

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

1	Introdução	1
1.1	Escopo do problema	1
1.2	Objetivos	2
2	Especificação	3
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
3	Elaboração	7
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
4	AOO – Análise Orientada a Objeto	13
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

5 Projeto	19
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
6 Implementação	25
6.1 Código fonte	25
7 Teste	120
7.1 Teste 1: Descrição	120
7.2 Teste 2: Regime Radial	121
7.3 Teste 2: Regime Linear	124
8 Documentação	129
8.1 Documentação do usuário	129
8.1.1 Como instalar o software	129
8.1.2 Como rodar o software	129
8.2 Documentação para desenvolvedor	130
8.2.1 Dependências	130
8.2.2 Como gerar a documentação usando doxygen	130

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;
 - Gerar gráficos de $p_D(r_D)$ 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

Nome	Software que resolve a Equação da Difusividade Hidráulica em termos adimensionais
Componentes principais	Desenvolvimento das soluções da equação da difusividade hidráulica
Missão	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).

2.2.1 Requisitos funcionais

Apresenta-se a seguir os requisitos funcionais.

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

2.2.2 Requisitos não funcionais

RNF-01	A resolução da equação será feita utilizando dados adimensionais para mostrar as soluções gerais que servem para qualquer sistemas de unidades.
RNF-02	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

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

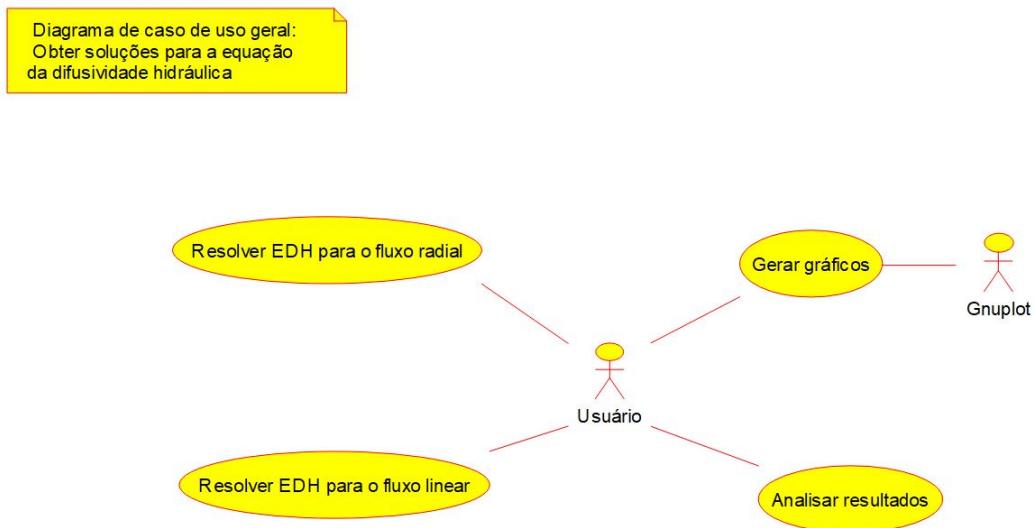


Figura 2.1: Caso de uso geral: Obter as soluções da Equação da Difusividade Hidráulica (EDH)

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.

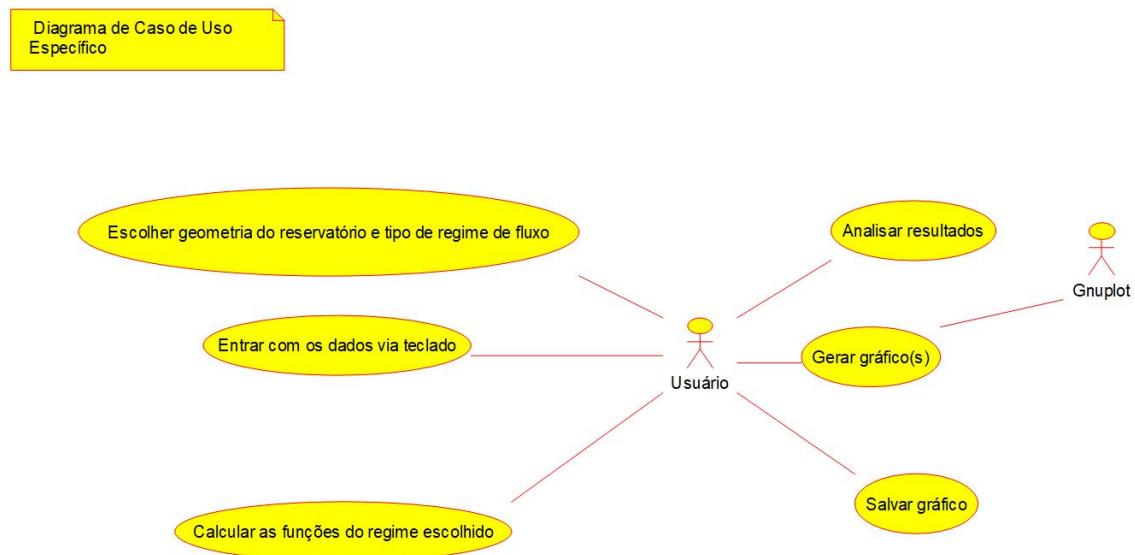


Figura 2.2: Diagrama de caso de uso específico: Cenário de teste

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^a ordem e 2^a 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 ΔA , a variação de massa no elemento é igual à diferença entre as quantidades que entram e saem do volume de controle como mostrado abaixo:

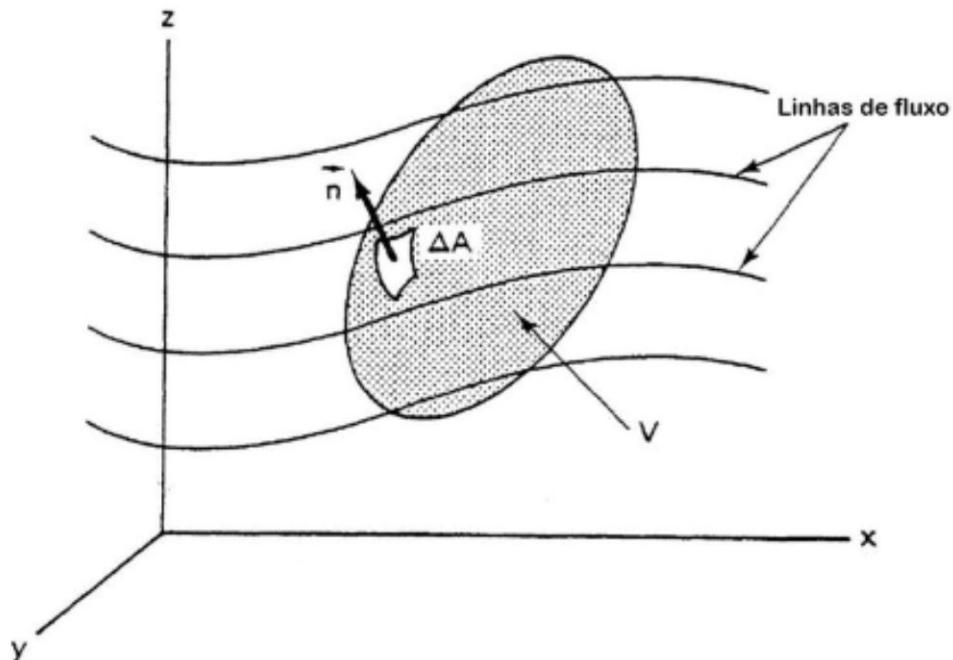


Figura 3.1: Volume de controle arbitrário no domínio de fluxo (Adaptado: LAKE *et al.*, 1989)

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 *et al.*, 2006).

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 (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.

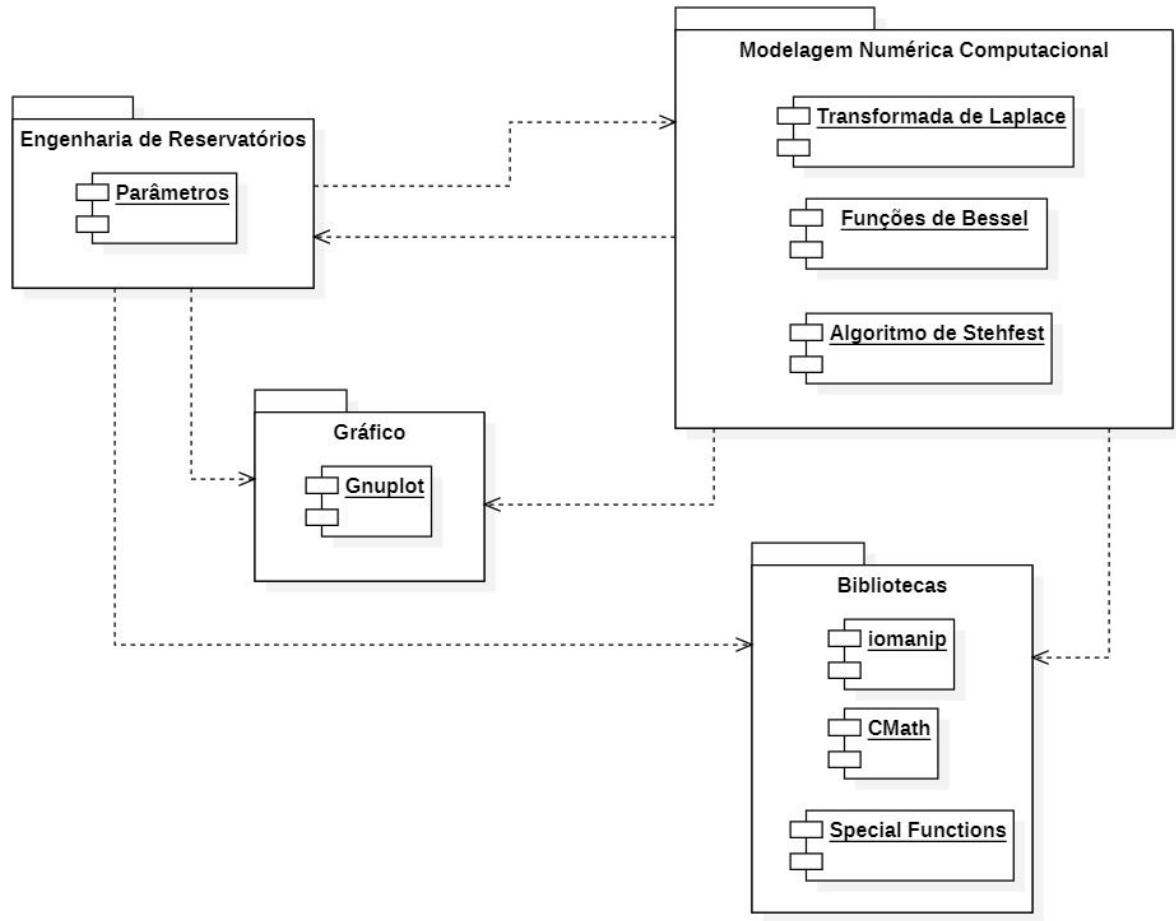


Figura 3.2: Diagrama de Pacotes

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, 2017). 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.

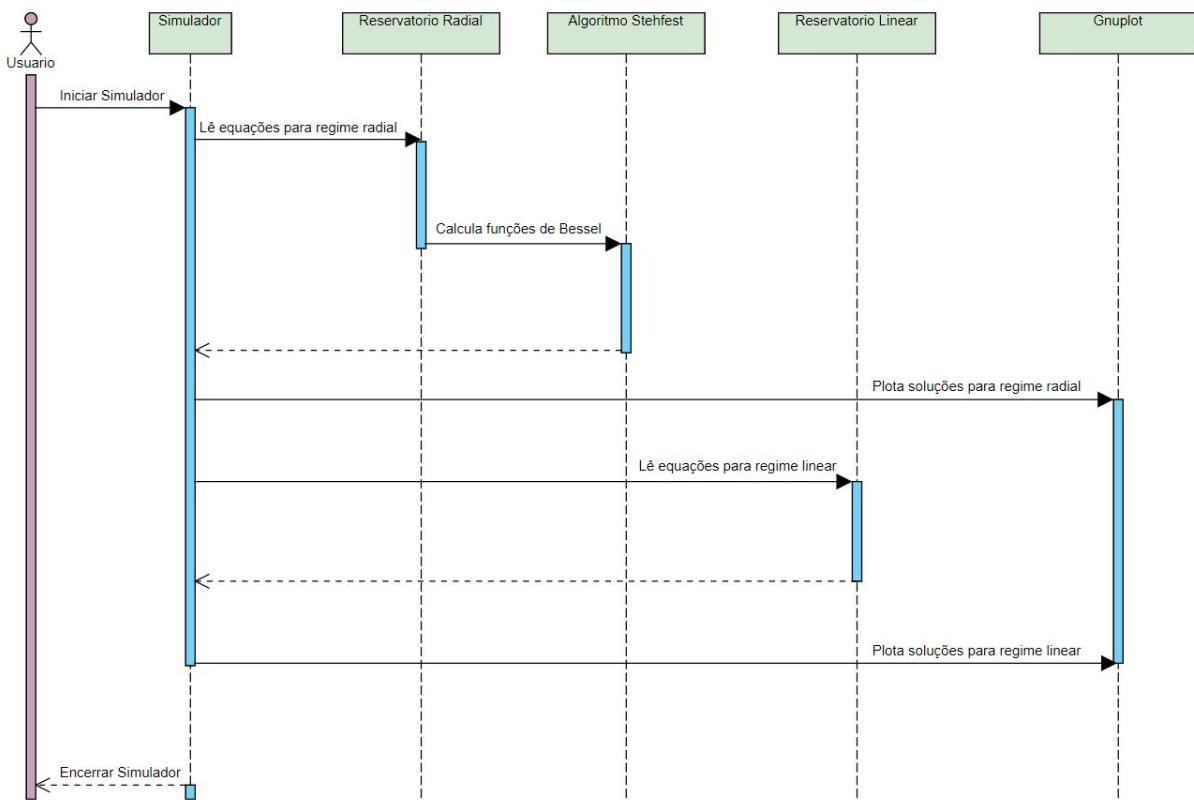


Figura 4.2: Diagrama de sequência

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.

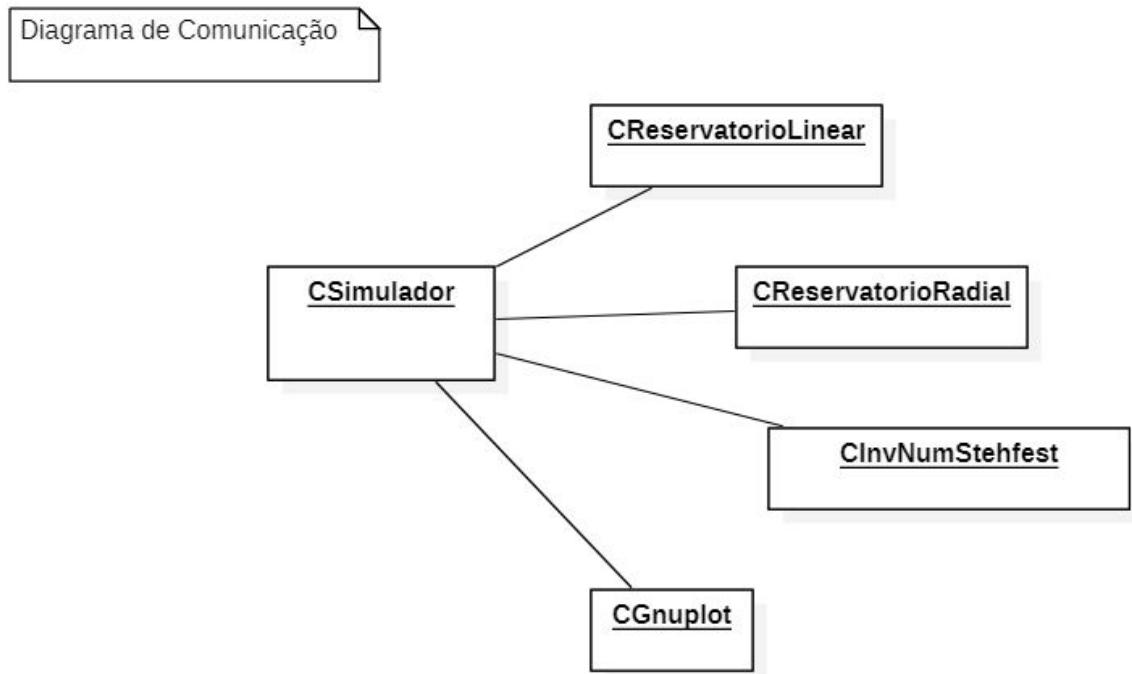


Figura 4.3: Diagrama de comunicação

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.

