

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

PROJETO DE ENGENHARIA
DESENVOLVIMENTO DO SOFTWARE
SIMULADOR DE SUBSTITUIÇÃO DE FLUIDOS E DE
MODELAGEM AVO
TRABALHO DA DISCIPLINA PROGRAMAÇÃO PRÁTICA

Maurício Laurindo de Matos
Rafael Boechat Baptista Bastos de Oliveira
Prof. André Duarte Bueno

MACAÉ - RJ
Janeiro - 2016

Sumário

1	Introdução	1
1.1	Escopo do problema	1
1.2	Objetivos	1
2	Especificação	3
2.1	Especificação do Software - Requisitos	3
2.1.1	Nome do produto e componentes	3
2.1.2	Especificação	3
2.1.3	Requisitos funcionais	4
2.1.4	Requisitos não funcionais	4
2.2	Casos de uso do Software	4
2.3	Diagrama de caso de uso geral do software	5
2.4	Diagrama de caso de uso específico do software	5
3	Elaboração	7
3.1	Análise de domínio	7
3.2	Formulação teórica	8
3.2.1	Substituição de Fluidos	8
3.2.2	Modelagem AVO (Amplitude versus offset)	16
3.3	Diagrama de pacotes	21
4	AOO – Análise Orientada a Objeto	23
4.1	Diagramas de classes	23
4.1.1	Dicionário de classes	23
4.2	Diagrama de seqüência – eventos e mensagens	25
4.2.1	Diagrama de sequência geral	25
4.3	Diagrama de comunicação – colaboração	27
4.4	Diagrama de máquina de estado	29
4.5	Diagrama de atividades	30
5	Projeto	33
5.1	Projeto do sistema	33

5.2	Projeto orientado a objeto – POO	35
5.3	Diagrama de componentes	36
5.4	Diagrama de implantação	37
6	Implementação	38
6.1	Código fonte	38
7	Teste	78
7.1	Teste 1: Substituição de fluidos e modelagem AVO	78
7.2	Teste 2: Substituição de fluidos a partir de dados de perfis	79
8	Documentação	89
8.1	Documentação do usuário	89
8.1.1	Como rodar o software	89
8.2	Documentação para desenvolvedor	90
8.2.1	Dependências	91
8.2.2	Documentação usando doxygen	91
8.3	Sugestões para trabalhos futuros	91
9	Referências Bibliográficas	93
10	Título do Apêndice	96
10.1	Sub-Título do Apêndice	96
11	Título do Apêndice	97
11.1	Roteiro Para Uso do Sistema de Citações Com Banco de Dados .bib . . .	98
11.1.1	Citações no meio do texto	99
11.1.2	Incluir nas referências bibliográficas (fim do documento), mas não citar	99
11.2	Informações adicionais	99

Capítulo 1

Introdução

No presente projeto de engenharia desenvolve-se o software “Simulador de substituição de fluidos e de modelagem AVO” que utiliza conceitos das disciplinas de perfilagem, métodos geofísicos e petrofísica. O simulador auxilia na caracterização dos fluidos e rochas de reservatório, permitindo identificar através dos perfis possíveis zonas de óleo.

1.1 Escopo do problema

Durante a etapa de exploração do petróleo utiliza-se uma série de métodos geofísicos para inferir as características das rochas e fluidos em subsuperfície, buscando identificar possíveis intervalos de acumulação de hidrocarbonetos.

Diante disso, a importância da geofísica está relacionada com a possibilidade de auxiliar na caracterização de um reservatório, fornecendo uma maior confiabilidade no potencial produtor de um sistema petrolífero. Tais métodos são utilizados em larga escala e para um conjunto de dados muito grande, o que destaca a relevância de se ter softwares de alto desempenho, que sejam capazes de trabalhar com esses dados.

Esta tendência é a motivação do nosso projeto de engenharia onde busca-se criar um simulador de substituição de fluidos e de modelagem AVO (*Amplitud versus offset*) com a finalidade de caracterizar intervalos de reservatórios, visando auxiliar o usuário na etapa de interpretação geofísica.

1.2 Objetivos

Os objetivos deste projeto de engenharia são:

- Objetivo geral:
 - Realizar a substituição de fluidos para plotar um gráfico de velocidade da onda compressional pela saturação de água e de velocidade da onda cisalhante pela saturação de água .

- Realizar a modelagem AVO de uma saturação de água escolhida, para identificar o tipo de anomalia ou classe AVO que qualifica o sistema petrolífero.
 - Plotar perfis para auxiliar na caracterização do reservatório e realizar a substituição de fluidos no intervalo analisado.
- Objetivos específicos:
 - Modelar fisicamente e matematicamente o problema.
 - Modelagem estática do software (diagramas de caso de uso, de pacotes, de classes).
 - Modelagem dinâmica do software (desenvolver algoritmos e diagramas exemplificando os fluxos de processamento).
 - Calcular as propriedades físicas da rocha.
 - Calcular as propriedades dos fluidos.
 - Simular (realizar simulações para teste do software desenvolvido).
 - Implementar manual simplificado de uso do software.

Capítulo 2

Especificação

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

2.1 Especificação do Software - Requisitos

2.1.1 Nome do produto e componentes

Nome	Simulador de substituição de fluidos e de modelagem AVO.
Componentes principais	Desenvolvimento de sistemas para a determinação de parâmetros geofísicos e caracterização de reservatórios.
Missão	Auxiliar profissionais e alunos na caracterização e interpretação de reservatórios por meio de parâmetros geofísicos.

2.1.2 Especificação

No projeto serão implementadas duas rotinas de trabalho. Na primeira, será proposto ao usuário uma configuração hipotética de uma rocha reservatório trapeada por uma rocha selante sobreposta horizontalmente. Será realizada a substituição de fluidos na rocha reservatório obtendo como as propriedades da camada (velocidades e densidade) variam com o aumento da saturação de água, processo comum no decorrer da produção. Posteriormente será realizada a modelagem AVO para um vários cenários de saturação, buscando uma melhor caracterização do reservatório.

Na segunda rotina, o usuário fornecerá dados de perfilagem de um intervalo do reservatório, os perfis serão plotados e será aplicada a substituição de fluidos no intervalo correspondente.

Portanto, o programa irá trabalhar com as propriedades dos fluidos e rochas. A interação com o usuário será através de interface em modo texto, onde serão expostas ao usuário diversas opções do que se deseja executar com os resultados sendo apresentados na tela. O programa será capaz de ler dados de entrada e gerar dados de saída em arquivos de disco em determinadas operações, além de plotar gráficos utilizando a biblioteca externa Gnuplot. Ademais o programa fornecerá ao usuário a opção de *help* que proporcionará um serviço de apoio e suporte para uma melhor compreensão.

Para fins acadêmicos, o presente software tem licença GPL 2, podendo ser livremente distribuído e copiado.

2.1.3 Requisitos funcionais

Apresenta-se a seguir os requisitos funcionais.

RF-01	O usuário deverá ter liberdade para escolher o que deseja simular.
RF-02	Os resultados deverão estar disponíveis em gráficos ou exportados como texto e/ou gráfico.
RF-03	O <i>software</i> deverá oferecer ao usuário a opção de <i>help</i> .
RF-04	O <i>software</i> deverá permitir o carregamento de dados pelo usuário.

2.1.4 Requisitos não funcionais

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

2.2 Casos de uso do Software

A Tabela 2.1 apresenta um caso de uso que descreve um ou mais cenários de uso do software, exemplos de uso, como o sistema interage com usuários externos (atores) e algumas das etapas a serem seguidas, representando uma sequência típica de uso do programa.

Tabela 2.1: Caso de uso do Softwares

Nome do caso de uso:	Simulação de substituição de fluidos e de modelagem AVO.
Resumo/descrição:	Simulação Geofísica.
Etapas:	<ol style="list-style-type: none"> 1. Escolher o tipo de simulação a ser executada. 2. Fornecer dados de entrada. 3. Realizar simulação. 4. Exibir resultados em gráficos e em modo texto. 5. Analisar resultados.
Cenários alternativos:	Inserir dados incompatíveis com as propriedades do sistema.

2.3 Diagrama de caso de uso geral do software

O diagrama apresentado na Figura 2.1 demonstra um cenário geral de uso do sistema pelo usuário.

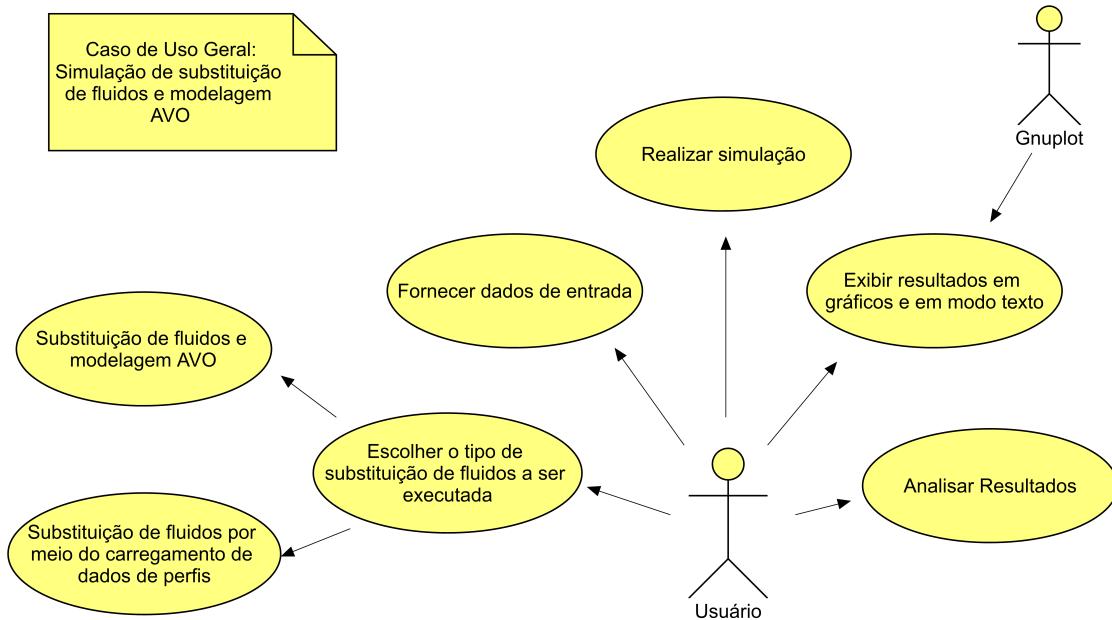


Figura 2.1: Diagrama de caso de uso – Caso de uso geral

2.4 Diagrama de caso de uso específico do software

Os diagramas a seguir representam um detalhamento da escolha do tipo de simulação escolhida pelo usuário.

Nas Figuras 2.2 e 2.3 apresentamos os diagramas de caso de uso específico da primeira e segunda simulação realizada pelo software respectivamente.

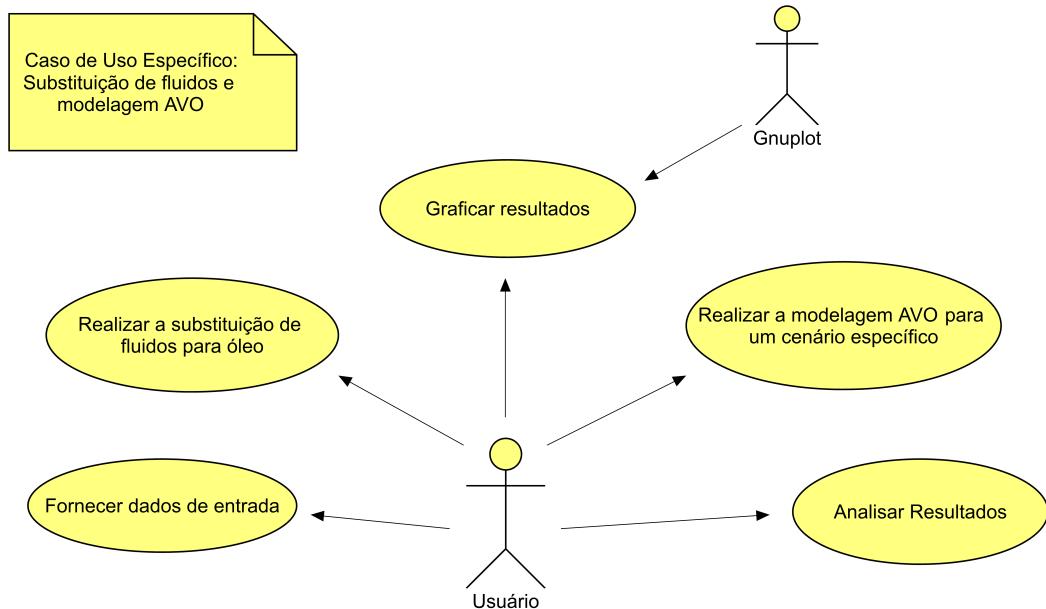


Figura 2.2: Diagrama de caso de uso específico: Substituição de fluidos e modelagem AVO

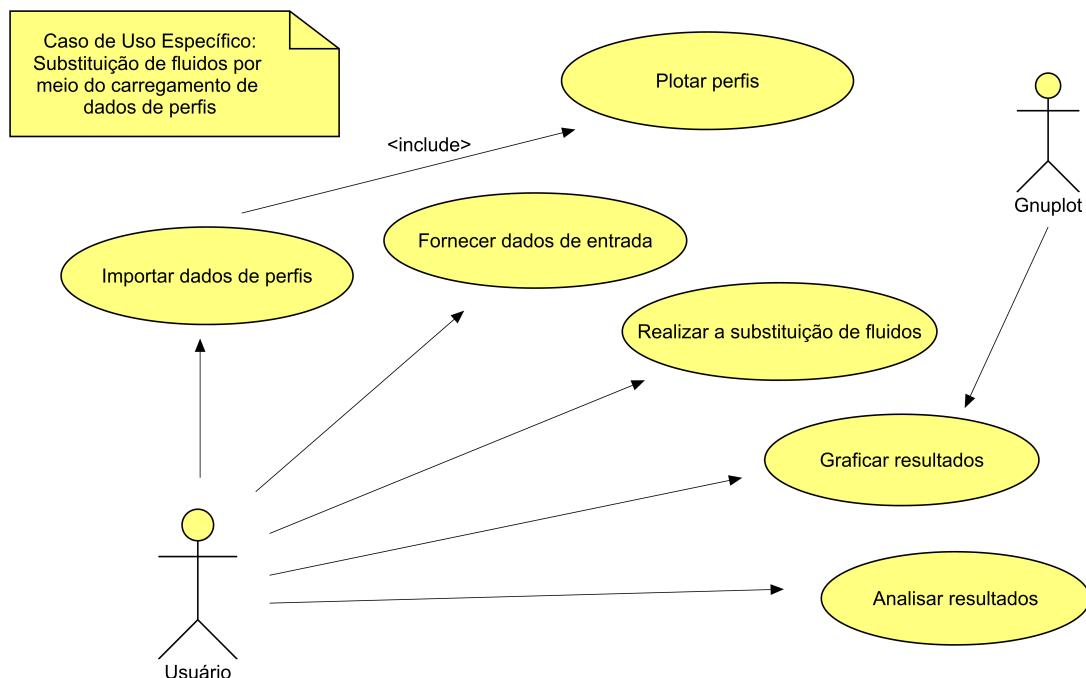


Figura 2.3: Diagrama de caso de uso específico .

Capítulo 3

Elaboração

Neste capítulo será apresentada a elaboração do *software*, que é feito através de pesquisas bibliográficas e entrevistas, que nos mostram o que é necessário para a formulação do programa. Ademais, será feita uma análise dos requisitos para o funcionamento do programa, avaliando as condições necessárias para o desenvolvimento de um sistema útil, de forma que satisfaça as necessidades do usuário e que permita, futuramente, uma extensão do *software*.

3.1 Análise de domínio

Neste tópico será apresentada uma análise do domínio do sistema a ser desenvolvido. Uma melhor compreensão deste trabalho parte do pressuposto que o usuário possui um conhecimento básico acerca das áreas e sub-áreas apresentadas.

Uma área relacionada ao desenvolvimento deste *software* é a geofísica. É uma área da engenharia de petróleo que atua na exploração de hidrocarbonetos, permitindo avaliar qualitativamente e quantitativamente o potencial de reservatórios.

As propriedades elásticas das rochas, assim como as propriedades dos fluidos que ocupam os seus poros, são as bases da investigação geofísica, portanto a petrofísica torna-se também uma área fundamental para a criação do *software*.

Outra área de grande importância para a formulação do programa será a modelagem numérica computacional, que trabalha com o desenvolvimento de modelos de numéricos para a solução de problemas físicos, aplicando esta solução por meio de algoritmos computacionais.

Uma sub-área envolvida será a sísmica, que baseia-se na medição dos tempos que as ondas sísmicas levam a atravessar as camadas sedimentares. Os estudos envolvem uma fonte de ondas elásticas que se propagam ao longo das rochas e dos reservatórios, sendo depois refletidos para receptores que registram a informação recebida.

Outra sub-área será a perfilação de poços, que através de perfis relacionam a profundidade com alguma propriedade física das rochas, permitindo avaliar o sistema petrolífero.

Devido ao *software* apresentar uma interface de fácil compreensão com o usuário e com uma base teórica relacionada à conteúdos apresentados no meio acadêmico, ele pode ser facilmente utilizado como subsídio para disciplinas derivadas das áreas e sub-áreas acima citadas.

3.2 Formulação teórica

3.2.1 Substituição de Fluidos

Equações de Gassman

O tipo de fluido presente nos poros das rochas (água, óleo ou gás) influencia de diversas formas as propriedades sísmicas. Com isso, é importante construir maneiras de testar e simular diversos aspectos da saturação. Segundo Smith et al., 2003, a substituição de fluidos é uma importante ferramenta no trabalho com atributos sísmicos, pois é capaz de modelar e quantificar vários cenários de fluidos de saturação que modificam uma observação final.

A técnica mais utilizada para realizar a substituição de fluidos é a aplicação das equações de Gassmann, que são utilizadas para obter o módulo de incompressibilidade da rocha saturada por um fluido, a partir de sua porosidade e dos módulos de incompressibilidade da rocha seca, dos fluidos de saturação e da matriz mineral.

A aplicação de tais equações é baseada em algumas suposições. Gassmann assume toda rocha como sendo isotrópica, com módulos de incompressibilidade e cisalhamento dos constituintes minerais constantes. Além disso, considera-se um comprimento de onda infinito, perfeita conectividade entre os poros e fluido com viscosidade zero. Esse pressuposto significa que uma onda sísmica se propagando na rocha produz um excesso na pressão de poros que pode ser desprezado, pois o fluido tem liberdade e tempo para fluir no espaço poroso, eliminando qualquer gradiente de pressão.

A rotina de trabalho para essa técnica se inicia com o levantamento das velocidades das ondas P e S para rochas saturadas com o fluido original de onde se pode extrair os módulos de incompressibilidade e de cisalhamento nessas condições. Posteriormente, após a substituição do fluido em saturação, é calculado o módulo de incompressibilidade da rocha saturada pelo novo fluido por Gassmann e as velocidades sísmicas são reconstituídas. As equações matemáticas que descrevem esse processo são apresentadas a seguir.

Para se modificar um tipo de fluido se faz necessário o conhecimento dos efeitos do fluido original sobre a rocha saturada. Encontrando as propriedades da rocha seca, pode-se então incluir os efeitos do novo fluido para que sejam obtidas as novas propriedades. Para isso, através da principal equação de Gassmann é possível obter o K_{dry} módulo de incompressibilidade da rocha seca [Kg/cm^2].

$$K_{dry} = \frac{K_{sat} \left(\frac{\phi K_{ma}}{K_{fl}} + 1,0 - \phi \right) - K_{ma}}{\phi \frac{K_{ma}}{K_{fl}} + \frac{K_{sat}}{K_{ma}} - 1,0 - \phi} \quad (3.1)$$

Onde:

- $K_{sat}[Kgf/cm^2]$ é o módulo de incompressibilidade da rocha saturada com o fluido original;
- $K_{fl} [Kgf/cm^2]$ é o módulo de incompressibilidade do fluido;
- $K_{ma}[Kgf/cm^2]$ é o módulo de incompressibilidade da matriz mineral;
- $\phi [m^3/m^3]$ é a porosidade da rocha;

O módulo de incompressibilidade da rocha saturada, $K_{sat} [Kgf/cm^2]$, com o fluido inicial pode ser obtido através do conhecimento dos valores das velocidades $V_p [m/s]$ e $V_s[m/s]$ e densidade nas condições iniciais.

$$K_{sat} = \rho \left(V_p^2 - \frac{4,0}{3,0} V_s^2 \right) \quad (3.2)$$

Onde:

- $V_p [m/s]$ é a velocidade da onda P;
- $V_s [m/s]$ é a velocidade da onda S;
- $\rho [g/cm^3]$ é a densidade.

Para o cálculo do módulo de incompressibilidade do fluido é considerado um modelo de fluido efetivo. Este modelo assume que as diferentes fases do fluido estão distribuídas uniformemente no meio poroso permitindo substituí-los por um fluido efetivo cuja compressibilidade é calculada através de uma média harmônica entre os módulos de incompressibilidade de cada fluido $K_i[kgf/cm^3]$ e suas saturações correspondentes $S_i[m^3/m^3]$.

$$K_{fl} = \left(\sum_{i=1}^n \frac{S_i}{K_i} \right)^{-1} \quad (3.3)$$

Onde:

- n é o número de fases.

As rochas são compostas por diversos minerais. Para o cálculo do módulo de incompressibilidade da matriz rochosa é necessário o conhecimento dessa composição e da fração dos componentes principais. A Tabela 3.1 apresenta diversos minerais comumente encontrados nas rochas e seus valores de módulo de incompressibilidade, cisalhamento e densidade. Assim como o cálculo para os fluidos, será considerado um mineral efetivo cujo módulo de

incompressibilidade pode ser calculado através da média de Voigh-Reuss-Hill, apresentada na Equação 3.6.

$$K_{voigh} = \left(\sum_{i=1}^n f_i K_i \right) \quad (3.4)$$

$$K_{reuss} = \left(\sum_{i=1}^n \frac{f_i}{K_i} \right)^{-1} \quad (3.5)$$

$$K_{vrh} = \frac{1}{2} (K_{voigh} + K_{reuss}) \quad (3.6)$$

Onde:

- $f_i [m^3/m^3]$ é a fração de cada mineral;
- $K_i [Kgf/cm^3]$ é o módulo de incompressibilidade de cada mineral;
- n é o número de minerais constituintes.

A porosidade pode ser obtida através de dados de perfilagem ou por análises em amostras de testemunhos. Obtido o módulo de incompressibilidade para rocha seca $K_{dry} [Kgf/cm^3]$ podemos rearranjar a Equação 3.1 de forma que fique em função do módulo de incompressibilidade da rocha saturada e assim modificar as condições para o novo fluido. Para a nova condição de saturação, valores dos módulos de incompressibilidade para a rocha seca e para a matriz serão os mesmos, bastando modificar o módulo de incompressibilidade do fluido influenciado pela nova condição de saturação.

$$K_{sat} = K_{dry} + \frac{\left(1,0 - \frac{K_{dry}}{K_{ma}}\right)^2}{\frac{\phi}{K_{fl}} + \frac{\phi}{K_{ma}} - \frac{K_{dry}}{K_{ma}^2}} \quad (3.7)$$

Assim, as novas velocidades podem ser calculadas pelas Equações 3.8 e 3.9.

$$V_p = \sqrt{\frac{K_{sat} + \frac{4,0}{3,0}\mu}{\rho}} \quad (3.8)$$

$$V_s = \sqrt{\frac{\mu}{\rho}} \quad (3.9)$$

Onde $\mu [MPa]$ é o módulo de cisalhamento, que pode ser calculado pela Equação 3.10.

$$\mu = \rho V_s^2 \quad (3.10)$$

Alternativamente, os módulos de incompressibilidade e de cisalhamento para a rocha seca podem ser obtidos pela correlação proposta por Murphy et. al (1993) para arenitos limpos ou pela correlação de Nur et. al (1995) apresentadas respectivamente nas Equações 3.11, 3.12, 3.13 e 3.14:

$$K_{dry} = 38,18 (1,0 - 3,34\phi + 1,95\phi^2) \quad (3.11)$$

$$\mu = 42,65 (1,0 - 3,48\phi + 2,19\phi^2) \quad (3.12)$$

$$\mu = \mu_{ma} \left(1,0 - \frac{\phi}{\phi_c} \right) \quad (3.13)$$

$$K_{dry} = K_{ma} \left(1,0 - \frac{\phi}{\phi_c} \right) \quad (3.14)$$

Onde:

- μ_{ma} [MPa] é o módulo de cisalhamento da matriz que pode ser obtido de fórmula análoga ao módulo de incompressibilidade através das equações de Voigh-Reuss-Hill. A Tabela 3.1 apresenta diferentes módulos de cisalhamentos para minerais formadores das rochas.
- $\phi_c[m^3/m^3]$ é a porosidade crítica da rocha, ou seja, limite máximo de porosidade que pode constituir uma rocha, pois, além disso ela é descaracterizada tornando-se fragmentos friáveis e sedimentos. A Tabela 3.2 apresenta a porosidade crítica média para diferentes rochas.

Tabela 3.1: Propriedades dos minerais

Mineral	Incompressibilidade (GPa)	Cisalhamento (GPa)	Densidade (gm/cm ³)
Quartzo	37,00	44,00	2,65
Feldspato	37,50	15,00	2,62
Plagioclásio	75,60	25,60	2,63
Argilominerais	30,00	8,20	2,80
Calcita	76,80	32,00	2,71
Dolomita	94,90	45,00	2,87
Anidrita	44,8	29,1	2,98
Pirita	147,4	135,5	4,93
Hematita	100,2	95,2	5,24

Tabela 3.2: Porosidade crítica de algumas rochas

Rochas	Porosidade Crítica
Arenitos	40%
Calcários	60%
Dolomitos	40%
Pomes	80%
Chalks	65%
Rochas ígneas crackeadas	5%

Para a obtenção das novas velocidades, descritas nas Equações 3.8 e 3.9, deve-se estimar a densidade da rocha saturada após a substituição do fluido usando o modelo de densidade efetiva, apresentada na Equação 3.15:

$$\rho = \rho_{ma} (1,0 - \phi) + \rho_{fl}\phi \quad (3.15)$$

Onde:

- $\rho_{ma} [g/cm^3]$ é a densidade da matriz;
- $\rho_{fl} [g/cm^3]$ é a densidade do fluido.

