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

PROJETO DE ENGENHARIA
DESENVOLVIMENTO DO SOFTWARE:
SIMULADOR DE CURVAS IPR UTILIZANDO MODELOS EMPÍRICOS EM POÇOS VERTICAIS

TRABALHO DA DISCIPLINA PROGRAMAÇÃO PRÁTICA

Versão 1:

ATILA JUNIOR
GIOVANNA MASSARDI
MARCELO BERNARDO
Prof. André Duarte Bueno

MACAÉ - RJ

Dezembro - 2023

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

No presente projeto de engenharia, desenvolve-se o software de Simulação de Curvas IPR (*Inflow Performance Relationship*) utilizando Modelos Empíricos em Poços verticais, um software aplicado a engenharia de petróleo e que utiliza o paradigma da orientação a objetos.

Este software tem como finalidade obter curvas de IPR no regime pseudopermanente a partir de dados inseridos pelo usuário. Para isso, é necessário que o índice de produtividade seja calculado com base nos dados de pressão e vazão ou propriedades do reservatório. Dessa forma, os modelos empíricos Linear, Fetkovich, Vogel e Vogel Generalizado poderão ser escolhidos para realização do cálculo da pressão de fundo e respectivas vazões bem como avaliar reservatórios estratificados a partir dos cálculos resultantes. Tais valores serão mostrados ao usuário juntamente com um gráfico com a curva de IPR e estas informações poderão ser salvas em disco.

1.1 Escopo do problema

A curva de IPR, que pode ser chamada de curva de influxo, curva de pressão disponível ou curva do índice de produtividade, é uma representação gráfica que descreve como a pressão disponível no fundo de um poço de petróleo ou gás varia em relação à taxa de fluxo de fluidos. Essas medidas são tomadas na profundidade em que o reservatório foi perfurado em um momento específico durante a vida útil do campo. Em outras palavras, essa curva fornece informações cruciais sobre como a pressão no poço reage quando se está produzindo hidrocarbonetos.

Através da análise dessa curva, as decisões estratégicas podem ser tomadas para maximizar a eficiência e recuperação de petróleo ou gás, além de demonstrar os cenários nos quais intervenções deverão ser realizadas assim como definir quando perfurar novos poços. Com isso, é possível analisar quando há possíveis limitações no desempenho do poço. Ademais, esta ferramenta é frequentemente utilizada em simulações de reservatório a fim de prever o comportamento futuro e auxiliar em um planejamento de longo prazo.

O cálculo da IPR pode sofrer variações dependendo das características do reservatório e se estes dados são conhecidos. Logo, alguns modelos matemáticos podem ser aplicados para obter a curva de IPR e inúmeros métodos podem ser escolhidos com base nos regimes de escoamento, tipos de fluxo e fluidos existentes. Portanto, estas equações e suas respectivas análises podem ser encontradas na literatura e, facilitar o cálculo e plotagem das curvas é algo extremamente relevante para simulação de reservatórios e para o entedimento de elevação e escoamento de fluidos. Neste trabalho será abordado apenas os modelos matemáticos descritos sob o regime pseudo-permanente, os quais pode-se destacar: IPR Linear, Fetkovich, Vogel e Vogel Generalizada.

1.2 Objetivos

Os objetivos deste projeto de engenharia são:

- Objetivos gerais:
  - Utilizar modelos empíricos e equações matemáticas propostas na literatura para o cálculo das pressões de poço e suas respectivas vazões ao longo da vida útil de um reservatório.
  - Plotar curvas de IPR em escoamentos monofásico e bifásico utilizando os modelos Linear, Vogel, Fetkovich e Vogel Generalizado considerando regime pseudopermanente a partir do sofware externo Gnuplot.
- Objetivos específicos:
  - Permitir que o usuário escolha qual modelo irá utilizar para realizar o cálculo dos parâmetros da curva de IPR.
  - Criar o software de modo que a curva de IPR possa ser plotada independente da existência dos parâmetros de reservatório.
  - A partir do cálculo dos modelos empíricos, analisar os tipos de IPR: Linear, Generalizada e Reservatórios Estratificados.

Capítulo 2

Especificação

Nesta seção do projeto de engenharia, é apresentada a especificação do software a ser desenvolvido para aplicação em sistemas de modelagem de curvas IPR, utilizado modelos empíricos em poços verticais. Para a indústria de Óleo e Gás, este processo é altamente relevante, pois ao dispor de um software capaz de construir curvas IPR (Figura 2.1), será possível estimar características importantes acerca dos poços analisados, dentre elas a sua produtividade. Além disso, o projeto visa estimar e fornecer ao usuário o momento em que será necessário utilizar-se de métodos de recuperação terciária, como a elevação artificial.

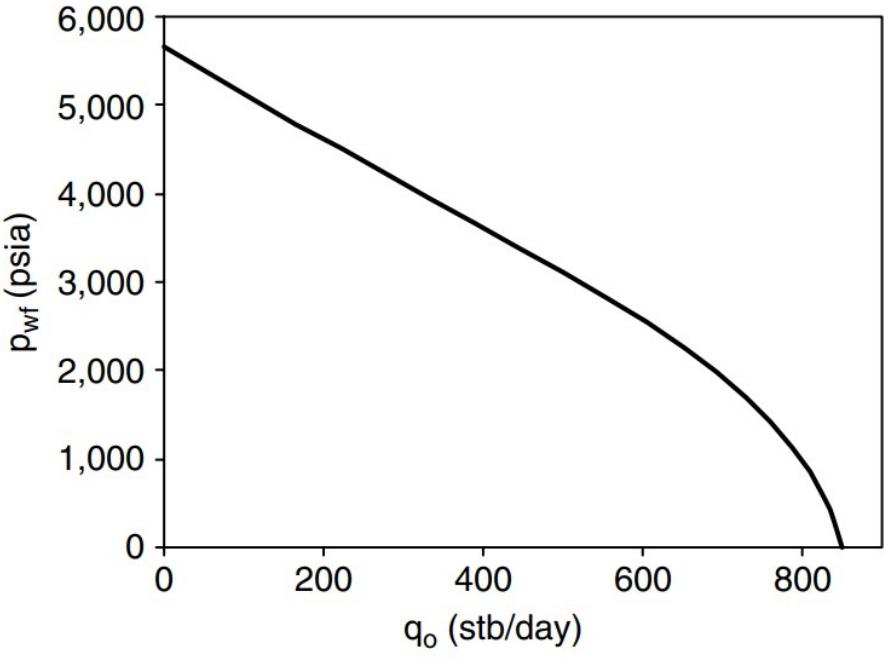

Figura 2.1: Curva IPR típica de um reservatório de óleo

2.1 Nome do sistema/produto

Nome	Simulação de Curvas IPR Utilizando Modelos Empíricos para Poços Verticais
Componentes principais	- Implementação de Modelos Empíricos de IPR: Será desenvolvida uma gama diversificada de modelos empíricos, os quais representarão com precisão os padrões de produção em poços verticais de petróleo. - Entrada Customizada: O software possibilitará aos usuários inserirem parâmetros específicos do poço, tais como propriedades dos fluidos e do reservatório, para conduzir simulações altamente personalizadas. - Cálculos de Produtividade: O software executará cálculos para estimar a produtividade prevista em um poço vertical, com base nos modelos empíricos selecionados e nos dados fornecidos. - Visualização Gráfica: O software será equipado com recursos de visualização gráfica, a fim de exibir de forma clara e compreensível os resultados, facilitando a análise e interpretação dos resultados.
Missão	A missão subjacente a este software é disponibilizar aos estudantes do curso de Elevação e Escoamento e à indústria uma ferramenta de simulação de curvas IPR de alta qualidade. Por meio de uma interface intuitiva e da capacidade de fornecer estimativas precisas de produtividade, o software tem por objetivo apoiar engenheiros, pesquisadores e profissionais do setor em suas decisões e na otimização da produção de poços verticais. Fundindo a precisão dos modelos com a flexibilidade de entrada dos dados, essa solução almeja tornar-se uma ferramenta indispensável para avaliação e planejamento de operações em poços de petróleo.

2.2 Especificação

O presente projeto tem como objetivo a criação de um software avançado de simulação de curvas IPR (*Inflow Performance Relationship*), utilizando modelos empíricos para poços verticais de petróleo. A simulação de curvas IPR permite estimativas precisas da produtividade dos poços e auxiliam na tomada de decisões estratégicas. O software a ser desenvolvido busca proporcionar uma ferramenta poderosa para engenheiros e profissionais do setor, que desejam analisar e otimizar a produção de poços verticais de maneira eficiente e eficaz.

O fundamento deste projeto conta com a aplicação de modelos empíricos conhecidos, incluindo as abordagens de Fetkovich, Vogel, e outras metodologias de IPR generalizada. Esses modelos têm sido fundamentais na avaliação da performance de poços de petróleo, considerando variáveis complexas como propriedades do reservatório, características dos fluidos e condições operacionais. O software que está sendo desenvolvido permitirá aos usuários empregarem esses modelos com facilidade, fornecendo uma plataforma de simulação confiável e precisa.

Os modelos empíricos como Fetkovich e Vogel têm sido amplamente utilizados para representar a relação entre a vazão de óleo e a pressão na entrada do poço. O Fetkovich é reconhecido pela sua aplicabilidade a diferentes regimes de fluxo, enquanto o modelo Vogel oferece uma abordagem simplificada para estimar a produtividade. Além disso, abordar-se-á a adaptação desses modelos para situações específicas, como poços estratificados, onde a heterogeneidade do reservatório é levada em consideração para uma simulação mais precisa.

Em suma, o projeto em questão fornece uma solução de simulação para a análise de curvas IPR em poços verticais. Com modelos empíricos consagrados, como Fetkovich e Vogel, juntamente com adaptações para poços estratificados e outras variações, o software busca simplificar e aprimorar a avaliação da produtividade de poços de petróleo. Com isso, engenheiros e profissionais do setor terão uma ferramenta valiosa para otimizar a produção e tomar decisões embasadas no setor de exploração e produção de petróleo.

O software será desenvolvido utilizando o conceito de programação orientada a objeto, interface intuitiva e utilizará o software externo Gnuplot para gerar e salvar os gráficos.

O software tem licença GPL 2.0.

2.2.1 Requisitos funcionais

Apresenta-se a seguir os requisitos funcionais.

RF-01	O usuário deve ter liberdade para escolher quais parâmetros de entrada utilizar, fornecendo-os através do teclado.

RF-02	O usuário deve escolher o modelo adequado para seu problema de engenharia específico.

RF-03	O usuário poderá plotar suas curvas em um gráfico utilizando o software externo Gnuplot. O gráfico poderá ser salvo como imagem.

2.2.2 Requisitos não funcionais

RNF-01	Os cálculos devem ser feitos utilizando-se formulações/modelos matemáticos conhecidos na literatura.

RNF-02	Usuário deve ter conhecimento básico e prévio de assuntos sobre Elevação e Escoamento para escolha do método mais adequado.

RNF-03	O programa deverá ser multi-plataforma, podendo ser executado em *Windows*, *GNU/Linux* ou *Mac*.

2.2.3 Casos de uso

Nesta seção, apresenta-se a Tabela 2.1 que especifica um caso de uso geral para o software. O diagrama de caso de uso geral (Figura 2.2) e o diagrama de caso de uso específico (Figura 2.3) para casos onde o reservatório é estratificado, isto é, cada camada possui diferentes propriedades.

Tabela 2.1: Caso de uso

Nome do caso de uso:	Simulação de curvas IPR de um reservatório.
Resumo/descrição:	Determinação das curvas IPR através das propriedades dos fluidos, rocha e poço para análise do comportamento do reservatório ao longo do tempo.
Etapas:	1. Definição do número de camadas. 2. Entrada de dados do reservatório, fluidos e poço via teclado. 3. Definir método adequado para a simulação. 4. Cálculo da vazão ao longo do tempo. 5. Gerar gráfico. 6. Analisar resultados.
Cenários alternativos:	Inserir valores negativos ou incompatíveis com a ordem de grandeza de um reservatório real.

2.2.4 Diagrama de caso de uso geral

O diagrama de caso de uso geral da Figura 2.2 mostra o usuário interagindo com o software para obter as curvas de IPR do reservatório. Nesse caso, o usuário insere os dados do problema (propriedades dos fluidos, reservatório e poço), selecionando o modelo mais adequado para o problema que ele possuir e analisando os resultados obtidos.

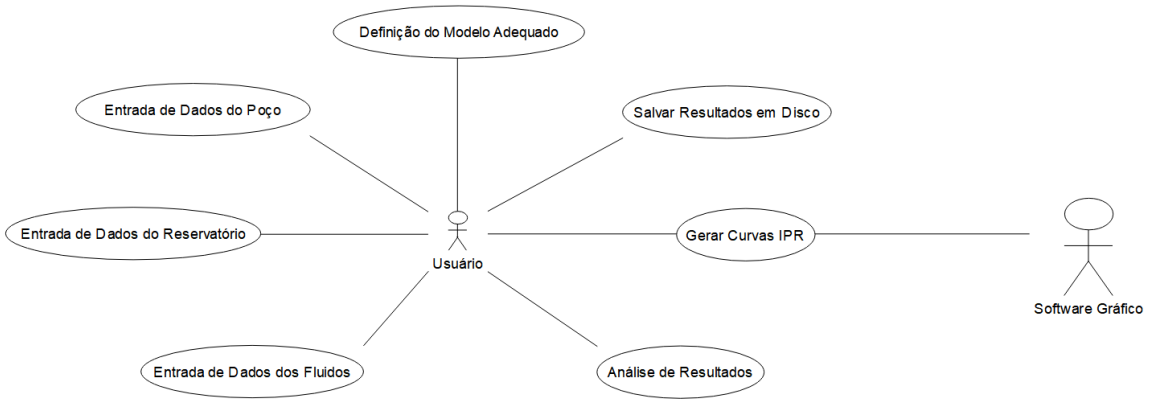

Figura 2.2: Diagrama de caso de uso – Caso de uso geral

2.2.5 Diagrama de caso de uso específico

O caso de uso para reservatórios homogêneos é descrito na Figura 2.2 e na Tabela 2.1. Para reservatórios estratificados o processo é detalhado na Figura 2.3. O usuário entrará com o número de camadas do reservatório e com as propriedades de cada camada do reservatório para obtenção das curvas.

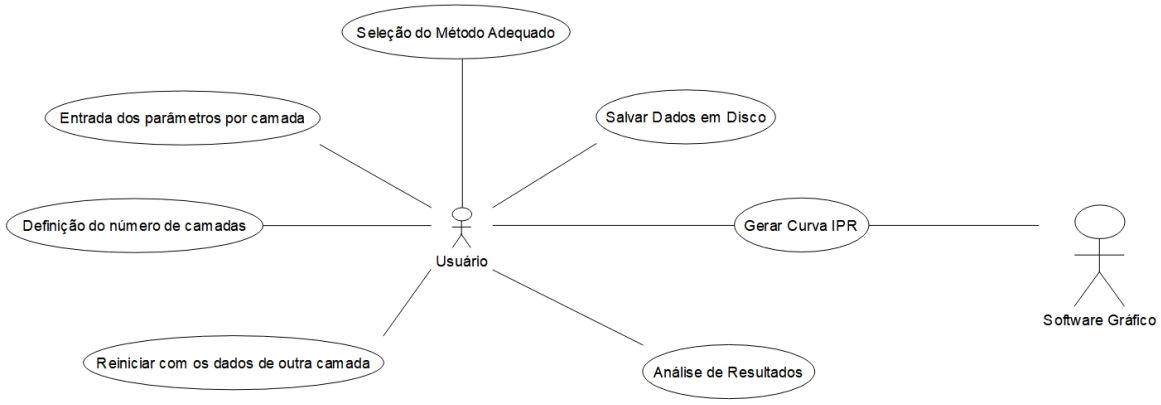

Figura 2.3: Diagrama de caso de uso específico – Reservatórios Estratificados

Capítulo 3

Elaboração

Neste capítulo, após a definição dos objetivos, da especificação do software e da montagem dos primeiros diagramas de caso de uso, será apresentada a etapa da elaboração que aborda estudos e análises de conceitos relacionados ao sistema desenvolvido, isto é, análise de domínio e identificação de pacotes. Neste sentido, será empregada uma análise de requisitos de modo a ajustá-los aos requisitos iniciais a fim de desenvolver um sistema útil, que atenda às necessidades do usuário e que possa ser, possivelmente, reutilizada e estendida.

3.1 Análise de domínio

O Software a ser desenvolvido aborda formas de estimar a produtividade de um poço vertical de petróleo através de modelagem de curvas IPR, utilizando modelos empíricos. O fato de se explorar este tema é de fundamental relevância no campo da produção de petróleo, pois traz informações importantes em relação ao rendimento e vida útil que o poço analisado terá. A capacidade de entrega do reservatório é definida como a taxa de produção alcançável de óleo ou gás a partir do reservatório em uma dada pressão no fundo do poço. É um fator importante que afeta a sua capacidade de entrega. Por sua vez, a capacidade de entrega do reservatório determina os tipos de completação e os métodos de elevação artificial a serem utilizados. Dessa forma, um conhecimento completo da produtividade do reservatório é essencial para engenheiros de produção. A produtividade do reservatório depende de diversos fatores, como:

- Pressão do Reservatório: A pressão do reservatório é um fator crítico que influencia diretamente a produtividade. Uma pressão mais alta pode impulsionar a migração do petróleo ou gás em direção ao poço, aumentando a taxa de produção.

- Espessura de *pay zone*: A *pay zone* se refere à camada do reservatório que contém o petróleo ou gás. Quanto mais espessa essa camada, mais espaço há para armazenar hidrocarbonetos, o que pode aumentar a produtividade.

- Permeabilidade: A permeabilidade está relacionada à capacidade do reservatório de permitir que o petróleo ou gás fluam através dele. Uma maior permeabilidade facilita a movimentação dos fluidos, contribuindo para uma maior produtividade.

- Propriedades dos fluidos do reservatório: As propriedades dos fluidos, como a viscosidade do petróleo e a composição do gás, influenciam a facilidade com que os fluidos podem ser extraídos. Fluidos mais viscosos podem dificultar o fluxo, reduzindo a produtividade.

- Permeabilidade Relativa do Reservatório: A permeabilidade relativa é a medida de quão facilmente os diferentes fluidos (por exemplo, óleo, gás e água) se movem no reservatório. Se a permeabilidade relativa do óleo for baixa em relação ao gás ou água, o petróleo pode ter dificuldades em ser deslocado, afetando a produtividade.

O software desenvolvido terá a capacidade de fornecer ao cliente resultadosda produtividade e vida útil do reservatório, levando em consideração todos os requisitos, especificações e conceitos de Engenharia de Petróleo apresentados na Introdução.

3.2 Formulação teórica

Nesta seção, apresenta-se a formulação teórica dos conceitos fundamentais abordados ao longo deste projeto. A mesma tem como base o livro[6]

3.2.1 *Inflow Performance Relationship*

A curva IPR (*Inflow Performance Relationship*) é usada para avaliar a capacidade de entrega do reservatório na engenharia de produção. É uma apresentação gráfica da relação entre a pressão de fluxo no fundo do poço e a taxa de produção de líquidos. A magnitude da inclinação da curva IPR é chamada de "índice de produtividade" (J):

$$J = \frac{q}{(p_e - p_{wf})} \tag{3.1}$$

.

As curvas IPR de poços são geralmente construídas usando modelos de vazão de reservatórios, que podem ser de base teórica ou empírica.

3.2.2 IPR Linear

A IPR Linear é aplicada na suposição de fluxo líquido monofásico e, dessa forma, funciona para zonas do reservatório acima do ponto de bolha ou para óleos subsaturados. As equações definem o índice de produtividade (J) (equação 3.2) para a pressão de fundo

de poço acima da pressão do ponto de bolha:

$$J^* = \frac{q}{(p_e - p_{wf})} = \frac{kh}{141.2B_o\left(\frac{1}{2}ln\frac{4A}{\gamma C_A r_w^2 + S}\right)} \tag{3.2}$$

Como o índice de produtividade (J) acima da pressão do ponto de bolha é independente da taxa de produção, a curva IPR para um reservatório monofásico (líquido) é uma linha reta traçada da pressão do reservatório até a pressão do ponto de bolha. Se a pressão do ponto de bolha for 0 $psig$, o fluxo aberto absoluto (AOF) é o índice de produtividade (J) vezes a pressão do reservatório .

$$AOF = JPe \tag{3.3}$$

3.2.3 Reservatórios Bifásicos

Acima da pressão de bolha, o óleo contido no reservatório se encontra subsaturado, ou seja, o gás contido no óleo se encontra todo dissolvido, fazendo com que a IPR apresente um comportamento linear. Abaixo da pressão do ponto de bolha, o gás em solução escapa do óleo e se torna gás livre.

