade, permeabilidade, volume de reservatório, etc.) e o comportamento sob produção. Ter um completo e sólido entendimento de como um reservatório de petróleo é e se comporta é impossível com as nossas atuais tecnologias. No entanto, os engenheiros de reservatório lutam por mais entendimento do comportamento de um reservatório, para que se possa fazer previsões cada vez mais condizentes com as medidas de campo e aumentar a segurança em dizer se um campo é viável ou não à exploração e por quanto tempo esse campo será viável.

O objeto de estudo deste software são os reservatórios de óleo com capa de gás. Tais reservatórios produzem a partir da expansão da capa, decorrente da queda de pressão devido à produção. Quanto mais se produz, mais a capa se expande e mais se produz óleo. Esse processo ocorre até que se comece a produzir uma vazão de gás que é economicamente inviável (tendo, claro, que o óleo sendo mais valioso de se explorar do que o gás, dada a nossa atual ordem geopolítica e econômica).

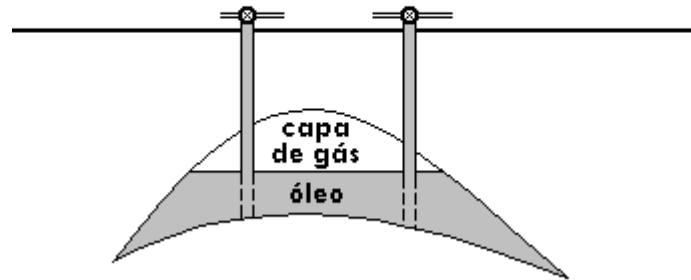


Figura 1.1: Reservatório de óleo com capa de gás

Por causa desse motivo, a previsão de parâmetros como: a produção acumulada de óleo (N_p), produção acumulada de gás (G_p), razão gás/óleo instantânea, etc.; se faz necessária para determinar a viabilidade econômica de um determinado reservatório de óleo que produz pelo mecanismo de capa de gás (viabilidade essa que não será determinada pelo software).

1.2 Objetivos

Os objetivos deste projeto de engenharia são:

- Objetivo geral:
 - Desenvolver um software para determinar os parâmetros comportamentais de um reservatório de óleo com capa de gás para a caracterização desse reservatório por meio da análise e cálculos a partir de dados de produção.
- Objetivos específicos:
 - Aprender e obter experiência em conceber e desenvolver projetos de engenharia com problemas do mundo real.
 - Modelar física e matematicamente o problema.
 - Modelagem estática do software (diagramas de caso de uso, de pacotes, de classes) usando a UML.
 - Modelagem dinâmica do software (desenvolver algoritmos e diagramas exemplificando os fluxos de processamento) usando a UML.
 - Implementar o código em C++.
 - Importar dados de produção a partir de um arquivo de texto.
 - Calcular o influxo de água (v1.0).

- Calcular a pressão no contato (v1.0).
- Calcular a saturação de óleo conforme dados iniciais (pressão, produção acumulada de HC) (v2.0).
- Prever a saturação de óleo para uma determinada pressão, usando o método numérico de Runge-Kutta de 4^a ordem (v2.0).
- Prever a produção de óleo e gás para uma determinada pressão (v2.0).
- Prever a razão gás/óleo instantânea (v2.0).
- Cálculo de vazões de produção e injeção de óleo e gás (v2.0).
- Gerar gráficos com os parâmetros comportamentais (v2.0).
- Realizar teste das classes.
- Encontrar quaisquer bugs.
- Simular problemas reais que se encontram em livros de engenharia de reservatórios e artigos.
- Comparar os resultados e verificar sua autenticidade e precisão.
- Implementar manual simplificado de uso do software.

Capítulo 2

Especificação

Apresenta-se neste capítulo do projeto de engenharia a concepção, a especificação do sistema a ser modelado e desenvolvido.

2.1 Especificação do sistema - descrição dos requisitos

Será desenvolvido (uma nova versão de um antigo) um software em modo texto para a previsão do comportamento de reservatórios que produzem pelos seguintes mecanismos de produção: Gás em Solução (Óleo), Capa de Gás (Óleo), Influxo de Água (Gás/Óleo). Os cálculos serão feitos a partir das propriedades do reservatório e do aquífero e as simulações serão feitas usando modelos como o de Muskat (Capa de Gás) e o de Van-Everdingen & Hurst (influxo de água). Portanto, já que estamos tratando de dezenas de dados de propriedades de fluidos assim como dados de produção, o usuário poderá escolher realizar a entrada de dados por meio de um arquivo de entrada salvo em disco ou entrada direta via teclado. Ao final de cada simulação, o usuário definirá a saída dos dados resultantes da simulação, podendo ser: na tela, no disco ou com um gráfico. Para as saídas gráficas serão apresentados gráficos de influxo de água e produção acumulada em função do tempo para reservatórios com influxo de água e gráficos de saturação de óleo e produção acumulada em função do tempo para reservatórios com capa de gás e gás em solução. Os gráficos serão gerados pelo programa **gnuplot**. Será fornecido um manual de ajuda para guiar aqueles em caso de dúvidas, explicando o funcionamento do software. O seguinte software será um software multi-plataforma desenvolvido para ambos GNU/LINUX e Windows com licença GPL.

2.2 Especificação do software - requisitos

2.2.1 Nome do sistema/produto

Nome	Software para previsão de comportamento de reservatórios que produzem pelo mecanismo de: gás em solução, capa de gás ou influxo de água.
Componentes principais	Sistema para cálculo de parâmetros comportamentais de reservatório.
Missão	Prever o comportamento do reservatório através de cálculos de parâmetros como: Saturação de óleo, influxo de água e produção acumulada de HC.

2.2.2 Requisitos funcionais

Apresenta-se a seguir os requisitos funcionais.

RF-01	O usuário deverá ter liberdade em escolher quais as maneiras em que dar-se-á a entrada de dados.
RF-02	O usuário deverá ter liberdade de escolher quais simulações realizar.
RF-03	Deve permitir a geração de gráficos com o resultado da simulação.
RF-04	Deve mostrar os resultados na tela.
RF-05	Deve salvar os resultados em um arquivo de disco em um diretório de escolha do usuário.

2.2.3 Requisitos não funcionais

RNF-01	O programa deverá ser multi-plataforma, podendo ser executado em <i>Windows</i> , <i>GNU/Linux</i> .
---------------	--

2.3 Casos de uso

Esta seção contém uma tabela que descreve um caso de uso do sistema, assim como os diagramas de caso de uso.

Tabela 2.1: Exemplo de caso de uso.

Nome do caso de uso:	Cálculo da saturação de óleo para uma determinada pressão
Resumo/descrição:	Cálculo da saturação de óleo através de propriedades dos fluídos de reservatório e dados de produção.
Etapas:	<ol style="list-style-type: none"> 1. Entrada de dados. 2. Cálculo da saturação inicial de óleo. 3. Resolução da EDO de Muskat para obter a saturação de óleo. 4. Analisar e validar dados resultantes. 5. Gerar gráficos. 6. Salvar dados em arquivo de texto e gráficos
Cenários alternativos:	Saturação de óleo não contida entre o intervalo $[0,1]$ encerrará a simulação por ser um dado inválido.

2.3.1 Diagrama de caso de uso geral

O diagrama de caso de uso geral da Figura 2.1 mostra o usuário realizando a simulação, isto é, escolhendo qual reservatório simular, entrando com os dados, calculando tais dados e analisando os subseqüentes resultados.

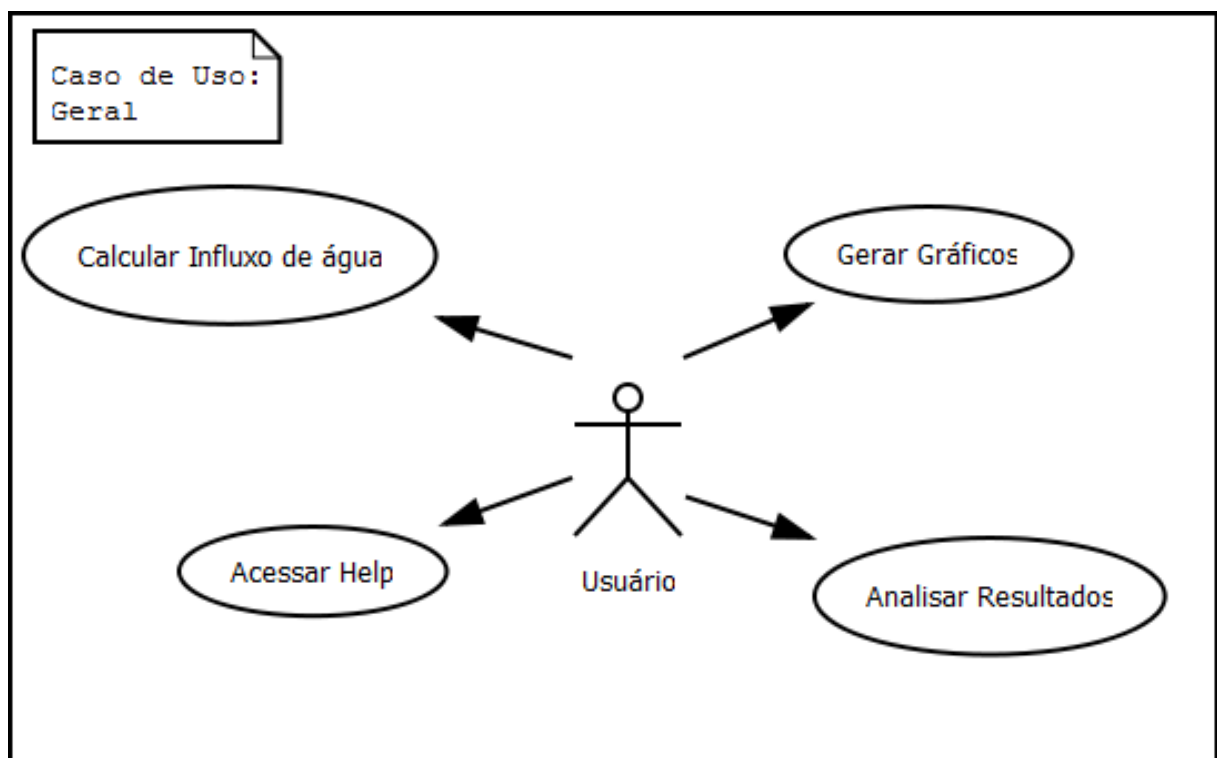


Figura 2.1: Diagrama de caso de uso – Caso de uso geral

2.3.2 Diagrama de caso de uso específico

Os diagramas de caso de uso a seguir detalham o cenário de teste realizado no capítulo 7, o primeiro diagrama contém o diagrama de caso de uso para a versão 1.0 enquanto o segundo diagrama contém para a versão 2.0.

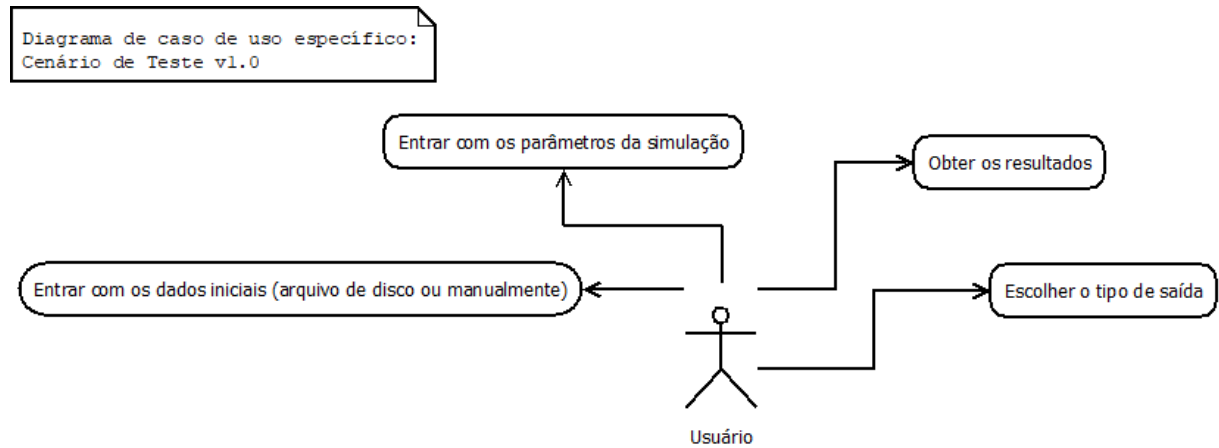


Figura 2.2: Diagrama de caso de uso específico v1.0- “Cenário de teste”.

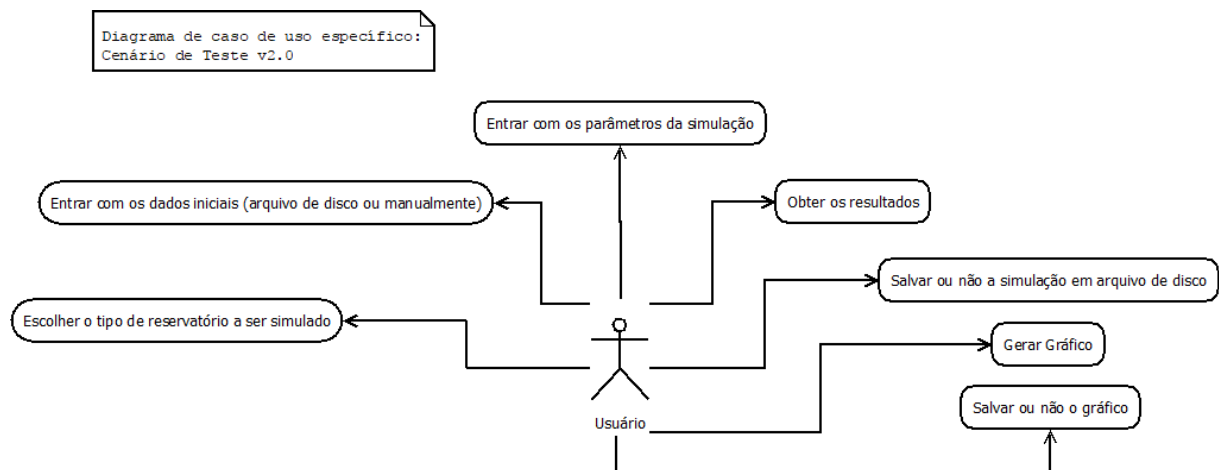


Figura 2.3: Diagrama de caso de uso específico v2.0- “Cenário de teste”.

Capítulo 3

Elaboração

Esta etapa envolve o estudo dos conceitos relacionados ao software, análise de domínio e a identificação dos pacotes para a análise dos requisitos, ajustando os requisitos iniciais de forma a desenvolver um sistema útil, que atenda às necessidades do usuário e, na medida do possível, permita seu reuso e futura extensão.

3.1 Análise de domínio

Depois de muito estudo e reflexão das especificações do sistema, do material acadêmico e do projeto anterior desse software, foi possível identificar as seguintes áreas abordadas pelo programa:

- Engenharia de Reservatórios: é a espinha dorsal no qual esse projeto se sustenta. O software aqui desenvolvido, utiliza conceitos como o de propriedades dos fluídos (B_o, B_g, R_s , etc.), propriedades de rochas (k_g, k_o), a Equação de Balanço (EBM), modelos de Influxo de Água (Fetkovitch e Van Everdingen & Hurst) e mecanismos de produção. O software irá aplicar todos esses conceitos na caracterização adicional do reservatório, fazendo uma predição do comportamento deste ao longo de sua produção.
- Modelagem Numérica Computacional utiliza conceitos matemáticos de Cálculo Numérico. Neste software serão utilizados por exemplo, a Transformada de Laplace, Algoritmo de Stehfest e o Método de Runge-Kutta de 4^a ordem.
- Pacote Gráfico: usar-se-á um pacote gráfico para a geração de gráficos de diferentes parâmetros de reservatórios para que haja uma melhor compreensão do comportamento desse mesmo.
- Software: Serão utilizadas classes e bibliotecas já existentes para a utilização dos modelos de influxo de água.

3.2 Formulação teórica

Nesta seção será discutida a fundamentação teórica do software desenvolvido.

3.2.1 Mecanismos de produção de reservatórios

Um mecanismo de produção de um reservatório é um conjunto de fatores que leva à produção de hidrocarbonetos. Quando esses mecanismos atuam, eles permitem que o óleo se desloque do poro da rocha para o fundo do poço produtor. Por exemplo, quando o mecanismo de influxo de água atua, a invasão da zona de óleo pela água do aquífero empurra o óleo do poro da rocha em direção ao poço produtor, fazendo que haja produção e ao mesmo tempo sustentando a pressão. Em um reservatório, usualmente há mais de um mecanismo de produção. Pode haver duas situações: (1) um mecanismo é preponderante em relação ao outro; (2) dois ou mais mecanismos exercem igual importância na produção de HC. Na maioria das vezes, temos a situação (1) e então, classificamos os reservatórios de acordo com o mecanismo dominante, tais são: influxo de água (Water Driven), gás em solução (Gas Solution), capa de gás (Gas Expansion), compactação de poros (Rock Driven ou Compaction Driven) e segregação gravitacional (Gravity Driven). Quando encontramos a situação (2), damos o nome: mecanismo combinado.

De acordo com Rosa et al. (2006), os principais mecanismos são: Influxo de Água, Gás em Solução e Capa de Gás. Os dois últimos são exclusivos para reservatórios de óleo enquanto o primeiro serve para ambos gás e óleo. Este software fará a previsão de comportamento desses reservatórios mais importantes.

Os reservatórios que produzem pelo influxo de água, requerem que o reservatório esteja em contato direto com uma grande acumulação de água (pelo menos 10x maior que o reservatório) e que tanto o reservatório quanto o aquífero estejam intimamente ligados, isto é, as alterações nas condições do reservatório causam alterações no aquífero e vice-versa. O mecanismo se manifesta assim: Com a produção de HC, o reservatório sente uma queda de pressão que é transmitida ao aquífero. Assim, a diminuição da pressão gerará uma diminuição do espaço poroso e expansão da água que acarretará na invasão da água na zona de óleo (influxo de água), invasão essa que manterá a pressão elevada na zona de óleo e deslocará o óleo/gás para a zona produtora. O índice de recuperação por este tipo de mecanismo varia entre 30 e 40% dependendo do reservatório.

Os reservatórios que produzem pela expansão da capa de gás, a capa localizando-se acima da zona de óleo (por ser menos densa) precisa ter um tamanho considerável (até 10 vezes o tamanho da região de óleo) para que o mecanismo seja atuante. Com a produção de óleo, a pressão decai e com a remoção de fluidos, o gás da capa se expande e vai ocupando os poros que antes tinham óleo. Assim, a pressão é sustentada pela capa. Nesse tipo de mecanismo é necessário um cuidado com a vazão de produção. Se a vazão for muito alta, os efeitos da decompressão não serão sentidos pela capa e ela não atuará,

então o período de surgência do reservatório cairá drasticamente. Este mecanismo de produção contém índice de recuperação entre 20 e 30%.

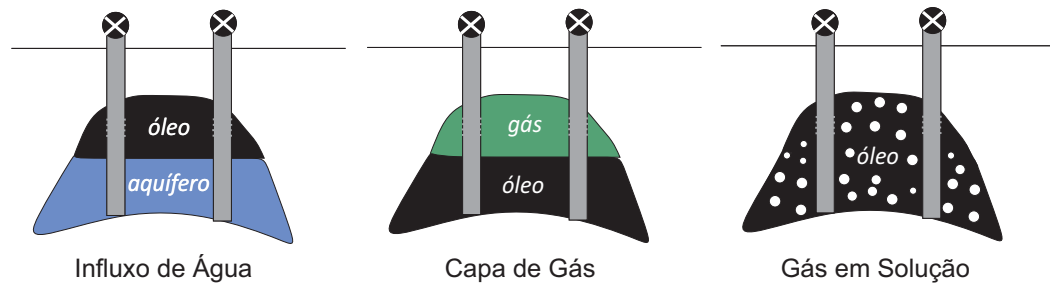


Figura 3.1: Tipos de reservatórios e mecanismos de produção - desenhos esquematizados

Os reservatórios que produzem por gás em solução, precisam que não sejam influenciados pelo ambiente externo. Toda a energia para produção se encontra na zona de óleo. Com a produção, a pressão cai até atingir o ponto de bolha, quando esse mecanismo começa a realmente atuar, assim, as frações mais leves de óleo vão vaporizar e esse gás gerado tem mais facilidade de se locomover no meio poroso que o óleo. Portanto, uma porção de HC com gás em solução tem muito mais facilidade de se locomover até o fundo do poço do que uma porção de HC de óleo puro. Porém, esse método só é benéfico para uma faixa de pressão, quando a pressão continua a cair, o que era antes algumas bolhas de gás, agora formarão uma fase contínua de gás para o fundo do poço, fazendo a RGO (Razão Gás-Óleo) explodir. Este método contém índice de recuperação baixo, inferior a 20%.

3.2.2 Previsão de comportamento dos reservatórios

A previsão de comportamento dos reservatórios ajudam os engenheiros de reservatório a terem uma melhor ideia de como um determinado reservatório se comporta. Saber como o reservatório se comporta (e comportará) é um estudo necessário para a determinação da viabilidade econômica do reservatório. Esse *ramo da engenharia de reservatórios*, tem como objetivo estimar parâmetros do reservatório como as *vazões de óleo, gás e água*, e outros parâmetros secundários como a *saturação de óleo, razão gás-óleo, pressão no contato água-óleo*, etc. Por exemplo, se a simulação resultar em uma razão gás-óleo alta, indica que a capa de gás chegará à área canhoneada e que precisará de uma intervenção para aquela determinada pressão de reservatório. Em termos operacionais, se já se espera uma intervenção futura, a equipe responsável pelo reservatório poderá se adiantar e planejar como se dará essa intervenção, reduzindo custos. Sem falar também, que a RGO poderá nos dizer o quão viável é explorar aquele reservatório.

Para os *reservatórios de óleo com capa de gás ou gás em solução*, é de interesse calcular a *produção acumulada de óleo, produção acumulada de gás, razão gás-óleo e saturação de*

óleo. Já para os reservatórios com influxo de água, é de interesse calcular principalmente o influxo de água e a pressão no contato.

3.2.3 Formulação matemática - reservatórios com influxo de água

O influxo natural de água tanto em reservatórios de óleo como de gás pode ser descrita pela equação de balanço de material do aquífero, independente do tempo (1.1):

$$W_e = c_t W(p_i - p) \quad (3.1)$$

Onde $W_e [m^3]$ é o Influxo natural de água, $W_i [m^3]$ é o volume inicial de água no aquífero, $p_i [kgf/cm^2]$ é a pressão inicial no aquífero/reservatório, $p [kgf/cm^2]$ é a pressão no contato óleo (ou gás) e água, e $c_t [cm^2/kgf]$ é a compressibilidade total do aquífero.

Porém, esta equação é apenas a definição básica de compressibilidade, e só é aplicável em aquíferos muito pequenos. Para aquíferos maiores, um modelo matemático é necessário incluindo a dependência do tempo, levando em conta o tempo finito que o aquífero leva para responder completamente a mudança de pressão no reservatório [5].

Neste trabalho, o cálculo do influxo de água no instante $t_n [ano]$ usando o modelo de van Everdingen & Hurst (1949) (1.2) é:

$$W_{en} = U \sum_{j=0}^{n-1} \Delta p_j W_d(t_{Dn} - t_{Dj}) \quad (3.2)$$

Sendo: $U [m^3/kgf/cm^2] =$ constante de influxo de água do aquífero, dada pela equação (1.3) para um aquífero radial:

$$U = 2\pi f \phi c_t h r_o^2 \quad (3.3)$$

Sendo $f [rad] = \frac{\theta}{2\pi}$, onde θ é o ângulo que representa o contato reservatório/aquífero em radianos, $\phi [adimensional]$ é a porosidade do aquífero, $c_t [cm^2/kgf]$ é a compressibilidade total do aquífero, $h [m]$ é a espessura média e $r_o [m]$ é o raio do reservatório.

Para um aquífero linear, a constante é dada pela equação (1.4):

$$U = w L h \phi c_t \quad (3.4)$$

Sendo $w [m]$ a largura do aquífero e $L [m]$ a largura do mesmo. A variável t_d é o tempo adimensional, dado pela equação (1.5):

$$t_d = \frac{kt}{\phi \mu c_t r_o^2} \quad (3.5)$$

Sendo $k [md]$ a permeabilidade do aquífero, e $\mu [cp]$ a viscosidade da água. Na equação (1.2), $\Delta p_j [kgf/cm^2]$ representa a queda de pressão no contato no tempo $t_j [ano]$, e

$W_d(t_{Dn} - t_{Dj})$ é influxo de água acumulado adimensional, obtido para o tempo adimensional $(t_{Dn} - t_{Dj})$, durante o qual o efeito de queda de pressão é sentido.

Na dedução desta equação, utiliza-se o conceito de transformada de Laplace para resolver a equação da difusividade do sistema aquífero-reservatório. Como as soluções são obtidas analiticamente apenas no campo de Laplace, é necessário um algoritmo de inversão numérica para se obter o comportamento de W_d em função de t_d . Um algoritmo normalmente utilizado para tal inversão é o algoritmo de Stehfest (1970). A seguir soluções para aquífero radial:

$$\bar{W}_d(u) = \frac{k_1(\sqrt{u})}{u^{3/2}k_0(\sqrt{u})} \quad (3.6)$$

Regime transiente

$$\bar{W}_d(u) = \frac{I_0(r_{eD}\sqrt{u})k_1(\sqrt{u}) + I_1(\sqrt{u})k_0(r_{eD}\sqrt{u})}{u^{3/2}[I_0(r_{eD}\sqrt{u})k_0(\sqrt{u}) - I_0(\sqrt{u})k_0(r_{eD}\sqrt{u})]} \quad (3.7)$$

Regime permanente

$$\bar{W}_d(u) = \frac{I_1(r_{eD}\sqrt{u})k_1(\sqrt{u}) - I_1(\sqrt{u})k_1(r_{eD}\sqrt{u})}{u^{3/2}[I_0(\sqrt{u})k_1(r_{eD}\sqrt{u}) + I_1(r_{eD}\sqrt{u})k_0(\sqrt{u})]} \quad (3.8)$$

Regime pseudopermanente

Para aquífero linear temos:

$$W_d(t_d) = 2\sqrt{\frac{t_d}{\pi}} \quad (3.9)$$

Regime transiente

$$\bar{W}_d(u) = \frac{1 + e^{-2\sqrt{u}}}{u^{3/2}[1 - e^{-2\sqrt{u}}]} \quad (3.10)$$

Regime permanente

$$\bar{W}_d(u) = \frac{1 - e^{-2\sqrt{u}}}{u^{3/2}[1 + e^{-2\sqrt{u}}]} \quad (3.11)$$

Regime pseudopermanente

No caso da solução para um aquífero linear em regime permanente, existe uma solução analítica com W_d em função de t_d . Para os outros casos é necessário utilizar o algoritmo de Stehfest. As soluções da Tabela 1.1 são escritas em termos das funções de Bessel modificadas de primeira espécie, I_0 e I_1 , e de segunda espécie, K_0 e K_1 , de ordens zero e um, respectivamente, eu é a variável de Laplace. Na literatura é muito comum a apresentação dos valores de W_d em função de t_d em forma de tabelas, para que não seja necessário o uso do algoritmo. A previsão do comportamento de reservatórios de óleo ou gás com mecanismo de influxo de água depende de duas equações básicas: a equação de influxo de água (1.2) e a equação de balanço de materiais. Estas podem ser resolvidas

simultaneamente, por um processo iterativo, para gerar a pressão no reservatório. Para um reservatório de gás, a equação do balanço de materiais se resume a equação (1.6) :

$$\frac{p}{z} = \frac{p_i}{z_i} \frac{(1 - \frac{G_p}{G})}{(1 - \frac{W_e}{GB_{gi}})} \quad (3.12)$$

Onde $p [kgf/cm^2]$ é a pressão no instante desejado, $p_i [kgf/cm^2]$ é a pressão inicial, $Z [adimensional]$ é o fator de compressibilidade calculado a partir da pressão pseudocrítica, $G_p [m^3]$ é o volume de gás produzido acumulado, $G [m^3]$ é o volume de gás inicial no reservatório, e $B_{gi} [m^3/m^3 std]$ é o fator volume-formação do gás. Para um reservatório de óleo subsaturado, a equação do balanço de materiais é dada pela equação (1.7):

$$W_e = N_p B_o - N B_{oi} c_{eo} \Delta p \quad (3.13)$$

Onde $N_p [m^3]$ é o volume de óleo recuperado acumulado, $N [m^3]$ é o volume de óleo inicial no reservatório, $B_o [m^3/m^3 std]$ é o fator volume-formação do óleo, $\Delta p [kgf/cm^2]$ é a queda de pressão no contato aquífero/reservatório, e $c_{eo} [cm^2/kgf]$ é a compressibilidade efetiva do óleo dada pela equação (1.8):

$$c_{eo} = \frac{c_o S_o + c_w S_w + c_f}{1 - S_w} \quad (3.14)$$

Sendo $c_o [cm^2/kgf]$, $c_w [cm^2/kgf]$, $c_f [cm^2/kgf]$ as compressibilidades do óleo, da água e da formação respectivamente, e $S_o [adimensional]$ e $S_w [adimensional]$ as saturações do óleo e da água respectivamente. Reescrevendo a equação (1.2) na forma da equação (1.6), temos a equação (1.9):

$$W_{en} = U \sum_{j=0}^{n-2} \Delta p_j W_d(t_{Dn} - t_{Dj}) + \frac{U}{2} (p_{n-2} - p_n) W_d(t_{Dn} - t_{Dn-1}) \quad (3.15)$$

Pode-se resolver os conjuntos de equações (1.4) e (1.6) ou (1.5) e (1.6) através de um método iterativo, visto que nestas equações as únicas variáveis desconhecidas são W_{en} e p_n .

3.2.4 Formulação Matemática - Reservatórios com Capa de Gás e Gás em Solução

Primeiramente, devemos definir as variáveis:

$S_o [adimensional]$ - Saturação de Óleo

$S_{wi} [adimensional]$ - Saturação de Água Irredutível

C - Razão de Ciclagem de Gás

$R [m^3 std/m^3 std]$ - Razão Gás-Óleo

$m [adimensional]$ - Razão entre os volumes da capa de gás e da zona de óleo

$\mu_o [cp]$ - Permeabilidade do Óleo

μ_g [cp] - Permeabilidade do Gás

p [kgf/cm²] - Pressão

k_g [md]- Permeabilidade Efetiva ao Gás

k_o [md]- Permeabilidade Efetiva ao Óleo

B_g [m³/m³std] - Fator Volume-Formação do Gás

B_o [m³/m³std]- Fator Volume-Formação do Óleo

R_s [m³std/m³std]- Razão de Solubilidade

O pontapé para a análise do comportamento do reservatório é feito através do cálculo da saturação de óleo através da equação de Muskat (1949):

$$\frac{dS_o}{dp} = \frac{S_o\lambda + (1 - S_o - S_{wi})\xi + S_o\eta(\Psi - \frac{CR}{\alpha}) + m(1 - S_{wi})\xi}{1 + (\frac{\mu_o}{\mu_g})(\Psi - \frac{CR}{\alpha})} \quad (3.16)$$

onde:

$$\eta = \frac{1}{Bo}(\frac{\mu_o}{\mu_g})(\frac{dB_o}{dp}) \quad (3.17)$$

$$\alpha = (\frac{Bo}{Bg})(\frac{\mu_o}{\mu_g}) \quad (3.18)$$

$$\lambda = \frac{Bg}{Bo} \frac{dR_s}{dp} \quad (3.19)$$

$$\Psi = \frac{k_g}{k_o} \quad (3.20)$$

$$\xi = B_g \frac{d}{dp}(\frac{1}{Bg}) \quad (3.21)$$

$$R = (\frac{k_g}{k_o})(\frac{Bo}{Bg})(\frac{\mu_o}{\mu_g}) + R_s \quad (3.22)$$

A equação (1.10) é uma EDO de 1^a ordem, do tipo:

$$y \equiv \frac{dy}{dx} = f(x, y) \quad (3.23)$$

A solução da equação (1.10) não pode ser resolvida analiticamente, devendo ser resolvida por um método numérico. Neste software será utilizado o método de Runge-Kutta de 4^a ordem por ser um método explícito, estável e mais preciso. A equação de Runge-Kutta que permite esse cálculo pode ser definida por:

$$y_{j+1} = y_j + \frac{1}{6}(k_1 + k_2 + k_3 + k_4) \quad (3.24)$$

onde:

$$k_1 = \Delta x_{j+1} f(x_j, y_j) \quad (3.25)$$

$$k_2 = \Delta x_{j+1} f(x_j + \frac{\Delta x_{j+1}}{2}, y_j + \frac{k_1}{2}) \quad (3.26)$$

$$k_3 = \Delta x_{j+1} f(x_j + \frac{\Delta x_{j+1}}{2}, y_j + \frac{k_2}{2}) \quad (3.27)$$

$$k_4 = \Delta x_{j+1} f(x_j + \Delta x_{j+1}, y_j + k_3) \quad (3.28)$$

e

$$\Delta x_{j+1} = x_{j+1} - x_j \quad (3.29)$$

A saturação de óleo inicial S_{oi} [adimensional], a produção acumulada de óleo N_p [m^3] e a produção acumulada de gás G_p [m^3] podem ser calculadas segundo as equações:

$$S_{oi} = (1 - \frac{N_p}{N}) (\frac{B_o}{B_{oi}}) (1 - S_{wi}) \quad (3.30)$$

$$N_p = N [1 - \frac{S_o}{(1 - S_{wi})} (\frac{B_{oi}}{B_o})] \quad (3.31)$$

$$G_p = N [(\frac{B_o}{B_g} - R_s) (1 - \frac{N_p}{N}) - (\frac{B_{oi}}{B_g} - R_{si})] \quad (3.32)$$

Para Reservatórios com Gás em Solução, o método de Muskat também pode ser utilizado bastando para isso substituir $m=0$ na Eq. (1.10). Assim temos:

$$\frac{dS_o}{dp} = \frac{S_o \lambda + (1 - S_o - S_{wi}) \xi + S_o \eta (\Psi - \frac{CR}{\alpha})}{1 + (\frac{\mu_o}{\mu_g}) (\Psi - \frac{CR}{\alpha})} \quad (3.33)$$

sendo o restante exatamente igual como foi descrito acima.

