

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

1	Introdução	1
1.1	Escopo do problema	1
1.2	Objetivos	1
2	Especificação	3
2.1	O que é a especificação?	3
2.2	Nome do sistema/produto	3
2.3	Especificação	3
2.3.1	Requisitos funcionais	4
2.3.2	Requisitos não funcionais	4
2.4	Casos de uso	4
2.4.1	Diagrama de caso de uso geral	5
2.4.2	Diagrama de caso de uso específico	5
3	Elaboração	7
3.1	Análise de domínio	7
3.2	Formulação teórica	8
3.2.1	Produtividade de Poços	8
3.2.2	Índice de Produtividade	8
3.2.3	Efeito Skin	8
3.2.4	Regime Permanente	9
3.2.5	Produtividade de Poços Horizontais	9
3.2.6	Cálculo do IP com anisotropia	10
3.3	Diagrama de pacotes – assuntos	11
4	AOO – Análise Orientada a Objeto	13
4.1	Diagramas de classes	13
4.1.1	Dicionário de classes	15
4.2	Diagrama de seqüência – eventos e mensagens	15
4.2.1	Diagrama de seqüência geral	15
4.3	Diagrama de comunicação – colaboração	16
4.4	Diagrama de máquina de estado	17

4.5	Diagrama de atividades	19
5	Projeto	20
5.1	Projeto do sistema	20
5.2	Projeto orientado a objeto – POO	21
5.3	Diagrama de componentes	25
5.4	Diagrama de implantação	26
6	Implementação	27
6.1	Código fonte	27
7	Teste	118
7.1	Teste 1: Descrição	118
7.2	Teste 2: Descrição	118
8	Documentação	120
8.1	Documentação do usuário	120
8.1.1	Como rodar o software	120
8.2	Documentação para desenvolvedor	121
8.2.1	Dependências	121
8.2.2	Como gerar a documentação usando doxygen	121

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 e RenardDupuy e Shedid.
 - Simular a influência de parâmetros 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 design (horizontal/vertical).

2.3 Especificação

Apresenta-se a seguir a especificação 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 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 <i>Windows</i> , <i>GNU/Linux</i> ou <i>Mac</i> .

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:	<ol style="list-style-type: none"> 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 análise 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.

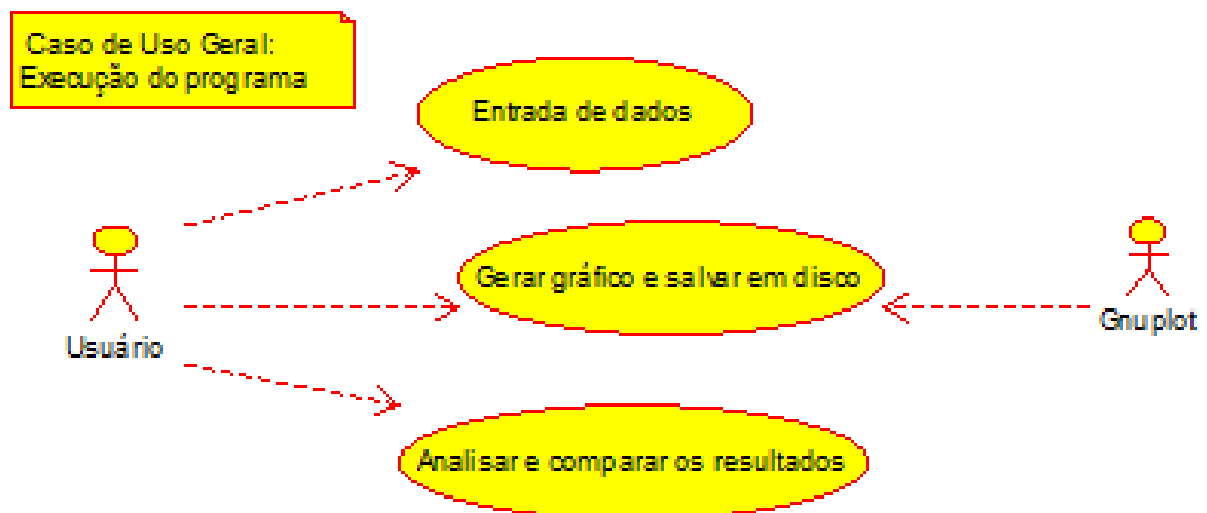


Figura 2.1: Diagrama de caso de uso geral – Cálculo do IP

2.4.2 Diagrama de caso de uso específico

O caso de uso Comparar IP de um reservatório anisotrópico e isotrópico 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.

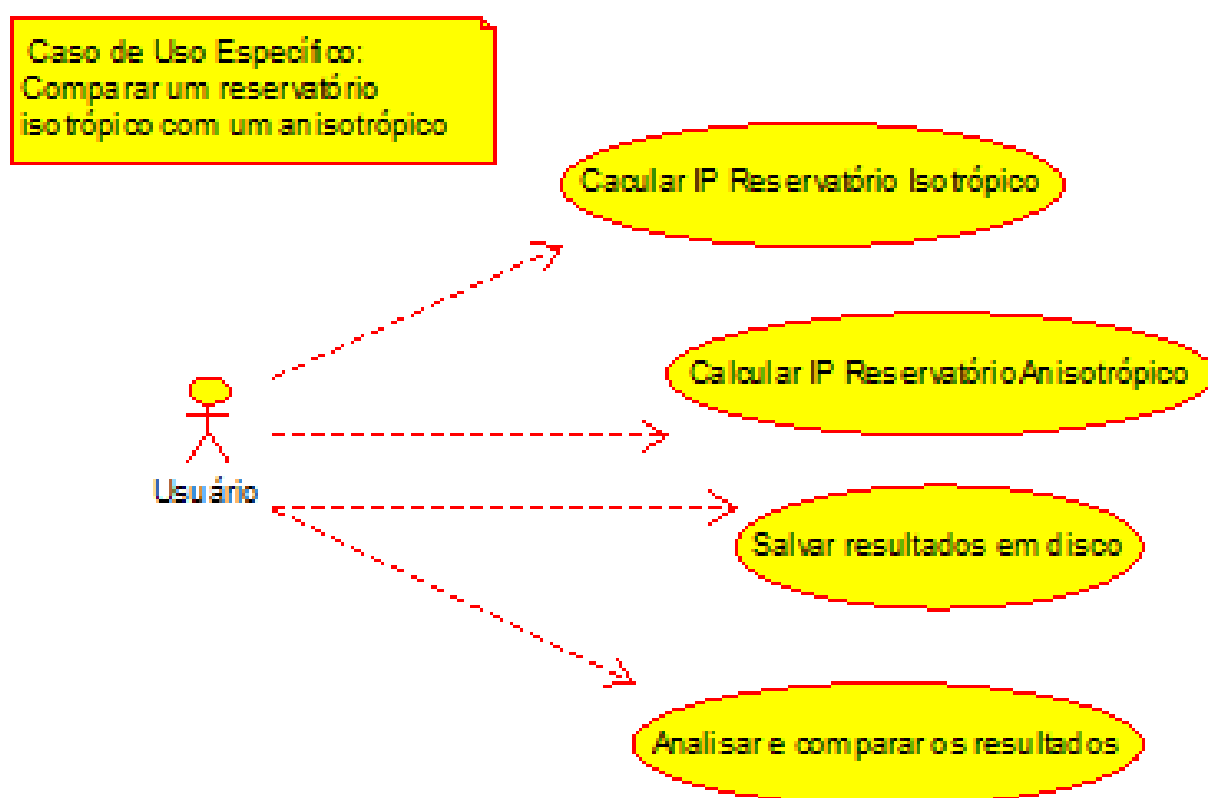


Figura 2.2: Diagrama de caso de uso específico – Comparando o IP de reservatório isotrópico com um anisotrópico

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 explicações 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 :

$$IP = \frac{Q}{P_e - P_w} \quad (3.1)$$

Onde:

$Q = \text{vazão} [m^3/d]$

$P_e = \text{pressão estática do reservatório} [kgf/cm^2]$

$P_w = \text{pressão de fluxo do poço} [kgf/cm^2]$

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:

$$r'_w = r e^{-s} \quad (3.2)$$

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:

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

- Giger:

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

- Joshi:

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

Onde:

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

IP = Índice de Produtividade

k_h = permeabilidade horizontal

h = altura do reservatório

μ = 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]:

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

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

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

Onde:

$$\beta = \sqrt{\frac{k_h}{k_v}} \quad (3.9)$$

ΔP é a queda de pressão no reservatório

- Modelo de Renard e Dupuy:

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

Onde:

$$r'_w = \frac{1 + \beta}{2\beta} r_w \quad (3.11)$$

$$X = \frac{2a}{L} \quad (3.12)$$

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 identificados os seguintes pacotes:

- Pacote Fluido: Engloba as características 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.

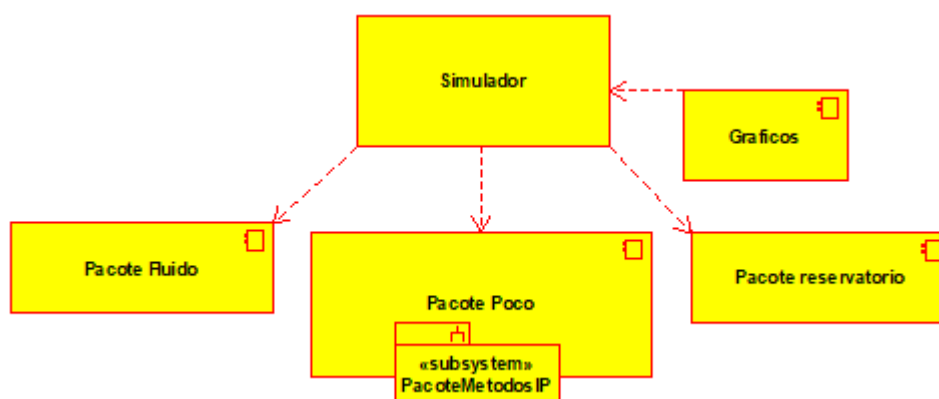


Figura 3.2: Diagrama de Pacotes

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 modificações do objeto com o tempo. O resultado da análise é um conjunto de diagramas que identificam os objetos e seus relacionamentos.

4.1 Diagramas de classes

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

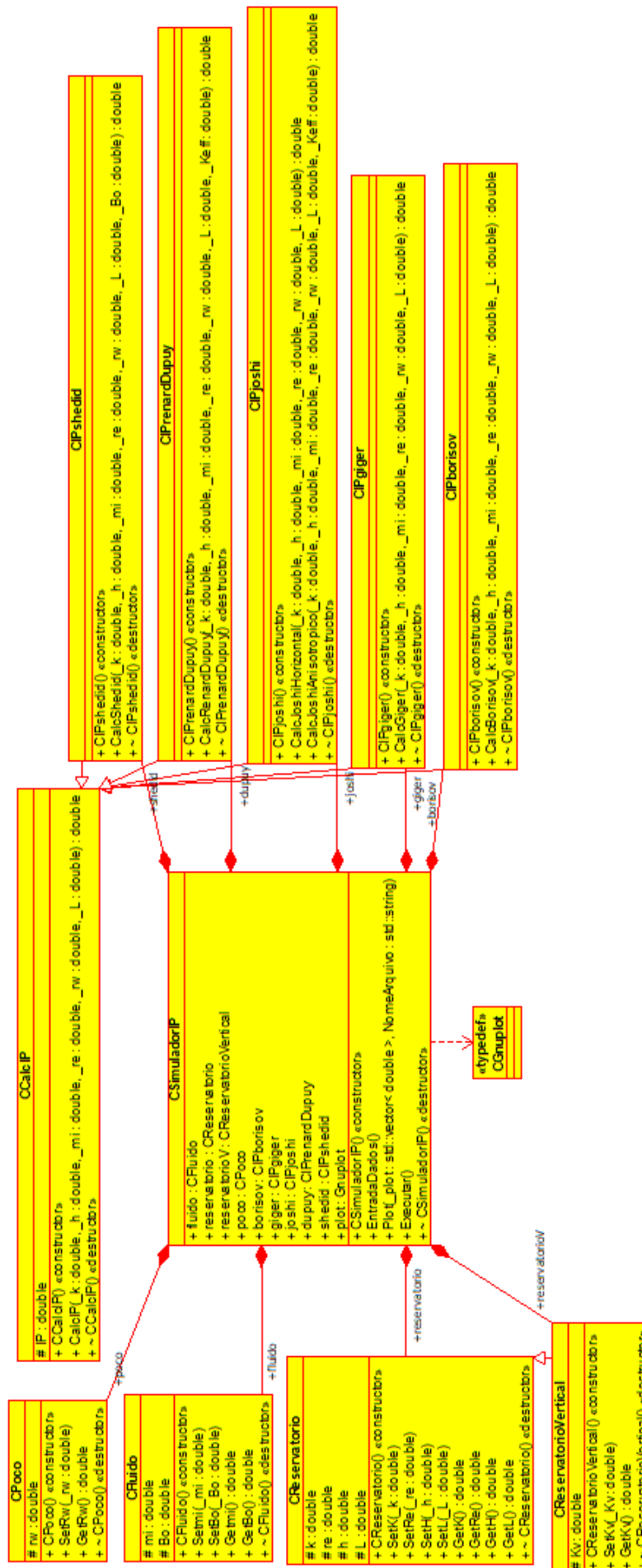


Figura 4.1: Diagrama de classes

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 CIPjoshiAnisotropico: Calcula o IP do poço pelo modelo de Joshi em casos de reservatórios anisotrópicos.
- Classe CIPjoshi: Calcula o IP do poço pelo modelo de Joshi.
- Classe CIPgiger: Calcula o IP do poço pelo modelo de Giger.
- Classe CIPgorisov: 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 seqüência geral

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

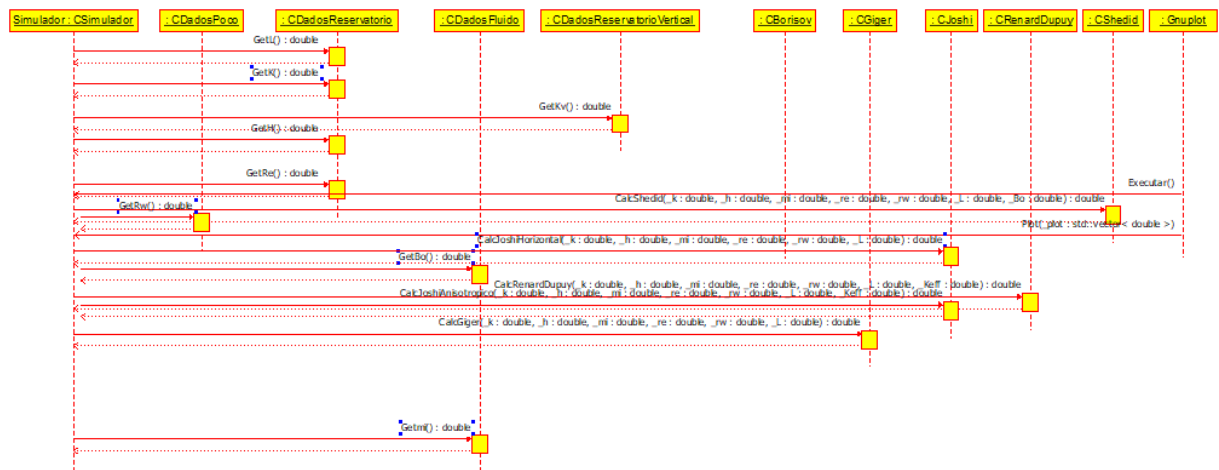


Figura 4.2: Diagrama de sequência geral

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 utilizado 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 vez chama a classe `CIPjoshi`. A classe `CSimuladorIP` então, após realizar os cálculos, envia esses resultados para a classe `CGnuplot` que gera a visualização e salva esses resultados de forma gráfica.

