

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

PROJETO ENGENHARIA
DESENVOLVIMENTO DO SOFTWARE
SIMULADOR DE HIDRÁULICA DE PERFURAÇÃO DE POÇO - SHPP
TRABALHO DA DISCIPLINA PROGRAMAÇÃO PRÁTICA

Versão 1:
GLEISON MONTEIRO ALVES
LUÍSA FIGUEIREDO CAMPBELL
Prof. André Duarte Bueno

MACAÉ - RJ
Julho - 2019

Sumário

1	Introdução	1
1.1	Escopo do problema	1
1.2	Objetivos	1
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	6
3.1	Análise de domínio	6
3.2	Formulação teórica	7
3.2.1	Geometria de poço	7
3.2.2	Modelo reológico	10
3.2.3	Formulação Físico-Matemática	11
3.3	Identificação de pacotes – assuntos	14
3.4	Diagrama de pacotes – assuntos	15
4	AOO – Análise Orientada a Objeto	16
4.1	Diagramas de classes	16
4.1.1	Dicionário de classes	16
4.2	Diagrama de sequência – eventos e mensagens	18
4.2.1	Diagrama de sequência geral	18
4.3	Diagrama de comunicação	18
4.4	Diagrama de máquina de estado	18
4.5	Diagrama de atividades	19

5	Projeto	20
5.1	Projeto do sistema	20
5.2	Projeto orientado a objeto – POO	21
5.3	Diagrama de componentes	22
5.4	Diagrama de implantação	23
6	Implementação	24
6.1	Código fonte	24
7	Teste	53
7.1	Teste 1: Perda de carga para a fase 1	53
7.2	Teste 2: Perda de carga de todas as fases	56
8	Documentação	60
8.1	Documentação do usuário	60
8.1.1	Como executar o software	60
8.2	Documentação para desenvolvedor	61
8.2.1	Dependências	61
8.2.2	Como gerar a documentação usando doxygen	62

Capítulo 1

Introdução

Neste projeto de engenharia será desenvolvido o Simulador de Hidráulica de Perfuração de Poço (SHPP), um software aplicado à engenharia de petróleo e que se baseia na orientação a objetos em linguagem C++. Esse software irá permitir que estudantes e profissionais sejam capazes de simular uma importante variável da perfuração de poços de petróleo: a perda de carga associada ao escoamento de um fluido de perfuração em tubulações.

1.1 Escopo do problema

O objetivo principal ao se projetar a hidráulica de perfuração do poço é dimensionar as perdas de carga associadas ao escoamento do fluido de perfuração pelo interior da coluna, pela broca e pelo anular. Essa informação é fundamental para a escolha e dimensionamento dos equipamentos de superfície, em especial da bomba de lama. A bomba deve ser capaz de bombear o fluido para o poço e de manter uma determinada pressão, suficiente para impedir o desmoronamento do poço e a ocorrência de fluxo indesejado de fluidos da formação (*kick*) ou, em casos mais extremos, um *blowout* e a volta do fluido para a superfície.

Dentre os diversos cálculos a serem efetuados, podem ser listados:

- Cálculo de velocidade dentro do anular e do tubo;
- Cálculo das perdas de carga do fluido para fluxo laminar;
- Cálculo das perdas de carga do fluido para fluxo turbulento;

1.2 Objetivos

Os objetivos deste projeto de engenharia são:

- Objetivo geral:

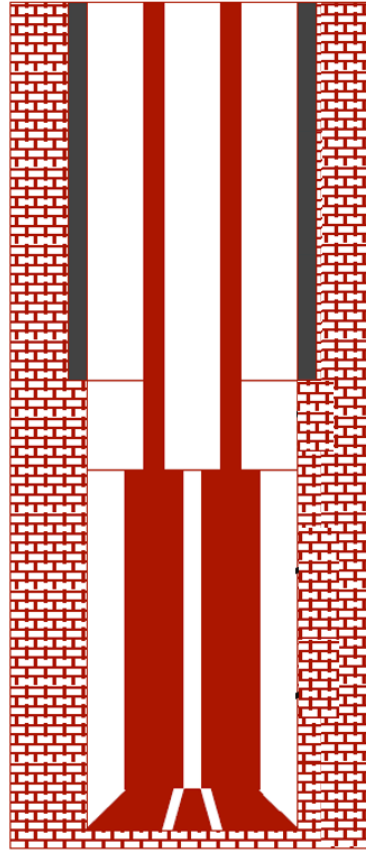


Figura 1.1: Perda de carga no poço: anular e tubos

- Desenvolver um simulador de hidráulica de poço que execute cálculos e gere gráficos. Para tal, deve-se usar as equações de perda de carga provenientes da mecânica dos fluidos adaptadas à indústria do petróleo.
- Objetivos específicos:
 - Modelar matematicamente as características de um poço, bem como a perda de carga associada a sua perfuração;
 - Calcular as velocidades no anular e nos tubos, de acordo com a geometria do poço;
 - Calcular as perdas de carga na tubulação;
 - Calcular as perdas de carga no anular;
 - Salvar os resultados em modo texto no disco;
 - Gerar um gráfico que mostre a perda de carga no poço por seção perfurada;

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	Simulador de Hidráulica de Perfuração de Poço - SHPP
Componentes principais	Sistema para cálculo de perda de carga previamente à perfuração de um poço de petróleo
Missão	Ferramenta de treinamento/ensino na área de engenharia de poço

2.2 Especificação

O software descrito nesse projeto tem como função simular os cálculos relacionados à hidráulica de perfuração de poço, em específico relacionados às perdas de cargas que ocorrem ao longo da coluna de perfuração, broca e anular. Os resultados desses cálculos estarão disponíveis em um arquivo em modo texto e em gráficos. O software irá interagir com o usuário, de forma a obter informações sobre a geometria do poço, tubulares e fluido de perfuração utilizados. A entrada do usuário será usada nos cálculos. Ao final da simulação, será gerado um gráfico de perda de carga por seção, e essa informação será exportada para um arquivo em modo texto.

Programas externos: o presente software utiliza o software externo *gnuplot* para plotar gráficos.

Licença: o presente software tem licença GPL 2.0, conforme consta no site <http://softwarelivre.org>.

2.2.1 Requisitos funcionais

Apresenta-se a seguir os requisitos funcionais do sistema.

RF-01	O usuário deverá entrar com informações sobre o fluido de perfuração.
RF-02	O usuário deverá informar o nome do arquivo texto de onde serão lidas as informações de geometria de poço.
RF-03	O usuário poderá escolher exportar os resultados como arquivo texto.
RF-04	O usuário poderá plotar seus resultados em um gráfico. O gráfico poderá ser salvo como imagem.
RF-05	Nos casos em que o software for plotar gráficos, o software externo <i>gnuplot</i> http://www.gnuplot.org deverá estar instalado no sistema.

2.2.2 Requisitos não funcionais

RNF-01	Os cálculos devem ser feitos utilizando-se fórmulas provenientes da Mecânica dos Fluidos.
RNF-02	O programa deverá ser multi-plataforma, podendo ser executado em <i>Windows</i> , <i>GNU/Linux</i> ou <i>Mac</i> .

2.3 Casos de uso

A Tabela 2.1 mostra um cenário de funcionamento do programa.

Tabela 2.1: Exemplo de caso de uso geral

Nome do caso de uso:	Cálculo de perda de carga em um poço.
Resumo/descrição:	Determinação da perda de carga por seção perfurada.
Etapas:	<ol style="list-style-type: none"> 1. Inserir dados do fluido de perfuração e da geometria do poço. 2. Calcular perdas de carga relacionadas à hidráulica de poço. 3. Analisar os resultados obtidos.
Cenários alternativos:	Um cenário alternativo envolve uma entrada errada do usuário (por exemplo, entrar com um elemento da coluna de perfuração maior do que o diâmetro da fase sendo perfurada), o que iria interferir com a grandeza do problema real.

2.3.1 Diagrama de caso de uso geral

O diagrama de caso de uso geral da Figura 2.1 mostra o usuário inserindo dados do fluido e do poço, calculando as perdas de carga e analisando os resultados.

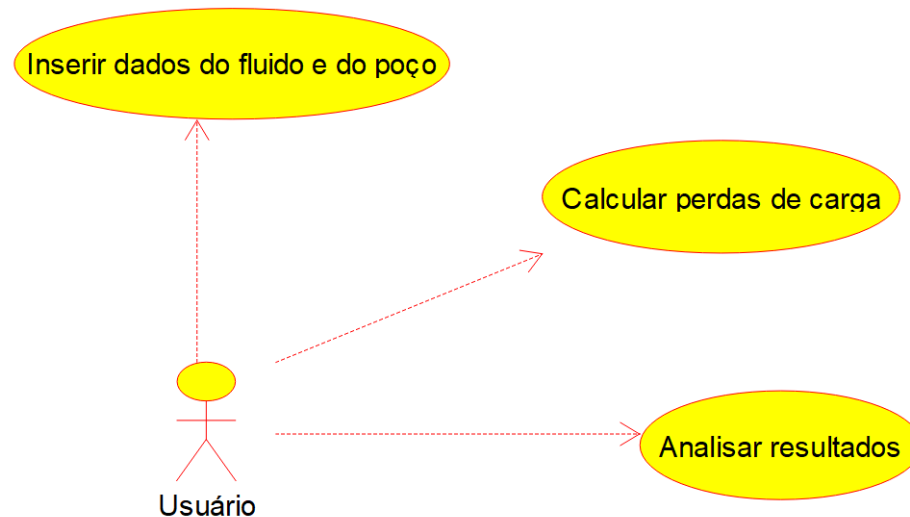


Figura 2.1: Diagrama de caso de uso geral

2.3.2 Diagrama de caso de uso específico

O diagrama de caso de uso específico da Figura 2.2 mostra o usuário inserindo dados do fluido e do poço. O simulador calcula, então, as perdas de carga. O usuário escolhe a opção de salvar em modo texto e plotar um gráfico com os resultados referentes à primeira fase do poço. Posteriormente, o usuário analisa os resultados.

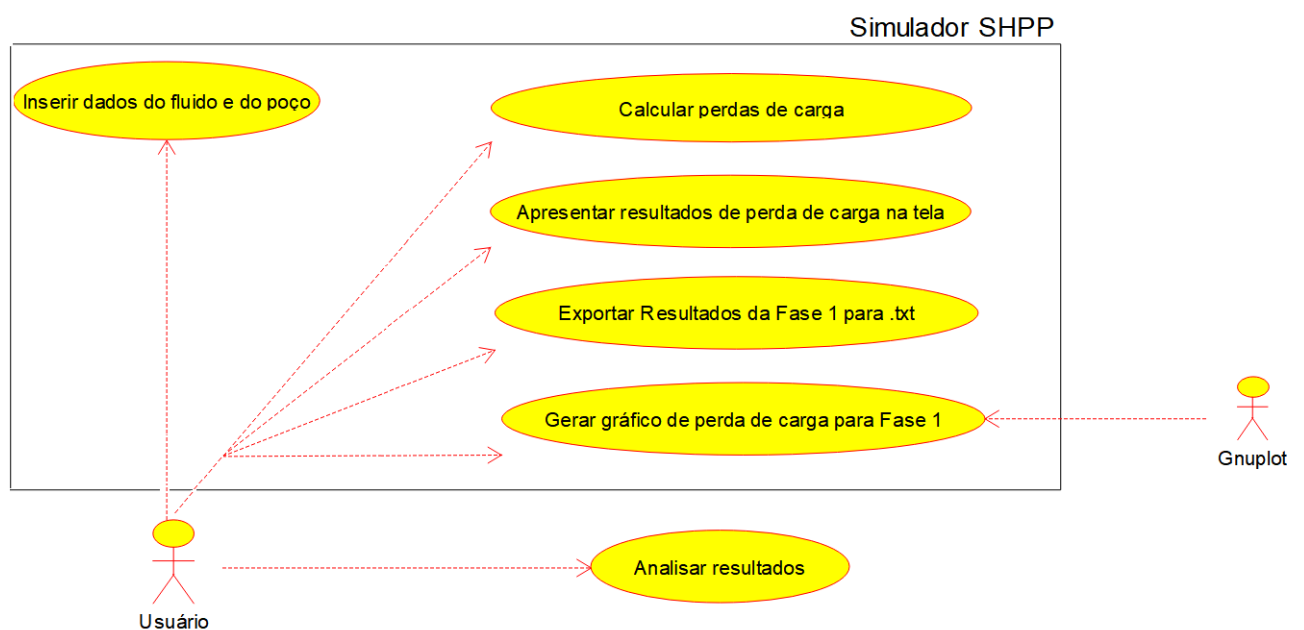


Figura 2.2: Diagrama de caso de uso específico - Analisando resultados para a Fase 1 do poço

Capítulo 3

Elaboração

Neste capítulo serão apresentados o estudo de conceitos relacionados ao sistema a ser desenvolvido, a análise de domínio e a identificação de pacotes.

3.1 Análise de domínio

- O *software* será desenvolvido na Universidade Estadual do Norte Fluminense Darcy Ribeiro com auxílio do corpo docente.
- O *software* envolve conceitos abordados nas disciplinas de Engenharia de Poço, Mecânica de Fluidos, Cálculo Numérico e Programação Orientada a Objeto em C++.

A engenharia de poço é uma disciplina dentro da engenharia de petróleo que envolve todos os processos pertinentes à criação de um poço. Representa uma área da indústria extremamente dinâmica, pois mesmo com experiência prévia e se baseando no histórico de perfuração de outros poços em uma mesma área, a perfuração de dois poços nunca será a mesma. Tratando-se de uma atividade de elevado custo e alta imprevisibilidade, é necessária a criação de projetos de poço a fim de se buscar o melhor resultado, no menor tempo e ao menor custo.

A mecânica dos fluidos é o ramo da engenharia que estuda o comportamento físico dos fluidos e suas propriedades. Os aspectos teóricos e práticos da mecânica dos fluidos são de fundamental importância para a solução de diversos problemas encontrados habitualmente na engenharia de petróleo, sendo suas principais aplicações destinadas ao estudo de escoamentos de líquidos e gases, máquinas hidráulicas, aplicações de pneumática e hidráulica industrial, entre outros. O estudo da mecânica dos fluidos é dividido basicamente em dois ramos, a estática dos fluidos e a dinâmica dos fluidos. A dinâmica dos fluidos é responsável pelo estudo e comportamento dos fluidos em regime de movimento acelerado, no qual se faz presente a ação de forças externas responsáveis pela perda de carga. A mecânica dos fluidos também

está diretamente relacionada à representação de sistemas de engenharia, ou seja, a transferência de calor e a manutenção dos equipamentos. Nesta primeira versão considera-se sistemas isotérmicos.

3.2 Formulação teórica

A engenharia de perfuração lida principalmente com o fluxo de fluidos através da coluna de perfuração e do espaço anular entre a coluna de perfuração e o poço aberto. Para determinar as perdas de carga durante a perfuração de um poço, é necessário compreender alguns conceitos básicos relacionados às disciplinas citadas. Dentre eles, pode-se destacar o modelo reológico utilizado para explicar o escoamento de um fluido de perfuração, a geometria de um poço, os regimes de fluxo de um fluido e as fórmulas existentes para a perda de carga de um fluido escoando em tubulações e no anular.

3.2.1 Geometria de poço

Para os cálculos de perda de carga, é necessário definir a geometria do poço, pois esses valores são utilizados nos cálculos de velocidade, representando as áreas por onde passam o fluido. É preciso conhecer as seguintes informações em relação ao poço e à coluna de perfuração:

1. Profundidade total do poço;
2. Diâmetro das fases sendo perfuradas, ou seja, o diâmetro do poço aberto (e da broca);
3. Diâmetro interno (ID) dos revestimentos usados em todas as fases;
4. Diâmetro externo (OD) e interno (ID) de todos os tubos de perfuração (*drill pipes* – DP);
5. Diâmetro externo (OD) e interno (ID) de todos os comandos de perfuração (*drill collars* – DC).

Para a elaboração deste software, são sugeridas geometrias usualmente utilizadas em poços do pré-sal no Brasil, como consta nas Figuras 3.1, 3.2, 3.3 e 3.4. Apesar de o usuário ter a opção de selecionar qualquer geometria de poço desejada, o uso de geometrias padronizadas garante a saída de resultados mais representativos.

