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

PROJETO ENGENHARIA
DESENVOLVIMENTO DO SOFTWARE
CÁLCULO DO ÍNDICE DE PRODUTIVIDADE DE POÇOS HORIZONTAIS E VERTICAIS

TRABALHO DA DISCIPLINA PROGRAMAÇÃO PRÁTICA

Versão 1:

CAROLINA BASTOS E DOUGLAS RIBEIRO

Prof. André Duarte Bueno

MACAÉ - RJ

Agosto - 2021

Sumário

Capítulo 1

Introdução

No presente projeto de engenharia desenvolve-se o software Cálculo do Índice de Produtividade de Poços Horizontais e Verticais, um software aplicado a engenharia de petróleo e que utiliza o paradigma da orientação a objetos.

O software é da área de engenharia de poços e permite simular a influência de parâmetros do reservatório e do fluido na produtividade dos poços, podendo os resultados serem comparados para definir qual melhor design de poço para cada cenário.

1.1 Escopo do problema

O projeto de perfuração de um poço horizontal é diferente de um projeto de perfuração de um poço vertical porque a produtividade do poço depende do comprimento do mesmo, além de fatores determinantes em ambos os projetos como permeabilidade, anisotropias, espessura permeável, viscosidade do óleo e vários aspectos relativos à perfuração do trecho horizontal. Para cada diferente cenário, haverá uma diferente solução de poço para desenvolver o campo. E isto engloba além do fator financeiro, a capacidade produtiva desses poços, se será ou não vantajoso a exploração do mesmo.

Por isso, é importante evidenciar em que situações, em termos de produtividade, qual design de poço seria mais recomendado através de uma comparação de resultados, alinhado com um embasamento teórico, sobre diversos parâmetros de reservatório que podem intervir na produtividade do poço horizontal e qual o ganho de produtividade em relação ao poço vertical. Assim os engenheiros de reservatório e de poço podem trabalhar de forma conjunta para escolher a técnica mais apropriada.

1.2 Objetivos

Os objetivos deste projeto de engenharia são:

- Objetivo geral:

  - Desenvolver um projeto de engenharia de software para resolver os diferentes modelos matemáticos de previsão de produtividade de poços horizontais e verticais e a influência dos parâmetros de reservatórios nos mesmos para analisar em que situações, em termos de produtividade, qual design de poço seria mais recomendado através das simulações.

- Objetivos específicos:

  - Modelar física e matematicamente o problema.
  - Modelagem estática do software (diagramas de caso de uso, de pacotes, de classes).
  - Modelagem dinâmica do software (desenvolver algoritmos e diagramas exemplificando o fluxo de processamento).
  - Calcular o Índice de Produtividade (IP) dos poços, a partir dos modelos analíticos de Borisov, Giger, Joshi, RenardDupuy e Shedid.
  - Simular a influência de parâmentros do reservatório, como a altura, a anisotropia, a centralização vertical e a viscosidade do fluido nos resultados do IP.
  - Implementar manual simplificado de uso do software.

Capítulo 2

Especificação

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

2.1 O que é a especificação?

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

2.2 Nome do sistema/produto

Nome	Cáculo do Índice de Produtividade de Poços Horizontais e Verticais
Componentes principais	Sistema para calcular a influência das propriedades do reservatório e do fluido na produtividade dos poços horizontais e verticais a fim de definir qual melhor design para o poço.
Missão	- Simular diferentes cenários do sistema fluido/reservatório e sua influência na produtividade dos poços. - Calcular IP dos poços. - Gerar gráficos que permita comparar IP de poços com diferentes designs (horizontal/vertical).

2.3 Especificação

Apresenta-se a seguir a especicação do software.

O projeto a ser desenvolvido consiste de um programa que deverá realizar cálculos de IP de poços horizontais e verticais, além de mostrar os resultados graficamente. Os cálculos serão feitos a partir de modelos matemáticos existentes na literatura e na dinâmica de execução do software, o usuário poderá escolher qual modelo deseja utilizar, qual o tipo de formação a ser atravessada e as características do fluido produzido. Além disso, o usuário deverá entrar com os dados do reservatório (permeabilidades horizontal e vertical, espessura, comprimento e raio do poço) e viscosidade e fator de formação do fluido - via arquivo .txt, ao final da simulação o usuário poderá salvar os resultados em arquivo .txt, ver os resultados em tela, gerar gráficos e salvá-los como imagem.

2.3.1 Requisitos funcionais

Apresenta-se a seguir os requisitos funcionais.

RF-01	O programa deverá solicitar os dados de entrada (parâmetros do poço, do reservatório e do fluido) ao usuário através de um arquivo .txt

RF-02	O usuário deverá ter liberdade para escolher o tipo de formação que o poço irá atravessar (isotrópica ou anisotrópica)

RF-03	O usuário deverá ter liberdade para escolher o modelo matemático para o cálculo do IP.

RF-04	O programa deverá mostrar os resultados dos cálculos de IP na tela.

RF-05	O usuário poderá plotar seus resultados em um gráfico, podendo este ser salvo como imagem e como arquivo .txt.

2.3.2 Requisitos não funcionais

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

RNF-02	O programa deverá ser multi-plataforma, podendo ser executado em *Windows*, *GNU/Linux* ou *Mac*.

2.4 Casos de uso

A tabela 2.1 apresenta um caso de uso do sistema.

Tabela 2.1: Exemplo de caso de uso

Nome do caso de uso:	Cálculo do IP de um poço horizontal
Resumo/descrição:	Determinar a capacidade produtiva de um poço do tipo horizontal, a partir de um modelo matemático a ser escolhido
Etapas:	1. Entrar com os dados do poço, do reservatório e do fluido (permeabilidade, espessura, viscosidade, etc.). 2. Definir o tipo de formação a ser atravessada pelo poço: isotrópica ou anisotrópica. 3. Definir o modelo matemático mais apropriado para aquele cenário de reservatório/fluido a partir da analise dos resultados. 4. Salvar resultados em disco.
Cenários alternativos:	Inserir valores negativos para parâmetros do reservatório ou incompatíveis com a ordem de grandeza do problema real.

2.4.1 Diagrama de caso de uso geral

O diagrama de caso de uso geral da Figura 2.1 mostra o usuário interagindo com o software para obter o IP de um poço. Neste caso de uso geral, o usuário insere os dados de entrada (via arquivo .txt) para então analisar o resultado obtido.

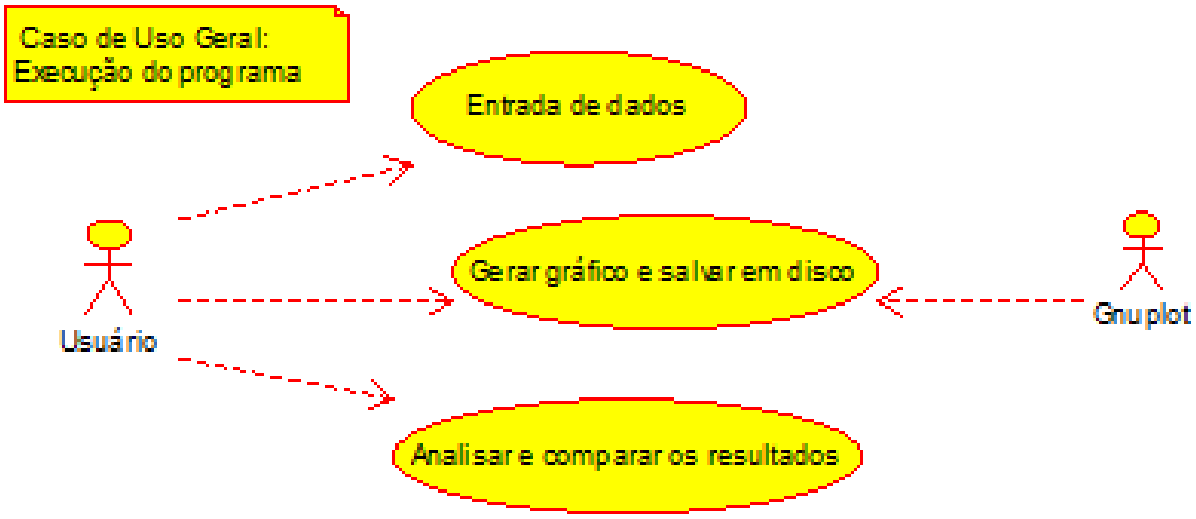

Figura 2.1: Diagrama de caso de uso geral – Cálculo do IP

2.4.2 Diagrama de caso de uso específico

O caso de uso Comparar IP de um reservatório anisotrópico e anisotrópico descrito na Figura 2.1 e na Tabela 2.1 é detalhado na Figura 2.2. O usuário pode variar os parâmetros do poço e do reservatório e então plotar esses diferentes cenários em um gráfico para fazer comparações e definir qual melhor se adequa ao projeto.

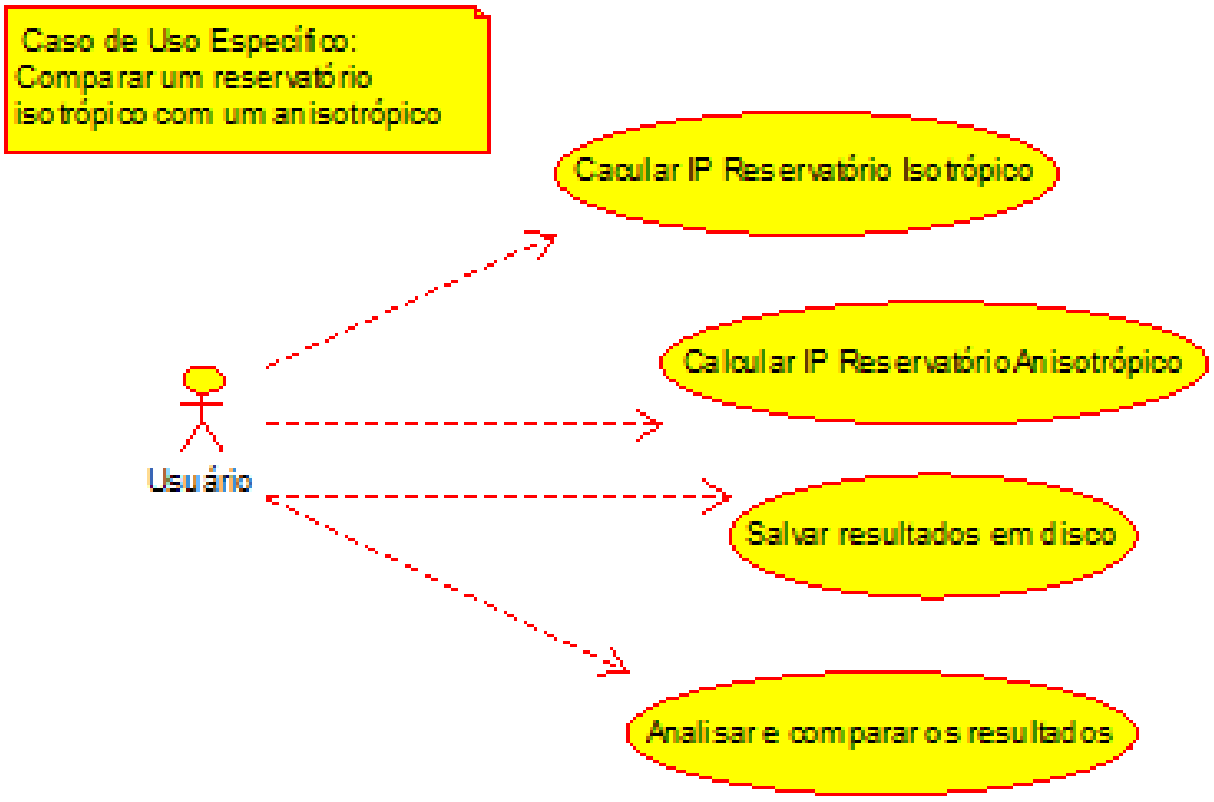

Figura 2.2: Diagrama de caso de uso específico – Comparando o IP de reservatório isotrópico com um anisotrópico

Capítulo 3

Elaboração

Depois da definição dos objetivos, da especificação do software e da montagem dos primeiros diagramas de caso de uso, a equipe de desenvolvimento do projeto de engenharia passa por um processo de elaboração que envolve o estudo de conceitos relacionados ao sistema a ser desenvolvido, a análise de domínio e a identificação de pacotes.