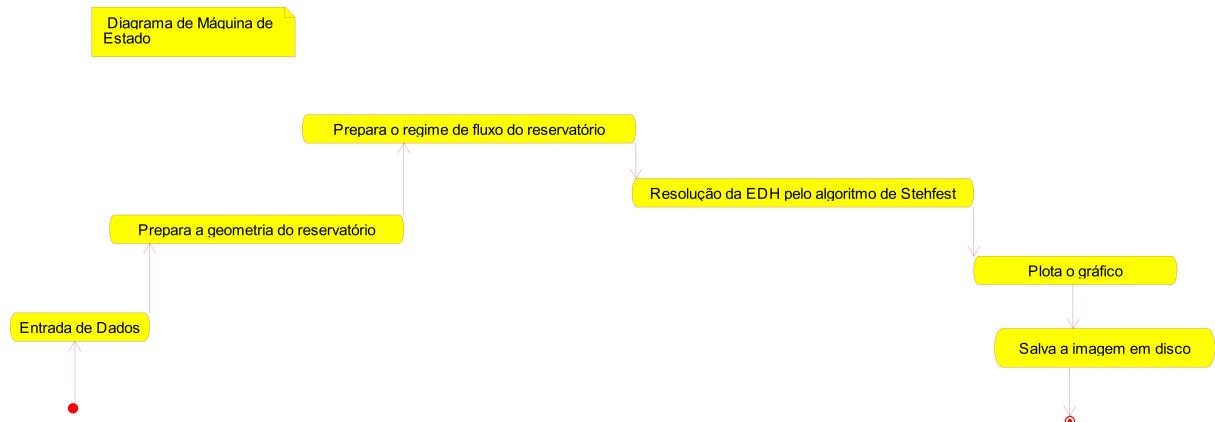


Figura 4.4: Diagrama de máquina de estado

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

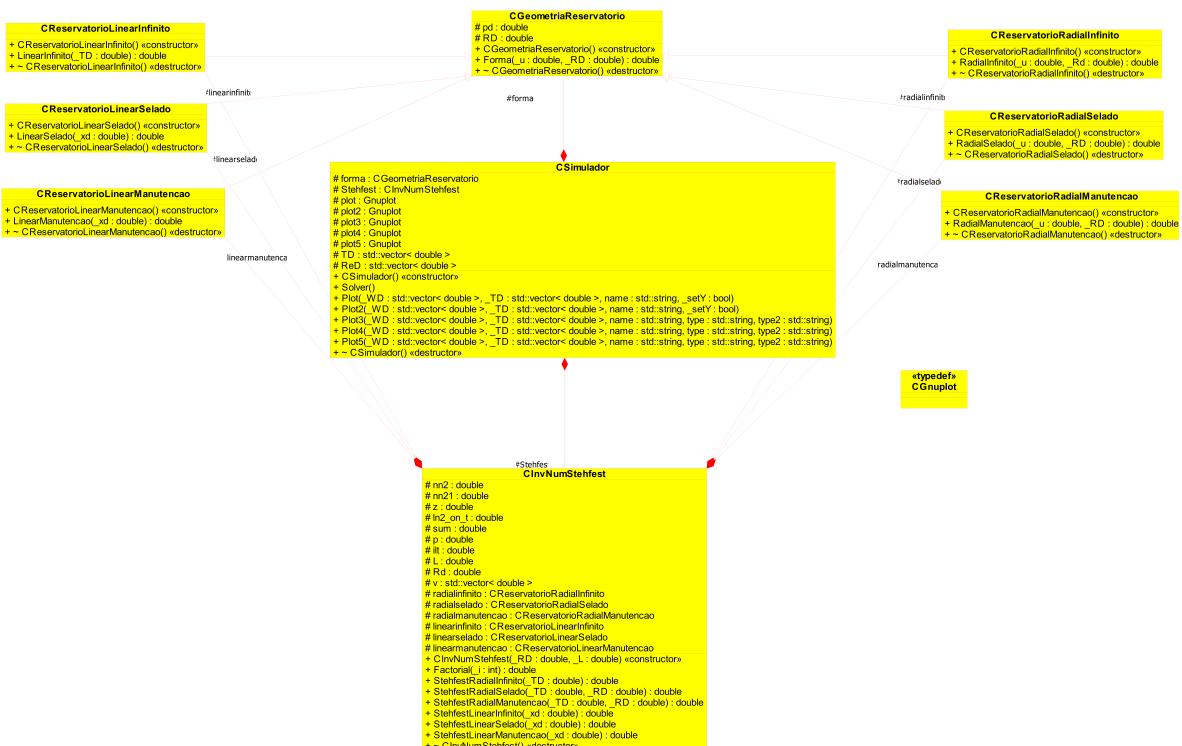


Figura 4.1: Diagrama de classes

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.

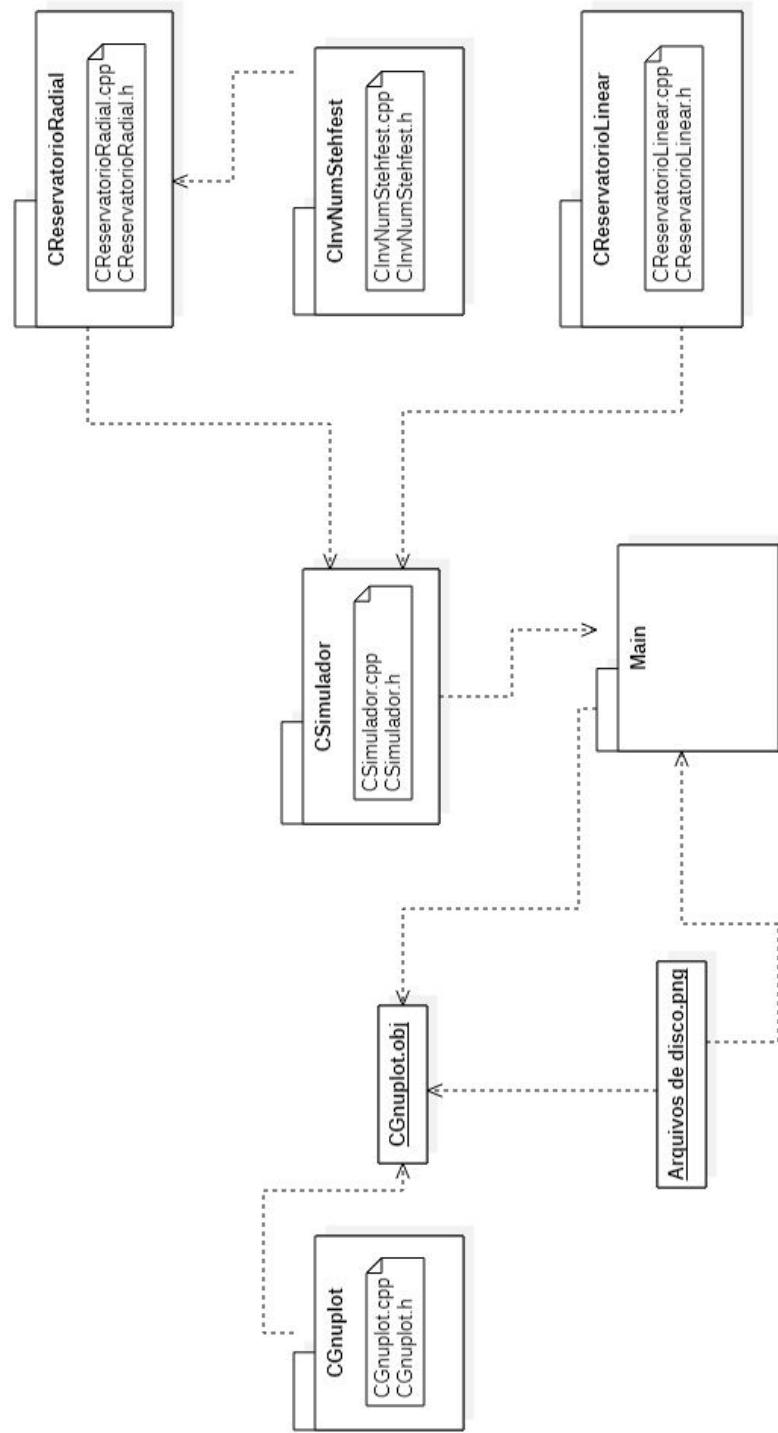


Figura 5.1: Diagrama de componentes do software Soluções Analíticas Adimensionais da Equação da Difusividade Hidráulica para Fluxo Linear e Radial

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.

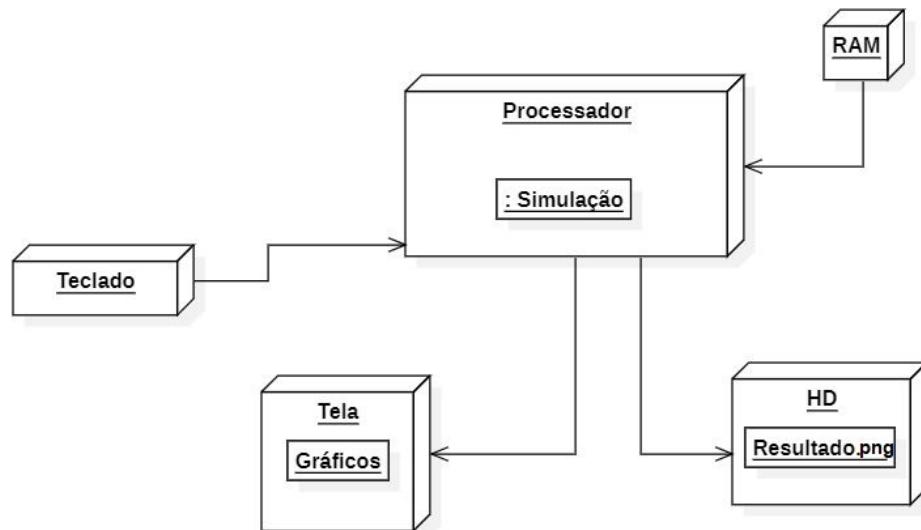


Figura 5.2: Diagrama de implantação.

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^AT_EX. Maiores detalhes de como a saída pode ser gerada estão disponíveis nos links abaixo.

- http://en.wikibooks.org/wiki/LaTeX/Source_Code_Listings.
- <http://mirrors.ctan.org/macros/latex/contrib/listings/listings.pdf>.

6.1 Código fonte

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

Apresenta-se na listagem 6.1 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 class CGeometriaReservatorio
5 {
6     protected:
7 }
```

```

8         double pd, RD; //Influxo Adimensional //Raio
9             externo admensional
10
11     public:
12
13         CGeometriaReservatorio() {};
14
15         virtual double Forma(double _u, double _RD);
16
17         ~CGeometriaReservatorio() {};
18
19 }
20 #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 //////////////////////////////////////////////////////////////////
3 //          Classe de Interface em C++ para o programa gnuplot
4 //          .
4 //          //////////////////////////////////////////////////////////////////
5 // Esta interface usa pipes e nao ira funcionar em sistemas que nao
6 // suportam
5 // o padrao POSIX pipe.
6 // O mesmo foi testado em sistemas Windows (MinGW e Visual C++) e
7 // Linux (GCC/G++)
7 // Este programa foi originalmente escrito por:
8 // Historico de versoes:
9 // 0. Interface para linguagem C

```

```
10 //      por N. Devillard (27/01/03)
11 // 1. Interface para C++: tradução direta da versão em C
12 //      por Rajarshi Guha (07/03/03)
13 // 2. Correções para compatibilidade com Win32
14 //      por V. Chyzhdzenka (20/05/03)
15 // 3. Novos métodos membros, correções para compatibilidade com
16 //      Win32 e Linux
17 //      por M. Burgis (10/03/08)
18 // 4. Tradução para Português, documentação - javadoc/doxygen,
19 //      e modificações na interface (adição de interface alternativa)
20 //      por Bueno.A.D. (30/07/08)
21 // Tarefas:
22 // (v1)
23 // Documentar toda classe
24 // Adicionar novos métodos, criando atributos adicionais se
25 // necessário.
26 // Adotar padrão C++, isto é, usar sobrecarga nas chamadas.
27 // (v2)
28 // Criar classe herdeira CGnuplot, que inclui somente a nova
29 // interface.
30 //
31 ///////////////////////////////////////////////////////////////////
32 // Requisitos:
33 // - O programa gnuplot deve estar instalado (veja http://www.gnuplot.info/download.html)
34 // - No Windows: setar a Path do Gnuplot (i.e. C:/program files/
35 //      gnuplot/bin)
36 //          ou setar a path usando: Gnuplot::set_GNUPlotPath(
37 //              const std::string &path);
38 //          Gnuplot::set_GNUPlotPath("C:/program files/gnuplot/
39 //              bin");
40 // - Para um melhor uso, consulte o manual do gnuplot,
41 //      no GNU/Linux digite: man gnuplot ou info gnuplot.
```

```
//////////  
  
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*/  
57class GnuplotException : public std::runtime_error  
58{  
59public:  
60    /// Construtor  
61    GnuplotException (const std::string & msg):std::runtime_error (msg) {}  
62};  
63  
64/**  
65@brief Classe de interface para acesso ao programa gnuplot.  
66@class Gnuplot  
67@file gnuplot_i.hpp  
68*/  
69class Gnuplot  
70{  
71private:  
72    //  
    -----  
    Atributos  
73    FILE * gnucmd;           ///  
                                Ponteiro para stream que escreve no  
                                pipe.  
74    bool valid;             ///  
                                Flag que indica se a sessao do gnuplot  
                                esta valida.  
75    bool two_dim;           ///  
                                true = verdadeiro = 2d, false = falso
```

```
= 3d.  
76 int      nplots;           ///< Numero de graficos (plots) na sessao.  
77 std::string pstyle;        ///< Estilo utilizado para visualizacao das  
    funcoes e dados.  
78 std::string smooth;       ///< interpolate and approximate data in  
    defined styles (e.g. spline).  
79 std::vector <std::string> tmpfile_list; //;< Lista com nome dos  
    arquivos temporarios.  
80  
81 //  
-----  
    flags  
82 bool fgrid;              ///< 0 sem grid,          1 com grid  
83 bool fhidden3d;           ///< 0 nao oculta,        1 oculta  
84 bool fcontour;            ///< 0 sem contorno,      1 com contorno  
85 bool fsurface;            ///< 0 sem superficie,   1 com superficie  
86 bool flegend;             ///< 0 sem legendad,     1 com legenda  
87 bool ftitle;              ///< 0 sem titulo,        1 com titulo  
88 bool fxlogscale;          ///< 0 desativa escala log, 1 ativa escala  
    log  
89 bool fylogscale;          ///< 0 desativa escala log, 1 ativa escala  
    log  
90 bool fzlogscale;          ///< 0 desativa escala log, 1 ativa escala  
    log  
91 bool fsmooth;             ///< 0 desativa,          1 ativa  
92  
93 //  
-----  
  
94 // Atributos estaticos (compartilhados por todos os objetos)  
95 static int tmpfile_num;           ///< Numero total de  
    arquivos temporarios (numero restrito).  
96 static std::string m_sGNUPlotFileName; //;< Nome do arquivo  
    executavel do gnuplot.  
97 static std::string m_sGNUPlotPath;    ///< Caminho para  
    executavel do gnuplot.  
98 static std::string terminal_std;     ///< Terminal padrao (  
    standart), usado para visualizacoes.  
99  
100 //  
-----  
    Metodos
```