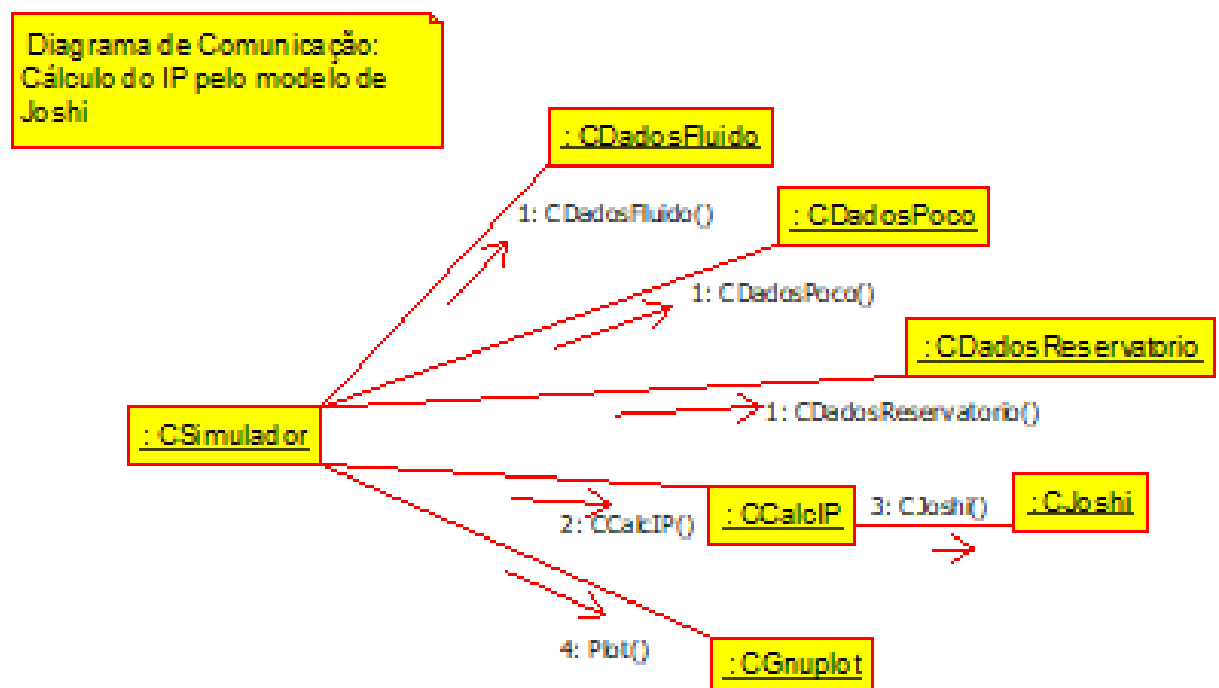


Figura 4.3: Diagrama de comunicação

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.

Diagrama Máquina de Estado: Sequência de cálculo dos métodos de IP

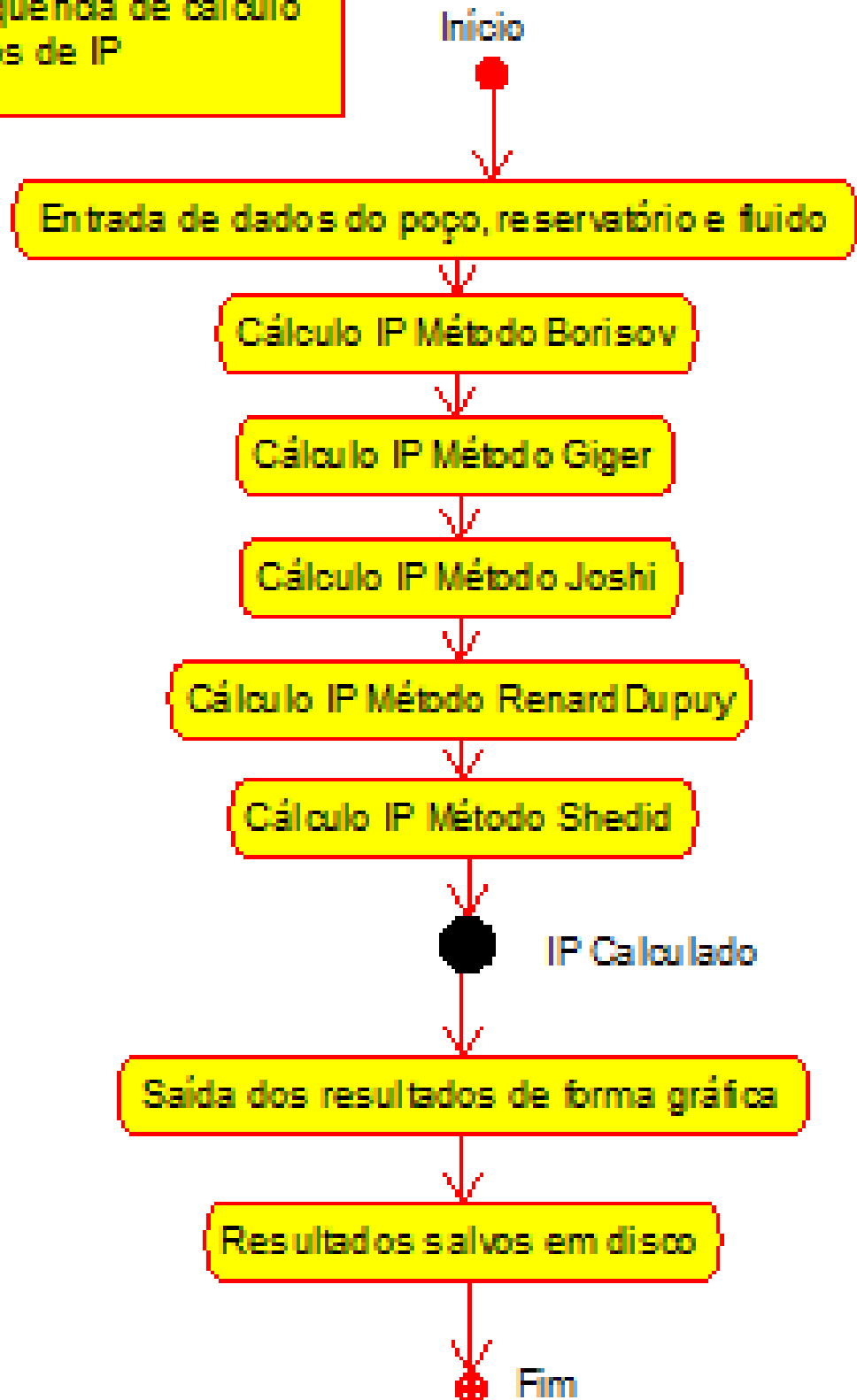


Figura 4.4: Diagrama de máquina de estado

4.5 Diagrama de atividades

O diagrama de atividades da figura 4.5 corresponde a uma atividade específica do diagrama de máquina 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 parâmetros do reservatório como raio externo, espessura, comprimento e permeabilidade vertical, que são informações necessárias para calcular IP. O modelo escolhido como exemplo é o Borisov.

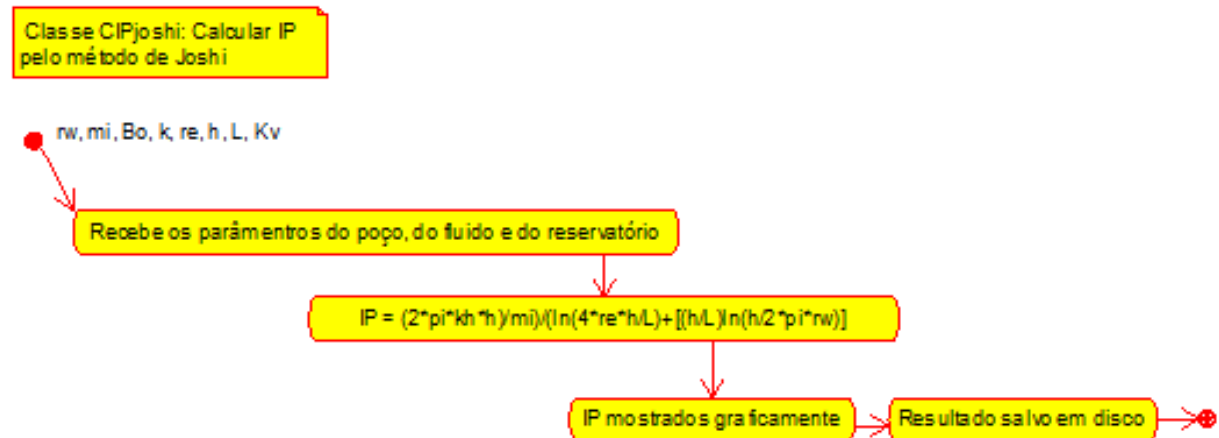


Figura 4.5: Diagrama de atividades

Capítulo 5

Projeto

Neste capítulo do projeto de engenharia veremos questões associadas ao projeto do sistema, incluindo protocolos, recursos, plataformas suportadas, 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 [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, ostream, 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 demais classes que calcula o índice de produtividade dos poços de acordo com modelos específicos, o diagrama de seqüência precisou ser revisado e a seqüência alterada para inclusão dessa etapa em que a CCalcIP acessasse os cálculos das classes herdeiras.
 - O mesmo se aplica ao diagrama de comunicação, com a inclusão dessa classe genérica CCalcIP a comunicação 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 construção do código que alterou a seqüência 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 calcula 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 relacionamento 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 atributos 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.

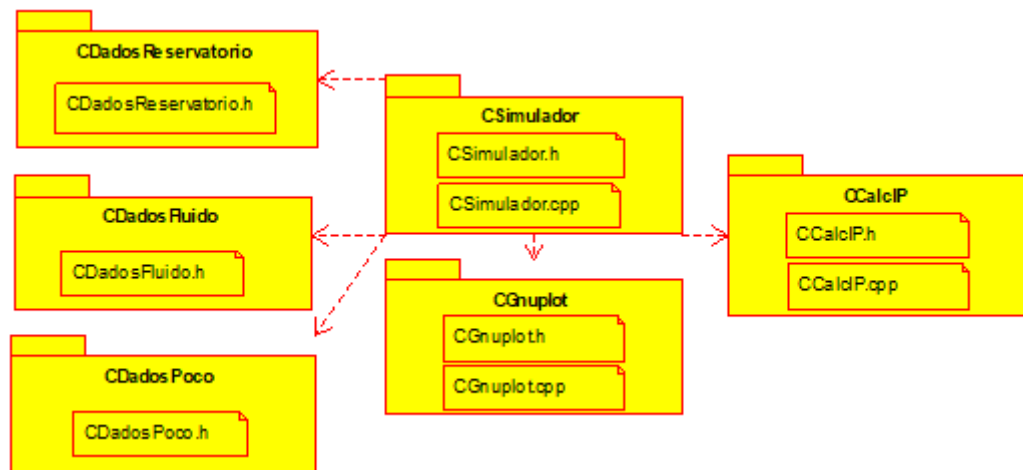


Figura 5.1: Diagrama de componentes

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.

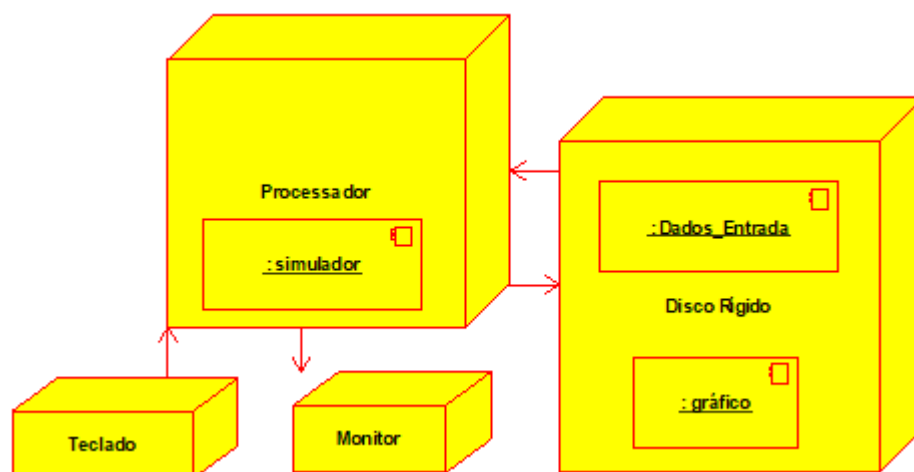


Figura 5.2: Diagrama de implantação

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
4 class 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
3 void CPoco::SetRw(double _rw)
4 {
5     rw = _rw;
6 }
7
8 double 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
4 class 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.

```

1 #define _USE_MATH_DEFINES
2 #include <math.h>
3
4 #include "CReservatorioVertical.h"
5
6         void CReservatorioVertical::SetKv(double _Kv)
7         {
8             Kv = _Kv;
9         }
10
11
12         double CReservatorioVertical::GetKv()
13         {
14             return Kv;
15         }

```

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

Listing 6.9: Arquivo de cabeçalho da classe CCalcIP.

```

1 #ifndef CCalcIP_H
2 #define CCalcIP_H
3
4 class CCalcIP
5 {
6
7     protected:
8
9         double IP;
10
11     public:
12
13         CCalcIP(){};
14
15         double CalcIP(double _k, double _h, double _mi,
16                     double _re, double _rw, double _L){return IP;};
17
18         ~CCalcIP(){};
19 };
20
21 #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
6 class CIPborisov : CCalcIP
7 {
8
9     public:
10
11         CIPborisov(){};
12
13         double CalcBorisov(double _k, double _h, double _mi,
14                             , double _re, double _rw, double _L);
15
16         ~CIPborisov(){};
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
8 double CIPborisov::CalcBorisov(double _k, double _h, double _mi,
9                                 double _re, double _rw, double _L)
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,
15                         double _re, double _rw, double _L);
16
17         ~CIPgiger(){};
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
7 double 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
6 class 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
8 double 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,
14         4.0)))));
15     A = (2.0*M_PI*_k*_h)/_mi;
16     B = log((a + (sqrt(pow(a, 2.0) - pow(_L/2.0, 2.0))))/(_L
17         /2.0));
18     C = (_h/_L)*log(_h/(2.0*_rw));
19
20     IP = A / (B + C);
21
22     return IP;
23 }
24
25 double CIPjoshi::CalcJoshiAnisotropico(double _k, double _h, double
   _mi, double _re, double _rw, double _L, double _Keff)
26 {
27
28     double a, A, B, C;
29
30     a = (_L/2.0)*sqrt(0.5 + (sqrt(0.25 + (pow((2.0*_re)/_L,
31         4.0)))));
32     A = (2.0*M_PI*_k*_h)/_mi;
33     B = log((a + (sqrt(pow(a, 2.0) - pow(_L/2.0, 2.0))))/(_L
34         /2.0));
35     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
14             _mi, double _re, double _rw, double _L, double
15             _Keff);
16
17         ~CIPrenardDupuy(){};
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,
7         double _h, double _mi, double _re, double _rw,
8         double _L, double _Keff)
9     {

```

```

8
9         double a, X, A, rw, B, C, c;
10
11         a = (_L/2.0)*sqrt(0.5 + (sqrt(0.25 + (pow
12             ((2.0*_re)/_L, 4.0)))));
13         X = (2.0*a)/_L;
14         A = (2.0*M_PI*_k*_h)/(_mi);
15         rw = ((1.0 + _Keff)/(2.0*_Keff))*_rw;
16         c = (_h)/(2.0*M_PI*rw);
17         B = (((_Keff*_h)/_L))*(log(c));
18         C = 1.0/((acosh(X)) + B);
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
6 class CIPshedid : CCalcIP
7 {
8
9     public:
10
11         CIPshedid(){};
12
13         double CalcShedid(double _k, double _h, double _mi,
14             double _re, double _rw, double _L, double _Bo);
15
16         ~CIPshedid(){};
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.

```

1 #define _USE_MATH_DEFINES
2 #include <cmath>
3
4 #include "CIPshedid.h"
5
6 double CIPshedid::CalcShedid(double _k, double _h, double _mi,
    double _re, double _rw, double _L, double _Bo)
7 {
8
9     double A, B, c, D, C;
10    if (_L > 0.0 && _L <= 1000.0)
11        C = 270.0;
12    else if (_L > 1000.0)
13        C = 470.0 - .2*_L;
14
15    A = ((2.0*M_PI*_k*_h)/(_mi*_Bo));
16    B = (_h/(2.0*_rw))/(_L/_h);
17    c = (.25 + (C/_L))*((1.0/_rw) - (2.0/_h));
18    D = log(B) + c;
19
20    IP = A/D;
21
22    return IP;
23 }

```

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

Listing 6.21: Arquivo de cabeçalho da classe CSimuladorIP.

