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

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

TRABALHO DA DISCIPLINA PROGRAMAÇÃO PRÁTICA

Versão 1:

CAROLINA BASTOS E DOUGLAS RIBEIRO

Prof. André Duarte Bueno

MACAÉ - RJ

Agosto - 2021

Sumário

Capítulo 1

Introdução

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

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

1.1 Escopo do problema

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

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

1.2 Objetivos

Os objetivos deste projeto de engenharia são:

- Objetivo geral:

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

- Objetivos específicos:

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

Capítulo 2

Especificação

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

2.1 O que é a especificação?

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

2.2 Nome do sistema/produto

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

2.3 Especificação

Apresenta-se a seguir a especicação do software.

O projeto a ser desenvolvido cosiste 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 - ou poderá inserir esses dados em um arquivo .txt e usá-lo como input do programa. 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.

RF-02	O programa também deverá permitir o carregamento desses dados de entrada a partir de um arquivo de disco criado pelo usuário.

RF-03	O usuário deverá ter liberdade para escolher o tipo de formação que o poço irá atravessar (isotrópica ou anisotrópica)

RF-04	O usuário deverá ter liberdade para escolher o modelo matemático para o cálculo do IP.

RF-05	O programa deverá mostrar os resultados dos cálculos de IP na tela.

RF-06	O usuário poderá plotar seus resultados em um gráfico, podendo este ser salvo como imagem.

2.3.2 Requisitos não funcionais

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

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

2.4 Casos de uso

A tabela 2.1 apresenta um caso de uso do sistema.

Tabela 2.1: Exemplo de caso de uso

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

2.4.1 Diagrama de caso de uso geral

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

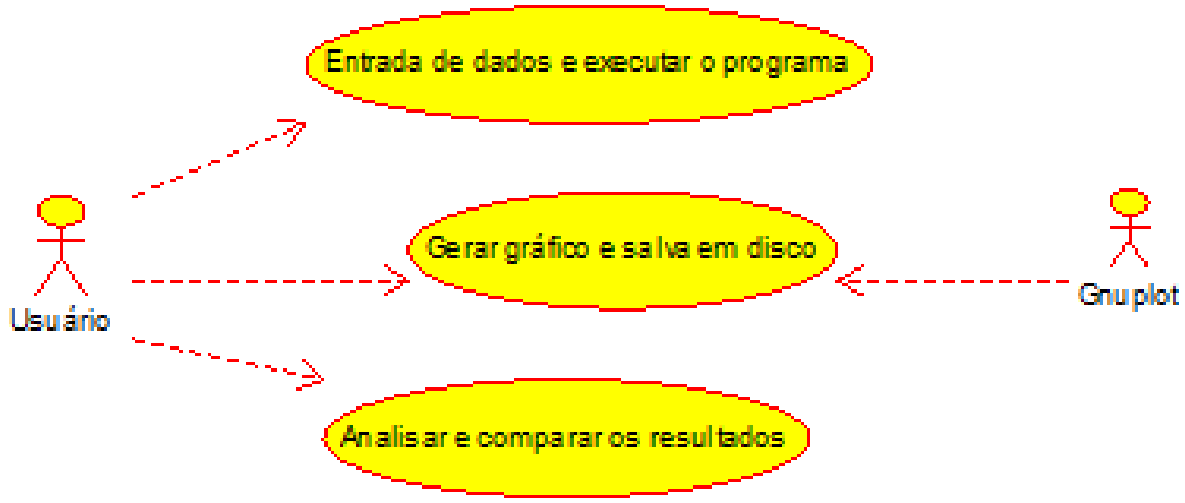

Figura 2.1: Diagrama de caso de uso geral – Cálculo do IP

2.4.2 Diagrama de caso de uso específico

O caso de uso Comparar IP de um reservatório anisotrópico e anisotrópico descrito na Figura 2.1 e na Tabela 2.1 é detalhado na Figura 2.2. O usuário pode variar os parâmetros

do poço e do reservatório e então plotar esses diferentes cenários em um gráfico para fazer comparações e definir qual melhor se adequa ao projeto.

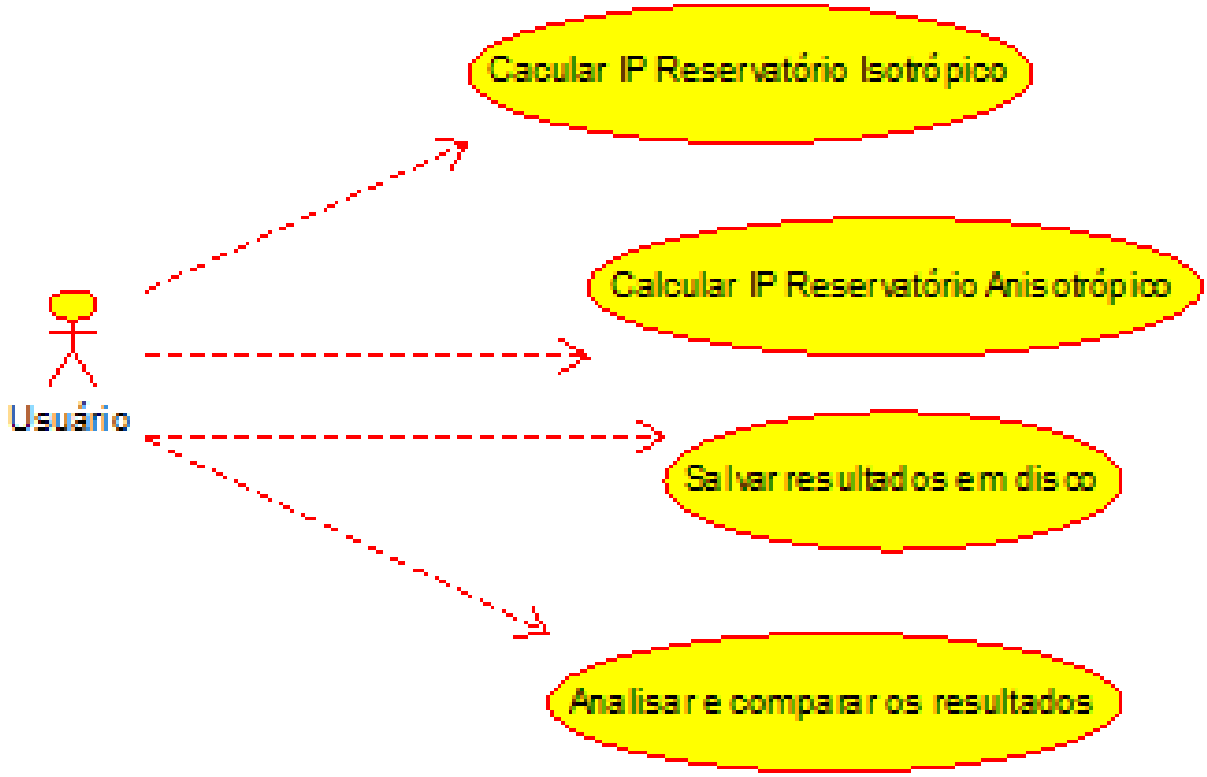

Figura 2.2: Diagrama de caso de uso específico – Comparando o IP de reservatório isotrópico com um anisotrópico

Capítulo 3

Elaboração

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

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

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

3.1 Análise de domínio

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

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

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

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

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

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

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

3.2 Formulação teórica

3.2.1 Produtividade de Poços

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

3.2.2 Índice de Produtividade

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

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

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

3.2.3 Efeito Skin

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

um fenômeno real, o dano à formação.

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

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

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

3.2.4 Regime Permanente

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

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

1. São de fácil dedução analítica;

2. Podem ser usados para obter soluções para o fluxo transiente, usa-se o artifício de aumentar o raio de drenagem com o tempo;

3. Podem ser usadas para se obter soluções para o fluxo pseudopermanente por meio do emprego do fator de Dietz, que permite o cálculo da pseudopressão para diversas geometrias do reservatório;

4. Podem ser verificadas experimentalmente por meio de modelos de laboratório [ROSA, 2006].

3.2.5 Produtividade de Poços Horizontais

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

- Borisov:

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

- Giger:

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

- Joshi:

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

Onde:

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

$IP = Índice\ de\ Produtividade$

$k_h = permeabilidade\ horizontal$

$h = altura\ do\ reservatório$

$\mu = viscosidade\ do\ óleo$

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

$L = comprimento\ horizontal\ do\ reservatório$

$r_w = raio\ do\ poço$

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

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

Onde:

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

E a contante C é mostrada na figura 3.1 abaixo:

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

Figura 3.1: Constante C

3.2.6 Cálculo do IP com anisotropia

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

Onde:

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

ΔP *é a queda de pressão no reservatório*

- Modelo de Renard e Dupuy:

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

Onde:

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

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

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

3.3 Diagrama de pacotes – assuntos

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

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

Veja Figura 3.2.

