

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

PROJETO ENGENHARIA  
DESENVOLVIMENTO DO SOFTWARE  
SIMULADOR DE SOLUÇÕES ANALÍTICAS ADIMENSIONAIS DA  
EQUAÇÃO DA DIFUSIVIDADE HIDRÁULICA PARA FLUXOS  
LINEAR E RADIAL  
TRABALHO DA DISCIPLINA PROGRAMAÇÃO PRÁTICA

Versão 1:  
Letícia Fernandes Sakai  
Lucas Rodrigues Tavares  
Prof. André Duarte Bueno

MACAÉ - RJ  
Março - 2022

# Sumário

<b>1</b>	<b>Introdução</b>	<b>1</b>
1.1	Escopo do problema . . . . .	1
1.2	Objetivos . . . . .	2
<b>2</b>	<b>Especificação</b>	<b>3</b>
2.1	Nome do sistema/produto . . . . .	3
2.2	Especificação . . . . .	3
2.2.1	Requisitos funcionais . . . . .	4
2.2.2	Requisitos não funcionais . . . . .	4
2.3	Casos de uso . . . . .	4
2.3.1	Diagrama de caso de uso geral . . . . .	5
2.3.2	Diagrama de caso de uso específico . . . . .	5
<b>3</b>	<b>Elaboração</b>	<b>7</b>
3.1	Análise de domínio . . . . .	7
3.2	Formulação teórica . . . . .	8
3.2.1	Lei de Darcy . . . . .	8
3.2.2	Princípio de Conservação de Massa . . . . .	8
3.2.3	Equação da Difusividade Hidráulica . . . . .	9
3.3	Identificação de pacotes – assuntos . . . . .	11
3.4	Diagrama de pacotes – assuntos . . . . .	11
<b>4</b>	<b>AOO – Análise Orientada a Objeto</b>	<b>13</b>
4.1	Diagramas de classes . . . . .	13
4.1.1	Dicionário de classes . . . . .	13
4.2	Diagrama de seqüência – eventos e mensagens . . . . .	14
4.2.1	Diagrama de sequência geral . . . . .	14
4.3	Diagrama de comunicação – colaboração . . . . .	15
4.4	Diagrama de máquina de estado . . . . .	16
4.5	Diagrama de atividades . . . . .	17

<b>5 Projeto</b>	<b>19</b>
5.1 Projeto do sistema . . . . .	19
5.2 Projeto orientado a objeto – POO . . . . .	21
5.3 Diagrama de componentes . . . . .	22
5.4 Diagrama de implantação . . . . .	24
<b>6 Implementação</b>	<b>25</b>
6.1 Código fonte . . . . .	25
<b>7 Teste</b>	<b>124</b>
7.1 Teste 1: Regime Radial . . . . .	124
7.2 Teste 2: Regime Linear . . . . .	128
7.3 Teste 3: Autenticidade do programa . . . . .	133
<b>8 Documentação</b>	<b>137</b>
8.1 Documentação do usuário . . . . .	137
8.1.1 Como instalar o software . . . . .	137
8.1.2 Como rodar o software . . . . .	137
8.2 Documentação para desenvolvedor . . . . .	138
8.2.1 Dependências . . . . .	138
8.2.2 Como gerar a documentação usando doxygen . . . . .	138

# Capítulo 1

## Introdução

No presente projeto de engenharia desenvolve-se o *software das soluções analíticas da Equação da Difusividade Hidráulica para fluxos linear e radial*, um software aplicado à engenharia de petróleo e que utiliza o paradigma da orientação a objetos.

Este software tem como finalidade obter as soluções adimensionais da Equação da Difusividade Hidráulica para os regimes de fluxos transiente, permanente e pseudopermanente, tanto com geometria linear, quanto radial. Além disso o sistema deverá ser capaz de interpretar as funções de Bessel modificadas de primeira e segunda espécies de ordem zero, inverter numericamente essas funções utilizando o Algoritmo de Stehfest e por fim, plotar os gráficos e salvar em disco.

### 1.1 Escopo do problema

A partir da descoberta de uma acumulação de petróleo diversas informações podem ser obtidas. Das mais importantes, pode-se citar como exemplo a quantidade de hidrocarbonetos que se pode retirar dessa jazida e o tempo em que essa produção se efetuará.

Dentro da engenharia de petróleo, engenheiros de reservatório constantemente buscam solucionar problemas envolvendo fluxos monofásicos de fluidos de baixa compressibilidade que partem das equações fundamentais da mecânica dos fluidos que descrevem o transporte de líquidos em meios porosos. A principal equação que rege esse estudo do fluxo em meios porosos é a Equação da Difusividade Hidráulica e ao resolver os problemas de valor inicial e de contorno formados por essa equação, é possível obter os modelos físicos de interesse que são encontrados no campo.

A necessidade de monitoramento do comportamento do reservatório é evidente e prever como será o seu comportamento auxilia diretamente na tomada de decisões a respeito das operações realizadas e serve de base para os testes de pressão que visam a obtenção de vários parâmetros do reservatório (fator de película, volume poroso drenado, limites do reservatório, etc).

## 1.2 Objetivos

Os objetivos deste projeto de engenharia são:

- Objetivo geral:

- Desenvolver um simulador para determinar as soluções gráficas adimensionais da Equação da Difusividade Hidráulica de um reservatório de óleo a partir da pressão em função do tempo ou da posição.

- Objetivos específicos:

- Gerar gráficos de  $p_D(x_D)$  para o fluxo linear;
  - Gerar gráficos de  $p_D(t_D)$  para o fluxo radial;
  - Disponibilizar opção para salvar estes resultados em disco;
  - Realizar teste das classes;
  - Encontrar quaisquer bugs;
  - Simular problemas reais que se encontram em livros de engenharia de reservatórios e artigos;
  - Comparar os resultados e verificar sua autenticidade e precisão.

# Capítulo 2

## Especificação

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

### 2.1 Nome do sistema/produto

<b>Nome</b>	Software que resolve a Equação da Difusividade Hidráulica em termos adimensionais
<b>Componentes principais</b>	Desenvolvimento das soluções da equação da difusividade hidráulica
<b>Missão</b>	Resolver a equação da difusividade hidráulica para duas geometrias de fluxo e plotar os resultados gráficos utilizando funções de Bessel e o algoritmo de inversão numérica de Stehfest

### 2.2 Especificação

O software a ser desenvolvido deverá resolver as equações para obter a solução para regimes transiente, permanente e pseudopermanente, em fluxo linear; e para regime transiente, permanente e pseudopermanente, em fluxo radial. Cada uma dessas soluções deve ser mostrada de forma gráfica e deve ser possível fazer a comparação entre os regimes de fluxo para uma mesma geometria de reservatório.

O software será desenvolvido em linguagem C++, com orientação à objeto, e poderá ser utilizado nos sistemas operacionais GNU/Linux, Windows, e OS X, sendo operado em modo texto, e contendo apenas uma janela. Sua licença é GPL (*General Public License*).

Após a realização dos cálculos, as soluções serão apresentadas na tela em forma de gráfico, e o usuário terá a opção de salvá-los em disco.

Os gráficos serão gerados pelo software externo *Gnuplot* ([www.gnuplot.info](http://www.gnuplot.info)).

### 2.2.1 Requisitos funcionais

Apresenta-se a seguir os requisitos funcionais.

<b>RF-01</b>	O sistema deve conter uma base de dados para comparação da autenticidade e precisão dos mesmos.
<b>RF-02</b>	Deve permitir o carregamento de arquivos criados pelo software.
<b>RF-03</b>	O usuário poderá plotar seus resultados em um gráfico, e ele poderá ser salvo como imagem.

### 2.2.2 Requisitos não funcionais

<b>RNF-01</b>	A resolução da equação será feita utilizando dados adimensionais para mostrar as soluções gerais que servem para qualquer sistemas de unidades.
<b>RNF-02</b>	O programa deverá ser multi-plataforma, podendo ser executado em <i>Windows</i> , <i>GNU/Linux</i> ou OS X.

## 2.3 Casos de uso

<b>Nome do caso de uso:</b>	Obter soluções para a equação da difusividade
<b>Resumo/descrição:</b>	Processos que o programa realiza desde o momento em que o usuário insere os dados desejados, até a armazenagem dos resultados em disco
<b>Etapas:</b>	<ol style="list-style-type: none"> <li>1. Calcular as funções do regime radial;</li> <li>2. Reconhecer as funções de Bessel no campo de Laplace para o caso de regime radial;</li> <li>3. Inverter numericamente e calcular as funções do regime radial utilizando o algoritmo de Stehfest;</li> <li>4. Calcular as funções do regime linear;</li> <li>5. Gerar gráficos das soluções;</li> <li>6. Analisar os resultados gráficos;</li> <li>7. Salvar a imagem ou os gráficos em disco</li> </ol>
<b>Cenários alternativos:</b>	Não aplicável

### 2.3.1 Diagrama de caso de uso geral

O diagrama de caso de uso geral da Figura 2.1 mostra o usuário realizando a simulação para resolver a equação da difusividade, em ambos os regimes e analisando os subsequentes resultados.

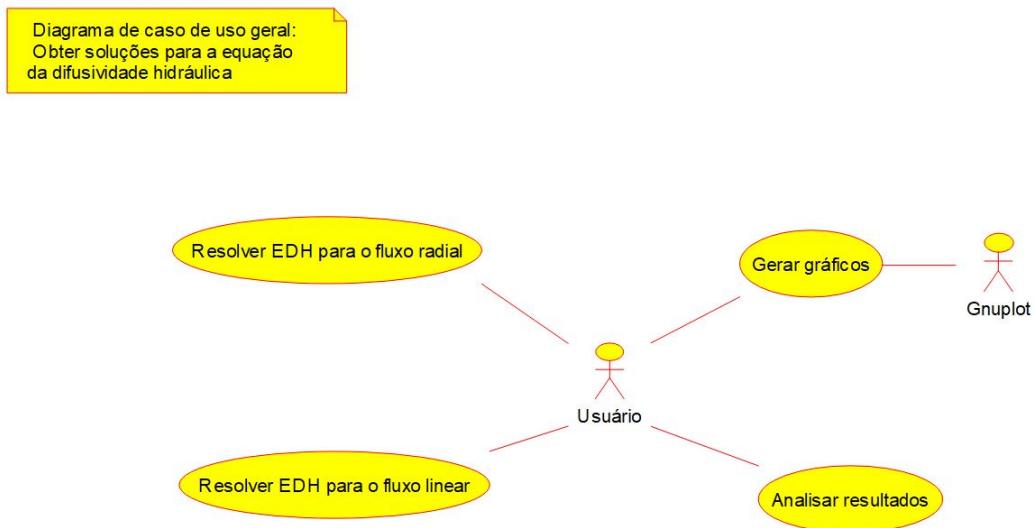


Figura 2.1: Caso de uso geral: Obter as soluções da Equação da Difusividade Hidráulica (EDH)

### 2.3.2 Diagrama de caso de uso específico

O diagrama de caso de uso a seguir detalha o cenário de teste que virá a ser realizado no Capítulo 7.

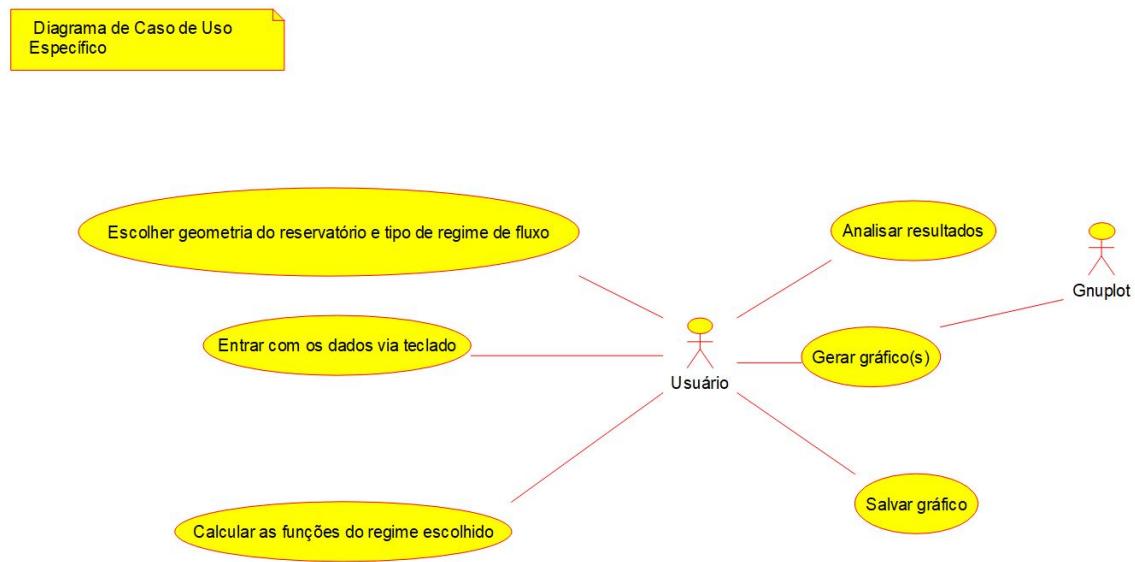


Figura 2.2: Diagrama de caso de uso específico: Cenário de teste

# 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

Após estudo dos requisitos/especificações do sistema, estudos de referências e disciplinas do curso foi possível identificar nosso domínio de trabalho:

- Engenharia de Reservatórios: é a espinha dorsal na qual esse projeto se sustenta. O software aqui desenvolvido, utiliza conceitos como a Equação de Conservação da Massa, a Lei de Darcy, compressibilidade da rocha e fluido, dentre outros. O software irá aplicar todos esses conceitos na resolução da Equação da Difusividade Hidráulica de acordo com a geometria do reservatório e do regime de fluxo escolhido.
- Modelagem Numérica Computacional: utiliza conceitos matemáticos de Cálculo Numérico. Neste software serão utilizados por exemplo, a Transformada de Laplace, o Algoritmo de Stehfest e as Funções de Bessel Modificadas de 1<sup>a</sup> ordem e 2<sup>a</sup> ordem.
- Pacote Gráfico: usar-se-á um pacote gráfico para a geração de gráficos de diferentes soluções que resolvam o problema apresentado.
- Software: Serão utilizadas classes e bibliotecas já existentes para a resolução da EDH por meio da transformada de Laplace e das Funções de Bessel.

## 3.2 Formulação teórica

Nesta seção consta a formulação física-matemática, as equações envolvidas, dentre outras propriedades desejadas utilizadas no software desenvolvido.

### 3.2.1 Lei de Darcy

O programa envolve conceitos de diversas áreas da engenharia e matemática. O princípio para o estudo de fluxo em meios porosos é baseado nos resultados experimentais de Henry Darcy. A análise destes resultados permitiu a Darcy formular a lei que se tornaria a base para a compreensão do fluxo em meios porosos.

A lei de Darcy foi generalizada para todos os fluidos, e é apresentada em forma diferencial:

$$v_x = -\frac{k}{\mu} \frac{dp}{dx} \quad (3.1)$$

### 3.2.2 Princípio de Conservação de Massa

O princípio da conservação de massa é representado matematicamente através da equação da continuidade. Considerando-se fluxo horizontal através de um volume de controle arbitrário  $V$  composto pelo somatório de diversos elementos de superfície  $\Delta A$ , a variação de massa no elemento é igual à diferença entre as quantidades que entram e saem do volume de controle como mostrado abaixo:

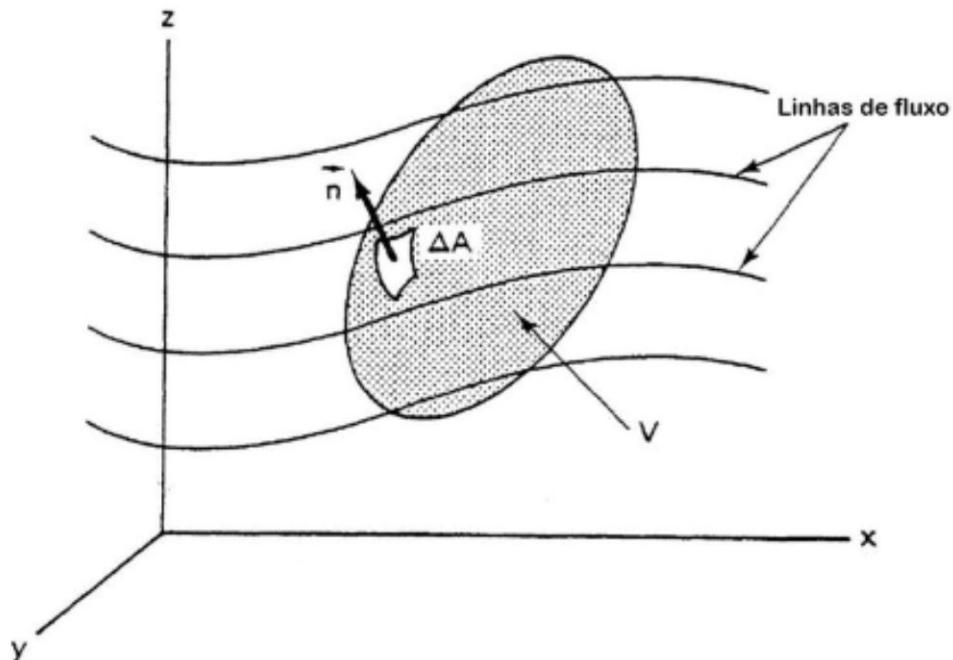


Figura 3.1: Volume de controle arbitrário no domínio de fluxo (Adaptado:[Lake, 1989])

Substituindo o termo da velocidade da Equação de Darcy  $v_x$ , a equação de conservação da massa se dá por:

$$\frac{\partial}{\partial x} \left( \rho - \frac{k}{\mu} \frac{dp}{dx} \right) + \frac{\partial}{\partial y} \left( \rho - \frac{k}{\mu} \frac{dp}{dy} \right) + \frac{\partial}{\partial z} \left( \rho - \frac{k}{\mu} \frac{dp}{dz} \right) = -\frac{\partial}{\partial t} (\phi \rho) \quad (3.2)$$

### 3.2.3 Equação da Difusividade Hidráulica

A equação da difusividade relaciona o comportamento da pressão no interior do reservatório com o tempo e é função da porosidade da rocha, viscosidade do fluido, compressibilidade total do sistema e da permeabilidade relativa ao fluido em consideração.

A equação da difusividade hidráulica, como é utilizada na engenharia de reservatórios, é obtida a partir da associação de três equações básicas: a equação da continuidade, que é uma equação da conservação da massa, a lei de Darcy, que é uma equação de transporte de massa, e uma equação de estado que tanto pode ser uma lei dos gases como a equação da compressibilidade para o caso dos líquidos ([Rosa et al., 2006]; [Ahmed, 1989]).

Na formulação desta equação serão admitidas algumas hipóteses:

- Fluxo estritamente horizontal;
- Meio poroso homogêneo de espessura constante;
- Fluxo monofásico;
- Uma única fase saturando o meio poroso;
- Sem reações químicas e sem absorção;
- Um único componente;
- Fluxo isotérmico;
- Viscosidade constante;
- Vazão constante;
- Fluido e rocha de compressibilidade pequena e constante;
- Pequenos gradientes de pressão.

#### Fluxo Linear

Para um sistema de fluxo linear, ou seja, quando há apenas fluxo na direção  $x$ , por exemplo, os termos referentes às direções  $y$  e  $z$  são iguais a zero e a equação da difusividade se dá por:

$$\frac{\partial^2 p}{\partial x^2} = -\frac{1}{\eta} \frac{\partial p}{\partial t} \quad (3.3)$$

onde  $\eta = \frac{k}{\phi\mu c_t}$  e é conhecida como a *constante de difusividade hidráulica*.

A partir desta forma reduzida serão desenvolvidas as soluções para os regimes de fluxo representados a seguir na forma adimensional:

$$p_D(t_D) = \sqrt{\frac{4t_D}{\pi}} \quad (3.4)$$

*Regime Transiente*

$$p_D(x_D) = x_D \quad (3.5)$$

*Regime Permanente*

$$p_D(x_D) = x_D - \frac{x_D^2}{2} \quad (3.6)$$

*Regime Pseudopermanente*

## Fluxo Radial

Para um sistema com fluxo radial, em coordenadas cilíndricas, a equação da difusividade é escrita da seguinte forma:

$$\frac{1}{r} \frac{\partial p}{\partial r} + \frac{\partial^2 p}{\partial r^2} + \frac{1}{r^2} \left( \frac{\partial^2 p}{\partial \theta^2} \right) + \frac{\partial^2 p}{\partial z^2} = \frac{\phi\mu c_t}{k} \frac{\partial p}{\partial t} \quad (3.7)$$

Como neste trabalho o fluxo é restrito apenas à direção  $r$  a Eq. 3.7 se reduz à expressão:

$$\frac{1}{r} \frac{\partial p}{\partial r} + \frac{\partial^2 p}{\partial r^2} = \frac{\phi\mu c_t}{k} \frac{\partial p}{\partial t} \quad (3.8)$$

A partir dessas equações serão calculadas as soluções para os três tipos de fluxo utilizando-se do conceito de transformada de Laplace para resolver a equação da difusividade. Como as soluções são obtidas analiticamente apenas no campo de Laplace, é necessário um algoritmo de inversão numérica para se obter o comportamento da queda de pressão adimensional  $p_D$  em função de  $r_D$  e  $t_D$ . Um algoritmo normalmente utilizado para tal inversão é o algoritmo de Stehfest [Stehfest, 1970]. A seguir as soluções adimensionais para o fluxo radial no campo de Laplace:

$$\overline{p}_D(r_D, u) = \frac{K_0(r_D\sqrt{u})}{u^{3/2} K_1(\sqrt{u})} \quad (3.9)$$

*Regime Transiente*

$$\overline{p}_D(r_D, u) = \frac{I_0(r_{eD}\sqrt{u})K_0(r_D\sqrt{u}) - K_0(r_{eD}\sqrt{u})I_0(r_D\sqrt{u})}{u^{3/2}[I_0(r_{eD}\sqrt{u})K_1(\sqrt{u}) + I_1(\sqrt{u})K_0(r_{eD}\sqrt{u})]} \quad (3.10)$$

*Regime Permanente*

$$\overline{p_D}(r_D, u) = \frac{I_1(r_{eD}\sqrt{u})K_0(r_D\sqrt{u}) + K_1(r_{eD}\sqrt{u})I_0(r_D\sqrt{u})}{u^{3/2}[I_1(r_{eD}\sqrt{u})K_1(\sqrt{u}) - I_1(\sqrt{u})K_1(r_{eD}\sqrt{u})]} \quad (3.11)$$

*Regime Pseudopermanente*

As soluções apresentadas são escritas em termos das funções de Bessel modicadas de primeira espécie,  $I_0$  e  $I_1$ , e de segunda espécie,  $K_0$  e  $K_1$ , de ordens zero e um, respectivamente, e onde  $u$  é a variável de Laplace. Após a aplicação do algoritmo de Stehfest, as soluções serão apresentadas de forma gráfica.

### 3.3 Identificação de pacotes – assuntos

.Engenharia de Reservatórios: Esse pacote recebe os parâmetros do usuário ( $p_D, r_D, t_D, r_{eD}$ ) ou lê do disco.

- Modelagem Numérica Computacional: Contém os algoritmos matemáticos necessários para a solução dos modelos de reservatório. Este pacote contém os algoritmos: Transformada de Laplace, Algoritmo de Inversão Numérica de Stehfest e Funções de Bessel. Este pacote permite ter uma maior reusabilidade do código, assim, estando separados, é possível aplicar este mesmo pacote para outros problemas de engenharia, como por exemplo o de análise de teste de pressão.
- Gráfico: Aqui se encontra a biblioteca do Gnuplot, necessária para a geração dos gráficos das soluções para cada reservatório.
- Biblioteca: Dentre as bibliotecas utilizadas, estarão as bibliotecas padrão de C++ (STL) e bibliotecas como a iostream, iomanip, etc. e a biblioteca GSL que fornecerá a base para as componentes do NCP.

### 3.4 Diagrama de pacotes – assuntos

A Figura 3.2 mostra o diagrama de pacotes do software.

