

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

PROJETO ENGENHARIA
DESENVOLVIMENTO DO SOFTWARE
ESP PERFORMANCE - SIMULADOR DE CURVAS DE DESEMPENHO
DE BCS: HEAD E PERDAS DE CARGA
TRABALHO DA DISCIPLINA PROGRAMAÇÃO PRÁTICA

AUTORES
Fernando Henrique Moreira Braga Fernandes
Maria Beatriz Lisboa Pereira

Prof. André Duarte Bueno

MACAÉ - RJ
Julho - 2017

Sumário

1	Introdução	1
1.1	Escopo do problema	1
1.2	Objetivos	1
2	Especificação	3
2.1	Especificação do programa - descrição dos requisitos	3
2.2	Nome do sistema/produto	3
2.3	Especificação	4
2.3.1	Requisitos funcionais	4
2.3.2	Requisitos não funcionais	4
2.4	Casos de uso do sistema – cenários	4
2.5	Casos de uso	5
2.5.1	Diagrama de caso de uso geral	5
2.5.2	Diagrama de caso de uso específico	6
3	Elaboração	7
3.1	Análise de domínio	7
3.2	Formulação teórica	7
3.3	Identificação de pacotes – assuntos	10
3.4	Diagrama de pacotes – assuntos	10
4	AOO – Análise Orientada a Objeto	12
4.1	Diagramas de classes	12
4.1.1	Dicionário de classes	13
4.2	Diagrama de seqüência – eventos e mensagens	15
4.2.1	Diagrama de seqüência geral	15
4.3	Diagrama de comunicação – colaboração	17
4.4	Diagrama de máquina de estado	17
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	22
6	Implementação	24
6.1	Código fonte	24
7	Teste	42
7.1	Teste 1: Recebendo dados do usuário via terminal	42
7.2	Teste 2: Cálculos do <i>head</i> teórico e alturas de elevação	43
7.3	Teste 3: Plotagem dos gráficos	44
8	Documentação	46
8.1	Documentação do usuário	46
8.1.1	Como instalar o software	46
8.1.2	Como rodar o software	46
8.2	Documentação para desenvolvedor	47
8.2.1	Dependências	47
8.2.2	Documentação usando doxygen	47
9	Sugestões	49
9.1	Classe CFluido	49
9.2	Classe CBombaBCS	49
9.3	Classe CPerdaCarga	49
9.4	Classe CSimuladorESPPerformance	49

Capítulo 1

Introdução

No presente projeto de engenharia desenvolve-se o software ESPPerformance, um software aplicado a engenharia de petróleo, mas especificamente à área de elevação artificial. O Software baseia-se principalmente na equação de altura de elevação de bombas centrífugas submersas (BCS) e a partir desta em cada termo componente da mesma. Dessa forma, leva em consideração as parametrizações principais para cada termo da equação, head teórico e perdas de carga, para simular as curvas de performance para diferentes condições de operação da BCS.

1.1 Escopo do problema

O software irá calcular, simular e apresentar resultados para análise em forma gráfica, obtidos a partir da equação da altura de elevação de bombas, e essa por sua vez dos termos componentes. Para realização dos cálculos de cada um dos termos da equação da altura de elevação, serão utilizados modelos presentes na literatura para cada um dos termos. O software contribuirá e auxiliará na previsibilidade do comportamento da bomba e do seu desempenho, os quais são fundamentais para definir condições de funcionamento, o que garante expressivas melhorias na delimitação de parâmetros de operação em fase de projeto e consequente aumento de eficiência na produção de petróleo utilizando BCS.

1.2 Objetivos

Os objetivos deste projeto de engenharia são:

- Objetivo geral:
 - Desenvolver um simulador que calcule as curvas de performance de operação de BCS utilizado em elevação artificial de petróleo a partir de parametrizações para escoamento de fluidos em bombas centrífugas.
- Objetivos específicos:

- Cálculo do *Head* teórico;
- Cálculo da perda de carga por atrito no rotor;
- Cálculo da perda de carga por atrito de disco;
- Cálculo da perda de carga por atrito no difusor;
- Cálculo da perda de carga por choques;
- Cálculo da perda de carga por recirculação;
- Cálculo do *Head* real da BCS;
- Plotar os resultados dos cálculos para o *Head* teórico subtraído das perdas de carga;

Capítulo 2

Especificação

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

2.1 Especificação do programa - descrição dos requisitos

A finalidade desse programa é calcular a altura de elevação da BCS, a partir da equação da altura de elevação de bomba, para a realização desse cálculo, o software irá calcular também o head teórico, e as devidas perdas de carga atribuídas ao escoamento de fluidos na bomba. Em seguida, o programa irá apresentar os resultados em forma gráfica e comparativa.

O projeto a ser desenvolvido consiste em um software que solicitará do usuário os dados de propriedades de fluidos, e escolher em um menu a condição desejada para operação da bomba. A partir dos dados recolhidos o programa realizará os devidos cálculos e retornará ao usuário em forma de gráfico.

O desenvolvimento do software utilizará o conceito de programação orientada a objeto (POO) em linguagem C++ e sua interface será em modo texto. O programa terá seu código aberto e ser passível de modificações.

2.2 Nome do sistema/produto

Nome	ESPPerformance
Componentes principais	Simulador de cálculo de desempenho de BCS
Missão	Calcular e plotar a altura de elevação de BCS

2.3 Especificação

Apresenta-se a seguir a especificação do software.

2.3.1 Requisitos funcionais

Apresenta-se a seguir os requisitos funcionais.

RF-01	O sistema deve conter uma base de dados contendo parametrizações que são funções de propriedades do fluido e características da bomba .
--------------	---

RF-02	O usuário deverá ter liberdade para escolher o tipo de fluido a ser usado, através da viscosidade.
--------------	--

RF-03	Deve permitir o carregamento de arquivos criados pelo software.
--------------	---

RF-04	Deve permitir a escolha da condição operacional da bomba, através da rotação.
--------------	---

RF-05	O programa plotará os resultados em forma de gráfico. .
--------------	---

2.3.2 Requisitos não funcionais

RNF-01	Os cálculos devem ser feitos utilizando equações presentes da literatura.
---------------	---

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

2.4 Casos de uso do sistema – cenários

Um *caso de uso* descreve um ou mais cenários de uso do software, exemplos de uso, como o sistema interage com usuários externos (atores). Ademais, ele deve representar uma seqüência típica de uso do software (a execução de determinadas tarefas-padrão). Também deve representar as exceções, casos em que o usuário comete algum erro, em que o sistema não consegue realizar as tarefas solicitadas.

Tabela 2.1: Caso de uso.

Nome do caso de uso:	Calcular e plotar a altura de elevação de BCS
Resumo/descrição:	Determinação da elevação do BCS a partir dos parâmetros inerentes ao escoamento em bomba.
Etapas:	<ol style="list-style-type: none"> 1. Criar objeto simulador. 2. Escolher a condição operacional da bomba. 3. Escolher a viscosidade do fluido. 4. Calcular os parâmetros. 5. Gerar gráficos de resultados.
Cenários alternativos:	Um cenário alternativo envolve uma entrada errada do usuário (por exemplo, escolher opção não existente para o valor da rotação da bomba. O software informará o usuário sobre a o erro.

2.5 Casos de uso

Nesta seção são apresentados os diagramas de caso de uso geral e caso de uso específico gerados pelo software *Umbrello*, disponível em [Umbrello 2017].

2.5.1 Diagrama de caso de uso geral

O diagrama de caso de uso geral da Figura 2.1 mostra o usuário acessando o simulador para entrar com os dados, calcular a altura de elevação da bomba e gerando gráficos a partir do Gnuplot.

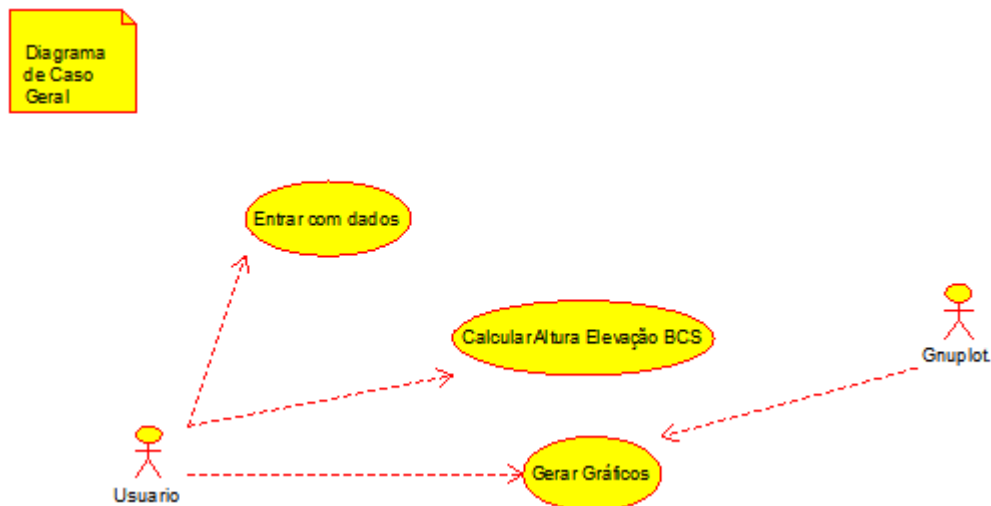


Figura 2.1: Diagrama de caso de uso – Caso de uso geral

2.5.2 Diagrama de caso de uso específico

O diagrama de caso de uso geral da Figura 2.1 mostra o usuário acessando o simulador para entrar com a viscosidade do fluido, escolher a rotação da bomba, calcular os parâmetros, calcular a altura de elevação da bomba, gerar gráficos a partir do **Gnuplot** e analisar os resultados.

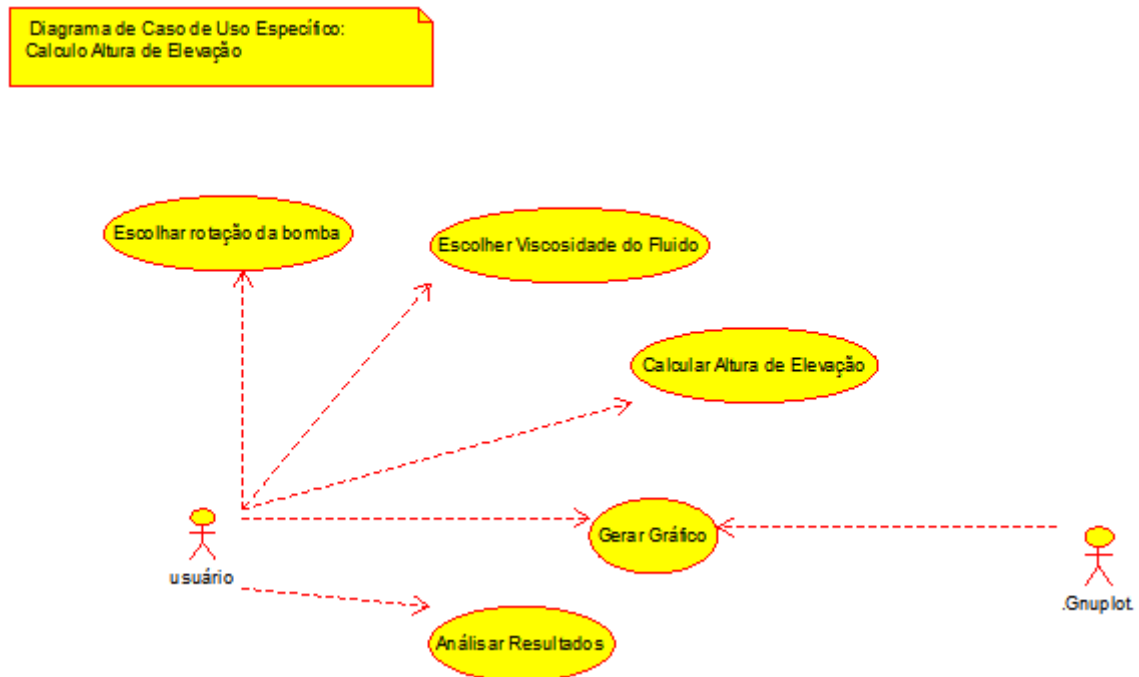


Figura 2.2: Diagrama de caso de uso específico – Cálculo da altura de elevação da bomba.

Capítulo 3

Elaboração

Definida a especificação do software e montados os diagramas de caso de uso, faz-se necessário agora desenvolver a elaboração do software. Definiremos as áreas de atuação e conceitos teóricos que envolvem a problemática, construiremos então, o diagrama de pacotes relacionando os assuntos.