Como dito, a matriz pode não ser considerada formada por um só mineral, portanto, se faz necessário estimar sua densidade através de uma média ponderada considerando a fração de cada mineral e sua densidade. As densidades dos principais minerais formadores das rochas estão expostas na Tabela 3.1 e a densidade da matriz pode ser econtrada pela Equação 3.16:

$$\rho_{ma} = \left(\sum_{i=1}^n f_i \rho_i \right) \quad (3.16)$$

A densidade do fluido depende das fases que compõem o fluido, como água, óleo e gás e suas proporções, podendo ser calculada através da Equação 3.17.

$$\rho_{fl} = \left(\sum_{i=1}^n S_i \rho_i \right) \quad (3.17)$$

Com isso temos todas as equações necessárias para realizar o processo de substituição de fluidos segundo Gassmann.

Modelo de Batzle & Wang para estimativa das propriedades dos fluidos

Nesse tópico serão apresentadas um conjunto de equações propostas por Batzle & Wang (1992) que complementam o trabalho proposto por Gassmann. Os autores correlacionam relações termodinâmicas com dados empíricos para estimar as propriedades dos fluidos nos poros da rocha. As correlações estão divididas em propriedades para água salgada, gás e óleo.

Água

A densidade da água salgada ou salmoura $\rho_b [g/cm^3]$ é função da temperatura [$^{\circ}C$], pressão [MPa], densidade da água pura [g/cm^3] e salinidade em porcentagem.

$$\rho_b = \rho_w + S\{0,668 + 0,44S + 10^{-6}[300,0P - 2400,0PS$$

$$+T(80,0 + 3,0T - 3300,0S - 13,0P + 47,0PS)]\} \quad (3.18)$$

Por sua vez a densidade da água pura pode ser calculada pela Equação 3.19:

$$\rho_w = 1,0 + 10^{-6}(-80,0T - 3,3T^2 + 0,00175T^3 + 489,0P - 2,0TP) \quad (3.19)$$

$$+ 10^{-6}[0,016T^2P - 1,3 \cdot 10^{-5}T^3P - (0,333P^2 - 0,02TP^2)] \quad (3.19)$$

A velocidade acústica [m/s] para a salmoura obtém-se pela Equação 3.20:

$$V_b = V_w + S(1170,0 - 9,6T + 0,055T^2 - 8,5 \cdot 10^{-5}T^3 + 2,6P - 0,0029TP - 0,0476P^2) \quad (3.20)$$

$$+ S^{1,5}(780,0 - 10,0P + 0,16P^2) - 1820,0S^2 \quad (3.20)$$

Onde V_w [m/s] representa a velocidade acústica da água doce em m/s dada pela Equação 3.21:

$$V_w = \sum_{i=0}^4 \sum_{j=0}^3 w_{ij} T^i P^j \quad (3.21)$$

Os valores dos coeficientes w_{ij} são apresentados na Tabela 3.3.

Tabela 3.3: Tabela com os valores de w_{ij}

$w_{0,0}=1402,85$	$w_{0,2}=3,347 \times 10^{-3}$
$w_{1,0}=4,871$	$w_{1,2}=1,739 \times 10^{-4}$
$w_{2,0}=-0,04783$	$w_{2,2}=-2,135 \times 10^{-6}$
$w_{3,0}=1,487 \times 10^{-4}$	$w_{3,2}=-1,455 \times 10^{-8}$
$w_{4,0}=-2,197 \times 10^{-7}$	$w_{4,2}=5,23 \times 10^{-11}$
$w_{0,1}=1,524$	$w_{0,3}=-1,197 \times 10^{-5}$
$w_{1,1}=-0,0111$	$w_{1,3}=-1,628 \times 10^{-6}$
$w_{2,1}=2,747 \times 10^{-4}$	$w_{2,3}=1,237 \times 10^{-8}$
$w_{3,1}=-6,503 \times 10^{-7}$	$w_{3,3}=1,327 \times 10^{-10}$
$w_{4,1}=7,987 \times 10^{-10}$	$w_{4,3}=-4,614 \times 10^{-11}$

Por fim, pode-se calcular o módulo de incompressibilidade para a salmoura K_b [MPa]:

$$K_b = V_b^2 \rho_b \quad (3.22)$$

Gás

O gás é caracterizado por sua gravidade específica G que representa a razão entre a densidade do gás e do ar a 15,6°C e pressão atmosférica variando de [0.56] Pa para metanos a 1.8 [Pa] para gases pesados (Mavko et al., 1998).

O primeiro passo é o cálculo da temperatura absoluta do gás [$^{\circ}K$].

$$T_a = T + 273,15 \quad (3.23)$$

Onde $T[{}^{\circ}C]$ é a temperatura.

Para o cálculo das pseudo-propriedade dos gás, usamos as Equações 3.24 e 3.25 :

$$T_r = \frac{T_a}{94,72 + 170,75G} \quad (3.24)$$

$$P_r = \frac{P}{4892,0 - 0,4048G} \quad (3.25)$$

O fator de compressibilidade Z pode ser calculado como função das propriedades pseudo-reduzidas da substância, mostrado a seguir.

$$Z = aP_r + b + cd \quad (3.26)$$

$$d = \exp \left\{ - \left[0,45 + 8,0 \left(0,56 - \frac{1}{T_r^2} \right)^2 \right] \frac{P_r^{1,2}}{T_r} \right\} \quad (3.27)$$

$$c = 0,109 (3,85 - T_r)^2 \quad (3.28)$$

$$b = 0,642T_r - 0,007T_r^4 - 0,52 \quad (3.29)$$

$$a = 0,03 + 0,00527 (3,5 - T_r)^3 \quad (3.30)$$

Com isso a densidade do gás pode ser calculada, pela Equação 3.31:

$$\rho_g = \frac{28,8GP}{ZRT} \quad (3.31)$$

Onde a constante do gás para essas condições é $R = 8,31441$ [J/mol.K] .

Para calcular o módulo de incompressibilidade do gás são usadas as seguintes relações:

$$K_g = \frac{P\gamma}{1 - \frac{P_r}{Z}f} \quad (3.32)$$

$$\gamma = 0,85 + \frac{5,6P_r}{P_r + 2} + \frac{27,1}{(P_r + 3,5)^2} - 8,7e^{-0,65(P_r+1)} \quad (3.33)$$

$$f = cdm + a \quad (3.34)$$

$$m = 1, 2 \left\{ - \left[0,45 + 8,0 \left(0,56 - \frac{1}{T_r} \right)^2 \right] \frac{P_r^{0,2}}{T_r} \right\} \quad (3.35)$$

E por fim, calcula-se a velocidade acústica do gás V_g [m/s].

$$V_g = \sqrt{\frac{K_g}{\rho_g}} \quad (3.36)$$

Óleo

Óleos naturais ou óleos crus são substâncias complexas variando desde líquidos leves a muito pesados. Com isso, a densidade de óleos em condições de reservatório podem variar de 0,5 a 1 [g/cm³] sendo mais comumente encontrados óleos entre 0,7 e 0,8 [g/cm³] (Mavko et al. 1998). A densidade de um óleo estabelecida pelo *Instituto Americano Petróleo* pode ser definida como:

$$API = \frac{141,5}{\rho_o} - 131,5 \quad (3.37)$$

Onde:

- ρ_o [g/cm³] é a densidade do óleo à 15,6°C e pressão atmosférica.

Os valores de densidade API variam de aproximadamente 10 para óleos muito pesados a 60 para óleos leves. A densidade do óleo morto, ou seja, óleo sem gás dissolvido pode ser estimada por:

$$\rho_d = \frac{\rho_p}{0,972 + (3,81 \cdot 10^{-4}) (T + 17,78)^{1,175}} \quad (3.38)$$

Em que:

$$\rho_p = \rho_o + (0,00277P - 1,71 \cdot 10^{-7}P^3)(\rho_o - 1,15)^2 + 3,49 \cdot 10^{-4}P \quad (3.39)$$

Onde ρ_p [g/cm³] é a densidade do óleo subsaturado.

Rearranjando a Equação 3.37, obtém-se:

$$\rho_o = \frac{API + 131,5}{141,5} \quad (3.40)$$

Para o óleo vivo (óleo com gás dissolvido), a densidade é função da pressão, temperatura e composição do óleo (Irineu, 2008). Logo, temos as seguintes relações para o cálculo da densidade do óleo vivo ρ_l [g/cm³].

$$\rho_l = \frac{\rho_o + 0,0012GR_l}{B_l} \quad (3.41)$$

$$R_l = 2,03 [P \exp(0,02878 API - 0,00377T)]^{1,205} \quad (3.42)$$

$$B_l = 0,972 + 0,0038 \left[2,4R_g \left(\frac{G}{\rho_o} \right)^{0,5} + T + 1,78 \right]^{1,175} \quad (3.43)$$

Onde:

- $R_g[m^3/m^3]$ é a razão de solubilidade de gás em óleo
- P [MPa] é a pressão
- T [°C] é a temperatura.

Segundo Batzle & Wang, a velocidade [m/s] para óleos com gás dissolvidos será:

$$V_o = 2096,0 \left(\frac{\rho'}{2,6 - \rho'} \right)^{0,5} - 3,7T + 4,64P + 0,0038 \left[2,4R_g \left(\frac{G}{\rho_o} \right)^{0,5} + T + 1,78 \right]^{1,175} \quad (3.44)$$

Onde a pseudo-densidade $\rho'[g/cm^3]$, calcula-se por:

$$\rho' = \rho_o \frac{(1 + 0,001R_l)^{-1}}{B_l} \quad (3.45)$$

Pode-se calcular V_o [m/s] em termos de API, por:

$$V_o = 15450,0 (77,1 + API)^{-0,5} - 3,7T + 4,64P + 0,0115 \left[4,12 \left(\frac{1,08}{\rho' - 1} \right)^{0,5} \right] TP \quad (3.46)$$

Por fim, o módulo de incompressibilidade do óleo vivo K_l [MPa] é dado por:

$$K_l = V_o^2 \rho_l \quad (3.47)$$

3.2.2 Modelagem AVO (Amplitude versus offset)

A modelagem AVO se processa através da integração das propriedades sísmicas e petrofísicas da rocha, de tal forma que se obtém a variação do coeficiente de reflexão com o ângulo de incidência, o que possibilita o estudo da resposta AVO para um determinado meio.

Segundo Castagna & Backus (1993), a variação dos coeficientes de reflexão e de transmissão com o ângulo de incidência, conhecida como variação da refletividade com afastamento, é o principal objeto de estudo da análise AVO. Essa variação é uma função das propriedades sísmicas das rochas, que por sua vez são dependentes das propriedades físicas do meio (litologia, porosidade e tipo de fluido). Correlacionando alguns parâmetros sísmicos da rocha, como a velocidade da onda compressional $V_p[m/s]$ e velocidade da onda cisalhante $V_s [m/s]$, e também com a densidade $\rho[g/cm^3]$ é possível determinar o coeficiente de reflexão e transmissão de uma onda plana incidente sobre uma interface plana.

Os coeficientes são, formalmente, obtidos pela partição de amplitudes que ocorre quando uma onda plana incide sobre uma interface plana separando dois meios de parâmetros elásticos distintos. Quando a incidência da onda plana é normal à superfície plana, o coeficiente de reflexão na interface pode ser obtido através do contraste das impedâncias acústicas das camadas. A impedância acústica é calculada através da velocidade da onda compressional, pela seguinte equação:

$$Ip = V_p \rho \quad (3.48)$$

Onde $Ip [m/s.g/cm^3]$ é a impedância acústica da onda P.

O coeficiente de reflexão acústico Rp pode ser calculado quando há uma incidência normal da onda compressional, por:

$$Rp = \frac{Ip_2 - Ip_1}{Ip_2 + Ip_1} \quad (3.49)$$

Em que $Ip_1 [m/s.g/cm^3]$ é a impedância acústica da onda compressional na camada anterior e $Ip_2 [m/s.g/cm^3]$ é a impedância acústica na camada posterior.

No entanto, quando a incidência da onda plana na interface entre dois meios é oblíqua, o cálculo do coeficiente de reflexão torna-se mais complexo.

Modelo de Zoeppritz Em 1919 Zoeppritz propôs uma solução do problema do coeficiente de reflexão e de transmissão em função do ângulo de incidência e das propriedades elásticas do meio. A referida solução relaciona o ângulo de incidência e as respectivas amplitudes das ondas P e S refletidas e refratadas.

O modelo de Zoeppritz considera os seguintes parâmetros:

- θ_1° é o ângulo de incidência da onda P ;
- θ_2° é o ângulo de refração da onda P ;
- ψ_1° é o ângulo de reflexão da onda S ;
- ψ_2° é o ângulo de refração da onda S ;

- $\alpha_1 [m/s]$ é a velocidade da onda P incidente ;
- $\alpha_2 [m/s]$ é a velocidade da onda P refratada;
- β_1° é o ângulo de reflexão da onda S ;
- β_2° é o ângulo de refração da onda S .

Considerando a Lei de Snell, em que:

$$p = \frac{\sin\theta_1}{\alpha_1} \quad (3.50)$$

Onde p é o parâmetro de raio.

Para obter o coeficiente de reflexão da onda P (R_{pp}), utilizamos a equação de Zoeppritz, em função das variáveis a seguir:

$$R_{pp} = \frac{\left[\left(b \frac{\cos\theta_1}{\alpha_1} - c \frac{\cos\theta_2}{\alpha_2} \right) F - \left(a + d \frac{\cos\theta_1 \cos\Psi_2}{\alpha_1 \beta_2} \right) H p^2 \right]}{D} \quad (3.51)$$

Onde:

$$a = (1, 0 - 2, 0 \beta_2^2 p^2) - \rho (1, 0 - 2, 0 \beta_1^2 p^2) \quad (3.52)$$

$$b = (1, 0 - 2, 0 \beta_2^2 p^2) + 2, 0 \rho_1 \beta_1^2 p^2 \quad (3.53)$$

$$c = 2, 0 \rho_2 \beta_1^2 p^2 + \rho (1, 0 - 2, 0 \beta_1^2 p^2) \quad (3.54)$$

$$d = 2, 0 (\rho_2 \beta_2^2 - \rho_1 \beta_1^2) \quad (3.55)$$

$$E = b \frac{\cos\theta_1}{\alpha_1} + c \frac{\cos\theta_2}{\alpha_2} \quad (3.56)$$

$$F = b \frac{\cos\Psi_1}{\beta_1} + c \frac{\cos\Psi_2}{\beta_2} \quad (3.57)$$

$$G = a - d \frac{\cos\theta_1}{\alpha_1} \frac{\cos\Psi_2}{\beta_2} \quad (3.58)$$

$$H = a - d \frac{\cos\theta_2}{\alpha_2} \frac{\cos\Psi_1}{\beta_1} \quad (3.59)$$

$$D = EF + GH p^2 \quad (3.60)$$

Modelo de Aki and Richards Visando simplificar a equação de Zoeppritz, Aki and Richards propuseram uma solução reduzida para o ângulo incidente e o coeficiente de reflexão, para utilização em aplicações práticas das mesmas (Castagna & Backus, 1993).

A equação de Aki and Richards trata-se de uma aproximação linear da equação de Zoeppritz, como mostram nas seguintes equações:

$$Vp = \frac{(Vp_1 + Vp_2)}{2,0} \quad (3.61)$$

$$\rho = \frac{(\rho_1 + \rho_2)}{2,0} \quad (3.62)$$

$$Vs = \frac{(Vs_1 + Vs_2)}{2,0} \quad (3.63)$$

$$\Delta Vp = Vp_2 - Vp_1 \quad (3.64)$$

$$\Delta Vs = Vs_2 - Vs_1 \quad (3.65)$$

$$\Delta \rho = \rho_2 - \rho_1 \quad (3.66)$$

$$p = \frac{\sin \theta}{Vp_1} \quad (3.67)$$

$$\theta = \frac{(\theta_1 + \theta_2)}{2,0} \quad (3.68)$$

Onde:

- p é o parâmetro do raio;
- Vp_1 [m/s] é a velocidade da onda p no meio 1;
- Vs_1 [m/s] é a velocidade da onda s no meio 1;
- Vp_2 [m/s] é a velocidade da onda p no meio 1;
- Vs_s [m/s] é a velocidade da onda s no meio 1;
- ρ_1 [g/cm^3] é a densidade do meio 1;
- ρ_2 [g/cm^3] é a densidade do meio 2.

Após encontrar os valores das variáveis, utiliza-se a equação de Aki and Richards:

$$Rpp(\theta) \approx \frac{(1,0 - 4,0p^2Vs^2)\Delta\rho}{2,0\rho} + \frac{\Delta Vp}{2,0 \cos \theta^2 Vp} - 4,0p^2Vs^2 \frac{\Delta Vs}{Vs} \quad (3.69)$$

Modelo de Shuey Da equação de Aki & Richards obteve-se aproximação de Shuey, com os parâmetros: intercept R_0 e gradiente G . Para isso, utilizamos as mesmas correlações que foram utilizadas em Aki & Richards. Após obter esses valores, calculamos o intercept R_0 e o gradiente G , por meio das seguintes equações:

$$R_0 = \frac{\left(\frac{\Delta V_p}{V_p} + \frac{\Delta \rho}{\rho} \right)}{2,0} \quad (3.70)$$

$$G = \frac{\Delta V_p}{2,0 V_p} - \frac{2,0 V_s^2}{V_p^2} \left(\frac{\Delta \rho}{\rho} + \frac{2,0 \Delta V_s}{V_s} \right) \quad (3.71)$$

Assim, por último, encontramos o coeficiente de reflexão em função do ângulo incidente:

$$Rpp(\theta) = R_0 + G \sin^2 \theta \quad (3.72)$$

Interpretação dos atributos AVO Rutherford & Williams (1989) definiram três classes distintas de anomalias de AVO. A Classe I ocorre quando o intercepto AVO é fortemente positivo, acarretando um decréscimo na magnitude da amplitude versus o afastamento e possível inversão de polaridade nos afastamentos longos, visto que a impedância da camada inferior é maior que a da camada superior. Neste caso A é positivo e B é negativo sendo plotados no quadrante IV. Pode representar arenitos consolidados com hidrocarbonetos, entretanto rochas desse tipo são pouco sensíveis ao fluido, podendo levar a ambiguidade na interpretação.

Na Classe II, pode apresentar um aumento ou diminuição da amplitude com o afastamento, o intercepto AVO pode ser positivo ou negativo, pode haver mudança de sinal nos afastamentos curtos a médios e a impedância acústica dos meios é quase a mesma. Pode representar arenitos limpos com hidrocarbonetos.

Na Classe III, conhecida como AVO clássico, o coeficiente de reflexão da incidência normal é negativo, assim como o gradiente, e tem-se o aumento da magnitude da amplitude com o afastamento, sendo plotados no quadrante III. Anomalia desse tipo podem ser indicativo de arenitos inconsolidados com gás.

Alem dessas três classes, pode ocorrer ainda a Classe IV de anomalia AVO, como proposto por Castagna & Swan (1997), conforme Figura 3.1. A Classe IV tem um coeficiente de reflexão normal negativo e o gradiente positivo, e tem-se a diminuição da magnitude da amplitude com o afastamento, sendo plotada no II quadrante. A anomalia desse tipo é rara e pode ocorrer com arenitos com gás capeados por folhelhos.

Na Tabela 3.4 encontra-se o resumo do comportamento AVO para as Classes I, II, III e IV.

Tabela 3.4: Comportamento AVO para as várias classes

Classe	Impedância Relativa	Quadrante	A	B	Resposta AVO
I	Maior que a unidade superior	IV	+	-	Diminui
II	Próxima da unidade superior	II, III ou IV	+	-	Aumenta ou Diminui
III	Menor que a unidade superior	III	-	-	Aumenta
IV	Menor que a unidade superior	IV	-	+	Diminui

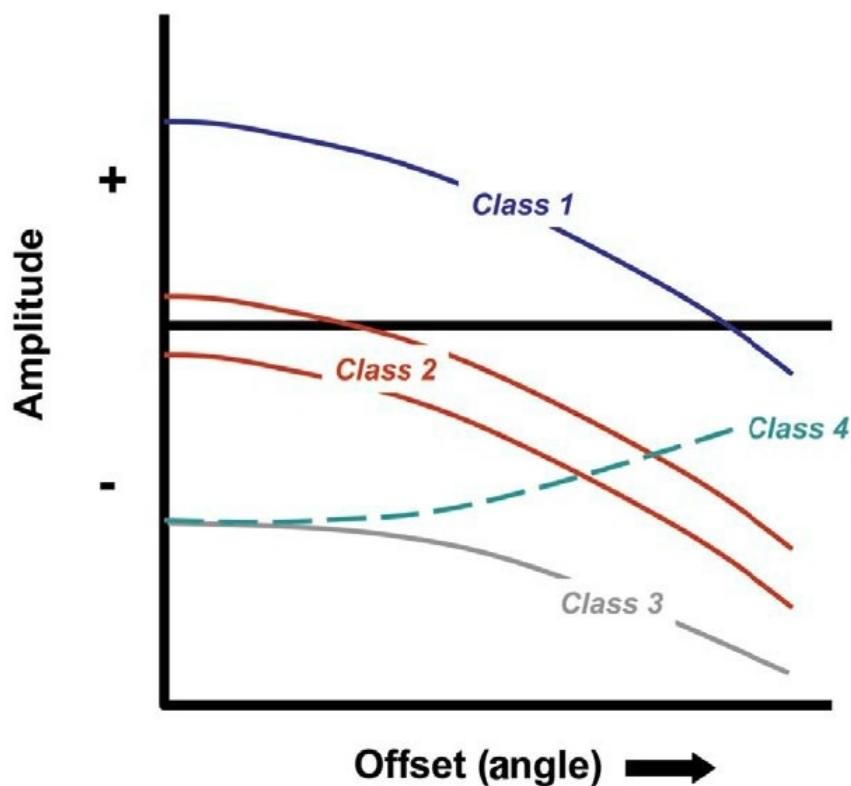


Figura 3.1: Classificação da resposta AVO

3.3 Diagrama de pacotes

Na Figura 3.2 temos o diagrama de pacotes para elaboração do software, como descrito na análise de domínio.

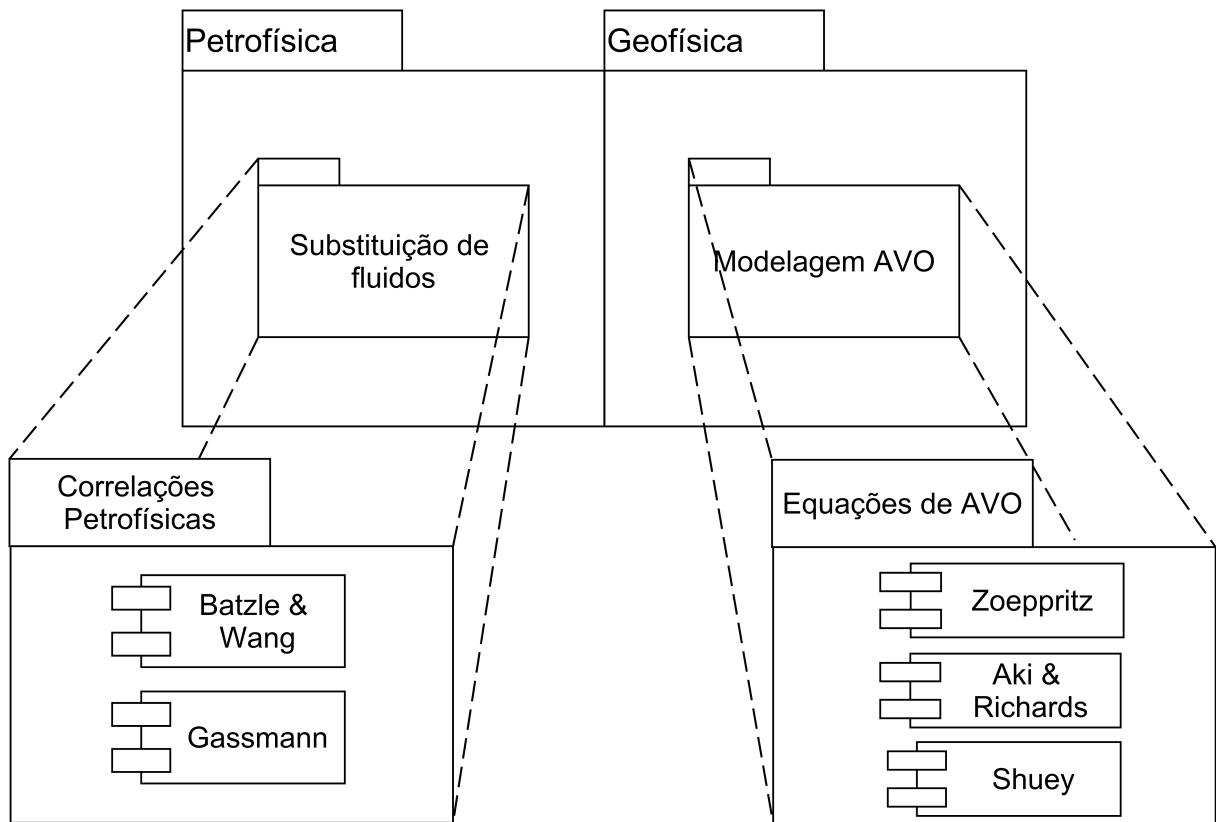


Figura 3.2: Diagrama de pacotes do simulador

Capítulo 4

AOO – Análise Orientada a Objeto

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

4.1 Diagramas de classes

O diagrama de classes é apresentado na Figura 4.1.

4.1.1 Dicionário de classes

Apresentamos nesse tópico um breve detalhamento sobre as classes e suas atividades.

- Classe CMistura: Classe que contem as propriedades da mistura de fluidos contidos no reservatório.
- Classe CAgua: Classe que contem os atributos e métodos específicos para o fluido água.
- Classe CGas: Classe que contem os atributos e métodos específicos para o fluido gás.
- Classe COleo: Classe que contem os atributos e métodos específicos para o fluido óleo.
- Classe CFluido: Classe base para os fluidos.
- Classe CRocha: Classe que contem as propriedades da matriz rochosa, em função do conteúdo mineralógico que a compõem.