Na elaboração fazemos uma análise dos requisitos, ajustando os requisitos iniciais de forma a desenvolver um sistema útil, que atenda às necessidades do usuário e, na medida do possível, permita seu reuso e futura extensão.

Eliminam-se os requisitos "impossíveis" e ajusta-se a idéia do sistema de forma que este seja flexível, considerando-se aspectos como custos e prazos.

3.1 Análise de domínio

A tecnologia de poços horizontais constitui o padrão de perfuração e implementação de poços de desenvolvimento na indústria do petróleo, ao lado da perfuração direcional, principalmente em ambientes offshore, devido ao alto custo de um poço. Antes do avanço da tecnologia para a perfuração de poços horizontais a desvantagem em relação a poços verticais era que apenas uma área poderia ser drenada por um mesmo poço.

A partir do surgimento de novas técnicas de perfuração passaram-se a perfurar poços horizontais multilaterais, assim um poço poderia drenar mais de um reservatório. A partir de um poço vertical perfuram-se vários trechos horizontais em diferentes camadas. O principal motivo para esse tipo é o grande aumento que se dá de produtividade, podendo apontar outras vantagens em relação ao poço vertical como menor gradiente de pressão, menor número de poços, maior exposição ao reservatório, poços de longo alcance, redução da produção de areia, entre outros. Porém, como qualquer outro método há desvantagens, por exemplo, se o poço horizontal for atingido pela água proveniente do contato óleo/água ascendente, dependendo da completação que foi utilizada no poço, ele deverá ser fechado ou transformado em um poço injetor, não podendo haver intervenção ou recompletação.

O projeto de perfuração de um poço horizontal é diferente de um poço vertical, porque

a sua produtividade depende de seu comprimento, além de fatores determinantes em ambos os projetos como viscosidade do óleo e permeabilidade da formação e vários aspectos relativos à perfuração do trecho horizontal.

Este projeto tem como objetivo evidenciar em que situações, em termos de produtividade, qual design de poço seria mais recomendado por meio de um estudo com embasamento teórico sobre diversos parâmetros de reservatório que podem intervir na produtividade do poço horizontal e qual o ganho de produtividade em relação a um vertical.

Depois de estudar as especificações do sistema e estudos de biblioteca e de disciplinas do curso foi possível identificar nosso domínio de trabalho:

- O software irá calcular vários índices de produtividade para um mesmo reservatório dado por meio dos métodos a depender do caso ser isotrópico ou anisotrópico;
- O software usará conceitos de engenharia de reservatório e da engenharia de poço para que se realize as simulações, aqui iremos ter explanações básicas de quando se usar cada método, porém é necessário que se tenha o conhecimento básico dessas disciplinas para a realização da simulação.
- O software plotará os resultados dos índices de produtividade para poços com diferentes design.

3.2 Formulação teórica

3.2.1 Produtividade de Poços

Inicialmente, serão apresentadas algumas definições de parâmetros para uma boa compreensão de termos e conceitos utilizados no decorrer do projeto.

3.2.2 Índice de Produtividade

O índice de produtividade, de forma simplificada, é dado pela equação 3.1 :

$$IP = \frac{Q}{Pe - Pw} \tag{3.1}$$

Onde:
$Q = vazão\,[m^3/d]$
$Pe = pressão\,estática\,do\,reservatório\ [kgf/cm^2]$
$P_w = pressão\,de\,fluxo\,do\,poço\ [kgf/cm^2]$

3.2.3 Efeito Skin

Segundo [JOSHI, 1988] o efeito de película ou de skin é um modelo matemático introduzido na indústria de petróleo por Van Everdingen & Hurst com o objetivo de simular

um fenômeno real, o dano à formação.

A partir da definição do fator de skin pode-se definir o raio efetivo do poço por meio da equação 3.2:

$$r'_w = re^{-s} \tag{3.2}$$

Onde:
$r'_w = raio\, efetivo\, do\, poço\ [cm]$
$r_w = raio\, do\, poço\ [cm]$
$s = fator\, de\, skin$

3.2.4 Regime Permanente

As soluções analíticas em estado estacionário ou permanente são a forma mais simples de soluções para poços horizontais. No regime de fluxo permanente, por hipótese admitimos que a pressão em qualquer ponto do reservatório é independente do tempo.

Na realidade são pouquíssimos casos de reservatórios que operam sob as condições do regime de fluxo permanente. Apesar disso, essas soluções são usadas em grande frequência segundo [JOSHI, 1988] pelos seguintes fatos:

1. São de fácil dedução analítica;
2. Podem ser usados para obter soluções para o fluxo transiente, usa-se o artifício de aumentar o raio de drenagem com o tempo;
3. Podem ser usadas para se obter soluções para o fluxo pseudopermanente por meio do emprego do fator de Dietz, que permite o cálculo da pseudopressão para diversas geometrias do reservatório;
4. Podem ser verificadas experimentalmente por meio de modelos de laboratório [ROSA, 2006].

3.2.5 Produtividade de Poços Horizontais

Os métodos abaixo são para formações isotrópicas, ou seja, com a permeabilidade vertical igual à horizontal.

- Borisov:

$$IP = \frac{\frac{2\pi k_h h}{\mu}}{ln(\frac{4r_{eh}}{L}) + [(\frac{h}{L})ln(\frac{h}{2\pi r_w})]} \tag{3.3}$$

- Giger:

$$IP = \frac{\frac{2\pi k_h h}{\mu}}{(\frac{L}{h})ln(\frac{1+\sqrt{1-(\frac{L}{2r_{eh}})^2}}{\frac{L}{2r_{eh}}}) + ln(\frac{h}{2\pi r_w})} \tag{3.4}$$

- Joshi:

$$IP = \frac{2\pi k_h h}{ln(\frac{a+\sqrt{a^2-(\frac{L}{2})^2}}{\frac{L}{2}}) + (\frac{h}{L})ln(\frac{h}{2r_w})} \tag{3.5}$$

Onde:

$$a = (\frac{L}{2})\sqrt{0.5 + \sqrt{0.25 + (\frac{2r_{eh}}{L})^4}} \tag{3.6}$$

$IP = Índice\ de\ Produtividade$

$k_h = permeabilidade\ horizontal$

$h = altura\ do\ reservatório$

$\mu = viscosidade\ do\ óleo$

$r_{eh} = raio\ exeterno\ do\ reservatório$

$L = comprimento\ horizontal\ do\ reservatório$

$r_w = raio\ do\ poço$

Na literatura também é apresentado uma solução que é independente do raio de drenagem r_{eh} do poço. segundo [SHEDID, 2001]:

$$IP = \frac{\frac{2\pi hk}{\mu B_o}}{[ln(\frac{h/2r_w)}{L/h} + (0.25 + \frac{C}{L})(\frac{1}{r_w} - \frac{2}{h})]} \tag{3.7}$$

Onde:

$B_o = fator\ volume\ de\ formação\ do\ óleo$

E a contante C é mostrada na figura 3.1 abaixo:

Horizontal well Length (L), ft	Value of (C) or equation to be used to calculate the constant (C), ft
>0-1000	270
>1000-3000	C = 470 - 0.20 * L

Figura 3.1: Constante C

3.2.6 Cálculo do IP com anisotropia

$$IP = \frac{Q}{\Delta P} = \frac{\frac{2\pi k_h h}{\mu}}{ln(\frac{a+\sqrt{a^2-(\frac{L}{2})^2}}{\frac{L}{2}}) + (\frac{\beta h}{L})ln(\frac{\beta h}{2r_w})} \tag{3.8}$$

Onde:

$$\beta = \sqrt{\frac{k_h}{k_v}} \tag{3.9}$$

ΔP *é a queda de pressão no reservatório*

- Modelo de Renard e Dupuy:

$$IP = \frac{2\pi k_h h}{\mu} \left(\frac{1}{cosh^{-1}(X) + \left(\frac{\beta h}{L}\right)\left(ln\left[\frac{h}{2\pi r'_w}\right]\right)} \right) \tag{3.10}$$

Onde:

$$r'_w = \frac{1+\beta}{2\beta} r_w \tag{3.11}$$

$$X = \frac{2a}{L} \tag{3.12}$$

Isso é usado para uma área elipsoidal, a é dado pela equação 3.6 e β pela equação 3.9.

3.3 Diagrama de pacotes – assuntos

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

- Pacote Fludo: Engloba as caracteríticas do fluido, como viscosidade e fator volume formação.
- Pacote Reservatório: Contém os dados relativos ao reservatório, incluindo o tipo de formação, se é isotrópica ou anisotrópica.
- Pacote Poço: Contém os dados relativos ao poço e os métodos que serão utilizados para o cálculo do índice de produtividade (subsistema MetodosIP).
- Pacote Gráficos: Usando o software Gnuplot, será possível gerar gráficos relacionando cada índice de produtividade com cada método.
- Pacote Simulador: Relaciona os pacotes acima, sendo responsável pela criação e destruição dos objetos.

Veja Figura 3.2.

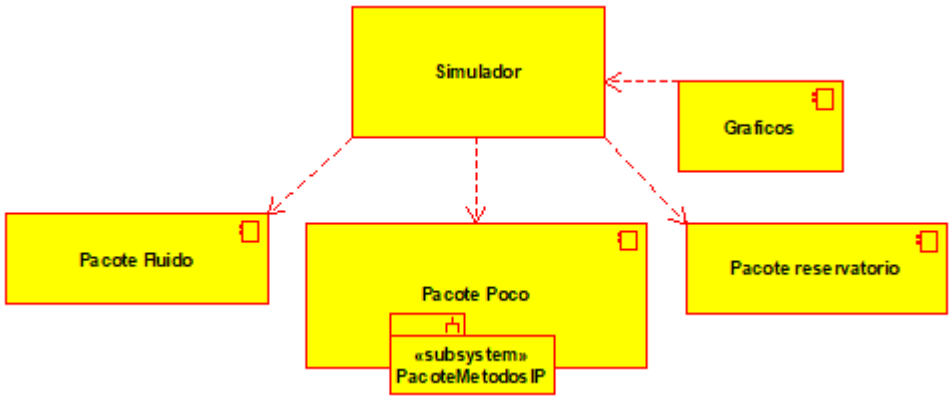

Figura 3.2: Diagrama de Pacotes

Capítulo 4

AOO – Análise Orientada a Objeto

Nesta etapa de desenvolvimento do projeto de engenharia, apresentamos a Análise Orientada a Objeto - AOO. Esta análise mostra as relações entre as classes, os atributos, os métodos e suas associações e consiste em modelos estruturais dos objetos e seus relacionamentos, e modelos dinâmicos, apresentando as modicações do objeto com o tempo. O resultado da análise é um conjunto de diagramas que identificam os objetos e seus relacionamentos.