```
101 // Funcoes membro (métodos membro) (funcoes auxiliares)
102 /// @brief Cria arquivo temporario e retorna seu nome.
103 /// Usa get_program_path(); e popen();
104 void init ();
105
106 /// @brief Cria arquivo temporario e retorna seu nome.
107 /// Usa get_program_path(); e popen();
108 void Init() { init(); }
109
110 /// @brief Cria arquivo temporario.
111 std::string create_tmpfile (std::ofstream & tmp);
112
113 /// @brief Cria arquivo temporario.
114 std::string CreateTmpFile (std::ofstream & tmp) { return
115     create_tmpfile(tmp); }
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
126 = 0);
127
128 /// @brief Checa se o arquivo existe.
129 static bool FileExists (const std::string & filename, int mode
130 = 0)
131     { return file_exists( filename, mode );
132
133 public:
134     // Opcional: Seta path do gnuplot manualmente
135     // 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
142      set_GNUPlotPath(path); }
143  /**
144   * @brief Opcional: Seta terminal padrao (standart), usado para
145   * visualizacao dos graficos.
146   * Valores padroes (default): Windows - win, Linux - x11, Mac -
147   * aqua
148  static void set_terminal_std (const std::string & type);
149
150  /// @brief Opcional: Seta terminal padrao (standart), usado para
151  * visualizacao dos graficos.
152  /// Para retornar para terminal janela precisa chamar
153  * ShowOnScreen().
154  /// Valores padroes (default): Windows - win, Linux - x11 ou wxt
155  * (fedora9), Mac - aqua
156  static void Terminal (const std::string & type) {
157      set_terminal_std(type); }

158  /**
-----  

159  * Construtores
160
161  /// @brief Construtor, seta o estilo do grafico na construcao.
162  Gnuplot (const std::string & style = "points");
163
164  /// @brief Construtor, plota um grafico a partir de um vector,
165  * diretamente na construcao.
166  Gnuplot (const std::vector < double >&x,
167          const std::string & title = "",
168          const std::string & style = "points",
169          const std::string & labelx = "x",
170          const std::string & labely = "y");
171
172  /// @brief Construtor, plota um grafico do tipo x_y a partir de
173  * vetores, diretamente na construcao.
```

```
164     Gnuplot (const std::vector < double >&x,
165                 const std::vector < double >&y,
166                 const std::string & title = "",
167                 const std::string & style = "points",
168                 const std::string & labelx = "x",
169                 const std::string & labely = "y");
170
171     /// @brief Construtor, plota um grafico de x_y_z a partir de
172     // vetores, diretamente na construcao.
173     Gnuplot (const std::vector < double >&x,
174               const std::vector < double >&y,
175               const std::vector < double >&z,
176               const std::string & title = "",
177               const std::string & style = "points",
178               const std::string & labelx = "x",
179               const std::string & labely = "y",
180               const std::string & labelz = "z");
181
182     /// @brief Destrutor, necessario para deletar arquivos
183     // temporarios.
184     ~Gnuplot ();
185
186     // -----
187
188     /// @brief Envia comando para o gnuplot.
189     Gnuplot & cmd (const std::string & cmdstr);
190
191     /// @brief Envia comando para o gnuplot.
192     Gnuplot & Cmd (const std::string & cmdstr)      { return cmd(
193         cmdstr); }
194
195     /// @brief Envia comando para o gnuplot.
196     Gnuplot & Command (const std::string & cmdstr) { return cmd(
197         cmdstr); }
198
199     /// @brief Sobrecarga operador <<, funciona como Comando.
200     Gnuplot & operator<< (const std::string & cmdstr);
201
202     // -----
203
```

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

```
224 Gnuplot & savetojpeg (const std::string & filename = "  
225     gnuplot_output");  
  
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     /// @brief Salva sessao do gnuplot para um arquivo filename,  
235     // usando o terminal_type e algum flag adicional  
236     /// Ex:  
237     /// grafico.SaveTo("pressao_X_temperatura","png", "enhanced size  
238         1280,960");  
239     /// Para melhor uso dos flags adicionais consulte o manual do  
240         gnuplot (help term)  
241     Gnuplot& SaveTo(const std::string &filename, const std::string &  
242         terminal_type, std::string flags="");  
  
243     //-----  
244     // set e unset  
245     /// @brief Seta estilos de linhas (em alguns casos sao  
246         necessarias informacoes adicionais).  
247     /// lines, points, linespoints, impulses, dots, steps, fsteps,  
248         histeps,  
249     /// boxes, histograms, filledcurves  
250     Gnuplot & set_style (const std::string & stylestr = "points");  
  
251     //-----  
252     // set e unset  
253     /// @brief Seta estilos de linhas (em alguns casos sao  
254         necessarias informacoes adicionais).  
255     /// lines, points, linespoints, impulses, dots, steps, fsteps,  
256         histeps,  
257     /// boxes, histograms, filledcurves  
258     Gnuplot & Style (const std::string & stylestr = "points")  
259     { return set_style  
260         (stylestr); }  
  
261     //-----  
262     // set e unset  
263     /// @brief Ativa suavizacao.  
264
```

```
250  /// Argumentos para interpolacoes e aproximacoes.  
251  /// csplines, bezier, acsplines (para dados com valor > 0),  
252  /// sbezier, unique,  
253  /// frequency (funciona somente com plot_x, plot_xy, plotfile_x,  
254  /// plotfile_xy (se a suavizacao esta ativa, set_style nao tem  
255  /// efeito na plotagem dos graficos)  
256  Gnuplot & set_smooth (const std::string & stylestr = "csplines")  
257  ;  
258  
259  /// @brief Desativa suavizacao.  
260  Gnuplot & unset_smooth (); // A suavizacao nao e  
261  setada por padrao (default)  
262  
263  /// @brief Ativa suavizacao.  
264  /// Argumentos para interpolacoes e aproximacoes.  
265  /// csplines, bezier, acsplines (para dados com valor > 0),  
266  /// sbezier, unique,  
267  /// frequency (funciona somente com plot_x, plot_xy, plotfile_x,  
268  /// plotfile_xy (se a suavizacao esta ativa, set_style nao tem  
269  /// efeito na plotagem dos graficos)  
270  Gnuplot & Smooth(const std::string & stylestr = "csplines")  
271  { return set_smooth  
272  (stylestr); }  
273  
274  Gnuplot & Smooth( int _fsmooth )  
275  { if( fsmooth =  
276  _fsmooth )  
277  return  
278  set_contour  
279  ();  
280  else  
281  return  
282  unset_contour  
283  ();  
284  }  
285  
286  /// @brief Desativa suavizacao.  
287  //Gnuplot & UnsetSmooth() { return  
288  unset_smooth (); }  
289  
290  /// @brief Escala o tamanho do ponto usado na plotagem.  
291  Gnuplot & set_pointsize (const double pointsize = 1.0);  
292
```

```

279 // @brief Escala o tamanho do ponto usado na plotagem.
280 Gnuplot & PointSize (const double pointsize = 1.0)
281                                         { return
282                                         set_pointsize(
283                                         pointsize); }
284
285
286 // @brief Ativa o grid (padrao = desativado).
287 Gnuplot & set_grid ();
288
289 // @brief Desativa o grid (padrao = desativado).
290 Gnuplot & unset_grid ();
291
292 // @brief Ativa/Desativa o grid (padrao = desativado).
293 Gnuplot & Grid(bool _fgrid = 1)
294                                         { if(fgrid = _fgrid
295                                         )
296                                         return set_grid
297                                         ();
298                                         else
299                                         return
300                                         unset_grid()
301                                         ; }
302
303 // @brief Seta taxa de amostragem das funcoes, ou dos dados de
304 //         interpolacao.
305 Gnuplot & set_samples (const int samples = 100);
306
307 // @brief Seta taxa de amostragem das funcoes, ou dos dados de
308 //         interpolacao.
309 Gnuplot & Samples(const int samples = 100) { return
310                                         set_samples(samples); }
311
312 // @brief Seta densidade de isolinhas para plotagem de funcoes
313 //         como superficies (para plotagen 3d).
314 Gnuplot & set_isosamples (const int isolines = 10);
315
316 // @brief Seta densidade de isolinhas para plotagem de funcoes
317 //         como superficies (para plotagen 3d).
318 Gnuplot & IsoSamples (const int isolines = 10){ return
319                                         set_isosamples(isolines); }
320
321 // @brief Ativa remocao de linhas ocultas na plotagem de

```

```
    superficies (para plotagen 3d).
309  Gnuplot & set_hidden3d ();
310
311 // @brief Desativa remocao de linhas ocultas na plotagem de
      superficies (para plotagen 3d).
312  Gnuplot & unset_hidden3d ();           // hidden3d nao e setado
      por padrao (default)
313
314 // @brief Ativa/Desativa remocao de linhas ocultas na plotagem
      de superficies (para plotagen 3d).
315  Gnuplot & Hidden3d(bool _fhidden3d = 1)
316
      { if (fhidden3d =
            _fhidden3d)
317
            return
            set_hidden3d
            ();
318
      else
319
            return
            unset_hidden3d
            ();
320
      }
321
322 // @brief Ativa desenho do contorno em superficies (para
      plotagen 3d).
323 // @param base, surface, both.
324  Gnuplot & set_contour (const std::string & position = "base");
325
326 // @brief Desativa desenho do contorno em superficies (para
      plotagen 3d).
327  Gnuplot & unset_contour ();           // contour nao e setado por
      default
328
329 // @brief Ativa/Desativa desenho do contorno em superficies (
      para plotagen 3d).
330 // @param base, surface, both.
331  Gnuplot & Contour(const std::string & position = "base")
332
      { return
            set_contour(
            position); }
333
334  Gnuplot & Contour( int _fcontour )
335
      { if ( fcontour =
```

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

```
361     Gnuplot & Legend(const std::string & position = "default")
362         { return set_legend
363             (position); }
364
365     /// @brief Ativa/Desativa a legenda (a legenda é setada por
366     /// o padrao).
367     Gnuplot & Legend(int _flegend)
368         { if(flegend =
369             _flegend)
370             return
371                 set_legend
372                 ();
373             else
374                 return
375                     unset_legend
376                     ();
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     /// o padrao (default)
385
386     /// @brief Ativa/Desativa o titulo da secao do gnuplot.
387     Gnuplot & Title(const std::string & title = "")
388         {
389             return set_title(
390                 title);
391         }
392
393     Gnuplot & Title(int _ftitle)
394         {
395             if(ftitle =
396                 _ftitle)
397                 return set_title
398                 ();
399             else
400                 return
401                     unset_title()
402                     ;
403         }
404
405     /// @brief Ativa/Desativa o titulo da secao do gnuplot.
406     Gnuplot & Title(const std::string & title = "")
407         {
408             if(ftitle =
409                 _ftitle)
410                 return set_title
411                 ();
412             else
413                 return
414                     unset_title()
415                     ;
416         }
417
418     /// @brief Ativa/Desativa o titulo da secao do gnuplot.
419     Gnuplot & Title(int _ftitle)
420         {
421             if(ftitle =
422                 _ftitle)
423                 return set_title
424                 ();
425             else
426                 return
427                     unset_title()
428                     ;
429         }
430
431     /// @brief Ativa/Desativa o titulo da secao do gnuplot.
432     Gnuplot & Title(const std::string & title = "")
433         {
434             if(ftitle =
435                 _ftitle)
436                 return set_title
437                 ();
438             else
439                 return
440                     unset_title()
441                     ;
442         }
443
444     /// @brief Ativa/Desativa o titulo da secao do gnuplot.
445     Gnuplot & Title(int _ftitle)
446         {
447             if(ftitle =
448                 _ftitle)
449                 return set_title
450                 ();
451             else
452                 return
453                     unset_title()
454                     ;
455         }
456
457     /// @brief Ativa/Desativa o titulo da secao do gnuplot.
458     Gnuplot & Title(const std::string & title = "")
459         {
460             if(ftitle =
461                 _ftitle)
462                 return set_title
463                 ();
464             else
465                 return
466                     unset_title()
467                     ;
468         }
469
470     /// @brief Ativa/Desativa o titulo da secao do gnuplot.
471     Gnuplot & Title(int _ftitle)
472         {
473             if(ftitle =
474                 _ftitle)
475                 return set_title
476                 ();
477             else
478                 return
479                     unset_title()
480                     ;
481         }
482
483     /// @brief Ativa/Desativa o titulo da secao do gnuplot.
484     Gnuplot & Title(const std::string & title = "")
485         {
486             if(ftitle =
487                 _ftitle)
488                 return set_title
489                 ();
490             else
491                 return
492                     unset_title()
493                     ;
494         }
495
496     /// @brief Ativa/Desativa o titulo da secao do gnuplot.
497     Gnuplot & Title(int _ftitle)
498         {
499             if(ftitle =
500                 _ftitle)
501                 return set_title
502                 ();
503             else
504                 return
505                     unset_title()
506                     ;
507         }
508
509     /// @brief Ativa/Desativa o titulo da secao do gnuplot.
510     Gnuplot & Title(const std::string & title = "")
511         {
512             if(ftitle =
513                 _ftitle)
514                 return set_title
515                 ();
516             else
517                 return
518                     unset_title()
519                     ;
520         }
521
522     /// @brief Ativa/Desativa o titulo da secao do gnuplot.
523     Gnuplot & Title(int _ftitle)
524         {
525             if(ftitle =
526                 _ftitle)
527                 return set_title
528                 ();
529             else
530                 return
531                     unset_title()
532                     ;
533         }
534
535     /// @brief Ativa/Desativa o titulo da secao do gnuplot.
536     Gnuplot & Title(const std::string & title = "")
537         {
538             if(ftitle =
539                 _ftitle)
540                 return set_title
541                 ();
542             else
543                 return
544                     unset_title()
545                     ;
546         }
547
548     /// @brief Ativa/Desativa o titulo da secao do gnuplot.
549     Gnuplot & Title(int _ftitle)
550         {
551             if(ftitle =
552                 _ftitle)
553                 return set_title
554                 ();
555             else
556                 return
557                     unset_title()
558                     ;
559         }
560
561     /// @brief Ativa/Desativa o titulo da secao do gnuplot.
562     Gnuplot & Title(const std::string & title = "")
563         {
564             if(ftitle =
565                 _ftitle)
566                 return set_title
567                 ();
568             else
569                 return
570                     unset_title()
571                     ;
572         }
573
574     /// @brief Ativa/Desativa o titulo da secao do gnuplot.
575     Gnuplot & Title(int _ftitle)
576         {
577             if(ftitle =
578                 _ftitle)
579                 return set_title
580                 ();
581             else
582                 return
583                     unset_title()
584                     ;
585         }
586
587     /// @brief Ativa/Desativa o titulo da secao do gnuplot.
588     Gnuplot & Title(const std::string & title = "")
589         {
590             if(ftitle =
591                 _ftitle)
592                 return set_title
593                 ();
594             else
595                 return
596                     unset_title()
597                     ;
598         }
599
599 }
```

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

```
427  /// @brief Seta intervalo do eixo z.  
428  Gnuplot & set_zrange (const int iFrom, const int iTo);  
429  
430  /// @brief Seta intervalo do eixo z.  
431  Gnuplot & ZRange (const int iFrom, const int iTo)  
432          { return set_zrange  
433              (iFrom,iTo); }  
434  
435  /// @brief Seta escalonamento automatico do eixo x (default).  
436  Gnuplot & set_xautoscale ();  
437  
438  /// @brief Seta escalonamento automatico do eixo x (default).  
439  Gnuplot & XAutoscale()           { return  
440      set_xautoscale (); }  
441  
442  /// @brief Seta escalonamento automatico do eixo y (default).  
443  Gnuplot & YAutoscale()         { return  
444      set_yautoscale (); }  
445  
446  /// @brief Seta escalonamento automatico do eixo z (default).  
447  Gnuplot & set_zautoscale ();  
448  
449  /// @brief Seta escalonamento automatico do eixo z (default).  
450  Gnuplot & ZAutoscale()        { return  
451      set_zautoscale (); }  
452  
453  /// @brief Ativa escala logaritma do eixo x (logscale nao e  
454      setado por default).  
455  Gnuplot & set_xlogscale (const double base = 10);  
456  
457  /// @brief Desativa escala logaritma do eixo x (logscale nao e  
458      setado por default).  
459  Gnuplot & unset_xlogscale ();  
460  
461  /// @brief Ativa escala logaritma do eixo x (logscale nao e  
462      setado por default).  
463  Gnuplot & XLogscale (const double base = 10) { //if(base)  
464          return  
465          set_xlogscale
```