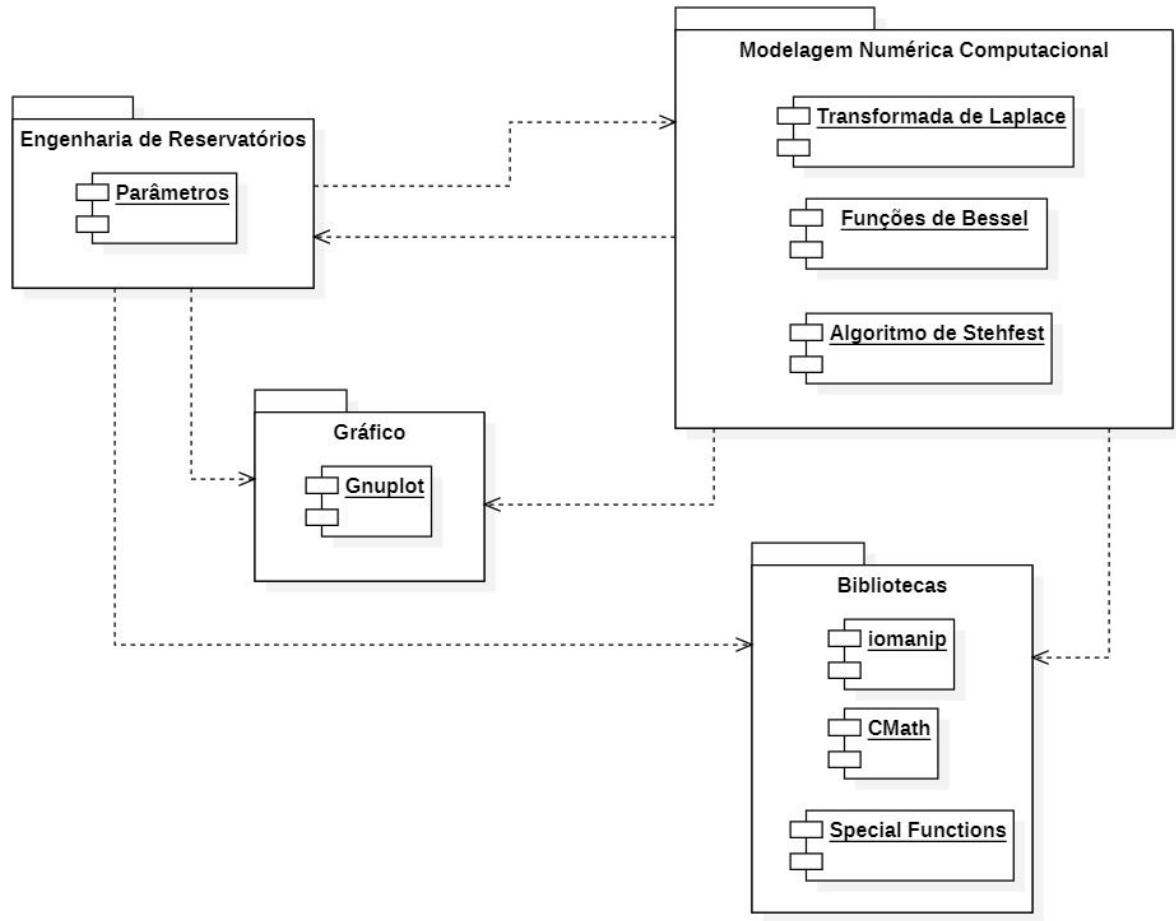


Figura 3.2: Diagrama de Pacotes

# Capítulo 4

## AOO – Análise Orientada a Objeto

A terceira etapa do desenvolvimento de um programa é a Análise Orientada a Objeto (AOO). A AOO utiliza algumas regras para identificar os objetos de interesse, as relações entre as classes, os atributos, os métodos, as heranças, as associações e as dependências (BUENO, 2003). Nesta etapa o primordial não é o programa em si, mas o que será desenvolvido. Aqui, faremos e analisaremos um conjunto de diagramas que permitirá visualizar, de várias formas, o software. A AOO abrange o desenvolvimento dos modelos estrutural e dinâmico.

### 4.1 Diagramas de classes

O diagrama de classes é apresentado na Figura 4.1.

#### 4.1.1 Dicionário de classes

- Classe CGeometriaReservatorio: Classe que reúne os atributos que definem a geometria do reservatório linear ou radial;
- Classe CInvNumStehfest: Classe que realiza a inversão numérica das equações adimensionais no Campo de Laplace;
- Classe CReservatorioLinearInfinito: Esta classe reúne os atributos e métodos necessários para fazer o cálculo dos parâmetros das equações que envolvem fluxo linear infinito, mostrar os resultados gráficos na tela do usuário e salvá-los em disco;
- Classe CReservatorioLinearSelado: Esta classe reúne os atributos e métodos necessários para fazer o cálculo dos parâmetros das equações que envolvem fluxo linear selado, mostrar os resultados gráficos na tela do usuário e salvá-los em disco;
- Classe CReservatorioLinearManutencao: Esta classe reúne os atributos e métodos necessários para fazer o cálculo dos parâmetros das equações que envolvem fluxo

linear com manutenção, mostrar os resultados gráficos na tela do usuário e salvá-los em disco;

- Classe CReservatorioRadialInfinito: Esta classe reúne os atributos e métodos necessários para fazer o cálculo dos parâmetros das equações que envolvem fluxo radial infinito, mostrar os resultados gráficos na tela do usuário e salvá-los em disco;
- Classe CReservatorioRadialSelado: Esta classe reúne os atributos e métodos necessários para fazer o cálculo dos parâmetros das equações que envolvem fluxo radial selado, mostrar os resultados gráficos na tela do usuário e salvá-los em disco;
- Classe CReservatorioRadialManutencao: Esta classe reúne os atributos e métodos necessários para fazer o cálculo dos parâmetros das equações que envolvem fluxo radial com manutenção, mostrar os resultados gráficos na tela do usuário e salvá-los em disco;
- Classe CGnuplot: Esta classe contém um conjunto de instruções para plotar o gráfico utilizado em CSimulador;
- Classe CSimulador: Esta classe utiliza todas as classes anteriores para resolver as equações propostas de acordo com a geometria e tipo de regime escolhido pelo usuário.

## 4.2 Diagrama de seqüência – eventos e mensagens

Mostra a sequência temporal pela qual as informações passam de uma classe para outra.

### 4.2.1 Diagrama de sequência geral

Veja o diagrama de seqüência na Figura 4.2. Ele representa uma ordem temporal pela qual as classes se relacionam entre si e com o usuário.

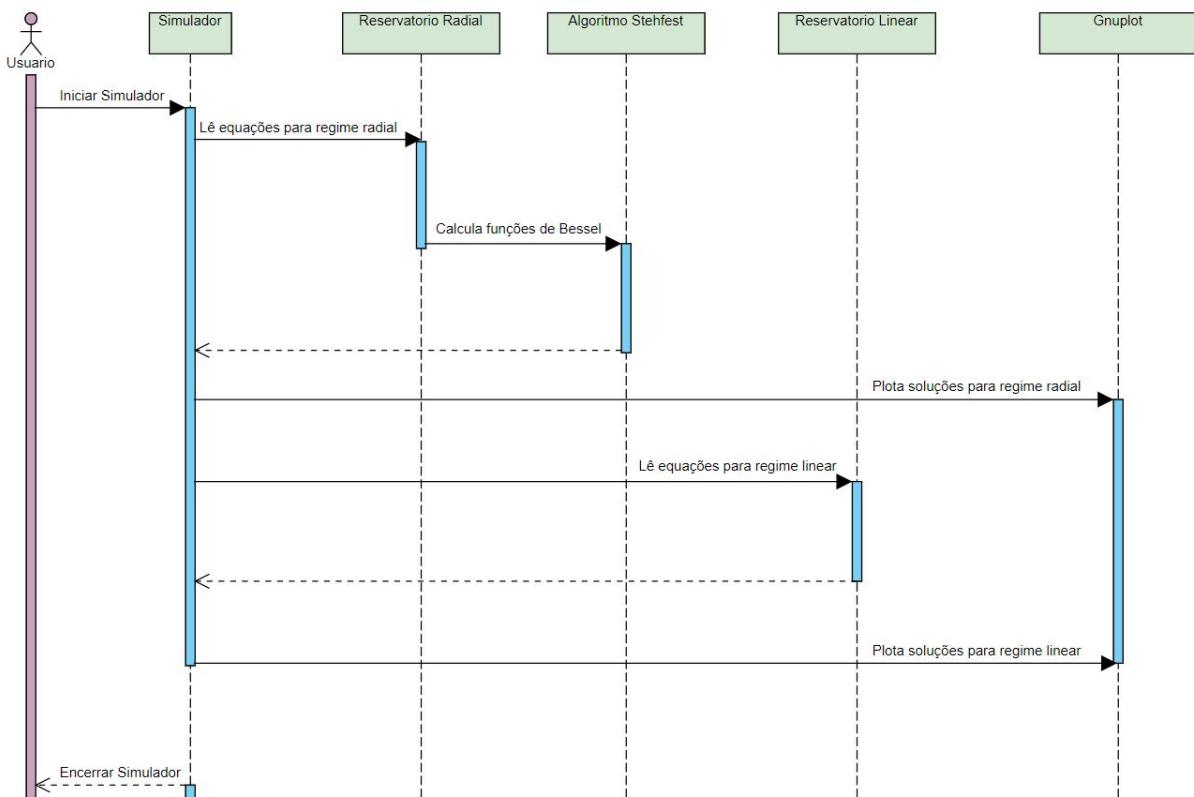


Figura 4.2: Diagrama de sequência

### 4.3 Diagrama de comunicação – colaboração

O diagrama de comunicação pode ser desenvolvido como uma extensão do diagrama de caso de uso, detalhando o mesmo por meio da inclusão de objetos, mensagens e parâmetros trocados entre objetos. No diagrama de comunicação, o foco é a interação e a troca de mensagens e dados entre os objetos.

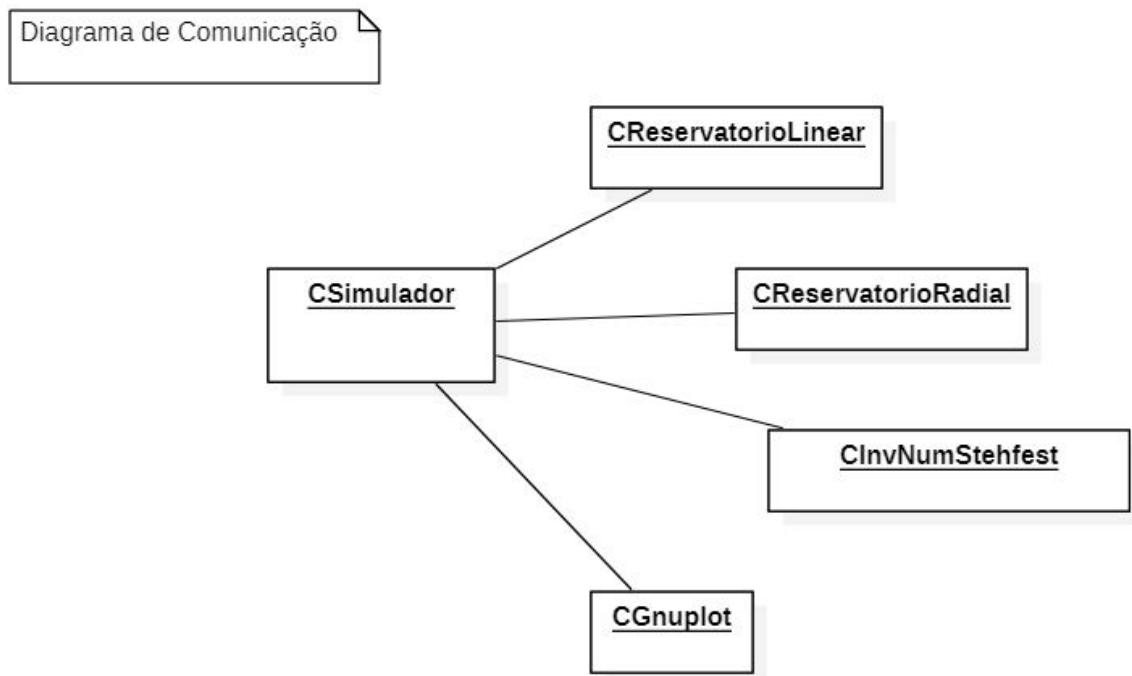


Figura 4.3: Diagrama de comunicação

## 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 de um objeto). Estes estados podem ser a realização de uma atividade demorada ou a espera de um evento.

Veja na Figura 4.4 o diagrama de máquina de estado para o objeto.

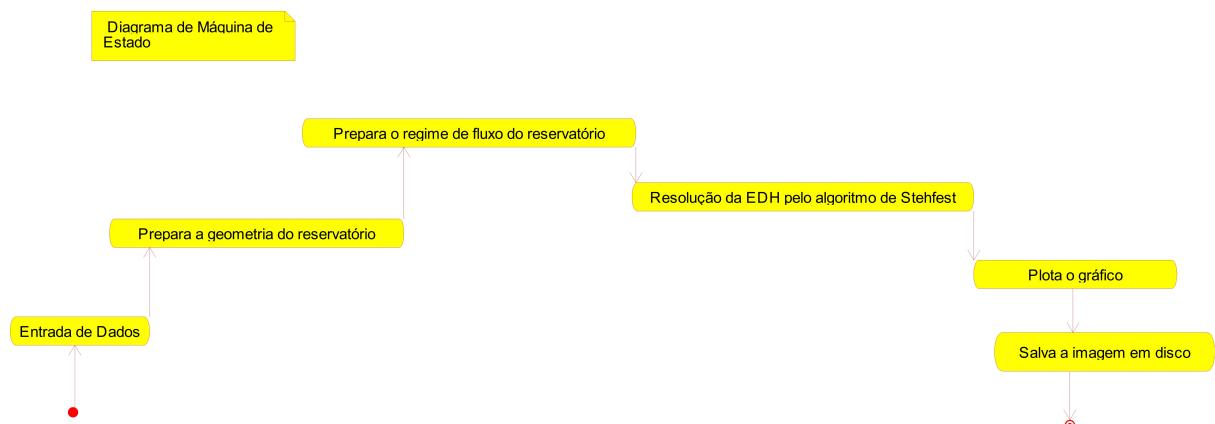


Figura 4.4: Diagrama de máquina de estado

## 4.5 Diagrama de atividades

O diagrama de atividades correspondente a uma atividade específica do diagrama de máquina de estado.

Veja na Figura 4.5 o diagrama de atividades correspondente a uma atividade específica do diagrama de máquina de estado.



Figura 4.5: Diagrama de atividades



Figura 4.1: Diagrama de classes

# Capítulo 5

## Projeto

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

### 5.1 Projeto do sistema

Depois da análise orientada a objeto, desenvolveu-se o projeto do sistema, o qual reúne 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. Foram definidos 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. O presente projeto do sistema foi elaborado para apresentar soluções, valendo-se dos seguintes itens como:

#### 1. Protocolos

- Definição dos protocolos de comunicação entre os diversos elementos externos (como dispositivos). Por exemplo: se o sistema envolve o uso dos nós de um cluster, devem ser considerados aspectos como o protocolo de comunicação entre os nós do cluster.
  - Esta versão do software comunica-se com o programa externo gnuplot.
- Definição dos protocolos de comunicação entre os diversos elementos internos (como objetos).

- O programa utilizará uma máquina computacional com HD, processador, teclado para a entrada de dados e o monitor para a saída de dados. Os arquivos gerados pelo programa estarão em formato de texto em um banco de dados.
  - Denição da interface API de suas bibliotecas e sistemas
- Utilizará uma biblioteca GSL para cálculos matemáticos especiais chamada special functions da GNU.
  - Definição do formato dos arquivos gerados pelo software. Por exemplo: prefira formatos abertos, como arquivos txt e xml.
- Os arquivos de texto com os resultados da simulação terão o formato .dat ou .txt.

## 2. Recursos

- Identificação e alocação dos recursos globais, como os recursos do sistema serão alocados, utilizados, compartilhados e liberados. Implicam modificações no diagrama de componentes.
- O simulador utiliza como recurso para plotagem de gráficos o programa externo Gnuplot. Que por sua vez plota os objetos da imagem rotulados e ajustados as formas geométricas. Além do básico, como HD, CPU, RAM e periféricos.

## 3. Controle

- Identificação e seleção da implementação de controle, sequencial ou concorrente, baseado em procedimentos ou eventos. Implicam modificações no diagrama de execução.
- Este software não requer um controle sequencial.
- Não requer processamento paralelo, visto que o programa e seus componentes executam cálculos que requerem pouco poder de processamento.

## 4. Plataformas

- Seleção do ambiente de desenvolvimento para montar a interface de desenvolvimento - IDE.
- Para a plataforma Windows, temos a seguinte IDE: Desktop com processador AMD FX-6100, 8GB RAM DDR3 e placa gráca AMD Radeon RX 470. O programa será escrito e compilado usando o programa DEV-C++.
- Para Linux: Desktop com processador Intel Core i7 3770, 8GB RAM DDR3, placa gráca Nvidia GT 620. O programa será escrito e compilado usando o programa Kate e gcc++.

## 5. Bibliotecas

- Será utilizada a biblioteca padrão da linguagem C++, incluindo *cmath*, *vector*, *string*, *iostream*, *fstream*, *sstream* e *boost*.
- Será utilizada a biblioteca Gnuplot neste software.

## 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-se a análise desenvolvida e as características da plataforma escolhida (hardware, sistema operacional e linguagem de programação). É realizado um maior detalhamento do funcionamento do programa, 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. Ainda no projeto orientado a objeto incluem-se as bibliotecas necessárias para acesso ao disco, cria um objeto específico para acessar o disco, podendo, portanto, acrescentar novas classes àquelas desenvolvidas na análise. Exemplo: na análise você define que existe um método para salvar um arquivo em disco, define um atributo NomeDoArquivo, mas não se preocupa com detalhes específicos da linguagem. Já no projeto, você inclui as bibliotecas necessárias para acesso ao disco, cria um objeto específico para acessar o disco, podendo, portanto, acrescentar novas classes àquelas desenvolvidas na análise.

### Efeitos do projeto no modelo estrutural

- A biblioteca gráfica utilizada será CGnuplot. Ela fornece acesso ao *software* externo Gnuplot, utilizado para plotar o gráfico.

### Efeitos do projeto nos métodos

- Neste projeto, os métodos de classe exercem a função de armazenar valores nos atributos, calcular ou executar atividades que suas respectivas classes propõem.

### Efeitos do projeto nas associações

- A classe CSimulador reunirá as informações contidas em CReservatorioLinear e CReservatorioRadial, que por sua vez, utilizará os métodos contidos na classe CInv-NumStehfest para calcular a inversão numérica das soluções no campo de Laplace.

### 5.3 Diagrama de componentes

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

Veja na Figura 5.1 o diagrama de componentes para nosso *software*. Observe que este inclui muitas dependências, ilustrando as relações entre os arquivos.

Algumas observações úteis para o diagrama de componentes:

- De posse do diagrama de componentes, temos a lista de todos os arquivos necessários para compilar e rodar o software.
- Observe que um assunto/pacote pode se transformar em uma biblioteca e será incluído no diagrama de componentes.

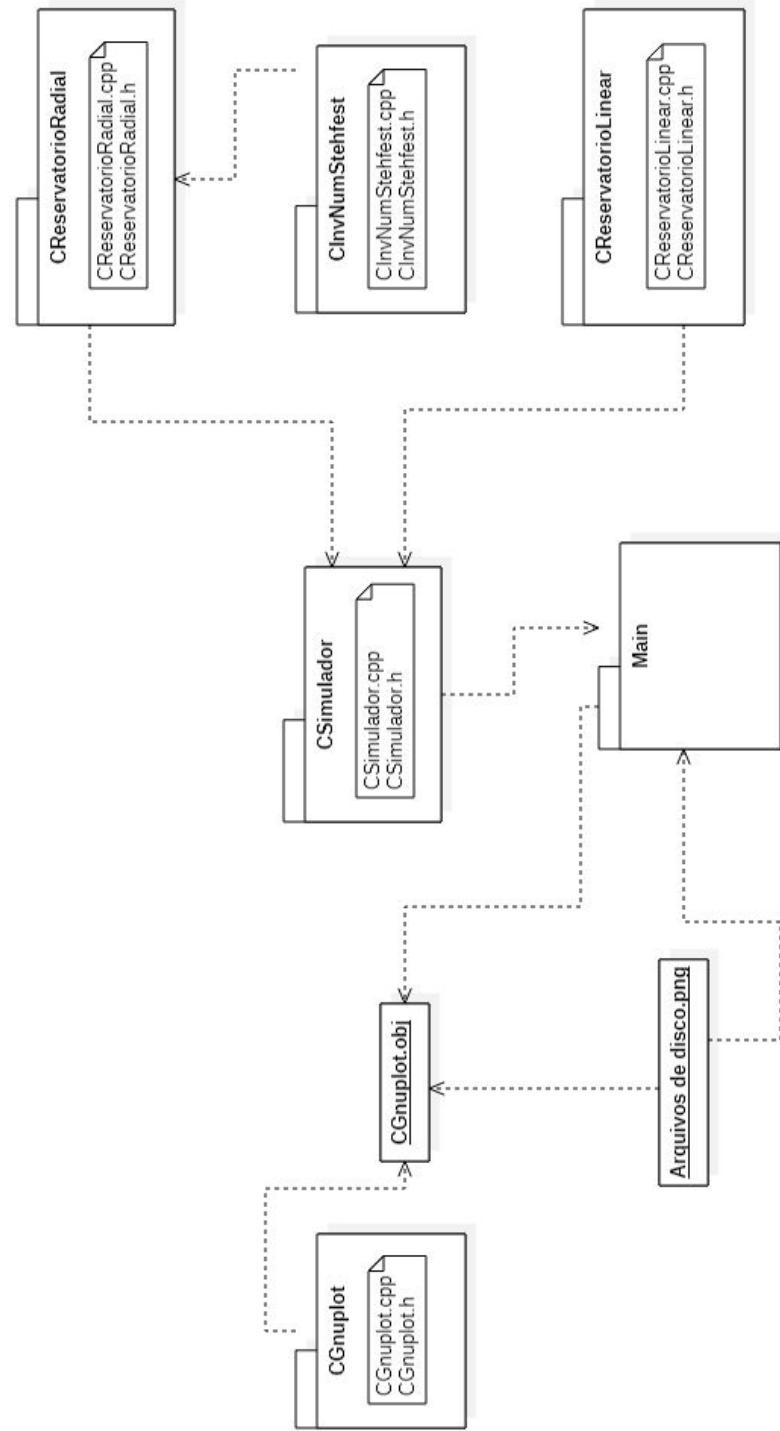


Figura 5.1: Diagrama de componentes do software Soluções Analíticas Adimensionais da Equação da Difusividade Hidráulica para Fluxo Linear e Radial

## 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 o diagrama de implantação do programa.

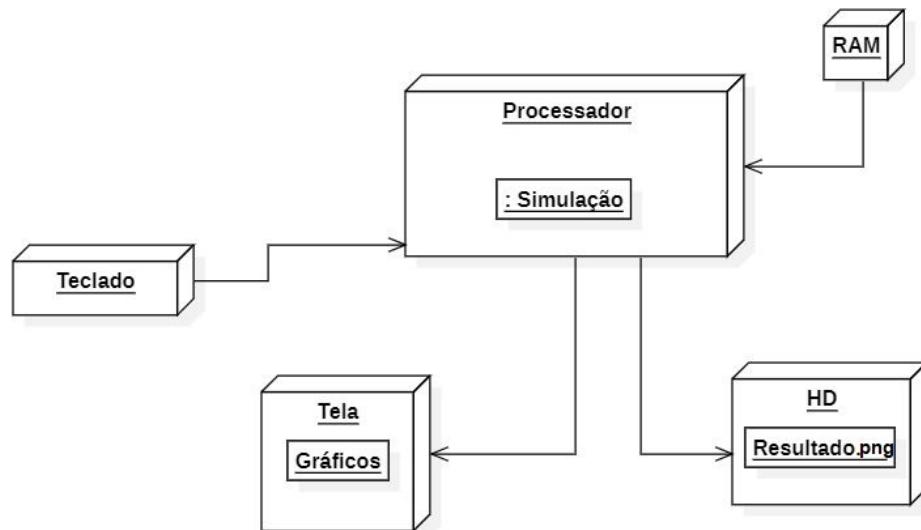


Figura 5.2: Diagrama de implantação.

# Capítulo 6

## Implementação

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

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

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

**Nota:** ao longo deste capítulo usamos inclusão direta de arquivos externos usando o pacote *listings* do L<sup>A</sup>T<sub>E</sub>X. Maiores detalhes de como a saída pode ser gerada estão disponíveis nos links abaixo.

- [http://en.wikibooks.org/wiki/LaTeX/Source\\_Code\\_Listings](http://en.wikibooks.org/wiki/LaTeX/Source_Code_Listings).
- <http://mirrors.ctan.org/macros/latex/contrib/listings/listings.pdf>.

### 6.1 Código fonte

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

Apresenta-se na listagem 6.1 o arquivo com código da classe **CGeometriaReservatorio**.