4.1 Diagramas de classes

O diagrama de classes do software desenvolvido é apresentado na Figura 4.1. Como podemos perceber, ele é constituído de doze classes.

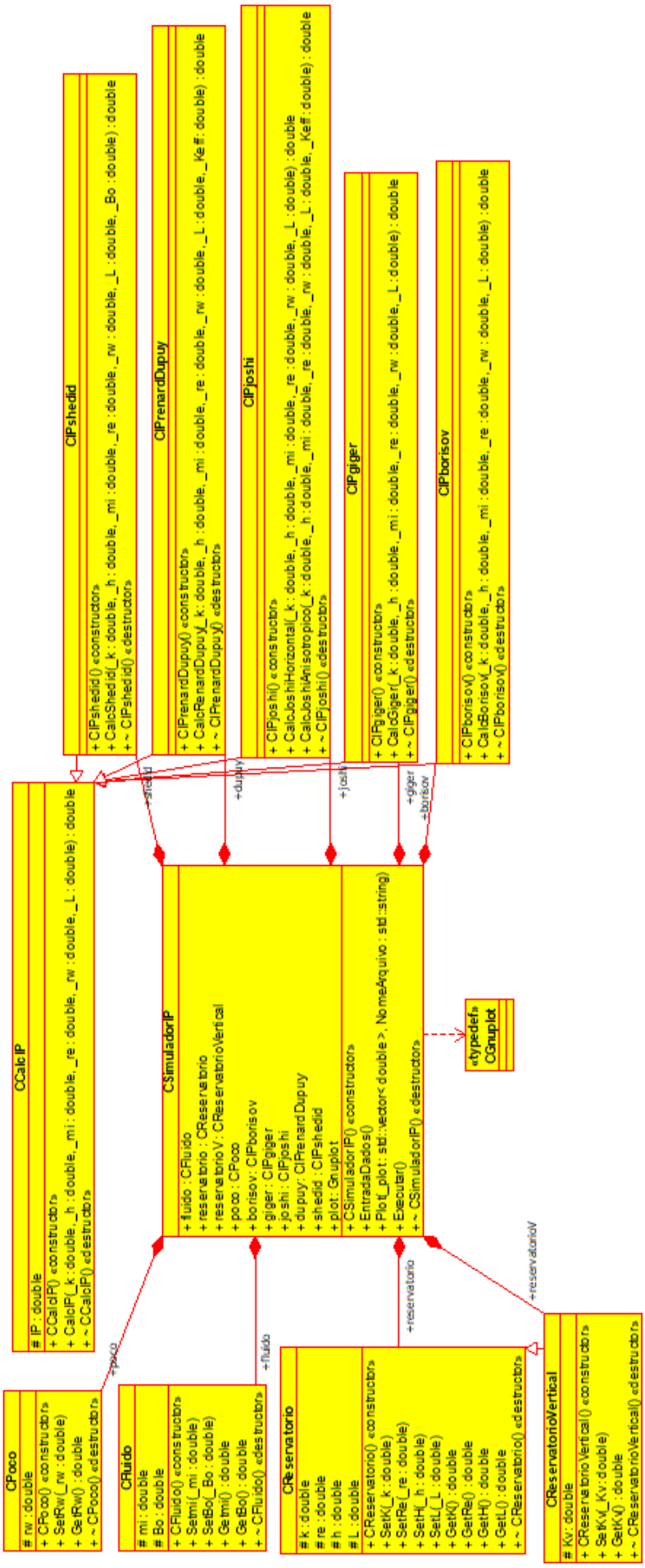

Figura 4.1: Diagrama de classes

4.1.1 Dicionário de classes

- Classe CPoco: Solicita ao usuário os dados do poço e armazena.
- Classe CFluido: Solicita ao usuário os dados do fluido e armazena.
- Classe CReservatorio: Solicita ao usuário os dados do reservatório e armazena.
- Classe CReservatorioVertical: Solicita ao usuário os dados do reservatório vertical e armazena.
- Classe CIPshedid: Calcula o IP do poço pelo modelo de Shedid.
- Classe CIPrenardDupuy: Calcula o IP do poço pelo modelo de Renard&Dupuy.
- Classe CIPjoshi: Calcula o IP do poço pelo modelo de Joshi.
- Classe CIPgiger: Calcula o IP do poço pelo modelo de Giger.
- Classe CIPborisov: Calcula o IP do poço pelo modelo de Borisov.
- Classe CCalcIP: Calcula o IP, utilizando os modelos disponíveis.
- Classe CSimuladorIP: Faz as simulações do índice de produtividades dos poços (classe *main* do programa)
- Classe GnuplotExcrption: Gera uma visualização gráfica dos resultados usando software externo Gnuplot.

4.2 Diagrama de seqüência – eventos e mensagens

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

4.2.1 Diagrama de sequência geral

O diagrama de seqüência geral do software é mostrado na figura 4.2.

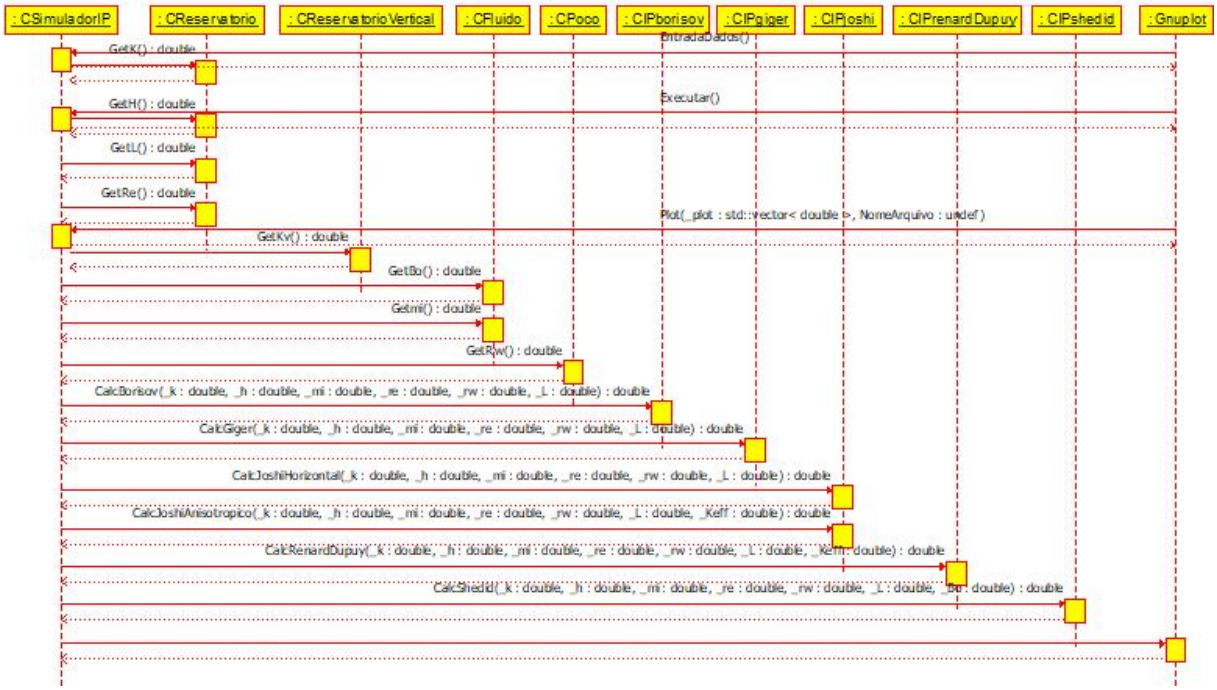

Figura 4.2: Diagrama de seqüência geral

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. Na figura 4.3 o diagrama de comunicação mostra a sequência do software para um caso em que o modelo de Joshi é utilizados para o cálculo do IP. Observe que a Classe CSimuladorIP acessa os parâmetros do fluido, do poço e do reservatório a partir das classes CFluido, CPoco e CReservatorio, respectivamente, que passa os atributos informados pelo usuário para as classes CCalcIP que por sua fez chama a classe CIPjoshi. A classe CSimuladorIP então, após realizar os cálculo, envia esses resultados para a classe CGnuplot que gera a visualização e salva esses resultados de forma gráfica.

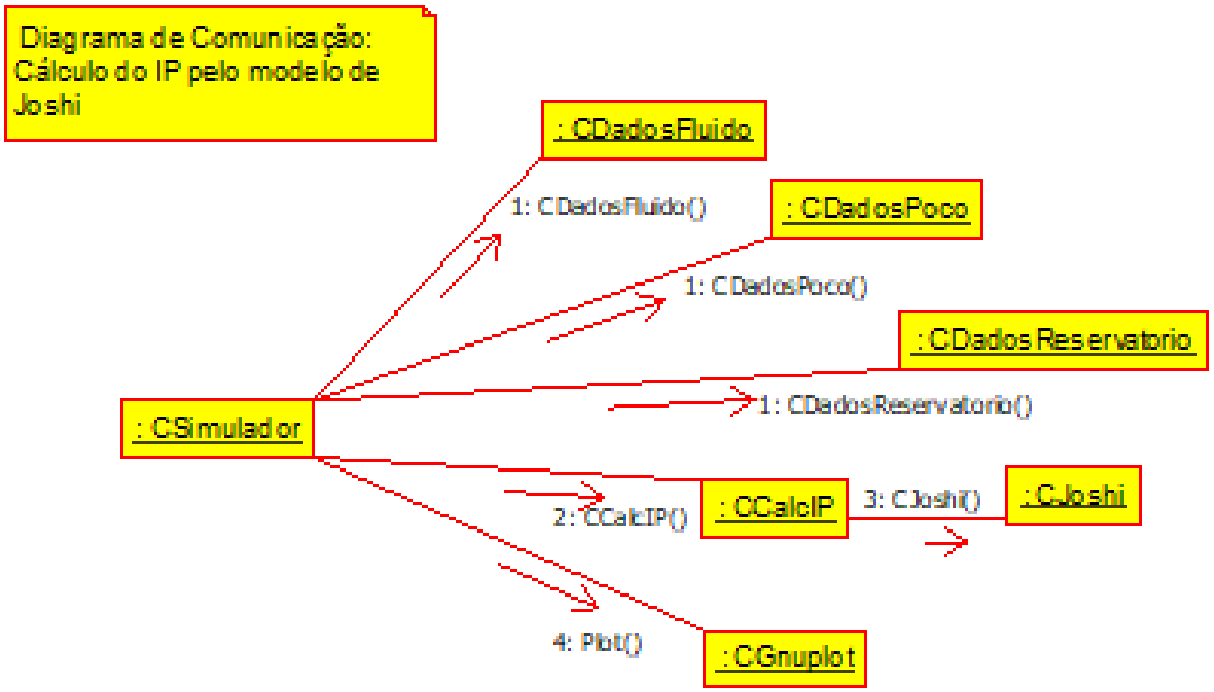

Figura 4.3: Diagrama de comunicação

4.4 Diagrama de máquina de estado

Um diagrama de máquina de estado representa os diversos estados que o objeto assume e os eventos que ocorrem ao longo de sua vida ou mesmo ao longo de um processo (histórico do objeto). É usado para modelar aspectos dinâmicos do objeto, como mostra a figura 4.4. Observe que, durante a execução do programa, o objeto passa por várias etapas.