```
        (base);
461
462    //else
463    //return
464    //unset_xlogscale
465    //}();
466
467    /// @brief Ativa/Desativa escala logaritma do eixo x (logscale
468    //nao e setado por default).
469    Gnuplot & XLogscale(bool _fxlogscale)
470    {
471        if(fxlogscale == _fxlogscale)
472            return
473        set_xlogscale
474        ();
475    }
476
477    /// @brief Ativa escala logaritma do eixo y (logscale nao e
478    //setado por default).
479    Gnuplot & set_ylogscale (const double base = 10);
480
481    /// @brief Ativa escala logaritma do eixo y (logscale nao e
482    //setado por default).
483    Gnuplot & YLogscale (const double base = 10) { return
484        set_ylogscale (base); }
485
486    /// @brief Desativa escala logaritma do eixo y (logscale nao e
487    //setado por default).
488    Gnuplot & unset_ylogscale ();
489
490    /// @brief Ativa/Desativa escala logaritma do eixo y (logscale
491    //nao e setado por default).
492    Gnuplot & YLogscale(bool _fylogscale)
493    {
494        if(fylogscale == _fylogscale)
495            return
496        set_ylogscale
497        ();
498    }
```

```
486                               else
487                               return
488                               unset_ylogscale
489                               ();
490
491     /// @brief Ativa escala logaritma do eixo y (logscale nao e
492     /// setado por default).
493     Gnuplot & set_zlogscale (const double base = 10);
494
495     /// @brief Ativa escala logaritma do eixo y (logscale nao e
496     /// setado por default).
497     Gnuplot & ZLogscale (const double base = 10) { return
498         set_zlogscale (base); }
499
500     /// @brief Desativa escala logaritma do eixo z (logscale nao e
501     /// setado por default).
502     Gnuplot & unset_zlogscale ();
503
504     /// @brief Ativa/Desativa escala logaritma do eixo y (logscale
505     /// nao e setado por default).
506     Gnuplot & ZLogscale(bool fzlogscale)
507
508         { if (fzlogscale =
509             _fzlogscale)
510             return
511                 set_zlogscale
512                 ();
513             else
514             return
515                 unset_zlogscale
516                 ();
517         }
518
519
520     /// @brief Seta intervalo da palette (autoscale por padrao).
521     Gnuplot & set_cbrange (const int iFrom, const int iTo);
522
523     /// @brief Seta intervalo da palette (autoscale por padrao).
524     Gnuplot & CBRange(const int iFrom, const int iTo)
525
526         { return
527             set_cbrange(
528                 iFrom, iTo); }
```

```
514
515 // -----
516
517 // @brief Plota dados de um arquivo de disco.
518 Gnuplot & plotfile_x (const std::string & filename,
519                         const int column = 1, const std::string & title = "")
520 ;
521
522 // @brief Plota dados de um arquivo de disco.
523 Gnuplot & PlotFile (const std::string & filename,
524                         const int column = 1, const std::string & title = "")
525
526 // @return plotfile_x
527 // (filename,
528 // column, title);
529
530
531 // @brief Plota dados de um vector.
532 Gnuplot & plot_x (const std::vector < double >&x, const std::
533 string & title = "");
534
535 // @brief Plota dados de um vector.
536 Gnuplot & PlotVector (const std::vector < double >&x, const std
537 ::string & title = "")
538
539 // @return plot_x( x,
540 // title ); }
541
542 // @brief Plota pares x,y a partir de um arquivo de disco.
543 Gnuplot & plotfile_xy (const std::string & filename,
544                         const int column_x = 1,
545                         const int column_y = 2, const std::string & title =
546                         "");
547
548 // @brief Plota pares x,y a partir de um arquivo de disco.
549 Gnuplot & PlotFile (const std::string & filename,
550                         const int column_x = 1,
551                         const int column_y = 2, const std::string & title =
552                         "")
553
554 // @return plotfile_xy(
555 // filename,
556 // column_x,
557 // column_y, title
```

```
        );
542    }
543
544    /// @brief Plota pares x,y a partir de vetores.
545    Gnuplot & plot_xy (const std::vector < double >&x,
546                        const std::vector < double >&y, const std::string &
547                        title = "");
548
549    /// @brief Plota pares x,y a partir de vetores.
550    Gnuplot & PlotVector (const std::vector < double >&x,
551                          const std::vector < double >&y, const std::string &
552                          title = "")
553
554    { return plot_xy (
555        x, y, title ); }

556
557    /// @brief Plota pares x,y com barra de erro dy a partir de um
558    arquivo.
559    Gnuplot & plotfile_xy_err (const std::string & filename,
560                               const int column_x = 1,
561                               const int column_y = 2,
562                               const int column_dy = 3, const std::string &
563                               title = "");

564
565    /// @brief Plota pares x,y com barra de erro dy a partir de um
566    arquivo.
567    Gnuplot & PlotFileXYErrorBar(const std::string & filename,
568                                  const int column_x = 1,
569                                  const int column_y = 2,
570                                  const int column_dy = 3, const std::string &
571                                  title = "")

572
573    { return
574        plotfile_xy_err
575        (filename,
576         column_x,
577         column_y,
578         column_dy,
579         title ); }

580
581    /// @brief Plota pares x,y com barra de erro dy a partir de
582    vetores.
583    Gnuplot & plot_xy_err (const std::vector < double >&x,
584                           const std::vector < double >&y,
```

```
570         const std::vector < double >&dy,
571         const std::string & title = "");
572
573     /// @brief Plota pares x,y com barra de erro dy a partir de
574     /// vetores.
575     Gnuplot & PlotVectorXYErrorBar(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 valores de x,y,z a partir de um arquivo de
581     /// disco.
582     Gnuplot & plotfile_xyz (const std::string & filename,
583                             const int column_x = 1,
584                             const int column_y = 2,
585                             const int column_z = 3, const std::string & title =
586                             "");
587
588     /// @brief Plota valores de x,y,z a partir de um arquivo de
589     /// disco.
590     Gnuplot & PlotFile (const std::string & filename,
591                         const int column_x = 1,
592                         const int column_y = 2,
593                         const int column_z = 3, const std::string & title =
594                         "")
595
596     /// @brief Plota valores de x,y,z a partir de vetores.
597     Gnuplot & plot_xyz (const std::vector < double >&x,
598                          const std::vector < double >&y,
599                          const std::vector < double >&z, const std::string &
600                          title = "");
```

```
598 // @brief Plota valores de x,y,z a partir de vetores.
599 Gnuplot & PlotVector(const std::vector< double >&x,
600                      const std::vector< double >&y,
601                      const std::vector< double >&z, const std::string &
602                      title = "")
603
604 // @brief Plota uma equacao da forma y = ax + b, voce fornece os
605 // coeficientes a e b.
606 Gnuplot & plot_slope (const double a, const double b, const std
607 :> string & title = "");
608
609 // @brief Plota uma equacao da forma y = ax + b, voce fornece os
610 // coeficientes a e b.
611 Gnuplot & PlotSlope (const double a, const double b, const std
612 :> string & title = "")
613
614 // @brief Plota uma equacao fornecida como uma std::string y=f(
615 // x).
616 // Escrever somente a funcao f(x) e nao y=
617 // A variavel independente deve ser x
618 // Os operadores binarios aceitos sao:
619 // ** exponenciacao,
620 // * multiplicacao,
621 // / divisao,
622 // + adicao,
623 // - subtracao,
624 // # modulo
625 // Os operadores unarios aceitos sao:
626 // - menos,
627 // ! fatorial
628 // Funcoes elementares:
629 // rand(x), abs(x), sgn(x), ceil(x), floor(x), int(x), imag(x),
630 // real(x), arg(x),
631 // sqrt(x), exp(x), log(x), log10(x), sin(x), cos(x), tan(x),
632 // asin(x), acos(x),
```

```
627  /// atan(x), atan2(y,x), sinh(x), cosh(x), tanh(x), asinh(x),
628  /// acosh(x), atanh(x)
629  /// Funções especiais:
630  /// erf(x), erfc(x), inverf(x), gamma(x), igamma(a,x), lgamma(x),
631  /// ibeta(p,q,x),
632  /// besj0(x), besj1(x), besy0(x), besy1(x), lambertw(x)
633  /// Funções estatísticas:
634  /// norm(x), invnorm(x)
635  Gnuplot & plot_equation (const std::string & equation,
636                           const std::string & title = "");
637
638  /// @brief Plota uma equação fornecida como uma std::string y=f(
639  /// x).
640  /// Escrever somente a função f(x) e não y=
641  /// A variável independente deve ser x.
642  /// Exemplo: gnuplot->PlotEquation(CFuncao& obj);
643  // Deve receber um CFuncao, que tem cast para string.
644  Gnuplot & PlotEquation(const std::string & equation,
645                         const std::string & title = "")
646                         { return
647                           plot_equation(
648                             equation, title );
649                         }
650
651  /// @brief Plota uma equação fornecida na forma de uma std::
652  /// string z=f(x,y).
653  /// Escrever somente a função f(x,y) e não z=, as variáveis
654  /// independentes são x e y.
655  Gnuplot & plot_equation3d (const std::string & equation, const
656                           std::string & title = "");
657
658  /// @brief Plota uma equação fornecida na forma de uma std::
659  /// string z=f(x,y).
660  /// Escrever somente a função f(x,y) e não z=, as variáveis
661  /// independentes são x e y.
662  // gnuplot->PlotEquation3d(CPolinomio());
663  Gnuplot & PlotEquation3d (const std::string & equation,
664                           const std::string & title = "")
665                           { return
666                           plot_equation3d(
667                             equation, title );
668                         }
```

```
655
656     /// @brief Plota uma imagem.
657     Gnuplot & plot_image (const unsigned char *ucPicBuf,
658                           const int iWidth, const int iHeight, const std::string & title = "");
659
660     /// @brief Plota uma imagem.
661     Gnuplot & PlotImage (const unsigned char *ucPicBuf,
662                           const int iWidth, const int iHeight, const std::string & title = "")
663                                         { return plot_image (
664                                           ucPicBuf, iWidth,
665                                           iHeight, title); }
666
667     // -----
668
669     // Repete o ultimo comando de plotagem, seja plot (2D) ou splot
670     // (3D)
671     // Usado para visualizar plotagens, aps mudar algumas opcoes de
672     // plotagem
673     // ou quando gerando o mesmo grafico para diferentes dispositivos
674     // (showonscreen, savetops)
675     Gnuplot & replot ();
676
677     // Repete o ultimo comando de plotagem, seja plot (2D) ou splot
678     // (3D)
679     // Usado para visualizar plotagens, aps mudar algumas opcoes de
680     // plotagem
681     // ou quando gerando o mesmo grafico para diferentes dispositivos
682     // (showonscreen, savetops)
683     Gnuplot & Replot()                                     { return replot(); }
684
685
686     // Reseta uma sessao do gnuplot (próxima plotagem apaga
687     // definicoes previas)
688     Gnuplot & reset_plot ();
689
690     // Reseta uma sessao do gnuplot (próxima plotagem apaga
691     // definicoes previas)
692     Gnuplot & ResetPlot()                                 { return reset_plot
693                                         (); }
```

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

85 //
86 // initialize static data
87 //
88 int Gnuplot::tmpfile_num = 0;
89
90 // Se estamos no windows
91 #if defined(WIN32) || defined(_WIN32) || defined(__WIN32__) ||
     defined(__TOS_WIN__)
```

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

```
126         return true;
127     }
128     else
129     {
130         Gnuplot::m_sGNUPlotPath.clear();
131         return false;
132     }
133 }
134
135 // -----
136 // define static member function: set standart terminal, used by
137 // showonscreen
138 void Gnuplot::set_terminal_std(const std::string &type)
139 {
140 #if defined(unix) || defined(__unix) || defined(__unix__)
141     if (type.find("x11") != std::string::npos && getenv("DISPLAY")
142         == NULL)
143     {
144         throw GnuplotException("Can't find DISPLAY variable");
145     }
146 #endif
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 = "\t\n")
```

```
160 {
161     const std::string::size_type len = in.length();
162     std::string::size_type i = 0;
163
164     while ( i < len )
165     {
166         // eat leading whitespace
167         i = in.find_first_not_of (delimiters, i);
168
169         if (i == std::string::npos)
170             return; // nothing left but white space
171
172         // find the end of the token
173         std::string::size_type j = in.find_first_of (delimiters, i)
174             ;
175
176         // push token
177         if (j == std::string::npos)
178             {
179                 container.push_back (in.substr(i));
180
181             return;
182         }
183
184         // set up for next loop
185         i = j + 1;
186     }
187
188     return;
189 }
190
191 // -----
192 //
193 // constructor: set a style during construction
194 //
195 Gnuplot::Gnuplot(const std::string &style)
196 {
197     this->init();
198     this->set_style(style);
```

```
199 }
200
201 // -----
202 // constructor: open a new session, plot a signal (x)
203 Gnuplot::Gnuplot(const std::vector<double> &x,
204                     const std::string &title,
205                     const std::string &style,
206                     const std::string &labelx,
207                     const std::string &labely)
208 {
209     this->init();
210
211     this->set_style(style);
212     this->set_xlabel(labelx);
213     this->set_ylabel(labely);
214
215     this->plot_x(x,title);
216 }
217
218 // -----
219 // constructor: open a new session, plot a signal (x,y)
220 Gnuplot::Gnuplot(const std::vector<double> &x,
221                     const std::vector<double> &y,
222                     const std::string &title,
223                     const std::string &style,
224                     const std::string &labelx,
225                     const std::string &labely)
226 {
227     this->init();
228
229     this->set_style(style);
230     this->set_xlabel(labelx);
231     this->set_ylabel(labely);
232
233     this->plot_xy(x,y,title);
234 }
235
236 //
```

```
-----  
237 // constructor: open a new session, plot a signal (x,y,z)  
238 Gnuplot::Gnuplot(const std::vector<double> &x,  
239                     const std::vector<double> &y,  
240                     const std::vector<double> &z,  
241                     const std::string &title,  
242                     const std::string &style,  
243                     const std::string &labelx,  
244                     const std::string &labely,  
245                     const std::string &labelz)  
246 {  
247     this->init();  
248  
249     this->set_style(style);  
250     this->set_xlabel(labelx);  
251     this->set_ylabel(labely);  
252     this->set_zlabel(labelz);  
253  
254     this->plot_xyz(x,y,z,title);  
255 }  
256  
257 //  
-----  
258 // Destructor: needed to delete temporary files  
259 Gnuplot::~Gnuplot()  
260 {  
261     if ((this->tmpfile_list).size() > 0)  
262     {  
263         for (unsigned int i = 0; i < this->tmpfile_list.size(); i++)  
264             remove( this->tmpfile_list[i].c_str() );  
265  
266         Gnuplot::tmpfile_num -= this->tmpfile_list.size();  
267     }  
268  
269     // A stream opened by popen() should be closed by pclose()  
270 #if defined(WIN32) || defined(_WIN32) || defined(__WIN32__) ||  
271     defined(__TOS_WIN__)  
272     if (_pclose(this->gnucmd) == -1)  
273 #elif defined(unix) || defined(__unix) || defined(__unix__) ||
```

```
    defined(__APPLE__)
273     if (pclose(this->gnucmd) == -1)
274 #endif
275     true;//throw GnuplotException("Problem closing
276     communication to gnuplot");
277 }
278 // -----
279 // Resets a gnuplot session (next plot will erase previous ones)
280 Gnuplot& Gnuplot::reset_plot()
281 {
282     if (this->tmpfile_list.size() > 0)
283     {
284         for (unsigned int i = 0; i < this->tmpfile_list.size(); i++)
285             remove(this->tmpfile_list[i].c_str());
286
287         Gnuplot::tmpfile_num -= this->tmpfile_list.size();
288         this->tmpfile_list.clear();
289     }
290
291     this->nplots = 0;
292
293     return *this;
294 }
295
296 // -----
297 // resets a gnuplot session and sets all variables to default
298 Gnuplot& Gnuplot::reset_all()
299 {
300     if (this->tmpfile_list.size() > 0)
301     {
302         for (unsigned int i = 0; i < this->tmpfile_list.size(); i++)
303             remove(this->tmpfile_list[i].c_str());
304
305         Gnuplot::tmpfile_num -= this->tmpfile_list.size();
306         this->tmpfile_list.clear();
307 }
```

```
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("linepoints") == 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 &&
371 //         // 3-5 columns of data are required
372 //         stylestr.find("xyerrorbars") == std::string::npos &&
373 //         // 4,6,7 columns of data are required
374 //         stylestr.find("xyerrorlines") == std::string::npos &&
375 //         // 4,6,7 columns of data are required
376 //         stylestr.find("boxxyerrorbars") == std::string::npos &&
377 //         // 4,6,7 columns of data are required
378 //         stylestr.find("financebars") == std::string::npos &&
379 //         // 5 columns of data are required
380 //         stylestr.find("candlesticks") == std::string::npos &&
381 //         // 5 columns of data are required
382 //         stylestr.find("vectors") == std::string::npos &&
383 //         stylestr.find("image") == std::string::npos &&
384 //         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
431 //      do arquivo
432 //      this->cmd(cmdstr.str());
433 //      this->replot();                                         // 
434 //          Replota o gráfico, agora salvando o arquivo ps
435 //      ShowOnScreen ();                                       // Volta
436 //          para terminal modo janela
437
438     this->cmd("set term png size 1800,1200");
439
440     std::ostringstream cmdstr;
441     cmdstr << "set output ./Src/" << filename << ".png\"";
442     this->cmd(cmdstr.str());
443     this->Replot();
```

```
441
442     return *this;
443 }
444 // -----
445 // saves a gnuplot session to a png file and return do on screen
446 // terminal
447 Gnuplot& Gnuplot::savetopng(const std::string &filename)
448 {
449 //           // Muda
450 //           o terminal
451 //           this->cmd("set term png enhanced size 1280,960"); // Tipo
452 //           de terminal (tipo de arquivo)
453 //           std::ostringstream cmdstr; // Muda
454 //           o nome do arquivo
455 //           cmdstr << "set output \"\" << filename << ".png\""; // Nome
456 //           do arquivo
457 //           this->cmd(cmdstr.str());
458 //           this->replot(); // //
459 //           Replota o gráfico, agora salvando o arquivo ps
460 //           // //
461 //           Retorna o terminal para o padrão janela
462 //           ShowOnScreen(); // Volta
463 //           para terminal modo janela
464 //           this->replot(); // //
465 //           Replota o gráfico, agora na tela
466 //           SaveTo(filename,"png", "enhanced_size_1280,960");
467
468     return *this;
469 }
470
471 // -----
472
473 // saves a gnuplot session to a jpeg file and return do on screen
474 // terminal
475 Gnuplot& Gnuplot::savetojpeg(const std::string &filename)
476 {
477 //           // Muda o
478 //           terminal
```

```

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