3.3 Identificação de pacotes – assuntos

Os seguintes pacotes foram identificados:

- Reservatório: Esse pacote recebe os dados do usuário ou lê do disco (há métodos para obtenção para ambos os métodos), os dados se separam, de acordo com a sua natureza: rocha, fluído, aquífero, dados de produção. Quando juntas fornecem uma caracterização do reservatório como um todo. Além disso, esse banco de dados serve como base para os cálculos da simulação.

- Simulador de Mecanismos de Produção - SMP : Aqui se encontram os modelos pelos quais a previsão dos parâmetros comportamentais é feita, isto é, as classes que descrevem o método de Muskat, método de Fetkovitch e método de Van Everdingen & Hurst, tais classes estão separadas nas componentes para os reservatórios que esses modelos são apropriados.
- Modelagem Numérica Computacional: Contém os algoritmos matemáticos necessários para a solução dos modelos do SMP. Este pacote contém os algoritmos: Transformada de Laplace, Algoritmo de Inversão Numérica de Stehfest e Método de Runge-Kutta de 4ª ordem. Este pacote está separado do SMP, pois um dos objetivos da AOO é ter uma maior reusabilidade do código, assim, estando separados, é possível aplicar este mesmo pacote para outros problemas de engenharia, como por exemplo o de análise de teste de pressão.
- Gráfico: Aqui se encontra a biblioteca do gnuplot, necessária para a geração dos gráficos dos parâmetros do reservatório para uma melhor análise, é sabido que existe uma demanda por programa que gere gráficos, assim portanto, este software implementará a saída gráfica dos dados calculados.
- Biblioteca: Dentre as bibliotecas utilizadas, estarão as bibliotecas padrão de C++ (STL) e bibliotecas como a iostream, iomanip, etc. e a biblioteca GSL que fornecerá a base para as componentes do NCP.

3.4 Diagrama de pacotes – assuntos

O diagrama de pacotes da Figura 3.2 mostra as relações existentes entre os pacotes deste software.

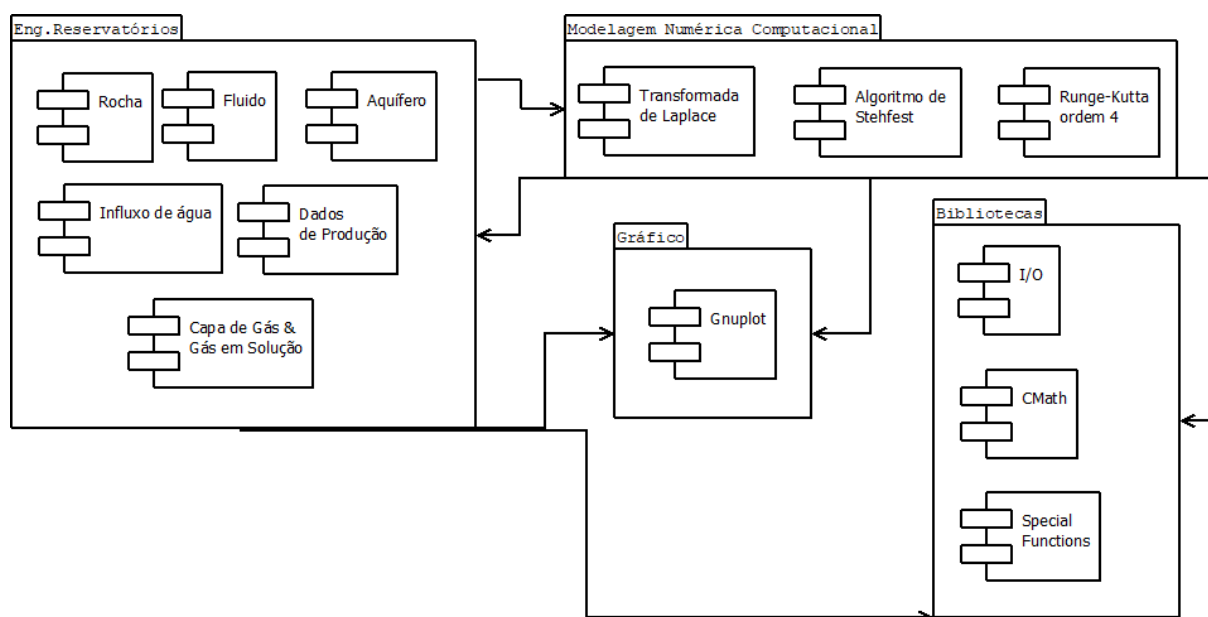


Figura 3.2: Diagrama de pacotes

Capítulo 4

AOO – Análise Orientada a Objeto

“A terceira etapa do desenvolvimento de um programa é a Análise Orientada a Objeto (AOO). A AOO utiliza algumas regras para identificar os objetos de interesse, as relações entre as classes, os atributos, os métodos, as heranças, as associações e as dependências” (Bueno, 2017). Nesta etapa o primordial não é o programa em si, mas o que será desenvolvido. Aqui, faremos e analisaremos um conjunto de diagramas que permitirá visualizar, de várias formas, o software. A AOO abrange o desenvolvimento dos modelos estrutural e dinâmico.

4.1 Diagramas de classes

O diagrama de classes é essencial para a montagem da versão inicial do código do software. Este diagrama é constituído pelas classes, seus métodos e atributos, além das diversas relações entre as classes. O diagrama de classes deste software é apresentado nas figuras 4.1 e 4.2. Note que o diagrama vem dividido em duas partes: a primeira diz respeito ao simulador de óleo com capa de gás ou gás em solução; enquanto a segunda diz respeito ao simulador de óleo/gás com influxo de água.

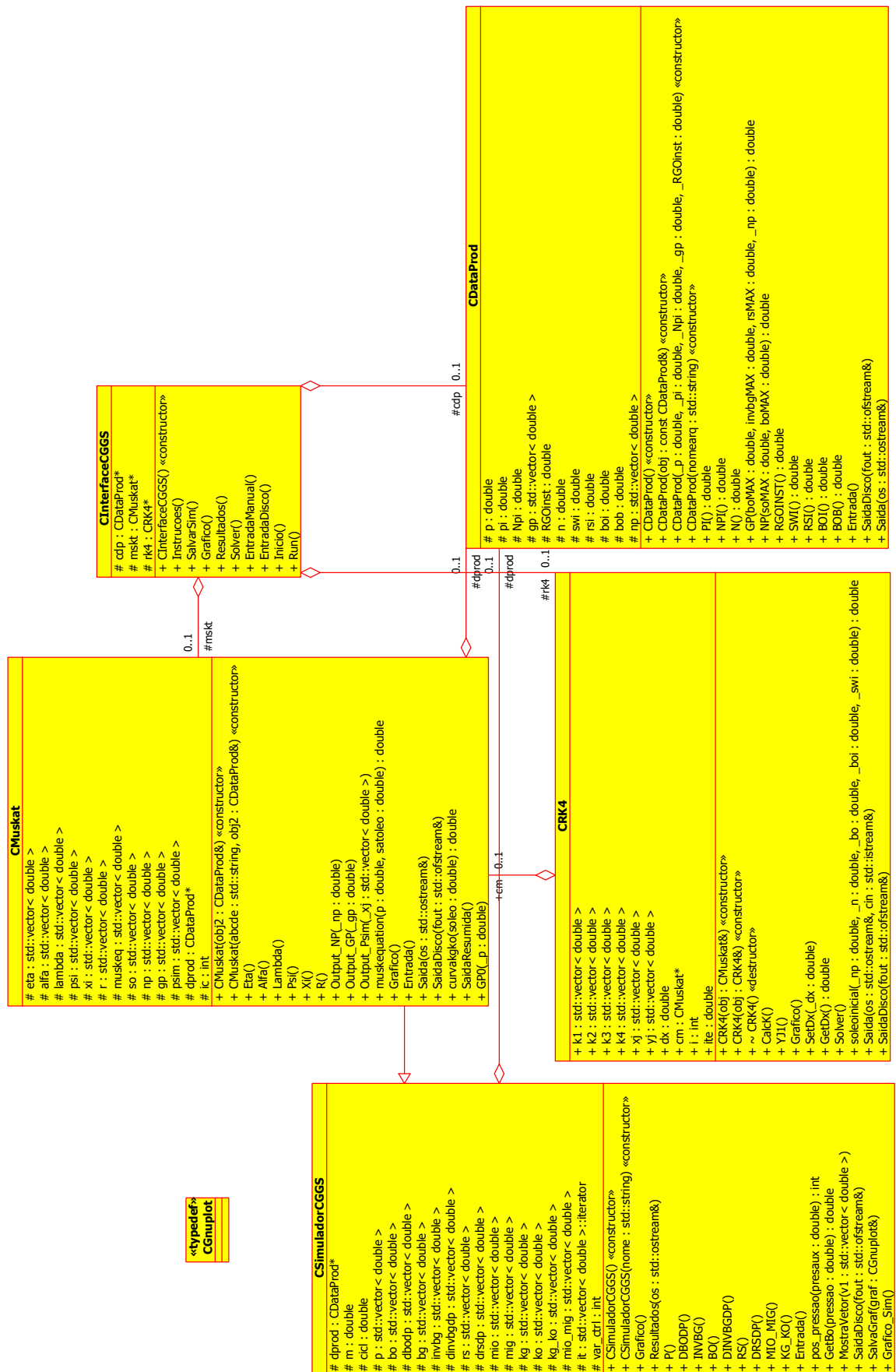


Figura 4.1: Diagrama de classes v2.0- parte 1



Figura 4.2: Diagrama de classes v1.0- parte 2

4.1.1 Dicionário de classes

- CAquifero: representa uma rocha aquífero, herda a classe CRocha e possui os atributos específicos do aquífero, que são permeabilidade, compressibilidade total e viscosidade da água.
- CDataProd: Classe que contém os dados de produção do reservatório.
- CFetkovich: herdeira de CSimulador, resolve o problema pelo método de Fetkovich.
- CFluido: representa um fluido e suas propriedades básicas: densidade, volume total e fator volume formação.
- CGás: representa um Gás, herdeira da classe CFluido, adiciona as propriedades do Gás como Temperatura e Pressão Pseudocrítica, temperatura do Gás e fator de compressibilidade. Define métodos para calcular este último, utilizando o método de Brill & Beggs ou a aproximação polinomial. Possui um método para calcular o fator volume formação de acordo com a pressão.
- CGeometria: Classe abstrata que representa a geometria da rocha. Ela é herdada pelas classes CLinear e CRadial.
- CGnuplot: Classe que fornece os métodos necessários para a geração de gráficos.
- CHurst: herdeira de CSimulador, resolve o problema pelo método de van Everdingen & Hurst.
- CInterface: Classe que interage com o usuário e que comanda o que será mostrado na tela para reservatórios com influxo de água. Cria um objeto CSimulador com os dados fornecidos pelo usuário e apresenta os resultados.
- CInterfaceCGGS: Classe que interage com o usuário e que comanda o que será mostrado na tela para reservatórios com capa de gás e gás em solução. Cria um objeto CSimuladorCGGS com os dados fornecidos pelo usuário e apresenta os resultados. A sigla CGGS significa (Capa de Gás e Gás em Solução - CGGS).
- CLaplaceWd: Classe que gera os influxos adimensionais no Campo de La Place para o método de van Everdingen & Hurst.
- CLinear: Classe herdeira de CGeometria, representa uma geometria linear, com largura e comprimento.
- CMuskat: Classe herdeira de CSimuladorCGGS, resolve o problema pelo método de Muskat.
- COleo: Classe que representa um Oleo, herdeira de CFluido, adiciona as propriedades do óleo como a compressibilidade do óleo e a compressibilidade efetiva.

- **CRadial**: Classe herdeira de **CGeometria**, representa uma geometria radial, com um raio.
- **CReservatorio**: representa uma rocha reservatório, herda a classe **CRocha** e possui os atributos específicos do reservatório, que são vazão e saturação de água.
- **CRocha**: Classe que representa uma rocha, possuindo os atributos pressão inicial, porosidade e espessura.média. Possui uma **CGeometria** que é definida no seu construtor. É herdada pelas classes **CReservatório** e **CAquifero**.
- **CRungeKutta4**: Classe que resolve a EDO do método de Muskat pela aplicação do método numérico de Runge-Kutta de 4ª ordem.
- **CSimulador**: Classe-mãe com os atributos necessários para a simulação de reservatórios com influxo de água, sendo os métodos **SolverGas()** e **SolverOleo()** definidos pelas classes herdeiras **CHurst** e **CFetkovich**. Possui os parâmetros de simulação: número de anos, tipo de influxo e passo. Possui objetos **CReservatorio**, **CFluido** e **CAquifero**.
- **CSimuladorCGGS**: Classe-mãe com os atributos necessários para a simulação de reservatórios com com capa de gás e gás em solução, sendo o método **MetMuskat()** definido pela classes herdeira **CMuskat**. Possui os parâmetros de simulação: pressão, derivada do fator volume-formação do óleo com a pressão, razão de solubilidade, permeabilidades relativa ao óleo e ao gás, viscosidades do óleo e gás, o inverso do fator volume-formação do gás, derivada da razão de solubilidade com a pressão, a derivada do inverso do fator volume-formação do gás com a pressão. Possui objetos **CReservatorio**, **CFluido** e **CDataProd**.
- **CStehfest**: Classe que realiza a inversão numérica dos influxos adimensionais no Campo de Laplace, provenientes da classe **CLaplaceWd**.

4.2 Diagrama de seqüência – eventos e mensagens

O diagrama de seqüências representa, no tempo, a troca de eventos e mensagens entre os objetos do sistema. Ele é parte do modelo dinâmico da análise orientada a objeto.

4.2.1 Diagrama de sequência geral

A Figura 4.3 mostra o diagrama de sequência versão v1.0 para a resolução de um problema de influxo de água. O usuário cria um objeto **CInterface** e fornece os dados necessários. Então é criado um objeto **CSimulador**, que possui agregado os objetos **CReservatório**, **CFluido** e **CAquifero**.

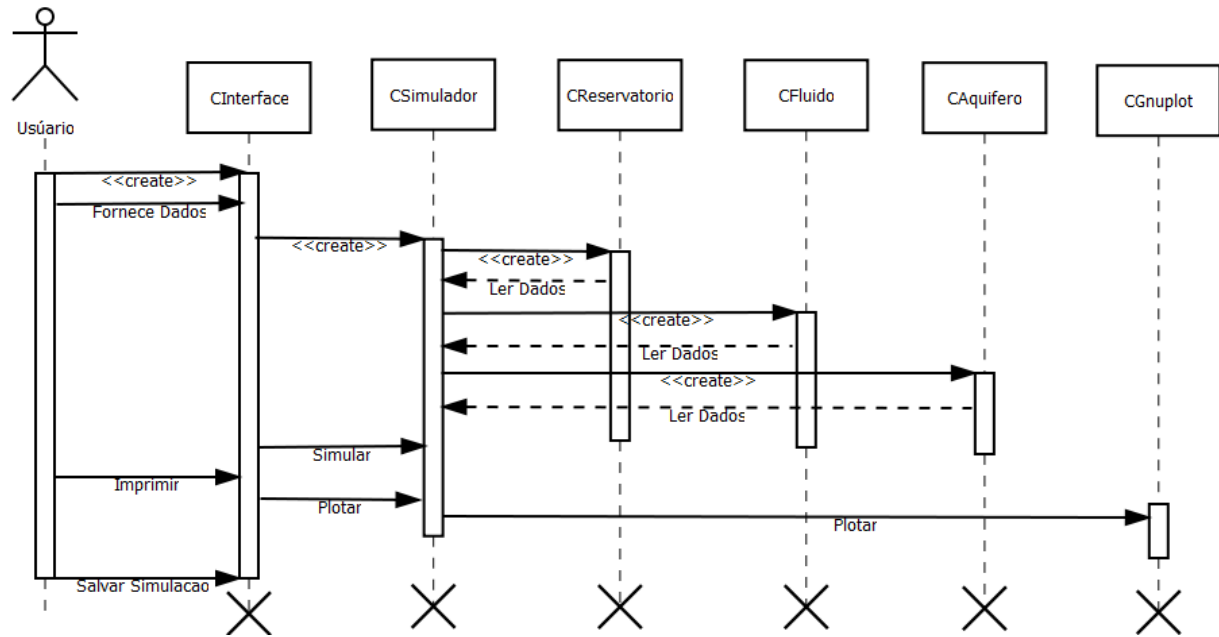


Figura 4.3: Diagrama de sequência v1.0

A Figura 4.4 mostra o diagrama de sequência versão v2.0 para a resolução de um problema de cálculo de parâmetros de reservatórios com capa de gás e gás em solução. O usuário cria um objeto CInterfaceCGGS, fornece os dados necessários e é criado os objetos agregados CSimuladorCGGS, CMuskat, CDataProd e CRK4.

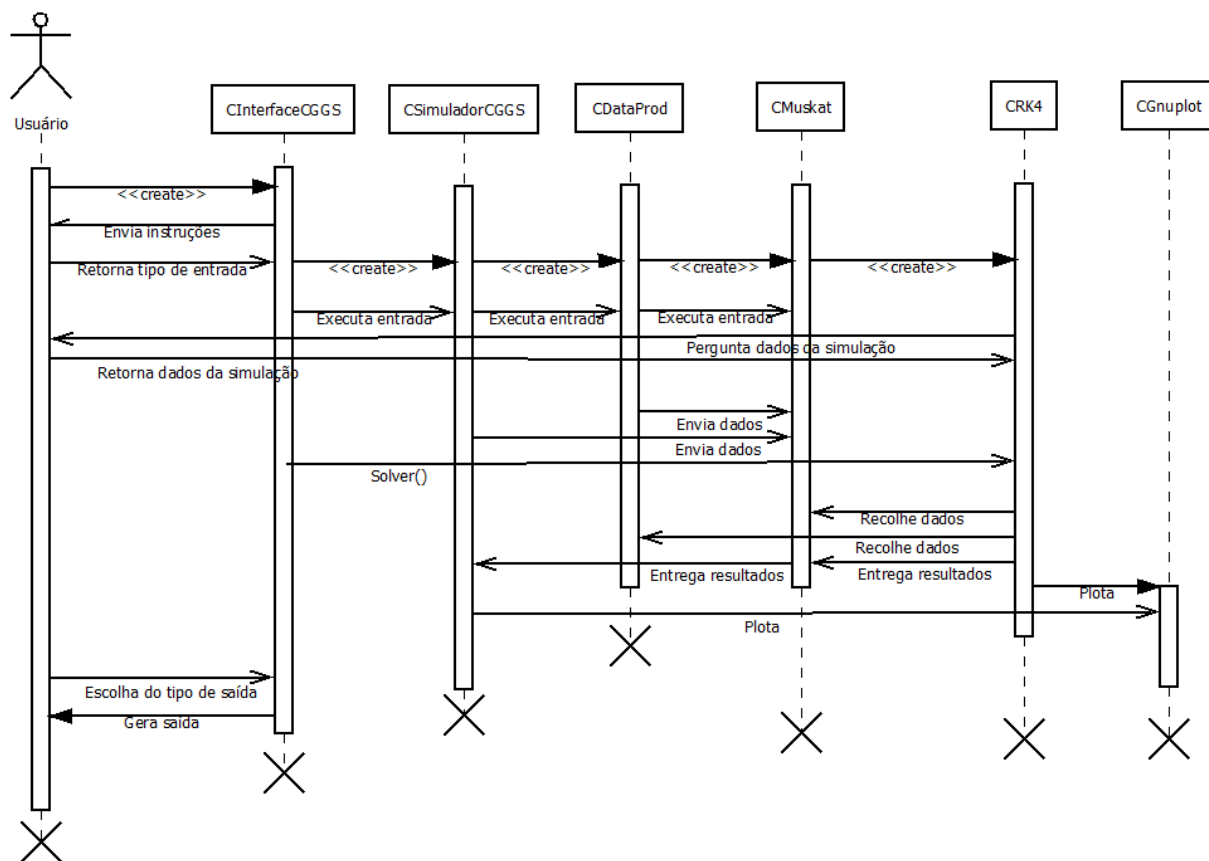


Figura 4.4: Diagrama de seqüência v2.0

4.3 Diagrama de comunicação – colaboração

“O diagrama de comunicação pode ser desenvolvido como uma extensão do diagrama de caso de uso, detalhando o mesmo por meio da inclusão de objetos, mensagens e parâmetros trocados entre objetos. No diagrama de comunicação, o foco é a interação e a troca de mensagens e dados entre os objetos”. A Figura 4.5 apresenta o diagrama de comunicação da classe CSimulador para a versão v1.0 e a Figura para a versão v2.0.

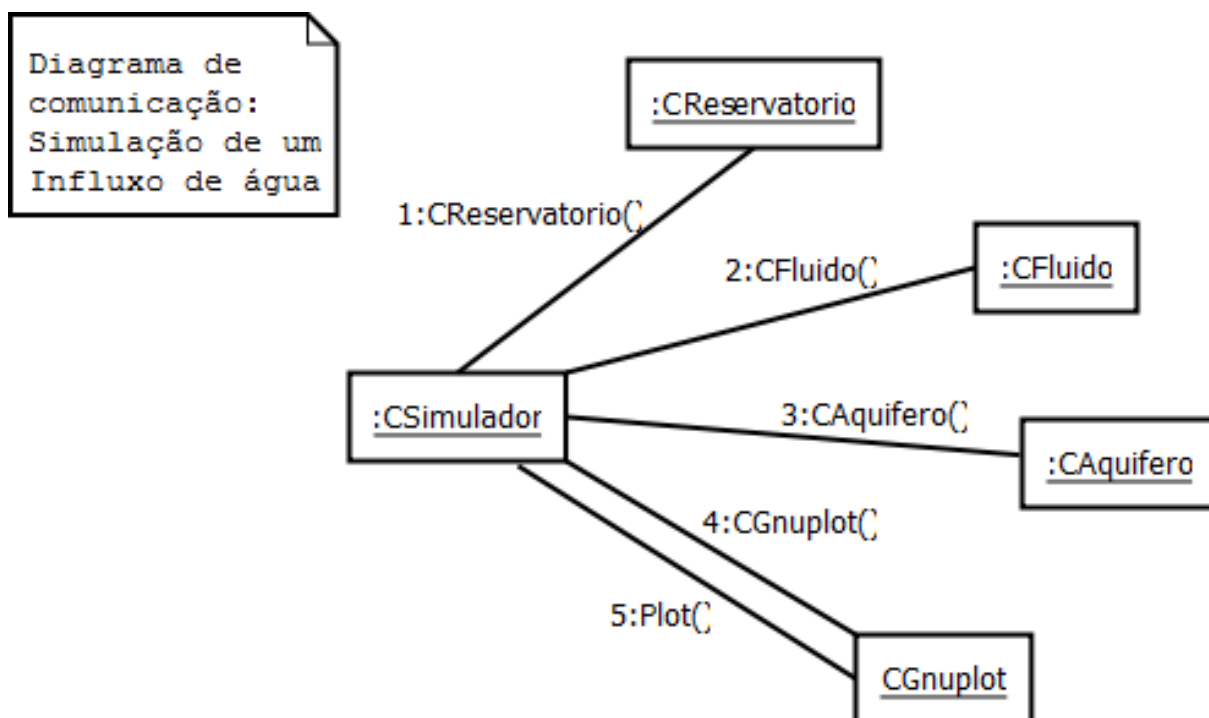


Figura 4.5: Diagrama de comunicação v1.0 : “Simulação de um influxo de água”

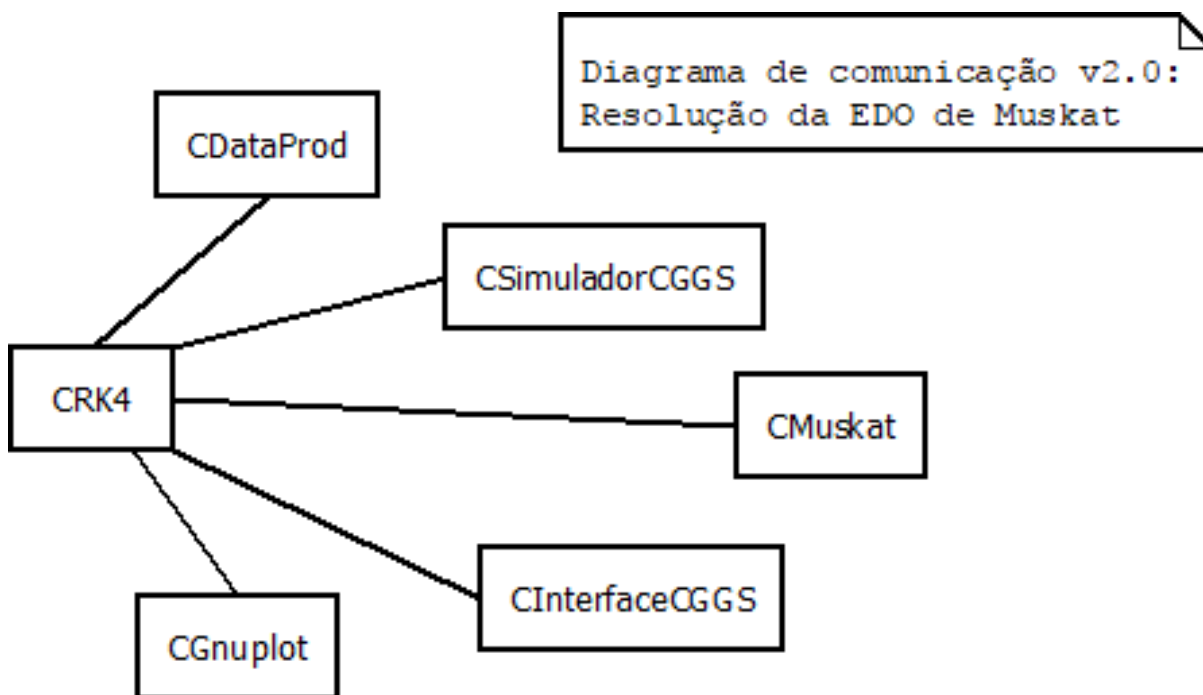


Figura 4.6: Diagrama de comunicação v2.0: “Resolução da EDO de Muskat”

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 de um objeto)” [3]. Estes estados podem ser a realização de uma atividade

demorada ou a espera de um evento. A Figura 4.7 apresenta este diagrama para a classe CSimulador que realiza o cenário de teste e a Figura 4.8 apresenta o diagrama para o cenário de teste da versão v2.0.

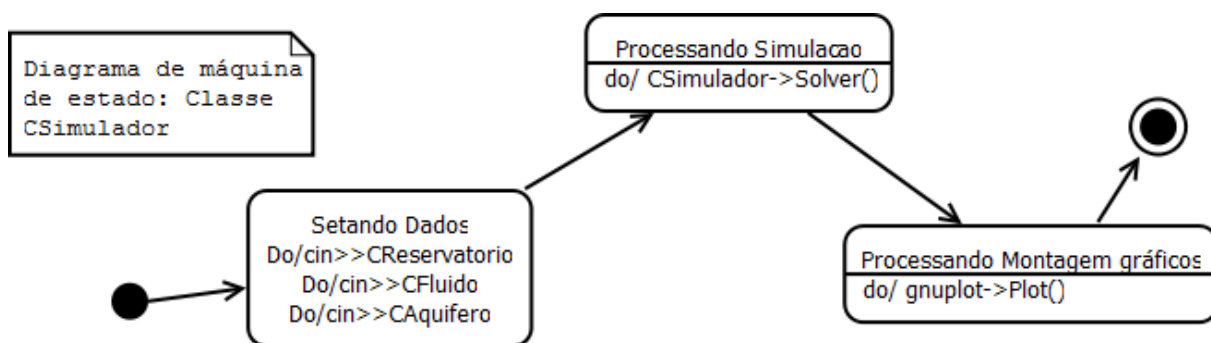


Figura 4.7: Diagrama de máquina de estado v1.0: "Cenário de teste (CSimulador)"

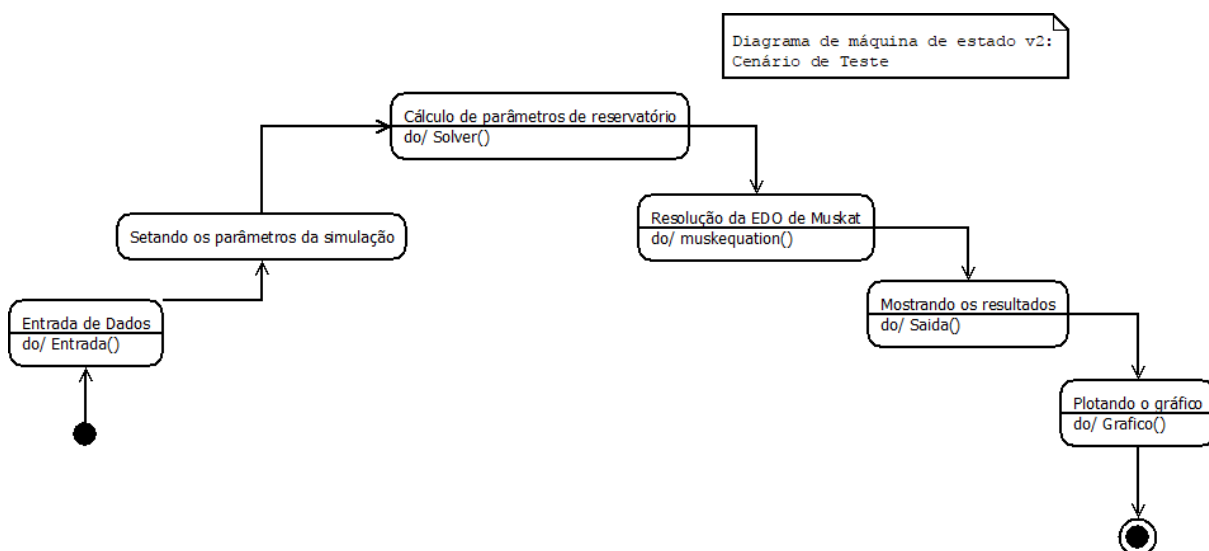


Figura 4.8: Diagrama de máquina de estado v2.0: "Cenário de teste"

4.5 Diagrama de atividades

"O diagrama de atividades correspondente a uma atividade específica do diagrama de máquina de estado". A Figura 4.9 apresenta um diagrama de atividades versão 1.0 detalhando um caso específico do diagrama de máquina de estado: "Processando Simulação - Método Solver()". A Figura 4.10 apresenta um diagrama de atividades versão 2.0 detalhando um caso específico do diagrama de máquina de estado: "Resolvendo a EDO de Muskat".

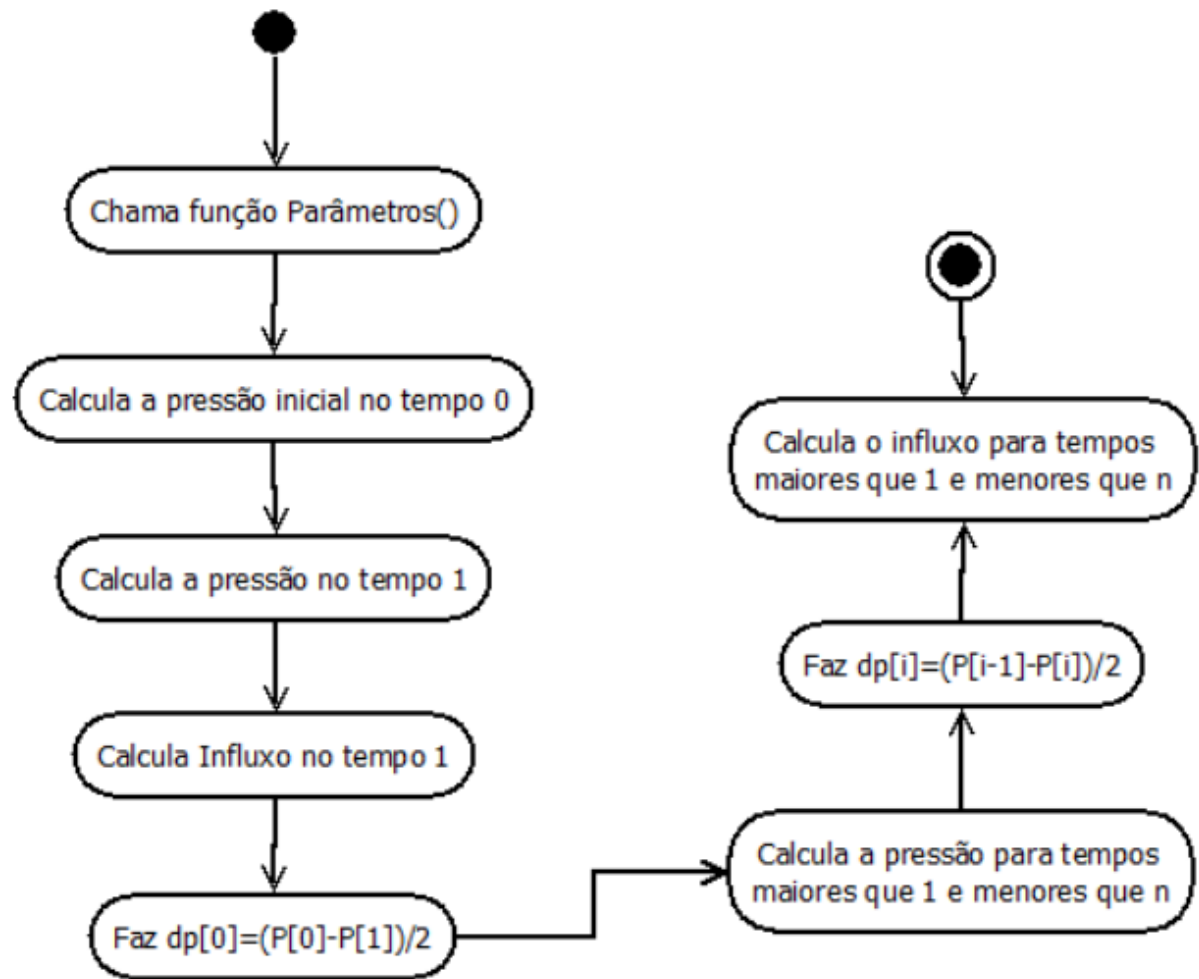


Figura 4.9: Diagrama de atividades v1.0: "Processando Simulação - Método Solver()"

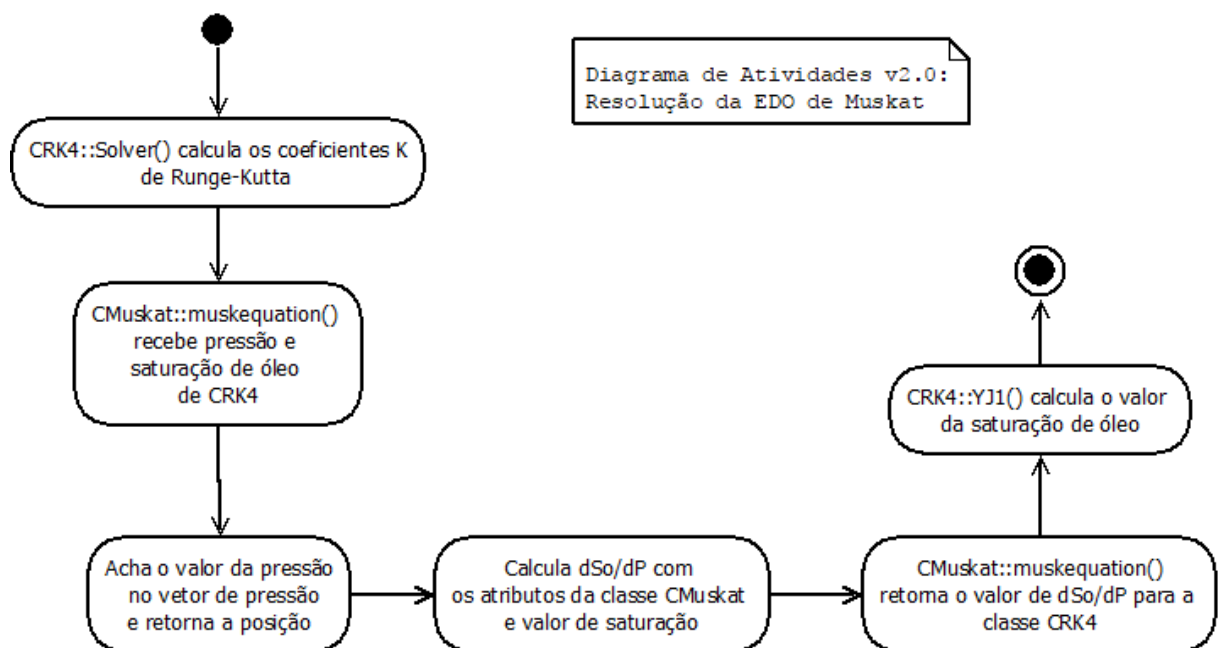


Figura 4.10: Diagrama de atividades v2.0: "Resolvendo a EDO de Muskat"

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 a objeto, desenvolveu-se o projeto do sistema, o qual reúne 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.