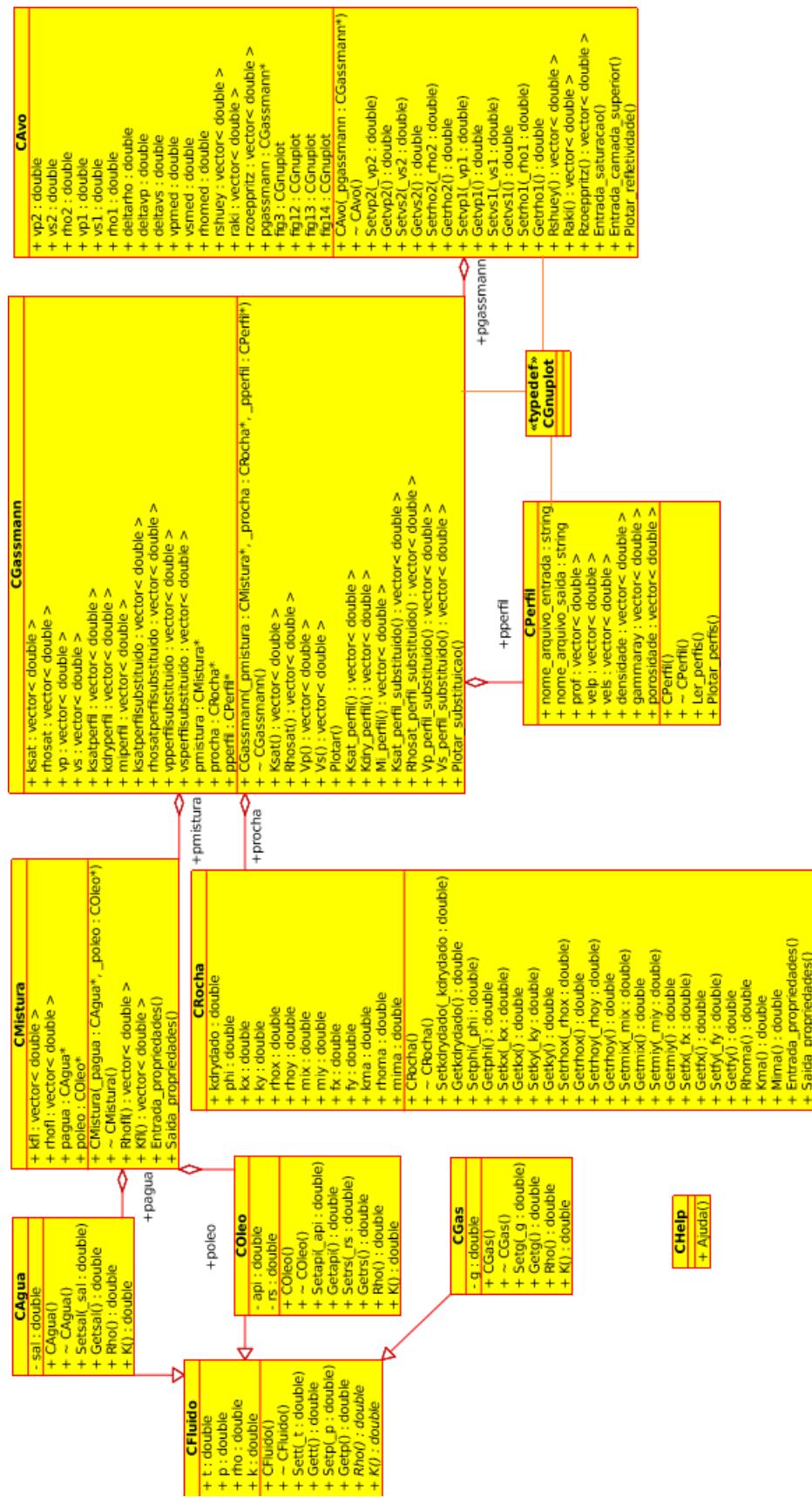


Figura 4.1: Diagrama de classes

- Classe CGassmann: Classe responsável pelos cálculos da substituição de fluidos utilizando a equação de Gassmann
- Classe CAVO: Classe responsável por realizar a modelagem AVO através das equações propostas por Zoeppritz, Aki & Richards e Shuey.
- Classe CGnuplot: Classe que possibilita a geração de gráficos usando o programa externo Gnuplot.
- Classe CPerfil: Carrega os dados de perfis.
- Classe CHelp: Classe que contém o método com informações a cerca do programa como ajuda.

4.2 Diagrama de seqüência – eventos e mensagens

As Figuras 4.2 e 4.3 apresentam os diagramas de sequência para as duas simulações propostas neste projeto. Apresentam a sequencia de eventos, em ordem temporal, da relação entre o usuário e o programa.

4.2.1 Diagrama de sequência geral

Na Figura 4.2, o usuário insere os dados no simulador, que por sua vez calcula as propriedades dos fluidos e da rocha possibilitando a substituição de fluidos pela equação de Gassmann, além de realizar a modelagem AVO. Por fim, gera os gráficos necessários que serão utilizados para interpretação geofísica.

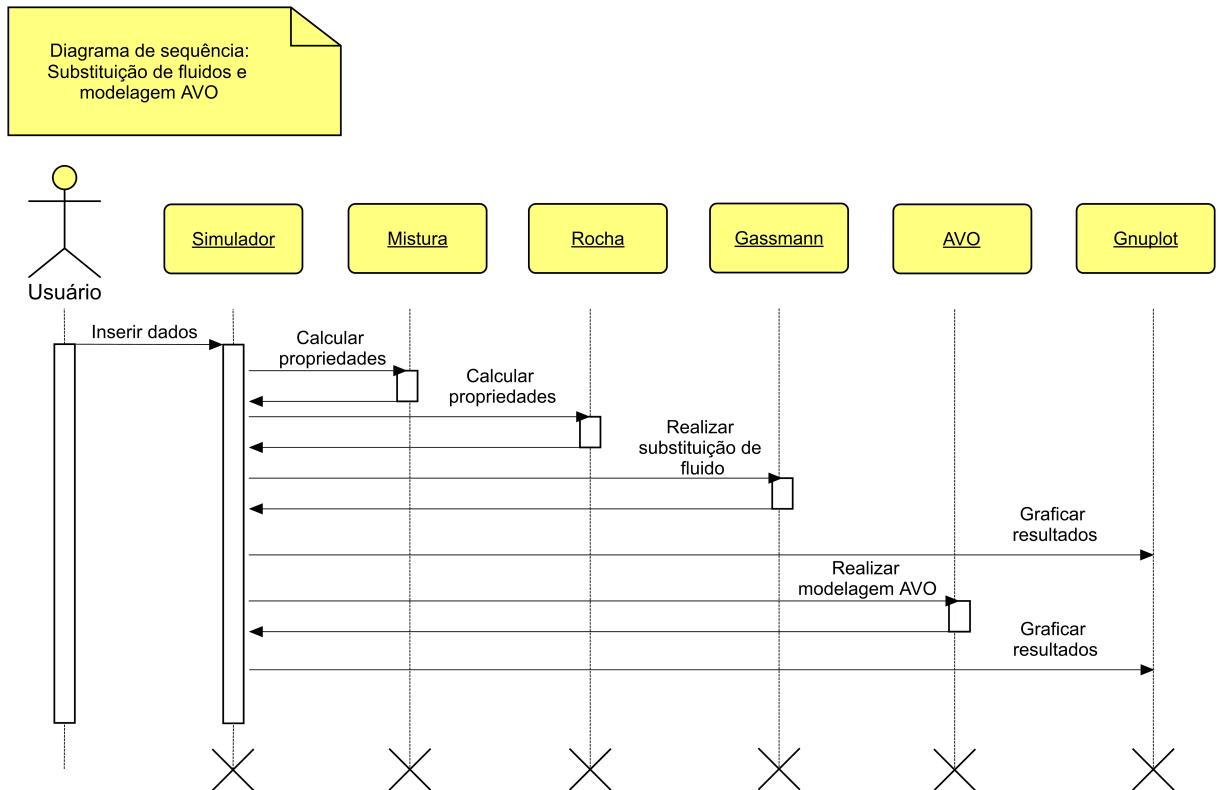


Figura 4.2: Diagrama de seqüência: Substituição de fluidos e modelagem AVO

Na Figura 4.3, arquivos de perfilagem são carregados no simulador e posteriormente plotado os perfis. As propriedades do fluido e rochas são calculadas, possibilitando a realização da substituição de fluidos.

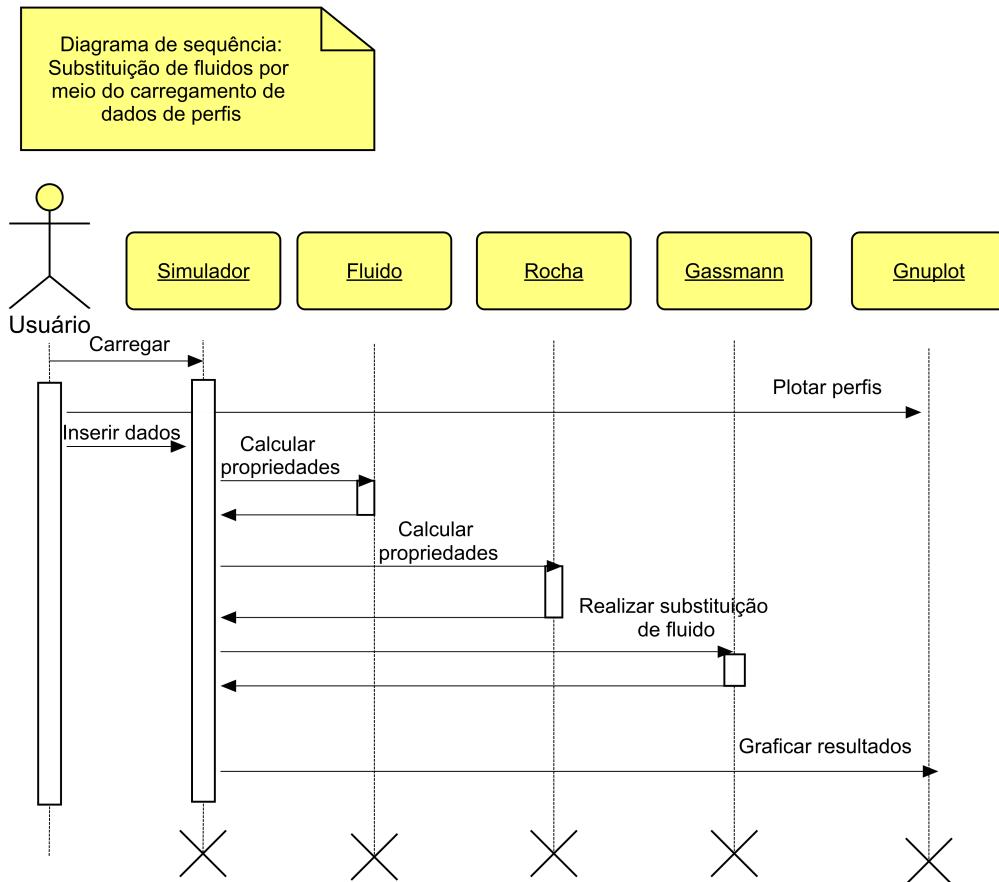


Figura 4.3: Diagrama de seqüência: Substituição de fluidos por meio do carregamento de dados de perfis

4.3 Diagrama de comunicação – colaboração

Nos diagramas de comunicação, o foco é a troca de mensagens e dados entre os objetos. Os diagramas de comunicação das Figuras 4.4 e 4.5 são baseados nos diagramas de sequência. Observe que o simulador carrega e calcula as propriedades do fluido e rocha e em seguior realiza a substituição de fluidos e modelagem AVO. Por fim, são gerados gráficos através do programa externo Gnuplot.

4 - Análise Orientada a Objeto

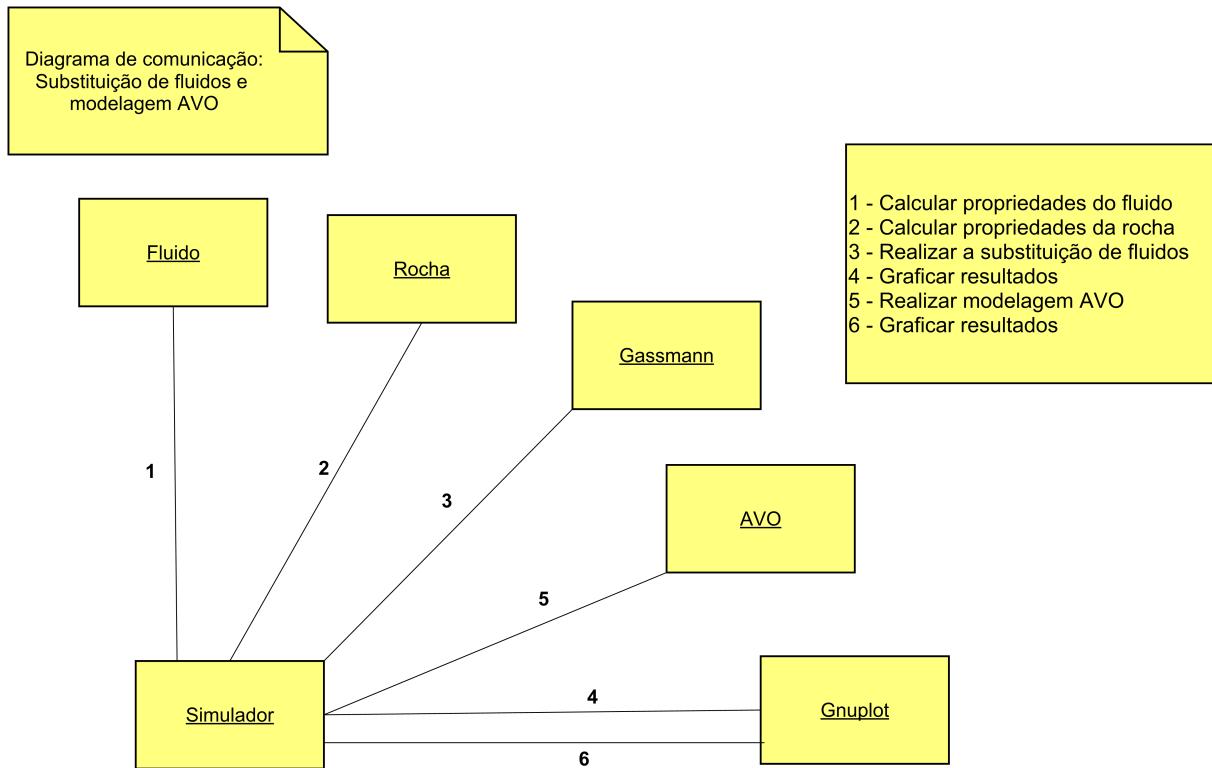


Figura 4.4: Diagrama de comunicação: Substituição de fluidos e modelagem AVO

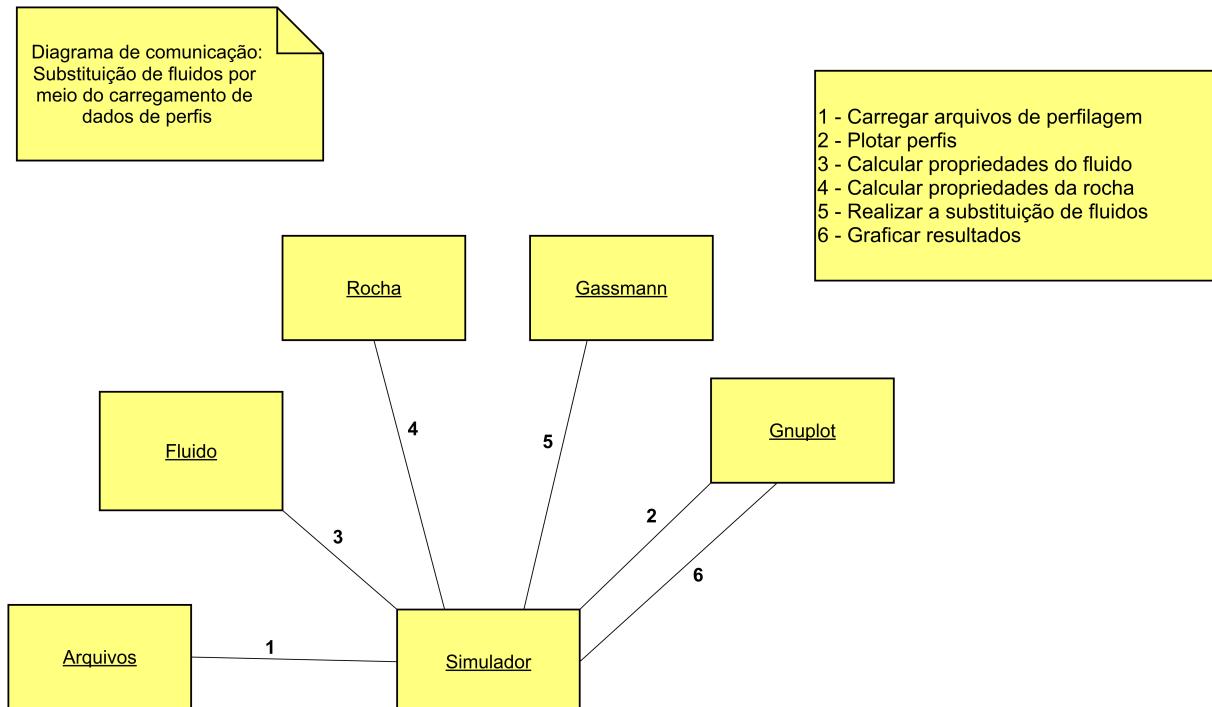


Figura 4.5: Diagrama de comunicação: Substituição de fluidos e modelagem AVO

4.4 Diagrama de máquina de estado

As Figuras 4.6 e 4.7 apresentam diagramas de máquina de estado para dois objetos representando os diversos estados que assumem durante suas vidas. São propostos modelos para a dinâmica dos objetos Gassmann e AVO.

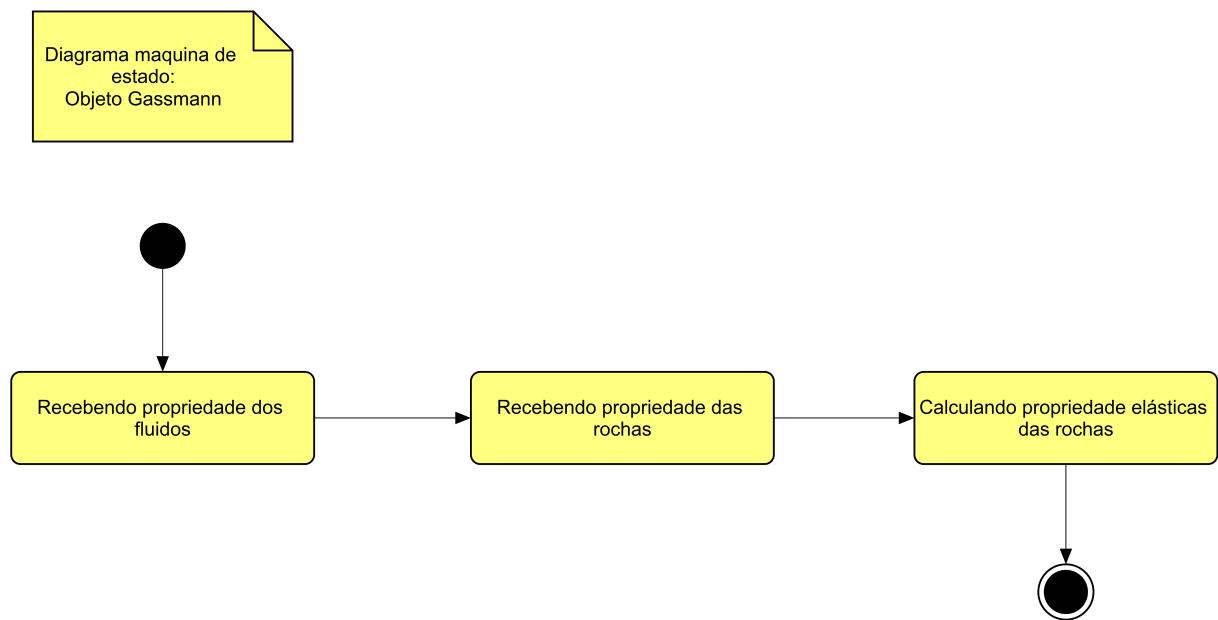


Figura 4.6: Diagrama de máquina de estado: Objeto Gassman

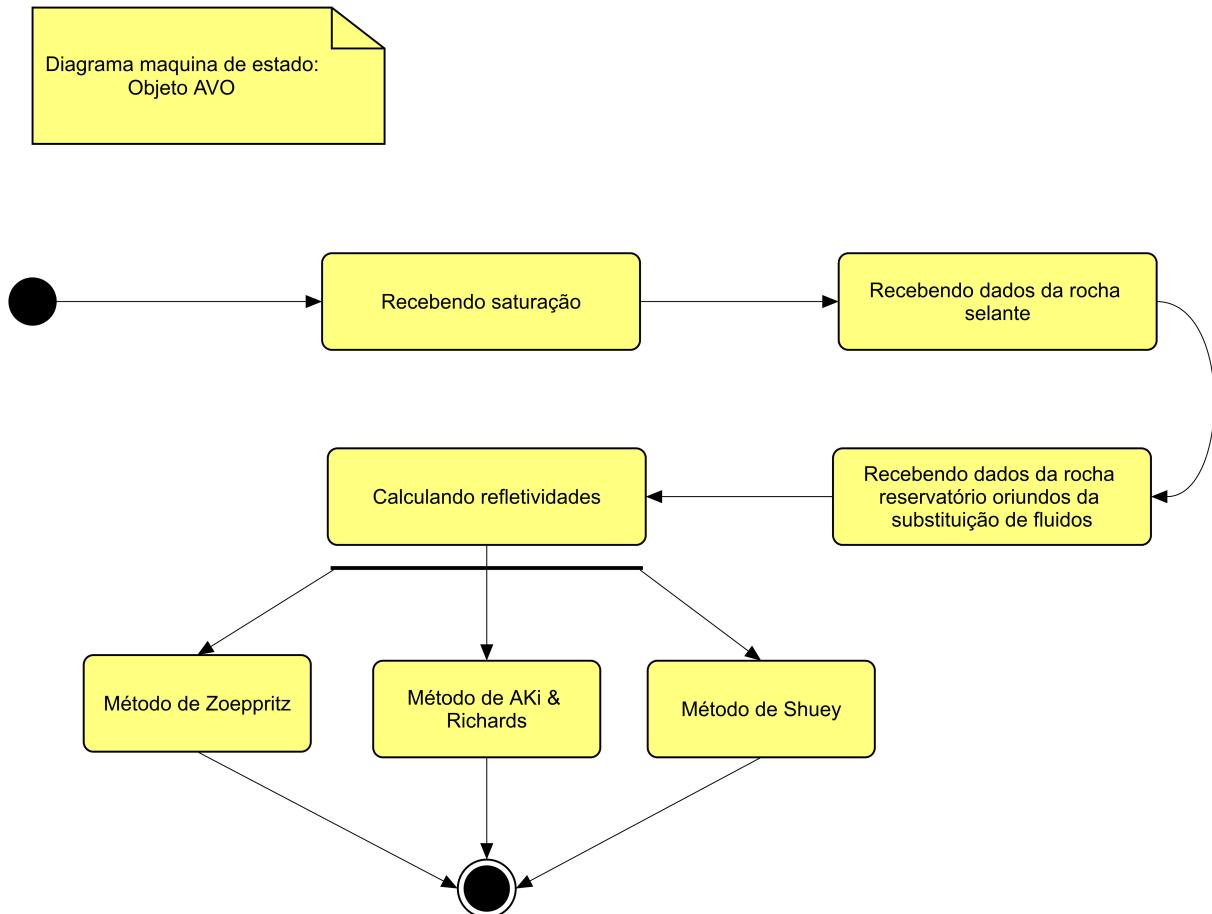


Figura 4.7: Diagrama de máquina de estado: Objeto AVO

4.5 Diagrama de atividades

No diagrama de atividades serão detalhados dois métodos apresentados no diagrama de máquina de estado. Um método é o cálculo da substituição de fluidos, utilizando a rotina proposta por Gassman, Figura 4.8 e o outro a modelagem AVO, através da equação de Shuey, observada na Figura 4.9 .

