

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

PROGRAMA EM C++ PARA AVALIAÇÃO DE FORMAÇÕES POR DADOS DE
TESTES DE PRESSÃO

DISCIPLINA: PROGRAMAÇÃO PRÁTICA

LEONAM DOS SANTOS BRAGA
RENAN MARCOS DE LIMA FILHO

MACAÉ - RJ
JANEIRO - 2014

Sumário

1	Introdução	1
1.1	Escopo do problema	1
1.2	Objetivos	1
2	Especificação	3
2.1	Especificação do programa	3
2.2	Casos de uso do programa	3
2.3	Diagrama de caso de uso geral do programa	5
2.4	Diagrama de caso de uso específico do programa	5
3	Elaboração	8
3.1	Análise de domínio	8
3.1.1	Testes de pressão	8
3.2	Formulação teórica	9
3.3	Identificação de pacotes – assuntos	11
3.4	Diagrama de pacotes – assuntos	12
4	AOO – Análise Orientada a Objeto	13
4.1	Diagramas de classes	13
4.1.1	Dicionário de classes	13
4.2	Diagrama de sequência – eventos e mensagens	21
4.2.1	Diagrama de sequência geral	21
4.2.2	Diagrama de sequência específico	21
4.3	Diagrama de comunicação – colaboração	21
4.4	Diagrama de máquina de estado	21
4.5	Diagrama de atividades	21
5	Implementação	27
5.1	Código fonte	27
6	Teste	68
6.1	Teste 1: Teste no Windows	68
6.2	Teste 2: Teste no GNU/Linux (Gerando o gráfico com o Gnuplot)	72

7 Documentação	77
7.1 Manual do Usuário	77

Capítulo 1

Introdução

No presente trabalho desenvolve-se um programa que possibilita ao usuário calcular de forma eficaz e simplificada os parâmetros de um reservatório. Tal método consiste na análise do comportamento da pressão do reservatório ao longo do tempo sob efeito de diferentes vazões de produção. Todas essas informações são importantes para a construção de um modelo confiável e o mais próximo do real visando avaliar o potencial de um reservatório. O diferencial desse projeto frente a outros já desenvolvidos, será a possibilidade de variação de um determinado parâmetro, e a visualização de como tal variação influenciará no comportamento do reservatório.

1.1 Escopo do problema

O estudo do comportamento das pressões nas formações produtoras de petróleo é de fundamental importância para a engenharia de reservatórios. Por meio de análises destes testes, é possível determinar potencialidades e características dos reservatórios, avaliar as reservas disponíveis de hidrocarbonetos, bem como fornecer previsões de produção dos fluidos existentes.

O domínio dos conceitos básicos da teoria de fluxo de fluidos através de meios porosos é necessário para aplicação e desenvolvimento das técnicas atuais de análise de dados de pressão em poços, particularmente no que se refere à avaliação das potencialidades e determinação das características dos horizontes produtores.

Por ocasião da descoberta de novos campos petrolíferos, as decisões sobre os recursos a serem investidos, dependem, por exemplo, de avaliações realizadas por intermédio de testes em poços. Estes testes duram pouco tempo e geram resultados confiáveis para avaliar a exploração de tais jazidas.

1.2 Objetivos

Os objetivos deste trabalho são:

- Objetivo geral:
 - Criar um software capaz de fornecer ao usuário parâmetros e características do reservatório, através da análise dos dados obtidos em testes de pressão em poços, possibilitando estimar as dimensões do campo e sua potencialidade econômica.
- Objetivos específicos:
 - Calcular permeabilidade efetiva, dano de formação, pressão inicial e dimensão do reservatório.
 - Calcular índice de produtividade.
 - Estimar efeito de estocagem e sua duração.
 - Fornecer ao usuário curvas do comportamento da pressão ao longo do tempo.
 - Criar um banco de dados que possibilite ao usuário comparar o reservatório que está sendo analisado com outros anteriormente analisados pelo programa.

Capítulo 2

Especificação

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

2.1 Especificação do programa

O projeto a ser desenvolvido consiste em um software que importe dados obtidos através de testes de pressão a partir de arquivos .dat. O programa realizará uma regressão linear semi-logaritmica entre a pressão medida no poço versus o tempo, gerando gráficos com auxílio da classe externa Gnuplot. O usuário fornecerá através do arquivo .dat, valores iniciais de porosidade, espessura do reservatório, viscosidade e fator volume de formação do fluido. Através da visualização e interpretação dos gráficos gerados, o programa terá como saída os valores de permeabilidade efetiva, pressão inicial, raio externo do reservatório, índice de produtividade, fator película de formação e efeito de estocagem (caso estes ocorram), além dos gráficos da variação da pressão no poço versus tempo.

O desenvolvimento do programa será feito utilizando a linguagem de programação C++, por se tratar de uma linguagem eficiente e possibilitar o reaproveitamento de códigos já desenvolvidos. Por se tratar de um programa científico, será utilizada uma interface em modo texto, que permitirá a entrada e saída de dados de forma simplificada.

2.2 Casos de uso do programa

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 sequência típica de uso do programa (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.

Nome do caso de uso:	Cálculo de parâmetro do reservatório
Resumo/descrição:	Determinação parâmetros do reservatório através de uma regressão linear com os dados do teste de pressão.
Etapas:	<ol style="list-style-type: none">1. Importar dados do arquivo.dat.2. Fornecer dados do reservatorio.3. Fornecer dados do fluido.4. Gerar gráficos.5. Analisar resultados.6. Gerar gráfico Variação da Pressão x Tempo.7. Exibir resultados na tela e exportar para um arquivo .dat.8. Selecionar uma variável e definir um intervalo de variação para a mesma.9. Gerar novos resultados e analisar a interferência da variação nestes.
Cenários alternativos:	Um cenário alternativo envolve uma entrada errada do usuário (por exemplo, valores de entrada da viscosidade, porosidade ou da altura do reservatório, negativos). O programa apresentará um bug quando valores que deveriam ser positivos forem menores que zero. Outro exemplo ocorre na entrada de dados para criação da função semi-logaritimica. O programa apresentará um bug quando for determinar o logarítmo de -1.

Tabela 2.1: Caso de uso do programa.

2.3 Diagrama de caso de uso geral do programa

A Figura 2.1 demonstra como o usuário vai interagir com o programa, através de ações simples.

2.4 Diagrama de caso de uso específico do programa

A Figura 2.2 exibe a interação do usuário com o programa, porém agora detalhando a solicitação de variação de um determinado parâmetro.

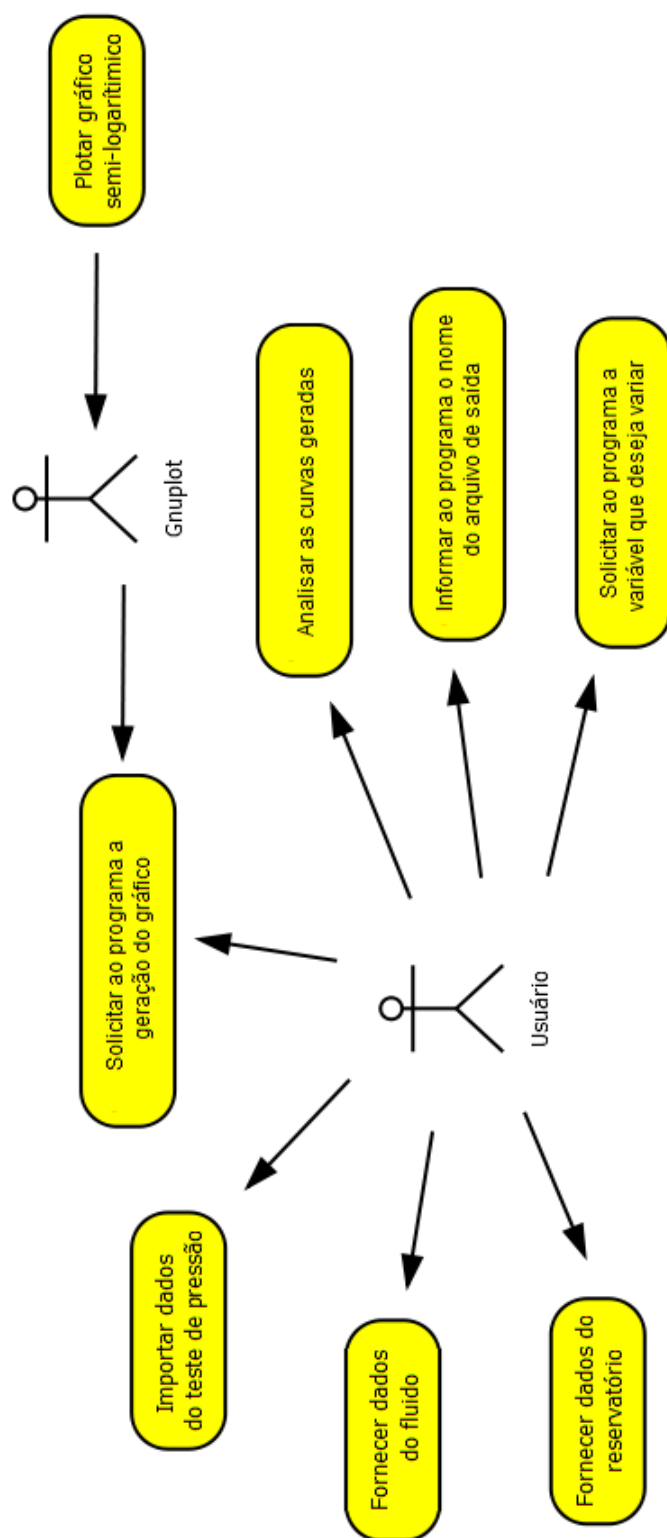


Figura 2.1: Diagrama de caso de uso geral.

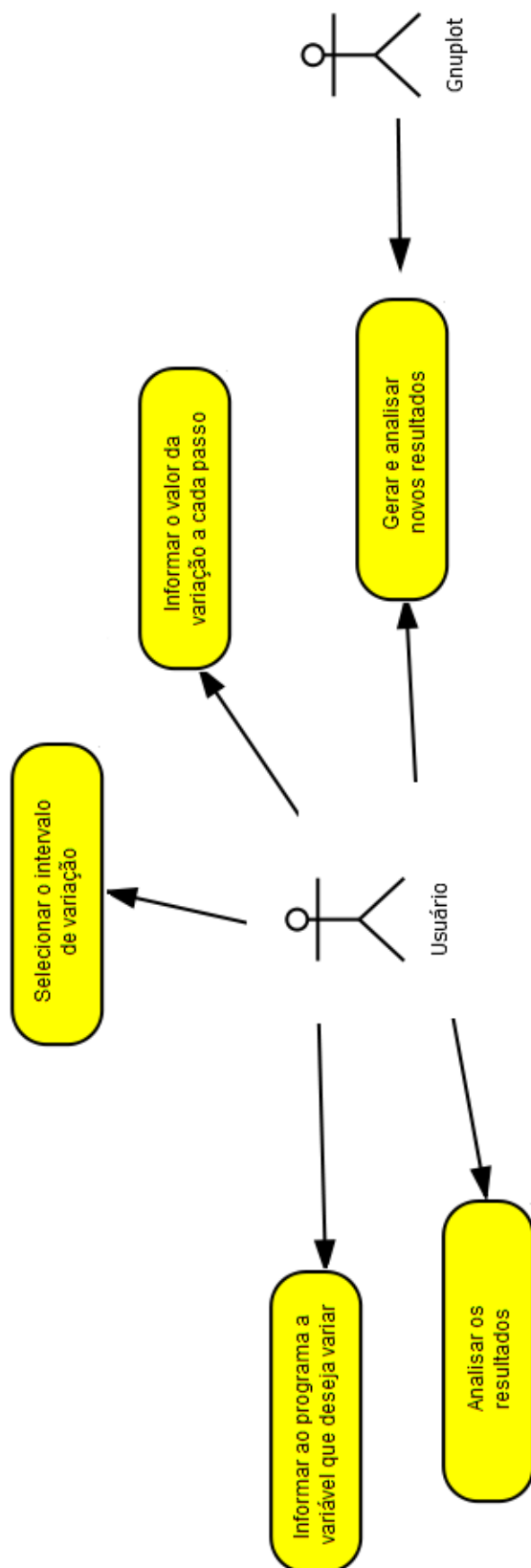


Figura 2.2: Diagrama de caso de uso específico.

Capítulo 3

Elaboração

Neste capítulo vamos fazer uma análise dos requisitos, e assim ajustar o programa de forma útil para o usuário ao incluir a possibilidade do mesmo variar determinados parâmetros e visualizar seu impacto no comportamento do reservatório, além de possibilitar uma futura extensão do programa.

3.1 Análise de domínio

A análise de domínio pode ser feita em conjunto com a especificação de pacotes, uma vez que os pacotes representam as áreas abordadas pelo programa.

3.1.1 Testes de pressão

A Engenharia de Reservatórios é uma subárea da Engenharia de Exploração e Produção de Petróleo. Um dos métodos mais utilizados nesta subárea são os testes de pressão. Estes consistem em acompanhar os dados de pressão no poço em função da vazão de produção utilizada. Através desse acompanhamento pode se determinar parâmetros de drenagem e outros fatores que interferem na produção do campo de petróleo.

A avaliação da formação consiste em um conjunto de atividades com vazão controlada, que têm como objetivos:

- obter o fluido contido na formação para análise do mesmo.
- avaliar a capacidade produtiva da formação.
- investigar a existência de danos de formação e efeito de estocagem.
- determinar a extensão do reservatório e sua pressão inicial.

Os testes de pressão seguem as seguintes etapas:

- completar o poço temporariamente para permitir a produção do fluido de forma segura.

- isolar o intervalo a ser testado.
- criar um diferencial de pressão entre poço e reservatório, afim de produzir o fluido.
- promover períodos intercalados de produção e fechamento do poço.
- registro contínuo de vazões em superfície e pressões no poço.

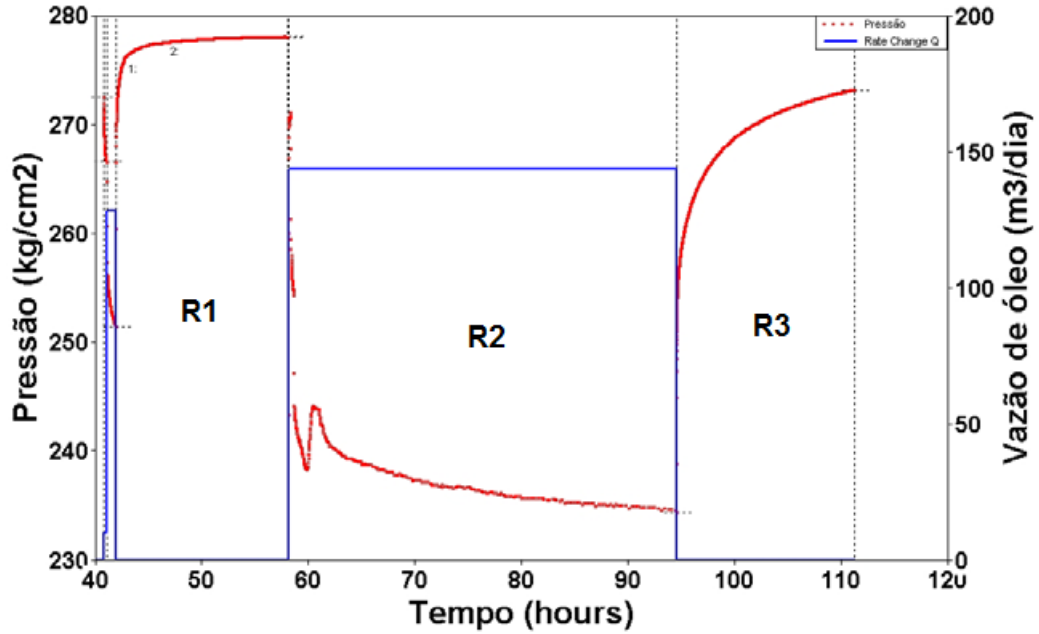


Figure 3.1: Exemplo de teste de poço convencional, onde é possível observar o comportamento da pressão em função da variação da vazão de produção. As regiões 1 e 3 mostram etapas com o poço fechado ($q=0$), onde é possível ver um aumento da pressão no reservatório. Já na região 2, o poço está produzindo com uma vazão constante, causando uma queda na pressão do reservatório.

3.2 Formulação teórica

A base para toda análise e obtenção das equações que regem o comportamento do reservatório é a Equação da Difusividade 3.1, que para geometria radial e fluxo monofásico é dada por:

$$\frac{1}{r} \frac{\partial}{\partial r} \left(\frac{k(r)}{\mu} r \frac{\partial p}{\partial r} \right) + \frac{k(r)}{\mu} C_f \left(\frac{\partial p}{\partial r} \right)^2 = \phi C_t \frac{\partial p}{\partial t} \quad (3.1)$$

A equação admite inúmeras soluções. Para obter a solução para um caso particular é necessário especificar as condições iniciais e de contorno de acordo com o tipo de reservatório.

Com os dados obtidos da pressão e o tempo em que cada uma foi medida (o teste inteiro pode variar a duração desde algumas horas a alguns dias), gera-se um gráfico

semi-logarítmico em que seu coeficiente angular possui uma relação intrínseca com a permeabilidade da formação e estima-se propriedades pelo método de Horner, ilustrado pela Figura 3.2.