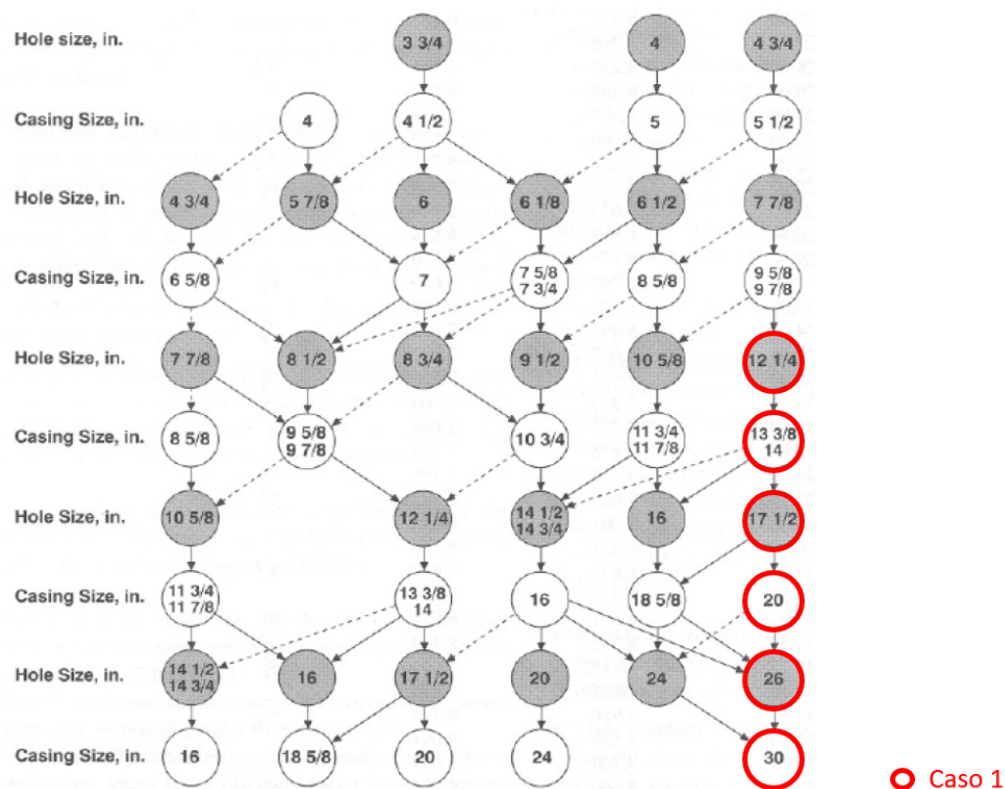


Figura 3.1: Sequência de diâmetros de revestimento e de poço aberto - Caso 1

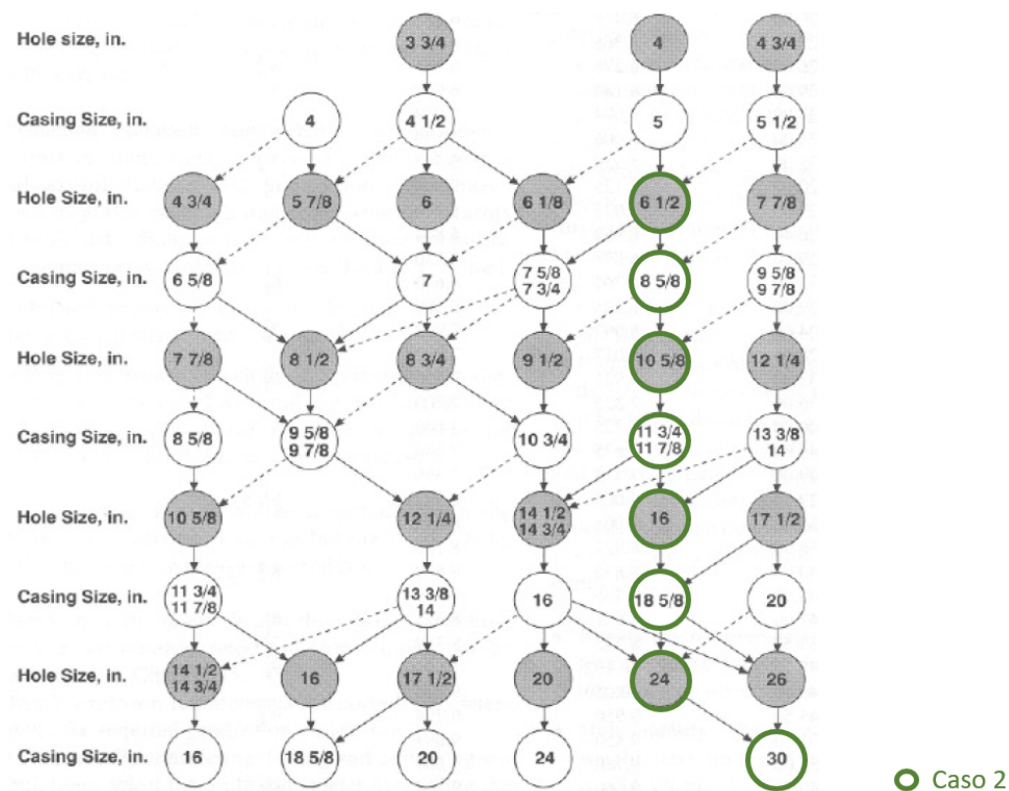


Figura 3.2: Sequência de diâmetros de revestimento e de poço aberto - Caso 2

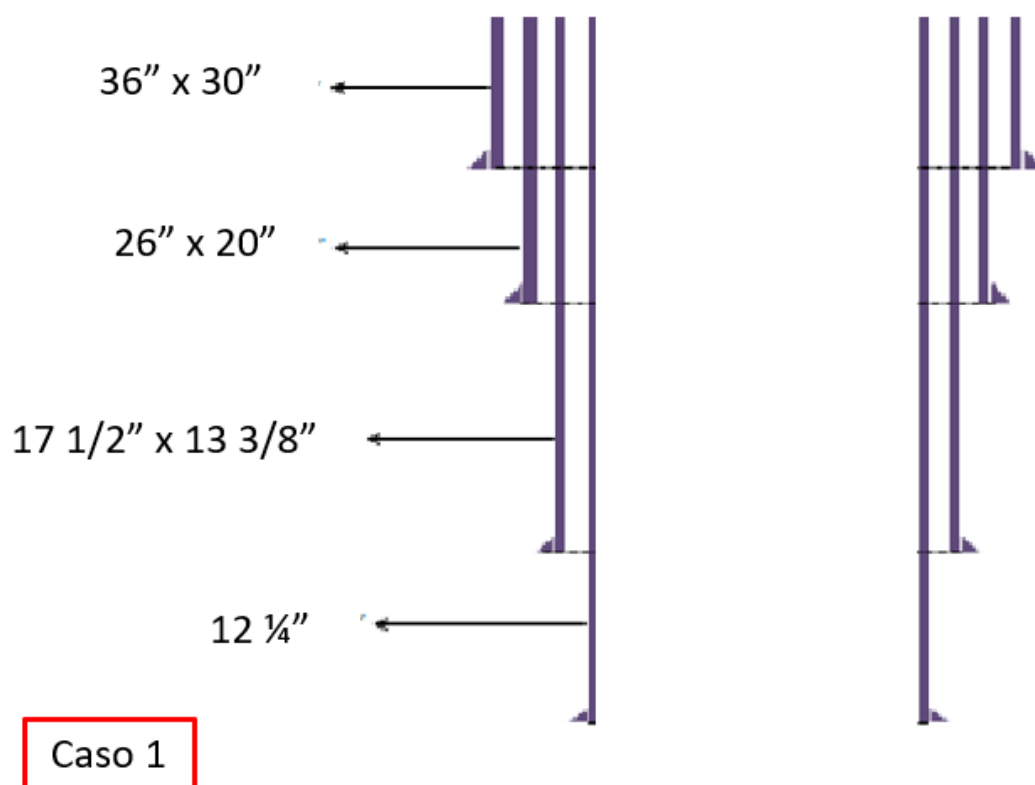


Figura 3.3: Geometria de Poço - Caso 1

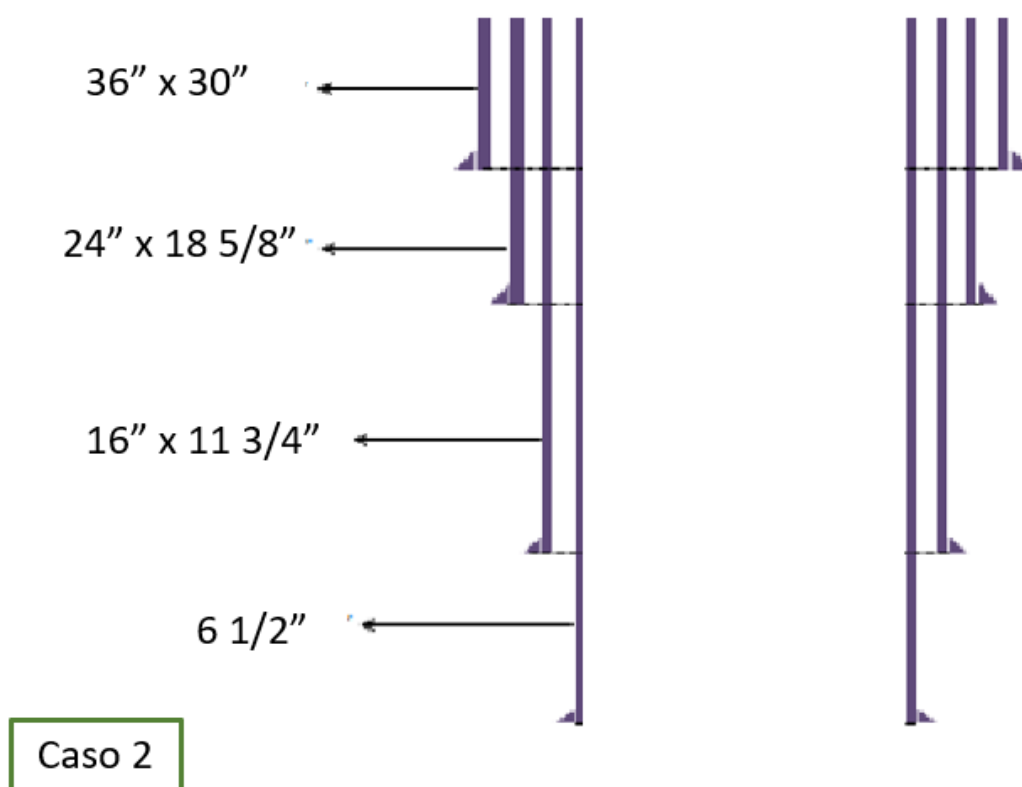


Figura 3.4: Geometria de Poço - Caso 2

Algumas simplificações serão utilizadas quanto à geometria do poço no desenvolvimento deste software:

1. Todos os poços serão verticais;
2. Todos os poços serão perfurados em quatro fases;
3. Todas as colunas de perfuração devem possuir uma fração de comandos de perfuração (DC), a fim de fornecer peso à broca no fundo do poço;
4. Todas as fases serão revestidas a partir da cabeça de poço, até chegar à sapata do revestimento;
5. Os cálculos de perda de carga serão realizados sempre na profundidade total da sapata do revestimento;
6. Ao atingir a formação, todos os poços possuirão 100m de reservatório perfurado.

Diâmetros padrão de tubulares

Constam abaixo os dados dos tubulares usados na perfuração dos casos de geometria propostos. Os dados foram retirados de tabelas da indústria e serão utilizados nos cálculos de perda de carga.

1. Revestimentos:

Diâmetro do Poço (in)	Casing OD (in)	Casing ID (in)	Wt (lb/ft)
36	30	28 7/8	250
26	20	19 1/8	94
17 1/2	13 3/8	12 5/7	48
12 1/4	-		

Figura 3.5: Dados relativos aos revestimentos (*casings*)

2. Tubulares (*drill pipes* e *drill collars*):

	Grade	OD (in)	ID (in)	Wall Thickness (in)	Wt (lb/ft)
DP	E-75	5 1/2	4 8/9	0,304	19,2
DC	6 5/8 REG	8	2 13/16	5,19	150

Figura 3.6: Dados relativos aos tubulares (*drill pipes* e *drill collars*)

3.2.2 Modelo reológico

Um modelo reológico é uma descrição matemática das forças viscosas experimentadas por um fluido e que são requeridas para o desenvolvimento das equações de perda de

carga. O modelo adotado nesse projeto de engenharia será o Newtoniano. Este modelo reológico é ideal para líquidos simples, como fluidos base-água, água salobra e fluidos base-óleo. Para um fluido Newtoniano, as forças viscosas presentes no fluido são uma constante. Esta constante é a viscosidade do fluido, cujo valor é determinado utilizando um reômetro ou um viscosímetro.

3.2.3 Formulação Físico-Matemática

Velocidade média:

Para o cálculo de perda de carga, é necessária a determinação das velocidades médias nas tubulações e no espaço anular. Para tal, são utilizadas a Equação 3.1 para a velocidade nas tubulações e a Equação 3.2 para a velocidade no anular.

$$v_t = \frac{q}{2,448(d^2)} \quad (3.1)$$

$$v_a = \frac{q}{2,448(d_2^2 - d_1^2)} \quad (3.2)$$

Onde:

q = vazão do fluido (galões por minuto - gpm);

v_t = velocidade média na tubulação (ft/s);

v_a = velocidade média no anular (ft/s);

d = diâmetro interno do tubo de perfuração (in);

d_2 = diâmetro do poço aberto ou diâmetro interno do revestimento (in);

d_1 = diâmetro externo do tubo de perfuração (in).

Regimes de fluxo

Os regimes de fluxo dizem respeito, em mecânica dos fluidos, a como os fluidos se comportam em relação a diversas variáveis. Quanto à direção da trajetória das partículas que o compõe, em relação a dependência do estado de organização do escoamento, os regimes podem ser divididos em:

Laminar - As partículas do fluido tendem a percorrer trajetórias paralelas em camadas (lâminas) bem definidas.

Turbulento - As trajetórias das partículas são curvilíneas, não paralelas e alteram-se em sentido, sendo irregulares. Apresentam um padrão de fluxo caótico, formando uma série de minúsculos redemoinhos.

O regime de fluxo é determinado fisicamente pelo número de Reynolds. A partir da comparação com o número de Reynolds crítico ($N_{Rec} = 2100$) encontrado na literatura, o

escoamento pode ser classificado como laminar, se $N_{Re} < N_{Rec}$, ou turbulento, se $N_{Re} > N_{Rec}$.

A Equação 3.3 é utilizada para determinar o número de Reynolds dentro de uma tubulação e a Equação 3.4 é utilizada para determinar o número de Reynolds no anular (entre a coluna de perfuração e o poço aberto/revestido).

$$N_{Re,t} = \frac{928\rho v d}{\mu} \quad (3.3)$$

$$N_{Re,a} = \frac{757\rho v (d_2 - d_1)}{\mu} \quad (3.4)$$

Onde:

$N_{Re,t}$ = número de Reynolds na tubulação;

$N_{Re,a}$ = número de Reynolds no anular;

ρ = densidade do fluido (*pounds per gallon* - ppg);

v = velocidade média (ft/s);

μ = viscosidade do fluido (cp);

d = diâmetro interno do tubo de perfuração (in);

d_2 = diâmetro do poço aberto ou diâmetro interno do revestimento (in);

d_1 = diâmetro externo das tubulações (in);

- Hipóteses simplificadoras

Se a vazão da bomba de lama for baixa o suficiente para que o fluxo seja laminar, o modelo Newtoniano pode ser aplicado para desenvolver as razões matemáticas entre vazão e queda de pressão. Neste desenvolvimento, algumas premissas simplificadoras são feitas para o fluxo laminar:

1. A coluna de perfuração é concêntrica ao revestimento ou poço aberto;
2. A coluna de perfuração não está sendo rotacionada;
3. As seções de poço aberto são circulares e de diâmetro conhecido;
4. O fluido de perfuração é incompressível;
5. O fluxo é isotérmico.

Na realidade, nenhuma dessas premissas é complementamente válida e o sistema de equações resultante não irá descrever perfeitamente o fluxo laminar do fluido de perfuração no poço. Entretanto, para a primeira versão do *software* e seu uso em sala de aula, considera-se suficiente.

Perda de carga

As fórmulas utilizadas para a determinação da perda de carga são dependentes do regime de fluxo presente na tubulação/anular. Dessa forma, é necessário determinar o tipo de fluxo previamente à determinação da perda de carga através das Equações 3.3 e 3.4.

Para o fluxo laminar, usa-se a Equação 3.5 para o cálculo de perda de carga nos tubos de perfuração e a Equação 3.6 para o cálculo de perda de carga no anular.

$$P_{c,t}^{Lam} = \frac{dP_f}{dL} = \frac{\mu v}{1500d^2} \quad (3.5)$$

$$P_{c,a}^{Lam} = \frac{dP_f}{dL} = \frac{\mu v}{1000(d_2 - d_1)^2} \quad (3.6)$$

Já para o fluxo turbulento, a Equação 3.7 para o cálculo de perda de carga nos tubos de perfuração e a Equação 3.8 para o cálculo de perda de carga no anular.

$$P_{c,t}^{Turb} = \frac{dP_f}{dL} = \frac{\rho^{0,75} v^{1,75} \mu^{0,25}}{1800d^{1,25}} \quad (3.7)$$

$$P_{c,a}^{Turb} = \frac{dP_f}{dL} = \frac{\rho^{0,75} v^{1,75} \mu^{0,25}}{1396(d_2 - d_1)^{1,25}} \quad (3.8)$$

Perda de carga total do poço

A perda de carga total do poço será igual à soma da perda de carga devido a cada seção do poço, considerando os diferentes diâmetros internos e externos dos tubos de perfuração (DP e DC) e diferentes diâmetros internos dos revestimentos, tal como exemplificado na Figura 3.7.