```

1 #ifndef CSIMULADOR_H_
2 #define CSIMULADOR_H_
3 #include <string>
4 #include <filesystem>
5
6 #include "CCalcIP.h"
7 #include "CFluido.h"
8 #include "CReservatorio.h"
9 #include "CReservatorioVertical.h"
10 #include "CPoco.h"
11 #include "CIPborisov.h"
12 #include "CIPgiger.h"
13 #include "CIPjoshi.h"
14 #include "CIPrenardDupuy.h"

```

```

15 #include "CIPshedid.h"
16 #include "CGnuplot.h"
17
18 class 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 <conio.h>
8 #include <dirent.h>
9

```

```

10 #include "CSimuladorIP.h"
11
12 using namespace std;
13
14 void CSimuladorIP::EntradaDados()
15 {
16
17     cout << "
18         #####
19         " << endl;
20     cout << "#_#####
21         _#####" << endl;
22     cout << "#_#####Importacao_de_dados_####
23         _#####" << endl;
24     cout << "#_#####
25         _#####" << endl;
26     cout << "
27         #####
28         " << endl << endl;
29
30     cout << "Digite_nome_do_arquivo_de_dados." << endl;
31
32     bool errado = true;
33
34     string path = "./Src/";
35
36     cout << "\nArquivos_Disponiveis\n" << endl;
37
38     for (const auto & file : filesystem::directory_iterator(path))
39         cout << file.path() << endl;
40
41     cout << endl;
42
43     do
44     {
45         string nomeArquivo;
46
47         cin.get();
48         getline (cin, nomeArquivo);
49
50         nomeArquivo="Src/"+nomeArquivo;
51     }

```

```

45     ifstream in;
46
47     in.open(nomeArquivo, fstream::in);
48
49     double tmp;
50
51     in >> tmp;
52     reservatorioV.SetKv(tmp);
53     in >> tmp;
54     poco.SetRw(tmp);
55     in >> tmp;
56     reservatorio.SetRe(tmp);
57     in >> tmp;
58     reservatorio.SetL(tmp);
59     in >> tmp;
60     reservatorio.SetK(tmp);
61     in >> tmp;
62     reservatorio.SetH(tmp);
63     in >> tmp;
64     fluido.Setmi(tmp);
65     in >> tmp;
66     fluido.SetBo(tmp);
67
68     in.close();
69
70     cout << "\n
71         #####
72         " << endl;
73     cout << "#_
74         _
75         _
76         _#" << endl;
77     cout << "#_
78         _
79         _
80         _
81         _
82         _
83         _
84         _
85         _
86         _
87         _
88         _
89         _
90         _
91         _
92         _
93         _
94         _
95         _
96         _
97         _
98         _
99         _
100        _
101        _
102        _
103        _
104        _
105        _
106        _
107        _
108        _
109        _
110        _
111        _
112        _
113        _
114        _
115        _
116        _
117        _
118        _
119        _
120        _
121        _
122        _
123        _
124        _
125        _
126        _
127        _
128        _
129        _
130        _
131        _
132        _
133        _
134        _
135        _
136        _
137        _
138        _
139        _
140        _
141        _
142        _
143        _
144        _
145        _
146        _
147        _
148        _
149        _
150        _
151        _
152        _
153        _
154        _
155        _
156        _
157        _
158        _
159        _
160        _
161        _
162        _
163        _
164        _
165        _
166        _
167        _
168        _
169        _
170        _
171        _
172        _
173        _
174        _
175        _
176        _
177        _
178        _
179        _
180        _
181        _
182        _
183        _
184        _
185        _
186        _
187        _
188        _
189        _
190        _
191        _
192        _
193        _
194        _
195        _
196        _
197        _
198        _
199        _
200        _
201        _
202        _
203        _
204        _
205        _
206        _
207        _
208        _
209        _
210        _
211        _
212        _
213        _
214        _
215        _
216        _
217        _
218        _
219        _
220        _
221        _
222        _
223        _
224        _
225        _
226        _
227        _
228        _
229        _
230        _
231        _
232        _
233        _
234        _
235        _
236        _
237        _
238        _
239        _
240        _
241        _
242        _
243        _
244        _
245        _
246        _
247        _
248        _
249        _
250        _
251        _
252        _
253        _
254        _
255        _
256        _
257        _
258        _
259        _
260        _
261        _
262        _
263        _
264        _
265        _
266        _
267        _
268        _
269        _
270        _
271        _
272        _
273        _
274        _
275        _
276        _
277        _
278        _
279        _
280        _
281        _
282        _
283        _
284        _
285        _
286        _
287        _
288        _
289        _
290        _
291        _
292        _
293        _
294        _
295        _
296        _
297        _
298        _
299        _
300        _
301        _
302        _
303        _
304        _
305        _
306        _
307        _
308        _
309        _
310        _
311        _
312        _
313        _
314        _
315        _
316        _
317        _
318        _
319        _
320        _
321        _
322        _
323        _
324        _
325        _
326        _
327        _
328        _
329        _
330        _
331        _
332        _
333        _
334        _
335        _
336        _
337        _
338        _
339        _
340        _
341        _
342        _
343        _
344        _
345        _
346        _
347        _
348        _
349        _
350        _
351        _
352        _
353        _
354        _
355        _
356        _
357        _
358        _
359        _
360        _
361        _
362        _
363        _
364        _
365        _
366        _
367        _
368        _
369        _
370        _
371        _
372        _
373        _
374        _
375        _
376        _
377        _
378        _
379        _
380        _
381        _
382        _
383        _
384        _
385        _
386        _
387        _
388        _
389        _
390        _
391        _
392        _
393        _
394        _
395        _
396        _
397        _
398        _
399        _
400        _
401        _
402        _
403        _
404        _
405        _
406        _
407        _
408        _
409        _
410        _
411        _
412        _
413        _
414        _
415        _
416        _
417        _
418        _
419        _
420        _
421        _
422        _
423        _
424        _
425        _
426        _
427        _
428        _
429        _
430        _
431        _
432        _
433        _
434        _
435        _
436        _
437        _
438        _
439        _
440        _
441        _
442        _
443        _
444        _
445        _
446        _
447        _
448        _
449        _
450        _
451        _
452        _
453        _
454        _
455        _
456        _
457        _
458        _
459        _
460        _
461        _
462        _
463        _
464        _
465        _
466        _
467        _
468        _
469        _
470        _
471        _
472        _
473        _
474        _
475        _
476        _
477        _
478        _
479        _
480        _
481        _
482        _
483        _
484        _
485        _
486        _
487        _
488        _
489        _
490        _
491        _
492        _
493        _
494        _
495        _
496        _
497        _
498        _
499        _
500        _
501        _
502        _
503        _
504        _
505        _
506        _
507        _
508        _
509        _
510        _
511        _
512        _
513        _
514        _
515        _
516        _
517        _
518        _
519        _
520        _
521        _
522        _
523        _
524        _
525        _
526        _
527        _
528        _
529        _
530        _
531        _
532        _
533        _
534        _
535        _
536        _
537        _
538        _
539        _
540        _
541        _
542        _
543        _
544        _
545        _
546        _
547        _
548        _
549        _
550        _
551        _
552        _
553        _
554        _
555        _
556        _
557        _
558        _
559        _
560        _
561        _
562        _
563        _
564        _
565        _
566        _
567        _
568        _
569        _
570        _
571        _
572        _
573        _
574        _
575        _
576        _
577        _
578        _
579        _
580        _
581        _
582        _
583        _
584        _
585        _
586        _
587        _
588        _
589        _
590        _
591        _
592        _
593        _
594        _
595        _
596        _
597        _
598        _
599        _
600        _
601        _
602        _
603        _
604        _
605        _
606        _
607        _
608        _
609        _
610        _
611        _
612        _
613        _
614        _
615        _
616        _
617        _
618        _
619        _
620        _
621        _
622        _
623        _
624        _
625        _
626        _
627        _
628        _
629        _
630        _
631        _
632        _
633        _
634        _
635        _
636        _
637        _
638        _
639        _
640        _
641        _
642        _
643        _
644        _
645        _
646        _
647        _
648        _
649        _
650        _
651        _
652        _
653        _
654        _
655        _
656        _
657        _
658        _
659        _
660        _
661        _
662        _
663        _
664        _
665        _
666        _
667        _
668        _
669        _
670        _
671        _
672        _
673        _
674        _
675        _
676        _
677        _
678        _
679        _
680        _
681        _
682        _
683        _
684        _
685        _
686        _
687        _
688        _
689        _
690        _
691        _
692        _
693        _
694        _
695        _
696        _
697        _
698        _
699        _
700        _
701        _
702        _
703        _
704        _
705        _
706        _
707        _
708        _
709        _
710        _
711        _
712        _
71
```

```

        #####" << endl;
75     cout << "
        #####
        " << endl << endl;

76
77     cin >> tmp;
78
79     bool tst = true;
80
81     do
82     if (tmp == 1)
83     {
84         errado = false;
85         tst = false;
86     }
87     else if (tmp == 2)
88     {
89         errado = true;
90         tst = false;
91     cout << "\n
        #####
        " << endl;
92     cout << "######
        #####" << endl;
93     cout << "######Importacao de dados#####
        #####" << endl;
94     cout << "######
        #####" << endl;
95     cout << "
        #####
        " << endl << endl;

96
97     cout << "Digite nome do arquivo de dados." << endl;
98     }
99     else
100    {
101    cout << "opcao invalida!!!" << endl;
102    cout << "\n
        #####
        " << endl;
103    cout << "######
        #####

```

```

        "#####" << endl;
104 cout << "######Dados_estao
        _corretos?_1_-sim_|_2_-nao#####
        " << endl;
105 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;
106 cout << "######
        #####
        " << endl;
107 cout << "
        #####
        " << endl << endl;
108 cin >> tmp;
109 }
110 while(tst);
111 }
112 while(erroado);
113
114 }
115
116 void CSimuladorIP::Plot(vector <double> _plot, string NomeArquivo)
117 {
118
119
120 cout << "\n
        #####
        " << endl;
121 cout << "######
        #####" << endl;
122 cout << "######Plotando_Graficos_
        #####" << endl;
123 cout << "######
        #####" << endl;
124 cout << "
        #####
        " << endl << endl;
125
126 Gnuplot::Terminal("qt");

```

```

127     plot.set_xlabel("Modelo");
128     plot.set_ylabel("IP");
129     plot.set_xrange(-1 , 6);
130     plot.Title("Indice_de_Produtividade");
131     plot.cmd("unset_xtics");
132     plot.Cmd("set_xtics(\"Borisov\"_0,\"Joshi\"_1,\"Joshi_
133         Anisotrópico\"_2,\"Giger\"_3,\"RenardDupuy\"_4,\"
134         Shedid\"_5)\");
135     plot.Cmd("set_boxwidth_0.5");
136     plot.Cmd("set_style_fill_solid_0.5");
137     plot.set_style("histograms");
138
139     plot.ShowOnScreen();
140
141     plot.plot_x(_plot);
142     plot.savetops(NomeArquivo);
143     cout << "Aperte qualquer tecla para continuar..." << endl;
144     _getch();
145 }
146
147 void CSimuladorIP::Executar()
148 {
149     cout << "\n
150         #####
151         " << endl;
152     cout << "#_
153         _
154         _#" << endl;
155     cout << "#_Projeto_Programacao_Pratica_-_Calculo_Indice_
156         de_produtividade_#" << endl;
157     cout << "#_
158         _#" << endl;
159     cout << "#_Professor:_Andre_Duarte_Bueno_
160         _#" << endl;
161     cout << "#_
162         _#" << endl;
163     cout << "#_Alunos:_Carolina_Bastos_
164         _#" << endl;
165     cout << "#_Douglas_Ribeiro_
166         _#" << endl;
167     cout << "#_
168         _"

```



```

        ######" << endl;
158     cout << "
        #####"
        " << endl << endl;

159
160     cout << "Gostaria de executar o programa? 1- Sim | 0- nao
        " << endl;

161
162     int opt;
163     bool tst=true;
164
165     cin >> opt;
166
167     do
168     if (opt!=1 && opt!=0)
169     {
170     cout << "opcao invalida\n" << endl;
171     cout << "Gostaria de executar o programa? 1- Sim | 0- nao
        " << endl;
172     cin >> opt;
173     }
174     else
175     tst=false;
176     while(tst);
177
178     while (opt==1)
179     {
180         EntradaDados();
181
182     cout << "\n
        #####"
        " << endl;
183     cout << "######
        ######" << endl;
184     cout << "######Qual nome do arquivo de saida de
        dados?#####" << endl;
185     cout << "######
        ######" << endl;
186     cout << "
        #####"
        " << endl << endl;
187

```

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

```

211         _plot.push_back(JOSHIANISIO);
212         _plot.push_back(GIGER);
213         _plot.push_back(DUPUY);
214         _plot.push_back(SHEDID);
215
216
217         out << "#Borisov_Joshi_JoshiVertical_Giger_
                RenardDupuy_Shedid" << endl;
218         out << BORISOV << "_" << JOSHI << "_" <<
                JOSHIANISIO << "_" << GIGER << "_" << DUPUY << "
                _" << SHEDID;
219
220     cout << "\n
                #####
                " << endl;
221     cout << "#_
                _" << endl;
222     cout << "#_Dados_Salvos!_
                _" << endl;
223     cout << "#_
                _" << endl;
224     cout << "
                #####
                " << endl << endl;
225
226
227     cout << "Entre_com_come_do_arquivo_de_imagem:\n" << endl;
228
229     string _nomeArquivo;
230
231     cin.get();
232     getline (cin, _nomeArquivo);
233
234     Plot(_plot, _nomeArquivo);
235
236     out.close();
237
238     cout << "Gostaria_de_executar_o_programa?_1_-_Sim_|
                _0_-_nao" << endl;
239     cin >> opt;
240
241     tst = true;

```

```

242
243     do
244     if (opt!=1 && opt!=0)
245     {
246     cout << "opcao_invalida\n" << endl;
247     cout << "Gostaria_de_executar_o_programa?_1_-_Sim_|_0_-_nao
        " << endl;
248     cin >> opt;
249     }
250     else
251     tst=false;
252     while(tst);
253     }
254
255
256 }

```

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

Listing 6.23: Arquivo de cabeçalho da classe CGnuplot.

```

1 //
    //////////////////////////////////////
2 //
    Classe de Interface em C++ para o programa gnuplot
    .
    //
3 //
    //////////////////////////////////////

4 // Esta interface usa pipes e nao ira funcionar em sistemas que nao
    suportam
5 // o padrao POSIX pipe.
6 // O mesmo foi testado em sistemas Windows (MinGW e Visual C++) e
    Linux(GCC/G++)
7 // Este programa foi originalmente escrito por:
8 // Historico de versoes:
9 // 0. Interface para linguagem C
10 //     por N. Devillard (27/01/03)
11 // 1. Interface para C++: tradução direta da versao em C
12 //     por Rajarshi Guha (07/03/03)
13 // 2. Correcoes para compatibilidade com Win32
14 //     por V. Chyzhdenka (20/05/03)
15 // 3. Novos métodos membros, correcoes para compatibilidade com
    Win32 e Linux

```

```

16//      por M. Burgis (10/03/08)
17// 4. Tradução para Português, documentação - javadoc/doxygen,
18//      e modificações na interface (adição de interface alternativa)
19//      por Bueno.A.D. (30/07/08)
20//      Tarefas:
21//      (v1)
22//      Documentar toda classe
23//      Adicionar novos métodos, criando atributos adicionais se
        necessario.
24//      Adotar padrão C++, isto é, usar sobrecarga nas chamadas.
25//      (v2)
26//      Criar classe herdeira CGnuplot, que inclui somente a nova
        interface.
27//      como é herdeira, o usuário vai poder usar nomes antigos.
28//      Vantagem: preserva classe original, cria nova interface, fica
        a critério do usuário
29//      qual interface utilizar.
30//
        //////////////////////////////////////
31// Requisitos:
32// - O programa gnuplot deve estar instalado (veja http://www.gnuplot.info/download.html)
33// - No Windows: setar a Path do Gnuplot (i.e. C:/program files/
        gnuplot/bin)
34//      ou setar a path usando: Gnuplot::set_GNUPlotPath(
        const std::string &path);
35//      Gnuplot::set_GNUPlotPath("C:/program files/gnuplot/
        bin");
36// - Para um melhor uso, consulte o manual do gnuplot,
37//      no GNU/Linux digite: man gnuplot ou info gnuplot.
38//
39// - Veja aula em http://www.lenep.uenf.br/~bueno/DisciplinaSL/
40//
41//
        //////////////////////////////////////
42
43
44#ifdef CGnuplot_h
45#define CGnuplot_h
46#include <iostream>                // Para teste

```

```

47#include <string>
48#include <vector>
49#include <stdexcept> // Heranca da classe std::
    runtime_error em GnuplotException
50#include <cstdio> // Para acesso a arquivos FILE
51
52/**
53@brief Erros em tempo de execucao
54@class GnuplotException
55@file GnuplotException.h
56*/
57class GnuplotException : public std::runtime_error
58{
59public:
60    /// Construtor
61    GnuplotException (const std::string & msg):std::runtime_error (
        msg) {}
62};
63
64/**
65@brief Classe de interface para acesso ao programa gnuplot.
66@class Gnuplot
67@file gnuplot_i.hpp
68*/
69class Gnuplot
70{
71private:
72    //
    -----
    Atributos
73    FILE *   gnuclmd; //< Ponteiro para stream que escreve no
        pipe.
74    bool     valid; //< Flag que indica se a sessao do gnuplot
        esta valida.
75    bool     two_dim; //< true = verdadeiro = 2d, false = falso
        = 3d.
76    int      nplots; //< Numero de graficos (plots) na sessao.
77    std::string pstyle; //< Estilo utilizado para visualizacao das
        funcoes e dados.
78    std::string smooth; //< interpolate and approximate data in
        defined styles (e.g. spline).
79    std::vector <std::string> tmpfile_list; //< Lista com nome dos

```

```

    arquivos temporarios.