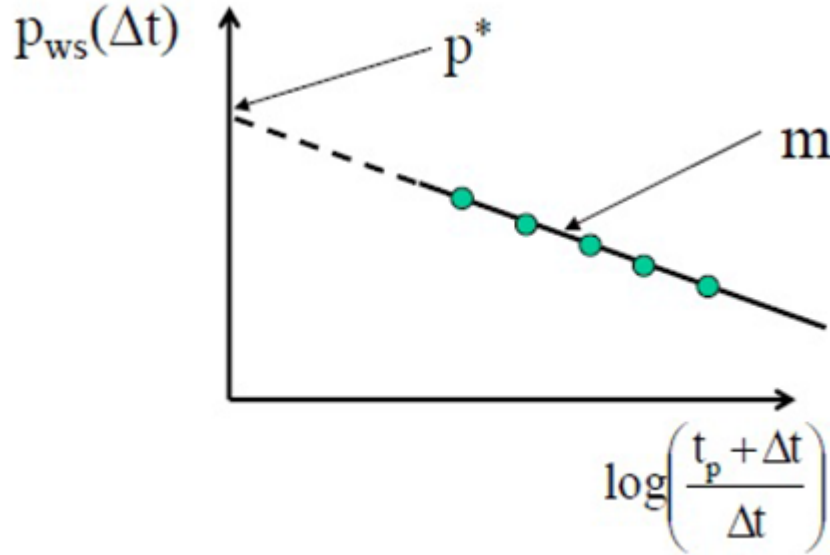


Figura 3.2: Gráfico semi-logaritimico obtido pelo Método de Horner, que fornece parâmetros referentes ao reservatório.

As equações apresentadas a seguir para determinação dos parâmetros são vistas em [Adalberto Rosa, 2006].

A permeabilidade, que é definida como a capacidade de um material (tipicamente uma rocha) de transmitir fluidos, pode ser inferida pela equação 3.2. Na equação, q [barris/d] é a vazão, B [adimensional] o fator volume-formação, h [pés] a altura da formação, μ [cp] a viscosidade do fluido, e m [adimensional] é o coeficiente angular.

$$k = \frac{1.151 \alpha_p q B \mu}{m h} \quad (3.2)$$

O fator de película S [adimensional] é definido como uma região ao redor do poço cuja permeabilidade foi alterada, reduzida (por fluido de perfuração/completação, inchamento de argilas, inversão de molhabilidade, etc) ou melhorada (através de processos de acidificação, fraturamento hidráulico, etc.) e se dá pela equação 3.3. É uma equação adimensional, onde ct [1/psi] é a compressibilidade total, r_w [pés] é o raio do poço, tp [dias] o tempo de produção do poço, Pwf [psi] é a pressão registrada no fechamento do poço, ΔP_{skin} [psi] é a queda de pressão devido ao dano e n [adimensional] é o coeficiente linear do gráfico. Quanto maior o valor do fator de película, maior o dano, resultando em menor produção e maior queda de pressão.

$$S = \frac{1.151.(m.log(tp + 1) + n - Pwf)}{(-m - log(k/(c_trw^2) + 3.23)} \quad (3.3)$$

$$\Delta P_{skin} = 0.869.(-m).s$$

A capacidade produtiva de um poço é caracterizada pelo índice de produtividade IP [barris/(dias.psi)], que indica a necessidade de injeção de fluidos para aumento da recuperação. A eficiência de fluxo EF [adimensional] indica o quanto a produção está sendo afetada pelo fator de película do poço. Esses fatores, dados em porcentagem, são importantes para a engenharia de reservatórios, pois definem a viabilidade de produção. São definidos pela equações 3.4 e 3.5, sendo Pi [psi] a pressão inicial do reservatório, indicada pelo gráfico. Valores maiores que 10 para o índice de produtividade são considerados bons.

$$IP = \frac{q}{Pi - P_{wf}} \quad (3.4)$$

$$EF = \frac{Pi - P_{wf} - \Delta P_{skin}}{Pi - P_{wf}} \quad (3.5)$$

O raio efetivo do poço rwe [pés] é definido como o tamanho teórico do poço incluindo o dano, calculado pela equação 3.6, sua unidade é [m]. Quanto maior o dano, menor o raio efetivo, pois o poço produz menos do que deveria.

$$rwe = rw.exp(-s) \quad (3.6)$$

O efeito de estocagem ocorre nos primeiros momentos da produção, fazendo com que a vazão do poço não seja igual à do reservatório, havendo uma estocagem de fluidos no interior do poço pela expansão e compressão do volume dos hidrocarbonetos. O coeficiente de estocagem C [barris/psi] é descrito pela equação 3.7, e sua duração, $twbs$ [horas], pela equação 3.8.

$$C = \frac{qB\Delta t}{24(P - P_{wf})} \quad (3.7)$$

$$t_{wbs} = \frac{60.0 + 3.5 * S}{(24C * \alpha_{pkh\mu})} \quad (3.8)$$

3.3 Identificação de pacotes – assuntos

- Pacote Testes: importa os dados do teste de pressão de um arquivo .dat e apresenta conceitos utilizados para estimar parâmetros do reservatório.
- Pacote Estatística: realiza a regressão linear dos dados passados no arquivo .dat. Deve apresentar conexão com o Pacote Reservatório, uma vez que recebe os dados

contidos nele.

- Pacote Gráfico: gera os gráficos com a curva semi-logaritmica dos dados da regressão linear feita no Pacote Matemático. Também plota os dados de entrada da pressão versus tempo.
- Pacote Reservatório: composto de diversas equações, calcula propriedades que podem ser obtidas pela curva semi-logaritmica. Os resultados dessas equações são parte da saída do programa.
- Pacote Banco de Dados: Os parâmetros calculados no Pacote Propriedades são fornecidos ao usuário através de um arquivo .dat e armazenados em um banco de dados. O mesmo é feito com as curvas de pressão x tempo, que são fornecidas ao usuário através de um arquivo de imagem e posteriormente armazenadas.

3.4 Diagrama de pacotes – assuntos

Um diagrama de pacotes é útil para mostrar as dependências entre as diversas partes do sistema. Pode incluir: sistemas, subsistemas, colaborações, casos de uso e componentes.

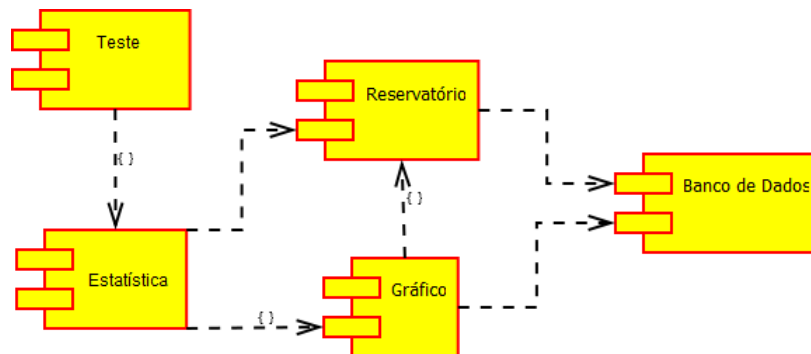


Figura 3.3: Diagrama de pacotes mostra as dependências entre os diversos pacotes do sistema.

Capítulo 4

AOO – Análise Orientada a Objeto

A terceira etapa do desenvolvimento de um software 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 nas Figuras 4.1, 4.2 e 4.3.

4.1.1 Dicionário de classes

- Classe CPoco: Classe que possui as características/atributos do poço, e tem uma função de entrada de dados por parte do usuário.
 - atributo vazao.
 - atributo tempoProducao.
 - atributo pressaoPoco.
 - atributo raioPoco.
 - método EntradaDados (): Método que pede ao usuário os parâmetros necessários para o programa.
 - método Erro (): Verifica e retorna uma mensagem de erro, caso haja alguma entrada equivocada do usuário.
 - método Vazao (_vazao): Método que seta o valor do atributo vazao.
 - método Vazao (): Método que retorna o valor do atributo vazao.
 - método TempoProducao (_tempoProducao): Método que seta o valor do atributo tempoProducao.

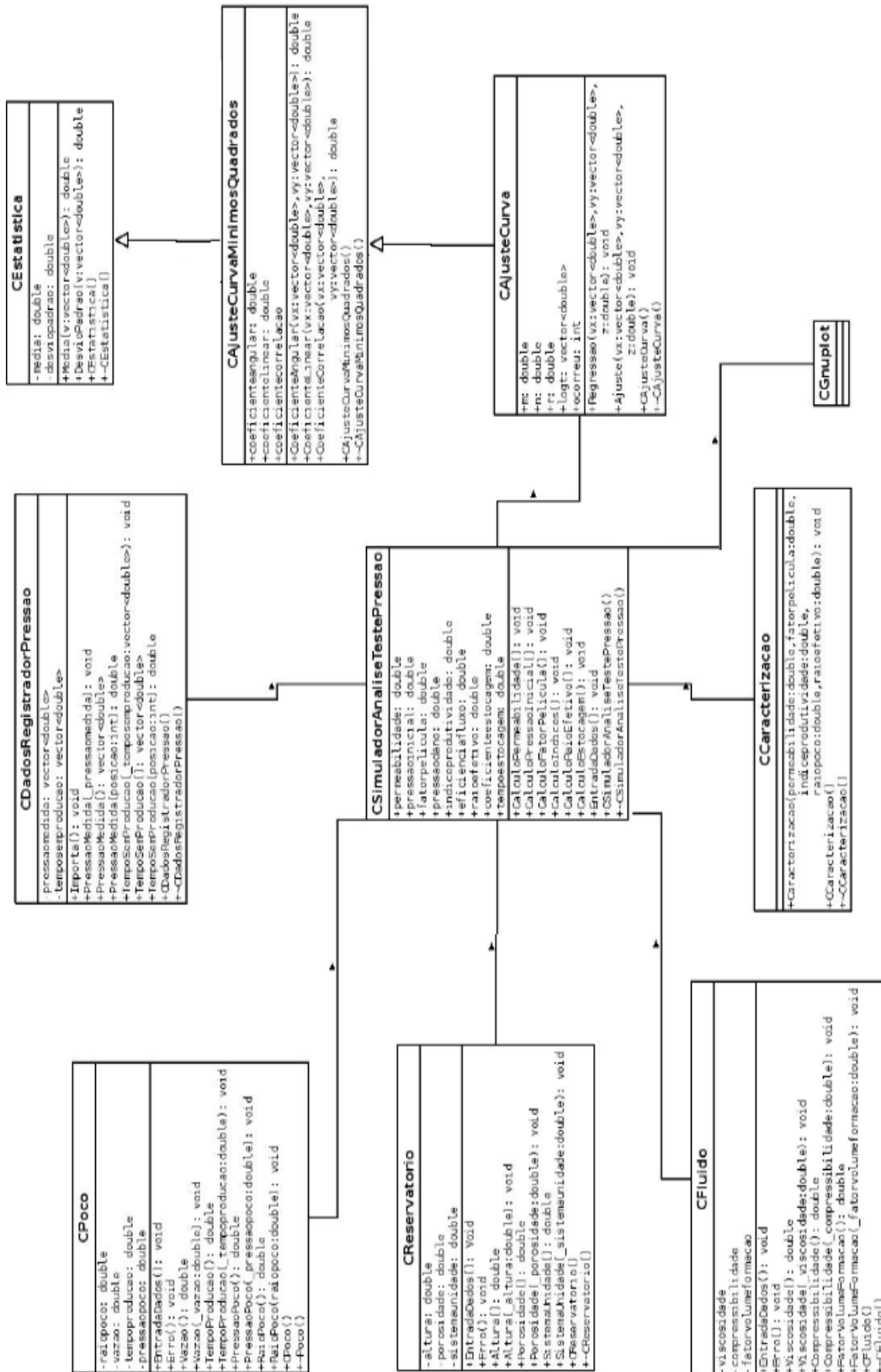


Figura 4.1: Diagrama de classes.

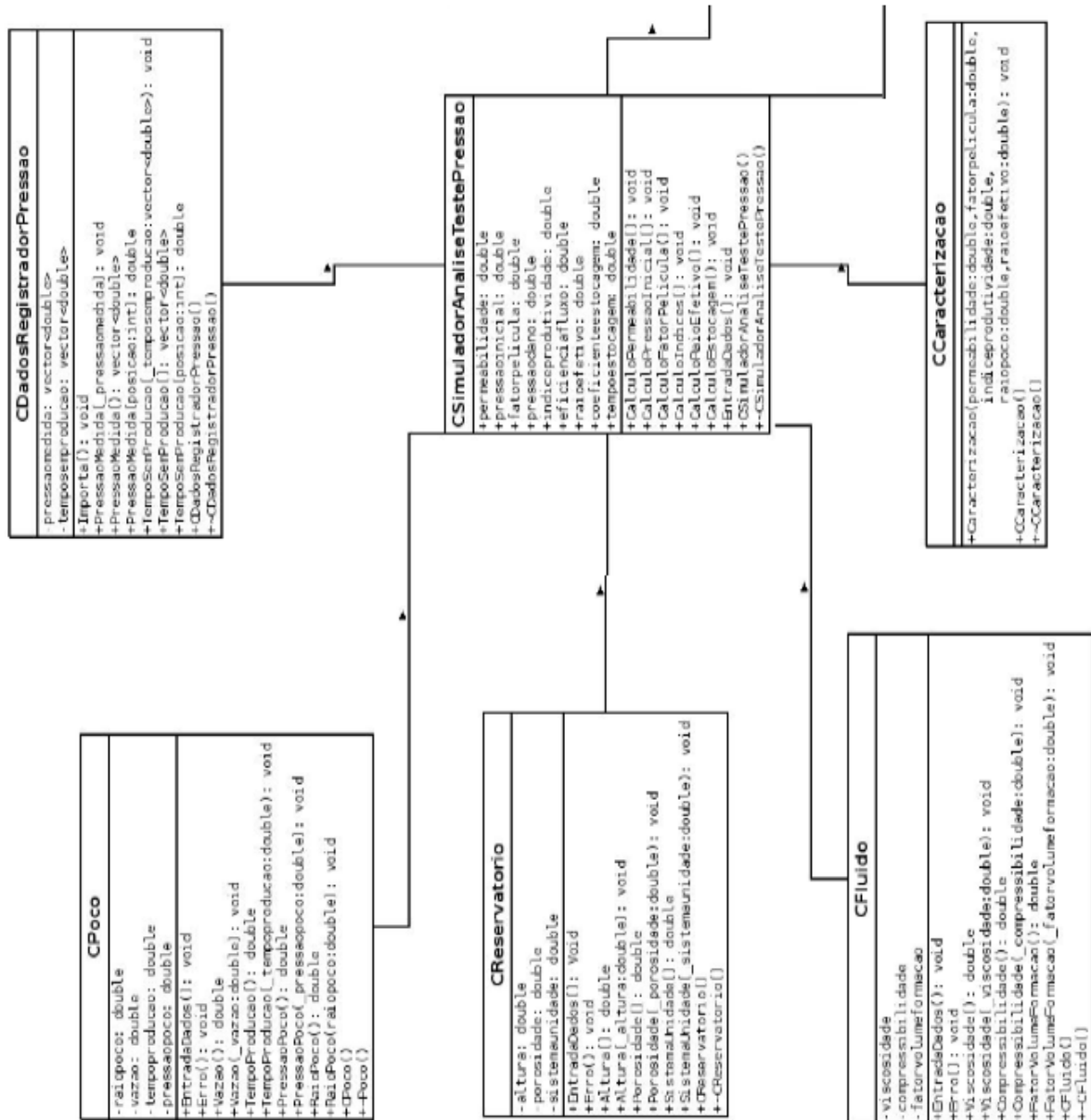


Figura 4.2: Diagrama de classes (a).

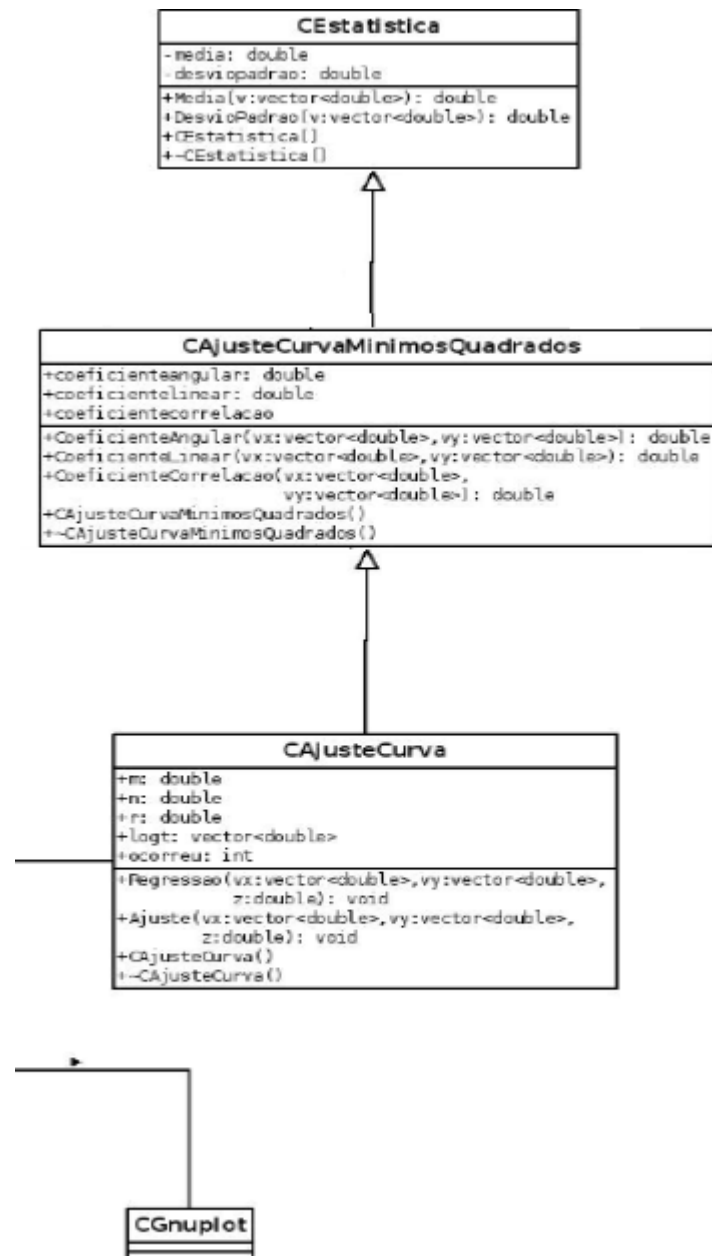


Figura 4.3: Diagrama de classes (b).