```
664         this ->cmd("setUcontourUbase");
665     }
666     else
667     {
668         this ->cmd("setUcontourU" + position);
669     }
670
671     return *this;
672 }
673
674 // -----
675 // disables contour drawing for surfaces
676 Gnuplot& Gnuplot::unset_contour()
677 {
678     this ->cmd("unsetUcontour");
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("setUsurface");
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("unsetUsurface");
697
698     return *this;
699 }
```

```
700
701 // -----
702 // Sets the title of a gnuplot session
703 Gnuplot& Gnuplot::set_title(const std::string &title)
704 {
705     std::ostringstream cmdstr;
706
707     cmdstr << "set_title\"" << title << "\"";
708     this->cmd(cmdstr.str());
709
710     return *this;
711 }
712
713 // -----
714 // Clears the title of a gnuplot session
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 xlabel \" " << label << "\"";  
754     this->cmd(cmdstr.str());  
755  
756     return *this;  
757 }  
758  
759 //  
-----  
760 // set range  
761 // set the xrange  
762 Gnuplot& Gnuplot::set_xrange(const int iFrom,  
763                               const int iTo)  
764 {  
765     std::ostringstream cmdstr;  
766  
767     cmdstr << "set xrange[" << iFrom << ":" << iTo << "]";  
768     this->cmd(cmdstr.str());  
769  
770     return *this;  
771 }
```

```
772
773 // -----
774 // set autoscale x
775 Gnuplot& Gnuplot::set_xautoscale()
776 {
777     this->cmd("set_xrange_restore");
778     this->cmd("set autoscale_x");
779
780     return *this;
781 }
782
783 // -----
784 // set the yrangle
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";
```

```
881     else
882         cmdstr << "plot ";
883
884     cmdstr << equation << "title\"\"";
885
886     if (title == "")
887         cmdstr << "f(x)=\" " << equation;
888     else
889         cmdstr << title;
890
891     cmdstr << "\\"with" << this->pstyle;
892
893     // Do the actual plot
894     this->cmd(cmdstr.str());
895
896     return *this;
897 }
898
899 // -----
900 // plot an equation supplied as a std::string y=(x)
901 Gnuplot& Gnuplot::plot_equation3d(const std::string &equation,
902                                     const std::string &title)
903 {
904     std::ostringstream cmdstr;
905
906     // command to be sent to gnuplot
907     if (this->nplots > 0 && this->two_dim == false)
908         cmdstr << "replot ";
909     else
910         cmdstr << "splot ";
911
912     cmdstr << equation << "title\"\"";
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
947     std::ostringstream cmdstr;
948
949     // command to be sent to gnuplot
950     if (this->nplots > 0 && this->two_dim == true)
951         cmdstr << "replot ";
952     else
953         cmdstr << "plot ";
954
955     cmdstr << "\\" << filename << "\\using " << column;
956
957     if (title == "")
958         cmdstr << "notitle";
959     else
```

```
958         cmdstr << "title\" " << title << "\\" ";
959
960         if (smooth == "") {
961             cmdstr << "with" << this->pstyle;
962         }
963         else
964             cmdstr << "smooth" << this->smooth;
965
966         // Do the actual plot
967         this->cmd(cmdstr.str()); //nplots++; two_dim = true; already
968             in this->cmd();
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;
1022
1023     // command to be sent to gnuplot
1024     if (this->nplots > 0 && this->two_dim == true)
1025         cmdstr << "replot ";
1026     else
1027         cmdstr << "plot ";
1028
1029     cmdstr << "\\" << filename << "\\using" << column_x << ":" <<
1030         column_y;
1031
1032     if (title == "")
1033         cmdstr << "\\notitle";
1034     else
1035         cmdstr << "\\title\" " << title << "\\";
1036
1037 }
```

```
1033     if (smooth == "")  
1034         cmdstr << "with" << this->pstyle;  
1035     else  
1036         cmdstr << "smooth" << this->smooth;  
1037  
1038     // Do the actual plot  
1039     this->cmd(cmdstr.str());  
1040  
1041     return *this;  
1042 }  
1043  
1044 //  
-----  
  
1045 // Plots a 2d graph from a list of doubles: x y  
1046 Gnuplot& Gnuplot::plot_xy(const std::vector<double> &x,  
1047                             const std::vector<double> &y,  
1048                             const std::string &title)  
1049 {  
1050     if (x.size() == 0 || y.size() == 0)  
1051     {  
1052         throw GnuplotException("std::vectors too small");  
1053         return *this;  
1054     }  
1055  
1056     if (x.size() != y.size())  
1057     {  
1058         throw GnuplotException("Length of the std::vectors differs"  
1059                     );  
1060         return *this;  
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
1083 Gnuplot& Gnuplot::plotfile_xy_err(const std::string &filename,
1084                                     const int column_x,
1085                                     const int column_y,
1086                                     const int column_dy,
1087                                     const std::string &title)
1088 {
1089     // check if file exists
1090     if( !(Gnuplot::file_exists(filename,4)) ) // check existence
1091         and read permission
1092     {
1093         std::ostringstream except;
1094         if( !(Gnuplot::file_exists(filename,0)) ) // check
1095             existence
1096             except << "File \"\\" << filename << "\" does not exist";
1097         else
1098             except << "No read permission for File \"\\" << filename
1099             << "\\"";
1100         throw GnuplotException( except.str() );
1101     }
1102     return *this;
1103 }
1104
1105 std::ostringstream cmdstr;
1106
1107 // command to be sent to gnuplot
1108 if (this->nplots > 0 && this->two_dim == true)
1109     cmdstr << "replot ";
1110 else
1111     cmdstr << "plot ";
1112
1113 cmdstr << "\\" << filename << "\\using" << column_x << ":" <<
```

```

        column_y;

1108
1109     if (title == "")
1110         cmdstr << "notitle";
1111     else
1112         cmdstr << "title\"" << title << "\"";
1113
1114     cmdstr << "with" << this->pstyle << ",\"" << filename << "\"
1115             using"
1116             << column_x << ":" << column_y << ":" << column_dy << "\"
1117             notitle"with"errorbars";

1118     // Do the actual plot
1119     this->cmd(cmdstr.str());
1120
1121     return *this;
1122
1123 // -----
1124 // plot x,y pairs with dy errorbars
1125 Gnuplot& Gnuplot::plot_xy_err(const std::vector<double> &x,
1126                                 const std::vector<double> &y,
1127                                 const std::vector<double> &dy,
1128                                 const std::string &title)
1129 {
1130     if (x.size() == 0 || y.size() == 0 || dy.size() == 0)
1131     {
1132         throw GnuplotException("std::vectors too small");
1133         return *this;
1134     }
1135
1136     if (x.size() != y.size() || y.size() != dy.size())
1137     {
1138         throw GnuplotException("Length of the std::vectors differs"
1139                               );
1140         return *this;
1141     }
1142
1143     std::ofstream tmp;
1144     std::string name = create_tmpfile(tmp);

```

```

1144     if (name == "")
1145         return *this;
1146
1147     // write the data to file
1148     for (unsigned int i = 0; i < x.size(); i++)
1149         tmp << x[i] << " " << y[i] << " " << dy[i] << std::endl;
1150
1151     tmp.flush();
1152     tmp.close();
1153
1154     // Do the actual plot
1155     this->plotfile_xy_err(name, 1, 2, 3, title);
1156
1157     return *this;
1158 }
1159
1160 // -----
1161 // Plots a 3d graph from a list of doubles (x y z) saved in a file
1162 Gnuplot& Gnuplot::plotfile_xyz(const std::string &filename,
1163                                 const int column_x,
1164                                 const int column_y,
1165                                 const int column_z,
1166                                 const std::string &title)
1167 {
1168
1169     // check if file exists
1170     if( !(Gnuplot::file_exists(filename,4)) ) // check existence
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     }
1182
1183     return *this;
1184 }
```

```

1181     std::ostringstream cmdstr;
1182
1183     // command to be sent to gnuplot
1184     if (this->nplots > 0 && this->two_dim == false)
1185         cmdstr << "replot";
1186     else
1187         cmdstr << "splot";
1188
1189     cmdstr << "\n" << filename << "\using" << column_x << ":" <<
1190         column_y << ":" << column_z;
1191
1192     if (title == "")
1193         cmdstr << "notitlewith" << this->pstyle;
1194     else
1195         cmdstr << "title\" << title << "\with" << this->
1196             pstyle;
1197
1198     // Do the actual plot
1199     this->cmd(cmdstr.str());
1200
1201
1202 // -----
1203
1204 // Plots a 3d graph from a list of doubles: x y z
1205 Gnuplot& Gnuplot::plot_xyz(const std::vector<double> &x,
1206                             const std::vector<double> &y,
1207                             const std::vector<double> &z,
1208                             const std::string &title)
1209 {
1210     if (x.size() == 0 || y.size() == 0 || z.size() == 0)
1211     {
1212         throw GnuplotException("std::vectors too small");
1213     }
1214
1215     if (x.size() != y.size() || x.size() != z.size())
1216     {
1217         throw GnuplotException("Length of the std::vectors differs"
1218                               );

```

```
1218         return *this;
1219     }
1220
1221
1222     std::ofstream tmp;
1223     std::string name = create_tmpfile(tmp);
1224     if (name == "")
1225         return *this;
1226
1227     // write the data to file
1228     for (unsigned int i = 0; i < x.size(); i++)
1229     {
1230         tmp << x[i] << " " << y[i] << " " << z[i] << std::endl;
1231     }
1232
1233     tmp.flush();
1234     tmp.close();
1235
1236
1237     this->plotfile_xyz(name, 1, 2, 3, title);
1238
1239     return *this;
1240 }
1241
1242
1243
1244 // -----
1245 // * note that this function is not valid for versions of GnuPlot
1246 // below 4.2
1247 GnuPlot& GnuPlot::plot_image(const unsigned char * ucPicBuf,
1248                               const int iWidth,
1249                               const int iHeight,
1250                               const std::string &title)
1251 {
1252     std::ofstream tmp;
1253     std::string name = create_tmpfile(tmp);
1254     if (name == "")
1255         return *this;
1256
1257     // write the data to file
```

```
1257     int iIndex = 0;
1258     for(int iRow = 0; iRow < iHeight; iRow++)
1259     {
1260         for(int iColumn = 0; iColumn < iWidth; iColumn++)
1261         {
1262             tmp << iColumn << " " << iRow << " " << static_cast<
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 ";
1275     else
1276         cmdstr << "plot ";
1277
1278     if (title == "")
1279         cmdstr << "\n" << name << "\nwith image";
1280     else
1281         cmdstr << "\n" << name << "\ntitle\n" << title << "\n"
1282             with image";
1283
1284     // Do the actual plot
1285     this->cmd(cmdstr.str());
1286
1287     return *this;
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
1307     // int fflush ( FILE * stream );
1308     // If the given stream was open for writing and the last i/o
1309     // operation was an output operation,
1310     // any unwritten data in the output buffer is written to the
1311     // file.
1312     // If the argument is a null pointer, all open files are
1313     // flushed.
1314     // The stream remains open after this call.
1315     fflush(this->gnucmd);
1316
1317
1318     if( cmdstr.find("replot") != std::string::npos )
1319     {
1320         return *this;
1321     }
1322     else if( cmdstr.find("splot") != std::string::npos )
1323     {
1324         this->two_dim = false;
1325         this->nplots++;
1326     }
1327     else if( cmdstr.find("plot") != std::string::npos )
1328     {
1329         this->two_dim = true;
1330         this->nplots++;
1331     }
1332     return *this;
1333 }
1334
1335 //
```

```
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
1356
1357     // if gnuplot not available
1358     if (!Gnuplot::get_program_path())
1359     {
1360         this->valid = false;
1361         throw GnuplotException("Can't find gnuplot");
1362     }
1363
1364     // open pipe
1365     std::string tmp = Gnuplot::m_sGNUPlotPath + "/" + Gnuplot::
1366     m_sGNUPlotFileName;
1367
1368     // FILE *popen(const char *command, const char *mode);
1369     // The popen() function shall execute the command specified by
```

```

        the string command,
1365    // create a pipe between the calling program and the executed
        command, and
1366    // return a pointer to a stream that can be used to either read
        from or write to the pipe.
1367 #if defined(WIN32) || defined(_WIN32) || defined(__WIN32__)
        defined(__TOS_WIN__)
1368     this->gnucmd = _popen(tmp.c_str(),"w");
1369 #elif defined(unix) || defined(__unix) || defined(__unix__)
        defined(__APPLE__)
1370     this->gnucmd = popen(tmp.c_str(),"w");
1371#endif
1372
1373    // popen() shall return a pointer to an open stream that can be
        used to read or write to the pipe.
1374    // Otherwise, it shall return a null pointer and may set errno
        to indicate the error.
1375    if (!this->gnucmd)
1376    {
1377        this->valid = false;
1378        throw GnuplotException("Couldn't open connection to gnuplot
        ");
1379    }
1380
1381    this->nplots = 0;
1382    this->valid = true;
1383    this->smooth = "";
1384
1385    // set terminal type
1386    this->showonscreen();
1387
1388    return;
1389}
1390
1391// -----
1392// Find out if a command lives in m_sGnuPlotPath or in PATH
1393bool Gnuplot::get_program_path()
1394{
1395    // first look in m_sGnuPlotPath for Gnuplot
1396    std::string tmp = Gnuplot::m_sGnuPlotPath + "/" + Gnuplot::

```

```
m_sGNUPlotFileName;

1397
1398 #if defined(WIN32) || defined(_WIN32) || defined(__WIN32__)
     defined(__TOS_WIN__)
1399     if ( Gnuplot::file_exists(tmp,0) ) // check existence
1400 #elif defined(unix) || defined(__unix) || defined(__unix__)
     defined(__APPLE__)
1401     if ( Gnuplot::file_exists(tmp,1) ) // check existence and
           execution permission
1402#endif
1403 {
1404     return true;
1405 }
1406
1407 // second look in PATH for Gnuplot
1408 char *path;
1409 // Retrieves a C string containing the value of the environment
     variable PATH
1410 path = getenv("PATH");
1411
1412 if (path == NULL)
1413 {
1414     throw GnuplotException("Path is not set");
1415     return false;
1416 }
1417 else
1418 {
1419     std::list<std::string> ls;
1420     // split path (one long string) into list ls of strings
1421 #if defined(WIN32) || defined(_WIN32) || defined(__WIN32__)
     defined(__TOS_WIN__)
1422     strtok(ls,path,";");
1423 #elif defined(unix) || defined(__unix) || defined(__unix__)
     defined(__APPLE__)
1424     strtok(ls,path,":");
1425#endif
1426     // scan list for Gnuplot program files
1427     for (std::list<std::string>::const_iterator i = ls.begin();
           i != ls.end(); ++i)
1428     {
1429         tmp = (*i) + "/" + Gnuplot::m_sGNUPlotFileName;
1430 #if defined(WIN32) || defined(_WIN32) || defined(__WIN32__)

```