---

Listing 6.1: Arquivo de cabeçalho da classe CGeometriaReservatorio.

```
1 #ifndef CGeometriaReservatorio_H_
2 #define CGeometriaReservatorio_H_
3
4 //Criacao da classe CGeometriaReservatorio.h
5 class CGeometriaReservatorio
6 {
7     ///Declaracao metodos protegidos
```

```

8     protected:
9
10    ///Pressao Adimensional, Raio externo adimensional
11    double pd, RD;
12
13    ///Declaracao metodos publicos
14    public:
15
16        ///Construtor default
17        CGeometriaReservatorio() {};
18
19        /**
20            Calcula um ponto de Laplace para determinada
21            forma de reservatorio.
22            @parametro u variavel do tempo no campo de
23            Laplace
24            @parametro RD raio externo adimensional
25            @retorna valor de funcao de Laplace calculada no
26            ponto (u, RD)
27
28        */
29
30        virtual double Forma(double _u, double _RD);
31
32    endif

```

---

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

Listing 6.2: Arquivo de implementação da classe CGeometriaReservatorio.

```

1 #include "CGeometriaReservatorio.h"
2
3 double CGeometriaReservatorio::Forma(double _u, double _RD)
4 {
5     return pd;
6 }

```

---

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

Listing 6.3: Arquivo de cabeçalho da classe CGnuplot.

```

1 //////////////////////////////////////////////////////////////////

```

---

```
2//          Classe de Interface em C++ para o programa gnuplot
3//          .
4//          ///////////////////////////////////////////////////////////////////
5// Esta interface usa pipes e nao ira funcionar em sistemas que nao
6// suportam
7// o padrao POSIX pipe.
8// O mesmo foi testado em sistemas Windows (MinGW e Visual C++) e
9// Linux(GCC/G++)
10// Este programa foi originalmente escrito por:
11// Historico de versoes:
12// 0. Interface para linguagem C
13//     por N. Devillard (27/01/03)
14// 1. Interface para C++: traducao direta da versao em C
15//     por Rajarshi Guha (07/03/03)
16// 2. Correcoes para compatibilidade com Win32
17//     por V. Chyzhdzenka (20/05/03)
18// 3. Novos metodos membros, correcoes para compatibilidade com
19//     Win32 e Linux
20//     por M. Burgis (10/03/08)
21// 4. Traducao para Portugues, documentacao - javadoc/doxygen ,
22//     e modificacoes na interface (adicao de interface alternativa)
23//     por Bueno.A.D. (30/07/08)
24// Tarefas:
25// (v1)
26// Documentar toda classe
27// Adicionar novos metodos, criando atributos adicionais se
28// necessario.
29// Adotar padrao C++, isto e, usar sobrecarga nas chamadas.
30// (v2)
31// Criar classe herdeira CGnuplot, que inclui somente a nova
32// interface.
33// como e herdeira, o usuario vai poder usar nome antigos.
34// Vantagem: preserva classe original, cria nova interface, fica
35// a critério do usuário
36// qual interface utilizar.
37// ///////////////////////////////////////////////////////////////////
38// 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     // -----
73     Atributos
74     FILE * gnuCMD;           ///< Ponteiro para stream que escreve no pipe.
75     bool valid;              ///< Flag que indica se a sessao do gnuplot esta valida.
76     bool two_dim;            ///< true = verdadeiro = 2d, false = falso = 3d.
77     int nplots;              ///< Numero de graficos (plots) na sessao.
78     std::string pstyle;      ///< Estilo utilizado para visualizacao das funcoes e dados.
79     std::string smooth;      ///< interpolate and approximate data in defined styles (e.g. spline).
80     std::vector<std::string> tmpfile_list; ///< Lista com nome dos arquivos temporarios.
81     // -----
82     flags
83     bool fgrid;              ///< 0 sem grid, 1 com grid
84     bool fhidden3d;          ///< 0 nao oculta, 1 oculta
85     bool fcontour;           ///< 0 sem contorno, 1 com contorno
86     bool fsurface;           ///< 0 sem superficie, 1 com superficie
87     bool flegend;             ///< 0 sem legendad, 1 com legenda
88     bool ftitle;              ///< 0 sem titulo, 1 com titulo
89     bool fxlogscale;         ///< 0 desativa escala log, 1 ativa escala log
90     bool fylogscale;         ///< 0 desativa escala log, 1 ativa escala log
91     bool fzlogscale;         ///< 0 desativa escala log, 1 ativa escala log
92     bool fsmooth;            ///< 0 desativa, 1 ativa
93     // -----
```

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

101           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
115           create_tmpfile(tmp); }
116
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,
157     // diretamente na construcao.
158 Gnuplot (const std::vector < double >&x,
159             const std::string & title = "",
160             const std::string & style = "points",
161             const std::string & labelx = "x",
162             const std::string & labely = "y");
163
164 /// @brief Construtor, plota um grafico do tipo x_y a partir de
165     // vetores, diretamente na construcao.
166 Gnuplot (const std::vector < double >&x,
167             const std::vector < double >&y,
168             const std::string & title = "",
169             const std::string & style = "points",
170             const std::string & labelx = "x",
171             const std::string & labely = "y");
172
173 /// @brief Construtor, plota um grafico de x_y_z a partir de
174     // vetores, diretamente na construcao.
175 Gnuplot (const std::vector < double >&x,
176             const std::vector < double >&y,
177             const std::vector < double >&z,
178             const std::string & title = "",
179             const std::string & style = "points",
180             const std::string & labelx = "x",
181             const std::string & labely = "y",
182             const std::string & labelz = "z");
183
184 /// @brief Destrutor, necessario para deletar arquivos
185     // temporarios.
186 ~Gnuplot ();
187
188 // -----
189
190
191 /// @brief Envia comando para o gnuplot.
192 Gnuplot & cmd (const std::string & cmdstr);
193
194
```

```
188  /// @brief Envia comando para o gnuplot.  
189  Gnuplot & Cmd (const std::string & cmdstr) { return cmd(  
190      cmdstr); }  
  
191  /// @brief Envia comando para o gnuplot.  
192  Gnuplot & Command (const std::string & cmdstr) { return cmd(  
193      cmdstr); }  
  
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  
199  terminal para terminal_std.  
200  Gnuplot & showonscreen (); // Janela de saida e setada  
201  como default (win/x11/aqua)  
  
202  /// @brief Mostrar na tela ou escrever no arquivo, seta o tipo de  
203  terminal para terminal_std.  
204  Gnuplot & ShowOnScreen () { return  
205      showonscreen(); };  
  
206  /// @brief Salva sessao do gnuplot para um arquivo postscript,  
207  informe o nome do arquivo sem extensao.  
208  /// Depois retorna para modo terminal  
209  Gnuplot & savetops (const std::string & filename = "  
210      gnuplot_output");  
  
211  /// @brief Salva sessao do gnuplot para um arquivo postscript,  
212  informe o nome do arquivo sem extensao  
213  /// Depois retorna para modo terminal  
214  Gnuplot & SaveTops (const std::string & filename = "  
215      gnuplot_output")  
216  { return savetops  
217      (filename); }  
  
218  /// @brief Salva sessao do gnuplot para um arquivo png, nome do  
219  arquivo sem extensao  
220  /// Depois retorna para modo terminal  
221  Gnuplot & savetopng (const std::string & filename = "
```

```
gnuplot_output");  
216  
217     /// @brief Salva sessao do gnuplot para um arquivo png, nome do  
218     /// arquivo sem extensao  
219     /// Depois retorna para modo terminal  
220     Gnuplot & SaveTopng (const std::string & filename = "  
221         gnuplot_output")  
222             { return  
223                 savetopng(  
224                     filename); }  
225  
226     /// @brief Salva sessao do gnuplot para um arquivo jpg, nome do  
227     /// arquivo sem extensao  
228     /// Depois retorna para modo terminal  
229     Gnuplot & savetojpeg (const std::string & filename = "  
230         gnuplot_output")  
231             { return  
232                 savetojpeg(  
233                     filename); }  
234  
235     /// @brief Salva sessao do gnuplot para um arquivo filename,  
236     /// usando o terminal_type e algum flag adicional  
237     /// Ex:  
238     /// grafico.SaveTo("pressao_X_temperatura","png", "enhanced size  
239     /// 1280,960");  
240     /// Para melhor uso dos flags adicionais consulte o manual do  
241     /// gnuplot (help term)  
242     Gnuplot& SaveTo(const std::string &filename, const std::string &  
243                     terminal_type, std::string flags="");  
244  
245     //  
246     -----  
247     set e unset  
248     /// @brief Seta estilos de linhas (em alguns casos sao  
249     /// necessarias informacoes adicionais).  
250     /// 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")  
           { return set_smooth  
     (stylestr); }  
265  
266  Gnuplot & Smooth( int _fsmooth )  
267           { if( fsmooth ==  
     _fsmooth )  
           return  
268  
269
```

```
        set_contour
        ();
270
271           else
272               return
273               unset_contour
274               ();
275
276           /// @brief Desativa suavizacao.
277           //Gnuplot & UnsetSmooth()
278           { return
279               unset_smooth (); }
280
281           /// @brief Escala o tamanho do ponto usado na plotagem.
282           Gnuplot & set_pointsize (const double pointsize = 1.0);
283
284           /// @brief Escala o tamanho do ponto usado na plotagem.
285           Gnuplot & PointSize (const double pointsize = 1.0)
286           { return
287               set_pointsize(
288                   pointsize); }
289
290           /// @brief Ativa o grid (padrao = desativado).
291           Gnuplot & set_grid ();
292
293           /// @brief Desativa o grid (padrao = desativado).
294           Gnuplot & unset_grid ();
295
296           /// @brief Ativa/Desativa o grid (padrao = desativado).
297           Gnuplot & Grid(bool _fgrid = 1)
298           { if(fgrid = _fgrid
299               )
300                   return set_grid
301               ();
302
303               else
304                   return
305                   unset_grid()
306               ; }
307
308           /// @brief Seta taxa de amostragem das funcoes, ou dos dados de
309           //interpolacao.
310           Gnuplot & set_samples (const int samples = 100);
311
312           /// @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)  
    { if (fhidden3d =  
        _fhidden3d)  
        return  
        set_hidden3d  
    ();  
    else  
        return  
        unset_hidden3d  
    ();  
}  
  
318  
319  
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 )
     return
     set_contour
     ();
336
     else
     return
     unset_contour
     ();
337
}
338
339  /// @brief Ativa a visualizacao da superficie (para plotagen 3d)
340
341  .
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)
     return
     set_surface
     ();
349
     else
     return
350
}
351
```

```
        unset_surface
        ());
352    }
353    /// @brief Ativa a legenda (a legenda é setada por padrao).
354    /// Posicao: inside/outside, left/center/right, top/center/bottom
355    , nobox/box
356    Gnuplot & set_legend (const std::string & position = "default");

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
361    // padrao).
362    Gnuplot & Legend(const std::string & position = "default")
363                                { return set_legend
364                                  (position); }

365    /// @brief Ativa/Desativa a legenda (a legenda é setada por
366    // padrao).
367    Gnuplot & Legend(int _flegend)
368                                { if(flegend =
369                                  _flegend)
370                                    return
371                                    set_legend
372                                    ();
373
374                                else
375                                    return
376                                    unset_legend
377                                    ();
378
379    /// @brief Ativa o titulo da secao do gnuplot.
380    Gnuplot & set_title (const std::string & title = "");
381
382    /// @brief Desativa o titulo da secao do gnuplot.
383    Gnuplot & unset_title (); // O title nao e setado por
384    // padrao (default)

385    /// @brief Ativa/Desativa o titulo da secao do gnuplot.
386    Gnuplot & Title(const std::string & title = "")
387                                {
```

```
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
419              (iFrom,iTo); }
420
421  /// @brief Seta intervalo do eixo y.
422  Gnuplot & set_yrange (const int iFrom, const int iTo);
423
424  /// @brief Seta intervalo do eixo y.
425  Gnuplot & YRange (const int iFrom, const int iTo)
426          { return set_yrange
427              (iFrom,iTo); }
428
429  /// @brief Seta intervalo do eixo z.
430  Gnuplot & set_zrange (const int iFrom, const int iTo);
431
432  /// @brief Seta intervalo do eixo z.
433  Gnuplot & ZRange (const int iFrom, const int iTo)
434          { return set_zrange
435              (iFrom,iTo); }
436
437  /// @brief Seta escalonamento automatico do eixo x (default).
438  Gnuplot & set_xautoscale ();
439
440  /// @brief Seta escalonamento automatico do eixo x (default).
441  Gnuplot & XAutoscale()           { return
442              set_xautoscale(); }
443
444  /// @brief Seta escalonamento automatico do eixo y (default).
445  Gnuplot & YAutoscale()          { return
446              set_yautoscale(); }
447
448  /// @brief Seta escalonamento automatico do eixo z (default).
449  Gnuplot & set_zautoscale ();
450
451  /// @brief Seta escalonamento automatico do eixo z (default).
452  Gnuplot & ZAutoscale()          { return
453              set_zautoscale(); }
```

```
451
452     /// @brief Ativa escala logaritma do eixo x (logscale nao e
453     //      setado por default).
454     Gnuplot & set_xlogscale (const double base = 10);
455
456     /// @brief Desativa escala logaritma do eixo x (logscale nao e
457     //      setado por default).
458     Gnuplot & unset_xlogscale ();
459
460     /// @brief Ativa escala logaritma do eixo x (logscale nao e
461     //      setado por default).
462     Gnuplot & XLogscale (const double base = 10) { //if(base)
463                                     return
464                                     set_xlogscale
465                                     (base);
466                                     //else
467                                     //return
468                                     unset_xlogscale
469                                     ();
470
471     }
472
473     /// @brief Ativa/Desativa escala logaritma do eixo x (logscale
474     //      nao e setado por default).
475     Gnuplot & XLogscale(bool _fxlogscale)
476                                     {
477                                         if(fxlogscale =
478                                         _fxlogscale)
479                                         return
480                                         set_xlogscale
481                                         ();
482                                         else
483                                         return
484                                         unset_xlogscale
485                                         ();
486
487                                         }
488
489     /// @brief Ativa escala logaritma do eixo y (logscale nao e
490     //      setado por default).
491     Gnuplot & set_ylogscale (const double base = 10);
492
493     /// @brief Ativa escala logaritma do eixo y (logscale nao e
494     //      setado por default).
495     Gnuplot & YLogscale (const double base = 10) { return
```

```
        set_ylogscale (base); }

478
479 /// @brief Desativa escala logaritma do eixo y (logscale nao e
      setado por default).
480 Gnuplot & unset_ylogscale ();
481
482 /// @brief Ativa/Desativa escala logaritma do eixo y (logscale
      nao e setado por default).
483 Gnuplot & YLogscale(bool _fylogscale)
484
485 { if(fylogscale =
      _fylogscale)
486     return
      set_ylogscale
      ();
487
488 else
489     return
      unset_ylogscale
      ();
490 }
491
492 /// @brief Ativa escala logaritma do eixo y (logscale nao e
      setado por default).
493 Gnuplot & set_zlogscale (const double base = 10);
494
495 /// @brief Ativa escala logaritma do eixo y (logscale nao e
      setado por default).
496 Gnuplot & ZLogscale (const double base = 10) { return
      set_zlogscale (base); }

497
498 /// @brief Desativa escala logaritma do eixo z (logscale nao e
      setado por default).
499 Gnuplot & unset_zlogscale ();
500
501 /// @brief Ativa/Desativa escala logaritma do eixo y (logscale
      nao e setado por default).
502 Gnuplot & ZLogscale(bool _fzlogscale)
503
504 { if(fzlogscale =
      _fzlogscale)
505     return
      set_zlogscale
      ();
506
507 else
```

```
504                                     return  
505                                     unset_zlogscale  
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  
514         set_cbrange(  
515         iFrom, iTo); }  
516  
517     //////////////////////////////////////////////////////////////////  
518     /// @brief Plota dados de um arquivo de disco.  
519     Gnuplot & plotfile_x (const std::string & filename,  
520                           const int column = 1, const std::string & title = "")  
521                           ;  
522  
523     /// @brief Plota dados de um arquivo de disco.  
524     Gnuplot & PlotFile (const std::string & filename,  
525                           const int column = 1, const std::string & title = "")  
526                           { return plotfile_x  
527                               (filename,  
528                               column, title); }  
529  
530     /// @brief Plota dados de um vector.  
531     Gnuplot & plot_x (const std::vector < double >&x, const std::string & title = "");  
532  
533     /// @brief Plota dados de um vector.  
534     Gnuplot & PlotVector (const std::vector < double >&x, const std::string & title = "")  
535     { return plot_x( x,  
536                      title ); }  
537  
538     /// @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          "");
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  {
542      return plotfile_xy(
543          filename,
544          column_x,
545          column_y, title
546          );
547  }
548
549  /// @brief Plota pares x,y a partir de vetores.
550  Gnuplot & plot_xy (const std::vector < double >&x,
551          const std::vector < double >&y, const std::string &
552          title = "");
553
554  /// @brief Plota pares x,y a partir de vetores.
555  Gnuplot & PlotVector (const std::vector < double >&x,
556          const std::vector < double >&y, const std::string &
557          title = "")
558  {
559      return plot_xy (
560          x, y, title );
561
562  /// @brief Plota pares x,y com barra de erro dy a partir de um
563  /// arquivo.
564  Gnuplot & plotfile_xy_err (const std::string & filename,
565          const int column_x = 1,
566          const int column_y = 2,
567          const int column_dy = 3, const std::string &
568          title = "");
569
570  /// @brief Plota pares x,y com barra de erro dy a partir de um
571  /// arquivo.
572  Gnuplot & PlotFileXYErrorBar(const std::string & filename,
573          const int column_x = 1,
574          const int column_y = 2,
```

```
563         const int column_dy = 3, const std::string &
564             title = "")  
565             { return  
566                 plotfile_xy_err  
567                     (filename,  
568                         column_x,  
569                         column_y,  
570                         column_dy,  
571                         title ); }  
572  
573     /// @brief Plota pares x,y com barra de erro dy a partir de  
574     vetores.  
575     Gnuplot & plot_xy_err (const std::vector < double >&x,  
576                             const std::vector < double >&y,  
577                             const std::vector < double >&dy,  
578                             const std::string & title = "");  
579  
580     /// @brief Plota pares x,y com barra de erro dy a partir de  
581     vetores.  
582     Gnuplot & PlotVectorXYErrorBar(const std::vector < double >&x,  
583                                     const std::vector < double >&y,  
584                                     const std::vector < double >&dy,  
585                                     const std::string & title = "")  
586             { return  
587                 plot_xy_err(  
588                     x, y, dy,  
589                     title); }  
590  
591     /// @brief Plota valores de x,y,z a partir de um arquivo de  
592     disco.  
593     Gnuplot & plotfile_xyz (const std::string & filename,  
594                             const int column_x = 1,  
595                             const int column_y = 2,  
596                             const int column_z = 3, const std::string & title =  
597                             "");  
598     /// @brief Plota valores de x,y,z a partir de um arquivo de  
599     disco.  
600     Gnuplot & PlotFile (const std::string & filename,  
601                             const int column_x = 1,  
602                             const int column_y = 2,  
603                             const int column_z = 3, const std::string & title =  
604                             "")
```

```
590                                     { return
591                                         plotfile_xyz
592                                         (filename,
593                                         column_x,
594                                         column_y,
595                                         column_z);
596                                         }
597
598     /// @brief Plota valores de x,y,z a partir de vetores.
599     Gnuplot & plot_xyz (const std::vector < double >&x,
600                         const std::vector < double >&y,
601                         const std::vector < double >&z, const std::string &
602                         title = "");
603
604     /// @brief Plota valores de x,y,z a partir de vetores.
605     Gnuplot & PlotVector(const std::vector < double >&x,
606                           const std::vector < double >&y,
607                           const std::vector < double >&z, const std::string &
608                           title = "")
609
610
611     /// @brief Plota uma equacao da forma y = ax + b, voce fornece os
612     /// coeficientes a e b.
613     Gnuplot & plot_slope (const double a, const double b, const std
614                           ::string & title = "");
615
616
617     /// @brief Plota uma equacao da forma y = ax + b, voce fornece os
618     /// coeficientes a e b.
619     Gnuplot & PlotSlope (const double a, const double b, const std
620                           ::string & title = "")
621
622                                     { return
623                                         plot_slope(a
624                                         ,b,title); }
625
626
627     /// @brief Plota uma equacao fornecida como uma std::string y=f(
628     /// x).
629     /// Escrever somente a funcao f(x) e nao y=
630     /// A variavel independente deve ser x
631     /// 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),
626  /// real(x), arg(x),
627  /// sqrt(x), exp(x), log(x), log10(x), sin(x), cos(x), tan(x),
628  /// asin(x), acos(x),
629  /// atan(x), atan2(y,x), sinh(x), cosh(x), tanh(x), asinh(x),
630  /// acosh(x), atanh(x)
631  /// Funcoes especiais:
632  /// erf(x), erfc(x), inverf(x), gamma(x), igamma(a,x), lgamma(x),
633  /// ibeta(p,q,x),
634  /// besj0(x), besj1(x), besy0(x), besy1(x), lambertw(x)
635  /// Funcoes estatisticas:
636  /// norm(x), invnorm(x)
637  Gnuplot & plot_equation (const std::string & equation,
638                               const std::string & title = "");
639
640  /// @brief Plota uma equacao fornecida como uma std::string y=f(
641  /// x).
642  /// Escrever somente a funcao f(x) e nao y=
643  /// A variavel independente deve ser x.
644  /// Exemplo: gnuplot->PlotEquation(CFuncao& obj);
645  // Deve receber um CFuncao, que tem cast para string.
646  Gnuplot & PlotEquation(const std::string & equation,
647                         const std::string & title = "")
648                         { return
649                           plot_equation(
650                             equation, title );
651                         }
652
653  /// @brief Plota uma equacao fornecida na forma de uma std::
654  /// string z=f(x,y).
655  /// Escrever somente a funcao f(x,y) e nao z=, as variaveis
656  /// independentes sao x e y.

```

```
647 Gnuplot & plot_equation3d (const std::string & equation, const
648   std::string & title = "");
649
650   /// @brief Plota uma equacao fornecida na forma de uma std::
651   /// string z=f(x,y).
652   /// Escrever somente a funcao f(x,y) e nao z=, as vaiaveis
653   /// independentes sao x e y.
654   // gnuplot->PlotEquation3d(CPolinomio());
655   Gnuplot & PlotEquation3d (const std::string & equation,
656                             const std::string & title = "")
657
658   { return
659     plot_equation3d(
660       equation, title );
661   }
662
663
664
665   ///
666   -----
667
668   // Repete o ultimo comando de plotagem, seja plot (2D) ou splot
669   // (3D)
670
671   // Usado para visualizar plotagens, aps mudar algumas opcoes de
672   // plotagem
673
674   // ou quando gerando o mesmo grafico para diferentes dispositivos
675   // (showonscreen, savetops)
676   Gnuplot & replot ();
677
678
679   // Repete o ultimo comando de plotagem, seja plot (2D) ou splot
680   // (3D)
681
682   // Usado para visualizar plotagens, aps 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 #endif

```

---

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

---

Listing 6.4: 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(),
69    popen()
70#include <cstdlib> // for getenv()
71#include "CGnuplot.h"
72// Se estamos no windows // defined for 32 and 64-bit environments
73#if defined(WIN32) || defined(_WIN32) || defined(__WIN32__)
74    defined(__TOS_WIN__)
75    #include <iostream> // for cout, endl
76    #define GP_MAX_TMP_FILES 27 // 27 temporary files it's
77        Microsoft restriction
78// Se estamos no unix, GNU/Linux, Mac Os X
79#elif defined(unix) || defined(__unix) || defined(__unix__)
80    defined(__APPLE__) //all UNIX-like OSs (Linux, *BSD, MacOSX,
81        Solaris, ...)
82    #include <unistd.h> // for access(), mkstemp()
83    #define GP_MAX_TMP_FILES 64
84#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__)
92    std::string Gnuplot::m_sGNUPlotFileName = "gnuplot.exe";
93    std::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__)
96    std::string Gnuplot::m_sGNUPlotFileName = "gnuplot";
97    std::string Gnuplot::m_sGNUPlotPath = "/usr/bin/";
98#endif
99
100// Se estamos no windows
101#if defined(WIN32) || defined(_WIN32) || defined(__WIN32__)
102    std::string Gnuplot::terminal_std = "windows";
103// Se estamos no unix, GNU/Linux
104#elif ( defined(unix) || defined(__unix) || defined(__unix__) ) &&
105    !defined(__APPLE__)
106    std::string Gnuplot::terminal_std = "x11";
107// Se estamos Mac Os X
108#elif defined(__APPLE__)
109    std::string Gnuplot::terminal_std = "aqua";
110#endif
111// -----
112//
113// define static member function: set Gnuplot path manual
114//   for windows: path with slash '/' not backslash '\'
```

```

115 //
116 bool Gnuplot::set_GNUPlotPath(const std::string &path)
117 {
118     std::string tmp = path + "/" + Gnuplot::m_sGNUPlotFileName;
119 #if defined(WIN32) || defined(_WIN32) || defined(__WIN32__)
120     if (Gnuplot::file_exists(tmp, 0)) // check existence
121 #elif defined(unix) || defined(__unix) || defined(__unix__)
122     if (Gnuplot::file_exists(tmp, 1)) // check existence and
123         execution permission
124 #endif
125     {
126         Gnuplot::m_sGNUPlotPath = path;
127         return true;
128     }
129     else
130     {
131         Gnuplot::m_sGNUPlotPath.clear();
132         return false;
133     }
134
135 //

-----


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