80
81 //
    -----

    flags
82 bool fgrid;           ///< 0 sem grid,          1 com grid
83 bool fhidden3d;       ///< 0 nao oculta,        1 oculta
84 bool fcontour;        ///< 0 sem contorno,      1 com contorno
85 bool fsurface;        ///< 0 sem superficie,  1 com superficie
86 bool flegend;         ///< 0 sem legendad,   1 com legenda
87 bool ftitle;          ///< 0 sem titulo,     1 com titulo
88 bool fxlogscale;      ///< 0 desativa escala log, 1 ativa escala
    log
89 bool fylogscale;      ///< 0 desativa escala log, 1 ativa escala
    log
90 bool fzlogscale;      ///< 0 desativa escala log, 1 ativa escala
    log
91 bool fsmooth;         ///< 0 desativa,       1 ativa
92
93 //
    -----

94 // Atributos estaticos (compartilhados por todos os objetos)
95 static int tmpfile_num;           ///< Numero total de
    arquivos temporarios (numero restrito).
96 static std::string m_sGNUPlotFileName; ///< Nome do arquivo
    executavel do gnuplot.
97 static std::string m_sGNUPlotPath;   ///< Caminho para
    executavel do gnuplot.
98 static std::string terminal_std;     ///< Terminal padrao (
    standart), usado para visualizacoes.
99
100 //
    -----

    Metodos
101 // Funcoes membro (métodos membro) (funcoes auxiliares)
102 /// @brief Cria arquivo temporario e retorna seu nome.
103 /// Usa get_program_path(); e popen();
104 void init ();
105
106 /// @brief Cria arquivo temporario e retorna seu nome.
107 /// Usa get_program_path(); e popen();

```

```

108 void Init() { init(); }
109
110 /// @brief Cria arquivo temporario.
111 std::string create_tmpfile (std::ofstream & tmp);
112
113 /// @brief Cria arquivo temporario.
114 std::string CreateTmpFile (std::ofstream & tmp) { return
    create_tmpfile(tmp); }
115
116 //
    -----

117 // Funcoes estaticas (static functions)
118 /// @brief Retorna verdadeiro se a path esta presente.
119 static bool get_program_path ();
120
121 /// @brief Retorna verdadeiro se a path esta presente.
122 static bool Path() { return get_program_path(); }
123
124 /// @brief Checa se o arquivo existe.
125 static bool file_exists (const std::string & filename, int mode
    = 0);
126
127 /// @brief Checa se o arquivo existe.
128 static bool FileExists (const std::string & filename, int mode
    = 0)
129
    { return file_exists( filename, mode );
    }
130
131 //
    -----

132 public:
133 // Opcional: Seta path do gnuplot manualmente
134 // No windows: a path (caminho) deve ser dada usando '/' e nao
    backslash '\'
135 /// @brief Seta caminho para path do gnuplot.
136 //ex: CGnuplot::set_GNUPlotPath ("\"C:/program files/gnuplot/bin
    /\");
137
138 static bool set_GNUPlotPath (const std::string & path);
139

```



```

140  /// @brief Seta caminho para path do gnuplot.
141  static bool Path(const std::string & path) { return
        set_GNUPlotPath(path); }
142  //
143  /// @brief Opcional: Seta terminal padrao (standart), usado para
        visualizacao dos graficos.
144  /// Valores padroes (default): Windows - win, Linux - x11, Mac -
        aqua
145  static void set_terminal_std (const std::string & type);
146
147  /// @brief Opcional: Seta terminal padrao (standart), usado para
        visualizacao dos graficos.
148  /// Para retornar para terminal janela precisa chamar
        ShowOnScreen().
149  /// Valores padroes (default): Windows - win, Linux - x11 ou wxt
        (fedora9), Mac - aqua
150  static void Terminal (const std::string & type) {
        set_terminal_std(type); }
151
152  //
        -----
        Construtores
153  /// @brief Construtor, seta o estilo do grafico na construcao.
154  Gnuplot (const std::string & style = "points");
155
156  /// @brief Construtor, plota um grafico a partir de um vector,
        diretamente na construcao.
157  Gnuplot (const std::vector < double >&x,
158           const std::string & title = "",
159           const std::string & style = "points",
160           const std::string & labelx = "x",
161           const std::string & labely = "y");
162
163  /// @brief Construtor, plota um grafico do tipo x_y a partir de
        vetores, diretamente na construcao.
164  Gnuplot (const std::vector < double >&x,
165           const std::vector < double >&y,
166           const std::string & title = "",
167           const std::string & style = "points",
168           const std::string & labelx = "x",
169           const std::string & labely = "y");
170

```

```

171  /// @brief Construtor, plota um grafico de x_y_z a partir de
      vetores, diretamente na construcao.
172  Gnuplot (const std::vector < double >&x,
173          const std::vector < double >&y,
174          const std::vector < double >&z,
175          const std::string & title = "",
176          const std::string & style = "points",
177          const std::string & labelx = "x",
178          const std::string & labely = "y",
179          const std::string & labelz = "z");
180
181  /// @brief Destrutor, necessario para deletar arquivos
      temporarios.
182  ~Gnuplot ();
183
184  //
      -----
185  /// @brief Envia comando para o gnuplot.
186  Gnuplot & cmd (const std::string & cmdstr);
187
188  /// @brief Envia comando para o gnuplot.
189  Gnuplot & Cmd (const std::string & cmdstr)      { return cmd(
      cmdstr); }
190
191  /// @brief Envia comando para o gnuplot.
192  Gnuplot & Command (const std::string & cmdstr) { return cmd(
      cmdstr); }
193
194  /// @brief Sobrecarga operador <<, funciona como Comando.
195  Gnuplot & operator<< (const std::string & cmdstr);
196
197  //
      -----
198  /// @brief Mostrar na tela ou escrever no arquivo, seta o tipo de
      terminal para terminal_std.
199  Gnuplot & showonscreen ();          // Janela de saida e setada
      como default (win/x11/aqua)
200
201  /// @brief Mostrar na tela ou escrever no arquivo, seta o tipo de
      terminal para terminal_std.

```

```
202 Gnuplot & ShowOnScreen () { return
    showonscreen(); };
203
204 /// @brief Salva sessao do gnuplot para um arquivo postscript,
    informe o nome do arquivo sem extensao.
205 /// Depois retorna para modo terminal
206 Gnuplot & savetops (const std::string & filename = "
    gnuplot_output");
207
208 /// @brief Salva sessao do gnuplot para um arquivo postscript,
    informe o nome do arquivo sem extensao
209 /// Depois retorna para modo terminal
210 Gnuplot & SaveTops (const std::string & filename = "
    gnuplot_output")
211 { return savetops
    (filename); }
212
213 /// @brief Salva sessao do gnuplot para um arquivo png, nome do
    arquivo sem extensao
214 /// Depois retorna para modo terminal
215 Gnuplot & savetopng (const std::string & filename = "
    gnuplot_output");
216
217 /// @brief Salva sessao do gnuplot para um arquivo png, nome do
    arquivo sem extensao
218 /// Depois retorna para modo terminal
219 Gnuplot & SaveTopng (const std::string & filename = "
    gnuplot_output")
220 { return
    savetopng(
    filename); }
221
222 /// @brief Salva sessao do gnuplot para um arquivo jpg, nome do
    arquivo sem extensao
223 /// Depois retorna para modo terminal
224 Gnuplot & savetjpeg (const std::string & filename = "
    gnuplot_output");
225
226 /// @brief Salva sessao do gnuplot para um arquivo jpg, nome do
    arquivo sem extensao
227 /// Depois retorna para modo terminal
228 Gnuplot & SaveTojpeg (const std::string & filename = "
```

```

        gnuplot_output")
229                                     { return
                                       savetojpeg(
                                       filename); }
230
231  /// @brief Salva sessao do gnuplot para um arquivo filename,
        usando o terminal_type e algum flag adicional
232  /// Ex:
233  /// grafico.SaveTo("pressao_X_temperatura","png", "enhanced size
        1280,960");
234  /// Para melhor uso dos flags adicionais consulte o manual do
        gnuplot (help term)
235  Gnuplot& SaveTo(const std::string &filename, const std::string &
        terminal_type, std::string flags="");
236
237  //
        -----
        set e unset
238  /// @brief Seta estilos de linhas (em alguns casos sao
        necessarias informacoes adicionais).
239  /// lines, points, linespoints, impulses, dots, steps, fsteps,
        histeps,
240  /// boxes, histograms, filledcurves
241  Gnuplot & set_style (const std::string & stylestr = "points");
242
243  /// @brief Seta estilos de linhas (em alguns casos sao
        necessarias informacoes adicionais).
244  /// lines, points, linespoints, impulses, dots, steps, fsteps,
        histeps,
245  /// boxes, histograms, filledcurves
246  Gnuplot & Style (const std::string & stylestr = "points")
247                                     { return set_style
                                       (stylestr); }
248
249  /// @brief Ativa suavizacao.
250  /// Argumentos para interpolacoes e aproximacoes.
251  /// csplines, bezier, acsplines (para dados com valor > 0),
        sbezier, unique,
252  /// frequency (funciona somente com plot_x, plot_xy, plotfile_x,
253  /// plotfile_xy (se a suavizacao esta ativa, set_style nao tem
        efeito na plotagem dos graficos)
254  Gnuplot & set_smooth (const std::string & stylestr = "csplines")

```

```

        ;
255
256 /// @brief Desativa suavizacao.
257 Gnuplot & unset_smooth (); // A suavizacao nao e
        setada por padrao (default)
258
259 /// @brief Ativa suavizacao.
260 /// Argumentos para interpolacoes e aproximacoes.
261 /// csplines, bezier, acsplines (para dados com valor > 0),
        sbezier, unique,
262 /// frequency (funciona somente com plot_x, plot_xy, plotfile_x,
263 /// plotfile_xy (se a suavizacao esta ativa, set_style nao tem
        efeito na plotagem dos graficos)
264 Gnuplot & Smooth(const std::string & stylestr = "csplines")
265                                     { return set_smooth
                                                (stylestr); }
266
267 Gnuplot & Smooth( int _fsmooth )
268                                     { if( fsmooth =
                                                _fsmooth )
269                                             return
                                                set_contour
                                                ();
270                                     else
271                                             return
                                                unset_contour
                                                ();
272                                     }
273 /// @brief Desativa suavizacao.
274 //Gnuplot & UnsetSmooth() { return
        unset_smooth (); }
275
276 /// @brief Escala o tamanho do ponto usado na plotagem.
277 Gnuplot & set_pointsize (const double pointsize = 1.0);
278
279 /// @brief Escala o tamanho do ponto usado na plotagem.
280 Gnuplot & PointSize (const double pointsize = 1.0)
281                                     { return
                                                set_pointsize(
                                                pointsize); }
282
283 /// @brief Ativa o grid (padrao = desativado).

```

```
284 Gnuplot & set_grid ();
285
286 /// @brief Desativa o grid (padrao = desativado).
287 Gnuplot & unset_grid ();
288
289 /// @brief Ativa/Desativa o grid (padrao = desativado).
290 Gnuplot & Grid(bool _fgrid = 1)
291                                     { if(fgrid = _fgrid
292                                     )
293                                     return set_grid
294                                     ();
295                                     else
296                                     return
297                                     unset_grid()
298                                     ; }
299
300 /// @brief Seta taxa de amostragem das funcoes, ou dos dados de
301 interpolacao.
302 Gnuplot & set_samples (const int samples = 100);
303
304 /// @brief Seta taxa de amostragem das funcoes, ou dos dados de
305 interpolacao.
306 Gnuplot & Samples(const int samples = 100) { return
307 set_samples(samples); }
308
309 /// @brief Seta densidade de isolinhas para plotagem de funcoes
310 como superficies (para plotagen 3d).
311 Gnuplot & set_isosamples (const int isolines = 10);
312
313 /// @brief Seta densidade de isolinhas para plotagem de funcoes
314 como superficies (para plotagen 3d).
315 Gnuplot & IsoSamples (const int isolines = 10){ return
316 set_isosamples(isolines); }
317
318 /// @brief Ativa remocao de linhas ocultas na plotagem de
319 superficies (para plotagen 3d).
320 Gnuplot & set_hidden3d ();
321
322 /// @brief Desativa remocao de linhas ocultas na plotagem de
323 superficies (para plotagen 3d).
324 Gnuplot & unset_hidden3d (); // hidden3d nao e setado
325 por padrao (default)
```

```

313
314 /// @brief Ativa/Desativa remocao de linhas ocultas na plotagem
      de superficies (para plotagen 3d).
315 Gnuplot & Hidden3d(bool _fhidden3d = 1)
316                                     { if(fhidden3d =
                                     _fhidden3d)
317                                         return
                                     set_hidden3d
                                     ();
318                                     else
319                                         return
                                     unset_hidden3d
                                     ();
320                                     }
321
322 /// @brief Ativa desenho do contorno em superficies (para
      plotagen 3d).
323 /// @param base, surface, both.
324 Gnuplot & set_contour (const std::string & position = "base");
325
326 /// @brief Desativa desenho do contorno em superficies (para
      plotagen 3d).
327 Gnuplot & unset_contour ();          // contour nao e setado por
      default
328
329 /// @brief Ativa/Desativa desenho do contorno em superficies (
      para plotagen 3d).
330 /// @param base, surface, both.
331 Gnuplot & Contour(const std::string & position = "base")
332                                     { return
                                     set_contour(
                                     position); }
333
334 Gnuplot & Contour( int _fcontour )
335                                     { if( fcontour =
                                     _fcontour )
336                                         return
                                     set_contour
                                     ();
337                                     else
338                                         return
                                     unset_contour

```

```

                                                    ();
339                                     }
340  /// @brief  Ativa a visualizacao da superficie (para plotagen 3d)
341  .
342  Gnuplot &  set_surface ();                // surface e setado por
343  padrao (default)
344
345  /// @brief  Desativa a visualizacao da superficie (para plotagen
346  3d).
347  Gnuplot &  unset_surface ();
348
349  /// @brief  Ativa/Desativa a visualizacao da superficie (para
350  plotagen 3d).
351  Gnuplot &  Surface( int _fsurface = 1 )
352
353                                     { if(fsurface =
354                                     _fsurface)
355                                     return
356                                     set_surface
357                                     ();
358                                     else
359                                     return
360                                     unset_surface
361                                     ();
362                                     }
363
364  /// @brief Ativa a legenda (a legenda é setada por padrao).
365  /// Posicao: inside/outside, left/center/right, top/center/bottom
366  , nobox/box
367  Gnuplot &  set_legend (const std::string & position = "default");
368
369  /// @brief Desativa a legenda (a legenda é setada por padrao).
370  Gnuplot &  unset_legend ();
371
372  /// @brief Ativa/Desativa a legenda (a legenda é setada por
373  padrao).
374  Gnuplot &  Legend(const std::string & position = "default")
375
376                                     { return set_legend
377                                     (position); }
378
379  /// @brief Ativa/Desativa a legenda (a legenda é setada por
380  padrao).
381  Gnuplot &  Legend(int _flegend)

```



```

366         { if(flegend =
              _flegend)
367             return
              set_legend
              ();
368         else
369             return
              unset_legend
              ();
370     }
371
372     /// @brief Ativa o titulo da secao do gnuplot.
373     Gnuplot & set_title (const std::string & title = "");
374
375     /// @brief Desativa o titulo da secao do gnuplot.
376     Gnuplot & unset_title ();           // 0 title nao e setado por
        padrao (default)
377
378     /// @brief Ativa/Desativa o titulo da secao do gnuplot.
379     Gnuplot & Title(const std::string & title = "")
380     {
381         return set_title(
            title);
382     }
383     Gnuplot & Title(int _ftitle)
384     {
385         if(ftitle =
            _ftitle)
386             return set_title
            ();
387         else
388             return
            unset_title()
            ;
389     }
390
391     /// @brief Seta o rotulo (nome) do eixo y.
392     Gnuplot & set_ylabel (const std::string & label = "y");
393
394     /// @brief Seta o rotulo (nome) do eixo y.
395     /// Ex: set ylabel "{/Symbol s}[MPa]" font "Times Italic, 10"
396     Gnuplot & YLabel(const std::string & label = "y")

```

```
397                                     { return set_ylabel
                                     (label); }
398
399 /// @brief Seta o rotulo (nome) do eixo x.
400 Gnuplot & set_xlabel (const std::string & label = "x");
401
402 /// @brief Seta o rotulo (nome) do eixo x.
403 Gnuplot & XLabel(const std::string & label = "x")
404                                     { return set_xlabel
                                     (label); }
405
406 /// @brief Seta o rotulo (nome) do eixo z.
407 Gnuplot & set_zlabel (const std::string & label = "z");
408
409 /// @brief Seta o rotulo (nome) do eixo z.
410 Gnuplot & ZLabel(const std::string & label = "z")
411                                     { return set_zlabel
                                     (label); }
412
413 /// @brief Seta intervalo do eixo x.
414 Gnuplot & set_xrange (const int iFrom, const int iTo);
415
416 /// @brief Seta intervalo do eixo x.
417 Gnuplot & XRange (const int iFrom, const int iTo)
418                                     { return set_xrange
                                     (iFrom,iTo); }
419
420 /// @brief Seta intervalo do eixo y.
421 Gnuplot & set_yrange (const int iFrom, const int iTo);
422
423 /// @brief Seta intervalo do eixo y.
424 Gnuplot & YRange (const int iFrom, const int iTo)
425                                     { return set_yrange
                                     (iFrom,iTo); }
426
427 /// @brief Seta intervalo do eixo z.
428 Gnuplot & set_zrange (const int iFrom, const int iTo);
429
430 /// @brief Seta intervalo do eixo z.
431 Gnuplot & ZRange (const int iFrom, const int iTo)
432                                     { return set_zrange
                                     (iFrom,iTo); }
```