3.1 Análise de domínio

Para uma melhor compreensão deste trabalho partiremos do pressuposto que o usuário possui conhecimentos básicos a cerca das áreas apresentadas.

Com o objetivo de entender o domínio e a abrangência do sistema em desenvolvimento, apresenta-se aqui a análise do domínio. Pensando de forma mais genérica identificamos os conceitos fundamentais que envolvem o sistema. Considerando que o objetivo principal do software é realizar o cálculo da altura de elevação da bomba centrífuga submersa, temos relacionamento entre as áreas:

- Mecânica dos Fluidos, envolvendo as propriedades do fluido, no caso a propriedade relacionada ao sistema é a densidade do fluido;
- Elevação e Escoamento de Petróleo, envolvendo conceitos sobre a utilização da BCS, suas propriedades (rotação da bomba) e as perdas de carga envolvidas na utilização;

3.2 Formulação teórica

Para estimar as alturas de elevação e as perdas de carga no estágio da bomba utiliza-se as combinações de parametrizações pesquisadas na literatura, com maior precisão em relação aos dados experimentais de Amaral(2007). A altura de elevação do estágio da bomba é dada por:

$$H = H_{teo} - h_{atr} - h_{cho} - h_{rec} - h_{dif} - h_{dis} \quad (3.1)$$

onde $H_{teo}[m]$ é a altura de elevação teórica da bomba, $h_{atr}[m]$ é a perda de carga por atritos no rotor da bomba, $h_{cho}[m]$ é a perda de carga por choques no rotor, $h_{rec}[m]$ é a perda de carga por recirculações no rotor, $h_{dif}[m]$ é a perda de carga por atritos no difusor da bomba, e $h_{dis}[m]$ é a perda de carga por atrito de disco da bomba, essas perdas são ilustradas em estágio de BCS na Figura 3.1.

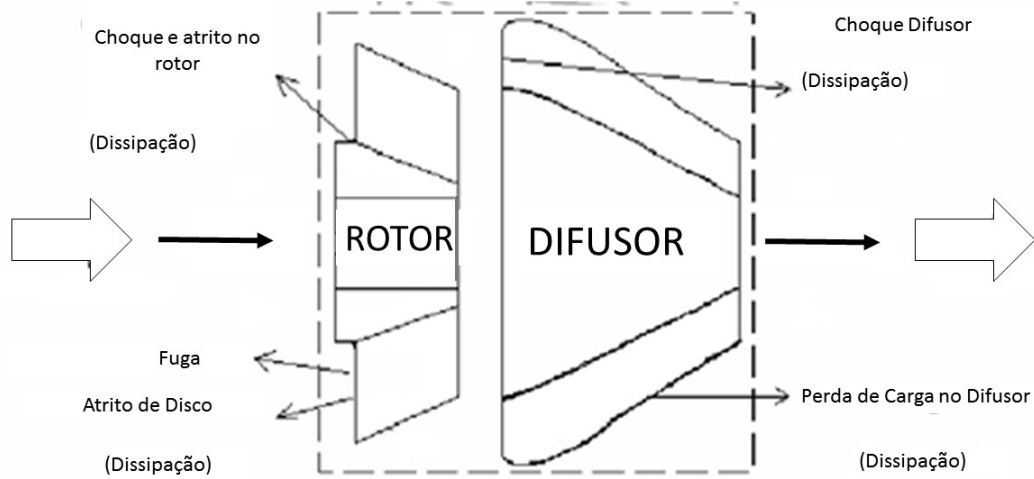


Figura 3.1: Estágio da bomba centrífuga de fluxo misto com indicações das perdas de carga [Amaral(2007)]

A seguir a definição física de cada um dos termos da equação da altura de elevação da bomba:

- **Head teórico:** Altura de elevação ideal ou teórica de uma bomba centrífuga sem considerar quaisquer perdas de carga no escoamento, formulação desenvolvida por Pfleiderer (1972) :

$$H_{teo} = C_a \left[\frac{U_2^2 - U_1^2}{g} - \frac{\omega Q}{2\pi g} \left(\frac{1}{b_2 \tan \beta_2} - \frac{1}{b_1 \tan \beta_1} \right) \right] \quad (3.2)$$

onde H_{teo} é a altura de elevação teórica $[m]$, C_a é o coeficiente de aletas, U é a velocidade periférica $[m/s]$, ω é a velocidade rotacional $[rpm]$, Q é a vazão $[m^3/s]$, b é a altura do canal do rotor $[m]$, β é o ângulo entre velocidade absoluta e direção tangencial das aletas $[graus]$ e os subscritos 1 e 2 se referam a entrea e saída respectivamente,

- **Perda de carga por atrito no rotor:** Definida como a perda linear causada nos limites das paredes no canal de passagem do fluido e sob os efeitos de viscosidade do fluido que considera o efeito da curvatura, efeito da forma retangular e efeito

da rotação do mesmo, formulação desenvolvida por Sun (2002) :

$$h_{atr} = \frac{f_{r\beta\omega} Q^2}{8gD_H \pi^2 b_m^2 \sin^2 \beta_h \cos \gamma} \frac{r_2 - r_1}{r_2 r_1} \quad (3.3)$$

onde $f_{r\beta\omega}$ é o fator de atrito para canal curvo, retangular e rotativo, γ é o ângulo entre aleta e plano horizontal [graus], D_H é o diâmetro hidráulico [m], r é o raio do canal do rotor [m] e β_h é a projeção do ângulo da aleta no plano horizontal.

- **Perda de carga por choques:** Sua ocorrência está relacionada a vazões fora da condição de máxima eficiência. Ocorre na entrada do rotor quando a vazão volumétrica descarregada pela bomba é diferente da vazão com componente velocidade tangente à aleta do rotor, formulação desenvolvida por Stepanoff (1959) :

$$h_{cho} = k_{cho} (Q - Q_{BEP})^2 \quad (3.4)$$

onde k_{cho} é a constante empírica para perdas de carga por choques e Q_{BEP} é a vazão no ponto de melhor eficiência [m/s].

- **Perda de carga por recirculação:** Uma perda volumétrica, ou seja, é a energia dissipada pela recirculação do fluido entre as regiões de entrada e saída do estágio devido às folgas naturais existentes entre os componentes internos da bomba. Por ser muito pequena é negligenciada pelas principais referências presentes na literatura, e aqui também será, a partir do modelo de Gullich (1999) e Amaral (2007):

$$h_{rec} \approx 0 \quad (3.5)$$

- **Perda de carga por atrito no difusor:** Perda hidráulica na entrada do difusor, que ocorre basicamente devido às perdas por atrito e às perdas geradas pela mudança de escoamento devido à não uniformidade da velocidade, formulação desenvolvida por Ito (1959):

$$h_{dif} = - \frac{(F_r F_\beta f) Q^2}{8gD_H \pi^2 b_m^2 \sin^3 \beta_m} \frac{(r_{3dif} - r_{2dif})}{(r_{3dif} r_{2dif})} \quad (3.6)$$

onde F_r é o fator de correção do fator de atrito para canal retangular, F_β é o fator de correção do fator de atrito para canal curvo, β_m é ângulo médio do escoamento entre entrada e saída do rotor [graus], b_m é altura média do canal do difusor [m], e r_{dif} é o raio do canal do difusor [m] e os subscritos 2 e 3 se referam a entrada e saída respectivamente.

- **Perda de carga por atrito de disco:** Energia dissipada externamente ao rotor, possivelmente a superposição de atritos com efeitos de irreversibilidade em mudanças de direção e velocidade no escoamento entre a saída do rotor e a entrada do difusor,

formulação desenvolvida por Thin (2008) :

$$h_{dis} = \frac{f_{dis}\rho\omega^3 (r_2)^5}{10^9 Q} \quad (3.7)$$

onde f_{dis} é o coeficiente de perda por atrito de disco e ρ é a massa específica do fluido $[kg/m^3]$.

É possível encontrar maiores detalhes sobre estudos de bombas centrífugas submersas, que envolvam características gerais, desempenho e perdas de carga nos livros de Gullich (2008), Tacacks (2009) e nas dissertações de Amaral (2007) e Vieira (2014).

3.3 Identificação de pacotes – assuntos

Apresentada a conceituação teórica, segue agora a identificação dos pacotes, que são por definição um agrupamento genérico de classes que fazem parte de um assunto e relacionam-se por um conceito comum. Um assunto é aquilo que é tratado ou abordado em uma discussão, em um estudo e é utilizado para orientar o leitor em um modelo amplo e complexo.

- **Propriedade do fluido:** Pacote contendo parâmetros que caracterizam o fluido, entre esses parâmetros está o valor de viscosidade informado pelo usuário do sistema. Essas características do fluido serão utilizadas no cálculo das perdas de carga devido a circulação de fluido na bomba.

- **Propriedades da bomba:** Pacote de parâmetros envolvendo a bomba centrífuga submersa, entre esses parâmetros está o valor da rotação da bomba escolhida pelo usuário. Esses dados serão utilizados para o cálculo das perdas de carga da bomba.

- **Perdas de carga:** É a perda de energia devido ao atrito provocado pela passagem de fluido. Utilizando os valores de bibliografia e dados fornecidos pelo usuário, os cálculos serão realizados;

- **Altura de elevação do BCS:** Cálculo objetivo do simulador. Utilizando as propriedades do fluido e da bomba, em segundo plano, calculará o head teórico da bomba, bem como as perdas de carga, para então calcular a Altura de Elevação do BCS.

- **Gráficos resultados:** Um software externo (**gnuplot**) irá gerar os gráficos a partir dos cálculos realizados pelo simulador.

- **Simulador:** Fará a ligação entre os demais pacotes e gerará os resultados de forma gráfica.

3.4 Diagrama de pacotes – assuntos

O diagrama de pacotes, Figura 3.2, mostra as dependências entre as diversas partes do sistema.

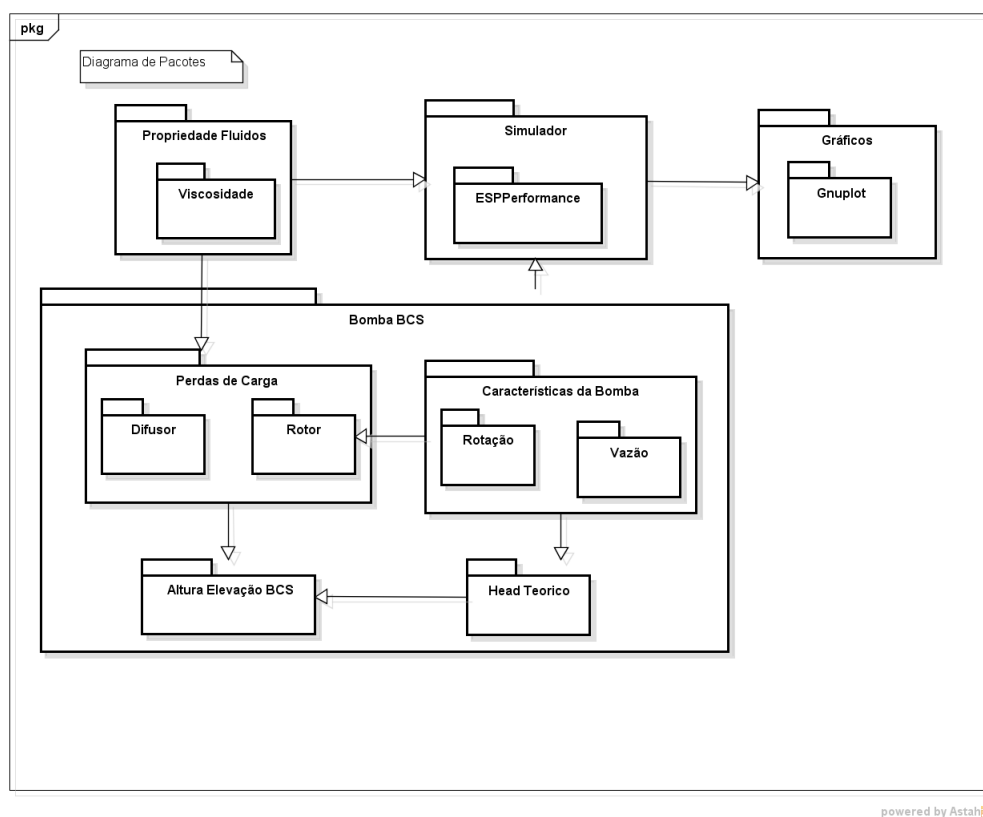


Figura 3.2: Diagrama de Pacotes

Capítulo 4

AOO – Análise Orientada a Objeto

A terceira etapa do desenvolvimento de um projeto de engenharia, no nosso caso um software aplicado a engenharia de petróleo, 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. O resultado da análise é um conjunto de diagramas que identificam os objetos e seus relacionamentos.