O gás livre ocupa alguma porção do espaço poroso, o que reduz o fluxo de óleo. Este efeito é quantificado pela permeabilidade relativa reduzida. Além disso, a viscosidade do óleo aumenta à medida que o conteúdo do gás em solução diminui. A combinação do efeito de permeabilidade relativa e do diminuição da viscosidade resulta em menor taxa de produção de petróleo a uma determinada pressão de fundo de poço. Isto faz com que a curva IPR se desvie da tendência linear abaixo da pressão do ponto de bolha. Quanto menor a pressão, maior o desvio. Se a pressão do reservatório estiver abaixo da pressão inicial do ponto de bolha, existe fluxo bifásico de petróleo e gás em todo o domínio do reservatório.

Apenas equações empíricas estão disponíveis para modelar IPR de reservatórios bifásicos. Essas equações empíricas incluem a equação de Vogel (1968) e a equação de Fetkovich (1973). A equação de Vogel ainda é amplamente utilizada na indústria.

3.2.4 Equação de Vogel

A equação de Vogel é descrita por:

$$q = q_{max}\left[1 - 0.2\left(\frac{p_{wf}}{\overline{p}}\right) - 0.8\left(\frac{p_{wf}}{\overline{p}}\right)^2\right] oup_{wf} = 0.125\overline{p}\left[\sqrt{81 - 80\left(\frac{q}{q_{max}}\right)} - 1\right] \tag{3.4}$$

,

onde $qmax[ft^3/d]$ é uma constante empírica e seu valor representa a capacidade máxima que o reservatório pode entregar, ou AOF. Para fluxo pseudo-permanente pode

ser calculado por:

$$q_{max} = \frac{J * \overline{p}}{1.8} \tag{3.5}$$

3.2.5 Equação de Fetkovich

A equação de Fetkovich é definida por:

$$q = q_{max}\left[1 - \left(\frac{p_{wf}}{\overline{p}}\right)^2\right]^n ouq = C\left(\overline{p}^2 - p_{wf}^2\right)^n \tag{3.6}$$

,

onde C e n são constantes empíricas relacionadas a vazão máxima, dada por: $C = \frac{q_{max}}{\overline{p^{2n}}}$.

3.2.6 IPR Generalizada

Acima do ponto de bolha a IPR apresenta comportamento linear e a vazão na pressão de bolha é dada por:

$$q = J^*(\overline{p} - p_b) \tag{3.7}$$

Baseado na equação de Vogel, a vazão adicional causada pela queda de pressão abaixo do ponto de bolha é expressada como:

$$\Delta q = q_v\left[1 - 0.2\left(\frac{p_{wf}}{p_b}\right) - 0.8\left(\frac{p_{wf}}{p_b}\right)^2\right] \tag{3.8}$$

sabendo que $q_v = \frac{J^* p_b}{1.8}$, onde qv é expressado em $[ft^3/d]$.

Portanto, a vazão para uma determinada pressão abaixo da pressão de bolha é:

$$q = J^*\left(\overline{p} - p_b\right) + \frac{J^* p_b}{1.8} * \left[1 - 0.2\left(\frac{p_{wf}}{p_b}\right) - 0.8\left(\frac{p_{wf}}{p_b}\right)^2\right] \tag{3.9}$$

3.2.7 Fluxo Monofásico

Para camadas de reservatório contendo óleos subsaturados, se a pressão de fluxo no fundo do poço estiver acima das pressões do ponto de bolha dos óleos em todas as camadas, é esperado fluxo monofásico.

$$\sum J^*(\overline{p}_i - p_{wf}) = q_{wh} \tag{3.10}$$

$$AOF = \sum J_i^* \overline{p}_i = \sum AOF_i \tag{3.11}$$

Também é possível se chegar a pressão de fundo. É importante ressaltar que Pwfo é uma pressão dinâmica de fundo de poço devido ao fluxo cruzado entre as camadas.

$$p_{wfo} = \frac{\sum J_i^* \overline{p_i}}{\sum J_i^*}. \tag{3.12}$$

3.2.8 Fluxo Bifásico

Para camadas de reservatório contendo óleos saturados, é esperado um fluxo bifásico. Utilizando Vogel:

$$\sum \frac{J_i^* \overline{p}_i}{1.8} \left[1 - 0.2 \left(\frac{p_{wf}}{\overline{p_i}} \right) - 0.8 \left(\frac{p_{wf}}{\overline{p_i}} \right)^2 \right] = q_{wh} \tag{3.13}$$

$$AOF = \sum \frac{J_i^* \overline{p}_i}{1.8} = \sum AOF_i \tag{3.14}$$

Também é possível se chegar a pressão de fundo:

$$p_{wfo} = \frac{\sqrt{80 \sum J_i^* \overline{p}_i \sum \frac{J_i^*}{\overline{p}_i} + \left(\sum J_i^* \right)^2} - \sum J_i^*}{8 \sum \frac{J_i^*}{\overline{p}_i}} \tag{3.15}$$

3.3 Identificação de pacotes – assuntos

A partir da análise dos modelos apresentados, identifica-se ose seguintes assuntos/pacotes:

- Pacote Reservatório:
  - Composto pelos parâmetros da rocha-reservatório(como porosidade, permeabilidade), fluidos(como viscosidade, densidade e fator volume formação) e do próprio reservatório(como espessura e pressão inicial) e outros.
- Pacote Dados de Produção:
  - Composto por dados de pressão de fundo e inicial do reservatório de modo que o cálculo da IPR seja possível sem necessitar dos parâmetros do pacote acima descrito.
- Pacote Modelos Empíricos de IPR:
  - Calcula os parâmetros necessários para as curvas utilizando os modelos listados. O usuário poderá escolher qual modelo deseja usar..
- Pacote SimuladorCurvasIPR:
  - Relaciona os pacotes listados, sendo responsável por interagir com o usuário através de um interface via texto para definir as ações a serem tomadas.

3.4 Diagrama de pacotes – assuntos

O diagrama de pacotes da Figura 3.1 mostra as relações e dependências existentes entre os pacotes deste software.

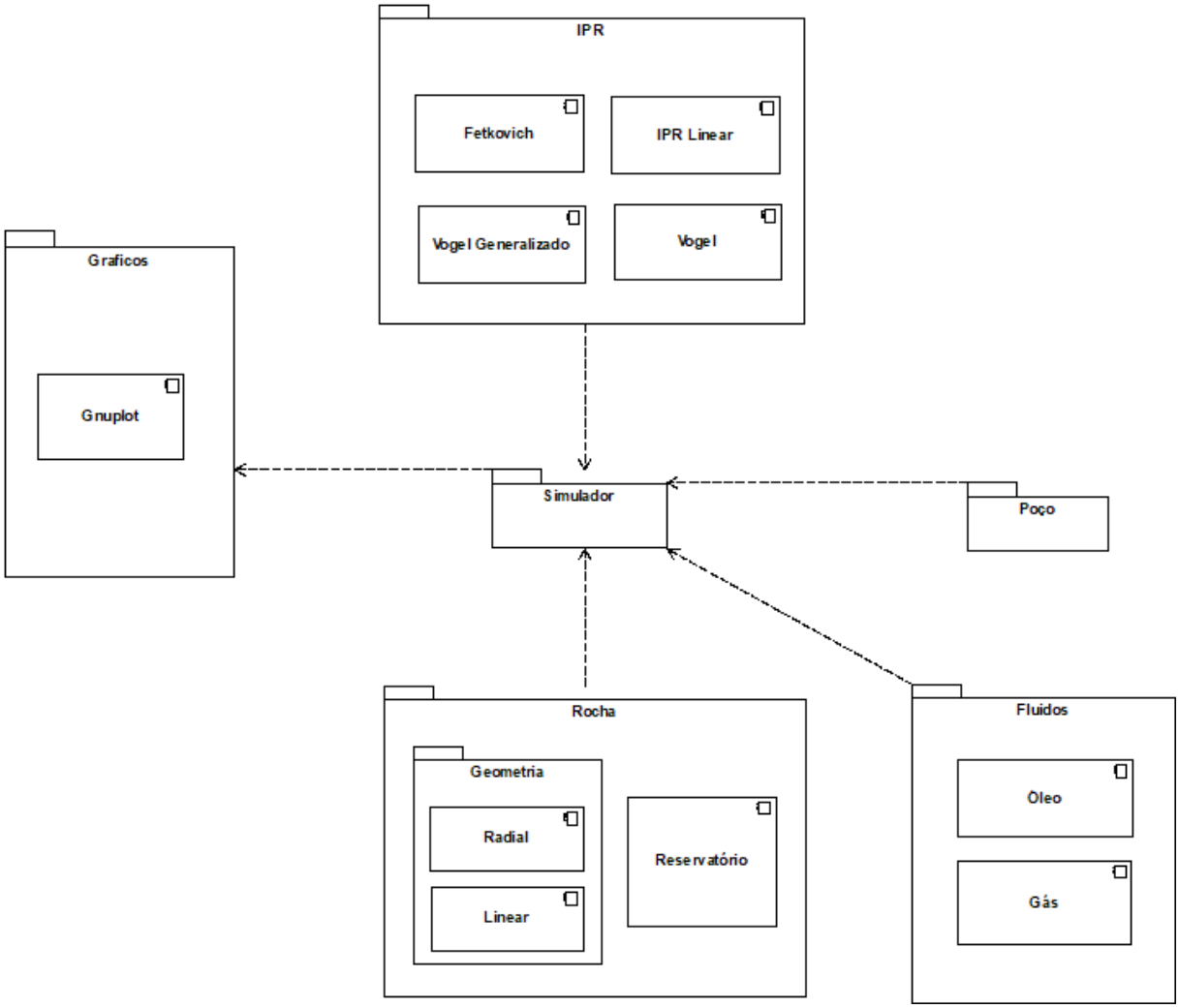

Figura 3.1: Diagrama de Pacotes

Capítulo 4

AOO – Análise Orientada a Objeto

A terceira etapa do desenvolvimento de um projeto de engenharia, no nosso caso um software aplicado a engenharia de petróleo, é a AOO – Análise Orientada a Objeto. A AOO utiliza algumas regras para identificar os objetos de interesse, as relacões entre os pacotes, as classes, os atributos, os métodos, as heranças, as associações, as agregações, as composições e as dependências.

4.1 Diagramas de classes

O diagrama de classes é apresentado na Figura 4.1.

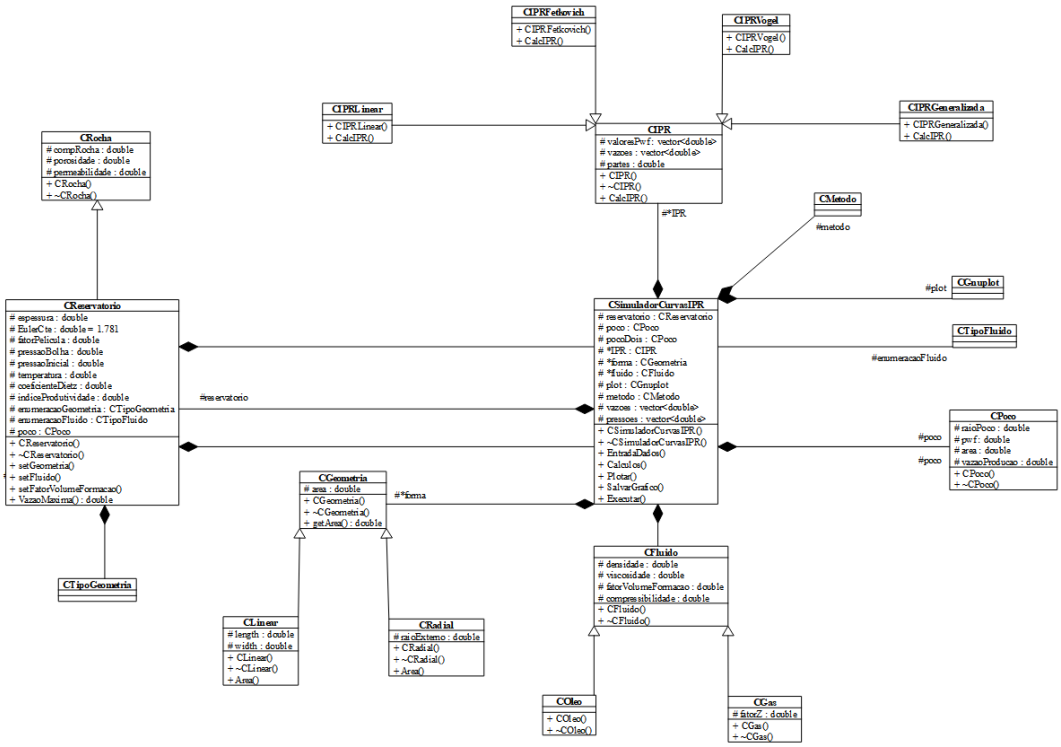

Figura 4.1: Diagrama de classes

4.1.1 Dicionário de classes

- Classe CPoco: representa a classe responsável por receber os parâmetros do poço
- Classe CFluido: representa a classe responsável por receber as propriedades do fluido.
- Classe COleo: representa a classe responsável por receber os parâmetros do óleo quando a intenção do usuário é simular curvas IPR de óleo.
- Classe CGas: representa a classe responsável por receber os parâmetros do gás quando a intenção do usuário é simular curvas IPR de gás.
- Classe CRocha: representa a classe responsável por receber os parâmetros de rocha.
- Classe CReservatorio: representa a classe responsável por receber todos os parâmetros referentes ao reservatório.
- Classe CGeometria: representa a classe responsável por definir a geometria do reservatório.
- Classe CGeometriaRadial: representa a classe responsável por receber e calcular dimensões do reservatório de geometria radial.
- Classe CGeometriaLinear: representa a classe responsável por receber e calcular dimensões do reservatório de geometria linear.
- Classe CIPR: representa a classe base que recebe os vetores de pressão de fundo e vazão.
- Classe CIPRLinear: representa a classe responsável por calcular a curva IPR para reservatórios acima da pressão de bolha.
- Classe CIPRFetkovich: representa a classe responsável pelo cálculo da curva IPR utilizando o modelo de Fetkovich.
- Classe CIPRVogel: representa a classe responsável pelo cálculo da curva IPR utilizando o modelo de Vogel.
- Classe CIPRGeneralizada: representa a classe responsável pelo cálculo da curva IPR utilizando o modelo de Vogel generalizado para reservatórios bifásicos.
- Classe EMetodo: enumeração que representa os tipos de métodos de cálculo de IPR.
- Classe ETipoGeometria: enumeração que representa os tipos de geometria do reservatório.
- Classe ETipoFluido: enumeração que representa os tipos de fluidos.

- Classe CSimuladorCurvasIPR: representa a classe responsável pela simulação do cálculo das curvas de IPR. Recebe as escolhas do usuário quanto aos parâmetros do poço, reservatório e fluidos, além da escolha do modelo adequado para a obtenção das curvas de IPR.

- Classe CGrafico: representa a classe responsável pela parte gráfica do programa, a partir do uso do software externo Gnuplot.

4.2 Diagrama de seqüência – eventos e mensagens

O diagrama de seqüência enfatiza a troca de eventos e mensagens e sua ordem temporal. Portanto, representa uma forma de diagrama interativo que delineia a maneira e a sequência em que um conjunto de objetos colabora.

4.2.1 Diagrama de sequência geral

Veja o diagrama de seqüência geral na Figura 4.2. Notar que a entrada de dados é realizada via teclado.

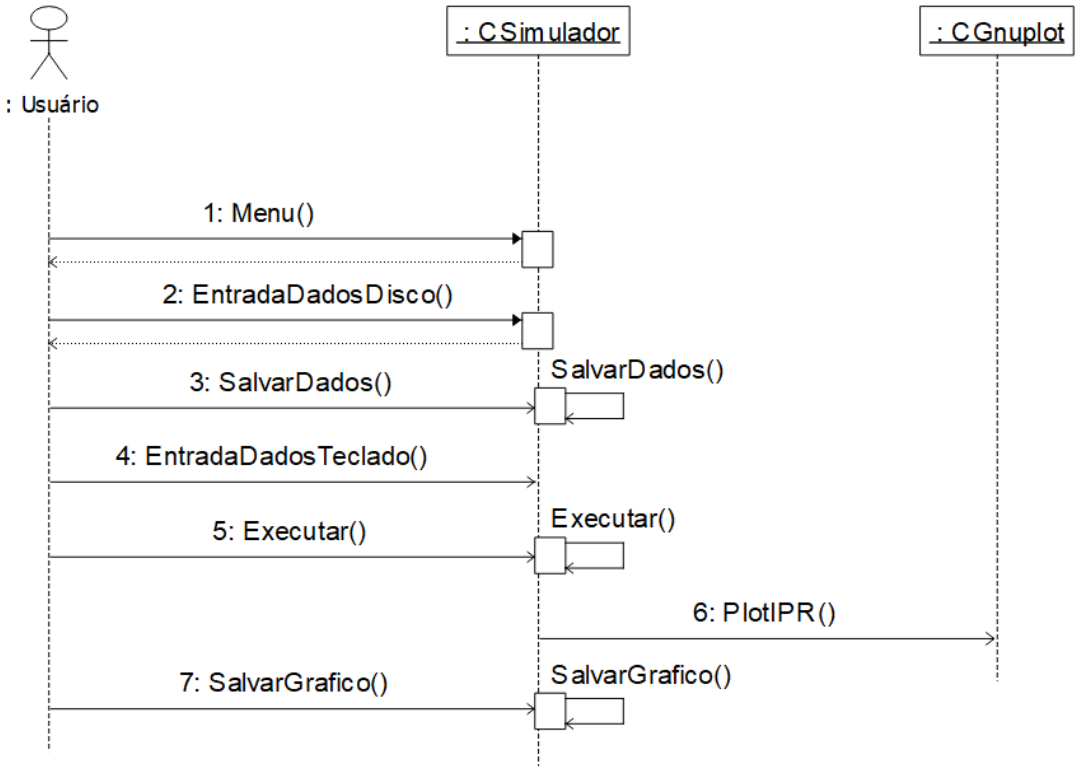

Figura 4.2: Diagrama de seqüência

4.2.2 Diagrama de sequência específico

Para o caso de um diagrama de sequência específico para cálculo dos parâmetros das curvas de IPR é necessário que o usuário forneça os dados de fluido, reservatório e poço, bem como escolha o método que deseja utilizar para calcular e obter os resultados que serão utilizados no gráfico final. Veja o diagrama de sequência na Figura 4.3.

4.3 Diagrama de comunicação – colaboração

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

Veja na Figura 4.3 o diagrama de comunicação mostrando a sequência de como as classes se comunicam entre si para o funcionamento do software. Observe que o método EntradaDados() pode ser representado através dos métodos EntradaDadosDisco() ou EntradaDadosTeclado(), ficando a critério do usuário selecionar qual modo de entrada de dados atende melhor à sua demanda.

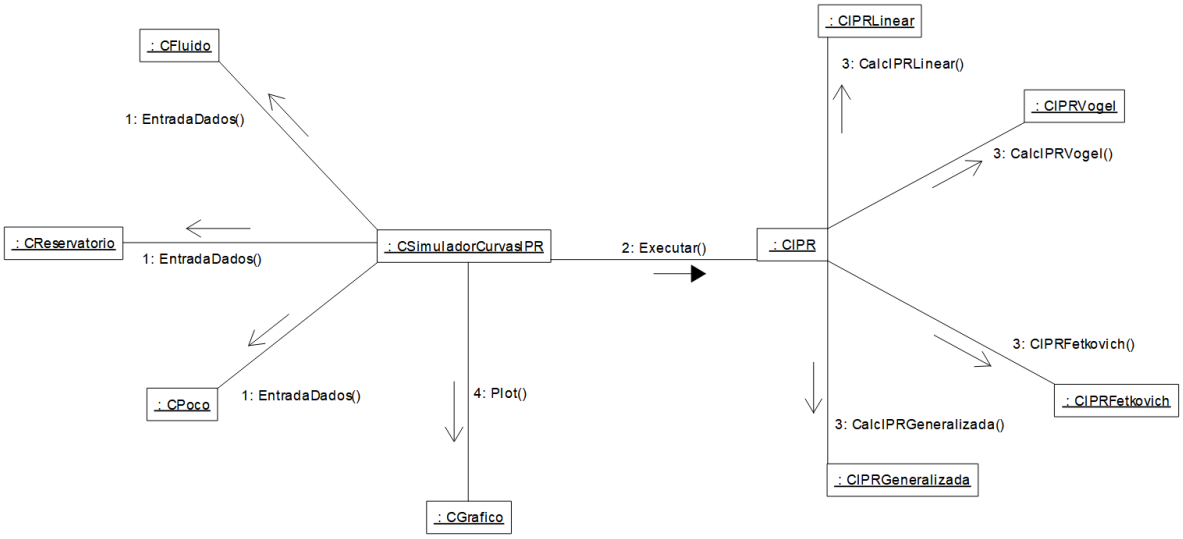

Figura 4.4: Diagrama de comunicação

4.4 Diagrama de máquina de estado

Um diagrama de máquina de estado, também conhecido como diagrama de estados ou diagrama de transição de estados, é uma representação visual que descreve o comportamento de uma classe ou entidade em relação aos diferentes estados em que pode estar e às transições entre esses estados. Essa ferramenta é amplamente utilizada na engenharia de software, engenharia de sistemas e em outros campos para modelar o comportamento de sistemas complexos.