Sendo assim, a perda de carga (ΔP) se dará por:

$$\Delta P = \Delta P_1 + \Delta P_2 + \Delta P_3 + \Delta P_4 + \Delta P_5$$

Onde:

ΔP = Perda de carga total no poço

ΔP_1 = Perda de carga no interior dos *drill pipes* (DP)

ΔP_2 = Perda de carga no interior dos *drill collars* (DC)

ΔP_3 = Perda de carga entre o poço aberto e os *drill collars* (DC)

ΔP_4 = Perda de carga entre o poço aberto e os *drill pipes* (DP)

ΔP_5 = Perda de carga entre o revestimento e os *drill pipes* (DP)

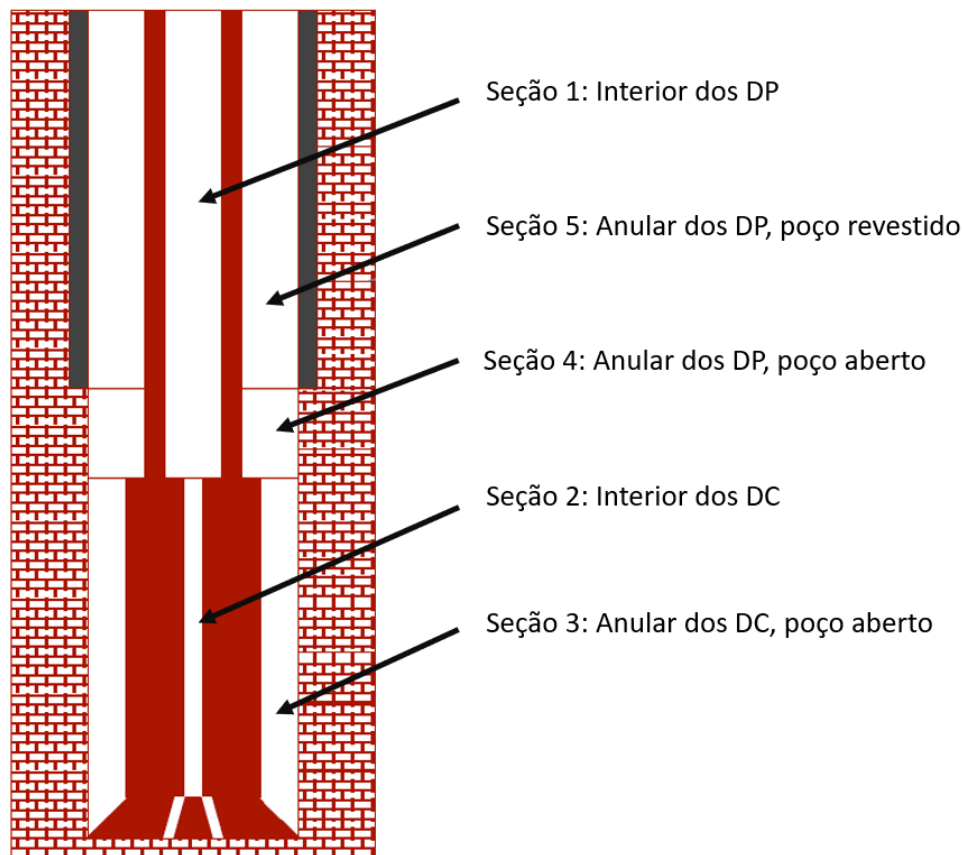


Figura 3.7: Perda de carga no poço por seção

3.3 Identificação de pacotes – assuntos

Definidos os conceitos físicos e matemáticos da Engenharia de Poço e Mecânica de Fluidos relacionados ao projeto, pode-se explicitar o conjunto de assuntos (ou pacotes) que serão desenvolvidos no software:

- Fluido: representa as características do fluido de perfuração sendo usado. Contém informações de viscosidade e densidade do mesmo. Usado para determinar o regime de fluxo e as perdas de carga do sistema.
- Geometria do poço: contém as informações geométricas referentes ao design do poço. Pode ser entendido como um conjunto de cilindros concêntricos, cada um com raios diferentes. Essas informações são necessárias para o cálculo das velocidades nos tubulares e no espaço anular, além de determinar quais fórmulas de perda de carga serão aplicadas.
- Perda de carga: representa a perda de carga que ocorre no sistema. Pode ser calculada por fase ou para todo o poço.
- Velocidades de fluxo: representa a velocidade com a qual o fluxo ocorre dentro das tubulações e nos espaços anulares. Se baseia na geometria do poço para a sua

determinação.

- Regime de fluxo: representa a forma como o fluxo está ocorrendo dentro da coluna de perfuração/poço/revestimento. Através do regime de fluxo, é possível determinar a forma como a perda de carga ocorre no sistema.
- Gráficos: trata-se de uma forma de representação dos resultados obtidos pelo software, onde as grandezas profundidade (metros) e perda de carga (psi) são plotadas.
- Resultados: representa as formas de visualização e exportação dos dados calculados pelo software.
- Simulador: é o gerenciador do sistema.

3.4 Diagrama de pacotes – assuntos

A Figura 3.8 representa o diagrama de pacotes para o software SHPP.

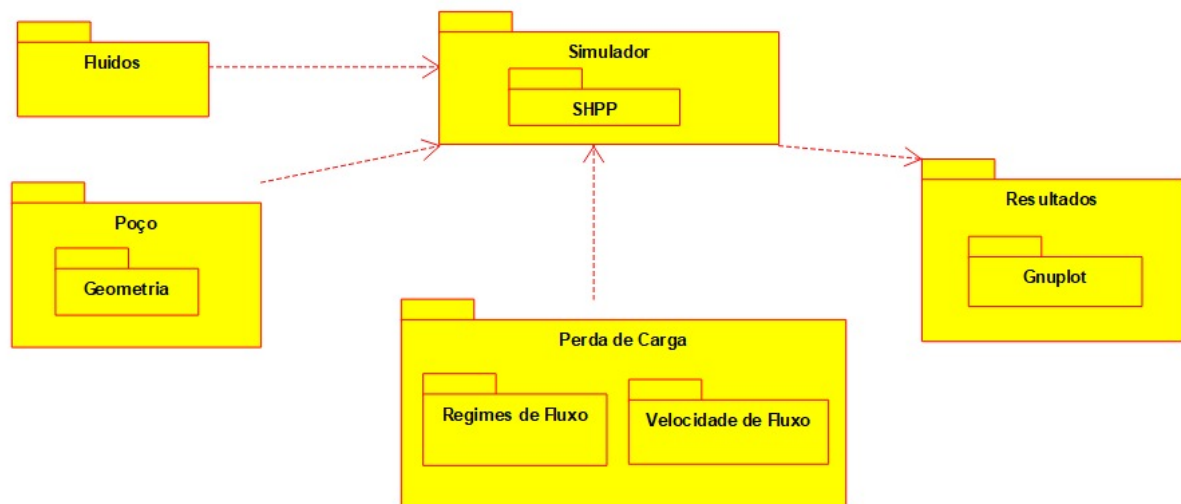


Figura 3.8: Diagrama de Pacotes para o *software* SHPP

Capítulo 4

AOO – Análise Orientada a Objeto

A terceira etapa do desenvolvimento de um projeto de engenharia, no nosso caso um software de cálculo de perda de carga é a AOO – Análise Orientada a Objeto. A AOO utiliza algumas regras para identificar os objetos de interesse, as relações entre os pacotes, as classes, os atributos, os métodos, as heranças, as associações, as agregações, as composições e as dependências.

4.1 Diagramas de classes

O diagrama de classes é apresentado na Figura 4.1.

4.1.1 Dicionário de classes

- Classe CFluido: é a classe responsável por receber as propriedades do fluido de perfuração do usuário.
- Classe CGeometriaPoco: é a classe que define a geometria do poço e das tubulações nas quais serão analisadas as perdas de carga.
- Classe CPerdaDeCarga: é a classe responsável pelo cálculo da perda de carga que ocorre no sistema.
- Classe CResultados: classe que fornece as opções de visualização e exportação dos dados calculados pelo software.
- Classe CSHPP: é a classe responsável por interconectar as demais classes e simular a perda de carga no poço.
- O programa externo Gnuplot é responsável pela geração dos gráficos e exportação de imagens.

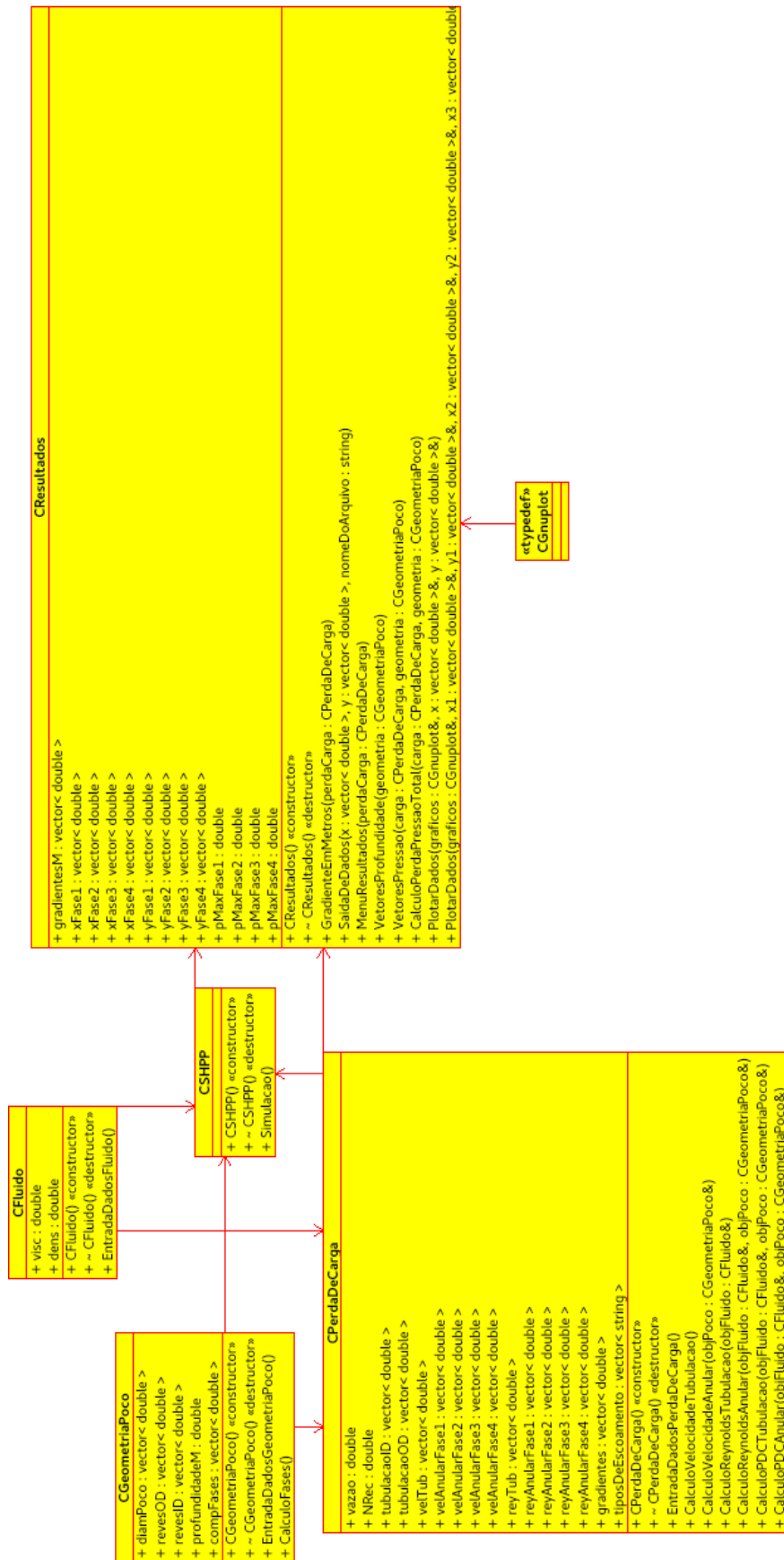


Figura 4.1: Diagrama de classes

4.2 Diagrama de sequência – eventos e mensagens

4.2.1 Diagrama de sequência geral

Veja o diagrama de sequência na Figura 4.2.

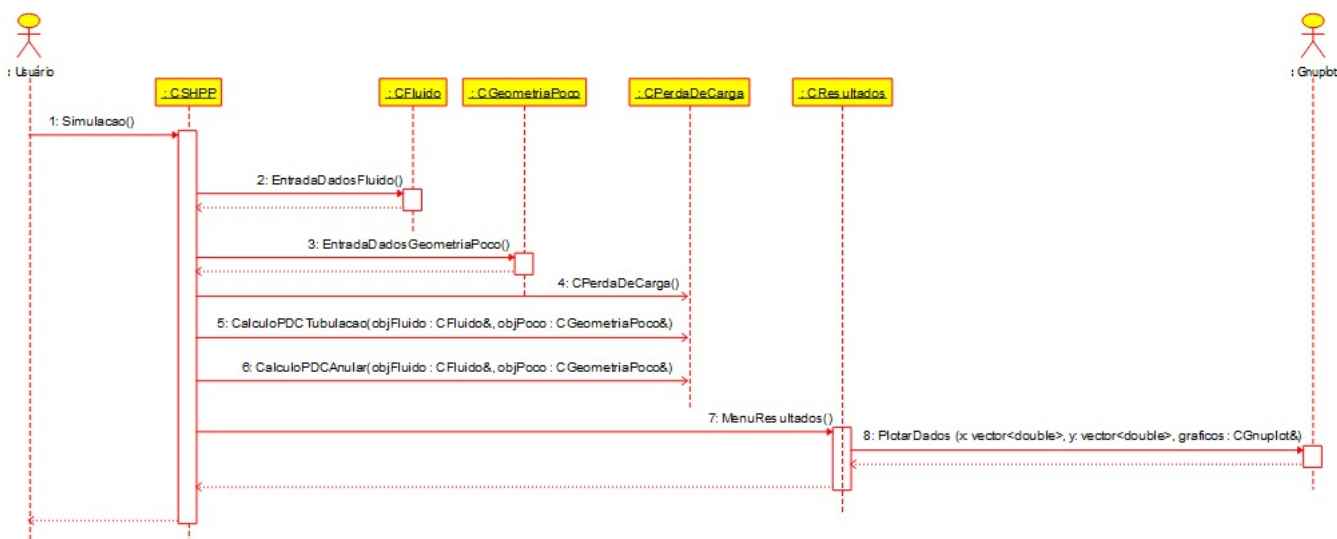


Figura 4.2: Diagrama de sequência

4.3 Diagrama de comunicação

A Figura 4.3 apresenta o diagrama de comunicação mostrando a sequência de cálculo de perda de carga e plotagem dos resultados em gráfico.

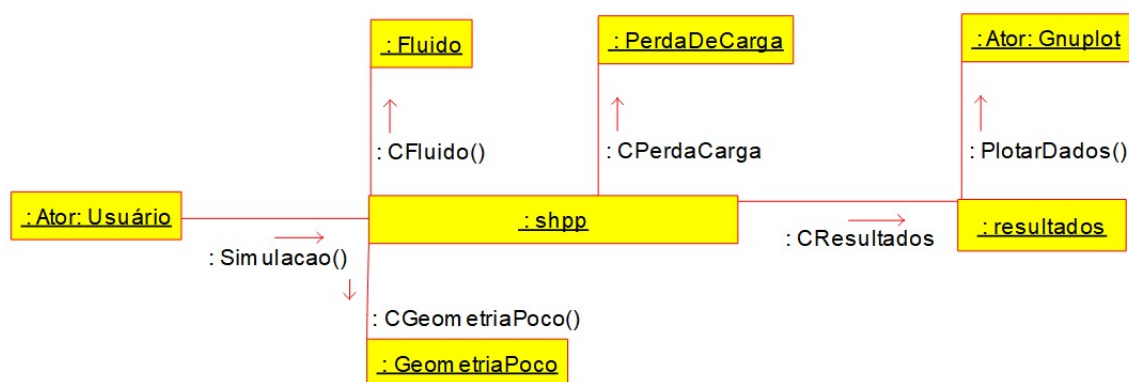


Figura 4.3: Diagrama de comunicação

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. É

usado para modelar aspectos dinâmicos do objeto.

Veja na Figura 4.4 o diagrama de máquina de estado para um objeto da classe CPerdaDeCarga.

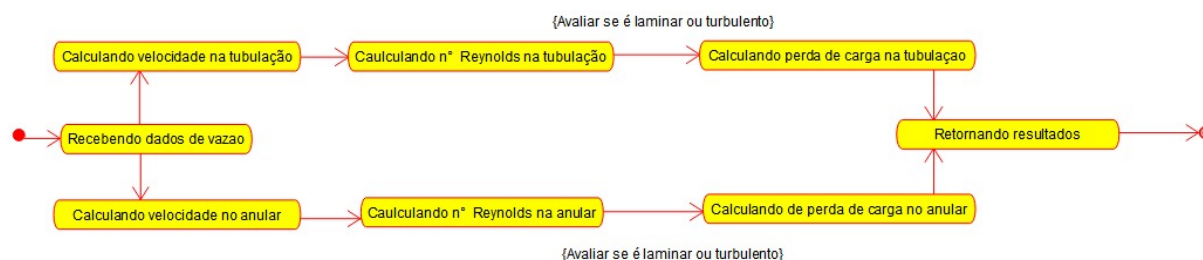


Figura 4.4: Diagrama de máquina de estado da classe CPerdaDeCarga

4.5 Diagrama de atividades

A Figura 4.5 representa o diagrama de atividades correspondente ao cálculo da perda de carga no anular. Observe que o método analisa o valor do número de Reynolds para verificar qual equação de perda de carga utilizar.