```

```
149     return;
150 }
151
152
153 // -----
154 // A string tokenizer taken from http://www.sunsite.ualberta.ca/
155 // Documentation/
156 // /Gnu/libstdc++-2.90.8/html/21_strings/stringtok_std_h.txt
157 template <typename Container>
158 void strtok (Container &container,
159               std::string const &in,
160               const char * const delimiters = "\u0009\u000a\u000d");
161 {
162     const std::string::size_type len = in.length();
163     std::string::size_type i = 0;
164
165     while ( i < len )
166     {
167         // eat leading whitespace
168         i = in.find_first_not_of (delimiters, i);
169
170         if (i == std::string::npos)
171             return; // nothing left but white space
172
173         // find the end of the token
174         std::string::size_type j = in.find_first_of (delimiters, i)
175         ;
176
177         // push token
178         if (j == std::string::npos)
179         {
180             container.push_back (in.substr(i));
181             return;
182         }
183         else
184             container.push_back (in.substr(i, j-i));
185
186         // set up for next loop
187         i = j + 1;
188     }
189 }
```

```
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 //
203 // constructor: open a new session, plot a signal (x)
204 Gnuplot::Gnuplot(const std::vector<double> &x,
205                   const std::string &title,
206                   const std::string &style,
207                   const std::string &labelx,
208                   const std::string &labely)
209 {
210     this->init();
211
212     this->set_style(style);
213     this->set_xlabel(labelx);
214     this->set_ylabel(labely);
215
216     this->plot_x(x,title);
217 }
218 // -----
219 //
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__) ||
271     defined(__TOS_WIN__)
272     if (_pclose(this->gnucmd) == -1)
273 #elif defined(unix) || defined(__unix) || defined(__unix__)
274     defined(__APPLE__)
275     if (pclose(this->gnucmd) == -1)
276 #endif
277     true;//throw GnuplotException("Problem closing
278         communication to gnuplot");
279 }
280
281 // -----
282
283 // Resets a gnuplot session (next plot will erase previous ones)
284 Gnuplot& Gnuplot::reset_plot()
285 {
286     if (this->tmpfile_list.size() > 0)
287     {
288         for (unsigned int i = 0; i < this->tmpfile_list.size(); i++)
289             remove(this->tmpfile_list[i].c_str());
290
291         Gnuplot::tmpfile_num -= this->tmpfile_list.size();
292         this->tmpfile_list.clear();
293     }
294
295     this->nplots = 0;
296
297     return *this;
298 }
299
300
```

```
296 //  
-----  
  
297 // resets a gnuplot session and sets all variables to default  
298 Gnuplot& Gnuplot::reset_all()  
299 {  
300     if (this->tmpfile_list.size() > 0)  
301     {  
302         for (unsigned int i = 0; i < this->tmpfile_list.size(); i++)  
303             remove(this->tmpfile_list[i].c_str());  
304  
305         Gnuplot::tmpfile_num -= this->tmpfile_list.size();  
306         this->tmpfile_list.clear();  
307     }  
308  
309     this->nplots = 0;  
310     this->cmd("reset");  
311     this->cmd("clear");  
312     this->pstyle = "points";  
313     this->smooth = "";  
314     this->showonscreen();  
315  
316     return *this;  
317 }  
318  
319 //  
-----  
  
320 // Find out if valid is true  
321 bool Gnuplot::is_valid()  
322 {  
323     return (this->valid);  
324 }  
325  
326 //  
-----  
  
327 // replot repeats the last plot or splot command  
328 Gnuplot& Gnuplot::replot()  
329 {  
330     if (this->nplots > 0)
```

```

331     {
332         this->cmd("replot");
333     }
334
335     return *this;
336 }
337
338
339 // -----
340 // Change the plotting style of a gnuplot session
341 Gnuplot& Gnuplot::set_style(const std::string &stylestr)
342 {
343     if (stylestr.find("lines") == std::string::npos &&
344         stylestr.find("points") == std::string::npos &&
345         stylestr.find("linespoints") == std::string::npos &&
346         stylestr.find("impulses") == std::string::npos &&
347         stylestr.find("dots") == std::string::npos &&
348         stylestr.find("steps") == std::string::npos &&
349         stylestr.find("fsteps") == std::string::npos &&
350         stylestr.find("histeps") == std::string::npos &&
351         stylestr.find("boxes") == std::string::npos &&
352             // 1-4 columns of data are required
353             stylestr.find("filledcurves") == std::string::npos &&
354             stylestr.find("histograms") == std::string::npos )
355             //only for one data column
356             stylestr.find("labels") == std::string::npos &&
357             // 3 columns of data are required
358             stylestr.find("xerrorbars") == std::string::npos &&
359             // 3-4 columns of data are required
360             stylestr.find("xerrorlines") == std::string::npos &&
361             // 3-4 columns of data are required
362             stylestr.find("errorbars") == std::string::npos &&
363             // 3-4 columns of data are required
364             stylestr.find("errorlines") == std::string::npos &&
365             // 3-4 columns of data are required
366             stylestr.find("yerrorbars") == std::string::npos &&
367             // 3-4 columns of data are required
368             stylestr.find("yerrorlines") == std::string::npos &&
369             // 3-4 columns of data are required
370             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
427 //         de terminal (tipo de arquivo)
428 //     std::ostringstream cmdstr;                                // Muda
429 //         o nome do arquivo
430 //     cmdstr << "set output \"\" << filename << ".ps\"";    // Nome
```

```
        do arquivo
430 //      this->cmd(cmdstr.str());
431 //      this->replot();                                //
        Replot o gráfico, agora salvando o arquivo ps
432 //
433 //      ShowOnScreen ();                            // Volta
        para terminal modo janela
434
435     this->cmd("set term png size 1800,1200");
436
437     std::ostringstream cmdstr;
438     cmdstr << "set output \"./Src/" << filename << ".png\"";
439     this->cmd(cmdstr.str());
440     this->Replot();
441
442     return *this;
443 }
444 //

-----
445 // saves a gnuplot session to a png file and return do on screen
        terminal
446 Gnuplot& Gnuplot::savetopng(const std::string &filename)
447 {
448 //                                // Muda
        o terminal
449 //      this->cmd("set term png enhanced size 1280,960"); // Tipo
        de terminal (tipo de arquivo)
450 //
451 //      std::ostringstream cmdstr;                                // Muda
        o nome do arquivo
452 //      cmdstr << "set output \" " << filename << ".png\""; // Nome
        do arquivo
453 //      this->cmd(cmdstr.str());
454 //      this->replot();                                //
        Replot o gráfico, agora salvando o arquivo ps
455 //
456 //      ShowOnScreen ();                            // Volta
        para terminal modo janela
457 //      this->replot();                                //
        Replot o gráfico, agora na tela
```

```
458     SaveTo(filename, "png", "enhanced_size_1280,960");
459
460     return *this;
461 }
462
463 // -----
464 // saves a gnuplot session to a jpeg file and return do on screen
464 // terminal
465 Gnuplot& Gnuplot::savetojpeg(const std::string &filename)
466 {
467
468 //      this->cmd("set term jpeg enhanced size 1280,960");    // Tipo
468 // de terminal (tipo de arquivo)
469 //
470 //      std::ostringstream cmdstr;                                // Muda o
470 // o nome do arquivo
471 //      cmdstr << "set output \"\" << filename << ".jpeg\"\"; // Nome
471 // do arquivo
472 //      this->cmd(cmdstr.str());
473 //      this->replot();                                         // 
473 // Replota o gráfico, agora salvando o arquivo ps
474 // 
474 //      Retorna o terminal para o padrão janela
475 //      ShowOnScreen();                                         // Volta
475 // para terminal modo janela
476 //      this->replot();                                         // 
476 // Replota o gráfico, agora na tela
477 //      SaveTo(filename, "jpeg", "enhanced_size_1280,960");
478
479     return *this;
480 }
481
482
483 // -----
484 // saves a gnuplot session to specific terminal and output file
485 // @filename: name of disc file
486 // @terminal_type: type of terminal
```

```
487 // @flags: aditional information specitif to terminal type
488 // Ex:
489 // grafico.SaveTo("pressao_X_temperatura","png", "enhanced size
1280,960");
490 // grafico.TerminalType("png").SaveFile(pressao_X_temperatura);
pense nisso?
491 Gnuplot& Gnuplot::SaveTo(const std::string &filename,const std::
string &terminal_type, std::string flags)
492 {                                                 // Muda o
terminal
493     this->cmd("set term " + terminal_type + " " + flags);      //
Tipo de terminal (tipo de arquivo) e flags adicionais
494     std::ostringstream cmdstr;                                     // Muda o
nome do arquivo
495     cmdstr << "set output \" " << filename << " ." << terminal_type
<< "\" ";
496     this->cmd(cmdstr.str());
497     this->replot();                                              // Replota
o gráfico, agora salvando o arquivo ps
498                                                 // Retorna
o
terminal
para o
padrão
janela
499     ShowOnScreen ();
500     this->replot();                                              // Replota
o gráfico, agora na tela
501
502     return *this;
503 }
504
505 // -----
506 // Switches legend on
507 Gnuplot& Gnuplot::set_legend(const std::string &position)
508 {
509     std::ostringstream cmdstr;
510     cmdstr << "set key " << position;
511
512     this->cmd(cmdstr.str());
```

```
513
514     return *this;
515 }
516
517 // -----
518 // Switches legend off
519 Gnuplot& Gnuplot::unset_legend()
520 {
521     this->cmd("unset key");
522
523     return *this;
524 }
525
526 // -----
527 // Turns grid on
528 Gnuplot& Gnuplot::set_grid()
529 {
530     this->cmd("set grid");
531
532     return *this;
533 }
534
535 // -----
536 // Turns grid off
537 Gnuplot& Gnuplot::unset_grid()
538 {
539     this->cmd("unset grid");
540
541     return *this;
542 }
543
544 // -----
545 // turns on log scaling for the x axis
546 Gnuplot& Gnuplot::set_xlogscale(const double base)
```

```
547 {
548     std::ostringstream cmdstr;
549
550     cmdstr << "set logscale x" << base;
551     this->cmd(cmdstr.str());
552
553     return *this;
554 }
555
556 // -----
557 // turns on log scaling for the y axis
558 Gnuplot& Gnuplot::set_ylogscale(const double base)
559 {
560     std::ostringstream cmdstr;
561
562     cmdstr << "set logscale y" << base;
563     this->cmd(cmdstr.str());
564
565     return *this;
566 }
567
568 // -----
569 // turns on log scaling for the z axis
570 Gnuplot& Gnuplot::set_zlogscale(const double base)
571 {
572     std::ostringstream cmdstr;
573
574     cmdstr << "set logscale z" << base;
575     this->cmd(cmdstr.str());
576
577     return *this;
578 }
579
580 // -----
581 // turns off log scaling for the x axis
582 Gnuplot& Gnuplot::unset_xlogscale()
```

```
583 {
584     this->cmd("unset logscale_x");
585     return *this;
586 }
587
588 // -----
589 // turns off log scaling for the y axis
590 Gnuplot& Gnuplot::unset_ylogscale()
591 {
592     this->cmd("unset logscale_y");
593     return *this;
594 }
595
596 // -----
597 // turns off log scaling for the z axis
598 Gnuplot& Gnuplot::unset_zlogscale()
599 {
600     this->cmd("unset logscale_z");
601     return *this;
602 }
603
604
605 // -----
606 // scales the size of the points used in plots
607 Gnuplot& Gnuplot::set_pointsize(const double pointsize)
608 {
609     std::ostringstream cmdstr;
610     cmdstr << "set pointsize " << pointsize;
611     this->cmd(cmdstr.str());
612
613     return *this;
614 }
615
616 // -----
```

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

628 // set isoline density (grid) for plotting functions as surfaces
629 Gnuplot& Gnuplot::set_isosamples(const int isolines)
630 {
631     std::ostringstream cmdstr;
632     cmdstr << "set isosamples" << isolines;
633     this->cmd(cmdstr.str());
634
635     return *this;
636 }
637
638 //
-----  

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

648 // disables hidden line removal for surface plotting
649 Gnuplot& Gnuplot::unset_hidden3d()
650 {
651     this->cmd("unset hidden3d");
652 }
```

```
653     return *this;
654 }
655
656 // -----
657 // enables contour drawing for surfaces set contour {base | surface
658 // | both}
659 Gnuplot& Gnuplot::set_contour(const std::string &position)
660 {
661     if (position.find("base") == std::string::npos &&
662         position.find("surface") == std::string::npos &&
663         position.find("both") == std::string::npos )
664     {
665         this->cmd("set contour base");
666     }
667     else
668     {
669         this->cmd("set contour " + position);
670     }
671     return *this;
672 }
673
674 // -----
675 // disables contour drawing for surfaces
676 Gnuplot& Gnuplot::unset_contour()
677 {
678     this->cmd("unset contour");
679
680     return *this;
681 }
682
683 // -----
684 // enables the display of surfaces (for 3d plot)
685 Gnuplot& Gnuplot::set_surface()
686 {
687     this->cmd("set surface");
```

```
688
689     return *this;
690 }
691
692 // -----
693 // disables the display of surfaces (for 3d plot)
694 Gnuplot& Gnuplot::unset_surface()
695 {
696     this->cmd("unset surface");
697
698     return *this;
699 }
700
701 // -----
702 // Sets the title of a gnuplot session
703 Gnuplot& Gnuplot::set_title(const std::string &title)
704 {
705     std::ostringstream cmdstr;
706
707     cmdstr << "set title " << title << "\n";
708     this->cmd(cmdstr.str());
709
710     return *this;
711 }
712
713 // -----
714 // Clears the title of a gnuplot session
715 Gnuplot& Gnuplot::unset_title()
716 {
717     this->set_title("");
718
719     return *this;
720 }
721
722 // -----
```

```
723 // set labels
724 // set the xlabel
725 Gnuplot& Gnuplot::set_xlabel(const std::string &label)
726 {
727     std::ostringstream cmdstr;
728
729     cmdstr << "set xlabel \"\" << label << \"\";";
730     this->cmd(cmdstr.str());
731
732     return *this;
733 }
734
735 // -----
736 // set the ylabel
737 Gnuplot& Gnuplot::set_ylabel(const std::string &label)
738 {
739     std::ostringstream cmdstr;
740
741     cmdstr << "set ylabel \"\" << label << \"\";";
742     this->cmd(cmdstr.str());
743
744     return *this;
745 }
746
747 // -----
748 // set the zlabel
749 Gnuplot& Gnuplot::set_zlabel(const std::string &label)
750 {
751     std::ostringstream cmdstr;
752
753     cmdstr << "set zlabel \"\" << label << \"\";";
754     this->cmd(cmdstr.str());
755
756     return *this;
757 }
758
759 //
```

```
-----  
760 // set range  
761 // set the xrange  
762 Gnuplot& Gnuplot::set_xrange(const int iFrom,  
763                               const int iTo)  
764 {  
765     std::ostringstream cmdstr;  
766  
767     cmdstr << "set xrange[" << iFrom << ":" << iTo << "]";  
768     this->cmd(cmdstr.str());  
769  
770     return *this;  
771 }  
772  
773 //  
-----  
  
774 // set autoscale x  
775 Gnuplot& Gnuplot::set_xautoscale()  
776 {  
777     this->cmd("set xrange restore");  
778     this->cmd("set autoscale x");  
779  
780     return *this;  
781 }  
782  
783 //  
-----  
  
784 // set the yrange  
785 Gnuplot& Gnuplot::set_yrange(const int iFrom, const int iTo)  
786 {  
787     std::ostringstream cmdstr;  
788  
789     cmdstr << "set yrange[" << iFrom << ":" << iTo << "]";  
790     this->cmd(cmdstr.str());  
791  
792     return *this;  
793 }  
794  
795 //
```

```
-----  
796 // set autoscale y  
797 Gnuplot& Gnuplot::set_yautoscale()  
798 {  
799     this->cmd("set_yrange_restore");  
800     this->cmd("set autoscale_y");  
801  
802     return *this;  
803 }  
804  
805 //  
-----  
  
806 // set the zrange  
807 Gnuplot& Gnuplot::set_zrange(const int iFrom,  
808                               const int iTo)  
809 {  
810     std::ostringstream cmdstr;  
811  
812     cmdstr << "set_zrange[" << iFrom << ":" << iTo << "]";  
813     this->cmd(cmdstr.str());  
814  
815     return *this;  
816 }  
817  
818 //  
-----  
  
819 // set autoscale z  
820 Gnuplot& Gnuplot::set_zautoscale()  
821 {  
822     this->cmd("set_zrange_restore");  
823     this->cmd("set autoscale_z");  
824  
825     return *this;  
826 }  
827  
828 //  
-----  
829 // set the palette range
```

```
830 Gnuplot& Gnuplot::set_cbrange(const int iFrom,
831                               const int iTo)
832 {
833     std::ostringstream cmdstr;
834
835     cmdstr << "set_cbrange[" << iFrom << ":" << iTo << "]";
836     this->cmd(cmdstr.str());
837
838     return *this;
839 }
840
841 // -----
842 // Plots a linear equation y=ax+b (where you supply the
843 // slope a and intercept b)
844 Gnuplot& Gnuplot::plot_slope(const double a,
845                               const double b,
846                               const std::string &title)
847 {
848     std::ostringstream cmdstr;
849
850     // command to be sent to gnuplot
851     if (this->nplots > 0 && this->two_dim == true)
852         cmdstr << "replot";
853     else
854         cmdstr << "plot";
855
856     cmdstr << a << "*x+" << b << "title\"";
857
858     if (title == "")
859         cmdstr << "f(x)=" << a << "*x+" << b;
860     else
861         cmdstr << title;
862
863     cmdstr << "\with" << this->pstyle;
864
865     // Do the actual plot
866     this->cmd(cmdstr.str());
867
868     return *this;
869 }
```

```
870
871 // -----
872 // Plot an equation which is supplied as a std::string y=f(x) (only
873 // f(x) expected)
874 Gnuplot& Gnuplot::plot_equation(const std::string &equation,
875                                     const std::string &title)
876 {
877     std::ostringstream cmdstr;
878
879     // command to be sent to gnuplot
880     if (this->nplots > 0 && this->two_dim == true)
881         cmdstr << "replot";
882     else
883         cmdstr << "plot";
884
885     cmdstr << equation << "\title\"\"";
886
887     if (title == "")
888         cmdstr << "f(x)=\" " << equation;
889     else
890         cmdstr << title;
891
892     cmdstr << "\with" << this->pstyle;
893
894     // Do the actual plot
895     this->cmd(cmdstr.str());
896
897 }
898
899 // -----
900 // plot an equation supplied as a std::string y=(x)
901 Gnuplot& Gnuplot::plot_equation3d(const std::string &equation,
902                                     const std::string &title)
903 {
904     std::ostringstream cmdstr;
905
906     // command to be sent to gnuplot
```

```

907     if (this->nplots > 0 && this->two_dim == false)
908         cmdstr << "replot";
909     else
910         cmdstr << "splot";
911
912     cmdstr << equation << "title\"\"";
913
914     if (title == "")
915         cmdstr << "f(x,y)=\" " << equation;
916     else
917         cmdstr << title;
918
919     cmdstr << "\\"with" << this->pstyle;
920
921     // Do the actual plot
922     this->cmd(cmdstr.str());
923
924     return *this;
925 }
926
927 // -----
928 // Plots a 2d graph from a list of doubles (x) saved in a file
929 Gnuplot& Gnuplot::plotfile_x(const std::string &filename,
930                               const int column,
931                               const std::string &title)
932 {
933     // check if file exists
934     if( !(Gnuplot::file_exists(filename,4)) ) // check existence
935         and read permission
936     {
937         std::ostringstream except;
938         if( !(Gnuplot::file_exists(filename,0)) ) // check
939             existence
940             except << "File\"\"" << filename << "\\"does\\not\\exist";
941         else
942             except << "No\\read\\permission\\for\\File\\\"\"" << filename
943             << "\\"";
944         throw GnuplotException( except.str() );
945     }
946     return *this;
947 }
```

```
944
945     std::ostringstream cmdstr;
946
947     // command to be sent to gnuplot
948     if (this->nplots > 0 && this->two_dim == true)
949         cmdstr << "replot";
950     else
951         cmdstr << "plot";
952
953     cmdstr << "\n" << filename << "\"using" << column;
954
955     if (title == "")
956         cmdstr << "notitle";
957     else
958         cmdstr << "title\" " << title << "\"";
959
960     if (smooth == "")
961         cmdstr << "with" << this->pstyle;
962     else
963         cmdstr << "smooth" << this->smooth;
964
965     // Do the actual plot
966     this->cmd(cmdstr.str()); //nplots++; two_dim = true; already
967     in this->cmd();
968
969 }
970
971 //
-----  

972 // Plots a 2d graph from a list of doubles: x
973 Gnuplot& Gnuplot::plot_x(const std::vector<double> &x,
974                           const std::string &title)
975 {
976     if (x.size() == 0)
977     {
978         throw GnuplotException("std::vector too small");
979         return *this;
980     }
981
982     std::ofstream tmp;
```

```
983     std::string name = create_tmpfile(tmp);
984     if (name == "")
985         return *this;
986
987     // write the data to file
988     for (unsigned int i = 0; i < x.size(); i++)
989         tmp << x[i] << std::endl;
990
991     tmp.flush();
992     tmp.close();
993
994     this->plotfile_x(name, 1, title);
995
996     return *this;
997 }
998
999 // -----
1000 // Plots a 2d graph from a list of doubles (x y) saved in a file
1001 Gnuplot& Gnuplot::plotfile_xy(const std::string &filename,
1002                               const int column_x,
1003                               const int column_y,
1004                               const std::string &title)
1005 {
1006     // check if file exists
1007     if( !(Gnuplot::file_exists(filename,4)) ) // check existence
1008         and read permission
1009     {
1010         std::ostringstream except;
1011         if( !(Gnuplot::file_exists(filename,0)) ) // check
1012             existence
1013             except << "File \" " << filename << "\" does not exist";
1014         else
1015             except << "No read permission for File \" " << filename
1016             << "\\"";
1017         throw GnuplotException( except.str() );
1018         return *this;
1019     }
1020
1021     std::ostringstream cmdstr;
```

```
1020     // command to be sent to gnuplot
1021     if (this->nplots > 0 && this->two_dim == true)
1022         cmdstr << "replot";
1023     else
1024         cmdstr << "plot";
1025
1026     cmdstr << "\n" << filename << "\using" << column_x << ":" <<
1027         column_y;
1028
1029     if (title == "")
1030         cmdstr << "notitle";
1031     else
1032         cmdstr << "title\"" << title << "\"";
1033
1034     if (smooth == "")
1035         cmdstr << "with" << this->pstyle;
1036     else
1037         cmdstr << "smooth" << this->smooth;
1038
1039     // Do the actual plot
1040     this->cmd(cmdstr.str());
1041
1042 }
1043
1044 // -----
1045 // Plots a 2d graph from a list of doubles: x y
1046 Gnuplot& Gnuplot::plot_xy(const std::vector<double> &x,
1047                             const std::vector<double> &y,
1048                             const std::string &title)
1049 {
1050     if (x.size() == 0 || y.size() == 0)
1051     {
1052         throw GnuplotException("std::vectors too small");
1053         return *this;
1054     }
1055
1056     if (x.size() != y.size())
1057     {
1058         throw GnuplotException("Length of the std::vectors differs")
```

```

        );
1059     return *this;
1060 }
1061
1062     std::ofstream tmp;
1063     std::string name = create_tmpfile(tmp);
1064     if (name == "")
1065         return *this;
1066
1067     // write the data to file
1068     for (unsigned int i = 0; i < x.size(); i++)
1069         tmp << x[i] << " " << y[i] << std::endl;
1070
1071     tmp.flush();
1072     tmp.close();
1073
1074     this->plotfile_xy(name, 1, 2, title);
1075
1076     return *this;
1077 }
1078
1079 // -----
1080 // Plots a 2d graph with errorbars from a list of doubles (x y dy)
1081 // saved in a file
1082 Gnuplot& Gnuplot::plotfile_xy_err(const std::string &filename,
1083                                     const int column_x,
1084                                     const int column_y,
1085                                     const int column_dy,
1086                                     const std::string &title)
1087 {
1088     // check if file exists
1089     if( !(Gnuplot::file_exists(filename,4)) ) // check existence
1090         and read permission
1091     {
1092         std::ostringstream except;
1093         if( !(Gnuplot::file_exists(filename,0)) ) // check
1094             existence
1095             except << "File " << filename << "\" does not exist";
1096         else
1097             except << "No read permission for File " << filename

```

```

        << "\\" ;
1095     throw GnuplotException( except.str() );
1096     return *this;
1097 }
1098
1099 std::ostringstream cmdstr;
1100
1101 // command to be sent to gnuplot
1102 if (this->nplots > 0 && this->two_dim == true)
1103     cmdstr << "replot" ;
1104 else
1105     cmdstr << "plot" ;
1106
1107 cmdstr << "\\" << filename << "\\" using " << column_x << ":" <<
1108     column_y;
1109
1110 if (title == "")
1111     cmdstr << "\notitle" ;
1112 else
1113     cmdstr << "\title\" << title << "\\" ;
1114
1115 cmdstr << "with" << this->pstyle << ",\\" << filename << "\\" using "
1116     << column_x << ":" << column_y << ":" << column_dy << "\"
1117     notitle with errorbars";
1118
1119 // Do the actual plot
1120 this->cmd(cmdstr.str());
1121
1122
1123 // -----
1124
1125 // plot x,y pairs with dy errorbars
1126 Gnuplot& Gnuplot::plot_xy_err(const std::vector<double> &x,
1127                                     const std::vector<double> &y,
1128                                     const std::vector<double> &dy,
1129                                     const std::string &title)
1130 {
1131     if (x.size() == 0 || y.size() == 0 || dy.size() == 0)

```