```
433
434 /// @brief Seta escalonamento automatico do eixo x (default).
435 Gnuplot & set_xautoscale ();
436
437 /// @brief Seta escalonamento automatico do eixo x (default).
438 Gnuplot & XAutoscale() { return
    set_xautoscale (); }
439
440 /// @brief Seta escalonamento automatico do eixo y (default).
441 Gnuplot & set_yautoscale ();
442
443 /// @brief Seta escalonamento automatico do eixo y (default).
444 Gnuplot & YAutoscale() { return
    set_yautoscale (); }
445
446 /// @brief Seta escalonamento automatico do eixo z (default).
447 Gnuplot & set_zautoscale ();
448
449 /// @brief Seta escalonamento automatico do eixo z (default).
450 Gnuplot & ZAutoscale() { return
    set_zautoscale (); }
451
452 /// @brief Ativa escala logaritma do eixo x (logscale nao e
    setado por default).
453 Gnuplot & set_xlogscale (const double base = 10);
454
455 /// @brief Desativa escala logaritma do eixo x (logscale nao e
    setado por default).
456 Gnuplot & unset_xlogscale ();
457
458 /// @brief Ativa escala logaritma do eixo x (logscale nao e
    setado por default).
459 Gnuplot & XLogscale (const double base = 10) { //if(base)
460
    return
        set_xlogscale
            (base);
461
    //else
462
    //return
        unset_xlogscale
            ();
463
    }
464
```

```
465  /// @brief Ativa/Desativa escala logaritma do eixo x (logscale
    nao e setado por default).
466  Gnuplot & XLogscale(bool _fxlogscale)
467
468                                     { if(fxlogscale =
                                         _fxlogscale)
                                         return
                                         set_xlogscale
                                         ();
469                                     else
470                                     return
                                         unset_xlogscale
                                         ();
471                                     }
472
473  /// @brief Ativa escala logaritma do eixo y (logscale nao e
    setado por default).
474  Gnuplot & set_ylogscale (const double base = 10);
475
476  /// @brief Ativa escala logaritma do eixo y (logscale nao e
    setado por default).
477  Gnuplot & YLogscale (const double base = 10) { return
    set_ylogscale (base); }
478
479  /// @brief Desativa escala logaritma do eixo y (logscale nao e
    setado por default).
480  Gnuplot & unset_ylogscale ();
481
482  /// @brief Ativa/Desativa escala logaritma do eixo y (logscale
    nao e setado por default).
483  Gnuplot & YLogscale(bool _fylogscale)
484
485                                     { if(fylogscale =
                                         _fylogscale)
                                         return
                                         set_ylogscale
                                         ();
486                                     else
487                                     return
                                         unset_ylogscale
                                         ();
488                                     }
489
490  /// @brief Ativa escala logaritma do eixo y (logscale nao e
```

```

        setado por default).
491 Gnuplot & set_zlogscale (const double base = 10);
492
493 /// @brief Ativa escala logaritma do eixo y (logscale nao e
        setado por default).
494 Gnuplot & ZLogscale (const double base = 10) { return
        set_zlogscale (base); }
495
496 /// @brief Desativa escala logaritma do eixo z (logscale nao e
        setado por default).
497 Gnuplot & unset_zlogscale ();
498
499 /// @brief Ativa/Desativa escala logaritma do eixo y (logscale
        nao e setado por default).
500 Gnuplot & ZLogscale(bool _fzlogscale)
501                                     { if(fzlogscale =
                                         _fzlogscale)
502                                         return
                                         set_zlogscale
                                         ();
503                                         else
504                                         return
                                         unset_zlogscale
                                         ();
505                                     }
506
507
508 /// @brief Seta intervalo da palette (autoscale por padrao).
509 Gnuplot & set_cbrange (const int iFrom, const int iTo);
510
511 /// @brief Seta intervalo da palette (autoscale por padrao).
512 Gnuplot & CBRange(const int iFrom, const int iTo)
513                                     { return
                                         set_cbrange(
                                         iFrom, iTo); }
514
515 //
    -----
516 /// @brief Plota dados de um arquivo de disco.
517 Gnuplot & plotfile_x (const std::string & filename,
518                      const int column = 1, const std::string & title = "")

```

```

;

519
520 /// @brief Plota dados de um arquivo de disco.
521 Gnuplot & PlotFile (const std::string & filename,
522                     const int column = 1, const std::string & title = "")
523                     { return plotfile_x
                      (filename,
                      column, title);
                      }

524
525 /// @brief Plota dados de um vector.
526 Gnuplot & plot_x (const std::vector < double >&x, const std::
                    string & title = "");

527
528 /// @brief Plota dados de um vector.
529 Gnuplot & PlotVector (const std::vector < double >&x, const std
                    ::string & title = "")

530                     { return plot_x( x,
                    title ); }

531
532 /// @brief Plota pares x,y a partir de um arquivo de disco.
533 Gnuplot & plotfile_xy (const std::string & filename,
534                       const int column_x = 1,
535                       const int column_y = 2, const std::string & title =
                    "");

536 /// @brief Plota pares x,y a partir de um arquivo de disco.
537 Gnuplot & PlotFile (const std::string & filename,
538                   const int column_x = 1,
539                   const int column_y = 2, const std::string & title =
                    "")

540                   {
541                   return plotfile_xy(
                    filename,
                    column_x,
                    column_y, title
                    );
                    }

542
543
544 /// @brief Plota pares x,y a partir de vetores.
545 Gnuplot & plot_xy (const std::vector < double >&x,
546                  const std::vector < double >&y, const std::string &
                    title = "");

```

```

547
548 /// @brief Plota pares x,y a partir de vetores.
549 Gnuplot & PlotVector (const std::vector < double >&x,
550                      const std::vector < double >&y, const std::string &
551                      title = "")
552
553                                     { return plot_xy (
554                                     x, y,title ); }
555
556
557 /// @brief Plota pares x,y com barra de erro dy a partir de um
558 arquivo.
559 Gnuplot & plotfile_xy_err (const std::string & filename,
560                           const int column_x = 1,
561                           const int column_y = 2,
562                           const int column_dy = 3, const std::string &
563                           title = "");
564
565
566 /// @brief Plota pares x,y com barra de erro dy a partir de um
567 arquivo.
568 Gnuplot & PlotFileXYErrorBar(const std::string & filename,
569                             const int column_x = 1,
570                             const int column_y = 2,
571                             const int column_dy = 3, const std::string &
572                             title = "")
573
574                                     { return
575                                     plotfile_xy_err
576                                     (filename,
577                                     column_x,
578                                     column_y,
579                                     column_dy,
580                                     title ); }
581
582
583 /// @brief Plota pares x,y com barra de erro dy a partir de
584 vetores.
585 Gnuplot & plot_xy_err (const std::vector < double >&x,
586                      const std::vector < double >&y,
587                      const std::vector < double >&dy,
588                      const std::string & title = "");
589
590
591 /// @brief Plota pares x,y com barra de erro dy a partir de
592 vetores.
593 Gnuplot & PlotVectorXYErrorBar(const std::vector < double >&x,
594                               const std::vector < double >&y,

```

```

576         const std::vector < double >&dy,
577         const std::string & title = "")
578                                     { return
                                     plot_xy_err(
                                     x, y, dy,
                                     title); }

579
580 /// @brief Plota valores de x,y,z a partir de um arquivo de
        disco.
581 Gnuplot & plotfile_xyz (const std::string & filename,
582                         const int column_x = 1,
583                         const int column_y = 2,
584                         const int column_z = 3, const std::string & title =
                            "");
585 /// @brief Plota valores de x,y,z a partir de um arquivo de
        disco.
586 Gnuplot & PlotFile (const std::string & filename,
587                   const int column_x = 1,
588                   const int column_y = 2,
589                   const int column_z = 3, const std::string & title =
                            "")
590                                     { return
                                     plotfile_xyz
                                     (filename,
                                     column_x,
591                                     column_y,
                                     column_z);
                                     }

592
593 /// @brief Plota valores de x,y,z a partir de vetores.
594 Gnuplot & plot_xyz (const std::vector < double >&x,
595                   const std::vector < double >&y,
596                   const std::vector < double >&z, const std::string &
                            title = "");
597
598 /// @brief Plota valores de x,y,z a partir de vetores.
599 Gnuplot & PlotVector(const std::vector < double >&x,
600                   const std::vector < double >&y,
601                   const std::vector < double >&z, const std::string &
                            title = "")
602                                     { return
                                     plot_xyz(x,

```



```

y, z, title)
; }

603
604 /// @brief Plota uma equacao da forma y = ax + b, voce fornece os
      coeficientes a e b.
605 Gnuplot & plot_slope (const double a, const double b, const std
      ::string & title = "");
606
607 /// @brief Plota uma equacao da forma y = ax + b, voce fornece os
      coeficientes a e b.
608 Gnuplot & PlotSlope (const double a, const double b, const std
      ::string & title = "")
609
      { return
        plot_slope(a
          ,b,title); }
610
611 /// @brief Plota uma equacao fornecida como uma std::string y=f(
      x).
612 /// Escrever somente a funcao f(x) e nao y=
613 /// A variavel independente deve ser x
614 /// Os operadores binarios aceitos sao:
615 /// ** exponenciacao,
616 /// * multiplicacao,
617 /// / divisao,
618 /// + adicao,
619 /// - subtracao,
620 /// % modulo
621 /// Os operadores unarios aceitos sao:
622 /// - menos,
623 /// ! fatorial
624 /// Funcoes elementares:
625 /// rand(x), abs(x), sgn(x), ceil(x), floor(x), int(x), imag(x),
      real(x), arg(x),
626 /// sqrt(x), exp(x), log(x), log10(x), sin(x), cos(x), tan(x),
      asin(x), acos(x),
627 /// atan(x), atan2(y,x), sinh(x), cosh(x), tanh(x), asinh(x),
      acosh(x), atanh(x)
628 /// Funcoes especiais:
629 /// erf(x), erfc(x), inverf(x), gamma(x), igamma(a,x), lgamma(x),
      ibeta(p,q,x),
630 /// besj0(x), besj1(x), besy0(x), besy1(x), lambertw(x)
631 /// Funcoes estatisticas:

```

```

632  /// norm(x), invnorm(x)
633  Gnuplot & plot_equation (const std::string & equation,
634                          const std::string & title = "");
635
636  /// @brief Plota uma equacao fornecida como uma std::string y=f(
        x).
637  /// Escrever somente a funcao f(x) e nao y=
638  /// A variavel independente deve ser x.
639  /// Exemplo: gnuplot->PlotEquation(CFuncao& obj);
640  // Deve receber um CFuncao, que tem cast para string.
641  Gnuplot & PlotEquation(const std::string & equation,
642                        const std::string & title = "")
643  { return
        plot_equation(
            equation, title );
644
645  /// @brief Plota uma equacao fornecida na forma de uma std::
        string z=f(x,y).
646  /// Escrever somente a funcao f(x,y) e nao z=, as variaveis
        independentes sao x e y.
647  Gnuplot & plot_equation3d (const std::string & equation, const
        std::string & title = "");
648
649  /// @brief Plota uma equacao fornecida na forma de uma std::
        string z=f(x,y).
650  /// Escrever somente a funcao f(x,y) e nao z=, as vaiaveis
        independentes sao x e y.
651  // gnuplot->PlotEquation3d(CPolinomio());
652  Gnuplot & PlotEquation3d (const std::string & equation,
653                          const std::string & title = "")
654  { return
        plot_equation3d(
            equation, title );
655
656  /// @brief Plota uma imagem.
657  Gnuplot & plot_image (const unsigned char *ucPicBuf,
658                      const int iWidth, const int iHeight, const std::
        string & title = "");
659
660  /// @brief Plota uma imagem.

```

```

661 Gnuplot & PlotImage (const unsigned char *ucPicBuf,
662                      const int iWidth, const int iHeight, const
663                      std::string & title = "")
664
665                      { return plot_image (
666                        ucPicBuf, iWidth,
667                        iHeight, title); }
668
669 //
670
671 -----
672
673 // Repete o ultimo comando de plotagem, seja plot (2D) ou splot
674 // (3D)
675 // Usado para visualizar plotagens, após mudar algumas opcoes de
676 // plotagem
677 // ou quando gerando o mesmo grafico para diferentes dispositivos
678 // (showonscreen, savetops)
679 Gnuplot & replot ();
680
681 // Repete o ultimo comando de plotagem, seja plot (2D) ou splot
682 // (3D)
683 // Usado para visualizar plotagens, após mudar algumas opcoes de
684 // plotagem
685 // ou quando gerando o mesmo grafico para diferentes dispositivos
686 // (showonscreen, savetops)
687 Gnuplot & Replot() { return replot();
688   }
689
690 // Reseta uma sessao do gnuplot (próxima plotagem apaga
691 // definicoes previas)
692 Gnuplot & reset_plot ();
693
694 // Reseta uma sessao do gnuplot (próxima plotagem apaga
695 // definicoes previas)
696 Gnuplot & ResetPlot() { return reset_plot
697   (); }
698
699 // Chama função reset do gnuplot
700 Gnuplot & Reset() { this->cmd("reset");
701   return *this; }
702
703 // Reseta uma sessao do gnuplot e seta todas as variaveis para o
704 // default

```

```

686 Gnuplot & reset_all ();
687
688 // Reseta uma sessao do gnuplot e seta todas as variaveis para o
    default
689 Gnuplot & ResetAll ()                                { return reset_all
    (); }
690
691 // Verifica se a sessao esta valida
692 bool is_valid ();
693
694 // Verifica se a sessao esta valida
695 bool IsValid ()                                    { return is_valid
    (); };
696
697 };
698 typedef Gnuplot CGnuplot;
699 #endif

```

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

Listing 6.24: Arquivo de implementação da classe CGnuplot.

```

1 //////////////////////////////////////
2 //
3 // A C++ interface to gnuplot.
4 //
5 // This is a direct translation from the C interface
6 // written by Nicolas Devillard (ndevilla@free.fr) which
7 // is available from http://ndevilla.free.fr/gnuplot/.
8 //
9 // As in the C interface this uses pipes and so wont
10 // run on a system that doesn't have POSIX pipe support
11 //
12 // Rajarshi Guha
13 // e-mail: rguha@indiana.edu, rajarshi@presidency.com
14 // http://cheminfo.informatics.indiana.edu/~rguha/code/cc++/
15 //
16 // 07/03/03
17 //
18 //////////////////////////////////////
19 //
20 // A little correction for Win32 compatibility
21 // and MS VC 6.0 done by V.Chyhdzenka
22 //

```

```

23// Notes:
24// 1. Added private method Gnuplot::init().
25// 2. Temporary file is created in the current
26//     folder but not in /tmp.
27// 3. Added #ifdef WIN32 e.t.c. where is needed.
28// 4. Added private member m_sGnUPlotFileName is
29//     a name of executed GnUPlot file.
30//
31// Viktor Chyzhdzenka
32// e-mail: chyzhdzenka@mail.ru
33//
34// 20/05/03
35//
36////////////////////////////////////
37//
38// corrections for Win32 and Linux compatibility
39//
40// some member functions added:
41//   set_GnUPlotPath, set_terminal_std,
42//   create_tmpfile, get_program_path, file_exists,
43//   operator<<, replot, reset_all, savetops, showonscreen,
44//   plotfile_*, plot_xy_err, plot_equation3d
45// set, unset: pointsize, grid, *logscale, *autoscale,
46// smooth, title, legend, samples, isosamples,
47// hidden3d, cbrange, contour
48//
49// Markus Burgis
50// e-mail: mail@burgis.info
51//
52// 10/03/08
53//
54////////////////////////////////////
55//
56// Modificacoes:
57// Traducao para o portugues
58// Adicao de novos nomes para os metodos(funcoes)
59// Uso de documentacao no formato javadoc/doxygen
60// Bueno A.D.
61// e-mail: bueno@lenep.uenf.br
62// 20/07/08
63//
64////////////////////////////////////

```

```

65#include <fstream>                // for std::ifstream
66#include <sstream>                // for std::ostringstream
67#include <list>                   // for std::list
68#include <cstdio>                 // for FILE, fputs(), fflush(),
    popen()
69#include <cstdlib>                // for getenv()
70#include "CGnuplot.h"
71
72// Se estamos no windows // defined for 32 and 64-bit environments
73#if defined(WIN32) || defined(_WIN32) || defined(__WIN32__) ||
    defined(__TOS_WIN__)
74#include <io.h>                   // for _access(), _mktemp()
75#define GP_MAX_TMP_FILES 27      // 27 temporary files it's
    Microsoft restriction
76// Se estamos no unix, GNU/Linux, Mac Os X
77#elif defined(unix) || defined(__unix) || defined(__unix__) ||
    defined(__APPLE__) //all UNIX-like OSs (Linux, *BSD, MacOSX,
    Solaris, ...)
78#include <unistd.h>               // for access(), mkstemp()
79#define GP_MAX_TMP_FILES 64
80#else
81#error unsupported or unknown operating system
82#endif
83
84//
    -----