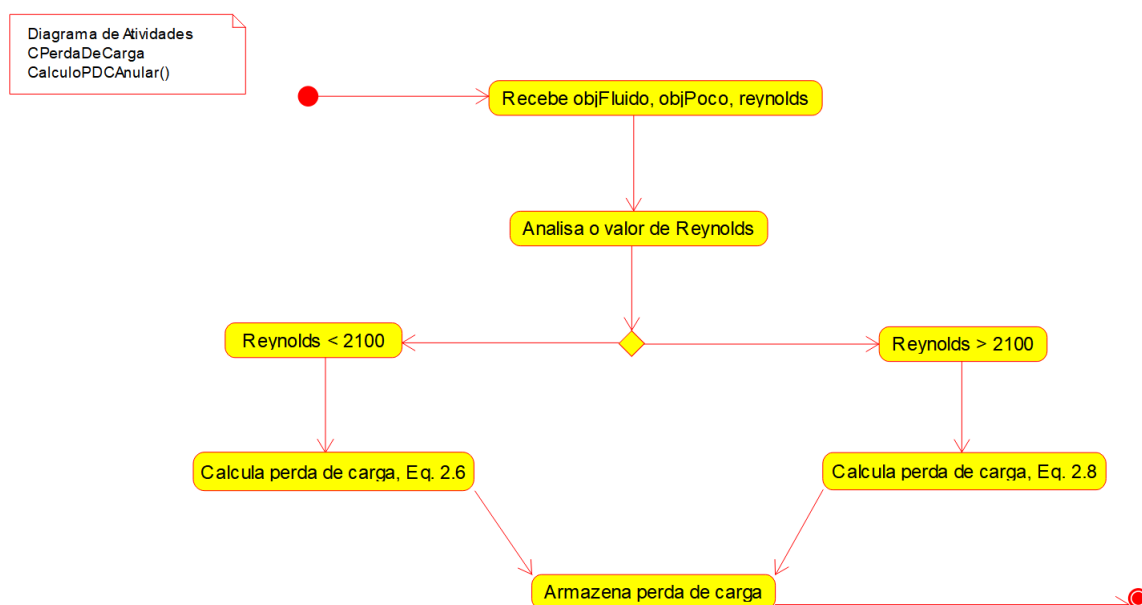


Figura 4.5: Diagrama de atividades da classe CPerdaDeCarga, método CalculoPDCAnular()

Capítulo 5

Projeto

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

5.1 Projeto do sistema

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

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

1. Plataformas

- O software irá funcionar no sistema operacional GNU/Linux, sendo desenvolvido no Windows e testado no GNU/Linux.
- O programa é multiplataforma, visto que a linguagem C++ possui suporte em diversos sistemas operacionais.

2. Bibliotecas

- Neste projeto será utilizada a biblioteca padrão da linguagem C++.
- O programa será desenvolvido por meio da interface Dev C++ e Visual Studio (Windows).

3. Arquivos externos

- O software SHPP gera arquivos no formato .txt e .png para apresentação dos resultados de perda de carga calculados por fase.

4. Recursos

- O programa utiliza o HD, o processador, o teclado, a memória, a tela e os demais componentes internos do computador.
- Será utilizado um arquivo de dados no formato .txt para leitura das informações de geometria de poço.

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 à análise desenvolvida as características da plataforma escolhida (hardware, sistema operacional e linguagem de softwareção). Passa pelo maior detalhamento do funcionamento do software, acrescentando atributos e métodos que envolvem a solução de problemas específicos não identificados durante a análise.

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

Efeitos do projeto no modelo estrutural

- Dependências e restrições do SHPP:
 - O software SHPP deve ser executado na plataforma GNU/Linux.
 - O programa depende da instalação do software externo Gnuplot.
 - A biblioteca gráfica CGnuplot é utilizada.

Efeitos do projeto no modelo dinâmico

- Essa etapa não envolveu mudanças no projeto, visto que os diagramas de sequência, comunicação, máquina de estado e atividades foram modificados durante o desenvolvimento do código.

Efeitos do projeto nos atributos

- Atributos para leitura de dados do disco foram implementados (arquivos .txt).

Efeitos do projeto nos métodos

- A classe externa Gnuplot permitiu o uso de uma função para salvar os gráficos gerados em formato png.
- Além da leitura de disco, o método de inserção de dados pelo usuário por meio do teclado foi implementado.

Efeitos do projeto nas heranças

- Essa etapa não envolveu mudanças no projeto.

Efeitos do projeto nas associações

- Essa etapa não envolveu mudanças no projeto.

Efeitos do projeto nas otimizações

- Após os primeiros testes, uma nova forma de representação dos resultados foi criada. O novo método de saída de dados consistiu em plotar a perda de carga para todas as fases em um mesmo gráfico. Esse gráfico permitiu comparar os valores de perda de carga correspondentes a cada fase.

5.3 Diagrama de componentes

O diagrama de componentes mostra a forma como os componentes do software se relacionam, suas dependências.

A Figura 5.1 apresenta o diagrama de componentes do software SHPP. O simulador acessa as bibliotecas C++. O subsistema biblioteca representa o simulador SHPP e contém os arquivos *header* e *implementation* das classes CFluido, CGeometriaPoco, CPerdaDeCarga e CResultados. O subsistema biblioteca inclui os arquivos das classes citadas, e a geração dos objetos fluido.obj, geometria.obj, perdaCarga.obj e resultados.obj depende dos arquivos CFluido.h, CFluido.cpp, CGeometriaPoco.h, CGeometriaPoco.cpp, CPerdaDeCarga.h, CPerdaDeCarga.cpp, CResultados.h e CResultados.cpp. O subsistema biblioteca Gnuplot, um subsistema externo para geração de gráficos, inclui os arquivos de código da biblioteca CGnuplot e a biblioteca em si. O software executável a ser gerado depende da biblioteca gerada, dos arquivos da biblioteca CGnuplot e do banco de dados.

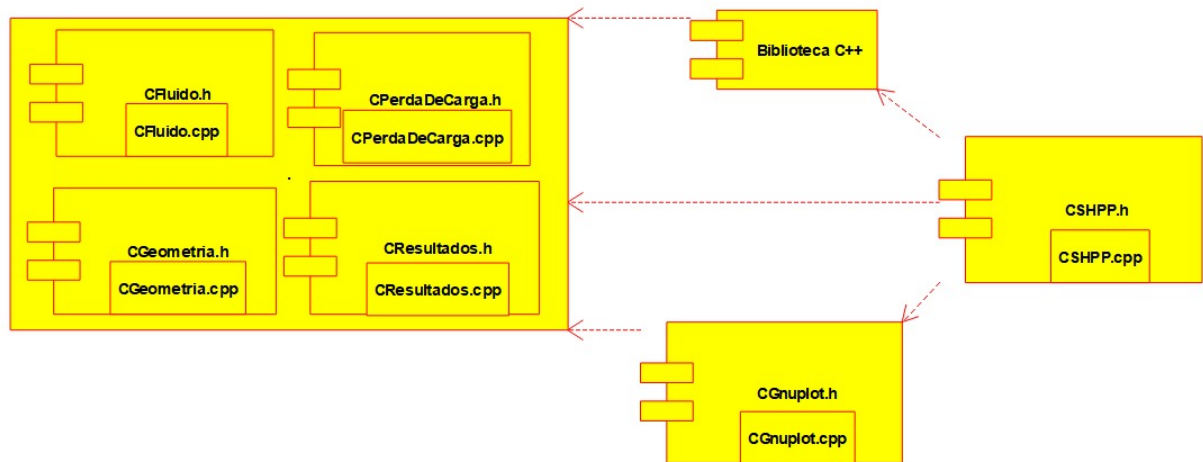


Figura 5.1: Diagrama de componentes

5.4 Diagrama de implantação

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

Veja na Figura 5.2 o diagrama de implantação do software SHPP. O Simulador acessa os arquivos de dados no disco rígido, contendo as informações de geometria do poço. Os insumos e resultados são dispostos ao usuário por meio do teclado e monitor. Os resultados das simulações são armazenados no disco rígido.

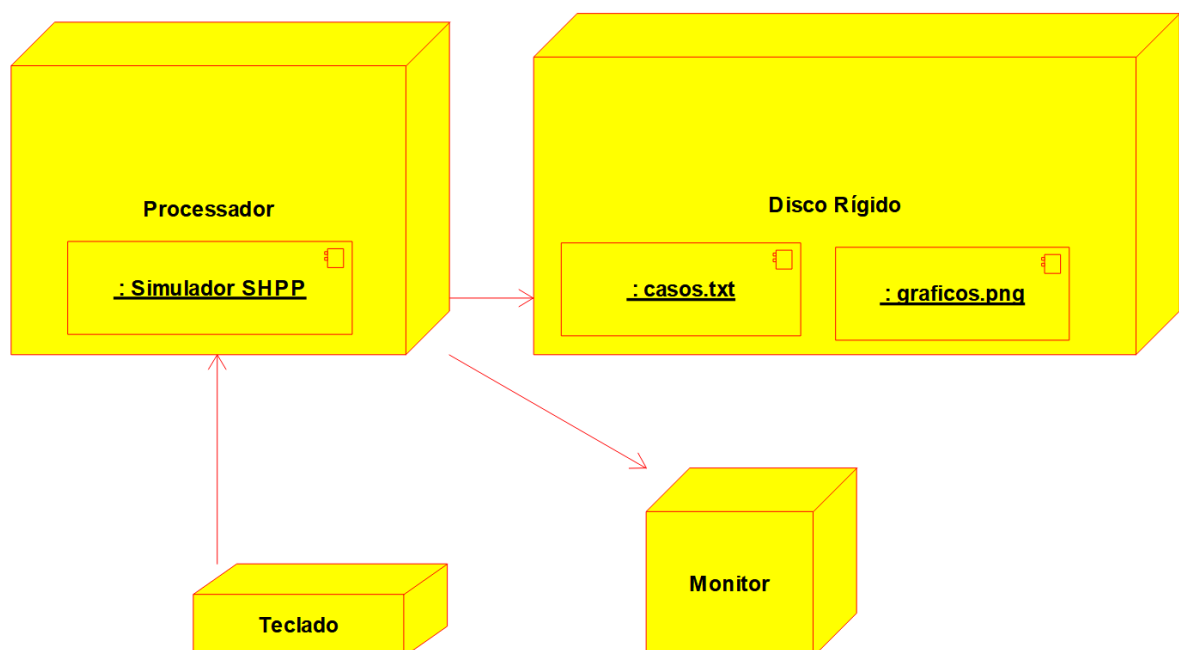


Figura 5.2: Diagrama de implantação.

Capítulo 6

Implementação

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

6.1 Código fonte

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

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

Listing 6.1: Arquivo de cabeçalho da classe CFluido.

```
1 /**
2  @autor Gleison Monteiro / Luisa Campbell
3  @file CFluido.h
4  @brief CFluido : E uma classe que representa o fluido usado na
      perfuracao do poço.
5  */
6
7 #ifndef CFluido_H
8 #define CFluido_H
9
10 //Inclusao de classes das bibliotecas basicas
11 #include <string>
12 #include <iostream>
13
14 using namespace std;
15
16 class CFluido {
17
18 public:
19     //ATRIBUTOS
20     double visc = 25; /// Viscosidade do fluido. OBS: Fluido
      polimerico
```

```

21     double dens = 10; /// Densidade do fluido. OBS: Fluido
        polimerico
22
23     //METODOS
24     CFluido(){} // Construtor Default
25     virtual ~CFluido(){} // Destrutor Default
26     void EntradaDadosFluido(); /// Metodo que permite a entrada de
        dados do fluido pelo usuario
27
28 };
29
30 #endif

```

Apresenta-se na listagem 6.2 o arquivo com código da classe CFluido.

Listing 6.2: Arquivo de implementação da classe CFluido.

```

31 /**
32  @autor Gleison Monteiro | Luisa Campbell
33  @file CFluido.cpp
34  @brief CFluido.cpp : Implementa os metodos da classe CFluido.h
35  */
36
37 // Inclusao de classes e bibliotecas
38
39 #include "CFluido.h" //Inclui o arquivo de cabecalho da classe CFluido
40
41 void CFluido::EntradaDadosFluido(){
42
43     bool flag = false;
44
45     cout << "\n Os valores padrao do fluido de perfuracao
        polimerico sao: viscosidade= 25cp (centipoise) e densidade=
        10ppg (ponds per gallon, ou libras por galoes). \n Deseja
        alterar as propriedades do fluido? \n Se sim, digite 's'; se
        nao, digite 'n'." << endl;
46     string resposta;
47     ///O loop ira garantir a entrada de uma opcao valida pelo
        usuario
48     while (!flag){
49
50         cin >> resposta;
51         cin.get();
52
53         if (resposta == "s"){ ///Nesta opcao, o usuario ira
            entrar manualmente com os valores de densidade e
            viscosidade do fluido
54             double nVisc, nDens;
55             cout << "\t Entre o novo valor da viscosidade (
                em cp):" << endl;

```

```

56         cin >> nVisc;
57         cin.get();
58         visc = nVisc;
59         cout << "\n";
60         cout << "\tEntre o novo valor da densidade (em
        ppg):" << endl;
61         cin >> nDens;
62         cin.get();
63         dens = nDens;
64         cout << "\n";
65         flag = true;
66     }
67     else if (resposta == "n"){    ///Nesta opcao, o usuario
        escolhe manter os valores de densidade e viscosidade
        do fluido sugeridos
68         cout << "Os dados serao mantidos." << endl;
69         cout << "\n";
70         flag = true;
71     }
72     else{    ///Esta opcao ira detectar erros de
        possiveis entradas do usuario
73         cout << "Voce nao entrou um valor valido.\n
        Tente novamente.\n" << endl;
74     }
75
76 }
77 };

```

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

Listing 6.3: Arquivo de cabeçalho da classe CGeometriaPoco.

```

78 /**
79 @autor Gleison Monteiro / Luisa Campbell
80 @file CGeometriaPoco.h
81 @brief CGeometriaPoco : E uma classe que representa a geometria do poco.
82 */
83
84 #ifndef CGeometriaPoco_H
85 #define CGeometriaPoco_H
86
87 /// Inclusao de classes das bibliotecas basicas
88 #include <string>
89 #include <iostream>
90 #include <fstream>
91 #include <vector>
92
93 using namespace std;
94
95 class CGeometriaPoco {

```

```

96
97 public:
98
99     //ATRIBUTOS
100     vector<double> diamPoco;    ///vetor representando os diâmetros de
        poco aberto
101     vector<double> revesOD;    ///vetor representando os diâmetros
        externos dos revestimentos
102     vector<double> revesID;    ///vetor representando os diâmetros
        internos dos revestimentos
103     double profundidadeM;      ///profundidade total do poco em
        metros
104     //double profundidadeFt;    ///profundidade total do poco em pes
105     vector<double> compFases;
106
107     //METODOS
108
109     CGeometriaPoco(){}          //Construtor Default
110     virtual ~CGeometriaPoco(){} //Destrutor Default
111
112     void EntradaDadosGeometriaPoco(); ///Metodo que permite que o
        usuario entre com dados de geometria de poco lendo de um
        arquivo texto
113     void CalculoFases();        ///Determina o comprimento de
        cada fase
114
115 };
116
117 #endif

```

Apresenta-se na listagem 6.4 o arquivo com código da classe CGeometriaPoco.