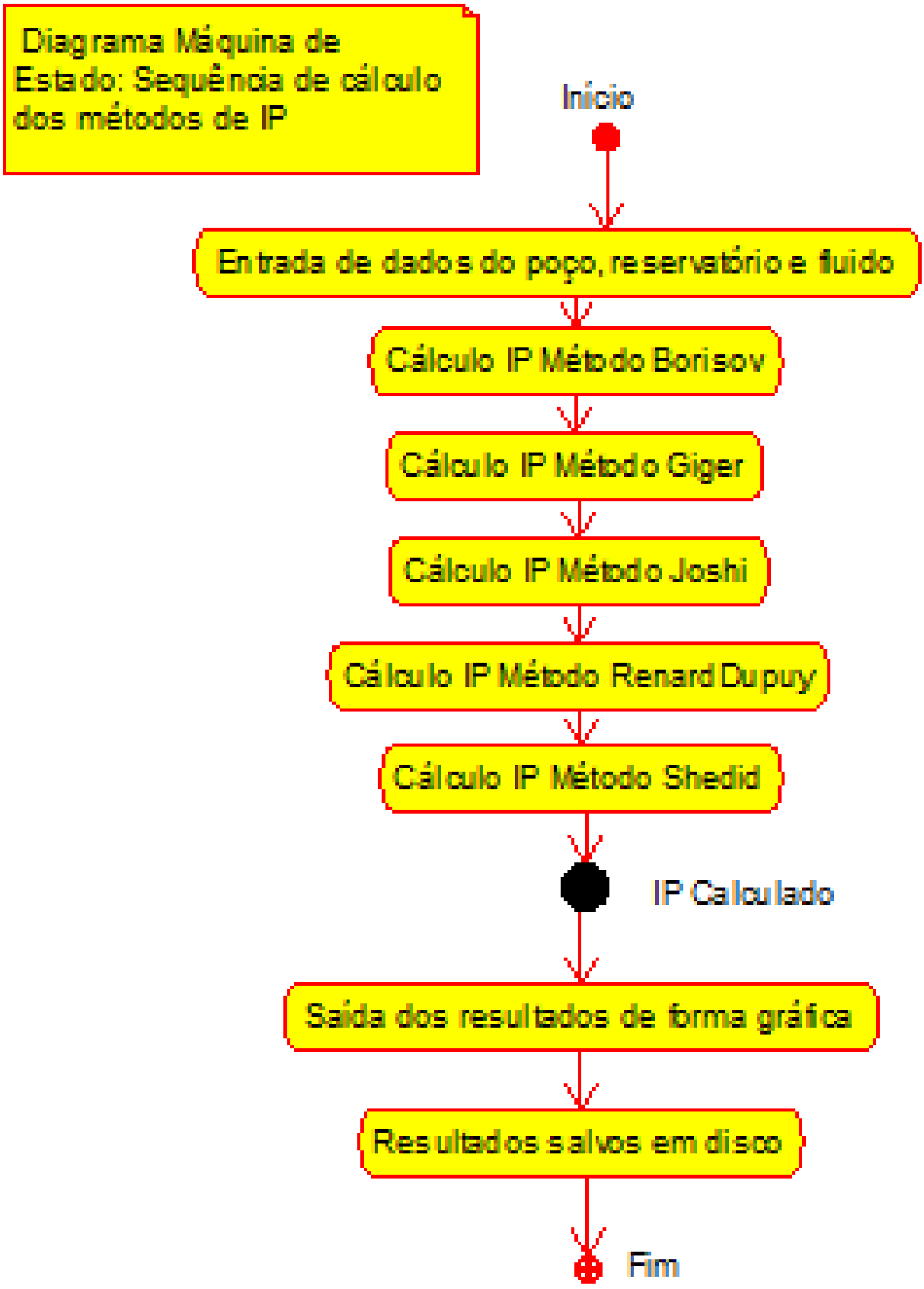

Figura 4.4: Diagrama de máquina de estado

4.5 Diagrama de atividades

O diagrama de atividades da figura 4.5 corresponde a uma atividade específica do diagrama de máguina de estado. Observe que foi escolhido um cenário fictício qualquer em que o poço recebe os dados do poço, raio interno, a viscosidade e fator volume formação do fluido e parâmentros do reservatório como raio externo, espessura, comprimento e permeabilidade vertical, que são informações necessáris para calcular IP. O modelo escolhido como exemplo é o Borisov.

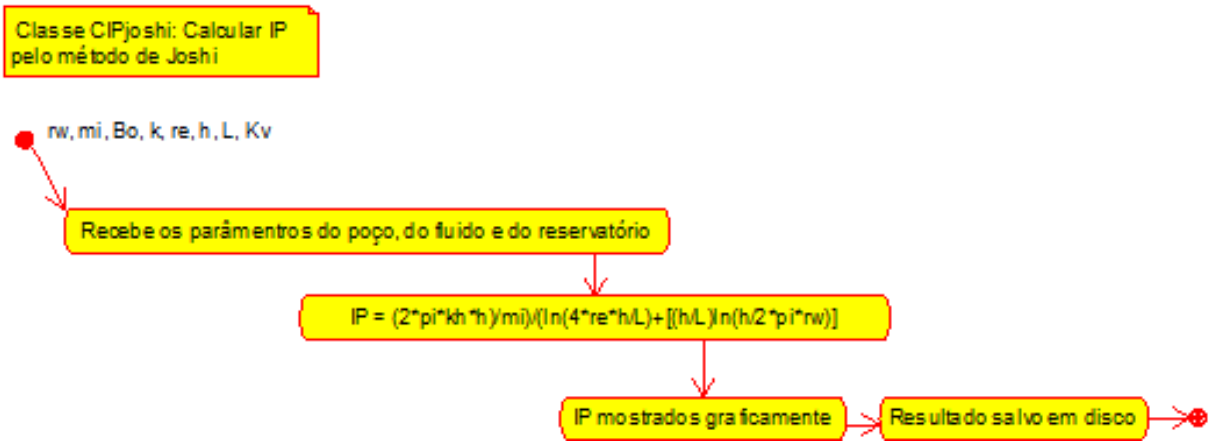

Figura 4.5: Diagrama de atividades

Capítulo 5

Projeto

Neste capítulo do projeto de engenharia veremos questões associadas ao projeto do sistema, incluindo protocolos, recursos, plataformas suportadas, inplicações 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 [Rumbaugh et al., 1994, Blaha and Rumbaugh, 2006], 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

    - No software,o usuário poderá entrar com os dados via arquivos no formato ASCII com extensão .txt.
    - A interface utilizada será em modo texto.
    - O software irá gerar saída de arquivos no formato ASCII com extensão .txt e .png.

2. Recursos

- Neste projeto,o programa irá necessitar de utilizar os componentes internos do computador, como, por exemplo, HD, processador, mouse e teclado.
- Os gráficos serão gerados no programa externo Gnuplot.

3. Controle

- Neste projeto,o controle será sequencial.
- Não irá haver necessidade de otimização, pois o software e seus componentes trabalham com dados pequenos.
- Identificação e definição de *loops* de controle e das escalas de tempo.
  - Não se aplica.

4. Plataformas

- O software irá funcionar nos sistema operacionais Windows e GNU/Linux, sendo desenvolvido no Windows e testado no Windows e GNU/Linux.
- A linguagem de programação padrão utilizada é C++.
- As bibliotecas que serão utilizadas neste projeto são: iomanip, iostream, fstream, string, vector, entre outras.
- O projeto será totalmente desenvolvido na IDE Dev C++ na versão 5.11.

5. Padrões de projeto

- Normalmente,os padrões de projeto são identificados e passam a fazer parte de uma biblioteca de padrões da empresa. Entretanto, isso só ocorre após a realização de diversos projetos. Portanto, não se aplica neste caso.

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 softwareção). Passa pelo maior detalhamento do funcionamento do software, acrescentando atributos e métodos que envolvem a solução de problemas específicos não identificados durante a análise.

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

Efeitos do projeto no modelo estrutural

- Adicionar nos diagramas de pacotes as bibliotecas e subsistemas selecionados no projeto do sistema (exemplo: a biblioteca gráfica selecionada).
  - Neste projeto foi adicionada a biblioteca gráfica CGnuplot.
- Novas classes e associações oriundas das bibliotecas selecionadas e da linguagem escolhida devem ser acrescentadas ao modelo.
  - Não se aplica a este projeto.
- Estabelecer as dependências e restrições associadas à plataforma escolhida.
  - O *Software* necessita das plataformas GNU/Linux ou Windows para ser executado.
  - No sistema operacional Windows, é necessário a instalação do *Software* Gnuplot para o funcionamento do programa.

Efeitos do projeto no modelo dinâmico

- Revisar os diagramas de seqüência e de comunicação considerando a plataforma escolhida.
  - Após necessidade de criação de uma classe CCalcIP que acessa todas as demias classes que calcula o índice de produtividade dos poços de acordo com modelos específicos, o diagrama de sequência precisou ser revisado e a sequência alterada para inclusão dessa etapa em que a CCalcIP acessasse os cáculos das classes herdeiras.
  - O mesmo se aplica ao diagrama de comunicação, com a inclusão dessa classe genérica CCalcIP a comunicaão entre as classes do programa precisou ser alterada.
- Verificar a necessidade de se revisar, ampliar e adicionar novos diagramas de máquinas de estado e de atividades.
  - Houve necessidade de revisar, por motivos de mudanças decorridas na forma de contrução do código que alteroou a sequência de alguns eventos. A classe que antes calculava diretamente o índice de produtividade a partir do modelo específico escolhido pelo usuário agora faz parte das muitas classes herdeiras que são acessadas pela classe "mãe" CCalcIP que cacula todos o índice de produtividade a partir de todos os métodos, permitindo uma comparação entre eles e definição de qual o melhor design de poço em termos de produtividade.

Efeitos do projeto nos atributos

- Atributos novos podem ser adicionados a uma classe, como, por exemplo, atributos específicos de uma determinada linguagem de softwareção (acesso a disco, ponteiros, constantes e informações correlacionadas).

  – O atributo de acesso ao disco precisou ser incluído durante a elaboração do código para que o usuário pudesse inserir os dados do poço, do reservatório e do fluido (dados de entrada) em um arquivo .txt utilizando-o como input no programa. Esse atributo também está sendo utilizado ao final da execução, pela classe CGnuplot que além de gerar os gráficos comparando os difenretes métodos também o salva como imagem em disco e como arquivo .txr.

Efeitos do projeto nos métodos

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

  – Não houve necessidade de alteração dos métodos.

- Algoritmos complexos podem ser subdivididos. Verifique quais métodos podem ser otimizados. Pense em utilizar algoritmos prontos como os da STL (algoritmos genéricos).