- método TempoProducao (): Método que retorna o valor do atributo tempo-Producao.
 - método PressaoPoco (_pressaoPoco): Método que seta o valor do atributo pressaoPoco.
 - método PressaoPoco (): Método que retorna o valor do atributo pressaoPoco.
 - método RaioPoco (_raioPoco): Método que seta o valor do atributo raioPoco.
 - método RaioPoco (): Método que retorna o valor do atributo raioPoco.
- Classe CReservatorio: Classe que possui as características/atributos do reservatório, e tem uma função de entrada de dados por parte do usuário.
 - atributo porosidade.
 - atributo altura.
 - atributo sistemaUnidade.
 - método EntradaDados (): Método que pede ao usuário os parâmetros necessários para o programa.
 - método SistemaUnidades (): Método do tipo void que pergunta ao usuário o sistema de unidades utilizado para os parâmetros fornecidos, através de um pequeno menu.
 - método SistemaUnidade (_sistemaUnidade): Método que seta o valor do atributo sistemaUnidade.
 - método SistemaUnidade (): Método que retorna o valor do atributo sistemaUnidade.
 - método Erro (): Verifica e retorna uma mensagem de erro, caso haja alguma entrada equivocada do usuário.
 - método Porosidade (_porosidade): Método que seta o valor do atributo porosidade.
 - método Porosidade (): Método que retorna o valor do atributo porosidade.
 - método Altura (_altura): Método que seta o valor do atributo altura.
 - método Altura (): Método que retorna o valor do atributo altura.
 - Classe CFluido: Classe que possui as características/atributos do fluido, e tem uma função de entrada de dados por parte do usuário.
 - atributo fatorVolumeFormacao.
 - atributo viscosidade.
 - atributo compressibilidade.

- método `EntradaDados ()`: Método do tipo void que pede ao usuário os parâmetros necessários para o programa.
 - método `Erro ()`: Verifica e retorna uma mensagem de erro, caso haja alguma entrada equivocada do usuário.
 - método `FatorVolumeFormacao (_fatorVolumeFormacao)`: Método que seta o valor do atributo `fatorVolumeFormacao`.
 - método `FatorVolumeFormacao ()`: Método que retorna o valor do atributo `fatorVolumeFormacao`.
 - método `Viscosidade (_viscosidade)`: Método que seta o valor do atributo `viscosidade`.
 - método `Viscosidade ()`: Método que retorna o valor do atributo `viscosidade`.
 - método `Compressibilidade (_compressibilidade)`: Método que seta o valor do atributo `compressibilidade`.
 - método `Compressibilidade ()`: Método que retorna o valor do atributo `compressibilidade`.
- Classe `CDadosRegistradorPressao`: Classe que cria 2 vetores e os preenche com os dados de teste de pressão importados de um arquivo de disco.
 - atributo `pressaoMedida`.
 - atributo `tempoSemProducao`.
 - método `Importa ()`: Método do tipo void que preenche os vetores.
 - método `PressaoMedida (_pressaoMedida)`: Método que seta o valor do atributo `pressaoMedida`.
 - método `PressaoMedida (_posicao)`: Método que seta o valor do atributo `pressaoMedida` na posição desejada.
 - método `PressaoMedida ()`: Método que retorna o valor do atributo `pressaoMedida`.
 - método `TempoSemProducao (_tempoSemProducao)`: Método que seta o valor do atributo `tempoSemProducao`.
 - método `TempoSemProducao (_posicao)`: Método que seta o valor do atributo `tempoSemProducao` na posição desejada.
 - método `TempoSemProducao ()`: Método que retorna o valor do atributo `tempoSemProducao`.
 - Classe `CEstatistica`: Classe que faz a média e desvio padrão de vetores, necessários para a regressão linear dos dados.

- atributo media.
 - atributo desvio.
 - método Media (v): Retorna a média do vetor v.
 - método DesvioPadrao (v): Retorna o desvio padrão de v.
- Classe CAjusteCurvaMinimosQuadrados: Classe que faz a regressão linear através do método dos mínimos quadrados.
 - atributo coeficienteAngular.
 - atributo coeficienteLinear.
 - atributo coeficienteCorrelacao.
 - método CoeficienteAngular (vx,vy): Retorna o coeficiente angular da reta obtida da regressão dos vetores vx e vy.
 - método CoeficienteLinear (vx,vy): Retorna o coeficiente linear da reta obtida da regressão dos vetores vx e vy.
 - método CoeficienteCorrelacao (vx,vy): Retorna o coeficiente correlação da reta obtida da regressão dos vetores vx e vy.
- Classe CAjusteCurva: Classe que executa a regressão linear (de uma reta semilogarítmica) dos dados obtidos e verifica se o coeficiente de correlação é satisfatório, caso não seja, descobre-se a melhor aproximação (o ponto) onde começa a reta da curva (a curva sendo o efeito de estocagem).
 - atributo m: Representa o coeficiente angular da reta obtida na regressão linear.
 - atributo n: Representa o coeficiente linear da reta obtida na regressão linear.
 - atributo r: Coeficiente de correlação da reta, quanto mais próximo de 1, melhor a regressão linear.
 - atributo logt: Vetor que relaciona as variáveis tp e o vetor deltat.
 - atributo efeitoEstocagem
 - método Regressao (vx, vy, z): Função que executa a regressão linear propriamente dita dos vetores, calculando os valores de m, n e r.
 - método Ajuste (vx, vy, z): Função que analisa se a regressão linear tem um fator de correlação de Pearson suficiente para o programa gerar resultados confiáveis.
- Classe CGnuplot: Classe que possibilita a geração de gráficos usando o programa externo Gnuplot.

- Classe CSimuladorAnaliseTestePressao: Classe principal, que se comunica com os objetos das outras classes para inferir parâmetros do reservatório e calcular outras variáveis a partir de equações de correlação.
 - atributo permeabilidade.
 - atributo pressaoInicial.
 - atributo fatorPelicula.
 - atributo pressaoDano.
 - atributo indiceProdutividade.
 - atributo eficienciaFluxo.
 - atributo raioEfetivo.
 - atributo coeficienteEstocagem.
 - atributo tempoEstocagem.
 - método EntradaDados (): Método que chama as funções de entrada das classes CFluido, CPoco e CReservatorio.
 - método CalculoPermeabilidade (): Função que calcula e exibe a permeabilidade do reservatório.
 - método CalculoPressaoInicial (): Função que calcula e exibe a pressão inicial pela extrapolação da reta.
 - método CalculoFatorPelicula (): Função que calcula e exibe o fator de película do reservatório e a queda de pressão devido à esse fator.
 - método CalculoIndices (): Função que calcula e exibe o índice de produtividade do reservatório e a eficiência de fluxo.
 - método CalculoRaioEfetivo (): Função que calcula e exibe o raio efetivo.
 - método Exporta(): Método que exporta os resultados para um arquivo .dat com um nome escolhido pelo usuário.
 - método Variacao(): Método que permite ao usuário variar um parâmetro selecionado e visualizar a mudança no comportamento do reservatório nos gráficos gerados.
- Classe CCaracterizacaoReservatorio: Classe que caracteriza o reservatório, interpretando os resultados obtidos.
 - método Caracterizacao (permeabilidade, fatorPelicula, indiceProdutividade, raioPoco, raioEfetivo): Função do tipo void que analisa os resultados e informa ao usuário a qualidade do reservatório submetido ao teste de pressão.

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. Costuma ser montado a partir de um diagrama de caso de uso e estabelece o relacionamento dos atores (usuários e sistemas externos) com alguns objetos do sistema.

4.2.1 Diagrama de seqüência geral

Veja o diagrama de seqüência na Figura 4.4.

4.2.2 Diagrama de seqüência específico

Veja o diagrama de seqüência específico na Figura 4.5.

4.3 Diagrama de comunicação – colaboração

Veja na Figura 4.6 o diagrama de comunicação. Observe que há muita interação entre os objetos de cada classe, iniciando pela entrada de dados para preencher os objetos de CPoco, CReservatorio e CFluido. Caso não ocorra erro na entrada, a classe DadosRegistradorPressões importa os dados do arquivo de texto e informa para a classe CAjusteCurva. Esta, por sua vez, utiliza a classe CAjusteCurvaMinimosQuadrados que faz a regressao de dois vetores usando a média e o desvio padrão obtidos da classe CEstatistica. Após a função Ajuste (r), que encontra o coeficiente de estocagem, o SimuladorAnaliseTestePressao faz os cálculos dos parâmetros do reservatório com esses dados de entrada e de importação. Finalmente, a classe CCaracterizacao caracteriza o reservatorio com a função Caracterizacao (permeabilidade, fatorPelícula, indiceProdutividade, raioPoco, raioEfetivo). Essas interações são de vital importância para o funcionamento do programa e a caracterização do reservatório.

4.4 Diagrama de máquina de estado

Veja na Figura 4.7 o diagrama de máquina de estado para o objeto da classe CSimuladorAnaliseTestePressao. Observe que o objeto possui atributos informados pelo usuário na seleção do parâmetro a ser variado, e como será feita tal variação.

4.5 Diagrama de atividades

Veja na Figura 4.8 o diagrama de atividades do programa, incluindo a inovação. Observe que o atributo porosidade do objeto da classe CReservatorio é informado pelo usuá-

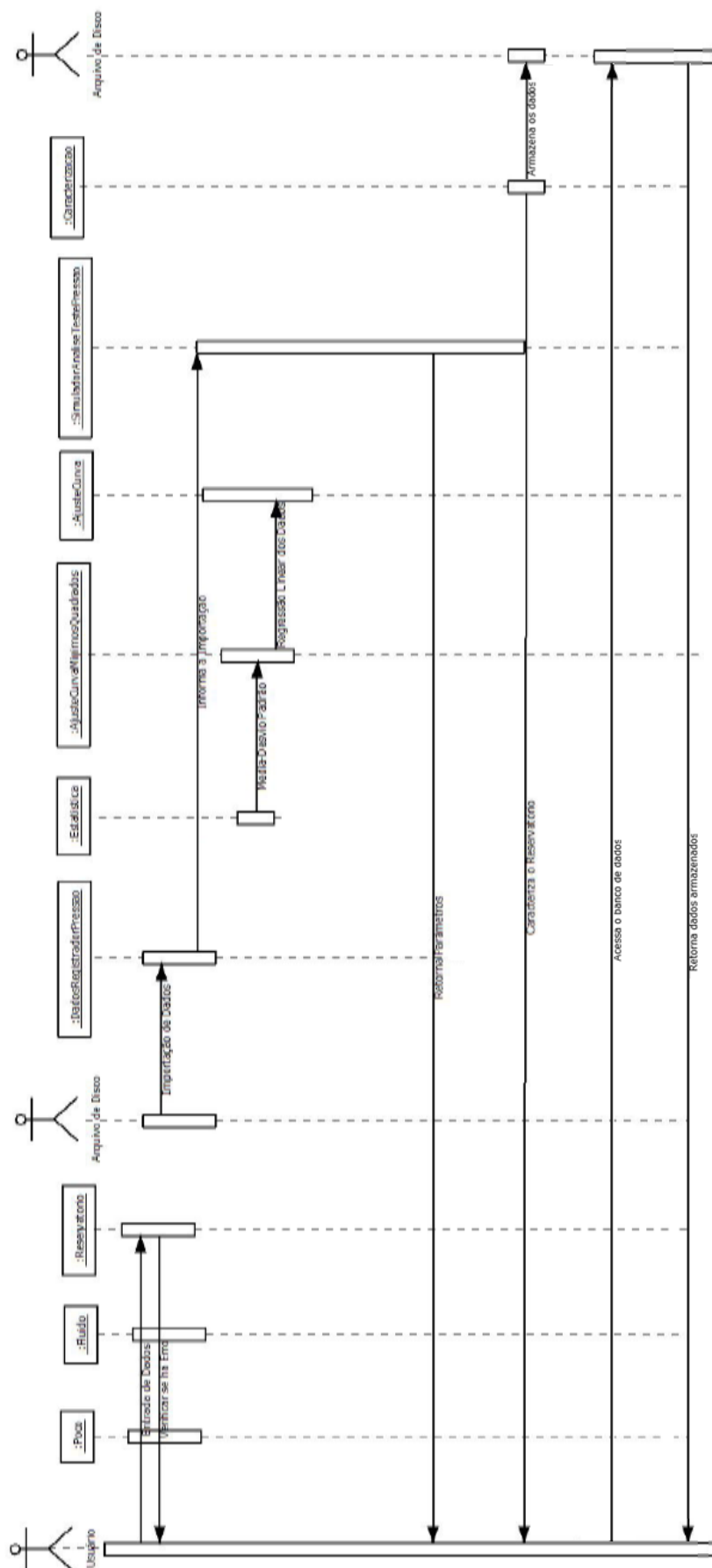


Figura 4.4: O diagrama de sequência geral mostra a execução do programa, com ordem temporal da troca de eventos e mensagens.

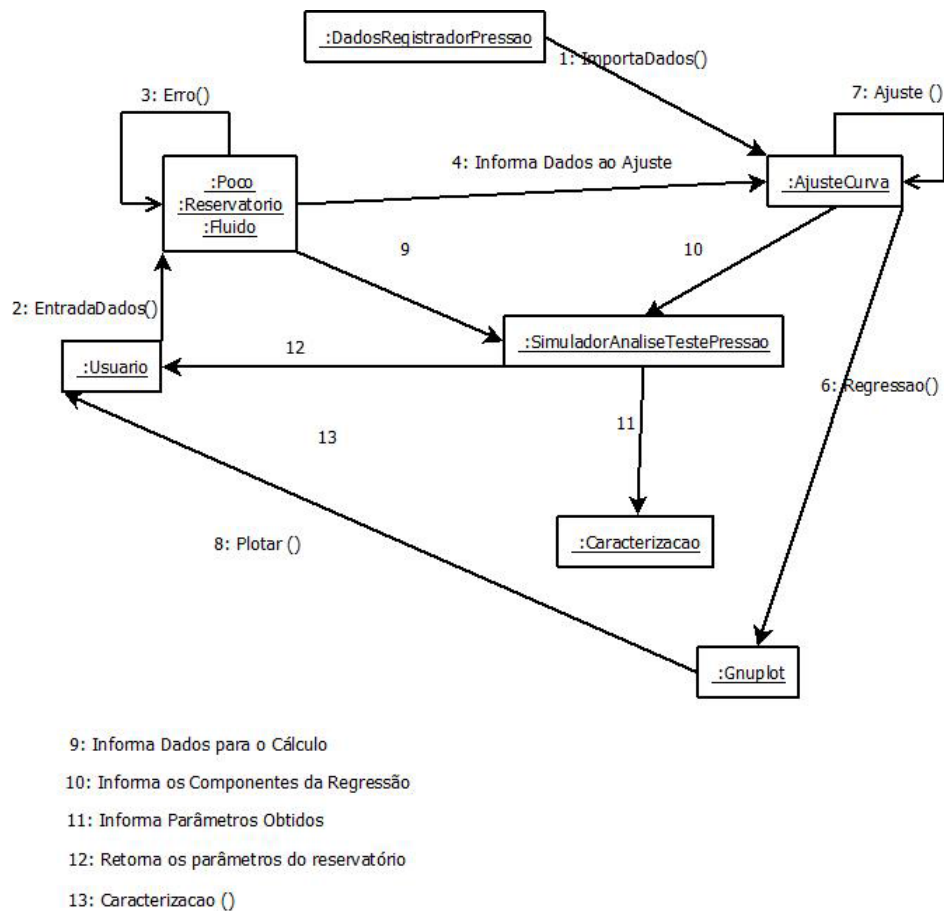


Figura 4.6: Diagrama de comunicação, que mostra o conjunto de objetos e seus relacionamentos, incluindo as mensagens trocadas entre eles.