Listing 6.4: Arquivo de implementação da classe CGeometriaPoco.

```

118 /**
119  @autor Gleison Monteiro / Luisa Campbell
120  @file CGeometriaPoco.cpp
121  @brief CGeometriaPoco.cpp : Implementa os metodos da classe
        CGeometriaPoco
122 */
123
124 // Inclusao de classes e bibliotecas
125 #include <fstream>
126 #include <string>
127
128 #include "CGeometriaPoco.h"    //Inclui o arquivo de cabecalho da classe
        CGeometriaPoco
129
130 using namespace std;
131

```

```

132 void CGeometriaPoco::EntradaDadosGeometriaPoco(){
133
134     bool flag = false;
135     cout << "Entre com o nome do arquivo texto de onde deseja ler os
        dados da geometria do poco - diametro do poco (in), diametro
        externo do revestimento (in) e diametro interno do
        revestimento (in):" << endl;
136     string nomeArquivo;
137
138     ///0 loop ira garantir que o usuario entre com um nome de
        arquivo existente no diretorio local
139     while (!flag){
140
141         cin >> nomeArquivo;
142         cin.get();
143         nomeArquivo.append(".txt");
144
145         ifstream fin(nomeArquivo);
146
147
148         if (!fin.is_open()){ ///Nesta opcao, o usuario entrou
        com um nome de arquivo invalido e tera que repetir a
        entrada
149             cout << "Erro: Arquivo nao encontrado." << endl;
150             cout << "Entre novamente o nome do arquivo." <<
                endl;
151         }
152
153         else { ///Nesta opcao, os dados serao lidos e
        armazenados
154
155             string cabecalho;
156             getline(fin, cabecalho);
157
158             while (!fin.eof()) {
159                 string _diamPoco, _revesOD, _revesID;
160
161                 fin >> _diamPoco;
162                 fin >> _revesOD;
163                 fin >> _revesID;
164
165                 diamPoco.push_back(stod(_diamPoco));
166                 revesOD.push_back(stod(_revesOD));
167                 revesID.push_back(stod(_revesID));
168             }
169
170
171             fin.close();

```

```

172             flag = true;
173         }
174
175     }
176
177     cout << "␣\n";
178 };
179
180
181 void CGeometriaPoco::CalculoFases(){
182
183     cout << "Entre␣com␣a␣profundidade␣total␣do␣poco,␣em␣metros:" <<
        endl;
184     cin >> profundidadeM;
185     cin.get();
186     cout << "␣\n";
187
188     //0 comprimento de cada fase foi baseado em um historico de
        perfuracao dos pocos do pre sal
189
190     compFases.push_back(profundidadeM*0.02); // Determinar a
        profundidade da fase 1 (superficie)
191     compFases.push_back(profundidadeM*0.38); // Determinar a
        profundidade da fase 2 (intermediaria 1)
192     compFases.push_back(profundidadeM*0.87); // Determinar a
        profundidade da fase 3 (intermediaria 2)
193     compFases.push_back(profundidadeM); // Determinar a profundidade
        da fase 4 (reservatorio)
194
195 };

```

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

Listing 6.5: Arquivo de cabeçalho da classe CPerdaDeCarga.

```

196 /**
197  @autor Gleison Monteiro / Luisa Campbell
198  @file CPerdaDeCarga.h
199  @brief CPerdaDeCarga.h : E uma classe que determina a perda de carga no
        poco.
200  */
201
202 #ifndef CPerdaDeCarga_H
203 #define CPerdaDeCarga_H
204
205 //Inclusao de classes das bibliotecas basicas
206 #include <string>
207 #include <iostream>
208 #include <fstream>
209 #include <cmath>

```



```

210 #include <vector>
211
212 #include "CFluido.h"           //Inclui o arquivo de cabecalho da classe
                                CFluido
213 #include "CGeometriaPoco.h"    //Inclui o arquivo de cabecalho da classe
                                CGeometriaPoco
214
215 using namespace std;
216
217 class CPerdaDeCarga {
218
219 public:
220
221     //ATRIBUTOS
222     double vazao;               ///representa a vazao da bomba
223     double NRec = 2100;        ///representa o numero de reynolds critico
224
225     // Dados relacionados as tubulacoes
226     vector<double> tubulacaoID; ///vetor com os diametros internos
                                das tubulacoes
227     vector<double> tubulacaoOD; ///vetor com os diametros externos
                                das tubulacoes
228
229     // Vetores para calculos de velocidade
230     vector<double> velTub;      ///vetor que armazena as
                                velocidades calculadas dentro da tubulacao
231     vector<double> velAnularFase1; ///vetor que armazena as
                                velocidades na espaco anular da fase 1
232     vector<double> velAnularFase2; ///vetor que armazena as
                                velocidades na espaco anular da fase 2
233     vector<double> velAnularFase3; ///vetor que armazena as
                                velocidades na espaco anular da fase 3
234     vector<double> velAnularFase4; ///vetor que armazena as
                                velocidades na espaco anular da fase 4
235
236     // Vetores para calculo do Numero de Reynolds
237     vector<double> reyTub;      ///vetor que armazena os numeros
                                de Reynolds dentro da tubulacao
238     vector<double> reyAnularFase1; ///vetor que armazena os numeros
                                de Reynolds no anular da fase 1
239     vector<double> reyAnularFase2; ///vetor que armazena os numeros
                                de Reynolds no anular da fase 2
240     vector<double> reyAnularFase3; ///vetor que armazena os numeros
                                de Reynolds no anular da fase 3
241     vector<double> reyAnularFase4; ///vetor que armazena os numeros
                                de Reynolds no anular da fase 4
242
243     vector<double> gradientes;  ///vetor que armazena os gradientes

```

```

        de perda de carga no poço
244     vector<string> tiposDeEscoamento;   ///vetor que armazena o tipo
        de escoamento nos intervalos
245
246     ///MÉTODOS
247     CPerdaDeCarga(){}                    ///Construtor Default
248     virtual ~CPerdaDeCarga(){}          ///Destrutor Default
249
250     void EntradaDadosPerdaDeCarga();      ///Método que permite a
        entrada de parâmetros da operação pelo usuário
251     void CalculoVelocidadeTubulacao();    ///Calcula as velocidades
        dentro da tubulação
252     void CalculoVelocidadeAnular(CGeometriaPoco& objPoco);    ///Calcula as velocidades no espaço anular
253     void CalculoReynoldsTubulacao(CFluido& objFluido);        ///Calcula os números de Reynolds na tubulação
254     void CalculoReynoldsAnular(CFluido& objFluido, CGeometriaPoco&
        objPoco);    ///Calcula os números de Reynolds no anular
255     void CalculoPDCTubulacao(CFluido& objFluido, CGeometriaPoco&
        objPoco);    ///Calcula o gradiente de perda de carga na
        tubulação considerando o tipo de regime
256     void CalculoPDCAnular(CFluido& objFluido, CGeometriaPoco&
        objPoco);    ///Calcula o gradiente de perda de carga no
        espaço anular considerando o tipo de regime
257
258 };
259
260 #endif

```

Apresenta-se na listagem 6.6 o arquivo com código da classe CPerdaDeCarga.

Listing 6.6: Arquivo de implementação da classe CPerdaDeCarga.

```

261 /**
262  @autor Gleison Monteiro | Luisa Campbell
263  @file CPerdaDeCarga.cpp
264  @brief CPerdaDeCarga.cpp : Implementa os métodos da classe CPerdaDeCarga
        .h
265 */
266
267 ///Inclusão de classes e bibliotecas
268
269 #include "CPerdaDeCarga.h"    ///Inclui o arquivo de cabeçalho da classe
        CPerdaDeCarga.h
270
271 void CPerdaDeCarga::EntradaDadosPerdaDeCarga(){
272
273     cout << "Entre com a vazão de bombeio do fluido de perfuração(
        em galões por minuto - gpm):" << endl;
274     double _vazao;

```

```

275         cin >> _vazao;
276         cin.get();
277         vazao = _vazao;
278         cout << "└─\n";
279
280         //Entrada dos valores de OD e ID das tubulacoes no vetor
281         tubulacaoID.push_back(4.88);
282         tubulacaoID.push_back(2.81);
283         tubulacaoOD.push_back(8.0);
284         tubulacaoOD.push_back(5.5);
285
286     };
287
288     void CPerdaDeCarga::CalculoVelocidadeTubulacao(){
289         double velocidade;
290         velocidade = vazao / (2.448*tubulacaoID[0] * tubulacaoID[0]);
291         velTub.push_back(velocidade);
292         velocidade = vazao / (2.448*tubulacaoID[1] * tubulacaoID[1]);
293         velTub.push_back(velocidade);
294
295     };
296
297     void CPerdaDeCarga::CalculoVelocidadeAnular(CGeometriaPoco& objPoco){
298         double velocidade;
299         vector<double> ajudaVelocidade;
300
301         //Fase 1
302         velocidade = vazao / (2.448 * ((objPoco.diamPoco[0])*(objPoco.
            diamPoco[0]) - tubulacaoOD[0] * tubulacaoOD[0])); //Poco
            aberto e DC;
303         velAnularFase1.push_back(velocidade);
304         velocidade = vazao / (2.448 * ((objPoco.diamPoco[0])*(objPoco.
            diamPoco[0]) - tubulacaoOD[1] * tubulacaoOD[1])); //Poco
            aberto e DP;
305         velAnularFase1.push_back(velocidade);
306
307         for (int i = 0; i < 3; i++){
308
309             velocidade = vazao / (2.448 * ((objPoco.diamPoco[i + 1])
                *(objPoco.diamPoco[i + 1]) - tubulacaoOD[0]*
                tubulacaoOD[0])); //Poco aberto e DC;
310             ajudaVelocidade.push_back(velocidade);
311             velocidade = vazao / (2.448 * ((objPoco.diamPoco[i + 1])
                *(objPoco.diamPoco[i + 1]) - tubulacaoOD[1]*
                tubulacaoOD[1])); //Poco aberto e DP;
312             ajudaVelocidade.push_back(velocidade);
313             velocidade = vazao / (2.448 * ((objPoco.revesID[i])*(
                objPoco.revesID[i]) - tubulacaoOD[1]*tubulacaoOD[1]))

```

```

        ; //Poco revestido e DP;
314         ajudaVelocidade.push_back(velocidade);
315
316     }
317
318     //Alimentando os vetores de velocidade anular por fases
319     for (int j = 0; j < 3; j++){
320         velAnularFase2.push_back(ajudaVelocidade[j]);
321         velAnularFase3.push_back(ajudaVelocidade[3 + j]);
322         velAnularFase4.push_back(ajudaVelocidade[6 + j]);
323     }
324
325 };
326
327 void CPerdaDeCarga::CalculoReynoldsTubulacao(CFluido& objFluido){
328
329     double reynolds;
330     reynolds = (928 * objFluido.dens * velTub[0] * tubulacaoID[0]) /
        objFluido.visc;
331     reyTub.push_back(reynolds);
332     reynolds = (928 * objFluido.dens * velTub[1] * tubulacaoID[1]) /
        objFluido.visc;
333     reyTub.push_back(reynolds);
334 };
335
336 void CPerdaDeCarga::CalculoReynoldsAnular(CFluido& objFluido,
        CGeometriaPoco& objPoco){
337
338     double reynolds;
339
340     //Fase 1
341     reynolds = (757 * objFluido.dens * velAnularFase1[0] * (objPoco.
        diamPoco[0] - tubulacaoOD[0])) / objFluido.visc; //Poco
        aberto e DC;
342     reyAnularFase1.push_back(reynolds);
343     reynolds = (757 * objFluido.dens * velAnularFase1[1] * (objPoco.
        diamPoco[0] - tubulacaoOD[1])) / objFluido.visc; //Poco
        aberto e DP;
344     reyAnularFase1.push_back(reynolds);
345
346     //Fase 2
347     reynolds = (757 * objFluido.dens * velAnularFase2[0] * (objPoco.
        diamPoco[1] - tubulacaoOD[0])) / objFluido.visc; //Poco
        aberto e DC;
348     reyAnularFase2.push_back(reynolds);
349     reynolds = (757 * objFluido.dens * velAnularFase2[1] * (objPoco.
        diamPoco[1] - tubulacaoOD[1])) / objFluido.visc; //Poco
        aberto e DP;

```

```

350     reyAnularFase2.push_back(reynolds);
351     reynolds = (757 * objFluido.dens * velAnularFase2[2] * (objPoco.
        revesID[0] - tubulacaoOD[1])) / objFluido.visc; // Poco
        revestido e DP;
352     reyAnularFase2.push_back(reynolds);
353
354     //Fase 3
355     reynolds = (757 * objFluido.dens * velAnularFase3[0] * (objPoco.
        diamPoco[2] - tubulacaoOD[0])) / objFluido.visc; // Poco
        aberto e DC;
356     reyAnularFase3.push_back(reynolds);
357     reynolds = (757 * objFluido.dens * velAnularFase3[1] * (objPoco.
        diamPoco[2] - tubulacaoOD[1])) / objFluido.visc; // Poco
        aberto e DP;
358     reyAnularFase3.push_back(reynolds);
359     reynolds = (757 * objFluido.dens * velAnularFase3[2] * (objPoco.
        revesID[1] - tubulacaoOD[1])) / objFluido.visc; // Poco
        revestido e DP;
360     reyAnularFase3.push_back(reynolds);
361
362     //Fase 4
363     reynolds = (757 * objFluido.dens * velAnularFase4[0] * (objPoco.
        diamPoco[3] - tubulacaoOD[0])) / objFluido.visc; // Poco
        aberto e DC;
364     reyAnularFase4.push_back(reynolds);
365     reynolds = (757 * objFluido.dens * velAnularFase4[1] * (objPoco.
        diamPoco[3] - tubulacaoOD[1])) / objFluido.visc; // Poco
        aberto e DP;
366     reyAnularFase4.push_back(reynolds);
367     reynolds = (757 * objFluido.dens * velAnularFase4[2] * (objPoco.
        revesID[2] - tubulacaoOD[1])) / objFluido.visc; // Poco
        revestido e DP;
368     reyAnularFase4.push_back(reynolds);
369
370 };
371
372 void CPerdaDeCarga::CalculoPDCTubulacao(CFluido& objFluido,
    CGeometriaPoco& objPoco) {
373     double _gradiente;
374
375     //Analisando a perda de carga no DP e DC
376
377     for (int i = 0; i < 2; i++){
378         if (reyTub[i] < 2100) {
379             _gradiente = (objFluido.visc * velTub[i]) /
                (1500 * tubulacaoID[i] * tubulacaoID[i]);
380             tiposDeEscoamento.push_back("Laminar");
381         }

```

```

382         else {
383             _gradiente = pow(objFluido.dens, 0.75) * pow(
384                 objFluido.visc, 0.25) * pow(velTub[i], 1.75)
385                 / (1800 * pow(tubulacaoID[i], 1.25));
386             tiposDeEscoamento.push_back("Turbulento");
387         }
388
389         gradientes.push_back(_gradiente);
390
391     };
392
393
394 void CPerdaDeCarga::CalculoPDCAnular(CFluido& objFluido, CGeometriaPoco&
395     objPoco) {
396     double _gradiente;
397
398     //FASE 1
399     for (int i = 0; i < 2; i++) {
400         if (reyAnularFase1[i] < 2100) {
401             _gradiente = (objFluido.visc * velAnularFase1[i]
402                 ) / (1000 * pow((objPoco.diamPoco[0] -
403                 tubulacaoOD[i]), 2)); //DC e DP
404             tiposDeEscoamento.push_back("Laminar");
405         }
406         else {
407             _gradiente = pow(objFluido.dens, 0.75) * pow(
408                 objFluido.visc, 0.25) * pow(velAnularFase1[i]
409                 , 1.75) / (1396 * pow((objPoco.diamPoco[0] -
410                 tubulacaoOD[i]), 1.25)); //DC e DP
411             tiposDeEscoamento.push_back("Turbulento");
412         }
413         gradientes.push_back(_gradiente);
414     }
415
416     //FASE 2
417
418     //DP e DC
419     for (int i = 0; i < 2; i++) {
420         if (reyAnularFase2[i] < 2100) {
421             _gradiente = (objFluido.visc * velAnularFase2[i]
422                 ) / (1000 * pow((objPoco.diamPoco[1] -
423                 tubulacaoOD[i]), 2)); //DC i=0 e DP i=1
424             tiposDeEscoamento.push_back("Laminar");

```

```

420     }
421     else {
422         _gradiente = pow(objFluido.dens, 0.75) * pow(
            objFluido.visc, 0.25) * pow(velAnularFase2[i
            ], 1.75) / (1396 * pow((objPoco.diamPoco[1] -
            tubulacaoOD[i]), 1.25));    //DC e DP
423         tiposDeEscoamento.push_back("Turbulento");
424     }
425     gradientes.push_back(_gradiente);
426
427 }
428
429 // Revestimento DP
430
431 if (reyAnularFase2[2] < 2100) {
432     _gradiente = (objFluido.visc * velAnularFase2[2]) /
        (1000 * pow((objPoco.revesID[0] - tubulacaoOD[1]), 2)
        );    //DC i=0 e DP i=1
433     tiposDeEscoamento.push_back("Laminar");
434     gradientes.push_back(_gradiente);
435 }
436 else {
437     _gradiente = pow(objFluido.dens, 0.75) * pow(objFluido.
        visc, 0.25) * pow(velAnularFase2[2], 1.75) / (1396 *
        pow((objPoco.revesID[0] - tubulacaoOD[1]), 1.25));
        //DC e DP
438     tiposDeEscoamento.push_back("Turbulento");
439     gradientes.push_back(_gradiente);
440 }
441
442
443 //CALCULOS FASE 3
444
445 //DP e DC
446 for (int i = 0; i < 2; i++) {
447
448     if (reyAnularFase3[i] < 2100) {
449         _gradiente = (objFluido.visc * velAnularFase3[i
            ]) / (1000 * pow((objPoco.diamPoco[2] -
            tubulacaoOD[i]), 2));    //DC i=0 e DP i=1
450         tiposDeEscoamento.push_back("Laminar");
451     }
452     else {
453         _gradiente = pow(objFluido.dens, 0.75) * pow(
            objFluido.visc, 0.25) * pow(velAnularFase3[i
            ], 1.75) / (1396 * pow((objPoco.diamPoco[2] -
            tubulacaoOD[i]), 1.25));    //DC e DP
454         tiposDeEscoamento.push_back("Turbulento");