```

defined(_ _TOS_WIN_ _)
1431           if ( Gnuplot::file_exists(tmp,0) ) // check existence
1432 #elif defined(unix) || defined(_ _unix) || defined(_ _unix_ _) ||
defined(_ _APPLE_ _)
1433           if ( Gnuplot::file_exists(tmp,1) ) // check existence
1434             and execution permission
1435 #endif
1436 {
1437     Gnuplot::m_sGNUPlotPath = *i; // set m_sGNUPlotPath
1438     return true;
1439 }
1440
1441 tmp = "Can't find gnuplot neither in PATH nor in \" " +
Gnuplot::m_sGNUPlotPath + "\" ";
1442 throw GnuplotException(tmp);
1443
1444 Gnuplot::m_sGNUPlotPath = "";
1445 return false;
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         return false;
1459     }
1460     // int _access(const char *path, int mode);
1461     // returns 0 if the file has the given mode,
1462     // it returns -1 if the named file does not exist or is not
1463     // accessible in the given mode
1464     // mode = 0 (F_OK) (default): checks file for existence only
1465     // mode = 1 (X_OK): execution permission

```

```

1464     // mode = 2 (W_OK): write permission
1465     // mode = 4 (R_OK): read permission
1466     // mode = 6          : read and write permission
1467     // mode = 7          : read, write and execution permission
1468 #if defined(WIN32) || defined(_WIN32) || defined(__WIN32__) ||
1469     defined(__TOS_WIN__)
1470     if (_access(filename.c_str(), mode) == 0)
1471 #elif defined(unix) || defined(__unix) || defined(__unix__)
1472     defined(__APPLE__)
1473     if (access(filename.c_str(), mode) == 0)
1474 #endif
1475     {
1476         return true;
1477     }
1478     else
1479     {
1480         return false;
1481     }
1482
1483 // -----
1484 // Opens a temporary file
1485 std::string GnuPlot::create_tmpfile(std::ofstream &tmp)
1486 {
1487 #if defined(WIN32) || defined(_WIN32) || defined(__WIN32__) ||
1488     defined(__TOS_WIN__)
1489     char name[] = "gnuplotiXXXXXX"; //tmp file in working directory
1490 #elif defined(unix) || defined(__unix) || defined(__unix__)
1491     defined(__APPLE__)
1492     char name[] = "/tmp/gnuplotiXXXXXX"; // tmp file in /tmp
1493 #endif
1494
1495     // check if maximum number of temporary files reached
1496     if (GnuPlot::tmpfile_num == GP_MAX_TMP_FILES - 1)
1497     {
1498         std::ostringstream except;
1499         except << "Maximum number of temporary files reached" <<
1500             GP_MAX_TMP_FILES
1501             << ") cannot open more files" << std::endl;

```

```
1499
1500         throw GnuplotException( except.str() );
1501         return "";
1502     }
1503
1504     //int mkstemp(char *name);
1505     // shall replace the contents of the string pointed to by "name"
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)
1527 #endif
1528     {
1529         std::ostringstream except;
1530         except << "Cannot create temporary file \""
1531             << name << "\""
1532             ;
1533         throw GnuplotException(except.str());
1534         return "";
1535     }
1536
1537     tmp.open(name);
1538     if (tmp.bad())
1539     {
1540         std::ostringstream except;
1541         except << "Cannot create temporary file \""
1542             << name << "\""
```

```

        ;
1531     throw GnuplotException(except.str());
1532     return "";
1533 }
1534
1535 // Save the temporary filename
1536 this->tmpfile_list.push_back(name);
1537 Gnuplot::tmpfile_num++;
1538
1539 return name;
1540 }
```

Apresenta-se na listagem 6.5 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 class CInvNumStehfest
14 {
15
16     protected:
17
18         double nn2, nn21, z, ln2_on_t, sum, p, ilt, L, Rd;
19         std::vector <double> v;
20         CReservatorioRadialInfinito radialinfinito;
21         CReservatorioRadialSelado radialselado;
22         CReservatorioRadialManutencao radialmanutencao;
23         CReservatorioLinearInfinito linearinfinito;
24         CReservatorioLinearSelado linearselado;
25         CReservatorioLinearManutencao linearmanutencao;
26
27     public:
```

```

29         CInvNumStehfest( double _RD = 1, double _L = 12 ) : L
30             (_L), Rd(_RD){};
31
32         double Factorial(int _i);
33         double StehfestRadialInfinito(double _TD);
34         double StehfestRadialSelado (double _TD, double _RD
35             );
36         double StehfestRadialManutencao (double _TD, double
37             _RD);
38         double StehfestLinearInfinito (double _xd);
39         double StehfestLinearSelado (double _xd);
40         double StehfestLinearManutencao (double _xd);
41
42     ~CInvNumStehfest(){}
43 };
44
45 #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
    Pi = 3.141516...)
2 #include <cmath> /////inclui biblioteca matematica padrao com
    funcoes bessel para c++17 ou superior
3
4 #include "CInvNumStehfest.h"
5 #include <iostream> //Biblioteca std usada somente pra dar cout e
    encontrar bugs vai ser apagada na versão final
6 #include <algorithm> // usar min(a, b)
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         return acumulador;
```

```
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            ++)
61             z = z + ((pow(k, nn2))*Factorial(2*k))/(Factorial(
62                 nn2-k)*Factorial(k)*Factorial(k-1)*Factorial(n-
63                 k)*Factorial(2*k - n));
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     v.push_back(pow((-1.0),(n+nn2))*z);
92 }
93 sum = 0.0;

```

```

93 ln2_on_t = log(2.0) / _TD;
94
95 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 class CReservatorioLinearInfinito : CGeometriaReservatorio
7 {
```

```

8
9     public:
10
11         CReservatorioLinearInfinito() {};
12
13         virtual double LinearInfinito(double _TD);
14
15         ~CReservatorioLinearInfinito() {};
16
17 };
18
19 #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 #include <cmath> /// inclui biblioteca matematica padrao com
4     funcoes bessel para c++17 ou superior
5
6 double CReservatorioLinearInfinito::LinearInfinito(double _TD)
7 {
8
9     pd = sqrt(4.0*_TD/M_PI);
10
11    return pd;
12 }

```

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

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

```

1 ifndef CRESERVATORIOLINEARMUTENCAO_H_
2 define CRESERVATORIOLINEARMUTENCAO_H_
3
4 include "CGeometriaReservatorio.h"
5
6 class CReservatorioLinearManutencao : CGeometriaReservatorio
7 {
8
9     public:
10

```

```
11             CReservatorioLinearManutencao() {};
12
13         double virtual LinearManutencao(double _xd);
14
15     ~CReservatorioLinearManutencao() {};
16
17 };
18
19 #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 <cmath> /// inclui biblioteca matematica padrao com
5     funcoes bessel para c++17 ou superior
6
7 double CReservatorioLinearManutencao::LinearManutencao(double _xd)
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
6 class CReservatorioLinearSelado : CGeometriaReservatorio
7 {
8
9     public:
10
11         CReservatorioLinearSelado() {};
12
13         virtual double LinearSelado(double _xd);
```

```

14
15         ~CReservatorioLinearSelado() {};
16     };
17
18 #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     double pd = _xd - ((_xd*_xd)/2.0);
12     return pd;
13 }

```

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
15     CReservatorioRadialInfinito(){}; //Construtor default
16
17     virtual double RadialInfinito(double _u, double _Rd);

```

```

18
19     ~CReservatorioRadialInfinito(){}; //Destrutor default
20
21 };
22
23 #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     pd = cyl_bessel_k(0, _Rd*sqrt(_u))/((pow(_u,(3.0/2.0)))*
15         cyl_bessel_k(1, sqrt(_u)));
16
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 class CReservatorioRadialManutencao : CGeometriaReservatorio
7 {
8

```

```

9         public:
10
11             CReservatorioRadialManutencao() {};
12
13             virtual double RadialManutencao(double _u, double
14                                         _RD);
15
16             ~CReservatorioRadialManutencao() {};
17 };
18
19 #endif

```

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

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))))));
16
17     return pd;
18
19 }

```

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
6 class CReservatorioRadialSelado : CGeometriaReservatorio
7 {
8
9     public:
10
11         CReservatorioRadialSelado(){};
12
13         virtual double RadialSelado(double _u, double _RD);
14
15         ~CReservatorioRadialSelado(){};
16
17
18 };
19
20
21 #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
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
18     return pd;

```

14 }

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 "CInvNumStehfest.h"
9 #include "CGnuplot.h"
10
11 class CSimulador
12 {
13
14     protected:
15
16         CGeometriaReservatorio forma;
17         CInvNumStehfest Stehfest;
18         Gnuplot plot, plot2, plot3, plot4, plot5;
19
20         std::vector <double> TD, ReD;
21
22     public:
23
24         CSimulador() {};
25
26         void Solver();
27         void Plot(std::vector <double> _WD, std::vector <
28                     double> _TD, std::string name, bool _setY);
29         void Plot2(std::vector <double> _WD, std::vector <
30                     double> _TD, std::string name, bool _setY);
31         void Plot3(std::vector <double> _WD, std::vector <
32                     double> _TD, std::string name, std::string type,
33                     std::string type2);
34         void Plot4(std::vector <double> _WD, std::vector <
35                     double> _TD, std::string name, std::string type,
36                     std::string type2);
37         void Plot5(std::vector <double> _WD, std::vector <
38                     double> _TD, std::string name, std::string type,
```

```
    std::string type2);  
32  
33         ~CSimulador(){};  
34  
35 };  
36  
37  
38 #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  
   ::string name, bool _setY)  
6 {  
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  
18    plot.set_xlogscale();  
19    plot.Legend("inside left top box");  
20    plot.ShowOnScreen();  
21    plot.plot_xy(_TD, _WD, name);  
22    plot.savetops(name);  
23    cout << "Aperte ENTER para continuar" << endl;  
24    cin.get();  
25  
26 }  
27  
28 void CSimulador::Plot2(vector <double> _WD, vector <double> _TD,  
   std::string name, bool _setY)
```

```
29 {
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("Pressão 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
55     Gnuplot::Terminal("qt");
56
57     plot3.Grid();
58     plot3.set_xlabel(type2);
59     plot3.set_ylabel("pD");
60     plot3.Title("Pressão Adimensional x "+type+" Adimensional")
61             ;
62     plot3.Legend("inside_left_top_box");
63     plot3.ShowOnScreen();
64     plot3.plot_xy(_TD, _WD, name);
65     plot3.savetops(name);
66     cout << "Aperte ENTER para continuar" << endl;
67     cin.get();
```

```
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("Pressão 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 x "+type+" Adimensional")
102  ;
103  plot5.Legend("inside_left_top_box");
104  plot5.ShowOnScreen();
105  plot5.plot_xy(_TD, _WD, name);
106  plot5.savetops(name);
```

```
106         cout << "Aperte ENTER para continuar" << endl;
107         cin.get();
108
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
123     vector <double> linearinfinito, linearmanutencao,
124             linearselado;
125
126
127     for (double i=0.01;i<=1000;i+=0.05)
128         TD.push_back(i);
129
130
131     for (double k=0; k < TD.size(); k++)
132         radialinfinito.push_back(Stehfest.
133             StehfestRadialInfinito(TD[k]));
134
135     cout << "
136         #####"
137         << endl;
138
139     cout << "#oooooooooooooooooooooooooooooooooooooooooooo#
140         #"
141         << endl;
142
143     cout << "#oooooooooooooooooooooooooooooooooooooooooooo#
144         #"
145         << endl;
146
147     cout << "#oooooooooooooooooooooooooooooooooooooooo#
148         #"
149         << endl;
150
151     cout << "
152         #####"
153         << endl;
154
155
156     Plot(radialinfinito, TD, "RadialInfinito", 1);
```

```
133
134         for (double k=0; k < TD.size(); k++)
135             radialselado2.push_back(Stehfest.
136                                     StehfestRadialSelado(TD[k], 2));
137
138         cout << "
139             #####"
140             << endl;
141
142         cout << "#oooooooooooooooooooooooooooooooooooooooooooo#
143             #"
144             << endl;
145
146         cout << "#oooooooooooooooooooooooooooooooooooooooooooo#
147             #"
148             << endl;
149
150         cout << "#oooooooooooooooooooooooooooooooooooooooooooo#
151             #"
152             << endl;
153
154         cout << "
155             #####"
156             << endl;
157
158         Plot(radialselado2, TD, "RadialSelado-Red=2",
159               0);
160
161         for (double k=0; k < TD.size(); k++)
162             radialselado3.push_back(Stehfest.
163                                     StehfestRadialSelado(TD[k], 3));
164
165         cout << "
166             #####"
167             << endl;
168
169         cout << "#oooooooooooooooooooooooooooooooooooooooooooo#
170             #"
171             << endl;
172
173         cout << "#oooooooooooooooooooooooooooooooooooooooooooo#
174             #"
175             << endl;
176
177         cout << "
178             #####"
179             << endl;
180
181         Plot(radialselado3, TD, "RadialSelado-Red=3",
182               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 << "#oooooooooooooooooooooooooooooooooooo#
165             oo# " << endl;
166         cout << "#oooooPlotando Radial Selado para ReD = 4oooo#
167             oo# " << endl;
168         cout << "#oooooooooooooooooooooooooooooooooooo#
169             oo# " << endl;
170         cout << "
171             #####"
172             << endl;
173         cout << "#oooooooooooooooooooooooooooooooooooo#
174             oo# " << endl;
175         cout << "
176             #####"
177             << endl;
178         Plot(radialselado5, TD, "RadialSelado - ReD = 5",
179               0);
180         for (double k=0; k < TD.size(); k++)
```



```

205                               StehfestRadialSelado(TD[k], 8));
206
207   cout << "
208     #####"
209   cout << "#"
210   cout << "#"
211
212
213   Plot(radialselado8, TD, "RadialSelado - ReD = 8",
214         0);
215
216   cout << "
217     #####"
218   cout << "#"
219   cout << "
220     #####"
221   Plot2(radialinfinito, TD, "RadialInfinito", 1);
222
223   for (double k=0; k < TD.size(); k++)
224     radialManutencao2.push_back(Stehfest.
225                               StehfestRadialManutencao(TD[k], 2));
226
227   cout << "
228     #####"
229
230   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.
236                StehfestRadialManutencao(TD[k], 3));
237
238        cout << "
        ##### #####
        #####
<< endl;
239        cout << "# ##### #####
        ##" << endl;
240        cout << "# Plotando Radial Manutencao para ReD=3 #####
        ##" << endl;
241        cout << "
        ##### #####
        #####
<< endl;
242
243        Plot2(radialManutencao3, TD, "RadialManutencao -"
        ReD=3",
        0);
244
245        for (double k=0; k < TD.size(); k++)
246            radialManutencao4.push_back(Stehfest.
247                StehfestRadialManutencao(TD[k], 4));
248
249        cout << "
        ##### #####
        #####
<< endl;
250        cout << "# Plotando Radial Manutencao para ReD=4 #####
        ##" << endl;
```



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

275 Plot2(radialManutencao6, TD, "RadialManutencao - ReD = 6",
0);

277     for (double k=0; k < TD.size(); k++)
279         radialManutencao7.push_back(Stehfest.
300             StehfestRadialManutencao(TD[k], 7));

280 cout << "
# ##### #####
<< endl;

281 cout << "#"
282     #<< endl;
283 cout << "# Plotando Radial Manutencao para ReD = 7"
284     #<< endl;
285 cout << "#"
286     #<< endl;
287 Plot2(radialManutencao7, TD, "RadialManutencao - ReD = 7",
0);

288     for (double k=0; k < TD.size(); k++)
289         radialManutencao8.push_back(Stehfest.
300             StehfestRadialManutencao(TD[k], 8));

291 cout << "
# ##### #####
<< endl;

292 cout << "#"
293     #<< endl;
294 cout << "# Plotando Radial Manutencao para ReD = 8"
295     #<< endl;
296 cout << "
```


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

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