Foram definidos 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.

O presente projeto do sistema foi elaborado para apresentar soluções, valendo-se dos seguintes itens como:

1. Protocolos

- Definição dos protocolos de comunicação entre os diversos elementos externos (como dispositivos). Por exemplo: se o sistema envolve o uso dos nós de um cluster, devem ser considerados aspectos como o protocolo de comunicação entre os nós do cluster.
 - Esta versão do software comunica-se com o programa externo `gnuplot`.
- Definição dos protocolos de comunicação entre os diversos elementos internos (como objetos).

- O programa utilizará uma máquina computacional com HD, processador, teclado para a entrada de dados e o monitor para a saída de dados. Os arquivos gerados pelo programa estarão em formato de texto em um banco de dados.
- Definição da interface API de suas bibliotecas e sistemas.
 - Utilizará uma biblioteca GSL para cálculos matemáticos especiais chamada *special functions* da GNU.
- Definição do formato dos arquivos gerados pelo software. Por exemplo: prefira formatos abertos, como arquivos txt e xml.
 - Os arquivos de texto com os resultados da simulação terão o formato .dat ou .txt.

2. Recursos

- Identificação e alocação dos recursos globais, como os recursos do sistema serão alocados, utilizados, compartilhados e liberados. Implicam modificações no diagrama de componentes.
 - O simulador utiliza como recurso para plotagem de gráficos o programa externo Gnuplot. Que por sua vez plota os objetos da imagem rotulados e ajustados as formas geométricas. Além do básico, como HD, CPU, RAM e periféricos.

3. Controle

- Identificação e seleção da implementação de controle, sequencial ou concorrente, baseado em procedimentos ou eventos. Implicam modificações no diagrama de execução.
 - Este software requer um controle sequencial.

4. Plataformas

- Identificação e definição das plataformas a serem suportadas: hardware, sistema operacional e linguagem de software.
 - O software é multiplataforma, funcionando em ambos Windows e GNU/Linux.
 - A linguagem de software utilizada é a C++ orientada a objeto.
- Seleção das bibliotecas externas a serem utilizadas.
 - Serão utilizadas a biblioteca `gnuplot` e a `GSL special functions` neste software.

- Seleção do ambiente de desenvolvimento para montar a interface de desenvolvimento – IDE.
 - Para a plataforma Windows, temos a seguinte IDE: Desktop com processador AMD FX-6100, 8GB RAM DDR3 e placa gráfica AMD Radeon RX 470. O programa será escrito e compilado usando o programa DEV-C++.
 - Para Linux: Desktop com processador Intel Core i7 3770, 8GB RAM DDR3, placa gráfica Nvidia GT 620. O programa será escrito e compilado usando o programa Kate e gcc++.

5. Padrões de projeto

- Normalmente os padrões de projeto são identificados e passam a fazer parte de uma biblioteca de padrões da empresa. Mas isto só ocorre após a realização de diversos projetos.
 - Não se aplica.

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-se a análise desenvolvida e as características da plataforma escolhida (hardware, sistema operacional e linguagem de programação).

É realizado um maior detalhamento do funcionamento do programa, 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.

Ainda no projeto orientado a objeto incluem-se as bibliotecas necessárias para acesso ao disco, cria um objeto específico para acessar o disco, podendo, portanto, acrescentar novas classes àquelas desenvolvidas na análise.

Exemplo: na análise você define que existe um método para salvar um arquivo em disco, define um atributo nomeDoArquivo, mas não se preocupa com detalhes específicos da linguagem. Já no projeto, você inclui as bibliotecas necessárias para acesso ao disco, cria um objeto específico para acessar o disco, podendo, portanto, acrescentar novas classes àquelas desenvolvidas na análise.

Efeitos do projeto no modelo estrutural

- Estabelecer as dependências e restrições associadas à plataforma escolhida.
 - Na plataforma Windows, o gnuplot necessita estar instalado para o correto funcionamento do software.

Efeitos do projeto no modelo dinâmico

- Não foi necessário nesta etapa do projeto.

Efeitos do projeto nos atributos

- Na classe CMuskat foram adicionados novos atributos: eta, alfa, lambda, psi, xi, r. Esses atributos são vetores do tipo double. Tais atributos servem para armazenar os valores das constantes da equação de Muskat, cada posição do vetor, corresponde a um valor da constante para uma determinada pressão.

Efeitos do projeto nos métodos

- Na classe CSimuladorCGGS foi adicionado o método FLUIPROP() que pede a entrada e calcula os seguintes atributos: p, dbodp, invbg, d(invbg)dp, rs, drsdp, mio, mig, kg, ko, mio_mig, kg_ko. A razão da existência deste método se explica pela intenção em deixar o código mais enxuto e pela intenção de agrupar algoritmos de mesma natureza em um só método. Caso esses atributos fossem inicializados no método sobrecarregado de entrada, o código ficaria confuso. Além disso foram adicionados os métodos *get*: KG(), KO(), MIG(), MIO().

Efeitos do projeto nas heranças

- As relações de herança continuam inalteradas.

Efeitos do projeto nas associações

- A classe CRungeKutta4 foi associada a classe CMuskat visto que a classe CRungeKutta4 faz parte da CMuskat (principalmente para a solução do método muskequation() da classe CMuskat).

Efeitos do projeto nas otimizações

- Não foi necessário rever nesta etapa do projeto.

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 na Figura 5.1 onde temos o diagrama de componentes. Podemos perceber as dependências de cada componente. Por exemplo, o componente CSimuladorCGGS depende da CInterfaceCGGS para funcionar, que por sua vez, depende do main.cpp. O componente CMuskat (NCP) dependa da CSimuladorCGGs p/funcionar. Com esse diagrama podemos saber das componentes necessárias para o software funcionar.

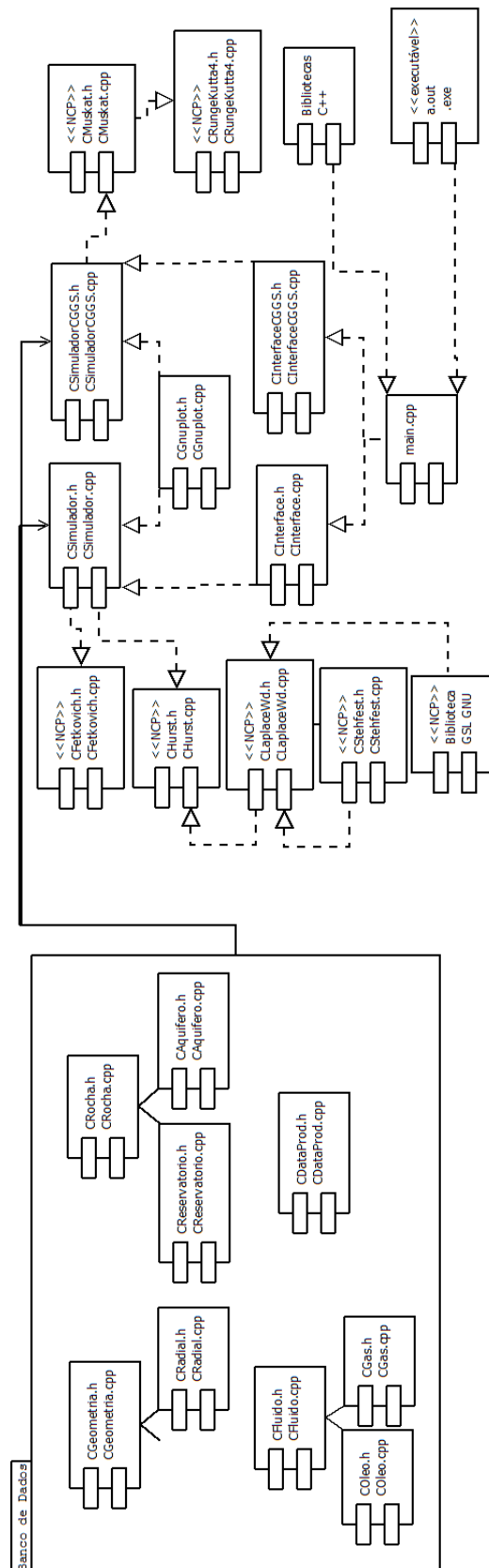


Figura 5.1: Diagrama de componentes

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.

Na figura 5.2, temos o diagrama de implantação deste software. Para que haja um correto e realístico desempenho da simulação provida pelo software, é necessário que haja primeiro, não só o computador com todos os hardwares requeridos (CPU, RAM, HD), mas também uma aquisição de dados, sendo requeridos dados PVT e dados do reservatório (ϕ , K_g , K_o , etc.)

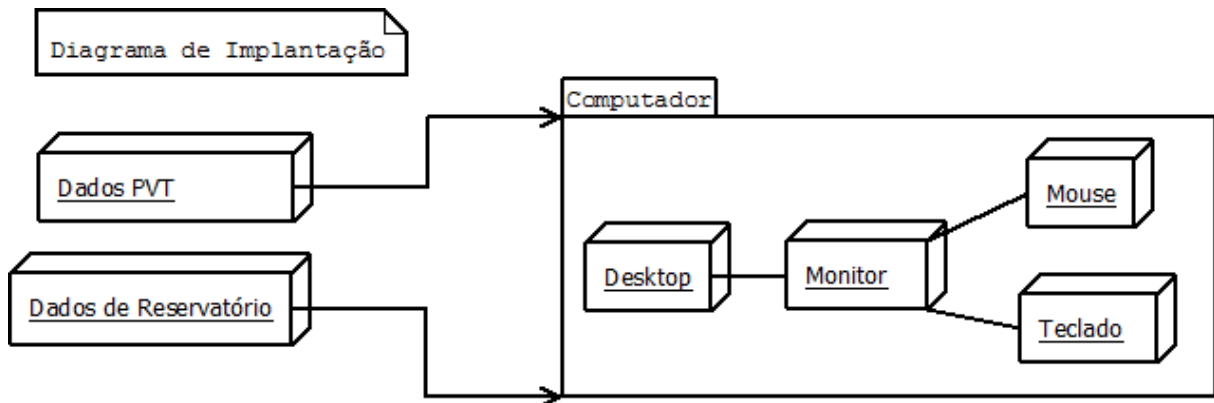


Figura 5.2: Diagrama de implantação.

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.

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

Listing 6.1: Arquivo de cabeçalho da classe CAquifero.

```
1 /** Previsao do Compartamento de Reservatorios  
2 de Gas e de Oleo com Influxo de Agua  
3 @author Carlos Andre Martins de Assis e Gabriel Clemente Franklin  
4  
5 */  
6  
7 /** @brief Classe com as propriedades do aquifero.  
8  
9  
10 @class CAquifero  
11 @file CAquifero.h  
12 */  
13 #ifndef CAquifero_h  
14 #define CAquifero_h  
15  
16 #include "CRocha.h"  
17  
18 class CAquifero: public CRocha  
19 {  
20     protected:  
21         double k; /// permeabilidade  
22         double ct; /// compressibilidade total  
23         double u; /// viscosidade da água
```

```

24     public:
25
26         ///brief Construtor default
27         CAquifero(int op):CRocha(op){};
28         ///brief Construtor sobrecarregado (quantidade de
29             parâmetros) radial
30         CAquifero(double _pi, double _poro, double _h, double _k,
31             double _ct, double _u, double _re)
32             :CRocha(_pi, _poro, _h, _re), k(_k), ct(_ct), u(
33                 _u){}
34         ///brief Construtor sobrecarregado (quantidade de
35             parâmetros) linear
36         CAquifero(double _pi, double _poro, double _h, double _k,
37             double _ct, double _u, double _l, double _w)
38             :CRocha(_pi, _poro, _h, _l, _w), k(_k), ct(
39                 _ct), u(_u){};
40         ///brief Construtor sobrecarregado de cópia
41         CAquifero(const CAquifero& obj):CRocha(obj), k(obj.k), ct(obj
42             .ct), u(obj.u){};
43         ///Metodos de Entrada e Saida padrao, diferenciados para
44             arquivo
45         ///brief Metodos de Entrada e Saida padrao, diferenciados
46             para arquivo
47         virtual void Entrada( std :: ostream & os , std :: istream
48             & is);
49         virtual void Entrada(std :: ifstream & is);
50         virtual void Saida( std :: ostream & os );
51         ///brief Membros publicos que retornam os parametros
52         double K(){return k;};
53         double Ct(){return ct;};
54         double U(){return u;};
55         ///brief Destrutor
56         virtual ~CAquifero(){};
57     };
58 #endif

```

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

Listing 6.2: Arquivo de implementação da classe CAquifero.

```

50 /* Previsão do Compartamento de Reservatórios
51     de Gás e de Óleo com Influxo de Água
52
53     @Autores: Carlos André Martins de Assis e Gabriel Clemente Franklin
54     @file CAquifero.cpp
55 */
56
57
58

```

```

59#include "CAquifero.h"
60
61using namespace std;
62
63void CAquifero::Entrada(ostream & os ,istream & is)
64{
65    os<<"\t-----\n\tCAquifero\n\t-----"<<endl;
66    CRocha::Entrada(os,is);
67    os<<"Entre com a Permeabilidade (md): ";
68    is>>k;
69    os<<"Entre com o Compressibilidade Total (cm2/kgf): ";
70    is>>ct;
71    os<<"Entre com o Viscosidade da Agua (cp): ";
72    is>>u;
73}
74
75void CAquifero::Entrada(istream & is)
76{
77
78    CRocha::Entrada(is);
79    is.ignore(100,':');
80    is>>k;
81    is.ignore(100,':');
82    is>>ct;
83    is.ignore(100,':');
84    is>>u;
85}
86
87
88void CAquifero::Saida(ostream & os)
89{
90    os<<"\t-----\n\tPropriedades do Aquifero\n\t-----"<<
        endl;
91    CRocha::Saida(os);
92    os<<"Permeabilidade (md): "<<k<<endl;
93    os<<"Compressibilidade Total (cm2/kgf): "<<ct<<endl;
94    os<<"Viscosidade da Agua (cp): "<<u<<endl;
95}

```

Apresenta-se na listagem 6.3 o arquivo com código da classe CDataProd.

Listing 6.3: Arquivo de cabeçalho da classe CDataProd.

```

96#ifndef CDataProd_h
97#define CDataProd_h
98
99#include<iostream>
100#include<cstdlib>
101#include<fstream>
102#include<functional>

```



```

103 #include <algorithm>
104 #include <vector>
105 #include <string>
106 #include "CGnuplot.h"
107 #include <utility>
108 #include <cmath>
109 #include <iomanip>
110
111 class CDataProd{
112     protected:
113     double p;
114     double pi; ///pressão inicial (pi < pi_res) - Pressão CRK4
115               inicio
116     double Npi; ///Produção de Óleo Acumulada até So/p=pi
117     std::vector<double> gp; ///Produção Acumulada de Gás GP
118     double RG0inst; ///Razão Gás-Óleo Inst
119     double n; ///Volume Original de Óleo
120     double swi; ///Saturação de Água Irredutível
121     double rsi; ///Rs inicial em pi_res
122     double boi; ///Bo inicial do óleo em pi_res
123     double bob; ///Bo na pressão de bolha
124     std::vector<double> np; ///Vetor de Armazenamento dos valores da
125                           Produção Acumulada de Óleo
126     public:
127     CDataProd()
128     {
129         std::cout <<"Criando Objeto CDataProd!"<<'\\n';
130         std::cout <<"Entre com a pressão inicial de análise pi em"
131                 <<"kgf/cm^2."<<'\\n';
132         std::cin>>pi; std::cin.get();
133         std::cout <<"Entre com a produção acumulada inicial de óleo"
134                 <<"Npi em m^3std."<<'\\n';
135         std::cin>>Npi; std::cin.get(); np.push_back(Npi);
136         std::cout <<"Entre com o Volume Original de Óleo N em m^3std"
137                 <<"."<<'\\n';
138         std::cin>>n; std::cin.get();
139         std::cout <<"Entre com a Razão Gás-Óleo Instantânea RG0inst"
140                 <<"em m^3std/m^3std."<<'\\n';
141         std::cin>>RG0inst; std::cin.get();
142         std::cout <<"Entre com a Saturação de Água Irredutível swi"<<
143                 <<'\\n';
144         std::cin>>swi; std::cin.get();
145         std::cout <<"Entre com a Rs inicial Rsi em m^3std/m^3std."
146                 <<'\\n';
147         std::cin>>rsi; std::cin.get();
148         std::cout <<"Entre com Bo inicial do óleo em pi_res em m^3"
149                 <<"std/m^3std."<<'\\n';
150         std::cin>>boi; std::cin.get();

```

```

142         std::cout <<"Entre com Bo na pressão de bolha em m^3 std/m^3
           std."<<'\\n';
143         std::cin>>bob; std::cin.get();
144     }; ///Construtor Default - Entrada Manual
145     CDataProd(const CDataProd& obj): p(obj.p), pi(obj.pi), Npi(obj.
           Npi), gp(obj.gp), RG0inst(obj.RG0inst){}; ///Construtor de
           Cópia
146     CDataProd(double _p, double _pi, double _Npi, double _gp, double
           _RG0inst): p(_p), pi(_pi), Npi(_Npi), gp(_gp), RG0inst(
           _RG0inst){}; ///Construtor Sobrecarregado
147     CDataProd(std::string nomearq){
148         std::ifstream fin(nomearq.c_str());
149         if(!fin.good()){
150             std::cout<<"Arquivo não encontrado"<<'\\n';
151             exit(0);
152         }
153         double auxvar=0;
154         std::cout<<"Lendo arquivo"<<'\\n';
155         fin.ignore(5000,'\\n');
156         fin.ignore(5000,'\\n');
157         fin.ignore(5000,'\\n');
158         fin.ignore(5000,'\\n');
159         fin.ignore(55,'\\n');
160         fin>>n; fin.ignore(55,'\\n');
161         fin.ignore(55,'\\n');
162         fin>>swi;fin.ignore(55,'\\n');
163         fin.ignore(55,'\\n');
164         fin>>rsi;fin.ignore(55,'\\n');
165         fin.ignore(55,'\\n');
166         fin>>boi;fin.ignore(55,'\\n');
167         fin.ignore(55,'\\n');
168         fin>>bob;fin.ignore(55,'\\n');
169         fin.ignore(55,'\\n');
170         fin>>Npi;fin.ignore(55,'\\n');
171         fin.ignore(55,'\\n');
172         fin>>RG0inst;fin.ignore(55,'\\n');
173         fin.ignore(55,'\\n');
174         fin>>pi;fin.ignore(55,'\\n');
175         fin.close();
176         std::cout<<"Leitura terminada CDataProd"<<'\\n';
177     } ///Construtor Default para Entrada em Arquivo de Disco
178     //~CDataProd();
179
180     double PI(){return pi;} ///Retorna a Pressão Inicial de Análise
181     double NPI(){return Npi;} ///Retorna a Produção Acumulada de
           Óleo avaliada na pressão inicial de análise
182     double N(){return n;} ///Retorna o Volume Original de Óleo no
           Reservatório

```

```

183     double GP(double boMAX, double invbgMAX, double rsMAX, double _np
        );///Método que calcula a Produção Acumulada de Gás (GP)
184     double NP(double soMAX, double boMAX);///Método que calcula a
        Produção Acumulada de Óleo (NP)
185     double RGOINST(){return RGOinst;} ///Retorna a Razão Gás Óleo
        Instantânea
186     double SWI(){return swi;} ///Retorna a Saturação de Água inicial
187     double RSI(){return rsi;} ///Retorna a Razão de Solubilidade
        Inicial
188     double BOI(){return boi;} ///Retorna BO inicial
189     double BOB(){return bob;} ///Retorna BO inicial no ponto de
        bolha
190
191     void Entrada(); ///Método de Entrada Manual
192     void SaidaDisco(std::ofstream& fout); ///Método para salvar em
        disco
193     void Saida(std::ostream& os); ///Método para saída na tela
194 };
195 #endif

```

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

Listing 6.4: Arquivo de implementação da classe CDataProd.

```

196 #include "CDataProd.h"
197
198
199 double CDataProd::GP(double boMAX, double invbgMAX, double rsMAX, double
    _np){
200     double aux= n*(((boMAX*invbgMAX-rsMAX)*(1-(_np/n)))-((
        boi*invbgMAX)-rsi));
201     gp.push_back(aux);
202     return aux;
203     }///Método que calcula a Produção Acumulada de Gás (GP)
204 double CDataProd::NP(double soMAX, double boMAX){
205     double aux= n*(1-((soMAX/(1-swi))*(boi/boMAX)));
206     np.push_back(aux);
207     return aux;
208 }
209
210 void CDataProd::Entrada(){
211     std::cout <<"Entre com a pressão inicial pi em kgf/cm^2.
        "<<'\\n';
212     std::cin>>pi; std::cin.get();
213     std::cout <<"Entre com a produção acumulada inicial de óleo Npi em
        m^3 std."<<'\\n';
214     std::cin>>Npi; std::cin.get();
215     std::cout <<"Entre com o Volume Original de Óleo N em m^3 std."
        <<'\\n';
216     std::cin>>n; std::cin.get();

```

```

217         std::cout <<"Entre com a Razão Gás Óleo Instantânea RG0inst em m
           ^3std/m^3std."<<'\\n';
218         std::cin>>RG0inst; std::cin.get();
219         std::cout <<"Entre com a Saturação de Água Irredutível swi"<<'\\n
           ';
220         std::cin>>swi; std::cin.get();
221         std::cout <<"Entre com a Rsi inicial Rsi em m^3std/m^3std."<<'\\
           n';
222         std::cin>>rsi; std::cin.get();
223         std::cin>>boi; std::cin.get();
224         std::cout <<"Entre com a pressão de bolha em m^3std/m^3std
           ."<<'\\n';
225         std::cin>>bob; std::cin.get();
226     }
227 void CDataProd::SaidaDisco(std::ofstream& fout){
228         //ofstream fout(s.c_str())
229         fout<<"A seguir os dados de produção do reservatório:"<<
           '\\n';
230         fout<<"
           -----
           "<<'\\n';
231         fout<<"A pressão inicial de análise é: " << PI() << " kgf/cm^2."
           <<'\\n';
232         fout<<"O volume original de óleo é: " << N() << " m^3std." <<'\\
           n';
233         fout<<"A razão gás-óleo instantânea é de: " <<RG0INST()<<" m^3
           std/m^3std."<<'\\n';
234         fout<<"A produção de óleo acumulada para a pressão inicial de
           análise é de: " <<NPI()<< " m^3std."<<'\\n';
235         fout<<"A saturação de água irredutível é de: " <<SWI()<<"%"<<'\\n'
           ;
236         fout<<"A razão de solubilidade inicial é de: " <<RSI()<<" m^3std/
           m^3std"<<'\\n';
237         fout<<"O fator volume-formação inicial do óleo é de: " <<BOI()<<"
           m^3std/m^3std"<<'\\n';
238         fout<<"O fator volume-formação do óleo na pressão de bolha é de:
           " <<BOB()<<" m^3std/m^3std"<<'\\n';
239         fout<<"A produção de gás acumulada para a pressão prevista foi
           de: " <<gp.back()<<" m^3std"<<'\\n';
240         fout<<"A produção de óleo acumulada para a pressão prevista é de
           : " <<np.back()<<" m^3std."<<'\\n';
241         fout<<"
           -----
           "<<'\\n';
242         fout<<'\\n';
243     } ///Método para salvar em disco
244 void CDataProd::Saida(std::ostream& os)
245     {

```

```

246         os<<"A seguir os dados de produção do reservatório:"<<'\\n';
247         os<<"
                -----
                "<<'\\n';
248         os<<"Gerando a saída de dados da classe CDataProd:"<<'\\n';
249         os<<"A pressão inicial de análise é: " << PI() << " kgf/cm^2."
                <<'\\n';
250         os<<"O volume original de óleo é: " << N() << " m^3 std." <<'\\n
                ';
251         os<<"A razão gás-óleo instantânea é de: " << RG0INST() << " m^3 std
                /m^3 std."<<'\\n';
252         os<<"A produção de óleo acumulada para a pressão inicial de
                análise é de: " << NPI() << " m^3 std."<<'\\n';
253         os<<"A saturação de água irreduzível é de: " << SWI() << "%"<<'\\n';
254         os<<"A razão de solubilidade inicial é de: " << RSI() << " m^3 std/m
                ^3 std"<<'\\n';
255         os<<"O fator volume-formação inicial do óleo é de: " << B0I() << " m
                ^3 std/m^3 std"<<'\\n';
256         os<<"O fator volume-formação do óleo na pressão de bolha é de:
                " << B0B() << " m^3 std/m^3 std"<<'\\n';
257         os<<"A produção de gás acumulada para a pressão prevista foi de
                : " << gp.back() << " m^3 std"<<'\\n';
258         os<<"A produção de óleo acumulada para a pressão prevista é de:
                " << np.back() << " m^3 std."<<'\\n';
259         os<<"
                -----
                "<<'\\n';
260 } // Método para saída na tela

```

Apresenta-se na listagem 6.5 o arquivo com código da classe CFetkovich.

Listing 6.5: Arquivo de cabeçalho da classe CFetkovich.

```

261 /** Previsao do Compartamento de Reservatorios
262     de Gas e de Oleo com Influxo de Agua
263     @author Carlos Andre Martins de Assis e Gabriel Clemente Franklin
264     @file CFetkovich.h
265 */
266
267 /** @brief Classe que implementa o metdo de Fetkovich
268     @class CFetkovich
269 */
270 #ifndef CFetkovich_h
271 #define CFetkovich_h
272
273 #include "CSimulador.h"
274
275
276
277 using namespace std;

```

```

278
279 class CFetkovich: public CSimulador
280
281 {
282     protected:
283         double J; ///< Índice de produtividade do Aquifero
284         double Wei; ///< Influxo Maximo
285
286     public:
287
288
289         ///

```

```

316         double EqSeg (double, double, double);
317         ///@brief Inicializa as constantes para caso Radial ou Linear
318         virtual void Constantes();
319
320         ///@brief Resolve o problema para o caso do gas
321         virtual void SolverGas();
322         ///@brief Resolve o problema para o caso do oleo
323         virtual void SolverOleo();
324         ///@brief Destrutor
325         virtual ~CFetkovich(){}
326     };
327
328 #endif

```

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

Listing 6.6: Arquivo de implementação da classe CFetkovich.

```

329 /* Previsão do Compartamento de Reservatórios
330    de Gás e de Óleo com Influxo de Água
331
332    @Autores: Carlos André Martins de Assis e Gabriel Clemente Franklin
333    @file CFetkovich.cpp
334 */
335 #include "CFetkovich.h"
336 #include "CGnuplot.h"
337
338 void CFetkovich::Constantes()
339 {
340     if (geom==1) //Caso Radial
341     {
342         J=0.05255*ra.K()*ra.H()/(ra.U()*(log(ra.g->R()/rr.g->R())
343             )-0.75));
344         Wei=3.1416*(ra.g->R()*ra.g->R()-rr.g->R()*rr.g->R())*ra.
345             Poro()*ra.H()*ra.Ct()*rr.Pi();
346     }
347     else //Caso Linear
348     {
349         //J=0.30662*ra.K()*ra.H()*ra.g->W()/(ra.U()*ra.g->L()); NAO
350             ENTENDEI
351         J = (ra.K()*ra.H()*ra.g->W()/(39.85*ra.U()*rr.g->L()));
352         Wei=ra.g->L()*ra.g->W()*ra.Poro()*ra.H()*ra.Ct()*rr.Pi();
353     }
354     Tempo();
355 }
356
357 double CFetkovich::EqSeg(double a, double b, double c)
358 {
359     double x1=(b+sqrt(b*b-4*a*c))/(a*(-2)); //Formulas de Bhaskara
360     double x2=(sqrt(b*b-4*a*c)-b)/(2*a);

```

```

358     if((x1>0)&&(x1<rr.Pi())) //Uma das raizes esta entre 0 e a Pressao
        Inicial, a outra esta fora deste intervalo
359     {
360         return x1;
361     }
362     else
363     {
364         return x2;
365     }
366 }
367
368
369 void CFetkovich::SolverGas()
370 {
371     Constantes();
372     vector<double> pm(n+1,0.0); //pressão média no aquífero
373     double D=Wei*(1-exp((-1)*J*rr.Pi()*dt/Wei))/rr.Pi(); //Constante
        simplificada
374     P[0]=pm[0]=rr.Pi();
375     double zi=fluido->BrillZ(P[0]);
376     //primeira iteracao
377     double z=zi;
378     //Define-se os coeficientes da Equacao do Segundo Grau
379     double c1=(-1)*z*rr.Pi()*(fluido->V()-rr.Q()*t[1])/zi;
380     double b1=fluido->V()-D*(pm[0]-P[0]/2)/fluido->B(P[0]);
381     double a1=D/(fluido->B(P[0])*2);
382     //Solucao da Equacao
383     P[1]=EqSeg(a1,b1,c1);
384     double pk; //Salva a pressao da ultima iteracao
385     do
386     {
387         pk=P[1];
388         z=fluido->BrillZ(P[1]);
389         c1=(-1)*z*rr.Pi()*(fluido->V()-rr.Q()*t[1])/zi;
390         P[1]=EqSeg(a1,b1,c1);
391     }while(fabs(P[1]-pk)>0.001);
392     Wen[1]=Wen[0]+D*(pm[0]-(P[0]+P[1])/2);
393     pm[1]=rr.Pi()*(1-Wen[1]/Wei);
394     //Iteracoes restantes
395     for(int i=2;i<=n;i++)
396     {
397         z=fluido->BrillZ(P[i-1]);
398         c1=(-1)*z*rr.Pi()*(fluido->V()-rr.Q()*t[i])/zi;
399         b1=fluido->V()-(Wen[i-1]/fluido->B(P[i-1]))-D*(pm[i-1]-P[i-1]/2)/fluido->B(P[i-1]);
400         a1=D/(fluido->B(P[i-1])*2);
401         P[i]=EqSeg(a1,b1,c1);
402         do

```



```

403     {
404         pk=P[i];
405         z=fluido->BrillZ(P[i]);
406         c1=(-1)*z*rr.Pi()*(fluido->V()-rr.Q()*t[i])/zi;
407         P[i]=EqSeg(a1,b1,c1);
408     }while(fabs(P[i]-pk)>0.001);
409     Wen[i]=Wen[i-1]+D*(pm[i-1]-(P[i-1]+P[i])/2);
410     pm[i]=rr.Pi()*(1-Wen[i]/Wei);
411 }
412 }
413
414 void CFetkovich::SolverOleo()
415 {
416     Constantes();
417     vector<double> pm(n+1,0.0); //pressão média no aquifero
418     double D=Wei*(1-exp((-1)*J*rr.Pi()*dt/Wei))/rr.Pi(); //Constante
419         simplificada
420     P[0]=pm[0]=rr.Pi();
421     for(int i=1;i<=n;i++)
422     {
423         double N=rr.Q()*t[i];
424         P[i]=(N*fluido->B()+(N*fluido->Co()-fluido->V()*fluido->Ceo()
425             )*fluido->B()*P[0]-Wen[i-1]-D*(pm[i-1]-P[i-1]/2))
426         /(fluido->B()*(N*fluido->Co()-fluido->V()*fluido->Ceo())-D/2)
427         ; //equacao explicita
428         Wen[i]=Wen[i-1]+D*(pm[i-1]-(P[i-1]+P[i])/2);
429         pm[i]=rr.Pi()*(1-Wen[i]/Wei);
430     }
431 }
432 }

```

Apresenta-se na listagem 6.7 o arquivo com código da classe CFluido.