```

```

455         }
456         gradientes.push_back(_gradiente);
457
458     }
459
460     // Revestimento DP
461
462     if (reyAnularFase3[2] < 2100) {
463         _gradiente = (objFluido.visc * velAnularFase3[2]) /
464             (1000 * pow((objPoco.revesID[1] - tubulacaoOD[1]), 2)
465             ); //DC i=0 e DP i=1
466         tiposDeEscoamento.push_back("Laminar");
467         gradientes.push_back(_gradiente);
468     }
469     else {
470         _gradiente = pow(objFluido.dens, 0.75) * pow(objFluido.
471             visc, 0.25) * pow(velAnularFase3[2], 1.75) / (1396 *
472             pow((objPoco.revesID[1] - tubulacaoOD[1]), 1.25));
473         //DC e DP
474         tiposDeEscoamento.push_back("Turbulento");
475         gradientes.push_back(_gradiente);
476     }
477
478     //CALCULOS FASE 4
479
480     //DP e DC
481     for (int i = 0; i < 2; i++) {
482
483         if (reyAnularFase4[i] < 2100) {
484             _gradiente = (objFluido.visc * velAnularFase4[i
485                 ]) / (1000 * pow((objPoco.diamPoco[3] -
486                 tubulacaoOD[i]), 2)); //DC i=0 e DP i=1
487             tiposDeEscoamento.push_back("Laminar");
488         }
489         else {
490             _gradiente = pow(objFluido.dens, 0.75) * pow(
491                 objFluido.visc, 0.25) * pow(velAnularFase4[i
492                 ], 1.75) / (1396 * pow((objPoco.diamPoco[3] -
493                 tubulacaoOD[i]), 1.25)); //DC e DP
494             tiposDeEscoamento.push_back("Turbulento");
495         }
496         gradientes.push_back(_gradiente);
497     }
498
499     // Revestimento DP
500
501     if (reyAnularFase4[2] < 2100) {

```



```

493         _gradiente = (objFluido.visc * velAnularFase4[2]) /
                    (1000 * pow((objPoco.revesID[2] - tubulacao0D[1]), 2)
                    );    //DC i=0 e DP i=1
494         tiposDeEscoamento.push_back("Laminar");
495         gradientes.push_back(_gradiente);
496     }
497     else {
498         _gradiente = pow(objFluido.dens, 0.75) * pow(objFluido.
                    visc, 0.25) * pow(velAnularFase4[2], 1.75) / (1396 *
                    pow((objPoco.revesID[2] - tubulacao0D[1]), 1.25));
                    //DC e DP
499         tiposDeEscoamento.push_back("Turbulento");
500         gradientes.push_back(_gradiente);
501     }
502 };

```

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

Listing 6.7: Arquivo de cabeçalho da classe CResultados.

```

503 /**
504 @autor Gleison Monteiro / Luisa Campbell
505 @file CResultados.h
506 @brief CResultados : E uma classe que disponibiliza ao usuario diversas
        formas de plotar os resultados.
507 */
508
509
510 #ifndef CResultados_H
511 #define CResultados_H
512
513 //Inclusao de classes das bibliotecas basicas
514 #include <string>
515 #include <iostream>
516 #include <fstream>
517 #include <vector>
518 #include <cmath>
519
520
521 #include "CFluido.h"          //Inclui o arquivo de cabeçalho da classe
        CFluido.h
522 #include "CGeometriaPoco.h"   //Inclui o arquivo de cabeçalho da classe
        CGeometriaPoco.h
523 #include "CPerdaDeCarga.h"    //Inclui o arquivo de cabeçalho da classe
        CPerdaDeCarga.h
524 #include "CGnuplot.h"         //Inclui a classe CGnuplot
525
526
527 using namespace std;
528

```

```

529 class CResultados {
530
531 public:
532
533     //ATRIBUTOS
534     vector<double> gradientesM;        ///vetor de gradientes de perda
        de carga em psi/metro
535
536     vector<double> xFase1;            ///vetor de profundidade para
        plotar a fase 1
537     vector<double> xFase2;            ///vetor de profundidade para
        plotar a fase 2
538     vector<double> xFase3;            ///vetor de profundidade para
        plotar a fase 3
539     vector<double> xFase4;            ///vetor de profundidade para
        plotar a fase 4
540
541     vector<double> yFase1;            ///vetor de queda de pressao
        acumulada para plotar a fase 1
542     vector<double> yFase2;            ///vetor de queda de pressao
        acumulada para plotar a fase 2
543     vector<double> yFase3;            ///vetor de queda de pressao
        acumulada para plotar a fase 3
544     vector<double> yFase4;            ///vetor de queda de pressao
        acumulada para plotar a fase 4
545
546     double pMaxFase1;                ///representa a
        perda de pressao total da fase 1
547     double pMaxFase2;                ///representa a
        perda de pressao total da fase 2
548     double pMaxFase3;                ///representa a
        perda de pressao total da fase 3
549     double pMaxFase4;                ///representa a
        perda de pressao total da fase 4
550
551     //double deltaPMax;                ///representa a queda de pressao
        maxima no poco
552
553
554     //METODOS
555
556     CResultados(){}                  // Construtor Default
557     virtual ~CResultados(){}         // Destrutor Default
558
559     void GradienteEmMetros(CPerdaDeCarga perdaCarga);
        ///Converte o gradiente para psi por metro
560     void SaidaDeDados(vector<double> x, vector<double> y, string
        nomeDoArquivo);    ///Permite salvar os dados em arquivo texto

```

```

561 void MenuResultados(CPerdaDeCarga perdaCarga);
                                     ///Oferece ao usuario opcoes
                                     de saida de dados
562 void VetoresProfundidade(CGeometriaPoco geometria);
                                     ///Define as profundidades para
                                     plotar graficos exportar os resultados
563 void VetoresPressao(CPerdaDeCarga carga, CGeometriaPoco
    geometria); ///Define as quedas de pressao em cada
               profundidade
564 void CalculoPerdaPressaoTotal (CPerdaDeCarga carga,
    CGeometriaPoco geometria); ///
565 void PlotarDados(CGnuplot& graficos, vector<double>& x, vector<
    double>& y); ///Define como salvar os graficos gerados
               pelo programa com o auxilio da biblioteca externa Gnuplot
566 void PlotarDados(CGnuplot& graficos, vector<double>& x1, vector<
    double>& y1, vector<double>& x2, vector<double>& y2, vector<
    double>& x3, vector<double>& y3, vector<double>& x4, vector<
    double>& y4); ///Define como salvar os graficos gerados
               pelo programa com o auxilio da biblioteca externa Gnuplot
567
568 };
569 #endif

```

Apresenta-se na listagem 6.8 o arquivo com código da classe CResultados.

Listing 6.8: Arquivo de implementação da classe CResultados.

```

570 /**
571 @autor Gleison Monteiro | Luisa Campbell
572 @file CResultados.cpp
573 @brief CResultados : É uma classe que disponibiliza ao usuario diversas
    formas de plotar os resultados.
574 */
575
576 #include "CResultados.h" ///Inclui o arquivo de cabecalho da classe
    CResultados.h
577
578 void CResultados::GradienteEmMetros(CPerdaDeCarga perdaCarga) {
579
580     for (int i = 0; i < perdaCarga.gradientes.size(); i++){
581         gradientesM.push_back(perdaCarga.gradientes[i] * 0.3048)
582         ;
583     }
584 }
585
586 void CResultados::VetoresProfundidade(CGeometriaPoco geometria){
587
588     double incremento1, incremento2, incremento3, incremento4;
589     incremento1 = geometria.compFases[0] / 100;

```

```

590     incremento2 = geometria.compFases[1] / 100;
591     incremento3 = geometria.compFases[2] / 100;
592     incremento4 = geometria.compFases[3] / 100;
593
594     //Preenchimento dos vetores de profundidade por fase
595     for (int i = 1; i < 101; i++) {
596
597         xFase1.push_back(i * incremento1);
598         xFase2.push_back(i * incremento2);
599         xFase3.push_back(i * incremento3);
600         xFase4.push_back(i * incremento4);
601
602     }
603
604
605 }
606
607 void CResultados::VetoresPressao(CPerdaDeCarga carga, CGeometriaPoco
    geometria){
608     double temp;
609     double compDC = (geometria.profundidadeM)*0.05;    ///comprimento
        da secao de DC para as fases 2,3 e 4
610
611     //Fase 1
612     for (int i = 0; i < xFase1.size(); i++) {
613
614         if (xFase1[i] <= 30) {
615             temp = (gradientesM[1] + gradientesM[2])
                *xFase1[i];
616             yFase1.push_back(temp);
617
618         }
619         else {
620             temp = ((gradientesM[0]+gradientesM[3])
                *(xFase1[i]-30))+((gradientesM[1] +
                gradientesM[2])*30);
621             yFase1.push_back(temp);
622         }
623
624     }
625
626
627     // Fase 2
628
629     for (int i = 0; i < xFase2.size(); i++){
630
631         // Menor que DC: apenas tem DC no poco revestido
632         if (xFase2[i] <= compDC) {

```

```

633         temp = (gradientesM[1] + gradientesM[4])*xFase2[
634             i];
635         yFase2.push_back(temp);
636     }
637     // Maior do que DC: > DC e DP no poco revestido (< que
638     // fase 1)
639     else if ((xFase2[i] >= compDC) && (xFase2[i] <= xFase1[(
640         xFase1.size() - 1)])) {
641         temp = (gradientesM[0] + gradientesM[6])*(xFase2
642             [i] - compDC) + ((gradientesM[1] +
643             gradientesM[4]) * compDC);
644         yFase2.push_back(temp);
645     }
646     // Profundidade e maior que a fase 1, porem menos que
647     // fase 1 e compDC: DC no poco aberto, DP no poco
648     // revestido
649     else if ((xFase2[i] >= xFase1[xFase1.size() - 1]) && (
650         xFase2[i] <= (compDC + xFase1[xFase1.size() - 1]))){
651         temp = ((gradientesM[0] + gradientesM[6]) * (
652             xFase2[i] - compDC)) + ((gradientesM[1] +
653             gradientesM[4]) * compDC);
654         yFase2.push_back(temp);
655     }
656     // Profundidade e maior que a fase 1 e o comprimento do
657     // DC, comecando a ter DP no poco aberto
658     else if ((xFase2[i] >= (compDC + xFase1[xFase1.size() -
659         1]))){
660         temp = (gradientesM[0] + gradientesM[5]) * (
661             xFase2[i] - compDC - xFase1[(xFase1.size() -
662             1)]) + (gradientesM[0] + gradientesM[6]) * (
663             xFase1[xFase1.size() - 1]) + ((gradientesM[1]
664             + gradientesM[4]) * compDC);
665         yFase2.push_back(temp);
666     }
667 }
668
669 //Fase 3
670
671 for (int i = 0; i < xFase3.size(); i++){
672
673     // Menor que DC: apenas tem DC no poco revestido
674     if (xFase3[i] <= compDC) {
675         temp = (gradientesM[1] + gradientesM[7])*xFase3[
676             i];

```

```

664         yFase3.push_back(temp);
665     }
666
667     // Maior do que DC: > DC e DP no poco revestido (< que
        fase 1)
668     else if ((xFase3[i] >= compDC) && (xFase3[i] <= xFase2[(
        xFase2.size() - 1)])) {
669         temp = (gradientesM[0] + gradientesM[9])*(xFase3
            [i] - compDC) + ((gradientesM[1] +
            gradientesM[7]) * compDC);
670         yFase3.push_back(temp);
671     }
672
673     // Profundidade e maior que a fase 1, porem menos que
        fase 1 e compDC: DC no poco aberto, DP no poco
        revestido
674     else if ((xFase3[i] >= xFase2[xFase2.size() - 1]) && (
        xFase3[i] <= (compDC + xFase2[xFase2.size() - 1]))){
675         temp = ((gradientesM[0] + gradientesM[9]) * (
            xFase3[i] - compDC)) + ((gradientesM[1] +
            gradientesM[7]) * compDC);
676         yFase3.push_back(temp);
677     }
678
679     // Profundidade e maior que a fase 1 e o comprimento do
        DC, comecando a ter DP no poco aberto
680     else if ((xFase3[i] >= (compDC + xFase2[xFase2.size() -
        1]))){
681         temp = (gradientesM[0] + gradientesM[8]) * (
            xFase3[i] - compDC - xFase2[(xFase2.size() -
            1)]) + (gradientesM[0] + gradientesM[9]) * (
            xFase2[xFase2.size() - 1]) + ((gradientesM[1]
            + gradientesM[7]) * compDC);
682         yFase3.push_back(temp);
683     }
684
685 }
686
687
688
689 //Fase 4
690
691 for (int i = 0; i < xFase4.size(); i++){
692
693     // Menor que DC: apenas tem DC no poco revestido
694     if (xFase4[i] <= compDC) {
695         temp = (gradientesM[1] + gradientesM[10])*xFase4
            [i];

```

```

696         yFase4.push_back(temp);
697     }
698
699     // Maior do que DC: > DC e DP no poco revestido (< que
700     // fase 1)
701     else if ((xFase4[i] >= compDC) && (xFase4[i] <= xFase3[(
702         xFase3.size() - 1)])) {
703         temp = (gradientesM[0] + gradientesM[12])*(
704             xFase4[i] - compDC) + ((gradientesM[1] +
705             gradientesM[10]) * compDC);
706         yFase4.push_back(temp);
707     }
708
709     // Profundidade e maior que a fase 1, porem menos que
710     // fase 1 e compDC: DC no poco aberto, DP no poco
711     // revestido
712     else if ((xFase4[i] >= xFase3[xFase3.size() - 1]) && (
713         xFase4[i] <= (compDC + xFase3[xFase3.size() - 1]))){
714         temp = ((gradientesM[0] + gradientesM[12]) * (
715             xFase4[i] - compDC)) + ((gradientesM[1] +
716             gradientesM[10]) * compDC);
717         yFase4.push_back(temp);
718     }
719
720     // Profundidade e maior que a fase 1 e o comprimento do
721     // DC, comecando a ter DP no poco aberto
722     else if ((xFase4[i] >= (compDC + xFase3[xFase3.size() -
723         1]))){
724         temp = (gradientesM[0] + gradientesM[11]) * (
725             xFase4[i] - compDC - xFase3[(xFase3.size() -
726             1)]) + (gradientesM[0] + gradientesM[12]) * (
727             xFase3[xFase3.size() - 1]) + ((gradientesM[1]
728             + gradientesM[10]) * compDC);
729         yFase4.push_back(temp);
730     }
731 }
732
733 void CResultados::CalculoPerdaPressaoTotal (CPerdaDeCarga carga,
734     CGeometriaPoco geometria){
735

```

```

728
729     pMaxFase1 = yFase1[0];
730     pMaxFase2 = yFase2[0];
731     pMaxFase3 = yFase3[0];
732     pMaxFase4 = yFase4[0];
733
734     for (int i = 1; i < xFase1.size(); i++) {
735         pMaxFase1 += yFase1[i];
736         pMaxFase2 += yFase2[i];
737         pMaxFase3 += yFase3[i];
738         pMaxFase4 += yFase4[i];
739     }
740
741
742
743
744 };
745
746 void CResultados::SaidaDeDados(vector<double> x, vector<double> y,
747     string nomeDoArquivo){
748
749     ofstream fout;
750     fout.open(nomeDoArquivo.c_str());
751
752     fout << "Profundidade_(metros)__" << "Queda_de_pressao_(psi)"
753         << endl;
754
755     for (int i = 0; i < y.size(); i++){
756         fout << x[i] << "_________________" << y[i] << endl;
757     }
758
759     fout.close();
760     cout << "Os_dados_foram_salvos_no_arquivo_" <<
761         nomeDoArquivo << "." << endl;
762     cout << "_\n";
763
764 }
765
766 void CResultados::MenuResultados(CPerdaDeCarga perdaCarga){
767
768     bool flag = true;
769     int escolha = 0;
770
771     cout << "\n
772     -----
773     " << endl;
774     cout << "____A_perda_de_carga_total_da_fase_1_foi_" <<
775         pMaxFase1 << "psi." << endl;

```



```

770     cout << "░░░░░A░perda░de░carga░total░da░fase░2░foi░" <<
        pMaxFase2 << "░psi." << endl;
771     cout << "░░░░░A░perda░de░carga░total░da░fase░3░foi░" <<
        pMaxFase3 << "░psi." << endl;
772     cout << "░░░░░A░perda░de░carga░total░da░fase░4░foi░" <<
        pMaxFase4 << "░psi." << endl;
773     cout << "
        -----
        " << endl;
774
775     ///Criando o objeto da classe Gnuplot e o configurando para as
        plotagens
776
777     CGnuplot gnu;
778     gnu.set_style("lines");
779     gnu.set_title("Perda░de░Carga░x░Profundidade");
780     gnu.set_xlabel("Profundidade░(m)");
781     gnu.set_ylabel("Perda░de░carga░(psi)");
782
783
784     ///O loop apresenta ao usuario as opcoes de entrega de
        resultados disponíveis pelo SHPP
785     while (flag) {
786
787         cout << "Este░e░um░menu░de░escolha░de░como░os░dados░
                serao░salvos.░Escolha░a░opcao░desejada:" << endl;
788     cout << "░\n";
789         cout << "Exportar░dados░de░profundidade░versus░queda░de░
                pressao░para░um░arquivo░texto:" << endl;
790     cout << "1░-░Fase░1." << endl;
791     cout << "2░-░Fase░2." << endl;
792     cout << "3░-░Fase░3." << endl;
793     cout << "4░-░Fase░4." << endl;
794     cout << "Plotar░dados░de░profundidade░versus░queda░de░pressao░
                utilizando░o░Gnuplot:" << endl;
795     cout << "5░-░Fase░1." << endl;
796     cout << "6░-░Fase░2." << endl;
797     cout << "7░-░Fase░3." << endl;
798     cout << "8░-░Fase░4." << endl;
799         cout << "9░-░Todas░as░fases." << endl;
800         cout << "10░-░Sair░do░programa." << endl;
801
802         cin >> escolha;
803         cin.get();
804
805         switch (escolha) {
806
807             case 1:

```

```
808             SaidaDeDados(xFase1, yFase1, "ResultadosFase1.
               txt");
809     cout << "┘\n";
810         break;
811
812
813     case 2:
814         SaidaDeDados(xFase2, yFase2, "ResultadosFase2.
               txt");
815     cout << "┘\n";
816         break;
817
818
819     case 3:
820         SaidaDeDados(xFase3, yFase3, "ResultadosFase3.
               txt");
821     cout << "┘\n";
822         break;
823
824
825     case 4:
826         SaidaDeDados(xFase4, yFase4, "ResultadosFase4.
               txt");
827     cout << "┘\n";
828         break;
829
830
831     case 5:
832         PlotarDados(gnu, xFase1, yFase1);
833     cout << "┘\n";
834         break;
835
836
837     case 6:
838         PlotarDados(gnu, xFase2, yFase2);
839     cout << "┘\n";
840         break;
841
842
843     case 7:
844         PlotarDados(gnu, xFase3, yFase3);
845     cout << "┘\n";
846         break;
847
848
849     case 8:
850         PlotarDados(gnu, xFase4, yFase4);
851     cout << "┘\n";
```

```

852             break;
853
854
855         case 9:
856             PlotarDados(gnu, xFase1, yFase1, xFase2, yFase2,
857                         xFase3, yFase3, xFase4, yFase4);
858             cout << "Grafico_comparativo_de_perda_de_carga_
859                     entre_as_fases" << endl;
860
861             cout << "\n";
862             break;
863
864         case 10:
865             flag = false;
866             break;
867
868             //Entrada invalida do usuario
869         default:
870             cout << "Esta_nao_e_uma_opcao_valida._Tente_
871                     novamente" << endl;
872
873             cout << "\n";
874
875     }
876
877 }
878
879 void CResultados::PlotarDados(CGnuplot& gnu, vector<double>& x, vector<
880 double>& y){
881     string nomeDoArquivo;
882
883     gnu.PlotVector(x, y);
884
885     cout << "Qual_o_nome_do_arquivo_em_que_voce_deseja_salvar_seu_grafico?"
886           << endl;
887     getline(cin, nomeDoArquivo);
888
889     gnu.savetopng(nomeDoArquivo);
890
891     cout << "O_grafico_foi_salvo_com_o_nome_" + nomeDoArquivo + ".png." <<
892           endl;
893
894     //gnu.replot();

```

```

894
895
896 }
897
898 void CResultados::PlotarDados(CGnuplot& gnu, vector<double>& x1, vector<
      double>& y1, vector<double>& x2, vector<double>& y2, vector<double>&
      x3, vector<double>& y3, vector<double>& x4, vector<double>& y4){
899
900 string nomeDoArquivo;
901
902 gnu.PlotVector(x1, y1);
903 gnu.PlotVector(x2, y2);
904 gnu.PlotVector(x3, y3);
905 gnu.PlotVector(x4, y4);
906
907 cout << "Qual o nome do arquivo em que voce deseja salvar seu grafico?"
      << endl;
908 getline(cin,nomeDoArquivo);
909
910 gnu.savetopng(nomeDoArquivo);
911
912 cout << "O grafico foi salvo com o nome " + nomeDoArquivo + ".png." <<
      endl;
913
914 //gnu.replot();
915
916 }

```

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

Listing 6.9: Arquivo de cabeçalho da classe CSHP.

```

919 /**
920 @autor Gleison Monteiro / Luisa Campbell
921 @file CSHP.h
922 @brief CSHP.h: E uma classe que representa o simulador de hidraulica de
      perfuracao
923 */
924
925 #ifndef CSHP_H
926 #define CSHP_H
927
928 //Inclusao das bibliotecas basicas
929 #include <string>
930 #include <iostream>
931
932 #include "CFluido.h" //Inclui o arquivo de cabecalho da classe
      CFluido.h
933 #include "CGeometriaPoco.h" //Inclui o arquivo de cabecalho da classe
      CGeometriaPoco.h

```

```

934 #include "CPerdaDeCarga.h"    //Inclui o arquivo de cabecalho da classe
    CPerdaDeCarga.h
935 #include "CResultados.h"      //Inclui o arquivo de cabecalho da classe
    CResultados.h
936 #include "CGnuplot.h"         //Inclui a classe CGnuplot
937
938 using namespace std;
939
940 class CSHPP {
941
942 public:
943
944     //ATRIBUTOS
945
946
947     //METODOS
948     CSHPP(){}                  // Construtor Default
949     virtual ~CSHPP(){}        // Destrutor Default
950     void Simulacao();          //Metodo que representa a simulacao de
        hidraulica
951
952
953
954
955
956
957 };
958
959
960
961
962 #endif

```

Apresenta-se na listagem 6.10 o arquivo com código da classe CSHPP.

Listing 6.10: Arquivo de implementação da classe CSHPP.

```

963 /**
964 @autor Gleison Monteiro / Luisa Campbell
965 @file CSHPP.cpp
966 @brief CSHPP.cpp : Implementa os metodos da classe CSHPP.h
967 */
968
969 // Inclusao de classes e bibliotecas
970 #include "CSHPP.h"
971
972 void CSHPP::Simulacao() {
973
974     cout << "

```

```

975         " << endl;
        cout << "|XXXXXXXXXXXXXXXXXXXXXXXXXXXXUniversidade_Estadual_do_Norte_
            Fluminense_Darcy_Ribeiro_-UENFXXXXXXXXXXXXXXXXXXXXXXXXXX|" << endl
            ;
976     cout << "|XXXXXXXXXXXXXXXXXXXXXXXXXXXXLaboratorio_de_Engenharia_e_
            Exploracao_de_Petroleo_-LENEPXXXXXXXXXXXXXXXXXXXXXXXXXX|" <<
            endl;
977     cout << "|XXXXXXXXXXXXXXXXXXXXXXXXXXXXProgramacao_
            Pratica_-Projeto_C++XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX|" <<
            endl;
978     cout << "|XXXXXXXXXXXXXXXXXXXXXXXXXXXXProfessor:_Andre
            _Duarte_BuenoXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX|" << endl;
979     cout << "|XXXXXXXXXXXXXXXXXXXXXXXXXXXXAlunos:_Gleison_
            Monteiro_e_Luisa_CampbellXXXXXXXXXXXXXXXXXXXXXXXXXX|" <<
            endl;
980     cout << "|XXXXXXXXXXXXXXXXXXXXXXXXXXXXSIMULADOR_DE_HIDRAULICA_DE_
            PERFURACAO_DE_POCO_-SHPPXXXXXXXXXXXXXXXXXXXXXXXXXX|" << endl;
981     cout << "
            -----
            " << endl;

982
983     CFluido fluido;
984     fluido.EntradaDadosFluido();
985
986     CGeometriaPoco geomPoco;
987     geomPoco.EntradaDadosGeometriaPoco();
988     geomPoco.CalculoFases();
989
990     CPerdaDeCarga perdaCarga;
991     perdaCarga.EntradaDadosPerdaDeCarga();
992     perdaCarga.CalculoVelocidadeTubulacao();
993     perdaCarga.CalculoVelocidadeAnular(geomPoco);
994     perdaCarga.CalculoReynoldsTubulacao(fluido);
995     perdaCarga.CalculoReynoldsAnular(fluido, geomPoco);
996     perdaCarga.CalculoPDCTubulacao(fluido, geomPoco);
997     perdaCarga.CalculoPDCAnular(fluido, geomPoco);
998
999     CResultados resultados;
1000     resultados.GradientEmMetros(perdaCarga);
1001     resultados.VetoresProfundidade(geomPoco);
1002     resultados.VetoresPressao(perdaCarga, geomPoco);
1003     resultados.CalculoPerdaPressaoTotal(perdaCarga, geomPoco);
1004     resultados.MenuResultados(perdaCarga);
1005
1006
1007 };

```

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

Listing 6.11: Arquivo de implementação da classe main.

```
1008 /**
1009  @autor Gleison Monteiro / Luisa Campbell
1010  @file main.cpp
1011  @brief Essa função é responsável por executar o programa
1012  */
1013
1014 #include "CFluido.h"
1015 #include "CGeometriaPoco.h"
1016 #include "CPerdaDeCarga.h"
1017 #include "CResultados.h"
1018 #include "CSHPP.h"
1019
1020
1021 int main(int argc, char** argv)
1022 {
1023
1024     CSHPP programa;
1025     programa.Simulacao();
1026
1027
1028     return 0;
1029 }
```

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: Perda de carga para a fase 1

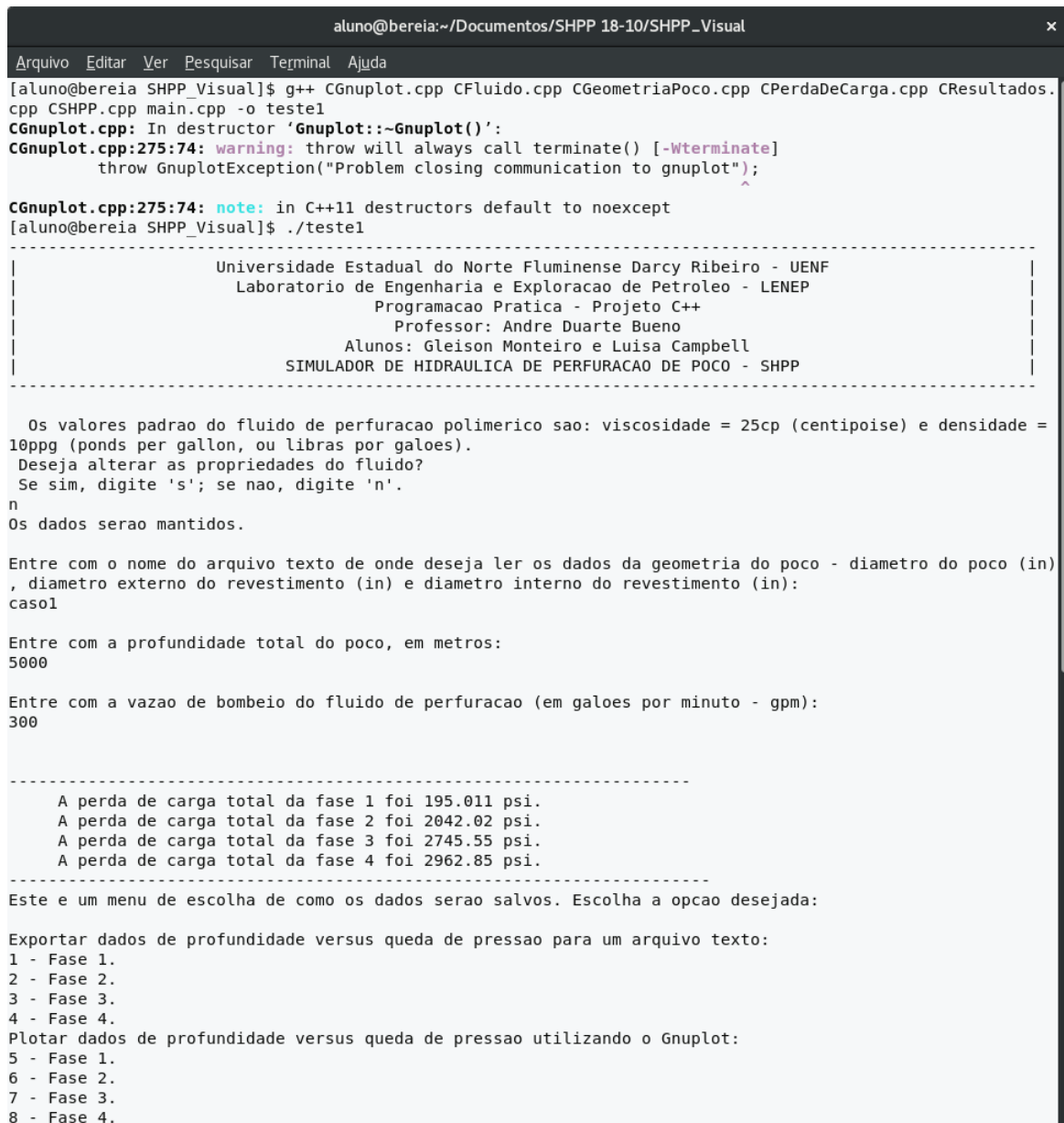
No primeiro teste, o usuário opta por manter as propriedades *default* do fluido sugeridas pelo software (viscosidade=25cp e densidade=10ppg). O usuário seleciona os dados de geometria de poço referentes ao Caso 1 através de entrada de dados por arquivo de disco, em que deve ser informado o nome do arquivo .txt (Figura 7.1). Os dados do arquivo selecionado são mostrados na Tabela 7.1.

Diâmetro de Poço	Casing OD (in)	Casing ID (in)
36.000	30.000	28.875
26.000	20.000	19.125
17.500	13.375	12.715
12.250	0.000	0.000

Tabela 7.1: Dados de geometria de poço referentes ao Caso 1

Após a leitura de dados do arquivo, foi testada a opção de salvar os resultados de perda de carga para a Fase 1 em arquivo .txt e, em seguida, plotar estes resultados em gráfico através do software Gnuplot (Figura 7.2).

A imagem foi salva no formato .png (Figura 7.3).



```

aluno@bereia:~/Documentos/SHPP 18-10/SHPP_Visual
Arquivo Editar Ver Pesquisar Terminal Ajuda
[aluno@bereia SHPP_Visual]$ g++ CGnuplot.cpp CFluido.cpp CGeometriaPoco.cpp CPerdaDeCarga.cpp CResultados.
cpp CSHPP.cpp main.cpp -o testel
CGnuplot.cpp: In destructor 'Gnuplot::~Gnuplot()':
CGnuplot.cpp:275:74: warning: throw will always call terminate() [-Wterminate]
    throw GnuplotException("Problem closing communication to gnuplot");
    ^
CGnuplot.cpp:275:74: note: in C++11 destructors default to noexcept
[aluno@bereia SHPP_Visual]$ ./testel
-----
                Universidade Estadual do Norte Fluminense Darcy Ribeiro - UENF
                Laboratorio de Engenharia e Exploracao de Petroleo - LENEP
                Programacao Pratica - Projeto C++
                Professor: Andre Duarte Bueno
                Alunos: Gleison Monteiro e Luisa Campbell
                SIMULADOR DE HIDRAULICA DE PERFURACAO DE POCO - SHPP
-----

Os valores padrao do fluido de perfuracao polimerico sao: viscosidade = 25cp (centipoise) e densidade =
10ppg (ponds per gallon, ou libras por galoes).
Deseja alterar as propriedades do fluido?
Se sim, digite 's'; se nao, digite 'n'.
n
Os dados serao mantidos.

Entre com o nome do arquivo texto de onde deseja ler os dados da geometria do poco - diametro do poco (in)
, diametro externo do revestimento (in) e diametro interno do revestimento (in):
caso1

Entre com a profundidade total do poco, em metros:
5000

Entre com a vazao de bombeio do fluido de perfuracao (em galoes por minuto - gpm):
300

-----
A perda de carga total da fase 1 foi 195.011 psi.
A perda de carga total da fase 2 foi 2042.02 psi.
A perda de carga total da fase 3 foi 2745.55 psi.
A perda de carga total da fase 4 foi 2962.85 psi.
-----

Este e um menu de escolha de como os dados serao salvos. Escolha a opcao desejada:

Exportar dados de profundidade versus queda de pressao para um arquivo texto:
1 - Fase 1.
2 - Fase 2.
3 - Fase 3.
4 - Fase 4.
Plotar dados de profundidade versus queda de pressao utilizando o Gnuplot:
5 - Fase 1.
6 - Fase 2.
7 - Fase 3.
8 - Fase 4.