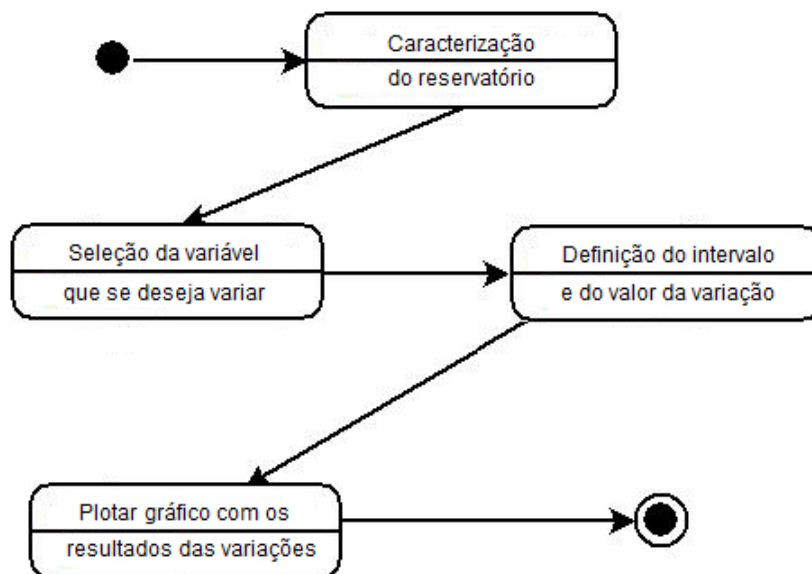


Figura 4.7: O diagrama de máquina de estado mostra os diversos estados que o objeto assume e os eventos que ocorrem ao longo do processo., modelando aspectos dinâmicos do objeto.

rio. Caso haja erro na entrada (porosidade informada menor do que zero), o programa pede uma nova entrada da porosidade, finalizando o preenchimento.

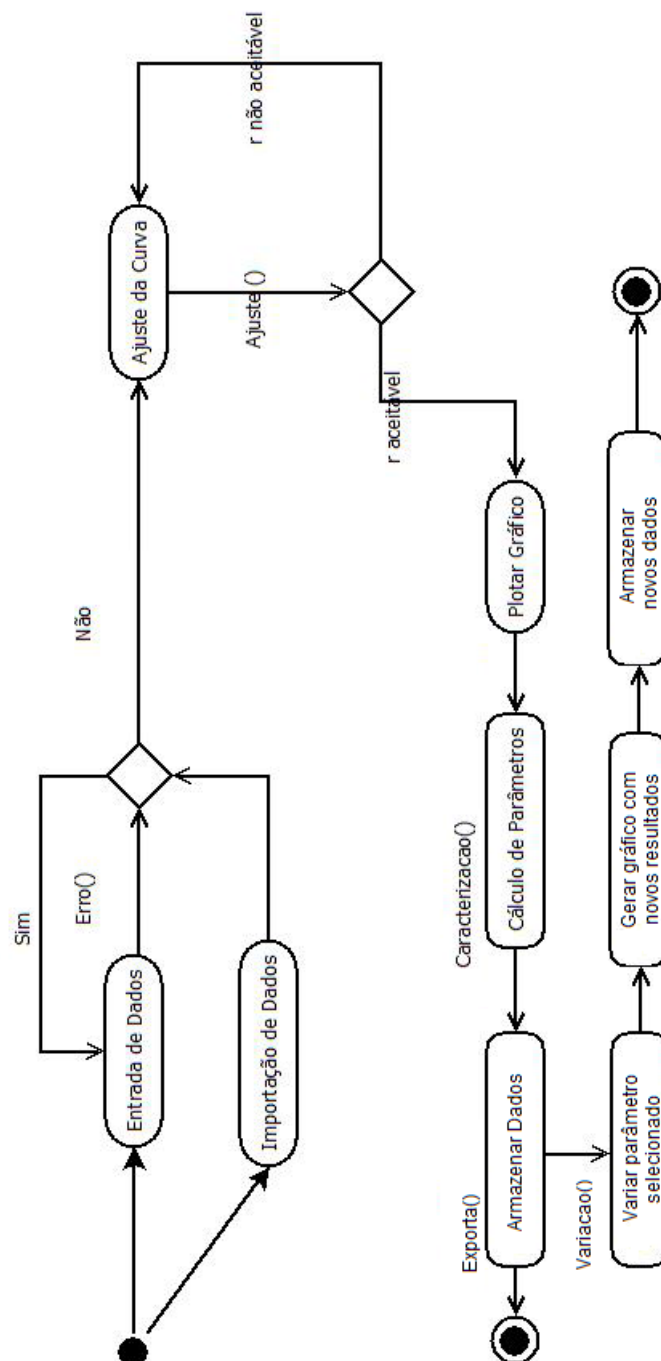


Figura 4.8: Diagrama de atividades, mostrando o fluxo de controle de uma atividade para outra e como serão empregados para fazer a modelagem de aspectos dinâmicos do sistema.

Capítulo 5

Implementação

Neste capítulo está listado o código fonte do programa propriamente dito.

5.1 Código fonte

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

Apresenta-se na listagem 5.1 o arquivo com código da classe CPoco.

Listing 5.1: Arquivo de cabeçalho da classe CPoco.h.

```
///Condicao para nao definir a classe mais de uma vez
#ifndef CPoco_h
#define CPoco_h

class CPoco;

///Classe contendo as caracteristicas do poco
class CPoco
{
    ///privados, so acessados por meio de funcoes get
    private:

        ///raio do poco
        double raioPoco;
        ///vazao de producao
        double vazao;
        ///tempo de producao
        double tempoProducao;
        ///pressao no poco
        double pressaoPoco;

    public:
```

```

    ///Funcao que recebe dados do usuario, preenchendo atributos
        raiopoco, vazao, tempoproducao, pressaopoco
void EntradaDados();

    ///Funcao que verifica se houve erro na entrada de dados e pede
        nova entrada ate nao ocorrer erro
void Erro();

    ///Funcao que seta o raiopoco
void RaioPoco(double _raioPoco);
    ///Funcao get do raiopoco
double RaioPoco() const;

    ///Funcao que seta a vazao
void Vazao(double _vazao);
    ///Funcao get da vazao
double Vazao() const;

    ///Funcao que seta o tempoproducao
void TempoProducao(double _tempoProducao);
    ///Funcao get do tempoproducao
double TempoProducao() const;

    ///Funcao que seta a pressaopoco
void PressaoPoco(double _pressaopoco);
    ///Funcao get da pressaopoco
double PressaoPoco() const;

};
#endif

```

Apresenta-se na listagem 5.2 o arquivo de implementação da classe CPoco.

Listing 5.2: Arquivo de implementação da classe CPoco.cpp.

```

#include "CPoco.h"

    ///inclui a biblioteca iostream pois usa funcoes de entrada e saida de
        dados para a tela
#include <iostream>

    ///usa funcoes pertencentes ao namespace std
using namespace std;

    ///funcao de entrada de dados da classe
void CPoco::EntradaDados()
{
    cout << "Informe a vazao de producao: " << endl;
    cin >> vazao;
    cin.get();
}

```

```

    cout << "Informe o tempo de producao: " << endl;
    cin >> tempoProducao;
    cin.get();

    cout << "Informe a pressao no poco: " << endl;
    cin >> pressaoPoco;
    cin.get();

    cout << "Informe o raio do poco: " << endl;
    cin >> raioPoco;
    cin.get();
}

//funcao que acusa e conserta erro de entrada
void CPoco::Erro()
{
    //repete a entrada enquanto o valor for equivocado

    while (tempoProducao<0.00)
    {
        cout << "Reinforme o tempo de producao: " << endl;
        cin >> tempoProducao;
        cin.get();
    }

    while (pressaoPoco<0.00)
    {
        cout << "Reinforme a pressão no poco: " << endl;
        cin >> pressaoPoco;
        cin.get();
    }

    while ((raioPoco<0.00) || (raioPoco>3.0))
    {
        cout << "Reinforme o raio do poco: " << endl;
        cin >> raioPoco;
        cin.get();
    }
}

//set
void CPoco::Vazao(double _vazao)
{
    vazao = _vazao;
}

```



```
//get
double CPoco::Vazao() const
{
    return vazao;
}

//set
void CPoco::TempoProducao(double _tempoProducao)
{
    tempoProducao = _tempoProducao;
}

//get
double CPoco::TempoProducao() const
{
    return tempoProducao;
}

//set
void CPoco::PressaoPoco(double _pressaoPoco)
{
    pressaoPoco = _pressaoPoco;
}

//get
double CPoco::PressaoPoco() const
{
    return pressaoPoco;
}

//set
void CPoco::RaioPoco(double _raioPoco)
{
    raioPoco = _raioPoco;
}

//get
double CPoco::RaioPoco() const
{
    return raioPoco;
}
```

Apresenta-se na listagem 5.3 o arquivo com código da classe CReservatorio.

Listing 5.3: Arquivo de cabeçalho da classe CReservatorio.h.

```
///Condicao para nao definir a classe mais de uma vez
#ifndef CReservatorio_h
#define CReservatorio_h
```

```
class CReservatorio;

///Classe contendo as caracteristicas do reservatorio
class CReservatorio
{
    ///privados, so acessados por meio de funcoes get
    private:

        ///porosidade do reservatorio
        double porosidade;
        ///altura do reservatorio
        double altura;
        ///sistema de unidades utilizado do reservatorio
        double sistemaUnidade;

public:

    ///Funcao que recebe dados do usuario, preenchendo atributos
        porosidade e altura
    void EntradaDados();

    ///Funcao que exibe um menu e recebe dado do usuario, esse dado
        preenche o atributo sistemaunidade
    void EntradaSistemaUnidades();

    ///Funcao que verifica se houve erro na entrada de dados e pede
        nova entrada ate nao ocorrer erro
    void Erro();

    ///Funcao que seta a porosidade
    void Porosidade(double _porosidade);
    ///Funcao get da porosidade
    double Porosidade() const;

    ///Funcao que seta a altura
    void Altura(double _altura);
    ///Funcao get da porosidade
    double Altura() const;

    ///Funcao que seta o sistemaunidade
    void SistemaUnidade(double _sistemaUnidade);
    ///Funcao get do sistemaunidade
    double SistemaUnidade() const;

};

#endif
```

Apresenta-se na listagem 5.4 o arquivo de implementação da classe CReservatorio.

Listing 5.4: Arquivo de implementação da classe CReservatorio.cpp.

```
#include "CReservatorio.h"

//inclui a biblioteca iostrem pois usa funcoes de entrada e saida de
    dados para a tela
#include <iostream>

//usa funcoes pertencentes ao namespace std
using namespace std;

//funcao de entrada de dados da classe
void CReservatorio::EntradaDados()
{

    cout << "Informe a porosidade da rocha reservatorio: " << endl;
    cin >> porosidade;
    cin.get();

    cout << "Informe a altura do reservatorio: " << endl;
    cin >> altura;
    cin.get();

}

//Funcao que preenche o sistema unidade por um pequeno menu
void CReservatorio::EntradaSistemaUnidades()
{
    cout << "Qual o sistema de unidades utilizado para informar os
        parametros: " << endl <<
        "1- Americano (Oilfield)" << endl << "2- Brasileiro (Petrobras)
        " << endl <<
        "3- Sistema Internacional" << endl;
    int i;
    cin >> i;
    cin.get();

    //repete a entrada enquanto o valor for equivocado
    while ((i!=1)&&(i!=2)&&(i!=3))
    {
        cout << "Reinforme o sistema de unidades utilizado." << endl;
        cin>>i;
        cin.get();
    }

    if(i==1)
        sistemaUnidade = 141.2;
```

```
        if(i==2)
            sistemaUnidade = 19.03;

        if (i==3)
            sistemaUnidade = 0.3183;
    }

    //funcao que acusa e conserta erro de entrada
    void CReservatorio::Erro()
    {
        //repete a entrada enquanto o valor for equivocado
        while (altura<0.00)
        {
            cout << "Reinforme a altura: " << endl;
            cin >> altura;
            cin.get();
        }

        while ((porosidade<0.00) || (porosidade>1.00))
        {
            cout << "Reinforme a porosidade: " << endl;
            cin >> porosidade;
            cin.get();
        }
    }

    //set
    void CReservatorio::Porosidade(double _porosidade)
    {
        porosidade = _porosidade;
    }

    //get
    double CReservatorio::Porosidade() const
    {
        return porosidade;
    }

    //set
    void CReservatorio::Altura(double _altura)
    {
        altura = _altura;
    }

    //get
    double CReservatorio::Altura() const
```

```

{
    return altura;
}

//set
void CReservatorio::SistemaUnidade(double _sistemaUnidade)
{
    sistemaUnidade = _sistemaUnidade;
}

//get
double CReservatorio::SistemaUnidade() const
{
    return sistemaUnidade;
}

```

Apresenta-se na listagem 5.5 o arquivo com código da classe CFluido.

Listing 5.5: Arquivo de cabeçalho da classe CFluido.h.

```

///Condicao para nao definir a classe mais de uma vez
#ifndef CFluido_h
#define CFluido_h

class CFluido;

///Classe contendo as caracteristicas do fluido produzido
class CFluido
{
    ///privados, so acessados por meio de funcoes get
private:

    ///Fator Volume formacao do fluido
    double fatorVolumeFormacao;
    ///Viscosidade
    double viscosidade;
    ///Compressibilidade Total
    double compressibilidade;

public:

    ///Funcao que recebe dados do usuario, preenchendo atributos
    fatorvolumeformacao, viscosidade,
    /// compressibilidade
    void EntradaDados();

    ///Funcao que verifica se houve erro na entrada de dados e pede
    nova entrada ate nao ocorrer erro
    void Erro();
}

```

```

    ///Funcao que seta o fatorvolumeformacao
    void FatorVolumeFormacao(double _fatorVolumeFormacao);
    ///Funcao get do fatorvolumeformacao
    double FatorVolumeFormacao() const;

    ///Funcao que seta a viscosidade
    void Viscosidade(double _viscosidade);
    ///Funcao get da viscosidade
    double Viscosidade() const;

    ///Funcao que seta a compressibilidade
    void Compressibilidade(double _compressibilidade);
    ///Funcao get da compressibilidade
    double Compressibilidade() const;

};
#endif

```

Apresenta-se na listagem 5.6 o arquivo de implementação da classe CFluido.

Listing 5.6: Arquivo de implementação da classe CFluido.cpp.

```

#include "CFluido.h"

//inclui a biblioteca iostream pois usa funcoes de entrada e saida de
    dados para a tela
#include <iostream>

using namespace std;

//funcao de entrada de dados da classe
void CFluido::EntradaDados()
{
    cout << "Informe o fator volume-formacao do fluido: " << endl;
    cin >> fatorVolumeFormacao;
    cin.get();

    cout << "Informe a viscosidade do fluido: " << endl;
    cin >> viscosidade;
    cin.get();

    cout << "Informe a compressibilidade total (fluido+rocha): " << endl;
    cin >> compressibilidade;
    cin.get();
}

//funcao que acusa e conserta erro de entrada
void CFluido::Erro()
{

```

```
//repete a entrada enquanto o valor for equivocado

while (fatorVolumeFormacao<0.00)
{
    cout << "Reinforme o Fator Volume-Formacao:" << endl;
    cin >> fatorVolumeFormacao;
    cin.get();
}

while (viscosidade<0.00)
{
    cout << "Reinforme a viscosidade." << endl;
    cin >> viscosidade;
    cin.get();
}

while (compressibilidade<0.00)
{
    cout << "Reinforme a compressibilidade." << endl;
    cin >> compressibilidade;
    cin.get();
}
}

//set
void CFluido::FatorVolumeFormacao(double _fatorVolumeFormacao)
{
    fatorVolumeFormacao = _fatorVolumeFormacao;
}

//get
double CFluido::FatorVolumeFormacao() const
{
    return fatorVolumeFormacao;
}

//set
void CFluido::Viscosidade(double _viscosidade)
{
    viscosidade = _viscosidade;
}

//get
double CFluido::Viscosidade() const
{
    return viscosidade;
}
```

```

}

//set
void CFluido::Compressibilidade(double _compressibilidade)
{
    compressibilidade = _compressibilidade;
}

//get
double CFluido::Compressibilidade() const
{
    return compressibilidade;
}

```

Apresenta-se na listagem 5.7 o arquivo com código da classe CDadosRegistradorPressao.

Listing 5.7: Arquivo de cabeçalho da classe CDadosRegistradorPressao.h.

```

///Condicao para nao definir a classe mais de uma vez
#ifndef CDadosRegistradorPressao_h
#define CDadosRegistradorPressao_h

///inclui a biblioteca vector pois ha declaracao de vetor
#include<vector>
#include<string>

class CDadosRegistradorPressao;

///Classe que contem dados registrados do registrador de pressao
class CDadosRegistradorPressao
{
    ///privados, so acessados por meio de funcoes get
private:
    ///pressao medida apos o fechamento da producao
    std::vector<double> pressaoMedida;
    ///tempo apos o fechamento da producao em que foi medida a
    pressao
    std::vector<double> tempoSemProducao;

public:

    ///Funcao que importa os dados registrados do arquivo .dat,
    preenchendo os atributos da classe
    void Importa();

    ///Funcao que seta a pressaomedida
    void PressaoMedida(std::vector<double> _pressaoMedida);
    ///Funcao get da posicao informada do vetor pressaomedida
    double PressaoMedida(int posicao) const;
    ///Funcao get da pressaomedida

```



```

        std::vector<double> PressaoMedida() const;

        ///Funcao que seta o temposemproducao
        void TempoSemProducao(std::vector<double> _tempoSemProducao);
        ///Funcao get da posicao informada do vetor temposemproducao
        double TempoSemProducao(int posicao) const;
        ///Funcao get do temposemproducao
        std::vector<double> TempoSemProducao() const;

};
#endif

```

Apresenta-se na listagem 5.8 o arquivo de implementação da classe CDadosRegistradorPressao.