Listing 6.7: Arquivo de cabeçalho da classe CFluido.

```

430 /** Previsao do Compartamento de Reservatorios
431     de Gas e de Oleo com Influxo de Agua
432     @author Carlos Andre Martins de Assis e Gabriel Clemente Franklin
433     @file CFluido.h
434 */
435
436 /** @brief Classe com as propriedades basicas de um fluido.
437     Esta classe e herdada pela classe CGas e COleo
438     @class CFluido
439 */
440
441 #ifndef CFluido_h
442 #define CFluido_h
443 #include <iostream>
444 #include <fstream>
445

```

```

446
447      //! Enumeracao com os tipos de fluidos
448      enum {egas=1, eoleo=2};
449
450 class CFluido
451 {
452     protected:
453         double d; ///< densidade
454         double v; ///< volume total
455         double b; ///< fator volume formacao
456     public:
457         ///
458         CFluido(double _d=0.0, double _v=0.0, double _b=0.0):d(_d),v(
            _v),b(_b){};
459         ///
460         CFluido(const CFluido& obj):d(obj.d),v(obj.v),b(obj.b){};
461         ///
            para arquivo
462         virtual void Entrada(std :: ostream & os , std :: istream &
            is);
463         virtual void Entrada(std :: ifstream & is);
464         virtual void Saida( std :: ostream & os );
465         ///
466         double D() {return d;};
467         ///
468         double V() {return v;};
469         ///
470         double B() {return b;};
471         ///
472         virtual double BrillZ(double p){};
473         ///
474         virtual double B(double P){};
475         ///
            herdeira COleo
476         virtual double Ceo(){};
477         ///
            COleo
478         virtual double Co(){};
479         ///
480         virtual ~CFluido(){};
481
482
483
484     };
485
486
487 #endif

```

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

Listing 6.8: Arquivo de implementação da classe CFluido.

```

488 /* Previsão do Compartamento de Reservatórios
489 de Gás e de Óleo com Influxo de Água
490
491 @Autores: Carlos André Martins de Assis e Gabriel Clemente Franklin
492 @file CFluido.cpp
493 */
494
495 #include "CFluido.h"
496 #include <iostream>
497 #include <fstream>
498
499 using namespace std;
500
501 void CFluido::Entrada(ostream & os ,istream & is)
502 {
503     os<<"Entre com a densidade:";
504     is>>d;
505     os<<"Entre com o volume inicial (m3 std):";
506     is>>v;
507     os<<"Entre com o fator Volume-Formacao (m3/m3 std):";
508     is>>b; is.get();
509 }
510
511 void CFluido::Entrada(ifstream & is)
512 {
513     is.ignore(100,':');
514     is>>d;
515     is.ignore(100,':');
516     is>>v;
517     is.ignore(100,':');
518     is>>b;
519 }
520
521 void CFluido::Saida(ostream & os)
522 {
523     os<<"Densidade:"<<d<<endl;
524     os<<"Volume inicial (m3 std):"<<v<<endl;
525     os<<"Fator Volume-Formacao (m3/m3 std):"<<b<<endl;
526 }

```

Apresenta-se na listagem 6.9 o arquivo com código da classe CGas.

Listing 6.9: Arquivo de cabeçalho da classe CGas.

```

527 /** Previsao do Compartamento de Reservatorios
528 de Gas e de Oleo com Influxo de Agua
529 @author Carlos Andre Martins de Assis e Gabriel Clemente Franklin

```

```

530  @file CGas.h
531  */
532
533  /** @brief Classe com as propriedade de um gas
534      //Herda as propriedade basicas de um fluido da classe CFluido
535      @class CGas
536  */
537  #ifndef CGas_h
538  #define CGas_h
539
540  #include "CFluido.h"
541
542  class CGas: public CFluido
543  {
544      private:
545          double Tpc;///< temperatura pseudocritica (é uma mistura)
546          double Ppc;///< temperatura pseudocritica (é uma mistura)
547          double z; ///< fator de compressibilidade
548          double t; ///< Temperatura do gás (mesma do reservatorio,
                    //aproximadamente constante na produção)
549
550
551      public:
552          ///<@brief Construtor default e sobrecarregado
553          CGas(double _d=0.0, double _v=0.0, double _B=0.0, double _Tpc
                    =0., double _Ppc=0., double _z=0., double _t=0.)
554              : CFluido(_d, _v, _B), Tpc(_Tpc), Ppc(_Ppc), z(_z), t(
                    _t){};
555          ///<@brief Construtor sobrecarregado de cópia
556          CGas(const CGas& obj): CFluido(obj), Tpc(obj.Tpc), Ppc(obj.Ppc
                    ), z(obj.z), t(obj.t){};
557
558
559          ///<@brief retorna o fator de compressibilidade
560          double Z() {return z;};
561          ///<@brief método polinomial para calculo de z
562          double PolinomioZ(double a, double b, double c, double p);
563          ///<@brief método de Brill & Begges para calculo de z
564          virtual double BrillZ(double p);
565          ///<@brief Calcula B para gás seco
566          virtual double B(double P);
567          ///<@brief Metodos de Entrada e Saida padrao, diferenciados
                    //para arquivo
568          virtual void Entrada( std :: ostream & os , std :: istream
                    & is);
569          virtual void Entrada(std :: ifstream & is);
570          virtual void Saida( std :: ostream & os );
571          ///<@brief Destrutor

```

```

572         virtual ~CGas(){};
573
574     };
575
576
577
578
579
580
581 #endif

```

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

Listing 6.10: Arquivo de implementação da classe `CGas`.

```

582 /* Previsão do Compartamento de Reservatórios
583 de Gás e de Óleo com Influxo de Água
584
585 @Autores: Carlos André Martins de Assis e Gabriel Clemente Franklin
586 @file CGas.cpp
587 */
588 #include <cmath>
589 #include "CGas.h"
590
591 using namespace std;
592
593
594 double CGas::BrillZ(double p)
595 {
596     double A=0.;
597     double B=0.;
598     double C=0.;
599     double D=0.;
600     double Ppr =0., Tpr=0.;
601     Ppr=p/Ppc;
602     Tpr=t/Tpc;
603
604     A=1.39*pow ((Tpr-0.92), 0.5 )-0.36*Tpr-0.101;
605     B= (0.62-0.23*Tpr)*Ppr+( 0.066/(Tpr-0.86) -0.037 )*pow(Ppr,2)+
        (0.32/(pow(10,9*(Tpr-1))))*pow(Ppr,6);
606     C=0.132-0.32*log10(Tpr);
607     D=pow(10,(0.3106-0.49*Tpr+0.1824*pow(Tpr,2)));
608
609     z=A+((1-A)/exp(B))+C*pow(Ppr,D);
610     return z;
611 }
612
613 double CGas::PolinomioZ(double a, double b, double c, double p)
614 {
615     z=a*p*p+b*p+c;

```

```

616
617     return z;
618
619 }
620
621
622 double CGas::B(double P)
623 {
624     b = ((1.0335122*z*t)/(288.89*P));
625
626     return b;
627 }
628
629 void CGas::Entrada(ostream & os ,istream & is)
630 {
631     os<<"\t-----\n\tCGas\n\t-----"<<endl;
632     CFluido::Entrada(os,is);
633     os<<"Entre com a Temperatura Pseudo-Critica Tpc (K): ";
634     is>>Tpc;
635     os<<"Entre com a Pressao Pseudo-Critica Ppc (kgf/cm2): ";
636     is>>Ppc;
637     os<<"Entre com o fator de compressibilidade z: ";
638     is>>z;
639     os<<"Entre com a Temperatura (K): ";
640     is>>t; is.get();
641 }
642
643 void CGas::Entrada(istream & is)
644 {
645
646     CFluido::Entrada(is);
647     is.ignore(100,':');
648     is>>Tpc;
649     is.ignore(100,':');
650     is>>Ppc;
651     is.ignore(100,':');
652     is>>z;
653     is.ignore(100,':');
654     is>>t;
655 }
656
657 void CGas::Saida(ostream & os)
658 {
659     os<<"\t-----\nPropriedades do Gas\n\t-----"<<endl;
660     CFluido::Saida(os);
661     os<<"Temperatura Pseudo-Critica Tpc (K): "<<Tpc<<endl;
662     os<<"Pressao Pseudo-Critica Ppc (kgf/cm2): "<<Ppc<<endl;
663     os<<"Fator de Compressibilidade z: "<<z<<endl;

```

```

664         os<<"Temperatura_□(K):_□"<<t<<endl;
665     }

```

Apresenta-se na listagem 6.11 o arquivo com código da classe CGeometria.

Listing 6.11: Arquivo de cabeçalho da classe CGeometria.

```

666 /** Previsao do Compartamento de Reservatorios
667 de Gas e de Oleo com Influxo de Agua
668 @author Carlos Andre Martins de Assis e Gabriel Clemente Franklin
669 @file CGeometria.h
670 */
671
672
673
674 /** @brief Classe abstrata para uma geometria.
675 *Classe abstrata herdada pelas geometrias radiais e lineares
676 @class CGeometria
677 */
678 #ifndef CGeometria_h
679 #define CGeometria_h
680
681 #include <iostream>
682
683 ///@brief enumeracao para tipo de geometria
684 enum {eradial=1, elinear=2};
685
686 class CGeometria
687 {
688     public:
689         ///@brief Metodos de Entrada e Saida padrao, diferenciados
para arquivo
690         virtual void Entrada( std :: ostream & os , std :: istream
            & is)=0;
691         virtual void Entrada(std :: ifstream & is)=0;
692         virtual void Saida( std :: ostream & os )=0;
693         ///@brief Metodos das classes herdeiras
694         virtual double R(){return 0;};
695         virtual double L(){return 0;};
696         virtual double W(){return 0;};
697         ///@brief Destrutor
698         virtual ~CGeometria(){};
699 };
700
701 #endif

```

Apresenta-se na listagem 6.12 o arquivo com código da classe CHurst.

Listing 6.12: Arquivo de cabeçalho da classe CHurst.

```

703 /** Previsao do Compartamento de Reservatorios

```

```

704  de Gas e de Oleo com Influxo de Agua
705  @author Carlos Andre Martins de Assis e Gabriel Clemente Franklin
706  @file CHurst.h
707 */
708
709
710 /** @brief Classe para o metodo de Van Everdingen and Hurst
711     /// Implemente o metodo de van Everdigen & Hurst para gas e oleo
712     @class CHurst
713 */
714
715
716 #ifndef CHurst_h
717 #define CHurst_h
718
719 #include "CSimulador.h"
720
721
722
723 using namespace std;
724
725 class CHurst: public CSimulador
726 {
727 {
728     protected:
729         double U; ///< Constante de influxo de agua do aquifero
730         vector<double>td; ///< tempo adimensional
731
732     public:
733
734
735         ///< @brief Construtor para o caso gás
736         CHurst(int _n,int _op,double _dt,double _pi,double _poro ,
737             double _h,double _sw,double _Q,double _T,double _ro ,
738             double _d,
739             double _v,double _B,double _Tpc, double _Ppc,double
740             _z,double a_pi,
741             double a_poro, double a_h,double _k,double _ct,
742             double _u,double _re)
743             :CSimulador (_n,_op,_dt,_pi,_poro,_h,_sw,_Q,_T,_ro,_d,_v,
744                 _B,_Tpc,_Ppc,_z,a_pi,a_poro,a_h,_k,_ct,_u,
745                 _re),td(_n+1)
746             {SolverGas(); titulo = "Hurst";}
747
748         ///< @brief construtor para o caso óleo
749         CHurst(int _n,int _op,double _dt,double _pi,double _poro ,
750             double _h,double _sw,double _Q,double _ro,double _d,
751             double _v,double _B,double _ceo, double _co,double

```



```

748         a_pi, double a_poro,
           double a_h, double _k, double _ct, double _u, double _re
           )
749         : CSimulador (_n, _op, _dt, _pi, _poro, _h, _sw, _Q, _ro, _d, _v,
750                     _B, _ceo, _co, a_pi, a_poro, a_h, _k, _ct, _u, _re),
           td(_n+1)
751         { SolverOleo(); titulo = "Hurst"; }
752
753         ///brief construtor usado no arquivo da interface
754         CHurst(int size): CSimulador(size)
755         { titulo = "Hurst"; }
756
757         ///brief construtor para arquivo do prompt
758         CHurst(int _n, int _op, double _dt, int _geom): CSimulador(_n,
759                     _op, _dt, _geom), td(_n+1)
760         { titulo = "Hurst"; }
761
762         ///brief calcula Wd, usando as classes ClaplaceWd e CStefehst
763         double InfluxoWd(double t);
764         ///brief Calcula o vetor tempo adimensional e tempo
765         virtual void Tempo();
766         ///brief Inicializa as constantes para caso Radial ou Linear
767         virtual void Constantes();
768         ///brief Resolve o problema para o caso do gas
769         virtual void SolverGas();
770         ///brief Resolve o problema para o caso do oleo
771         virtual void SolverOleo();
772         ///brief Destrutor
773         virtual ~CHurst(){}
774
775     };
776
777 #endif

```

Apresenta-se na listagem 6.13 o arquivo de implementação da classe CHurst.

Listing 6.13: Arquivo de implementação da classe CHurst.

```

778 /* Previsão do Compartamento de Reservatórios
779 de Gás e de Óleo com Influxo de Água
780
781 @Autores: Carlos André Martins de Assis e Gabriel Clemente Franklin
782 @file CHurst.cpp
783 */
784
785 #include "CHurst.h"
786 #include "CGnuplot.h"
787
788 void CHurst::Constantes()

```

```

789 {
790     if (geom==1) //Caso Radial
791     {
792         U=ra.Poro()*ra.Ct()*ra.H()*2*3.1416*rr.g->R()*rr.g->R();
793     }
794     else //Caso Linear
795     {
796         U=ra.Poro()*ra.Ct()*ra.H()*ra.g->W()*ra.g->L();
797     }
798     Tempo();
799 }
800
801 double CHurst::InfluxoWd(double t)
802 {
803     CStehfest ob;
804     if (geom==1) //Caso Radial
805     {
806
807         switch(op)
808         {
809             case 1:
810                 return ob.Inversao(CLaplaceWd::RadInf,t); break;
811             case 2:
812                 return ob.Inversao(CLaplaceWd::RadRea,ra.g->R(),rr.g->R
813                     (),t); break;
814             case 3:
815                 return ob.Inversao(CLaplaceWd::RadSel,ra.g->R(),rr.g->R
816                     (),t); break;
817             default:
818                 //CASO PADRAO
819                 !!!!
820
821                 return ob.Inversao(CLaplaceWd::RadInf,t); break;
822         }
823     }
824     else //Caso Linear
825     {
826         switch(op)
827         {
828             case 1:
829                 return CLaplaceWd::LinInf(t); break;
830             case 2:
831                 return ob.Inversao(CLaplaceWd::LinRea,t); break;
832             case 3:
833                 return ob.Inversao(CLaplaceWd::LinSel,t); break;
834             default:
835                 //CASO PADRAO
836                 !!!!

```

```

833         return CLaplaceWd::LinInf(t); break;
834     }
835 }
836 }
837
838 void CHurst::Tempo()
839 {
840     CSimulador::Tempo();
841     double r2; // Constante que depende da geometria
842     if(geom==1)
843     {
844         r2=rr.g->R()*rr.g->R();
845     }
846     else
847     {
848         r2=ra.g->L()*ra.g->L();
849     }
850     td[0]=0.;
851     double a=0.008362;
852     for (int i=1;i<=n;i++)
853     {
854         td[i]=(ra.K()*dt*a*i)/(ra.Poro()*ra.U()*ra.Ct()*r2);
855     }
856 }
857
858 }
859
860 void CHurst::SolverGas()
861 {
862     Constantes();
863     P[0]=rr.Pi();
864     double zi=fluido->BrillZ(P[0]);
865     double soma=0.;
866     double z=zi;
867     vector<double> dp(n);
868
869     Wen[0]=0.; //lebrando que P[0]=pi
870
871     double pk=0.; // Pn da iteração anterior para comparacao
872
873     //Inicio da Primeira iteração
874
875     P[1]=P[0]*z*(1-(rr.Q()*dt)/fluido->V())/(zi*(1-U*soma/(fluido->V()*
        fluido->B(P[0]))));
876     do
877     {
878         //passo 3)
879         soma=((P[0]-P[1])/2)*InfluxoWd(td[1]);

```

```

880     Wen[1]=U*soma;
881     z=fluido->BrillZ(P[1]);
882     pk=P[1];
883     P[1]=P[0]*z*(1-((rr.Q()*dt)/fluido->V()))/(zi*(1-Wen[1]/(fluido->V())
        *fluido->B(P[0])))); //CHECAR!
884     dp[0] =(P[0]-P[1])/2;
885     }while(fabs(P[1]-pk)>0.001);
886     for (int i=2;i<=n;i++)
887     {
888         soma=0.;
889         for (int j=0;j<=i-2;j++)
890         {
891             soma+=dp[j]*InfluxoWd(td[i]-td[j]);
892         }
893         //Passo 2)
894         z=fluido->BrillZ(P[i-1]);
895         P[i]=P[0]*z*(1-(rr.Q()*dt*i)/fluido->V()))/(zi*(1-U*soma/(fluido->V())*
            fluido->B(P[i-1]))));
896
897         //PASSO 3)
898         do
899         {
900             z=fluido->BrillZ(P[i]);
901
902             Wen[i]=U*soma+U*(P[i-2]-P[i])*InfluxoWd(td[i]-td[i-1])/2;
903             pk=P[i];
904             P[i]=P[0]*z*(1-(rr.Q()*dt*i)/fluido->V()))/(zi*(1-Wen[i]/(fluido
                ->V()*fluido->B(P[i]))));
905
906
907             dp[i-1]=(P[i-2]-P[i])/2;
908
909         }while (fabs(P[i]-pk)>0.001);
910     }
911 }
912
913 void CHurst::SolverOleo()
914 {
915     Constantes();
916     op=1; //0 metodo simplificado de Hurst só funciona para aquifero
        infinito e subsaturado (Fetkovich tambem)
917     vector<double> dp(n);
918     P[0]=rr.Pi();
919     P[1]=(P[0]*(U*InfluxoWd(td[1])+fluido->V()*fluido->B()*fluido->Ceo()
        )-rr.Q()*t[1]*fluido->B()))
920         /(fluido->V()*fluido->B()*fluido->Ceo()+U*InfluxoWd(td[1]));
        //Equacao explicita
921     Wen[1]=rr.Q()*t[1]*fluido->B()-fluido->V()*fluido->B()*fluido->Ceo()

```

```

        *(P[0]-P[1]);
922     dp[0] =(P[0]-P[1])/2;
923
924     for(int i=2;i<=n;i++)
925     {
926         double soma=0.;
927         for (int j=0;j<=i-2;j++)
928         {
929             soma+=U*dp[j]*InfluxoWd(td[i]-td[j]);
930         }
931         P[i]=(soma+U*P[i-2]*InfluxoWd(td[i]-td[i-1])/2+fluido->V()*
            fluido->B()*fluido->Ceo()*P[0]-rr.Q()*t[i]*fluido->B())
932         /(fluido->V()*fluido->B()*fluido->Ceo()+U*InfluxoWd(td[i]-td
            [i-1])/2);
933         Wen[i]=rr.Q()*t[i]*fluido->B()-fluido->V()*fluido->B()*
            fluido->Ceo()*(P[0]-P[i]);
934         dp[i-1]=(P[i-2]-P[i])/2;
935     }
936 }

```

Apresenta-se na listagem 6.14 o arquivo com código da classe CInterface.

Listing 6.14: Arquivo de cabeçalho da classe CInterface.

```

939 /** Previsao do Compartamento de Reservatorios
940 de Gas e de Oleo com Influxo de Agua
941 @author Carlos Andre Martins de Assis e Gabriel Clemente Franklin
942 @file CInterface.h
943 */
944
945 /** @brief Classe de interface em modo texto.
946 @class CInterface
947 */
948 #ifndef CInterface_h
949 #define CInterface_h
950
951 #include <string>
952 #include "CSimulador.h"
953
954 enum {ehurst =0, fet = 1};
955
956 class CInterface
957 {
958     private:
959
960         string linha;
961         string erro;
962         string nomeArq; ///< Pega do teclado o nome do arquivo
963         int p; ///< Atributo que controla o menu
964         int tipo; ///< Tipo de fluido, Gas(1) ou Oleo(2)

```

```

965         int geom;          ///< Tipo de Geometria, Radial(1) ou
                          Linear(2)
966         int metodo;      ///< Metodo de calculo, Hurst(1) ou
                          Fetkovich(2)
967         ifstream *fin;    ///< arquivo de entrada
968         CSimulador *Sim1; ///< Necessario para o polimorfismo,
                          ponteiro para a base CSimulador
969
970         //Estes proximos objetos sao criados apenas para usar seus
                          metodos de entrada;
971         CReservatorio *rr; ///< rocha reservatorio
972         CFluido *fluido;  ///< fluido que satura a rocha
                          reservatorio
973         CAquifero *ra;     ///< rocha aquifero
974
975
976     public:
977
978         ///<@brief Construtor
979         CInterface():linha("\n-----\n")
                          ,erro("\n\tOpcao inválida. Tente novamente!"),
980                             p(0),tipo(0),geom(0),fin(NULL),rr(NULL),ra(
                          NULL),fluido(NULL){Inicio();};
981
982         ///<@brief Inicia a interacao com o usuario.
983         void Inicio();
984         ///<@brief Pergunta ao usuario o que ele deseja fazer com os
                          dados simulados
985         void MostrarResultados();
986         ///<@brief Pergunta o tipo de grafico a ser gerado
987         void Comparacao();
988         ///<@brief Seta os dados de simulacao e executa a simulacao
989         void Set();
990         ///<@brief Salva os dados fornecidos pelo usuario via
                          teclado
991         void SalvarDados();
992         ///<@brief Salva os resultados da simulacao
993         void SalvarSimulacao();
994         ///<@brief Valida a entrada do usuario via teclado
995         void ValidaEntrada( int &opcao,int min,int max=200);
996         ///<@brief Destrutor
997         ~CInterface();
998
999 };
1000
1001 #endif

```

Apresenta-se na listagem 6.15 o arquivo de implementação da classe CInterface.

Listing 6.15: Arquivo de implementação da classe CInterface.

```

1002 /* Previsão do Compartamento de Reservatórios
1003 de Gás e de Óleo com Influxo de Água
1004
1005 @Autores: Carlos André Martins de Assis e Gabriel Clemente Franklin
1006 @file CInterface.cpp
1007 */
1008
1009 #include "CInterface.h"
1010 #include "CGnuplot.h"
1011 #include "CHurst.h"
1012 #include "CFetkovich.h"
1013
1014
1015 using namespace std;
1016
1017 void CInterface::Inicio()
1018 {
1019
1020     cout<<linha<<"\tComo_era_entrada_de_dados_de_poco?"<<linha
1021     <<"1) Via_arquivo_de_disco;\n2) Via_teclado;"<<endl;
1022     ValidaEntrada(p,1, 2);
1023     switch(p)
1024     {
1025     case 1:
1026         {
1027             cout<<"\nDigite_o_nome_do_arquivo:"<<endl;
1028             cin>>nomeArq; cin.get();
1029             fin=new ifstream(nomeArq.c_str());
1030             while(!fin->is_open())
1031             {
1032                 cout<<"\nArquivo_inexistente!\n
1033                 \nDigite_o_nome_do_arquivo:"<<endl
1034                 ;
1035                 cin>>nomeArq; cin.get();
1036                 fin->open(nomeArq.c_str());
1037             }
1038         }
1039     case 2:
1040         {
1041             cout<<linha<<"\tQual_a_geometria_das_rochas?"<<linha
1042             <<"1) Radial;\n2) Linear;"<<endl;
1043             ValidaEntrada( geom,1, 2);
1044             cout<<linha<<"\tQual_o_fluido_que_preenche_o_reservatorio
1045             ?:"<<linha

```

```

1046         ValidaEntrada(tipo,egas,eoleo);
1047         if(tipo==egas){fluido=new CGas;}
1048         else {fluido=new COleo;}
1049         rr=new CReservatorio(geom); //rocha reservatorio
1050         ra=new CAquifero(geom); //rocha aquifero
1051         rr->Entrada(cout,cin);
1052         fluido->Entrada(cout,cin);
1053         ra->Entrada(cout,cin);
1054         cin.get();
1055         SalvarDados();
1056         fin=new ifstream(nomeArq.c_str());
1057         break;
1058     }
1059 }
1060
1061 Set();
1062 }
1063
1064 void CInterface::Set()
1065 {
1066     int n,op=3,f;
1067
1068     cout<<linha<<"Escolha os parametros da simulacao:"<<linha<<endl;
1069     cout<<"\nEntre com n (numero de anos):";
1070     ValidaEntrada(n,0);
1071     cout<<"\nQue fracao de ano sera o passo? (ex.: 1=1 ano, 4=1/4 de
        ano...):";
1072     ValidaEntrada(f,1);
1073     cout<<"\nEntre com o metodo:\n0-Hurst\n1-Fetkovich:";
1074     ValidaEntrada(metodo,0,1);
1075     //cria o objeto de simulacao de acordo com o metodo
1076     (*fin)>>geom;
1077     if (metodo == ehurst)
1078     {
1079         Sim1=new CHurst(f*n,op,365/f,geom);
1080     }else if (metodo == fet)
1081     {
1082         Sim1 = new CFetkovich (f*n,op,365/f,geom);
1083     }
1084
1085     (*fin)>>(*Sim1);
1086
1087     fin->close();
1088
1089     if ( metodo == ehurst && Sim1->Tipo() == egas)
1090     {
1091         cout<<"\nEntre com o fluxo:\n1-Infinito\n2-Realimentado\n3-Selado:"
            ;

```



```

1092     ValidaEntrada(op,1,3);
1093 }
1094
1095     Sim1->SetInfluxo(op);
1096     //chama o solver de acordo com o fluido
1097     if (Sim1->Tipo() == egas)
1098     {
1099         Sim1->SolverGas();
1100     }else
1101     {
1102         Sim1->SolverOleo();
1103     }
1104
1105
1106     MostrarResultados();
1107 }
1108
1109 void CInterface::MostrarResultados()
1110 {
1111     while(p)
1112     {
1113         system("cls");
1114         cout<<linha<<"\t-Menu -\nEscolha uma das opcoes:"<<linha
1115         <<"1)Mostrar Simulacao\n2) Grafico da solucao\n3) Exportar
1116             simulacao como txt\n4) Dados da Simulacao"<<endl
1117         <<"0) Sair\n0 que deseja fazer?"<<endl;
1118         ValidaEntrada(p,0,4);
1119         switch(p)
1120         {
1121             case 0:
1122                 break;
1123             case 1:
1124                 Sim1->Resultados(cout);
1125                 system("pause"); break;
1126             case 2:
1127                 Comparacao(); break;
1128             case 3:
1129                 SalvarSimulacao(); break;
1130             case 4:
1131                 cout<<*Sim1;
1132                 system("pause"); break;
1133         }
1134     }
1135
1136
1137 void CInterface::Comparacao()
1138 {

```

```

1139     int tipo=4;
1140     cout<<endl<<"Escolha o tipo de grafico:"<<endl;
1141     cout<<"0) Pressao no contato x Tempo (dias)"<<endl<<"1) Influxo
        acumulado x Tempo (dias)"
1142     <<endl<<"2) Influxo acumulado x Pressao no contato"<<endl;
1143     ValidaEntrada(tipo,0,2);
1144     Sim1->Grafico(tipo);
1145 }
1146
1147 void CInterface::SalvarDados()
1148 {
1149     cout<<"\nOs dados de pouco serao salvos em disco para uso posterior.
        \nDigite o nome do arquivo:"<<endl;
1150     getline(cin,nomeArq);
1151     ofstream fout(nomeArq.c_str());
1152     fout<<geom<<endl;
1153     fout<<tipo<<endl;
1154     rr->Saida(fout);
1155     fluido->Saida(fout);
1156     ra->Saida(fout);
1157     fout.close();
1158 }
1159
1160 void CInterface::SalvarSimulacao()
1161 {
1162     cout<<"\n\nEntre com o nome do arquivo:";
1163     getline(cin,nomeArq);
1164     ofstream fout(nomeArq.c_str());
1165     Sim1->Resultados(fout);
1166     fout.close();
1167     cout<<"\n\nSimulacao Exportada com sucesso!"<<endl;
1168     system("pause");
1169 }
1170
1171
1172 void CInterface::ValidaEntrada(int &opcao,int min, int max)
1173 {
1174     while (true)
1175     {
1176         cin>>opcao;
1177
1178         if((opcao>=min) && (opcao<=max) && cin.good())
1179         {
1180             cin.get();
1181             break;
1182         }else
1183         {
1184             cin.clear();
1185             string entrada;
1186             getline(cin,entrada);

```

```

1185                                     cout<<erro<<endl;
1186
1187                                     }
1188                                 }
1189 }
1190
1191 CInterface::~CInterface()
1192 {
1193     if(fin!=NULL) delete fin;
1194     if(rr!=NULL) delete rr;
1195     if(ra!=NULL) delete ra;
1196     if(fluido!=NULL) delete fluido;
1197     if(Sim1!=NULL) delete Sim1;
1198 }

```

Apresenta-se na listagem 6.16 o arquivo com código da classe CInterfaceCGGS.

Listing 6.16: Arquivo de cabeçalho da classe CInterfaceCGGS.

```

1200 #ifndef CInterfaceCGGS_h
1201 #define CInterfaceCGGS_h
1202
1203 #include <iostream>
1204 #include "CSimuladorCGGS.h"
1205 #include <cstdlib>
1206 #include <fstream>
1207 #include <functional>
1208 #include <algorithm>
1209 #include <vector>
1210 #include <string>
1211 #include <iterator>
1212 #include "CGnuplot.h"
1213 #include <locale.h>
1214 #include <utility>
1215 #include <cmath>
1216 #include <iomanip>
1217 #include "CDataProd.h"
1218 #include "CMuskat.h"
1219 #include "CRK4.h"
1220
1221 using namespace std;
1222
1223 class CInterfaceCGGS
1224 {
1225     protected:
1226         CDataProd* cdp;
1227         CMuskat* mskt;
1228         CRK4* rk4;
1229     public:
1230         CInterfaceCGGS() {Run();}

```

```

1231     void Instrucoes();
1232     void SalvarSim();
1233     void Grafico();
1234     void Resultados();
1235     void Solver();
1236     void EntradaManual();
1237     void EntradaDisco();
1238     void Inicio();
1239     void Run();
1240 };
1241 #endif

```

Apresenta-se na listagem 6.17 o arquivo de implementação da classe CInterfaceCGGS.

Listing 6.17: Arquivo de implementação da classe CInterfaceCGGS.

```

1242 #include "CInterfaceCGGS.h"
1243
1244 void CInterfaceCGGS::Instrucoes(){
1245     cout<<"
        =====
        "<<endl;
1246     cout<<"Simulador de Previsão de Comportamento de
        Reservatórios de Óleo com Capa de Gás ou Gás em
        Solução"<<endl;
1247     cout<<"Programação Prática 2016.2 / LENEP - UENF"<<endl;
1248     cout<<"Autor: Thiago Couto de Almeida Chaves"<<endl;
1249     cout<<"Licença GPL 2.0"<<endl;
1250     cout<<"
        =====
        "<<endl;
1251     cout<<endl;
1252     cout<<"=====
        INSTRUÇÕES DE USO
        =====
        "<<endl;
1253     cout<<"Bem-vindo ao simulador, para utilizá-lo tenha em
        mãos os dados PVT do fluido de reservatório assim
        como dados de produção"<<endl;
1254     cout<<"& reservatório para começar. Você pode inserir
        tais dados pelo teclado, porém é altamente
        recomendado que a entrada seja"<<endl;
1255     cout<<"por arquivo de disco já que para uma simulação
        adequada muitos dados são requeridos."<<endl;
1256     cout<<endl;
1257     cout<<"Para inserir como arquivo de disco, olhe o modelo
        exemplo que segue com esse programa e somente EDITE
        os dados ali"<<endl;
1258     cout<<"demonstrados. Caso altere o formato do arquivo,
        terá que alterar o código também. Após editar os

```

```

        dados_que_queres, _salve"<<endl;
1259     cout<<"o_arquivo_no_mesmo_diretório_do_programa._No_
        programa, _digite_o_nome_do_arquivo_e_extensão_para_
        começar_a_simulação."<<endl;
1260     cout<<endl;
1261     cout<<"Todos_os_dados, _para_resultar_em_uma_simulação_
        real_e_de_proveito, _precisarão_estar_no_sistema_
        PETROBRAS_de_unidades"<<endl;
1262     cout<<"
        =====
        "<<endl;
1263     cout<<endl;
1264 }
1265 void CInterfaceCGGS::SalvarSim(){
1266     cout<<"Deseja_salvar_a_simulação?_1-SIM_2-NÃO"<<endl;
1267     int op;
1268     cin>>op; cin.get();
1269     if (op==1){
1270         cout<<"Qual_o_nome_de_arquivo_que_desejas_salvar
            ?"<<endl;
1271         string s;
1272         getline(cin,s);
1273         s=s+".txt";
1274         ofstream fout(s.c_str());
1275         cout<<"Cheguei_aq"<<endl;
1276         cdp->SaidaDisco(fout);
1277         mskt->SaidaDisco(fout);
1278         rk4->SaidaDisco(fout);
1279     }
1280 };
1281 void CInterfaceCGGS::Grafico(){
1282     mskt->Grafico();
1283     cout<<"Mostrando_agora_o_gráfico_da_Saturação_de_Óleo_
        versus_Pressão"<<'\\n';
1284     rk4->Grafico();
1285 }
1286 void CInterfaceCGGS::Resultados(){
1287     cout<<"Deseja_ver_a_saída_completa_ou_resumida_dos_
        dadosx?"<<endl;
1288     int op; cin>>op; cin.get();
1289     if (op==1){
1290         mskt->Saida(cout);
1291         rk4->Saida(cout, cin);
1292     }
1293     if (op==2){
1294         mskt->SaidaResumida();
1295     }
1296 };

```

```

1297     void CInterfaceCGGS::Solver(){
1298         rk4->Solver();
1299         cout<<"Sequência de cálculo terminada!"<<endl;
1300     }
1301     void CInterfaceCGGS::EntradaManual(){
1302         cout<<"Entrada Manual selecionada!"<<endl;
1303         cdp= new CDataProd;
1304         mskt= new CMuskat(*cdp);
1305         mskt->Entrada();
1306         rk4= new CRK4(*mskt);
1307     }
1308     void CInterfaceCGGS::EntradaDisco(){
1309         string nome;
1310         cout<<"Entrada por Disco selecionada!"<<endl;
1311         cout<<"Qual nome do arquivo + extensão (i.e. arquivo ou
            arquivo.dat)?"<<'\\n';
1312         getline(cin,nome);
1313         //CDataProd dados(nome);
1314         cdp= new CDataProd(nome);
1315         //CMuskat obj(nome,dados);
1316         mskt= new CMuskat(nome,*cdp);
1317         //CRK4 eq(obj);
1318         rk4= new CRK4(*mskt);
1319     }
1320     void CInterfaceCGGS::Inicio(){
1321         int op;
1322         cout<<"Como se dará a entrada de dados? Digite 1-Manual
            por Teclado / 2-Arquivo de Disco"<<endl;
1323         cin>>op; cin.get();
1324         if (op==1){
1325             EntradaManual();
1326         }
1327         if(op==2){
1328             EntradaDisco();
1329         }
1330         if(op!=1 && op!=2)
1331         {
1332             cout<<"Erro! Tipo de Entrada não encontrado";
1333             exit(0);
1334         }
1335     }
1336     void CInterfaceCGGS::Run(){
1337         Instrucoes(); Inicio(); Solver(); Resultados(); Grafico
            (); SalvarSim();
1338     }

```

Apresenta-se na listagem 6.18 o arquivo com código da classe CLaplaceWd.