É possível observar na figura 4.5 o diagrama de máquina de estado para os objetos da classe CSimuladorCurvasIPR.

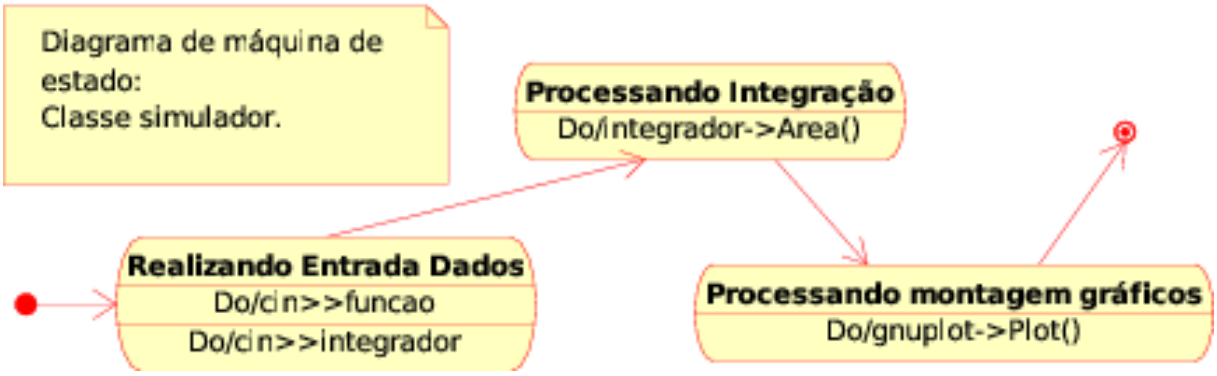

Figura 4.5: Diagrama de máquina de estado da classe XXX

4.5 Diagrama de atividades

Um diagrama de atividades é um tipo de diagrama da Linguagem de Modelagem Unificada (UML) que representa o fluxo de controle de atividades em um sistema. Ele é essencialmente um gráfico de fluxo que mostra o fluxo de controle de uma atividade para outra.

Veja na Figura 4.6 o diagrama de atividades correspondente a uma atividade específica do diagrama de máguina de estado. Neste caso específico, é realizado o cálculo da produtividade de um poço vertical, utilizando-se o modelo empírico de Fetkovich, onde se é necessário o usuário possuir dados de pressão de fundo e vazão de dois diferentes poços para que seja realizado o cálculo das constantes do modelo e essas sejam utilizadas no cálculo de vazões para o poço de interesse.

Figura 4.6: Diagrama de atividades da classe X método Y

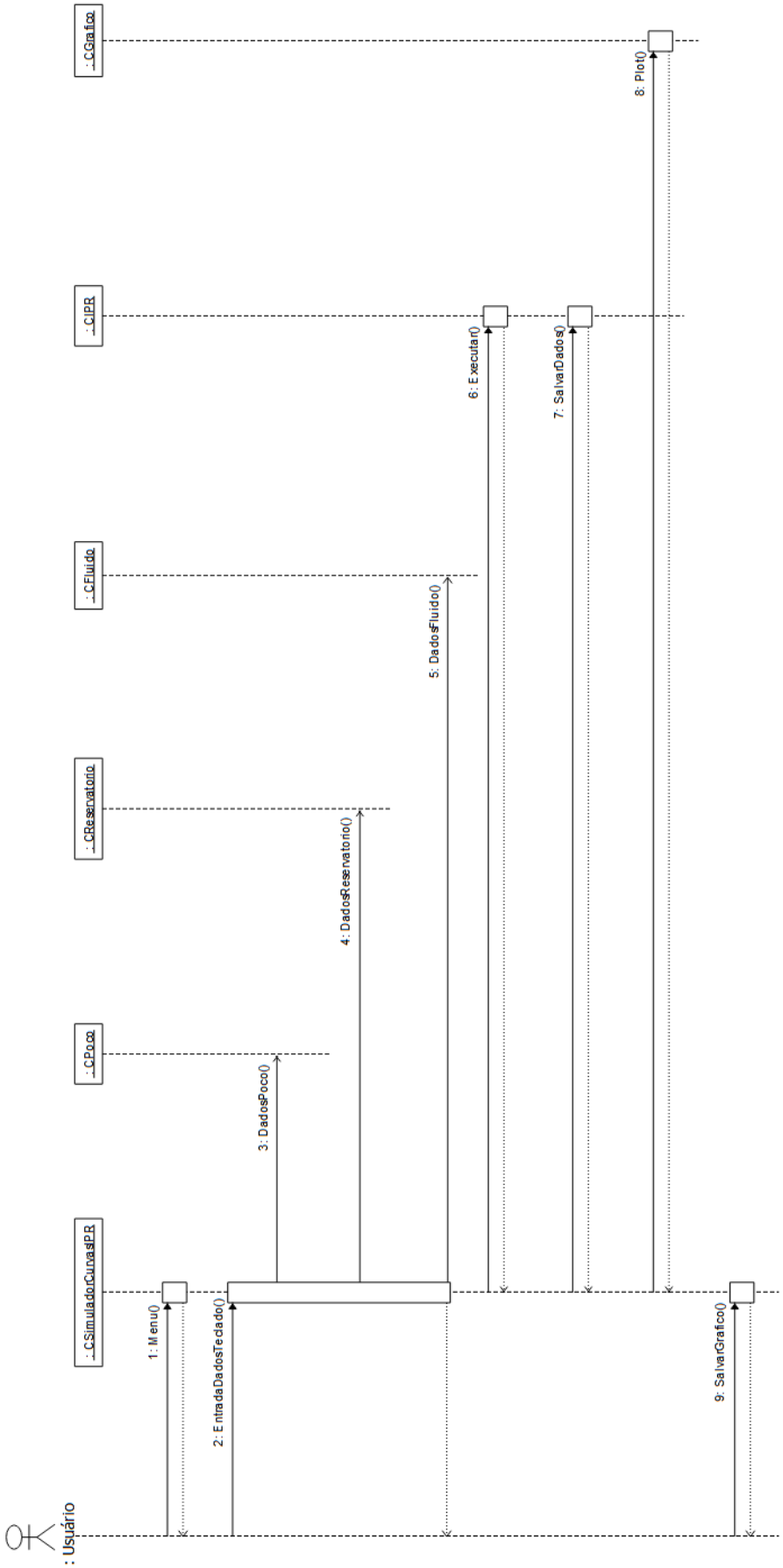

Figura 4.3: Diagrama de seqüência

Capítulo 5

Projeto

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

5.1 Projeto do sistema

Depois da análise orientada a objeto 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.

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

Segundo [**?**, **?**], o projeto do sistema é a estratégia de alto nível para resolver o problema e elaborar uma solução. Você deve se preocupar com itens como:

1. Protocolos

    - Definição dos protocolos de comunicação entre os diversos elementos externos
        - Neste projeto o software irá se comunicar com o componente externo Gnuplot.
    - Definição do formato dos arquivos gerados pelo software. Por exemplo: prefira formatos abertos, como arquivos txt e xml.
        - Neste projeto será gerado um arquivo .txt com os valores de pressão e vazão obtidos dos resultados.

  - Neste projeto será gerado um arquivo .png com as curvas IPR geradas.

2. Recursos

   - Identificação e alocação dos recursos globais, como os recursos do sistema serão alocados, utilizados, compartilhados e liberados. Implicam modificações no diagrama de componentes.
     - Neste projeto serão utilizados todos os componentes do computador que estão dentro do gabinete: HD, processador, memória;
     - Neste projeto serão utilizados todos os componentes :teclado, mouse e tela.

3. Controle

   - Identificação da necessidade de otimização.
     - Neste projeto não haverá a necessidade de grande quantidade de memória visto que não haverá grande quantidade de dados.
   - Identificação de concorrências – quais algoritmos podem ser implementados usando processamento paralelo.
     - O software não necessitará de uma grande escala de processamento, logo, não haverá necessidade de processamento paralelo.

4. Plataformas

   - Identificação e definição das plataformas a serem suportadas: hardware, sistema operacional e linguagem de software.
     - Neste projeto será utilizado a linguagem de programação C++.
     - O software deverá ser multiplataforma, podendo ser executado em Windows e GNU/Linux.
     - O software será desenvolvido no sistema operacional Windows 11 em máquinas com processador Intel Core i5-11ª geração e Intel Core i3-10ª geração.
   - Seleção das bibliotecas externas a serem utilizadas.
     - Neste projeto será utilizada a biblioteca padrão da linguagem C++, incluindo vector, string, iostream, fstream, sstream e iremos utilizar a classe CGnuplot, que fornece acesso ao programa externo Gnuplot (link).
   - Seleção do ambiente de desenvolvimento para montar a interface de desenvolvimento – IDE.
     - Neste projeto a IDE utilizada será o software Embarcadero na versão 6.3 (link) e o DEV C++ na versão 5.11 (link)..

5.2 Projeto orientado a objeto – POO

O projeto orientado a objeto é a etapa posterior ao projeto do sistema. Baseia-se na análise, mas considera as decisões do projeto do sistema. Acrescenta a análise desenvolvida e as características da plataforma escolhida (hardware, sistema operacional e linguagem de software). 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.

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

5.2.0.1 Efeitos do projeto no modelo estrutural

- Adicionar nos diagramas de pacotes as bibliotecas e subsistemas selecionados no projeto do sistema (exemplo: a biblioteca gráfica selecionada).
  - Neste projeto, foi utilizada a classe `CGnuplot` a fim de gerar as curvas para realização de análises gráficas. Para isto, é necessário que haja a instalação do software Gnuplot para que o funcionamento do Simulador seja totalmente garantido.
- Novas classes e associações oriundas das bibliotecas selecionadas e da linguagem escolhida devem ser acrescentadas ao modelo.
  - Neste projeto, foram feitas associações entre as classes CGnuplot e CSimuladorCurvasIPR para que ocorra a geração de gráficos dos parâmetros calculados;
  - As classes CRocha, CReservatorio, CPoco e CFluido estão associadas a CSimuladorCurvasIPR pois configuram os dados fundamentais que serão utilizados nos modelos empíricos;
  - Por fim, as classes CIPRVogel, CIPRGeneralizada,CIPRLinear e CIPRFetkovich também estão relacionadas a CSimuladorCurvasIPR visto que são os modelos empíricos que serão utilizados nos cálculos que resultarão nos parâmetros principais das curvas de IPR (pressão e vazão).
- Estabelecer as dependências e restrições associadas à plataforma escolhida.

  - O software utiliza o HD, processador e o teclado do computador;
  - Pode ser executado nas plataformas GNU/Linux, Windows ou Mac;
  - Nos Sistemas Operacionais citados, há necessidade de instalação do software Gnuplot para o funcionamento total do programa.

5.2.0.2 Efeitos do projeto no modelo dinâmico

- Revisar os diagramas de seqüência e de comunicação considerando a plataforma escolhida.
  - Os diagramas de sequência e comunicação serão revisados, se necessário, durante as etapas de desenvolvimento do código.
- Verificar a necessidade de se revisar, ampliar e adicionar novos diagramas de máquinas de estado e de atividades.
  - Os diagramas de máquina de estado e atividades sofrerão correções se surgir a necessidade à medida que o código for desenvolvido..

5.2.0.3 Efeitos do projeto nos atributos

- Atributos novos podem ser adicionados a uma classe, como, por exemplo, atributos específicos de uma determinada linguagem de software (acesso a disco, ponteiros, constantes e informações correlacionadas).
  - Neste projeto, os atributos fin e fout deverão ser criados a fim de possibilitar a leitura dos dados a partir de um arquivo de disco bem como criar um arquivo com a saída de dados.

5.2.0.4 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).
  - Neste projeto, além da possibilidade de entrada de dados por um arquivo de disco, estes poderão também ser digitados pelo usuário utilizando o teclado do computador de maneira a não se limitar somente na utilização de uma das vias.
- Revise os diagramas de classes, de sequência e de máquina de estado.

  - As classes deverão ser revisadas a medida que for notada a necessidade de adicionar uma ou mais classes durante o desenvolvimento do código. Atualmente, foram adicionadas as classes CRocha, CGeometria, CRadial, CLinear, CIPR, COleo e CGas, com o intuito solucionar uma maior gama de problemas relacionados a IPR de gás e para reservatórios de diferentes geometrias.

5.2.0.5 Efeitos do projeto nas heranças

- Reorganização das classes e dos métodos (criar métodos genéricos com parâmetros que nem sempre são necessários e englobam métodos existentes).

  - Foi criada uma classe-base CIPR para as classes CIPRLinear, CIPRVogel, CIPRFetkovich e CIPRGeneralizada, duas classes herdeiras da classe CFluido, COleo e CGas, e as classes foram reordenadas de modo que a classe CRocha seja uma classe-base para CReservatorio, essa seja uma classe-base para CGeometria e, por fim, essa seja uma classe-base para duas classes herdeiras, CRadial e CLinear.

5.2.0.6 Efeitos do projeto nas associações

- Deve-se definir na fase de projeto como as associações serão implementadas, se obedecerão um determinado padrão ou não.

  - O projeto, ao referir-se a classes e associações, sofrerá mudanças futuras quando o código começar a ser desenvolvido.

5.2.0.7 Efeitos do projeto nas otimizações

- A ordem de execução pode ser alterada.

  - Um menu com opções distintas irá aparecer na tela de modo que o usuário poderá escolher como gostaria de entrar com os dados, via teclado ou disco, o modelo empírico que gostaria de utilizar para realização dos cálculos de pressão e vazão. Logo após, o gráfico será gerado e o usuário poderá salvá-lo ou não.

As dependências das bibliotecas e arquivos são demonstrados pelo diagrama de componentes e as dependências entre o sistema e o hardware são ilustradas pelo diagrama de implantação.

5.3 Diagrama de componentes

O diagrama de componentes mostra a forma como os componentes do software se relacionam, suas dependências. Inclui itens como: componentes, subsistemas, executáveis,

nós, associações, dependências, generalizações, restrições e notas. Exemplos de componentes são bibliotecas estáticas, bibliotecas dinâmicas, dlls, componentes Java, executáveis, arquivos de disco, código-fonte.

Veja na Figura 5.1 o de diagrama de componentes. Dessa forma, um diagrama de componentes é uma ferramenta valiosa para o seu projeto de desenvolvimento de software para simulação de curvas IPR na engenharia de petróleo por várias razões:

- Visualiza a estrutura do sistema;
- Mostra as dependências entre os elementos;
- Organiza o código em módulos;
- Facilita a comunicação na equipe;
- Identifica áreas críticas do sistema;
- Serve como documentação.

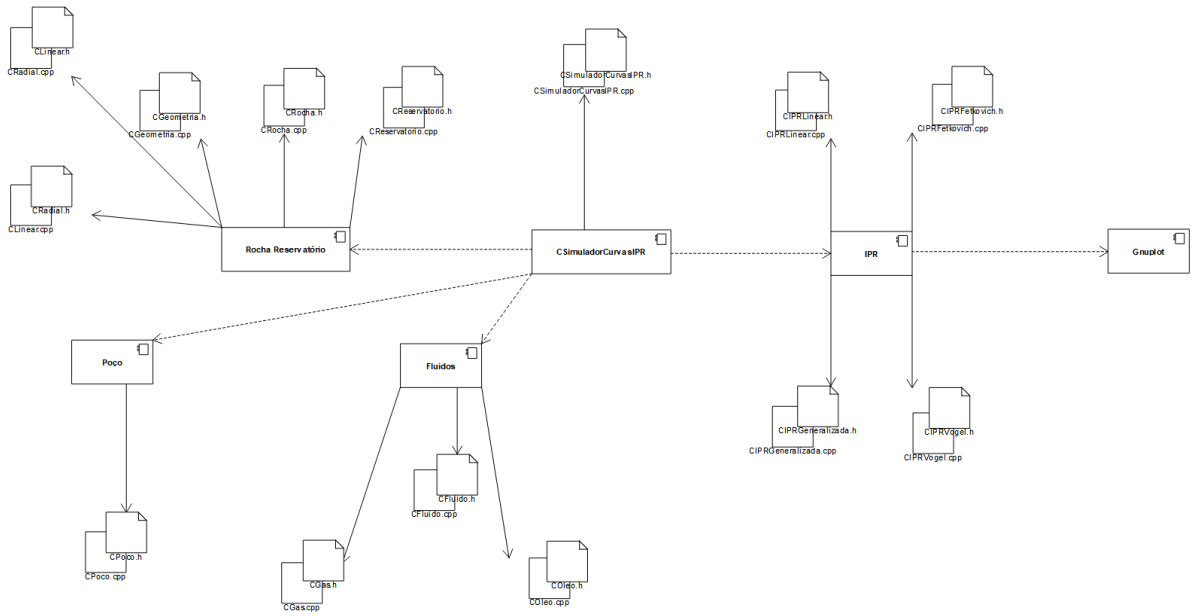

Figura 5.1: Diagrama de componentes

5.4 Diagrama de implantação

O diagrama de implantação é um diagrama de alto nível que inclui relações entre o sistema e o hardware e que se preocupa com os aspectos da arquitetura computacional escolhida. Seu enfoque é o hardware, a configuração dos nós em tempo de execução. O diagrama de implantação deve incluir os elementos necessários para que o sistema seja colocado em funcionamento: computador, periféricos, processadores, dispositivos, nós, relacionamentos de dependência, associação, componentes, subsistemas, restrições e

notas. O diagrama de implantação desempenha um papel crucial no projeto de engenharia de petróleo com simulações de curvas IPR para Poços Verticais. Ele se aplica de forma significativa ao ajudar a visualizar como os diversos componentes de software e hardware interagem em seu ambiente de execução. Neste contexto, o diagrama de implantação permite a:

- Visualização da Infraestrutura;
- Modelagem de Conexões;
- Planejamento de Recursos;
- Identificação de Pontos de Falha;
- Escalabilidade.

Veja na Figura 5.2 um exemplo de diagrama de implantação.

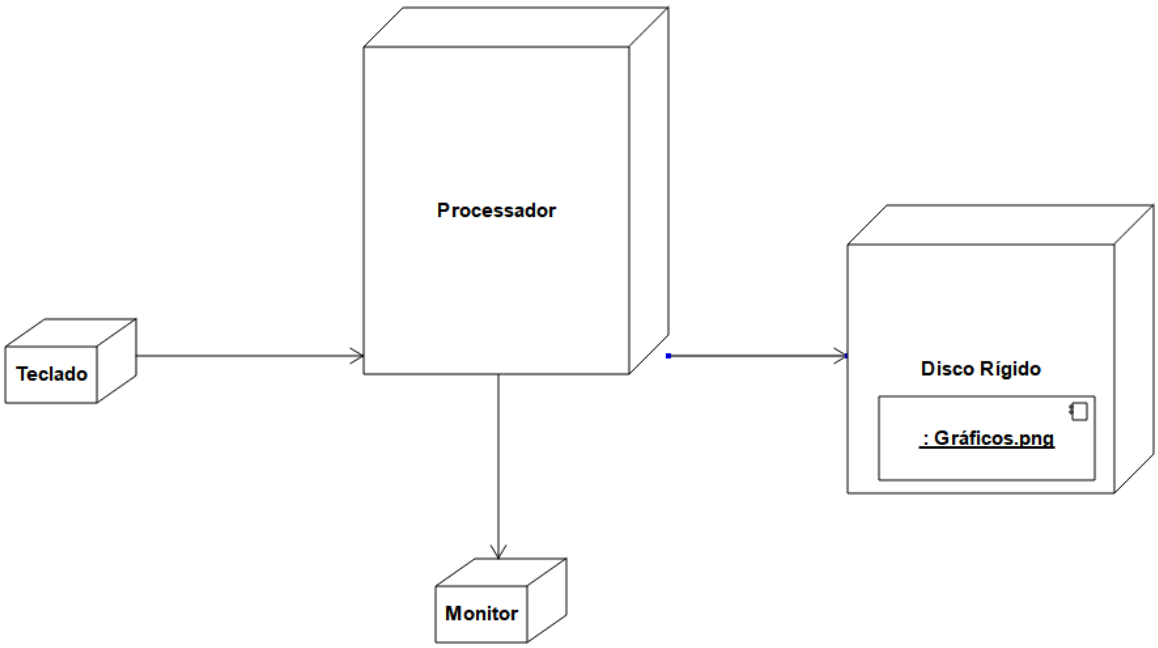

Figura 5.2: Diagrama de implantação

5.4.1 Lista de características <<features>>

No final do ciclo de concepção e análise chegamos a uma lista de características *<<features>>* que teremos de implementar.