Listing 5.8: Arquivo de implementação da classe CDadosRegistradorPressao.cpp.

```

#include "CDadosRegistradorPressao.h" //inclui o cabeçalho da classe

//inclui biblioteca para importacao de arquivos de disco
#include <fstream>

//inclui a biblioteca iostream pois usa funcoes cin, cout
#include <iostream>

//inclui a biblioteca vector pois usa vetores
#include <vector>

//inclui a biblioteca string pois usa variaveis string
#include <string>

//usa funcoes pertencentes ao namespace std
using namespace std;

void CDadosRegistradorPressao::Importa()
{
    //limpa os vetores de importacao para novo preenchimento
    tempoSemProducao.resize(0);
    pressaoMedida.resize(0);

    //indica o que o eixo x representa
    string eixoX;
    //indica o que o eixo y representa
    string eixoY;
    double x;
    double y;
    //nome do arquivo com os dados a serem importados
    string nomeArquivo;

    cout << "Informe o nome do arquivo com os dados do registrador de  

    pressao: " << endl;

```

```

//armazena a string digitada em nomearquivo
getline (cin,nomeArquivo);

//cria objeto de importacao
ifstream fin;
//converte a string de c++ para c, necessario para funcao
fin.open (nomeArquivo.c_str());

//pega o primeiro valor, o nome do eixo x
fin >> eixox;
//pega o segundo valor, o nome do eixo y
fin >> eixoy;

//fazer ate o fim do arquivo
while (!fin.eof())
{
//valores de x e y alternados e separados por um espaco
    fin >> x;
    //para adicionar no fim do vetor, otimizando memoria
    tempoSemProducao.push_back (x);
    fin >> y;
    pressaoMedida.push_back (y);
}
}

//set
void  CDadosRegistradorPressao::PressaoMedida(vector<double>
    _pressaoMedida)
{
    pressaoMedida = _pressaoMedida;
}

//get da posicao
double  CDadosRegistradorPressao::PressaoMedida(int posicao) const
{
    return pressaoMedida[posicao];
}

//get
vector<double>  CDadosRegistradorPressao::PressaoMedida() const
{
    return pressaoMedida;
}

//set
void  CDadosRegistradorPressao::TempoSemProducao(vector<double>
    _tempoSemProducao)
{

```

```

        tempoSemProducao = _tempoSemProducao;
    }

    //get da posicao
    double CDadosRegistradorPressao::TempoSemProducao(int posicao) const
    {
        return tempoSemProducao[posicao];
    }

    //get
    vector<double> CDadosRegistradorPressao::TempoSemProducao() const
    {
        return tempoSemProducao;
    }

```

Apresenta-se na listagem 5.9 o arquivo com código da classe CEstadística.

Listing 5.9: Arquivo de cabeçalho da classe CEstadística.h.

```

///Condicao para nao definir a classe mais de uma vez
#ifndef CEstadistica_h
#define CEstadistica_h

///inclui vector pois ha parametros declarados que sao vetores
#include <vector>

class CEstadistica;

///Classe que calcula estatisticas do vetor, como media e desvio padrao,
    util para regressao
class CEstadistica
{
private:

    double media;
    double desvio;

public:

    ///retorna a media do vetor informado
    double Media(std::vector<double> v);

    ///retorna o desvio padrao do vetor informado
    double DesvioPadrao(std::vector<double> v);

};

#endif

```

Apresenta-se na listagem 5.10 o arquivo de implementação da classe CEstadística.

Listing 5.10: Arquivo de implementação da classe CEstatistica.cpp.

```
#include "CEstatistica.h"

//inclui a biblioteca vector pois usa a funcao size
#include <vector>

//inclui a biblioteca cmath pois usa a funcao pow
#include <cmath>

using namespace std;

double CEstatistica::Media(vector<double> v)
{
    double soma = 0.0;
    //loop que faz a soma de todos os elementos do vetor
    for ( int i = 0 ; i < (v.size()) ; i++)
        soma = soma + v[i];

    return media = soma/v.size();
}

double CEstatistica::DesvioPadrao(vector<double> v)
{
    double soma = 0.0;
    double vquadrado = 0.0;
    desvio = 0.0;

    //loop que faz a soma dos elementos do vetor elevados ao quadrado
    for ( int i = 0 ; i < (v.size()) ; i++)
    {
        soma = soma + v[i];
        vquadrado = vquadrado + (v[i]*v[i]);
    }

    return desvio = pow(((vquadrado - ((1.0/v.size())*soma*soma))/(v.size()
        -1.0)),0.5);
}
```

Apresenta-se na listagem 5.11 o arquivo com código da classe CAjusteCurvaMinimosQuadrados.

Listing 5.11: Arquivo de cabeçalho da classe CAjusteCurvaMinimosQuadrados.h.

```
///Condicao para nao definir a classe mais de uma vez
#ifndef CAjusteCurvaMinimosQuadrados_h
#define CAjusteCurvaMinimosQuadrados_h

//inclui o cabeçalho da classe que sera utilizada
#include "CEstatistica.h"
```

```

///inclui vector pois ha parametros declarados que sao vetores
#include <vector>

class CAjusteCurvaMinimosQuadrados;

///Classe que obtem os coeficiente da regressao linear por meio do
metodo dos minimos quadrados
class CAjusteCurvaMinimosQuadrados
{

    ///encapsulamento que permite o acesso para a classe e para a classe
    herdeira
protected:
    ///coeficiente angular da reta do tipo y=ax+b
    double coeficienteAngular;
    ///coeficiente linear da reta do tipo y=ax+b
    double coeficienteLinear;
    ///coeficiente de correlacao da reta do tipo y=ax+b
    double coeficienteCorrelacao;

    ///cria um objeto da classe CEstatistica para ser utilizado
    funcoes de calculo
    CEstatistica estatistica;

protected:

    ///Funcao que retorna o valor do coeficiente angular
    double CoeficienteAngular (std::vector<double> vx, std::vector<
        double> vy);

    ///Funcao que retorna o valor do coeficiente linear
    double CoeficienteLinear (std::vector<double> vx, std::vector<
        double> vy);

    ///Funcao que retorna o valor do coeficiente de correlacao
    double CoeficienteCorrelacao (std::vector<double> vx, std:::
        vector<double> vy);

};

#endif

```

Apresenta-se na listagem 5.12 o arquivo de implementação da classe CAjusteCurvaMinimosQuadrados.

Listing 5.12: Arquivo de implementação da classe CAjusteCurvaMinimosQuadrados.cpp.

```

#include "CAjusteCurvaMinimosQuadrados.h"

```

```

///inclui a biblioteca cmath pois usa a funcao logaritmica
#include <cmath>

```

```

//inclui a biblioteca vector pois usa funcoes dos vetores: push_back,
    resize
#include <vector>

//usa funcoes pertencentes ao namespace std
using namespace std;

//retorna o valor do coeficiente angular
double CAjusteCurvaMinimosQuadrados::CoeficienteAngular (vector<double>
    vx, vector<double> vy)
{
    double mnum = 0.0; //termo do denominador
    double mden = 0.0; //termo do numerador

    double mediax = estatistica.Media(vx);
    double mediay = estatistica.Media(vy);

    for (int j=0 ; j<vx.size() ; j++) //percorre todo o vetor
    {
        //metodo dos minimos quadrados
        mnum = mnum + (vx[j] * (vy[j] - mediay));
        mden = mden + (vx[j] * (vx[j] - mediax));
    }

    return coeficienteAngular = mnum/mden;
}

//retorna o valor do coeficiente linear
double CAjusteCurvaMinimosQuadrados::CoeficienteLinear (vector<double>
    vx, vector<double> vy)
{
    return coeficienteLinear = estatistica.Media(vy) - (
        coeficienteAngular * estatistica.Media(vx));;
}

//retorna o valor do coeficiente de correlacao
double CAjusteCurvaMinimosQuadrados::CoeficienteCorrelacao(vector<double>
    > vx, vector<double> vy)
{
    double somar = 0.0;
    double variax = 0.0;
    double variay = 0.0;

    double mediax = estatistica.Media(vx);
    double mediay = estatistica.Media(vy);

```

```

    for (int j=0 ; j<vx.size() ; j++)
    {
        somar = somar + ((vx[j] - mediox) * (vy[j] - mediy));
        variax = variax + (pow ((vx[j] - mediox),2));
        variay = variay + (pow ((vy[j] - mediy),2));
    }

    return coeficienteCorrelacao = -somar / pow ((variax * variay)
        ,0.5);
}

```

Apresenta-se na listagem 5.13 o arquivo com código da classe CAjusteCurva.

Listing 5.13: Arquivo de cabeçalho da classe CAjusteCurva.h.

```

///Condicao para nao definir a classe mais de uma vez
#ifndef CAjusteCurva_h
#define CAjusteCurva_h

///inclui a biblioteca vector pois ha declaracao de vetor
#include <vector>

///inclui o cabecalho da classe pai
#include "CAjusteCurvaMinimosQuadrados.h"

///Declaracao da Classe filha de CAjusteCurvaMinimosQuadrados
class CAjusteCurva;

///Classe que executa a regressao linear e ajusta ate a correlacao
satisfatoria
class CAjusteCurva: public CAjusteCurvaMinimosQuadrados
{
public:
    ///coef. angular
    double m;
    ///coef. linear
    double n;
    ///coef. de correlacao
    double r;
    ///indica que o coef. de correlacao nao foi satisfatorio (ha
    estocagem)
    bool efeitoEstocagem;
    ///ajuste de variavel para logaritmica
    std::vector<double> logt;

public:
    ///Funcao que executa a regressao linear atraves do metodo dos
    minimos quadrados
    void Regressao(std::vector<double> vx, std::vector<double> vy,
        double z);

```

```

        ///Funcao que ajusta a regressao para o periodo correto,
        removendo os pontos referentes a estocagem
        void Ajuste(std::vector<double> vx, std::vector<double> vy,
            double z);

};

///Fim da condicao de definicao da classe
#endif

```

Apresenta-se na listagem 5.14 o arquivo de implementação da classe CAjusteCurva.

Listing 5.14: Arquivo de implementação da classe CAjusteCurva.cpp.

```

#include "CAjusteCurva.h"

///inclui a biblioteca iostream pois usa funcoes de entrada e saida de
dados para a tela
#include <iostream>

///inclui a biblioteca cmath pois usa a funcao logaritmica
#include <cmath>

///inclui a biblioteca vector pois usa funcoes dos vetores: push_back,
resize
#include <vector>

///usa funcoes pertencentes ao namespace std
using namespace std;

/// Funcao que cria a variavel logaritmica a partir dos parametros 1 e 3
da funcao, executa a
regressao linear atraves do metodo dos minimos quadrados.
void CAjusteCurva::Regressao(vector<double> vx, vector<double> vy,
    double z)
{
    ///limpa o vetor do eixo x para novo preenchimento
    logt.resize(0);

    ///loop que percorre todo o vetor vx e preenche logt
    for(int i=0 ; i<vx.size() ; i++)
        ///transformacao da variavel em logaritmica
        logt.push_back (log10((z+vx[i]) / vx[i]));

    m = CoeficienteAngular (logt,vy);
    n = CoeficienteLinear (logt,vy);
    r = CoeficienteCorrelacao (logt,vy);

    cout << "EQUACAO: y = " << m << " * x + " << n << endl << "r = " << r
        << endl;
}

```



```

}

//Funcao que ajusta a regressao para o periodo correto, removendo os
    pontos referentes a estocagem
void CAjusteCurva::Ajuste(vector<double> vx, vector<double> vy, double z
    )
{
    // variavel que contem os coef. de correlacao
    vector<double> coef(vx.size()/2,r);

    //variavel que ajusta o eixo y
    vector<double> y;

    //variavel que ajusta o eixo x
    vector<double> t;

    //loop principal que vai aumentando o valor de k e retirando as
        primeiras posicoes dos vetores (estocagem)
    for (int k=1 ; k<(vx.size()/2) ; k++)
    {
        //repete o loop ate achar o coef. de correlacao aceitavel
        if(coef[k-1]<0.9900)
        {
            //ocorre estocagem se cair na condicao
            efeitoEstocagem = true;
            cout << "Necessario_novo_Ajuste." << endl;

            //limpa os vetores
            t.resize(0);
            y.resize(0);

            for(int l=0 ; l<vx.size() ; l++)
            {
                //define o eixo x
                t.push_back (log10((z + vx[l])/vx[l]));
                //define o eixo y
                y.push_back (vy[l]);
            }

            for(int w=0 ; w<(vx.size()-k) ; w++)
            {
                //retira o primeiro valor do vetor (estocagem)
                t[w] = t[w+k];
                //se repetir o if, vai retirando
                y[w] = vy[w+k];
                //até terminar a estocagem (coef. sera bom)
            }
        }
    }
}

```

```

//redefine o tamanho dos vetores
t.resize (vx.size()-k);
y.resize (vx.size()-k);

//nova regressao linear
m = CoeficienteAngular (t,y);
n = CoeficienteLinear (t,y);
//novo coeficiente de correlacao
coef[k] = CoeficienteCorrelacao (t,y);

cout << "EQUACAO: y=" << m << " * x + " << n << endl <<
        "r=" << coef[k] << endl;

if (1.2*coef[k]<coef[0])
    cout << "Regressao Linear nao tao perfeita,
            Indicativo de Reservatorio Heterogeneo" <<
            endl;

if (k>1) //apos a segunda regressao
{
    if ((1.2*coef[k])<coef[k-1])
    {
        k = vx.size()/2; //para terminar o loop
        cout << "Maximo Coeficiente de Correlacao
                alcançado." << endl;
    }
}

} // Fecha a condicao inicial.
} // Fecha loop.
} // Fecha o Metodo.

```

Apresenta-se na listagem 5.15 o arquivo com código da classe CCaracterizacaoReservatorio.

Listing 5.15: Arquivo de cabeçalho da classe CCaracterizacaoReservatorio.h.

```

///Condicao para nao definir a classe mais de uma vez
#ifndef CCaracterizacaoReservatorio_h
#define CCaracterizacaoReservatorio_h

class CCaracterizacaoReservatorio;

///Classe que caracteriza o reservatorio.
class CCaracterizacaoReservatorio
{
public:
    ///Funcao que analisa os resultados e caracteriza o reservatorio.

```

```

        void Caracterizacao(double permeabilidade, double fatorpelicula,
                             double indiceprodutividade, double raio pouco, double
                             raioefetivo);

};
#endif

```

Apresenta-se na listagem 5.16 o arquivo de implementação da classe CCaracterizacao.

Listing 5.16: Arquivo de implementação da classe CCaracterizacaoReservatorio.cpp.

```

#include "CCaracterizacaoReservatorio.h"

//inclui a biblioteca iostream pois usa funcoes de entrada e saida de
//dados para a tela
#include <iostream>

//usa funcoes pertencentes ao namespace std
using namespace std;

//Funcao que caracteriza o reservatorio, dado os parametros calculados
void CCaracterizacaoReservatorio::Caracterizacao (double permeabilidade,
double fatorpelicula, double indiceprodutividade, double raio pouco,
double raioefetivo)
{
    //condicoes que se satisfeitas, escrevem na tela caracteristicas do
    //reservatorio.

    if (permeabilidade<=10)
        cout << "1-Reservatorio com permeabilidade ruim." << endl;

    if ((permeabilidade<100)&&(permeabilidade>10))
        cout << "1-Reservatorio com permeabilidade boa." << endl;

    if (permeabilidade>=100)
        cout << "1-Reservatorio com permeabilidade excelente." << endl;

    if (fatorpelicula==0)
        cout << "2-Reservatorio sem dano e sem estimulo." << endl;

    if (fatorpelicula<0)
        cout << "2-Reservatorio estimulado" << endl;

    if ((fatorpelicula>0)&&(fatorpelicula<5))
        cout << "2-Reservatorio com dano baixo, nao necessita de
        processos de acidificacao e/ou fraturamento hidraulico." << endl
        ;

    if ((fatorpelicula>5)&&(fatorpelicula<10))

```

```

        cout << "2-Reservatorio com dano intermediario, pode ser usado
        processos de acidificacao e/ou fraturamento hidraulico." << endl
        ;

    if (fatorpelicula>10)
        cout << "2-Reservatorio com dano alto, necessita de processos de
        acidificacao e/ou fraturamento hidraulico." << endl;

    if (indiceprodutividade<=0.01)
        cout << "3-Reservatorio com produtividade baixo, considerar uso
        de tecnicas de recuperacao secundaria." << endl;

    if ((indiceprodutividade>0.01)&&(indiceprodutividade<0.1))
        cout << "3-Reservatorio com produtividade regular, considerar uso
        de tecnicas de recuperacao secundaria." << endl;

    if (indiceprodutividade>0.1)
        cout << "3-Reservatorio com produtividade boa." << endl;

    if ((raioefetivo/raiopoco)<=0.0001)
        cout << "4-Razao de dano alto, pois o raio efetivo e muito menor
        que o raio do poco real, afetando a produtividade." << endl;

}

```

Apresenta-se na listagem 5.17 o arquivo com código da classe CSimuladorAnaliseTestePressao.