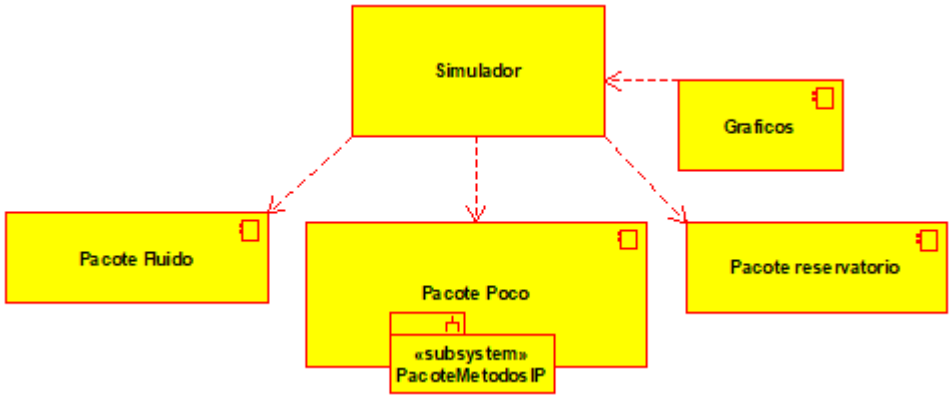

Figura 3.2: Diagrama de Pacotes

Capítulo 4

AOO – Análise Orientada a Objeto

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

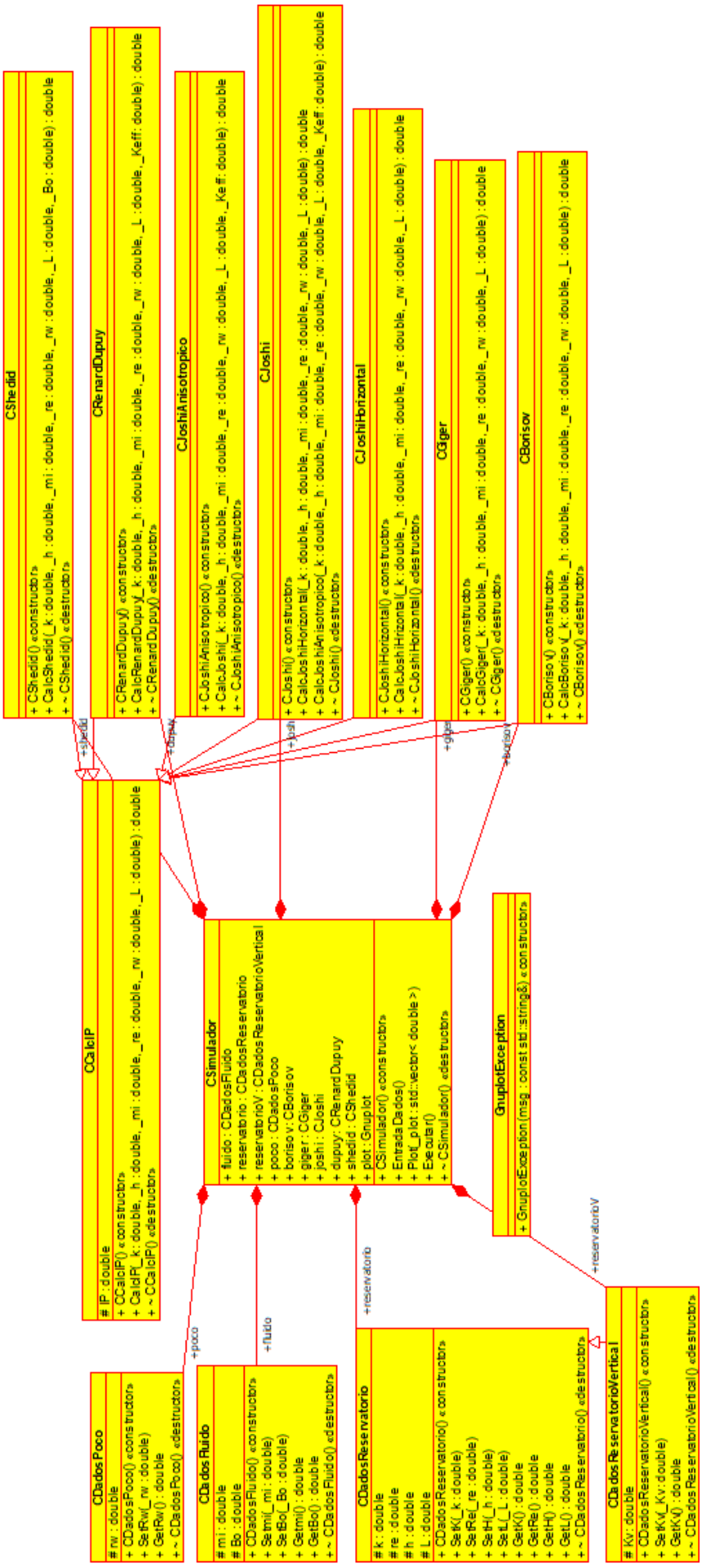

Figura 4.1: Diagrama de classes

4.1.1 Dicionário de classes

- Classe CDadosPoco: Solicita ao usuário os dados do poço e armazena.
- Classe CDadosFluido: Solicita ao usuário os dados do fluido e armazena.
- Classe CDadosReservatorio: Solicita ao usuário os dados do reservatório e armazena.
- Classe CDadosReservatorioVertical: Solicita ao usuário os dados do reservatório vertical e armazena.
- Classe CShedid: Calcula o IP do poço pelo modelo de Shedid.
- Classe CRenardDupuy: Calcula o IP do poço pelo modelo de Renard&Dupuy.
- Classe CJoshiAnisotropico: Calcula o IP do poço pelo modelo de Joshi em casos de reservatórios anisotrópicos.
- Classe CJoshi: Calcula o IP do poço pelo modelo de Joshi.
- Classe CJoshiHorizontal: Calcula o IP do poço horizontal pelo modelo de Joshi.
- Classe CGiger: Calcula o IP do poço pelo modelo de Giger.
- Classe CBorisov: Calcula o IP do poço pelo modelo de Borisov.
- Classe CCalcIP: Calcula o IP, utilizando os modelos disponíveis.
- Classe CSimulador: Faz as simulações do índice de produtividades dos poços (classe *main* do programa)
- Classe GnuplotExcrption: Gera uma visualização gráfica dos resultados usando software externo Gnuplot.

4.2 Diagrama de seqüência – eventos e mensagens

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

4.2.1 Diagrama de sequência geral

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

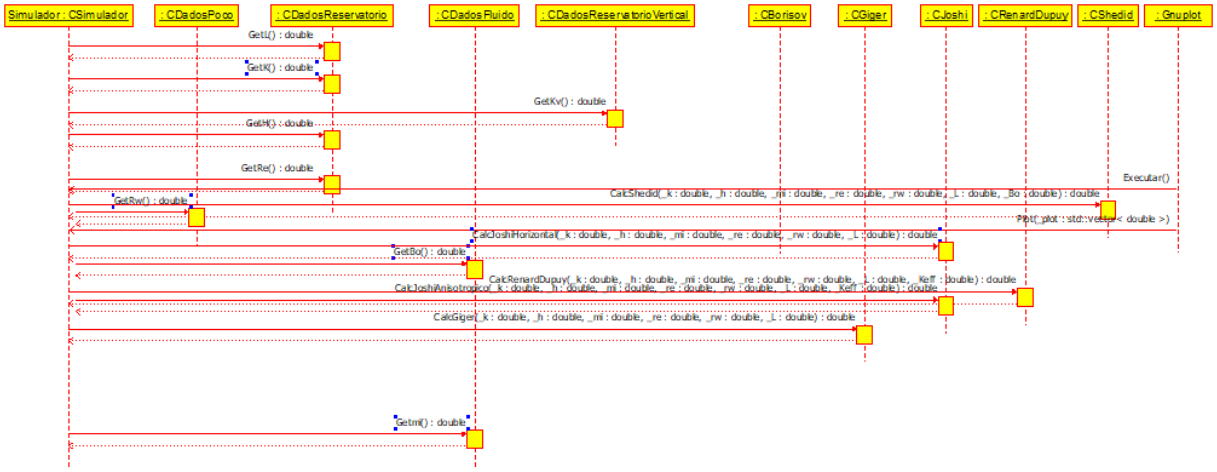

Figura 4.2: Diagrama de seqüência geral

4.3 Diagrama de comunicação – colaboração

No diagrama de comunicação o foco é a interação e a troca de mensagens e dados entre os objetos. Na figura 4.3 o diagrama de comunicação mostra a sequência do software para um caso em que o modelo de Joshi é utilizados para o cálculo do IP. Observe que a Classe CSimulador acessa os parâmetros do fluido, do poço e do reservatório a partir das classes CDadosFluido, CDadosPoco e CDadosReservatorio, respectivamente, que passa os atributos informados pelo usuário para as classes CCalcIP que por sua fez chama a classe CJoshi. A classe CSimulador então, após realizar os cálculo, envia esses resultados para a classe CGnuplotException que gera a visualização e salva esses resultados de forma gráfica.