```

1131     {
1132         throw GnuplotException("std::vectors too small");
1133         return *this;
1134     }
1135
1136     if (x.size() != y.size() || y.size() != dy.size())
1137     {
1138         throw GnuplotException("Length of the std::vectors differs"
1139             );
1140         return *this;
1141     }
1142
1143     std::ofstream tmp;
1144     std::string name = create_tmpfile(tmp);
1145     if (name == "")
1146         return *this;
1147
1148     // write the data to file
1149     for (unsigned int i = 0; i < x.size(); i++)
1150         tmp << x[i] << " " << y[i] << " " << dy[i] << std::endl;
1151
1152     tmp.flush();
1153     tmp.close();
1154
1155     // Do the actual plot
1156     this->plotfile_xy_err(name, 1, 2, 3, title);
1157
1158 }
1159
1160 // -----
1161 // Plots a 3d graph from a list of doubles (x y z) saved in a file
1162 Gnuplot& Gnuplot::plotfile_xyz(const std::string &filename,
1163                                 const int column_x,
1164                                 const int column_y,
1165                                 const int column_z,
1166                                 const std::string &title)
1167 {
1168
1169     // check if file exists

```

```

1170     if( !(Gnuplot::file_exists(filename,4)) ) // check existence
1171         and read permission
1172     {
1173         std::ostringstream except;
1174         if( !(Gnuplot::file_exists(filename,0)) ) // check
1175             existence
1176             except << "File \"\\" << filename << "\" does not exist";
1177         else
1178             except << "No read permission for File \"\\" << filename
1179             << "\\"";
1180         throw GnuplotException( except.str() );
1181         return *this;
1182     }
1183
1184     std::ostringstream cmdstr;
1185
1186     // command to be sent to gnuplot
1187     if (this->nplots > 0 && this->two_dim == false)
1188         cmdstr << "replot";
1189     else
1190         cmdstr << "splot";
1191
1192     cmdstr << "\\" << filename << "\" using " << column_x << ":" <<
1193         column_y << ":" << column_z;
1194
1195     if (title == "")
1196         cmdstr << "notitle with " << this->pstyle;
1197     else
1198         cmdstr << "title \"\\" << title << "\" with " << this->
1199             pstyle;
1200
1201
1202 // -----
1203 // Plots a 3d graph from a list of doubles: x y z
1204 Gnuplot& Gnuplot::plot_xyz(const std::vector<double> &x,

```

```
1205         const std::vector<double> &y,
1206         const std::vector<double> &z,
1207         const std::string &title)
1208 {
1209     if (x.size() == 0 || y.size() == 0 || z.size() == 0)
1210     {
1211         throw GnuplotException("std::vectors too small");
1212         return *this;
1213     }
1214
1215     if (x.size() != y.size() || x.size() != z.size())
1216     {
1217         throw GnuplotException("Length of the std::vectors differs"
1218                               );
1219         return *this;
1220     }
1221
1222     std::ofstream tmp;
1223     std::string name = create_tmpfile(tmp);
1224     if (name == "")
1225         return *this;
1226
1227     // write the data to file
1228     for (unsigned int i = 0; i < x.size(); i++)
1229     {
1230         tmp << x[i] << " " << y[i] << " " << z[i] << std::endl;
1231     }
1232
1233     tmp.flush();
1234     tmp.close();
1235
1236
1237     this->plotfile_xyz(name, 1, 2, 3, title);
1238
1239     return *this;
1240 }
1241
1242
1243
1244 //
```

---

```
1245 // * note that this function is not valid for versions of Gnuplot
1246 // below 4.2
1247
1248
1249
1250 {
1251     std::ofstream tmp;
1252     std::string name = create_tmpfile(tmp);
1253     if (name == "")
1254         return *this;
1255
1256     // write the data to file
1257     int iIndex = 0;
1258     for(int iRow = 0; iRow < iHeight; iRow++)
1259     {
1260         for(int iColumn = 0; iColumn < iWidth; iColumn++)
1261         {
1262             tmp << iColumn << "\u00d7" << iRow << "\u00d7" << static_cast<
1263             float>(ucPicBuf[iIndex++]) << std::endl;
1264         }
1265     }
1266     tmp.flush();
1267     tmp.close();
1268
1269
1270     std::ostringstream cmdstr;
1271
1272     // command to be sent to gnuplot
1273     if (this->nplots > 0 && this->two_dim == true)
1274         cmdstr << "replot\u00d7";
1275     else
1276         cmdstr << "plot\u00d7";
1277
1278     if (title == "")
1279         cmdstr << "\\" << name << "\\with\u00d7image";
1280     else
1281         cmdstr << "\\" << name << "\\title\u00d7\" << title << "\\u
1282         with\u00d7image";
```

```
1283     // Do the actual plot
1284     this->cmd(cmdstr.str());
1285
1286     return *this;
1287 }
1288
1289 // -----
1290 // Sends a command to an active gnuplot session
1291 Gnuplot& Gnuplot::cmd(const std::string &cmdstr)
1292 {
1293     if( !(this->valid) )
1294     {
1295         return *this;
1296     }
1297
1298     // int fputs ( const char * str, FILE * stream );
1299     // writes the string str to the stream.
1300     // The function begins copying from the address specified (str)
1301     // until it reaches the
1302     // terminating null character ('\0'). This final null-character
1303     // is not copied to the stream.
1304     fputs( (cmdstr+"\n").c_str(), this->gnucmd );
1305
1306     // int fflush ( FILE * stream );
1307     // If the given stream was open for writing and the last i/o
1308     // operation was an output operation,
1309     // any unwritten data in the output buffer is written to the
1310     // file.
1311     // If the argument is a null pointer, all open files are
1312     // flushed.
1313     // The stream remains open after this call.
1314     fflush(this->gnucmd);
1315
1316     if( cmdstr.find("replot") != std::string::npos )
1317     {
1318         return *this;
1319     }
1320     else if( cmdstr.find("splot") != std::string::npos )
1321     {
```

```
1318     this->two_dim = false;
1319     this->nplots++;
1320 }
1321 else if( cmdstr.find("plot") != std::string::npos )
1322 {
1323     this->two_dim = true;
1324     this->nplots++;
1325 }
1326 return *this;
1327 }
1328
1329 // -----
1330 // Sends a command to an active gnuplot session, identical to cmd()
1331 Gnuplot& Gnuplot::operator<<(const std::string &cmdstr)
1332 {
1333     this->cmd(cmdstr);
1334     return *this;
1335 }
1336
1337 // -----
1338 // Opens up a gnuplot session, ready to receive commands
1339 void Gnuplot::init()
1340 {
1341     // char * getenv ( const char * name );   get value of an
1342     // environment variable
1343     // Retrieves a C string containing the value of the environment
1344     // variable whose
1345     // name is specified as argument.
1346     // If the requested variable is not part of the environment
1347     // list, the function returns a NULL pointer.
1348 #if ( defined(unix) || defined(__unix) || defined(__unix__) ) && !
1349     defined(__APPLE__)
1350     if (getenv("DISPLAY") == NULL)
1351     {
1352         this->valid = false;
1353         throw GnuplotException("Can't find DISPLAY variable");
1354     }
1355 #endif
```

```
1352
1353     // if gnuplot not available
1354     if (!Gnuplot::get_program_path())
1355     {
1356         this->valid = false;
1357         throw GnuplotException("Can't find gnuplot");
1358     }
1359
1360     // open pipe
1361     std::string tmp = Gnuplot::m_sGNUPlotPath + "/" + Gnuplot::
1362             m_sGNUPlotFileName;
1363
1364     // FILE *popen(const char *command, const char *mode);
1365     // The popen() function shall execute the command specified by
1366     // the string command,
1367     // create a pipe between the calling program and the executed
1368     // command, and
1369     // return a pointer to a stream that can be used to either read
1370     // from or write to the pipe.
1371 #if defined(WIN32) || defined(_WIN32) || defined(__WIN32__)
1372     defined(__TOS_WIN__)
1373     this->gnucmd = _popen(tmp.c_str(),"w");
1374 #elif defined(unix) || defined(__unix) || defined(__unix__)
1375     defined(__APPLE__)
1376     this->gnucmd = popen(tmp.c_str(),"w");
1377 #endif
1378
1379     // popen() shall return a pointer to an open stream that can be
1380     // used to read or write to the pipe.
1381     // Otherwise, it shall return a null pointer and may set errno
1382     // to indicate the error.
1383     if (!this->gnucmd)
1384     {
1385         this->valid = false;
1386         throw GnuplotException("Couldn't open connection to gnuplot
1387             ");
1388     }
1389
1390     this->nplots = 0;
1391     this->valid = true;
1392     this->smooth = "";
1393
1394
```

```
1385     //set terminal type
1386     this->showonscreen();
1387
1388     return;
1389 }
1390
1391 // -----
1392 // Find out if a command lives in m_sGNUPlotPath or in PATH
1393 bool Gnuplot::get_program_path()
1394 {
1395     // first look in m_sGNUPlotPath for Gnuplot
1396     std::string tmp = Gnuplot::m_sGNUPlotPath + "/" + Gnuplot::
1397         m_sGNUPlotFileName;
1398 #if defined(WIN32) || defined(_WIN32) || defined(__WIN32__)
1399     if ( Gnuplot::file_exists(tmp,0) ) // check existence
1400 #elif defined(unix) || defined(__unix) || defined(__unix__)
1401     if ( Gnuplot::file_exists(tmp,1) ) // check existence and
1402         execution permission
1403 #endif
1404 {
1405     return true;
1406 }
1407 // second look in PATH for Gnuplot
1408 char *path;
1409 // Retrieves a C string containing the value of the environment
1410 // variable PATH
1411 path = getenv("PATH");
1412
1413 if (path == NULL)
1414 {
1415     throw GnuplotException("Path is not set");
1416     return false;
1417 }
1418 {
1419     std::list<std::string> ls;
```

```
1420         //split path (one long string) into list ls of strings
1421 #if defined(WIN32) || defined(_WIN32) || defined(__WIN32__) ||
1422     defined(__TOS_WIN__)
1423     strtok(ls,path,";");
1424 #elif defined(unix) || defined(__unix) || defined(__unix__)
1425     defined(__APPLE__)
1426     strtok(ls,path,":");
1427 #endif
1428
1429         // scan list for Gnuplot program files
1430         for (std::list<std::string>::const_iterator i = ls.begin();
1431             i != ls.end(); ++i)
1432         {
1433             tmp = (*i) + "/" + Gnuplot::m_sGNUMPlotFileName;
1434 #if defined(WIN32) || defined(_WIN32) || defined(__WIN32__) ||
1435     defined(__TOS_WIN__)
1436             if ( Gnuplot::file_exists(tmp,0) ) // check existence
1437 #elif defined(unix) || defined(__unix) || defined(__unix__)
1438     defined(__APPLE__)
1439             if ( Gnuplot::file_exists(tmp,1) ) // check existence
1440                 and execution permission
1441 #endif
1442             {
1443                 Gnuplot::m_sGNUMPlotPath = *i; // set m_sGNUMPlotPath
1444                 return true;
1445             }
1446         }
1447
1448
1449 // -----
1450 // check if file exists
1451 bool Gnuplot::file_exists(const std::string &filename, int mode)
1452 {
```

```

1453     if ( mode < 0 || mode > 7)
1454     {
1455         throw std::runtime_error("In function \"Gnuplot::"
1456                                 "file_exists\": mode has to be an integer between 0 and 7"
1457                                 );
1458
1459         return false;
1460     }
1461
1462     // int _access(const char *path, int mode);
1463     // returns 0 if the file has the given mode,
1464     // it returns -1 if the named file does not exist or is not
1465     // accessible in the given mode
1466     // mode = 0 (F_OK) (default): checks file for existence only
1467     // mode = 1 (X_OK): execution permission
1468     // mode = 2 (W_OK): write permission
1469     // mode = 4 (R_OK): read permission
1470     // mode = 6       : read and write permission
1471     // mode = 7       : read, write and execution permission
1472 #if defined(WIN32) || defined(_WIN32) || defined(__WIN32__)
1473     defined(__TOS_WIN__)
1474     if (_access(filename.c_str(), mode) == 0)
1475 #elif defined(unix) || defined(__unix) || defined(__unix__)
1476     defined(__APPLE__)
1477     if (access(filename.c_str(), mode) == 0)
1478 #endif
1479     {
1480         return true;
1481     }
1482     else
1483     {
1484         return false;
1485     }
1486 }
1487
-----
```

```

1484 // Opens a temporary file
1485 std::string Gnuplot::create_tmpfile(std::ofstream &tmp)
1486 {
1487 #if defined(WIN32) || defined(_WIN32) || defined(__WIN32__)

```

```
defined(_TOS_WIN_)

1488     char name[] = "gnuplotXXXXXX"; //tmp file in working directory
1489 #elif defined(unix) || defined(__unix) || defined(__unix__)
1490     char name[] = "/tmp/gnuplotXXXXXX"; // tmp file in /tmp
1491#endif

1492
1493     // check if maximum number of temporary files reached
1494     if (Gnuplot::tmpfile_num == GP_MAX_TMP_FILES - 1)
1495     {
1496         std::ostringstream except;
1497         except << "Maximum number of temporary files reached"
1498             << GP_MAX_TMP_FILES
1499             << " cannot open more files" << std::endl;
1500
1501         throw GnuplotException( except.str() );
1502     }
1503
1504     //int mkstemp(char *name);
1505     // shall replace the contents of the string pointed to by "name"
1506     // by a unique filename,
1507     // and return a file descriptor for the file open for reading
1508     // and writing.
1509     // Otherwise, -1 shall be returned if no suitable file could be
1510     // created.
1511     // The string in template should look like a filename with six
1512     // trailing 'X' s;
1513     // mkstemp() replaces each 'X' with a character from the
1514     // portable filename character set.
1515     // The characters are chosen such that the resulting name does
1516     // not duplicate the name of an existing file at the time of a
1517     // call to mkstemp()

1518
1519
1520     // open temporary files for output
1521 #if defined(WIN32) || defined(_WIN32) || defined(__WIN32__)
1522     defined(_TOS_WIN_)
1523     if (_mktemp(name) == NULL)
1524 #elif defined(unix) || defined(__unix) || defined(__unix__)
1525     defined(__APPLE__)
1526     if (mkstemp(name) == -1)
```

```

1518 #endif
1519 {
1520     std::ostringstream except;
1521     except << "Cannot create temporary file \""
1522         << name << "\""
1523         ;
1524     throw GnuplotException(except.str());
1525     return "";
1526 }
1527 tmp.open(name);
1528 if (tmp.bad())
1529 {
1530     std::ostringstream except;
1531     except << "Cannot create temporary file \""
1532         << name << "\""
1533         ;
1534     throw GnuplotException(except.str());
1535     return "";
1536 }
1537 // Save the temporary filename
1538 this->tmpfile_list.push_back(name);
1539 Gnuplot::tmpfile_num++;
1540

```

---

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

Listing 6.5: Arquivo de cabeçalho da classe CInvNumStehfest.

```

1 #ifndef CInvNumStehfest_H_
2 #define CInvNumStehfest_H_
3
4 #include <vector>
5
6 #include "CReservatorioRadialInfinito.h"
7 #include "CReservatorioRadialSelado.h"
8 #include "CReservatorioRadialManutencao.h"
9 #include "CReservatorioLinearInfinito.h"
10 #include "CReservatorioLinearSelado.h"
11 #include "CReservatorioLinearManutencao.h"
12
13 //Criacao da classe CInvNumStehfest.h
14 class CInvNumStehfest

```

```
15 {
16     ///Declaracao metodos protegidos
17     protected:
18
19         ///Variaveis auxiliares do metodo de inversao
20         ///numerica de Stehfest
21         double nn2, nn21, z, ln2_on_t, sum, p, ilt, L, Rd;
22         std::vector <double> v;
23
24         ///Declaracoes dos objetos das classes
25         CReservatorioRadialInfinito radialinfinito;
26         CReservatorioRadialSelado radialselado;
27         CReservatorioRadialManutencao radialmanutencao;
28         CReservatorioLinearInfinito linearinfinito;
29         CReservatorioLinearSelado linearselado;
30         CReservatorioLinearManutencao linearmanutencao;
31
32     ///Declaracao metodos publicos
33     public:
34
35         ///Construtor
36         CInvNumStehfest(double _RD = 1, double _L = 12) : L
37             (_L), Rd(_RD){};
38
39         /**
40             Calcula fatorial
41             @parametro i inteiro a ser fatorado
42             @retorna o inteiro da variavel i fatorado
43         */
44         double Factorial(int _i);
45
46         /**
47             Calcula a inversao numerica de Stehfest para a
48             forma de reservatorio radial infinito
49             @parametro TD variavel do tempo adimensional no
50             campo real
51             @retorna a inversao numerica do algoritmo de
52             Stehfest no campo real
53         */
54         double StehfestRadialInfinito(double _TD);
```

```
52             Calcula a inversao numerica de Stehfest para a
53             forma de reservatorio radial selado
54             @parametro TD variavel do tempo adimensional no
55             campo real
56             @parametro RD variavel do raio externo
57             adimensional
58             @retorna a inversao numerica do algoritmo de
59             Stehfest no campo real
60             */
61             double StehfestRadialSelado (double _TD, double _RD
62             );
63
64             /**
65             Calcula a inversao numerica de Stehfest para a
66             forma de reservatorio radial com manutencao
67             @parametro TD variavel do tempo adimensional no
68             campo real
69             @parametro RD variavel do raio externo
70             adimensional
71             @retorna a inversao numerica do algoritmo de
72             Stehfest no campo real
73             */
74             double StehfestRadialManutencao (double _TD, double
75             _RD);
76
77             /**
78             Calcula a forma de reservatorio linear infinito
79             @parametro xd variavel do tempo adimensional
80             @retorna valor para o reservatorio linear
81             infinito
82             */
83             double StehfestLinearInfinito (double _xd);
84
85             /**
86             Calcula a forma de reservatorio linear selado
87             @parametro xd variavel da posicao (espaco)
88             adimensional
89             @retorna valor para o reservatorio linear selado
90             */
91             double StehfestLinearSelado (double _xd);
92
93             /**
94
```

```

82             Calcula a forma de reservatorio linear com
83                 manutencao
84             @parametro xd variavel da posicao (espaço)
85                 adimensional
86             @retorna valor para o reservatorio linear com
87                 manutencao
88         */
89         double StehfestLinearManutencao (double _xd);
90     };
91
92 #endif

```

---

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

Listing 6.6: Arquivo de implementação da classe `CInvNumStehfest`.

```

1 #define _USE_MATH_DEFINES // define constantes matematicas (exemplo
2     Pi = 3.141516...)
3
4 #include <cmath> ///inclui biblioteca matematica padrao com
5     funcoes bessel para c++17 ou superior
6
7
8 using namespace std;
9
10 double CInvNumStehfest::Factorial(int _i)
11 {
12     double acumulador = 1;
13     if (_i==0 || _i==1)
14         return acumulador;
15     else
16     {
17         for (double j = 1; j <= _i; j++)
18             acumulador *=j;
19     }
20 }
21
22

```

```
23 double CInvNumStehfest::StehfestRadialInfinito(double _TD)
24 {
25
26 nn2 = L/2.0;
27 nn21= nn2+1.0;
28
29 for (double n=1 ; n<=L; n++)
30 {
31     z=0.0;
32     for (int k = floor( ( n + 1.0 ) / 2.0 ); k <= min(n,nn2); k++)
33         z = z + ((pow(k, nn2))*Factorial(2*k))/(Factorial(
34             nn2-k)*Factorial(k)*Factorial(k-1)*Factorial(n-
35                 k)*Factorial(2*k - n));
36
37 }
38 sum = 0.0;
39 ln2_on_t = log(2.0) / _TD;
40
41 for (double n = 1.0; n <= L; n++)
42 {
43     p = n * ln2_on_t;
44     sum = sum + v[n-1.0]*radialinfinito.RadialInfinito(p, Rd);
45 }
46     ilt = sum * ln2_on_t;
47     return ilt;
48 }
49
50 double CInvNumStehfest::StehfestRadialSelado (double _TD, double
51 _RD)
52 {
53 nn2 = L/2.0;
54 nn21= nn2+1.0;
55
56 for (double n=1 ; n<=L; n++)
57 {
58     z=0.0;
59     for (int k = floor( ( n + 1.0 ) / 2.0 ); k <= min(n,nn2); k++)
60         z = z + ((pow(k, nn2))*Factorial(2*k))/(Factorial(
61             nn2-k)*Factorial(k)*Factorial(k-1)*Factorial(n-
62                 k)*Factorial(2*k - n));
```

```
60             z = z + ((pow(k, nn2))*Factorial(2*k))/(Factorial(
61                         nn2-k)*Factorial(k)*Factorial(k-1)*Factorial(n-
62                         k)*Factorial(2*k - n));
63
64 }
65 sum = 0.0;
66 ln2_on_t = log(2.0) / _TD;
67
68 for (double n = 1.0; n <= L; n++)
69 {
70     p = n * ln2_on_t;
71     sum = sum + v[n-1.0]*radialSelado.RadialSelado(p, _RD);
72 }
73     ilt = sum * ln2_on_t;
74     return ilt;
75 }
76
77 double CInvNumStehfest::StehfestRadialManutencao (double _TD,
78           double _RD)
79 {
80 nn2 = L/2.0;
81 nn21= nn2+1.0;
82
83 for (double n=1 ; n<=L; n++)
84 {
85     z=0.0;
86     for (int k = floor( ( n + 1.0 ) / 2.0 ); k <= min(n,nn2); k
87        ++)
88         z = z + ((pow(k, nn2))*Factorial(2*k))/(Factorial(
89                         nn2-k)*Factorial(k)*Factorial(k-1)*Factorial(n-
90                         k)*Factorial(2*k - n));
91
92     v.push_back(pow((-1.0),(n+nn2))*z);
93
94 }
95 sum = 0.0;
96 ln2_on_t = log(2.0) / _TD;
97
98 for (double n = 1.0; n <= L; n++)
```

```

96     {
97         p = n * ln2_on_t;
98         sum = sum + v[n-1.0]*radialmanutencao.RadialManutencao(p, _RD);
99     }
100    ilt = sum * ln2_on_t;
101    return ilt;
102 }
103
104 double CInvNumStehfest::StehfestLinearInfinito (double _TD)
105 {
106
107     ilt = linearinfinito.LinearInfinito(_TD);
108     return ilt;
109
110 }
111
112 double CInvNumStehfest::StehfestLinearSelado (double _xd)
113 {
114
115     ilt = linearselado.LinearSelado(_xd);
116     return ilt;
117 }
118
119 double CInvNumStehfest::StehfestLinearManutencao (double _xd)
120 {
121
122     ilt = linearmanutencao.LinearManutencao(_xd);
123     return ilt;
124 }
```

---

Apresenta-se na listagem 6.7 o arquivo com código da classe CReservatorioLinearInfinito.

Listing 6.7: Arquivo de cabeçalho da classe CReservatorioLinearInfinito.

```

1 #ifndef CRESERVATORIOLINEARINFINITO_H_
2 #define CRESERVATORIOLINEARINFINITO_H_
3
4 #include "CGeometriaReservatorio.h"
5
6 //////////////Criacao da classe CReservatorioLinearInfinito.h
7 class CReservatorioLinearInfinito : CGeometriaReservatorio
8 {
9     //Declaracao metodos publicos
10    public:
```

```

11
12         ///Construtor default
13         CReservatorioLinearInfinito() {};
14
15         /**
16             Calcula um ponto da funcao para a forma de
17                 reservatorio linear infinito
18             @parametro TD variavel do tempo adimensional
19             @retorna valor de funcao de calculada para
20                 reservatorio linear infinito
21
22         */
23         virtual double LinearInfinito(double _TD);
24
25     };
26
27 #endif

```

---

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

Listing 6.8: Arquivo de implementação da classe `CReservatorioLinearInfinito`.

```

1 #define _USE_MATH_DEFINES // define constantes matematicas (exemplo
2     Pi = 3.141516...)
3
4 #include <cmath> /////inclui biblioteca matematica padrao com
5     funcoes bessel para c++17 ou superior
6
7 double CReservatorioLinearInfinito::LinearInfinito(double _TD)
8 {
9
10    pd = sqrt(4.0*_TD/M_PI);
11
12    return pd;
13}

```

---

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

Listing 6.9: Arquivo de cabeçalho da classe `CReservatorioLinearManutencao`.