85//
86// initialize static data
87//
88int Gnuplot::tmpfile_num = 0;
89
90// Se estamos no windows
91#if defined(WIN32) || defined(_WIN32) || defined(__WIN32__) ||
    defined(__TOS_WIN__)
92std::string Gnuplot::m_sGnuplotFileName = "gnuplot.exe";
93std::string Gnuplot::m_sGnuplotPath = "C:/gnuplot/bin/";
94// Se estamos no unix, GNU/Linux, Mac Os X
95#elif defined(unix) || defined(__unix) || defined(__unix__) ||
    defined(__APPLE__)
96std::string Gnuplot::m_sGnuplotFileName = "gnuplot";
97std::string Gnuplot::m_sGnuplotPath = "/usr/bin/";

```

```

98 #endif
99
100 // Se estamos no windows
101 #if defined(WIN32) || defined(_WIN32) || defined(__WIN32__) ||
    defined(__TOS_WIN__)
102 std::string Gnuplot::terminal_std = "windows";
103 // Se estamos no unix, GNU/Linux
104 #elif ( defined(unix) || defined(__unix) || defined(__unix__) ) &&
    !defined(__APPLE__)
105 std::string Gnuplot::terminal_std = "x11";
106 // Se estamos Mac Os X
107 #elif defined(__APPLE__)
108 std::string Gnuplot::terminal_std = "aqua";
109 #endif
110
111 //
    -----
112 //
113 // define static member function: set Gnuplot path manual
114 //   for windows: path with slash '/' not backslash '\'
115 //
116 bool Gnuplot::set_GNUPlotPath(const std::string &path)
117 {
118     std::string tmp = path + "/" + Gnuplot::m_sGNUPlotFileName;
119     #if defined(WIN32) || defined(_WIN32) || defined(__WIN32__) ||
        defined(__TOS_WIN__)
120         if ( Gnuplot::file_exists(tmp,0) ) // check existence
121     #elif defined(unix) || defined(__unix) || defined(__unix__) ||
        defined(__APPLE__)
122         if ( Gnuplot::file_exists(tmp,1) ) // check existence and
            execution permission
123 #endif
124     {
125         Gnuplot::m_sGNUPlotPath = path;
126         return true;
127     }
128     else
129     {
130         Gnuplot::m_sGNUPlotPath.clear();
131         return false;
132     }

```

```

133}
134
135//
-----

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

154// A string tokenizer taken from http://www.sunsite.ualberta.ca/
    Documentation/
155// /Gnu/libstdc++-2.90.8/html/21_strings/stringtok_std_h.txt
156template <typename Container>
157void stringtok (Container &container,
158                std::string const &in,
159                const char * const delimiters = "\t\n")
160{
161    const std::string::size_type len = in.length();
162    std::string::size_type i = 0;
163
164    while ( i < len )
165    {
166        // eat leading whitespace

```



```

167         i = in.find_first_not_of (delimiters, i);
168
169         if (i == std::string::npos)
170             return;    // nothing left but white space
171
172         // find the end of the token
173         std::string::size_type j = in.find_first_of (delimiters, i)
174             ;
175
176         // push token
177         if (j == std::string::npos)
178         {
179             container.push_back (in.substr(i));
180             return;
181         }
182         else
183             container.push_back (in.substr(i, j-i));
184
185         // set up for next loop
186         i = j + 1;
187     }
188
189     return;
190 }
191 //
192
193 // constructor: set a style during construction
194 //
195 Gnuplot::Gnuplot(const std::string &style)
196 {
197     this->init();
198     this->set_style(style);
199 }
200
201 //
202
203 // constructor: open a new session, plot a signal (x)
204 Gnuplot::Gnuplot(const std::vector<double> &x,

```

```
204         const std::string &title,
205         const std::string &style,
206         const std::string &labelx,
207         const std::string &labely)
208 {
209     this->init();
210
211     this->set_style(style);
212     this->set_xlabel(labelx);
213     this->set_ylabel(labely);
214
215     this->plot_x(x,title);
216 }
217
218 //
-----
219 // constructor: open a new session, plot a signal (x,y)
220 Gnuplot::Gnuplot(const std::vector<double> &x,
221                 const std::vector<double> &y,
222                 const std::string &title,
223                 const std::string &style,
224                 const std::string &labelx,
225                 const std::string &labely)
226 {
227     this->init();
228
229     this->set_style(style);
230     this->set_xlabel(labelx);
231     this->set_ylabel(labely);
232
233     this->plot_xy(x,y,title);
234 }
235
236 //
-----
237 // constructor: open a new session, plot a signal (x,y,z)
238 Gnuplot::Gnuplot(const std::vector<double> &x,
239                 const std::vector<double> &y,
240                 const std::vector<double> &z,
241                 const std::string &title,
```

```

242         const std::string &style,
243         const std::string &labelx,
244         const std::string &labely,
245         const std::string &labelz)
246 {
247     this->init();
248
249     this->set_style(style);
250     this->set_xlabel(labelx);
251     this->set_ylabel(labely);
252     this->set_zlabel(labelz);
253
254     this->plot_xyz(x,y,z,title);
255 }
256
257 //
-----
258 // Destructor: needed to delete temporary files
259 Gnuplot::~Gnuplot()
260 {
261     if ((this->tmpfile_list).size() > 0)
262     {
263         for (unsigned int i = 0; i < this->tmpfile_list.size(); i
                ++)
264             remove( this->tmpfile_list[i].c_str() );
265
266         Gnuplot::tmpfile_num -= this->tmpfile_list.size();
267     }
268
269     // A stream opened by popen() should be closed by pclose()
270 #if defined(WIN32) || defined(_WIN32) || defined(__WIN32__) ||
    defined(__TOS_WIN__)
271     if (_pclose(this->gnucmd) == -1)
272 #elif defined(unix) || defined(__unix) || defined(__unix__) ||
    defined(__APPLE__)
273     if (pclose(this->gnucmd) == -1)
274 #endif
275         true; //throw GnuplotException("Problem closing
                communication to gnuplot");
276 }
277

```

```

278 //
-----

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

297 // resets a gnuplot session and sets all variables to default
298 Gnuplot& Gnuplot::reset_all()
299 {
300     if (this->tmpfile_list.size() > 0)
301     {
302         for (unsigned int i = 0; i < this->tmpfile_list.size(); i
            ++))
303             remove(this->tmpfile_list[i].c_str());
304
305         Gnuplot::tmpfile_num -= this->tmpfile_list.size();
306         this->tmpfile_list.clear();
307     }
308
309     this->nplots = 0;
310     this->cmd("reset");
311     this->cmd("clear");
312     this->pstyle = "points";
313     this->smooth = "";

```

```

314     this->showonscreen();
315
316     return *this;
317 }
318
319 //
-----

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

327 // replot repeats the last plot or splot command
328 Gnuplot& Gnuplot::replot()
329 {
330     if (this->nplots > 0)
331     {
332         this->cmd("replot");
333     }
334
335     return *this;
336 }
337
338
339 //
-----

340 // Change the plotting style of a gnuplot session
341 Gnuplot& Gnuplot::set_style(const std::string &stylestr)
342 {
343     if (stylestr.find("lines")           == std::string::npos &&
344         stylestr.find("points")          == std::string::npos &&
345         stylestr.find("linespoints")     == std::string::npos &&
346         stylestr.find("impulses")        == std::string::npos &&
347         stylestr.find("dots")            == std::string::npos &&
348         stylestr.find("steps")           == std::string::npos &&
349         stylestr.find("fsteps")          == std::string::npos &&

```

```

350     stylestr.find("histeps")           == std::string::npos  &&
351     stylestr.find("boxes")             == std::string::npos  &&
        // 1-4 columns of data are required
352     stylestr.find("filledcurves")      == std::string::npos  &&
353     stylestr.find("histograms")        == std::string::npos  )
        //only for one data column
354 //     stylestr.find("labels")          == std::string::npos  &&
        // 3 columns of data are required
355 //     stylestr.find("xerrorbars")      == std::string::npos  &&
        // 3-4 columns of data are required
356 //     stylestr.find("xerrorlines")    == std::string::npos  &&
        // 3-4 columns of data are required
357 //     stylestr.find("errorbars")      == std::string::npos  &&
        // 3-4 columns of data are required
358 //     stylestr.find("errorlines")    == std::string::npos  &&
        // 3-4 columns of data are required
359 //     stylestr.find("yerrorbars")     == std::string::npos  &&
        // 3-4 columns of data are required
360 //     stylestr.find("yerrorlines")    == std::string::npos  &&
        // 3-4 columns of data are required
361 //     stylestr.find("boxerrorbars")   == std::string::npos  &&
        // 3-5 columns of data are required
362 //     stylestr.find("xyerrorbars")    == std::string::npos  &&
        // 4,6,7 columns of data are required
363 //     stylestr.find("xyerrorlines")   == std::string::npos  &&
        // 4,6,7 columns of data are required
364 //     stylestr.find("boxxyerrorbars") == std::string::npos  &&
        // 4,6,7 columns of data are required
365 //     stylestr.find("financebars")    == std::string::npos  &&
        // 5 columns of data are required
366 //     stylestr.find("candlesticks")   == std::string::npos  &&
        // 5 columns of data are required
367 //     stylestr.find("vectors")        == std::string::npos  &&
368 //     stylestr.find("image")          == std::string::npos  &&
369 //     stylestr.find("rgbimage")       == std::string::npos  &&
370 //     stylestr.find("pm3d")           == std::string::npos  )
371 {
372     this->pstyle = std::string("points");
373 }
374 else
375 {
376     this->pstyle = stylestr;

```

```
377     }
378
379     return *this;
380 }
381
382 //
-----
383 // smooth: interpolation and approximation of data
384 Gnuplot& Gnuplot::set_smooth(const std::string &stylestr)
385 {
386     if (stylestr.find("unique")      == std::string::npos    &&
387         stylestr.find("frequency")  == std::string::npos    &&
388         stylestr.find("csplines")   == std::string::npos    &&
389         stylestr.find("acsplines")  == std::string::npos    &&
390         stylestr.find("bezier")     == std::string::npos    &&
391         stylestr.find("sbezier")    == std::string::npos    )
392     {
393         this->smooth = "";
394     }
395     else
396     {
397         this->smooth = stylestr;
398     }
399
400     return *this;
401 }
402
403 //
-----
404 // unset smooth
405 Gnuplot& Gnuplot::unset_smooth()
406 {
407     this->smooth = "";
408
409     return *this;
410 }
411
412 //
```

```

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

423 // saves a gnuplot session to a postscript file
424 Gnuplot& Gnuplot::savetops(const std::string &filename)
425 {
426 //     this->cmd("set terminal postscript color");           // Tipo
de terminal (tipo de arquivo)
427 //
428 //     std::ostringstream cmdstr;                         // Muda
o nome do arquivo
429 //     cmdstr << "set output \"" << filename << ".ps\"";   // Nome
do arquivo
430 //     this->cmd(cmdstr.str());
431 //     this->replot();                                     //
Replota o gráfico, agora salvando o arquivo ps
432 //
433 //     ShowOnScreen ();                                   // Volta
para terminal modo janela
434
435     this->cmd("set_term_png_size_1800,1200");
436
437     std::ostringstream cmdstr;
438     cmdstr << "set_output_\"./Src/Imagens/" << filename << ".png\""
;
439     this->cmd(cmdstr.str());
440     this->Replot();
441
442     return *this;
443 }
444 //
-----

```



```

445 // saves a gnuplot session to a png file and return do on screen
      terminal
446 Gnuplot& Gnuplot::savetopng(const std::string &filename)
447 {
448 //                                     // Muda
      o terminal
449 //      this->cmd("set term png enhanced size 1280,960"); // Tipo
      de terminal (tipo de arquivo)
450 //
451 //      std::ostringstream cmdstr; // Muda
      o nome do arquivo
452 //      cmdstr << "set output \"" << filename << ".png\""; // Nome
      do arquivo
453 //      this->cmd(cmdstr.str());
454 //      this->replot(); //
      Replota o gráfico, agora salvando o arquivo ps
455 //                                     //
      Retorna o terminal para o padrão janela
456 //      ShowOnScreen (); // Volta
      para terminal modo janela
457 //      this->replot(); //
      Replota o gráfico, agora na tela
458      SaveTo(filename,"png", "enhanced_size_1280,960");
459
460      return *this;
461 }
462
463 //
      -----
464 // saves a gnuplot session to a jpeg file and return do on screen
      terminal
465 Gnuplot& Gnuplot::savetojpeg(const std::string &filename)
466 {
467                                     // Muda o
                                     terminal
468 //      this->cmd("set term jpeg enhanced size 1280,960"); // Tipo
      de terminal (tipo de arquivo)
469 //
470 //      std::ostringstream cmdstr; // Muda
      o nome do arquivo
471 //      cmdstr << "set output \"" << filename << ".jpeg\""; // Nome

```

```

do arquivo
472//      this->cmd(cmdstr.str());
473//      this->replot();                                     //
      Replota o gráfico, agora salvando o arquivo ps
474//                                     //
      Retorna o terminal para o padrão janela
475//      ShowOnScreen ();                                     // Volta
      para terminal modo janela
476//      this->replot();                                     //
      Replota o gráfico, agora na tela
477      SaveTo(filename,"jpeg", "enhanced_size_1280,960");
478
479      return *this;
480}
481
482
483//
-----

484// saves a gnuplot session to specific terminal and output file
485// @filename: name of disc file
486// @terminal_type: type of terminal
487// @flags: additional information specific to terminal type
488// Ex:
489// grafico.SaveTo("pressao_X_temperatura","png", "enhanced size
      1280,960");
490// grafico.TerminalType("png").SaveFile(pressao_X_temperatura);
      pense nisso?
491Gnuplot& Gnuplot::SaveTo(const std::string &filename,const std::
      string &terminal_type, std::string flags)
492{                                     // Muda o
      terminal
493      this->cmd("set term " + terminal_type + " " + flags);    //
      Tipo de terminal (tipo de arquivo) e flags adicionais
494      std::ostream cmdstr;                                     // Muda o
      nome do arquivo
495      cmdstr << "set output \" " << filename << "." << terminal_type
      << "\"";
496      this->cmd(cmdstr.str());
497      this->replot();                                     // Replota
      o gráfico, agora salvando o arquivo ps
498                                     // Retorna

```

```

o
terminal
para o
padrão
janela

499 ShowOnScreen ();
500 this->replot(); // Replota
    o gráfico, agora na tela
501
502 return *this;
503 }
504
505 //
-----

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

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

527 // Turns grid on
528 Gnuplot& Gnuplot::set_grid()

```

```

529 {
530     this->cmd("set_grid");
531
532     return *this;
533 }
534
535 //
-----

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

545 // turns on log scaling for the x axis
546 Gnuplot& Gnuplot::set_xlogscale(const double base)
547 {
548     std::ostringstream cmdstr;
549
550     cmdstr << "set_logscale_x" << base;
551     this->cmd(cmdstr.str());
552
553     return *this;
554 }
555
556 //
-----

557 // turns on log scaling for the y axis
558 Gnuplot& Gnuplot::set_ylogscale(const double base)
559 {
560     std::ostringstream cmdstr;
561
562     cmdstr << "set_logscale_y" << base;
563     this->cmd(cmdstr.str());
564

```

```
565     return *this;
566 }
567
568 //
-----

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

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

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

597 // turns off log scaling for the z axis
598 Gnuplot& Gnuplot::unset_zlogscale()
```

```
599 {
600     this->cmd("unset logscale z");
601     return *this;
602 }
603
604
605 //
-----

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

617 // set isoline density (grid) for plotting functions as surfaces
618 Gnuplot& Gnuplot::set_samples(const int samples)
619 {
620     std::ostringstream cmdstr;
621     cmdstr << "set samples" << samples;
622     this->cmd(cmdstr.str());
623
624     return *this;
625 }
626
627 //
-----

628 // set isoline density (grid) for plotting functions as surfaces
629 Gnuplot& Gnuplot::set_isosamples(const int isolines)
630 {
631     std::ostringstream cmdstr;
632     cmdstr << "set isosamples" << isolines;
633     this->cmd(cmdstr.str());
634
```

```

635     return *this;
636 }
637
638 //
-----

639 // enables hidden line removal for surface plotting
640 Gnuplot& Gnuplot::set_hidden3d()
641 {
642     this->cmd("set_□hidden3d");
643
644     return *this;
645 }
646
647 //
-----

648 // disables hidden line removal for surface plotting
649 Gnuplot& Gnuplot::unset_hidden3d()
650 {
651     this->cmd("unset_□hidden3d");
652
653     return *this;
654 }
655
656 //
-----

657 // enables contour drawing for surfaces set contour {base | surface
    | both}
658 Gnuplot& Gnuplot::set_contour(const std::string &position)
659 {
660     if (position.find("base")      == std::string::npos  &&
661         position.find("surface")    == std::string::npos  &&
662         position.find("both")       == std::string::npos  )
663     {
664         this->cmd("set_□contour_□base");
665     }
666     else
667     {
668         this->cmd("set_□contour_□" + position);
669     }

```