Após a análises desenvolvidas e considerando o requisito de que este material deve ter um formato didático, chegamos a seguinte lista:

- v0.1
  - Lista de classes a serem implementadas: CFluido, COleo, CGas, CPoco, CRocha, CReservatorio, CGeometria, CGeometriaLinear, CGeometriaRadial, CIPR, CIPRLinear, CIPRVogel, CIPRFetkovich, CIPRGeneralizada, CSimuladorCurvasIPR e CGnuplot.
  - Testes:
    * O software deve ser capaz de exibir um menu com uma lista de opções ao usuário;
    * O software deve carregar informações via disco ou ler as informações inseridas pelo usuário via teclado;
    * O software deve disponibilizar uma opção para que o usuário selecione o modelo empírico a ser utilizado, desde que seja adequado para seu problema
    * O software deve, a partir dos dados inseridos, realizar cálculos iterativos de vazões e exportar os resultados para um arquivo externo;
    * O software deve plotar, utilizando a biblioteca externa Gnuplot, gráfico com a curva IPR do modelo selecionado;
    * O software deve oferecer ao usuário a possibilidade de salvar o gráfico gerado caso seja de necessidade do mesmo;
- v0.3
  - Lista de classes a serem implementadas
  - Testes
- v0.5
  - Lista de classes a serem implementadas
  - Testes

5.4.2 Tabela classificação sistema

A Tabela a seguir é utilizada para classificação do sistema desenvolvido. Deve ser preenchida na etapa de projeto e revisada no final, quando o software for entregue na sua versão final.

Licença:	[X] livre GPL-v3 [ ] proprietária
Engenharia de software:	[ ] tradicional [X] ágil [ ] outras
Paradigma de programação:	[ ] estruturada [X] orientado a objeto - POO [ ] funcional
Modelagem UML:	[X] básica [X] intermediária [ ] avançada
Algoritmos:	[X] alto nível [X] baixo nível
	implementação: [ ] recursivo ou [X] iterativo; [X] determinístico ou [ ] não-determinístico; [ ] exato ou [X] aproximado
	concorrências: [X] serial - síncrona [] concorrente [] paralelo
	paradigma: [X] dividir para conquistar [ ] programação linear [ ] transformação/ redução [ ] busca e enumeração [ ] heurístico e probabilístico [ ] baseados em pilhas
Software:	[ ] de base [X] aplicados [ ] de cunho geral [X] específicos para determinada área [X] educativo [X] científico
	instruções: [X] alto nível [ ] baixo nível
	otimização: [X] serial não otimizado [X] serial otimizado [] concorrente [] paralelo [ ] vetorial
	interface do usuário: [ ] kernel numérico [ ] linha de comando [ ] modo texto [X] híbrida (texto e saídas gráficas) [ ] modo gráfico (ex: Qt) [ ] navegador
Recursos de C++:	[X] C++ básico (FCC): variáveis padrões da linguagem, estruturas de controle e repetição, estruturas de dados, struct, classes(objetos, atributos, métodos), funções; entrada e saída de dados (*streams*), funções de cmath
	[X] C++ intermediário: funções lambda. Ponteiros, referências, herança, herança múltipla, polimorfismo, sobrecarga de funções e de operadores, tipos genéricos (templates), *smarth pointers*. Diretrizes de pré-processador, classes de armazenamento e modificadores de acesso. Estruturas de dados: enum, uniões. Bibliotecas: entrada e saída acesso com arquivos de disco, redirecionamento. Bibliotecas: *filesystem*
	[] C++ intermediário 2: A biblioteca de gabaritos de C++ (a STL), containers, iteradores, objetos funções e funções genéricas. Noções de processamento paralelo (múltiplas threads, uso de *thread, join* e *mutex*). Bibliotecas: *random, threads*

	[ ] C++ avançado: Conversão de tipos do usuário, especializações de templates, exceções. Cluster de computadores, processamento paralelo e concorrente, múltiplos processos (pipes, memória compartilhada, sinais). Bibliotecas: *expressões regulares, múltiplos processos*
Bibliotecas de C++:	[X] Entrada e saída de dados (*streams*) [X] cmath [X] *filesystem* [ ] *random* [X] *threads* [ ] *expressões regulares* [ ] *múltiplos processos [X] Vector [X] Locale [X] String*
Bibliotecas externas:	[X] `CGnuplot` [] `QCustomPlot` [ ] `Qt diálogos` [ ] QT Janelas/menus/BT_____
Ferramentas auxiliares:	Montador: [] make [] cmake [] qmake
IDE:	[ ] Editor simples: kate/gedit/emacs [ ] kdevelop [ ] QT-Creator [X] Embarcadero [X] Dev C++
SCV:	[ ] cvs [ ] svn [X] git
Disciplinas correlacionadas	[ ] estatística [ ] cálculo numérico [ ] modelamento numérico [ ] análise e processamento de imagens [X] elevação e escoamento [X] engenharia de reservatório

Capítulo 6

Ciclos Construção - Implementação

Neste capítulo, são apresentados os códigos fonte implementados.

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 LaTeX. 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 6.1o arquivo com código da classe `CFluido`.

Listing 6.1: Arquivo de cabeçalho da classe CFluido

```
1 #ifndef CFluido_h
2 #define CFluido_h
3
4 #include <iostream>
5
6 class CFluido
7 {
8
9         protected:
```

```
10
11          double densidade; // Densidade do fluido
12          double viscosidade; // Viscosidade do fluido
13          double fatorVolumeFormacao; //Fator Volume formacao do fluido
14          double compressibilidade; // Compressibilidade do fluido
15
16          public:
17
18                  CFluido(){}; // Construtor
19                  virtual ~CFluido(){}; // Destrutor
20
21                  double Densidade() const {return densidade;} // Metodo
                        get para a densidade do fluido
22                  void Densidade (double _densidade) {densidade =
                        _densidade;} // Metodo set para a densidade do fluido
23
24                  double Viscosidade() const {return viscosidade;} //
                        Metodo get para a viscosidade do fluido
25                  void Viscosidade (double _viscosidade) {viscosidade =
                        _viscosidade;} // Metodo set para a viscosidade do
                        fluido
26
27                  double FatorVolumeFormacao() const {return
                        fatorVolumeFormacao;} // Metodo get para o fator
                        volume-formacao do oleo
28                  void FatorVolumeFormacao (double _fatorVolumeFormacao) {
                        fatorVolumeFormacao = _fatorVolumeFormacao;} //
                        Metodo set para o fator volume-formacao do oleo
29
30          double getCompressibilidade() {return compressibilidade; } //
               Metodo get para a compressibilidade do oleo
31          void setCompressibilidade(double _compressibilidade) {
               compressibilidade = _compressibilidade; } // Metodo set para
               a compressibilidade do oleo
32
33};
34
35#endif
```

Apresenta-se na listagem 6.2o arquivo com código da classe `COleo`.

Listing 6.2: Arquivo de cabeçalho da classe COleo

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

```
7
8 //Declaracao de COleo que herda de CFluido
9
10 class COleo : public CFluido {
11
12
13         public:
14
15                 COleo(){}; // Construtor
16                 virtual ~COleo() = default; // Destrutor
17
18 };
19
20 #endif
```

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

Listing 6.3: Arquivo de cabeçalho da classe CGas

```
1 #ifndef CGas_h
2 #define CGas_h
3
4 #include "CFluido.h"
5
6 #include <iostream>
7
8 //Declaracao de CGas que herda de CFluido
9
10 class CGas : public CFluido {
11
12         protected:
13
14                 double fatorZ; // Constante dos gases
15
16         public:
17
18                 CGas(){}; // Construtor
19                 ~CGas(){}; // Destrutor
20
21                 double FatorZ() const {return fatorZ;} // Metodo get
                       para a constante dos gases
22                 void FatorZ (double _fatorZ) {fatorZ = _fatorZ;} //
                       Metodo set para a constante dos gases
23
24 };
25
26 #endif
```

Apresenta-se na listagem 7.4o arquivo com código da classe `CTipoFluido`.

Listing 6.4: Arquivo de cabeçalho da classe CTipoFluido

```
#ifndef CTipoFluido_h
#define CTipoFluido_h

#include <iostream>

enum class CTipoFluido {

        none = 0, oleo, gas // Definicao da geometria por enumeracao

        // O usuario fica aberto a adicionar mais geometrias

};

#endif
```

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

Listing 6.5: Arquivo de cabeçalho da classe CRocha.

```
#ifndef CRocha_h
#define CRocha_h

#include <iostream>

// Classe CRocha que representa as propriedades de rocha

class CRocha {

        protected:

                double compRocha; // Compressibilidade da rocha
                double porosidade; // Porosidade da rocha
                double permeabilidade; // Permeabilidade da rocha


        public:

                CRocha(){}; // Construtor
                ~CRocha(){}; // Destrutor

                double CompRocha() const {return compRocha;} // Metodo
                    get para a compressibilidade da rocha
                void CompRocha (double _compRocha) {compRocha =
                    _compRocha;} // Metodo set para a compressibilidade
                    da rocha

                double Porosidade() const {return porosidade;} // Metodo
                     get para a porosidade da rocha
```

```
26                 void Porosidade (double _porosidade) {porosidade =
                       _porosidade;} // Metodo set para a porosidade da
                       rocha
27
28                 double Permeabilidade() const {return permeabilidade;}
                       // Metodo get para a permeabilidade da rocha
29                 void Permeabilidade (double _permeabilidade) {
                       permeabilidade = _permeabilidade;} // Metodo set para
                        a permeabilidade da rocha
30
31 };
32
33 #endif
```

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

Listing 6.6: Arquivo de cabeçalho da classe CReservatorio

```
1 #ifndef CReservatorio_h
2 #define CReservatorio_h
3
4 // Inclusao dos arquivos de cabecalho
5
6 #include "CRocha.h"
7 #include "CGeometria.h"
8 #include "CLinear.h"
9 #include "CRadial.h"
10 #include "CTipoGeometria.h"
11 #include "CTipoFluido.h"
12 #include "CFluido.h"
13 #include "COleo.h"
14 #include "CGas.h"
15 #include "CPoco.h"
16
17 #include <iostream>
18
19 // Classe CReservatorio que representa o reservatorio
20
21 class CReservatorio: public CRocha {
22
23         protected:
24
25                 double espessura; // Espessura do reservatorio
26                 double EulerCte = 1.781; // Constante de Euler
27                 double fatorPelicula; // Fator de pelicula do
                       reservatorio
28                 double pressaoBolha; // Pressao de bolha do reservatorio
29                 double pressaoInicial; // Pressao media inicial do
                       reservatorio
```

```
30                double temperatura; // Temperatura
31                double coeficienteDietz; // Coeficiente de Dietz
32                double indiceProdutividade; // Indice de Produtividade
                     do reservatorio
33                CTipoGeometria enumeracaoGeometria; // Enumeracao para
                     escolha do tipo de geometria
34                CTipoFluido enumeracaoFluido; // Enumeracao para escolha
                      do tipo de fluido
35                CPoco poco; // Objeto de CPoco associado a reservatorio
36
37        public:
38
39                CReservatorio () {} // Construtor default
40
41                ~CReservatorio () {} // Destrutor
42
43                double Espessura () const {return espessura;} // Metodo
                     get para a espessura
44                void Espessura (double _espessura) {espessura =
                     _espessura;} // Metodo set para a espessura
45
46                double FatorPelicula() const {return fatorPelicula;} //
                     Metodo get para a fatorPelicula
47                void FatorPelicula (double _fatorPelicula) {
                     fatorPelicula = _fatorPelicula;} // Metodo set para a
                      fatorPelicula
48
49                double PressaoBolha() const {return pressaoBolha;} //
                     Metodo get para a pressaoBolha
50                void PressaoBolha (double _pressaoBolha) {pressaoBolha =
                      _pressaoBolha;} // Metodo set para a pressaoBolha
51
52                double PressaoInicial() const {return pressaoInicial;}
                     // Metodo get para a pressaoInicial
53                void PressaoInicial (double _pressaoInicial) {
                     pressaoInicial = _pressaoInicial;} // Metodo set para
                      a pressaoInicial
54
55                double Temperatura() const {return temperatura;} //
                     Metodo get para a temperatura
56                void Temperatura (double _temperatura) {temperatura =
                     _temperatura;} // Metodo set para a temperatura
57
58                void setGeometria(CTipoGeometria _enumeracaoGeometria);
                     // Metodo set para o tipo de geometria
59
60                void setFluido(CTipoFluido _enumeracaoFluido); // Metodo
                      set para o tipo de fluido
```

```
61
62                 void setFatorVolumeFormacao(double _fatorVolumeFormacao)
                       ; // Metodo set para o fator volume-formacao
63
64                 void CoeficienteDietz(CTipoGeometria
                       _enumeracaoGeometria); // Metodo para calculo do
                       Coeficiente de Dietz a ser utilizado
65         double getCoeficienteDietz() { return coeficienteDietz; }
66
67                 void IndiceProdutividade(CFluido* _fluido, CGeometria*
                       _geometria, CPoco& _poco); // Metodo para calculo do
                       Indice de Produtividade
68         double getIndiceProdutividade() {return indiceProdutividade; }
69
70                 double VazaoMaxima() {return ((indiceProdutividade *
                       pressaoInicial)/1.8);} // Metodo para calculo da
                       maxima vazao obtida pelo reservatorio (AOF)
71
72};
73
74#endif
```

Apresenta-se na listagem 6.7 o arquivo de implementação da classe `CReservatorio`.

Listing 6.7: Arquivo de implementação da classe CReservatorio

```
1#include <iostream>
2#include <fstream>
3#include <string>
4#include <cmath>
5
6#include "CReservatorio.h"
7
8using namespace std;
9
10// Implementacao do metodo CoeficienteDietz da classe CReservatorio
11void CReservatorio::CoeficienteDietz(CTipoGeometria _enumeracaoGeometria
    ) {
12
13// Switch para atribuir o coeficiente de Dietz baseado no tipo de
    geometria
14         switch (_enumeracaoGeometria) {
15
16                 case CTipoGeometria::linear:
17                         coeficienteDietz = 30.8828;
18                         break;
19
20                 case CTipoGeometria::radial:
21                         coeficienteDietz = 31.62;
```

```
22                          break;
23
24                  default:
25                          cout << "Geometria nao encontrada";
26                          break;
27
28          }
29
30}
31
32// Implementacao do metodo IndiceProdutividade da classe CReservatorio
33void CReservatorio::IndiceProdutividade(CFluido* _fluido, CGeometria*
    _geometria, CPoco& _poco) {
34
35          // Calculo do Indice de Produtividade usando os parametros do
               reservatorio
36          indiceProdutividade = ((permeabilidade * espessura) / (141.2 *
               _fluido->FatorVolumeFormacao() * _fluido->Viscosidade() *
               (0.5 * log((4 * _geometria->getArea()) / (EulerCte *
               coeficienteDietz * _poco.getRaioPoco() * _poco.getRaioPoco())
               ) + fatorPelicula)));
37
38}
```

Apresenta-se na listagem 6.8o arquivo com código da classe `CGeometria`.

Listing 6.8: Arquivo de cabeçalho da classe CGeometria

```
1#ifndef CGeometria_h
2#define CGeometria_h
3
4#include <iostream>
5
6//Classe CGeometria que representa a geometria da rocha reservatorio
7
8class CGeometria{
9
10          protected:
11
12                  double area;
13
14          public:
15
16                  CGeometria(){};
17                  virtual ~CGeometria() = default; //Destrutor
18
19                  virtual void Area() {}
20
21                  double getArea () {return area;} // Puxar area da
```

```
                    geometria em questÃ£o
22
23};
24
25#endif
```

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

Listing 6.9: Arquivo de cabeçalho da classe CLinear

```
1#ifndef CLinear_h
2#define CLinear_h
3
4#include <iostream>
5#include <fstream>
6
7#include "CGeometria.h"
8
9class CLinear: public CGeometria {
10
11        protected:
12
13        double length; // Comprimento da rocha reservatorio
14        double width; // Largura da rocha reservatorio
15
16        public:
17
18                CLinear(); // Construtor default
19                ~CLinear() {}; // Destrutor
20
21                double Length() {return length;} // Metodo get para o
                     comprimento do reservatorio
22                void Length (double _length) {length = _length;} //
                     Metodo set para o comprimento do reservatorio
23
24                double Width() const {return width;} // Metodo get para
                     a largura do reservatorio
25                void Width (double _width) {width = _width;} // Metodo
                     set para a largura do reservatorio
26
27                virtual void Area() override;
28
29};
30
31#endif
```

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

Listing 6.10: Arquivo de implementação da classe CLinear

```
#include <iostream>
#include <fstream>

#include "CLinear.h"

using namespace std;

CLinear::CLinear(): CGeometria() {}

void CLinear::Area() {

    area = width * length;

}
```

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

Listing 6.11: Arquivo de cabeçalho da classe CRadial

```
#ifndef CRadial_h
#define CRadial_h

#include <iostream>
#include <cmath>
#include <fstream>

#include "CGeometria.h"

class CRadial: public CGeometria {

        protected:

                double raioExterno; // Raio da rocha reservatorio

        public:

                CRadial() {}; // Construtor default
                ~CRadial() {}; // Destrutor

                double RaioExterno() const {return raioExterno;} //
                    Metodo get para o raio do reservatorio
                void RaioExterno (double _raioExterno) {raioExterno =
                    _raioExterno;} // Metodo set para o raio do
                    reservatorio

                virtual void Area() override;

};

```

```
28#endif
```

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

Listing 6.12: Arquivo de implementação da classe CRadial

```
1#include <iostream>
2#include <fstream>
3
4#include <cmath>
5
6#include "CRadial.h"
7
8using namespace std;
9
10void CRadial::Area() {
11
12        area = M_PI*pow(raioExterno, 2.0);
13
14}
```

Apresenta-se na listagem 6.13o arquivo com código da classe `CTipoGeometria`.

Listing 6.13: Arquivo de cabeçalho da classe CTipoGeometria

```
1#ifndef CTipoGeometria_h
2#define CTipoGeometria_h
3
4#include <iostream>
5
6enum class CTipoGeometria {
7
8        none = 0, linear, radial // Definicao da geometria por
             enumeracao
9
10       //O usuario fica aberto a adicionar mais geometrias
11
12};
13
14#endif
```

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

Listing 6.14: Arquivo de cabeçalho da classe CPoco

```
1#ifndef CPoco_h
2#define CPoco_h
3
4#include <iostream>
5
6class CPoco{
7
```

```
8         protected:
9
10                double raioPoco; // Raio do poco
11                double pwf; // Pressao de fundo
12        double area; // Area do poco
13        double vazaoProducao; // Vazao de producao
14
15        public:
16
17                CPoco(){}; // Construtor
18                ~CPoco(){}; // Destrutor
19
20                void CalcArea(); // Metodo de calculo da area
21        double getArea() {return area; }
22
23                double getRaioPoco() const {return raioPoco;} // Metodo
                      get para o raio do poco
24                void setRaioPoco (double _raioPoco) {raioPoco =
                      _raioPoco;} // Metodo set para o raio do poco
25
26                double getPressao() const {return pwf;} // Metodo get
                      para a pressao de fundo
27                void setPressao (double _pwf) {pwf = _pwf;} // Metodo
                      set para a pressao de fundo
28
29        double getVazaoProducao() {return vazaoProducao;} // Metodo get
             para a vazao de producao
30        void setVazaoProducao(double _vazaoProducao) { vazaoProducao =
             _vazaoProducao;} // Metodo set para a vazao de producao
31
32};
33#endif
```

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

Listing 6.15: Arquivo de implementação da classe CPoco

```
1#include "CPoco.h"
2#include<iostream>
3#include<cmath>
4
5void CPoco::CalcArea(){
6
7         area = (2.0 * M_PI * raioPoco * raioPoco) / 4.0; // CÃ¡lculo
             area do poco
8
9};
```

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

Listing 6.16: Arquivo de cabeçalho da classe CIPR

```
#ifndef CIPR_H
#define CIPR_H

#include "CFluido.h"
#include "CReservatorio.h"
#include "CPoco.h"
#include <vector>

class CIPR
{

public:

    CIPR();
    virtual void CalcIPR(CFluido* fluido, CReservatorio& reservatorio,
       CPoco& poco, CPoco& parametrosSegundoPoco);
    virtual void CalcIPR(CFluido* fluido, CReservatorio& reservatorio,
       CPoco& poco);
    std::vector<double> getPWF() {return valoresPwf;}
    void setPWF(std::vector<double> _valoresPwf) { valoresPwf =
       _valoresPwf; }
    std::vector<double> getVazao() {return vazao; }
    void iprVariacaoPwf(double _pwf);
    void setPartes(double _partes) { partes = _partes; }

protected:

    std::vector<double> valoresPwf;
    std::vector<double> vazao ;
    double partes;


};

#endif
```

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

Listing 6.17: Arquivo de implementação da classe CIPR

```
#include "CIPR.h"
#include <vector>

CIPR::CIPR()
{
}

// Declarando e definindo o metodo CalcIPR na classe CIPR
```

```
9 void CIPR::CalcIPR(CFluido* fluido, CReservatorio& reservatorio, CPoco&
    poco, CPoco& parametrosSegundoPoco) {
10
11 }
12
13 void CIPR::CalcIPR(CFluido *fluido, CReservatorio &reservatorio, CPoco &
    poco)
14 {
15 }
16
17 void CIPR::iprVariacaoPwf(double _pwf) // Funcao responsavel pela
    variacao de Pwf
18 {
19
20   double deltaP = _pwf/partes; // seta o Delta P que eh obtido pelo
       usuario. Divisao de Pwf e q sao obtidas pelo usuario
21
22   while ( _pwf >= 0)
23   {
24       valoresPwf.push_back(_pwf); //Pwf armazenado eh decrementado em
           Delta P
25       _pwf -= deltaP;
26
27    if(_pwf < 0)
28       valoresPwf.push_back(0.0);
29
30   }
31
32 }
```

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

Listing 6.18: Arquivo de cabeçalho da classe CIPRLinear

```
1 #ifndef MODELOIPRLINEAR_H
2 #define MODELOIPRLINEAR_H
3
4 #include "CIPR.h"
5
6 class CIPRLinear : public CIPR
7 {
8
9 public:
10     CIPRLinear(){}; // Construtor padrao
11     void CalcIPR(CFluido* fluido, CReservatorio& reservatorio, CPoco&
          poco) override;
12
13 };
14
```

```
15#endif
```

Apresenta-se na listagem 6.19 o arquivo de implementação da classe `CIPRLinear`.