```

1 ifndef CRESERVATORIOLINEARMANUTENCAO_H_
2 define CRESERVATORIOLINEARMANUTENCAO_H_
3

```

```

4 #include "CGeometriaReservatorio.h"
5 ///Criacao da classe CReservatorioLinearManutencao.h
6 class CReservatorioLinearManutencao : CGeometriaReservatorio
7 {
8     ///Declaracao metodos publicos
9     public:
10         ///Construtor default
11         CReservatorioLinearManutencao() {};
12
13         /**
14             Calcula um ponto da funcao para a forma de
15                 reservatorio linear com manutencao
16             @parametro xd variavel do espaco adimensional
17             @retorna valor de funcao de calculada para
18                 reservatorio linear com manutencao
19
20         */
21         double virtual LinearManutencao(double _xd);
22
23 };
24
25 #endif

```

---

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

Listing 6.10: Arquivo de implementação da classe CReservatorioLinearManutencao.

```

1 #define _USE_MATH_DEFINES // define constantes matematicas (exemplo
2     Pi = 3.141516...)
3
4 #include "CReservatorioLinearManutencao.h"
5
6 double CReservatorioLinearManutencao::LinearManutencao(double _xd)
7 {
8
9     pd = _xd;
10
11    return pd;
12 }

```

---

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

Listing 6.11: Arquivo de cabeçalho da classe CReservatorioLinearSelado.

```

1 #ifndef CRESERVATORIOLINEARSELADO_H_
2 #define CRESERVATORIOLINEARSELADO_H_
3
4 #include "CGeometriaReservatorio.h"
5 ///Criacao da classe CReservatorioLinearSelado.h
6 class CReservatorioLinearSelado : CGeometriaReservatorio
7 {
8     ///Declaracao metodos publicos
9     public:
10
11         ///Construtor default
12         CReservatorioLinearSelado() {};
13
14     /**
15         Calcula um ponto da funcao para a forma de
16             reservatorio linear selado
17             @parametro xd variavel do espaco adimensional
18             @retorna valor de funcao de calculada para
19                 reservatorio linear selado
20
21         */
22         virtual double LinearSelado(double _xd);
23 };
24
25#endif

```

---

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

Listing 6.12: Arquivo de implementação da classe CReservatorioLinearSelado.

```

1 #define _USE_MATH_DEFINES // define constantes matematicas (exemplo
2     Pi = 3.141516...)
3
4 #include <cmath> ///inclui biblioteca matematica padrao com
5     funcoes bessel para c++17 ou superior
6
7 #include "CReservatorioLinearSelado.h"
8
9 double CReservatorioLinearSelado::LinearSelado(double _xd)
10 {
11     pd = _xd - ((_xd*_xd)/2.0);

```

```

10
11         return pd;
12 }
```

---

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

Listing 6.13: Arquivo de cabeçalho da classe CReservatorioRadialInfinito.

---

```

1 #ifndef CRESERVATORIORADIALINFINITO_H_
2 #define CRESERVATORIORADIALINFINITO_H_
3
4 #include "CGeometriaReservatorio.h"
5
6 // /Criacao da classe CReservatorioRadialInfinito.h
7
8 class CReservatorioRadialInfinito : CGeometriaReservatorio
9 {
10
11     ///Declaracao metodos publicos
12     public:
13
14     ///Construtor default
15     CReservatorioRadialInfinito(){};
16
17     /**
18      Calcula um ponto de Laplace para a forma de reservatorio radial
19      infinito
20      @parametro u variavel do tempo no campo de Laplace
21      @parametro Rd raio externo adimensional
22      @retorna valor de funcao de Laplace calculada no ponto (u,
23      Rd)
24
25     */
26     virtual double RadialInfinito(double _u, double _Rd);
27
28 };
29
30#endif
```

---

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

Listing 6.14: Arquivo de implementação da classe CReservatorioRadialInfinito.

---

```

1 #define _USE_MATH_DEFINES // define constantes matematicas (exemplo
2     Pi = 3.141516...)
3
4 #include <cmath> /////inclui biblioteca matematica padrao com
5     funcoes bessel para c++17 ou superior
6
7
8 //Forma da equação solucao do modelo de van everdinger para
9     reservaatorio infinito e radial
10
11 double CReservatorioRadialInfinito::RadialInfinito(double _u,
12             double _Rd)
13 {
14
15     pd = cyl_bessel_k(0, _Rd*sqrt(_u))/((pow(_u, (3.0/2.0)))*
16         cyl_bessel_k(1, sqrt(_u)));
17 }
```

Apresenta-se na listagem 6.15 o arquivo com código da classe CReservatorioRadialManutencao.

Listing 6.15: Arquivo de cabeçalho da classe CReservatorioLinearRadial.

```

1 ifndef CRESERVATORIORADIALMANUTENCAO_H_
2 define CRESERVATORIORADIALMANUTENCAO_H_
3
4 include "CGeometriaReservatorio.h"
5
6 //Criacao da classe CReservatorioRadialManutencao.h
7 class CReservatorioRadialManutencao : CGeometriaReservatorio
8 {
9     //Declaracao metodos publicos
10    public:
11
12        //Construtor default
13        CReservatorioRadialManutencao() {};
14
15        /**
16            Calcula um ponto de Laplace para a forma de
17            reservatorio radial com manutencao
```

```

17             @parametro u variavel do tempo no campo de
18                 Laplace
19             @parametro RD raio externo adimensional
20             @retorna valor de funcao de Laplace calculada no
21                 ponto (u, RD)
22             */
23         virtual double RadialManutencao(double _u, double
24             _RD);
25
26 };
27
28 #endif

```

---

Apresenta-se na listagem ?? o arquivo de implementação da classe CReservatorioRadialManutencao.

Listing 6.16: Arquivo de implementação da classe CReservatorioRadialManutencao.

```

1 #define _USE_MATH_DEFINES // define constantes matematicas (exemplo
2     Pi = 3.141516...)
3 #include <cmath> ///inclui biblioteca matematica padrao com
4     funcoes bessel para c++17 ou superior
5
6 using namespace std;
7
8 double CReservatorioRadialManutencao::RadialManutencao(double _u,
9     double _ReD)
10 {
11     pd = (((cyl_bessel_i(0, _ReD*sqrt(_u))*cyl_bessel_k(0,
12         sqrt(_u)))-(cyl_bessel_k(0, _ReD*sqrt(_u))*cyl_bessel_i
13         (0, sqrt(_u))))/((pow(_u, (3.0/2.0)))*((cyl_bessel_i(0,
14         _ReD*sqrt(_u))*cyl_bessel_k(1, sqrt(_u)))+(cyl_bessel_i
15         (1, sqrt(_u))*cyl_bessel_k(0, _ReD*sqrt(_u))))));

```

---

Apresenta-se na listagem 6.17 o arquivo com código da classe CReservatorioRadialSelado.

Listing 6.17: Arquivo de cabeçalho da classe CReservatorioRadialSelado.

```

1 #ifndef CRESERVATORIORADIALSELADO_H_
2 #define CRESERVATORIORADIALSELADO_H_
3
4 #include "CGeometriaReservatorio.h"
5 ///Criacao da classe CReservatorioRadialSelado.h
6 class CReservatorioRadialSelado : CGeometriaReservatorio
7 {
8     //Declaracao metodos publicos
9     public:
10
11         ///Construtor default
12         CReservatorioRadialSelado() {};
13
14         /**
15             Calcula um ponto de Laplace para a forma de
16                 reservatorio radial selado
17             @parametro u variavel do tempo no campo de
18                 Laplace
19             @parametro RD raio externo adimensional
20             @retorna valor de funcao de Laplace calculada no
21                 ponto (u, RD)
22         */
23         virtual double RadialSelado(double _u, double _RD);
24
25
26 };
27
28
29#endif

```

---

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

Listing 6.18: Arquivo de implementação da classe CReservatorioRadialSelado.

```

1 #define _USE_MATH_DEFINES // define constantes matematicas (exemplo
2     Pi = 3.141516...)
3
4 #include <cmath> ///inclui biblioteca matematica padrao com
5     funcoes bessel para c++17 ou superior
6
7 #include "CReservatorioRadialSelado.h"

```

```

5
6 using namespace std;
7
8 double CReservatorioRadialSelado::RadialSelado(double _u, double
9   _ReD)
10 {
11     pd = (((cyl_bessel_i(1, _ReD*sqrt(_u))*cyl_bessel_k(0,
12       sqrt(_u)))+(cyl_bessel_k(1, _ReD*sqrt(_u))*cyl_bessel_i
13       (0, sqrt(_u))))/((pow(_u, (3.0/2.0)))*((cyl_bessel_i(1,
14       _ReD*sqrt(_u))*cyl_bessel_k(1, sqrt(_u)))-(cyl_bessel_i
15       (1, sqrt(_u))*cyl_bessel_k(1, _ReD*sqrt(_u))))));
16
17     return pd;
18 }

```

---

Apresenta-se na listagem 6.19 o arquivo com código da classe CSimulador.

Listing 6.19: Arquivo de cabeçalho da classe CSimulador.

```

1 #ifndef CSimulador_H_
2 #define CSimulador_H_
3
4 #include <vector>
5 #include <iostream>
6 #include <string>
7
8 #include "CIInvNumStehfest.h"
9 #include "CGnuplot.h"
10
11 //Criacao da classe CSimulador.h
12 class CSimulador
13 {
14     ///Declaracao metodos protegidos
15     protected:
16
17         CGeometriaReservatorio forma;
18         CIInvNumStehfest Stehfest;
19         Gnuplot plot, plot2, plot3, plot4, plot5;
20
21         std::vector <double> TD, ReD;
22
23     ///Declaracao metodos publicos
24     public:

```

```

25
26         ///Construtor default
27         CSimulador() {};
28
29         ///Funções auxiliares para plotar com retorno vazio
30         void Solver();
31         void Plot(std::vector <double> _WD, std::vector <
32             double> _TD, std::string name, bool _setY);
33         void Plot2(std::vector <double> _WD, std::vector <
34             double> _TD, std::string name, bool _setY);
35         void Plot3(std::vector <double> _WD, std::vector <
36             double> _TD, std::string name, std::string type,
37             std::string type2);
38         void Plot4(std::vector <double> _WD, std::vector <
39             double> _TD, std::string name, std::string type,
40             std::string type2);
41         void Plot5(std::vector <double> _WD, std::vector <
42             double> _TD, std::string name, std::string type,
43             std::string type2);
44
45         ///Destrutor default
46         ~CSimulador() {};
47
48     };
49
50 #endif

```

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

Listing 6.20: Arquivo de implementação da classe CSimulador.

```

1 #include "CSimulador.h"
2
3 using namespace std;
4
5 void CSimulador::Plot(vector <double> _WD, vector <double> _TD, std
6   ::string name, bool _setY)
7 {
8     Gnuplot::Terminal("qt");
9
10    if (_setY)
11        plot.set_yrange(0, 10);

```

```
12
13     plot.Grid();
14     plot.set_xlabel("tD");
15     plot.set_ylabel("pD");
16     plot.Title("Pressao Adimensional x Tempo Adimensional");

17     plot.set_xlogscale();
18     plot.Legend("inside left top box");
19     plot.ShowOnScreen();
20     plot.plot_xy(_TD, _WD, name);
21     plot.savetops(name);
22     cout << "Aperte ENTER para continuar" << endl;
23     cin.get();

24
25
26 }
27

28 void CSimulador::Plot2(vector <double> _WD, vector <double> _TD,
29   std::string name, bool _setY)
30 {
31     Gnuplot::Terminal("qt");

32
33     if (_setY)
34         plot2.set_yrange(0, 5);

35
36     plot2.Grid();
37     plot2.set_xlabel("tD");
38     plot2.set_ylabel("pD");
39     plot2.Title("Pressao Adimensional x Tempo Adimensional");

40     plot2.set_xlogscale();
41     plot2.Legend("inside left top box");
42     plot2.ShowOnScreen();
43     plot2.plot_xy(_TD, _WD, name);
44     plot2.savetops(name);
45     cout << "Aperte ENTER para continuar" << endl;
46     cin.get();

47
48
49 }
50
```

```
51 void CSimulador::Plot3(vector <double> _WD, vector <double> _TD,
52                         std::string name, std::string type, std::string type2)
53 {
54     Gnuplot::Terminal("qt");
55
56     plot3.Grid();
57     plot3.set_xlabel(type2);
58     plot3.set_ylabel("pD");
59     plot3.Title("Pressao Adimensional x "+type+" Adimensional")
60             ;
61     plot3.Legend("inside left top box");
62     plot3.ShowOnScreen();
63     plot3.plot_xy(_TD, _WD, name);
64     plot3.savetops(name);
65     cout << "Aperte ENTER para continuar" << endl;
66     cin.get();
67
68 }
69
70 void CSimulador::Plot4(vector <double> _WD, vector <double> _TD,
71                         std::string name, std::string type, std::string type2)
72 {
73     Gnuplot::Terminal("qt");
74
75     plot4.Grid();
76     plot4.set_xrange(0, 2);
77     plot4.set_yrange(0, 1);
78     plot4.set_xlabel(type2);
79     plot4.set_ylabel("pD");
80     plot4.Title("Pressao Adimensional x "+type+" Adimensional")
81             ;
82     plot4.Legend("inside left top box");
83     plot4.ShowOnScreen();
84     plot4.plot_xy(_TD, _WD, name);
85     plot4.savetops(name);
86     cout << "Aperte ENTER para continuar" << endl;
87     cin.get();
88 }
```

```
89 }
90
91 void CSimulador::Plot5(vector <double> _WD, vector <double> _TD,
92   std::string name, std::string type, std::string type2)
93 {
94     Gnuplot::Terminal("qt");
95
96     plot5.Grid();
97     //plot5.set_xrange(0, 2);
98     //plot5.set_yrange(0, 1);
99     plot5.set_xlabel(type2);
100    plot5.set_ylabel("pD");
101    plot5.Title("Pressão Adimensional "+type+" Adimensional")
102    ;
103    plot5.Legend("inside_left_top_box");
104    plot5.ShowOnScreen();
105    plot5.plot_xy(_TD, _WD, name);
106    plot5.savetops(name);
107    cout << "Aperte ENTER para continuar" << endl;
108    cin.get();
109
110 }
111
112 void CSimulador::Solver()
113 {
114
115     vector <double> radialinfinito, radialselado2,
116         radialselado3, radialselado4, radialselado5,
117         radialselado6, radialselado7, radialselado8,
118         radialselado9, radialselado10, radialManutencao2,
119         radialManutencao3, radialManutencao4, radialManutencao5,
120         radialManutencao6, radialManutencao7, radialManutencao8,
121         radialManutencao9, radialManutencao10;
122     vector <double> linearinfinito, linearmanutencao,
123         linearselado;
124
125
126     for (double i=0.01;i<=1000;i+=0.05)
127         TD.push_back(i);
128
129
130
131 }
```



```
147                     radialselado3.push_back(Stehfest.
148                                     StehfestRadialSelado(TD[k], 3));
149
150         cout << "
151             #####"
152             << endl;
153         cout << "#"
154             "#####"
155             << endl;
156         Plot(radialselado3, TD, "RadialSelado\u2014Red\u20143", 0);
157         for (double k=0; k < TD.size(); k++)
158             radialselado4.push_back(Stehfest.
159             StehfestRadialSelado(TD[k], 4));
160
161         cout << "
162             #####"
163             << endl;
164         cout << "#"
165             "#####"
166             << endl;
167         Plot(radialselado4, TD, "RadialSelado\u2014Red\u20144", 0);
168         for (double k=0; k < TD.size(); k++)
169             radialselado5.push_back(Stehfest.
170             StehfestRadialSelado(TD[k], 5));
```



```
194     cout << "
195         #####"
196         << endl;
197     cout << "#uuuuuuPlotando\u00d7Radial\u00d7Selado\u00d7para\u00d7ReD\u00d7=7uuuuuuuuuu
198         uu\u00d7" << endl;
199     cout << "#uuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuu
200         uu\u00d7" << endl;
201     cout << "#uuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuu
202         uu\u00d7" << endl;
203     for (double k=0; k < TD.size(); k++)
204         radialselado8.push_back(Stehfest.
205             StehfestRadialSelado(TD[k], 8));
206
207     cout << "
208         #####"
209         << endl;
210     cout << "#uuuuuuPlotando\u00d7Radial\u00d7Selado\u00d7para\u00d7ReD\u00d7=8uuuuuuuuuu
211         uu\u00d7" << endl;
212     cout << "#uuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuu
213         uu\u00d7" << endl;
214     cout << "
215         #####"
216         << endl;
217     cout << "#uuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuu
```

```
        ##" << endl;
217    cout << "#Criando novo terminal do gnuplot"
        ##" << endl;
218    cout << "#"
        ##" << endl;
219    cout << "
        #####"
        << endl;
220
221        Plot2(radialinfinito, TD, "RadialInfinito", 1);
222
223        for (double k=0; k < TD.size(); k++)
224            radialManutencao2.push_back(Stehfest.
                StehfestRadialManutencao(TD[k], 2));
225
226    cout << "
        #####"
        << endl;
227    cout << "#"
        ##" << endl;
228    cout << "#Plotando Radial Manutencao para ReD=2"
        ##" << endl;
229    cout << "#"
        ##" << endl;
230    cout << "
        #####"
        << endl;
231
232        Plot2(radialManutencao2, TD, "RadialManutencao -"
            ReD=2", 0);
233
234        for (double k=0; k < TD.size(); k++)
235            radialManutencao3.push_back(Stehfest.
                StehfestRadialManutencao(TD[k], 3));
236
237    cout << "
        #####"
        << endl;
238    cout << "#"
        ##" << endl;
239    cout << "#Plotando Radial Manutencao para ReD=3"
        ##" << endl;
```









```

334         cout << "
335             #####"
336             << endl;
337             cout << "# Plotando Linear com Manutenção de Pressão"
338             << endl;
339             cout << "
340                 Plot5(linearmanutencao, TD, "Linear com Manutenção de"
341                         "Pressão", "Espaço", "xD");
342             cout << "
343                 #####"
344             << endl;
345             cout << "# Todos arquivos de saída estão salvos na pasta ./"
346             "Src#" << endl;
347             cout << "# Pressione ENTER para destruir arquivos"
348             "temporários#" << endl;
349             cout << "
350                 #####"
351             << endl;
352
353 }

```

Apresenta-se na listagem 6.21 o programa que usa a classe `main`.

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

---