Listing 6.18: Arquivo de cabeçalho da classe CLaplaceWd.

```

1339 /** Previsao do Compartamento de Reservatorios
1340 de Gas e de Oleo com Influxo de Agua
1341 @author Carlos Andre Martins de Assis e Gabriel Clemente Franklin
1342 @file CLaplaceWD.h
1343 */
1344 /** @brief Classe que gera influxo adimensional wd no Campo de
1345 Laplace
1346 Bibliografia: Previsao de Comportamento de Reservatorios
1347 de Petroleo. Apendices F,G e H.
1348 @class CLaplaceWd
1349 */
1350
1351 #ifndef CLaplaceWd_h
1352 #define CLaplaceWd_h
1353
1354
1355 class CLaplaceWd
1356 {
1357     public:
1358     ///@brief Acha a solucao no campo de Laplace para o caso Radial
1359     Infinito
1360     static double RadInf(double x);
1361     ///@brief Acha a solucao no campo de Laplace para o caso Radial
1362     Realimentado
1363     static double RadRea(double x,double re,double ro);
1364     ///@brief Acha a solucao no campo de Laplace para o caso Radial
1365     Selado
1366     static double RadSel(double x,double re,double ro);
1367     ///@brief Acha a solucao no campo de Laplace para o caso Linear
1368     Infinito
1369     static double LinInf(double x);
1370     ///@brief Acha a solucao no campo de Laplace para o caso Linear
1371     Realimentado
1372     static double LinRea(double x);
1373     ///@brief Acha a solucao no campo de Laplace para o caso Linear
1374     Selado
1375     static double LinSel(double x);
1376
1377 };
1378 #endif

```

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

Listing 6.19: Arquivo de implementação da classe CLaplaceWd.

```

1379 /* Previsão do Compartamento de Reservatórios
1380 de Gás e de Óleo com Influxo de Água
1381
1382 @Autores: Carlos André Martins de Assis e Gabriel Clemente Franklin
1383 @file CLaplaceWd.cpp
1384 */
1385
1386 #include <iostream>
1387 #include <cmath>
1388 #include <vector>
1389 #include <algorithm>
1390
1391 #include "specialfunctions.h" //funções deessel-biblioteca da internet
1392 #include "CLaplaceWd.h"
1393
1394 using namespace std;
1395 using namespace alglib;
1396
1397
1398
1399 double CLaplaceWd::RadInf(double x) // Aquifero Radial Infinito
1400 {
1401
1402     double a=sqrt(x);
1403     double b=pow(a,3);
1404
1405     return  besseli0(a)/(b*besseli1(a)); //besseli0 e besseli1 da
        biblioteca specialfunctions
1406 };
1407
1408 double CLaplaceWd::RadRea(double x,double re,double ro) //Aquifero
    Radial Realimentado
1409 {
1410
1411
1412     double a=sqrt(x);
1413     double b=pow(a,3);
1414     double r=re/ro;
1415
1416     return  ( besseli0(r*a)*besseli1(a)+besseli1(a)*besseli0(r*a) )/(
        b*( besseli0(a)*besseli0(r*a)-besseli1(a)*besseli1(r*a) ) );
1417 };
1418
1419
1420

```



```

1421 double CLaplaceWd::RadSel(double x,double re,double ro) //aquifero
      Radial Selado
1422 {
1423
1424     double a=sqrt(x);
1425     double b=pow(a,3);
1426     double r=re/ro;
1427
1428     return ( besseli1(r*a)*besselk1(a)-besseli1(a)*besselk1(r*a) )/( b*(
        besseli0(a)*besselk1(r*a)+besselk0(a)*besseli1(r*a) ) );
1429 };
1430
1431 double CLaplaceWd::LinInf(double x) //aquifero Linear Infinito
1432 {
1433     return 2*sqrt(x/3.1416);
1434 };
1435 double CLaplaceWd::LinRea(double x) //aquifero Linear Realimentado
1436 {
1437
1438     double a=sqrt(x);
1439     double b=pow(a,3);
1440
1441
1442
1443     return ( (1+exp(-2*a))/( b*( 1-exp(-2*a) ) ) );
1444
1445 };
1446
1447 double CLaplaceWd::LinSel(double x) //Aquifero linear selado
1448 {
1449     double a=sqrt(x);
1450     double b=pow(a,3);
1451
1452
1453
1454
1455
1456     return ( (1-exp(-2*a) )/( b*( 1+exp(-2*a)) ) );
1457 };

```

Apresenta-se na listagem 6.20 o arquivo com código da classe CLinear.

Listing 6.20: Arquivo de cabeçalho da classe CLinear.

```

1458 /** Previsao do Compartamento de Reservatorios
1459     de Gas e de Oleo com Influxo de Agua
1460     @author Carlos Andre Martins de Assis e Gabriel Clemente Franklin
1461     @file CLinear.h
1462 */
1463 /** @brief Classe que representa uma geometria Linear

```

```

1464         @class CLinear
1465 */
1466
1467 #ifndef CLinear_h
1468 #define CLinear_h
1469
1470 #include "CGeometria.h"
1471
1472 class CLinear: public CGeometria
1473 {
1474     protected:
1475         double w; ///< Largura
1476         double l; ///< Comprimento
1477     public:
1478         ///< @brief Construtor default e sobrecarregado
1479         CLinear(double _w=0.0, double _l=0.0):w(_w),l(_l){};
1480         ///< @brief Construtor sobrecarregado de copia
1481         CLinear(const CLinear &obj):w(obj.w),l(obj.l){};
1482         ///< @brief Metodos de Entrada e Saida padrao, diferenciados
            para arquivo
1483         virtual void Entrada( std :: ostream & os , std :: istream
            & is);
1484         virtual void Entrada(std :: ifstream & is);
1485         virtual void Saida( std :: ostream & os );
1486         ///< @brief Membros publicos que retornam os parametros
1487         virtual double L(){return l;};
1488         virtual double W(){return w;};
1489         ///< @brief Destrutor
1490         virtual ~CLinear(){};
1491 };
1492
1493 #endif

```

Apresenta-se na listagem 6.21 o arquivo de implementação da classe CLinear.

Listing 6.21: Arquivo de implementação da classe CLinear.

```

1494 /* Previsão do Compartamento de Reservatórios
1495    de Gás e de Óleo com Influxo de Água
1496
1497    @Autores: Carlos André Martins de Assis e Gabriel Clemente Franklin
1498    @file CLinear.cpp
1499 */
1500
1501 #include "CLinear.h"
1502 #include <iostream>
1503 #include <fstream>
1504
1505 using namespace std;
1506

```

```

1507 void CLinear::Entrada(ostream & os ,istream & is)
1508 {
1509     os<<"Entre com a Largura(m): ";
1510     is>>w;
1511     os<<"Entre com o Comprimento(m): ";
1512     is>>l; is.get();
1513
1514 }
1515
1516 void CLinear::Saida(ostream & os)
1517 {
1518     os<<"Largura(m): " <<w<<endl;
1519     os<<"Comprimento(m): " <<l<<endl;
1520
1521 }
1522
1523 void CLinear::Entrada(istream & is)
1524 {
1525     is.ignore(100, ':');
1526     is>>w;
1527     is.ignore(100, ':');
1528     is>>l;
1529 }

```

Apresenta-se na listagem 6.22 o arquivo com código da classe CMuskat.

Listing 6.22: Arquivo de cabeçalho da classe CMuskat.

```

1530 #ifndef CMuskat_h
1531 #define CMuskat_h
1532
1533 #include<iostream>
1534 #include<cstdlib>
1535 #include<fstream>
1536 #include<functional>
1537 #include<algorithm>
1538 #include<vector>
1539 #include<string>
1540 #include<iterator>
1541 #include "CGnuplot.h"
1542 #include<utility>
1543 #include<cmath>
1544 #include<iomanip>
1545 #include "CSimuladorCGGS.h"
1546 #include "CDataProd.h"
1547
1548 class CMuskat: protected CSimuladorCGGS //implementar método para
    devolução da string da Eq. de Muskat
1549
1550 {

```

```

1551     protected:
1552     std::vector<double> eta; ///Parâmetro Eta da Eq. de Muskat
1553     std::vector<double> alfa; ///Parâmetro Alfa da Eq. de Muskat
1554     std::vector<double> lambda; ///Parâmetro Lambda da Eq. de Muskat
1555     std::vector<double> psi; ///Parâmetro Psi da Eq. de Muskat
1556     std::vector<double> xi; ///Parâmetro Xi da Eq. de Muskat
1557     std::vector<double> r; ///Parâmetro R da Eq. de Muskat
1558     std::vector<double> muskeq; ///Vetor que contém os valores de
        dSo/dp
1559     std::vector<double> so; ///Vetor de saída da saturação de óleo
1560     std::vector<double> np; ///Vetor de saída para a produção
        acumulada de óleo
1561     std::vector<double> gp; ///Vetor de saída para a produção
        acumulada de gás
1562     std::vector<double> psim; ///Vetor de saída para as pressões de
        simulação
1563     CDataProd* dprod; /// Ponteiro de linkagem para a classe
        CDataProd
1564     int ic=0; ///variável de controle auxiliar
1565     public:
1566     CMuskat(CDataProd& obj2){
1567         dprod=&obj2;
1568         r.push_back(dprod->RG0INST());
1569         np.push_back(dprod->NPI());
1570         ///Eta(); Alfa(); Lambda(); Xi();
1571     }; ///Construtor Default - Recebe objeto CDataProd por
        referência
1572     CMuskat(std::string abcde,CDataProd& obj2):CSimuladorCGGS(abcde)
        {
1573         dprod=&obj2;
1574         ///std::cout<<"Qual o razão M da capa de gás?" <<'\\n';
1575         ///std::cin>>m;
1576         ///std::cout<<"Qual a razão de ciclagem C"<<'\\n';
1577         ///std::cin>>cicl;
1578         r.push_back(dprod->RG0INST());
1579         np.push_back(dprod->NPI());
1580         Eta(); Alfa(); Lambda(); Xi();
1581     }; ///Construtor Padrão da Classe para entrada em arquivo de
        disco
1582
1583     ///~CMuskat();
1584     void Eta(); ///Método para cálculo do Parâmetro Eta
1585     void Alfa(); ///Método para cálculo do Parâmetro Alfa
1586     void Lambda(); ///Método para cálculo do Parâmetro Lambda
1587     void Psi(); ///Método para cálculo do Parâmetro Psi
1588     void Xi(); ///Método para cálculo do Parâmetro Xi
1589     void R(); ///Método para cálculo do Parâmetro R
1590     void Output_NP(double _np){np.push_back(_np);} ///Método para

```

```

        preencher os vetores de saída
1591 void Output_GP(double _gp){gp.push_back(_gp);} ///Método para
        preencher os vetores de saída
1592 void Output_Psim(std::vector<double> _xj){psim=_xj;} ///Método
        para preencher os vetores de saída
1593 double muskequation(double p, double satoleo); ///Método que
        contém a equação de Muskat
1594 void Grafico(); ///Método Gráfico Gnuplot da Classe
1595 void Entrada(){
1596     CSimuladorCGGS::Entrada();
1597     //std::cout<<"Qual o razão M da capa de gás? .....
        ";
1598     //std::cin>>m;
1599     //std::cout<<"\n";
1600     //std::cout<<"Qual a razão de ciclagem C? ..... ";
1601     //std::cin>>cicl;
1602     std::cout<<"\n";
1603     Eta(); Alfa(); Lambda(); Xi();
1604 }; ///Método de Entrada
1605 void Saida(std::ostream& os); ///Método de Saída para a Tela
1606 void SaidaDisco(std::ofstream& fout); ///Método de Saída para o
        Disco
1607 double curvakgko(double soleo); ///Método que contém a curva de
        permeabilidade
1608 void SaidaResumida(); ///Método para a saída condensada dos
        resultados de interesse da simulação (exclui variáveis de
        cálculo)
1609 void GP0(double _p){
1610     int i=pos_pressao(_p);
1611     double aux=dprod->N()*(((bo[i]*invbg[i])-rs[i])*(1-(
        dprod->NPI()/dprod->N()))-((dprod->BOI()*invbg[i])-
        dprod->RSI())));
1612     gp.push_back(aux);
1613 } ///Método para calcular a produção acumulada para a pressão
        inicial de análise
1614 friend class CRK4;
1615
1616 };
1617 #endif

```

Apresenta-se na listagem 6.23 o arquivo de implementação da classe CMuskat.

Listing 6.23: Arquivo de implementação da classe CMuskat.

```

1618 #include "CMuskat.h"
1619 #include <math.h>
1620 //-----CMUSKATCPP
        -----
1621 void CMuskat::Eta(){
1622     double etaux;

```

```

1623     for(int i=0;i<var_ctrl;i++)
1624     {
1625         etaux=mio_mig[i]*dbodp[i]*(1/bo[i]);
1626         eta.push_back(etaux);
1627     }
1628 }
1629 void CMuskat::Alfa(){
1630     double alfaaux;
1631     for(int i=0;i<var_ctrl;i++)
1632     {
1633         alfaaux=mio_mig[i]*bo[i]*invbg[i];
1634         alfa.push_back(alfaaux);
1635     }
1636 }
1637 void CMuskat::Lambda(){
1638     double lambaux;
1639     for(int i=0;i<var_ctrl;i++)
1640     {
1641         lambaux=drsdp[i]*((1/invbg[i])/bo[i]);
1642         lambda.push_back(lambaux);
1643     }
1644 }
1645 void CMuskat::Psi(){
1646     double psiaux;
1647     for(int i=0;i<var_ctrl;i++)
1648     {
1649         psiaux=kg_ko[i];
1650         psi.push_back(psiaux);
1651     }
1652 }
1653 void CMuskat::Xi(){
1654     double xiaux;
1655     for(int i=0;i<var_ctrl;i++)
1656     {
1657         xiaux=(1/invbg[i])*dinvbgdp[i];
1658         xi.push_back(xiaux);
1659     }
1660 }
1661 void CMuskat::R() {
1662     double raux;
1663     for(int i=0;i<var_ctrl;i++)
1664     {
1665         raux=(kg_ko[i]*mio_mig[i]*(bo[i]*invbg[i])+rs[i]);
1666         r.push_back(raux);
1667     }
1668 }
1669 double CMuskat::muskequation(double p, double satoleo){ //revisar
1670     int i=pos_pressao(p); ic++;

```

```

1671     double aux;
1672     aux=((satoleo*lambda[i]) + ((1-satoleo-dprod->SWI())*xi[i]) + (
        satoleo*eta[i])*(curvavgko(satoleo)-(cicl*(curvavgko(satoleo)*
        mio_mig[i]*bo[i]*invbg[i]+rs[i])/alfa[i]))+ (m*(1-dprod->SWI()
        )*xi[i]))/
1673     (1+(mio_mig[i]*(curvavgko(satoleo)-(cicl*(curvavgko(satoleo)*
        mio_mig[i]*bo[i]*invbg[i]+rs[i])/alfa[i]))));
1674     muskeq.push_back(aux);
1675     if(ic%2!=1){r.push_back(curvavgko(satoleo)*mio_mig[i]*bo[i]*invbg
        [i]+rs[i]);so.push_back(satoleo);}
1676     return aux;
1677 }
1678 void CMuskat::Saida(std::ostream& os){
1679     CSimuladorCGGS::Resultados(std::cout);
1680     os<<'\\n';
1681     os<<"
        -----
        "<<'\\n';
1682     os<<"A seguir, os parâmetros de cálculo da Eq. de Muskat
        : "<<'\\n';
1683     os<<"
        -----
        "<<'\\n';
1684     os<< "Eta" << std::setw(12) << "Alfa" << std::setw(12)
        << "Lambda" /*<< std::setw(12) << "Psi" */<<std::setw
        (12) << "Xi" <<std::setw(12) << "R" <<std::endl;
1685     for(int j=0; j<var_ctrl; j++)
1686     {
1687     os<<std::showpoint<<std::setprecision(7)<< eta[j] <<std::setw
        (12)<<std::setprecision(7)<< alfa[j] <<std::setw(12)<<std::
        setprecision(7);
1688     os<<lambda[j] /*<<std::setw(12)<<std::setprecision(7)<<psi[j]*/
        <<std::setw(12)<<std::setprecision(7)<<xi[j] <<std::setw(12)
        <<std::setprecision(7);
1689     os<<r[j]<<'\\n';
1690     }
1691     os<<"
        -----
        "<<'\\n';
1692     os<<"\\n";
1693     }
1694 void CMuskat::SaidaDisco(std::ofstream& fout){
1695     //ofstream fout(s.c_str());
1696     CSimuladorCGGS::SaidaDisco(fout);
1697     fout<<'\\n';
1698     fout<<"
        -----
        "<<'\\n';

```

```

1699      fout<<"A seguir, os parâmetros de cálculo da Eq. de Muskat:"<<'\\
        n';
1700      fout<<"
        -----
        "<<'\\n';
1701      fout << "Eta" << std::setw(12) << "Alfa" << std::setw(12) << "
        Lambda" /*<< std::setw(12) << "Psi" */<<std::setw(12) << "Xi"
        <<std::setw(12) << "R" <<std::endl;
1702      for(int j=0; j<var_ctrl; j++)
1703      {
1704          fout<<std::showpoint<<std::setprecision(7)<< eta[j] <<std::setw(12)
          <<std::setprecision(7)<< alfa[j] <<std::setw(12)<<std:::
          setprecision(7);
1705          fout<<lambda[j] /*<<std::setw(12)<<std::setprecision(7)<<psi[j]*/
          <<std::setw(12)<<std::setprecision(7)<<xi[j] <<std::setw(12)<<
          std::setprecision(7);
1706          fout<<r[j]<<'\\n';
1707      }
1708      fout<<"
        -----
        "<<'\\n';
1709      fout<<"\\n";
1710 }
1711 void CMuskat::Grafico(){
1712     std::cout<<"Método Gráfico CMuskat"<<'\\n';
1713     CSimuladorCGGS::Grafico_Sim();}
1714 double CMuskat::curvakgko(double soleo){ //log(kg/ko)=-6,1484sl+3,5070
        para 0,2<sl<0.95
1715     double sl=soleo+dprod->SWI();
1716     //if(sl>0.2 88 sl<0.95){
1717     double aux=(-6.1484*(sl))+3.5070);
1718     double aux2=pow(10,aux);
1719     kg_ko.push_back(aux2);
1720     return aux2;
1721 }
1722 void CMuskat::SaidaResumida(){
1723     using namespace std;
1724     #if defined(WIN32) || defined(_WIN32) || defined(__WIN32__) ||
        defined(__TOS_WIN__)
1725     system("cls");
1726     #elif defined(unix) || defined(__unix) || defined(__unix__) ||
        defined(__APPLE__)
1727     system("clear");
1728     #endif
1729     cout<<"A tabela a seguir contém os resultados resumidos de maior
        interesse para o usuário do programa"<<'\\n';
1730     cout<<"Pressão"<<"_"<<"Saturação de Óleo"<<"_"<<"Produção
        Acumulada de Óleo"<<"_"<<"Produção Acumulada de Gás"<<"_"<<

```



```

        "Razão_Gás-Óleo_Instantânea"<<'\\n';
1731     for (int i=0;i<psim.size();i++){
1732         cout<<right<<psim[i]<<"_<<right<<setw(17)<<setprecision(8)<<so
            [i+1]<<"_";
1733         cout<<right<<setw(26)<<setprecision(8)<<np[i]<<"_<<right<<setw
            (25)<<setprecision(8)<<gp[i]<<"_<<right<<setw(26)<<
            setprecision(8)<<r[i]<<endl;
1734     }
1735 }

```

Apresenta-se na listagem 6.24 o arquivo com código da classe COleo.

Listing 6.24: Arquivo de cabeçalho da classe COleo.

```

1736 /** Previsao do Compartamento de Reservatorios
1737 de Gas e de Oleo com Influxo de Agua
1738 @author Carlos Andre Martins de Assis e Gabriel Clemente Franklin
1739 @file COleo.h
1740 */
1741 /** @brief Classe que represeta um Oleo
1742 @class COleo
1743 */
1744
1745 #ifndef COleo_h
1746 #define COleo_h
1747
1748 #include "CFluido.h"
1749 #include <iostream>
1750
1751 class COleo: public CFluido
1752 {
1753     protected:
1754         double ceo; ///compressibilidade efetiva do oleo
1755         double co;  ///compressibilidade do oleo
1756     public:
1757         ///@brief Construtor default e sobrecarregado
1758         COleo(double _d=0.0,double _v=0.0,double _B=0.0,double _ceo
            =0.0,double _co=0.0)
1759             :CFluido(_d,_v,_B),ceo(_ceo),co(_co){};
1760         ///@brief Construtor sobrecarregado de copia
1761         COleo(const COleo &o):CFluido(o),ceo(o.ceo),co(o.co){};
1762         ///@brief Calcula Ceo
1763         double Ceo(double,double,double);
1764         ///@brief Membros publicos que retornam os parametros
1765         double Ceo(){return ceo;};
1766         double Co(){return co;};
1767         ///@brief Metodos de Entrada e Saida padrao, diferenciados
            para arquivo
1768         virtual void Entrada( std :: ostream & os , std :: istream
            & is);

```

```

1769         virtual void Entrada(std :: ifstream & is);
1770         virtual void Saida( std :: ostream & os );
1771         ///@brief Destrutor
1772         virtual ~COleo(){};
1773 };
1774
1775 #endif

```

Apresenta-se na listagem 6.25 o arquivo de implementação da classe COleo.

Listing 6.25: Arquivo de implementação da classe COleo.

```

1777 /* Previsão do Compartamento de Reservatórios
1778 de Gás e de Óleo com Influxo de Água
1779
1780 @Autores: Carlos André Martins de Assis e Gabriel Clemente Franklin
1781 @file COleo.cpp
1782 */
1783
1784 #include "COleo.h"
1785 #include <iostream>
1786
1787 using namespace std;
1788
1789 double COleo::Ceo(double sw,double cw,double cf)
1790 {
1791     //Calcula ceo a partir da Saturacao de Agua(sw), Comp. da
1792     formacao(cf) e Comp. da Agua(cw)
1793     ceo=co+(cw*sw+cf)/(1-sw);
1794     return ceo;
1795 }
1796
1797 void COleo::Entrada(ostream & os ,istream & is)
1798 {
1799     os<<"\t-----\n\tCOleo\n\t-----"<<endl;
1800     CFluido::Entrada(os,is);
1801     os<<"Entre com a Compressibilidade do Oleo (cm2/kgf):";
1802     is>>co; is.get();
1803     os<<"Entre com a Compressibilidade Efetiva do Oleo (cm2/kgf):";
1804     is>>ceo; is.get();
1805 }
1806
1807 void COleo::Entrada(ifstream & is)
1808 {
1809     CFluido::Entrada(is);
1810     is.ignore(100,':');
1811     is>>co;
1812     is.ignore(100,':');
1813     is>>ceo;

```

```

1814 }
1815
1816 void COleo::Saida(ostream & os)
1817 {
1818     os<<"\t-----\n\tPropriedades do Oleo\n\t-----"<<endl;
1819     CFluido::Saida(os);
1820     os<<"Compressibilidade do Oleo (cm2/kgf):"<<co<<endl;
1821     os<<"Compressibilidade Efetiva do Oleo (cm2/kgf):"<<ceo<<endl;
1822 }

```

Apresenta-se na listagem 6.26 o arquivo com código da classe CRadial.

Listing 6.26: Arquivo de cabeçalho da classe CRadial.

```

1823 /** Previsao do Compartamento de Reservatorios
1824 de Gas e de Oleo com Influxo de Agua
1825 @author Carlos Andre Martins de Assis e Gabriel Clemente Franklin
1826 @file CRadial.h
1827 */
1828 /** @brief Classe que representa uma geometria Radial
1829 @class CRadial
1830 */
1831
1832 #ifndef CRadial_h
1833 #define CRadial_h
1834
1835 #include "CGeometria.h"
1836
1837 class CRadial: public CGeometria
1838 {
1839     protected:
1840         double r; ///< Raio
1841     public:
1842         ///<@brief Construtor default e sobrecarregado
1843         CRadial(double _r=0.0):r(_r){};
1844         ///<@brief Construtor sobrecarregado de copia
1845         CRadial(const CRadial &obj):r(obj.r){};
1846         ///<@brief Metodos de Entrada e Saida padrao, diferenciados
1847         para arquivo
1848         virtual void Entrada( std :: ostream & os , std :: istream
1849             & is);
1850         virtual void Entrada(std :: ifstream & is);
1851         virtual void Saida( std :: ostream & os );
1852         virtual double R(){return r;}; ///

```

Apresenta-se na listagem 6.27 o arquivo de implementação da classe `CRadial`.

Listing 6.27: Arquivo de implementação da classe `CRadial`.

```

1856 /* Previsão do Compartamento de Reservatórios
1857 de Gás e de Óleo com Influxo de Água
1858
1859 @Autores: Carlos André Martins de Assis e Gabriel Clemente Franklin
1860 @file CRadial.cpp
1861 */
1862
1863 #include "CRadial.h"
1864 #include <iostream>
1865 #include <fstream>
1866
1867 using namespace std;
1868
1869 void CRadial::Entrada(ostream & os ,istream & is)
1870 {
1871     os<<"Entre com o Raio (m): ";
1872     is>>r; is.get();
1873 }
1874
1875 void CRadial::Saida(ostream & os)
1876 {
1877     os<<"Raio (m): " <<r<<endl;
1878 }
1879
1880 void CRadial::Entrada(istream & is)
1881 {
1882     is.ignore(100,':');
1883     is>>r;
1884 }

```

Apresenta-se na listagem 6.28 o arquivo com código da classe `CReservatorio`.

Listing 6.28: Arquivo de cabeçalho da classe `CReservatorio`.

```

1884 /** Previsao do Compartamento de Reservatorios
1885 de Gas e de Oleo com Influxo de Agua
1886 @author Carlos Andre Martins de Assis e Gabriel Clemente Franklin
1887 @file CReservatorio.h
1888 */
1889
1890 /** @brief Classe com as propriedade do Reservatorio
1891 @class CReservatorio
1892 */
1893
1894
1895 #ifndef CReservatorio_h
1896 #define CReservatorio_h
1897

```

```

1898 #include "CRocha.h"
1899
1900
1901 class CReservatorio: public CRocha
1902 {
1903     protected:
1904         double sw; ///< saturacao de agua
1905         double q; ///< vazao
1906
1907     public:
1908         ///

```

Apresenta-se na listagem 6.29 o arquivo de implementação da classe CReservatorio.

Listing 6.29: Arquivo de implementação da classe CReservatorio.

```

1933 /* Previsão do Compartamento de Reservatórios

```

```

1934   de Gás e de Óleo com Influxo de Água
1935
1936   @Autores: Carlos André Martins de Assis e Gabriel Clemente Franklin
1937   @file CReservatorio.cpp
1938 */
1939
1940 #include "CReservatorio.h"
1941
1942 using namespace std;
1943
1944 void CReservatorio::Entrada(ostream & os ,istream & is)
1945 {
1946     os<<"\t-----\n\tCReservatorio\n\t-----"<<endl;
1947     CRocha::Entrada(os,is);
1948     os<<"Entre com a Saturacao da Agua (fracao): ";
1949     is>>sw;
1950     os<<"Entre com o Vazao (m3 std/d): ";
1951     is>>q;
1952 }
1953
1954 void CReservatorio::Entrada(istream & is)
1955 {
1956     CRocha::Entrada(is);
1957     is.ignore(100,':');
1958     is>>sw;
1959     is.ignore(100,':');
1960     is>>q;
1961 }
1962
1963 void CReservatorio::Saida(ostream & os)
1964 {
1965     os<<"\t-----\n\tPropriedades do Reservatorio\n\t-----"
        <<endl;
1966     CRocha::Saida(os);
1967     os<<"Saturacao da Agua (fracao): "<<sw<<endl;
1968     os<<"Vazao (m3 std/d): "<<q<<endl;
1969 }

```

Apresenta-se na listagem 6.30 o arquivo com código da classe CRK4.

Listing 6.30: Arquivo de cabeçalho da classe CRK4.

```

1970 #ifndef CRK4_h
1971 #define CRK4_h
1972
1973 #include <iostream>
1974 #include <cstdlib>
1975 #include <fstream>
1976 #include <functional>
1977 #include <algorithm>

```

```

1978 #include <vector>
1979 #include <string>
1980 #include <iterator>
1981 #include "CGnuplot.h"
1982 #include <locale.h>
1983 #include <utility>
1984 #include <cmath>
1985 #include <iomanip>
1986 #include "CMuskat.h"
1987
1988 class CRK4
1989 {
1990     // yj+1 = yj + (k1 + 2k2 + 2k3 + k4) / 6
1991     // k1 = dx f(xj, yj)
1992     // k2 = dx f(xj + k1/2, yj + dx/2)
1993     // k3 = dx f(xj + k2/2, yj + h/2)
1994     // k4 = dx f(xj + k3, yj + dx)
1995     public:
1996         std::vector<double> k1, k2, k3, k4, xj, yj;
1997         double dx;
1998         CMuskat* cm;
1999         int i;
2000         double ite;
2001     public:
2002         CRK4(CMuskat& obj): i(0) {
2003             cm = &obj;
2004             double aux; double aux2;
2005             aux = obj.dprod->PI();
2006             xj.push_back(aux); cm->GP0(aux);
2007             std::cout << std::endl;
2008             std::cout << "Entre com o passo (Se a pressão
                estiver caindo, entre o passo com sinal
                negativo):" << '\n';
2009             std::cin >> dx;
2010             std::cin.get();
2011             std::cout << "Até que pressão deseja prever?" <<
                '\n';
2012             std::cin >> aux2;
2013             std::cin.get();
2014             if(dx < 0) {ite = (aux - aux2) / (-dx);}
2015             else {ite = (aux - aux2) / (dx);}
2016             for (int p = 0; p < ite; p++)
2017             {
2018                 double aux3 = xj[p] + dx;
2019                 xj.push_back(aux3);
2020             }
2021             cm->Output_Psim(xj);
2022             yj.resize(xj.size());

```

```

2023         k1.resize(xj.size()); k2.resize(xj.size()); k3.
           resize(xj.size()); k4.resize(xj.size());
2024         yj[0]=soleo inicial(obj.dprod->NPI(),obj.dprod->
           N(),obj.GetBo(aux),obj.dprod->BOI(),obj.
           dprod->SWI());
2025     };
2026     CRK4(CRK4& obj): k1(obj.k1),k2(obj.k2),k3(obj.k3),k4(obj
           .k4), dx(obj.dx),xj(obj.xj),yj(obj.yj){};
2027     ~CRK4(){};
2028
2029     void CalcK(); /// Calcula os coeficientes K
2030     void YJ1(); /// Calcula vetor yj
2031     void Grafico(); /// Grafica os resultados
2032     void SetDx(double _dx){dx=_dx;}; /// Seta o dx
2033     double GetDx(){return dx;}; /// Devolve o dx
2034     void Solver(); /// Resolve a EDO
2035         double soleo inicial(double _np, double _n,
           double _bo, double _boi, double _swi){
2036             double aux= ((1-(_np/_n))*(_bo/_boi)*(1-
           _swi)); cm->so.push_back(aux); return
           aux;
2037             } /// Calcula Saturação Inicial - YJ
           [0]
2038     void Saida(std::ostream& os, std::istream& cin); ///
           operador sobrecarregado de saída
2039     void SaidaDisco(std::ofstream& fout); /// Salva em Disco
2040 };
2041 #endif

```

Apresenta-se na listagem 6.31 o arquivo de implementação da classe `alglibinternal`.