Listing 6.19: Arquivo de implementação da classe CIPRLinear

```
1#include "CIPRLinear.h"
2#include <iostream>
3#include <cmath>
4
5void CIPRLinear::CalcIPR(CFluido* fluido, CReservatorio& reservatorio,
    CPoco& poco) {
6
7    double j = reservatorio.getIndiceProdutividade();
8    double pi = reservatorio.PressaoInicial();
9    double pb = reservatorio.PressaoBolha();
10
11    iprVariacaoPwf(pi);  // Inicializa a classe CIPRVariacaoPwf com
       parametro pi
12
13        // Uma estrutura condicional verifica se a pressao inicial eh
           maior que a pressao de bolha
14    if (pi > pb) {
15        // Loop para calcular a vazao para cada valor de Pwf
16        for (double pwf : valoresPwf) {
17            vazao.push_back(j * (pi - pwf));
18
19            std::cout << "Pwf:␣" << pwf << ",␣Vazao:␣" << j * (pi - pwf)
                  << std::endl;
20        }
21    } else {
22        // Mostrar mensagem de erro e encerrar o codigo
23        std::cout << "A␣pressao␣do␣reservatorio␣(pi)␣eh␣menor␣ou␣igual␣a
              ␣pressao␣de␣bolha␣(pb).␣Selecione␣um␣valor␣acima␣de␣pb." <<
              std::endl;
24        exit(1); // Encerra o programa com um codigo de erro
25    }
26
27}
```

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

Listing 6.20: Arquivo de cabeçalho da classe CIPRFetkovich

```
1#ifndef CFetkovich_h
2#define CFetkovich_h
3
4#include "CIPR.h"
5
6class CIPRFetkovich : public CIPR{
```

```
7public:
8    CIPRFetkovich(){};
9    virtual void CalcIPR(CFluido* fluido, CReservatorio& reservatorio,
        CPoco& poco, CPoco& pocoDois) override;
10
11};
12
13#endif
```

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

Listing 6.21: Arquivo de implementação da classe CIPRFetkovich

```
1#include "CIPRFetkovich.h"
2#include <cmath>
3#include <iostream>
4
5using namespace std;
6
7void CIPRFetkovich::CalcIPR(CFluido* fluido, CReservatorio& reservatorio
    , CPoco& poco, CPoco& pocoDois)
8{
9
10    double pi = reservatorio.PressaoInicial();
11        double pwf = pi; // Inicializa o valor de pwf como pi
12
13    // Solicita os valores de teste do usuario
14    double pwf1, pwf2, q1, q2;
15
16
17    pwf1 = poco.getPressao();
18
19
20    pwf2 = pocoDois.getPressao();
21
22
23    q1 = poco.getVazaoProducao();
24
25    q2 = pocoDois.getVazaoProducao();
26
27        //Definicao das constantes empiricas n e C
28    double n = ( log(q1/q2) ) / ( log ( ( pow (pi,2) - pow(pwf1,2) )/ (
        pow(pi,2) - pow(pwf2,2) ) ) );
29    double C = q1 / pow( (pow(pi,2) - pow(pwf1,2)), n);
30
31        // Calculo de qmax com os valores de teste
32    double qmax = C * pow(pi, 2*n);
33
34
```

```
35      iprVariacaoPwf(pwf);
36
37      // Loop para variar pwf
38      for (double pwf : valoresPwf) {
39          if (pwf == pi) {
40              vazao.push_back(0); // Vazao zero quando pwf eh igual a pi
41          } else if (pwf == 0) {
42              vazao.push_back(qmax); // Vazao maxima quando pwf eh igual a
                    zero
43          } else {
44              vazao.push_back(qmax * (1 - pow(pwf/pi, 2 ) ));
45          }
46          cout << "Pwf: " << pwf << ", Vazao: " << qmax * (1 - pow(pwf/pi,
                2 ) ) << endl;
47      }
48  }
```

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

Listing 6.22: Arquivo de cabeçalho da classe CIPRVogel

```
1 #ifndef CVogel_H
2 #define CVogel_H
3
4 #include "CFluido.h"
5 #include "CReservatorio.h"
6 #include "CPoco.h"
7 #include "CIPR.h"
8
9 class CIPRVogel : public CIPR {
10 public:
11     CIPRVogel(){};
12     virtual void CalcIPR(CFluido* fluido, CReservatorio& reservatorio,
           CPoco& poco) override;
13
14 };
15
16 #endif
```

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

Listing 6.23: Arquivo de implementação da classe CIPRVogel

```
1 #include "CIPRVogel.h"
2 #include <cmath>
3 #include <iostream>
4
5 using namespace std;
6
7
```

```
8 void CIPRVogel::CalcIPR(CFluido* fluido, CReservatorio& reservatorio,
      CPoco& poco) {
9     double pi = reservatorio.PressaoInicial(); // Pressao inicial do
          reservatorio
10    double j = reservatorio.getIndiceProdutividade(); //Indice de
          Produtividade
11
12    // Calculo de qmax com os valores de teste
13    double qmax = j*(pi)/1.8;
14
15    // Inicializa a classe CIPRVariacaoPwf
16    iprVariacaoPwf(pi);
17
18    // Loop para variar pwf
19    for (double pwf : valoresPwf) {
20        double q;
21        if (pwf == pi) {
22            vazao.push_back(0); // Vazao zero quando pwf eh igual a pi
23        } else if (pwf == 0) {
24            vazao.push_back(qmax); // Vazao maxima quando pwf eh igual a
                  zero
25        } else {
26            vazao.push_back(qmax * (1 - 0.2 * (pwf / pi) - 0.8 * pow(pwf
                  / pi, 2)));
27        }
28        cout << "Pwf: " << pwf << ", Vazao: " << (qmax * (1 - 0.2 * (pwf
              / pi) - 0.8 * pow(pwf / pi, 2))) << endl;
29    }
30
31 }
```

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

Listing 6.24: Arquivo de cabeçalho da classe CIPRGeneralizada

```
1 #ifndef CIPRGeneralizada_H
2 #define CIPRGeneralizada_H
3
4 #include "CIPR.h"
5
6 class CIPRGeneralizada : public CIPR {
7 public:
8     CIPRGeneralizada(){};
9     virtual void CalcIPR(CFluido* fluido, CReservatorio& reservatorio,
          CPoco& poco);
10
11 };
12
13 #endif // MODELOIPRGENERALIZADA_H
```

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

Listing 6.25: Arquivo de implementação da classe CIPRGeneralizada

```
#include "CIPRGeneralizada.h"
#include<iostream>
#include <cmath>

void CIPRGeneralizada::CalcIPR(CFluido* fluido, CReservatorio&
    reservatorio, CPoco& poco) {
    double j = reservatorio.getIndiceProdutividade();
    double pi = reservatorio.PressaoInicial();
    double pb = reservatorio.PressaoBolha();

    // Inicializa a classe CIPRVariacaoPwf
    iprVariacaoPwf(pi);

    if (pi >= pb) {
        // Loop para calcular a vazao para cada valor de Pwf
        for (double pwf : valoresPwf) {
            vazao.push_back(j * (pi - pb) + ((j * pb) / 1.8) * (1.0 -
                0.2 * (pwf / pb) - 0.8 * ((pwf / pb) * (pwf / pb))));

            std::cout << "Pwf: " << pwf << ", Vazao: " << j * (pi - pwf)
                << std::endl;
        }
    } else {
        for (double pwf : valoresPwf) {
            vazao.push_back(j * (pi - pwf));

            std::cout << "Pwf: " << pwf << ", Vazao: " << j * (pi - pwf)
                << std::endl;
        }
    }

}
```

Apresenta-se na listagem **??**o arquivo com código da classe `CMetodo`.

Listing 6.26: Arquivo de cabeçalho da classe CReservatorio

```
#ifndef CReservatorio_h
#define CReservatorio_h

// Inclusao dos arquivos de cabecalho

#include "CRocha.h"
#include "CGeometria.h"
#include "CLinear.h"
#include "CRadial.h"
```

```
10#include "CTipoGeometria.h"
11#include "CTipoFluido.h"
12#include "CFluido.h"
13#include "COleo.h"
14#include "CGas.h"
15#include "CPoco.h"
16
17#include <iostream>
18
19// Classe CReservatorio que representa o reservatorio
20
21class CReservatorio: public CRocha {
22
23        protected:
24
25                double espessura; // Espessura do reservatorio
26                double EulerCte = 1.781; // Constante de Euler
27                double fatorPelicula; // Fator de pelicula do
                     reservatorio
28                double pressaoBolha; // Pressao de bolha do reservatorio
29                double pressaoInicial; // Pressao media inicial do
                     reservatorio
30                double temperatura; // Temperatura
31                double coeficienteDietz; // Coeficiente de Dietz
32                double indiceProdutividade; // Indice de Produtividade
                     do reservatorio
33                CTipoGeometria enumeracaoGeometria; // Enumeracao para
                     escolha do tipo de geometria
34                CTipoFluido enumeracaoFluido; // Enumeracao para escolha
                      do tipo de fluido
35                CPoco poco; // Objeto de CPoco associado a reservatorio
36
37        public:
38
39                CReservatorio () {} // Construtor default
40
41                ~CReservatorio () {} // Destrutor
42
43                double Espessura () const {return espessura;} // Metodo
                     get para a espessura
44                void Espessura (double _espessura) {espessura =
                     _espessura;} // Metodo set para a espessura
45
46                double FatorPelicula() const {return fatorPelicula;} //
                     Metodo get para a fatorPelicula
47                void FatorPelicula (double _fatorPelicula) {
                     fatorPelicula = _fatorPelicula;} // Metodo set para a
                      fatorPelicula
```

```
48
49                 double PressaoBolha() const {return pressaoBolha;} //
                       Metodo get para a pressaoBolha
50                 void PressaoBolha (double _pressaoBolha) {pressaoBolha =
                       _pressaoBolha;} // Metodo set para a pressaoBolha
51
52                 double PressaoInicial() const {return pressaoInicial;}
                       // Metodo get para a pressaoInicial
53                 void PressaoInicial (double _pressaoInicial) {
                       pressaoInicial = _pressaoInicial;} // Metodo set para
                        a pressaoInicial
54
55                 double Temperatura() const {return temperatura;} //
                       Metodo get para a temperatura
56                 void Temperatura (double _temperatura) {temperatura =
                       _temperatura;} // Metodo set para a temperatura
57
58                 void setGeometria(CTipoGeometria _enumeracaoGeometria);
                       // Metodo set para o tipo de geometria
59
60                 void setFluido(CTipoFluido _enumeracaoFluido); // Metodo
                        set para o tipo de fluido
61
62                 void setFatorVolumeFormacao(double _fatorVolumeFormacao)
                       ; // Metodo set para o fator volume-formacao
63
64                 void CoeficienteDietz(CTipoGeometria
                       _enumeracaoGeometria); // Metodo para calculo do
                       Coeficiente de Dietz a ser utilizado
65         double getCoeficienteDietz() { return coeficienteDietz; }
66
67                 void IndiceProdutividade(CFluido* _fluido, CGeometria*
                       _geometria, CPoco& _poco); // Metodo para calculo do
                       Indice de Produtividade
68         double getIndiceProdutividade() {return indiceProdutividade; }
69
70                 double VazaoMaxima() {return ((indiceProdutividade *
                       pressaoInicial)/1.8);} // Metodo para calculo da
                       maxima vazao obtida pelo reservatorio (AOF)
71
72};
73
74#endif
```

Apresenta-se na listagem 6.27o arquivo com código da classe `CSimuladorCurvasIPR`.

Listing 6.27: Arquivo de cabeçalho da classe CSimuladorCurvasIPR

```
1#ifndef CSimuladorCurvasIPR_h
```

```
2#define CSimuladorCurvasIPR_h
3
4#include <iostream>
5#include <string>
6#include <vector>
7
8#include "CPoco.h"
9#include "CFluido.h"
10#include "COleo.h"
11#include "CGas.h"
12#include "CGeometria.h"
13#include "CLinear.h"
14#include "CRadial.h"
15#include "CReservatorio.h"
16#include "CRocha.h"
17#include "CIPRLinear.h"
18#include "CIPRFetkovich.h"
19#include "CIPRVogel.h"
20#include "CIPRGeneralizada.h"
21#include "CIPR.h"
22#include "CGnuplot.h"
23#include "CMetodo.h"
24#include "CTipoGeometria.h"
25
26class CSimuladorCurvasIPR {
27
28        private:
29
30        CReservatorio reservatorio; // Criando objeto reservatorio
31        CPoco poco; // Criando objeto poco
32        CPoco pocoDois; // Criando segundo objeto poco para IPR do tipo
             Fetkovich
33        CIPR* IPR; // Ponteiro para apontar para o tipo de metodo
34        CGeometria* forma; // Ponteiro para apontar a geometria do
             reservatorio
35        CFluido* fluido; // Ponteiro para apontar o tipo de fluido: gas
             ou oleo
36        Gnuplot plot; // Criando objeto plot
37        CMetodo metodo; // Criando objeto para enumeracao do metodo
38
39        std::vector<double> vazoes; // Criando vetor de vazoes
40        std::vector<double> pressoes; // Criando vetor de pressoes de
             fundo
41
42        public:
43
44                CSimuladorCurvasIPR(){}; // Construtor
45                ~CSimuladorCurvasIPR(); // Destrutor
```

```
46
47                 void EntradaDados(); // Metodo para entrada de dados do
                       usuario
48
49         void Calculos(); // Metodo para calculos da simulacao
50
51                 void Plotar(); // Metodo para plotar as curvas
52
53                 void SalvarGrafico(); // Metodo para salvar o grafico
54
55                 void Executar(); // Metodo para execucao do programa
56
57};
58
59#endif
```