```
#include "CSimulador.h"
```

```
2
3
4 int main(void){
5
6     CSimulador executa;
7
8     executa.Solver();
9
10    return 0;
11}
```

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

**Nota:**

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

# Capítulo 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: Regime Radial

Primeiramente, ao abrir o programa, será plotado o gráfico para o regime radial infinito (pressão *vs.* tempo), com os valores de pressão e tempo adimensionais. Veja Figura 7.1.

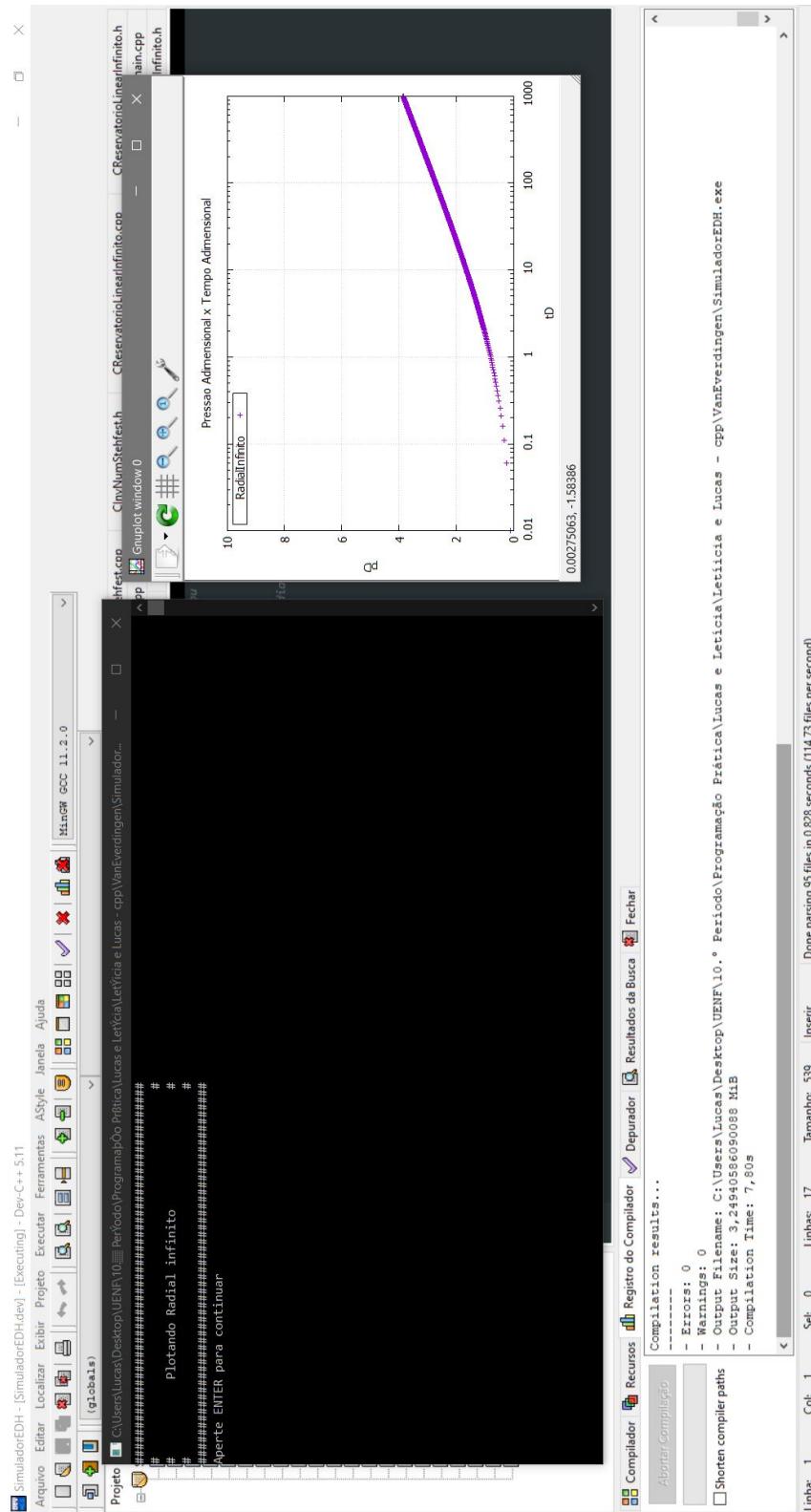


Figura 7.1: Tela inicial do programa mostrando o regime radial infinito

Em seguida, será solicitado ao usuário que clique *enter*, de modo que o regime radial será diagnosticado no reservatório selado (pressão *vs.* tempo), para valores distintos de pressão e tempo. Veja Figura 7.2.

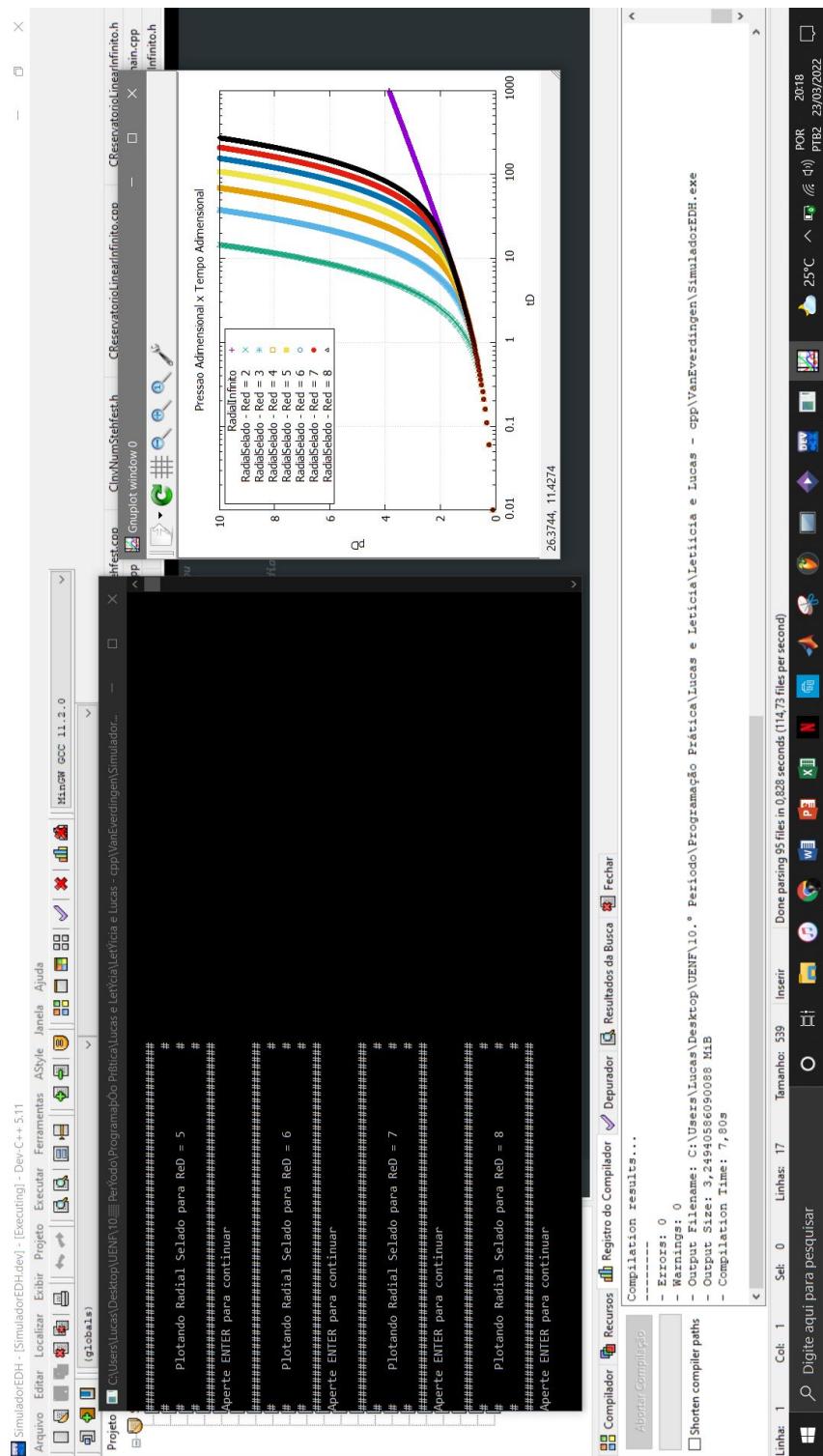


Figura 7.2: Tela do programa mostrando o regime radial selado para valores de pressão e tempo distintos

Será solicitado ao usuário que clique *enter* novamente, de modo que o programa abrirá o terminal com o reservatório no regime radial infinito (pressão *vs.* tempo), com os valores de pressão e tempo adimensionais. Veja Figura 7.3.

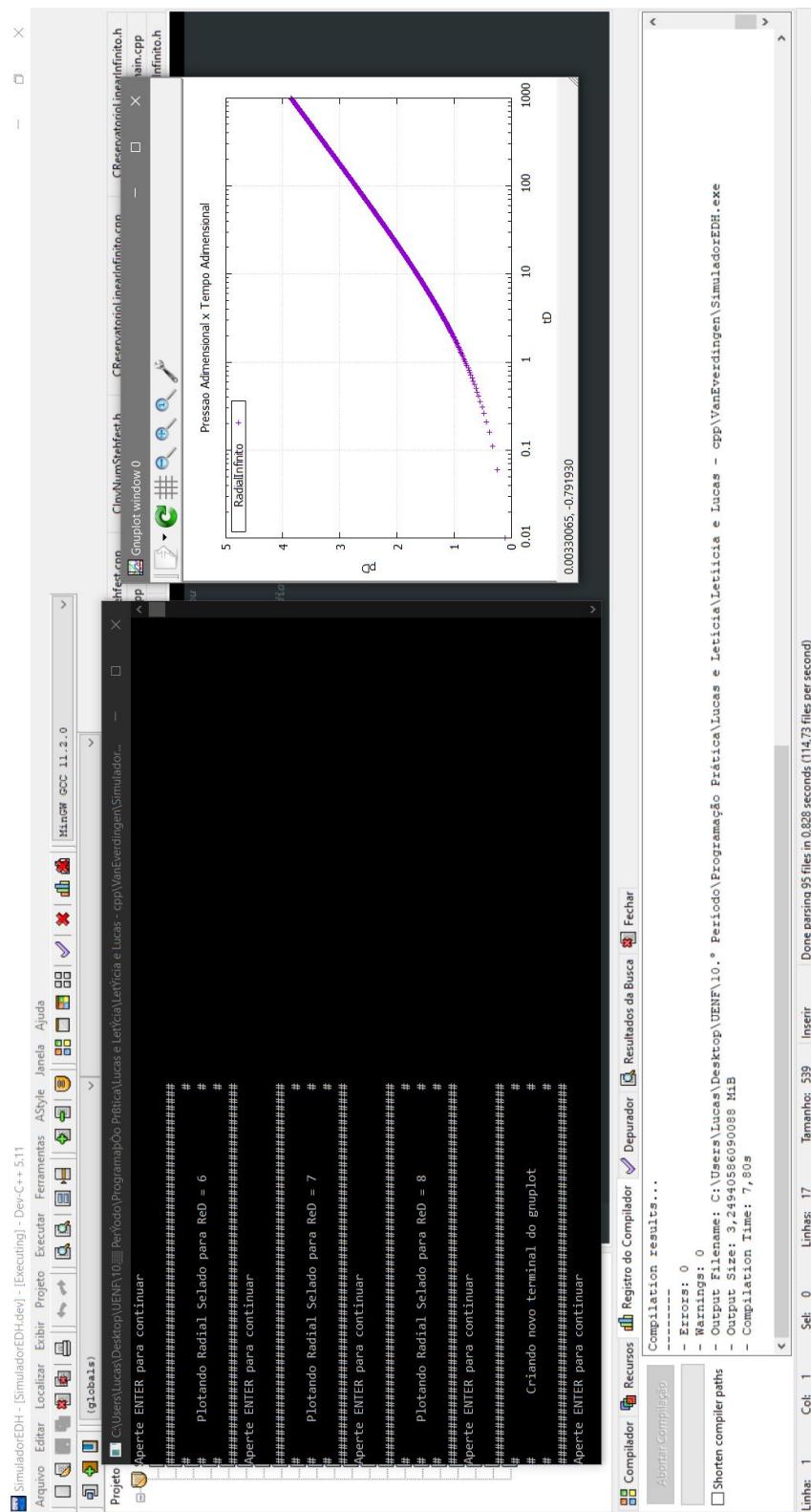


Figura 7.3: Tela do programa mostrando o terminal do Gnuplot para o regime radial infinito

E então, será solicitado ao usuário que novamente clique *enter*, de modo que o regime radial será diagnosticado no reservatório com manutenção de pressão (*pressão vs. tempo*), para valores distintos de pressão e tempo. Veja Figura 7.4.

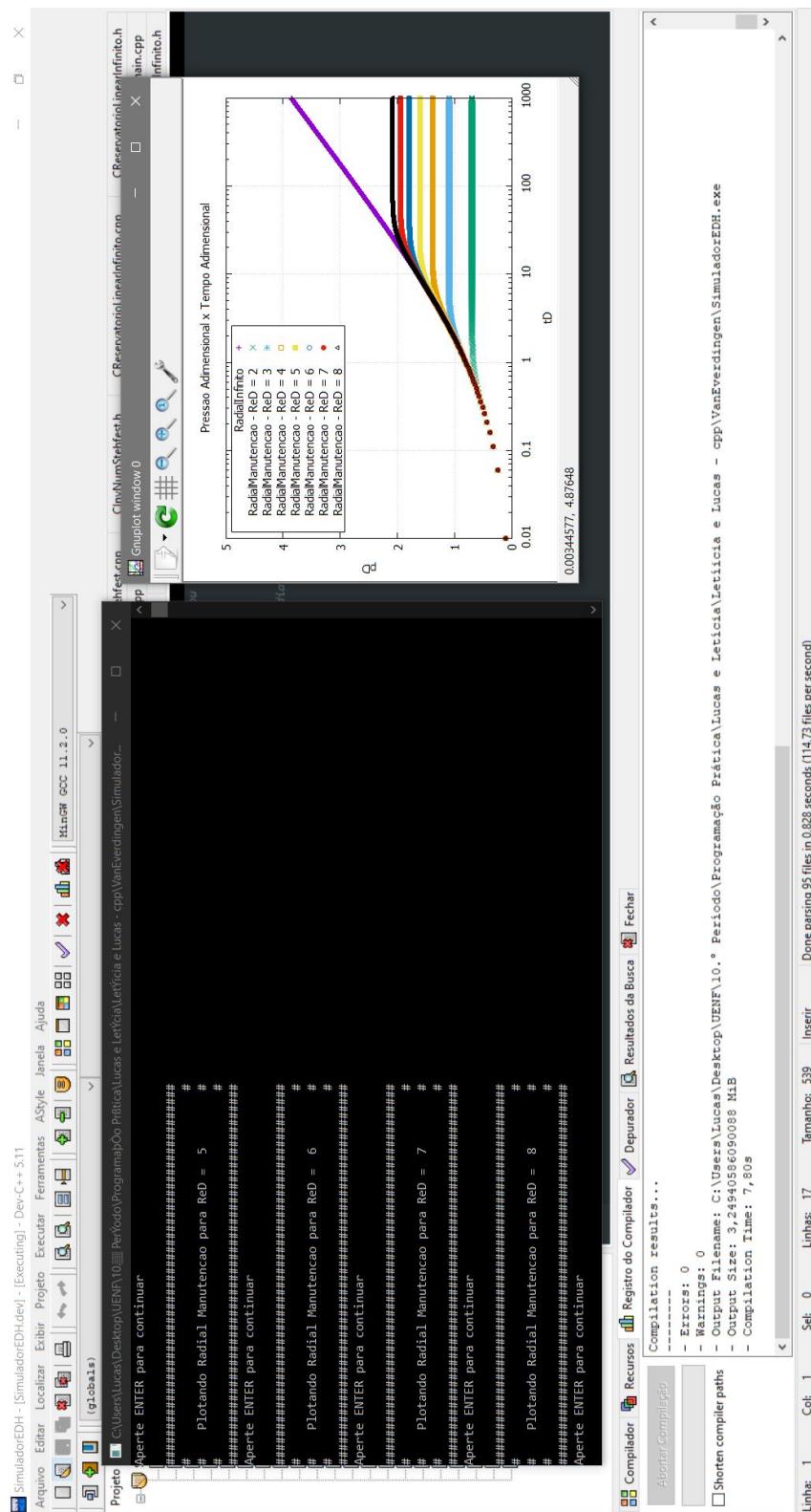


Figura 7.4: Tela do programa mostrando o regime radial com manutenção de pressão

## 7.2 Teste 2: Regime Linear

Será solicitado ao usuário que clique *enter*, de modo que o programa irá plotar o gráfico para o regime linear infinito (pressão *vs.* tempo), com valores de pressão e tempo

adimensionais. Veja Figura 7.5.

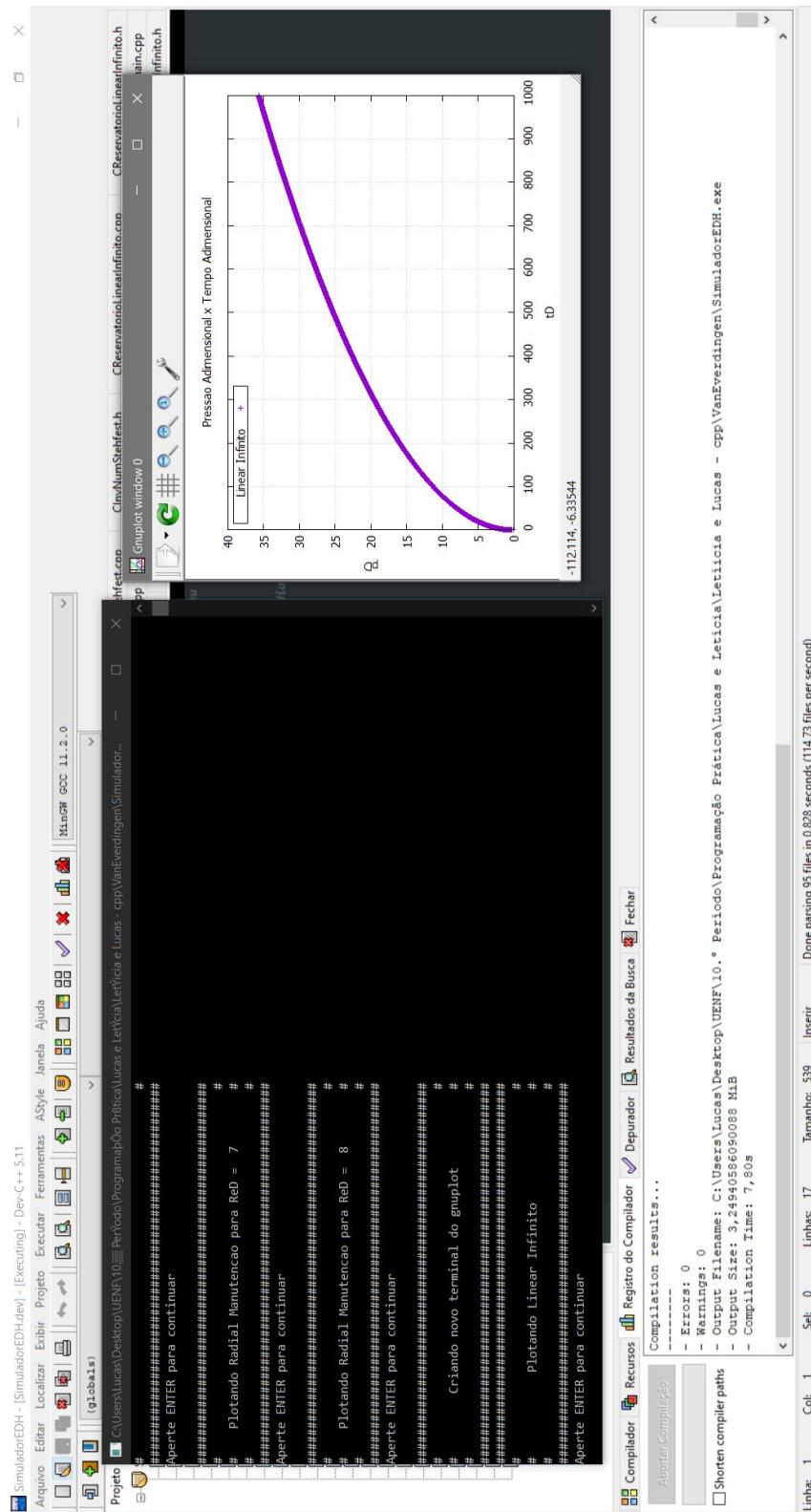


Figura 7.5: Tela do programa mostrando o regime linear

Em seguida, será solicitado ao usuário que clique *enter*, de modo que o regime linear será diagnosticado no reservatório selado (*pressão vs. posição*), para valores distintos de pressão e posição. Veja Figura 7.6

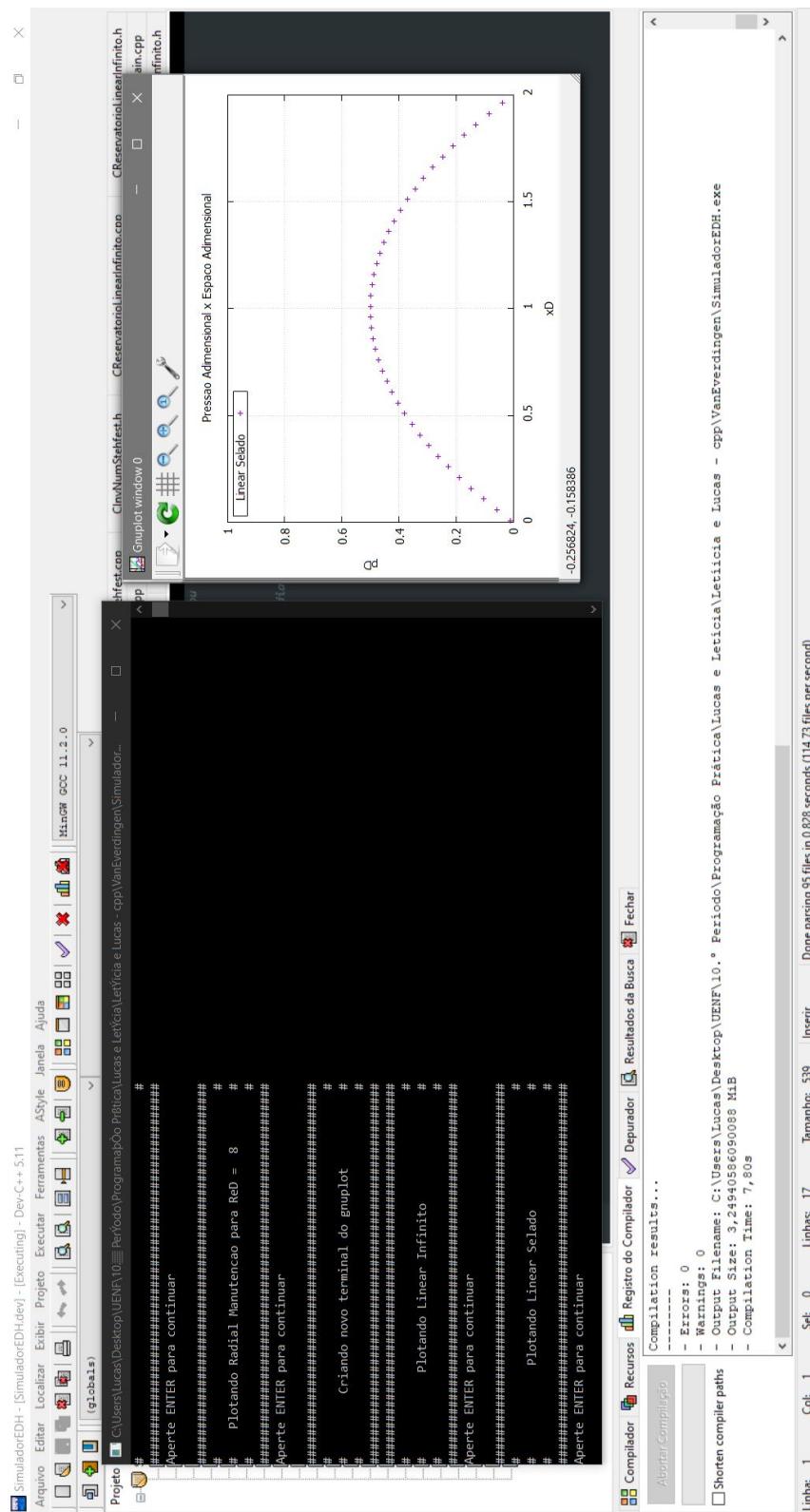


Figura 7.6: Tela do programa mostrando o regime linear selado para valores de pressão e posição distintos

A seguir, o programa mostra o reservatório linear com manutenção de pressão (pressão vs. posição), com valores distintos de pressão e posição. Veja Figura 7.7.

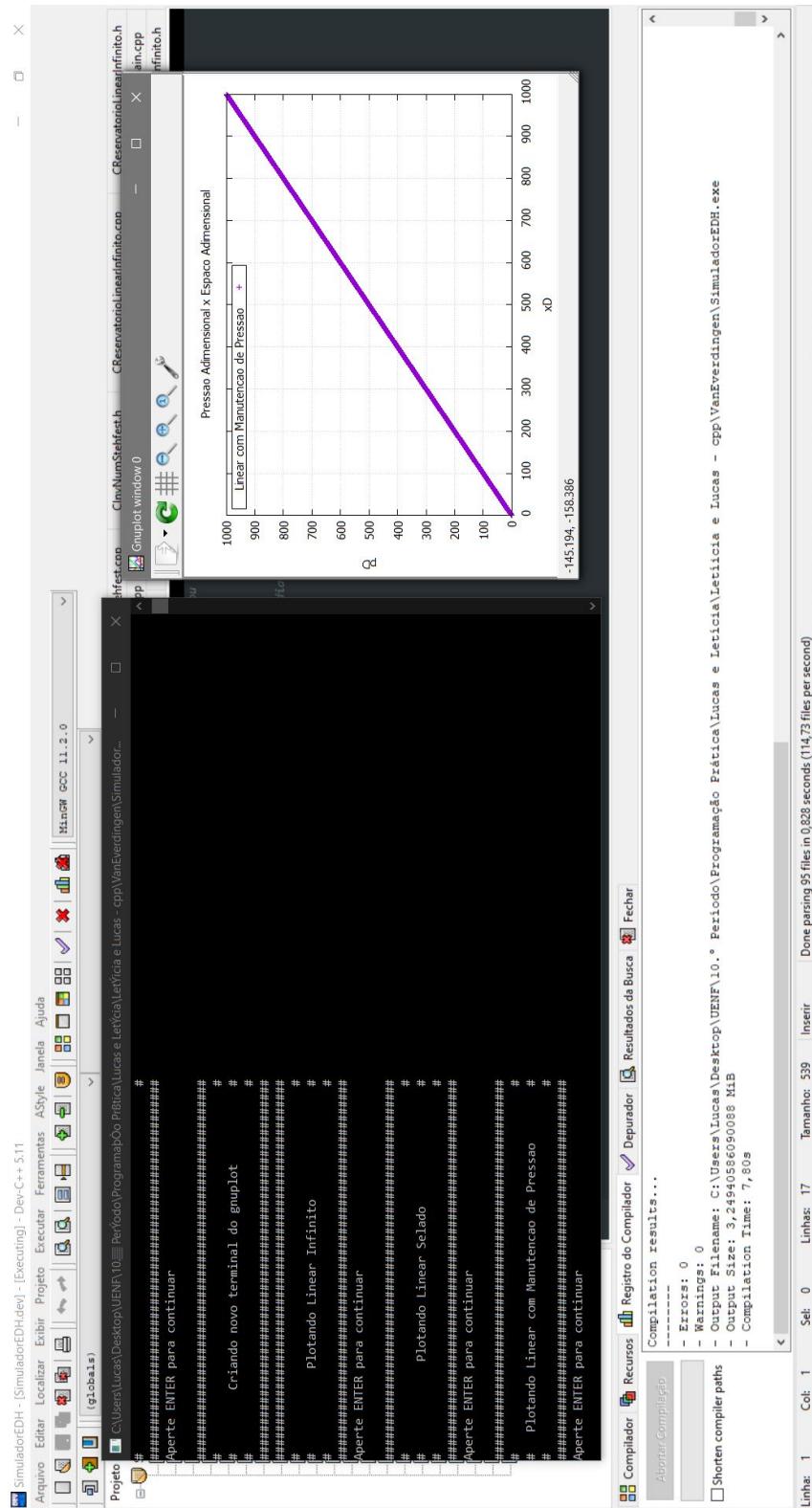


Figura 7.7: Tela do programa mostrando o regime linear com manutenção de pressão

A tela final indica que os resultados gráficos foram salvos em disco. Veja Figura 7.8.

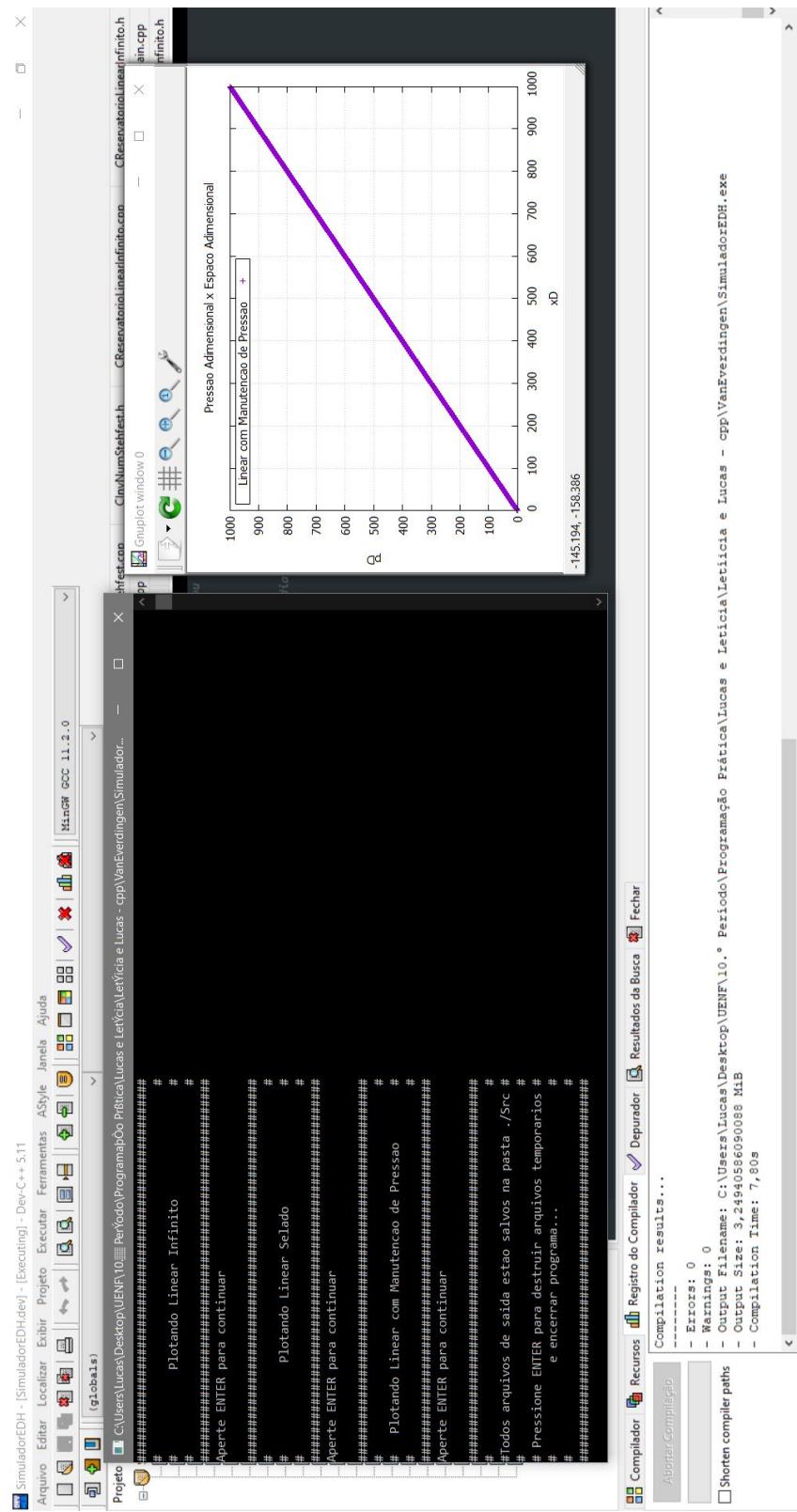


Figura 7.8: Tela do programa mostrando que os gráficos plotados foram salvos

### 7.3 Teste 3: Autenticidade do programa

Para fins comparativos, segue abaixo alguns gráficos da apostila “Análise de Testes de Poços - PETROBRAS”.

Primeiramente, veja a Figura 7.9 para analisar o comportamento do reservatório radial com manutenção de pressão.

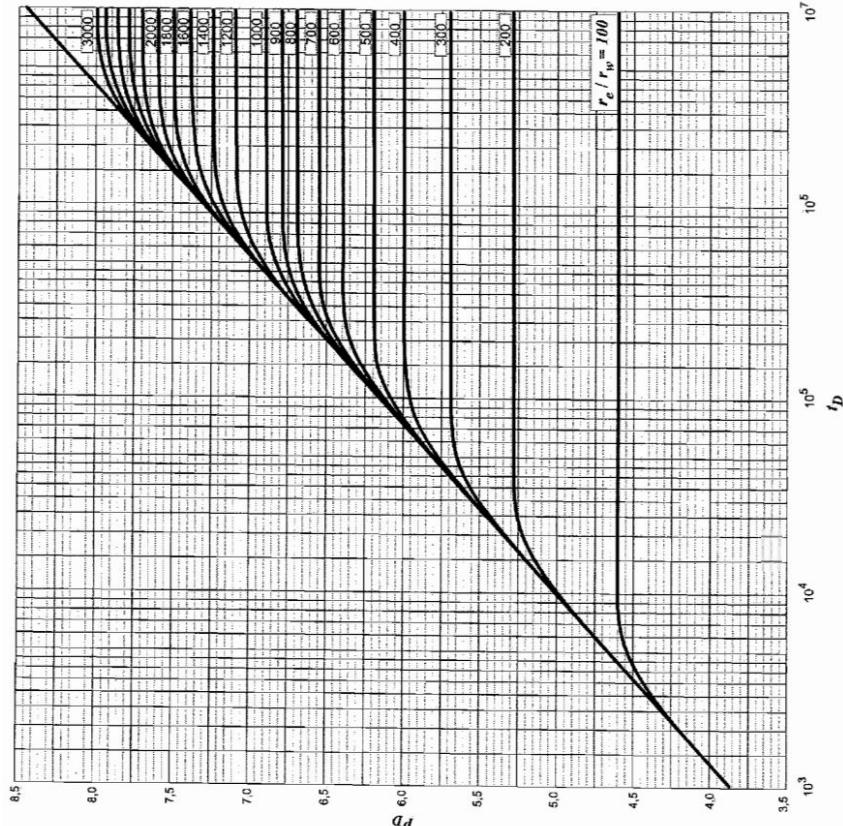


Figura 7.9: Reservatório Radial com Manutenção de Pressão

Já a Figura 7.10 representa os resultados gráficos obtidos pelo programa.

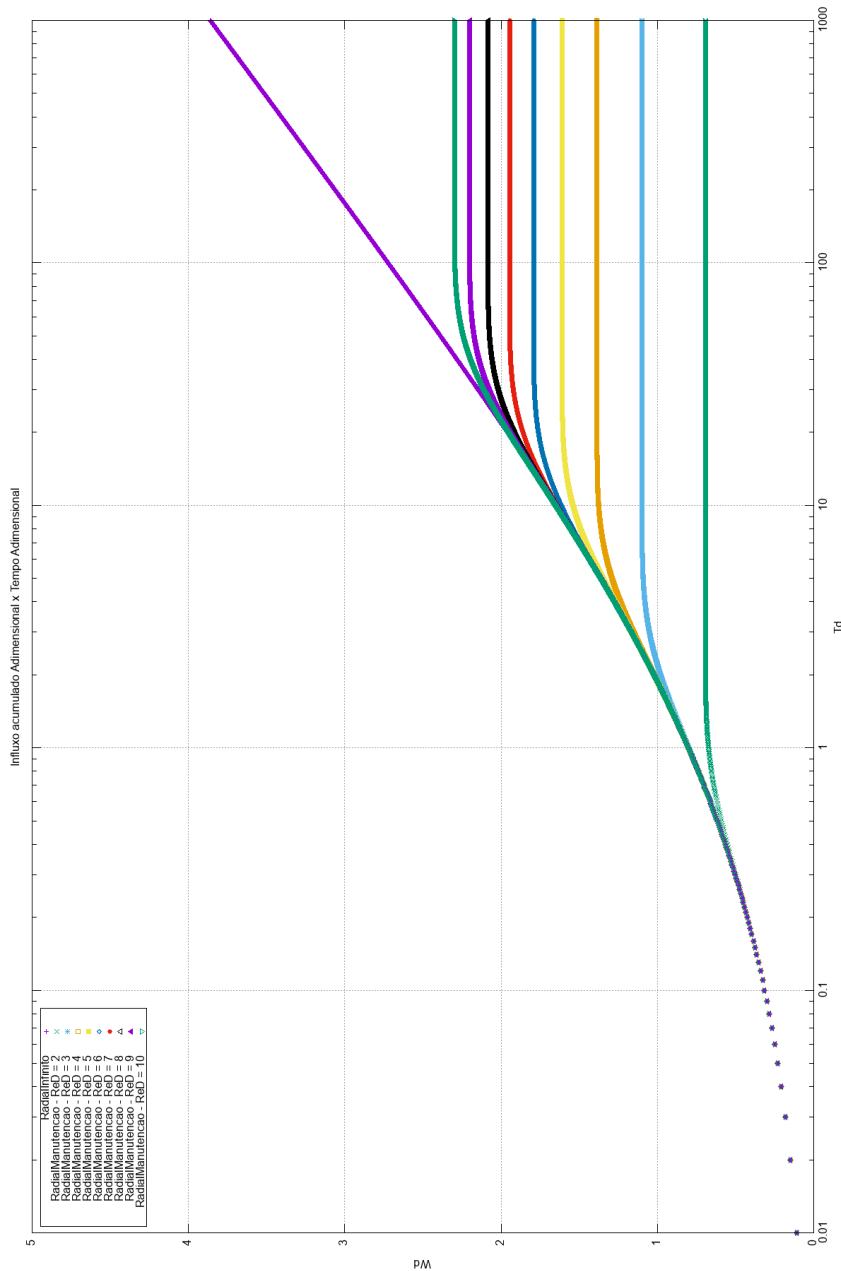


Figura 7.10: Reservatório Radial com Manutenção de Pressão

Primeiramente, veja a Figura 7.11 para analisar o comportamento do reservatório radial selado.

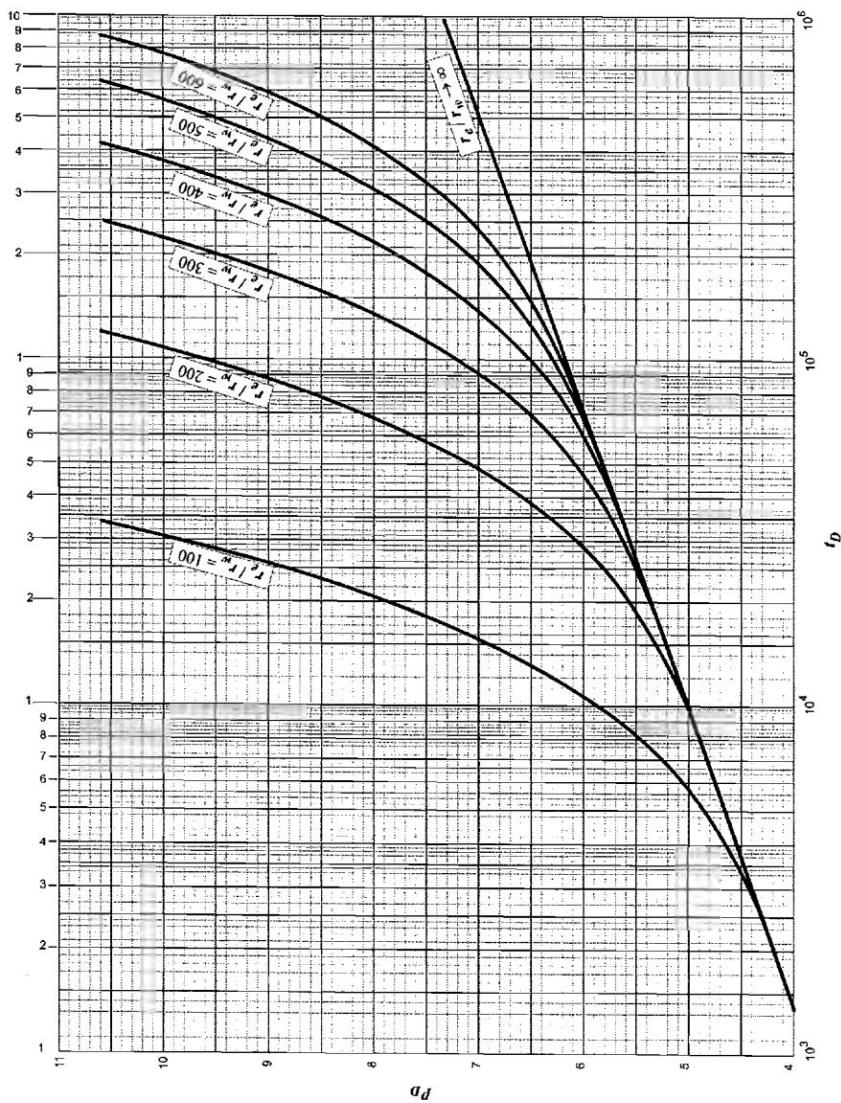


Figura 7.11: Reservatório Radial Selado

Já a Figura 7.12 representa os resultados gráficos obtidos pelo programa.

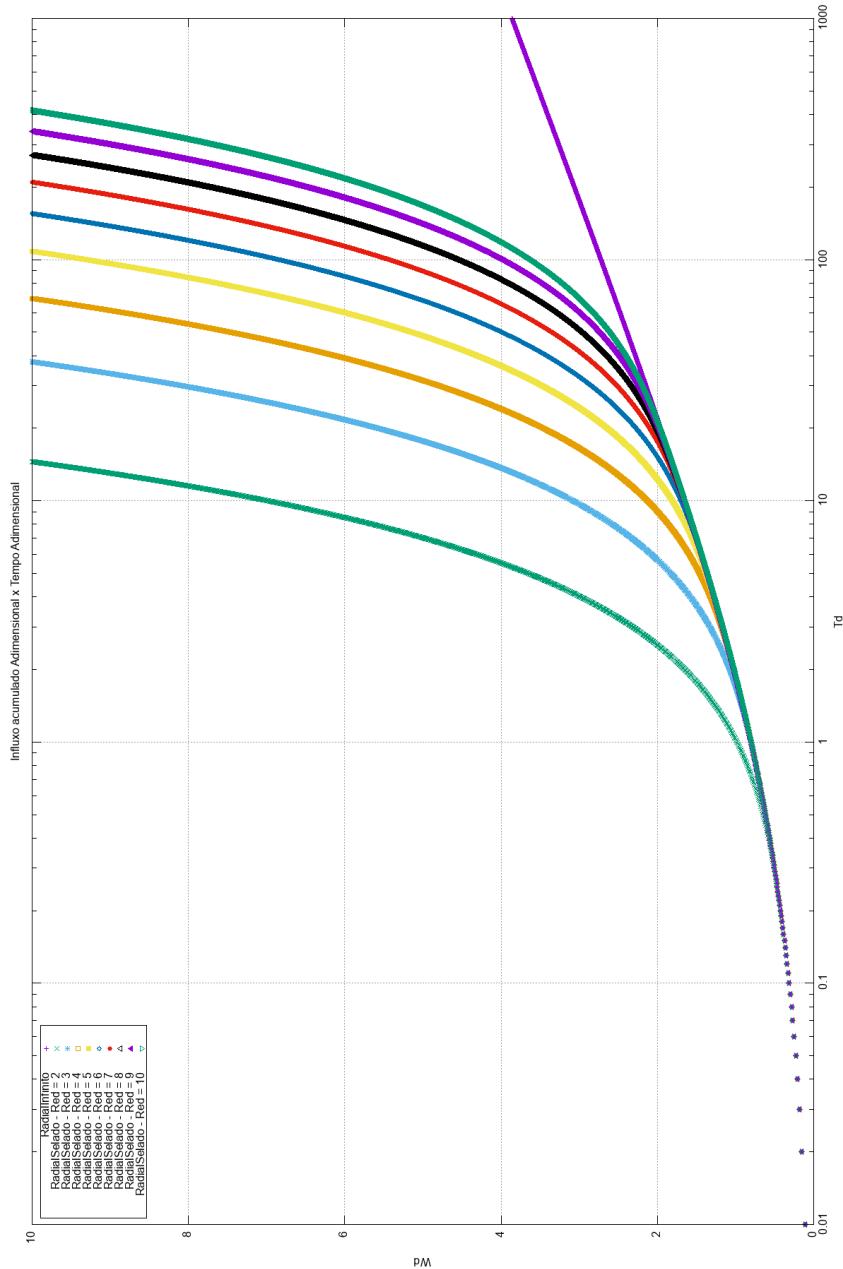


Figura 7.12: Reservatório Radial com Manutenção de Pressão

# Capítulo 8

## Documentação

Todo projeto de engenharia precisa ser bem documentado. Neste sentido, apresenta-se neste capítulo a documentação de uso do “Simulador de Soluções Analíticas da Equação da Difusividade Hidráulica Para Fluxo Linear e Radial”. Esta documentação tem o formato de uma apostila que explica o passo a passo de como usar o software.

### 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 instalar o software

Para instalar o software execute o seguinte passo a passo:

- Em Linux: Abra o terminal, vá para o diretório onde está o simulador, faça a compilação e depois, o execute.
- Em Windows: Faça o download de um compilador, como por exemplo o Dev C++ disponível em <https://dev-c.softonic.com.br/>. Compile o simulador e execute-o.

#### 8.1.2 Como rodar o software

Após compilado, o programa irá gerar os resultados gráficos dos regimes radiais e lineares para a Equação da Difusividade Hidráulica. Para o regime radial, o usuário poderá comparar os resultados para o reservatório selado e com manutenção, com o infinito. Já para o regime linear, os resultados são gerados em janelas separadas. Todos os resultados gráficos são armazenados na pasta *Src*.

## 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`.

### 8.2.2 Como gerar a documentação usando doxygen

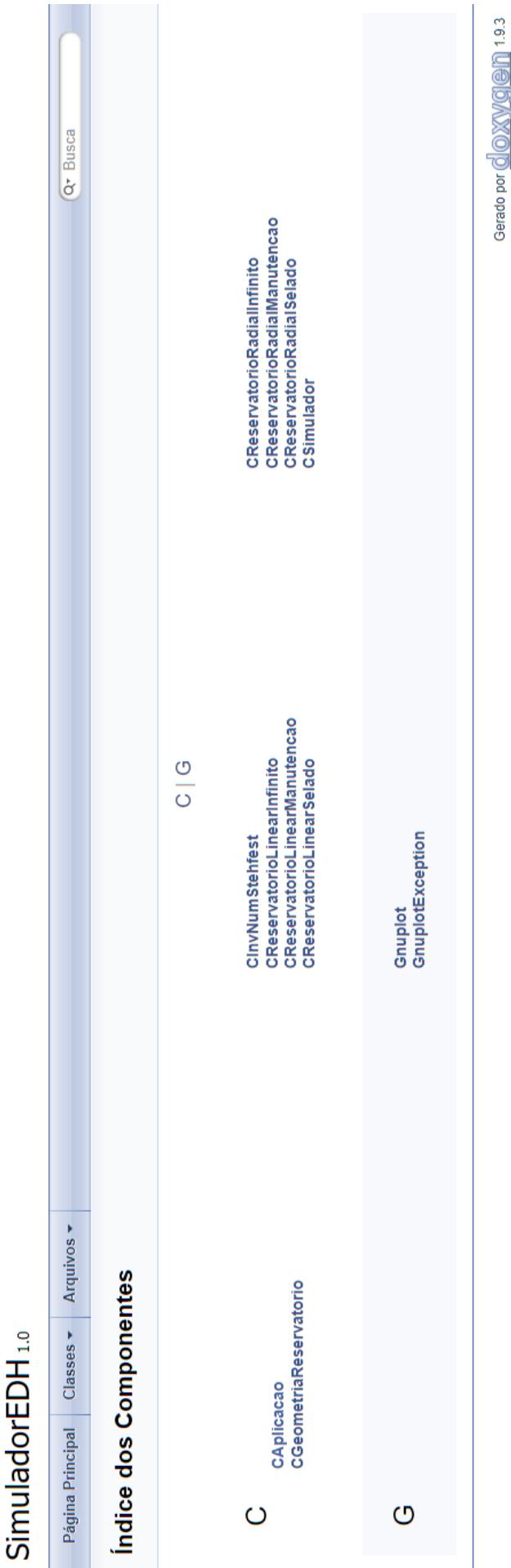
A documentação do código do software deve ser feita usando o padrão JAVADOC, conforme apresentada no Capítulo - Documentação, do livro texto da disciplina. Depois de documentar o código, use o software `doxygen` para gerar a documentação do desenvolvedor no formato html. O software `doxygen` lê os arquivos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