  – Não se aplica.

- Responda a pergunta: os métodos da classes estão dando resposta às responsabilidades da classe?

  – Os métodos que foram construídos estão gerando resultados coerentes com o que é abordado na literatura.

- Revise os diagramas de classes, de seqüência e de máquina de estado.

  – Foram realizadas várias revisões dos diagramas a medida que o código foi sendo construído e consequentemente havendo necessidade de tais alterações. O número de classes também mudou ao londo do processo, chegando à versão final que é a apresentadas neste documento.

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).
  - Está sendo realizada uma reformulação das classes, separando-as em classes menores e conceitos independentes. Por exemplo, tínhamos elaborado uma classe para cada modelo de cálculo de IP e agora rearranjamos para que fique uma classe reunindo os modelos para poços do tipo horizontal e uma outra classe com os do tipo vertical.
  - Além disso, foi criada uma classe genérica de cálculo de IP para que ela seja acessada pela CSimuladorIP e a partir daí acessar as classes herdeiras que calculam o IP a partir de modelos específicos. Anteriormente a CSimuladorIP acessava diretamente todas essas classes.
- Abstração do comportamento comum (duas classes podem ter uma superclasse em comum).
  - Não se aplica a este projeto.
- Utilização de delegação para compartilhar a implementação (quando você cria uma herança irreal para reaproveitar código). Usar com cuidado.
  - Não se aplica a este projeto.
- Revise as heranças no diagrama de classes.
  - Foi criado relaciomaneto de herança entre a classe genérica CCalcIP e as demais CIPjoshi, CIPgiger.. que calculam o IP a partir de um modelo específico.

Efeitos do projeto nas associações

- Deve-se definir na fase de projeto como as associações serão implementadas, se obedecerão um determinado padrão ou não.
  - As associações foram criadas e modificadas ao longo do desenvolvimento do código, respeitando a hierarquia das classes.
- Se existe uma relação de "muitos", pode-se implementar a associação com a utilização de um dicionário, que é um mapa das associações entre objetos. Assim, o objeto A acessa o dicionário fornecendo uma chave (um nome para o objeto que deseja acessar) e o dicionário retorna um valor (um ponteiro) para o objeto correto.
  - Não se aplica a este projeto.

- Evite percorrer várias associações para acessar dados de classes distantes. Pense em adicionar associações diretas.

  – Não se aplica a este projeto. Só houve criação de associações diretas.

Efeitos do projeto nas otimizações

- Faça uma análise de aspectos relativos à otimização do sistema. Lembrando que a otimização deve ser desenvolvida por analistas/desenvolvedores experientes.

  – Inicialmente pensamos em solicitar ao usuário os dados de entrada via terminal, ao longo do desenvolvimento implementamos a funcionalidade de colocar os dados em um arquivo externo que será lido pelo programa ao ser executado.

- Identifique pontos a serem otimizados em que podem ser utilizados processos concorrentes.

  – Não identificamos.

- Se o acesso a determinados objetos (atributos/métodos) requer um caminho longo (exemplo: A->B->C->D.atributo), pense em incluir associações extras (exemplo: A-D.atributo).

  – Não se aplica a este projeto, todos os atriutos estão sendo acessados de forma direta.

- Revise as associações nos diagramas de classes.

  – Foram revisadas a medida que desenvolvemos o código.

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

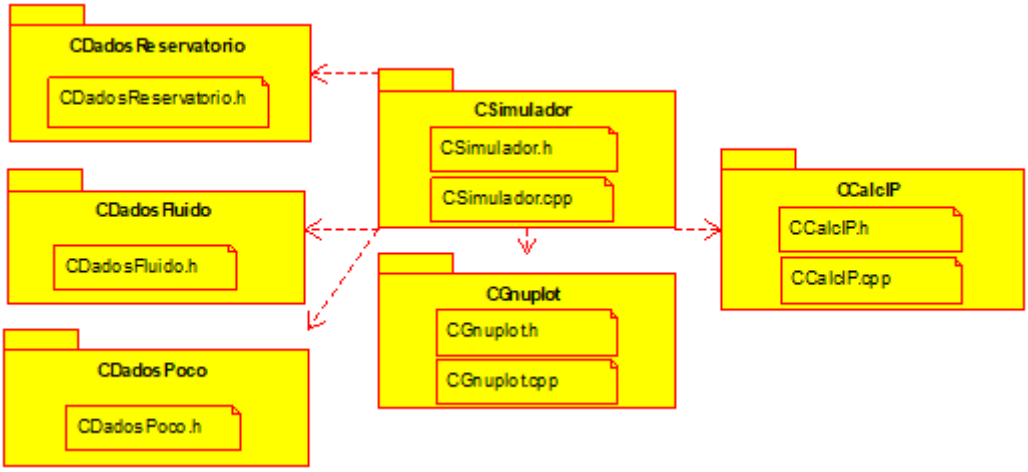

Figura 5.1: Diagrama de componentes

5.4 Diagrama de implantação

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

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

Veja na Figura 5.2 um exemplo de diagrama de implantação utilizado.

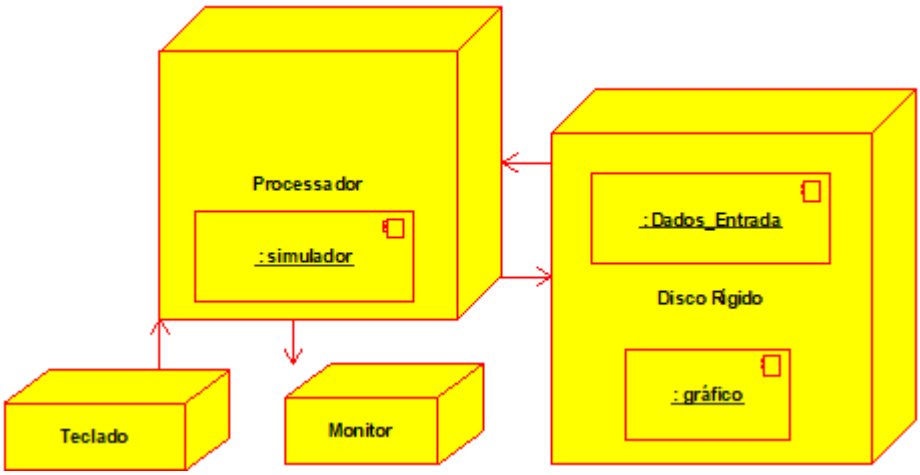

Figura 5.2: Diagrama de implantação

Capítulo 6

Implementação

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

6.1 Código fonte

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

Apresenta-se na listagem **??** o 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
5 class CFluido {
6
7         protected:
8
9         double mi, Bo;
10
11        public:
12
13                CFluido(){};
14
15                void Setmi(double _mi);
16                void SetBo(double _Bo);
17                double Getmi();
18                double GetBo();
19
```

```
20                  ~CFluido(){};
21
22
23};
24
25#endif
```

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

Listing 6.2: Arquivo de implementação da classe CFluido.

```
1#include "CFluido.h"
2
3                   void CFluido::Setmi(double _mi)
4                   {
5                           mi = _mi;
6                   }
7                   double CFluido::Getmi()
8                   {
9                           return mi;
10                  }
11
12                  void CFluido::SetBo(double _Bo)
13                  {
14                          Bo = _Bo;
15                  }
16
17                  double CFluido::GetBo()
18                  {
19                          return Bo;
20                  }
```

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

Listing 6.3: Arquivo de cabeçalho da classe CPoco.

```
1#ifndef CPOCO_H_
2#define CPOCO_H_
3
4class CPoco
5{
6
7       protected:
8
9                   double rw;
10
```

```
11          public:
12
13                  CPoco(){};
14
15                  void SetRw(double _rw);
16                  double GetRw();
17
18                  ~CPoco(){};
19
20};
21
22#endif
```

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

Listing 6.4: Arquivo de implementação da classe CPoco.

```
1#include "CPoco.h"
2
3void CPoco::SetRw(double _rw)
4{
5          rw = _rw;
6}
7
8double CPoco::GetRw()
9{
10          return rw;
11}
```

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

Listing 6.5: Arquivo de cabeçalho da classe CReservatorio.

```
1#ifndef CRESERVATORIO_H_
2#define CRESERVATORIO_H_
3
4class CReservatorio {
5
6          protected:
7
8                  double k, re, h, L;
9
10         public:
11
12         CReservatorio(){};
13
```

```
14          void SetK (double _k);
15          void SetRe (double _re);
16          void SetH (double _h);
17          void SetL (double _L);
18          double GetK();
19          double GetRe();
20          double GetH();
21          double GetL();
22
23          ~CReservatorio(){};
24
25 };
26
27
28
29 #endif
```

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

Listing 6.6: Arquivo de implementação da classe CReservatorio.

```
1 #include "CReservatorio.h"
2
3           void CReservatorio::SetK(double _k)
4           {
5                   k = _k;
6           }
7           void CReservatorio::SetRe (double _re)
8           {
9                   re = _re;
10          }
11          void CReservatorio::SetH (double _h)
12          {
13                  h = _h;
14          }
15          void CReservatorio::SetL (double _L)
16          {
17                  L=_L;
18          }
19          double CReservatorio::GetK()
20          {
21                  return k;
22          }
23          double CReservatorio::GetRe()
```

```
24          {
25                  return re;
26          }
27          double CReservatorio::GetH()
28          {
29                  return h;
30          }
31          double CReservatorio::GetL()
32          {
33                  return L;
34          }
```

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

Listing 6.7: Arquivo de cabeçalho da classe CReservatorioVertical.

```
1 #ifndef CRESERVATORIOVERTICAL_H_
2 #define CRESERVATORIOVERTICAL_H_
3
4 #include "CReservatorio.h"
5
6 class CReservatorioVertical : CReservatorio
7 {
8
9         protected:
10
11                  double Kv;
12
13          public:
14
15                  CReservatorioVertical(){};
16
17                  void SetKv(double _Kv);
18                  double GetKv();
19
20
21                  ~CReservatorioVertical(){};
22
23
24 };
25
26 #endif
```

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

Listing 6.8: Arquivo de implementação da classe CReservatorioVertical.

```
#define _USE_MATH_DEFINES
#include <math.h>

#include "CReservatorioVertical.h"

                void CReservatorioVertical::SetKv(double _Kv)
                {
                        Kv = _Kv;
                }


                double CReservatorioVertical::GetKv()
                {
                        return Kv;
                }
```

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

Listing 6.9: Arquivo de cabeçalho da classe CCalcIP.

```
#ifndef CCalcIP_H
#define CCalcIP_H

class CCalcIP
{

        protected:

                double IP;

        public:

                CCalcIP(){};

                double CalcIP(double _k, double _h, double _mi,
                   double _re, double _rw, double _L){return IP;};

                ~CCalcIP(){};

};

#endif
```

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

Listing 6.10: Arquivo de implementação da classe CCalcIP.

```
1#include "CCalcIP.h"
```

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

Listing 6.11: Arquivo de cabeçalho da classe CIPborisov.