4.1 Diagramas de classes

O diagrama de classes é apresentado na Figura 4.1.

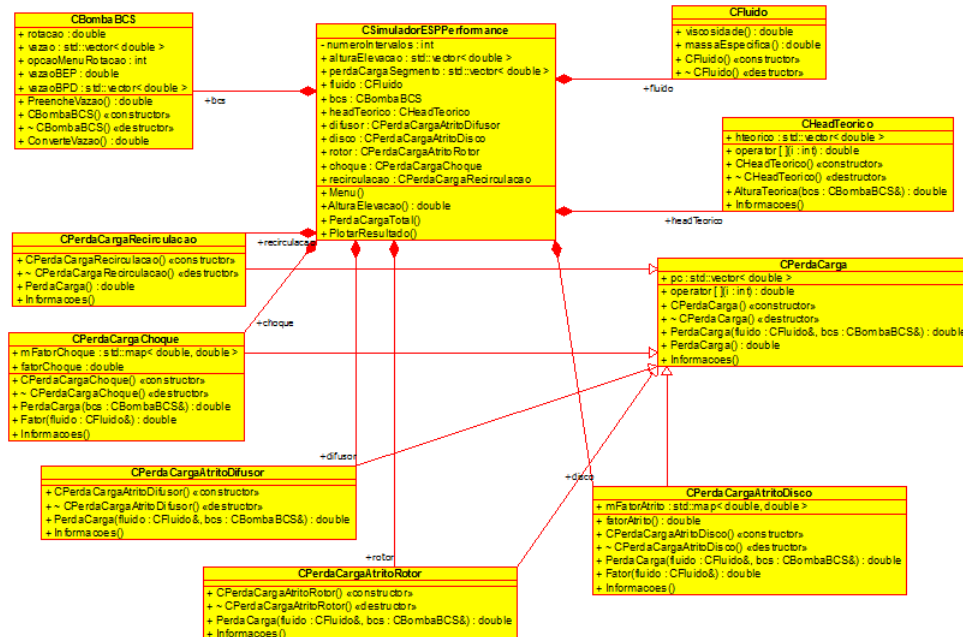


Figura 4.1: Diagrama de classes

4.1.1 Dicionário de classes

- Classe CFluido: representa as propriedades do fluido.
 - Atributo viscosidade: representa a viscosidade do fluido escolhido $[cP]$, podendo ser água ou glicerina como fluido de referência.
 - Atributo massaEspecificas: representa a massa específica do fluido escolhido $[Kg/m^3]$.
- Classe CBombaBCS: representa as características da bomba centrífuga submersa.
 - Atributo rotacao: representa a rotação da bomba $[rpm]$.
 - Atributo vazao: representa a vazão na saída da bomba em m^3/s .
 - Atributo vazaoBPD: vetor vazao na saída da bomba em bbl/d .
 - Atributo vazaoBEP: representa a vazão do ponto de maior eficiência para cada caso $[m^3/s]$
 - Método ConverteVazao: método que recebe os valores de vazão em m^3/s de arquivo externo e os converte para bbl/d .
- Classe CPerdaCarga: classe base de perdas de cargas gerais que recebe fluido e bomba BCS
 - Atributo pc: vetor das perdas de carga por seguimento $[m]$
 - Método PerdaCarga: método para calcular a perda de carga
 - Método Informacoes: informações sobre equações e parâmetros utilizados
- Classe CHeadTeorico: efetua o cálculo do *head* teórico.
 - Atributo hteorico: representa o *head* teórico $[m]$ calculado.
 - Método AlturaTeorica: calcula o *head* teórico.
 - Método Informacoes: método para informar as constantes presentes na formulação do *head* teórico
- Classe CPerdaAtritoRotor: efetua o cálculo da perda de carga por atrito do rotor
 - Método PerdaCarga: método para calcular a perda de carga por atrito no rotor.
 - Método Informacoes: método para informar as constantes presentes na formulação da perda por atrito no rotor
- Classe CPerdaAtritoDifusor: efetua o cálculo da perda de carga por atrito no difusor

- Método PerdaCarga: método para calcular a perda de carga por atrito no difusor.
- Método Informacoes: método para informar as constantes presentes na formulação da perda por atrito no difusor.
- Classe CPerdaChoque: efetua o cálculo da perda de carga por choques no rotor
 - Atributo fatorChoque: fator empírico para o calculo da perda de carga por choques no rotor;
 - Atributo mFatorChoque: map que associa valores de viscosidade a fatores empiricos de perda por choques
 - Método Fator: método para definir qual dos fatores de choque será aplicado ao calculo da perda de carga em função da viscosidade;
 - Método PerdaCarga: método para calcular a perda de carga por choques.
- Classe CPerdaRecirculacao: efetua o cálculo da perda de carga por recirculações
 - Método PerdaCarga: método para calcular a perda de carga por recirculações.
 - Método Informacoes: método para informar as constantes presentes na formulação da perda por recirculações.
- Classe CPerdaAtritoDisco: efetua o cálculo da perda de carga por atrito de disco.
 - Atributo fatorAtrito: fator empírico para o calculo da perda de carga por atrito de disco;
 - Atributo mFatorAtrito: map que associa valores de viscosidade a fatores empiricos de atrito de disco;
 - Método Fator: método para definir qual dos fatores de choque será aplicado ao calculo da perda de carga em função da viscosidade;
 - Método PerdaCarga: método para calcular a perda de carga por atrito de disco;
- Classe CSimuladorESPPerformance: classe simulador que gera os resultados, salva e plota os gráficos;
 - Atributo alturaElevacao: vetor altura de elevação do BCS $[m]$;
 - Atributo perdaCargaSegmento: vetor somatório de todas as perdas de carga $[m]$;
 - Atributo fluido: representa a classe CFluido;
 - Atributo bcs: representa a classe CBombaBCS;
 - Atributo headTeorico: representa a classe CHeadTeorico;

- Atributo difusor: representa a classe CPerdaCargaAtritoDifusor;
- Atributo disco: representa a classe CPerdaCargaAtritoDisco;
- Atributo rotor: representa a classe CPerdaCargaAtritoRotor;
- Atributo choque: representa a classe CPerdaCargaAtritoChoque;
- Atributo recirculacao: representa a classe CPerdaCargaRecirculacao;
- Método Menu: metodo que fornece opcoes viscosidade e rotacao para a simulacao e atribui os devidos parametros para os calculos de acordo com a combinacao escolhida;
- Método AlturaElevacao: método para cálculo e implementação da equação da altura de elevação de bombas centrífugas, bem como gerar resultados em tabela;
- Método PerdaCargaTotal: método para o cálculo do somatório de todas as perdas de carga;
- Método PlotarResultado: método para plotar os resultados obtidos (head teórico X vazão) e (altura de elevação X vazão).

4.2 Diagrama de seqüência – eventos e mensagens

O diagrama de seqüência enfatiza a troca de eventos e mensagens e sua ordem temporal. Contém informações sobre o fluxo de controle do programa. Estabelece o relacionamento dos atores (usuários e sistemas externos) com alguns objetos do sistema.

4.2.1 Diagrama de sequêcia geral

Veja o diagrama de seqüência na Figura 4.2.

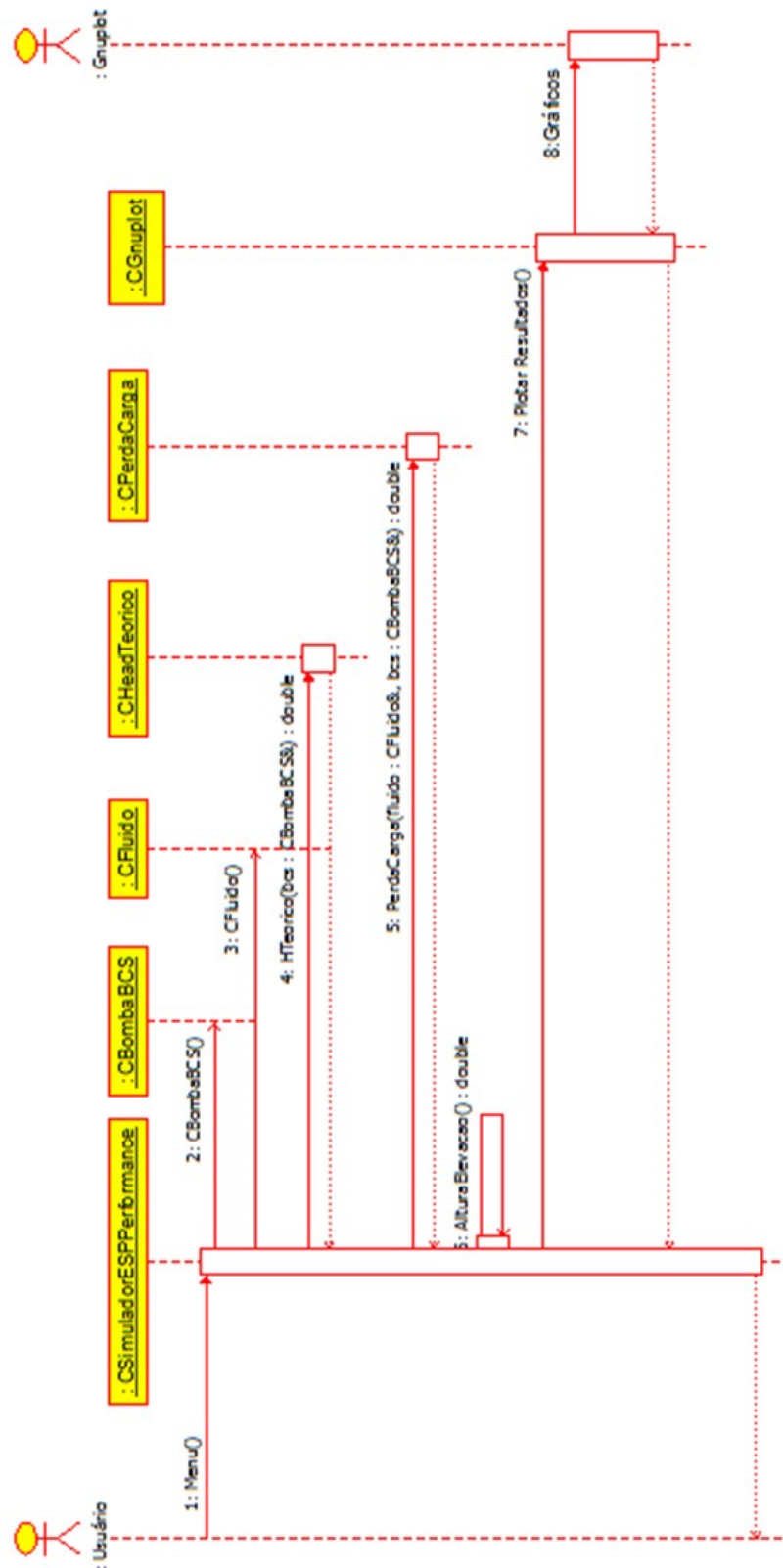


Figura 4.2: Diagrama de Sequência Geral

4.3 Diagrama de comunicação – colaboração

No diagrama de comunicação o foco é a interação e a troca de mensagens e dados entre os objetos.

Veja na Figura 4.3 o diagrama de comunicação, o mesmo ilustra a interação e a troca de mensagens e dados entre os objetos.

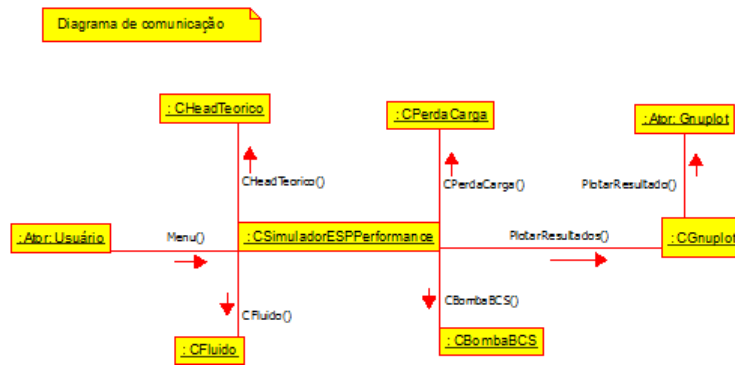


Figura 4.3: Diagrama de comunicação entre classes

4.4 Diagrama de máquina de estado

Um diagrama de máquina de estado representa os diversos estados que um determinado objeto assume e os eventos que ocorrem ao longo de sua vida ou mesmo ao longo de um processo (histórico do objeto). É usado para modelar aspectos dinâmicos do objeto.