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

Listing 6.28: Arquivo de implementação da classe CSimuladorCurvasIPR

```
1#include "CSimuladorCurvasIPR.h"
2
3#include <iostream>
4
5using namespace std;
6
7// Sobrecarga de operador >> para o metodos
8
9istream &operator>>(istream &is, CMetodo &metodo) {
10
11   unsigned int a;
12   is >> a;
13   metodo = static_cast<CMetodo>(a);
14
15   return is;
16
17}
18
19// Sobrecarga de operador >> para o tipo de fluido
20
21istream &operator>>(istream &is, CTipoFluido &fluido) {
22
23   unsigned int a;
24   is >> a;
25   fluido = static_cast<CTipoFluido>(a);
26
27   return is;
28
29}
```

```
30
31// Sobrecarga de operador >> para geometria do reservatorio
32istream &operator>>(istream &is, CTipoGeometria &geometria) {
33
34   unsigned int a;
35   is >> a;
36   geometria = static_cast<CTipoGeometria>(a);
37
38   return is;
39
40}
41
42CSimuladorCurvasIPR::~CSimuladorCurvasIPR() {
43
44    delete forma;
45    forma = nullptr;
46
47}
48
49void CSimuladorCurvasIPR::EntradaDados() {
50
51    cout << endl;
52    cout << "---------------␣SIMULADOR␣DE␣CURVAS␣IPR␣PARA␣POCOS␣
         VERTICAIS␣---------------" << endl;
53    cout << "
         --------------------------------------------------------------------------
         " << endl;
54    cout << "------------␣Atila␣Junior,␣Giovanna␣Massardi␣e␣Marcelo␣
         Bernardo␣------------" << endl;
55    cout << "--------------------------------␣UENF/LENEP␣
         --------------------------------" << endl;
56    cout << endl;
57
58        cout << "Qual␣o␣metodo␣sera␣utilizado?" << endl;
59    cout << "1␣-␣IPR␣Linear" << endl;
60    cout << "2␣-␣IPR␣Generalizada" << endl;
61    cout << "3␣-␣IPR␣de␣Fetkovich␣(Ideal␣para␣reservatorios␣de␣gas)" <<
         endl;
62    cout << "4␣-␣IPR␣de␣Vogel" << endl;
63    cout << "
         --------------------------------------------------------------------------
         " << endl;
64
65    cin >> metodo; // Recebe o tipo de metodo que o usuario entrar
66
67    cin.get();
68
69        // Switch para o tipo de metodo de calculo
```

```
70      switch (metodo) {
71
72          case CMetodo::IPR_LINEAR:
73
74                  {
75
76                      CTipoGeometria geometria = CTipoGeometria::linear;
77              cin.get();
78
79              forma = new CLinear;
80              CLinear* cast = dynamic_cast<CLinear*>(forma);
81
82              reservatorio.CoeficienteDietz(geometria);
83
84              cout << "Entre com o comprimento do reservatorio (FT)" <<
                    endl;
85
86              double esp;
87
88              cin >> esp;
89
90              cout << "Entre com a largura do reservatorio (FT)" << endl;
91
92              double lar;
93
94              cin >> lar;
95
96              cast-> Length(esp);
97              cast-> Width(lar);
98              cast-> Area();
99
100             cout << "Entre com o raio do poco (FT):" << endl;
101
102             double raio;
103
104             cin >> raio;
105
106             poco.setRaioPoco(raio);
107
108             cout << "Entre com a pressao inicial do reservatorio (PSIA):
                    " << endl;
109
110             double pi;
111
112             cin >> pi;
113
114             reservatorio.PressaoInicial(pi);
115
```

```
116             cout << "Entre com a pressao de bolha do reservatorio (PSIA)
                    : " << endl;
117
118             double pb;
119
120             cin >> pb;
121
122             reservatorio.PressaoBolha(pb);
123
124             poco.CalcArea();
125
126             cout << "Qual tipo de fluido presente no reservatorio?" <<
                    std::endl;
127             cout << "1 - Oleo" << endl;
128             cout << "2 - Gas" << endl;
129             cout << "
                    ------------------------------------------------------------------
                    " << endl;
130
131             CTipoFluido tipoFluido;
132
133             cin >> tipoFluido;
134             cin.get();
135
136             bool chs = true;
137
138             while(chs)
139
140                         {
141
142                                 switch (tipoFluido) // Switch para
                                        definicao do tipo de fluido
143
144                                 {
145
146                                         case CTipoFluido::oleo: // Caso
                                                oleo, cria um novo objeto da
                                                classe COleo
147
148                                                 {
149
150                         fluido = new COleo;
151                         cout << "Entre com fator volume de formacao do
                                oleo: " << endl;
152                         double Bo;
153                         cin >> Bo;
154                         fluido->FatorVolumeFormacao(Bo);
155                         cout << "Entre com a viscosidade do fluido (CP):
```

```
                    " << endl;
156                 double viscosidade;
157                 cin >> viscosidade;
158                 fluido->Viscosidade(viscosidade);
159
160                 chs = false;
161                 break;
162
163                                 }
164
165                             case CTipoFluido::gas: // Caso
                                    gas, cria um novo objeto da
                                    classe CGas
166
167                                 {
168
169                 fluido = new CGas;
170                 cout << "Entre com fator volume de formacao do
                        gas:" << endl;
171                 double Bg;
172                 cin >> Bg;
173                 fluido->FatorVolumeFormacao(Bg);
174                 cout << "Entre com a viscosidade do fluido (CP):
                        " << endl;
175                 double viscosidade;
176                 cin >> viscosidade;
177                 fluido->Viscosidade(viscosidade);
178
179                 chs = false;
180                 break;
181
182                                 }
183
184                             default:
185                 cout << "Entrada invalida!" << endl;
186                 cout << "Qual tipo de fluido presente no
                        reservatorio? " << endl;
187                 cin >> tipoFluido;
188                 cin.get();
189
190                         }
191
192         }
193
194         cout << "Entre com a espessura do reservatorio (FT):" <<
                endl;
195
196         double espreservatorio;
```

```
197
198             cin >> espreservatorio;
199
200             reservatorio.Espessura(espreservatorio);
201
202             cout << "Entre com a permeabilidade da rocha reservatorio (
                    MD):" << endl;
203
204             double perm;
205
206             cin >> perm;
207
208             reservatorio.Permeabilidade(perm);
209
210             cout << "Entre com fator de pelicula do reservatorio:" <<
                    endl;
211
212             double s;
213
214             cin >> s;
215
216             reservatorio.FatorPelicula(s);
217
218             reservatorio.IndiceProdutividade(fluido, forma, poco);
219
220             cout << "IP = " << reservatorio.getIndiceProdutividade() <<
                    endl << endl;
221
222             IPR = new CIPRLinear();
223
224             break;
225
226                 }
227
228                 case CMetodo::IPR_GENERALIZADA:
229
230                 {
231
232                         CTipoGeometria geometria = CTipoGeometria::
                            radial; // Caso geometria radial cria objeto
                            do tipo CRadial
233             cin.get();
234
235             forma = new CRadial;
236             CRadial* cast = dynamic_cast<CRadial*>(forma);
237
238             reservatorio.CoeficienteDietz(geometria);
239
```

```
240             cout << "Entre␣com␣o␣raio␣externo␣do␣reservatorio␣(FT)" <<
                    endl;
241
242             double Re;
243
244             cin >> Re;
245
246             cast->RaioExterno(Re);
247             cast->Area();
248
249             cout << "Entre␣com␣a␣raio␣do␣poco␣(FT)" << endl;
250
251             double raio;
252
253             cin >> raio;
254
255             poco.setRaioPoco(raio);
256
257             cout << "Entre␣com␣a␣pressao␣inicial␣(PSIA):" << endl;
258
259             double pi;
260
261             cin >> pi;
262
263             reservatorio.PressaoInicial(pi);
264
265             cout << "Entre␣com␣a␣pressao␣de␣bolha␣(PSIA):␣" << endl;
266
267             double pb;
268
269             cin >> pb;
270
271             reservatorio.PressaoBolha(pb);
272
273             poco.CalcArea();
274
275             cout << "Qual␣tipo␣de␣fluido␣presente␣no␣reservatorio?" <<
                    endl;
276             cout << "1␣-␣Oleo" << endl;
277             cout << "2␣-␣Gas" << endl;
278             cout << "
                    ----------------------------------------------------------------
                    " << endl;
279
280             CTipoFluido tipoFluido;
281
282             cin >> tipoFluido;
283             cin.get();
```

```
284
285             bool chs = true;
286
287             while(chs)
288
289                         {
290
291                                 switch (tipoFluido)
292
293                                 {
294
295                                         case CTipoFluido::oleo:
296
297                                                 {
298
299                         fluido = new COleo;
300                         cout << "Entre com fator volume de formacao do
                               oleo:" << endl;
301                         double Bo;
302                         cin >> Bo;
303                         fluido->FatorVolumeFormacao(Bo);
304                         cout << "Entre com a viscosidade do fluido (CP):
                               " << endl;
305                         double viscosidade;
306                         cin >> viscosidade;
307                         fluido->Viscosidade(viscosidade);
308
309                         chs = false;
310                         break;
311
312                                                 }
313
314                                         case CTipoFluido::gas:
315
316                                                 {
317
318                         fluido = new CGas;
319                         cout << "Entre com fator volume de formacao do
                               gas:" << endl;
320                         double Bg;
321                         cin >> Bg;
322                         fluido->FatorVolumeFormacao(Bg);
323                         cout << "Entre com a viscosidade do fluido (CP):
                               " << endl;
324                         double viscosidade;
325                         cin >> viscosidade;
326                         fluido->Viscosidade(viscosidade);
327
```

```
328                         chs = false;
329                         break;
330 
331                                             }
332 
333                                         default:
334                         cout << "Entrada invalida!" << endl;
335                         cout << "Qual tipo de fluido presente no
                               reservatorio?" << endl;
336                         cin >> tipoFluido;
337                         cin.get();
338 
339                                 }
340 
341             }
342 
343             cout << "Entre com a espessura do reservatorio (FT):" <<
                    endl;
344 
345             double espreservatorio;
346 
347             cin >> espreservatorio;
348 
349             reservatorio.Espessura(espreservatorio);
350 
351             cout << "Entre com a permeabilidade da rocha (MD):" << endl;
352 
353             double perm;
354 
355             cin >> perm;
356 
357             reservatorio.Permeabilidade(perm);
358 
359             cout << "Entre com fator de pelicula do reservatorio:" <<
                    endl;
360 
361             double s;
362 
363             cin >> s;
364 
365             reservatorio.FatorPelicula(s);
366 
367             reservatorio.IndiceProdutividade(fluido, forma, poco);
368 
369             IPR = new CIPRGeneralizada();
370 
371             break;
372 
```

```
373                 }
374
375                 case CMetodo::IPR_FETKOVICH:
376
377                         {
378
379                         CTipoGeometria geometria = CTipoGeometria::
                               radial;
380             cin.get();
381
382             forma = new CRadial;
383             CRadial* cast = dynamic_cast<CRadial*>(forma);
384
385             reservatorio.CoeficienteDietz(geometria);
386
387             cout << "Entre␣com␣o␣raio␣externo␣do␣reservatorio␣(FT)" <<
                    endl;
388
389             double Re;
390
391             cin >> Re;
392
393             cast->RaioExterno(Re);
394             cast->Area();
395
396             cout << "Entre␣com␣a␣raio␣do␣poco␣." << endl;
397
398             double raio;
399
400             cin >> raio;
401
402             poco.setRaioPoco(raio);
403
404             cout << "Entre␣com␣a␣pressao␣media␣do␣reservatorio␣(PSIA):␣"
                        << endl;
405
406             double pi;
407
408             cin >> pi;
409
410             reservatorio.PressaoInicial(pi);
411
412             cout << "Entre␣com␣a␣pressao␣de␣bolha␣(PSIA):␣" << endl;
413
414             double pb;
415
416             cin >> pb;
417
```

```
418             reservatorio.PressaoBolha(pb);
419
420
421             cout << "Entre␣com␣a␣pressao␣de␣fundo␣no␣poco␣A␣(PSIA):" <<
                    endl;
422
423             double pwf1;
424
425             cin >> pwf1;
426
427             poco.setPressao(pwf1);
428
429             cout << "Entre␣com␣a␣vazao␣no␣poco␣A␣(STB/d):" << endl;
430
431             double q1;
432
433             cin >> q1;
434
435             poco.setVazaoProducao(q1);
436
437             cout << "Entre␣com␣a␣pressao␣de␣fundo␣no␣poco␣B␣(PSIA):" <<
                    endl;
438
439             double pwf2;
440
441             cin >> pwf2;
442
443             pocoDois.setPressao(pwf2);
444
445             cout << "Entre␣com␣a␣vazao␣no␣poco␣B␣(STB/d):" << endl;
446
447             double q2;
448
449             cin >> q2;
450
451             pocoDois.setVazaoProducao(q2);
452
453             poco.CalcArea();
454
455             cout << "Qual␣tipo␣de␣fluido␣presente␣no␣reservatorio?" <<
                    endl;
456             cout << "1␣-␣Oleo" << endl;
457             cout << "2␣-␣Gas" << endl;
458             cout << "
                    ----------------------------------------------------------------
                    " << endl;
459
460             CTipoFluido tipoFluido;
```

```
461
462             cin >> tipoFluido;
463             cin.get();
464
465             bool chs = true;
466
467             while(chs)
468
469                         {
470
471                                 switch (tipoFluido)
472
473                                 {
474                     case CTipoFluido::oleo:
475
476                                                         {
477
478                         fluido = new COleo;
479                         cout << "Entre com fator volume de formacao do
                               oleo:" << endl;
480                         double Bo;
481                         cin >> Bo;
482                         fluido->FatorVolumeFormacao(Bo);
483                         cout << "Entre com a viscosidade do fluido (CP):
                               " << endl;
484                         double viscosidade;
485                         cin >> viscosidade;
486                         fluido->Viscosidade(viscosidade);
487
488                         chs = false;
489                         break;
490
491                                                         }
492
493                                         case CTipoFluido::gas:
494
495                                                         {
496
497                         fluido = new CGas;
498                         cout << "Entre com fator volume de formacao do
                               gas:" << endl;
499                         double Bg;
500                         cin >> Bg;
501                         fluido->FatorVolumeFormacao(Bg);
502                         cout << "Entre com a viscosidade do fluido (CP):
                               " << endl;
503                         double viscosidade;
504                         cin >> viscosidade;
```

```
505                         fluido->Viscosidade(viscosidade);
506
507                         chs = false;
508                         break;
509
510                                             }
511
512                                     default:
513                         cout << "Entrada invalida!" << endl;
514                         cout << "Qual tipo de fluido presente no
                               reservatorio?" << endl;
515                         cin >> tipoFluido;
516                         cin.get();
517
518                                 }
519
520             }
521
522             cout << "Entre com a espessura do reservatorio (FT):" <<
                    endl;
523
524             double espreservatorio;
525
526             cin >> espreservatorio;
527
528             reservatorio.Espessura(espreservatorio);
529
530             cout << "Entre com a permeabilidade da rocha reservatorio (
                    MD):" << endl;
531
532             double perm;
533
534             cin >> perm;
535
536             reservatorio.Permeabilidade(perm);
537
538             cout << "Entre com fator de pelicula do reservatorio:" <<
                    endl;
539
540             double s;
541
542             cin >> s;
543
544             reservatorio.FatorPelicula(s);
545
546             reservatorio.IndiceProdutividade(fluido, forma, poco);
547
548             IPR = new CIPRFetkovich();
```

```
549
550             break;
551
552                 }
553
554                 case CMetodo::IPR_VOGEL:
555
556                         {
557
558                         CTipoGeometria geometria = CTipoGeometria::
                            radial;
559             cin.get();
560
561             forma = new CRadial;
562             CRadial* cast = dynamic_cast<CRadial*>(forma);
563
564             reservatorio.CoeficienteDietz(geometria);
565
566             cout << "Entre␣com␣o␣raio␣externo␣do␣reservatorio␣(FT)" <<
                    endl;
567
568             double Re;
569
570             cin >> Re;
571
572             cast->RaioExterno(Re);
573             cast->Area();
574
575             cout << "Entre␣com␣a␣raio␣do␣poco␣(FT):" << endl;
576
577             double raio;
578
579             cin >> raio;
580
581             poco.setRaioPoco(raio);
582
583             cout << "Entre␣com␣a␣pressao␣inicial␣do␣reservatorio␣(PSIA):
                    " << endl;
584
585             double pi;
586
587             cin >> pi;
588
589             reservatorio.PressaoInicial(pi);
590
591             cout << "Entre␣com␣a␣pressao␣de␣bolha␣(PSIA):␣" << endl;
592
593             double pb;
```

```
594
595             cin >> pb;
596
597             reservatorio.PressaoBolha(pb);
598
599             poco.CalcArea();
600
601             cout << "Qual tipo de fluido presente no reservatorio?" <<
                    endl;
602             cout << "1 - Oleo" << endl;
603             cout << "2 - Gas" << endl;
604             cout << "
                    ------------------------------------------------------------------
                    " << endl;
605
606             CTipoFluido tipoFluido;
607
608             cin >> tipoFluido;
609             cin.get();
610
611             bool chs = true;
612
613             while(chs)
614
615                         {
616
617                                 switch (tipoFluido)
618
619                                 {
620
621                                         case CTipoFluido::oleo:
622
623                                                 {
624
625                         fluido = new COleo;
626                         cout << "Entre com fator volume de formacao do
                                oleo:" << endl;
627                         double Bo;
628                         cin >> Bo;
629                         fluido->FatorVolumeFormacao(Bo);
630                         cout << "Entre com a viscosidade do fluido (CP):
                                " << endl;
631                         double viscosidade;
632                         cin >> viscosidade;
633                         fluido->Viscosidade(viscosidade);
634
635                         chs = false;
636                         break;
```

```
637
638                                                 }
639
640                                             case CTipoFluido::gas:
641
642                                                 {
643
644                         fluido = new CGas;
645                         cout << "Entre com fator volume de formacao do
                               gas:" << endl;
646                         double Bg;
647                         cin >> Bg;
648                         fluido->FatorVolumeFormacao(Bg);
649                         cout << "Entre com a viscosidade do fluido (CP):
                               " << endl;
650                         double viscosidade;
651                         cin >> viscosidade;
652                         fluido->Viscosidade(viscosidade);
653
654                         chs = false;
655                         break;
656
657                                             }
658
659                                         default:
660                         cout << "Entrada invalida!" << endl;
661                         cout << "Qual tipo de fluido presente no
                               reservatorio? " << endl;
662                         cin >> tipoFluido;
663                         cin.get();
664
665                                 }
666
667             }
668
669             cout << "Entre com a espessura do reservatorio (FT):" <<
                    endl;
670
671             double espreservatorio;
672
673             cin >> espreservatorio;
674
675             reservatorio.Espessura(espreservatorio);
676
677             cout << "Entre com a permeabilidade da rocha (MD):" << endl;
678
679             double perm;
680
```

```
681             cin >> perm;
682 
683             reservatorio.Permeabilidade(perm);
684 
685             cout << "Entre com fator de pelicula do reservatorio:" <<
                    endl;
686 
687             double s;
688 
689             cin >> s;
690 
691             reservatorio.FatorPelicula(s);
692 
693             reservatorio.IndiceProdutividade(fluido, forma, poco);
694 
695             IPR = new CIPRVogel();
696 
697             break;
698 
699                 }
700 
701                 default:
702             break;
703 
704     }
705 
706 }
707 
708 void CSimuladorCurvasIPR::Calculos() {
709 
710     cout << endl << endl;
711     cout << "Calculando IPR..." << endl;
712     cout << endl << endl;
713 
714     cout << "Deseja realizar o calculo de pressoes em quantas partes?"
           << endl; // Realizara os calculos com base na quantidade de
           pontos solicitados ate que atinja a pressao igual a zero
715     cout << "Digite um numero inteiro."  << endl;
716 
717     int partes; // Variavel que recebe a entrada do usuario
718 
719     cin >> partes;
720 
721     IPR->setPartes(partes);
722 
723     if(metodo == CMetodo::IPR_FETKOVICH)
724         IPR->CalcIPR(fluido, reservatorio, poco, pocoDois); // Para
               Fetkovich eh necessario dois pocos para calculo das
```

```
            constantes
725     else
726         IPR->CalcIPR(fluido, reservatorio, poco);
727 
728     vazoes = IPR->getVazao();
729     pressoes = IPR->getPWF();
730 
731 }
732 
733 // Metodo para plotar o grafico
734 
735 void CSimuladorCurvasIPR::Plotar() {
736 
737     cout << "Deseja plotar o grafico?" << std::endl;
738     cout << " 1 - Sim" << endl;
739     cout << " 2 - Nao " << endl;
740     cout << "
        ------------------------------------------------------------------------
        " << endl;
741 
742     char ans; // Variavel para receber resposta do usuario
743 
744     cin >> ans;
745     cin.get();
746 
747     bool boleano = true;
748 
749     while(boleano)
750 
751         {
752 
753         switch(ans)
754 
755                 {
756 
757                         case '1':
758 
759                         {
760 
761                 plot.XLabel("Vazao (STB/d)");
762                 plot.YLabel("Pressao de Fundo (PSIA)");
763                 plot.Title("Curva IPR");
764                 plot.Style("lines");
765                 plot.plot_xy(vazoes, pressoes);
766                 cin.get();
767                 boleano = false;
768                 break;
769 
```

```
770                         }
771
772                         case '2':
773                 boleano = false;
774                 break;
775             default:
776                 cout << "Opcao invalida!" << endl;
777
778                 }
779
780     }
781
782}
783
784// Metodo para salvar o grafico
785void CSimuladorCurvasIPR::SalvarGrafico() {
786
787     plot.savetops("GraficoIPR");
788
789}
790
791// Metodo para executar o software
792void CSimuladorCurvasIPR::Executar() {
793
794     bool run = true;
795
796     cout << endl;
797     cout << "---------------- SIMULADOR DE CURVAS IPR PARA POCOS
          VERTICAIS ----------------" << endl;
798     cout << "
          ------------------------------------------------------------------------
          " << endl;
799     cout << "------------ Atila Junior , Giovanna Massardi e Marcelo
          Bernardo ------------" << endl;
800     cout << "--------------------------------- UENF/LENEP
          ---------------------------------" << endl;
801     cout << endl;
802
803         EntradaDados();
804     Calculos();
805     Plotar();
806
807     cout << "Deseja salvar o grafico?" << endl;
808     cout << " 1 - Sim" << endl;
809         cout << " 2 - Nao " << endl;
810         cout << "
             ------------------------------------------------------------------------
             " << endl;
```

```
811
812     bool test = true;
813
814     char chse; // Variavel para receber a resposta do usuario
815
816     cin >> chse;
817     cin.get();
818
819     while(test)
820
821         {
822
823                 switch(chse)
824
825                 {
826
827                         case '1':
828
829                         {
830
831                                 SalvarGrafico();
832                 test = false;
833                 break;
834
835                         }
836
837                         case '2':
838
839                         {
840
841                                 test = false;
842                 break;
843
844                         }
845
846             // Caso usuario entre com uma opcao diferente das
                    apresentadas retorna mensagem de erro
847
848             default:
849
850                         {
851
852                                 cout << "Opcao Invalida!" << endl;
853                 cout << "Deseja salvar o grafico?" << endl;
854                         cout << " 1 - Sim" << endl;
855                                 cout << " 2 - Nao " << endl;
856                                 cout << "
                                    ----------------------------------------
```

```
                                        " << endl;
857                 cin >> chse;
858                 break;
859
860                         }
861
862                 }
863
864     }
865
866     cout << "Deseja␣realizar␣os␣calculos␣utilizando␣outro␣metodo?" <<
            endl;
867     cout << "1␣-␣Sim" << endl;
868     cout << "2␣-␣Nao␣" << endl;
869     cout << "
            ---------------------------------------------------------------------
            " << endl;
870
871     char ans; // Variavel para receber a resposta do usuario
872
873     cin >> ans;
874     cin.get();
875
876// Caso usuario entre com uma opcao diferente das apresentadas retorna
       mensagem de erro
877
878         while (run)
879
880         {
881
882                 while ( ans!= '1' && ans != '2')
883
884                 {
885
886                         cout << "Opcao␣invalida" << endl;
887                 cout << "Deseja␣realizar␣os␣calculos␣utilizando␣outro␣
                        metodo?" << endl;
888                 cout << "1␣-␣Sim" << endl;
889                 cout << "2␣-␣Nao␣" << endl;
890                 cout << "
                        ---------------------------------------------------------
                        " << endl;
891
892                 cin >> ans;
893                 cin.get();
894
895             }
896
```

```
897             switch(ans)
898
899                 {
900
901                     case '1': // Caso resposta positiva do usuario,
                            executa o programa
902
903                     {
904
905                             EntradaDados();
906                 Calculos();
907                 Plotar();
908
909                 cout << "Deseja salvar o grafico?" << endl;
910                 cout << " 1 - Sim" << endl;
911                     cout << " 2 - Nao " << endl;
912                     cout << "
                        ----------------------------------------------------
                        " << endl;
913
914                 bool test = true;
915
916                 char chse; // Variavel para receber a resposta do
                        usuario
917
918                 cin >> chse;
919                 cin.get();
920
921                 while(test)
922
923                         {
924
925                                 switch(chse)
926
927                                 {
928
929                                         case '1':
930
931                                         {
932
933                                                 SalvarGrafico();
934                             test = false;
935                             break;
936
937                                         }
938
939                                         case '2':
940
```

```
941                                         {
942
943                                             test = false;
944                         break;
945
946                                         }
947
948                     // Caso usuario entre com uma opcao
                           diferente das apresentadas retorna
                           mensagem de erro
949
950                     default:
951
952                                         {
953
954                                             cout << "Opcao␣
                                                   Invalida!" <<
                                                    endl;
955                         cout << "Deseja␣salvar␣o␣grafico?" <<
                               endl;
956                                 cout << "␣1␣-␣Sim" << endl;
957                                         cout << "␣2␣-␣Nao␣" <<
                                               endl;
958                                         cout << "
                                               ------------------------------
                                               " << endl;
959                         cin >> chse;
960                         break;
961
962                                         }
963
964                                 }
965
966             }
967
968             cout << "Deseja␣realizar␣os␣calculos␣utilizando␣
                   outro␣metodo?" << endl;
969             cout << "1␣-␣Sim" << endl;
970                 cout << "2␣-␣Nao␣" << endl;
971                 cout << "
                       ------------------------------------------------------
                       " << endl;
972             cin >> ans;
973             cin.get();
974
975             while ( ans!= '1' && ans != '2')
976
977                         {
```

```
978
979                                         cout << "Opcao invalida" << endl
                                                ;
980                         cout << "Deseja realizar os calculos utilizando
                                outro metodo?" << endl;
981                         cout << "1 - Sim" << endl;
982                                 cout << "2 - Nao " << endl;
983                                 cout << "
                                    ------------------------------------------
                                    " << endl;
984
985                         cin >> ans;
986                         cin.get();
987
988                     }
989
990                                 break;
991
992                         }
993
994                         case '2':
995                     run = false;
996                     cout << "Obrigado por utilizar o software!" << endl;
997                     break;
998                 default:
999                     break;
1000
1001            }
1002
1003        }
1004
1005        }
```

Apresenta-se na listagem 6.29 a implementação da função `main`.