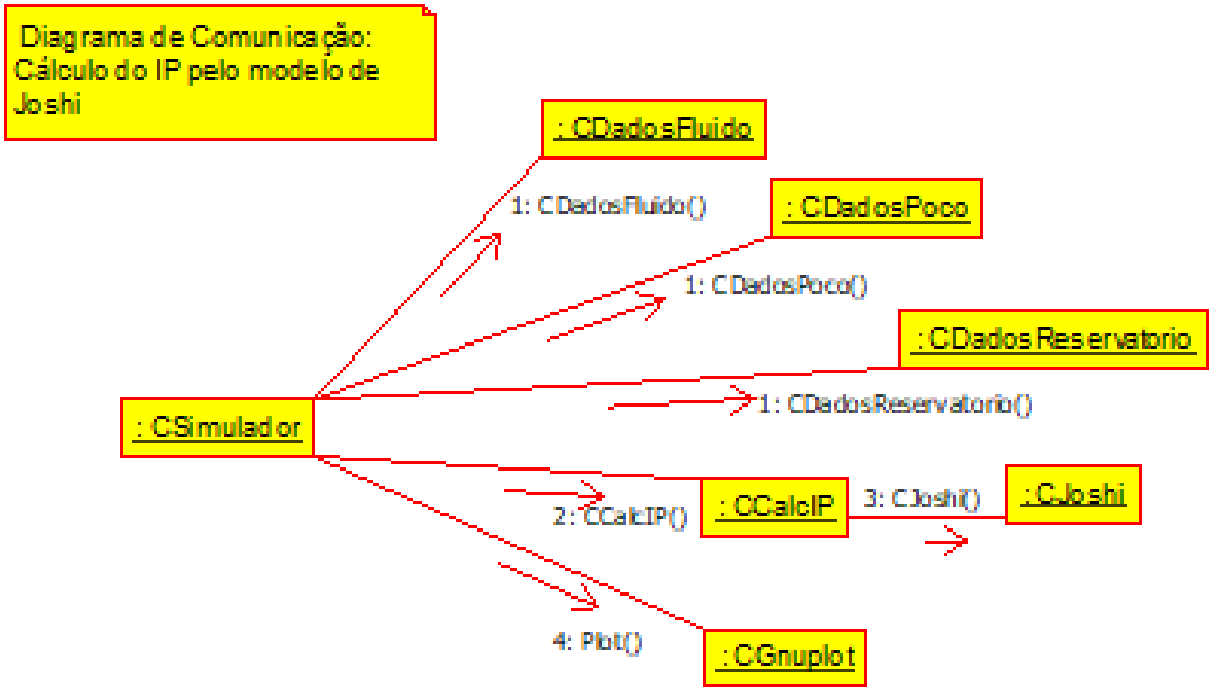

Figura 4.3: Diagrama de comunicação

4.4 Diagrama de máquina de estado

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

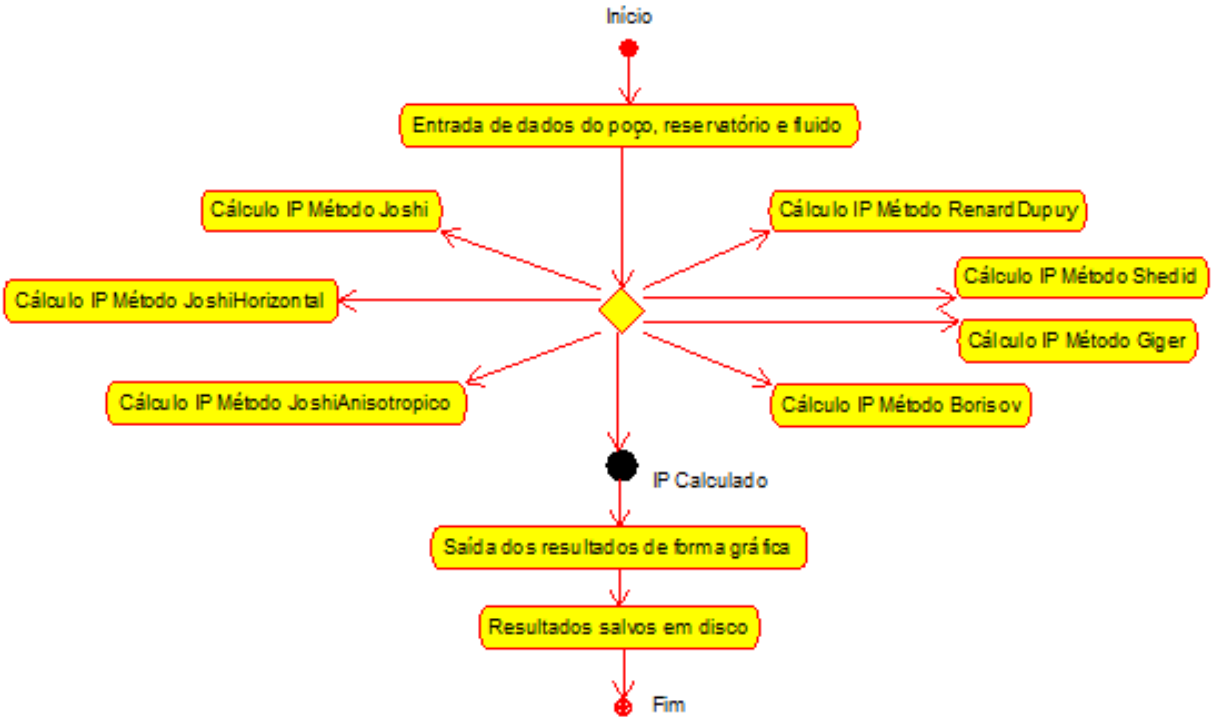

Figura 4.4: Diagrama de máquina de estado

4.5 Diagrama de atividades

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

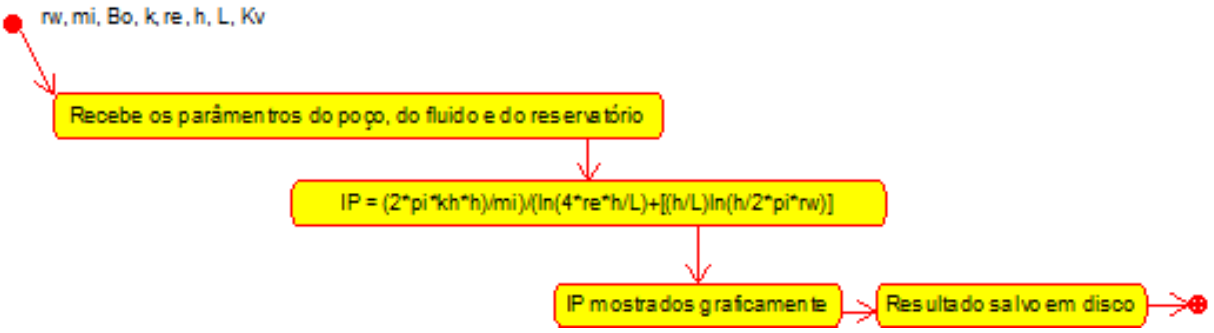

Figura 4.5: Diagrama de atividades

Capítulo 5

Projeto

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

5.1 Projeto do sistema

Depois da análise orientada a objeto desenvolve-se o projeto do sistema, qual envolve etapas como a definição dos protocolos, da interface API, o uso de recursos, a subdivisão do sistema em subsistemas, a alocação dos subsistemas ao hardware e a seleção das estruturas de controle, a seleção das plataformas do sistema, das bibliotecas externas, dos padrões de projeto, além da tomada de decisões conceituais e políticas que formam a infraestrutura do projeto.

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

Segundo [Rumbaugh et al., 1994, Blaha and Rumbaugh, 2006], o projeto do sistema é a estratégia de alto nível para resolver o problema e elaborar uma solução. Você deve se preocupar com itens como:

1. Protocolos

    - No software,o usuário poderá escolher se irá entrar com os dados do problema de forma manual ou se a entrada de dados será feita de modo que se importe 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.

2. Recursos

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

3. Controle

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

4. Plataformas

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

5. Padrões de projeto

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

5.2 Projeto orientado a objeto – POO

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

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

Efeitos do projeto no modelo estrutural

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

Efeitos do projeto no modelo dinâmico

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

Efeitos do projeto nos atributos

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

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

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 CSimulador e a partir daí acessar as classes herdeiras que calculam o IP a partir de modelos específicos. Anteriormente a CSimulador acessava diretamente todas essas classes.

- Abstração do comportamento comum (duas classes podem ter uma superclasse em comum).

  - Não se aplica a este projeto.

- Utilização de delegação para compartilhar a implementação (quando você cria uma herança irreal para reaproveitar código). Usar com cuidado.

  - Não se aplica a este projeto.

- Revise as heranças no diagrama de classes.

  - Foi criado relaciomaneto de herança entre a classe genérica CCalcIP e as demais CJoshi, CGiger.. que calculam o IP a partir de um modelo específico.

Efeitos do projeto nas associações

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

  - As associações foram criadas e modificadas ao longo do desenvolvimento do código, respeitando a hierarquia das classes.

- Se existe uma relação de "muitos", pode-se implementar a associação com a utilização de um dicionário, que é um mapa das associações entre objetos. Assim, o objeto A acessa o dicionário fornecendo uma chave (um nome para o objeto que deseja acessar) e o dicionário retorna um valor (um ponteiro) para o objeto correto.

  - Não se aplica a este projeto.

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

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

Efeitos do projeto nas otimizações

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

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

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

  - Não identificamos.

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

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

- Revise as associações nos diagramas de classes.

  - Foram revisadas a medida que desenvolvemos o código.

5.3 Diagrama de componentes

O diagrama de componentes mostra a forma como os componentes do software se relacionam, suas dependências. Inclui itens como: componentes, subsistemas, executáveis, nós, associações, dependências, generalizações, restrições e notas. Exemplos de componentes são bibliotecas estáticas, bibliotecas dinâmicas, dlls, componentes Java, executáveis, arquivos de disco, código-fonte.

Veja na Figura 5.1 um exemplo de diagrama de componentes. Observe que este inclui muitas dependências, ilustrando as relações entre os arquivos.

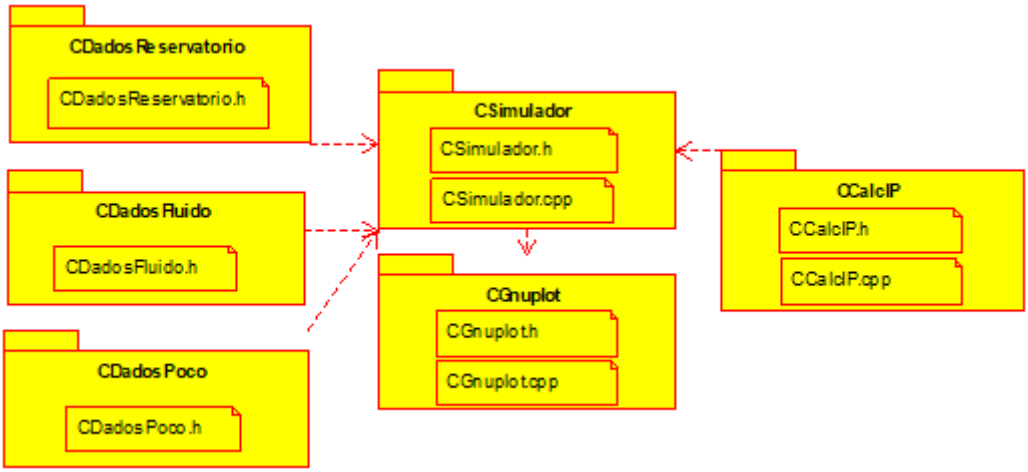

Figura 5.1: Diagrama de componentes

5.4 Diagrama de implantação

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

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

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

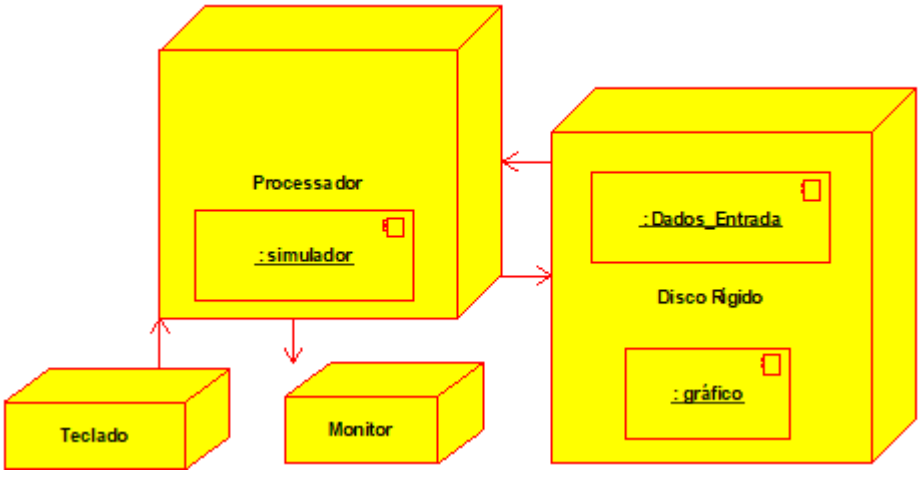

Figura 5.2: Diagrama de implantação

Capítulo 6

Implementação

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

6.1 Código fonte

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

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

Listing 6.1: Arquivo de cabeçalho da classe CDadosFluido.

```
1 #ifndef CDADOSFLUIDO_H_
2 #define CDADOSFLUIDO_H_
3
4
5 class CDadosFluido {
6
7         protected:
8
9         double mi, Bo;
10
11        public:
12
13                CDadosFluido(){};
14
15                void Setmi(double _mi);
16                void SetBo(double _Bo);
17                double Getmi();
18                double GetBo();
19
```

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

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

Listing 6.2: Arquivo de implementação da classe CDadosFluido.

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

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

Listing 6.3: Arquivo de cabeçalho da classe CDadosPoco.

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

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

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

Listing 6.4: Arquivo de implementação da classe CDadosPoco.

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

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

Listing 6.5: Arquivo de cabeçalho da classe CDadosReservatorio.

```
1#ifndef CDADOSRESERVATORIO_H_
2#define CDADOSRESERVATORIO_H_
3
4class CDadosReservatorio {
5
6          protected:
7
8                  double k, re, h, L;
9
10         public:
11
12         CDadosReservatorio(){};
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          ~CDadosReservatorio(){};
24
25};
26
27
28
29#endif
```

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

Listing 6.6: Arquivo de implementação da classe CDadosReservatorio.

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

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

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

Listing 6.7: Arquivo de cabeçalho da classe CDadosReservatorioVertical.

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

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

Listing 6.8: Arquivo de implementação da classe CDadosReservatorioVertical.

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

#include "CDadosReservatorioVertical.h"

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


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

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

Listing 6.9: Arquivo de cabeçalho da classe CCalcIP.

```
#ifndef CCalcIP_H
#define CCalcIP_H

class CCalcIP
{

        protected:

                double IP;

        public:

                CCalcIP(){};

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

                ~CCalcIP(){};

};

#endif
```

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

Listing 6.10: Arquivo de implementação da classe CCalcIP.

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

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

Listing 6.11: Arquivo de cabeçalho da classe CBorisov.

```
#ifndef CBORISOV_H_
#define CBORISOV_H_

#include "CCalcIP.h"

class CBorisov : CCalcIP
{

        public:

                CBorisov(){};

                double CalcBorisov(double _k, double _h, double _mi
                    , double _re, double _rw, double _L);

                ~CBorisov(){};

};

#endif
```

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

Listing 6.12: Arquivo de implementação da classe CBorisov.

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

#include "CBorisov.h"


double CBorisov::CalcBorisov(double _k, double _h, double _mi,
    double _re, double _rw, double _L)
{

        double A, B, C;
```

```
12
13          A = (2*M_PI*_k*_h)/_mi ;
14          B = log((4*_re)/_L);
15          C = (_h/_L)*log(_h/(2*M_PI*_rw));
16
17          IP = A / (B+C);
18
19          return IP;
20 }
```

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

Listing 6.13: Arquivo de cabeçalho da classe CGiger.

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

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

Listing 6.14: Arquivo de implementação da classe CGiger.

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

```
6
7double CGiger::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*M_PI*_k*_L)/_mi;
13          B = _L/_h;
14          C = (1 + (sqrt((1 - pow(( _L/(2*_re)) ,2)))));
15          D = (_L/(2*_re));
16          E = log(_h/(2*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 CJoshi.