Veja na Figura 4.4 o diagrama de máquina de estado.

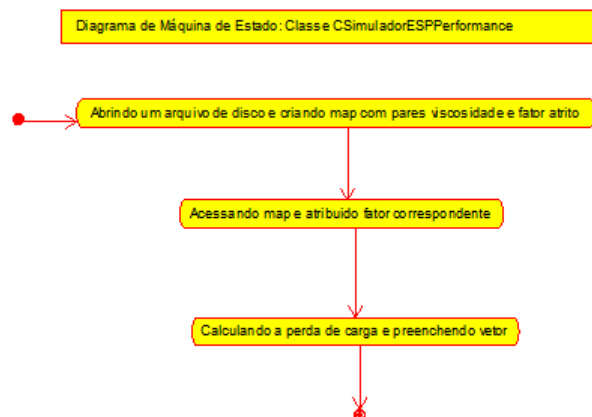


Figura 4.4: Diagrama de máquina de estado da classe CPerdaCargaAtritoDisco

4.5 Diagrama de atividades

Veja na Figura 4.5 o diagrama de atividades correspondente a atividade principal do programa que é calcular a altura de elevação da BCS.

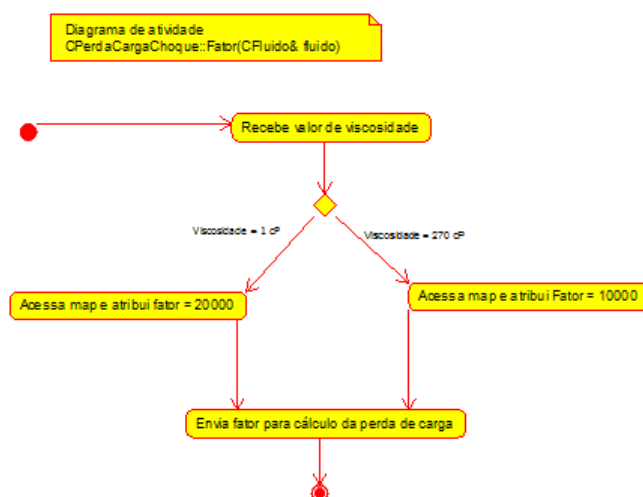


Figura 4.5: Diagrama de atividades: CPerdaCargaChoque:: Fator(CFluido& fluido)

Capítulo 5

Projeto

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

5.1 Projeto do sistema

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

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

O projeto do sistema é a estratégia de alto nível para resolver o problema e elaborar uma solução. Você deve se preocupar com itens como:

1. Protocolos

- Definição dos protocolos de comunicação entre os diversos elementos externos (como dispositivos).
 - O acesso ao programa externo `gnuplot` é feito usando fires.
- Definição dos protocolos de comunicação entre os diversos elementos internos (como objetos).
 - Os métodos são chamados seguindo uma ordem de conceitos de objetos do mundo real (bomba e fluido), depois os objetos das perdas de carga e altura de elevação.

- Definição do formato dos arquivos gerados pelo software.
 - Os arquivos de texto serão gerados em formato ASCII (não formatado)
 - Os arquivos de imagem (gráficos) são gerados com extensão png.

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 programa utilizará uma máquina computacional com HD, processador, teclado para a entrada de dados e o monitor para a saída de dados.
- Identificação da necessidade do uso de banco de dados. Implicam em modificações nos diagramas de atividades e de componentes.
 - O programa implementará cálculos de desempenho de bomba, baseados em conjunto de vazões pré-definidas.

3. Controle

- Identificação da necessidade de otimização.
- Identificação e definição de processamento sequencial linear de controle e das escalas de tempo.

4. Plataformas

- Identificação das estruturas arquitetônicas comuns.:
 - O programa será desenvolvido em linguagem de programação orientada a objeto C++, versão C++ 14.
- Identificação e definição das plataformas a serem suportadas: hardware, sistema operacional e linguagem de programação.
 - O programa será desenvolvido e testado em computador Intel 32/64 bits, com sistema operacional Windows 64 bits usando linguagem C++ orientada a objeto, e compilador gcc.
- Além disso, o simulador exige a instalação software externo Gnuplot.
- O simulador funciona em modo texto, executado através de uma janela do terminal

5.2 Projeto orientado a objeto – POO

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

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

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

- Novas classes e composições devem ser acrescentadas ao modelo.
 - Neste projeto foram feitas composições que podem ser visualizadas no diagrama de classe, Figura 4.1.

Efeitos do projeto no modelo dinâmico

- Os diagramas de sequência e comunicação foram modificados a medida que a implementação foi desenvolvida, Figuras 4.2 e 4.3.

Efeitos do projeto nos atributos

- Atributos novos podem ser adicionados a uma classe, como, por exemplo, atributos específicos de uma determinada linguagem de softwareção (acesso a disco, ponteiros, constantes e informações correlacionadas).
 - Foi adicionados novos atributos como exemplo vazaoBPD, que converte a unidade de vazao usada nos cálculos (m^3/s) para bbl/d.
 - Foi removido o atributo perdaCargaTotal, que realizava o somatório das perdas carga.

Efeitos do projeto nas heranças

- Reorganização da herança no diagrama de classes.

- Todas as heranças foram retiradas e substituídas por composições.

Efeitos do projeto nas otimizações

- Otimização do sistema.
 - A retirada das heranças fez com que o programa ao invés de criar o objeto da classe CPropriedadesEscoamento várias vezes, passou a criar uma vez apenas os objetos das classes de perdas de carga e *head* teórico.

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, executáveis, arquivos de disco, código-fonte. Veja na Figura 5.1 um exemplo de diagrama de componentes. A geração da biblioteca depende dos arquivos de classe de extensão .h e .cpp. O software executável a ser gerado depende da biblioteca gerada, dos arquivos da biblioteca, dos arquivos desta e do banco de dados.

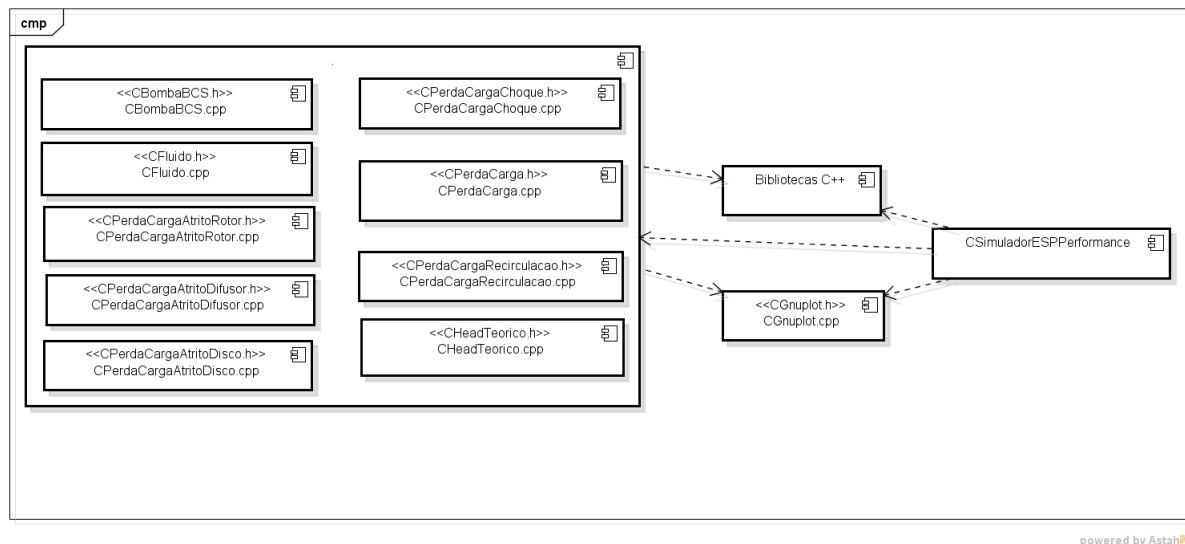


Figura 5.1: Diagrama de componentes

5.4 Diagrama de implantação

O diagrama de implantação é um diagrama que inclui relações entre o sistema e o hardware e deve incluir elementos necessários para que o sistema seja colocado em funcionamento:

computador, periféricos, processadores, dispositivos, nós, relacionamentos de dependência, associação, componentes, subsistemas, restrições e notas.

Veja na Figura 5.2 um exemplo de diagrama de implantação do software.

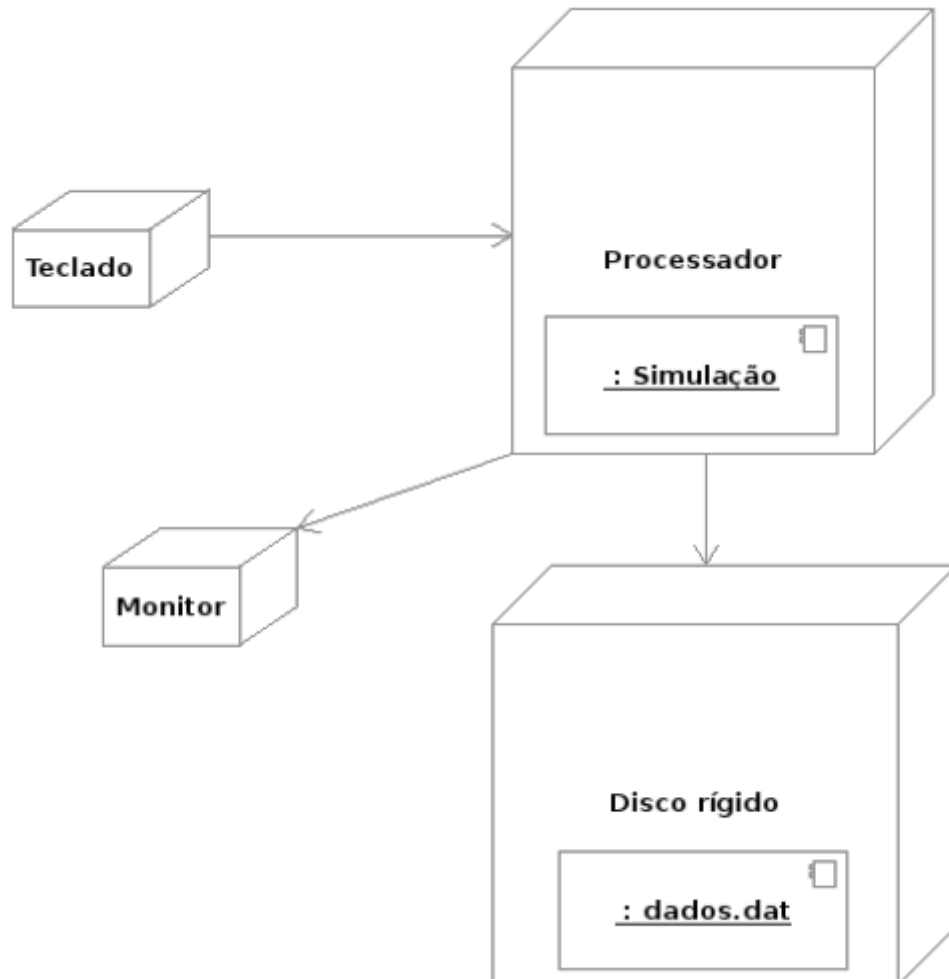


Figura 5.2: Diagrama de implantação.