```
1 #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: Descrição

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

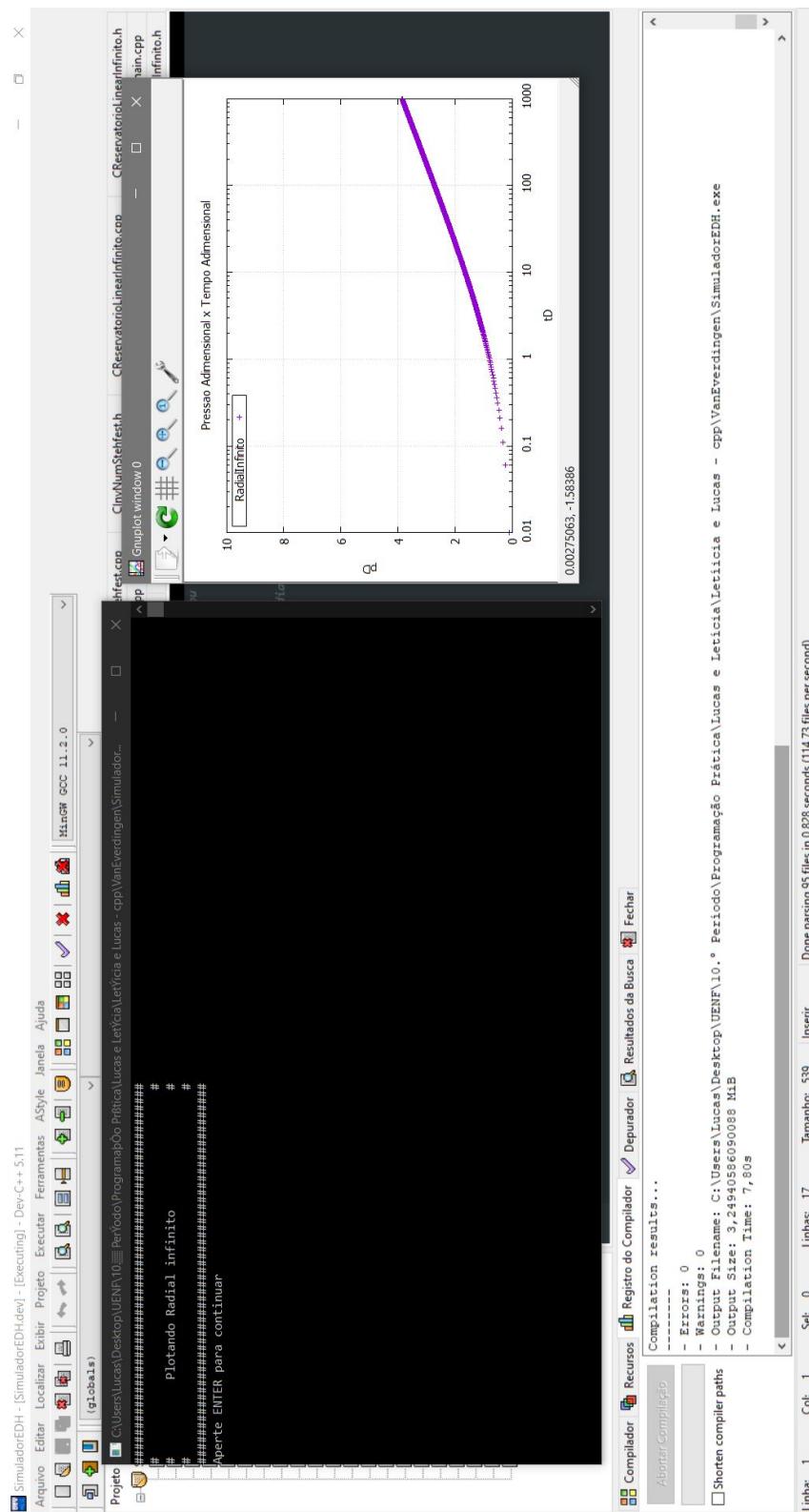


Figura 7.1: Tela inicial do programa mostrando o regime radial infinito

7.2 Teste 2: Regime Radial

Será solicitado ao usuário que clique *enter*, de modo que o regime radial será diagnosticado no reservatório selado para valores distintos de pressão e tempo. Veja Figura

7.2.

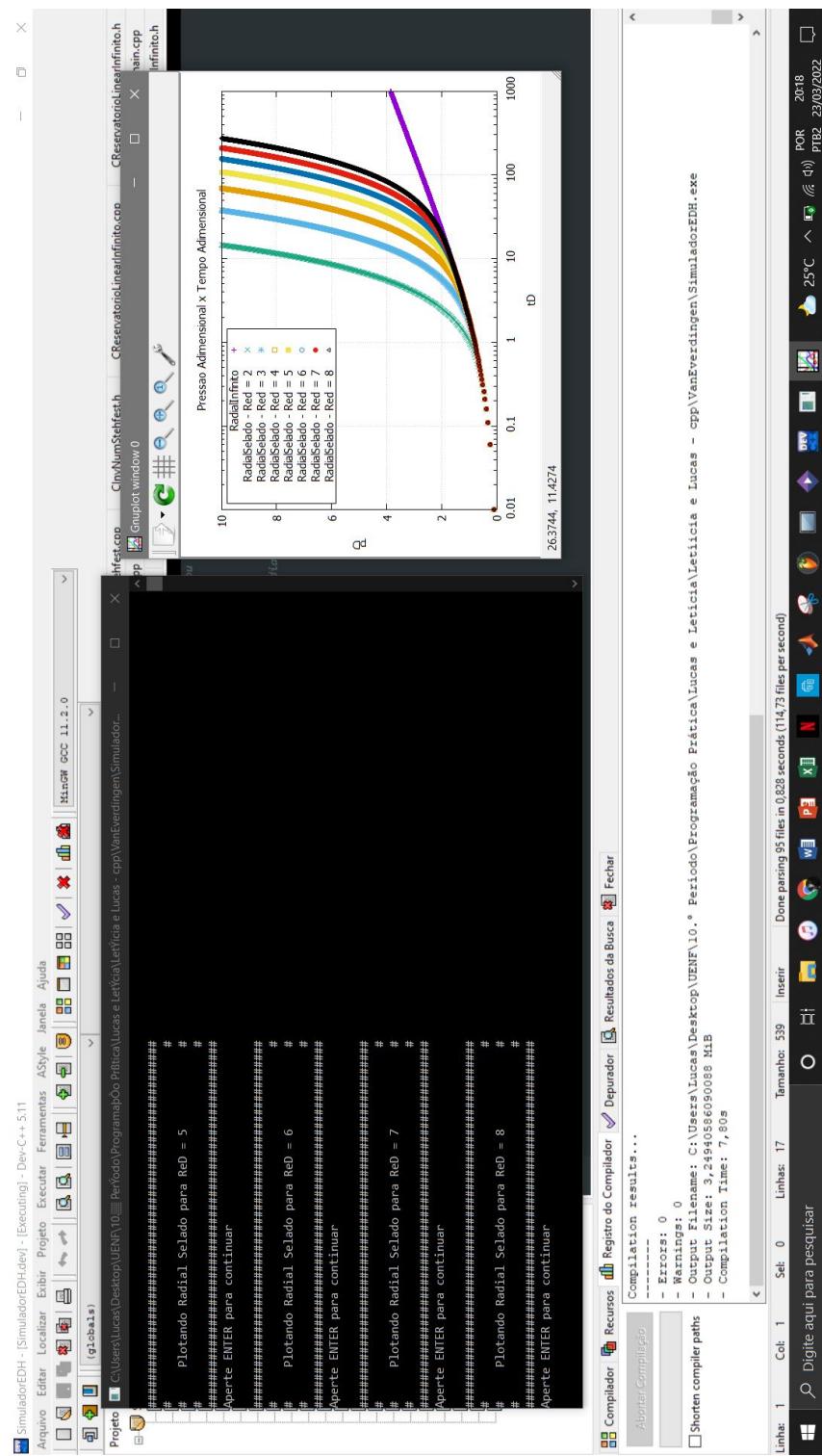


Figura 7.2: Tela do programa mostrando o regime radial selado para valores de pressão e tempo distintos

Será solicitado ao usuário que clique *enter* novamente, de modo que o regime radial será diagnosticado no reservatório infinito. Veja Figura 7.2.

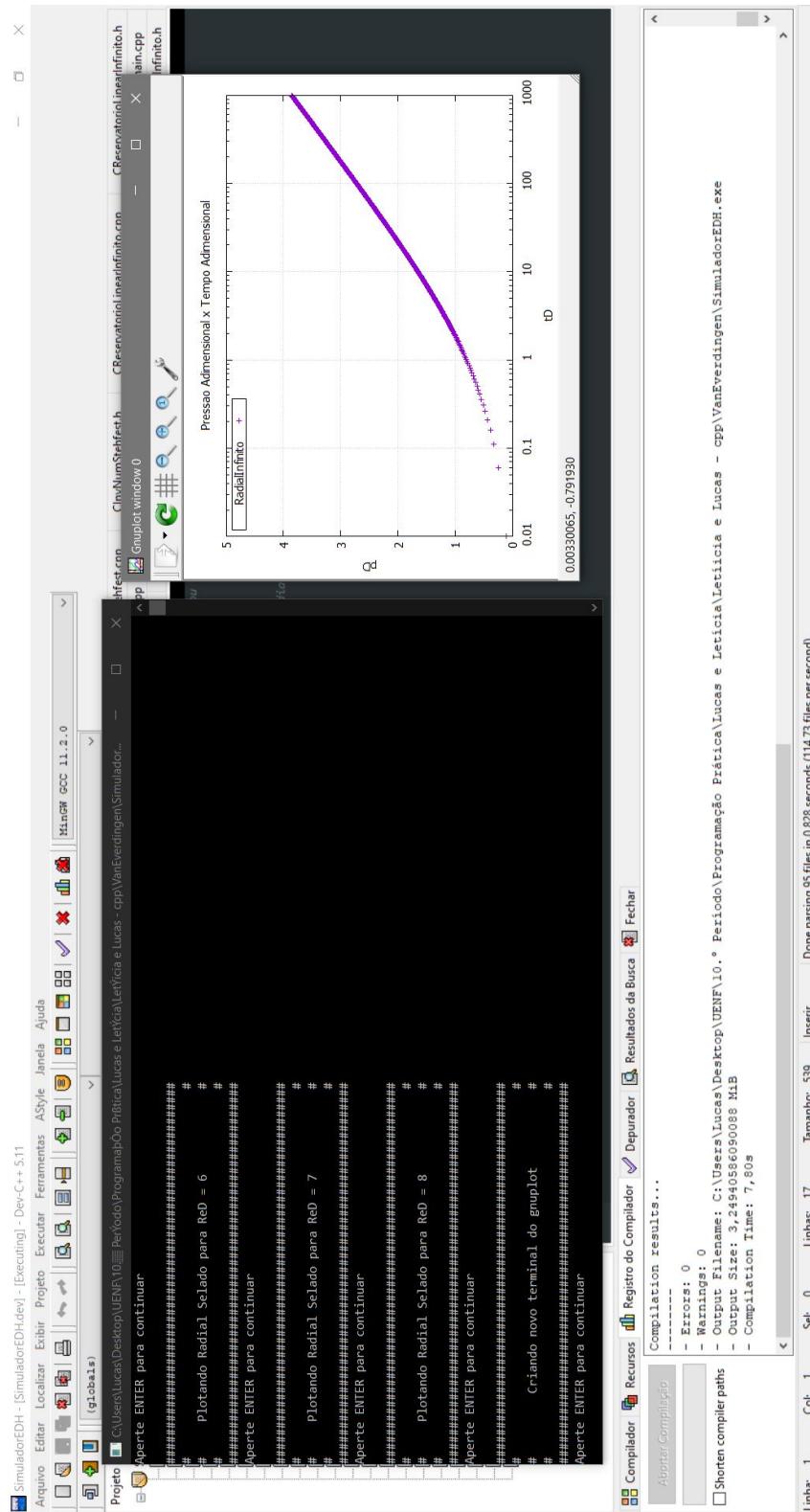


Figura 7.3: Tela do programa mostrando o terminal do Gnuplot para o regime radial infinito

Será solicitado ao usuário que clique *enter*, de modo que o regime radial será diagnosticado no reservatório em manutenção para valores distintos de pressão e tempo. Veja Figura 7.4.

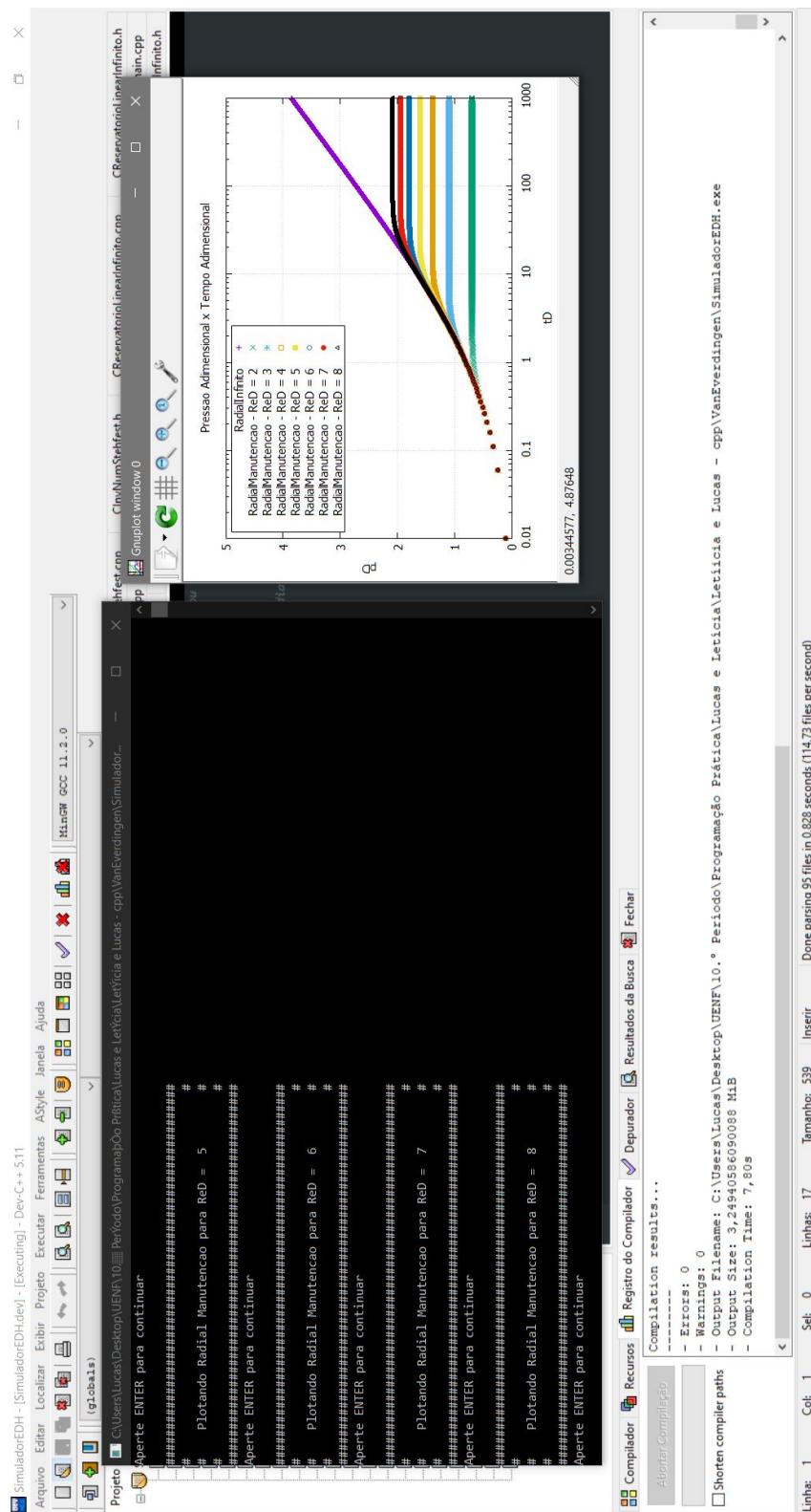


Figura 7.4: Tela do programa mostrando o regime radial com manutenção de pressão

7.3 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, com valores de pressão e tempo adimensionais. Veja Figura

7.5.