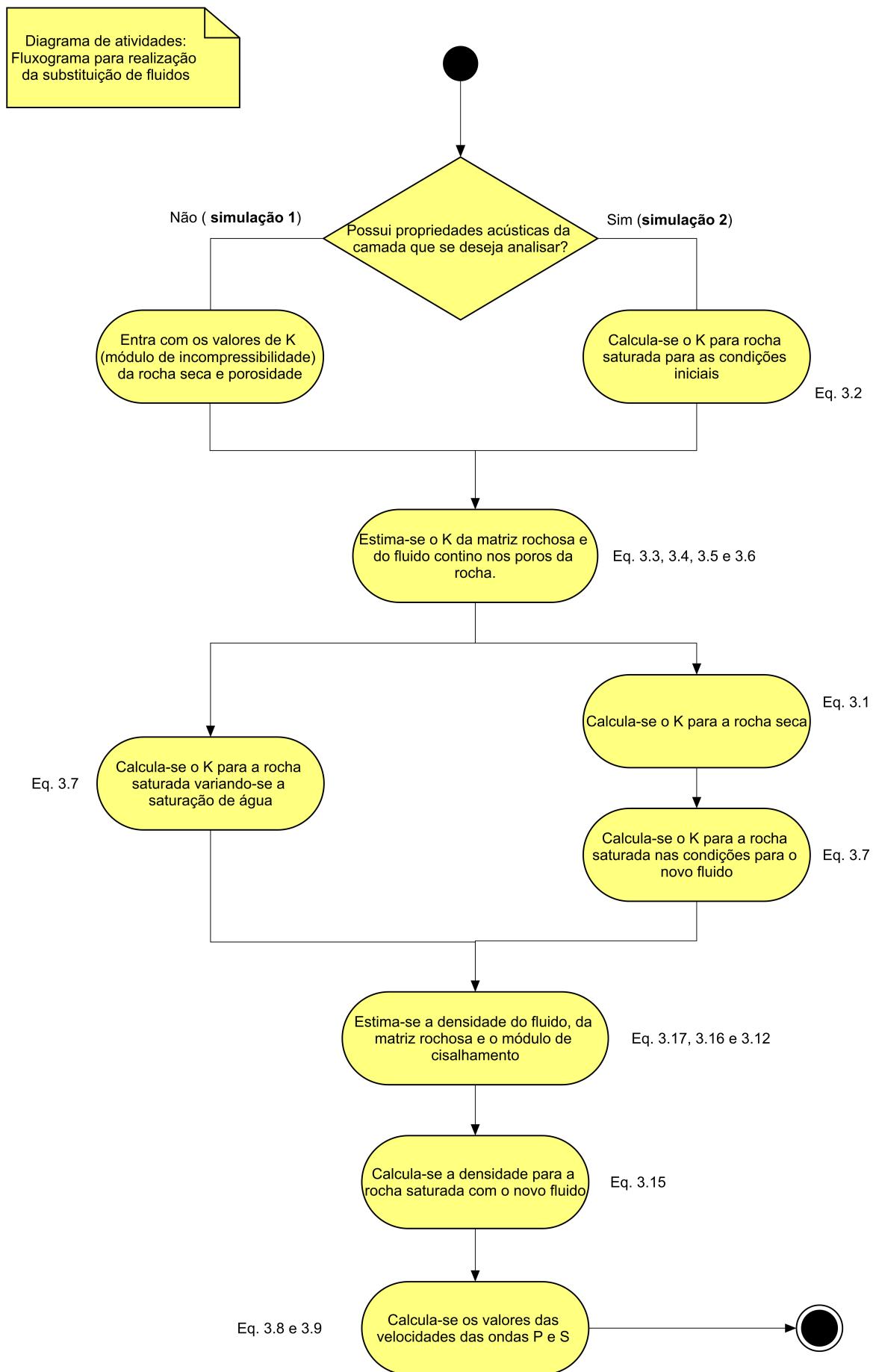


Figura 4.8: Diagrama de atividades:Fluxograma para a realização da substituição de fluidos

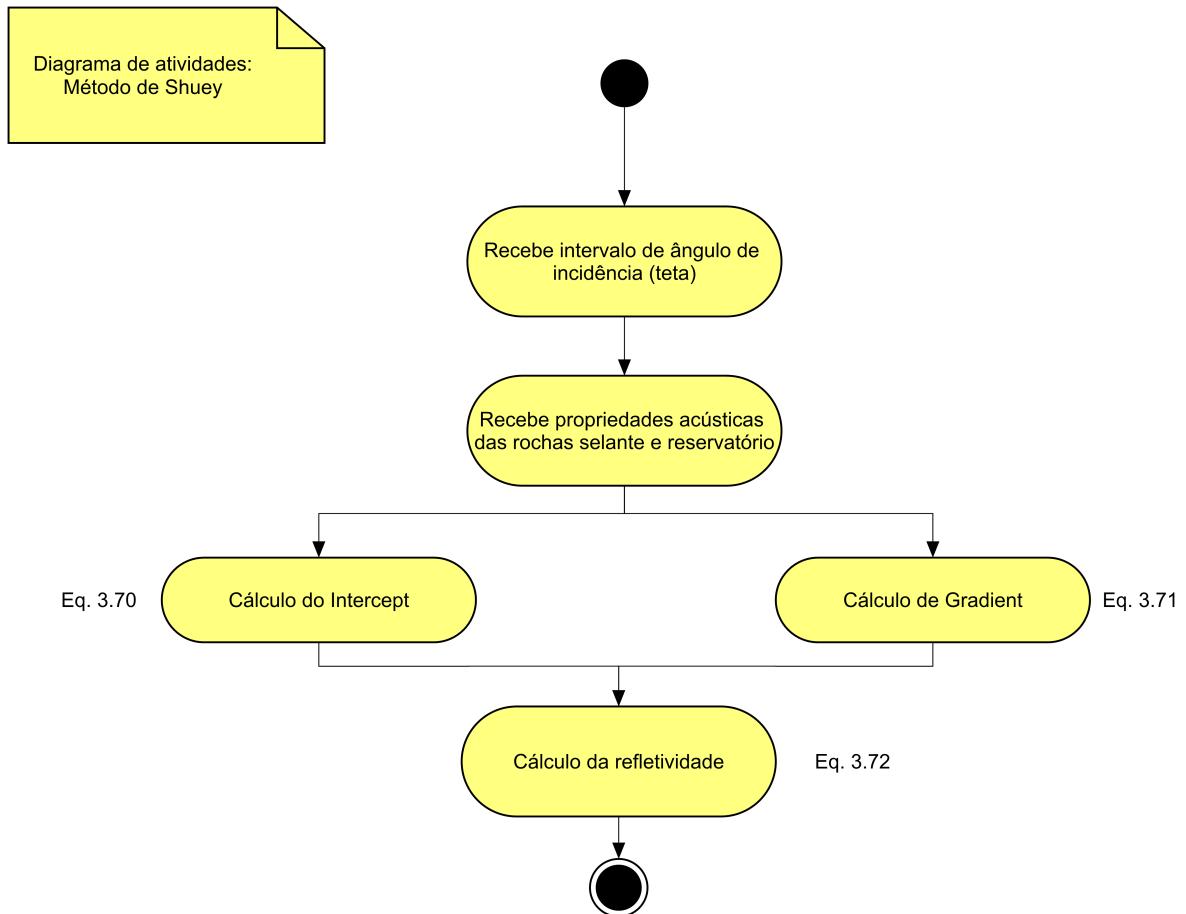


Figura 4.9: Diagrama de atividades: Método de Shuey

Capítulo 5

Projeto

Apresentamos neste capítulo questões associadas ao projeto do sistema, incluindo protocolos, recursos, plataformas suportadas, implicações nos diagramas feitos anteriormente, bibliotecas, 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 desenvolve-se o projeto do sistema, qual envolve etapas como a definição dos protocolos, da interface API, o uso de recursos, a subdivisão do sistema em subsistemas, a alocação dos subsistemas ao hardware e a seleção das estruturas de controle, a seleção das plataformas do sistema, das bibliotecas externas, dos padrões de projeto, além da tomada de decisões conceituais e políticas que formam a infraestrutura do projeto.

1. Protocolos

- O programa terá como entrada e saída arquivos de extensão .txt e perfis em arquivos de extensão .png.
- Não será necessário incluir intercomunicações entre o programa e componentes externos, uma vez que o programa oferecerá a possibilidade de salvar os resultados em um arquivo.

2. Recursos

- O presente programa precisará utilizar o HD, o processador, o teclado, a tela, o mouse, a memória e demais componentes internos do computador.
- Será necessária também a entrada de um banco de dados provenientes da geofísica com informações acerca de um poço. O arquivo de entrada deverá ter a estrutura mostrada na Figura 5.1.

3. Controle

- Não haverá necessidade de grande espaço na memória visto que o programa e seus componentes trabalham com dados relativamente pequenos.
- Neste projeto não ha necessidade de uso de processos de processamento paralelo, pois os cálculos realizados requerem pouco esforço de processamento.

4. Plataformas

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

5. Bibliotecas

- Será utilizada a biblioteca padrão da linguagem C++.
- Bibliotecas a serem utilizadas na implementação, cmath, vector, fstream.

Well2.txt (~/Documentos/MauricioRafaelBoechat) - gedit					
Arquivo	Editar	Ver	Pesquisar	Ferramentas	Documentos
Abrir	Salvar	Desfazer			
Well2.txt					
%'depth(m)'	'Vp(km/s)'	'Vs(km/s)'	'rho(gm/cc)'	'GR'	'nphi'
2183.0264	2.9061	1.5124	2.1410	63.3513	.3206
2183.1787	2.8651	1.5445	2.1525	66.8294	.3056
2183.3313	2.8755	1.5445	2.1740	74.7242	.3182
2183.4836	2.9090	1.4554	2.1812	75.0442	.3272
2183.6360	2.9153	1.4525	2.1587	69.8180	.3286
2183.7883	2.8960	1.4611	2.1401	64.2003	.3014
2183.9409	2.9244	1.5936	2.1406	61.7122	.3036
2184.0933	2.7595	1.5936	2.1430	62.5664	.2948
2184.2456	2.7185	1.4505	2.1475	63.4999	.3109
2184.3979	2.7255	1.3445	2.1438	65.3123	.2963
2184.5503	2.6564	1.2835	2.1495	68.6323	.2952
2184.7029	2.4892	1.2256	2.1335	68.8848	.2818
2184.8552	2.4561	1.2256	2.1208	74.4677	.3300
2185.0076	2.5192	1.1130	2.1003	79.5344	.3619
2185.1599	2.4990	1.0745	2.1006	83.0246	.3646
2185.3125	2.5012	.9893	2.0766	77.3263	.3156
2185.4648	2.5029	1.1188	2.0061	80.9366	.3139
2185.6172	2.6444	1.1238	1.9683	84.7465	.4072

Figura 5.1: Estrutura do arquivo de entrada

5.2 Projeto orientado a objeto – POO

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

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

Efeitos do projeto no modelo estrutural

- Adicionar nos diagramas de pacotes as bibliotecas e subsistemas selecionados no projeto do sistema.
 - Neste projeto foi utilizada como biblioteca gráfica CGnuplot
- Novas classes e associações oriundas das bibliotecas selecionadas e da linguagem escolhida devem ser acrescentadas ao modelo.
 - Neste projeto foi feito uma associação entre a biblioteca CGnuplot com CPerfil, CAvo e CGassman para a geração dos gráficos.

Efeitos do projeto no modelo dinâmico

- Revisar os diagramas de seqüência e de comunicação considerando a plataforma escolhida.
 - Não se aplica
- Verificar a necessidade de se revisar, ampliar e adicionar novos diagramas de máquinas de estado e de atividades.
 - Não se aplica

Efeitos do projeto nos métodos

- Em função da plataforma escolhida, verifique as possíveis alterações nos métodos. O projeto do sistema costuma afetar os métodos de acesso aos diversos dispositivos (exemplo: hd, rede).
 - Não se aplica

- De maneira geral os métodos devem ser divididos em dois tipos: i) tomada de decisões, métodos políticos ou de controle; devem ser claros, legíveis, flexíveis e usam polimorfismo. ii) realização de processamentos, podem ser otimizados e em alguns casos o polimorfismo deve ser evitado.
- Responda a pergunta: os métodos da classes estão dando resposta às responsabilidades da classe?
 - Neste projeto todos os métodos das classes estão dando respostas às responsabilidade da classe

5.3 Diagrama de componentes

O diagrama de componentes mostra a forma como os componentes do software se relacionam, suas dependências.

Na Figura 5.2 temos o diagragama de componentes, em que a geração dos objetos depende dos arquivos de classe de extensão .h e .cpp. O subsistema dados de entrada apresenta as propriedades dos fluidos e das rochas. O programaçao executável a ser gerado depende das bibliotecas, dos arquivos .h e .cpp e dos arquivos de entrada.

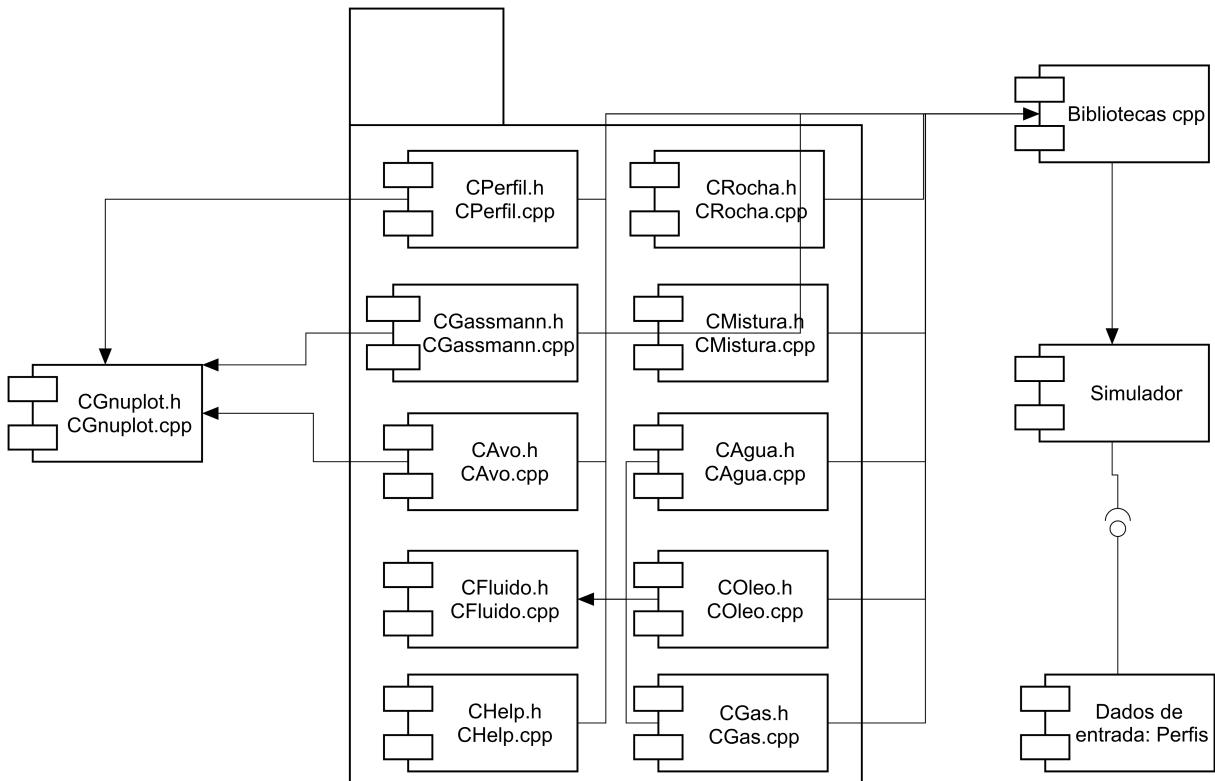


Figura 5.2: Diagrama de componentes

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.

Na Figura ?? apresenta o diagrama de implantação do sistema. Para que o sistema seja colocado em funcionamento faz-se necessário a obtenção de arquivos de poço, através da perfilagem de poços. Esses arquivos são salvos em extensão .txt e carregados pelo computador. O monitor e o mouse são necessários para que o operador e o sistema se relacione.

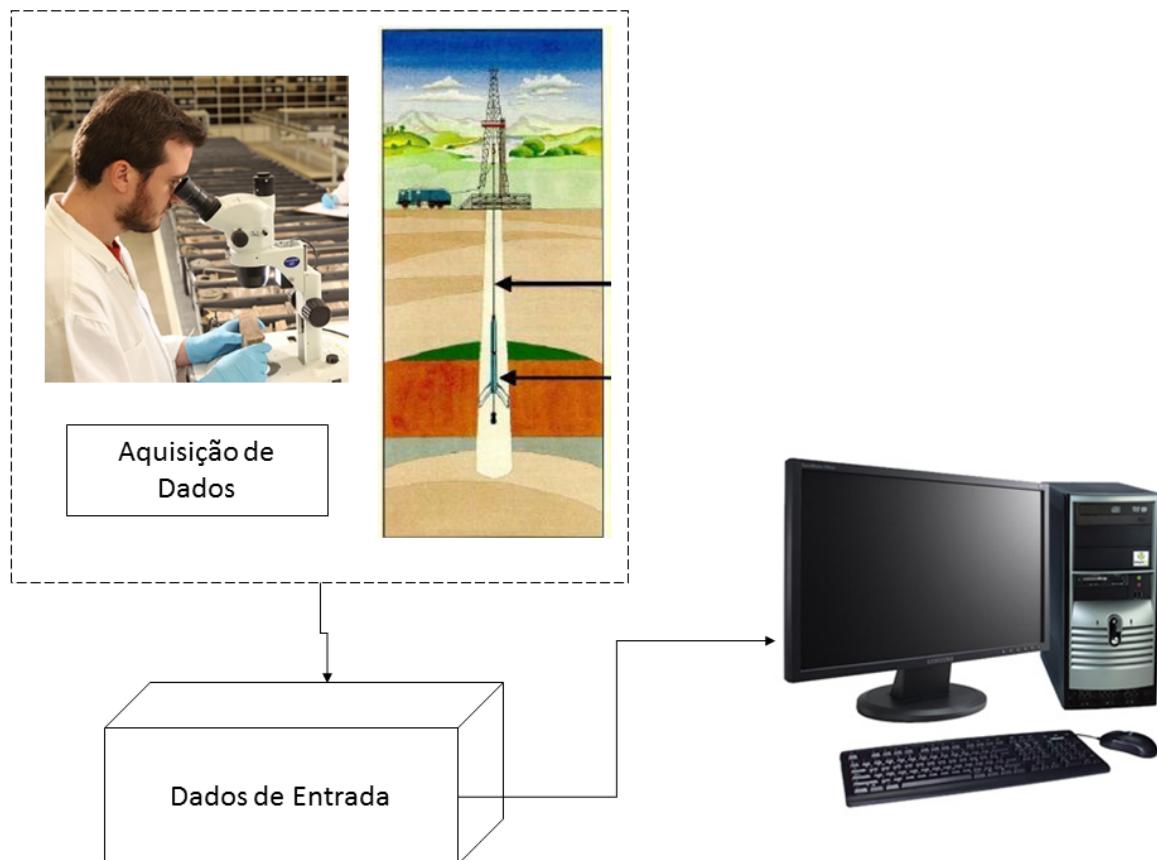


Figura 5.3: Estrutura do arquivo de entrada

Capítulo 6

Implementação

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

Nota: os códigos devem ser documentados usando padrão **javadoc**. Posteriormente usar o programa **doxygen** para gerar a documentação no formato html.

- Veja informações gerais aqui <http://www.doxygen.org/>.
- Veja exemplo aqui <http://www.stack.nl/~dimitri/doxygen/manual/docblocks.html>.

Nota: ao longo deste capítulo usamos inclusão direta de arquivos externos usando o pacote *listings* do L^AT_EX. Maiores detalhes de como a saída pode ser gerada estão disponíveis nos links abaixo.

- http://en.wikibooks.org/wiki/LaTeX/Source_Code_Listings.
- <http://mirrors.ctan.org/macros/latex/contrib/listings/listings.pdf>.

6.1 Código fonte

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

Apresenta-se na listagem seguinte o arquivo com código da classe **CAqua**.

Listing 6.1: Arquivo de cabeçalho da classe CAqua.

```
1#ifndef CAGUA_H
2#define CAGUA_H
3
4#include "CFluido.h"
5
6#include <iostream>
7
```

```

8/// Esta classe representa o fluido água e contém atributos e métodos
9// de este. Herdeira de CFLuido.
9class CAgua: public CFLuido
10
11{
12    private:
13        double sal; ///Salinidade
14
15    public:
16
17        CAgua() {}; ///Construtor
18        ~CAgua() {}; ///Destruitor
19        void Setsal(double _sal); ///Set
20        double Getsal(); ///Get
21        virtual double Rho(); ///Cálculo da densidade da fase
22            salmoura (brine)
23        virtual double K(); ///Cálculo do mód. de
24            Incompressibilidade da fase salmoura (brine)
25};

25#endif

```

Apresenta-se na listagem seguinte o arquivo de implementação da classe CAgua.

Listing 6.2: Arquivo de implementação da classe CAgua.

```

26#include "CAgua.h"
27
28#include <cmath>
29
30void CAgua::Setsal(double _sal){sal=_sal;}
31
32double CAgua::Getsal(){return sal;}
33
34double CAgua::Rho()
35{
36    double salu; ///Salinidade em unidades corrigidas
37    salu = sal/1000000.0;
38    double rhow; ///Densidade da água doce
39    double t2 = t*t; double t3 = t2*t; double t4 = t2*t2;
40    double p2 = p*p; double p3 = p*p2; double p4 = p2*p2;
41    rhow = 1.0 + 1.0e-6*(-80.0*t - 3.3*t2 + 0.00175*t3 + 489.0*p-
42        2.0*t*p + 0.016*t2*p - 1.3e-5*t3*p - 0.333*p2 - 0.002*t*p2);
43    rho = rhow + salu*( 0.668 + 0.44*salu + 1.0e-6*(300.0*p-
44        2400.0*p*salu + t*( 80.0 + 3.0*t - 3300.0*salu - 13.0*p +
45        47.0*p*salu)));
46    return rho;
47}
48
49double CAgua::K()

```

```

47     {
48         double salu; ///Salinidade em unidades corrigidas
49         salu = sal/1000000.0;
50         double vw; ///Velocidade na água doce
51         double t2 = t*t; double t3 = t2*t; double t4 = t2*t2;
52         double p2 = p*p;double p3 = p*p2;double p4 = p2*p2;
53         vw = 1402.85 + 1.524*p + 3.437e-3*p2 -1.197e-5*p3 + 4.871*t
54             -0.0111*t*p + 1.739e-4*t*p2 -1.628e-6*t*p3 - 0.04783*t2 +
55             2.747e-4*t2*p -2.135e-6*t2*p2 + 1.237e-8*t2*p3 + 1.487e-4*
56             t3 -6.503e-7*t3*p -1.455e-8*t3*p2 + 1.327e-10*t3*p3 - 2.197
57             e-7*t4 + 7.987e-10*t4*p + 5.230e-11*t4*p2 - 4.614e-11*t4*p3
58             ;
59         double vb; ///Velocidade na água salina
60         vb = vw + salu*(1170.0 - 9.6*t + 0.055*t2 - 8.5e-5*t3 + 2.6*p
61             - 0.0029*t*p - 0.0476*p2 + pow(salu,1.5)*(780.0 - 10.0*p +
62             0.16*p2) - 1820.0*pow(salu,2.0));
63         k = (pow(vb,2.0)*rho)/1000.0;
64         return k;
65     }

```

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

Listing 6.3: Arquivo de cabeçalho da classe COleo.

```

60#ifndef COLEO_H
61#define COLEO_H
62
63#include "CFluido.h"
64
65#include <iostream>
66
67/// Esta classe representa o fluido óleo e contém atributos e métodos
68/// deste. Herdeira de CFluido.
69class COleo: public CFluido
70{
71    private:
72        double api; /// Densidade em API
73        double rs; /// Razão de solubilidade
74    public:
75        COleo() {}; /// Construtor
76        ~COleo() {}; /// Destruitor
77        void Setapi(double _api); /// Set
78        double Getapi(); ///Get
79        void Setsrs(double _rs); ///Set
80        double Getsrs(); ///Get
81        virtual double Rho(); ///Cálculo da densidade da fase
82            óleo
83        virtual double K(); ///Cálculo do mód. de
84            Incompressibilidade da fase óleo

```

```

83 } ;
84
85 #endif

```

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

Listing 6.4: Arquivo de implementação da classe COleo.

```

86 #include "COleo.h"
87
88 #include <cmath>
89
90 void COleo::Setapi(double _api){api=_api;}
91
92 double COleo::Getapi(){return api;}
93
94 void COleo::Setrs(double _rs){rs=_rs;}
95
96 double COleo::Getrs(){return rs;}
97
98 double COleo::Rho() /// Considerando óleo "morto" (sem gás dissolvido)
99 {
100     double rhoostd;
101     rhoostd = 141.5/(api + 131.5);
102     double rhoop;
103     rhoop = rhoostd + (0.00277*p -1.71e-7*p*p*p*pow((rhoostd -
104         1.15),2.0) +(3.4e-4*p));
105     rho = rhoop/(0.972 + 3.81e-4*pow((t + 17.78),1.175));
106     return rho;
107 }
108
109 double COleo::K()
110 {
111     double vo; /// Velocidade na fase óleo
112     vo = 2096.0*pow((rho/(2.6 - rho)),0.5) - 3.7*t + 4.64*p +
113         0.0115*(4.12*(pow((1.08/rho - 1.0),1.2) -1.0))*t*p;
114     k = (pow(vo,2.0)*rho)/1000.0;
115     return k;
116 }

```

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

Listing 6.5: Arquivo de cabeçalho da classe CGas.

```

115 #ifndef CGAS_H
116 #define CGAS_H
117
118 #include "CFluido.h"
119
120 #include <iostream>
121

```

```

122 /// Esta classe representa o fluido gás e contém atributos e métodos
123 // deste. Herdeira de CFLuido.
124
125 {
126     private:
127         double g; //Gravidade específica do gas
128
129     public:
130
131         CGas() {}; //Construtor
132         ~CGas() {}; //Destruitor
133         void Setg(double _g); // Set
134         double Getg(); // Get
135         virtual double Rho(); //Cálculo da densidade da fase
136             óleo
137         virtual double K(); //Cálculo do mód. de
138             Incompressibilidade da fase óleo
139 };
140
141
142 #endif

```

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

Listing 6.6: Arquivo de implementação da classe CGas.

```

140 #include "CGas.h"
141
142 #include <cmath>
143
144 void CGas::Setg(double _g){g=_g;}
145
146 double CGas::Getg(){return g;}
147
148 double CGas::Rho()
149 {
150
151     double tk; ///Temperatura em Kelvin
152     tk = t + 273.17;
153
154     double pr; ///Pressão reduzida
155     pr = p/(4.892 - 0.4048*g);
156
157     double tr; ///Temperatura reduzida
158     tr = tk/(94.72 + 170.75*g);
159
160     double r=8.31441; //Constante dos gases
161
162     double a;
163     a = 0.03 + 0.00527*pow((3.5 - tr),3.0);

```

```
164
165     double b;
166     b = 0.642*tr - 0.007*pow(tr,4.0) - 0.52;
167
168     double c;
169     c = 0.109*pow((3.85 - tr),2.0);
170
171     double d;
172     d = exp(-1.0*(0.45 + 8.0*pow((0.56 - (1.0/tr)),2.0))*(pow(pr
173                 ,1.2)/tr));
174
175     double z; //Fator de Incompressibilidade
176     z = a*pr + b + c*d;
177
178     rho = (28.8*g*p)/(z*r*tk);
179
180     return rho;
181 }
182
183 double CGas::K() //Correlação apresentando erros
184 {
185
186     double pr; //Pressão reduzida
187     pr = p/(4.892 - 0.4048*g);
188
189     double gamma;
190     gamma = 0.85 + (5.6/(pr + 2.0)) + (27.1/pow((pr + 3.5),2.0) -
191                 8.7*exp(-0.65*(pr +1.0)));
192
193     double tk; //Temperatura em Kelvin
194     tk = t + 273.17;
195
196     double tr; //temperatura reduzida
197     tr = tk/(94.72 + 170.75*g);
198
199     double m;
200     m = 1.2*(-1*(0.45 + 8.0*pow((0.56 - (1.0/tr)),2.0)*(pow(pr,0.2)/
201                 tr)));
202
203     double c;
204     c = 0.109*pow((3.85 - tr),2.0);
205
206     double d;
207     d = exp(-1.0*(0.45 + 8.0*pow((0.56 - (1.0/tr)),2.0))*(pow(pr
208                 ,1.2)/tr));
```

```

208     double b;
209     b = 0.642*tr - 0.007*pow(tr,4.0) - 0.52;
210
211     double a;
212     a = 0.03 + 0.00527*pow((3.5 - tr),3.0);
213
214     double z; //Fator de Incompressibilidade
215     z = a*pr + b + c*d;
216
217     double f;
218     f = c*d*m + a;
219
220     double kg;
221     k = (p*gamma)/(1.0 - pr/(z*f));
222
223     return k;
224 }
```

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

Listing 6.7: Arquivo de cabeçalho da classe CFluido.

```

226#ifndef CFLUIDO_H
227#define CFLUIDO_H
228
229#include <iostream>
230
231///Esta classe representa um fluido e é base para os fluidos utilizados
232///no simulador.
233class CFluido
234{
235    public:
236        double t; ///Temperatura
237        double p; ///Pressão
238        double rho; ///Densidade do fluido
239        double k; ///Módulo de incompressibilidade do fluido
240
241    public:
242        CFluido() {};//Construtor
243        ~CFluido() {};//Destruitor
244        void Sett(double _t); ///Set
245        double Gett(); ///Get
246        void Setp(double _p); ///Set
247        double Getp(); ///Get
248        virtual double Rho() = 0; ///Cálculo da densidade
249        virtual double K() = 0; ///Cálculo do mód. de
250        Incompressibilidade
251};
```

```
252#endif
```

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

Listing 6.8: Arquivo de implementação da classe CFluido.

```
253#include "CFluido.h"
254
255#include <cmath>
256
257void CFluido::Sett(double _t){t=_t;}
258
259double CFluido::Gett(){return t;}
260
261void CFluido::Setp(double _p){p=_p;}
262
263double CFluido::Getp(){return p;}
```

Apresenta-se na listagem seguinte o arquivo com código da classe CMistura.