Capítulo 6

Implementação

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

6.1 Código fonte

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

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

Listing 6.1: Arquivo de cabeçalho da classe CFluido.

```
1 #ifndef CFluido_H
2 #define CFluido_H
3
4 #include <iostream>
5 #include <vector>
6
7 ///Classe fluido com os atributos referentes ao escoamento que sao
   inerentes ao fluido, viscosidade e massa especifica
8
9 class CFluido
10 {
11     public:
12     ///atributos fluido:
13     double viscosidade{1};           ///< viscosidade do fluido (1 cP
        ou 270 cP)
14     double massaEspecific{997};      ///< tomadas como referencia as
        da agua (997 kg/m3) ou da glicerina (1260 kg/m3)
15
16     /// Metodos
17     CFluido(){};                    // Construtor
18     virtual ~CFluido(){};          // Destrutor
19 };
20 #endif
```

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

Listing 6.2: Arquivo de implementação da classe CFluido.

```
21 #include "CFluido.h"
```

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

Listing 6.3: Arquivo de cabeçalho da classe CBombaBCS.

```
24 #ifndef CBombaBCS_H
25 #define CBombaBCS_H
26
27 #include <iostream>
28 #include <vector>
29
30 ///Classe bomba BCS com os atributos referentes ao escoamento que sao
    inerentes ao BCS, rotacao e vazao, e que armazena as vazoes pre-
    definidas e faz as devidas conversoes de unidade
31
32 class CBombaBCS
33 {
34     public:
35     /// atributos bomba:
36     double rotacao; ///< rotacao da bomba (1800 rpm ou 3500
        rpm)
37     std::vector<double> vazao; ///<vetor de vazoes (m3/s) como dados de
        entrada tomadas baseadas nos dados experimentais de Amaral
        (2007)
38     double PreencheVazao ();
39     int opcaoMenuRotacao; ///< usuario escolhe qual das duas
        rotacoes disponiveis ira simular
40     double vazaoBEP; ///< vazao no ponto de melhor eficiencia
        para cada um dos 4 casos operacionais disponiveis
41     std::vector<double> vazaoBPD; ///< vazao em barris por dia
42
43     /// Metodos
44     CBombaBCS(){}; // Construtor
45     virtual ~CBombaBCS(){}; // Destrutor
46
47     double ConverteVazao (); ///< metodo para converter vazao
        de m3/s para bbl/d
48
49 };
50 #endif
```

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

Listing 6.4: Arquivo de implementação da classe CBombaBCS.

```
51 #include "CBombaBCS.h"
52
```

```

53#include <iostream>
54#include <cmath>
55#include <fstream>
56#include <vector>
57using namespace std;
58
59double CBombaBCS::ConverteVazao(){
60
61    ifstream fin ("vazao.txt"); ///< abre arquivo com valores de
        vazao
62
63    ///< preenche vetor vazao
64    double v;
65    vazao.clear(); ///< zera vetor de vazao
66
67    fin.ignore(256, '\n'); ///< ignora a primeira linha
68
69    while (!fin.eof()) {
70        fin >> v;
71        vazao.push_back(v);
72    }
73
74    fin.close(); ///< fecha arquivo de dados
75
76    vazaoBPD.resize(10);
77
78    for (int i = 0; i<vazao.size(); i++) {
79        vazaoBPD[i] = vazao[i] * 543456; ///< conversao de vazao
        para saida dos resultados
80        vazaoBPD.push_back(i);
81    }
82}

```

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

Listing 6.5: Arquivo de cabeçalho da classe CHeadTeorico.

```

86#ifndef CHeadTeorico_H
87#define CHeadTeorico_H
88
89#include <vector>
90
91#include "CBombaBCS.h"
92
93///< Classe que calcula o head teorico (ou altura de elevacao teorica)
94
95class CHeadTeorico
96{
97    public:
98        ///

```

```

99      std:: vector <double> hteorico;///altura de elevacao teorica (
        ou head teorico) do BCS (m)
100      double operator[](int i) { return hteorico[i]; } ;///<
        CHeadTeorico obj; obj[i]
101
102      public:
103          CHeadTeorico():hteorico(10){};                ///construtor
104          virtual ~CHeadTeorico(){};                    ///destrutor
105
106          double AlturaTeorica (CBombaBCS& bcs);///< metodo que calcula o
        head teorico
107          void Informacoes();///< metodo para informar as constantes
        presentes na formulacao do head teorico (Pfleiderer, 1972)
108 };
109 #endif // CHEADTEORICO_H

```

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

Listing 6.6: Arquivo de implementação da classe CHeadTeorico.

```

112 #include <vector>
113 #include <iostream>
114 #include <cmath>
115
116 #include "CHeadTeorico.h"
117 #include "CBombaBCS.h"
118
119 using namespace std;
120
121 void CHeadTeorico::Informacoes(){
122     cout << "
        =====
        n";
123     cout << "\n";
124     cout << "Para o calculo da altura de elevacao teorica foram considerados
        os seguintes parametros\n";
125     cout << "constantes inerentes ao modelo de BCS utilizado nas simulacoes
        (SLB-REDA-GN7000):\n";
126     cout << "angulo entre velocidade relativa e direcao tangencial na
        entrada do rotor = 28 graus\n";
127     cout << "angulo entre velocidade relativa e direcao tangencial na saida
        do rotor = 36 graus\n";
128     cout << "altura do canal na entrada do rotor = 0,0173 m\n";
129     cout << "altura do canal na saida do rotor = 0,0157 m\n";
130     cout << "raio da entrada do rotor = 0,0445 m\n";
131     cout << "raio da saida do rotor = 0,0255 m\n";
132     cout << "numero de aletas = 7\n";
133     cout << "fator de correcao para numero finito de aletas = 0,71\n";
134     cout << "aceleracao da gravidade = 9,81 m\n";
135     cout << "\n";

```

```

136 cout <<"
      =====
      "<<endl;
137 }
138
139 double CHeadTeorico:: AlturaTeorica (CBombaBCS& bcs) {
140     for (int i = 0; i< hteorico.size(); i++){
141         hteorico [i] = 0.07238*((pow(0.0445*bcs.rotacao, 2)) - (pow(0.0255*
            bcs.rotacao, 2))) + 0.2424*bcs.rotacao*bcs.vazao[i]; // formulação
            como função das variáveis
142     }
143 }

```

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

Listing 6.7: Arquivo de cabeçalho da classe CPerdaCarga.

```

144 #ifndef CPERDACARGA_H
145 #define CPERDACARGA_H
146
147 #include <vector>
148 #include "CFluido.h"
149 #include "CBombaBCS.h"
150
151 /// Classe base de perdas de cargas gerais que recebe fluido e bomba BCS
152 class CPerdaCarga
153 {
154     // private:
155     //     CFluido& fluido;
156     //     CBombaBCS& bcs;
157
158     public:
159         std::vector<double>pc; ///< Vetor com a perda de carga por
            seguimento.
160         double operator[](int i) { return pc[i]; } ///< retorna a perda
            de carga no segmento i
161
162     public:
163         /// Construtor, recebe fluido e bombabcs.
164         /// CPerdaCarga(CFluido& _fluido, CBombaBCS& _bcs) fluido(_fluido),
            bcs(_bcs){};
165         CPerdaCarga():pc(10){};
166
167         /// Destrutor
168         virtual ~CPerdaCarga(){};
169
170         /// Método para calcular a perda de carga
171         double PerdaCarga (CFluido& fluido, CBombaBCS& bcs) {return
            0.0;};
172

```

```

173      /// Método para calcular a perda de carga
174      double PerdaCarga () {return 0.0;};
175
176      // Informacoes sobre equacoes e parametros utilizados
177      void Informacoes() {};
178 };
179 #endif // CPERDACARGA_H

```

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

Listing 6.8: Arquivo de implementação da classe CPerdaCarga.

```

180
181 #include "CPerdaCarga.h"

```

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

Listing 6.9: Arquivo de cabeçalho da classe CPerdaCargaAtritoRotor.

```

182 #ifndef CPERDACARGAATRITOROTOR_H
183 #define CPERDACARGAATRITOROTOR_H
184
185 #include "CPerdaCarga.h"
186
187 /// Classe que realiza o calculo da perda de carga por atrito no rotor.
188
189 class CPerdaCargaAtritoRotor: public CPerdaCarga
190 {
191     public:
192         CPerdaCargaAtritoRotor(){};
193         virtual ~CPerdaCargaAtritoRotor(){};
194
195         /// Metodo para calcular a perda de carga
196         double PerdaCarga (CFluido& fluido, CBombaBCS& bcs);
197
198         void Informacoes (); ///< metodo para informar as constantes
            presentes na formulacao da perda por atrito no rotor (Sun e
            Prado, 2003)
199 };
200 #endif // CPERDACARGAATRITOROTOR_H

```

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

Listing 6.10: Arquivo de implementação da classe CPerdaCargaAtritoRotor.

```

201 #include <vector>
202 #include <iostream>
203 #include <cmath>
204
205 #include "CPerdaCargaAtritoRotor.h"
206
207 using namespace std;

```


[illegible]

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

Listing 6.11: Arquivo de cabeçalho da classe CPerdaCargaChoque.

```
231 #ifndef CPERDACARGACHOQUE_H
232 #define CPERDACARGACHOQUE_H
233
234 #include "CPerdaCarga.h"
235 #include <map>
236
237 /// Classe que realiza o calculo da perda de carga por atrito de choque
238
239 class CPerdaCargaChoque: public CPerdaCarga
240 {
```

```

241     public:
242         CPerdaCargaChoque(){};
243         virtual ~CPerdaCargaChoque(){};
244
245         std::map<double,double> mFatorChoque;///map que associa a
            viscosidade a fator empirico de perda por choque, a partir de
            tabela em arquivo externo
246
247         double fatorChoque;///fator empirico para o calculo da perda
            de carga por choques no rotor
248
249         double PerdaCarga (CBombaBCS& bcs);///metodo para calcular a
            perda de carga por choques no rotor
250         double Fator(CFluido& fluido);///metodo para definir qual dos
            fatores de choque sera aplicado ao calculo da perda de carga
            em funcao da viscosidade
251     };
252 #endif // CPERDACARGACHOQUE_H

```

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

Listing 6.12: Arquivo de implementação da classe CPerdaCargaChoque.

```

253 #include <vector>
254 #include <iostream>
255 #include <cmath>
256 #include <fstream>
257
258 #include "CPerdaCargaChoque.h"
259
260 using namespace std;
261
262 double CPerdaCargaChoque::Fator(CFluido& fluido){
263
264
265     ifstream in ("fatorChoque.txt");
266     in.ignore(256, '\n');
267     while (!in.eof()){
268         double viscosidade;
269         double fatCho;
270         in>> viscosidade;
271         in>> fatCho;
272         mFatorChoque.insert(make_pair(viscosidade, fatCho));
273     }
274     in.close();
275
276     if (mFatorChoque.find(fluido.viscosidade) != mFatorChoque.end()) {
277         map<double,double>::const_iterator iter;
278         iter = mFatorChoque.find(fluido.viscosidade);
279         fatorChoque=iter->second;}

```

```

280
281     cout<<"\n";
282     cout<<"0_fator_de_perda_por_choque_empirico_e:"<<fatorChoque;
283     cout<<"\n";
284
285     }
286
287 double CPerdaCargaChoque::PerdaCarga(CBombaBCS& bcs){
288     for (int i = 0; i < pc.size(); i++){
289         pc [i] = fatorChoque*(pow(bcs.vazao[i]- bcs.vazaoBEP,2));
290     }
291
292 }

```

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

Listing 6.13: Arquivo de cabeçalho da classe CPerdaCargaRecirculacao.

```

295 #ifndef CPERDACARGARECIRCULACAO_H
296 #define CPERDACARGARECIRCULACAO_H
297
298 #include <vector>
299 #include "CPerdaCarga.h"
300
301 /// Classe que realiza o calculo da perda de carga por recirculacao.
302
303 class CPerdaCargaRecirculacao : public CPerdaCarga
304 {
305     public:
306         CPerdaCargaRecirculacao(){};
307         virtual ~CPerdaCargaRecirculacao(){};
308
309         double PerdaCarga();///< metodo para calcular a perda de carga
por recirculacoes no rotor
310
311         void Informacoes();
312 };
313 #endif // CPERDACARGARECIRCULACAO_H

```

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

Listing 6.14: Arquivo de implementação da classe CPerdaCargaRecirculacao.

```

316 #include <iostream>
317 #include "CPerdaCargaRecirculacao.h"
318
319 using namespace std;
320
321 double CPerdaCargaRecirculacao::PerdaCarga(){
322     for (int i = 0; i < pc.size(); i++){
323         pc[i]=0.0;

```

```

324     }
325 }
326
327 void CPerdaCargaRecirculacao::Informacoes() {
328     cout
329     << "
330         =====
331         n"
332     << "Para a perda de carga por recirculacao e usada a definicao de
333         Gullich (1999), que considera essa perda em\n"
334     << "escoamentos em bomba centrifuga desprezivel.\n"
335     << "
336         =====
337     " << endl;
338 }