```
1#ifndef CIPBORISOV_H_
2#define CIPBORISOV_H_
3
4#include "CCalcIP.h"
5
6class CIPborisov : CCalcIP
7{
8
9        public:
10
11                CIPborisov(){};
12
13                double CalcBorisov(double _k, double _h, double _mi
                      , double _re, double _rw, double _L);
14
15                ~CIPborisov(){};
16
17};
18
19#endif
```

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

Listing 6.12: Arquivo de implementação da classe CIPborisov.

```
1#define _USE_MATH_DEFINES
2#include <math.h>
3#include <iostream>
4
5#include "CIPborisov.h"
6
7
8double CIPborisov::CalcBorisov(double _k, double _h, double _mi,
    double _re, double _rw, double _L)
9{
10
11        double A, B, C;
```

```
12
13          A = (2.0*M_PI*_k*_h)/_mi ;
14          B = log((4.0*_re)/_L);
15          C = (_h/_L)*log(_h/(2.0*M_PI*_rw));
16
17          IP = A / (B+C);
18
19          return IP;
20 }
```

Apresenta-se na listagem ?? o arquivo com código da classe CIPgiger.

Listing 6.13: Arquivo de cabeçalho da classe CIPgiger.

```
1 #ifndef CIPGIGER_H_
2 #define CIPGIGER_H_
3
4 #include "CCalcIP.h"
5
6 class CIPgiger : CCalcIP
7 {
8
9
10          public:
11
12                  CIPgiger(){};
13
14                  double CalcGiger(double _k, double _h, double _mi,
                        double _re, double _rw, double _L);
15
16                  ~CIPgiger(){};
17
18
19 };
20
21 #endif
```

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

Listing 6.14: Arquivo de implementação da classe CIPgiger.

```
1 #define _USE_MATH_DEFINES
2 #include <math.h>
3
4 #include "CIPgiger.h"
5
```

```
6
7double CIPgiger::CalcGiger(double _k, double _h, double _mi, double
      _re, double _rw, double _L)
8{
9
10          double A, B, C, D, E;
11
12          A = (2.0*M_PI*_k*_L)/_mi;
13          B = _L/_h;
14          C = (1.0 + (sqrt((1.0 - pow(( _L/(2.0*_re)) , 2.0)))));
15          D = (_L/(2.0*_re));
16          E = log(_h/(2.0*M_PI*_rw));
17
18          IP = A/((B*log(C/D)) + E);
19
20          return IP;
21
22}
```

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

Listing 6.15: Arquivo de cabeçalho da classe CIPjoshi.

```
1#ifndef CIPJOSHI_H_
2#define CIPJOSHI_H_
3
4#include "CCalcIP.h"
5
6class CIPjoshi : CCalcIP
7{
8
9          public:
10
11                    CIPjoshi(){};
12
13                    double CalcJoshiHorizontal(double _k, double _h,
                         double _mi, double _re, double _rw, double _L);
14                    double CalcJoshiAnisotropico(double _k, double _h,
                         double _mi, double _re, double _rw, double _L,
                         double _Keff);
15
16                    ~CIPjoshi(){};
17
18};
```

```
19
20#endif
```

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

Listing 6.16: Arquivo de implementação da classe CIPjoshi.

```
1#define _USE_MATH_DEFINES
2#include <math.h>
3
4
5#include "CIPjoshi.h"
6
7
8double CIPjoshi::CalcJoshiHorizontal(double _k, double _h, double
    _mi, double _re, double _rw, double _L)
9{
10
11        double a, A, B, C;
12
13        a = (_L/2.0)*sqrt(0.5 + (sqrt(0.25 + (pow((2.0*_re)/_L,
             4.0)))));
14        A = (2.0*M_PI*_k*_h)/_mi;
15        B = log((a + (sqrt(pow(a, 2.0) - pow(_L/2.0, 2.0))))/(_L
             /2.0));
16        C = (_h/_L)*log(_h/(2.0*_rw));
17
18        IP = A / (B + C);
19
20        return IP;
21}
22
23double CIPjoshi::CalcJoshiAnisotropico(double _k, double _h, double
     _mi, double _re, double _rw, double _L, double _Keff)
24{
25
26        double a, A, B, C;
27
28        a = (_L/2.0)*sqrt(0.5 + (sqrt(0.25 + (pow((2.0*_re)/_L,
             4.0)))));
29        A = (2.0*M_PI*_k*_h)/_mi;
30        B = log((a + (sqrt(pow(a, 2.0) - pow(_L/2.0, 2.0))))/(_L
             /2.0));
31        C = (_h*_Keff/_L)*log(_h*_Keff/(2.0*_rw));
```

```
32
33          IP = A / (B + C);
34
35          return IP;
36
37 }
```

Apresenta-se na listagem ?? o arquivo com código da classe CIPrenardDupuy.

Listing 6.17: Arquivo de cabeçalho da classe CIPrenardDupuy.

```
1 #ifndef CIPRENARDDUPUY_H_
2 #define CIPRENARDDUPUY_H_
3
4 #include "CCalcIP.h"
5
6 class CIPrenardDupuy : CCalcIP
7 {
8
9          public:
10
11                 CIPrenardDupuy(){};
12
13                 double CalcRenardDupuy(double _k, double _h, double
                       _mi, double _re, double _rw, double _L, double
                      _Keff);
14
15                 ~CIPrenardDupuy(){};
16
17 };
18
19 #endif
```

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

Listing 6.18: Arquivo de implementação da classe CIPrenardDupuy.

```
1 #define _USE_MATH_DEFINES
2 #include <math.h>
3
4 #include "CIPrenardDupuy.h"
5
6                  double CIPrenardDupuy::CalcRenardDupuy(double _k,
                      double _h, double _mi, double _re, double _rw,
                      double _L, double _Keff)
7                  {
```

```
8
9                       double a, X, A, rw, B, C, c;
10
11                      a = (_L/2.0)*sqrt(0.5 + (sqrt(0.25 + (pow
                            ((2.0*_re)/_L, 4.0)))));
12                      X = (2.0*a)/_L;
13                      A = (2.0*M_PI*_k*_h)/(_mi);
14                      rw = ((1.0 + _Keff)/(2.0*_Keff))*_rw;
15                      c = (_h)/(2.0*M_PI*rw);
16                      B = (((_Keff*_h)/_L))*(log(c));
17                      C = 1.0/((acosh(X)) + B);
18
19
20                      IP = A*(C);
21
22                      return IP;
23
24              }
```

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

Listing 6.19: Arquivo de cabeçalho da classe CIPshedid.

```
1#ifndef CIPSHEDID_H
2#define CIPSHEDID_H
3
4#include "CCalcIP.h"
5
6class CIPshedid : CCalcIP
7{
8
9       public:
10
11              CIPshedid(){};
12
13              double CalcShedid(double _k, double _h, double _mi,
                    double _re, double _rw, double _L, double _Bo);
14
15              ~CIPshedid(){};
16
17};
18
19#endif
```

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

Listing 6.20: Arquivo de implementação da classe CIPshedid.

```
#define _USE_MATH_DEFINES
#include <cmath>

#include "CIPshedid.h"

double CIPshedid::CalcShedid(double _k, double _h, double _mi,
    double _re, double _rw, double _L, double _Bo)
{

        double A, B, c, D, C;
        if (_L> 0.0 && _L<=1000.0)
                C= 270.0;
        else if (_L > 1000.0)
                C= 470.0 - .2*_L;

        A = ((2.0*M_PI*_k*_h)/(_mi*_Bo));
        B = (_h/(2.0*_rw))/(_L/_h);
        c = (.25 + (C/_L))*((1.0/_rw) - (2.0/_h));
        D = log(B) + c;

        IP = A/D;

        return IP;
}
```

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

Listing 6.21: Arquivo de cabeçalho da classe CSimuladorIP.

```
#ifndef CSIMULADOR_H_
#define CSIMULADOR_H_
#include <string>
#include <filesystem>

#include "CCalcIP.h"
#include "CFluido.h"
#include "CReservatorio.h"
#include "CReservatorioVertical.h"
#include "CPoco.h"
#include "CIPborisov.h"
#include "CIPgiger.h"
#include "CIPjoshi.h"
#include "CIPrenardDupuy.h"
```

```
15#include "CIPshedid.h"
16#include "CGnuplot.h"
17
18class CSimuladorIP
19{
20
21        public:
22
23                CFluido fluido;
24                CReservatorio reservatorio;
25                CReservatorioVertical reservatorioV;
26                CPoco poco;
27                CIPborisov borisov;
28                CIPgiger giger;
29                CIPjoshi joshi;
30                CIPrenardDupuy dupuy;
31                CIPshedid shedid;
32                Gnuplot plot;
33
34                CSimuladorIP(){};
35
36                void EntradaDados();
37                void Plot(std::vector <double> _plot, std::string
                     NomeArquivo);
38                void Executar();
39
40                ~CSimuladorIP(){};
41};
42
43#endif
```

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

Listing 6.22: Arquivo de implementação da classe CSimuladorIP.

```
1#define _USE_MATH_DEFINES
2#include <math.h>
3#include <iostream>
4#include <fstream>
5#include <string>
6#include <vector>
7#include <dirent.h>
8
9#include "CSimuladorIP.h"
```

```
10
11 using namespace std;
12
13 void CSimuladorIP::EntradaDados()
14 {
15
16         cout << "
               ##################################################################
               " << endl;
17         cout << "#                                                              
               #" << endl;
18         cout << "#                        Importacao de dados                   
               #" << endl;
19         cout << "#                                                              
               #" << endl;
20         cout << "
               ##################################################################
               " << endl << endl;
21
22         cout << "Digite nome do arquivo de dados." <<endl;
23
24         bool errado = true;
25
26         string path = ".";
27
28         cout << "\nArquivos Disponiveis\n" << endl;
29
30     for (const auto & file : filesystem::directory_iterator(path))
31         cout << file.path() << endl;
32
33         cout << endl;
34
35         do
36         {
37         string nomeArquivo;
38
39         cin.get();
40         getline (cin, nomeArquivo);
41
42         //nomeArquivo="Src/"+nomeArquivo;
43
44         ifstream in;
```

```
45
46          in.open(nomeArquivo, fstream::in);
47
48          double tmp;
49
50          in >> tmp;
51          reservatorioV.SetKv(tmp);
52          in >> tmp;
53          poco.SetRw(tmp);
54          in >> tmp;
55          reservatorio.SetRe(tmp);
56          in >> tmp;
57          reservatorio.SetL(tmp);
58          in >> tmp;
59          reservatorio.SetK(tmp);
60          in >> tmp;
61          reservatorio.SetH(tmp);
62          in >> tmp;
63          fluido.Setmi(tmp);
64          in >> tmp;
65          fluido.SetBo(tmp);
66
67          in.close();
68
69          cout << "\n
                ##################################################################
                " << endl;
70          cout << "#                                                              
                                                                                
                                                                                
                            #" << endl;
71          cout << "#                                  Dados estao
                corretos? 1 - sim   | 2 - nao                                    
                            #" << endl;
72          cout << "#    " << "Kh = " << reservatorio.GetK() << " | Rw
                = " << poco.GetRw() << " | Re = " << reservatorio.GetRe
                () << " | L = " << reservatorio.GetL() << " | Kv = " <<
                reservatorioV.GetKv() << " | H = " << reservatorio.GetH
                () << " | mi = " << fluido.Getmi() << " | Bo = " <<
                fluido.GetBo() << "             #" << endl;
73          cout << "#                                                              
                                                                                