Listing 6.15: Arquivo de cabeçalho da classe CJoshi.

```
1#ifndef CJOSHI_H_
2#define CJOSHI_H_
3
4#include "CCalcIP.h"
5
6class CJoshi : CCalcIP
7{
8
9          public:
10
11                    CJoshi(){};
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                    ~CJoshi(){};
17
18};
```

```
19
20#endif
```

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

Listing 6.16: Arquivo de implementação da classe CJoshi.

```
1#define _USE_MATH_DEFINES
2#include <math.h>
3
4
5#include "CJoshi.h"
6
7
8double CJoshi::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)*sqrt(0.5 + (sqrt(0.25 + (pow((2*_re)/_L, 4)))));
14        A = (2*M_PI*_k*_h)/_mi;
15        B = log((a + (sqrt(pow(a, 2) - pow(_L/2, 2))))/(_L/2));
16        C = (_h/_L)*log(_h/(2*_rw));
17
18        IP = A / (B + C);
19
20        return IP;
21}
22
23double CJoshi::CalcJoshiAnisotropico(double _k, double _h, double
    _mi, double _re, double _rw, double _L, double _Keff)
24{
25
26        double a, A, B, C;
27
28        a = (_L/2)*sqrt(0.5 + (sqrt(0.25 + (pow((2*_re)/_L, 4)))));
29        A = (2*M_PI*_k*_h)/_mi;
30        B = log((a + (sqrt(pow(a, 2) - pow(_L/2, 2))))/(_L/2));
31        C = (_h*_Keff/_L)*log(_h*_Keff/(2*_rw));
32
33        IP = A / (B + C);
34
35        return IP;
```

```
36
37}
```

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

Listing 6.17: Arquivo de cabeçalho da classe CJoshiAnisotropico.

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

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

Listing 6.18: Arquivo de implementação da classe CJoshiAnisotropico.

```
1#define _USE_MATH_DEFINES
2#include <math.h>
3
4#include "CJoshiAnisotropico.h"
5
6double CJoshiAnisotropico::CalcJoshi(double _k, double _h, double
    _mi, double _re, double _rw, double _L, double _Keff)
7{
8
9        double a, A, B, C;
10
11       a = (_L/2)*sqrt(0.5 + (sqrt(0.25 + (pow((2*_re)/_L, 4)))));
12       A = (2*M_PI*_k*_h)/_mi;
13       B = log((a + (sqrt(pow(a, 2) - pow(_L/2, 2))))/(_L/2));
```

```
14          C = (_h*_Keff/_L)*log(_h*_Keff/(2*_rw));
15
16          IP = A / (B + C);
17
18          return IP;
19
20}
```

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

Listing 6.19: Arquivo de cabeçalho da classe CRenardDupuy.

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

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

Listing 6.20: Arquivo de implementação da classe CRenardDupuy.

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

```
7                 {
8
9                         double a, X, A, rw, B, C, c;
10
11                        a = (_L/2)*sqrt(0.5 + (sqrt(0.25 + (pow((2*
                              _re)/_L, 4)))));
12                        X = (2*a)/_L;
13                        A = (2*M_PI*_k*_h)/(_mi);
14                        rw = ((1 + _Keff)/(2*_Keff))*_rw;
15                        c = (_h)/(2*M_PI*rw);
16                        B = (((_Keff*_h)/_L))*(log(c));
17                        C = 1/((acosh(X)) + B);
18
19
20                        IP = A*(C);
21
22                        return IP;
23
24                }
```

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

Listing 6.21: Arquivo de cabeçalho da classe CShedid.

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

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

Listing 6.22: Arquivo de implementação da classe CShedid.

```
#define _USE_MATH_DEFINES
#include <cmath>

#include "CShedid.h"

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

        double A, B, c, D, C;
        if (_L> 0 && _L<=1000)
                C= 270;
        else if (_L > 1000)
                C= 470 - .2*_L;

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

        IP = A/D;

        return IP;
}
```

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

Listing 6.23: Arquivo de cabeçalho da classe CSimulador.

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

#include "CCalcIP.h"
#include "CDadosFluido.h"
#include "CDadosReservatorio.h"
#include "CDadosReservatorioVertical.h"
#include "CDadosPoco.h"
#include "CBorisov.h"
#include "CGiger.h"
#include "CJoshi.h"
```

```
14#include "CRenardDupuy.h"
15#include "CShedid.h"
16#include "CGnuplot.h"
17
18class CSimulador
19{
20
21        public:
22
23                CDadosFluido fluido;
24                CDadosReservatorio reservatorio;
25                CDadosReservatorioVertical reservatorioV;
26                CDadosPoco poco;
27                CBorisov borisov;
28                CGiger giger;
29                CJoshi joshi;
30                CRenardDupuy dupuy;
31                CShedid shedid;
32                Gnuplot plot;
33
34                CSimulador(){};
35
36                void EntradaDados();
37                void Plot(std::vector <double> _plot);
38                void Executar();
39
40                ~CSimulador(){};
41};
42
43#endif
```

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

Listing 6.24: Arquivo de implementação da classe CSimulador.

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

```
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 << "
               ##################################################################
               " << endl;
71          cout << "#                                                              
               
               
               #" << endl;
72          cout << "#                                    Dados estao
               corretos? 1 -sim   | 2- nao                                  
               #" << endl;
73          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;
74          cout << "#                                                              
               
```

```
            ␣␣␣␣␣␣␣␣␣␣␣␣#" << 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 << "
               ##################################################################
               " << 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 << "
               ##################################################################
               " << endl;
103         cout << "#␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
               ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
```

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

}

void CSimulador::Plot(vector <double> _plot)
{


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

        plot.set_xlabel("Modelo");
```

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

        plot.plot_x(_plot);
        _getch();
}

void CSimulador::Executar()
{

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

```

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

        int opt;
        bool tst=true;

        cin >> opt;

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

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

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


                string arquivoSaida;

                cin.get();
                getline(cin, arquivoSaida);
```

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

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

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


                Plot(_plot);

                out.close();

                system("gnusave.png");

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

        tst = true;

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

```
241         tst=false;
242         while(tst);
243         }
244
245
246 }
```

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

Listing 6.25: 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 compatibilidadde com Win32
14 //    por V. Chyzhdzenka (20/05/03)
15 // 3. Novos métodos membros, correcoes para compatibilidade com
    Win32 e Linux
16 //    por M. Burgis (10/03/08)
17 // 4. Traducao para Portugues, documentacao - javadoc/doxygen,
18 //    e modificacoes na interface (adicao 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 padrao C++, isto e, usar sobrecarga nas chamadas.
```

```
25 //   (v2)
26 //   Criar classe herdeira CGnuplot, que inclui somente a nova
       interface.
27 //   como e herdeira, o usuario vai poder usar nome 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 #ifndef 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 */
57 class GnuplotException : public std::runtime_error
58 {
59 public:
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 */
69 class Gnuplot
70 {
71 private:
72   //
       ----------------------------------------------------------------------
        Atributos
73   FILE *  gnucmd;       ///< 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 &  savetojpeg (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                                                    return set_grid
```

```
                                                    ();
293                                               else
294                                                 return
                                                       unset_grid()
                                                       ; }
295
296 /// @brief Seta taxa de amostragem das funcoes, ou dos dados de
       interpolacao.
297 Gnuplot &  set_samples (const int samples = 100);
298
299 /// @brief Seta taxa de amostragem das funcoes, ou dos dados de
       interpolacao.
300 Gnuplot &  Samples(const int samples = 100)   { return
       set_samples(samples); }
301
302 /// @brief Seta densidade de isolinhas para plotagem de funcoes
       como superficies (para plotagen 3d).
303 Gnuplot &  set_isosamples (const int isolines = 10);
304
305 /// @brief Seta densidade de isolinhas para plotagem de funcoes
       como superficies (para plotagen 3d).
306 Gnuplot &  IsoSamples (const int isolines = 10){ return
       set_isosamples(isolines); }
307
308 /// @brief Ativa remocao de linhas ocultas na plotagem de
       superficies (para plotagen 3d).
309 Gnuplot &  set_hidden3d ();
310
311 /// @brief Desativa remocao de linhas ocultas na plotagem de
       superficies (para plotagen 3d).
312 Gnuplot &  unset_hidden3d ();          // hidden3d nao e setado
       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   Gnuplot &  set_surface ();            // surface e setado por
         padrao (default)
342
343   /// @brief  Desativa a visualizacao da superficie (para plotagen
         3d).
344   Gnuplot &  unset_surface ();
```

```
345
346 /// @brief  Ativa/Desativa a visualizacao da superficie (para
        plotagen 3d).
347 Gnuplot &  Surface( int _fsurface = 1 )
348                                                 { if(fsurface =
                                                       _fsurface)
349                                                     return
                                                           set_surface
                                                           ();
350                                                   else
351                                                     return
                                                           unset_surface
                                                           ();
352                                                 }
353 /// @brief Ativa a legenda (a legenda é setada por padrao).
354 /// Posicao: inside/outside, left/center/right, top/center/bottom
        , nobox/box
355 Gnuplot &  set_legend (const std::string & position = "default");

356
357 /// @brief Desativa a legenda (a legenda é setada por padrao).
358 Gnuplot &  unset_legend ();
359
360 /// @brief Ativa/Desativa a legenda (a legenda é setada por
        padrao).
361 Gnuplot &  Legend(const std::string & position = "default")
362                                                 { return set_legend
                                                       (position); }
363
364 /// @brief Ativa/Desativa a legenda (a legenda é setada por
        padrao).
365 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 ();            // O 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                                                  { if(fxlogscale =
                                                      _fxlogscale)
468                                                    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                                             { if(fylogscale =
                                                  _fylogscale)
485                                                 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 &
                 title = "")
551                                                 { return plot_xy (
                                                        x, y,title ); }
552
553 /// @brief Plota pares x,y com barra de erro dy a partir de um
       arquivo.
```

```
554 Gnuplot &  plotfile_xy_err (const std::string & filename,
555                 const int column_x = 1,
556                 const int column_y = 2,
557                 const int column_dy = 3, const std::string &
                       title = "");
558
559 /// @brief Plota pares x,y com barra de erro dy a partir de um
       arquivo.
560 Gnuplot &  PlotFileXYErrorBar(const std::string & filename,
561                 const int column_x = 1,
562                 const int column_y = 2,
563                 const int column_dy = 3, const std::string &
                       title = "")
564                                                 { return
                                                    plotfile_xy_err
                                                    (filename,
565                                                  column_x,
                                                     column_y,
                                                     column_dy,
                                                     title ); }
566
567 /// @brief Plota pares x,y com barra de erro dy a partir de
       vetores.
568 Gnuplot &  plot_xy_err (const std::vector < double >&x,
569               const std::vector < double >&y,
570               const std::vector < double >&dy,
571               const std::string & title = "");
572
573 /// @brief Plota pares x,y com barra de erro dy a partir de
       vetores.
574 Gnuplot &  PlotVectorXYErrorBar(const std::vector < double >&x,
575               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
                            std::string & title = "")
663                                                 { return plot_image (
                                                      ucPicBuf, iWidth,
                                                      iHeight, title); }
664
665 //
    ---------------------------------------------------------------------
```

```
666  // Repete o ultimo comando de plotagem, seja plot (2D) ou splot
       (3D)
667  // Usado para visualizar plotagens, após mudar algumas opcoes de
       plotagem
668  // ou quando gerando o mesmo grafico para diferentes dispositivos
        (showonscreen, savetops)
669  Gnuplot &  replot ();
670
671  // Repete o ultimo comando de plotagem, seja plot (2D) ou splot
       (3D)
672  // Usado para visualizar plotagens, após mudar algumas opcoes de
       plotagem
673  // ou quando gerando o mesmo grafico para diferentes dispositivos
        (showonscreen, savetops)
674  Gnuplot &  Replot()                              { return replot();
       }
675
676  // Reseta uma sessao do gnuplot (próxima plotagem apaga
       definicoes previas)
677  Gnuplot &  reset_plot ();
678
679  // Reseta uma sessao do gnuplot (próxima plotagem apaga
       definicoes previas)
680  Gnuplot &  ResetPlot()                           { return reset_plot
       (); }
681
682  // Chama função reset do gnuplot
683  Gnuplot &  Reset()                       { this->cmd("reset");
       return *this; }
684
685  // Reseta uma sessao do gnuplot e seta todas as variaveis para o
       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 6.26 o arquivo de implementação da classe CGnuplot.