```

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

Listing 6.15: Arquivo de cabeçalho da classe CPerdaCargaAtritoDifusor.

```

334 #ifndef CPERDACARGAATRITODIFUSOR_H
335 #define CPERDACARGAATRITODIFUSOR_H
336
337 #include "CPerdaCarga.h"
338
339 /// Classe que realiza o calculo da perda de carga por atrito no difusor
340 .
341
342 class CPerdaCargaAtritoDifusor : public CPerdaCarga
343 {
344     public:
345     CPerdaCargaAtritoDifusor(){};
346     virtual ~CPerdaCargaAtritoDifusor(){};
347
348     /// Metodo para calcular a perda de carga
349     double PerdaCarga (CFluido& fluido, CBombaBCS& bcs);
350
351     void Informacoes(); ///< metodo para informar as constantes
352     presentes na formulacao da perda por atrito no difusor (Ito,
353     1959)
354 };
355
356 #endif // CPERDACARGAATRITODIFUSOR_H

```

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

Listing 6.16: Arquivo de implementação da classe CPerdaCargaAtritoDifusor.

```

354 #include <vector>
355 #include <iostream>
356 #include <cmath>

```

```

357
358 #include "CPerdaCargaAtritoDifusor.h"
359
360 using namespace std;
361
362 void CPerdaCargaAtritoDifusor::Informacoes(){
363     cout <<"
        =====
        n";
364     cout << "
        =====
        =====
        =====\n";
365     cout<<"Para o calculo da perda de carga por atrito no rotor foram
        considerados os seguintes parametros\n";
366     cout<<"constantes inerentes ao modelo de BCS utilizado nas
        simulacoes (SLB-REDA-GN7000):\n";
367     cout<<"raio medio da entrada do difusor = 0,04775 m\n";
368     cout<<"raio medio da saida do difusor = 0,0290 m\n";
369     cout<<"diametro hidraulico da secao do difusor = 0,02843 m\n";
370     cout<<"altura media do canal do difusor = 0,01425 m\n";
371     cout<<"angulo relativo medio do escoamento entre entrada e saida do
        difusor = 32 graus\n";
372     cout << "
        =====
        =====\n";
373     cout <<"
        =====
        "<<endl;
374 }
375
376 double CPerdaCargaAtritoDifusor::PerdaCarga(CFluido& fluido, CBombaBCS&
        bcs){
377     for (int i = 0; i<pc.size(); i++){
378         pc[i] = 2752 * (pow (bcs.vazao[i], 2))*(pow((8.58 *fluido.
            massaEspecific *bcs.vazao[i]) / (fluido.viscosidade * pow
            (10,-3)), -0.611));
379     }
380 }

```

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

Listing 6.17: Arquivo de cabeçalho da classe CPerdaCargaAtritoDisco.

```

381 #ifndef CPERDACARGAATRITODISCO_H
382 #define CPERDACARGAATRITODISCO_H
383
384 #include <map>
385 #include "CPerdaCarga.h"
386
387 /// Classe que realiza o calculo da perda de carga por atrito de disco.
388

```

```

389 class CPerdaCargaAtritoDisco: public CPerdaCarga
390 {
391     public:
392         double fatorAtrito;///fator empirico para o calculo da perda
            de carga por atrito de disco
393
394     public:
395         CPerdaCargaAtritoDisco(){};
396         virtual ~CPerdaCargaAtritoDisco(){};
397
398         std::map<double,double> mFatorAtrito; ///map que associa a
            viscosidade a fator empirico de perda por atrito de disco, a
            partir de tabela em arquivo externo
399
400         double PerdaCarga (CFluido& fluido, CBombaBCS& bcs);///metodo
            para calcular a perda de carga por atrito de disco
401         double Fator(CFluido& fluido);///metodo para definir qual dos
            fatores de atrito será aplicado ao calculo da perda de carga
            em funcao da viscosidade
402
403
404
405 };
406 #endif // CPERDACARGAATRITODISCO_H

```

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

Listing 6.18: Arquivo de implementação da classe CPerdaCargaAtritoDisco.

```

407 #include <vector>
408 #include <iostream>
409 #include <cmath>
410 #include <fstream>
411
412 #include "CPerdaCargaAtritoDisco.h"
413
414 using namespace std;
415
416 double CPerdaCargaAtritoDisco::Fator(CFluido& fluido){
417
418     ifstream in ("fatorDisco.txt");
419     in.ignore(256, '\n');
420     while (!in.eof()){
421         double viscosidade;
422         double fatDis;
423         in>> viscosidade;
424         in>> fatDis;
425         mFatorAtrito.insert(make_pair(viscosidade, fatDis));
426     }
427     in.close();

```

```

428     if (mFatorAtrito.find(fluido.viscosidade) != mFatorAtrito.end()) {
429         map<double, double>::const_iterator iter;
430         iter = mFatorAtrito.find(fluido.viscosidade);
431         fatorAtrito=iter->second;}
432
433         cout<<"\n";
434         cout<<"0_fator_de_atrito_de_disco_empirico_para_esta_opcao_e:"<<
            fatorAtrito;
435         cout<<"\n";
436
437     }
438
439
440 double CPerdaCargaAtritoDisco::PerdaCarga (CFluido& fluido, CBombaBCS&
        bcs){
441     for (int i = 0; i < pc.size(); i++){
442         pc[i] = ((4.5917 * (pow (10,-17))) * fatorAtrito *fluido.
            massaEspecific * (pow (bcs.rotacao,3)))/ (bcs.vazao[i]);
443         //pc.push_back(i);
444     }
445 }

```

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

Listing 6.19: Arquivo de cabeçalho da classe CSimuladorESPPerformance.

```

446 #ifndef CSIMULADORESPPERFORMANCE_H
447 #define CSIMULADORESPPERFORMANCE_H
448
449 #include <vector>
450 #include <iomanip>
451 #include <fstream>
452 #include <iostream>
453
454 #include "CFluido.h"
455 #include "CBombaBCS.h"
456 #include "CHeadTeorico.h"
457 #include "CPerdaCargaAtritoDifusor.h"
458 #include "CPerdaCargaAtritoDisco.h"
459 #include "CPerdaCargaAtritoRotor.h"
460 #include "CPerdaCargaChoque.h"
461 #include "CPerdaCargaRecirculacao.h"
462 #include "CGnuplot.h"
463
464 /// Classe simulador que gera os resultados, salva e plota os graficos
465
466 class CSimuladorESPPerformance
467 {
468 private:
469

```

```

470     int numeroIntervalos; /// Numero de intervalos usados nos calculos
        das perdas de carga.
471 public:
472     std::vector<double> alturaElevacao; /// altura de elevacao do
        BCS
473     std::vector<double> perdaCargaSegmento; /// perda de carga
        segmento = soma de todas as perdas de carga
474     CFluido fluido;
475     CBombaBCS bcs;
476     CHeadTeorico headTeorico;
477     CPerdaCargaAtritoDifusor difusor;
478     CPerdaCargaAtritoDisco disco;
479     CPerdaCargaAtritoRotor rotor;
480     CPerdaCargaChoque choque;
481     CPerdaCargaRecirculacao recirculacao;
482
483 public:
484     CSimuladorESPPerformance(){};
485     virtual ~CSimuladorESPPerformance(){};
486
487     void Menu(); /// metodo que fornece opcoes viscosidade e
        rotacao para a simulacao e atribui os devidos parametros para
        os calculos de acordo com a combinacao escolhida
488     double AlturaElevacao (); /// metodo para calculo e
        implementacao da equacao da altura de elevacao de bombas
        centrifugas, bem como gerar resultados em tabela
489     void PerdaCargaTotal(); /// metodo para o calculo do somatorio
        de todas as perdas de carga
490     void PlotarResultado(); /// metodo para plotar os resultados
        obtidos (Head teorico X vazao) e (altura de elevacao X vazao)
491
492 };
493 #endif // CSIMULADORESPPERFORMANCE_H

```

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

Listing 6.20: Arquivo de implementação da classe CSimuladorESPPerformance.

```

495 #include <iostream>
496 #include <vector>
497 #include <iomanip>
498 #include <fstream>
499 #include <cmath>
500
501 #include "CSimuladorESPPerformance.h"
502 // #include "CGnuplot.h"
503 // #include "CFluido.h"
504 // #include "CBombaBCS.h"
505 // #include "CHeadTeorico.h"
506 // #include "CPerdaCargaAtritoDifusor.h"

```



```

507 // #include "CPerdaCargaAtritoDisco.h"
508 // #include "CPerdaCargaAtritoRotor.h"
509 // #include "CPerdaCargaChoque.h"
510 // #include "CPerdaCargaRecirculacao.h"
511
512 using namespace std;
513
514
515 void CSimuladorESPPerformance::Menu(){
516 cout << "\n
=====
";
517 cout << "\n=====ESP
PERFORMANCE===== ";
518 cout << "\n=====SIMULADOR SIMPLIFICADO DE DESEMPENHO DE
BCS BASEADO NAS ALTURAS DE ELEVACAO===== ";
519 cout << "\n
=====
n";
520 cout << "\nDeseja simular com fluido de viscosidade igual a: 1 cP (
Digite 1) ou 270 cP (Digite 2)?\n";
521 int opcaoMenuViscosidade;
522 cin >> opcaoMenuViscosidade; cin.get();
523 cout << "\nDeseja simular com bomba operando com rotacao de: 3500 rpm (
Digite 1) ou 1800 rpm (Digite 2)?\n";
524 int opcaoMenuRotacao;
525 cin >> opcaoMenuRotacao;
526 cin.get();
527 cout << "\n
=====
"<<endl;
528
529 numeroIntervalos = 10;
530 alturaElevacao.resize(numeroIntervalos);
531 perdaCargaSegmento.resize(numeroIntervalos);
532
533 if (opcaoMenuViscosidade == 1 and opcaoMenuRotacao == 1){
534     fluido.viscosidade = 1;    ///< Viscosidade em centiPoise (cP)
535     fluido.massaEspecificas = 997; ///< massa especifica em Kg/m3
536     bcs.rotacao = (3500*2*M_PI)/60;    ///< Em rotacao por minuto (
rpm), e necessario converter para radiano por segundo (rad/s)
537     bcs.vazaoBEP = 0.01394 ;    ///< vazao em m3/s
538     cout << "A simulacao ocorrera com fluido de massa especifica
igual a 997 Kg/m3"<<endl;
539 }
540
541 else if (opcaoMenuViscosidade == 2 and opcaoMenuRotacao == 1){
542     fluido.viscosidade = 270;    ///< Viscosidade em centiPoise (cP)

```

```

543     fluido.massaEspecifica = 1260; ///< massa especifica em Kg/m3
544     bcs.rotacao = (3500*2*M_PI)/60;    ///< Em rotacao por minuto (
        rpm), e necessario converter para radiano por segundo (rad/s)
545     bcs.vazaoBEP = 0.0107 ;    ///< vazao em m3/s
546     cout << "A simulacao ocorrera com fluido de massa especifica
        igual a 1260 Kg/m3"<<endl;
547 }
548
549 else if (opcaoMenuViscosidade == 1 and opcaoMenuRotacao == 2){
550     fluido.viscosidade = 1;    ///< Viscosidade em centiPoise (cP)
551     fluido.massaEspecifica = 997; ///< massa especifica em Kg/m3
552     bcs.rotacao = (1800*2*M_PI)/60;    ///< Em rotacao por minuto (
        rpm), e necessario converter para radiano por segundo (rad/s)
553     bcs.vazaoBEP = 0.0083 ;    ///< vazao em m3/s
554     cout << "A simulacao ocorrera com fluido de massa especifica
        igual a 997 Kg/m3"<<endl;;
555 }
556
557 else if (opcaoMenuViscosidade == 2 and opcaoMenuRotacao == 2){
558     fluido.viscosidade = 270;    ///< Viscosidade em centiPoise (cP)
559     fluido.massaEspecifica = 1260; ///< massa especifica em Kg/m3
560     bcs.rotacao = (1800*2*M_PI)/60;    ///< Em rotacao por minuto (
        rpm), e necessario converter para radiano por segundo (rad/s)
561     bcs.vazaoBEP = 0.00475 ;    ///< vazao em m3/s
562     cout << "A simulacao ocorrera com fluido de massa especifica
        igual a 1260 Kg/m3"<<endl;
563 }
564 }
565
566 void CSimuladorESPPerformance::PerdaCargaTotal() {
567     for (int i = 0; i< perdaCargaSegmento.size(); i++){
568
569         perdaCargaSegmento[i] = rotor[i] + choque[i] + recirculacao[i] +
            difusor[i] + disco[i];}
570 }
571
572 double CSimuladorESPPerformance :: AlturaElevacao(){
573
574     ///< Mostra menu que solicita dados fluido e bomba; seta
        numeroIntervalos
575     Menu();
576
577     ///< Converte a vazão
578     bcs.ConverteVazao();
579
580     ///< Calcula head teorico e informa constantes
581     headTeorico.AlturaTeorica (bcs);
582     headTeorico.Informacoes();