Listing 6.29: Arquivo de implementação da função `main()`

```
1#include "CSimuladorCurvasIPR.h"
2
3
4int main ()
5{
6
7    CSimuladorCurvasIPR simulador;
8
9    simulador.Executar();
10
11    return 0;
12}
```

Capítulo 7

Teste

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

7.1 Teste 1: Entrada de dados

O software inicia com a entrada de dados referentes ao problema como: tipo de fluido, raio de poço, viscosidade, fator volume formação do fluido e etc. Os dados são inseridos via teclado pelo usuário 7.1. É importante notar que os tipos de dados inseridos está condicionado ao método de IPR que será utilizado nos cálculos. Abaixo, encontra-se o primeiro exemplo o qual utiliza o modelo da IPR Generalizada.

```
-------------- SIMULADOR DE CURVAS IPR PARA POCOS VERTICAIS ---------------
--------------------------------------------------------------------------------
------------ Atila Junior, Giovanna Massardi e Marcelo Bernardo ------------
-------------------------------- UENF/LENEP --------------------------------


-------------- SIMULADOR DE CURVAS IPR PARA POCOS VERTICAIS ---------------
--------------------------------------------------------------------------------
------------ Atila Junior, Giovanna Massardi e Marcelo Bernardo ------------
-------------------------------- UENF/LENEP --------------------------------

Qual o metodo sera utilizado?
1 - IPR Linear
2 - IPR Generalizada
3 - IPR de Fetkovich (Ideal para reservatorios de gas)
4 - IPR de Vogel
--------------------------------------------------------------------------------
2

Entre com o raio externo do reservatorio (FT)
2980
Entre com a raio do poco (FT)
0.328
Entre com a pressao inicial (PSIA):
5651
Entre com a pressao de bolha (PSIA):
5651
Qual tipo de fluido presente no reservatorio?
1 - Oleo
2 - Gas
--------------------------------------------------------------------------------
1
Entre com fator volume de formacao do oleo:
1.1
Entre com a viscosidade do fluido (CP):
1.7
Entre com a espessura do reservatorio (FT):
53
```

Figura 7.1: Software - Entrada de dados IPR Generalizada

Listing 7.1: Exemplo teste 1 - IPR Generalizada

```
-------------- SIMULADOR DE CURVAS IPR PARA POCOS VERTICAIS
   ---------------
--------------------------------------------------------------------------------


------------ Atila Junior, Giovanna Massardi e Marcelo Bernardo
   ------------
-------------------------------- UENF/LENEP
   --------------------------------



-------------- SIMULADOR DE CURVAS IPR PARA POCOS VERTICAIS
   ---------------
--------------------------------------------------------------------------------
```

```
------------ Atila Junior, Giovanna Massardi e Marcelo Bernardo
   ------------
--------------------------------- UENF/LENEP
   ---------------------------------

Qual o metodo sera utilizado?
1 - IPR Linear
2 - IPR Generalizada
3 - IPR de Fetkovich (Ideal para reservatorios de gas)
4 - IPR de Vogel
--------------------------------------------------------------------------------

2

Entre com o raio externo do reservatorio (FT)
2980
Entre com a raio do poco (FT)
0.328
Entre com a pressao inicial (PSIA):
5651
Entre com a pressao de bolha (PSIA):
5651
Qual tipo de fluido presente no reservatorio?
1 - Oleo
2 - Gas
--------------------------------------------------------------------------------

1
Entre com fator volume de formacao do oleo:
1.1
Entre com a viscosidade do fluido (CP):
1.7
Entre com a espessura do reservatorio (FT):
53
Entre com a permeabilidade da rocha (MD):
8.2
Entre com fator de pelicula do reservatorio:
0


Calculando IPR...


Deseja realizar o calculo de pressoes em quantas partes?
Digite um numero inteiro.
100
Pwf: 5651, Vazao: 0
Pwf: 5594.49, Vazao: 11.1199
```

```
Pwf: 5537.98, Vazao: 22.2398
Pwf: 5481.47, Vazao: 33.3597
Pwf: 5424.96, Vazao: 44.4796
Pwf: 5368.45, Vazao: 55.5996
Pwf: 5311.94, Vazao: 66.7195
Pwf: 5255.43, Vazao: 77.8394
Pwf: 5198.92, Vazao: 88.9593
Pwf: 5142.41, Vazao: 100.079
Pwf: 5085.9, Vazao: 111.199
Pwf: 5029.39, Vazao: 122.319
Pwf: 4972.88, Vazao: 133.439
Pwf: 4916.37, Vazao: 144.559
Pwf: 4859.86, Vazao: 155.679
Pwf: 4803.35, Vazao: 166.799
Pwf: 4746.84, Vazao: 177.919
Pwf: 4690.33, Vazao: 189.038
Pwf: 4633.82, Vazao: 200.158
Pwf: 4577.31, Vazao: 211.278
Pwf: 4520.8, Vazao: 222.398
Pwf: 4464.29, Vazao: 233.518
Pwf: 4407.78, Vazao: 244.638
Pwf: 4351.27, Vazao: 255.758
Pwf: 4294.76, Vazao: 266.878
Pwf: 4238.25, Vazao: 277.998
Pwf: 4181.74, Vazao: 289.118
Pwf: 4125.23, Vazao: 300.238
Pwf: 4068.72, Vazao: 311.358
Pwf: 4012.21, Vazao: 322.477
Pwf: 3955.7, Vazao: 333.597
Pwf: 3899.19, Vazao: 344.717
Pwf: 3842.68, Vazao: 355.837
Pwf: 3786.17, Vazao: 366.957
Pwf: 3729.66, Vazao: 378.077
Pwf: 3673.15, Vazao: 389.197
Pwf: 3616.64, Vazao: 400.317
Pwf: 3560.13, Vazao: 411.437
Pwf: 3503.62, Vazao: 422.557
Pwf: 3447.11, Vazao: 433.677
Pwf: 3390.6, Vazao: 444.796
Pwf: 3334.09, Vazao: 455.916
Pwf: 3277.58, Vazao: 467.036
Pwf: 3221.07, Vazao: 478.156
Pwf: 3164.56, Vazao: 489.276
Pwf: 3108.05, Vazao: 500.396
Pwf: 3051.54, Vazao: 511.516
Pwf: 2995.03, Vazao: 522.636
Pwf: 2938.52, Vazao: 533.756
Pwf: 2882.01, Vazao: 544.876
```

```
Pwf: 2825.5, Vazao: 555.996
Pwf: 2768.99, Vazao: 567.115
Pwf: 2712.48, Vazao: 578.235
Pwf: 2655.97, Vazao: 589.355
Pwf: 2599.46, Vazao: 600.475
Pwf: 2542.95, Vazao: 611.595
Pwf: 2486.44, Vazao: 622.715
Pwf: 2429.93, Vazao: 633.835
Pwf: 2373.42, Vazao: 644.955
Pwf: 2316.91, Vazao: 656.075
Pwf: 2260.4, Vazao: 667.195
Pwf: 2203.89, Vazao: 678.315
Pwf: 2147.38, Vazao: 689.435
Pwf: 2090.87, Vazao: 700.554
Pwf: 2034.36, Vazao: 711.674
Pwf: 1977.85, Vazao: 722.794
Pwf: 1921.34, Vazao: 733.914
Pwf: 1864.83, Vazao: 745.034
Pwf: 1808.32, Vazao: 756.154
Pwf: 1751.81, Vazao: 767.274
Pwf: 1695.3, Vazao: 778.394
Pwf: 1638.79, Vazao: 789.514
Pwf: 1582.28, Vazao: 800.634
Pwf: 1525.77, Vazao: 811.754
Pwf: 1469.26, Vazao: 822.873
Pwf: 1412.75, Vazao: 833.993
Pwf: 1356.24, Vazao: 845.113
Pwf: 1299.73, Vazao: 856.233
Pwf: 1243.22, Vazao: 867.353
Pwf: 1186.71, Vazao: 878.473
Pwf: 1130.2, Vazao: 889.593
Pwf: 1073.69, Vazao: 900.713
Pwf: 1017.18, Vazao: 911.833
Pwf: 960.67, Vazao: 922.953
Pwf: 904.16, Vazao: 934.073
Pwf: 847.65, Vazao: 945.192
Pwf: 791.14, Vazao: 956.312
Pwf: 734.63, Vazao: 967.432
Pwf: 678.12, Vazao: 978.552
Pwf: 621.61, Vazao: 989.672
Pwf: 565.1, Vazao: 1000.79
Pwf: 508.59, Vazao: 1011.91
Pwf: 452.08, Vazao: 1023.03
Pwf: 395.57, Vazao: 1034.15
Pwf: 339.06, Vazao: 1045.27
Pwf: 282.55, Vazao: 1056.39
Pwf: 226.04, Vazao: 1067.51
Pwf: 169.53, Vazao: 1078.63
```

```
Pwf: 113.02, Vazao: 1089.75
Pwf: 56.51, Vazao: 1100.87
Pwf: 0, Vazao: 1111.99
Deseja plotar o grafico?
 1 - Sim
 2 - Nao
--------------------------------------------------------------------------------

1

Deseja salvar o grafico?
 1 - Sim
 2 - Nao
--------------------------------------------------------------------------------

1
Deseja realizar os calculos utilizando outro metodo?
1 - Sim
2 - Nao
--------------------------------------------------------------------------------

2
Obrigado por utilizar o software!
```

O segundo teste foi realizado utilizando o modelo IPR Linear e possui as mesmas premissas da entrada de dados do modelo anterior. Logo, é possível observar abaixo:

Listing 7.2: Exemplo teste 2 - IPR Linear

```

--------------- SIMULADOR DE CURVAS IPR PARA POCOS VERTICAIS
   ---------------
--------------------------------------------------------------------------------

------------ Atila Junior, Giovanna Massardi e Marcelo Bernardo
   ------------
--------------------------------- UENF/LENEP
   ---------------------------------


--------------- SIMULADOR DE CURVAS IPR PARA POCOS VERTICAIS
   ---------------
--------------------------------------------------------------------------------

------------ Atila Junior, Giovanna Massardi e Marcelo Bernardo
   ------------
--------------------------------- UENF/LENEP
   ---------------------------------

```

```
Qual o metodo sera utilizado?
1 - IPR Linear
2 - IPR Generalizada
3 - IPR de Fetkovich (Ideal para reservatorios de gas)
4 - IPR de Vogel
--------------------------------------------------------------------------------

1

Entre com o comprimento do reservatorio (FT)
2980
Entre com a largura do reservatorio (FT)
600
Entre com o raio do poco (FT):
0.328
Entre com a pressao inicial do reservatorio (PSIA):
5651
Entre com a pressao de bolha do reservatorio (PSIA):
50
Qual tipo de fluido presente no reservatorio?
1 - Oleo
2 - Gas
--------------------------------------------------------------------------------

1
Entre com fator volume de formacao do oleo:
1.1
Entre com a viscosidade do fluido (CP):
1.7
Entre com a espessura do reservatorio (FT):
53
Entre com a permeabilidade da rocha reservatorio (MD):
8.2
Entre com fator de pelicula do reservatorio:
0
IP = 0.23505



Calculando IPR...


Deseja realizar o calculo de pressoes em quantas partes?
Digite um numero inteiro.
25
Pwf: 5651, Vazao: 0
Pwf: 5424.96, Vazao: 53.1307
Pwf: 5198.92, Vazao: 106.261
```

```
Pwf: 4972.88, Vazao: 159.392
Pwf: 4746.84, Vazao: 212.523
Pwf: 4520.8, Vazao: 265.653
Pwf: 4294.76, Vazao: 318.784
Pwf: 4068.72, Vazao: 371.915
Pwf: 3842.68, Vazao: 425.045
Pwf: 3616.64, Vazao: 478.176
Pwf: 3390.6, Vazao: 531.307
Pwf: 3164.56, Vazao: 584.437
Pwf: 2938.52, Vazao: 637.568
Pwf: 2712.48, Vazao: 690.699
Pwf: 2486.44, Vazao: 743.829
Pwf: 2260.4, Vazao: 796.96
Pwf: 2034.36, Vazao: 850.091
Pwf: 1808.32, Vazao: 903.221
Pwf: 1582.28, Vazao: 956.352
Pwf: 1356.24, Vazao: 1009.48
Pwf: 1130.2, Vazao: 1062.61
Pwf: 904.16, Vazao: 1115.74
Pwf: 678.12, Vazao: 1168.87
Pwf: 452.08, Vazao: 1222.01
Pwf: 226.04, Vazao: 1275.14
Pwf: 8.52651e-13, Vazao: 1328.27
Pwf: 0, Vazao: 1328.27
Deseja plotar o grafico?
 1 - Sim
 2 - Nao
--------------------------------------------------------------------------------

1

Deseja salvar o grafico?
 1 - Sim
 2 - Nao
--------------------------------------------------------------------------------

1
Deseja realizar os calculos utilizando outro metodo?
1 - Sim
2 - Nao
--------------------------------------------------------------------------------
```

O terceito teste, assim como os anteriores, também possui a mesma interface de entrada de dados, porém foi compilado com modelo IPR Vogel e está a seguir:

Listing 7.3: Exemplo teste 3 - IPR Vogel

```
--------------- SIMULADOR DE CURVAS IPR PARA POCOS VERTICAIS
   ---------------
--------------------------------------------------------------------------------

------------ Atila Junior, Giovanna Massardi e Marcelo Bernardo
   ------------
-------------------------------- UENF/LENEP
   --------------------------------


--------------- SIMULADOR DE CURVAS IPR PARA POCOS VERTICAIS
   ---------------
--------------------------------------------------------------------------------

------------ Atila Junior, Giovanna Massardi e Marcelo Bernardo
   ------------
-------------------------------- UENF/LENEP
   --------------------------------

Qual o metodo sera utilizado?
1 - IPR Linear
2 - IPR Generalizada
3 - IPR de Fetkovich (Ideal para reservatorios de gas)
4 - IPR de Vogel
--------------------------------------------------------------------------------

4

Entre com o raio externo do reservatorio (FT)
2980
Entre com a raio do poco (FT):
0.328
Entre com a pressao inicial do reservatorio (PSIA):
5651
Entre com a pressao de bolha (PSIA):
3000
Qual tipo de fluido presente no reservatorio?
1 - Oleo
2 - Gas
--------------------------------------------------------------------------------

1
Entre com fator volume de formacao do oleo:
1.1
Entre com a viscosidade do fluido (CP):
1.7
Entre com a espessura do reservatorio (FT):
```

```
53
Entre com a permeabilidade da rocha (MD):
8.2
Entre com fator de pelicula do reservatorio:
0


Calculando IPR...


Deseja realizar o calculo de pressoes em quantas partes?
Digite um numero inteiro.
25
Pwf: 5651, Vazao: 0
Pwf: 5424.96, Vazao: 43.6889
Pwf: 5198.92, Vazao: 85.7963
Pwf: 4972.88, Vazao: 126.322
Pwf: 4746.84, Vazao: 165.267
Pwf: 4520.8, Vazao: 202.63
Pwf: 4294.76, Vazao: 238.411
Pwf: 4068.72, Vazao: 272.611
Pwf: 3842.68, Vazao: 305.229
Pwf: 3616.64, Vazao: 336.266
Pwf: 3390.6, Vazao: 365.722
Pwf: 3164.56, Vazao: 393.595
Pwf: 2938.52, Vazao: 419.888
Pwf: 2712.48, Vazao: 444.599
Pwf: 2486.44, Vazao: 467.728
Pwf: 2260.4, Vazao: 489.276
Pwf: 2034.36, Vazao: 509.243
Pwf: 1808.32, Vazao: 527.627
Pwf: 1582.28, Vazao: 544.431
Pwf: 1356.24, Vazao: 559.653
Pwf: 1130.2, Vazao: 573.293
Pwf: 904.16, Vazao: 585.352
Pwf: 678.12, Vazao: 595.83
Pwf: 452.08, Vazao: 604.726
Pwf: 226.04, Vazao: 612.04
Pwf: 8.52651e-13, Vazao: 617.773
Pwf: 0, Vazao: 617.773
Deseja plotar o grafico?
 1 - Sim
 2 - Nao
--------------------------------------------------------------------------------


1


Deseja salvar o grafico?
```

```
 1 - Sim
 2 - Nao
--------------------------------------------------------------------------------


1
Deseja realizar os calculos utilizando outro metodo?
1 - Sim
2 - Nao
--------------------------------------------------------------------------------
```

Por último, há o modelo Fetkovich que também foi testado. Observe a seguir as informações que abrangem este modelo:

Listing 7.4: Exemplo teste 4- IPR Fetkovich

```

--------------- SIMULADOR DE CURVAS IPR PARA POCOS VERTICAIS
   ---------------
--------------------------------------------------------------------------------

------------ Atila Junior, Giovanna Massardi e Marcelo Bernardo
   ------------
--------------------------------- UENF/LENEP
   ---------------------------------


--------------- SIMULADOR DE CURVAS IPR PARA POCOS VERTICAIS
   ---------------
--------------------------------------------------------------------------------

------------ Atila Junior, Giovanna Massardi e Marcelo Bernardo
   ------------
--------------------------------- UENF/LENEP
   ---------------------------------

Qual o metodo sera utilizado?
1 - IPR Linear
2 - IPR Generalizada
3 - IPR de Fetkovich (Ideal para reservatorios de gas)
4 - IPR de Vogel
--------------------------------------------------------------------------------

3

Entre com o raio externo do reservatorio (FT)
2980
Entre com a raio do poco .
0.328
```

```
Entre com a pressao media do reservatorio (PSIA):
5000
Entre com a pressao de bolha (PSIA):
3000
Entre com a pressao de fundo no poco A (PSIA):
4000
Entre com a vazao no poco A (STB/d):
300
Entre com a pressao de fundo no poco B (PSIA):
2000
Entre com a vazao no poco B (STB/d):
900
Qual tipo de fluido presente no reservatorio?
1 - Oleo
2 - Gas
--------------------------------------------------------------------------------

1
Entre com fator volume de formacao do oleo:
1.1
Entre com a viscosidade do fluido (CP):
1.7
Entre com a espessura do reservatorio (FT):
53
Entre com a permeabilidade da rocha reservatorio (MD):
8.2
Entre com fator de pelicula do reservatorio:
0


Calculando IPR...


Deseja realizar o calculo de pressoes em quantas partes?
Digite um numero inteiro.
25
Pwf: 5000, Vazao: 0
Pwf: 4800, Vazao: 88.4583
Pwf: 4600, Vazao: 173.306
Pwf: 4400, Vazao: 254.543
Pwf: 4200, Vazao: 332.17
Pwf: 4000, Vazao: 406.186
Pwf: 3800, Vazao: 476.592
Pwf: 3600, Vazao: 543.387
Pwf: 3400, Vazao: 606.571
Pwf: 3200, Vazao: 666.145
Pwf: 3000, Vazao: 722.109
Pwf: 2800, Vazao: 774.461
```

```
Pwf: 2600, Vazao: 823.204
Pwf: 2400, Vazao: 868.336
Pwf: 2200, Vazao: 909.857
Pwf: 2000, Vazao: 947.767
Pwf: 1800, Vazao: 982.068
Pwf: 1600, Vazao: 1012.76
Pwf: 1400, Vazao: 1039.84
Pwf: 1200, Vazao: 1063.3
Pwf: 1000, Vazao: 1083.16
Pwf: 800, Vazao: 1099.41
Pwf: 600, Vazao: 1112.05
Pwf: 400, Vazao: 1121.07
Pwf: 200, Vazao: 1126.49
Pwf: 0, Vazao: 1128.29
Pwf: 0, Vazao: 1128.29
Deseja plotar o grafico?
 1 - Sim
 2 - Nao
--------------------------------------------------------------------------------

1

Deseja salvar o grafico?
 1 - Sim
 2 - Nao
--------------------------------------------------------------------------------

1
Deseja realizar os calculos utilizando outro metodo?
1 - Sim
2 - Nao
--------------------------------------------------------------------------------
```

7.2 Teste 2: Cálculos

Após a entrada de dados é realizado o cálculo de vazões com base no método selecionado. A saída dos resultados é gerada com o valor de vazão para a pressão de fundo em questão7.2.

```
Deseja realizar o calculo de pressoes em quantas partes?
Digite um numero inteiro.
20
Pwf: 5651, Vazao: 0
Pwf: 5368.45, Vazao: 53.5463
Pwf: 5085.9, Vazao: 107.093
Pwf: 4803.35, Vazao: 160.639
Pwf: 4520.8, Vazao: 214.185
Pwf: 4238.25, Vazao: 267.731
Pwf: 3955.7, Vazao: 321.278
Pwf: 3673.15, Vazao: 374.824
Pwf: 3390.6, Vazao: 428.37
Pwf: 3108.05, Vazao: 481.916
Pwf: 2825.5, Vazao: 535.463
Pwf: 2542.95, Vazao: 589.009
Pwf: 2260.4, Vazao: 642.555
Pwf: 1977.85, Vazao: 696.102
Pwf: 1695.3, Vazao: 749.648
Pwf: 1412.75, Vazao: 803.194
Pwf: 1130.2, Vazao: 856.74
Pwf: 847.65, Vazao: 910.287
Pwf: 565.1, Vazao: 963.833
Pwf: 282.55, Vazao: 1017.38
Pwf: 0, Vazao: 1070.93
Deseja plotar o grafico?
 1 - Sim
 2 - Nao
--------------------------------------------------------------------------------
```

Figura 7.2: Cálculo de vazão.