Listing 6.26: 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.Chyzhdzenka
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_qt.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__)
102std::string Gnuplot::terminal_std = "windows";
103// Se estamos no unix, GNU/Linux
104#elif ( defined(unix) || defined(__unix) || defined(__unix__) ) &&
    !defined(__APPLE__)
105std::string Gnuplot::terminal_std = "x11";
```

```
106// Se estamos Mac Os X
107#elif defined(__APPLE__)
108std::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//
116bool 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
156 template <typename Container>
157 void 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         // push token
```

```
176         if (j == std::string::npos)
177         {
178             container.push_back (in.substr(i));
179             return;
180         }
181         else
182             container.push_back (in.substr(i, j-i));
183 
184         // set up for next loop
185         i = j + 1;
186     }
187 
188     return;
189 }
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 // constructor: open a new session, plot a signal (x)
203 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 varibles 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 \"" << filename << ".png\"";
439     this->cmd(cmdstr.str());
440
441     return *this;
442 }
443 //
444 // saves a gnuplot session to a png file and return do on screen terminal
445 Gnuplot& Gnuplot::savetopng(const std::string &filename)
446 {
447 //                                                       // Muda o terminal
448 //     this->cmd("set term png enhanced size 1280,960");  // Tipo de terminal (tipo de arquivo)
449 //
450 //     std::ostringstream cmdstr;                           // Muda o nome do arquivo
451 //     cmdstr << "set output \"" << filename << ".png\"";  // Nome
```

```
    do arquivo
452 //     this->cmd(cmdstr.str());
453 //     this->replot();                                          //
    Replota o gráfico, agora salvando o arquivo ps
454 //                                                              //
    Retorna o terminal para o padrão janela
455 //     ShowOnScreen ();                                         // Volta
     para terminal modo janela
456 //     this->replot();                                          //
    Replota o gráfico, agora na tela
457     SaveTo(filename,"png", "enhanced␣size␣1280,960");
458
459     return *this;
460 }
461
462 //
    ------------------------------------------------------------------------


463 // saves a gnuplot session to a jpeg file and return do on screen
    terminal
464 Gnuplot& Gnuplot::savetojpeg(const std::string &filename)
465 {
466                                                              // Muda o
                                                                    terminal
467 //     this->cmd("set term jpeg enhanced size 1280,960");   // Tipo
     de terminal (tipo de arquivo)
468 //
469 //     std::ostringstream cmdstr;                               // Muda
    o nome do arquivo
470 //     cmdstr << "set output \"" << filename << ".jpeg\"";   // Nome
     do arquivo
471 //     this->cmd(cmdstr.str());
472 //     this->replot();                                          //
    Replota o gráfico, agora salvando o arquivo ps
473 //                                                              //
    Retorna o terminal para o padrão janela
474 //     ShowOnScreen ();                                         // Volta
     para terminal modo janela
475 //     this->replot();                                          //
    Replota o gráfico, agora na tela
476     SaveTo(filename,"jpeg", "enhanced␣size␣1280,960");
477
```

```
478     return *this;
479 }
480 
481 
482 //
    ------------------------------------------------------------------------
483 // saves a gnuplot session to spectific terminal and output file
484 // @filename: name of disc file
485 // @terminal_type: type of terminal
486 // @flags: aditional information specitif to terminal type
487 // Ex:
488 // grafico.SaveTo("pressao_X_temperatura","png", "enhanced size
    1280,960");
489 // grafico.TerminalType("png").SaveFile(pressao_X_temperatura);
    pense nisso?
490 Gnuplot& Gnuplot::SaveTo(const std::string &filename,const std::
    string &terminal_type, std::string flags)
491 {                                                          // Muda o
    terminal
492     this->cmd("set␣term␣" + terminal_type + "␣" + flags);    //
        Tipo de terminal (tipo de arquivo) e flags adicionais
493     std::ostringstream cmdstr;                             // Muda o
        nome do arquivo
494     cmdstr << "set␣output␣\"" << filename << "." << terminal_type
        << "\"";
495     this->cmd(cmdstr.str());
496     this->replot();                                        // Replota
        o gráfico, agora salvando o arquivo ps
497                                                            // Retorna
                                                               o
                                                               terminal
                                                                para o
                                                               padrão
                                                               janela
498     ShowOnScreen ();
499     this->replot();                                        // Replota
        o gráfico, agora na tela
500 
501     return *this;
502 }
503 
```

```
504 //
    ------------------------------------------------------------------------

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

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

526 // Turns grid on
527 Gnuplot& Gnuplot::set_grid()
528 {
529     this->cmd("set␣grid");
530
531     return *this;
532 }
533
534 //
    ------------------------------------------------------------------------

535 // Turns grid off
536 Gnuplot& Gnuplot::unset_grid()
537 {
```

```
538     this->cmd("unset␣grid");
539 
540     return *this;
541 }
542 
543 //
    ------------------------------------------------------------------------

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

556 // turns on log scaling for the y axis
557 Gnuplot& Gnuplot::set_ylogscale(const double base)
558 {
559     std::ostringstream cmdstr;
560 
561     cmdstr << "set␣logscale␣y␣" << base;
562     this->cmd(cmdstr.str());
563 
564     return *this;
565 }
566 
567 //
    ------------------------------------------------------------------------

568 // turns on log scaling for the z axis
569 Gnuplot& Gnuplot::set_zlogscale(const double base)
570 {
571     std::ostringstream cmdstr;
572 
573     cmdstr << "set␣logscale␣z␣" << base;
```

```
574     this->cmd(cmdstr.str());
575
576     return *this;
577 }
578
579 //
      ------------------------------------------------------------------------
580 // turns off log scaling for the x axis
581 Gnuplot& Gnuplot::unset_xlogscale()
582 {
583     this->cmd("unset␣logscale␣x");
584     return *this;
585 }
586
587 //
      ------------------------------------------------------------------------
588 // turns off log scaling for the y axis
589 Gnuplot& Gnuplot::unset_ylogscale()
590 {
591     this->cmd("unset␣logscale␣y");
592     return *this;
593 }
594
595 //
      ------------------------------------------------------------------------
596 // turns off log scaling for the z axis
597 Gnuplot& Gnuplot::unset_zlogscale()
598 {
599     this->cmd("unset␣logscale␣z");
600     return *this;
601 }
602
603
604 //
      ------------------------------------------------------------------------
605 // scales the size of the points used in plots
606 Gnuplot& Gnuplot::set_pointsize(const double pointsize)
607 {
```

```
608     std::ostringstream cmdstr;
609     cmdstr << "set␣pointsize␣" << pointsize;
610     this->cmd(cmdstr.str());
611 
612     return *this;
613 }
614 
615 //
    ------------------------------------------------------------------------

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

627 // set isoline density (grid) for plotting functions as surfaces
628 Gnuplot& Gnuplot::set_isosamples(const int isolines)
629 {
630     std::ostringstream cmdstr;
631     cmdstr << "set␣isosamples␣" << isolines;
632     this->cmd(cmdstr.str());
633 
634     return *this;
635 }
636 
637 //
    ------------------------------------------------------------------------

638 // enables hidden line removal for surface plotting
639 Gnuplot& Gnuplot::set_hidden3d()
640 {
641     this->cmd("set␣hidden3d");
642 
643     return *this;
```

```
644 }
645
646 //
      ---------------------------------------------------------------------
647 // disables hidden line removal for surface plotting
648 Gnuplot& Gnuplot::unset_hidden3d()
649 {
650     this->cmd("unset␣hidden3d");
651
652     return *this;
653 }
654
655 //
      ---------------------------------------------------------------------
656 // enables contour drawing for surfaces set contour {base | surface
      | both}
657 Gnuplot& Gnuplot::set_contour(const std::string &position)
658 {
659     if (position.find("base")    == std::string::npos  &&
660         position.find("surface") == std::string::npos  &&
661         position.find("both")    == std::string::npos  )
662     {
663         this->cmd("set␣contour␣base");
664     }
665     else
666     {
667         this->cmd("set␣contour␣" + position);
668     }
669
670     return *this;
671 }
672
673 //
      ---------------------------------------------------------------------
674 // disables contour drawing for surfaces
675 Gnuplot& Gnuplot::unset_contour()
676 {
677     this->cmd("unset␣contour");
678
```

```
679     return *this;
680 }
681
682 //
    ------------------------------------------------------------------------