```

```

583
584     /// Calcula perda de carga difusor e informa constantes
585     difusor.PerdaCarga(fluido, bcs);
586     difusor.Informacoes();
587
588     /// Define fator de atrico e calcula perda de carga
589     disco.Fator(fluido);
590     disco.PerdaCarga(fluido, bcs);
591
592     /// Calcula perda de carga rotor e informa constantes
593     rotor.PerdaCarga(fluido, bcs);
594     rotor.Informacoes();
595
596     /// Calcula fator choque e perda de carga choque
597     choque.Fator(fluido);
598     choque.PerdaCarga (bcs);
599
600     recirculacao.Informacoes();
601     recirculacao.PerdaCarga();
602
603     /// Calcula a perda de carga total
604     PerdaCargaTotal();
605
606     cout<<"Vazao_(bbl/d)"<<setw(15)<<"Head_teorico_(m)"<<setw(15)<<"
        Altura_de_elevacao_(m)\n";
607
608     for (int i = 0; i<10; i++){
609         alturaElevacao[i] = headTeorico[i] - perdaCargaSegmento[i];}
610
611     fstream fout; ///< Salvar resultados da tabela em arquivo
612     fout.open("resultados.txt", ios::out);
613     fout.clear();
614     fout << "Vazao_(bbl/d)" <<setw(13) << "Head_teorico_(m)" <<setw(13)
        << "Altura_de_elevacao_(m)\n";
615     for (int i = 0; i<10; i++){
616
617         fout <<setw(15)<<bcs.vazaoBPD[i]<<setw(15)<< headTeorico[i]<<setw
            (15)<<alturaElevacao[i]<<setw(15)<< "\n";}
618     fout.close();
619
620     for (int i = 0; i<10; i++){
621
622         cout << bcs.vazaoBPD[i] << setw(16) << headTeorico[i] << setw(16) <<
            alturaElevacao[i]<<"\n";}
623 }
624
625
626

```

```

627 void CSimuladorESPPerformance::PlotarResultado(){
628
629     static CGnuplot plotar1;
630     static CGnuplot plotar2;
631
632     plotar1 << "set term png\n";
633     plotar1 << "set out \"HeadTeorico_x_Vazao.png\"\n";
634     plotar1.set_style("lines");
635     plotar1.set_title("HeadTeorico_x_Vazao");
636     plotar1.set_xlabel("vazao (bbl/d)");
637     plotar1.set_ylabel("Head Teorico (m)");
638     plotar1.Grid();
639     plotar1.plotfile_xy("resultados.txt", 1,2);
640
641     plotar2 << "set term png\n";
642     plotar2 << "set out \"AlturaElevacao_x_Vazao.png\"\n";
643     plotar2.set_style("lines");
644     plotar2.set_title("AlturaElevacao_x_Vazao");
645     plotar2.set_xlabel("vazao (bbl/d)");
646     plotar2.set_ylabel("Altura Elevacao (m)");
647     plotar2.Grid();
648     plotar2.plotfile_xy("resultados.txt", 1,3);
649 }

```

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

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

```

650 #include <iostream>
651 #include "CSimuladorESPPerformance.h"
652
653 /* run this program using the console pauser or add your own getch,
    system("pause") or input loop */
654 int main(int argc, char** argv) {
655     CSimuladorESPPerformance simulador;
656     char continuar{'s'};
657     do
658     {
659         simulador.AlturaElevacao();
660         simulador.PlotarResultado();
661         std::cout << "\n\n aContinuar?(s)(n): ";
662         std::cin >> continuar;
663     } while(continuar == 's' or continuar == 'S');
664     return 0;
665 }

```

Capítulo 7

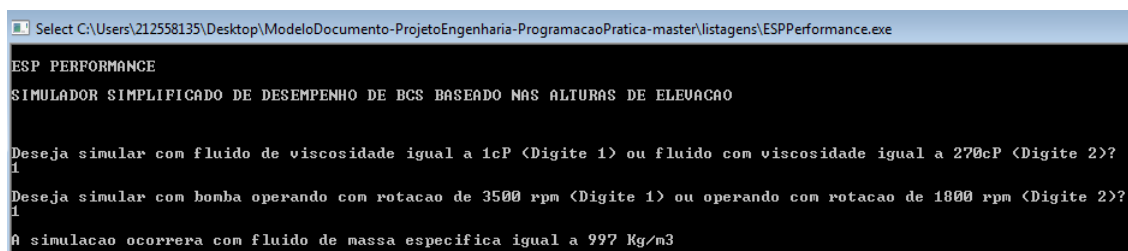
Teste

Todo projeto de engenharia passa por uma etapa de testes. Neste capítulo apresentamos alguns testes do software desenvolvido.

7.1 Teste 1: Recebendo dados do usuário via terminal

O programa interage com o usuário solicitando que o mesmo escolha entre as opções disponíveis de viscosidade de fluido e rotação do BCS. O teste é realizado rodando o programa via terminal de comando, de forma a verificar se o programa está recebendo a entrada do usuário. Existem quatro combinações a depender das opções escolhidas pelo usuário e a partir destas combinações são definidos a massa específica do fluido e os fatores de perda de carga por atrito de disco e choque

Na terminal de comando é solicitado ao usuário as opções e a seguir definidos a massa específica do fluido e os fatores e efetuados os devidos cálculos. Veja Figura 7.1, no caso em que o usuário escolhe a opção 1 de viscosidade (1 cP) e a opção 1 de rotação (3500 rpm)



```
Select C:\Users\212558135\Desktop\ModeloDocumento-ProjetoEngenharia-ProgramacaoPratica-master\listagens\ESPPerformance.exe

ESP PERFORMANCE
SIMULADOR SIMPLIFICADO DE DESEMPENHO DE BCS BASEADO NAS ALTURAS DE ELEUACAO

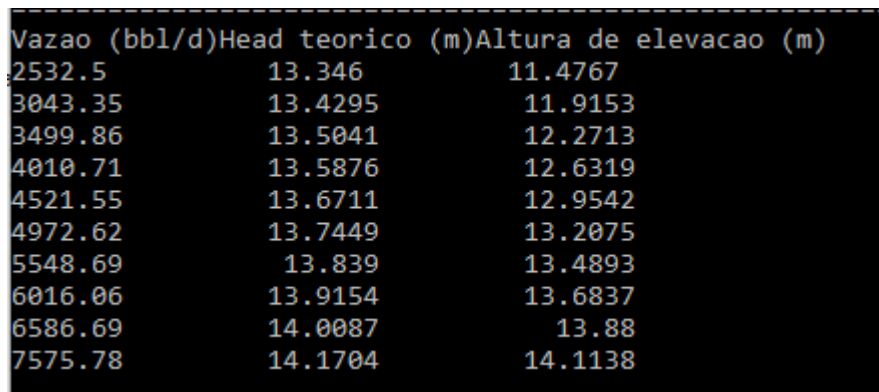
Deseja simular com fluido de viscosidade igual a 1cP <Digite 1> ou fluido com viscosidade igual a 270cP <Digite 2>?
1
Deseja simular com bomba operando com rotacao de 3500 rpm <Digite 1> ou operando com rotacao de 1800 rpm <Digite 2>?
1
A simulacao ocorrera com fluido de massa especifica igual a 997 Kg/m3
```

Figura 7.1: Tela do programa mostrando: inserção das opções de viscosidade e rotação e definição da massa específica do fluido.

7.2 Teste 2: Cálculos do *head* teórico e alturas de elevação

O programa deve calcular o *head* teórico, as perdas de carga a fim de calcular a altura de elevação final. O teste é para verificar se os cálculos estão sendo realizados devidamente, sendo mostrados na tela, no terminal e também se estão sendo gravados em arquivo txt. A validação é feita a partir de conferência dos valores gerados, se estão coerentes com o esperado e se estão sendo salvos no arquivo externo.

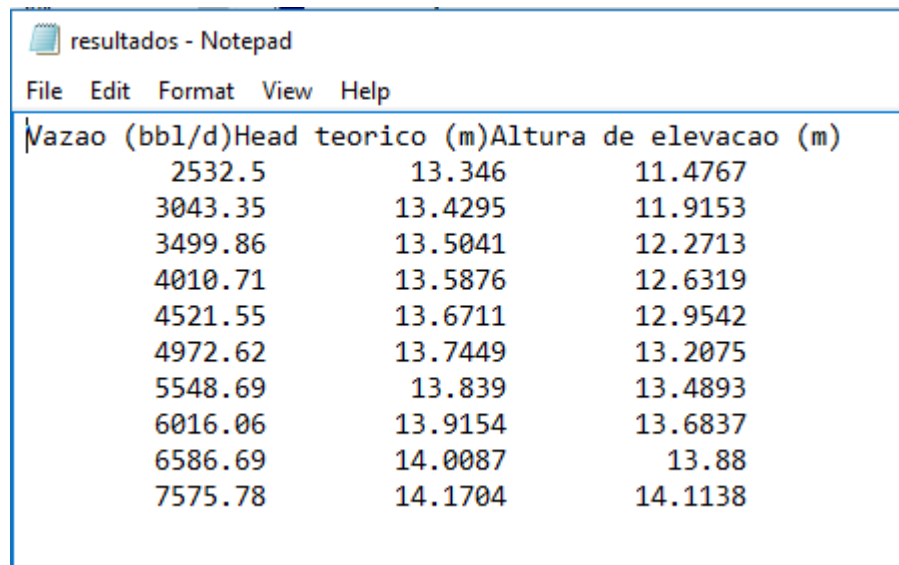
Na figura 7.2 são mostradas as saídas dos resultados dos cálculos de *head* teórico (m) e alturas de elevação (m) dentro da faixa de vazões (bbl/d) determinada para simulações, em terminal para uma combinação de rotação (3500 rpm) e viscosidade (1 cP).



Vazao (bbl/d)	Head teorico (m)	Altura de elevacao (m)
2532.5	13.346	11.4767
3043.35	13.4295	11.9153
3499.86	13.5041	12.2713
4010.71	13.5876	12.6319
4521.55	13.6711	12.9542
4972.62	13.7449	13.2075
5548.69	13.839	13.4893
6016.06	13.9154	13.6837
6586.69	14.0087	13.88
7575.78	14.1704	14.1138

Figura 7.2: Tela do programa mostrando os resultados dos cálculos do *head* teórico e alturas de elevação, para a combinação de rotação (1800 rpm) e viscosidade (1 cP).

Na figura 7.3 são mostradas as saídas dos resultados dos cálculos de *head* teórico (m) e alturas de elevação (m) dentro da faixa de vazões (bbl/d) determinada para simulações, sendo salvas em arquivo de disco (resultados.txt) para para uma combinação de rotação (3500 rpm) e viscosidade (1 cP).



Vazao (bbl/d)	Head teorico (m)	Altura de elevacao (m)
2532.5	13.346	11.4767
3043.35	13.4295	11.9153
3499.86	13.5041	12.2713
4010.71	13.5876	12.6319
4521.55	13.6711	12.9542
4972.62	13.7449	13.2075
5548.69	13.839	13.4893
6016.06	13.9154	13.6837
6586.69	14.0087	13.88
7575.78	14.1704	14.1138

Figura 7.3: Arquivo resultados.txt onde são salvos os resultados dos cálculos do *head* teórico e alturas de elevação, para a combinação de rotação (3500 rpm) e viscosidade (1 cP).

7.3 Teste 3: Plotagem dos gráficos

O programa deve gerar gráficos dos resultados dos cálculos do *head* teórico e altura de elevação. Os gráficos devem apresentar no eixo y os valores de altura de elevação ou *head* teórico em metros e no eixo x as vazões em barris por dia. São apresentados as curvas de *head* teórico e altura de elevação em cada gráfico.

Na figura 7.4 são mostrados os os gráficos plotados das curvas de *head* teórico (m) e alturas de elevação (m) dentro da faixa de vazões (bbl/d) determinada para simulações, para uma combinação de rotação (1800 rpm) e viscosidade (270 cP).