Listing 6.31: Arquivo de implementação da classe `CRK4`.

```

2042 #include "CRK4.h"
2043
2044 void CRK4::Saida(std::ostream& os, std::istream& cin)
2045 {
2046     using namespace std;
2047     ostream_iterator<double> output(os," ");
2048     cm->dprod->Saida(os);
2049     os<<"Qual vetor queres que sejas mostrado? 1-Pressão 2-Saturação
           de Óleo 3-K1 4-K2 5-K3 6-K4 7-Todos" << std::endl;
2050     int op;
2051     cin>>op;
2052     os << "Mostrando os resultados numéricos..." << std::endl;
2053     switch(op)
2054     {
2055     case 1:
2056     {
2057         os<< "XJ: " << '\n';

```



```

2058     copy(xj.begin(),xj.end(),output); os<<std::endl; break;
2059 }
2060 case 2:
2061 {
2062     os<< "YJ:_" << '\n';
2063     copy(yj.begin(),yj.end(),output); os<<std::endl; break;
2064 }
2065 case 3:
2066 {
2067     os<< "K1:_" << '\n';
2068     copy(k1.begin(),k1.end(),output); os<<std::endl; break;
2069 }
2070 case 4:
2071 {
2072     os<< "K2:_" << '\n';
2073     copy(k2.begin(),k2.end(),output); os<<std::endl; break;
2074 }
2075 case 5:
2076 {
2077     os<< "K3:_" << '\n';
2078     copy(k3.begin(),k3.end(),output); os<<std::endl; break;
2079 }
2080 case 6:
2081 {
2082     os<< "K4:_" << '\n';
2083     copy(k4.begin(),k4.end(),output); os<<std::endl; break;
2084 }
2085 case 7:
2086 {
2087     os << "P" << std::setw(16) << "So" << std::setw(16) << "K1" <<
        std::setw(16) << "K2" <<std::setw(16) << "K3" <<std::setw(16)
        << "K4" << std::endl;
2088     os << setprecision(15) << showpoint;
2089     for(int j=0; j<xj.size(); j++)
2090     {
2091         os << xj[j] <<"_"<< yj[j] <<"_"<<k1[j] <<"_"<<k2[j] <<"_"<< k3
            [j]<<"_"<< k4[j] <<std::endl;
2092     }
2093     break;
2094 }
2095 }
2096
2097 }
2098
2099 void CRK4::SaidaDisco(std::ofstream& fout){
2100     fout<<"
        -----
        "<<'\n';

```

```

2101      fout<<"A seguir os dados provenientes da integração da EDO de
          Muskat:"<<'\\n';
2102      fout<<"
          -----
          "<<'\\n';
2103      fout << "P" << std::setw(16) << "So" << std::setw(16) << "K1" << std
          ::setw(16) << "K2" <<std::setw(16) << "K3" <<std::setw(16) << "K4
          " << std::endl;
2104      fout << std::setprecision(15) << std::showpoint;
2105      for(int j=0; j<xj.size(); j++)
2106      {
2107          fout << xj[j] <<" " << yj[j] <<" " <<k1[j] <<" " <<k2[j] <<" " << k3
          [j] <<" " << k4[j] <<std::endl;
2108      }
2109      fout.close();
2110  }
2111  void CRK4::CalcK() {
2112      k1[i]=dx*(cm->muskequation(xj[i-1],yj[i-1]));
2113      k2[i]=dx*(cm->muskequation(xj[i-1]+(dx/2), yj[i-1]+(k1[i]/2)));
2114      k3[i]=dx*(cm->muskequation(xj[i-1]+(dx/2), yj[i-1]+(k2[i]/2)));
2115      k4[i]=dx*(cm->muskequation(xj[i-1]+dx,yj[i-1]+k3[i]));
2116  }
2117  void CRK4::YJ1() {
2118      yj[i]=yj[i-1]+(k1[i]+2*k2[i]+2*k3[i]+k4[i])/6;
2119      int j=cm->pos_pressao(xj[i]);
2120      double aux=cm->dprod->NP(yj[i],cm->bo[j]);
2121      double aux2=cm->dprod->GP(cm->bo[j],cm->invbg[j],cm->rs[j],aux);
2122      cm->Output_NP(aux); cm->Output_GP(aux2);
2123  }
2124  void CRK4::Solver() {
2125      std::cout<<"Resolvendo a Equação Diferencial Ordinária de Muskat:"
          << '\\n';
2126      for (int s=0; s<ite; s++)
2127      {
2128          i++;
2129          CalcK();
2130          YJ1();
2131      }
2132      std::cout<<"Cálculo Terminado!" << '\\n';
2133  }
2134  void CRK4::Grafico() {
2135      std::cout<<"Deseja ver o gráfico da Saturação de Óleo versus
          Pressão do Reservatório? 1-SIM 2-NÃO"<<'\\n';
2136      int op; std::cin>>op; std::cin.get();
2137      if (op==1) {
2138          Gnuplot::set_GNUPlotPath("C:/PROGRA~1/gnuplot/bin");
2139          CGnuplot grafico(xj,yj,"Saturação de Óleo contra Pressão do
          Reservatório","linespoints","Pressão","Saturação de Óleo");
          system("Pause");

```

```

2140         std::cout<<"Deseja salvar o gráfico? 1-SIM 2-NAO"<<'\\n';
2141         std::cin>>op; std::cin.get();
2142         if(op==1){
2143             std::cout<<"Qual o nome desejado para o gráfico?"<<'\\n';
2144             std::string nome;
2145             getline(std::cin, nome);
2146             std::string salvar="set_out\\"+nome+".png\\"+\\n";
2147             grafico << salvar;
2148             grafico <<"set_term_png\\n";
2149             grafico.Replot(); std::cout<<"Pressione ENTER para continuar"<<'\\n';
2150             std::cin.get();
2151             grafico << "exit";}}
2152 }

```

Apresenta-se na listagem 6.32 o arquivo com código da classe CRocha.

Listing 6.32: Arquivo de cabeçalho da classe CRocha.

```

2154 /** Previsao do Compartamento de Reservatorios
2155 de Gas e de Oleo com Influxo de Agua
2156 @author Carlos Andre Martins de Assis e Gabriel Clemente Franklin
2157 @file CRocha.h
2158 */
2159 /** @brief Classe abstrata que representa uma rocha porosa (aquifero ou
2160 reservatorio)
2161 @class CRocha
2162 */
2163 #ifndef CRocha_h
2164 #define CRocha_h
2165
2166 #include <iostream>
2167 #include <fstream>
2168 #include "CLinear.h"
2169 #include "CRadial.h"
2170
2171 class CRocha
2172 {
2173     protected:
2174         double pi; ///pressao inicial
2175         double poro; ///porosidade
2176         double h; ///espessura média
2177     public:
2178         ///@brief Geometria da Rocha, Linear ou Radial
2179         CGeometria *g;
2180         ///@brief Construtor default, define a CGeometria
2181         CRocha(int geom);
2182         ///@brief Construtor default e sobrecarregado
2183         CRocha(double _pi, double _poro, double _h, double _r)

```

```

2184         :pi(_pi),poro(_poro),h(_h) {g=new CRadial(_r)
                };
2185     CRocha(double _pi,double _poro, double _h,double _l,double
        _w)
2186         :pi(_pi),poro(_poro),h(_h) {g=new CLinear(_l,
        _w);};
2187     ///brief Construtor sobrecarregado de copia
2188     CRocha(const CRocha &o):pi(o.pi),poro(o.poro),h(o.h),g(o.g)
        {};
2189     ///brief Metodos de Entrada e Saida padrao, diferenciados
        para arquivo
2190     virtual void Entrada( std :: ostream & os , std :: istream
        & is);
2191     virtual void Entrada(std :: ifstream & is);
2192     virtual void Saida( std :: ostream & os );
2193     ///brief Membros publicos que retornam os parametros
2194     double Pi(){return pi;};
2195     double Poro(){return poro;};
2196     double H(){return h;};
2197     ///brief Destrutor
2198     virtual ~CRocha(){if (g!=NULL) delete g;};
2199 };
2200
2201 #endif

```

Apresenta-se na listagem 6.33 o arquivo de implementação da classe `CRocha`.

Listing 6.33: Arquivo de implementação da classe `CRocha`.

```

2202 /* Previsão do Compartamento de Reservatórios
2203 de Gás e de Óleo com Influxo de Água
2204
2205 @Autores: Carlos André Martins de Assis e Gabriel Clemente Franklin
2206 @file CRocha.cpp
2207 */
2208
2209 #include "CRocha.h"
2210
2211 using namespace std;
2212
2213 CRocha::CRocha(int geom)
2214 {
2215     if(geom==eradial)
2216     {
2217         g=new CRadial;
2218     }
2219     if(geom==elinear)
2220     {
2221         g=new CLinear;
2222     }

```

```

2223 }
2224
2225 void CRocha::Entrada(ostream & os ,istream & is)
2226 {
2227     os<<"Entre com a Pressao Inicial(kgf/cm2):";
2228     is>>pi;
2229     os<<"Entre com o Porosidade(fracao):";
2230     is>>poro;
2231     os<<"Entre com o Espessura Media(m):";
2232     is>>h; is.get();
2233     g->Entrada(os,is);
2234 }
2235
2236 void CRocha::Entrada(istream & is)
2237 {
2238     is.ignore(100,':');
2239     is>>pi;
2240     is.ignore(100,':');
2241     is>>poro;
2242     is.ignore(100,':');
2243     is>>h;
2244     g->Entrada(is);
2245 }
2246
2247 void CRocha::Saida(ostream & os)
2248 {
2249     os<<"Pressao Inicial(kgf/cm2):"<<pi<<endl;
2250     os<<"Porosidade(fracao):"<<poro<<endl;
2251     os<<"Espessura Media(m):"<<h<<endl;
2252     g->Saida(os);
2253 }

```

Apresenta-se na listagem 6.34 o arquivo com código da classe CSimulador.

Listing 6.34: Arquivo de cabeçalho da classe CSimulador.

```

2255 /** Previsao do Compartamento de Reservatorios
2256     de Gas e de Oleo com Influxo de Agua
2257     @author Carlos Andre Martins de Assis e Gabriel Clemente Franklin
2258     @file CSimulador.h
2259 */
2260
2261 /** @brief Classe Mae para os metodos de Hurst e Fetkovich.
2262     @class CSimulador
2263     */
2264
2265 #ifndef CSimulador_h
2266 #define CSimulador_h
2267
2268 #include "CAquifero.h"

```

```

2269 #include "CReservatorio.h"
2270 #include "CGas.h"
2271 #include "COleo.h"
2272 #include "CLaplaceWd.h"
2273 #include "CStehfest.h"
2274 #include <cmath>
2275 #include <vector>
2276 #include <string>
2277 #include <typeinfo>
2278
2279
2280
2281
2282 //class MainWindow;
2283
2284
2285 using namespace std;
2286
2287 ///@enum enumeração para geracao de graficos
2288 enum {PxT = 0, WxT = 1, WxP = 2};
2289
2290
2291 class CSimulador
2292 {
2293 {
2294     // friend class MainWindow; //DEIXAR APENAS O MÉTODO QUE USAR OS
        VETORES COMO FRIEND!!!!
2295
2296     protected:
2297
2298
2299
2300         int n; ///< numero de anos a serem simulados
2301         int op; ///< tipo de influxo: infinito(1), realimentado
            (2), selado(3)
2302         double dt; ///< Passo em dias do vetor td (default é 365
            dias)
2303         int tipo; ///< Tipo de fluido: Gas(1), Oleo(2)
2304         int geom; ///< Geometria das rochas: Radial(1), Linear(2)
2305
2306         CReservatorio rr; ///< rocha reservatorio
2307         CAquifero ra; ///< rocha aquifero
2308         CFluido *fluido; ///< fluido na rocha reservatorio
2309
2310         vector<double>P; ///< vetor de pressões no contato
            aquifero/reservatório
2311         vector<double>Wen; ///< vetor de influxo acumulado
2312         vector<double>t; ///< Tempo

```

```

2313
2314
2315         string titulo; /// titulo recebe hurst ou fetkovich,
                inicia como padrao
2316     public:
2317
2318         ///@brief Construtor Gas uso na Interface grafica
2319
2320         CSimulador(int _n,int _op,double _dt,double _pi,double
                _poro, double _h,double _sw,double _Q,
2321         double _T,double _ro,double _d,double _v,double _B,double
                _Tpc, double _Ppc,double _z,
2322         double a_pi,double a_poro, double a_h,double _k,double
                _ct,double _u,double _re)
2323         :n(_n),op(_op),dt(_dt),t(_n+1,0.0),P(n+1),Wen(n+1)
2324         ,rr(_pi,_poro,_h,_sw,_Q,_ro),ra(a_pi,a_poro,a_h,_k,_ct,_u
                ,_re)
2325         {fluido=new CGas(_d,_v,_B,_Tpc,_Ppc,_z,_T); titulo = "
                Padrao"; }
2326
2327
2328         ///@brief Construtor oleo uso na interface grafica
2329         CSimulador(int _n,int _op,double _dt,double _pi,double
                _poro, double _h,double _sw,double _Q, ///@brief Aqui
                nao usa mais T
2330         double _ro,double _d,double _v,double _B,double _ceo,
                double _co,
2331         double a_pi,double a_poro, double a_h,double _k,
                double _ct,double _u,double _re)
2332         :n(_n),op(_op),dt(_dt),t(_n+1,0.0),P(n+1),Wen(n+1)
2333         ,rr(_pi,_poro,_h,_sw,_Q,_ro),
2334         ra(a_pi,a_poro,a_h,_k,_ct,_u,_re)
2335         {fluido=new COleo(_d,_v,_B,_ceo,_co); titulo = "
                Padrao"; }
2336
2337         ///@brief uso com Interface grafica
2338         CSimulador(int size):P(size),Wen(size),t(size),rr
                (0.0,0.0,0.0,0.0,0.0,0.0),
2339         ra(0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0)
2340         { titulo = "Padrao";}
2341
2342         ///@brief uso com arquivo do prompt
2343         CSimulador(int _n,int _op,double _dt,int _geom):n(_n),op(
                _op),dt(_dt),t(_n+1,0.0),Wen(_n+1),
2344         P(_n+1),rr(_geom),ra(_geom),geom(_geom)
2345         {titulo = "Padrao";}
2346         ///@brief calcula os vetores de tempo
2347         virtual void Tempo();

```

```

2348      ///Retorna o tipo de fluido
2349      double Tipo(){return tipo;};
2350      ///Metodos das classes herdeiras
2351      virtual void SolverGas()=0;
2352      virtual void SolverOleo()=0;
2353      virtual void Constantes()=0;
2354      ///Gera os graficos a partir das opcoes disponiveis
2355      void Grafico(int op);
2356
2357      ///Sobrecarga para receber os dados (arquivo)
2358      friend void operator>>(ifstream & is,CSimulador &obj);
2359      ///Sobrecarga para enviar os dados (arquivo)
2360      friend void operator<<(ofstream & os,CSimulador &obj);
2361      ///Sobrecarga para enviar os dados (tela)
2362      friend void operator<<(ostream & os,CSimulador &obj);
2363      ///Seta o valor de op
2364      void SetInfluxo(int _op){ op = _op;};
2365      ///Mostra os resultados
2366      void Resultados(ostream & os);
2367
2368      ///Destructor
2369      ~CSimulador(){delete fluido;};
2370
2371 };
2372
2373 #endif

```

Apresenta-se na listagem 6.35 o arquivo de implementação da classe CSimulador.

Listing 6.35: Arquivo de implementação da classe CSimulador.

```

2374 /* Previsão do Compartamento de Reservatórios
2375    de Gás e de Óleo com Influxo de Água
2376
2377    @Autores: Carlos André Martins de Assis e Gabriel Clemente Franklin
2378    @file CSimulador.cpp
2379 */
2380
2381
2382 #include "CSimulador.h"
2383 #include "CGnuplot.h"
2384 #include <iostream>
2385 #include <fstream>
2386
2387
2388 //Biblioteca do Qt esta aqui para usar o comando system no Qt
2389 // #include <QtGui>
2390
2391 void CSimulador::Tempo()
2392 {

```



```

2393
2394         for (int i=1;i<=n;i++)
2395     {
2396
2397
2398         t[i]=i*dt; //0 vetor t é o tempo em dias
2399     }
2400 }
2401
2402
2403
2404
2405 void CSimulador::Resultados(ostream &os)
2406 {
2407     os<<"\t-----\n\tMetodo_de_"<<titulo<<"\n\t-----"<<endl;
2408
2409     os<<"Pressao_(kgf/cm2)"<<"\tInfluxo_Acumulado_(m3)"<<"\tTempo_(dias
2410         )"<<endl;
2411     for (int i=0;i<=n;i++)
2412     {
2413
2414         // os<<"p["<<i<<"]="<<P[i]<<";\t\tWen["<<i<<"]="
2415         // <<"\t"<<Wen[i]<<";\t\t["<<i<<"]="<<"\t"<<t[i]<<endl;
2416
2417         os<<P[i]<<"\t\t\t"<<Wen[i]<<"\t\t\t"<<t[i]<<endl;
2418     }
2419
2420 }
2421
2422
2423 void operator>>(ifstream & is,CSimulador &obj)
2424 {
2425     is>>obj.tipo;
2426     if(obj.tipo==egas)
2427     {
2428         obj.fluido=new CGas;
2429     }
2430     if(obj.tipo==eoleo)
2431     {
2432         obj.fluido=new COleo;
2433     }
2434
2435
2436     obj.rr.Entrada(is);
2437
2438     obj.fluido->Entrada(is);
2439

```

```
2440     obj.ra.Entrada(is);
2441
2442
2443 }
2444
2445
2446
2447 void operator<<(ofstream & os,CSimulador &obj)
2448 {
2449     obj.rr.Saida(os);
2450     obj.fluido->Saida(os);
2451     obj.ra.Saida(os);
2452 }
2453
2454 void operator<<(ostream & os,CSimulador &obj)
2455 {
2456     obj.rr.Saida(os);
2457     obj.fluido->Saida(os);
2458     obj.ra.Saida(os);
2459 }
2460
2461
2462
2463
2464 void CSimulador::Grafico(int op)
2465 {
2466     Gnuplot :: set_GNUPlotPath ("C://gnuplot//binary");
2467     Gnuplot :: Terminal ("win");
2468     CGnuplot obj("lines");
2469     obj.Grid();
2470
2471
2472     if(op == PxT){
2473         obj.set_ylabel ("Pressão no Contato");
2474         obj.set_xlabel ("Tempo (Dias)");
2475         obj.PlotVector (t,P,titulo);
2476         system ("pause");
2477     }
2478
2479     else if(op == WxT){
2480         obj.set_ylabel ("Influxo Acumulado");
2481         obj.set_xlabel ("Tempo");
2482         obj.PlotVector (t,Wen,titulo);
2483         system( "pause" );
2484     }
2485
2486     else if(op == WxP){
2487         obj.set_xlabel("Influxo Acumulado");
```

```

2488 obj.set_ylabel("Pressão no Contato");
2489 obj.PlotVector(Wen,P,titulo);
2490 system("pause");
2491 }
2492 }

```

Apresenta-se na listagem 6.36 o arquivo com código da classe CSimuladorCGGS.

Listing 6.36: Arquivo de cabeçalho da classe CSimuladorCGGS.

```

2493 #ifndef CSimuladorCGGS_h
2494 #define CSimuladorCGGS_h
2495
2496 #include<iostream>
2497 #include<cstdlib>
2498 #include<fstream>
2499 #include<functional>
2500 #include<algorithm>
2501 #include<vector>
2502 #include<string>
2503 #include<iterator>
2504 #include "CGnuplot.h"
2505 #include<locale.h>
2506 #include<stdlib.h>
2507 #include<utility>
2508 #include<cmath>
2509 #include<iomanip>
2510 #include "CDataProd.h"
2511
2512 class CSimuladorCGGS{
2513     protected:
2514     CDataProd* dprod;
2515     double m; ///Razão entre a capa de gás e a seção de óleo do
                reservatório
2516     double cicl; ///Razão de Ciclagem
2517     std::vector<double> p; ///Pressão
2518     std::vector<double> bo; ///Fator Volume-Formação do Óleo
2519     std::vector<double> dbodp; ///Derivada de Bo com a pressão
2520     std::vector<double> bg; ///Fator Volume-Formação do Gás
2521     std::vector<double> invbg; ///Inverso do Bg
2522     std::vector<double> dinvbgdp; ///Derivada do Inverso do Bg com a
                pressão
2523     std::vector<double> rs; ///Razão de Solubilidade
2524     std::vector<double> drsdp; ///Derivada da Razão de Solubilidade
                com a pressão
2525     std::vector<double> mio; ///Viscosidade do Óleo
2526     std::vector<double> mig; ///Viscosidade do Gás
2527     std::vector<double> kg; ///Permeabilidade do Gás
2528     std::vector<double> ko; ///Permeabilidade do Óleo

```

```

2529     std::vector<double> kg_ko; ///Armazena a fração entre as
        permeabilidades
2530     std::vector<double> mio_mig; ///Armazena a fração entre as
        viscosidades
2531     std::vector<double>::iterator it;
2532     int var_ctrl; ///variável de controle
2533
2534     public:
2535     CSimuladorCGGS(){}; ///Construtor Default
2536     CSimuladorCGGS(std::string nome){
2537         std::ifstream fin(nome.c_str());
2538         if(!fin.good()){
2539             std::cout<<"Arquivo não encontrado"<<'\n';
2540             exit(0);
2541         }
2542         double auxvar=0;
2543         std::cout<<"Lendo arquivo"<<'\n';
2544         fin.ignore(5000, '\n');
2545         fin.ignore(5000, '\n');
2546         fin.ignore(61, '\n');
2547         fin>>m; fin.ignore(55, '\n');
2548         fin.ignore(55, '\n');
2549         fin>>cicl; fin.ignore(55, '\n');
2550
2551         for(int i=0; i<18; i++)
2552             {fin.ignore(5000, '\n');}
2553         var_ctrl=0;
2554         do{
2555             fin>>auxvar; std::cout<<auxvar; p.push_back(
                auxvar);
2556             fin>>auxvar; std::cout<<auxvar; bo.push_back(
                auxvar);
2557             fin>>auxvar; std::cout<<auxvar; invbg.push_back(
                auxvar);
2558             fin>>auxvar; std::cout<<auxvar; rs.push_back(
                auxvar);
2559             fin>>auxvar; std::cout<<auxvar; mio_mig.push_back
                (auxvar);
2560             fin>>auxvar; std::cout<<auxvar; if(auxvar!=0){
                dbodp.push_back(auxvar);}
2561             fin>>auxvar; std::cout<<auxvar; if(auxvar!=0){
                dinvbgdp.push_back(auxvar);}
2562             fin>>auxvar; std::cout<<auxvar; if(auxvar!=0){
                drsdp.push_back(auxvar);}
2563             fin.ignore(5000, '\n');
2564             var_ctrl++;
2565         }while(!fin.eof());
2566         if (dbodp.size()==0){DBODP();}

```

```

2567         if (dinvgdp.size()==0){DINVBGDP();}
2568         if (drsdp.size()==0){DRSDP();}
2569         p.pop_back();
2570         bo.pop_back();
2571         invbg.pop_back();
2572         rs.pop_back();
2573         mio_mig.pop_back();
2574         dbodp.pop_back();
2575         dinvgdp.pop_back();
2576         drsdp.pop_back();
2577         fin.close();
2578         std::cout<<"Leitura_terminada!\_CSimuladorCGGS"<<'\n';
2579     }; ///Construtor Padrão da Classe
2580     //~CSimuladorCGGS();
2581     void Grafico(); ///Método Gráfico Gnuplot
2582     void Resultados(std::ostream& os); ///Mostra os Resultados
2583     //double MetMuskat(); //Solver?
2584     void P(); ///Entrada da Pressão
2585     void DBODP(); ///Cálculo da Derivada de Bo com a pressão
2586     void INVBG(); ///Entrada da Pressão
2587     void BO(); ///Entrada do Fator Volume-Formação do Óleo
2588     void DINVBGDP(); ///Cálculo da Derivada do Inverso do Bg com a
        pressão
2589     void RS(); ///Entrada da Razão de Solubilidade
2590     void DRSDP(); ///Cálculo da Derivada da Razão de Solubilidade
        com a pressão
2591     void MIO_MIG(); ///Entrada das Viscosidades
2592     void KG_KO(); ///Entrada das Permeabilidades - caso não haja uma
        curva
2593     void Entrada(){P(); BO(); RS(); DBODP(); INVBG(); DINVBGDP();
        DRSDP();MIO_MIG();std::cout <<"Entrada_Terminada" <<'\n';} ///
        /Método de Entrada
2594     int pos_pressao(double presaux){
2595         it=find(p.begin(),p.end(),presaux);
2596         int pos = distance(p.begin(), it);
2597         return pos;
2598     } ///Método para orientação em qual posição do vetor o cálculo será
        baseado
2599     friend std::ifstream& operator>>(std::ifstream& fin,
        CSimuladorCGGS& obj); ///Operador Sobrecarregado para salvar
        em disco
2600     friend std::ostream& operator<<(std::ostream& out,
        CSimuladorCGGS& obj); ///Operador Sobrecarregado para mostrar
        na tela os conteúdos dos vetores
2601     double GetBo(double pressao); ///Método Get para retornar o
        valor de Bo
2602     void MostraVetor(std::vector<double> v1); ///Método para mostrar
        o vetor na tela

```

```

2603     void SaidaDisco(std::ofstream& fout); ///Método para salvar no
        disco
2604     void SalvaGraf (CGnuplot& graf); ///Método para salvar o gráfico
2605     void Grafico_Sim(); ///Método Gráfico Gnuplot
2606 };
2607 #endif

```

Apresenta-se na listagem 6.37 o arquivo de implementação da classe CSimuladorCGGS.

Listing 6.37: Arquivo de implementação da classe CSimuladorCGGS.

```

2608 #include "CSimuladorCGGS.h"
2609
2610 ///----- CSimuladorCGGS
        -----
2611 void CSimuladorCGGS::SalvaGraf(CGnuplot& graf){
2612     std::cout<<"Deseja salvar o gráfico? 1-SIM 2-NAO"<<'\\n';
2613     int op;
2614     std::cin>>op; std::cin.get();
2615     if(op==1){
2616         std::cout<<"Qual o nome desejado para o grafico?"<<'\\n';
2617         std::string nome;
2618         getline(std::cin, nome);
2619         std::string salvar="set out\\"+nome+".png\\n";
2620         graf << salvar;
2621         graf <<"set term png\\n";
2622         graf.Replot(); std::cout<<"Pressione ENTER para continuar"<<'\\n';
                ;
2623         std::cin.get();}
2624 }
2625 void CSimuladorCGGS::Grafico_Sim(){
2626     int opc,opcx;
2627     Gnuplot::set_GNUPlotPath("C:/PROGRA~1/gnuplot/bin");
2628     std::cout<<"Qual gráfico das propriedades dos fluidos de
        reservatório deseja plotar?" <<'\\n';
2629     std::cout<<"1-Bo contra P" <<'\\n';
2630     std::cout<<"2-Rs contra P" <<'\\n';
2631     std::cout<<"3-Inverso de Bg contra P" <<'\\n';
2632     std::cout<<"4-Curva de Permeabilidade Kg contra Ko" <<'\\n';
2633     std::cout<<"5-Nenhum"<<'\\n';
2634     std::cin>>opc; std::cin.get(); std::cout<<"Pressione ENTER para
        continuar"<<'\\n'; std::cin.get();
2635
2636     if(opc==1){
2637         CGnuplot grafico(p,bo,"Bo contra P","linespoints","Pressão","
            Fator Volume-Formação do óleo"); SalvaGraf(grafico); std::cin
            .get();}
2638     if(opc==2){
2639         CGnuplot grafico(p,rs,"Rs contra P","linespoints","
            Pressão","Razão de solubilidade"); SalvaGraf(grafico);

```

```

        std::cin.get();}

2640     if(opc==3){
2641         CGnuplot grafico(p,bo,"1/Bg_contra_P","linespoints","Pressão","
            Inverso_do_Fator_Volume-Formação_do_Gás"); SalvaGraf(grafico)
            ; std::cin.get();}

2642     if(opc==4){
2643         CGnuplot grafico(ko,kg,"Kg_contra_Ko","linespoints","Ko","Kg");
            SalvaGraf(grafico); std::cin.get();}

2644     if(opc<=4 && opc>0){
2645         std::cout<<"Deseja_ver_outro_gráfico_dentre_os_quatro_aqui_
            apresentados?_1-SIM_2-NAO"<<'\\n';
2646         std::cin>>opc; std::cin.get();
2647         if (opc==1){CSimuladorCGGS::Grafico_Sim();}
2648     }
2649 void CSimuladorCGGS::Resultados(std::ostream& os){
2650     os<<"A_seguir_os_dados_da_simulação:"<<'\\n';
2651     os<<"
        -----
        "<<'\\n';
2652 os << "Pressão" << std::setw(12) << "Bo" << std::setw(12) << "dBo/dP" <<
        std::setw(12) << "1/Bg" <<std::setw(12) << "d(1/Bg)/dP" <<std::setw
        (12) << "Rs" <<std::setw(12) << "dRs/dP" <<std::setw(12) << "uo/ug"
        <<std::endl;
2653 os << "kgf/cm^2" << std::setw(12) << "m^3/m^3_std" << std::setw(12) << "
        1/kgf/cm^2" << std::setw(12) << "m^3std/m" <<std::setw(12) << "1/kgf/
        cm^2" <<std::setw(12) << "m^3std/m^3std" <<std::setw(12) << "1/kgf/cm
        ^2" <<std::setw(12) << "cp/cp" <<std::endl;
2654 for(int j=0; j<p.size(); j++)
2655 {
2656     os <<std::showpoint<<std::setprecision(7)<< p[j] <<std::setw(12)<<std
        ::setprecision(7)<< bo[j] <<std::setw(12)<<std::setprecision(7)<<
        dbodp[j] <<std::setw(12)<<std::setprecision(7)<<invgb[j] <<std::
        setw(12)<<std::setprecision(7)<<dinvgb[j]<<std::setw(12)<<std::
        setprecision(7)<<rs[j]<<std::setw(12)<<std::setprecision(7)<<drsd
        p[j]<<std::setw(12)<<std::setprecision(7)<<mio_mig[j]<<std::setw
        (12)<<std::setprecision(7)<<std::endl;
2657 } //chamar Saída Tabela de CMuskat e CRunge
2658 } //tabela resultados
2659 void CSimuladorCGGS::SaidaDisco(std::ofstream& fout){
2660     fout<<"A_seguir_os_dados_da_simulação:"<<'\\n';
2661     fout<<"
        -----
        "<<'\\n';
2662     fout << "Pressão" << std::setw(12) << "Bo" << std::setw(12) << "
        dBo/dP" << std::setw(12) << "1/Bg" <<std::setw(12) << "d(1/Bg
        )/dP" <<std::setw(12) << "Rs" <<std::setw(12) << "dRs/dP" <<
        std::setw(12) << "uo/ug" <<std::endl;
2663     fout << "kgf/cm^2" << std::setw(12) << "m^3/m^3_std" << std::setw

```

```

(12) << "1/kgf/cm^2" << std::setw(12) << "m^3std/m" <<std::setw
(12) << "1/kgf/cm^2" <<std::setw(12) << "m^3std/m^3std" <<std::
setw(12) << "1/kgf/cm^2" <<std::setw(12) << "cp/cp" <<std::endl;
2664 for(int j=0; j<p.size(); j++)
2665 {
2666     fout<<std::showpoint<<std::setprecision(7)<< p[j] <<std::setw(12)<<
        std::setprecision(7)<< bo[j] <<std::setw(12)<<std::setprecision
        (7)<<dbodp[j] <<std::setw(12)<<std::setprecision(7)<<invbg[j] <<
        std::setw(12)<<std::setprecision(7)<<dinvbgdp[j]<<std::setw(12)
        <<std::setprecision(7)<<rs[j]<<std::setw(12)<<std::setprecision
        (7)<<drsdp[j]<<std::setw(12)<<std::setprecision(7)<<mio_mig[j]<<
        std::setw(12)<<std::setprecision(7)<<std::endl;
2667 }
2668 }
2669 void CSimuladorCGGS::P(){
2670     double pressure;
2671     std::cout <<"Entre com os dados da pressões medidas de
        reservatório! Digite um valor menor que 0 para sair do loop(
        visto que não há pressões negativas no reservatório)"<<'\\n';
2672     do{
2673         std::cout <<"Digite a pressão em kgf/cm^2"<<'\\n';
2674         std::cin>>pressure; std::cin.get();
2675         p.push_back(pressure);
2676     }while(pressure>0);
2677     std::cout <<"Entrada terminada!" <<'\\n';
2678     p.pop_back();
2679     var_ctrl=p.size(); //recebe tamanho do vetor para definir a
        variável de controle de cálculo
2680     MostraVetor(p);
2681 }
2682 void CSimuladorCGGS::B0(){
2683     double bo_aux;
2684     std::cout <<"Entre com os valores de Bo referentes à cada pressão
        preenchida por você anteriormente neste programa." <<'\\n';
2685     for(int i=0;i<var_ctrl;i++){
2686         std::cout << "Digite o valor de Bo para a pressão de" << p[i]
            <<" kgf/cm^2" << '\\n';
2687         std::cin>>bo_aux; std::cin.get();
2688         bo.push_back(bo_aux);
2689     }
2690     //incluir depois condição para invalidar entrada se o tamanho de
        bo passar do tamanho de P
2691     std::cout <<"Entrada terminada B0!" <<'\\n';
2692     MostraVetor(bo);
2693 }
2694 void CSimuladorCGGS::DBODP(){
2695     double varaux; dbodp.resize(var_ctrl);
2696     // if bo.size() < 4 and pressao.size() < 4, do something else.