Listing 5.17: Arquivo de cabeçalho da classe CSimuladorAnaliseTestePressao.h.

```

#ifndef CSimuladorAnaliseTestePressao_h
#define CSimuladorAnaliseTestePressao_h

#include <fstream>
///inclusao de arquivos de classe necessarios

#include "cgplot.h"
#include "CReservatorio.h"
#include "CFluido.h"
#include "CPoco.h"
#include "CDadosRegistradorPressao.h"
#include "CEstatistica.h"
#include "CAjusteCurva.h"
#include "CCaracterizacaoReservatorio.h"

///criacao do objeto caracterizar da classe CCaracterizacao.h

class CSimuladorAnaliseTestePressao;

```

*///Classe que faz a analise do teste de pressao realizado no campo e
infere as propriedades do reservatorio*

```
class CSimuladorAnaliseTestePressao
{
```

```
public:
```

```
    ///Nome do Arquivo exportado
    std::string nome;
    ///permeabilidade do reservatorio
    double permeabilidade;
    ///pressao inicial que se encontrava o reservatorio
    double pressaoInicial;
    ///skin factor do poço
    double fatorPelícula;
    ///queda de pressao referente ao fator de película
    double pressaoDano;
    ///indice de produtividade do reservatorio
    double indiceProdutividade;
    ///eficiencia de fluxo do reservatorio
    double eficienciaFluxo;
    ///raio efetivo do poço
    double raioEfetivo;
    ///coeficiente de estocagem do poço
    double coeficienteEstocagem;
    ///tempo de duracao do efeito de estocagem
    double tempoEstocagem;
    ///cria objeto de armazenamento de dados
    std::ofstream fout;
    ///criacao do objeto poço da classe CPoco
    CPoco poco;
    ///criacao do objeto fluido da classe CFluido
    CFluido fluido;
    ///criacao do objeto reservatorio da classe CReservatorio
    CReservatorio reservatorio;
    ///criacao do objeto registrador da classe CRegistrador
    CDadosRegistradorPressao registrador;
    ///criacao do objeto ajuste da classe CAjuste
    CAjusteCurva ajuste;
    ///criacao do objeto caracterizar da classe CCaracterizacao
    CCaracterizacaoReservatorio caracterizar;
    ///criacao do objeto plot da classe CGnuplot
    //CGnuplot plot;
```

```
public:
```

```
    ///Funcao que calcula e preenche o atributo permeabilidade
    void CalculoPermeabilidade ();
    ///Funcao que calcula e preenche o atributo pressao inicial
```

```

void CalculoPressaoInicial ();
///Funcao que calcula e preenche o atributo fatorpelicula
void CalculoFatorPelicula ();
///Funcao que calcula e preenche o atributo raioefetivo
void CalculoRaioEfetivo ();
///Funcao que calcula e preenche os atributos
    indiceprodutividade, eficienciafluxo, pressaodano
void CalculoIndices ();
///Funcao que calcula e preenche os atributos
    coeficienteestocagem e tempoestocagem
void CalculoEstocagem ();
///Funcao que chama as entradas de dados necessarias
void EntradaDados();
///Funcao principal que executa a simulacao do teste
void Executar();
///Caria parametros de reservatorio
void Variacao();
///Funcao que exporta os dados para um arquivo.dat
void Exporta ();

};

```

```
#endif
```

Apresenta-se na listagem 5.18 o arquivo de implementação da classe CSimuladorAnaliseTestePre

Listing 5.18: Arquivo de implementação da classe CSimuladorAnaliseTestePressao.cpp.

```

#include "CSimuladorAnaliseTestePressao.h"

///inclui a biblioteca iostream pois usa funcoes de entrada e saida de
    dados para a tela
#include <iostream>

///inclui a biblioteca cmath pois usa a funcao logaritmica
#include <cmath>

///inclui a biblioteca vector pois usa funcoes dos vetores: push_back,
    resize
#include <vector>

///inclui a biblioteca que guarda os dados
#include <fstream>

///inclui biblioteca string que le caracteres
#include <string>

///biblioteca que permite manipulaÃ§Ã£o de variaveis
#include <sstream>

```

```

//usa funcoes pertencentes ao namespace std
using namespace std;

///Funcao principal que executa a simulacao do teste
void CSimuladorAnaliseTestePressao::Executar()
{
    cout << endl << "PROGRAMA_PARA_CALCULO_DE_PARAMETROS_DE_RESERVATORIO
        _POR_TESTES_DE_PRESSAO" << endl
        << endl << "1-Rodar_o_Programa" << endl << "2-Sair" << endl <<
        endl;
    int i = 0;
    cin >> i;

    while (i==1) //quando terminar a execucao do programa, se o usuario
        quiser, o programa roda novamente
    {
        cout << "Entrada_de_Dados_do_teste_de_pressao_realizado_" <<
            endl
            << "-----"
            << endl;

        EntradaDados();

        cout << "Entrada_de_Dados_Finalizada" << endl
            << "-----"
            << endl << endl;
        cout << "Importacao_dos_dados_do_registrador_de_Pressao" <<
            endl
            << "-----"
            << endl << endl;

        registrador.Importa();
        cout << "Dados_do_registrador_importados_com_sucesso" <<
            endl << endl;

        cout << "Regressao_Linear_dos_Dados" << endl
            << "-----"
            << endl << endl;

        ajuste.Regressao (registrador.TempoSemProducao(),
            registrador.PressaoMedida(), poco.TempoProducao());

        cin.get ();
        cout << "Localizando_o_Periodo_Transiente" << endl
            << "-----"
            << endl << endl;

        ajuste.Ajuste(registrador.TempoSemProducao(), registrador.

```

```

    PressaoMedida(), poco.TempoProducao());

cout << "Regressao_linear_feita_com_sucesso" << endl
    << "-----"
    << endl;

//GERA O GRAFICO CASO USUARIO QUEIRA

cout << "Deseja_gerar_o_grafico:" << endl << "1-Sim" <<
    endl << "2-Nao" << endl << endl << endl;
int j;
cin >> j;

if (j==1)
{
    CGnuplot plot;
    //gera o grafico com a reta perfeita obtida
    plot.plot_slope (ajuste.m,ajuste.n);
    cin.get();
    if (ajuste.efeitoEstocagem==1)
    //compara com os pontos originais
    plot.plot_xy (ajuste.logt,registrador.
        PressaoMedida());
    cin.get();
}

cout << "Parametros_do_Reservatorio" << endl
    << "-----"
    << endl << endl;
//CALCULOS
CalculoPermeabilidade();
CalculoPressaoInicial();
CalculoFatorPelícula(); //guardar
CalculoRaioEfetivo();
CalculoIndices();
Exporta();
cin.get();

//se nao ocorreu estocagem
if (ajuste.efeitoEstocagem==false)
    cout << "Reservatorio_sem_o_periodo_de_estocagem" <<
        endl;
else
{
    CalculoEstocagem ();
    //zera o valor em caso de novo calculo
    ajuste.efeitoEstocagem = false;
}

```



```

    }

    cout << "Caracterizacao do Reservatorio." << endl
    << "-----"
    << endl;

    caracterizar.Caracterizacao(permeabilidade, fatorPelícula,
        indiceProdutividade, poco.RaioPoco(), raioEfetivo);
    cin.get();

    Variacao();

    //Nova Escolha
    cout << "\n1-Rodar o Programa" << endl << "2-Sair" << endl
    << endl;
    cin >> i;
}

}

//chama as outras entradas de dados
void CSimuladorAnaliseTestePressao::EntradaDados()
{
    reservatorio.EntradaSistemaUnidades();
    reservatorio.EntradaDados();
    reservatorio.Erro();
    fluido.EntradaDados();
    fluido.Erro();
    poco.EntradaDados();
    poco.Erro();
}

//Calcula a permeabilidade
void CSimuladorAnaliseTestePressao::CalculoPermeabilidade ()
{
    permeabilidade = (1.151 * reservatorio.SistemaUnidade() * poco.
        Vazao() * fluido.FatorVolumeFormacao() *
        fluido.Viscosidade()) / (-ajuste.m * reservatorio.
        Altura());

    cout << "Permeabilidade: " << permeabilidade;

    if ((reservatorio.SistemaUnidade()==141.2) || (reservatorio.
        SistemaUnidade()==19.03))
        cout << " milidarcy" << endl;

    if (reservatorio.SistemaUnidade()==0.3183)
        cout << " metros quadrados" << endl; //criacao do objeto
        armazena da classe CSimuladorAnaliseTestePressao

```

```

}

//Calcula a pressao inicial
void CSimuladorAnaliseTestePressao::CalculoPressaoInicial()
{
    pressaoInicial = ajuste.n;
    cout << "Pressao_Inicial:" << pressaoInicial;

    if (reservatorio.SistemaUnidade()==141.2)
        cout << "_psi" << endl;

    if (reservatorio.SistemaUnidade()==19.03)
        cout << "_kgf/cm2" << endl;

    if (reservatorio.SistemaUnidade()==0.3183)
        cout << "_Pascal" << endl;

}

//Calcula fator pelicula
void CSimuladorAnaliseTestePressao::CalculoFatorPelicula ()
{
    fatorPelicula = 1.151 * (((ajuste.m * log10(poco.TempoProducao()
        )) + ajuste.n - poco.PressaoPoco())/
        -ajuste.m) - log10((reservatorio.SistemaUnidade()
        * permeabilidade) /
        (reservatorio.Porosidade() * fluido.Viscosidade()
        * fluido.Compressibilidade()
        * poco.RaioPoco () * poco.RaioPoco())) - 0.3514 +
        log10 (poco.TempoProducao()+1));

    cout << "Fator_de_Pelicula:" << fatorPelicula << endl;
}

//Calcula raio efetivo
void CSimuladorAnaliseTestePressao::CalculoRaioEfetivo()
{
    raioEfetivo = poco.RaioPoco() * exp(-fatorPelicula);
    cout << "Raio_Efetivo:" << raioEfetivo;

    if ((reservatorio.SistemaUnidade()==0.3183) || (reservatorio.
        SistemaUnidade()==19.03))
        cout << "_metros" << endl;
}

```

```

        if (reservatorio.SistemaUnidade()==141.2)
            cout << "␣ft" << endl;
    }

    //Calcula do indice de produtividade, eficiencia de fluxo e queda de
    //pressao referente ao dano
    void CSimuladorAnaliseTestePressao::CalculoIndices ()
    {
        pressaoDano = 0.869 * (-ajuste.m) * fatorPelicula;

        indiceProdutividade = poco.Vazao() / (pressaoInicial - poco.
            PressaoPoco());

        eficienciaFluxo = (pressaoInicial - poco.PressaoPoco() -
            pressaoDano) / (pressaoInicial - poco.PressaoPoco());

        cout << "Queda␣de␣Pressao␣devido␣ao␣dano:␣" << pressaoDano <<
            endl;

        //Exibir em porcentagens
        cout << "Indice␣de␣Produtividade:␣" << indiceProdutividade*100.0
            << "␣%␣" << endl <<
            "Eficiencia␣de␣Fluxo:␣" << eficienciaFluxo*100.0 << "␣%␣" <<
            endl;
    }

    void CSimuladorAnaliseTestePressao::Variacao()
    {
        cout << "-----" << endl;
        cout << "Deseja␣variar␣algum␣parametro?" << endl;
        cout << "1-␣Sim␣|␣2-␣Nao" << endl;
        int resp;
        cin >> resp;
        cin.get();
        do
        {
            if (resp!=1 && resp!=2)
            {
                cout << "Alternativa␣Invalida" << endl;
                cout << "1-␣Sim␣|␣2-␣Nao" << endl;
                cin >> resp;
                cin.get();
            }
        }
        while (resp!=1 && resp!=2);
        if (resp == 1)

```

```

{
    cout << "\nQual parametro deseja variar?" << endl;
    cout << "| 1 - Porosidade | 2 - Fator Volume de Formacao | 3 -  

        Compressibilidade | 4 - Viscosidade |" << endl;
    cin >> resp;
    cin.get();
do
{
    if (resp!=1 && resp!=2 && resp!=3 && resp!=4)
    {
        cout << "Alternativa Invalida" << endl;
        cout << "| 1 - Porosidade | 2 - Fator Volume de Formacao | 3 -  

            Compressibilidade | 4 - Viscosidade |" << endl;
        cin >> resp;
        cin.get();
    }
}
while (resp!=1 && resp!=2 && resp!=3 && resp!=4);
int intervalo;
double dp;
    switch (resp)
    {
        case 1:
            cout << "Digite um valor inicial para porosidade" << endl;
            double porosidadeInicial;
            cin >> porosidadeInicial;
            cin.get();
            while ((porosidadeInicial < 0.00) || (porosidadeInicial > 1.00))
            {
                cout << "Reinforme a porosidade Inicial:" << endl;
                cin >> porosidadeInicial;
                cin.get();
            }
            cout << "Digite um valor Final para porosidade" << endl;
            double porosidadeFinal;
            cin >> porosidadeFinal;
            cin.get();
            while ((porosidadeFinal < 0.00) || (porosidadeFinal > 1.00) || (
                porosidadeInicial > porosidadeFinal))
            {
                cout << "Reinforme a porosidade Final:" << endl;
                cin >> porosidadeFinal;
                cin.get();
            }
            cout << "Em quantos intervalos deseja dividir a porosidade?" <<
                endl;
            cin >> intervalo;
            cin.get();

```

```

        dp=(porosidadeFinal - porosidadeInicial)/intervalo;
        for (int x = porosidadeInicial;porosidadeInicial <= porosidadeFinal;
            porosidadeInicial=porosidadeInicial+dp)
        {
            reservatorio.Porosidade(porosidadeInicial);
        cout << "\nCalculo para porosidade: " << porosidadeInicial << endl;
        cout << "-----" << endl;
            CalculoPermeabilidade();
            CalculoPressaoInicial();
            CalculoFatorPelícula(); //guardar
            CalculoRaioEfetivo();
            CalculoIndices();
        cout << "-----" << endl;

        ostringstream str;
        str << porosidadeInicial;
        string porosidadeString = str.str();

        string formato = ".dat";
        string Saida = nome+"porosidade("+porosidadeString+formato;
            ///abre arquivo
        fout.open (Saida.c_str());

        fout << "Permeabilidade: " << permeabilidade;
        if ((reservatorio.SistemaUnidade()==141.2) || (reservatorio.
            SistemaUnidade()==19.03))
        fout << "milidarcy" << endl;
        if (reservatorio.SistemaUnidade()==0.3183)
            fout << "metros quadrados" << endl; ///criacao do objeto
                armazena da classe CSimuladorAnaliseTestePressao
        fout << "Pressao Inicial: " << pressaoInicial;

        if (reservatorio.SistemaUnidade()==141.2)
            fout << "psi" << endl;

        if (reservatorio.SistemaUnidade()==19.03)
            fout << "kgf/cm2" << endl;

        if (reservatorio.SistemaUnidade()==0.3183)
            fout << "Pascal" << endl;
            fout << "Fator de Película: " << fatorPelícula << endl;
            fout << "Raio Efetivo: " << raioEfetivo;

        if ((reservatorio.SistemaUnidade()==0.3183) || (reservatorio.
            SistemaUnidade()==19.03))
            fout << "metros" << endl;

        if (reservatorio.SistemaUnidade()==141.2)

```

```

        fout << "ft" << endl;
        fout << "Queda de Pressao devido ao dano:" << pressaoDano <<
            endl;

        //Exibir em porcentagens
        fout << "Indice de Produtividade:" << indiceProdutividade*100.0 << "
            %" << endl << "Eficiencia de Fluxo:" << eficienciaFluxo*100.0 << "
            %" << endl;
        fout.close();
        cout << "Dados Salvos com sucesso!" << endl;
        cout << "-----" << endl;
        cin.get();
    }
    break;

    case 2:
        cout << "Digite um valor inicial para fator volume de formacao"
            << endl;
        double FatorVolformacaoInicial;
        cin >> FatorVolformacaoInicial;
        cin.get();
        while ((FatorVolformacaoInicial < 0.00))
        {
            cout << "Reinforme o fator volume de formacao Inicial:" <<
                endl;
            cin >> FatorVolformacaoInicial;
            cin.get();
        }
        cout << "Digite um valor Final para fator volume de formacao" <<
            endl;
        double FatorVolformacaoFinal;
        cin >> FatorVolformacaoFinal;
        cin.get();
        while ((FatorVolformacaoFinal < 0.00) || (FatorVolformacaoInicial >
            FatorVolformacaoFinal))
        {
            cout << "Reinforme o fator volume de formacao Final:" << endl
                ;
            cin >> FatorVolformacaoFinal;
            cin.get();
        }
        cout << "Em quantos intervalos deseja dividir o fator volume de
            formacao?" << endl;
        cin >> intervalo;
        cin.get();
        dp=(FatorVolformacaoFinal - FatorVolformacaoInicial)/intervalo;
        for (int x = FatorVolformacaoInicial; FatorVolformacaoInicial <=
            FatorVolformacaoFinal; FatorVolformacaoInicial+=dp)