Listing 6.9: Arquivo de cabeçalho da classe CMistura.

```
264#ifndef CMISTURA_H
265#define CMISTURA_H
266
267#include "CAgua.h"
268#include "COleo.h"
269
270#include <iostream>
271#include <vector>
272#include <fstream>
273
274using namespace std;
275
276///Classe que representa a mistura de fluidos, chamado fluido efetivo e
277// contém seus atributos e métodos
277class CMistura
278
279{
280    public:
281        vector <double> kfl; ///Vetor com os valores de módulo
282                                // de incompressibilidade para a mistura de fluidos
282        vector <double> rhofl; ///Vetor com os valores de
283                                // densidade para a mistura de fluidos
283        CAqua* pagua; ///Cria ponteiro para objeto da classe
284                                // CAqua
284        COleo* poleo; ///Cria ponteiro para objeto da classe
285                                // COleo
285
286    public:
```

```

288         CMistura(CAgua* _pagua, COleo* _poleo): pagua(_pagua),
289             poleo(_poleo) {}; ///Construtor
290         ~CMistura() {}; ///Destruitor
291         vector <double> Rhofl(); ///Cálculo da densidade para
292             mistura de fluidos
293         vector <double> Kfl(); ///Cálculo do módulo de
294             incompressibilidade para a mistura de fluidos
295         void Entrada_propriedades(); ///Utilizado para pedir o
296             usuário as propriedades dos fluidos
297         void Saída_propriedades(); ///Utilizado para escrever em
298             arquivo externo as propriedades fornecidas e
299             calculadas
300     };
301
302 #endif

```

Apresenta-se na listagem seguinte o arquivo de implementação da classe CMistura.

Listing 6.10: Arquivo de implementação da classe CMistura.

```

297 #include "CMistura.h"
298
299 #include <cmath>
300 #include <vector>
301 #include <fstream>
302
303 using namespace std;
304
305 vector <double> CMistura::Rhofl()
306 {
307     rhofl.resize(101);
308
309     for(int i = 0; i <= 100; i++)
310     {
311         rhofl[i] = ((i/100.0) * pagua->rho + ((1.0-(i/100.0))*poleo->
312             rho));
313     }
314
315     return rhofl;
316 }
317
318 vector <double> CMistura::Kfl()
319 {
320
321     kfl.resize(101);
322     for(int i = 0; i <= 100; i++)
323     {

```

```
326         kfl[i] = (1.0/(((i/100.0)/pagua->k)+(1.0-(i/100.0))/poleo->k))
327         ;
328     }
329     return kfl;
330 }
331
332 void CMistura::Entrada_propriedades()
333 {
334     double _t;
335     cout << "Entre com o valor da temperatura inicial em graus Celsius
336             : ";
337     cin >> _t;
338     cout << endl;
339     pagua->Sett(_t);
340     poleo->Sett(_t);
341
342     double _p;
343     cout << "Entre com o valor da pressão de poros em MPa: ";
344     cin >> _p;
345     cout << endl;
346     pagua->Setp(_p);
347     poleo->Setp(_p);
348
349
350     double _sal;
351     cout << "Entre com o valor da salinidade da água em ppm: ";
352     cin >> _sal;
353     cout << endl;
354     pagua->Setsal(_sal);
355
356
357     double _api;
358     cout << "Entre com o valor da densidade em API do óleo: ";
359     cin >> _api;
360     cout << endl;
361     poleo->Setapi(_api);
362 }
363
364
365 void CMistura::Saida_propriedades()
366 {
367     string nome_arquivo_saida;
368     nome_arquivo_saida = "Resultados_fluidos.txt";
369     ofstream fout(nome_arquivo_saida.c_str());
370     cin.get();
```

```

372
373     fout << "PROPRIEDADES DA ÁGUA (SALMOURA)" << "\n";
374     fout << "Pressão:" << pagua->Getp() << " MPa" << "\n";
375     fout << "Temperatura:" << pagua->Gett() << " graus Celsius" <<
376         "\n";
377     fout << "Salinidade:" << pagua->Getsal() << " ppm" << "\n";
378     fout << "Densidade:" << pagua->Rho() << " ug/cm3" << "\n";
379     fout << "Módulo de incompressibilidade:" << pagua->K() << "
380         MPa" << "\n\n";
381
382     fout << "PROPRIEDADES DO ÓLEO" << "\n";
383     fout << "Pressão:" << poleo->Getp() << " MPa" << "\n";
384     fout << "Temperatura:" << poleo->Gett() << " graus Celsius" <<
385         "\n";
386     fout << "Densidade API:" << poleo->Getapi() << " API" << "\n";
387     fout << "Densidade:" << poleo->Rho() << " ug/cm3" << "\n";
388     fout << "Módulo de incompressibilidade:" << poleo->K() << "
389         MPa" << "\n\n";
390
391     Rhofl();
392     Kfl();
393
394     fout << "PARA O FLUIDO EFETIVO (ÓLEO + ÁGUA)" << "\n\n";
395     fout << "DENSIDADE" << "\n";
396
397     for(int i = 0; i <= 100; i++)
398     {
399         fout << "Sw=" << i << "%" << rhofl[i] << "\n";
400     }
401
402     fout << "\nMÓDULO DE INCOMPRESSIBILIDADE" << "\n";
403
404     for(int i = 0; i <= 100; i++)
405     {
406         fout << "Sw=" << i << "%" << kfl[i] << "\n";
407     }
408
409     cout << "Os dados inicializados e calculados para os fluidos
410         foram salvos no arquivo externo" << nome_arquivo_saida << "
411         .Aperte ENTER para continuar." << "\n";
412     cin.get();
413 }
```

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

Listing 6.11: Arquivo de cabeçalho da classe CRocha.

```

410#ifndef CROCHA_H
411#define CROCHA_H
412
```

```
413 #include <iostream>
414 #include <fstream>
415
416 ///Classe que representa a rocha, com seus atributos e métodos
417 class CRocha
418
419 {
420     public:
421
422         double kdrydado; ///Módulo de incompressibilidade
423                         fornecido
424         double phi; ///Porosidade
425         double kx; ///Módulo de incompressibilidade do primeiro
426                         componente
427         double ky; ///Módulo de incompressibilidade do segundo
428                         componente
429         double rhox; ///Densidade do primeiro componente
430         double rhoy; ///Densidade do segundo componente
431         double mix; ///Módulo de cisalhamento do primeiro
432                         componente
433         double miy; ///Módulo de cisalhamento do segundo
434                         componente
435         double fx; ///Fração do primeiro componente na rocha
436         double fy; ///Fração do segundo componente na rocha
437         double kma; ///Módulo de incompressibilidade da matriz
438                         considerando os dois componentes
439         double rhoma; ///Densidade da matriz considerando os
440                         dois componentes
441         double mima; ///Módulo de cisalhamento da matriz
442                         considerando os dois componentes
443
444     public:
445
446         CRocha() {}; ///Construtor
447         ~CRocha() {}; ///Destruitor
448         void Setkdrydado(double _kdrydado);
449         double Getkdrydado(); ///Get
450         void Setphi(double _phi); ///Set
451         double Getphi(); ///Get
452         void Setkx(double _kx); ///Set
453         double Getkx(); ///Get
454         void Setky(double _ky); ///Set
455         double Getky(); ///Get
456         void Setrhoux(double _rhox); ///Set
457         double Getrhoux(); ///Get
458         void Setrhoy(double _rhoy); ///Set
459         double Getrhoy(); ///Get
460         void Setmix(double _mix); ///Set
```

```

453     double Getmix(); ///Get
454     void Setmiy(double _miy); ///Set
455     double Getmiy(); ///Get
456     void Setfx(double _fx); ///Set
457     double Getfx(); ///Get
458     void Setfy(double _fy); ///Set
459     double Getfy(); ///Get
460     double Rhoma(); ///Cálculo da densidade para a matriz da
461     rocha
462     double Kma(); ///Cálculo do módulo de
463     incompressibilidade para a matriz da rocha
464     double Mima(); ///Cálculo do módulo de cisalhamento para
465     a matriz da rocha
466     void Entrada_propriedades(); ///Utilizado para pedir ao
467     usuário as propriedades da rocha e seus componentes
468     void Saida_propriedades(); ///Utilizado para escrever em
469     arquivo externo as propriedades fornecidas e
470     calculadas
471 };
472
473 #endif

```

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

Listing 6.12: Arquivo de implementação da classe CRocha.

```

468 #include "CRocha.h"
469
470 #include <cmath>
471 #include <fstream>
472
473 using namespace std;
474
475 void CRocha::Setkdrydado(double _kdrydado){kdrydado=_kdrydado;}
476
477 double CRocha::Getkdrydado(){return kdrydado;}
478
479 void CRocha::Setphi(double _phi){phi=_phi;}
480
481 double CRocha::Getphi(){return phi;}
482
483 void CRocha::Setkx(double _kx){kx=_kx;}
484
485 double CRocha::Getkx(){return kx;}
486
487 void CRocha::Setrhou(double _rhox){rhox=_rhox;}
488
489 double CRocha::Getrhou(){return rhox;}
490
491 void CRocha::Setmix(double _mix){mix=_mix;}

```

```
492
493 double CRocha::Getmix(){return mix;}
494
495 void CRocha::Setfx(double _fx){fx=_fx;}
496
497 double CRocha::Getfx(){return fx;}
498
499 void CRocha::Setky(double _ky){ky=_ky;}
500
501 double CRocha::Getky(){return ky;}
502
503 void CRocha::Setrhoy(double _rhoy){rhoy=_rhoy;}
504
505 double CRocha::Getrhoy(){return rhoy;}
506
507 void CRocha::Setmiy(double _miy){miy=_miy;}
508
509 double CRocha::Getmiy(){return miy;}
510
511 void CRocha::Setfy(double _fy){fy=_fy;}
512
513 double CRocha::Getfy(){return fy;}
514
515 double CRocha::Rhoma()
516 {
517     rhoma = fx*rhox + fy*rhoy;
518     return rhoma;
519 }
520
521 double CRocha::Kma()
522 {
523     /// Utiliza-se a correlação de Voigh-Reuss-Hills
524     double kvoigh;
525     kvoigh = fx*kx + fy*ky;
526
527     double kreuss;
528     kreuss = 1.0/ (fx/kx + fy/ky);
529
530     kma = 0.5* (kvoigh + kreuss);
531     return kma;
532 }
533
534 double CRocha::Mima()
535 {
536     /// Utiliza-se a correlação de Voigh-Reuss-Hills
537     double mivoigh;
538     mivoigh = fx*(1.0-phi)*kx + fy*(1.0-phi)*ky;
539 }
```

```

540         double mireuss;
541         mireuss = 1.0/ ((fx*(1.0-phi))/kx + (fy*(1.0-phi))/ky);
542
543         mima = 0.5* (mivoigh + mireuss);
544         return mima;
545     }
546
547 void CRocha::Entrada_propriedades()
548 {
549     cout << "Entre com o valor da porosidade média da camada em "
550         "fração: ";
551     cin >> phi;
552     cout << endl;
553
554     cout << "Nessa simulação de teste a matriz será modelada "
555         "considerando ser composta por 2 componentes: " << "\n";
556
557     cout << "A tabela com as propriedades de minerais constituintes "
558         "das rochas mais comuns é apresentada abaixo como auxílio para "
559         "a entrada dos dados subsequentes" << "\n\n";
560     cout << "Adaptado da TABELA 3.1" << "\n";
561     cout << "MINERAL" << " " << "K(MPa)" << " " <<
562         "Mi(Mpa)" << " " << "Rho(g/cm3)" << "\n";
563     cout << "Quartzo" << " " << "37000" << " " <<
564         "44000" << " " << "2.65" << "\n";
565     cout << "Feldspato" << " " << "37500" << " " <<
566         "15000" << " " << "2.62" << "\n";
567     cout << "Plagioclásio" << " " << "75600" << " " <<
568         "25600" << " " << "2.63" << "\n";
569     cout << "Argilominerais" << " " << "30000" << " " <<
570         "8200" << " " << "2.80" << "\n";
571     cout << "Calcita" << " " << "76800" << " " <<
572         "32000" << " " << "2.71" << "\n";
573     cout << "Dolomita" << " " << "94900" << " " <<
574         "45000" << " " << "2.87" << "\n";
575     cout << "Anidrita" << " " << "44800" << " " <<
576         "29100" << " " << "2.98" << "\n";
577     cout << "Pirita" << " " << "147400" << " " <<
578         "135500" << " " << "4.93" << "\n";
579     cout << "Hematita" << " " << "100200" << " " <<
580         "95200" << " " << "5.24" << "\n\n";
581
582     cout << "Entre com o valor do módulo de incompressibilidade do "
583         "primeiro componente da matriz da rocha em Mpa: ";
584     cin >> kx;
585     cout << endl;
586
587     cout << "Entre com o valor da densidade do primeiro componente "

```

```
      da matriz da rocha em g/cm3: ";
573  cin >> rhox;
574  cout << endl;
575
576  cout << "Entre com o valor do modulo de cisalhamento do primeiro
      componente da matriz da rocha em MPa: ";
577  cin >> mix;
578  cout << endl;
579
580  cout << "Entre com o valor do modulo de incompressibilidade do
      segundo componente da matriz da rocha em MPa: ";
581  cin >> ky;
582  cout << endl;
583
584  cout << "Entre com o valor da densidade do segundo componente da
      matriz da rocha em g/cm3: ";
585  cin >> rhoy;
586  cout << endl;
587
588  cout << "Entre com o valor do modulo de cisalhamento do segundo
      componente da matriz da rocha em MPa: ";
589  cin >> miy;
590  cout << endl;
591
592  cout << "Entre com a fração do primeiro componente na rocha: ";
593  cin >> fx;
594  cout << endl;
595
596  cout << "Entre com a fração do segundo componente na rocha: ";
597  cin >> fy;
598  cout << endl;
599
600  double ftotal;
601  ftotal = fx + fy;
602  if( ftotal != 1)
603  {
604      cout << "Você entrou com frações dos componentes da rocha
          somadas diferentes de 1, ou que é fisicamente irreal nesta
          modelagem, Sugerimos reiniciar a simulação." << "\n\n";
605  }
606
607  cout << "Entre com o valor do modulo de incompressibilidade da
      rocha seca em MPa: ";
608  cin >> kdrydado;
609  cout << endl;
610  }
611
612 void CRocha::Saida_propriedades()
```

```

613     {
614         string nome_arquivo_saida;
615         nome_arquivo_saida = "Resultados_rocha.txt";
616         ofstream fout(nome_arquivo_saida.c_str());
617         cin.get();
618
619         fout << "Porosidade:" << phi << "\n";
620         fout << "Módulo de incompressibilidade da rocha seca:" <<
621             kdrydado << "\n\n";
622
623         fout << "PROPRIEDADES DO PRIMEIRO COMPONENTE" << "\n";
624         fout << "Módulo de incompressibilidade:" << kx << " MPa" << "\n";
625         fout << "Densidade:" << rhox << " ug/cm3" << "\n";
626         fout << "Módulo de cisalhamento:" << mix << " MPa" << "\n";
627         fout << "Fração de volume na rocha:" << fx << endl << "\n";
628
629         fout << "PROPRIEDADES DO SEGUNDO COMPONENTE" << "\n\n";
630         fout << "Módulo de incompressibilidade:" << ky << " MPa" << "\n";
631         fout << "Densidade:" << rhoy << " ug/cm3" << "\n";
632         fout << "Módulo de cisalhamento:" << miy << " MPa" << "\n";
633         fout << "Fração de volume na rocha:" << fy << "\n\n";
634
635         fout << "PROPRIEDADES DA MATRIZ DA ROCHA (CONSIDERANDO OS DOIS
636             COMPONENTES)" << "\n";
637         fout << "Módulo de incompressibilidade:" << Kma() << " MPa" <<
638             "\n";
639         fout << "Densidade:" << Rhoma() << " ug/cm3" << "\n";
640         fout << "Módulo de cisalhamento:" << Mima() << " MPa" << "\n";
641
642         cout << "Os dados inicializados e calculados foram salvos no
643             arquivo externo" << nome_arquivo_saida << ". Aperte ENTER
644             para continuar." << "\n";
645
646         cin.get();
647     }

```

Apresenta-se na listagem seguinte o arquivo com código da classe CPerfil.

Listing 6.13: Arquivo de cabeçalho da classe CPerfil.

```

647#ifndef CPERFIL_H
648#define CPERFIL_H
649
650#include <iostream>
651#include <vector>
652
653#include "CGnuplot.h"

```

```

654
655 using namespace std;
656
657 /// Classe com atributos e métodos para trabalhar com os perfis
658 class CPerfil
659
660 {
661     public:
662
663         string nome_arquivo_entrada; ///Armazena o nome do
664             arquivo com os perfis
665         string nome_arquivo_saida;
666         vector <double> prof; ///Vetor com os valores de
667             porosidade obtido a partir dos perfis
668         vector <double> velp; ///Vetor com os valores de
669             velocidade compressional obtido a partir dos perfis
670         vector <double> vels; ///Vetor com os valores de
671             velocidade cisalhante obtido a partir dos perfis
672         vector <double> densidade; ///Vetor com os valores de
673             densidade obtido a partir dos perfis
674         vector <double> gammaray; ///Vetor com os valores de
675             raios gama obtido a partir dos perfis
676         vector <double> porosidade; ///Vetor com os valores de
677             porosidade obtido a partir dos perfis
678 };
679
680 #endif

```

Apresenta-se na listagem seguinte o arquivo de implementação da classe CPerfil.

Listing 6.14: Arquivo de implementação da classe CPerfil.

```

681 #include "CPerfil.h"
682 #include "CGnuplot.h"
683
684 #include <vector>
685 #include <cmath>
686 #include <fstream>
687
688 using namespace std;
689
690 void CPerfil::Ler_perfis()

```

```
691
692     {
693         double prof_, velp_, vels_, densidade_, gammaray_, porosidade_
694             ;
695
696         string s;
697         cout<<"Informe o nome do arquivo de entrada (com a extensão): "
698             <<"\n";
699         cin>>nome_arquivo_entrada;
700         cin.get();
701
702         ifstream fin(nome_arquivo_entrada.c_str());
703
704         while(!fin.eof())
705     {
706             fin>>prof_; fin>>velp_; fin>>vels_; fin>>densidade_; fin>>
707                 gammaray_; fin>>porosidade_;
708
709             prof.push_back(prof_);
710             velp.push_back(velp_);
711             vels.push_back(vels_);
712             densidade.push_back(densidade_);
713             gammaray.push_back(gammaray_);
714             porosidade.push_back(porosidade_);
715         }
716
717         fin.close();
718
719     }
720
721 void CPerfil::Plotar_perfis()
722
723     {
724         Gnuplot fig4("lines");
725         fig4.Grid();
726         fig4.set_xlabel("Velocidade P (m/s)");
727         fig4.set_ylabel("Profundidade (m)");
728         fig4 << "set xrange [*:*] reverse";
729         fig4.plot_xy(velp, prof, "Perfil de velocidade P");
730         fig4 << "set terminal png size 1200,900";
731         fig4 << "set output 'Perfil de Velocidade P.png'";
732         fig4.plot_xy(velp, prof, "Perfil de velocidade P");
733
734         Gnuplot fig5("lines");
735         fig5.Grid();
```

```

736     fig5.set_xlabel("Velocidade\u209cS\u209c(m/s)");
737     fig5.set_ylabel("Profundidade\u209c(m)");
738     fig5 << "set\u209ayrange[*:]reverse";
739     fig5.plot_xy(vels,prof,"Perfil\u209de\u209avelocidade\u209cS");
740     fig5 << "set\u209eterminal\u209cpng\u209csize\u209c1200,900";
741     fig5 << "set\u209output\u209c'Perfil\u209de\u209aVelocidade\u209cS.png'";
742     fig5.plot_xy(vels,prof,"Perfil\u209de\u209aVelocidade\u209cS");

743
744     Gnuplot fig6("lines");
745     fig6.Grid();
746     fig6.set_xlabel("Raios\u209cGama\u209c(API)");
747     fig6.set_ylabel("Profundidade\u209c(m)");
748     fig6 << "set\u209ayrange[*:]reverse";
749     fig6.plot_xy(gammaRay,prof,"Perfil\u209de\u209craios\u209cgama");
750     fig6 << "set\u209eterminal\u209cpng\u209csize\u209c1200,900";
751     fig6 << "set\u209output\u209c'Perfil\u209de\u209aRaios\u209cGama.png'";
752     fig6.plot_xy(gammaRay,prof,"Perfil\u209de\u209craios\u209cgama");

753
754     Gnuplot fig7("lines");
755     fig7.Grid();
756     fig7.set_xlabel("Densidade\u209c(g/cm\u00b3)");
757     fig7.set_ylabel("Profundidade\u209c(m)");
758     fig7 << "set\u209ayrange[*:]reverse";
759     fig7.plot_xy(density,prof,"Perfil\u209de\u209cdensidade");
760     fig7 << "set\u209eterminal\u209cpng\u209csize\u209c1200,900";
761     fig7 << "set\u209output\u209c'Perfil\u209de\u209aDensidade.png'";
762     fig7.plot_xy(density,prof,"Perfil\u209de\u209cdensidade");

763
764     Gnuplot fig8("lines");
765     fig8.Grid();
766     fig8.set_xlabel("Porosidade\u209c(fracao)");
767     fig8.set_ylabel("Profundidade\u209c(m)");
768     fig8 << "set\u209ayrange[*:]reverse";
769     fig8.plot_xy(porosity,prof,"Perfil\u209de\u209cporosidade");
770     fig8 << "set\u209eterminal\u209cpng\u209csize\u209c1200,900";
771     fig8 << "set\u209output\u209c'Perfil\u209de\u209aPorosidade.png'";
772     fig8.plot_xy(porosity,prof,"Perfil\u209de\u209cporosidade");

773
774     cin.get();
775
776 }

```

Apresenta-se na listagem seguinte o arquivo com código da classe CGassmann.

Listing 6.15: Arquivo de cabeçalho da classe CGassmann.

```

778#ifndef CGASSMANN_H
779#define CGASSMANN_H
780
781#include "CRocha.h"

```

```
782#include "CMistura.h"
783#include "CAgua.h"
784#include "COleo.h"
785#include "CPerfil.h"
786#include "CGnuplot.h"
787
788#include <iostream>
789#include <vector>
790
791using namespace std;
792
793///Classe que contém as equações de Gassmann para a substituição de
794///fluidos
794class CGassmann
795
796{
797    public:
798        vector <double> ksat; ///Vetor que armazena os valores
799                    ///calculados do módulo de incompressibilidade da rocha
800                    ///saturada
801        vector <double> rhosat; ///Vetor que armazena os valores
802                    ///calculados da densidade da rocha saturada
803        vector <double> vp; ///Vetor que armazena os valores
804                    ///calculados da velocidade P da rocha saturada
805        vector <double> vs; ///Vetor que armazena os valores
806                    ///calculados da velocidade S da rocha saturada
807
808        vector <double> ksatperfil; ///Vetor que armazena os
809                    ///valores calculados do módulo de incompressibilidade
810                    ///da rocha saturada a partir dos perfis
811        vector <double> kdryperfil; ///Vetor que armazena os
812                    ///valores calculados do módulo de incompressibilidade
813                    ///da seca a partir dos perfis
814        vector <double> miperfil; ///Vetor que armazena os
815                    ///valores calculados do módulo de cisalhamento da rocha
816                    ///saturada a partir dos perfis
817
818        vector <double> ksatperfilsubstituido; ///Vetor que
819                    ///armazena os valores calculados do módulo de
820                    ///incompressibilidade da rocha saturada a partir dos
821                    ///perfis após a substituição de fluidos
822        vector <double> rhosatperfilsubstituido; ///Vetor que
823                    ///armazena os valores calculados da densidade da rocha
824                    ///saturada a partir dos perfis após a substituição de
825                    ///fluidos
826        vector <double> vpperfilsubstituido; ///Vetor que
827                    ///armazena os valores calculados da velocidade P da
828                    ///rocha saturada após a substituição de fluidos
```

```
810         vector <double> vsperfilsubstituido; ///Vetor que
811             armazena os valores calculados da velocidade S da
812             rocha saturada após a substituição de fluidos
813
814             CMistura* pmistura; ///Ponteiro para objeto da classe
815                 CMistura
816
817             CRocha* procha; ///Ponteiro para objeto da classe CRocha
818             CPerfil* pperfil; ///Ponteiro para objeto da classe
819                 CPerfil
820
821
822             public:
823
824             CGassmann(CMistura* _pmistura, CRocha* _procha, CPerfil*
825                 _pperfil): pmistura(_pmistura), procha(_procha),
826                     pperfil(_pperfil) {}; /// Construtor
827             ~CGassmann() {}; ///Destruitor
828             vector <double> Ksat(); ///Cálculo do vetor de módulo de
829                 incompressibilidade da rocha saturada
830             vector <double> Rhosat(); ///Cálculo do vetor de
831                 densidade da rocha saturada
832             vector <double> Vp(); ///Cálculo do vetor de velocidade
833                 compressional da rocha saturada
834             vector <double> Vs(); ///Cálculo do vetor de velocidade
835                 cisalhante da rocha saturada
836             void Plotar(); ///Plota os gráficos de Vp, Vs e rho em
837                 função da saturação de água
838
839             ///Todos os métodos abaixo, com o final "perfil" são
840                 métodos utilizados na parte 2 do trabalho onde os
841                 cálculos iniciais são feitos a partir de dados de
842                 perfis.
843             vector <double> Ksat_perfil(); ///Cálculo do vetor de
844                 módulo de incompressibilidade da rocha saturada a
845                 partir dos perfis (considerando 100% saturada com
846                 água)
847             vector <double> Kdry_perfil(); ///Cálculo do vetor de
848                 módulo de incompressibilidade da rocha seca a partir
849                 dos perfis
850             vector <double> Mi_perfil(); ///Cálculo do vetor de
851                 módulo cisalhamento da rocha saturada a partir dos
852                 perfis
853             vector <double> Ksat_perfil_substituido(); ///Cálculo do
854                 vetor de módulo de incompressibilidade da rocha
855                 saturada após a substituição de fluidos (considerando
856                 100% saturada com óleo)
857             vector <double> Rhosat_perfil_substituido(); ///Cálculo
858                 do vetor de densidade da rocha saturada a partir dos
```

```

    perfis após a substituição de fluidos
833     vector <double> Vp_perfil_substituido(); //Cálculo do
          vetor de velocidade compressional da rocha saturada
          após a substituição de fluidos
834     vector <double> Vs_perfil_substituido(); //Cálculo do
          vetor de velocidade cisalhante da rocha saturada após
          a substituição de fluidos
835     void Plotar_substituicao(); //Plota os gráficos de Vp,
          Vs e rho antes e depois da substituição de fluidos
836 };
837
838#endif

```

Apresenta-se na listagem seguinte o arquivo de implementação da classe CGassmann.