```



```

2697     for(int i=0;i<var_ctrl-2;i++){dbodp[i+1]=(bo[i]-bo[i+2])/(p[i]-p[i+2])
        ;}
2698     double diff=dbodp[1]-dbodp[2];
2699     varaux=var_ctrl-1;
2700     dbodp[0]=dbodp[1]+diff;
2701     dbodp[varaux]=dbodp[varaux-1]-diff;
2702     std::cout <<"Entrada_terminada_DBODP()" <<'\\n';
2703     MostraVetor(dbodp);
2704 }
2705 void CSimuladorCGGS::INVBG(){
2706     double bg_aux, invbgaux; int i=0;
2707     std::cout <<"Entre_com_os_valores_de_Bg_referentes_à_cada_pressão
        _preenchida_por_você_anteriormente_neste_programa." <<'\\n';
2708     for(int i=0;i<var_ctrl;i++){
2709         std::cout << "Digite_o_valor_de_Bg_para_a_pressão_de" << p[i]
            ]<<"_kgf/cm^2"<< '\\n';
2710         std::cin >>bg_aux; std::cin.get();
2711         bg.push_back(bg_aux);
2712         invbgaux=1/bg_aux;
2713         invbg.push_back(invbgaux);
2714         i++;
2715     }
2716     std::cout <<"Entrada_terminada!_INVBG()" <<'\\n';
2717     MostraVetor(invbg);
2718 }
2719 void CSimuladorCGGS::DINVBGDP(){
2720     double varaux; dinvbgdp.resize(var_ctrl);
2721     // if invbg.size() < 4 and pressao.size() < 4, do something else.
2722     for(int i=0;i<var_ctrl-2;i++){dinvbgdp[i+1]=(invbg[i]-invbg[i+2])/(
        p[i]-p[i+2]);}
2723     double diff=dinvbgdp[1]-dinvbgdp[2];
2724     varaux=var_ctrl-1;
2725     dinvbgdp[0]=dinvbgdp[1]+diff;
2726     dinvbgdp[varaux]=dinvbgdp[varaux-1]-diff;
2727     std::cout <<"Entrada_terminada_DINVBGDP()" <<'\\n';
2728     MostraVetor(dinvbgdp);
2729 }
2730 void CSimuladorCGGS::RS(){
2731     double rs_aux; int i=0;
2732     std::cout <<"Entre_com_os_valores_de_Rs_referentes_à_cada_pressão
        _preenchida_por_você_anteriormente_neste_programa." <<'\\n';
2733     for(int i=0;i<var_ctrl;i++){
2734         std::cout << "Digite_o_valor_de_Rs_para_a_pressão_de" << p[i]
            ]<<"_kgf/cm^2"<< '\\n';
2735         std::cin >>rs_aux; std::cin.get();
2736         rs.push_back(rs_aux);
2737     }
2738     //incluir depois condição para invalidar entrada se o tamanho de

```

```

        rs passar do tamanho de P
2739         std::cout <<"Entrada_terminada!_RS()" <<'\\n';
2740         MostraVetor(rs);
2741     }
2742 void CSimuladorCGGS::DRSDP(){
2743     double varaux; drsdp.resize(var_ctrl);
2744     // if invbg.size() < 4 and pressao.size() < 4, do something else.
2745     for(int i=0;i<var_ctrl-2;i++){drsdp[i+1]=(rs[i]-rs[i+2])/(p[i]-p[i
        +2]);}
2746     double diff=drsdp[1]-drsdp[2];
2747     varaux=var_ctrl-1;
2748     drsdp[0]=drsdp[1]+diff;
2749     drsdp[varaux]=drsdp[varaux-1]-diff;
2750
2751     std::cout <<"Entrada_terminada_DRSDP!" <<'\\n';
2752     MostraVetor(drsdp);
2753 }
2754 void CSimuladorCGGS::MIO_MIG(){
2755     std::cout<<"Insira_os_valores_das_viscosidades_do_óleo_em_cp" <<'
        \\n';
2756     double miaux;
2757     for(int i=0;i<var_ctrl;i++){
2758         std::cout << "Digite_o_valor_de_Mio_para_a_pressão_de"<<
            p[i]<<"_kgf/cm^2"<< '\\n';
2759         std::cin>>miaux; std::cin.get();
2760         mio.push_back(miaux);
2761     }
2762
2763     std::cout<<"Insira_os_valores_das_viscosidades_do_gás_em_cp" <<'\\
        n';
2764     for(int j=0;j<var_ctrl;j++){
2765         std::cout << "Digite_o_valor_de_Mig_para_a_pressão_de"<<
            p[j]<<"_kgf/cm^2"<< '\\n';
2766         std::cin>>miaux; std::cin.get();
2767         mig.push_back(miaux);
2768     }
2769     mio_mig.resize(var_ctrl);
2770     std::cout<<"Vetor_mio:_"; MostraVetor(mio); std::cout<<'\\n';
2771     std::cout<<"Vetor_mig:_"; MostraVetor(mig); std::cout<<'\\n';
2772     std::transform(mio.begin(),mio.end(),mig.begin(),mio_mig.begin(),
        std::divides<double>());
2773     std::cout<<"Vetor_mio_mig:_"; MostraVetor(mio_mig); std::cout<<'\\
        n';
2774 }
2775 void CSimuladorCGGS::KG_KO(){
2776     std::cout<<"Insira_os_valores_das_permeabilidades_do_óleo_em_cp"
        <<'\\n';
2777     double koaux;

```

```

2778     for(int i=0;i<var_ctrl;i++){
2779         std::cout << "Digite o valor de Ko para a pressão de " << p
                [i]<<" kgf/cm^2" << '\n';
2780         std::cin>>koaux; std::cin.get();
2781         ko.push_back(koaux);
2782     }
2783     std::cout<<"Insira os valores das permeabilidades do gás em cp"
        <<'\n';
2784     double kgaux;
2785     for(int j=0;j<var_ctrl;j++){
2786         std::cout << "Digite o valor de Kg para a pressão de " << p
                [j]<<" kgf/cm^2" << '\n';
2787         std::cin>>kgaux; std::cin.get();
2788         kg.push_back(kgaux);
2789     }
2790     kg_ko.resize(var_ctrl);
2791     std::cout<<"Vetor kg: "; MostraVetor(ko); std::cout<<'\n';
2792     std::cout<<"Vetor ko: "; MostraVetor(kg); std::cout<<'\n';
2793     std::transform(kg.begin(),kg.end(),ko.begin(),kg_ko.begin(),std::
        divides<double>());
2794     std::cout<<"Vetor kg_ko: "; MostraVetor(kg_ko); std::cout<<'\n';
2795 }
2796 void CSimuladorCGGS::MostraVetor(std::vector<double> v1){
2797
2798     std::ostream_iterator<double> output(std::cout," ");
2799     copy(v1.begin(),v1.end(),output);
2800     std::cout<<'\n';
2801 }
2802 std::ifstream& operator>>(std::ifstream& fin, CSimuladorCGGS& obj){
2803     double auxvar=0;
2804     fin.ignore(5000,'\n');
2805     fin.ignore(5000,'\n');
2806     fin>>obj.var_ctrl;
2807     fin.ignore(5000,'\n');
2808     fin.ignore(5000,'\n');
2809     for (int i=0;i<obj.var_ctrl; i++){
2810         fin>>auxvar;
2811         obj.p.push_back(auxvar);
2812     }
2813     fin.ignore(5000,'\n');
2814     fin.ignore(5000,'\n');
2815     for (int i=0;i<obj.var_ctrl; i++){
2816         fin>>auxvar;
2817         obj.bo.push_back(auxvar);
2818     }
2819     fin.ignore(5000,'\n');
2820     fin.ignore(5000,'\n');
2821     for (int i=0;i<obj.var_ctrl; i++){

```

```

2822         fin>>auxvar;
2823         obj.invgb.push_back(auxvar);
2824     }
2825     fin.ignore(5000, '\n');
2826     fin.ignore(5000, '\n');
2827     for (int i=0; i<obj.var_ctrl; i++){
2828         fin>>auxvar;
2829         obj.rs.push_back(auxvar);
2830     }
2831     fin.ignore(5000, '\n');
2832     fin.ignore(5000, '\n');
2833     for (int i=0; i<obj.var_ctrl; i++){
2834         fin>>auxvar;
2835         obj.mio_mig.push_back(auxvar);
2836     }
2837     fin.ignore(5000, '\n');
2838     fin.ignore(5000, '\n');
2839     for (int i=0; i<obj.var_ctrl; i++){
2840         fin>>auxvar;
2841         obj.dbodp.push_back(auxvar);
2842     }
2843     fin.ignore(5000, '\n');
2844     fin.ignore(5000, '\n');
2845     for (int i=0; i<obj.var_ctrl; i++){
2846         fin>>auxvar;
2847         obj.dinvgdp.push_back(auxvar);
2848     }
2849     fin.ignore(5000, '\n');
2850     fin.ignore(5000, '\n');
2851     for (int i=0; i<obj.var_ctrl; i++){
2852         fin>>auxvar;
2853         obj.drmdp.push_back(auxvar);
2854     }
2855     fin.close();
2856     std::cout<<"Leitura_terminada!"<<'\\n';
2857     return fin;
2858 }

2859 std::ostream& operator<<(std::ostream& out, CSimuladorCGGS& obj){
2860     std::cout<<"Vamos_mostrar_os_vetores_até_agora:"<<'\\n';
2861     std::cout<<"Vetor_Pressão:"<<'\\n';
2862     obj.MostraVetor(obj.p);
2863     std::cout<<"Vetor_Bo:"<<'\\n';
2864     obj.MostraVetor(obj.bo);
2865     std::cout<<"Vetor_1/Bg:"<<'\\n';
2866     obj.MostraVetor(obj.invgb);
2867     std::cout<<"Vetor_Rs:"<<'\\n';
2868     obj.MostraVetor(obj.rs);
2869     std::cout<<"Vetor_Mio_Mig:"<<'\\n';

```

```

2870     obj.MostraVetor(obj.mio_mig);
2871     std::cout<<"Vetor_dBo/dP:"<<'\\n';
2872     obj.MostraVetor(obj.dbodp);
2873     std::cout<<"Vetor_dInvBg/dP:"<<'\\n';
2874     obj.MostraVetor(obj.dinvbgdp);
2875     std::cout<<"Vetor_dRs/dP:"<<'\\n';
2876     obj.MostraVetor(obj.drscp);
2877     return out;
2878 }
2879 double CSimuladorCGGS::GetBo(double pressao){
2880     int i=pos_pressao(pressao);
2881     return bo[i];
2882 }

```

Apresenta-se na listagem 6.38 o arquivo com código da classe CStehfest.

Listing 6.38: Arquivo de cabeçalho da classe CStehfest.

```

2883 /** Previsao do Compartamento de Reservatorios
2884 de Gas e de Oleo com Influxo de Agua
2885 @author Carlos Andre Martins de Assis e Gabriel Clemente Franklin
2886 @file CStehfest.h
2887
2888 */
2889
2890 /** @brief Algoritmo de Stehfest para inversao numerica do campo de
2891 Laplace, Apendice H.
2892 @Class CStehfest
2893
2894 */
2895 #ifndef CStehfest_h
2896 #define CStehfest_h
2897
2898 #include <vector>
2899 class CStehfest
2900 {
2901     public:
2902
2903         int n; ///< numero de iteracoes
2904         std::vector<double>v; ///< vetor v do algoritmo de Stehfest
2905
2906     public:
2907         ///  
brief Construtor default e sobrecarregado que
2908         inicializa v
2909         CStehfest(int _n=10):n(_n),v(_n)
2910         {SetV();};
2911         ///  
brief Construtor de copia
2912         CStehfest(const CStehfest& obj):v(obj.v),n(obj.n){};
2913         ///  
brief Metodo que calcula o fatorial de N

```

```

2913         double Fatorial(int N);
2914         ///@brief Define o vetor V
2915         void SetV();
2916         ///@Inverte a solucao do campo de Laplace para o campo real
2917         double Inversao(double (*f)(double),double x);
2918         ///@Inverte a solucao do campo de Laplace para o campo real
                (sobrecarga)
2919         double Inversao(double (*f)(double,double,double),double re,
                double ro,double x);
2920         ///@brief Destrutor
2921         ~CStehfest(){};
2922
2923     };
2924
2925
2926
2927 #endif

```

Apresenta-se na listagem 6.39 o arquivo de implementação da classe CStehfest.

Listing 6.39: Arquivo de implementação da classe CStehfest.

```

2928 /* Previsão do Compartamento de Reservatórios
2929    de Gás e de Óleo com Influxo de Água
2930
2931    @Autores: Carlos André Martins de Assis e Gabriel Clemente Franklin
2932    @file CStehfest.cpp
2933 */
2934 #include <iostream>
2935 #include <cmath>
2936 #include <vector>
2937 #include <algorithm>
2938 #include "specialfunctions.h"
2939 #include "CStehfest.h"
2940
2941 using namespace std;
2942 using namespace alglib;
2943
2944
2945 double CStehfest::Fatorial(int N)
2946 {
2947     double x = 1;
2948     if (N > 1)
2949     {
2950         for (int i = 2; i <= N; i++)
2951             x = i * x;
2952     }
2953     return x;
2954
2955 }

```

```

2956
2957 void CStehfest::SetV() //indices começam em zero
2958 {
2959
2960     int N2 = n / 2;
2961     int NV = 2 * N2;
2962     int sign = 1;
2963     if ((N2 % 2) != 0)
2964         sign = -1;
2965     for (int i = 0; i < n; i++)
2966     {
2967         int kmin = (i + 2) / 2;
2968         int kmax = i + 1;
2969         if (kmax > N2)
2970             kmax = N2;
2971         v[i] = 0;
2972         sign = -sign;
2973         for (int k = kmin; k <= kmax; k++)
2974         {
2975             double y=k;
2976             v[i] = v[i] + (pow(y, N2) / Fatorial(k)) * (Fatorial(2 *
                k)
2977                 / Fatorial(2 * k - i - 1)) / Fatorial(N2 - k)
2978                 / Fatorial(k - 1) / Fatorial(i + 1 - k);
2979         }
2980         v[i] = sign * v[i];
2981     }
2982
2983
2984 double CStehfest::Inversao(double (*f)(double),double x)
2985
2986 {
2987
2988     double u=0.;
2989     double l=0.;
2990     double soma=0.;
2991     u=log(2)/x;
2992     l=1.;
2993
2994     for (int i=0;i<n;i++)
2995     {
2996         soma+=v[i]*f(u*l);
2997         l++;
2998     }
2999
3000     return u*soma;
3001 }
3002 double CStehfest::Inversao(double (*f)(double,double,double),double re,

```

```

        double ro, double x)
3003
3004 {
3005
3006     double u=0.;
3007     double l=0.;
3008     double soma=0.;
3009     u=log(2)/x;
3010     l=1.;
3011
3012     for (int i=0;i<n;i++)
3013     {
3014         soma+=v[i]*f(u*l,re,ro);
3015         l++;
3016     }
3017
3018     return u*soma;
3019 }

```

Apresenta-se na listagem 6.40 o programa que usa a classe `CAplicacao`.

Listing 6.40: Arquivo de implementação da função `main()`.

```

3020 #include <iostream>
3021 #include "CInterfaceCGGS.cpp"
3022 #include "CInterface.h"
3023 #include <locale.h>
3024 /* run this program using the console pauser or add your own getch,
    system("pause") or input loop */
3025
3026 using namespace std;
3027 int main() {
3028     setlocale(LC_ALL, "");
3029     int p=1;
3030     system("color_0F");
3031     #if defined(WIN32) || defined(_WIN32) || defined(__WIN32__) ||
        defined(__TOS_WIN__)
3032     system("cls");
3033     #elif defined(unix) || defined(__unix) || defined(__unix__) ||
        defined(__APPLE__)
3034     system("clear");
3035     #endif
3036     cout<<"Simulador de Parâmetros de Comportamento de Reservatório"
        <<endl;
3037     cout<<"LENEP - UENF - Programação Prática"<<'\\n';
3038     cout<<"Professor: André Duarte Bueno, Dr."<<'\\n';
3039     cout<<"
        -----
        "<<endl;

```



```

3040     cout<<"Simulador de Reservatório de Óleo/Gás com Influxo de Água
        (2010)"<<endl;
3041     cout<<"Autores: Gabriel Clemente Franklin e Carlos Andre Martins
        de Assis"<<endl;
3042     cout<<"
        -----
        "<<endl;
3043     cout<<"Simulador de Reservatório de Óleo com capa de gás ou gás
        em solução (2016)"<<endl;
3044     cout<<"Autor: Thiago Couto de Almeida Chaves"<<endl;
3045     cout<<"
        -----
        "<<endl;
3046     cout<<"Para começar, escolha o mecanismo de produção do
        reservatório estudado:"<<endl;
3047     cout<<"1-Reservatório de Óleo/Gás com Influxo de Água"<<endl;
3048     cout<<"2-Reservatório de Óleo com capa de gás ou gás em solução"
        <<endl;
3049     int op; cin>>op; cin.get();
3050
3051     CInterface *intface=NULL;
3052     CInterfaceCGGS *intface_cggs=NULL;
3053
3054     #if defined(WIN32) || defined(_WIN32) || defined(__WIN32__) ||
        defined(__TOS_WIN__)
3055     system("cls");
3056     #elif defined(unix) || defined(__unix) || defined(__unix__) ||
        defined(__APPLE__)
3057     system("clear");
3058     #endif
3059
3060     if (op==1){
3061         while(p!=0)
3062         {
3063             #if defined(WIN32) || defined(_WIN32) || defined(
                __WIN32__) || defined(__TOS_WIN__)
3064             system("color 17");
3065             #elif defined(unix) || defined(__unix) || defined(
                __unix__) || defined(__APPLE__)
3066             system("setterm -background blue -foreground white -
                store");
3067             #endif
3068             intface= new CInterface;
3069             system("cls");
3070             cout<<"\n\tDeseja iniciar uma nova simulacao?(0-nao):"<<endl;
3071             cin>>p; cin.get();
3072             delete intface;
3073             intface=NULL;

```

```

3074         }
3075
3076     }
3077
3078     if(op==2){
3079         while(p!=0)
3080         {
3081             #if defined(WIN32) || defined(_WIN32) || defined(
3082                 __WIN32__) || defined(__TOS_WIN__)
3083             system("color F0");
3084             #elif defined(unix) || defined(__unix) || defined(
3085                 __unix__) || defined(__APPLE__)
3086             system("setterm -background white -foreground black -
3087                 store");
3088             #endif
3089             intface_cggs= new CInterfaceCGGS;
3090             #if defined(WIN32) || defined(_WIN32) || defined
3091                 (__WIN32__) || defined(__TOS_WIN__)
3092             system("cls");
3093             #elif defined(unix) || defined(__unix) ||
3094                 defined(__unix__) || defined(__APPLE__)
3095             system("clear");
3096             #endif
3097             cout<<"\n\tDeseja iniciar uma nova simulacao?(0-nao):"<<endl;
3098             cin>>p; cin.get();
3099             delete intface_cggs;
3100             intface_cggs=NULL;
3101         }
3102     }
3103
3104     cout<<"Deseja iniciar uma nova simulação com outro tipo de
3105         reservatório? 1-SIM2-NAO"<<endl;
3106     int op2; cin>>op2; cin.get();
3107     if(op2==1){main();}
3108     return 0;
3109 }
3110 Bem vindo ao C++!

```

Nota:

Não perca de vista a visão do todo; do projeto de engenharia como um todo. Cada capítulo, cada seção, cada parágrafo deve se encaixar. Este é um diferencial fundamental do engenheiro em relação ao técnico, a capacidade de desenvolver projetos, de ver o todo e suas diferentes partes, de modelar processos/sistemas/produtos de engenharia.

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 1: Tela inicial

Este software se apresenta em modo texto. Veja na Figura 7.1, a tela que servirá para que o usuário decida qual tipo de reservatório utilizar:

7.2 Teste 2: Interfaces do programa

Dependendo da escolha do usuário, uma ou outra interface será utilizada. A seguir na Figura 7.2 temos a interface para o reservatório de óleo/gás com influxo de água. Na figura 7.3, temos a interface para o reservatório de óleo com capa de gás ou gás em solução.

7.3 Teste 3: Calculando o influxo de água e plotando gráfico associado

As imagens a seguir mostram o cálculo do influxo de água do programa, assim como um gráfico das propriedades calculadas.

7.4 Teste 4: Calculando a saturação de óleo, outros parâmetros e plotando gráfico associado

As imagens a seguir mostra a previsão da saturação de óleo do reservatório mediante um declínio na pressão, cálculo de outros parâmetros (como a produção acumulada de óleo) e um gráfico da saturação de óleo versus pressão.

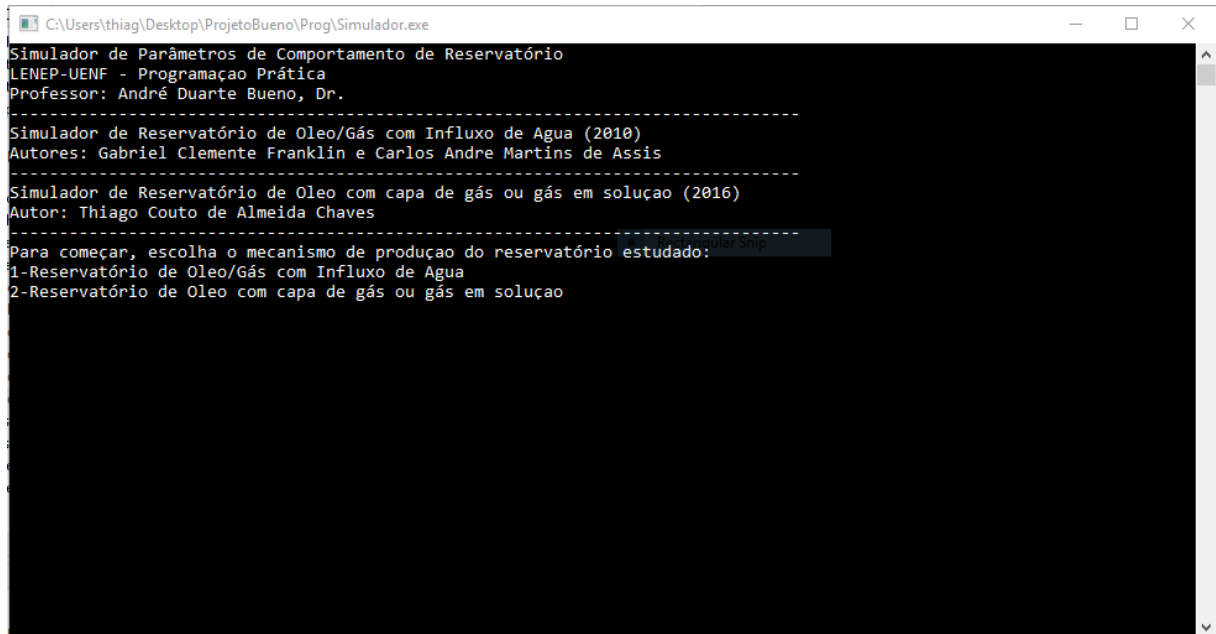


Figura 7.1: Tela Inicial

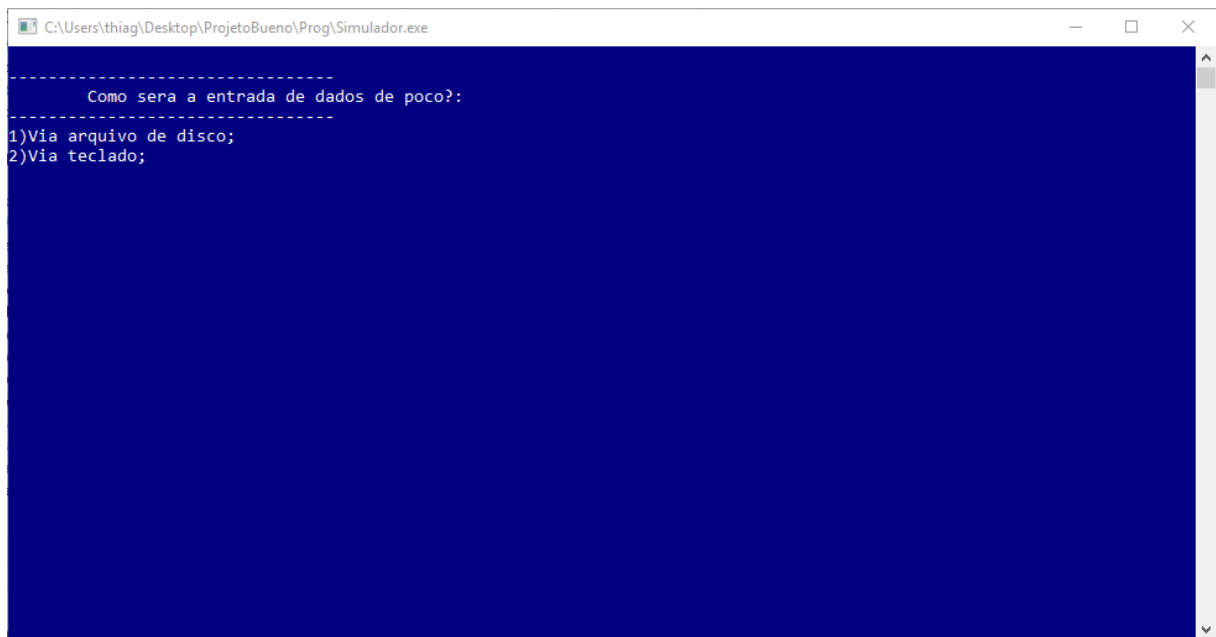
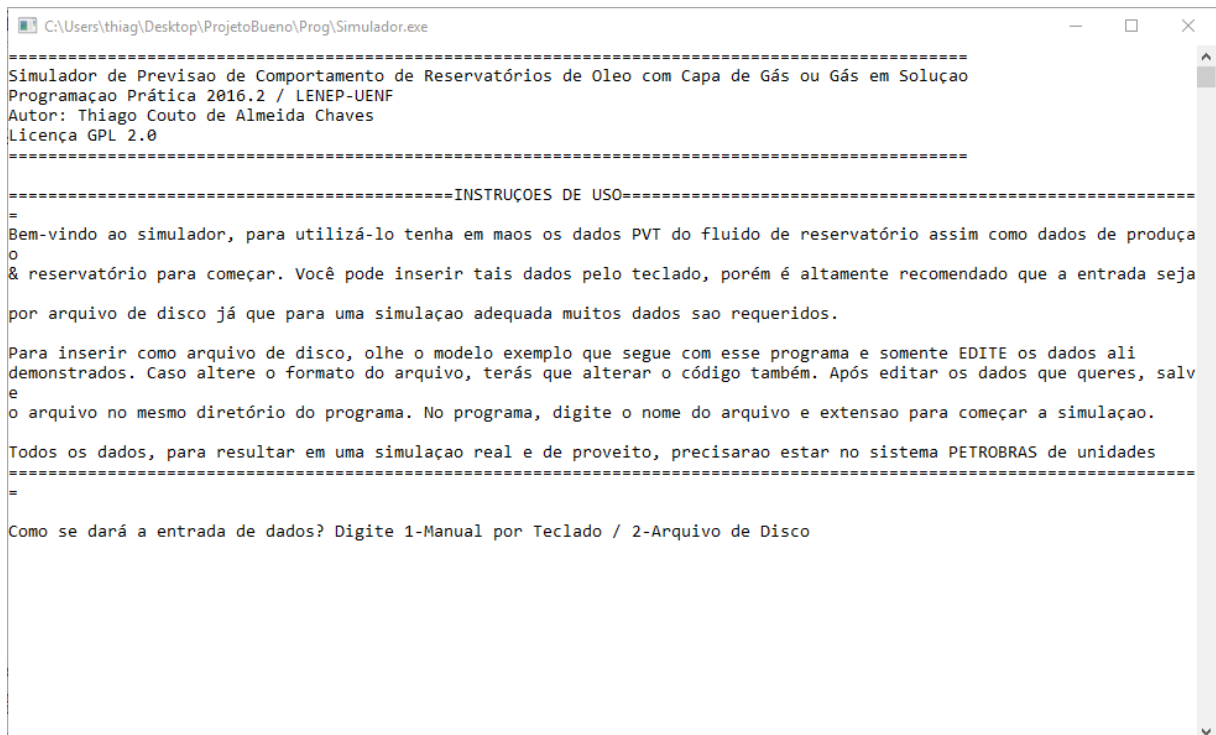


Figura 7.2: Interface-Influxo de Água



```
C:\Users\thiag\Desktop\ProjetoBueno\Prog\Simulador.exe

=====
Simulador de Previsao de Comportamento de Reservatórios de Oleo com Capa de Gás ou Gás em Solução
Programação Prática 2016.2 / LENEP-UENF
Autor: Thiago Couto de Almeida Chaves
Licença GPL 2.0
=====

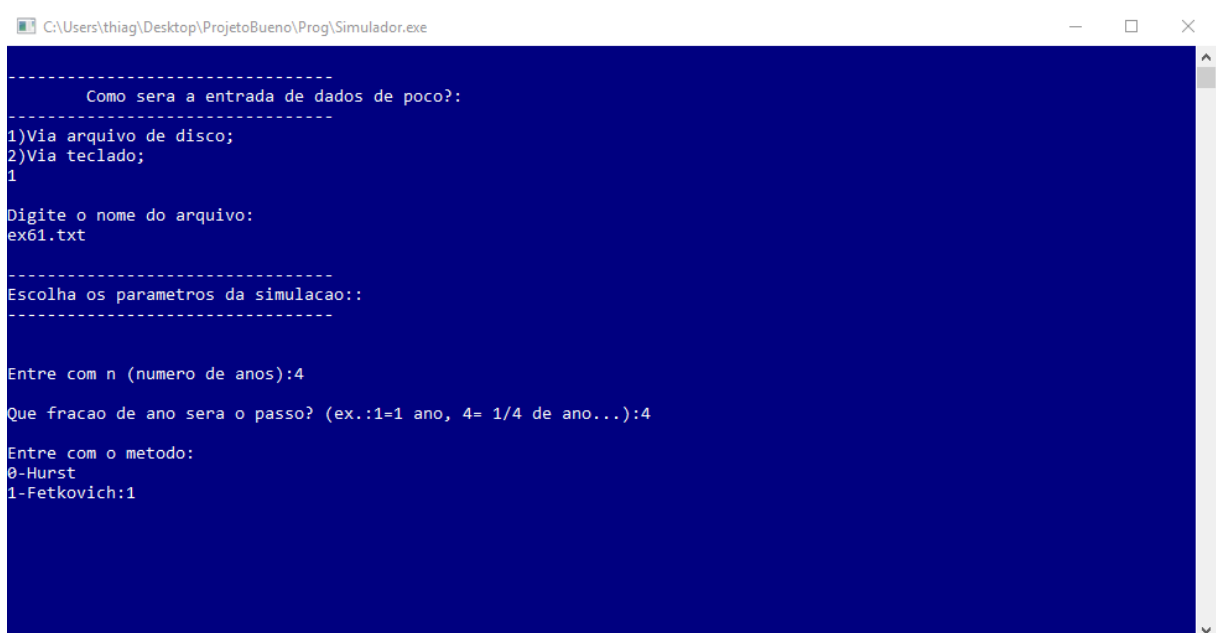
=====INSTRUÇÕES DE USO=====
=
Bem-vindo ao simulador, para utilizá-lo tenha em mãos os dados PVT do fluido de reservatório assim como dados de produção
e reservatório para começar. Você pode inserir tais dados pelo teclado, porém é altamente recomendado que a entrada seja
por arquivo de disco já que para uma simulação adequada muitos dados são requeridos.

Para inserir como arquivo de disco, olhe o modelo exemplo que segue com esse programa e somente EDITE os dados ali
demonstrados. Caso altere o formato do arquivo, terá que alterar o código também. Após editar os dados que quiser, salve
o arquivo no mesmo diretório do programa. No programa, digite o nome do arquivo e extensão para começar a simulação.

Todos os dados, para resultar em uma simulação real e de proveito, precisarão estar no sistema PETROBRAS de unidades
=====
=

Como se dará a entrada de dados? Digite 1-Manual por Teclado / 2-Arquivo de Disco
```

Figura 7.3: Interface-Capa de Gás/Gás em Solução



```
C:\Users\thiag\Desktop\ProjetoBueno\Prog\Simulador.exe

-----
Como será a entrada de dados de poço?:
-----
1)Via arquivo de disco;
2)Via teclado;
1

Digite o nome do arquivo:
ex61.txt

-----
Escolha os parâmetros da simulação::
-----

Entre com n (número de anos):4
Que fração de ano será o passo? (ex.:1=1 ano, 4= 1/4 de ano...):4

Entre com o método:
0-Hurst
1-Fetkovich:1
```