                                                                                
                            #" << endl;
```

```
74          cout << "
                ##################################################################
                " << endl << endl;
75
76          cin >> tmp;
77
78          bool tst =true;
79
80          do
81          if (tmp == 1)
82          {
83          errado = false;
84          tst = false;
85          }
86          else if (tmp == 2)
87          {
88          errado = true;
89          tst = false;
90          cout << "\n
                ##################################################################
                " << endl;
91          cout << "#                                                              
                                  #" << endl;
92          cout << "#                          Importacao de dados                 
                                  #" << endl;
93          cout << "#                                                              
                                  #" << endl;
94          cout << "
                ##################################################################
                " << endl << endl;
95
96          cout << "Digite nome do arquivo de dados." <<endl;
97          }
98          else
99          {
100         cout << "opcao invalida!!!" <<endl;
101         cout << "\n
                ##################################################################
                " << endl;
102         cout << "#                                                              
                                                                                
                            #" << endl;
```

```
        cout << "#                                       Dados estao corretos? 1 - sim  | 2 - nao                                        #" << endl;
        cout << "#  " << "Kh = " << reservatorio.GetK() << " | Rw = " << poco.GetRw() << " | Re = " << reservatorio.GetRe() << " | L = " << reservatorio.GetL() << " | Kv = " << reservatorioV.GetKv() << " | H = " << reservatorio.GetH() << " | mi = " << fluido.Getmi() << " | Bo = " << fluido.GetBo() << "           #" << endl;
        cout << "#                                                                                                                               #" << endl;
        cout << "##################################################################" << endl << endl;
        cin >> tmp;
        }
        while(tst);
        }
        while(errado);

}

void CSimuladorIP::Plot(vector <double> _plot, string NomeArquivo)
{


        cout << "\n##################################################################" << endl;
        cout << "#                                                                                #" << endl;
        cout << "#                             Plotando Graficos                                 #" << endl;
        cout << "#                                                                                #" << endl;
        cout << "##################################################################" << endl << endl;

        Gnuplot::Terminal("qt");

```

```
        plot.set_xlabel("Modelo");
        plot.set_ylabel("IP");
        plot.set_xrange(-1 , 6);
        plot.Title("Indice de Produtividade");
        plot.cmd("unset xtics");
        plot.Cmd("set xtics (\"Borisov\" 0, \"Joshi\" 1, \"Joshi Anisotrópico\" 2, \"Giger\" 3, \"RenardDupuy\" 4, \"Shedid\" 5)");
        plot.Cmd("set boxwidth 0.5");
        plot.Cmd("set style fill solid 0.5");
        plot.set_style("histograms");

        plot.ShowOnScreen();

        plot.plot_x(_plot);
        plot.savetops(NomeArquivo);
        cout << "Aperte Enter para continuar ..." << endl;
        cin.get();
}

void CSimuladorIP::Executar()
{

        cout << "\n #################################################################
" << endl;
        cout << "#                                                                            #" << endl;
        cout << "#    Projeto Programacao Pratica - Calculo Indice de produtividade          #" << endl;
        cout << "#                                                                            #" << endl;
        cout << "# Professor: Andre Duarte Bueno                                             #" << endl;
        cout << "#                                                                            #" << endl;
        cout << "#    Alunos: Carolina Bastos                                                #" << endl;
        cout << "#            Douglas Ribeiro                                                #" << endl;
        cout << "#                                                                            #" << endl;
```

```
        cout << "################################################################" << endl << endl;

        cout << "Gostaria de executar o programa? 1 - Sim | 0 - nao " << endl;

        int opt;
        bool tst=true;

        cin >> opt;

        do
        if (opt!=1 && opt!=0)
        {
        cout << "opcao invalida\n" << endl;
        cout << "Gostaria de executar o programa? 1 - Sim | 0 - nao " << endl;
        cin >> opt;
        }
        else
        tst=false;
        while(tst);

        while (opt==1)
        {
                EntradaDados();

        cout << "\n################################################################" << endl;
        cout << "#                                                                           #" << endl;
        cout << "#             Qual nome do arquivo de saida de dados?                          #" << endl;
        cout << "#                                                                           #" << endl;
        cout << "################################################################" << endl << endl;


```

```
188                 string arquivoSaida;
189
190                 cin.get();
191                 getline(cin, arquivoSaida);
192
193                 arquivoSaida = "Src/Saida/"+arquivoSaida;
194
195                 ofstream out;
196                 out.open(arquivoSaida, fstream::out);
197
198                 double BORISOV = borisov.CalcBorisov(reservatorio.
                       GetK(), reservatorio.GetH(), fluido.Getmi(),
                       reservatorio.GetRe(), poco.GetRw(), reservatorio
                       .GetL());
199                 double JOSHI = joshi.CalcJoshiHorizontal(
                       reservatorio.GetK(), reservatorio.GetH(), fluido
                       .Getmi(), reservatorio.GetRe(), poco.GetRw(),
                       reservatorio.GetL());
200                 double keff = sqrt(reservatorio.GetK()/
                       reservatorioV.GetKv());
201                 double JOSHIANISIO = joshi.CalcJoshiAnisotropico(
                       reservatorio.GetK(), reservatorio.GetH(), fluido
                       .Getmi(), reservatorio.GetRe(), poco.GetRw(),
                       reservatorio.GetL(), keff);
202                 double GIGER = giger.CalcGiger(reservatorio.GetK(),
                        reservatorio.GetH(), fluido.Getmi(),
                       reservatorio.GetRe(), poco.GetRw(), reservatorio
                       .GetL());
203                 double DUPUY = dupuy.CalcRenardDupuy(reservatorio.
                       GetK(), reservatorio.GetH(), fluido.Getmi(),
                       reservatorio.GetRe(), poco.GetRw(), reservatorio
                       .GetL(), keff);
204                 double SHEDID = shedid.CalcShedid(reservatorio.GetK
                       (), reservatorio.GetH(), fluido.Getmi(),
                       reservatorio.GetRe(), poco.GetRw(), reservatorio
                       .GetL(), fluido.GetBo());
205
206                 vector <double> _plot;
207
208                 _plot.push_back(BORISOV);
209                 _plot.push_back(JOSHI);
210                 _plot.push_back(JOSHIANISIO);
```

```
                _plot.push_back(GIGER);
                _plot.push_back(DUPUY);
                _plot.push_back(SHEDID);


                out << "#Borisov Joshi JoshiVertical Giger RenardDupuy Shedid" << endl;
                out << BORISOV << " " << JOSHI << " " << JOSHIANISIO << " " << GIGER << " " << DUPUY << " " << SHEDID;

        cout << "\n#################################################################" << endl;
        cout << "#                                                                           #" << endl;
        cout << "#                              Dados Salvos!                                #" << endl;
        cout << "#                                                                           #" << endl;
        cout << "#################################################################" << endl << endl;


        cout << "Entre com come do arquivo de imagem:\n" << endl;

        string _nomeArquivo;

        cin.get();
        getline (cin, _nomeArquivo);

                Plot(_plot, _nomeArquivo);

                out.close();

                cout << "Gostaria de executar o programa? 1 - Sim | 0 - nao" << endl;
                cin >> opt;