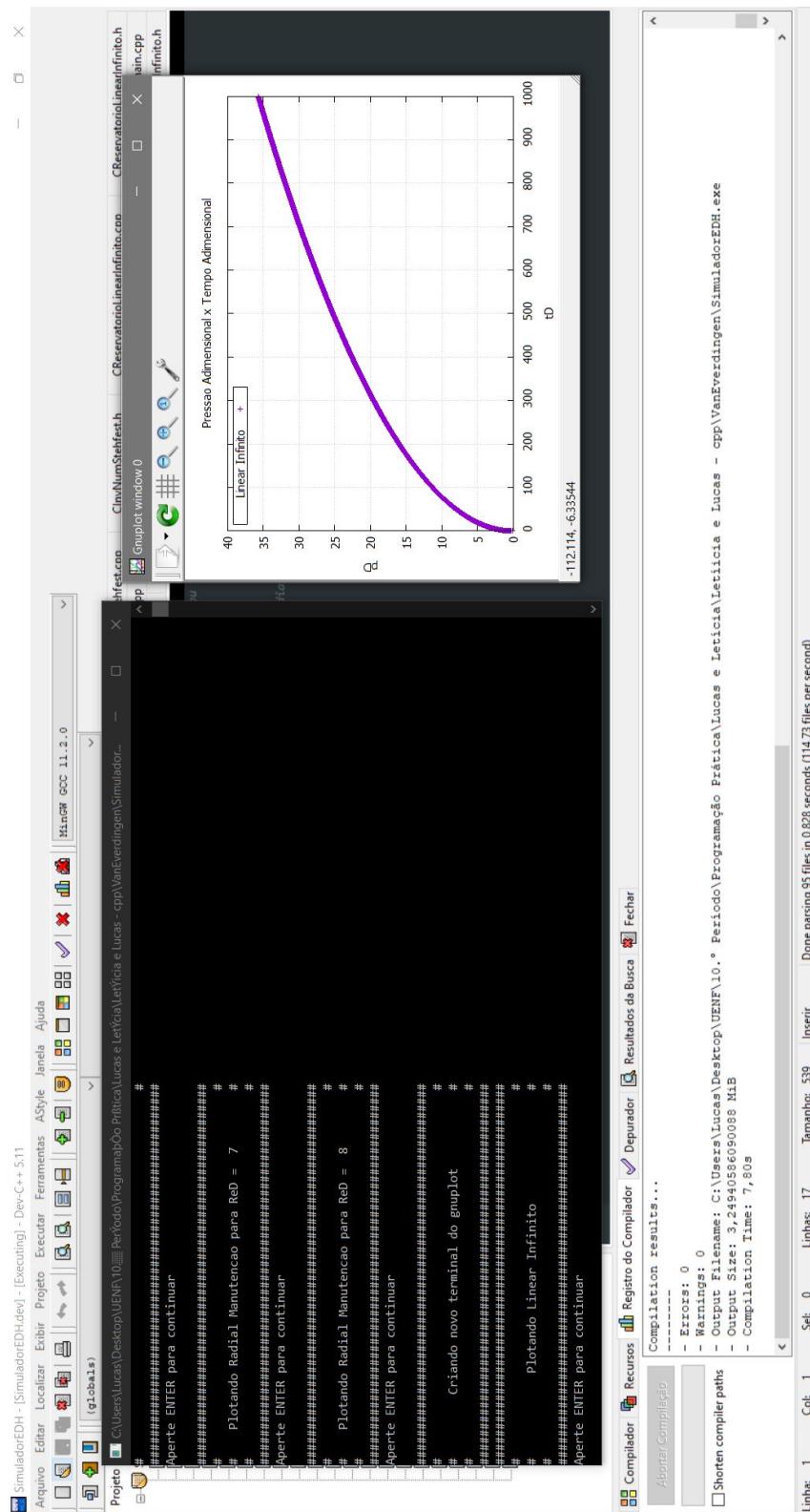


Figura 7.5: Tela do programa mostrando o regime linear

Será solicitado ao usuário que clique *enter*, de modo que o regime radial será diagnosticado no reservatório selado para valores distintos de pressão e tempo. Veja Figura 7.4

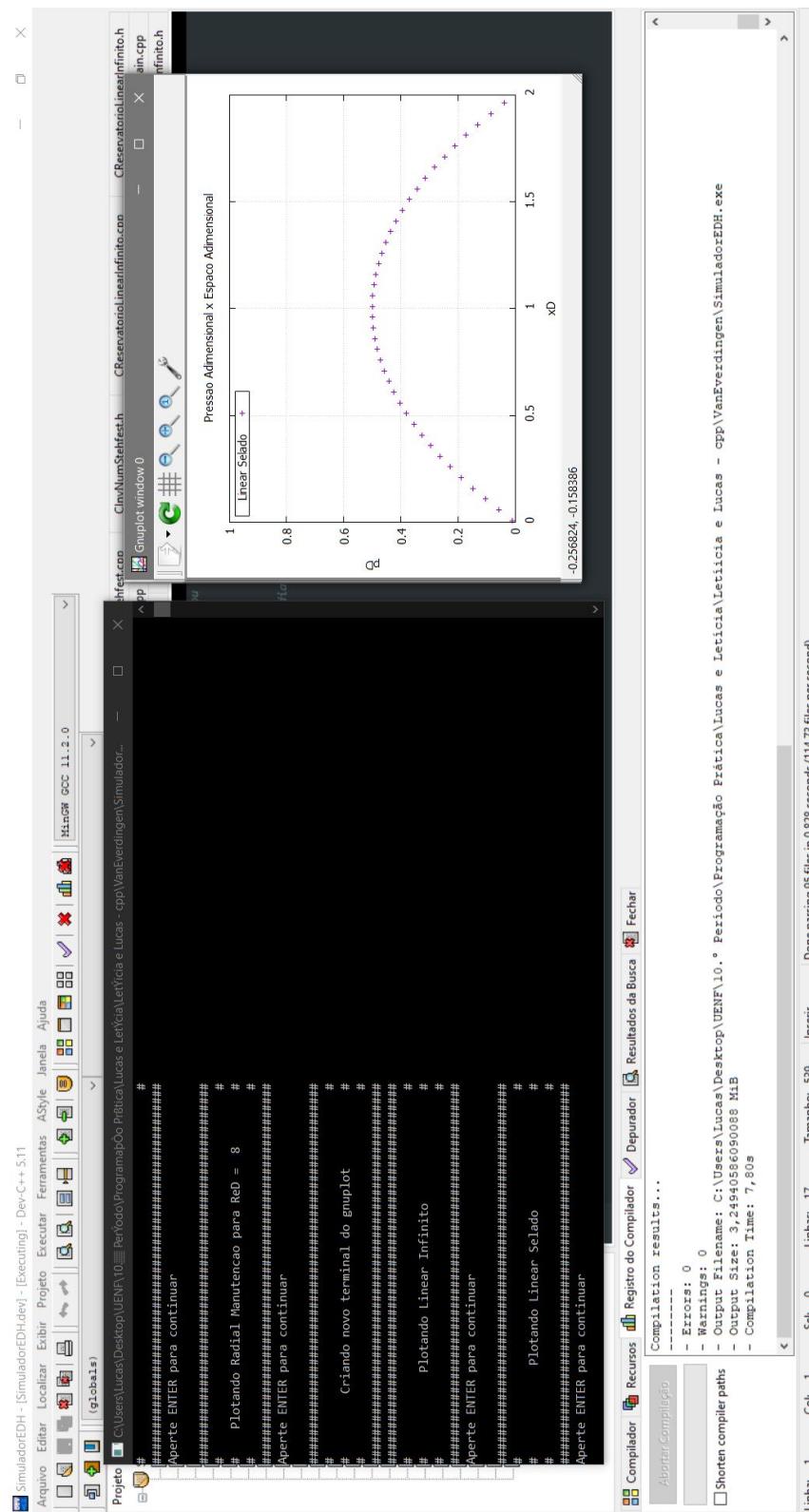


Figura 7.6: Tela do programa mostrando o regime linear selado para valores de pressão e posição distintos

A tela final do programa mostra que todos os resultados gráficos foram salvos em disco. Veja Figura 7.7.

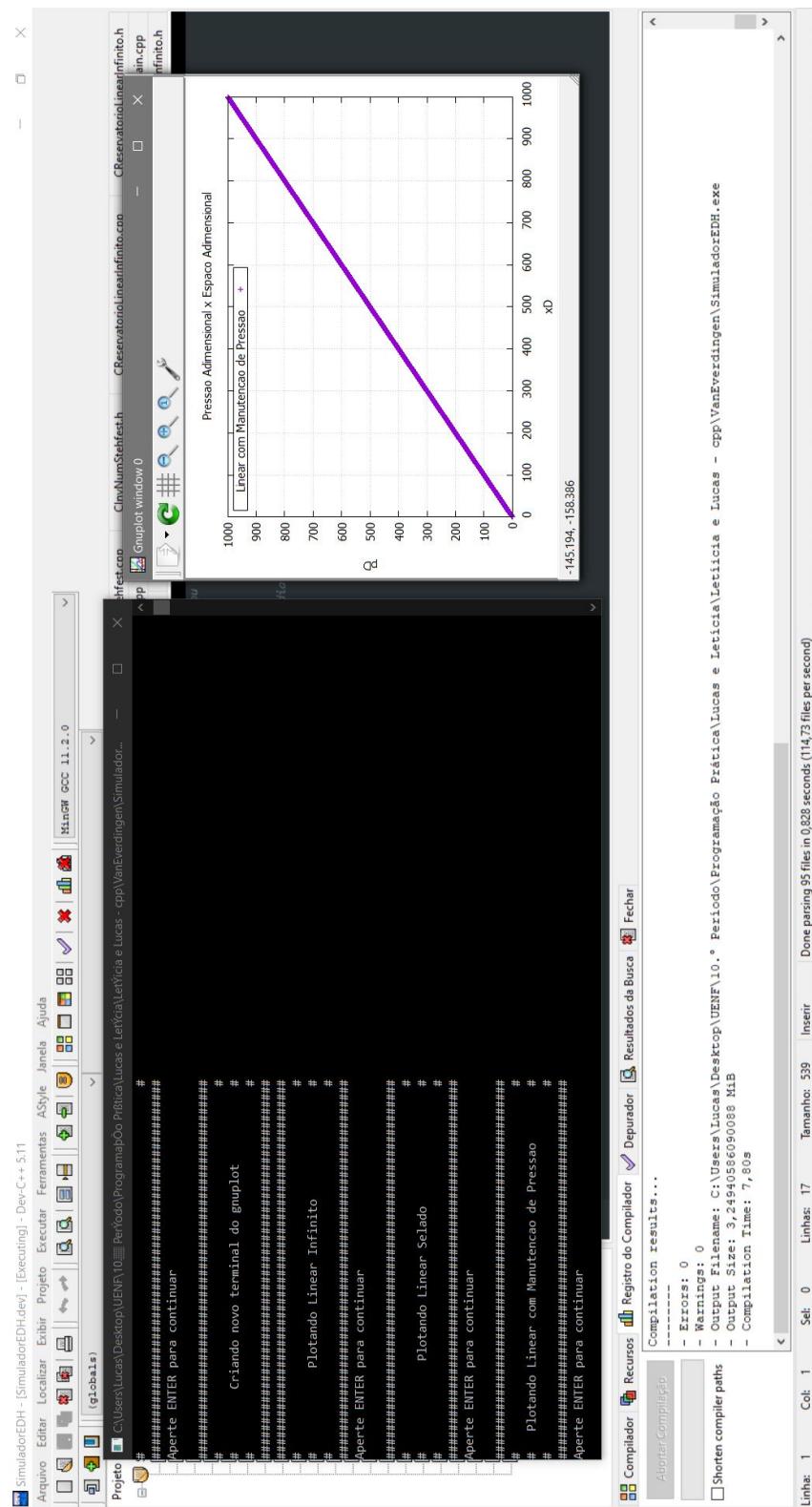


Figura 7.7: Tela do programa mostrando o regime linear com manutenção de pressão

Será solicitado ao usuário que clique *enter*, de modo que o regime linear será diagnosticado no reservatório em manutenção para valores distintos de pressão e posição. Veja Figura 7.8.

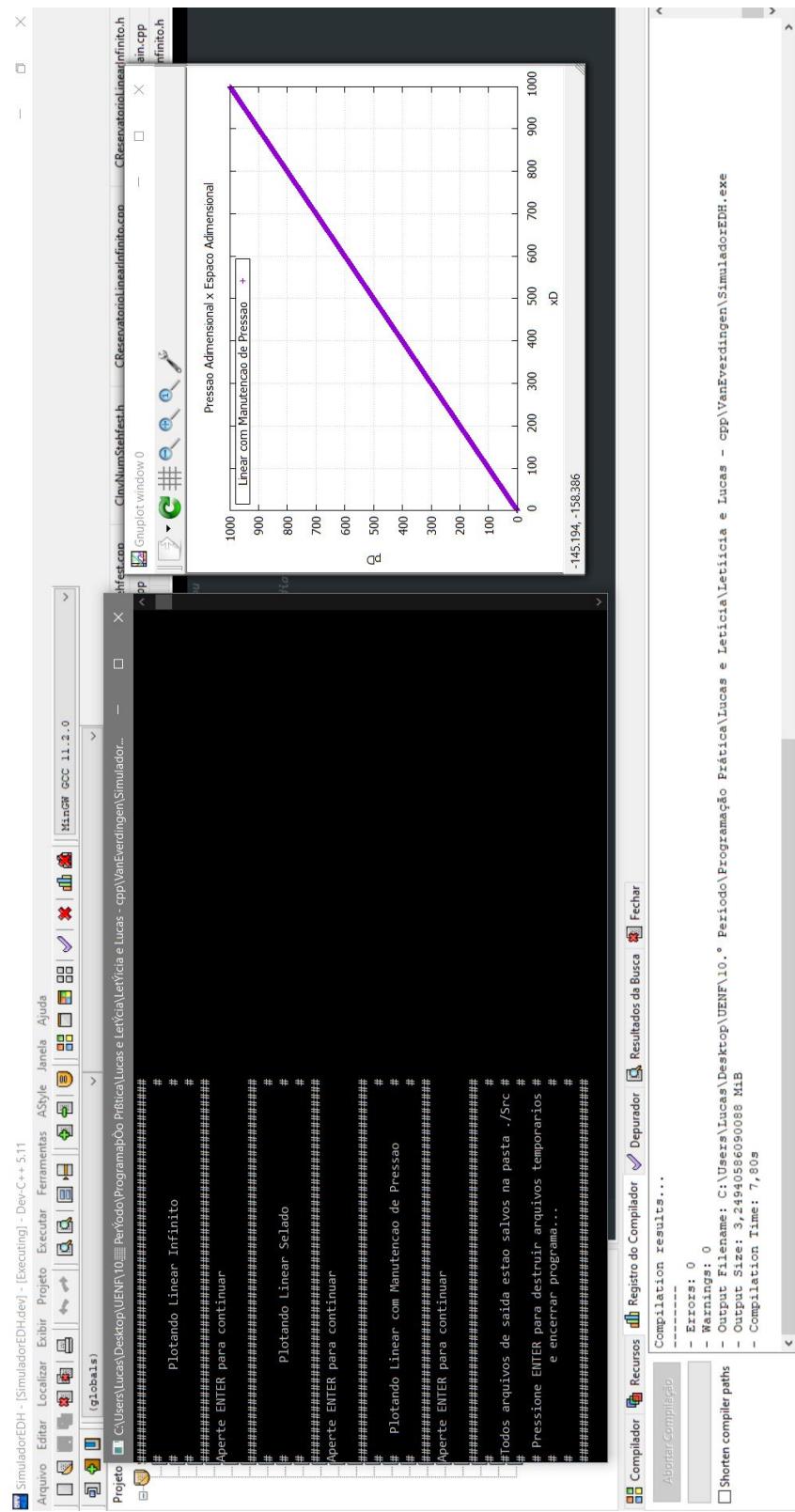


Figura 7.8: Tela do programa mostrando que os gráficos plotados foram salvos

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

Índice dos Componentes

	C G
C	CReservatorioLinearInfinito CReservatorioLinearManutencao CReservatorioLinearSelado CReservatorioRadialInfinito CReservatorioRadialManutencao CReservatorioRadialSelado CSimulador CGeometriaReservatorio CInvNumSteifest
G	Gnuplot GnuplotException

Gerado por  doxygen 1.9.3