```

```

    {
        fluido.FatorVolumeFormacao(FatorVolformacaoInicial);
        cout << "\nCalculo para fator volume de formacao: " <<
            FatorVolformacaoInicial << endl;
        cout << "-----" << endl;
            CalculoPermeabilidade();
            CalculoPressaoInicial();
            CalculoFatorPelícula(); //guardar
            CalculoRaioEfetivo();
            CalculoIndices();
        cout << "-----" << endl;

        ostringstream str;
        str << FatorVolformacaoInicial;
        string FatorVolformacaoString = str.str();

        string formato = ".dat";
        string Saida = nome+"FatorVolF("+FatorVolformacaoString+formato;
            ///abre arquivo
        fout.open (Saida.c_str());

        fout << "Permeabilidade: " << permeabilidade;
        if ((reservatorio.SistemaUnidade()==141.2) || (reservatorio.
            SistemaUnidade()==19.03))
        fout << "milidarcy" << endl;
        if (reservatorio.SistemaUnidade()==0.3183)
            fout << "metros quadrados" << endl; ///criacao do objeto
                armazena da classe CSimuladorAnaliseTestePressao
        fout << "Pressao Inicial: " << pressaoInicial;

        if (reservatorio.SistemaUnidade()==141.2)
            fout << "psi" << endl;

        if (reservatorio.SistemaUnidade()==19.03)
            fout << "kgf/cm2" << endl;

        if (reservatorio.SistemaUnidade()==0.3183)
            fout << "Pascal" << endl;
            fout << "Fator de Película: " << fatorPelícula << endl;
            fout << "Raio Efetivo: " << raioEfetivo;

        if ((reservatorio.SistemaUnidade()==0.3183) || (reservatorio.
            SistemaUnidade()==19.03))
            fout << "metros" << endl;

        if (reservatorio.SistemaUnidade()==141.2)
            fout << "ft" << endl;
            fout << "Queda de Pressao devido ao dano: " << pressaoDano <<

```

```

        endl;

        //Exibir em porcentagens
        fout << "Indice de Produtividade: " << indiceProdutividade*100.0 << "
        %" << endl << "Eficiencia de Fluxo: " << eficienciaFluxo*100.0 << "
        %" << endl;
        fout.close();
        cout << "Dados Salvos com sucesso!" << endl;
        cout << "-----" << endl;
        cin.get();
    }

    break;

    case 3:
        cout << "Digite um valor inicial para Compressibilidade" << endl
        ;
        double CompressibilidadeInicial;
        cin >> CompressibilidadeInicial;
        cin.get();
        while ((CompressibilidadeInicial < 0.00))
        {
            cout << "Reinforme a compressibilidade: " << endl;
            cin >> CompressibilidadeInicial;
            cin.get();
        }
        cout << "Digite um valor Final para Compressibilidade" << endl;
        double CompressibilidadeFinal;
        cin >> CompressibilidadeFinal;
        cin.get();
        while ((CompressibilidadeFinal < 0.00) || (CompressibilidadeInicial
        > CompressibilidadeFinal))
        {
            cout << "Reinforme a Compressibilidade: " << endl;
            cin >> CompressibilidadeFinal;
            cin.get();
        }
        cout << "Em quantos intervalos deseja dividir o
        Compressibilidade?" << endl;
        cin >> intervalo;
        cin.get();
        dp=(CompressibilidadeFinal - CompressibilidadeInicial)/intervalo
        ;
        for (int x = CompressibilidadeInicial; CompressibilidadeInicial <=
        CompressibilidadeFinal; CompressibilidadeInicial+=dp)
        {
            fluido.Compressibilidade(CompressibilidadeInicial);
            cout << "\nCalculo para Compressibilidade: " <<

```



```

        CompressibilidadeInicial << endl;
    cout << "-----" << endl;
        CalculoPermeabilidade();
        CalculoPressaoInicial();
        CalculoFatorPelícula(); //guardar
        CalculoRaioEfetivo();
        CalculoIndices();
    cout << "-----" << endl;

    ostringstream str;
    str << CompressibilidadeInicial;
    string CompressibilidadeString = str.str();

    string formato = ".dat";
    string Saida = nome+"Compressibilidade("+CompressibilidadeString+
        formato;
        ///abre arquivo
    fout.open (Saida.c_str());

    fout << "Permeabilidade:_" << permeabilidade;
        if ((reservatorio.SistemaUnidade()==141.2) || (reservatorio.
            SistemaUnidade()==19.03))
    fout << "_milidarcy" << endl;
        if (reservatorio.SistemaUnidade()==0.3183)
            fout << "_metros_quadrados" << endl; ///criacao do objeto
                armazena da classe CSimuladorAnaliseTestePressao
    fout << "Pressao_Inicial:_" << pressaoInicial;

    if (reservatorio.SistemaUnidade()==141.2)
        fout << "_psi" << endl;

    if (reservatorio.SistemaUnidade()==19.03)
        fout << "_kgf/cm2" << endl;

    if (reservatorio.SistemaUnidade()==0.3183)
        fout << "_Pascal" << endl;
        fout << "Fator_de_Película:_" << fatorPelícula << endl;
        fout << "Raio_Efetivo:_" << raioEfetivo;

    if ((reservatorio.SistemaUnidade()==0.3183) || (reservatorio.
        SistemaUnidade()==19.03))
        fout << "_metros" << endl;

    if (reservatorio.SistemaUnidade()==141.2)
        fout << "_ft" << endl;
        fout << "Queda_de_Pressao_devido_ao_dano:_" << pressaoDano <<
            endl;

```

```

        //Exibir em porcentagens
        fout << "Indice de Produtividade:_" << indiceProdutividade*100.0 << "_\n" << endl << "Eficiencia de Fluxo:_" << eficienciaFluxo*100.0 << "_%\n" << endl;
        fout.close();
        cout << "Dados Salvos com sucesso!" << endl;
        cout << "-----" << endl;
        cin.get();
    }
    break;

    case 4:
        cout << "Digite um valor inicial para Viscosidade" << endl;
        double ViscosidadeInicial;
        cin >> ViscosidadeInicial;
        cin.get();
        while ((ViscosidadeInicial < 0.00))
        {
            cout << "Reinforme a compressibilidade:_" << endl;
            cin >> ViscosidadeInicial;
            cin.get();
        }
        cout << "Digite um valor Final para Viscosidade" << endl;
        double ViscosidadeFinal;
        cin >> ViscosidadeFinal;
        cin.get();
        while ((ViscosidadeFinal < 0.00) || (ViscosidadeInicial > ViscosidadeFinal))
        {
            cout << "Reinforme a Viscosidade:_" << endl;
            cin >> ViscosidadeFinal;
            cin.get();
        }
        cout << "Em quantos intervalos deseja dividir o Viscosidade?" << endl;
        cin >> intervalo;
        cin.get();
        dp=(ViscosidadeFinal - ViscosidadeInicial)/intervalo;
        for (int x = ViscosidadeInicial; ViscosidadeInicial <= ViscosidadeFinal; ViscosidadeInicial+=dp)
        {
            fluido.Viscosidade(ViscosidadeInicial);
            cout << "\nCalculo para Viscosidade:_" << ViscosidadeInicial << endl;
            cout << "-----" << endl;
            CalculoPermeabilidade();
            CalculoPressaoInicial();
            CalculoFatorPelícula(); //guardar
            CalculoRaioEfetivo();
        }
    }
}

```

```

        CalculoIndices();
cout << "-----" << endl;

    ostreamstream strs;
    strs << ViscosidadeInicial;
    string ViscosidadeString = strs.str();

    string formato = ".dat";
    string Saida = nome+"Viscosidade("+ViscosidadeString+formato;
        ///abre arquivo
fout.open (Saida.c_str());

fout << "Permeabilidade:_" << permeabilidade;
    if ((reservatorio.SistemaUnidade()==141.2) || (reservatorio.
        SistemaUnidade()==19.03))
fout << "_milidarcy" << endl;
    if (reservatorio.SistemaUnidade()==0.3183)
        fout << "_metros_quadradados" << endl; ///criacao do objeto
            armazena da classe CSimuladorAnaliseTestePressao
fout << "Pressao_Inicial:_" << pressaoInicial;

    if (reservatorio.SistemaUnidade()==141.2)
        fout << "_psi" << endl;

    if (reservatorio.SistemaUnidade()==19.03)
        fout << "_kgf/cm2" << endl;

    if (reservatorio.SistemaUnidade()==0.3183)
        fout << "_Pascal" << endl;
        fout << "Fator_de_Película:_" << fatorPelícula << endl;
        fout << "Raio_Efetivo:_" << raioEfetivo;

    if ((reservatorio.SistemaUnidade()==0.3183) || (reservatorio.
        SistemaUnidade()==19.03))
        fout << "_metros" << endl;

    if (reservatorio.SistemaUnidade()==141.2)
        fout << "_ft" << endl;
        fout << "Queda_de_Pressao_devido_ao_dano:_" << pressaoDano <<
            endl;

        //Exibir em porcentagens
fout << "Indice_de_Produtividade:_" << indiceProdutividade*100.0 << "_
    %_" << endl << "Eficiencia_de_Fluxo:_" << eficienciaFluxo*100.0 << "
    %_" << endl;
fout.close();
cout << "Dados_Salvos_com_sucesso!" << endl;
cout << "-----" << endl;

```

```

        cin.get();
    }

    break;
}
}
}

void CSimuladorAnaliseTestePressao::Exporta()
{
    //armazena a string digitada em nomeSaida

    cout << "\nInforme o nome do arquivo de saida com os parametros
        calculados pelo simulador:" << endl;
    cin >> nome;
    cin.get();
    //    getline (cin, nome);
    string formato = ".dat";
    string Saida = nome+formato;
        ///abre arquivo
    fout.open (Saida.c_str());

    fout << "Permeabilidade:" << permeabilidade;
    if ((reservatorio.SistemaUnidade()==141.2) || (reservatorio.
        SistemaUnidade()==19.03))
    fout << " milidarcy" << endl;
    if (reservatorio.SistemaUnidade()==0.3183)
        fout << " metros quadrados" << endl; ///criacao do objeto
            armazena da classe CSimuladorAnaliseTestePressao
    fout << "Pressao Inicial:" << pressaoInicial;

    if (reservatorio.SistemaUnidade()==141.2)
        fout << " psi" << endl;

    if (reservatorio.SistemaUnidade()==19.03)
        fout << " kgf/cm2" << endl;

    if (reservatorio.SistemaUnidade()==0.3183)
        fout << " Pascal" << endl;
        fout << " Fator de Pelicula:" << fatorPelicula << endl;
        fout << " Raio Efetivo:" << raioEfetivo;

    if ((reservatorio.SistemaUnidade()==0.3183) || (reservatorio.
        SistemaUnidade()==19.03))
        fout << " metros" << endl;

    if (reservatorio.SistemaUnidade()==141.2)
        fout << " ft" << endl;

```

```

        fout << "Queda de Pressao devido ao dano: " << pressaoDano <<
            endl;

        //Exibir em porcentagens
        fout << "Indice de Produtividade: " << indiceProdutividade*100.0 << "
            % " << endl << "Eficiencia de Fluxo: " << eficienciaFluxo*100.0 << "
            % " << endl;
        fout.close();
        cout << "Dados Salvos com sucesso!" << endl;
        cout << "-----" << endl;
    }
    //Calcula os parametros da estocagem, se houver
    void CSimuladorAnaliseTestePressao::CalculoEstocagem()
    {
        coeficienteEstocagem = (poco.Vazao() * fluido.FatorVolumeFormacao()
            * registrador.TempoSemProducao(0))
            / (24.0 * (registrador.PressaoMedida(0) -
                poco.PressaoPoco()));

        tempoEstocagem = ((60.0 + 3.5 * fatorPelícula)/(permeabilidade *
            reservatorio.Altura()))
            * reservatorio.SistemaUnidade() * 24.0 *
            coeficienteEstocagem * fluido.Viscosidade();

        cout << "Coeficiente de Estocagem: " << coeficienteEstocagem <<
            endl
            << "Tempo de Estocagem: " << tempoEstocagem;

        if ((reservatorio.SistemaUnidade()==141.2) || (reservatorio.
            SistemaUnidade()==19.03))
            cout << " horas " << endl;

        if (reservatorio.SistemaUnidade()==0.3183)
            cout << " segundos " << endl;
    }
}

```

Apresenta-se na listagem 5.19 o programa que usa as classes listadas acima.

Listing 5.19: Arquivo de implementação da função main().

```

#include "CSimuladorAnaliseTestePressao.h"

int main()
{
    // cria o objeto simulador da classe CSimuladorAnaliseTestePressao
    CSimuladorAnaliseTestePressao simulador;

    //executa a funcao de analise do teste de pressao
    simulador.Executar();
}

```

```
    //retorna 0 se o programa rodou normalmente  
    return 0;  
}
```

Capítulo 6

Teste

Neste capítulo se apresenta os testes realizados para assegurar que o programa esteja funcionando corretamente.

6.1 Teste 1: Teste no Windows

O teste realizado no sistema operacional Windows 8 (plataforma onde foi desenvolvida a maior parte do código do programa), com auxílio do compilador 'Dev C++', teve como objetivo: Verificar se havia algum tipo de '*bug*' no programa, se ele retornava os valores corretos dos parâmetros do reservatório, e se os métodos condicionais do programa funcionariam, como uma entrada de dados errada por parte do usuário e uma nova regressão linear caso o fator de correlação não fosse satisfatório.

Primeiramente, como mostra a Figura 6.1, o programa pede a seleção do sistema de unidades e a entrada dos parâmetros.

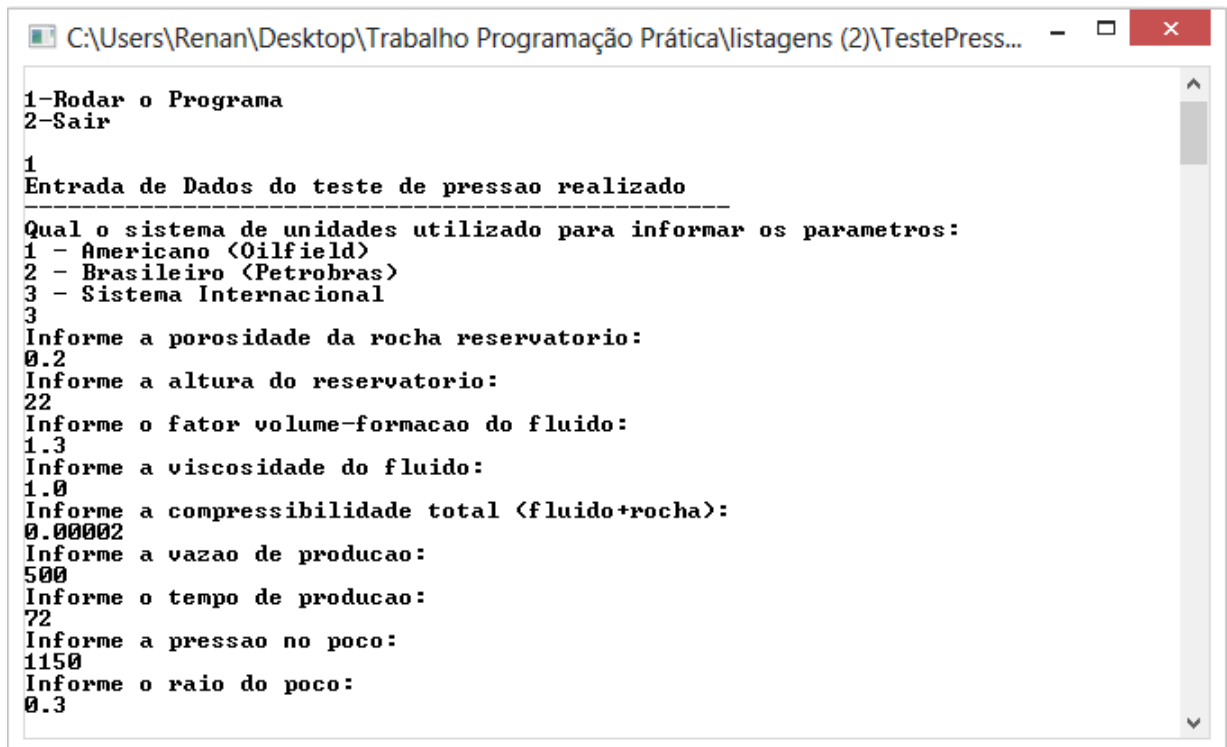


Figura 6.1: Tela do programa mostrando a entrada de dados.

Em seguida, como mostrado na Figura 6.2, o programa solicita os dados do registrador de pressão, e realiza a regressão linear, exibindo o resultado na tela.