        tst = true;

```

```
242         do
243         if (opt!=1 && opt!=0)
244         {
245         cout << "opcao invalida\n" << endl;
246         cout << "Gostaria de executar o programa? 1 - Sim | 0 - nao
               " << endl;
247         cin >> opt;
248         }
249         else
250         tst=false;
251         while(tst);
252         }
253
254
255 }
```

Apresenta-se na listagem 6.23 o programa main().

Listing 6.23: Arquivo de implementação da função `main()`.

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

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: Descrição

O presente trabalho apresenta interface em modo texto. O software, primeiramente, pergunta ao usuário se ele deseja a execução do programa e, após isso, o usuário entra com os dados do sistema poço-reservatório com extensão .dat e .txt. Veja Figura 7.1.

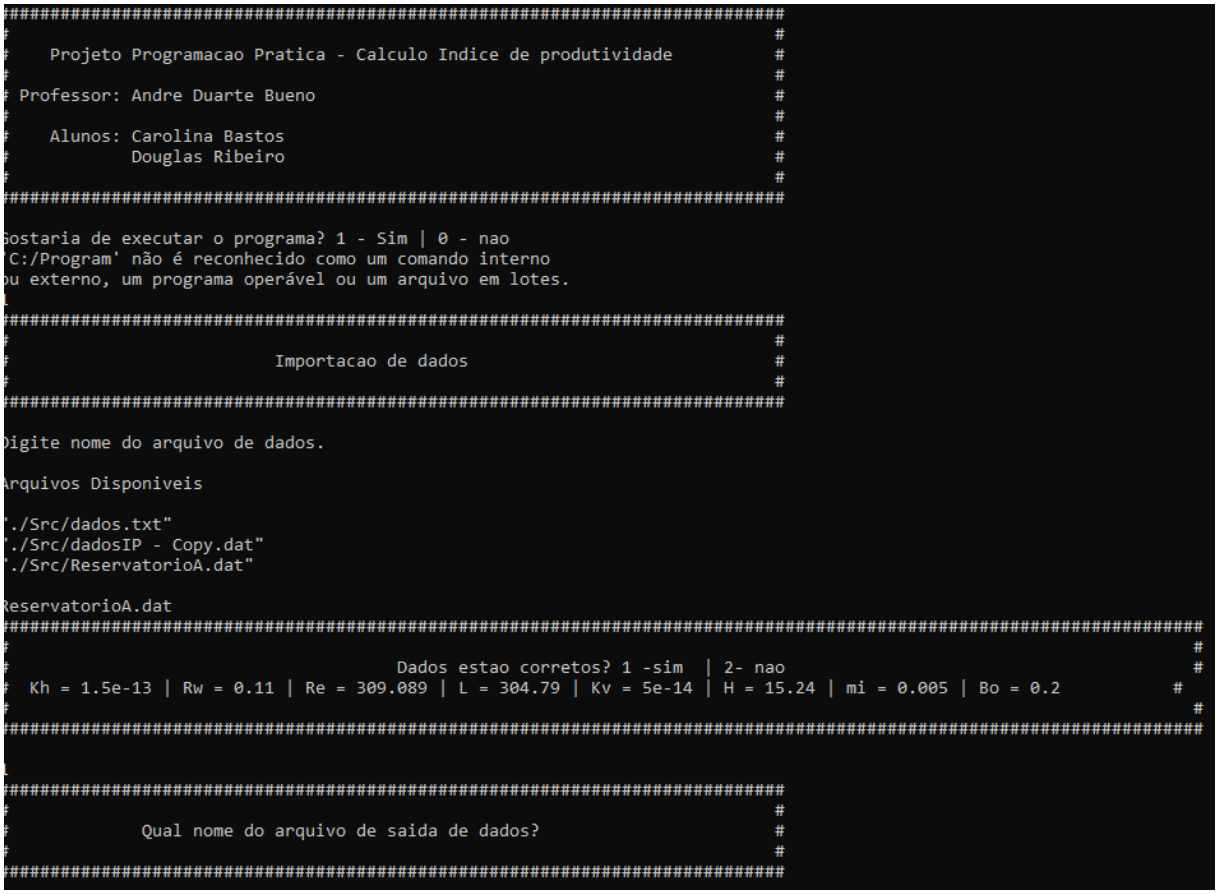

Figura 7.1: Interface com usuário do programa.

7.2 Teste 2: Descrição

No início apresente texto explicativo do teste:

- Neste programa, serão calculados os índices de produtividade de um determinado reservatório pelos seguintes métodos da literatura: Borisov, Joshi, Joshi Anisotrópico, Giger, Renard & Dupuy e Sheddid.
- O programa, primeiramente, precisa ser aberto no diretório onde o código se encontra e, depois disso, compila-se o código em um programa como o visual Studio 2019, neste caso, o programa foi compilado direto utilizando-se o dev C++.
- O programa pôde ser validado com base nos resultados obtidos na monografia em que foi baseado, neste caso, todos os dados possuem saída em .txt após a simulação para cada método.

Veja as figuras 7.2 e 7.3.

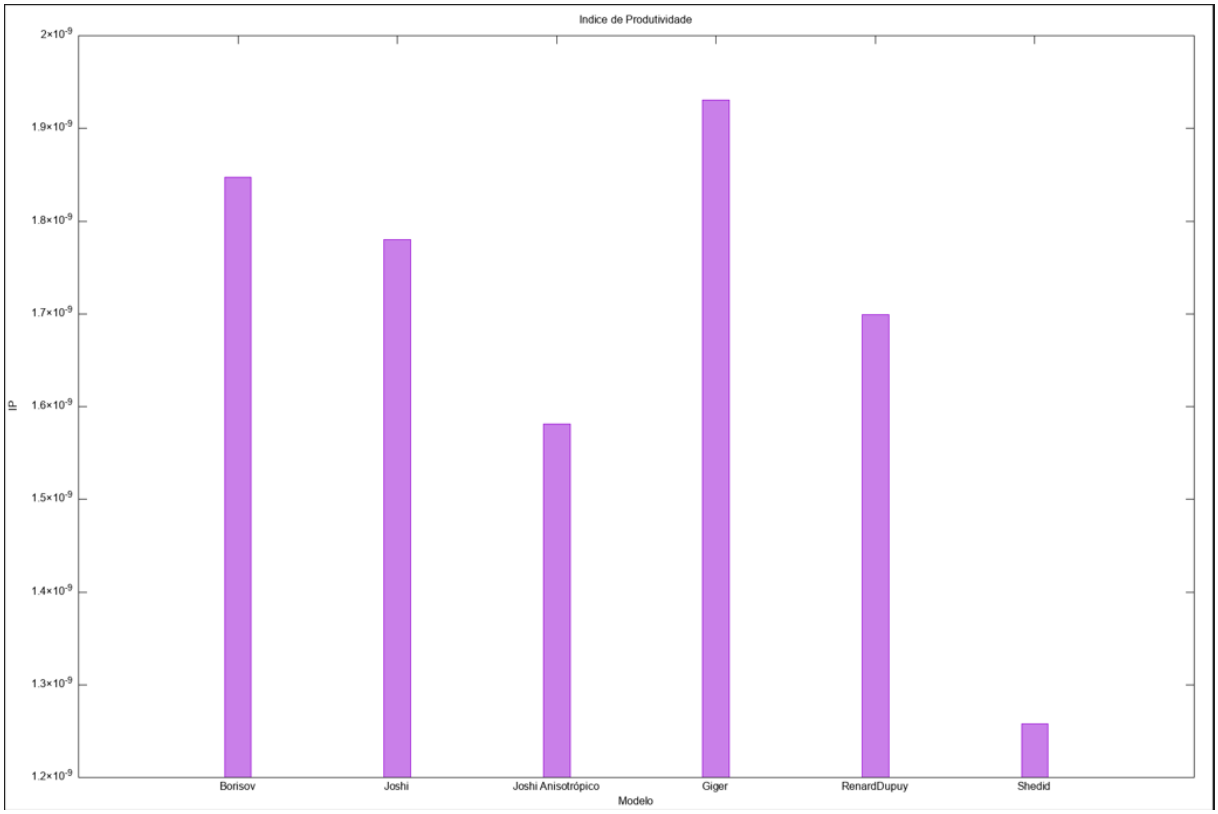

Figura 7.2: Histograma com os resultados para cada método do cálculo do Índice de Produtividade.

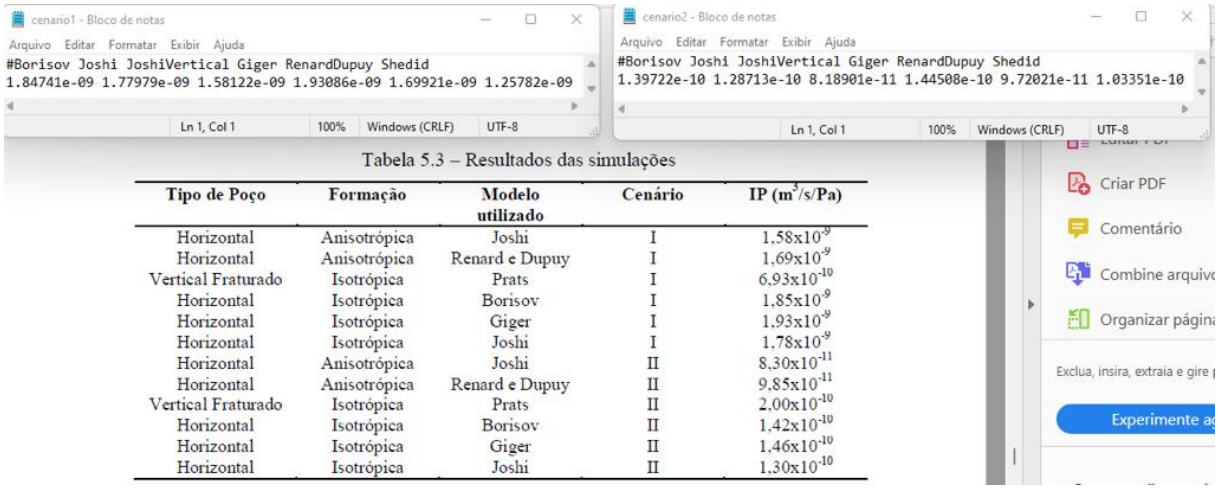

Tabela 5.3 – Resultados das simulações

Tipo de Poço	Formação	Modelo utilizado	Cenário	IP ($m^3/s/Pa$)
Horizontal	Anisotrópica	Joshi	I	$1{,}58x10^{-9}$
Horizontal	Anisotrópica	Renard e Dupuy	I	$1{,}69x10^{-9}$
Vertical Fraturado	Isotrópica	Prats	I	$6{,}93x10^{-10}$
Horizontal	Isotrópica	Borisov	I	$1{,}85x10^{-9}$
Horizontal	Isotrópica	Giger	I	$1{,}93x10^{-9}$
Horizontal	Isotrópica	Joshi	I	$1{,}78x10^{-9}$
Horizontal	Anisotrópica	Joshi	II	$8{,}30x10^{-11}$
Horizontal	Anisotrópica	Renard e Dupuy	II	$9{,}85x10^{-11}$
Vertical Fraturado	Isotrópica	Prats	II	$2{,}00x10^{-10}$
Horizontal	Isotrópica	Borisov	II	$1{,}42x10^{-10}$
Horizontal	Isotrópica	Giger	II	$1{,}46x10^{-10}$
Horizontal	Isotrópica	Joshi	II	$1{,}30x10^{-10}$

Figura 7.3: Validação dos cenários I e II.

Capítulo 8

Documentação

A presente documentação refere-se ao uso do "Programa em C++ para o cálculo do Índice de Produtividade de Poços Horizontais e Verticais. Esta documentação tem o formato de uma apostila que explica passo a passo ao usuário como usar o programa.

8.1 Documentação do usuário

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

8.1.1 Como rodar o software

Abra o terminal, vá para o diretório onde está o projeto, compile o programa e depois o execute. Logo após, siga os seguintes passos:

1. O programa apresentará uma pequena interface e perguntará se o usuário quer continuar a execução, sendo 1 para continuar e 0 para não, caso o usuário digite outro valor, aparecerá uma mensagem de erro pedindo que digite os valores anteriormente ditos.

2. Selecionando-se a opção 1, o programa vai para o diretório onde se encontra os arquivos em formato .txt, que possuem nesta sequência e nesta ordem as propriedades do poço e do reservatório e esses dados devem ser postos no sistema S.I de unidades para gerar o índice de produtividade correto: permeabilidade horizontal, raio do poço, raio do reservatório, comprimento do poço, permeabilidade vertical, altura do reservatório, viscosidade do fluido, fator volume-formação do fluido.

3. Após o usuário escolher o sistema reservatório-poço a ser analisado na pasta “Src” , o software vai mostrar os dados na tela e perguntará se estão corretos, sendo 1 para confirmar e 2 sinalizará que os dados estão incorretos.

4. Em seguida, após a confirmação dos dados, o programa pedirá um nome para o arquivo de saída a qual ficará à escolha do usuário.

5. Feita a escolha do nome, o programa mostrará na tela que está a plotar os gráficos e basta apertar enter para que a figura com um histograma apareça na tela com os resultados

obtidos dos índices de produtividade de cada tipo: Borisov, Joshi, Joshi Anisotrópico, Giger, Renard & Dupuy e Sheddid.

6. A figura será salva no diretório onde se encontra o código fonte e um arquivo .txt será gerado com os valores dos índices de produtividade.

7. Por último, o programa vai perguntar ao usuário se ele quer fazer uma nova simulação.

8.2 Documentação para desenvolvedor

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

8.2.1 Dependências

Para compilar o software é necessário atender as seguintes dependências:

- Instalar o compilador g++ da GNU disponível em http://gcc.gnu.org. Para instalar no GNU/Linux use o comando `yum install gcc`.
- Biblioteca CGnuplot; os arquivos para acesso a biblioteca CGnuplot devem estar no diretório com os códigos do software;
- O software `gnuplot`, disponível no endereço http://www.gnuplot.info/, deve estar instalado. É possível que haja necessidade de setar o caminho para execução do `gnuplot`.
- É recomendado usar versões mais atualizadas de compiladores para se conseguir usar a biblioteca filesystem, que somente pode ser usada para versões C++17 a C++20.

8.2.2 Como gerar a documentação usando doxygen

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

- 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`.

Figura 8.1: Documentação do código no Doxygen.

Referências Bibliográficas

[Blaha and Rumbaugh, 2006] Blaha, M. and Rumbaugh, J. (2006). *Modelagem e Projetos Baseados em Objetos com UML 2.* Campus, Rio de Janeiro. 20

[JOSHI, 1988] JOSHI, S. D. (1988). *Production Forecasting Methods for Horizontal Wells.* SPE 17850, Tianjin, China. 8, 9

[ROSA, 2006] ROSA, A. (2006). *Engenharia de Reservatorios de Petroleo.* Editora Interciencia, Rio de Janeiro. 9

[Rumbaugh et al., 1994] Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., and Lorensen, W. (1994). *Modelagem e Projetos Baseados em Objetos.* Edit. Campus, Rio de Janeiro. 20

[SHEDID, 2001] SHEDID, S. A. (2001). *Sensitivity Analysis of Horizontal Well Productivity under Steady-State Conditions.* SPE 72121, Kuala Lumpur. 10

Índice Remissivo