Listing 6.16: Arquivo de implementação da classe CGassmann.

```

839#include "CGassmann.h"
840
841#include <vector>
842#include <cmath>
843
844using namespace std;
845
846vector <double> CGassmann::Ksat()
847{
848    ksat.resize(101);
849
850    for(int i = 0; i <= 100; i++)
851    {
852        ksat[i] = (procha->kdrydado + ((pow((1.0 - procha->
853            kdrydado/procha->kma),2.0))/(procha->phi/(pmistura->kfl
854            [i]) + (1.0 - procha->phi)/procha->kma - (procha->
855            kdrydado/pow(procha->kma,2.0))));}
856
857    return ksat;
858}
859
860vector <double> CGassmann::Rhosat()
861{
862    rhosat.resize(101);
863
864    for (int i = 0; i <= 100; i++)
865    {
866        rhosat[i] = (procha->rhomax*(1.0-procha->phi) + pmistura->
867            rhofl[i]*(procha->phi));
868    }

```

```
868
869         return rhosat;
870     }
871
872 vector <double> CGassmann::Vp()
873
874 {
875     vp.resize(101);
876
877     for (int i = 0; i <= 100; i++)
878     {
879         vp[i] = (sqrt(((ksat[i]*pow(10.0,6.0)) + (((4.0/3.0)*procha
880             ->mima)*pow(10.0,6.0)))/(rhosat[i]*pow(10.0,3.0)))) ;
881     }
882
883     return vp;
884 }
885
886 vector <double> CGassmann::Vs()
887
888 {
889     vs.resize(101);
890
891     for (int i = 0; i <= 100; i++)
892     {
893         vs[i] = (sqrt((procha->mima*pow(10.0,6.0))/(rhosat[i]*pow
894             (10.0,3.0)))); ;
895     }
896
897     return vs;
898 }
899
900 vector <double> CGassmann::Ksat_perfil()
901
902 {
903     for(int i = 0; i < pperfil->prof.size(); i++)
904     {
905         ksatperfil.push_back((pperfil->densidade[i]*(pow(pperfil->
906             velp[i] * 1000.0,2.0) - (4.0/3.0) * pow(pperfil->vels[i]
907             * 1000.0,2.0)))/1000.0);
908     }
909
910     return ksatperfil;
911 }
```

```
912 vector <double> CGassmann::Kdry_perfil()
913
914     {
915
916         for(int i = 0; i < pperfil->prof.size(); i++)
917
918         {
919             kdryperfil.push_back((ksatperfil[i]*(((pperfil->porosidade[i]
920                 ] * procha->kma)/pmistura->kfl[100]) + 1 - pperfil->
921                 porosidade[i]) - procha->kma)/(((pperfil->porosidade[i] *
922                 procha->kma)/pmistura->kfl[100]) + ksatperfil[i]/procha
923                 ->kma - 1.0 - pperfil->porosidade[i]));
924
925         }
926
927         return kdryperfil;
928     }
929
930
931     for(int i = 0; i < pperfil->prof.size(); i++)
932
933     {
934         miperfil.push_back((pperfil->densidade[i] * pow(pperfil->
935             vels[i] * 1000.0,2.0))/1000.0);
936     }
937
938     return miperfil;
939 }
940
941
942     for(int i = 0; i < pperfil->prof.size(); i++)
943
944     {
945         ksatperfilsubstituido.push_back(kdryperfil[i] + (pow(1.0 -
946             kdryperfil[i]/procha->kma),2.0))/((pperfil->porosidade[i]
947             ]/pmistura->kfl[0]) + ((1.0 - pperfil->porosidade[i])/
948             procha->kma) - kdryperfil[i]/pow(procha->kma,2.0)));
949     }
950
951     return ksatperfilsubstituido;
952 }
953
954
955     vector <double> CGassmann::Rhosat_perfil_substituido()
```

```
952
953     {
954
955         for(int i = 0; i < pperfil->prof.size(); i++)
956
957         {
958             rhosatperfilsustituido.push_back(procha->rhomax * (1.0 -
959                                         pperfil->porosidade[i]) + pmistura->rhofl[0] * pperfil->
960                                         porosidade[i]);
961         }
962
963
964     return rhosatperfilsustituido;
965
966 }
967
968
969     for(int i = 0; i < pperfil->prof.size(); i++)
970
971     {
972         vpperfilsustituido.push_back(sqrt(((ksatperfilsustituido[i]
973                                         ]*pow(10.0,6.0)) + (((4.0/3.0)*miperfil[i])*pow(10.0,6.0)
974                                         ))/(rhosatperfilsustituido[i]*pow(10.0,3.0))));
975     }
976
977
978     return vpperfilsustituido;
979
980 }
981
982
983     vector <double> sw;
984
985     for (int i = 0; i <= 100; i++) //Criando o vetor de saturação
986         (0-100)
987     {
988         sw.push_back(i);
989
990         Gnuplot fig("lines");
991         fig.Grid();
992         fig.set_xlabel("Saturação");
993         fig.set_ylabel("Velocidade P");
994         fig.plot_xy(sw, vp, "Variação da velocidade da onda P com o aumento da saturação");
```

```

994
995     Gnuplot fig1("lines");
996     fig1.Grid();
997     fig1.set_xlabel("Saturacao");
998     fig1.set_ylabel("Velocidade_S");
999     fig1.plot_xy(sw,vs,"Variacao_da_velocidade_da_onda_S_com_o
1000         aumento_da_saturacao");

1001    Gnuplot fig2("lines");
1002    fig2.Grid();
1003    fig2.set_xlabel("Saturacao");
1004    fig2.set_ylabel("Densidade");
1005    fig2.plot_xy(sw,rhosat,"Variacao_da_densidade_com_o_aumento_da_
1006        saturacao");

1007    cin.get();

1008}

1009

1010

1011 void CGassmann::Plotar_substituicao()
1012 {
1013
1014     vector <double> velp1;
1015
1016
1017     for (int i = 0; i < pperfil->prof.size(); i++)
1018     {
1019         velp1.push_back
1020             (pperfil->velp[i]*1000.0);
1021     }
1022
1023     Gnuplot fig10("lines");
1024     fig10.Grid();
1025     fig10.set_xlabel("Velocidade_P_(m/s)");
1026     fig10.set_ylabel("Profundidade_(m)");
1027     fig10.plot_xy(velp1,pperfil->prof,"_Velocidade_P_(antes_da_
1028         substituicao:_agua)");
1029     fig10.replot();
1030     fig10.plot_xy(vpperfilsustituido,pperfil->prof,"_Velocidade_
1031         _P_(apos_a_substituicao:_oleo)");
1032
1033     cin.get();
1034 }
```

Apresenta-se na listagem seguinte o arquivo com código da classe CAVO.

Listing 6.17: Arquivo de cabeçalho da classe CAVO.

```
1034 #ifndef CAVO_H
```

```
1035#define CAVO_H
1036
1037#include <iostream>
1038#include <vector>
1039
1040#include "CGassmann.h"
1041#include "CGnuplot.h"
1042
1043using namespace std;
1044
1045///Classe com atributos e métodos para a modelagem AVO.
1046class Cavo
1047{
1048public:
1049    double vp2; ///Velocidade compressional da camada
1050        inferior
1051    double vs2; ///Velocidade cisalhante na camada inferior
1052    double rho2; ///Densidade da camada inferior
1053    double vp1; ///Velocidade compressional da camada
1054        superior
1055    double vs1; ///Velocidade cisalhante da camada superior
1056    double rho1; ///Densidade da camada superior
1057    double deltarho; ///Diferença entre as densidades das
1058        camadas
1059    double deltavp; ///Diferença entre as velocidades
1060        compressionais das camadas
1061    double deltavs; ///Diferença entre as velocidades
1062        cisalhantes das camadas
1063    double vpmed; ///Velocidade compressional média entre as
1064        camadas
1065    double vsmed; ///Velocidade cisalhante média entre as
1066        camadas
1067    double rhomed; ///Densidade média entre as camadas
1068    vector <double> rshuey; ///Vetor que armazena os valores
1069        da refletividade calculado pelo método de Shuey
    vector <double> raki; ///Vetor que armazena os valores
        da refletividade calculado pelo método de Aki &
        Richards
    vector <double> rzoepritz; ///Vetor que armazena os
        valores da refletividade calculado pelo método de
        Zoepritz
    CGassmann* pgassmann; ///Cria ponteiro para a classe
        CGassmann
    CGnuplot fig3; ///Cria objeto do tipo CGnuplot
    CGnuplot fig12; ///Cria objeto do tipo CGnuplot
    CGnuplot fig13; ///Cria objeto do tipo CGnuplot
    CGnuplot fig14; ///Cria objeto do tipo CGnuplot
```

```

1070     public:
1071
1072         CAvo(CGassmann* _pgassmann): pgassmann(_pgassmann) {};
1073         ///Construtor
1074         ~CAvo() {}; //Destruitor
1075         void Setvp2(double _vp2); ///Set
1076         double Getvp2(); ///Get
1077         void Setvs2(double _vs2); ///Set
1078         double Getvs2(); ///Get
1079         void Setrho2(double _rho2); ///Set
1080         double Getrho2(); ///Get
1081         void Setvp1(double _vp1); ///Set
1082         double Getvp1(); ///Get
1083         void Setvs1(double _vs1); ///Set
1084         double Getvs1(); ///Get
1085         void Setrho1(double _rho1); ///Set
1086         double Getrho1(); ///Get
1087         vector <double> Rshuey(); ///Cálculo da refletividade
1088             pelo método de Shuey
1089         vector <double> Raki(); ///Cálculo da refletividade pelo
1090             método de Aki & Richards
1091         vector <double> Rzoepritz(); ///Cálculo da
1092             refletividade pelo método de Zoeppritz
1093         void Entrada_saturacao(); ///Usuário escolhe o valor da
1094             saturação em que se deseja fazer a modelagem AVO e os
1095             valores de Vp, Vs e densidade para essa saturação
1096             são selecionados.
1097         void Entrada_camada_superior(); ///Pede ao usuário os
1098             dados de Vp, Vs e densidade da camada superior (
1099                 sugerida como folhelho)
1100         void Plotar_refletividade(); ///Plota os gráficos da
1101             refletividade pelos métodos propostos
1102     };
1103 #endif

```

Apresenta-se na listagem seguinte o arquivo de implementação da classe CAVO.

Listing 6.18: Arquivo de implementação da classe CAvo.

```

1096 #include "CAvo.h"
1097
1098 #include <cmath>
1099 #include <vector>
1100
1101 using namespace std;
1102
1103 void CAvo::Setvp2(double _vp2){vp2=_vp2;}
1104

```

```
1105 double CAvo::Getvp2(){return vp2;}
1106
1107 void CAvo::Setrho2(double _rho2){rho2=_rho2;}
1108
1109 double CAvo::Getrho2(){return rho2;}
1110
1111 void CAvo::Setvs2(double _vs2){vs2=_vs2;}
1112
1113 double CAvo::Getvs2(){return vs2;}
1114
1115 void CAvo::Setvp1(double _vp1){vp1=_vp1;}
1116
1117 double CAvo::Getvp1(){return vp1;}
1118
1119 void CAvo::Setrho1(double _rho1){rho1=_rho1;}
1120
1121 double CAvo::Getrho1(){return rho1;}
1122
1123 void CAvo::Setvs1(double _vs1){vs1=_vs1;}
1124
1125 double CAvo::Getvs1(){return vs1;}
1126
1127 vector <double> CAvo::Rshuey()
1128 {
1129     deltarho = rho2 - rho1;
1130     deltavp = vp2 - vp1;
1131     deltavs = vs2 - vs1;
1132     rhomed = (rho2 + rho1)/2.0;
1133     vpmmed = (vp2 + vp1)/2.0;
1134     vsmed = (vs2 + vs1)/2.0;
1135
1136     rshuey.resize(41);
1137     for(int i = 0; i <= 40; i++)
1138     {
1139         rshuey[i] = (0.5*(deltavp/vpmmed + deltarho/rhomed) + (0.5*(deltavp/vpmmed) - 2.0*(pow(vsmed,2.0)/pow(vpmmed,2.0)))*((deltarho/rhomed) + ((2.0*deltavs)/vsmed)))*pow(sin((i*3.14)/(180.0)),2.0));
1140     }
1141 }
1142
1143     return rshuey;
1144 }
1145
1146 vector <double> CAvo::Raki()
1147
1148 {
1149     deltarho = rho2 - rho1;
```

```
1150     deltavp = vp2 - vp1;
1151     deltavs = vs2 - vs1;
1152     rhomed = (rho2 + rho1)/2.0;
1153     vpmed = (vp2 + vp1)/2.0;
1154     vsmed = (vs2 + vs1)/2.0;
1155
1156     raki.resize(41);
1157     for(int i = 0; i <= 40; i++)
1158     {
1159         double alfa, beta, gama;
1160
1161         alfa = 1.0/(2.0*pow(cos((i*3.14)/(180.0)),2.0));
1162
1163         beta = -4.0 * ((pow(vsmed,2.0))/(pow(vpmed,2.0))) * pow(sin((i
1164             *3.14)/(180.0)),2.0);
1165
1166         gama = 0.5 - 2 * ((pow(vsmed,2))/(pow(vpmed,2))) * pow(sin((i
1167             *3.14)/(180.0)),2.0);
1168
1169         raki[i] = (deltavp/vpmed*alfa + deltavs/vsmed*beta + deltarho/
1170             rhomed*gama);
1171     }
1172
1173 }
1174
1175 vector <double> CAvo::Rzoepritz()
1176
1177 {
1178     double r0,r1,r2,r3,r4;
1179     r0 = 1.0;
1180     r1 = vp2/vp1;
1181     r2 = vs1/vp1;
1182     r3 = vs2/vp1;
1183     r4 = rho2/rho1;
1184
1185     rzoepritz.resize(41);
1186
1187     for(int i = 0; i <= 40; i++)
1188     {
1189         double t0,t1,t2,t3;
1190
1191         t0 = (r0 * sin((i*3.14)/(180.0)))/(pow((1 - pow(r0,2.0) * pow(
1192             sin((i*3.14)/(180.0)),2.0)),0.5));
1193         t1 = (r1 * sin((i*3.14)/(180.0)))/(pow((1 - pow(r1,2.0) * pow(
```

```
        sin((i*3.14)/(180.0)),2.0)),0.5));
1194 t2 = (r2 * sin((i*3.14)/(180.0)))/(pow((1 - pow(r2,2.0) * pow(
1195     sin((i*3.14)/(180.0)),2.0)),0.5));
t3 = (r3 * sin((i*3.14)/(180.0)))/(pow((1 - pow(r3,2.0) * pow(
1196     sin((i*3.14)/(180.0)),2.0)),0.5));
1197
1198 double q,a,b;
1199
q = 2.0 * pow(sin((i*3.14)/(180.0)),2.0) * (r4 * pow(r3,2.0)
1200     - pow(r2,2.0));
a = pow((r4-q),2.0) * t1 * t3;
1201 b = pow((r4-q-1.0),2.0)*t0*t1*t2*t3;
1202
1203 rzoeppritz[i] = ((pow(q,2.0) - r4*t0*t3 + r4*t1*t2 - ((pow
1204     ((1.0 + q),2.0))*t0*t2) + a + b)/(((pow((1.0 + q),2.0)*t0*
1205     t2) + a + b) + (pow(q,2.0) + r4*t0*t3 + r4*t1*t2)));
1206 }
1207
1208 }
1209
1210 void CAvo::Entrada_camada_superior()
1211 {
1212     cout << "Entre com valor de Vp da camada superior em /s: ";
1213     cin >> vp1;
1214     cout << endl;
1215
1216     cout << "Entre com valor de Vs da camada superior em /s: ";
1217     cin >> vs1;
1218     cout << endl;
1219
1220     cout << "Entre com valor da densidade da camada superior em
1221         /s: ";
1222     cin >> rho1;
1223     cout << endl;
1224     cin.get();
1225 }
1226
1227 void CAvo::Plotar_refletividade()
1228 {
1229
1230     cout << "Pressione ENTER para continuar." << "\n";
1231
1232     vector <double> teta;
1233
1234     for (int i = 0; i <= 40; i++)
```

```
1235
1236    {
1237        teta.push_back(i);
1238    }
1239
1240    int saturacao;
1241    double vp2;
1242    double vs2;
1243    double rho2;
1244
1245    saturacao = 0;
1246    vp2 = pgassmann->vp[saturacao];
1247    Setvp2(vp2);
1248    vs2 = pgassmann->vs[saturacao];
1249    Setvs2(vs2);
1250    rho2 = pgassmann->rhosat[saturacao];
1251    Setrho2(rho2);
1252
1253    fig3.set_style("lines");
1254    fig3.Grid();
1255    fig3.set_title("Modelagem AVO para reservatorio 100% saturado com oleo.");
1256    fig3.set_xlabel("Angulo (graus)");
1257    fig3.set_ylabel("Refletividade");
1258    fig3.plot_xy(teta, Rshuey(), "Refletividade vs Angulo: Metodo de Shuey");
1259    fig3.replot();
1260    fig3.plot_xy(teta, Raki(), "Refletividade vs Angulo: Metodo de Aki & Richards");
1261    fig3.replot();
1262    fig3.plot_xy(teta, Rzoepritz(), "Refletividade vs Angulo: Metodo de Zoepritz");
1263
1264    saturacao = 100;
1265    vp2 = pgassmann->vp[saturacao];
1266    Setvp2(vp2);
1267    vs2 = pgassmann->vs[saturacao];
1268    Setvs2(vs2);
1269    rho2 = pgassmann->rhosat[saturacao];
1270    Setrho2(rho2);
1271
1272    fig13.set_style("lines");
1273    fig13.Grid();
1274    fig13.set_title("Modelagem AVO para reservatorio 100% saturado com agua.");
1275    fig13.set_xlabel("Angulo (graus)");
1276    fig13.set_ylabel("Refletividade");
1277    fig13.plot_xy(teta, Rshuey(), "Refletividade vs Angulo: Metodo
```

```

        de_Shuey");
1278    fig13.replot();
1279    fig13.plot_xy(teta,Raki(),"Refletividade vs Angulo: Metodo de
1280                  Aki & Richards");
1280    fig13.replot();
1281    fig13.plot_xy(teta,Rzoepritz(),"Refletividade vs Angulo:-
1282                  Metodo de Zoepritz");
1282    cin.get();
1283
1284}

```

Apresenta-se na listagem seguinte o arquivo com código da classe CHelp.

Listing 6.19: Arquivo de cabeçalho da classe CHelp.

```

1285 #ifndef CHELP_H
1286 #define CHELP_H
1287
1288 #include <iostream>
1289
1290 //Esta classe apresenta um pequeno manual do programa
1291 class CHelp
1292
1293 {
1294     public:
1295
1296     public:
1297
1298     void Ajuda(); // Ajuda
1299 };
1300
1301#endif

```

Apresenta-se na listagem seguinte o arquivo de implementação da classe CHelp.

Listing 6.20: Arquivo de implementação da classe CHelp.

```

1302 #include "CHelp.h"
1303
1304 using namespace std;
1305
1306 void CHelp::Ajuda()
1307 {
1308     cout << "\nSistema de Ajuda do Simulador de Substituição
1309          de Fluidos e Modelagem AV0" << "\n";
1309     cout << "Segue um abreviado manual sobre o uso do
1310          simulador proposto" << "\n";
1310     cout << "Simular ao capítulo 8 do Projeto de Engenharia"
1310          << "\n";
1311
1312     cout << "SOBRE O SIMULADOR" << "\n\n";

```

```
1313     cout << "Desenvolvido por Maurício Matos e Rafael  
Boechat, como projeto de engenharia para a disciplina  
de Programação Prática no segundo semestre de 2015"  
<< "\n\n";  
1314  
1315     cout << "Sobre o tema" << "\n\n";  
1316     cout << "Durante a etapa de exploração do petróleo,  
utiliza-se uma série de métodos geofísicos como  
ferramentas para inferir sobre as características das  
rochas e fluidos em subsuperfície, buscando  
identificar possíveis intervalos de acumulação de  
hidrocarbonetos." << "\n";  
1317     cout << "Diante disso, a importância da geofísica está  
relacionada com a possibilidade de caracterizar a maior  
área reservatório, fornecendo uma maior  
confiabilidade no potencial produtor de um sistema  
petrolífero. Tais métodos são utilizados em larga  
escala para um conjunto de dados muito grande, o que  
destaca a relevância de se ter softwares que  
sejam capazes de trabalhar com esses dados." << "\n";  
1318     cout << "Esta tendência é a motivação do nosso projeto  
de engenharia onde busca-se criar um simulador de  
substituição de fluidos e modelagem AVO com a  
finalidade de caracterizar intervalos de  
reservatórios, visando auxiliar o usuário na etapa de  
interpretação geofísica." << "\n\n";  
1319  
1320     cout << "PRÉ-REQUISITOS E DEPENDÊNCIAS" << "\n\n";  
1321     cout << "- Ter o compilador g++ instalado" << "\n";  
1322     cout << "- Ter o software GnuPlot instalado" << "\n";  
1323     cout << "- Ter um arquivo com dados de perfilagem a  
disposição" << "\n\n";  
1324  
1325     cout << "Sobre o funcionamento" << "\n\n";  
1326     cout << "Previamente, deve-se assegurar que os dados de  
perfis de entrada estejam na mesma pasta que o  
programa. Também é importante ressaltar que os dados de  
entrada sejam fisicamente coerentes, caso  
contrário resultados incoerentes serão obtidos." <<  
"\n";  
1327     cout << "No cabeçalho do programa, tem-se o nome, a  
universidade, a data, a versão e os autores do  
programa. Abaixo o usuário deve optar por realizar  
uma substituição de fluidos com modelagem AVO, a  
substituição de fluidos a partir de dados de perfis  
ou obter ajuda." << "\n\n";  
1328     cout << "1. Caso digite 1, a primeira opção será  
executada." << "\n";
```

```

1329     cout << "a. .... Primeiramente serão solicitados ao usuário os dados de entrada dos fluidos, água e óleo ou da rocha e seus componentes principais." << "\n";
1330     cout << "b. .... A matriz rochosa nesse trabalho foi modelada supondo ser composta por dois minerais, então serão solicitadas propriedades elásticas referentes a esses dois minerais bem como a fração em volume de cada um na matriz da rocha." << "\n";
1331     cout << "c. .... Caso a soma das frações dos dois minerais não seja igual a um, uma mensagem de erro será exibida já que seria fisicamente impossível nessa modelagem. Fica a cargo do usuário prosseguir ou reiniciar a execução do programa." << "\n";
1332     cout << "d. .... Após a entrada de dados o programa calculará as propriedades dos fluidos e a rocha efetivos e utilizará as equações de Gassmann implementadas para realizar a substituição dos fluidos. Comissão será gerado três gráficos de propriedades do meio com o aumento da saturação de água." << "\n";
1333     cout << "e. .... As propriedades calculadas para a rocha e fluido efetivos são salvos nos arquivos externos Resultados_rocha.txt e Resultados_fluido.txt respectivamente. Ambos podem ser localizados na mesma pasta do programa." << "\n";
1334     cout << "f. .... O próximo passo é fornecer ao programa os dados para a modelagem AV0. A partir do que já foi calculado, deve-se fornecer o valor de saturação em que se deseja realizar a modelagem. Além disso, é necessário fornecer as propriedades elásticas da camada superior ao reservatório." << "\n";
1335     cout << "g. .... De posse de todos os dados, será realizada a modelagem AV0 para os três métodos implementados e um gráfico de refletividade por ângulo será gerado e a simulação encerrada." << endl
           << endl;
1336     cout << "2. .... Caso digite 2, a segunda opção será executada." << "\n";
1337     cout << "a. .... Será solicitado ao usuário o nome do arquivo e formato com os dados de perfilagem." << "\n";
1338     cout << "b. .... Após o carregamento dos perfis, serão realizados os mesmos procedimentos apresentados nos itens a, b e c do caso anterior." << "\n";
1339     cout << "c. .... Os perfis carregados são salvos na pasta do programa juntamente com os dados calculados como descrito no item e do caso acima." << "\n";
1340     cout << "d. .... Com isso, será realizada a substituição

```

```

de fluidos de água para óleo no perfil carregado. □
Como dito, partimos do pressuposto que o perfil
carregado é uma formação totalmente saturada com água
e que passará a ser totalmente saturada com óleo. □
O comportamento da velocidade na camada com a mudança
do fluido antes e depois pode ser visto no gráfico
gerado, encerrando assim esta simulação." << "\n\n";
1341
1342         cout << "\nVersão 1.0\n" << "\n";
1343
1344     }

```

Apresenta-se na listagem o main.cpp.