```
670
671     return *this;
672 }
673
674 //
-----

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

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

693 // disables the display of surfaces (for 3d plot)
694 Gnuplot& Gnuplot::unset_surface()
695 {
696     this->cmd("unset surface");
697
698     return *this;
699 }
700
701 //
-----

702 // Sets the title of a gnuplot session
703 Gnuplot& Gnuplot::set_title(const std::string &title)
```



```
704{
705    std::ostringstream cmdstr;
706
707    cmdstr << "set_title\" << title << "\"";
708    this->cmd(cmdstr.str());
709
710    return *this;
711}
712
713//
-----

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

723// set labels
724// set the xlabel
725Gnuplot& Gnuplot::set_xlabel(const std::string &label)
726{
727    std::ostringstream cmdstr;
728
729    cmdstr << "set_xlabel\" << label << "\"";
730    this->cmd(cmdstr.str());
731
732    return *this;
733}
734
735//
-----

736// set the ylabel
737Gnuplot& Gnuplot::set_ylabel(const std::string &label)
738{
739    std::ostringstream cmdstr;
```

```

740
741     cmdstr << "set_ylabel\" << label << "\"";
742     this->cmd(cmdstr.str());
743
744     return *this;
745 }
746
747 //
-----

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

760 // set range
761 // set the xrange
762 Gnuplot& Gnuplot::set_xrange(const int iFrom,
763                             const int iTo)
764 {
765     std::ostringstream cmdstr;
766
767     cmdstr << "set_xrange[" << iFrom << ":" << iTo << "]";
768     this->cmd(cmdstr.str());
769
770     return *this;
771 }
772
773 //
-----

774 // set autoscale x
775 Gnuplot& Gnuplot::set_xautoscale()

```

```
776 {
777     this->cmd("set_xrange_restore");
778     this->cmd("set_autoscale_x");
779
780     return *this;
781 }
782
783 //
-----
784 // set the yrange
785 Gnuplot& Gnuplot::set_yrange(const int iFrom, const int iTo)
786 {
787     std::ostringstream cmdstr;
788
789     cmdstr << "set_yrange[" << iFrom << ":" << iTo << "]";
790     this->cmd(cmdstr.str());
791
792     return *this;
793 }
794
795 //
-----
796 // set autoscale y
797 Gnuplot& Gnuplot::set_yautoscale()
798 {
799     this->cmd("set_yrange_restore");
800     this->cmd("set_autoscale_y");
801
802     return *this;
803 }
804
805 //
-----
806 // set the zrange
807 Gnuplot& Gnuplot::set_zrange(const int iFrom,
808                               const int iTo)
809 {
810     std::ostringstream cmdstr;
811
```

```

812     cmdstr << "set_zrange[" << iFrom << ":" << iTo << "];
813     this->cmd(cmdstr.str());
814
815     return *this;
816 }
817
818 //
-----

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

829 // set the palette range
830 Gnuplot& Gnuplot::set_cbrange(const int iFrom,
831                               const int iTo)
832 {
833     std::ostringstream cmdstr;
834
835     cmdstr << "set_cbrange[" << iFrom << ":" << iTo << "];
836     this->cmd(cmdstr.str());
837
838     return *this;
839 }
840
841 //
-----

842 // Plots a linear equation y=ax+b (where you supply the
843 // slope a and intercept b)
844 Gnuplot& Gnuplot::plot_slope(const double a,
845                               const double b,
846                               const std::string &title)
847 {

```

```

848     std::ostringstream cmdstr;
849
850     // command to be sent to gnuplot
851     if (this->nplots > 0 && this->two_dim == true)
852         cmdstr << "replot_";
853     else
854         cmdstr << "plot_";
855
856     cmdstr << a << "_*_x_+_ " << b << "_title_\n";
857
858     if (title == "")
859         cmdstr << "f(x)_=_ " << a << "_*_x_+_ " << b;
860     else
861         cmdstr << title;
862
863     cmdstr << "\"_with_ " << this->pstyle;
864
865     // Do the actual plot
866     this->cmd(cmdstr.str());
867
868     return *this;
869 }
870
871 //
-----
872 // Plot an equation which is supplied as a std::string y=f(x) (only
      f(x) expected)
873 Gnuplot& Gnuplot::plot_equation(const std::string &equation,
874                                 const std::string &title)
875 {
876     std::ostringstream cmdstr;
877
878     // command to be sent to gnuplot
879     if (this->nplots > 0 && this->two_dim == true)
880         cmdstr << "replot_";
881     else
882         cmdstr << "plot_";
883
884     cmdstr << equation << "_title_\n";
885
886     if (title == "")

```

```

887         cmdstr << "f(x)_=_ " << equation;
888     else
889         cmdstr << title;
890
891     cmdstr << "\"_with_" << this->pstyle;
892
893     // Do the actual plot
894     this->cmd(cmdstr.str());
895
896     return *this;
897 }
898
899 //
-----
900 // plot an equation supplied as a std::string y=(x)
901 Gnuplot& Gnuplot::plot_equation3d(const std::string &equation,
902                                   const std::string &title)
903 {
904     std::ostringstream cmdstr;
905
906     // command to be sent to gnuplot
907     if (this->nplots > 0 && this->two_dim == false)
908         cmdstr << "replot_";
909     else
910         cmdstr << "splot_";
911
912     cmdstr << equation << "_title_\n";
913
914     if (title == "")
915         cmdstr << "f(x,y)_=_ " << equation;
916     else
917         cmdstr << title;
918
919     cmdstr << "\"_with_" << this->pstyle;
920
921     // Do the actual plot
922     this->cmd(cmdstr.str());
923
924     return *this;
925 }
926

```

927 //

```

928 // Plots a 2d graph from a list of doubles (x) saved in a file
929 Gnuplot& Gnuplot::plotfile_x(const std::string &filename,
930                               const int column,
931                               const std::string &title)
932 {
933     // check if file exists
934     if( !(Gnuplot::file_exists(filename,4)) ) // check existence
        and read permission
935     {
936         std::ostringstream except;
937         if( !(Gnuplot::file_exists(filename,0)) ) // check
            existence
938             except << "File_" << filename << "_" << "does_not_exist";
939         else
940             except << "No_read_permission_for_File_" << filename
                << "_";
941         throw GnuplotException( except.str() );
942         return *this;
943     }
944
945     std::ostringstream cmdstr;
946
947     // command to be sent to gnuplot
948     if (this->nplots > 0 && this->two_dim == true)
949         cmdstr << "replot_";
950     else
951         cmdstr << "plot_";
952
953     cmdstr << "\"" << filename << "_" << "using_" << column;
954
955     if (title == "")
956         cmdstr << "_notitle_";
957     else
958         cmdstr << "_title_" << title << "_";
959
960     if(smooth == "")
961         cmdstr << "with_" << this->pstyle;
962     else
963         cmdstr << "smooth_" << this->smooth;

```

```

964
965     // Do the actual plot
966     this->cmd(cmdstr.str()); //nplots++; two_dim = true; already
        in this->cmd();
967
968     return *this;
969 }
970
971 //
-----

972 // Plots a 2d graph from a list of doubles: x
973 Gnuplot& Gnuplot::plot_x(const std::vector<double> &x,
974                          const std::string &title)
975 {
976     if (x.size() == 0)
977     {
978         throw GnuplotException("std::vector too small");
979         return *this;
980     }
981
982     std::ofstream tmp;
983     std::string name = create_tmpfile(tmp);
984     if (name == "")
985         return *this;
986
987     // write the data to file
988     for (unsigned int i = 0; i < x.size(); i++)
989         tmp << x[i] << std::endl;
990
991     tmp.flush();
992     tmp.close();
993
994     this->plotfile_x(name, 1, title);
995
996     return *this;
997 }
998
999 //
-----

1000 // Plots a 2d graph from a list of doubles (x y) saved in a file

```



```

1001 Gnuplot& Gnuplot::plotfile_xy(const std::string &filename,
1002                               const int column_x,
1003                               const int column_y,
1004                               const std::string &title)
1005 {
1006     // check if file exists
1007     if( !(Gnuplot::file_exists(filename,4)) ) // check existence
        and read permission
1008     {
1009         std::ostringstream except;
1010         if( !(Gnuplot::file_exists(filename,0)) ) // check
            existence
1011             except << "File_\n" << filename << "\n_does_not_exist";
1012         else
1013             except << "No_read_permission_for_File_\n" << filename
                << "\n";
1014         throw GnuplotException( except.str() );
1015         return *this;
1016     }
1017
1018     std::ostringstream cmdstr;
1019
1020     // command to be sent to gnuplot
1021     if (this->nplots > 0 && this->two_dim == true)
1022         cmdstr << "replot\n";
1023     else
1024         cmdstr << "plot\n";
1025
1026     cmdstr << "\"\" << filename << "\"_using\" << column_x << ":" <<
        column_y;
1027
1028     if (title == "")
1029         cmdstr << "_notitle\n";
1030     else
1031         cmdstr << "_title_\n" << title << "\"_";
1032
1033     if(smooth == "")
1034         cmdstr << "with_" << this->pstyle;
1035     else
1036         cmdstr << "smooth_" << this->smooth;
1037
1038     // Do the actual plot

```

```

1039     this->cmd(cmdstr.str());
1040
1041     return *this;
1042 }
1043
1044 //
-----
1045 // Plots a 2d graph from a list of doubles: x y
1046 Gnuplot& Gnuplot::plot_xy(const std::vector<double> &x,
1047                             const std::vector<double> &y,
1048                             const std::string &title)
1049 {
1050     if (x.size() == 0 || y.size() == 0)
1051     {
1052         throw GnuplotException("std::vectors too small");
1053         return *this;
1054     }
1055
1056     if (x.size() != y.size())
1057     {
1058         throw GnuplotException("Length of the std::vectors differs"
1059                                 );
1059         return *this;
1060     }
1061
1062     std::ofstream tmp;
1063     std::string name = create_tmpfile(tmp);
1064     if (name == "")
1065         return *this;
1066
1067     // write the data to file
1068     for (unsigned int i = 0; i < x.size(); i++)
1069         tmp << x[i] << " " << y[i] << std::endl;
1070
1071     tmp.flush();
1072     tmp.close();
1073
1074     this->plotfile_xy(name, 1, 2, title);
1075
1076     return *this;
1077 }

```

```

1078
1079 //
-----

1080 // Plots a 2d graph with errorbars from a list of doubles (x y dy)
      saved in a file
1081 Gnuplot& Gnuplot::plotfile_xy_err(const std::string &filename,
1082                                   const int column_x,
1083                                   const int column_y,
1084                                   const int column_dy,
1085                                   const std::string &title)
1086 {
1087     // check if file exists
1088     if( !(Gnuplot::file_exists(filename,4)) ) // check existence
        and read permission
1089     {
1090         std::ostringstream except;
1091         if( !(Gnuplot::file_exists(filename,0)) ) // check
            existence
1092             except << "File_" << filename << "\"_does_not_exist";
1093         else
1094             except << "No_read_permission_for_File_" << filename
                << "\"";
1095         throw GnuplotException( except.str() );
1096         return *this;
1097     }
1098
1099     std::ostringstream cmdstr;
1100
1101     // command to be sent to gnuplot
1102     if (this->nplots > 0 && this->two_dim == true)
1103         cmdstr << "replot_";
1104     else
1105         cmdstr << "plot_";
1106
1107     cmdstr << "\"" << filename << "\"_using_" << column_x << ":" <<
        column_y;
1108
1109     if (title == "")
1110         cmdstr << "_notitle_";
1111     else
1112         cmdstr << "_title_" << title << "\"_";

```

```

1113
1114     cmdstr << "with_" << this->pstyle << ",_" << filename << "\"_
        using_"
1115         << column_x << ":" << column_y << ":" << column_dy << "_
        notitle_with_errorbars";

1116
1117     // Do the actual plot
1118     this->cmd(cmdstr.str());
1119
1120     return *this;
1121 }
1122
1123 //
-----

1124 // plot x,y pairs with dy errorbars
1125 Gnuplot& Gnuplot::plot_xy_err(const std::vector<double> &x,
1126                               const std::vector<double> &y,
1127                               const std::vector<double> &dy,
1128                               const std::string &title)
1129 {
1130     if (x.size() == 0 || y.size() == 0 || dy.size() == 0)
1131     {
1132         throw GnuplotException("std::vectors_too_small");
1133         return *this;
1134     }
1135
1136     if (x.size() != y.size() || y.size() != dy.size())
1137     {
1138         throw GnuplotException("Length_of_the_std::vectors_differs"
                                );
1139         return *this;
1140     }
1141
1142     std::ofstream tmp;
1143     std::string name = create_tmpfile(tmp);
1144     if (name == "")
1145         return *this;
1146
1147     // write the data to file
1148     for (unsigned int i = 0; i < x.size(); i++)
1149         tmp << x[i] << "_" << y[i] << "_" << dy[i] << std::endl;

```

```

1150
1151     tmp.flush();
1152     tmp.close();
1153
1154     // Do the actual plot
1155     this->plotfile_xy_err(name, 1, 2, 3, title);
1156
1157     return *this;
1158 }
1159
1160 //
-----
1161 // Plots a 3d graph from a list of doubles (x y z) saved in a file
1162 Gnuplot& Gnuplot::plotfile_xyz(const std::string &filename,
1163                                const int column_x,
1164                                const int column_y,
1165                                const int column_z,
1166                                const std::string &title)
1167 {
1168
1169     // check if file exists
1170     if( !(Gnuplot::file_exists(filename,4)) ) // check existence
        and read permission
1171     {
1172         std::ostringstream except;
1173         if( !(Gnuplot::file_exists(filename,0)) ) // check
            existence
1174             except << "File_" << filename << "_" << "does not exist";
1175         else
1176             except << "No read permission for File_" << filename
                << "_";
1177         throw GnuplotException( except.str() );
1178         return *this;
1179     }
1180
1181     std::ostringstream cmdstr;
1182
1183     // command to be sent to gnuplot
1184     if (this->nplots > 0 && this->two_dim == false)
1185         cmdstr << "replot_";
1186     else

```

```

1187         cmdstr << "splot_";
1188
1189         cmdstr << "\" << filename << "\"_using_" << column_x << ":" <<
            column_y << ":" << column_z;
1190
1191         if (title == "")
1192             cmdstr << "_notitle_with_" << this->pstyle;
1193         else
1194             cmdstr << "_title_" << title << "\"_with_" << this->
                pstyle;
1195
1196         // Do the actual plot
1197         this->cmd(cmdstr.str());
1198
1199         return *this;
1200 }
1201
1202 //
-----
1203 // Plots a 3d graph from a list of doubles: x y z
1204 Gnuplot& Gnuplot::plot_xyz(const std::vector<double> &x,
1205                             const std::vector<double> &y,
1206                             const std::vector<double> &z,
1207                             const std::string &title)
1208 {
1209     if (x.size() == 0 || y.size() == 0 || z.size() == 0)
1210     {
1211         throw GnuplotException("std::vectors_too_small");
1212         return *this;
1213     }
1214
1215     if (x.size() != y.size() || x.size() != z.size())
1216     {
1217         throw GnuplotException("Length_of_the_std::vectors_differs"
                                );
1218         return *this;
1219     }
1220
1221
1222     std::ofstream tmp;
1223     std::string name = create_tmpfile(tmp);

```

```

1224     if (name == "")
1225         return *this;
1226
1227     // write the data to file
1228     for (unsigned int i = 0; i < x.size(); i++)
1229     {
1230         tmp << x[i] << " " << y[i] << " " << z[i] <<std::endl;
1231     }
1232
1233     tmp.flush();
1234     tmp.close();
1235
1236
1237     this->plotfile_xyz(name, 1, 2, 3, title);
1238
1239     return *this;
1240 }
1241
1242
1243
1244 //
-----
1245 /// * note that this function is not valid for versions of GNUPlot
      below 4.2
1246 Gnuplot& Gnuplot::plot_image(const unsigned char * ucPicBuf,
1247                               const int iWidth,
1248                               const int iHeight,
1249                               const std::string &title)
1250 {
1251     std::ofstream tmp;
1252     std::string name = create_tmpfile(tmp);
1253     if (name == "")
1254         return *this;
1255
1256     // write the data to file
1257     int iIndex = 0;
1258     for(int iRow = 0; iRow < iHeight; iRow++)
1259     {
1260         for(int iColumn = 0; iColumn < iWidth; iColumn++)
1261         {
1262             tmp << iColumn << " " << iRow << " " << static_cast<

```