7.3 Teste 3: Saída de dados

Após a entrada de dados e cálculo dos resultados o usuário pode optar pela plotagem das curvas. Os resultados quando comparados à literatura foram satisfatórios e podem ser analisados a seguir.

Por fim, o usuário pode salvar o gráfico em seu computador selecionando o item 1 no Simulador como a imagem 7.7 demonstra.

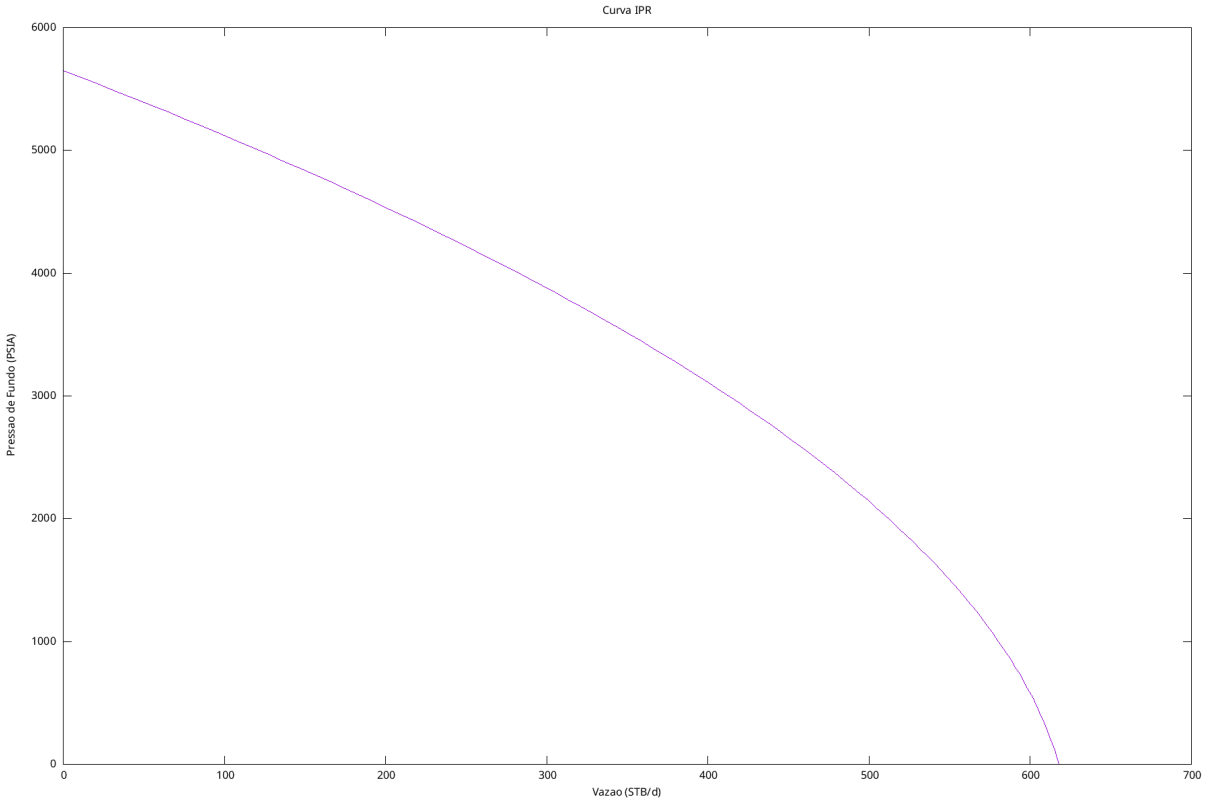

Figura 7.3: Gráfico IPR Generalizada

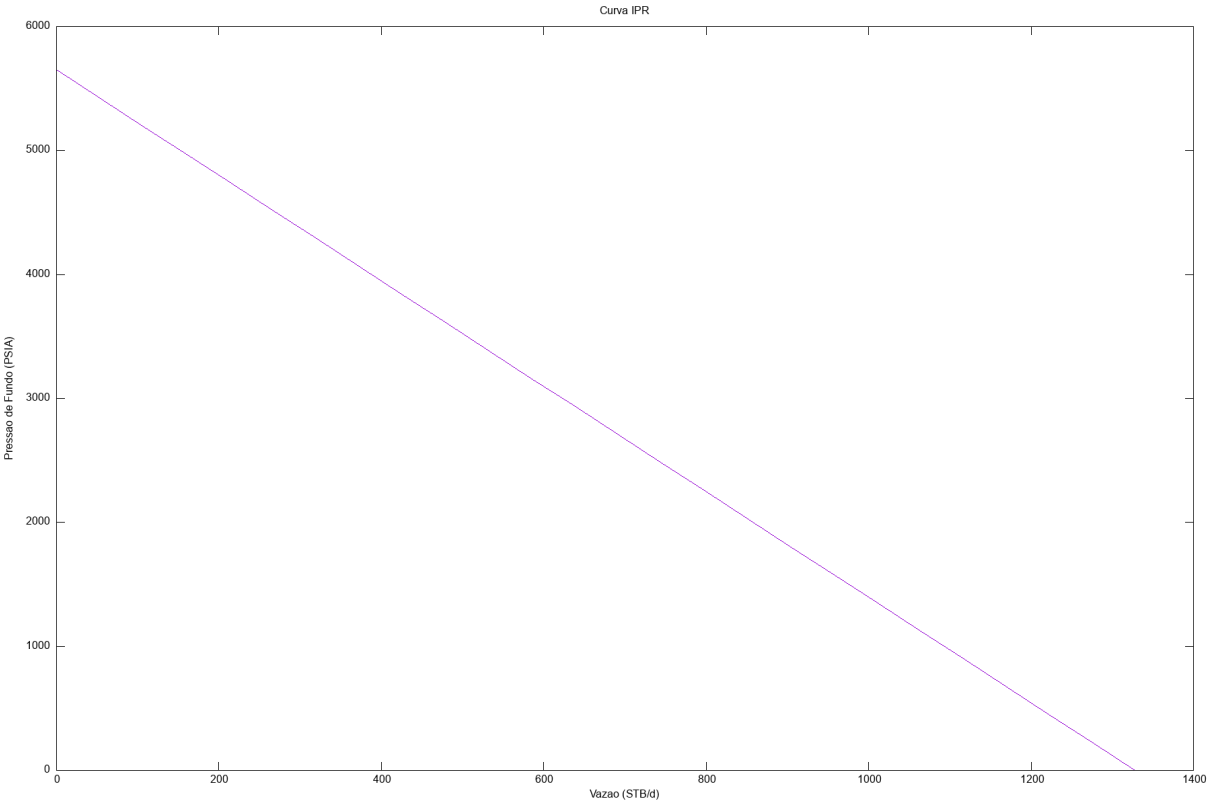

Figura 7.4: Gráfico IPR Linear

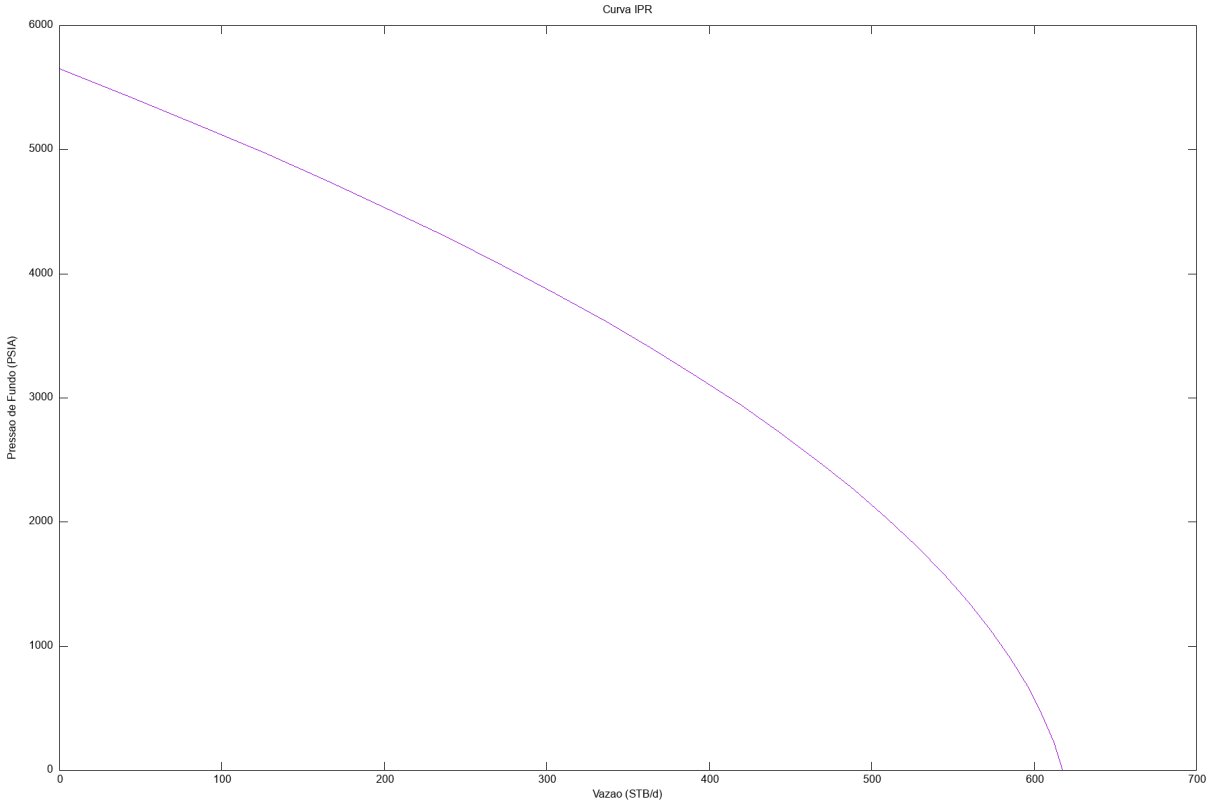

Figura 7.5: Gráfico IPR Vogel

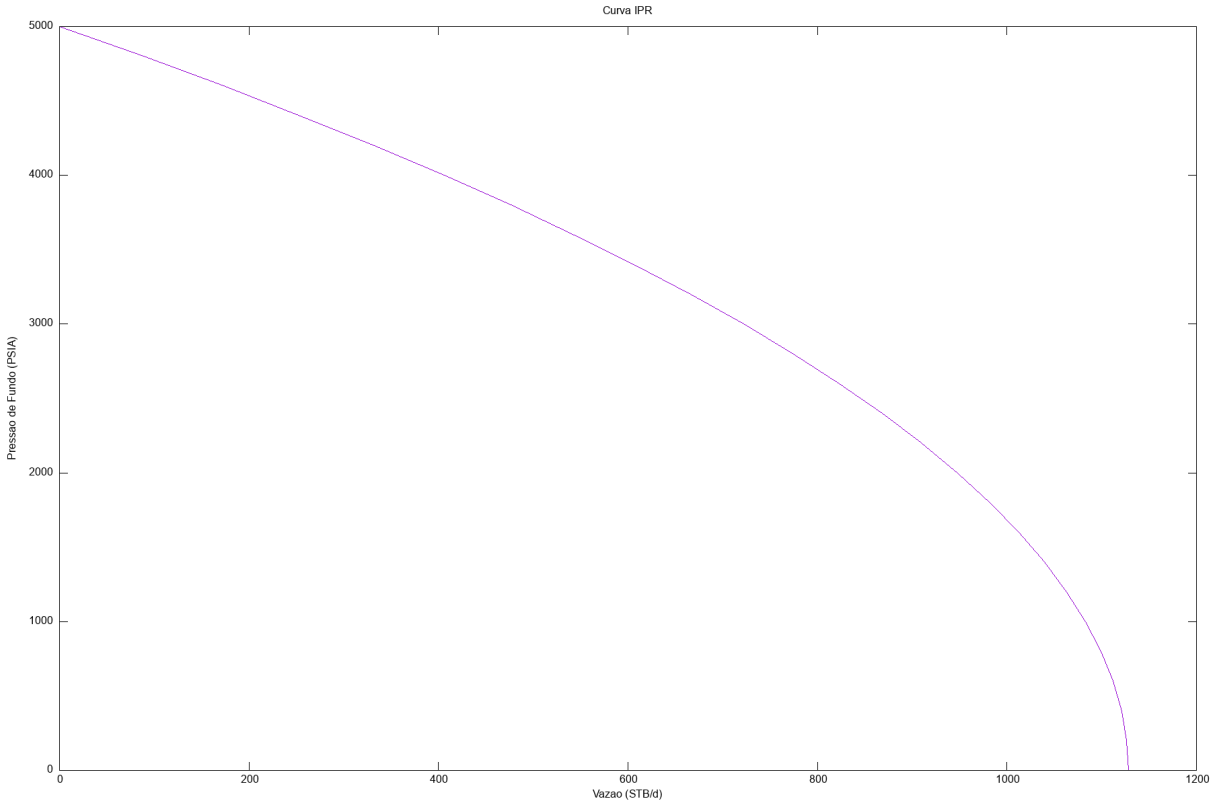

Figura 7.6: Gráfico IPR Fetkovich

```
Pwf: 4746.84, Vazao: 177.919
Pwf: 4520.8, Vazao: 222.398
Pwf: 4294.76, Vazao: 266.878
Pwf: 4068.72, Vazao: 311.358
Pwf: 3842.68, Vazao: 355.837
Pwf: 3616.64, Vazao: 400.317
Pwf: 3390.6, Vazao: 444.796
Pwf: 3164.56, Vazao: 489.276
Pwf: 2938.52, Vazao: 533.756
Pwf: 2712.48, Vazao: 578.235
Pwf: 2486.44, Vazao: 622.715
Pwf: 2260.4, Vazao: 667.195
Pwf: 2034.36, Vazao: 711.674
Pwf: 1808.32, Vazao: 756.154
Pwf: 1582.28, Vazao: 800.634
Pwf: 1356.24, Vazao: 845.113
Pwf: 1130.2, Vazao: 889.593
Pwf: 904.16, Vazao: 934.073
Pwf: 678.12, Vazao: 978.552
Pwf: 452.08, Vazao: 1023.03
Pwf: 226.04, Vazao: 1067.51
Pwf: 8.52651e-13, Vazao: 1111.99
Pwf: 0, Vazao: 1111.99
Deseja plotar o grafico?
 1 - Sim
 2 - Nao
--------------------------------------------------------------------------------
1

Deseja salvar o grafico?
 1 - Sim
 2 - Nao
--------------------------------------------------------------------------------
1
Deseja realizar os calculos utilizando outro metodo?
1 - Sim
2 - Nao
--------------------------------------------------------------------------------
```

Figura 7.7: Salvar gráfico

Capítulo 8

Documentação para o Desenvolvedor

Todo projeto de engenharia precisa ser bem documentado. Neste sentido, apresenta-se neste capítulo documentações extras para o desenvolvedor. Ou seja, instruções para pessoas que venham a dar continuidade a este projeto de engenharia.

Nota: O manual do usuário é apresentado em um documento separado. Veja diretório "doc/ManualDoUsuario".

8.1 Dependências para compilar o software

Para compilar o software é necessário atender as seguintes dependências:

- Instalar o compilador g++ da GNU disponível em http://gcc.gnu.org. Para instalar no GNU/Linux use o comando `yum install gcc`.
- Biblioteca CGnuplot; os arquivos para acesso a biblioteca CGnuplot devem estar no diretório com os códigos do software;
- O software `gnuplot`, disponível no endereço http://www.gnuplot.info/, deve estar instalado. É possível que haja necessidade de setar o caminho para execução do `gnuplot`.
- .
- .

8.2 Como gerar a documentação usando doxygen

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

- Veja informações sobre uso do formato JAVADOC em:
  - http://www.stack.nl/∼dimitri/doxygen/manual/docblocks.html
- Veja informações sobre o software `doxygen` em
  - http://www.stack.nl/∼dimitri/doxygen/

Passos para gerar a documentação usando o `doxygen`.

- Documente o código usando o formato JAVADOC. Um bom exemplo de código documentado é apresentado nos arquivos da biblioteca CGnuplot, abra os arquivos `CGnuplot.h` e `CGnuplot.cpp` no editor de texto e veja como o código foi documentado.
- Abra um terminal.
- Vá para o diretório onde esta o código.

  ```
  cd /caminho/para/seu/codigo
  ```

- Peça para o `doxygen` gerar o arquivo de definições (arquivo que diz para o doxygem como deve ser a documentação).

  ```
  dogygen -g
  ```

- Peça para o `doxygen` gerar a documentação.

  ```
  doxygen
  ```

- Verifique a documentação gerada abrindo o arquivo html/index.html.

  ```
  firefox html/index.html
  ```

  ou

  ```
  chrome html/index.html
  ```

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

Nota:

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

Capítulo 9

Sugestões para Trabalhos Futuros

Com a finalidade de melhorar o projeto, sugere-se que os tópicos abaixo sejam incorporados de modo que o código e formulação teórica sejam ampliados a outros casos.

- Especificação:
  - Apresenta-se a seguir os requisitos funcionais.

RF-01	O usuário deve ter liberdade para escolher quais parâmetros de entrada utilizar, utilizando teclado ou um arquivo .txt.

  - .

- Formulação Teórica:
  - Adicionar a IPR que abrange reservatórios estratificados, ou seja, reservatórios com mais de uma camada e possibilitar o cálculo utilizando os modelos empíricos.
  - Adicionar a IPR Futura, método utilizado para prever produtividade.
  - Implementar outros regimes de escoamento, como o permanente e transiente.
  - Considerar os cálculos para poços horizontais ou direcionais.

- Formulação do Código:
  - Permitir que o usuário entre com os dados na forma de arquivo de disco.
  - Permitir que o usuário salve os resultados de pressão e vazão calculados pelo método escolhido.

Referências Bibliográficas

[1] Tarek Ahmed. *Reservoir Engineering Book*. ELSEVIER, Oxford, 2006.

[2] James P. Brill. *Multiphase Flow in Wells*. SPE, Pennsylvania, 1999.

[3] Andre Duarte Bueno. *Apostila de Programacao Orientada a Objeto*. Florianópolis, 1997.

[4] Andre Duarte Bueno. *Programacao Orientada a Objeto com C++ - Aprenda a Programar em Ambiente Multiplataforma com Software Livre*. Novatec, Sao Paulo, 2003.

[5] Boyun Guo and Ali Ghalambor. *Well Productivity Handbook*. GPC, 2008.

[6] Boyun Guo, William C. Lyons, and Ali Ghalambor. *Petroleum Production Engineering*. ELSEVIER, Oxford, 2007.

[7] Adalberto Jose Rosa and Renato de Souza Carvalho. *Engenharia de Reservatorio de Petroleo*. Interciência, 2006.

[8] Moshood Sanni. *Petroleum Engineering*. Wiley, 2018.