Listing 6.21: Arquivo de implementação da função main().

```

1345 #include "CFluido.h"
1346 #include "CAgua.h"
1347 #include "COleo.h"
1348 #include "CGas.h"
1349 #include "CGnuplot.h"
1350 #include "CMistura.h"
1351 #include "CRocha.h"
1352 #include "CPerfil.h"
1353 #include "CAvo.h"
1354 #include "CHelp.h"
1355
1356 using namespace std;
1357
1358 int main()
1359 {
1360     /// Criando os objetos
1361     CAgua agua;
1362     COleo oleo;
1363     CRocha rocha;
1364     CPerfil perfil;
1365     CMistura mistura(&agua,&oleo);
1366     CGassmann gassmann(&mistura, &rocha, &perfil);
1367     CAvo avo(&gassmann);
1368     CHelp help;
1369
1370     /// Cabeçalho
1371     cout << "\n
1372             -----" <<
1373             endl;
1374     cout << "SIMULADOR DE SUBSTITUIÇÃO DE FLUIDOS E DE MODELAGEM AVO
1375             " << endl << endl;
1376     cout << "UNIVERSIDADE ESTADUAL DO NORTE FLUMINENSE DARCY RIBEIRO
1377             " << endl;

```

```
1375     cout << "..... JANEIRO DE 2016....."
1376     " << endl;
1377     cout << "-----"
1378     " << endl;
1379     cout << "AUTORES: RAFAEL BOECHAT E MAURÍCIO MATOS" << endl;
1380     cout << "VERSÃO 1.0" << endl << endl;
1381
1382     int resp;
1383     cout << "Qual simulação deseja realizar?" << endl;
1384     cout << "Digite 1 para Substituição de fluidos e modelagem AVO."
1385     << endl;
1386     cout << "Digite 2 para Substituição de fluidos por meio do
1387     carregamento de dados de perfis." << endl;
1388     cout << "Digite 3 para Ajuda." << endl;
1389     cin >> resp;
1390
1391     switch (resp)
1392     {
1393         case 1:
1394
1395             cout << "\n---PARTE A1---PROPRIEDADES DOS FLUIDOS PELAS
1396             EQUAÇÕES DE BATZLE & WANG---" << endl << endl;
1397
1398             mistura.Entrada_propriedades(); //Pedimos ao usuário os
1399             dados de entrada (p, t, api, salinidade...) dos fluidos.
1400
1401             mistura.Saida_propriedades(); //As propriedades dos fluidos
1402             são calculadas e salvas em um arquivo externo.
1403
1404             cout << "\n---PARTE A2---PROPRIEDADES DA ROCHE E FLUIDOS
1405             PELAS EQUAÇÕES DE GASSMANN---" << endl << endl;
1406
1407             rocha.Entrada_propriedades(); //Pedimos ao usuário os dados
1408             de entrada da rocha e seus componentes.
1409
1410             rocha.Saida_propriedades(); //As propriedades da rocha são
1411             calculadas e salvas em um arquivo externo.
1412
1413             //Calculo das propriedades da rocha saturada
1414             gassmann.Ksat();
1415             gassmann.Rhosat();
1416             gassmann.Vp();
1417             gassmann.Vs();
1418
1419             gassmann.Plotar(); //Plotando os gráficos das velocidades e
1420             densidade com o aumento da saturação.
```

```

1412
1413         cout << "\n---PARTE\uA3\u-MODELAGEM\uAVO---" << endl << endl;
1414
1415         avo.Entrada_camada_superior(); //Pedimos ao usuário os
1416             valores de Vp, Vs e densidade da camada superior.
1417
1418         avo.Plotar_refletividade(); //Plota em um mesmo gráfico as
1419             refletividades calculadas pelos três métodos.
1420
1421         cout << "Simulação\u1uterminada!\a" << endl;
1422         cout << "Aperte\uENTER\upara\uasair\a" << endl;
1423
1424         break;
1425
1426     case 2:
1427
1428         cout << "\n\n---PARTE\uB1\u-CARREGANDO\uE\uPLOTANDO\uOS\uPERFIS
1429             ---" << endl << endl;
1430
1431         perfil.Ler_perfis(); //Carrega os perfis
1432
1433         perfil.Plotar_perfis(); //Plota os perfis e salva-os
1434             externamente.
1435
1436         cout << "\n---PARTE\uB2\u-PROPRIEDADES\uDOS\uFLUIDOS\uPELAS\u
1437             EQUA\u00c7\u00f5ES\uDE\uBATZLE\u&\uWANG---" << endl << endl;
1438
1439         mistura.Entrada_propriedades(); //Pedimos ao usuário os
1440             dados de entrada (p, t, api, salinidade...) dos fluidos.
1441
1442         mistura.Saida_propriedades(); //As propriedades dos fluidos
1443             são calculadas e salvas em um arquivo externo.
1444
1445         cout << "\n---PARTE\uB3\u-PROPRIEDADES\uDA\uROCHA\uE\uFLUIDOS\u
1446             PELAS\uEQUA\u00c7\u00f5ES\uDE\uGASSMANN---" << endl << endl;
1447
1448         rocha.Entrada_propriedades(); //Pedimos ao usuário os dados
1449             de entrada da rocha e seus componentes.
1450
1451         rocha.Saida_propriedades(); //As propriedades da rocha são
1452             calculadas e salvas em um arquivo externo.
1453
1454         cout << "\n---PARTE\uB4\u-REALIZANDO\uA\uSUBSTITUI\u00c7\u00f5O\uDE\u
1455             FLUIDOS\uNO\uARQUIVO\uCARREGADO---" << endl << endl;
1456
1457         ///Cálculo das propriedades do perfil antes da substituição.
1458         gassmann.Ksat_perfil();

```

```
1449         gassmann.Kdry_perfil();
1450         gassmann.Mi_perfil();
1451         ///Cálculo das propriedades do perfil após a substituição.
1452         gassmann.Ksat_perfil_substituido();
1453         gassmann.Rhosat_perfil_substituido();
1454         gassmann.Vp_perfil_substituido();

1455
1456         gassmann.Plotar_substituicao(); ///Plota o perfil das
1457         //velocidades antes e após a substituição de fluidos.

1458         cout << "Simulação terminada!\a" << endl;
1459         cout << "Aperte ENTER para sair\a" << endl;
1460
1461         break;
1462
1463     case 3:
1464
1465         help.Ajuda();
1466         cout << "Aperte ENTER para sair\a" << endl;
1467
1468         break;
1469
1470     default:
1471
1472         cout << "Opção incorreta!\a" << endl;
1473         cout << "Aperte ENTER para sair\a" << endl;
1474
1475         break;
1476
1477     }
1478
1479     cin.get();
1480     return 0;
1481
1482 }
```

Capítulo 7

Teste

Neste capítulo apresentamos alguns testes do software desenvolvido. Como já dito, esse software trabalha com dois tipos de simulação que serão apresentadas nas sessões seguintes. A Figura 7.1 apresenta o cabeçalho do programa e a escolha da simulação a ser executada.

7.1 Teste 1: Substituição de fluidos e modelagem AVO

Os dados de entrada fornecidos pelo usuário estão dispostos na Tabela 7.1 e na Tabela 7.2.

Tabela 7.1: Dados do reservatório

Pressão	30 MPa
Temperatura	80 °C
Salinidade da água	50000 ppm
°API do óleo	35
Porosidade	0.2
K_{dry} [MPa]	10000

Tabela 7.2: Dados dos componentes da matriz

	Componente 1 (quartzo)	Componente 2 (argilomineral)
K [MPa]	37000	23000
ρ [g/cm ³]	2.65	2.58
μ [MPa]	44000	8000
Fração da matriz da rocha	0.8	0.2

A Figura 7.2 apresenta a entrada dos dados fornecido pelo usuário.

A Figura 7.3 apresenta o arquivo externo onde foram armazenados os dados de entrada e o resultados dos cálculos das propriedades dos fluidos.

A Figura 7.4 apresenta o arquivo externo onde foram armazenados os dados de entrada e o resultados dos cálculos das propriedades da rocha.

As Figuras 7.5, 7.6 e 7.7 apresentam os gráficos gerados pelo programa.

Para a modelagem AVO faz-se necessário entrar com os valores médio da camada selante, suposta como folhelho. Tais dados são apresentados na Tabela 7.3.

Tabela 7.3: Propriedades da rocha selante

V_p	3030 m/s
V_s	1500 m/s
ρ	2.4 g/cm ³

A Figura 7.8 apresenta o gráfico gerado pelo software a partir dos dados fornecidos para a modelagem AVO pelos métodos de Zoeppritz, Aki and Richards e Shuey.

O software gera dois gráficos de modelagem AVO, um para reservatórios saturados totalmente por água e outro por óleo. Ambos são similares, evidenciando a dificuldade da modelagem AVO na discriminação de fluidos.

7.2 Teste 2: Substituição de fluidos a partir de dados de perfis

Nesse teste, a primeira etapa será o carregamento de dados de perfis de um arquivo em extensão .txt, exemplificado na Figura 5.1.

A Figura 7.9 apresenta os perfis gerados.

Após o carregamento dos perfis foram realizados os mesmos cálculos das propriedades dos fluidos e da rocha do Teste 1, utilizando os mesmos dados apresentados nas Tabelas 1 e 2 .

Com posse das propriedades do reservatório concluiu-se a substituição de fluido, que está apresentado no gráfico da Figura 7.10, onde tem-se o antes e o depois da substituição de fluidos com a rocha totalmente saturada com água e óleo respectivamente.

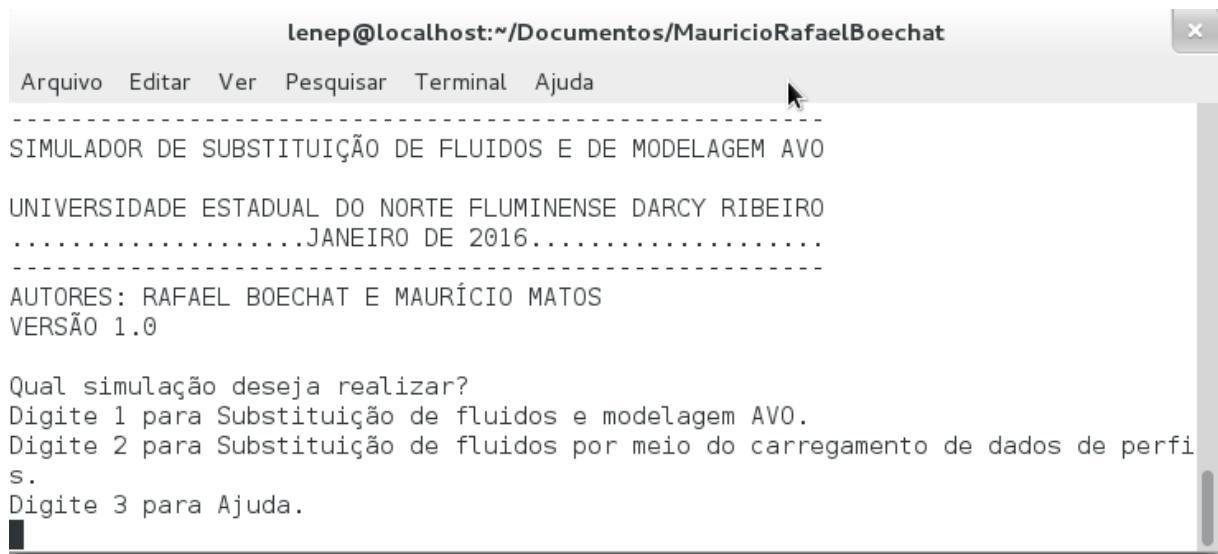


Figura 7.1: Tela do programa mostrando a opção de escolha de simulação

```
lenep@localhost:~/Documentos/MauricioRafaelBoechat
Arquivo Editar Ver Pesquisar Terminal Ajuda

---PARTE A1 - PROPRIEDADES DOS FLUIDOS PELAS EQUAÇÕES DE BATZLE & WANG---

Entre com valor da temperatura in situ em graus Celsius: 80
Entre com valor da pressão de poros em MPa: 30
Entre com valor da salinidade da água em ppm: 50000
Entre com valor da densidade em API do óleo: 28
Os dados inicializados e calculados para os fluidos foram salvos no arquivo externo
Resultados_fluidos.txt

---PARTE A2 - PROPRIEDADES DA ROCHA E FLUIDOS PELAS EQUAÇÕES DE GASSMANN---

Entre com o valor da porosidade média da camada em fração: 0.2
Nessa simulação de teste a matriz será modelada considerando ser composta por 2 componentes:
Entre com o valor do módulo de incompressibilidade do primeiro componente da matriz
da rocha: 37000
Entre com o valor da densidade do primeiro componente da matriz da rocha: 2.65
Entre com o valor do modulo de cisalhamento do primeiro componente da matriz da rocha
a: 44000
Entre com o valor do módulo de incompressibilidade do segundo componente da matriz d
a rocha: 23000
Entre com o valor da densidade do segundo componente da matriz da rocha: 2.58
Entre com o valor do modulo de cisalhamento do segundo componente da matriz da rocha
: 8000
Entre com a fração do primeiro componente na rocha: 0.8
Entre com a fração do segundo componente na rocha: 0.2
Entre com o valor do módulo de incompressibilidade da rocha seca: 10000
Os dados inicializados e calculados foram salvos no arquivo externo Resultados_rocha
```

Figura 7.2: Tela do programa mostrando a entrada de dados

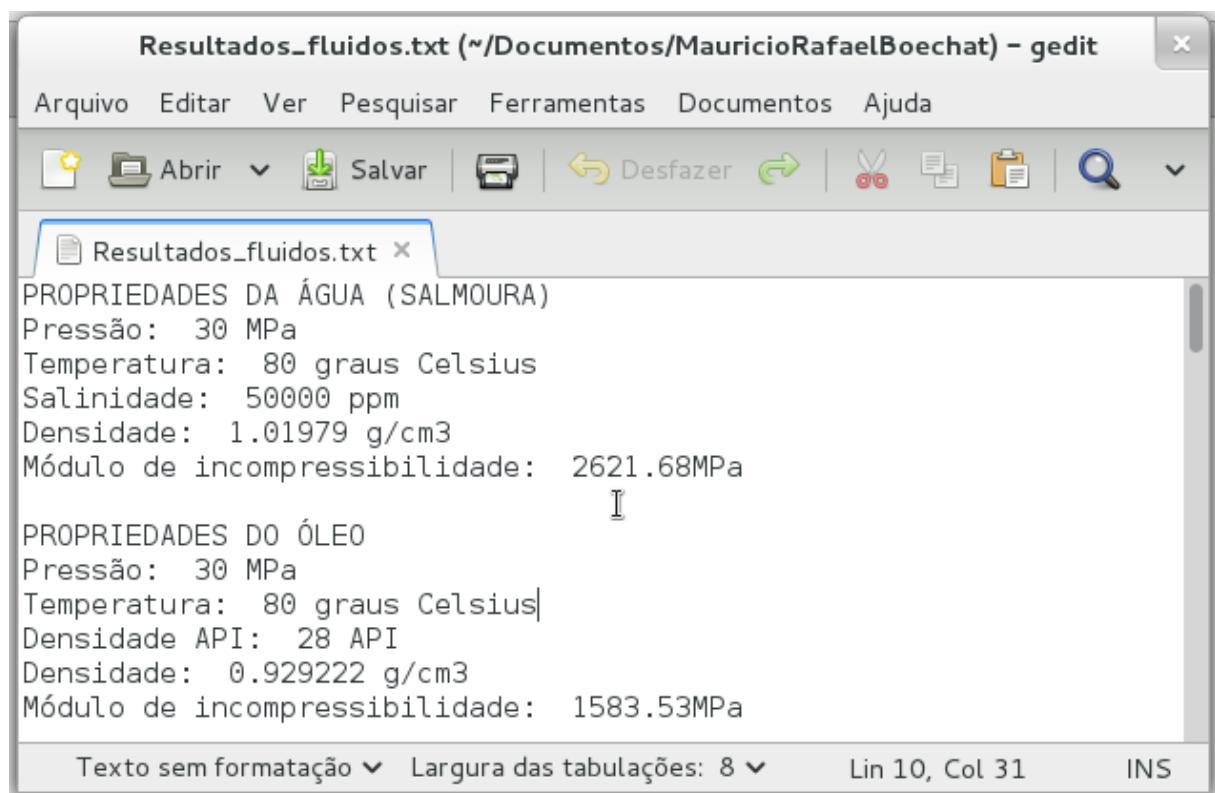


Figura 7.3: Tela do programa mostrando os resultados e dados de entrada

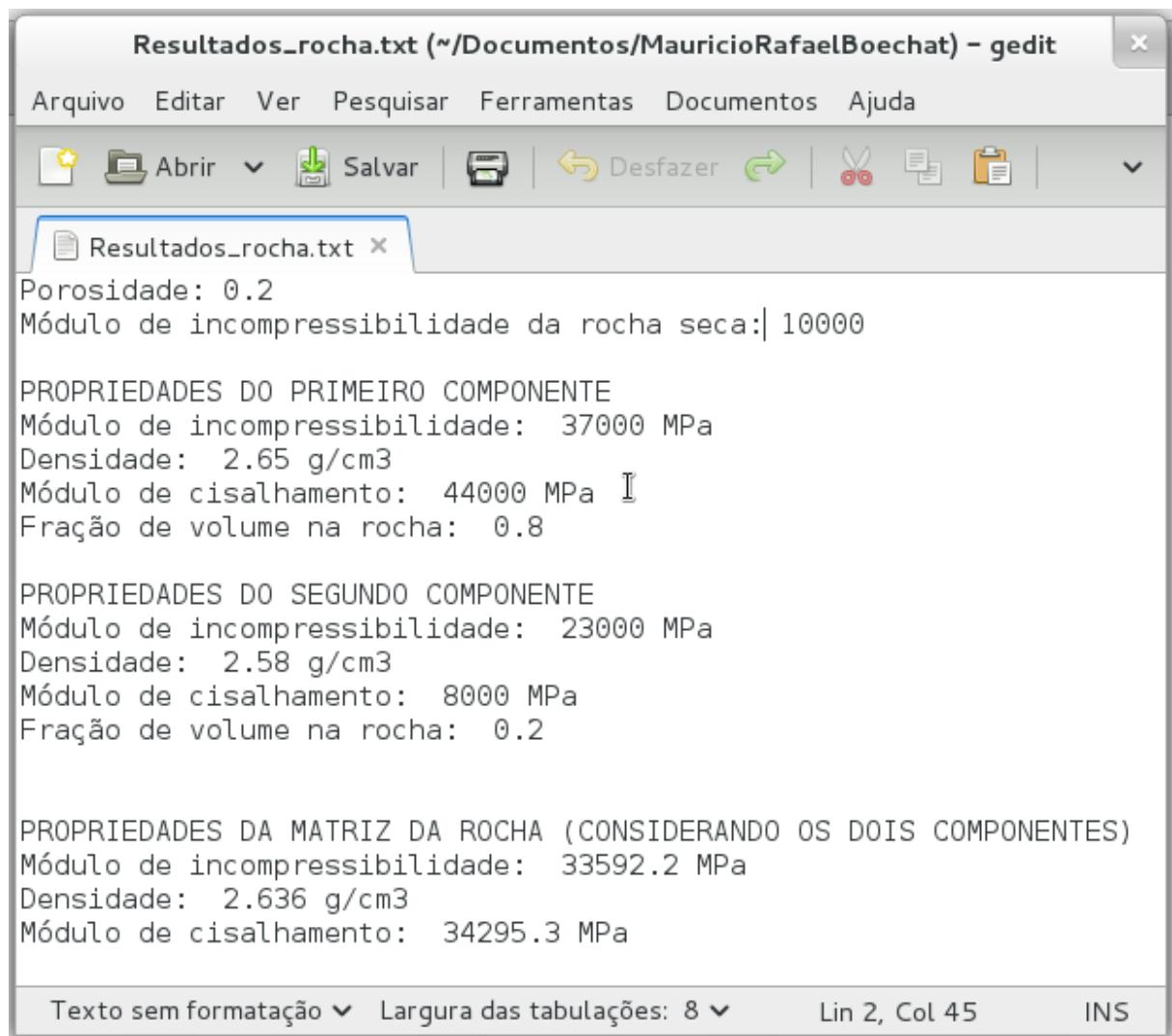


Figura 7.4: Tela do programa mostrando os dados de entrada para as rocha

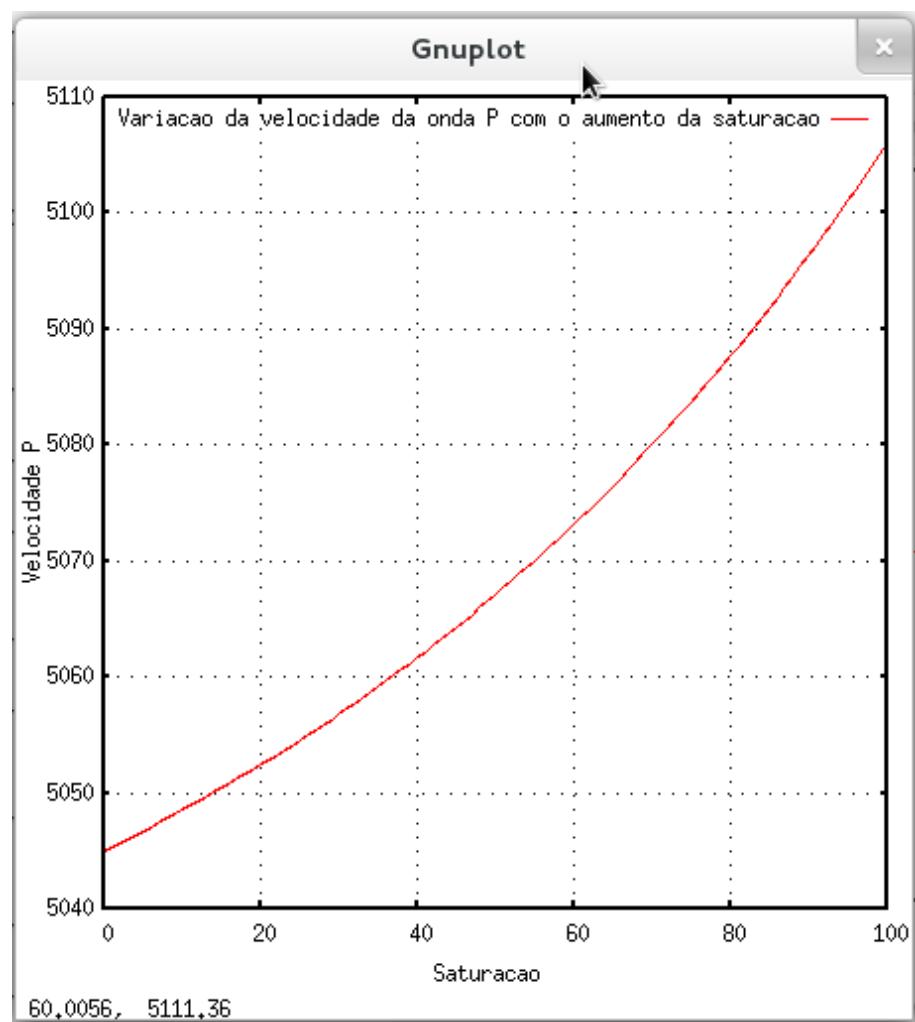
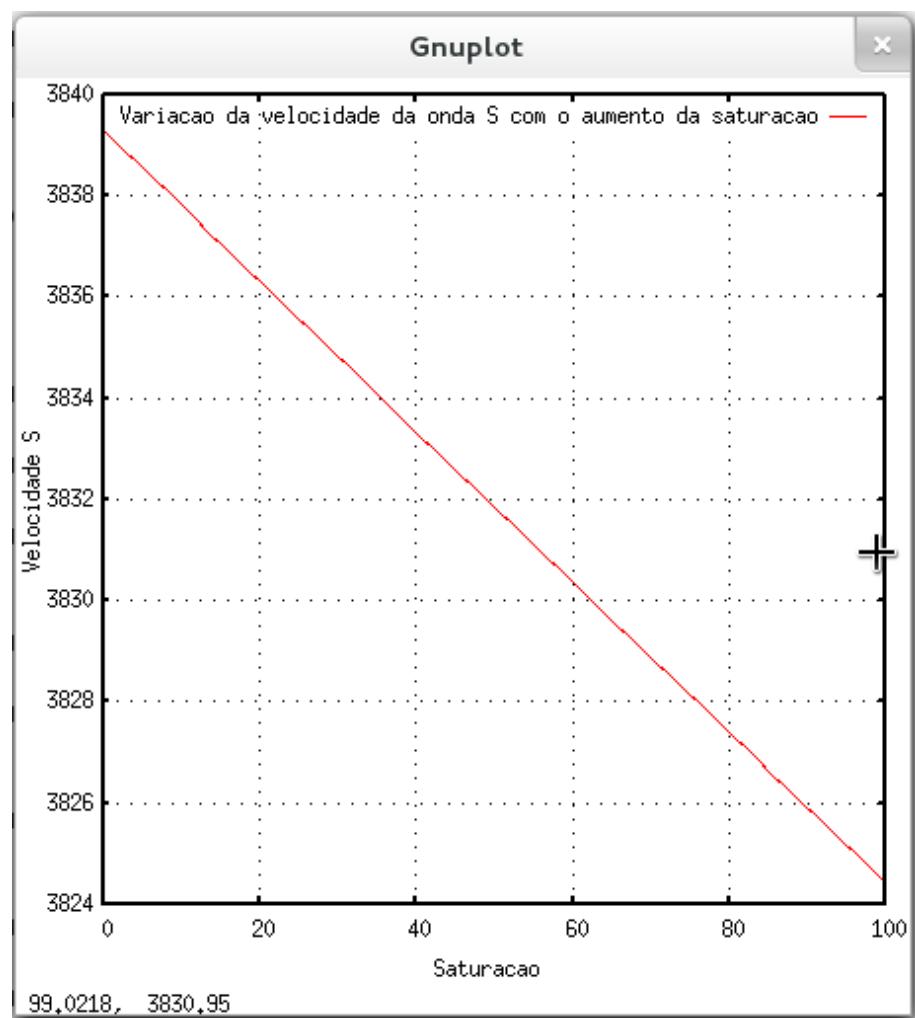