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

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

701 // Sets the title of a gnuplot session
702 Gnuplot& Gnuplot::set_title(const std::string &title)
703 {
704     std::ostringstream cmdstr;
705
706     cmdstr << "set␣title␣\"" << title << "\"";
707     this->cmd(cmdstr.str());
708
709     return *this;
710 }
711
712 //
    ------------------------------------------------------------------------
```

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

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

735 // set the ylabel
736 Gnuplot& Gnuplot::set_ylabel(const std::string &label)
737 {
738     std::ostringstream cmdstr;
739 
740     cmdstr << "set␣ylabel␣\"" << label << "\"";
741     this->cmd(cmdstr.str());
742 
743     return *this;
744 }
745 
746 //
      ---------------------------------------------------------------------------

747 // set the zlabel
748 Gnuplot& Gnuplot::set_zlabel(const std::string &label)
```

```
749 {
750     std::ostringstream cmdstr;
751 
752     cmdstr << "set␣zlabel␣\"" << label << "\"";
753     this->cmd(cmdstr.str());
754 
755     return *this;
756 }
757 
758 //
      ------------------------------------------------------------------------
759 // set range
760 // set the xrange
761 Gnuplot& Gnuplot::set_xrange(const int iFrom,
762                              const int iTo)
763 {
764     std::ostringstream cmdstr;
765 
766     cmdstr << "set␣xrange[" << iFrom << ":" << iTo << "]";
767     this->cmd(cmdstr.str());
768 
769     return *this;
770 }
771 
772 //
      ------------------------------------------------------------------------
773 // set autoscale x
774 Gnuplot& Gnuplot::set_xautoscale()
775 {
776     this->cmd("set␣xrange␣restore");
777     this->cmd("set␣autoscale␣x");
778 
779     return *this;
780 }
781 
782 //
      ------------------------------------------------------------------------
783 // set the yrange
784 Gnuplot& Gnuplot::set_yrange(const int iFrom, const int iTo)
```

```
785 {
786     std::ostringstream cmdstr;
787 
788     cmdstr << "set␣yrange[" << iFrom << ":" << iTo << "]";
789     this->cmd(cmdstr.str());
790 
791     return *this;
792 }
793 
794 //
    ------------------------------------------------------------------------

795 // set autoscale y
796 Gnuplot& Gnuplot::set_yautoscale()
797 {
798     this->cmd("set␣yrange␣restore");
799     this->cmd("set␣autoscale␣y");
800 
801     return *this;
802 }
803 
804 //
    ------------------------------------------------------------------------

805 // set the zrange
806 Gnuplot& Gnuplot::set_zrange(const int iFrom,
807                              const int iTo)
808 {
809     std::ostringstream cmdstr;
810 
811     cmdstr << "set␣zrange[" << iFrom << ":" << iTo << "]";
812     this->cmd(cmdstr.str());
813 
814     return *this;
815 }
816 
817 //
    ------------------------------------------------------------------------

818 // set autoscale z
819 Gnuplot& Gnuplot::set_zautoscale()
820 {
```

```
821     this->cmd("set␣zrange␣restore");
822     this->cmd("set␣autoscale␣z");
823 
824     return *this;
825 }
826 
827 //
    ------------------------------------------------------------------------

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

841 // Plots a linear equation y=ax+b (where you supply the
842 // slope a and intercept b)
843 Gnuplot& Gnuplot::plot_slope(const double a,
844                              const double b,
845                              const std::string &title)
846 {
847     std::ostringstream cmdstr;
848 
849     // command to be sent to gnuplot
850     if (this->nplots > 0  &&  this->two_dim == true)
851         cmdstr << "replot␣";
852     else
853         cmdstr << "plot␣";
854 
855     cmdstr << a << "␣*␣x␣+␣" << b << "␣title␣\"";
856 
857     if (title == "")
858         cmdstr << "f(x)␣=␣" << a << "␣*␣x␣+␣" << b;
```

```
859     else
860         cmdstr << title;
861 
862     cmdstr << "\" with " << this->pstyle;
863 
864     // Do the actual plot
865     this->cmd(cmdstr.str());
866 
867     return *this;
868 }
869 
870 //
    ---------------------------------------------------------------------------

871 // Plot an equation which is supplied as a std::string y=f(x) (only
    f(x) expected)
872 Gnuplot& Gnuplot::plot_equation(const std::string &equation,
873                                 const std::string &title)
874 {
875     std::ostringstream cmdstr;
876 
877     // command to be sent to gnuplot
878     if (this->nplots > 0   &&   this->two_dim == true)
879         cmdstr << "replot ";
880     else
881         cmdstr << "plot ";
882 
883     cmdstr << equation << " title \"";
884 
885     if (title == "")
886         cmdstr << "f(x) = " << equation;
887     else
888         cmdstr << title;
889 
890     cmdstr << "\" with " << this->pstyle;
891 
892     // Do the actual plot
893     this->cmd(cmdstr.str());
894 
895     return *this;
896 }
897 
```

```
898 //
    ------------------------------------------------------------------------

899 // plot an equation supplied as a std::string y=(x)
900 Gnuplot& Gnuplot::plot_equation3d(const std::string &equation,
901                                   const std::string &title)
902 {
903     std::ostringstream cmdstr;
904 
905     // command to be sent to gnuplot
906     if (this->nplots > 0   &&   this->two_dim == false)
907         cmdstr << "replot ";
908     else
909         cmdstr << "splot ";
910 
911     cmdstr << equation << " title \"";
912 
913     if (title == "")
914         cmdstr << "f(x,y) = " << equation;
915     else
916         cmdstr << title;
917 
918     cmdstr << "\" with " << this->pstyle;
919 
920     // Do the actual plot
921     this->cmd(cmdstr.str());
922 
923     return *this;
924 }
925 
926 //
    ------------------------------------------------------------------------

927 // Plots a 2d graph from a list of doubles (x) saved in a file
928 Gnuplot& Gnuplot::plotfile_x(const std::string &filename,
929                              const int column,
930                              const std::string &title)
931 {
932     // check if file exists
933     if( !(Gnuplot::file_exists(filename,4)) ) // check existence
        and read permission
934     {
```

```
935         std::ostringstream except;
936         if( !(Gnuplot::file_exists(filename,0)) ) // check
                existence
937             except << "File␣\"" << filename << "\"␣does␣not␣exist";
938         else
939             except << "No␣read␣permission␣for␣File␣\"" << filename
                    << "\"";
940         throw GnuplotException( except.str() );
941         return *this;
942     }
943
944     std::ostringstream cmdstr;
945
946     // command to be sent to gnuplot
947     if (this->nplots > 0   &&   this->two_dim == true)
948         cmdstr << "replot␣";
949     else
950         cmdstr << "plot␣";
951
952     cmdstr << "\"" << filename << "\"␣using␣" << column;
953
954     if (title == "")
955         cmdstr << "␣notitle␣";
956     else
957         cmdstr << "␣title␣\"" << title << "\"␣";
958
959     if(smooth == "")
960         cmdstr << "with␣" << this->pstyle;
961     else
962         cmdstr << "smooth␣" << this->smooth;
963
964     // Do the actual plot
965     this->cmd(cmdstr.str()); //nplots++; two_dim = true;  already
        in this->cmd();
966
967     return *this;
968 }
969
970 //
    ------------------------------------------------------------------------

971 // Plots a 2d graph from a list of doubles: x
```

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

999 // Plots a 2d graph from a list of doubles (x y) saved in a file
1000 Gnuplot& Gnuplot::plotfile_xy(const std::string &filename,
1001                               const int column_x,
1002                               const int column_y,
1003                               const std::string &title)
1004 {
1005     // check if file exists
1006     if( !(Gnuplot::file_exists(filename,4)) ) // check existence
        and read permission
1007     {
1008         std::ostringstream except;
1009         if( !(Gnuplot::file_exists(filename,0)) ) // check
            existence
```

```
1010            except << "File␣\"" << filename << "\"␣does␣not␣exist";
1011        else
1012            except << "No␣read␣permission␣for␣File␣\"" << filename
                    << "\"";
1013        throw GnuplotException( except.str() );
1014        return *this;
1015    }
1016
1017    std::ostringstream cmdstr;
1018
1019    // command to be sent to gnuplot
1020    if (this->nplots > 0  &&  this->two_dim == true)
1021        cmdstr << "replot␣";
1022    else
1023        cmdstr << "plot␣";
1024
1025    cmdstr << "\"" << filename << "\"␣using␣" << column_x << ":" <<
        column_y;
1026
1027    if (title == "")
1028        cmdstr << "␣notitle␣";
1029    else
1030        cmdstr << "␣title␣\"" << title << "\"␣";
1031
1032    if(smooth == "")
1033        cmdstr << "with␣" << this->pstyle;
1034    else
1035        cmdstr << "smooth␣" << this->smooth;
1036
1037    // Do the actual plot
1038    this->cmd(cmdstr.str());
1039
1040    return *this;
1041}
1042
1043//
    ------------------------------------------------------------------------


1044// Plots a 2d graph from a list of doubles: x y
1045Gnuplot& Gnuplot::plot_xy(const std::vector<double> &x,
1046                          const std::vector<double> &y,
1047                          const std::string &title)
```

```
1048 {
1049     if (x.size() == 0 || y.size() == 0)
1050     {
1051         throw GnuplotException("std::vectors too small");
1052         return *this;
1053     }
1054 
1055     if (x.size() != y.size())
1056     {
1057         throw GnuplotException("Length of the std::vectors differs"
                ");
1058         return *this;
1059     }
1060 
1061     std::ofstream tmp;
1062     std::string name = create_tmpfile(tmp);
1063     if (name == "")
1064         return *this;
1065 
1066     // write the data to file
1067     for (unsigned int i = 0; i < x.size(); i++)
1068         tmp << x[i] << " " << y[i] << std::endl;
1069 
1070     tmp.flush();
1071     tmp.close();
1072 
1073     this->plotfile_xy(name, 1, 2, title);
1074 
1075     return *this;
1076 }
1077 
1078 //
      ---------------------------------------------------------------------------

1079 // Plots a 2d graph with errorbars from a list of doubles (x y dy)
      saved in a file
1080 Gnuplot& Gnuplot::plotfile_xy_err(const std::string &filename,
1081                                   const int column_x,
1082                                   const int column_y,
1083                                   const int column_dy,
1084                                   const std::string &title)
1085 {
```

```
1086     // check if file exists
1087     if( !(Gnuplot::file_exists(filename,4)) ) // check existence
            and read permission
1088     {
1089         std::ostringstream except;
1090         if( !(Gnuplot::file_exists(filename,0)) ) // check
               existence
1091             except << "File␣\"" << filename << "\"␣does␣not␣exist";
1092         else
1093             except << "No␣read␣permission␣for␣File␣\"" << filename
                   << "\"";
1094         throw GnuplotException( except.str() );
1095         return *this;
1096     }
1097
1098     std::ostringstream cmdstr;
1099
1100     // command to be sent to gnuplot
1101     if (this->nplots > 0  &&  this->two_dim == true)
1102         cmdstr << "replot␣";
1103     else
1104         cmdstr << "plot␣";
1105
1106     cmdstr << "\"" << filename << "\"␣using␣" << column_x << ":" <<
         column_y;
1107
1108     if (title == "")
1109         cmdstr << "␣notitle␣";
1110     else
1111         cmdstr << "␣title␣\"" << title << "\"␣";
1112
1113     cmdstr << "with␣" << this->pstyle << ",␣\"" << filename << "\"␣
        using␣"
1114             << column_x << ":" << column_y << ":" << column_dy << "␣
                notitle␣with␣errorbars";
1115
1116     // Do the actual plot
1117     this->cmd(cmdstr.str());
1118
1119     return *this;
1120 }
1121
```

```
1122 //
     ------------------------------------------------------------------------

1123 // plot x,y pairs with dy errorbars
1124 Gnuplot& Gnuplot::plot_xy_err(const std::vector<double> &x,
1125                               const std::vector<double> &y,
1126                               const std::vector<double> &dy,
1127                               const std::string &title)
1128 {
1129     if (x.size() == 0 || y.size() == 0 || dy.size() == 0)
1130     {
1131         throw GnuplotException("std::vectors too small");
1132         return *this;
1133     }
1134
1135     if (x.size() != y.size() || y.size() != dy.size())
1136     {
1137         throw GnuplotException("Length of the std::vectors differs"
                );
1138         return *this;
1139     }
1140
1141     std::ofstream tmp;
1142     std::string name = create_tmpfile(tmp);
1143     if (name == "")
1144         return *this;
1145
1146     // write the data to file
1147     for (unsigned int i = 0; i < x.size(); i++)
1148         tmp << x[i] << " " << y[i] << " " << dy[i] << std::endl;
1149
1150     tmp.flush();
1151     tmp.close();
1152
1153     // Do the actual plot
1154     this->plotfile_xy_err(name, 1, 2, 3, title);
1155
1156     return *this;
1157 }
1158
1159 //
     ------------------------------------------------------------------------
```

```
1160// Plots a 3d graph from a list of doubles (x y z) saved in a file
1161Gnuplot& Gnuplot::plotfile_xyz(const std::string &filename,
1162                                const int column_x,
1163                                const int column_y,
1164                                const int column_z,
1165                                const std::string &title)
1166{
1167
1168    // check if file exists
1169    if( !(Gnuplot::file_exists(filename,4)) ) // check existence
           and read permission
1170    {
1171        std::ostringstream except;
1172        if( !(Gnuplot::file_exists(filename,0)) ) // check
              existence
1173            except << "File␣\"" << filename << "\"␣does␣not␣exist";
1174        else
1175            except << "No␣read␣permission␣for␣File␣\"" << filename
                  << "\"";
1176        throw GnuplotException( except.str() );
1177        return *this;
1178    }
1179
1180    std::ostringstream cmdstr;
1181
1182    // command to be sent to gnuplot
1183    if (this->nplots > 0  &&  this->two_dim == false)
1184        cmdstr << "replot␣";
1185    else
1186        cmdstr << "splot␣";
1187
1188    cmdstr << "\"" << filename << "\"␣using␣" << column_x << ":" <<
        column_y << ":" << column_z;
1189
1190    if (title == "")
1191        cmdstr << "␣notitle␣with␣" << this->pstyle;
1192    else
1193        cmdstr << "␣title␣\"" << title << "\"␣with␣" << this->
              pstyle;
1194
1195    // Do the actual plot
```