```

        float>(ucPicBuf[iIndex++]) << std::endl;
1263     }
1264 }
1265
1266 tmp.flush();
1267 tmp.close();
1268
1269
1270 std::ostringstream cmdstr;
1271
1272 // command to be sent to gnuplot
1273 if (this->nplots > 0 && this->two_dim == true)
1274     cmdstr << "replot_";
1275 else
1276     cmdstr << "plot_";
1277
1278 if (title == "")
1279     cmdstr << "\"" << name << "\"_with_image";
1280 else
1281     cmdstr << "\"" << name << "\"_title\"" << title << "\"_
        with_image";
1282
1283 // Do the actual plot
1284 this->cmd(cmdstr.str());
1285
1286 return *this;
1287 }
1288
1289 //
    -----
1290 // Sends a command to an active gnuplot session
1291 Gnuplot& Gnuplot::cmd(const std::string &cmdstr)
1292 {
1293     if( !(this->valid) )
1294     {
1295         return *this;
1296     }
1297
1298     // int fputs ( const char * str, FILE * stream );
1299     // writes the string str to the stream.
1300     // The function begins copying from the address specified (str)

```



```

        until it reaches the
1301 // terminating null character ('\0'). This final null-character
        is not copied to the stream.
1302 fputs( (cmdstr+"\n").c_str(), this->gnucmd );
1303
1304 // int fflush ( FILE * stream );
1305 // If the given stream was open for writing and the last i/o
        operation was an output operation,
1306 // any unwritten data in the output buffer is written to the
        file.
1307 // If the argument is a null pointer, all open files are
        flushed.
1308 // The stream remains open after this call.
1309 fflush(this->gnucmd);
1310
1311
1312 if( cmdstr.find("replot") != std::string::npos )
1313 {
1314     return *this;
1315 }
1316 else if( cmdstr.find("splot") != std::string::npos )
1317 {
1318     this->two_dim = false;
1319     this->nplots++;
1320 }
1321 else if( cmdstr.find("plot") != std::string::npos )
1322 {
1323     this->two_dim = true;
1324     this->nplots++;
1325 }
1326 return *this;
1327 }
1328
1329 //
        -----
1330 // Sends a command to an active gnuplot session, identical to cmd()
1331 Gnuplot& Gnuplot::operator<<(const std::string &cmdstr)
1332 {
1333     this->cmd(cmdstr);
1334     return *this;
1335 }

```

```

1336
1337 //
-----

1338 // Opens up a gnuplot session, ready to receive commands
1339 void Gnuplot::init()
1340 {
1341     // char * getenv ( const char * name ); get value of an
        environment variable
1342     // Retrieves a C string containing the value of the environment
        variable whose
1343     // name is specified as argument.
1344     // If the requested variable is not part of the environment
        list, the function returns a NULL pointer.
1345 #if ( defined(unix) || defined(__unix) || defined(__unix__) ) && !
        defined(__APPLE__)
1346     if (getenv("DISPLAY") == NULL)
1347     {
1348         this->valid = false;
1349         throw GnuplotException("Can't find DISPLAY variable");
1350     }
1351 #endif
1352
1353     // if gnuplot not available
1354     if (!Gnuplot::get_program_path())
1355     {
1356         this->valid = false;
1357         throw GnuplotException("Can't find gnuplot");
1358     }
1359
1360     // open pipe
1361     std::string tmp = Gnuplot::m_sGnUPlotPath + "/" + Gnuplot::
        m_sGnUPlotFileName;
1362
1363     // FILE *popen(const char *command, const char *mode);
1364     // The popen() function shall execute the command specified by
        the string command,
1365     // create a pipe between the calling program and the executed
        command, and
1366     // return a pointer to a stream that can be used to either read
        from or write to the pipe.
1367 #if defined(WIN32) || defined(_WIN32) || defined(__WIN32__) ||

```

```

        defined(__TOS_WIN__)
1368     this->gnucmd = _popen(tmp.c_str(), "w");
1369 #elif defined(unix) || defined(__unix) || defined(__unix__) ||
        defined(__APPLE__)
1370     this->gnucmd = popen(tmp.c_str(), "w");
1371 #endif
1372
1373     // popen() shall return a pointer to an open stream that can be
        used to read or write to the pipe.
1374     // Otherwise, it shall return a null pointer and may set errno
        to indicate the error.
1375     if (!this->gnucmd)
1376     {
1377         this->valid = false;
1378         throw GnuplotException("Couldn't open connection to gnuplot
            ");
1379     }
1380
1381     this->nplots = 0;
1382     this->valid = true;
1383     this->smooth = "";
1384
1385     //set terminal type
1386     this->showonscreen();
1387
1388     return;
1389 }
1390
1391 //
    -----
1392 // Find out if a command lives in m_sGnuplotPath or in PATH
1393 bool Gnuplot::get_program_path()
1394 {
1395     // first look in m_sGnuplotPath for Gnuplot
1396     std::string tmp = Gnuplot::m_sGnuplotPath + "/" + Gnuplot::
        m_sGnuplotFileName;
1397
1398 #if defined(WIN32) || defined(_WIN32) || defined(__WIN32__) ||
        defined(__TOS_WIN__)
1399     if ( Gnuplot::file_exists(tmp,0) ) // check existence
1400 #elif defined(unix) || defined(__unix) || defined(__unix__) ||

```

```

        defined(__APPLE__)
1401     if ( Gnuplot::file_exists(tmp,1) ) // check existence and
        execution permission
1402 #endif
1403     {
1404         return true;
1405     }
1406
1407     // second look in PATH for Gnuplot
1408     char *path;
1409     // Retrieves a C string containing the value of the environment
        variable PATH
1410     path = getenv("PATH");
1411
1412     if (path == NULL)
1413     {
1414         throw GnuplotException("Path_is_not_set");
1415         return false;
1416     }
1417     else
1418     {
1419         std::list<std::string> ls;
1420         //split path (one long string) into list ls of strings
1421 #if defined(WIN32) || defined(_WIN32) || defined(__WIN32__) ||
        defined(__TOS_WIN__)
1422         stringtok(ls,path,";");
1423 #elif defined(unix) || defined(__unix) || defined(__unix__) ||
        defined(__APPLE__)
1424         stringtok(ls,path,":");
1425 #endif
1426         // scan list for Gnuplot program files
1427         for (std::list<std::string>::const_iterator i = ls.begin();
            i != ls.end(); ++i)
1428         {
1429             tmp = (*i) + "/" + Gnuplot::m_sGNUPlotFileName;
1430 #if defined(WIN32) || defined(_WIN32) || defined(__WIN32__) ||
        defined(__TOS_WIN__)
1431             if ( Gnuplot::file_exists(tmp,0) ) // check existence
1432 #elif defined(unix) || defined(__unix) || defined(__unix__) ||
        defined(__APPLE__)
1433             if ( Gnuplot::file_exists(tmp,1) ) // check existence
                and execution permission

```

```

1434 #endif
1435         {
1436             Gnuplot::m_sGnuplotPath = *i; // set m_sGnuplotPath
1437             return true;
1438         }
1439     }
1440
1441     tmp = "Can't find gnuplot neither in PATH nor in \" +
1442           Gnuplot::m_sGnuplotPath + "\"";
1443     throw GnuplotException(tmp);
1444
1445     Gnuplot::m_sGnuplotPath = "";
1446     return false;
1447 }
1448
1449 //
-----
1450 // check if file exists
1451 bool Gnuplot::file_exists(const std::string &filename, int mode)
1452 {
1453     if ( mode < 0 || mode > 7)
1454     {
1455         throw std::runtime_error("In function \"Gnuplot::
1456           file_exists\": mode has to be an integer between 0 and 7
1457           ");
1458         return false;
1459     }
1460
1461     // int _access(const char *path, int mode);
1462     // returns 0 if the file has the given mode,
1463     // it returns -1 if the named file does not exist or is not
1464     // accessible in the given mode
1465     // mode = 0 (F_OK) (default): checks file for existence only
1466     // mode = 1 (X_OK): execution permission
1467     // mode = 2 (W_OK): write permission
1468     // mode = 4 (R_OK): read permission
1469     // mode = 6 : read and write permission
1470     // mode = 7 : read, write and execution permission
1471 #if defined(WIN32) || defined(_WIN32) || defined(__WIN32__) ||
1472     defined(__TOS_WIN__)

```

```

1469     if (_access(filename.c_str(), mode) == 0)
1470 #elif defined(unix) || defined(__unix) || defined(__unix__) ||
        defined(__APPLE__)
1471     if (access(filename.c_str(), mode) == 0)
1472 #endif
1473     {
1474         return true;
1475     }
1476     else
1477     {
1478         return false;
1479     }
1480
1481 }
1482
1483 //
-----
1484 // Opens a temporary file
1485 std::string Gnuplot::create_tmpfile(std::ofstream &tmp)
1486 {
1487 #if defined(WIN32) || defined(_WIN32) || defined(__WIN32__) ||
        defined(__TOS_WIN__)
1488     char name[] = "gnuplotiXXXXXX"; //tmp file in working directory
1489 #elif defined(unix) || defined(__unix) || defined(__unix__) ||
        defined(__APPLE__)
1490     char name[] = "/tmp/gnuplotiXXXXXX"; // tmp file in /tmp
1491 #endif
1492
1493     // check if maximum number of temporary files reached
1494     if (Gnuplot::tmpfile_num == GP_MAX_TMP_FILES - 1)
1495     {
1496         std::ostringstream except;
1497         except << "Maximum number of temporary files reached (" <<
            GP_MAX_TMP_FILES
1498             << "): cannot open more files" << std::endl;
1499
1500         throw GnuplotException( except.str() );
1501         return "";
1502     }
1503
1504     // int mkstemp(char *name);

```

```

1505 // shall replace the contents of the string pointed to by "name
      " by a unique filename,
1506 // and return a file descriptor for the file open for reading
      and writing.
1507 // Otherwise, -1 shall be returned if no suitable file could be
      created.
1508 // The string in template should look like a filename with six
      trailing 'X' s;
1509 // mkstemp() replaces each 'X' with a character from the
      portable filename character set.
1510 // The characters are chosen such that the resulting name does
      not duplicate the name of an existing file at the time of a
      call to mkstemp()

1511
1512
1513 // open temporary files for output
1514 #if defined(WIN32) || defined(_WIN32) || defined(__WIN32__) ||
      defined(__TOS_WIN__)
1515     if (_mktemp(name) == NULL)
1516 #elif defined(unix) || defined(__unix) || defined(__unix__) ||
      defined(__APPLE__)
1517     if (mkstemp(name) == -1)
1518 #endif
1519     {
1520         std::ostringstream except;
1521         except << "Cannot create temporary file\" << name << "\"
            ;
1522         throw GnuplotException(except.str());
1523         return "";
1524     }
1525
1526 tmp.open(name);
1527 if (tmp.bad())
1528 {
1529     std::ostringstream except;
1530     except << "Cannot create temporary file\" << name << "\"
        ;
1531     throw GnuplotException(except.str());
1532     return "";
1533 }
1534
1535 // Save the temporary filename

```

```
1536     this->tmpfile_list.push_back(name);
1537     Gnuplot::tmpfile_num++;
1538
1539     return name;
1540 }
```

Apresenta-se na listagem 6.25 o programa main().

Listing 6.25: 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.

```
#####
Projeto Programacao Pratica - Calculo Indice de produtividade
Professor: Andre Duarte Bueno
Alunos: Carolina Bastos
      Douglas Ribeiro
#####
Gostaria de executar o programa? 1 - Sim | 0 - nao
C:/Program' não é reconhecido como um comando interno
ou externo, um programa operável ou um arquivo em lotes.
#####
Importacao de dados
#####
Digite nome do arquivo de dados.
Arquivos Disponíveis
"./Src/dados.txt"
"./Src/dadosIP - Copy.dat"
"./Src/ReservatorioA.dat"
ReservatorioA.dat
#####
Dados estao corretos? 1 -sim | 2- nao
Kh = 1.5e-13 | Rw = 0.11 | Re = 309.089 | L = 304.79 | Kv = 5e-14 | H = 15.24 | mi = 0.005 | Bo = 0.2
#####
Qual nome do arquivo de saida de dados?
#####
```

Figura 7.1: Interface com usuário do programa.

7.2 Teste 2: Descrição

Instruções::

- 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 Shedd.
- 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.

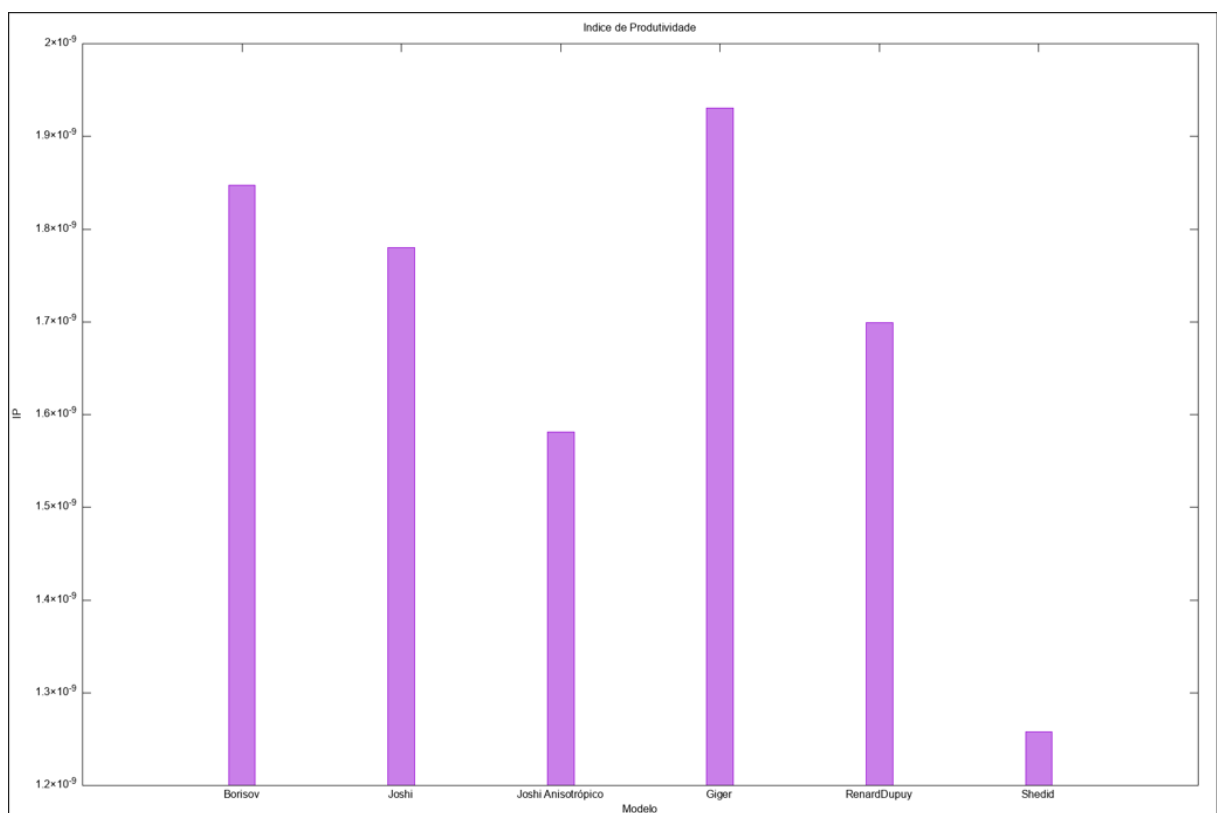


Figura 7.2: Histograma com os resultados para cada método do cálculo do Índice de Produtividade.

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 Sheddidd.

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 está o código.

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

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

```
doxygen -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`.

Lista de Arquivos	
Esta é a lista de todos os arquivos e suas respectivas descrições:	
CBorisov.cpp	
CBorisov.h	
CCalcIP.cpp	
CCalcIP.h	
CDadosFluido.cpp	
CDadosFluido.h	
CDadosPoco.cpp	
CDadosPoco.h	
CDadosReservatorio.cpp	
CDadosReservatorio.h	
CDadosReservatorioVertical.cpp	
CDadosReservatorioVertical.h	
CGiger.cpp	
CGiger.h	
CGnuplot.cpp	
CGnuplot.h	
CJoshi.cpp	
CJoshi.h	
CJoshiAnisotropico.cpp	
CJoshiAnisotropico.h	
CRenardDunuv.cpp	

Figura 8.1: Histograma com os resultados para cada método do cálculo do Índice de Produtividade.

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 Reservatórios de Petróleo*. Editora Interciencia, Rio de Janeiro. 9
- [Rumbaugh et al., 1994] Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., and Lorenzen, 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

A

Análise orientada a objeto, 13
AOO, 13
Associações, 24
atributos, 23

C

Casos de uso, 4
colaboração, 16
comunicação, 16
Concepção, 3
Controle, 21

D

Diagrama de colaboração, 16
Diagrama de componentes, 25
Diagrama de execução, 26
Diagrama de máquina de estado, 17
Diagrama de sequência, 15

E

Efeitos do projeto nas associações, 24
Efeitos do projeto nas heranças, 24
Efeitos do projeto nos métodos, 23
Elaboração, 7
especificação, 3
Especificações, 3
estado, 17
Eventos, 15

H

Heranças, 24
heranças, 24

I

Implementação, 27

M

Mensagens, 15
métodos, 23
modelo, 22

O

otimizações, 25

P

Plataformas, 21
POO, 21
Projeto do sistema, 20
Projeto orientado a objeto, 21
Protocolos, 20

R

Recursos, 20