Figura 7.5: Gráfico gerado pelo Gnuplot de $V_p \times S_w$

Figura 7.6: Gráfico gerado pelo Gnuplot de $V_s \times S_w$

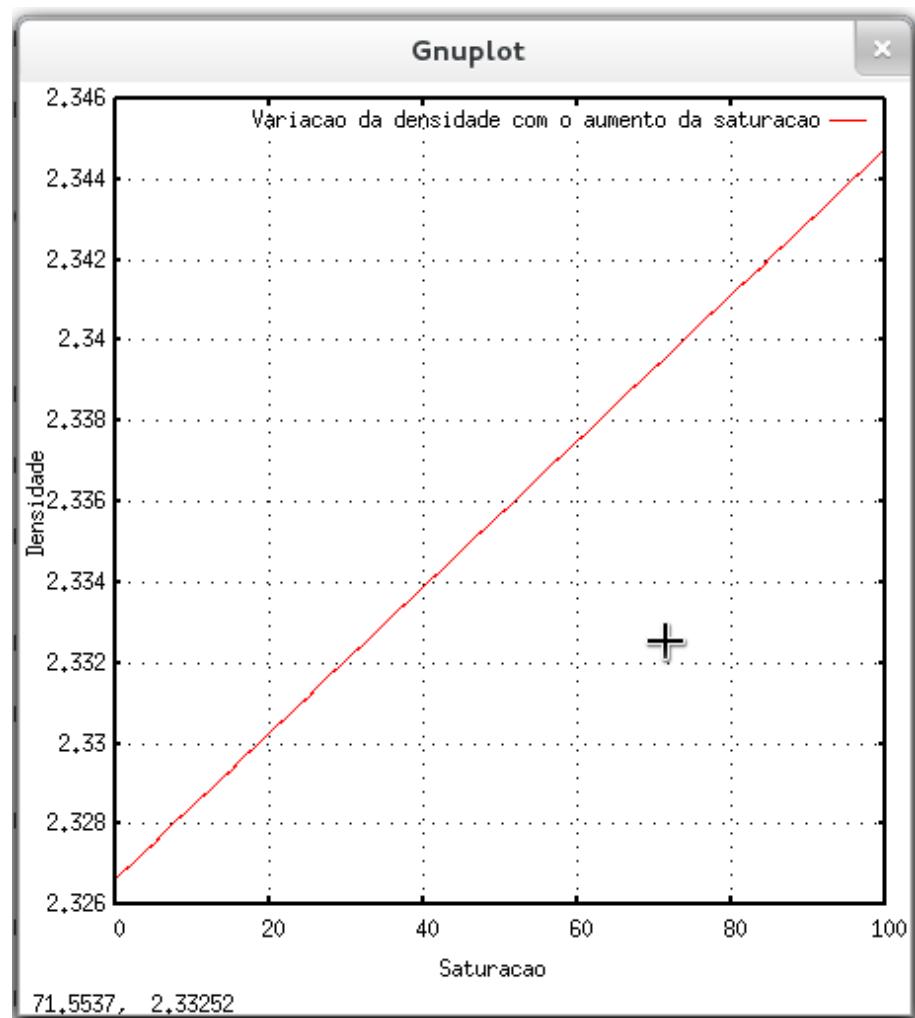


Figura 7.7: Gráfico gerado pelo programa de $\rho x S_w$

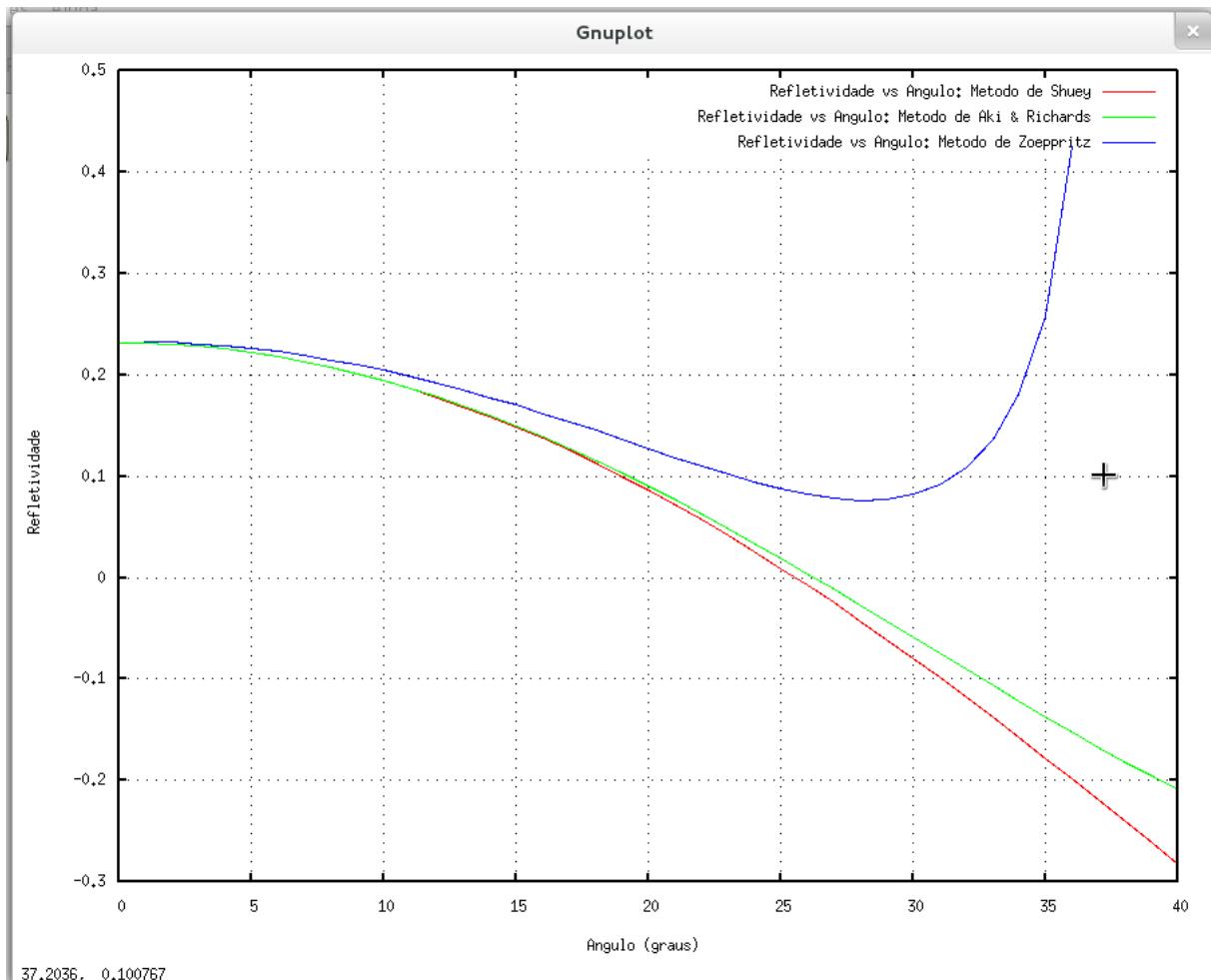


Figura 7.8: Gráfico gerado pelo Gnuplot representando a modelagem AVO

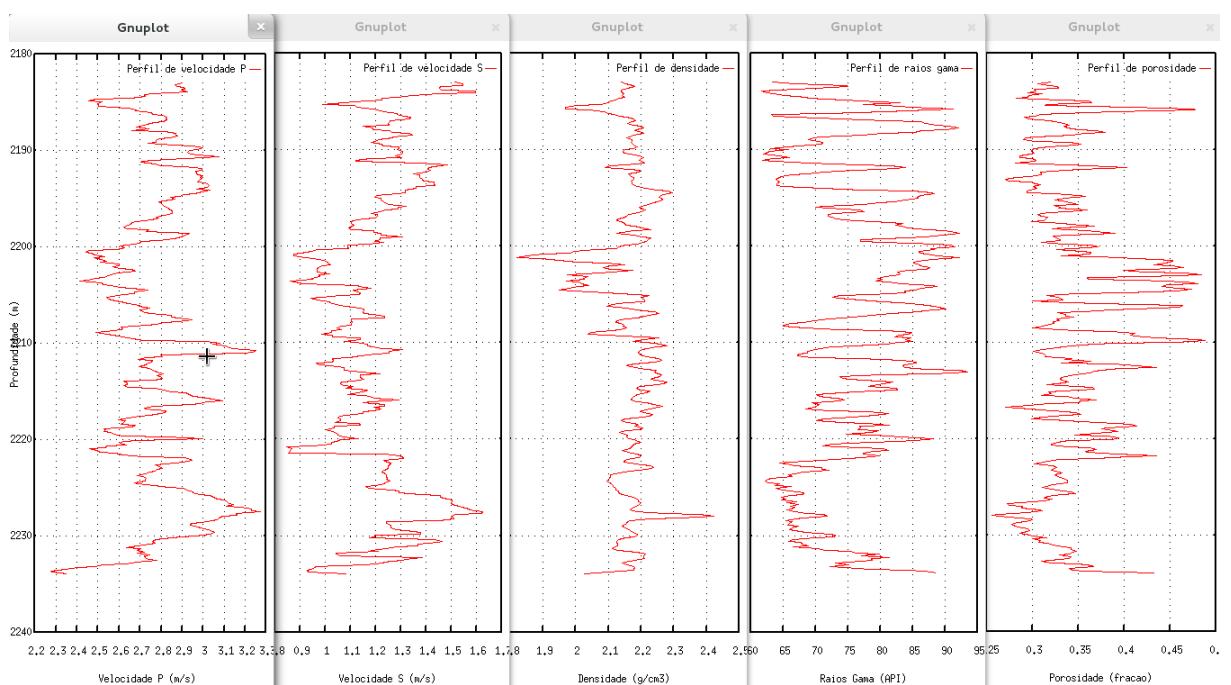


Figura 7.9: Perfis gerados pelo Gnuplot

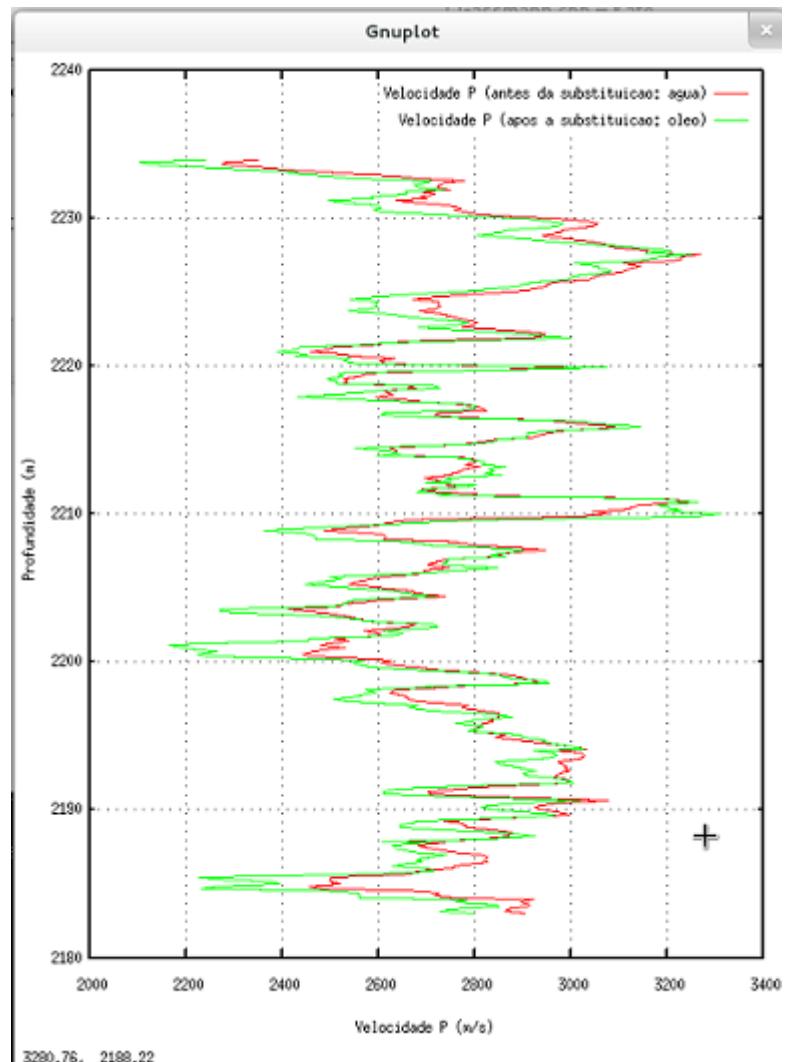


Figura 7.10: Perfil gerado com a substituição de fluidos

Capítulo 8

Documentação

Apresenta-se neste capítulo a documentação de uso do software “SIMULAÇÃO DE SUBSTITUIÇÃO DE FLUIDOS E MODELAGEM AVO – Versão 1.0”.

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 rodar o software

Previvamente, deve-se assegurar que os dados de perfis de entrada estejam na mesma pasta que o programa. Também é importante ressaltar que os dados de entrada sejam fisicamente coerentes, caso contrário resultados incompatíveis serão obtidos.

Para a geração de gráficos, o software livre “Gnuplot” deve estar instalado no computador. Com isso, abra o terminal, vá ao diretório onde está o código, compile e execute o programa. Após executar, os procedimentos descritos abaixo são apresentados.

No cabeçalho do programa, tem-se o nome, a universidade, a data, a versão e os autores do programa. Abaixo o usuário deve optar por realizar uma substituição de fluidos com modelagem AVO, substituição de fluidos a partir de dados de perfis ou obter ajuda.

Caso digite 1, a primeira opção será executada:

- Primeiramente serão solicitados ao usuário dados de entrada dos fluidos, água e óleo e da rocha e seus componentes principais.
- A matriz rochosa nesse trabalho foi modelada supondo ser composta por dois minerais, então serão solicitado propriedades elásticas referentes a esses dois minerais bem como a fração em volume de cada um na matriz da rocha. A Tabela 3.1 apresenta propriedades de alguns minerais constituintes das rochas e pode ser usada como auxílio nessa etapa.

- Caso a soma das frações dos dois minerais não seja igual a um, uma mensagem de erro será exibida já que seria fisicamente impossível nessa modelagem. Fica a cargo de o usuário prosseguir ou reiniciar a execução do programa.
- Após a entrada de dados o programa calculará as propriedades dos fluidos e rocha efetivos e utilizará as equações de Gassmann para realizar a substituição de fluidos. Com isso serão gerados três gráficos de propriedades do meio com o aumento da saturação de água.
- As propriedades calculadas para a rocha e fluido efetivos são salvos nos arquivos externos Resultados_rocha.txt e Resultados_fluido.txt respectivamente. Ambos podem ser localizados na mesma pasta do programa.
- O próximo passo é fornecer ao programa os dados para a modelagem AVO.
- De posse de todos os dados, será realizada a modelagem AVO para os três métodos implementados e um gráfico de refletividade por ângulo será gerado e a simulação encerrada.

Caso digite 2, a segunda opção será executada:

- Será solicitado ao usuário o nome do arquivo e formato com os dados de perfilagem. Os dados devem ter a mesma estrutura de colunas que o mostrado na figura xxxx. Após carregados, os perfis são plotados.
- Após o carregamento dos perfis, serão realizados os mesmos procedimentos apresentados nos itens a, b e c do caso anterior.
- Os perfis carregados são salvos na pasta do programa juntamente com os dados calculados como descrito no item e do caso acima.
- Com isso, será realizada a substituição de fluidos de água para óleo no perfil carregado. Como dito, partimos do pressuposto que o perfil carregado é uma formação totalmente saturada com água e que passará a ser totalmente saturada com óleo. O comportamento da velocidade na camada com a mudança do fluido antes e depois pode ser visto no gráfico gerado, encerrando assim esta simulação.

Caso digite 3, um menu com opções de ajuda será exibido ao usuário.

Caso digite qualquer número diferente dos relacionados às opções acima, uma mensagem de erro será exibida e o programa reiniciado.

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.

8.2.2 Documentação usando doxygen

A documentação do código do software foi feita usando o padrão JAVADOC. Depois de documentar o código, o software doxygen foi usado para gerar a documentação do desenvolvedor no formato html. O software doxygen lê os arquivos com os códigos (*.h e *.cpp) e gera uma documentação muito útil e de fácil navegação no formato html. A Figura 8.1 apresenta a listagem de classes gerado pelo doxygen.

The screenshot shows the 'My Project' doxygen-generated documentation. At the top, there is a navigation bar with tabs: 'Main Page' (selected), 'Classes' (highlighted in blue), and 'Files'. Below the tabs are sub-links: 'Class List' (selected), 'Class Index', 'Class Hierarchy', and 'Class Members'. The main content area is titled 'Class List' and contains the following text: 'Here are the classes, structs, unions and interfaces with brief descriptions:'. Below this, there is a table listing various classes with their descriptions:

N std	Esta classe representa o fluido água e contém atributos e métodos deste. Herdeira de CFLuido
C Agua	Classe com atributos e métodos para a modelagem AVO
C Avo	Classe que contém as equações de Gassmann para a substituição de fluidos
C CFLuido	Esta classe representa um fluido e é base para os fluidos utilizados no simulador
C CGas	Classe que contém os atributos e métodos deste. Herdeira de CFLuido
C CGassmann	Classe que apresenta um pequeno manual do programa
C CHelp	Classe que representa a mistura de fluidos, chamado fluido efetivo e contém seus atributos e métodos
C CMistura	Classe que representa o fluido óleo e contém atributos e métodos deste. Herdeira de CFLuido
C COleo	Classe com atributos e métodos para trabalhar com os perfis
C CPerfil	Classe que representa a rocha, com seus atributos e métodos
C CRocha	Classe de interface para acesso ao programa gnuplot
C Gnuplot	Erros em tempo de execução
C GnuplotException	

Figura 8.1: Listas

8.3 Sugestões para trabalhos futuros

- Realizar a substituição de fluidos entre água e gás. As funções para o cálculo das propriedades do gás já estão implementadas bastando realizar os mesmos procedimentos feitos entre água e óleo.

- Procurar na literatura funções que apresentem resultados mais coerentes para o módulo de incompressibilidade do gás. As equações de Batzle-Wang implementadas não apresentaram resultados satisfatórios.
- Melhorar o sistema de ajuda.
- Implementar mais equações de refletividade no programa. No programa já está implementada os três métodos mais conhecidos, no entanto existe outros na literatura.
- Otimizar o código.
- Implementar mais atributos AVO, como meios interpretativos para a física das rochas.
- Modelar o óleo como óleo vivo.
- Modelar a matriz das rochas como compostas por mais de dois minerais.

Capítulo 9

Referências Bibliográficas

- MAKVO, G. et al. The Rock Physics Handbook. Second Edition. Nova York: Cambridge University Press, 2009. 503 p.
- SMITH, T. D. et al. Gassmann Fluid Substitution: A Tutorial. GEOPHYSICS, vol. 68, n^o 2 (march-april 2003); p. 430–440.
- NETO, I. A. L. Análise De Avo E Estudo De Viabilidade Sísmica 4d Para Reservatórios Carbonáticos.. 111 p. Dissertação. UENF. Setembro, 2008.
- BORÇOI, D. R. Análise Da Assinatura De Avo Em Reservatórios Delgados: Modelagem e Inversão. Monografia. UENF. Julho, 2006. • MARQUES, D. S. Simulação Numérica Da Resposta Sísmica De Modelos Geológicos De Reservatórios De Petróleo E Gás. Monografia. UFF. Dezembro de 2011. • BATZLE, B.; WANG Z. Seismic properties of pore fluids. GEOPHYSICS, vol. 57, N^o 11 (NOVEMBER 1992); P. 1396-1408 .
- BUENO, A. D. Programação Orientada a Objeto com C++. 2 ed. Novatec Editora. 2013.
- BACON, M. et al. 3D Seismic Interpretation. Cambridge. 2003.
- Notas de aula da disciplina Interpretação Integrada Geologia – Geofísica.
- BATZLE, M. & WANG, Z. Seismic Properties of Pore Fluid. Geophysics. Vol 57 n^o11. 1992. Pag 1396-1408.
- AVSETH, P. et al. Quantitative Seismic Interpretation: Applying Rock Physics Tools to Reduce Interpretation Risk. Cambridge. 2005.
- PEREIRA, B. Análise Quantitativa de Dados Sísmicos: Inversão e AVO. Dissertação. Universidade de Lisboa. 2014.

Referências

Referências Bibliográficas

- [e Patrick W. Daly, 1995] e Patrick W. Daly, H. K. (1995). *A Guide to Latex 2e*. Addison-Wesley, New York, 2 edition. 98, 99
- [Grossens et al., 1993] Grossens, M., Mittelbach, F., and Samarin, A. (1993). *Latex Companion*. Addison-Wesley, New York. 98, 99
- [Karger, 2004] Karger, A. (2004). *O Tutorial de Lyx*. LyX Team - <http://www.lyx.org>. 98, 99
- [Knuth, 1986] Knuth, D. E. (1986). *The Texbook*. Addison-Wesley. 98, 99
- [Lamport, 1985] Lamport, L. (1985). *Latex - A Document Preparation System*. Addison-Wesley. 98, 99
- [LyX-Team, 2004a] LyX-Team, editor (2004a). *Extended LyX Features*. LyX Team - <http://www.lyx.org>. 98, 99
- [LyX-Team, 2004b] LyX-Team, editor (2004b). *The LyX User's Guide*. LyX Team - <http://www.lyx.org>. 98, 99
- [Steding-Jessen, 2000] Steding-Jessen, K. (2000). *Latex demo: Exemplo com Latex 2e*. 98, 99

Capítulo 10

Título do Apêndice

Descreve-se neste apêndice ...

- Os anexos ou apêndices contém material auxiliar. Por exemplo, tabelas, gráficos, resultados de experimentos, algoritmos, códigos e simulações.
- Um apêndice pode incluir assuntos mais gerais (geral demais para estar no núcleo do trabalho) ou mais específicos (detalhado demais para estar no núcleo do trabalho).
- Pode conter um artigo de auxílio fundamental ao trabalho.
- Pode conter artigos publicados.
- [tudo aquilo que for importante para a tese mas não essencial, deve ser colocado em apêndices]
- [como exemplo, revisão de metodologias, técnicas, modelos matemáticos, ítems desenvolvidos por terceiros]
- [algoritmos e programas devem ser colocados no apêndice]
- [imagens detalhadas de programas desenvolvidos devem ser colocados no apêndice]

10.1 Sub-Título do Apêndice

.....conteúdo..

Capítulo 11

Título do Apêndice

Descreve-se neste apêndice ...

[tudo aquilo que for importante para a tese mas não essencial, deve ser colocado em apêndices]

[como exemplo, revisão de metodologias, técnicas, modelos matemáticos, ítems desenvolvidos por terceiros]

[algoritmos e programas devem ser colocados no apêndice]

[imagens de programas desenvolvidos/utilizados devem ser colocados no apêndice]

11.1 Roteiro Para Uso do Sistema de Citações Com Banco de Dados .bib

O sistema de referências usando bibtex é extremamente simples e muito prático. O mesmo é composto de uma base de dados (um arquivo .bib que contém a lista de referencias a ser utilizada). Por exemplo, o arquivo andre.bib, inclui referencias bibliograficas no formato bib (de uma olhada agora no arquivo andre.bib usando um editor de texto como o emacs). A seguir, você deve incluir no arquivo do lyx, o nome de sua base de dados. Finalmente, você precisa incluir as referencias cruzadas.

Veja a seguir um roteiro:

1. Você deve fazer uma copia do arquivo andre.bib com seu nome, e a seguir usar um editor qualquer (mas preferencialmente o emacs) para incluir suas referências bibliográficas. Ou seja, inclua no arquivo seuNome.bib todas as citações e referências bibliográficas a serem incluídas em sua tese (tudo que você leu, e que pode ser incluído na citação da tese e de outros artigos. É sua base de dados de citações).
 - (a) Você pode incluir ítems no arquivo .bib que não irão fazer parte da tese, mas poderão ser citadas em artigos futuros.
2. Para fazer uma citação é necessário incluir no arquivo do lyx um "Insert-> Lists & Toc->Bibtex reference". Vai aparecer um diálogo pedindo para você incluir o nome do arquivo com a base de dados de citações (digite seuNome.bib).
3. Finalmente, faça referencias cruzadas usando o ítem de menu "Insert Cross-Reference".
4. Aqui um exemplo, vou citar material sobre LyX e Latex. Veja maiores informações sobre latex em [Grossens et al., 1993, Knuth, 1986, Steding-Jessen, 2000, e Patrick W. Daly, 1985, LyX-Team, 2004a, Karger, 2004, LyX-Team, 2004b].

11.1.1 Citações no meio do texto

Segundo [Grossens et al., 1993] asldkjasldkajsdlkajsdlaksjd

Segundo [Grossens et al., 1993, Knuth, 1986, Steding-Jessen, 2000, e Patrick W. Daly, 1995, Lamport, 1985, LyX-Team, 2004a, Karger, 2004, LyX-Team, 2004b] asldkjasldkajsdlkajsdlaksjd

11.1.2 Incluir nas referências bibliográficas (fim do documento), mas não citar

asldkjasldkajsdlkajsdlaksjd

asldkjasldkajsdlkajsdlaksjd

11.2 Informações adicionais

- Manuais do LYX (precisa ler!)
- <http://chem-e.org/comando-cite-e-citeonline-no-abntex/>
- <http://win.ua.ac.be/~nschloe/content/bibtex-how-cite-website.>
- <http://chem-e.org/comando-apud-e-apudonline-no-abntex/.>
- http://en.wikibooks.org/wiki/LaTeX/Bibliography_Management

Referências Bibliográficas

- [e Patrick W. Daly, 1995] e Patrick W. Daly, H. K. (1995). *A Guide to Latex 2e*. Addison-Wesley, New York, 2 edition. 98, 99
- [Grossens et al., 1993] Grossens, M., Mittelbach, F., and Samarin, A. (1993). *Latex Companion*. Addison-Wesley, New York. 98, 99
- [Karger, 2004] Karger, A. (2004). *O Tutorial de Lyx*. LyX Team - <http://www.lyx.org>. 98, 99
- [Knuth, 1986] Knuth, D. E. (1986). *The Texbook*. Addison-Wesley. 98, 99
- [Lamport, 1985] Lamport, L. (1985). *Latex - A Document Preparation System*. Addison-Wesley. 98, 99
- [LyX-Team, 2004a] LyX-Team, editor (2004a). *Extended LyX Features*. LyX Team - <http://www.lyx.org>. 98, 99
- [LyX-Team, 2004b] LyX-Team, editor (2004b). *The LyX User's Guide*. LyX Team - <http://www.lyx.org>. 98, 99
- [Steding-Jessen, 2000] Steding-Jessen, K. (2000). *Latex demo: Exemplo com Latex 2e*. 98, 99
- ...

Índice Remissivo

- Análise orientada a objeto, 23
- AOO, 23
 - Casos de uso, 4
 - Cenário, 4
 - Citações, 98
 - Citações no meio do texto, 99
 - colaboração, 27
 - comunicação, 27
 - Concepção, 3
 - Controle, 34
- Diagrama de colaboração, 27
- Diagrama de componentes, 36
- Diagrama de execução, 37
- Diagrama de máquina de estado, 29
- Diagrama de sequência, 25
- Efeitos do projeto nos métodos, 35
- Elaboração, 7
- especificação, 3
- Especificações, 3
- estado, 29
- Eventos, 25
- Implementação, 38
- métodos, 35
- Mensagens, 25
- modelo, 35
- Plataformas, 34
- POO, 35
- Projeto do sistema, 33
- Projeto orientado a objeto, 35
- Protocolos, 33
- Recursos, 33