```
1196     this->cmd(cmdstr.str());
1197
1198     return *this;
1199 }
1200
1201 //
      ------------------------------------------------------------------------

1202 // Plots a 3d graph from a list of doubles: x y z
1203 Gnuplot& Gnuplot::plot_xyz(const std::vector<double> &x,
1204                            const std::vector<double> &y,
1205                            const std::vector<double> &z,
1206                            const std::string &title)
1207 {
1208     if (x.size() == 0 || y.size() == 0 || z.size() == 0)
1209     {
1210         throw GnuplotException("std::vectors too small");
1211         return *this;
1212     }
1213
1214     if (x.size() != y.size() || x.size() != z.size())
1215     {
1216         throw GnuplotException("Length of the std::vectors differs"
              );
1217         return *this;
1218     }
1219
1220
1221     std::ofstream tmp;
1222     std::string name = create_tmpfile(tmp);
1223     if (name == "")
1224         return *this;
1225
1226     // write the data to file
1227     for (unsigned int i = 0; i < x.size(); i++)
1228     {
1229         tmp << x[i] << " " << y[i] << " " << z[i] <<std::endl;
1230     }
1231
1232     tmp.flush();
1233     tmp.close();
1234
```

```
1235
1236     this->plotfile_xyz(name, 1, 2, 3, title);
1237
1238     return *this;
1239 }
1240
1241
1242
1243 //
      ------------------------------------------------------------------------
1244 /// *  note that this function is not valid for versions of GNUPlot
       below 4.2
1245 Gnuplot& Gnuplot::plot_image(const unsigned char * ucPicBuf,
1246                              const int iWidth,
1247                              const int iHeight,
1248                              const std::string &title)
1249 {
1250     std::ofstream tmp;
1251     std::string name = create_tmpfile(tmp);
1252     if (name == "")
1253         return *this;
1254
1255     // write the data to file
1256     int iIndex = 0;
1257     for(int iRow = 0; iRow < iHeight; iRow++)
1258     {
1259         for(int iColumn = 0; iColumn < iWidth; iColumn++)
1260         {
1261             tmp << iColumn << "␣" << iRow  << "␣" << static_cast<
                    float>(ucPicBuf[iIndex++]) << std::endl;
1262         }
1263     }
1264
1265     tmp.flush();
1266     tmp.close();
1267
1268
1269     std::ostringstream cmdstr;
1270
1271     // command to be sent to gnuplot
1272     if (this->nplots > 0   &&   this->two_dim == true)
```

```
1273            cmdstr << "replot␣";
1274        else
1275            cmdstr << "plot␣";
1276
1277        if (title == "")
1278            cmdstr << "\"" << name << "\"␣with␣image";
1279        else
1280            cmdstr << "\"" << name << "\"␣title␣\"" << title << "\"␣
                with␣image";
1281
1282        // Do the actual plot
1283        this->cmd(cmdstr.str());
1284
1285        return *this;
1286    }
1287
1288    //
        ------------------------------------------------------------------------

1289    // Sends a command to an active gnuplot session
1290    Gnuplot& Gnuplot::cmd(const std::string &cmdstr)
1291    {
1292        if( !(this->valid) )
1293        {
1294            return *this;
1295        }
1296
1297        // int fputs ( const char * str, FILE * stream );
1298        // writes the string str to the stream.
1299        // The function begins copying from the address specified (str)
               until it reaches the
1300        // terminating null character ('\0'). This final null-character
               is not copied to the stream.
1301        fputs( (cmdstr+"\n").c_str(), this->gnucmd );
1302
1303        // int fflush ( FILE * stream );
1304        // If the given stream was open for writing and the last i/o
           operation was an output operation,
1305        // any unwritten data in the output buffer is written to the
           file.
1306        // If the argument is a null pointer, all open files are
           flushed.
```

```
1307     // The stream remains open after this call.
1308     fflush(this->gnucmd);
1309 
1310 
1311     if( cmdstr.find("replot") != std::string::npos )
1312     {
1313         return *this;
1314     }
1315     else if( cmdstr.find("splot") != std::string::npos )
1316     {
1317         this->two_dim = false;
1318         this->nplots++;
1319     }
1320     else if( cmdstr.find("plot") != std::string::npos )
1321     {
1322         this->two_dim = true;
1323         this->nplots++;
1324     }
1325     return *this;
1326 }
1327 
1328 //
      ---------------------------------------------------------------------------

1329 // Sends a command to an active gnuplot session, identical to cmd()
1330 Gnuplot& Gnuplot::operator<<(const std::string &cmdstr)
1331 {
1332     this->cmd(cmdstr);
1333     return *this;
1334 }
1335 
1336 //
      ---------------------------------------------------------------------------

1337 // Opens up a gnuplot session, ready to receive commands
1338 void Gnuplot::init()
1339 {
1340     // char * getenv ( const char * name );  get value of an
          environment variable
1341     // Retrieves a C string containing the value of the environment
           variable whose
1342     // name is specified as argument.
```

```
1343     // If the requested variable is not part of the environment
            list, the function returns a NULL pointer.
1344 #if ( defined(unix) || defined(__unix) || defined(__unix__) ) && !
     defined(__APPLE__)
1345      if (getenv("DISPLAY") == NULL)
1346      {
1347          this->valid = false;
1348          throw GnuplotException("Can't␣find␣DISPLAY␣variable");
1349      }
1350 #endif
1351
1352      // if gnuplot not available
1353      if (!Gnuplot::get_program_path())
1354      {
1355          this->valid = false;
1356          throw GnuplotException("Can't␣find␣gnuplot");
1357      }
1358
1359      // open pipe
1360      std::string tmp = Gnuplot::m_sGNUPlotPath + "/" + Gnuplot::
            m_sGNUPlotFileName;
1361
1362      // FILE *popen(const char *command, const char *mode);
1363      // The popen() function shall execute the command specified by
            the string command,
1364      // create a pipe between the calling program and the executed
            command, and
1365      // return a pointer to a stream that can be used to either read
             from or write to the pipe.
1366 #if defined(WIN32) || defined(_WIN32) || defined(__WIN32__) ||
     defined(__TOS_WIN__)
1367      this->gnucmd = _popen(tmp.c_str(),"w");
1368 #elif defined(unix) || defined(__unix) || defined(__unix__) ||
     defined(__APPLE__)
1369      this->gnucmd = popen(tmp.c_str(),"w");
1370 #endif
1371
1372      // popen() shall return a pointer to an open stream that can be
             used to read or write to the pipe.
1373      // Otherwise, it shall return a null pointer and may set errno
            to indicate the error.
1374      if (!this->gnucmd)
```

```
1375     {
1376         this->valid = false;
1377         throw GnuplotException("Couldn't open connection to gnuplot
                ");
1378     }
1379
1380     this->nplots = 0;
1381     this->valid = true;
1382     this->smooth = "";
1383
1384     //set terminal type
1385     this->showonscreen();
1386
1387     return;
1388 }
1389
1390 //
      ---------------------------------------------------------------------------

1391 // Find out if a command lives in m_sGNUPlotPath or in PATH
1392 bool Gnuplot::get_program_path()
1393 {
1394     // first look in m_sGNUPlotPath for Gnuplot
1395     std::string tmp = Gnuplot::m_sGNUPlotPath + "/" + Gnuplot::
            m_sGNUPlotFileName;
1396
1397 #if defined(WIN32) || defined(_WIN32) || defined(__WIN32__) ||
      defined(__TOS_WIN__)
1398     if ( Gnuplot::file_exists(tmp,0) ) // check existence
1399 #elif defined(unix) || defined(__unix) || defined(__unix__) ||
      defined(__APPLE__)
1400     if ( Gnuplot::file_exists(tmp,1) ) // check existence and
            execution permission
1401 #endif
1402     {
1403         return true;
1404     }
1405
1406     // second look in PATH for Gnuplot
1407     char *path;
1408     // Retrieves a C string containing the value of the environment
          variable PATH
```

```
1409     path = getenv("PATH");
1410
1411     if (path == NULL)
1412     {
1413         throw GnuplotException("Path is not set");
1414         return false;
1415     }
1416     else
1417     {
1418         std::list<std::string> ls;
1419         //split path (one long string) into list ls of strings
1420#if defined(WIN32) || defined(_WIN32) || defined(__WIN32__) ||
      defined(__TOS_WIN__)
1421         stringtok(ls,path,";");
1422#elif defined(unix) || defined(__unix) || defined(__unix__) ||
      defined(__APPLE__)
1423         stringtok(ls,path,":");
1424#endif
1425         // scan list for Gnuplot program files
1426         for (std::list<std::string>::const_iterator i = ls.begin();
                 i != ls.end(); ++i)
1427         {
1428             tmp = (*i) + "/" + Gnuplot::m_sGNUPlotFileName;
1429#if defined(WIN32) || defined(_WIN32) || defined(__WIN32__) ||
      defined(__TOS_WIN__)
1430             if ( Gnuplot::file_exists(tmp,0) ) // check existence
1431#elif defined(unix) || defined(__unix) || defined(__unix__) ||
      defined(__APPLE__)
1432             if ( Gnuplot::file_exists(tmp,1) ) // check existence
                    and execution permission
1433#endif
1434             {
1435                 Gnuplot::m_sGNUPlotPath = *i; // set m_sGNUPlotPath
1436                 return true;
1437             }
1438         }
1439
1440         tmp = "Can't find gnuplot neither in PATH nor in \"" +
                Gnuplot::m_sGNUPlotPath + "\"";
1441         throw GnuplotException(tmp);
1442
1443         Gnuplot::m_sGNUPlotPath = "";
```

```
1444            return false;
1445        }
1446    }
1447
1448    //
        ------------------------------------------------------------------------


1449    // check if file exists
1450    bool Gnuplot::file_exists(const std::string &filename, int mode)
1451    {
1452        if ( mode < 0 || mode > 7)
1453        {
1454            throw std::runtime_error("In␣function␣\"Gnuplot::
                    file_exists\":␣mode␣has␣to␣be␣an␣integer␣between␣0␣and␣7
                    ");
1455            return false;
1456        }
1457
1458        // int _access(const char *path, int mode);
1459        //  returns 0 if the file has the given mode,
1460        //  it returns -1 if the named file does not exist or is not
                accessible in the given mode
1461        // mode = 0 (F_OK) (default): checks file for existence only
1462        // mode = 1 (X_OK): execution permission
1463        // mode = 2 (W_OK): write permission
1464        // mode = 4 (R_OK): read permission
1465        // mode = 6       : read and write permission
1466        // mode = 7       : read, write and execution permission
1467    #if defined(WIN32) || defined(_WIN32) || defined(__WIN32__) ||
        defined(__TOS_WIN__)
1468        if (_access(filename.c_str(), mode) == 0)
1469    #elif defined(unix) || defined(__unix) || defined(__unix__) ||
        defined(__APPLE__)
1470        if (access(filename.c_str(), mode) == 0)
1471    #endif
1472        {
1473            return true;
1474        }
1475        else
1476        {
1477            return false;
1478        }
```

```
1479
1480}
1481
1482//
      ------------------------------------------------------------------------