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

## SimuladorEDH 1.0

Página Principal	Classes ▾	Arquivos ▾																																				
<b>Lista de Classes</b>																																						
Aqui estão as classes, estruturas, uniões e interfaces e suas respectivas descrições:																																						
<table border="1"><tbody><tr><td><a href="#">C CAplicacao</a></td><td colspan="2">Criacao da classe <a href="#">CGeometriaReservatorio.h</a></td></tr><tr><td><a href="#">C CGeometriaReservatorio</a></td><td colspan="2">Criacao da classe <a href="#">CGeometriaReservatorio.h</a></td></tr><tr><td><a href="#">C CInvNumStehfest</a></td><td colspan="2">Criacao da classe <a href="#">CInvNumStehfest.h</a></td></tr><tr><td><a href="#">C CReservatorioLinearInfinito</a></td><td colspan="2">Criacao da classe <a href="#">CReservatorioLinearInfinito.h</a></td></tr><tr><td><a href="#">C CReservatorioLinearManutencao</a></td><td colspan="2">Criacao da classe <a href="#">CReservatorioLinearManutencao.h</a></td></tr><tr><td><a href="#">C CReservatorioLinearSelado</a></td><td colspan="2">Criacao da classe <a href="#">CReservatorioLinearSelado.h</a></td></tr><tr><td><a href="#">C CReservatorioRadialInfinito</a></td><td colspan="2">Criacao da classe <a href="#">CReservatorioRadialInfinito.h</a></td></tr><tr><td><a href="#">C CReservatorioRadialManutencao</a></td><td colspan="2">Criacao da classe <a href="#">CReservatorioRadialManutencao.h</a></td></tr><tr><td><a href="#">C CReservatorioRadialSelado</a></td><td colspan="2">Criacao da classe <a href="#">CReservatorioRadialSelado.h</a></td></tr><tr><td><a href="#">C CSimulador</a></td><td colspan="2">Criacao da classe <a href="#">CSimulador.h</a></td></tr><tr><td><a href="#">C Gnuplot</a></td><td colspan="2">Classe de interface para acesso ao programa gnuplot</td></tr><tr><td><a href="#">C GnuplotException</a></td><td colspan="2">Erros em tempo de execucao</td></tr></tbody></table>			<a href="#">C CAplicacao</a>	Criacao da classe <a href="#">CGeometriaReservatorio.h</a>		<a href="#">C CGeometriaReservatorio</a>	Criacao da classe <a href="#">CGeometriaReservatorio.h</a>		<a href="#">C CInvNumStehfest</a>	Criacao da classe <a href="#">CInvNumStehfest.h</a>		<a href="#">C CReservatorioLinearInfinito</a>	Criacao da classe <a href="#">CReservatorioLinearInfinito.h</a>		<a href="#">C CReservatorioLinearManutencao</a>	Criacao da classe <a href="#">CReservatorioLinearManutencao.h</a>		<a href="#">C CReservatorioLinearSelado</a>	Criacao da classe <a href="#">CReservatorioLinearSelado.h</a>		<a href="#">C CReservatorioRadialInfinito</a>	Criacao da classe <a href="#">CReservatorioRadialInfinito.h</a>		<a href="#">C CReservatorioRadialManutencao</a>	Criacao da classe <a href="#">CReservatorioRadialManutencao.h</a>		<a href="#">C CReservatorioRadialSelado</a>	Criacao da classe <a href="#">CReservatorioRadialSelado.h</a>		<a href="#">C CSimulador</a>	Criacao da classe <a href="#">CSimulador.h</a>		<a href="#">C Gnuplot</a>	Classe de interface para acesso ao programa gnuplot		<a href="#">C GnuplotException</a>	Erros em tempo de execucao	
<a href="#">C CAplicacao</a>	Criacao da classe <a href="#">CGeometriaReservatorio.h</a>																																					
<a href="#">C CGeometriaReservatorio</a>	Criacao da classe <a href="#">CGeometriaReservatorio.h</a>																																					
<a href="#">C CInvNumStehfest</a>	Criacao da classe <a href="#">CInvNumStehfest.h</a>																																					
<a href="#">C CReservatorioLinearInfinito</a>	Criacao da classe <a href="#">CReservatorioLinearInfinito.h</a>																																					
<a href="#">C CReservatorioLinearManutencao</a>	Criacao da classe <a href="#">CReservatorioLinearManutencao.h</a>																																					
<a href="#">C CReservatorioLinearSelado</a>	Criacao da classe <a href="#">CReservatorioLinearSelado.h</a>																																					
<a href="#">C CReservatorioRadialInfinito</a>	Criacao da classe <a href="#">CReservatorioRadialInfinito.h</a>																																					
<a href="#">C CReservatorioRadialManutencao</a>	Criacao da classe <a href="#">CReservatorioRadialManutencao.h</a>																																					
<a href="#">C CReservatorioRadialSelado</a>	Criacao da classe <a href="#">CReservatorioRadialSelado.h</a>																																					
<a href="#">C CSimulador</a>	Criacao da classe <a href="#">CSimulador.h</a>																																					
<a href="#">C Gnuplot</a>	Classe de interface para acesso ao programa gnuplot																																					
<a href="#">C GnuplotException</a>	Erros em tempo de execucao																																					

Figura 8.1: Lista de Classes Doxygen



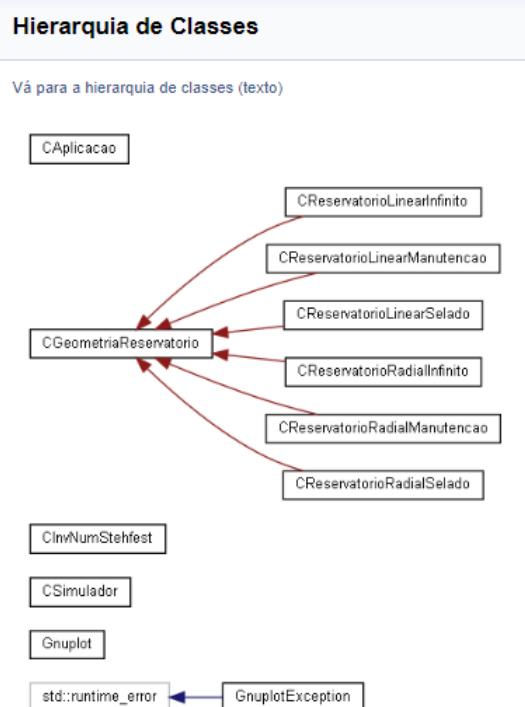


Figura 8.3: Hierarquia das Classes Doxygen



# Referências Bibliográficas

- [Ahmed, 1989] Ahmed, T. (1989). *Hydrocarbon phase behavior*. 9
- [Lake, 1989] Lake, L. W. (1989). Enhanced oil recovery. 8
- [Rosa et al., 2006] Rosa, A. J., de Souza Carvalho, R., and Xavier, J. A. D. (2006). *Engenharia de reservatórios de petróleo*. Interciênciac. 9
- [Stehfest, 1970] Stehfest, H. (1970). Algorithm 368: Numerical inversion of laplace transforms [d5]. *Communications of the ACM*, 13(1):47–49. 10