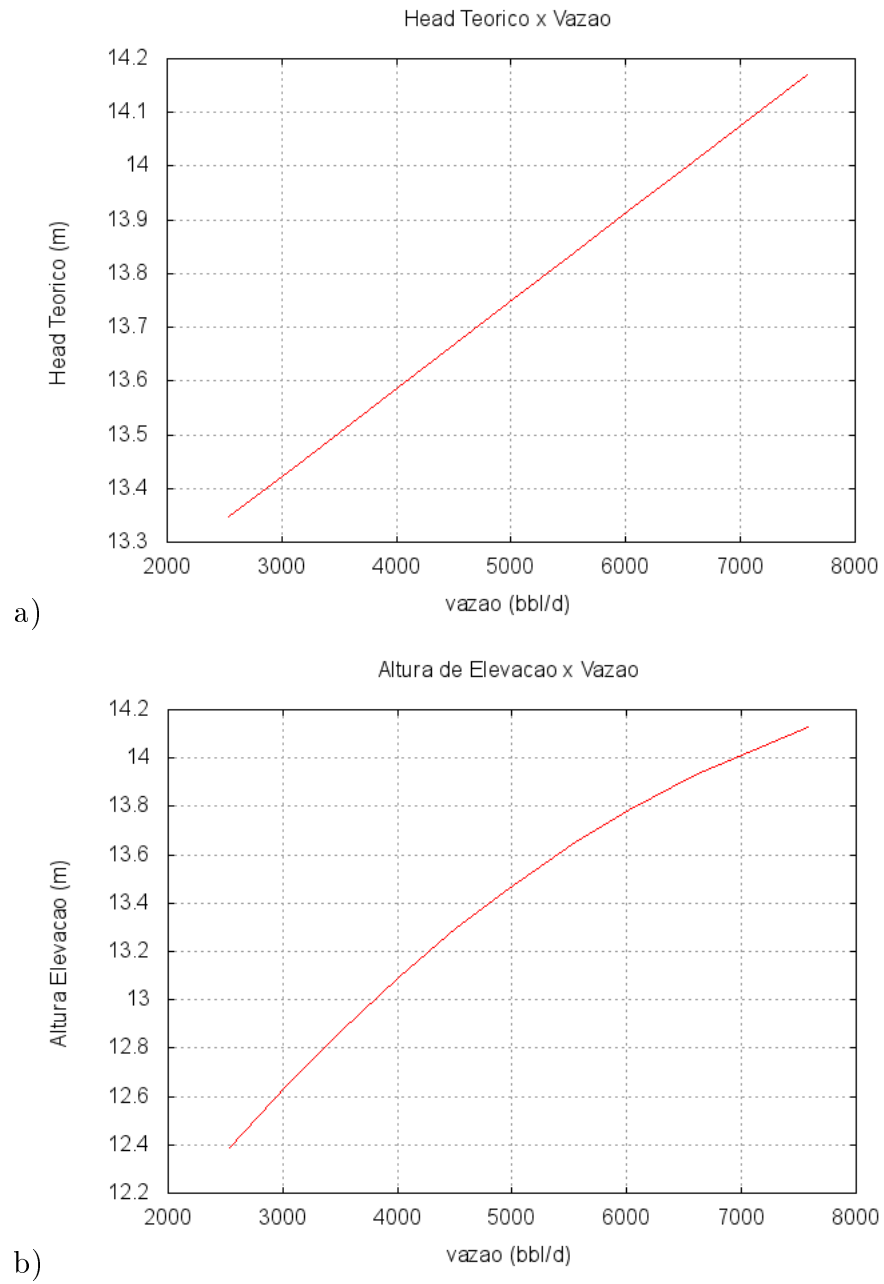


Figura 7.4: Gráficos das curvas de: a) *head* teórico e b) alturas de elevação, para a combinação rotação (1800 rpm) e viscosidade (270 cP).

Capítulo 8

Documentação

Todo projeto de engenharia precisa ser bem documentado. Neste sentido, apresenta-se neste capítulo a documentação de uso do "software ESPPerformance".

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:

1. Obtenha o arquivo executável ESPPerformance, copie e salve em seu computador.
2. Observação: Para o correto funcionamento, é necessário a instalação de um software auxiliar para geração dos gráficos, no caso o software utilizado pelo programa é o Gnuplot.

8.1.2 Como rodar o software

Para rodar o software é necessário utilizar o arquivo executável do programa.

1. Abra o arquivo executável.
2. Entre com a opção desejada:
 - (a) Viscosidade - opção 1 (fluido de 1 cP) ou opção 2 (fluido de 270 cP)
 - (b) Rotação do BCS - opção 1 (3500 rpm) ou opção 2 (1800 rpm)
3. As opções geram 4 diferentes combinações que afetam a definição parâmetros inertes ao cálculos realizados. Veja no Capítulo 7 - Teste, 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:

- Instalar o compilador gcc 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 Documentação usando doxygen

A documentação do código do software foi feita usando o padrão JAVADOC. Depois de documentar o código, utilizou-se o software `doxygen` para gerar a documentação do desenvolvedor no formato `html` e `latex`. 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` e `latex`, tais como hierarquia textual e gráfica de classes, vide Figuras 8.1 e 8.2, e documentação completa dos métodos.

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

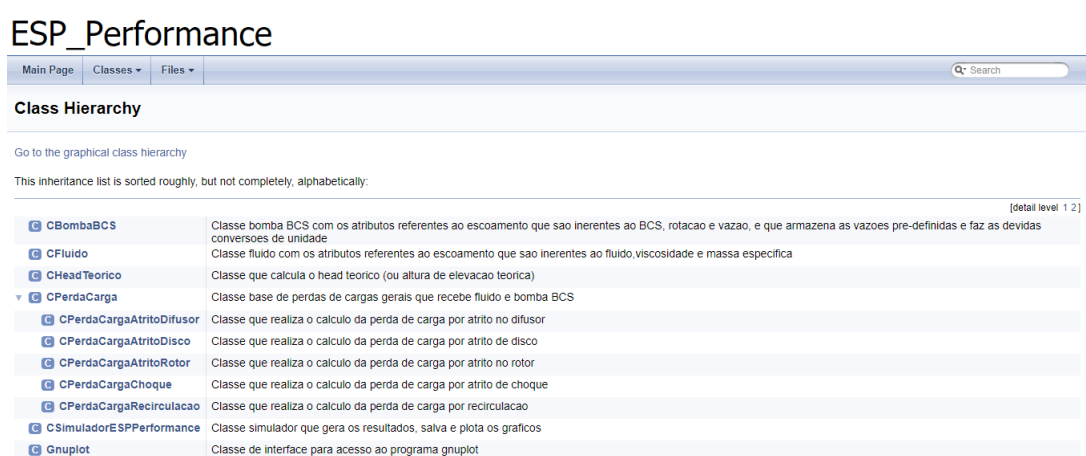


Figura 8.1: Documentação do projeto gerada pelo doxygen

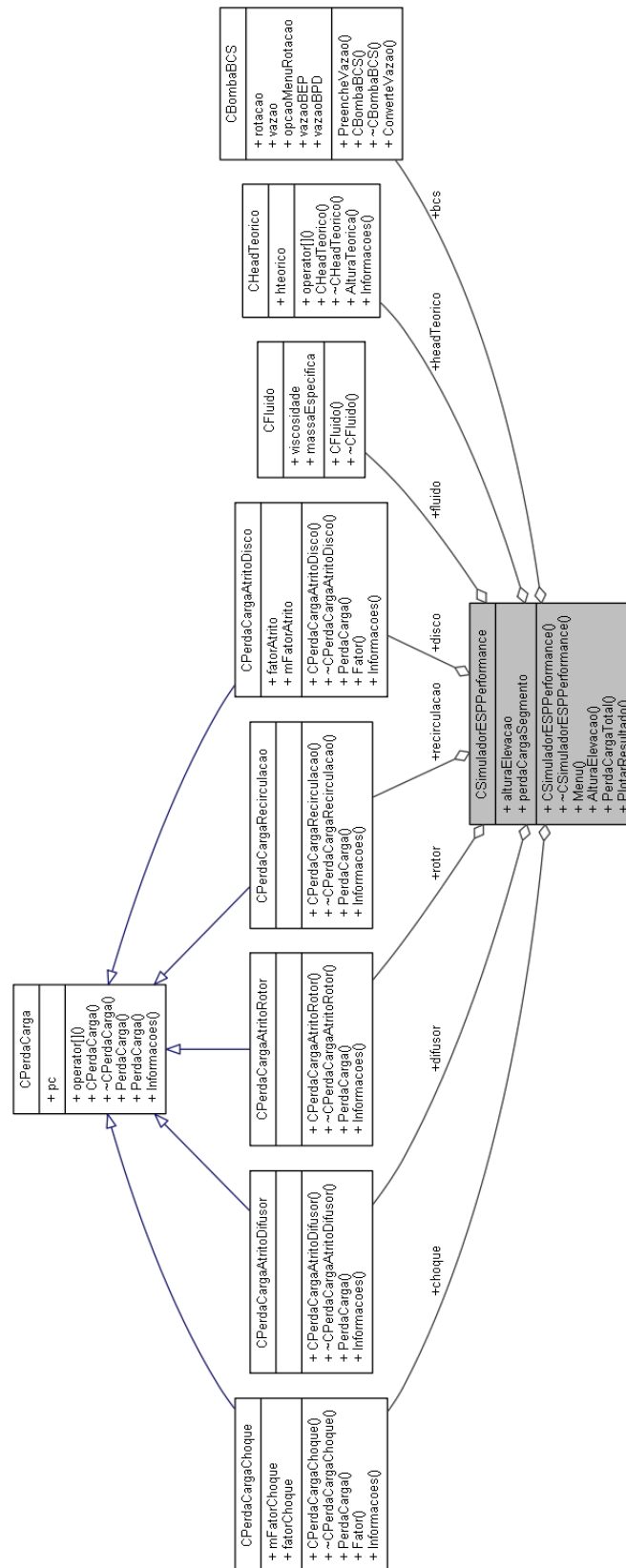


Figura 8.2: Diagrama gerado pelo doxygen: Relação entre as classes

Capítulo 9

Sugestões

Neste capítulo são apresentadas algumas sugestões que podem ser inseridas ao programa de forma a complementá-lo, aumentando o número de funcionalidades:

9.1 Classe CFluido

- Realizar novas pesquisas bibliográficas e encontrar possibilidades de aumentar as opções de fluidos de testes para a bomba, além da água e glicerina que foram as utilizadas nesse programa, aumentando assim as opções de viscosidade.

9.2 Classe CBombaBCS

- Realizar novas pesquisas bibliográficas e implementar mais opções de funcionamento da bomba, aumentando as opções de rotação da bomba.

9.3 Classe CPerdaCarga

- A classe CPerdaCarga permite o cálculo de perdas de carga em outros equipamentos, como tubulações, por exemplo, assim, há a possibilidade de complementação no código de forma a atribuir outras funcionalidades ao ESPPerformance, além das relacionadas à Bomba Centrífuga Submersa (BCS).

9.4 Classe CSimuladorESPPerformance

- Ao se implementar novas equações para realização do cálculo da perda de carga, novas combinações de equações são feitas, assim, pode-se incluir o cálculo para medir a eficiência da combinação, como função da vazão e altura de elevação.

Referências Bibliográficas

- [ITO 1959]ITO, H. Friction factors for turbulent flow in curved pipes. *Journal Basic Engineering, Transaction of the ASME*, 1959.
- [PFLEIDERER 1972]PFLEIDERER, C. *Máquinas de Fluxo*. Rio de Janeiro: Livros Técnicos e científicos Editora, 1972.
- [STEPANOFF 1957]STEPANOFF, A. J. *Centrifugal and Axial Flow Pumps Theory, Design and Application*. Nova Iorque: Second edition. John Wiley Sons, 1957.
- [SUN 2002]SUN, D. e. a. Single-phase model for esp's performance. In: . Tulsa/Oklahoma: TUALP Annual Advisory Board Meeting, 2002.
- [THIN 2008]THIN, K. C. e. a. Design and performance analysis of centrifugal pump. In: . China: World Academy of Science, Engineering and Technology., 2008.
- [Umbrello 2017]UMBRELLO, P. *Umbrello Project*. 2017. Disponível em: <<https://umbrello.kde.org/>>. 5

Índice Remissivo

A

Análise orientada a objeto, 12

AOO, 12

atributos, 21

C

Casos de uso, 4, 5

Cenário, 4

colaboração, 17

comunicação, 17

Concepção, 3

Controle, 20

D

Diagrama de colaboração, 17

Diagrama de componentes, 22

Diagrama de execução, 22

Diagrama de máquina de estado, 17

Diagrama de sequência, 15

E

Efeitos do projeto nas heranças, 21

Elaboração, 7

especificação, 3, 4

Especificações, 3

estado, 17

Eventos, 15

H

Heranças, 21

heranças, 21

I

Implementação, 24

M

Mensagens, 15

O

otimizações, 22

P

Plataformas, 20

POO, 21

Projeto do sistema, 19

Projeto orientado a objeto, 21

Protocolos, 19

R

Recursos, 20