1483// Opens a temporary file
1484std::string Gnuplot::create_tmpfile(std::ofstream &tmp)
1485{
1486#if defined(WIN32) || defined(_WIN32) || defined(__WIN32__) ||
      defined(__TOS_WIN__)
1487     char name[] = "gnuplotiXXXXXX"; //tmp file in working directory
1488#elif defined(unix) || defined(__unix) || defined(__unix__) ||
      defined(__APPLE__)
1489     char name[] = "/tmp/gnuplotiXXXXXX"; // tmp file in /tmp
1490#endif
1491
1492     // check if maximum number of temporary files reached
1493     if (Gnuplot::tmpfile_num == GP_MAX_TMP_FILES - 1)
1494     {
1495          std::ostringstream except;
1496          except << "Maximum␣number␣of␣temporary␣files␣reached␣(" <<
                GP_MAX_TMP_FILES
1497               << "):␣cannot␣open␣more␣files" << std::endl;
1498
1499          throw GnuplotException( except.str() );
1500          return "";
1501     }
1502
1503     // int mkstemp(char *name);
1504     // shall replace the contents of the string pointed to by "name
          " by a unique filename,
1505     // and return a file descriptor for the file open for reading
          and writing.
1506     // Otherwise, -1 shall be returned if no suitable file could be
           created.
1507     // The string in template should look like a filename with six
          trailing 'X' s;
1508     // mkstemp() replaces each 'X' with a character from the
          portable filename character set.
1509     // 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()
1510
1511
1512    // open temporary files for output
1513#if defined(WIN32) || defined(_WIN32) || defined(__WIN32__) ||
    defined(__TOS_WIN__)
1514    if (_mktemp(name) == NULL)
1515#elif defined(unix) || defined(__unix) || defined(__unix__) ||
    defined(__APPLE__)
1516    if (mkstemp(name) == -1)
1517#endif
1518    {
1519        std::ostringstream except;
1520        except << "Cannot␣create␣temporary␣file␣\"" << name << "\""
               ;
1521        throw GnuplotException(except.str());
1522        return "";
1523    }
1524
1525    tmp.open(name);
1526    if (tmp.bad())
1527    {
1528        std::ostringstream except;
1529        except << "Cannot␣create␣temporary␣file␣\"" << name << "\""
               ;
1530        throw GnuplotException(except.str());
1531        return "";
1532    }
1533
1534    // Save the temporary filename
1535    this->tmpfile_list.push_back(name);
1536    Gnuplot::tmpfile_num++;
1537
1538    return name;
1539}
```

Apresenta-se na listagem 6.27 o programa main().

Listing 6.27: Arquivo de implementação da função `main()`.

```
1#include "CSimulador.h"
2
3int main (){
4
```

```
5         CSimulador simular;
6
7         simular.Executar();
8
9         return 0;
10}
```

```
1Bem vindo ao C++!
```

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.

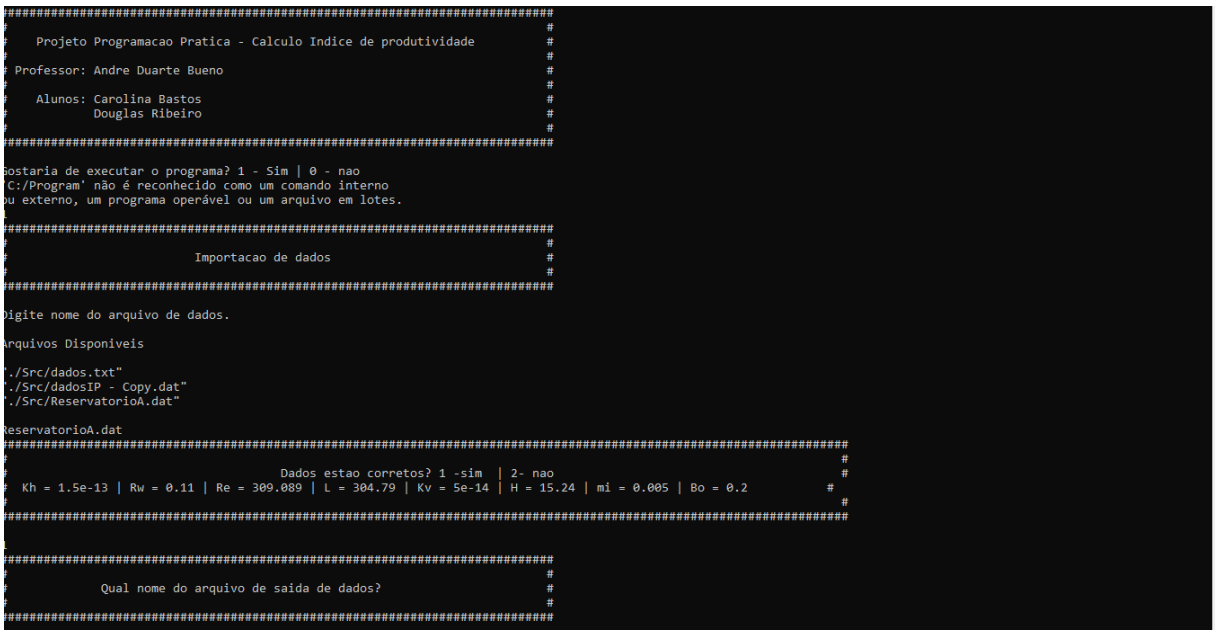

Figura 7.1: Interface com usuário do programa.

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 Sheddid.
- O programa, primeiramente, precisa ser aberto no diretório onde o código se encontra e, depois disso, compila-se o código em um programa como o visual Studio 2019, neste caso, o programa foi compilado direto utilizando-se o dev C++.
- O programa pôde ser validado com base nos resultados obtidos na monografia em que foi baseado, neste caso, todos os dados possuem saída em .txt após a simulação para cada método.

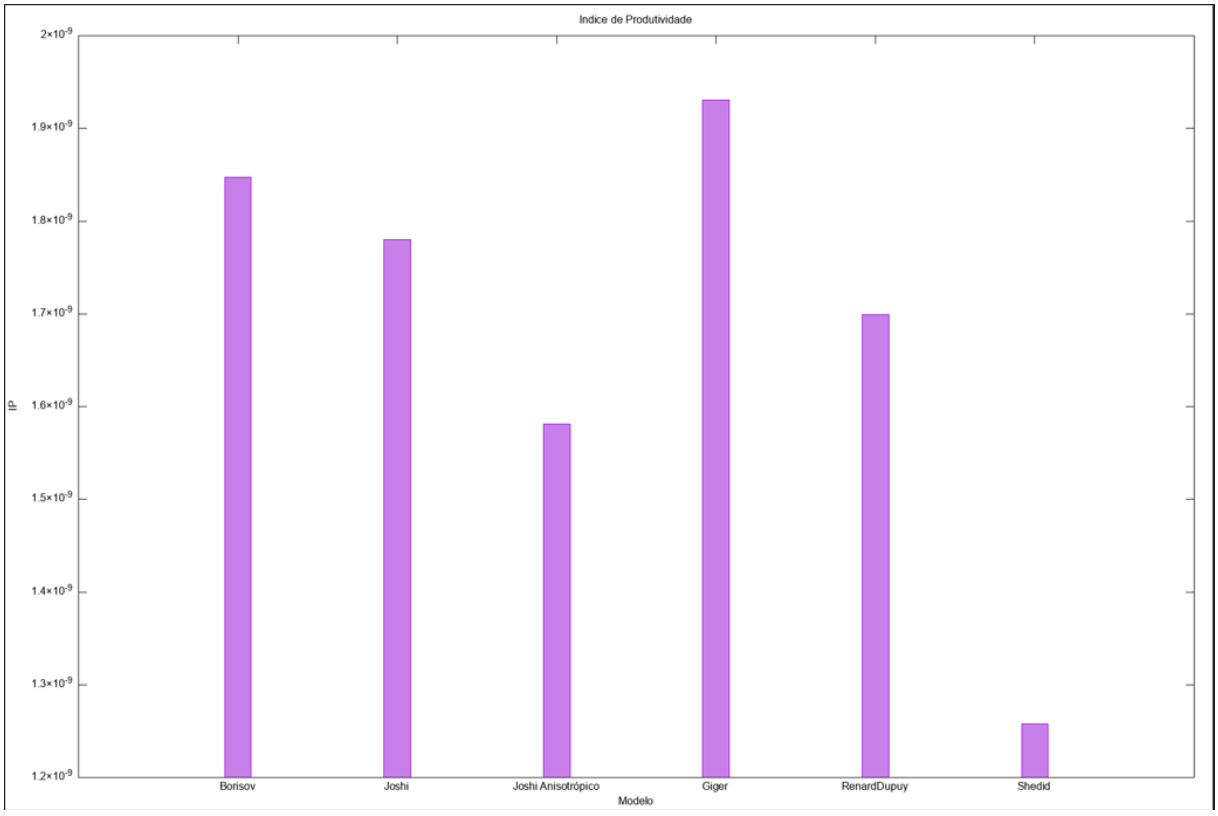

Figura 7.2: Histograma com os resultados para cada método do cálculo do Índice de Produtividade.

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 Sheddid.

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

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

8.2 Documentação para desenvolvedor

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

8.2.1 Dependências

Para compilar o software é necessário atender as seguintes dependências:

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

8.2.2 Como gerar a documentação usando doxygen

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

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

  - http://www.stack.nl/~dimitri/doxygen/

Passos para gerar a documentação usando o `doxygen`.

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

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

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

  ```
  dogygen -g
  ```

- Peça para o `doxygen` gerar a documentação.

  ```
  doxygen
  ```

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

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

  ou

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

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

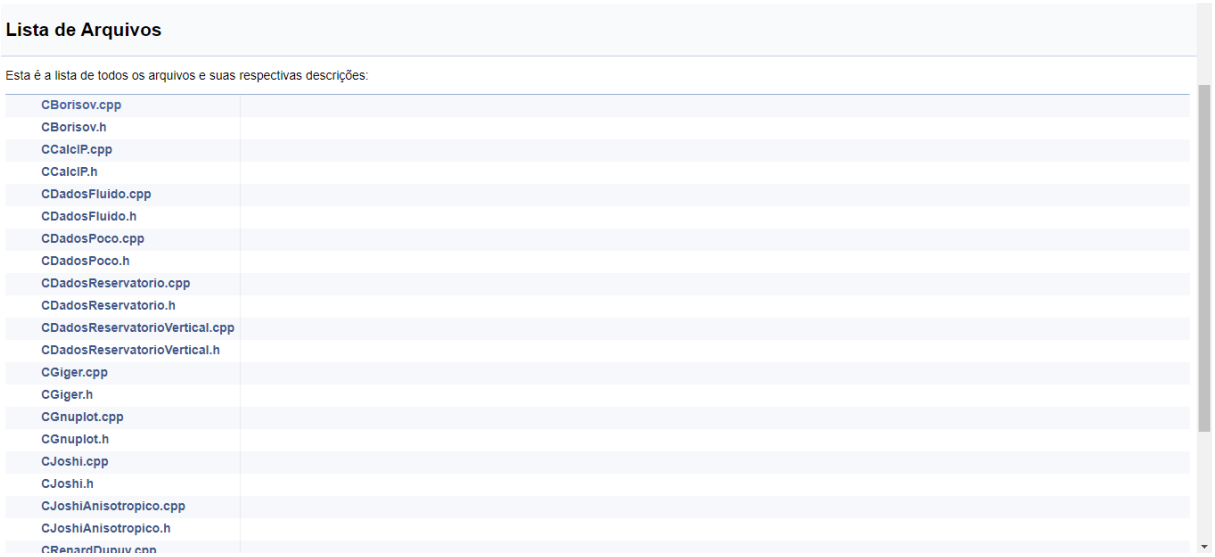

Figura 8.1: Histograma com os resultados para cada método do cálculo do Índice de Produtividade.

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. 19

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

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

[Rumbaugh et al., 1994] Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., and Lorensen, W. (1994). *Modelagem e Projetos Baseados em Objetos*. Edit. Campus, Rio de Janeiro. 19

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

Índice Remissivo