```

Figura 7.1: Tela inicial do programa mostrando o menu para entrada de dados



```
aluno@bereia:~/Documentos/SHPP 18-10/SHPP_Visual
Arquivo Editar Ver Pesquisar Terminal Ajuda
Este e um menu de escolha de como os dados serao salvos. Escolha a opcao desejada:

Exportar dados de profundidade versus queda de pressao para um arquivo texto:
1 - Fase 1.
2 - Fase 2.
3 - Fase 3.
4 - Fase 4.
Plotar dados de profundidade versus queda de pressao utilizando o Gnuplot:
5 - Fase 1.
6 - Fase 2.
7 - Fase 3.
8 - Fase 4.
9 - Todas as fases.
10 - Sair do programa.
1
Os dados foram salvos no arquivo ResultadosFase1.txt.

Este e um menu de escolha de como os dados serao salvos. Escolha a opcao desejada:

Exportar dados de profundidade versus queda de pressao para um arquivo texto:
1 - Fase 1.
2 - Fase 2.
3 - Fase 3.
4 - Fase 4.
Plotar dados de profundidade versus queda de pressao utilizando o Gnuplot:
5 - Fase 1.
6 - Fase 2.
7 - Fase 3.
8 - Fase 4.
9 - Todas as fases.
10 - Sair do programa.
5
Qual o nome do arquivo em que voce deseja salvar seu grafico?
grafico1
O grafico foi salvo com o nome grafico1.png.

Este e um menu de escolha de como os dados serao salvos. Escolha a opcao desejada:

Exportar dados de profundidade versus queda de pressao para um arquivo texto:
1 - Fase 1.
2 - Fase 2.
3 - Fase 3.
4 - Fase 4.
Plotar dados de profundidade versus queda de pressao utilizando o Gnuplot:
5 - Fase 1.
6 - Fase 2.
7 - Fase 3.
8 - Fase 4.
9 - Todas as fases.
10 - Sair do programa.
10
[aluno@bereia SHPP_Visual]$
```

Figura 7.2: Menu de escolha para exportar os dados

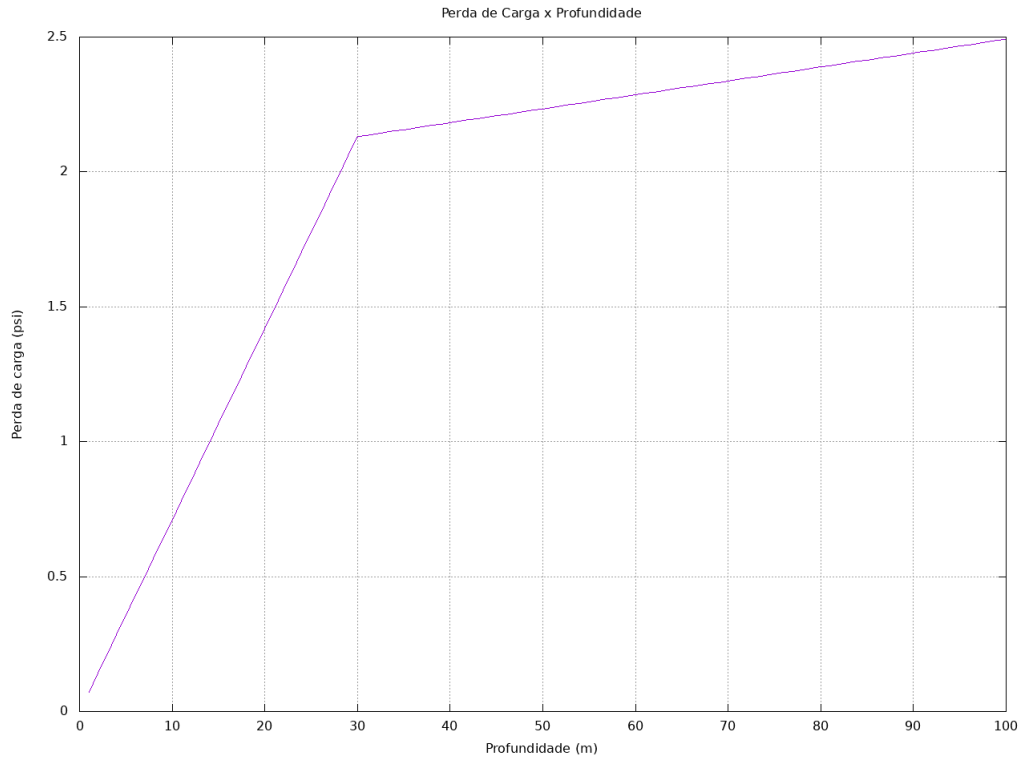


Figura 7.3: Gráfico gerado para a perda de carga na Fase 1

7.2 Teste 2: Perda de carga de todas as fases

No segundo teste, o usuário opta por modificar as propriedades *default* do fluido (viscosidade=25cp e densidade=10ppg). Assim, a função de entrada de dados pelo usuário. O usuário deve informar o novo valor de viscosidade e o novo valor de densidade.

Posteriormente, o usuário seleciona os dados de geometria de poço referentes ao Caso 2 (Figura 7.4), fornecendo o nome do arquivo no qual estão salvas as informações. Os dados de geometria do Caso 2 são mostrados na Tabela 7.2.

Diâmetro de Poço	Casing OD (in)	Casing ID (in)
36.000	30.000	28.875
26.000	18.625	17.755
16.000	11.750	11.000
10.625	0.000	0.000

Tabela 7.2: Dados de geometria de poço referentes ao Caso 2

Após a leitura de dados do arquivo, foi testada a opção de plotar o resultado de perda de carga para todas as fases através do software Gnuplot (Figura 7.5).

A imagem foi salva no formato .png (Figura 7.6).

```

aluno@bereia:~/Documentos/SHPP 18-10/SHPP_Visual
Arquivo Editar Ver Pesquisar Terminal Ajuda
[aluno@bereia SHPP_Visual]$ g++ CGnuplot.cpp CFluido.cpp CGeometriaPoco.cpp CPerdaDeCarga.cpp CResultados.cpp
p CSHPP.cpp main.cpp -o teste2
CGnuplot.cpp: In destructor 'Gnuplot::~Gnuplot()':
CGnuplot.cpp:275:74: warning: throw will always call terminate() [-Wterminate]
    throw GnuplotException("Problem closing communication to gnuplot");
    ^
CGnuplot.cpp:275:74: note: in C++11 destructors default to noexcept
[aluno@bereia SHPP_Visual]$ ./teste2
-----
|                               Universidade Estadual do Norte Fluminense Darcy Ribeiro - UENF                               |
|                               Laboratorio de Engenharia e Exploracao de Petroleo - LENEP                               |
|                               Programacao Pratica - Projeto C++                               |
|                               Professor: Andre Duarte Bueno                               |
|                               Alunos: Gleison Monteiro e Luisa Campbell                               |
|                               SIMULADOR DE HIDRAULICA DE PERFURACAO DE POCO - SHPP                               |
|-----|

Os valores padrao do fluido de perfuracao polimerico sao: viscosidade = 25cp (centipoise) e densidade = 10
ppg (pounds per gallon, ou libras por galoes).
Deseja alterar as propriedades do fluido?
Se sim, digite 's'; se nao, digite 'n'.
s
    Entre o novo valor da viscosidade (em cp):
20

    Entre o novo valor da densidade (em ppg):
13

Entre com o nome do arquivo texto de onde deseja ler os dados da geometria do poco - diametro do poco (in),
diametro externo do revestimento (in) e diametro interno do revestimento (in):
caso2


Entre com a profundidade total do poco, em metros:
5000

Entre com a vazao de bombeio do fluido de perfuracao (em galoes por minuto - gpm):
300

-----
A perda de carga total da fase 1 foi 224.536 psi.
A perda de carga total da fase 2 foi 2351.08 psi.
A perda de carga total da fase 3 foi 3166.36 psi.
A perda de carga total da fase 4 foi 3619.43 psi.
-----

```

Figura 7.4: Tela inicial do programa mostrando o menu para entrada de dados



```
aluno@bereia:~/Documentos/SHPP 18-10/SHPP_Visual
Arquivo Editar Ver Pesquisar Terminal Ajuda

-----
A perda de carga total da fase 1 foi 224.536 psi.
A perda de carga total da fase 2 foi 2351.08 psi.
A perda de carga total da fase 3 foi 3166.36 psi.
A perda de carga total da fase 4 foi 3619.43 psi.
-----

Este e um menu de escolha de como os dados serao salvos. Escolha a opcao desejada:

Exportar dados de profundidade versus queda de pressao para um arquivo texto:
1 - Fase 1.
2 - Fase 2.
3 - Fase 3.
4 - Fase 4.
Plotar dados de profundidade versus queda de pressao utilizando o Gnuplot:
5 - Fase 1.
6 - Fase 2.
7 - Fase 3.
8 - Fase 4.
9 - Todas as fases.
10 - Sair do programa.
9
Qual o nome do arquivo em que voce deseja salvar seu grafico?
grafico2
0 grafico foi salvo com o nome grafico2.png.
Grafico comparativo de perda de carga entre as fases

Este e um menu de escolha de como os dados serao salvos. Escolha a opcao desejada:

Exportar dados de profundidade versus queda de pressao para um arquivo texto:
1 - Fase 1.
2 - Fase 2.
3 - Fase 3.
4 - Fase 4.
Plotar dados de profundidade versus queda de pressao utilizando o Gnuplot:
5 - Fase 1.
6 - Fase 2.
7 - Fase 3.
8 - Fase 4.
9 - Todas as fases.
10 - Sair do programa.
10
[aluno@bereia SHPP_Visual]$
```

Figura 7.5: Menu de escolha para exportar os dados

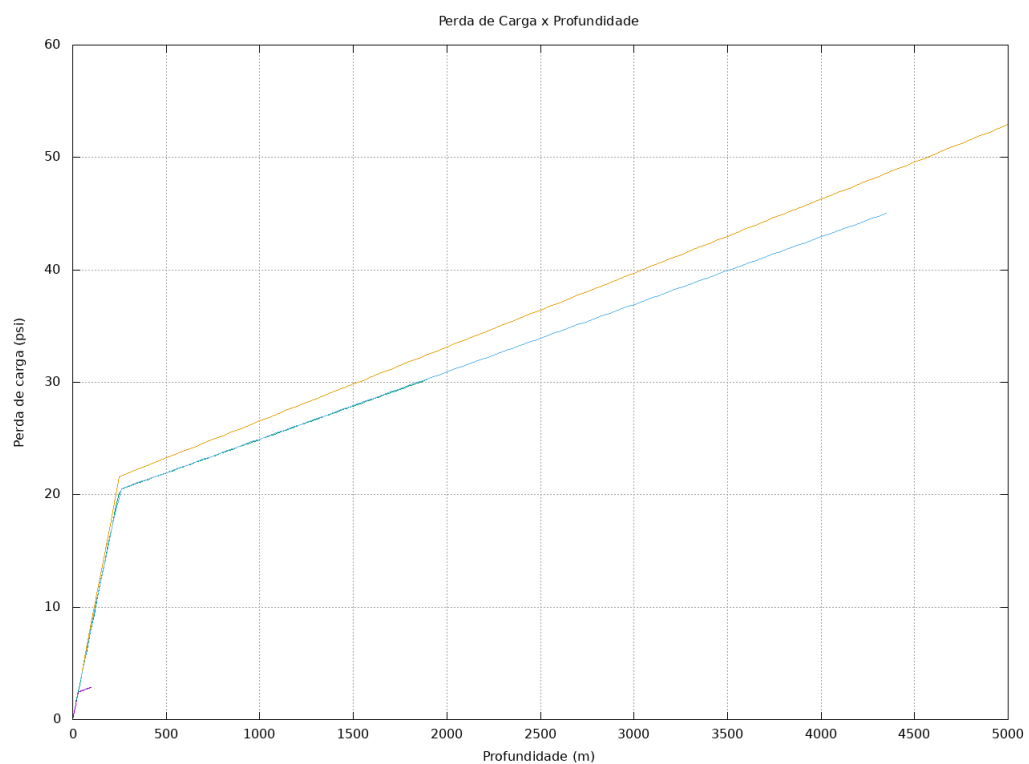


Figura 7.6: Gráfico gerado para a perda de carga em todas as fases

Capítulo 8

Documentação

Apresenta-se neste capítulo a documentação de uso do software SHPP - Simulador de Hidráulica de Perfuração de Poço. Esta documentação tem o formato de uma apostila que explica passo a passo 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 executar o software

Abrir o terminal, ir para o diretório onde os códigos se encontram, compilar o programa e executá-lo.

1. Para compilar o programa, digitar o seguinte comando no terminal:

```
g++ CGnuplot.cpp CFluido.cpp CGeometriaPoco.cpp CPerdaDeCarga.cpp CRe-  
sultados.cpp CSHPP.cpp main.cpp -o NomedoArquivoTeste
```

2. Para executar o programa, digitar o seguinte comando no terminal:

```
./NomedoArquivoTeste
```

3. A interface inicial do software será executada.

4. Inicialmente, o usuário pode escolher entre a opção de manter as propriedades *default* do fluido ou modificá-las, escolhendo entre as opções:

- (a) Opção 1: Manter as propriedades - digite 'n'
- (b) Opção 2: Modificar as propriedades - digite 's'

Caso o usuário escolha a Opção 2, o software irá pedir os novos valores para as propriedades do fluido, e serão dadas as instruções de entrada de dados pelo teclado.

5. Posteriormente, independente da opção escolhida, o usuário deverá entrar com o nome de um arquivo texto no qual estão armazenados os dados de geometria do poço, fornecidos pelo software. Existem dois arquivos possíveis: caso1 e caso2. O nome do arquivo deve ser digitado, sem a necessidade de informar o formato da extensão.
6. Após carregar os dados, o software irá informar os valores de perda de carga para cada fase do poço selecionado.
7. Em seguida, um menu com diversas opções de saída de dados será mostrado. O usuário pode escolher entre as opções abaixo:
 - (a) Saída de dados em arquivo .txt: Os dados de perda de carga para a fase escolhida serão salvos em um arquivo .txt.
 - (b) Plotar os dados com auxílio do Gnuplot: O gráfico de perda de carga será plotado para a fase escolhida, e será salvo em formato .png.Caso o usuário escolha a opção de plotar os dados, deverá ser informado o nome do arquivo em que o gráfico será salvo no diretório.
8. O usuário pode escolher mais de uma opção de saída de dados ou optar por sair do programa, selecionando a última opção do menu.

Veja no Capítulo 7 - Teste alguns exemplos de uso do software.

8.2 Documentação para desenvolvedor

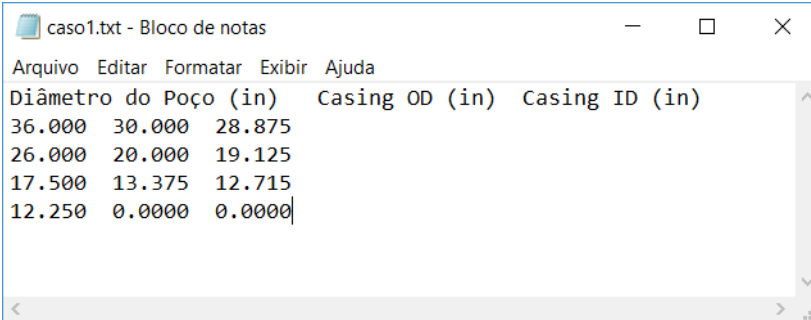
Apresenta-se nesta seção a documentação para o desenvolvedor, isto é, informações para usuários que queiram modificar, aperfeiçoar ou ampliar este software.

8.2.1 Dependências

Para compilar o software é necessário atender as seguintes dependências:

- No sistema operacional GNU/Linux:
 - Instalar o compilador g++ da GNU disponível em <http://gcc.gnu.org>.
 - Para instalar no GNU/Linux use o comando `yum install gcc`.
- No sistema operacional Windows:
 - Instalar um compilador de sua escolha.
 - Recomenda-se a instalação do Dev C++.

- Software externo Gnuplot:
 - 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**.
- Dependência de arquivos com dados:
 - O programa depende da existência de um arquivo contendo os dados de geometria de poço, no formato `.txt`.
 - O formato dos dados no arquivo é mostrado na Figura 8.1.



The screenshot shows a text editor window with the title 'caso1.txt - Bloco de notas'. The window contains a table with three columns: 'Diâmetro do Poço (in)', 'Casing OD (in)', and 'Casing ID (in)'. The data is as follows:

Diâmetro do Poço (in)	Casing OD (in)	Casing ID (in)
36.000	30.000	28.875
26.000	20.000	19.125
17.500	13.375	12.715
12.250	0.0000	0.0000

Figura 8.1: Formato de preenchimento do arquivo contendo os dados de geometria de poço

8.2.2 Como gerar a documentação usando doxygen

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

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

Passos para gerar a documentação usando o **doxygen**.

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

- Abra um terminal.
- Vá para o diretório onde está o código.

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

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

```
doxygen -g
```

- Peça para o `doxygen` gerar a documentação.

```
doxygen
```

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

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

ou

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


Referências Bibliográficas

- [Bourgoyne et al., 1986] Bourgoyne, A. T., Millheim, K. K., Chenevert, M. E., and Young, F. S. (1986). *Applied drilling engineering*. Society of Petroleum Engineers Richardson.
- [Gabolde and Nguyen, 2006] Gabolde, G. and Nguyen, J.-P. (2006). *Drilling Data Handbook 7th*. Editions Technip.
- [Lapeyrouse, 2002] Lapeyrouse, N. J. (2002). *Formulas and calculations for drilling, production, and workover*. Gulf professional publishing.
- [Prideco, 2003] Prideco, G. (2003). *Drill Pipe Data Tables*.
- [Smith, 2009] Smith (2009). *Dimensional Data Handbook*, 4th edition.