Figura 7.4: Incluindo parâmetros da simulação

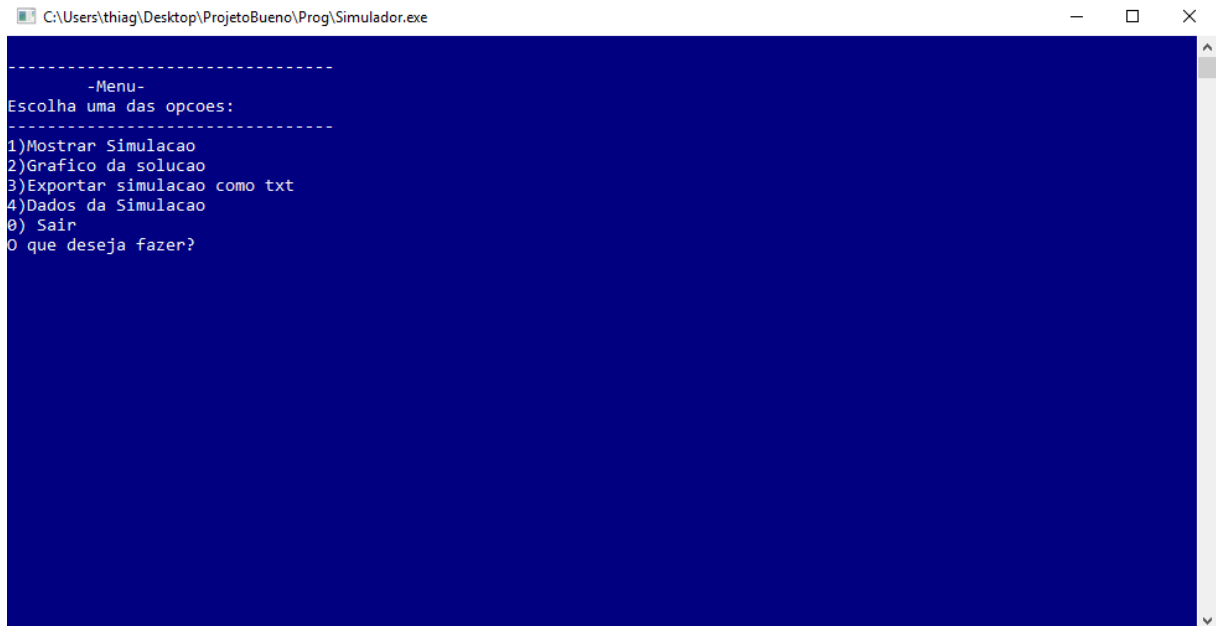


Figura 7.5: Menu de opções da simulação

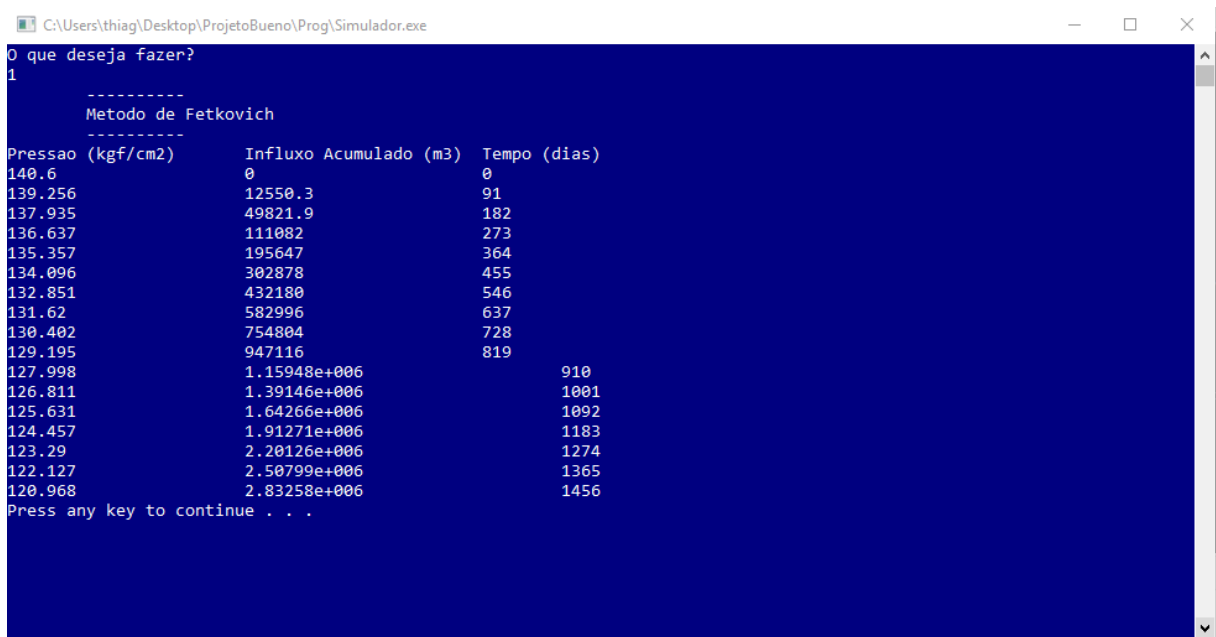


Figura 7.6: Resultados do cálculo de influxo de água

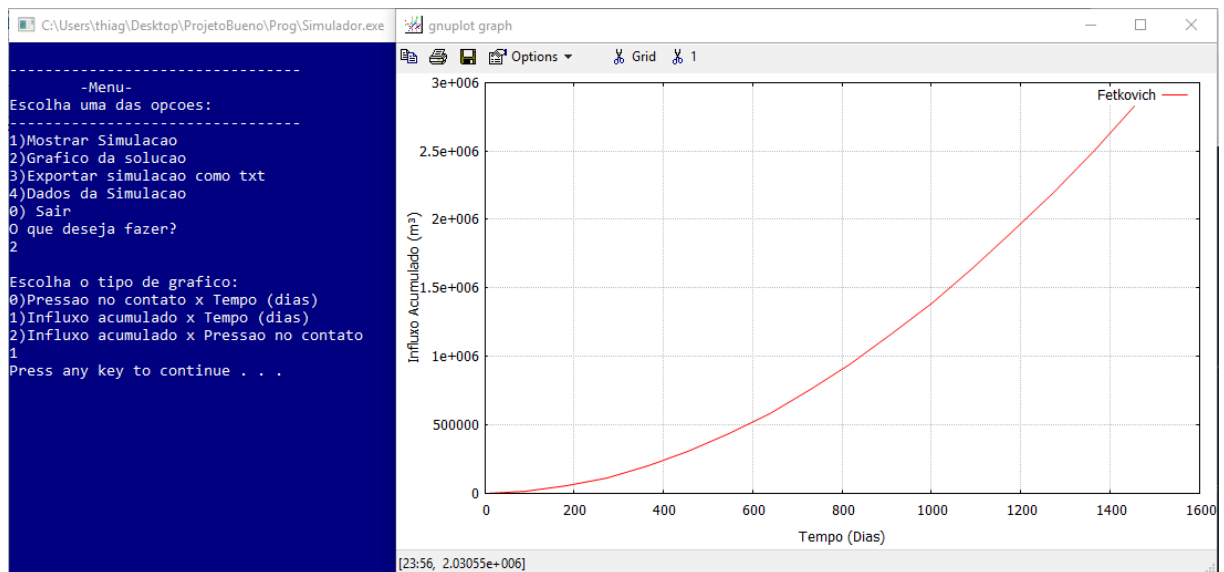


Figura 7.7: Gráfico do Influxo de Água Acumulado

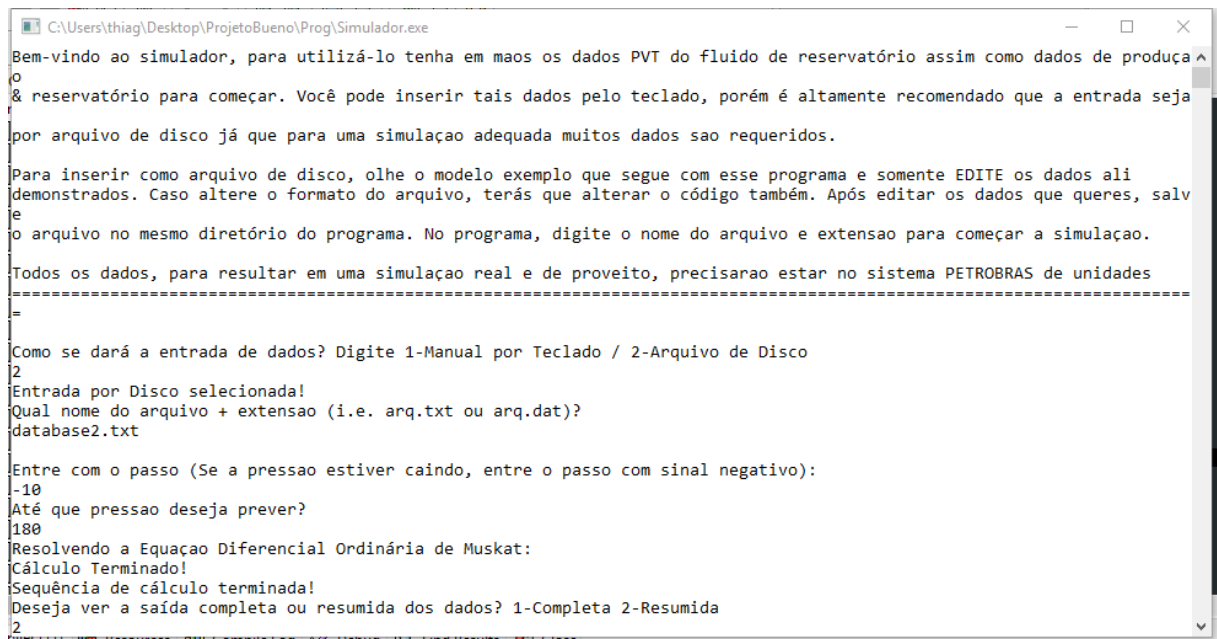


Figura 7.8: Entrada por arquivo de disco e escolha dos parâmetros de reservatório

C:\Users\thiag\Desktop\ProjetoBueno\Prog\Simulador.exe

A tabela a seguir contém os resultados resumidos de maior interesse para o usuário do programa

Pressao	Saturação de Oleo	Produção Acumulada de Oleo	Produção Acumulada de Gás	Razao Gás-Oleo Instantânea
kgf/cm²	adimensional	m3std	m3std	m3std/m3std
210	0.58259533	3500000	2.00489e+008	356
200	0.57816693	3667409	2.6220437e+008	370.16375
190	0.57373769	3843006.7	3.2822529e+008	385.73533
180	0.56923833	4026702.6	3.9805858e+008	402.52675

Qual gráfico das propriedades dos fluídos de reservatório deseja plotar?

- 1- Bo contra P
- 2- Rs contra P
- 3- Inverso de Bg contra P
- 4- Curva de Permeabilidade Kg contra Ko
- 5- Nenhum

Figura 7.9: Saída resumida da simulação

C:\Users\thiag\Desktop\ProjetoBueno\Prog\Simulador.exe

Deseja salvar o gráfico? 1-SIM 2-NAO

2

Deseja ver outro gráfico dentre os quatro aqui apresentados? 1-SIM 2-NAO

1

Qual gráfico das propriedades dos fluídos de reservatório deseja plotar?

- 1- Bo contra P
- 2- Rs contra P
- 3- Inverso de Bg contra P
- 4- Curva de Permeabilidade Kg contra Ko
- 5- Nenhum

2

Pressione ENTER para continuar

Deseja salvar o gráfico? 1-SIM 2-NAO

1

Qual o nome desejado para o graficox?

plt

Pressione ENTER para continuar

Deseja ver outro gráfico dentre os quatro aqui apresentados? 1-SIM 2-NAO

2

Deseja ver o gráfico da Saturação de Oleo versus Pressao do Reservatório? 1-SIM 2-NAO

1

Press any key to continue . . .

Figura 7.10: Método gráfico (note que há a opção de salvar o gráfico)

7.5 Teste 5: Salvando simulação e saindo do programa

As imagens a seguir mostram o processo simples de exportar a simulação em um arquivo de disco. A imagem 7.14 mostra a tela final do programa (comum às duas interfaces).

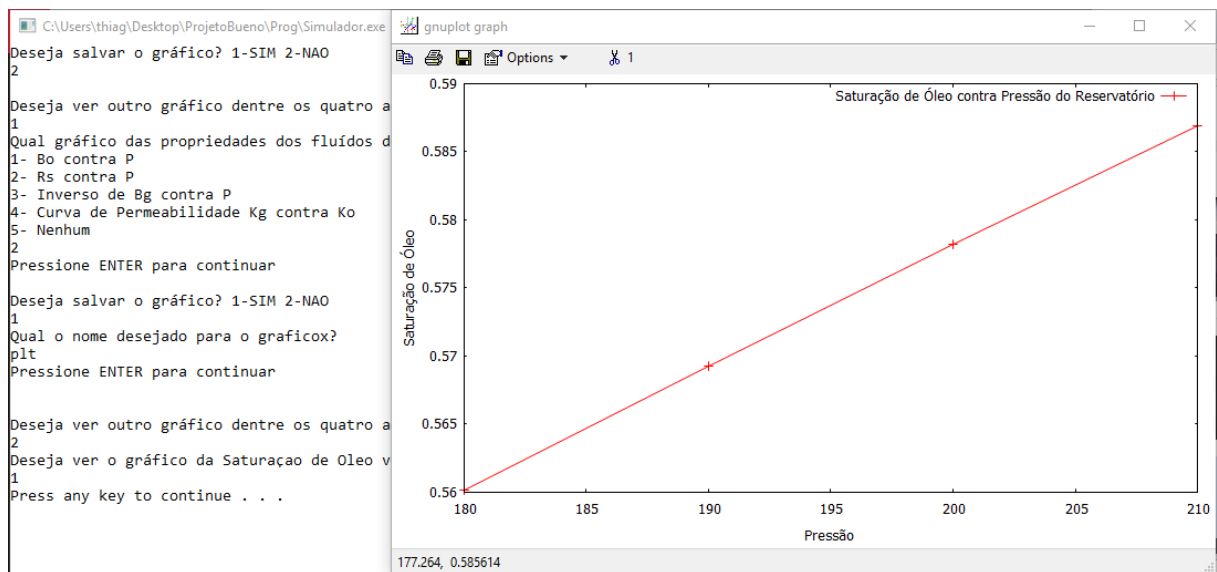


Figura 7.11: Gráfico da saturação de óleo versus pressão

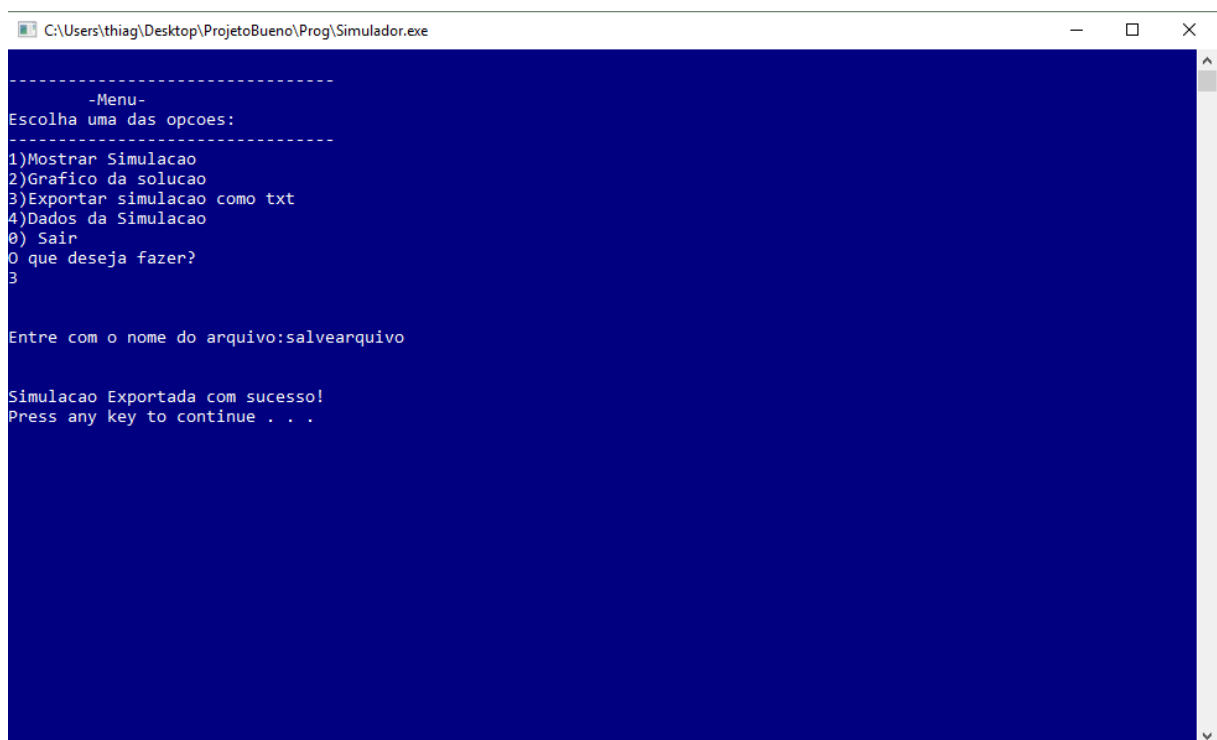


Figura 7.12: Exportação da simulação do influxo de água



Figura 7.13: Exportação da simulação da saturação de óleo

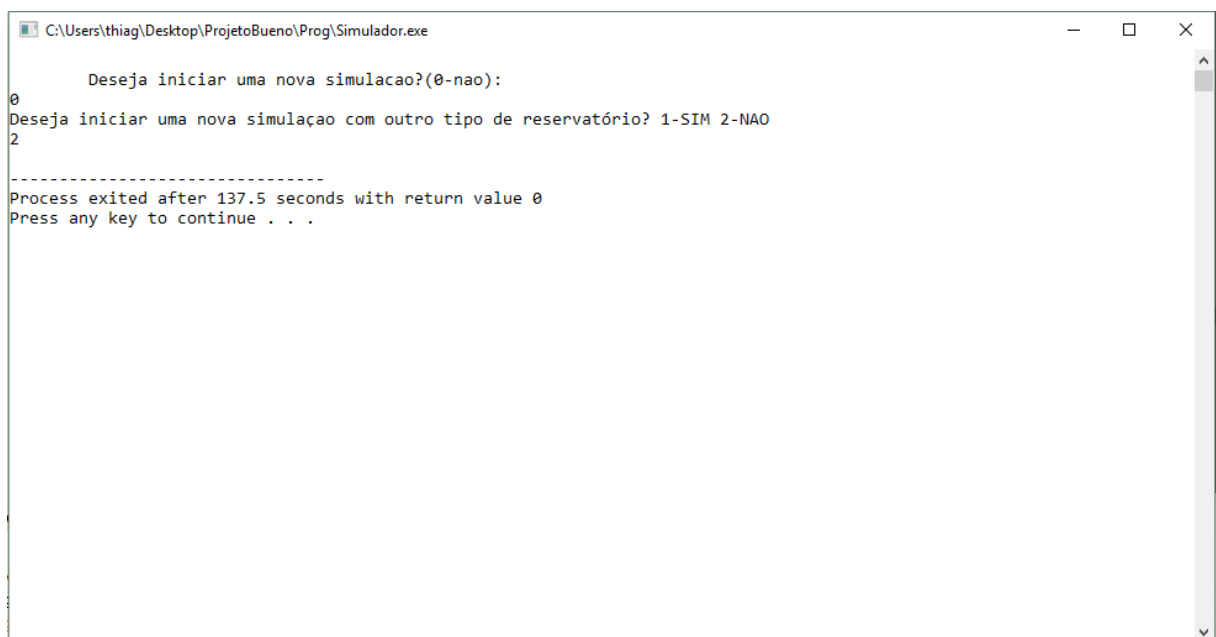


Figura 7.14: Tela final do programa

Capítulo 8

Documentação

Todo projeto de engenharia precisa ser bem documentado. Neste sentido, apresenta-se neste capítulo a documentação de uso do “Simulador de parâmetros de comportamento de reservatório com influxo de água, capa de gás ou gás em solução”. 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 o software execute o seguinte passo a passo:

Em Linux: Abra o terminal, vá para o diretório onde está o simulador, faça a compilação e depois, o execute.

Em Windows: Faça o download de um compilador, como por exemplo o *Dev C++* disponível em <https://dev-c.softonic.com.br/>. Compile o simulador e execute-o.

8.1.2 Como rodar o software

Após compilado, preencha o arquivo de texto nos moldes que vem com o simulador (caso escolha pela entrada por arquivo de disco). Por mais que sejam arquivos-exemplo, eles são modelos de funcionamento e sua organização não pode ser alterada, caso seja, o simulador não rodará adequadamente, erros fatais são esperados. Um exemplo do arquivo-exemplo é o que está na figura 8.1.

É preferível que a entrada seja por arquivo de disco, o que tornará a simulação menos laborosa. Porém, caso prefira, a entrada de dados pode ser manual diretamente na tela do simulador. Ao executar, o usuário será levado à tela inicial, onde escolherá o tipo de reservatório a qual simular. Escolhido o tipo, uma interface será aberta para a entrada de

dados e posterior previsão dos parâmetros comportamentais do reservatório. O usuário poderá ver os resultados dependendo de sua escolha no menu da interface. Gráficos também podem ser gerados se o usuário assim escolher na tela do simulador. É dada a opção de salvar a simulação e de começar outra simulação com o mesmo tipo de reservatório ou com outro tipo de reservatório. Todo esse processo já foi ilustrado no capítulo 7.

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

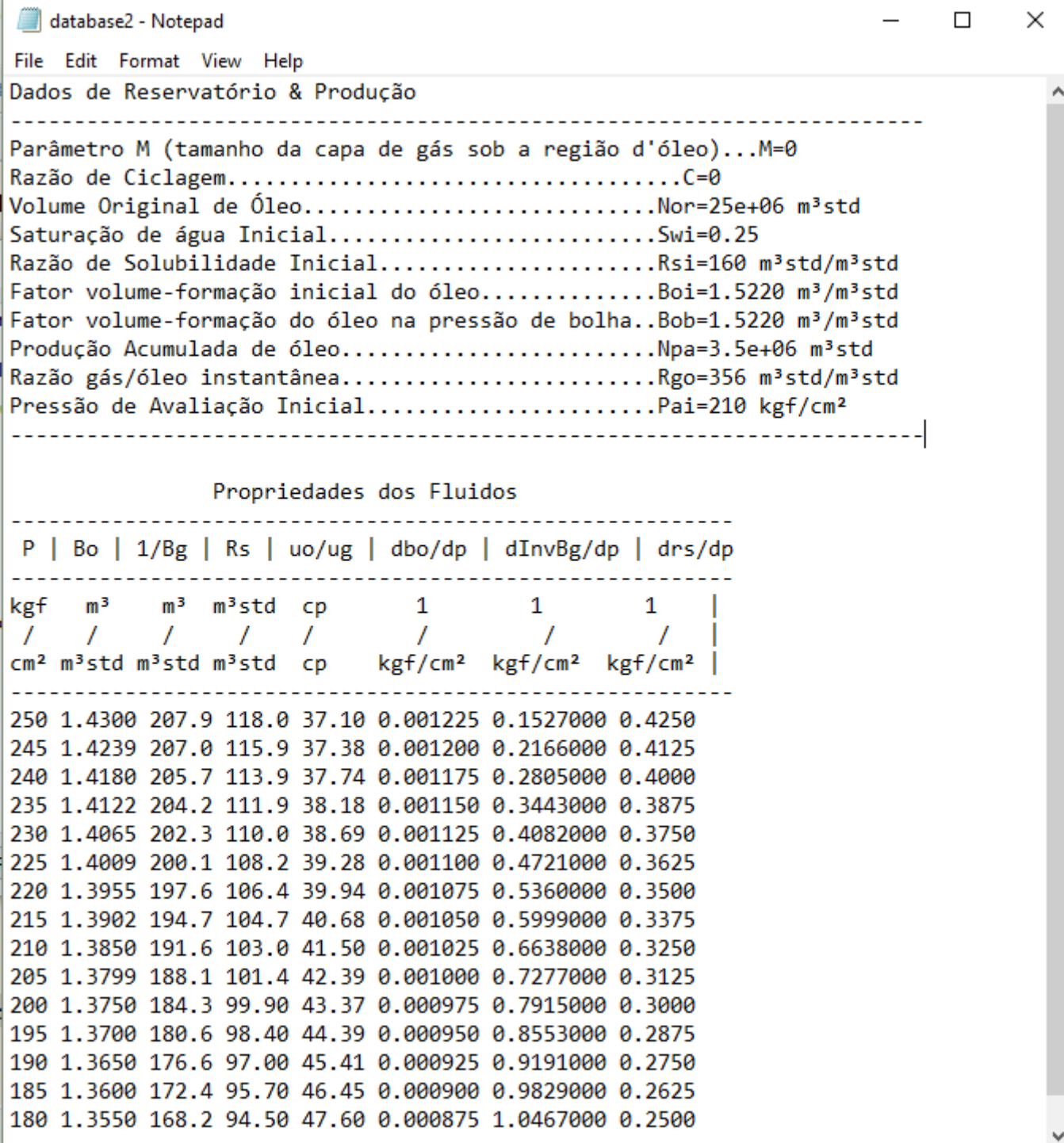
Para compilar o software é necessário atender as seguintes dependências:

- 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`.
- Arquivos com dados de reservatório, é imperativo que a ordem dos dados não seja mudada. Caso queira modificar a ordem, é necessária uma alteração no código.
- É possível alterar as cores do terminal para as diferentes interfaces, os comandos estão no arquivo `main.cpp`.

8.2.2 Como gerar a documentação usando doxygen

A documentação do código do simulador foi feita segundo o padrão JAVADOC, através do software `doxygen` que lê os arquivos com os códigos (*.h e *.cpp) e gera uma documentação muito útil e de fácil navegação no formato html.

Apresenta-se a seguir algumas imagens com as telas das saídas geradas pelo software `doxygen`.



```

database2 - Notepad
File Edit Format View Help
Dados de Reservatório & Produção
-----
Parâmetro M (tamanho da capa de gás sob a região d'óleo)...M=0
Razão de Ciclagem.....C=0
Volume Original de Óleo.....Nor=25e+06 m³std
Saturação de água Inicial.....Swi=0.25
Razão de Solubilidade Inicial.....Rsi=160 m³std/m³std
Fator volume-formação inicial do óleo.....Boi=1.5220 m³/m³std
Fator volume-formação do óleo na pressão de bolha..Bob=1.5220 m³/m³std
Produção Acumulada de óleo.....Npa=3.5e+06 m³std
Razão gás/óleo instantânea.....Rgo=356 m³std/m³std
Pressão de Avaliação Inicial.....Pai=210 kgf/cm²
-----

Propriedades dos Fluidos
-----
P | Bo | 1/Bg | Rs | uo/ug | dbo/dp | dInvBg/dp | drs/dp
-----
kgf m³ m³ m³std cp 1 1 1 |
/ / / / / / / / |
cm² m³std m³std m³std cp kgf/cm² kgf/cm² kgf/cm² |
-----
250 1.4300 207.9 118.0 37.10 0.001225 0.1527000 0.4250
245 1.4239 207.0 115.9 37.38 0.001200 0.2166000 0.4125
240 1.4180 205.7 113.9 37.74 0.001175 0.2805000 0.4000
235 1.4122 204.2 111.9 38.18 0.001150 0.3443000 0.3875
230 1.4065 202.3 110.0 38.69 0.001125 0.4082000 0.3750
225 1.4009 200.1 108.2 39.28 0.001100 0.4721000 0.3625
220 1.3955 197.6 106.4 39.94 0.001075 0.5360000 0.3500
215 1.3902 194.7 104.7 40.68 0.001050 0.5999000 0.3375
210 1.3850 191.6 103.0 41.50 0.001025 0.6638000 0.3250
205 1.3799 188.1 101.4 42.39 0.001000 0.7277000 0.3125
200 1.3750 184.3 99.90 43.37 0.000975 0.7915000 0.3000
195 1.3700 180.6 98.40 44.39 0.000950 0.8553000 0.2875
190 1.3650 176.6 97.00 45.41 0.000925 0.9191000 0.2750
185 1.3600 172.4 95.70 46.45 0.000900 0.9829000 0.2625
180 1.3550 168.2 94.50 47.60 0.000875 1.0467000 0.2500

```

Figura 8.1: Arquivo de entrada de disco modelo

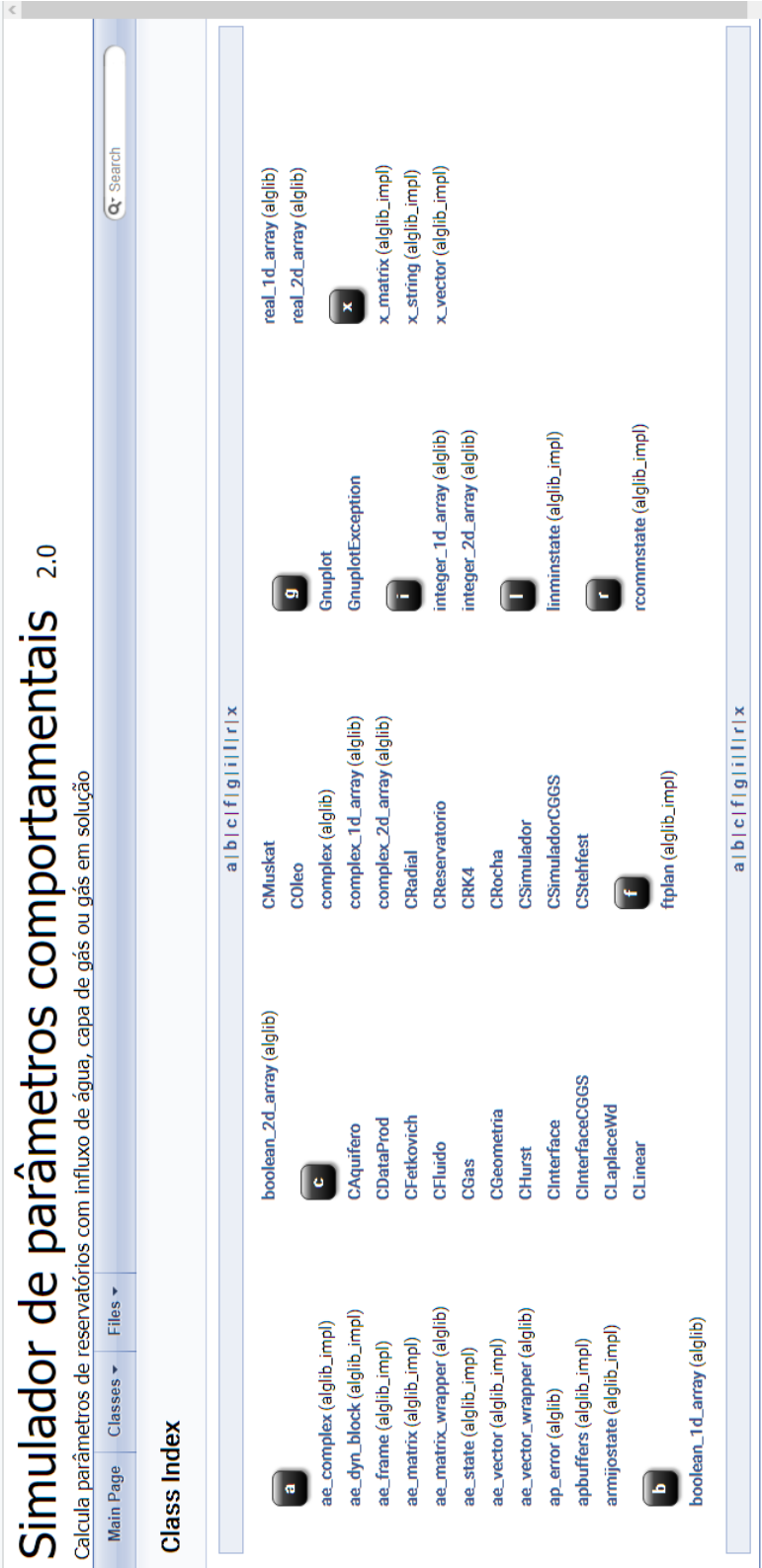


Figura 8.2: Lista de Classes Doxygen

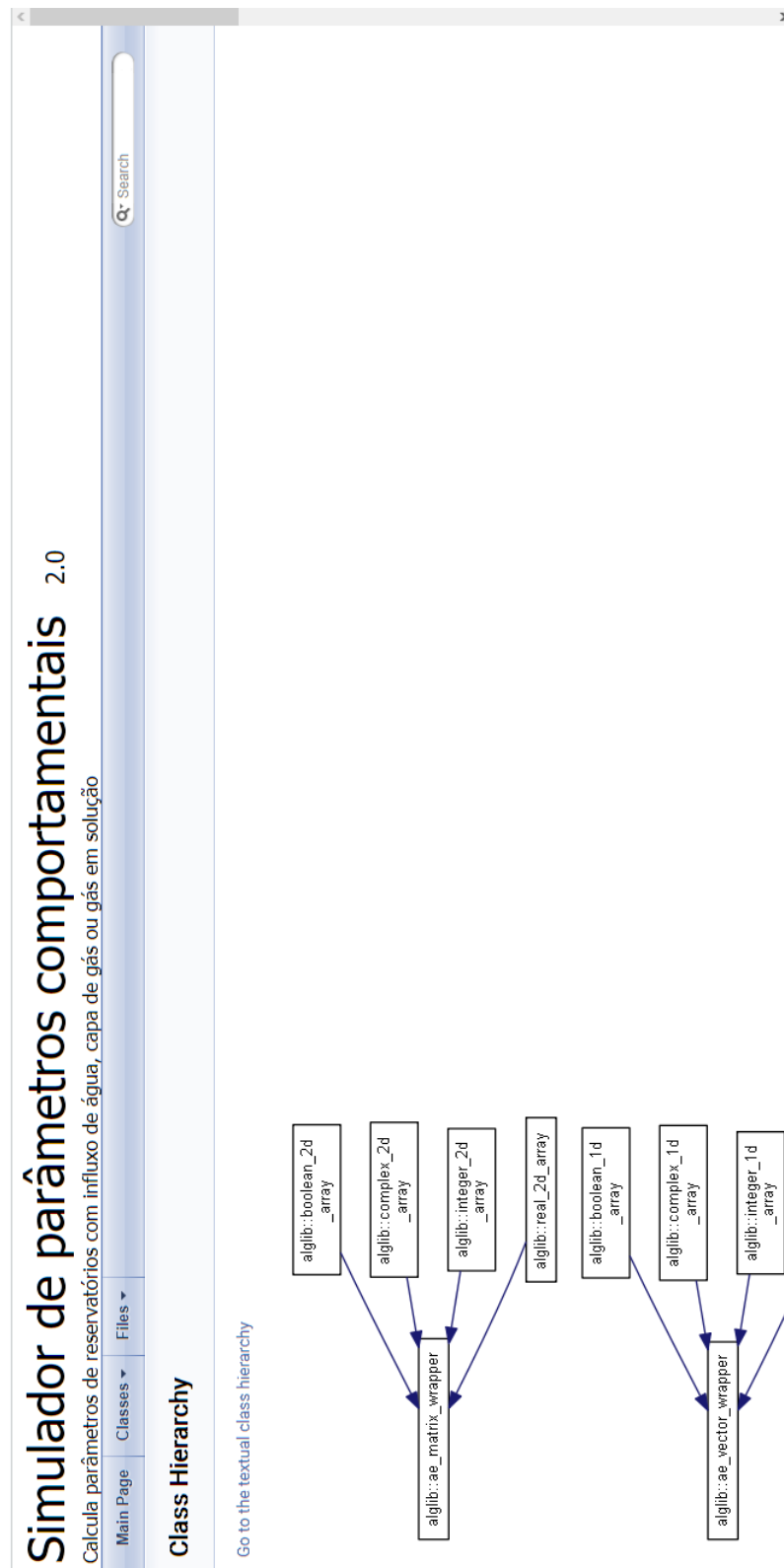


Figura 8.3: Diagrama de hierarquia de classes gerado pelo programa Dot do pacote graphviz - Doxygen

Índice Remissivo

A

Análise orientada a objeto, 18
AOO, 18
Associações, 31
atributos, 31

C

Casos de uso, 5
colaboração, 24
comunicação, 24
Concepção, 4
Controle, 29

D

Diagrama de colaboração, 24
Diagrama de componentes, 31
Diagrama de execução, 34
Diagrama de máquina de estado, 25
Diagrama de sequência, 22

E

Efeitos do projeto nas associações, 31
Efeitos do projeto nas heranças, 31
Efeitos do projeto nos métodos, 31
Elaboração, 8
especificação, 4
Especificações, 4
estado, 25
Eventos, 22

H

Heranças, 31
heranças, 31

I

Implementação, 35

M

Mensagens, 22
métodos, 31
modelo, 30, 31

O

otimizações, 31

P

Plataformas, 29
POO, 30
Projeto do sistema, 28
Projeto orientado a objeto, 30
Protocolos, 28

R

Recursos, 29