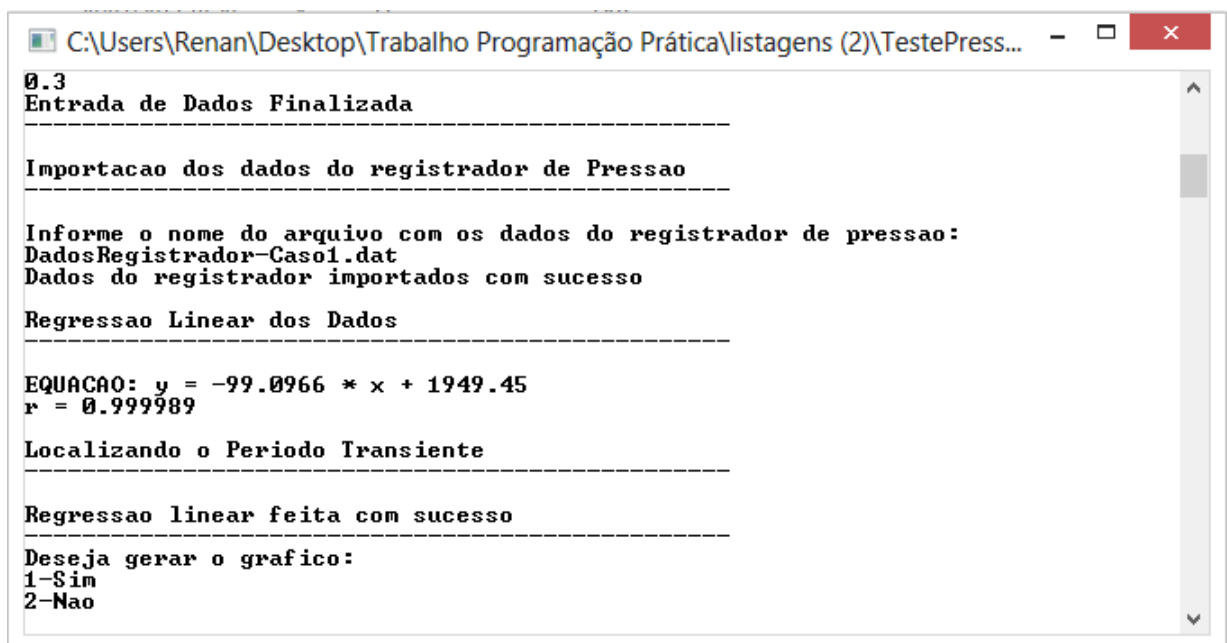


Figura 6.2: Tela do programa mostrando seleção do arquivo de entrada, e posterior regressão linear e o ajuste da curva.

Após o a realização da regressão linear, o usuário pode solicitar a geração do gráfico, como mostrado nas Figuras 6.3 e 6.4. Os resultados são exibidos em tela, assim como podem ser exportados para um arquivo .dat, com o nome escolhido pelo usuário.

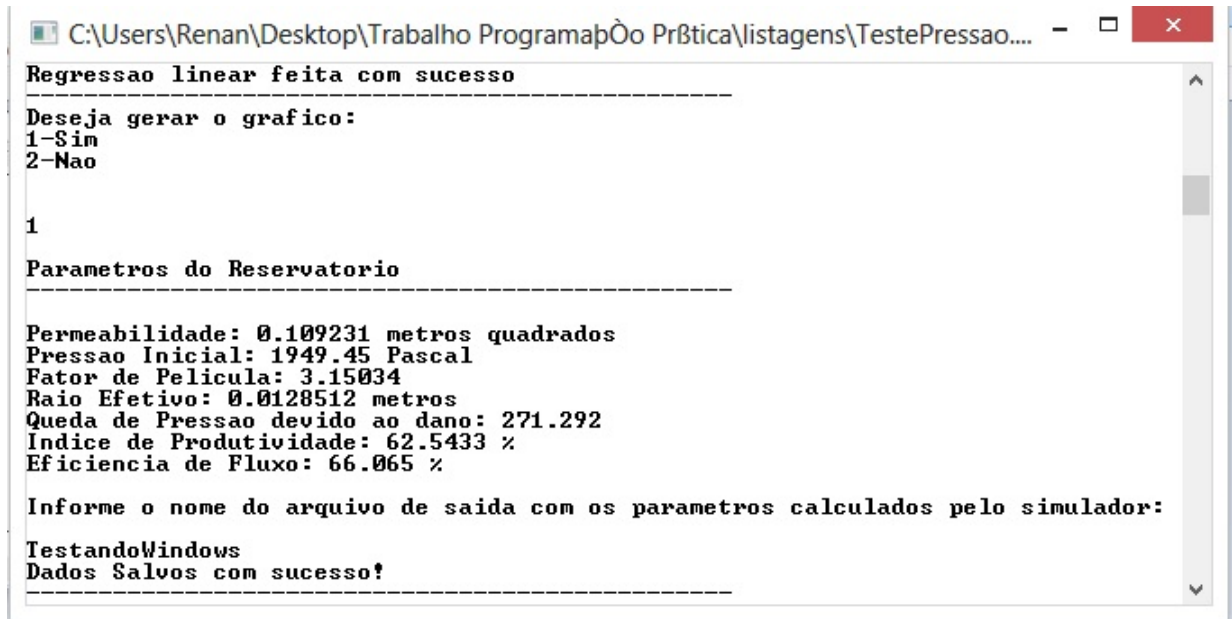


Figura 6.3: Tela do programa mostrando a possibilidade de geração do gráfico, a exportação dos resultados para um arquivo de saída .dat, e a caracterização do reservatório.

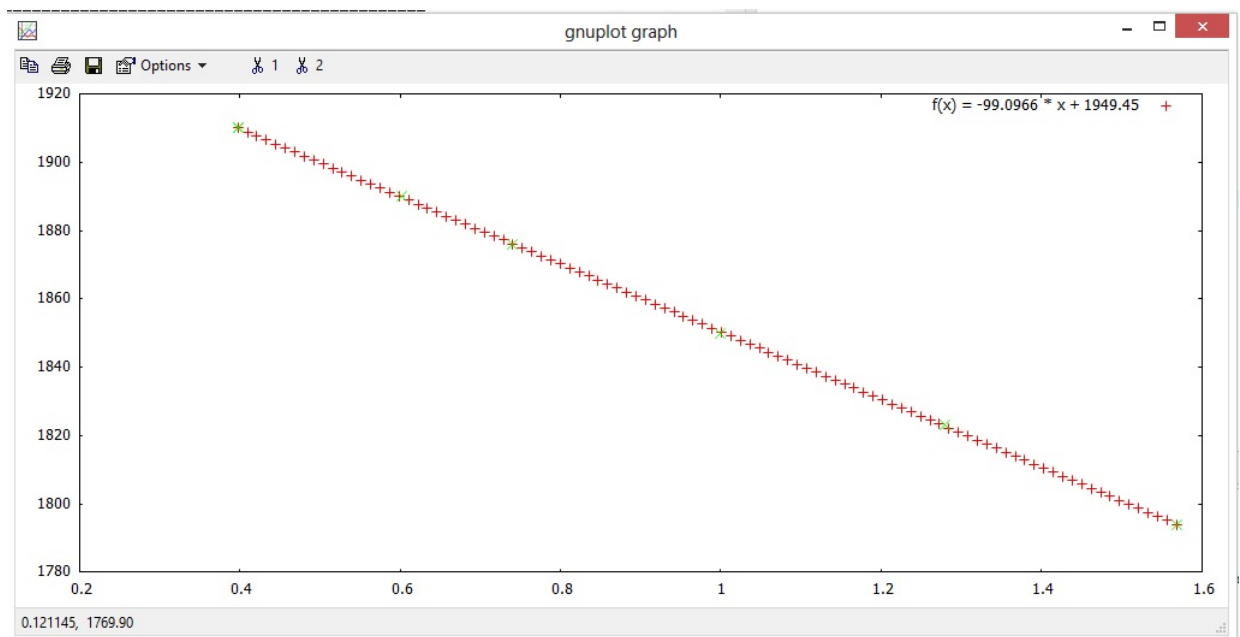


Figura 6.4: Gráfico de Pws vs. $\log[(Tp + \Delta t)/\Delta t]$ gerado pelo Gnuplot.

A Figura 6.5 mostra a última etapa, onde o usuário pode realizar a variação de uma determinada variável. Para isso, deve entrar com os valores inicial e final, além da quantidade de intervalos em que deseja realizar tal variação. Assim, o programa gera o resultado para todos os valores calculados e exporta para um arquivo externo .dat automaticamente, como exibido na Figura 6.6

```

C:\Users\Renan\Desktop\Trabalho Programação Prática\listagens\TestePressao...
Deseja variar algum parametro?
1- Sim ; 2 - Nao
1
Qual parametro deseja variar?
! 1 - Porosidade ; 2 - Fator Volume de Formacao ; 3 - Compressibilidade ; 4 - Viscosidade ;
1
Digite um valor inicial para porosidade
.2
Digite um valor Final para porosidade
.4
Em quantos intervalos deseja dividir a porosidade?
2

Calculo para porosidade : 0.2
-----
Permeabilidade: 0.109231 metros quadrados
Pressao Inicial: 1949.45 Pascal
Fator de Pelicula: 3.15034
Raio Efetivo: 0.0128512 metros
Queda de Pressao devido ao dano: 271.292
Indice de Produtividade: 62.5433 %
Eficiencia de Fluxo: 66.065 %
-----
Dados Salvos com sucesso!
-----

Calculo para porosidade : 0.3
-----
Permeabilidade: 0.109231 metros quadrados
Pressao Inicial: 1949.45 Pascal
Fator de Pelicula: 3.35302
Raio Efetivo: 0.0104935 metros
Queda de Pressao devido ao dano: 288.745
Indice de Produtividade: 62.5433 %
Eficiencia de Fluxo: 63.8818 %
-----
Dados Salvos com sucesso!
-----

Calculo para porosidade : 0.4
-----
Permeabilidade: 0.109231 metros quadrados
Pressao Inicial: 1949.45 Pascal
Fator de Pelicula: 3.49683
Raio Efetivo: 0.00908799 metros
Queda de Pressao devido ao dano: 301.129
Indice de Produtividade: 62.5433 %
Eficiencia de Fluxo: 62.3327 %
-----
Dados Salvos com sucesso!
-----

```

Figura 6.5: Tela do programa mostrando a possibilidade de se variar um parâmetro, e comparar os resultados, além de exportá-los para um arquivo externo.

Todos os arquivos exportados são salvos no diretório onde se encontram as listagens. Como é destacado na Figura 6.6, os arquivos tem o nome dado pelo usuário, e na realização da variação, o nome é automaticamente composto pelo nome escolhido pelo usuário, mais o nome do parâmetro variado, e o valor de tal parâmetro.

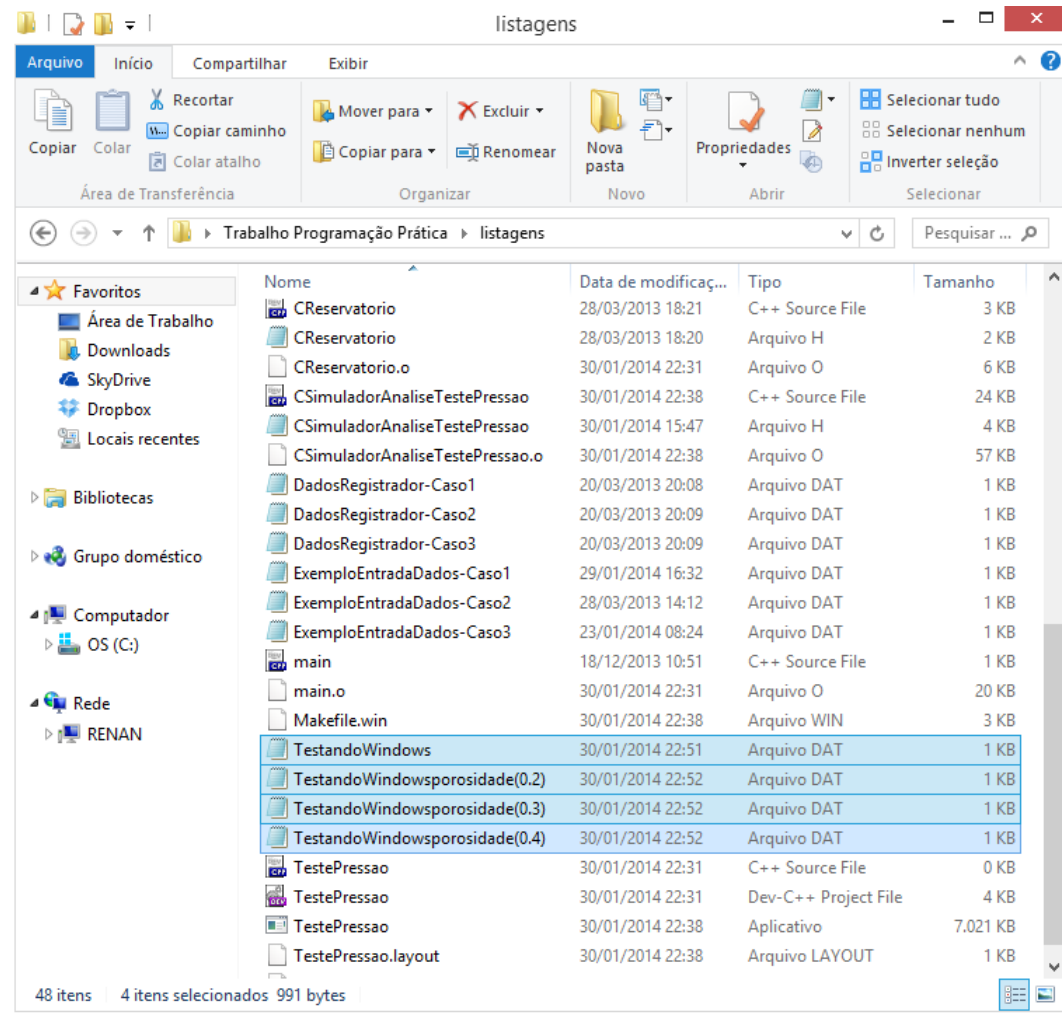
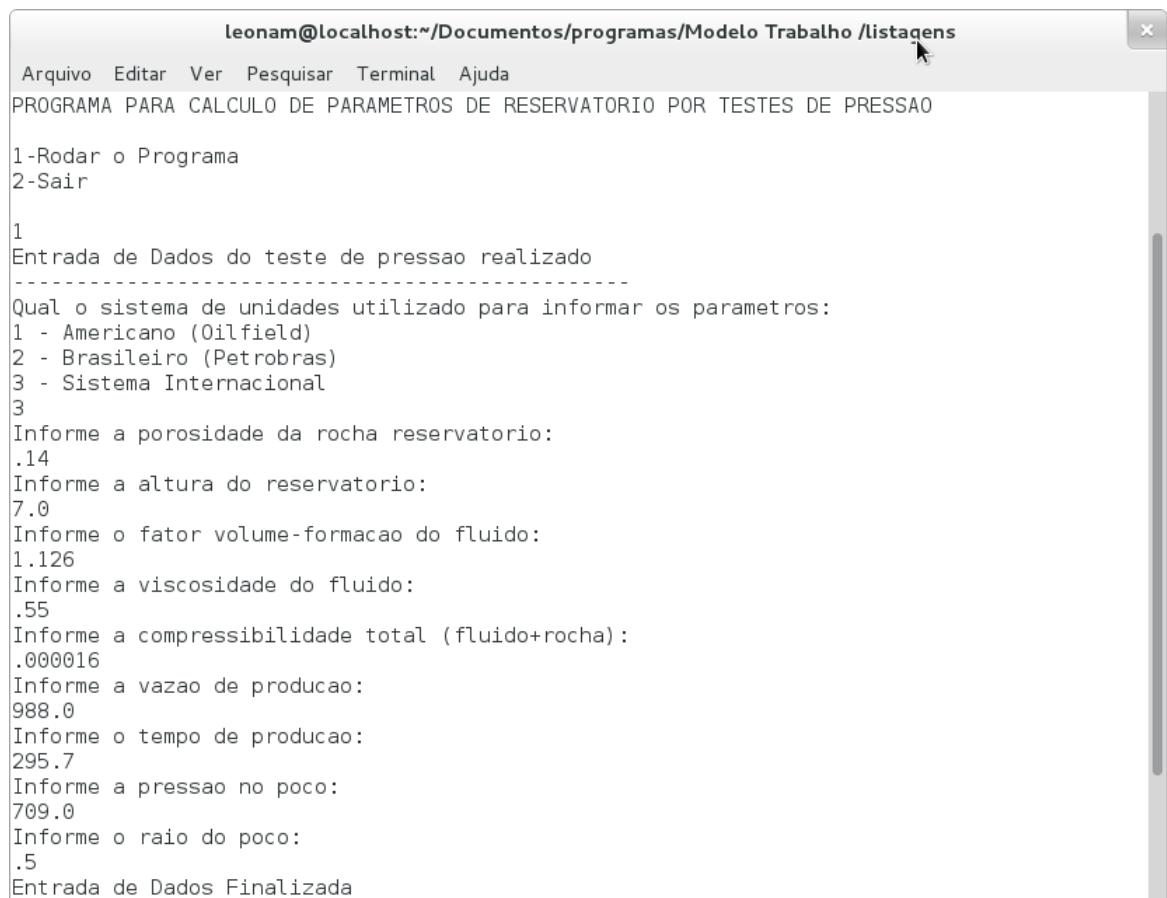


Figura 6.6: Diretório onde se encontram as listagens e onde são salvos os arquivos de saída (em destaque).

6.2 Teste 2: Teste no GNU/Linux (Gerando o gráfico com o Gnuplot)

O teste realizado no sistema operacional Linux Fedora, com auxílio do editor 'Kate' e dos comandos do terminal "g++ *.cpp" e "./a.out", teve como objetivo: verificar se havia algum tipo de 'bug' no programa em outro sistema, mas principalmente se o gráfico iria ser gerado corretamente pelo programa "Gnuplot". A entrada de dados do fluido, do poço e da rocha é mostrada na Figura 6.7. Os próximos passos, mostrados nas Figuras 6.8, 6.9 e 6.10 foram idênticos aos feitos no teste no Windows. Na Figura 6.11 há uma entrada equivocada (compressibilidade menor que 0), onde o programa pede que seja reinformada a compressibilidade. O mesmo processo ocorre em caso de erro na entrada dos demais parâmetros.

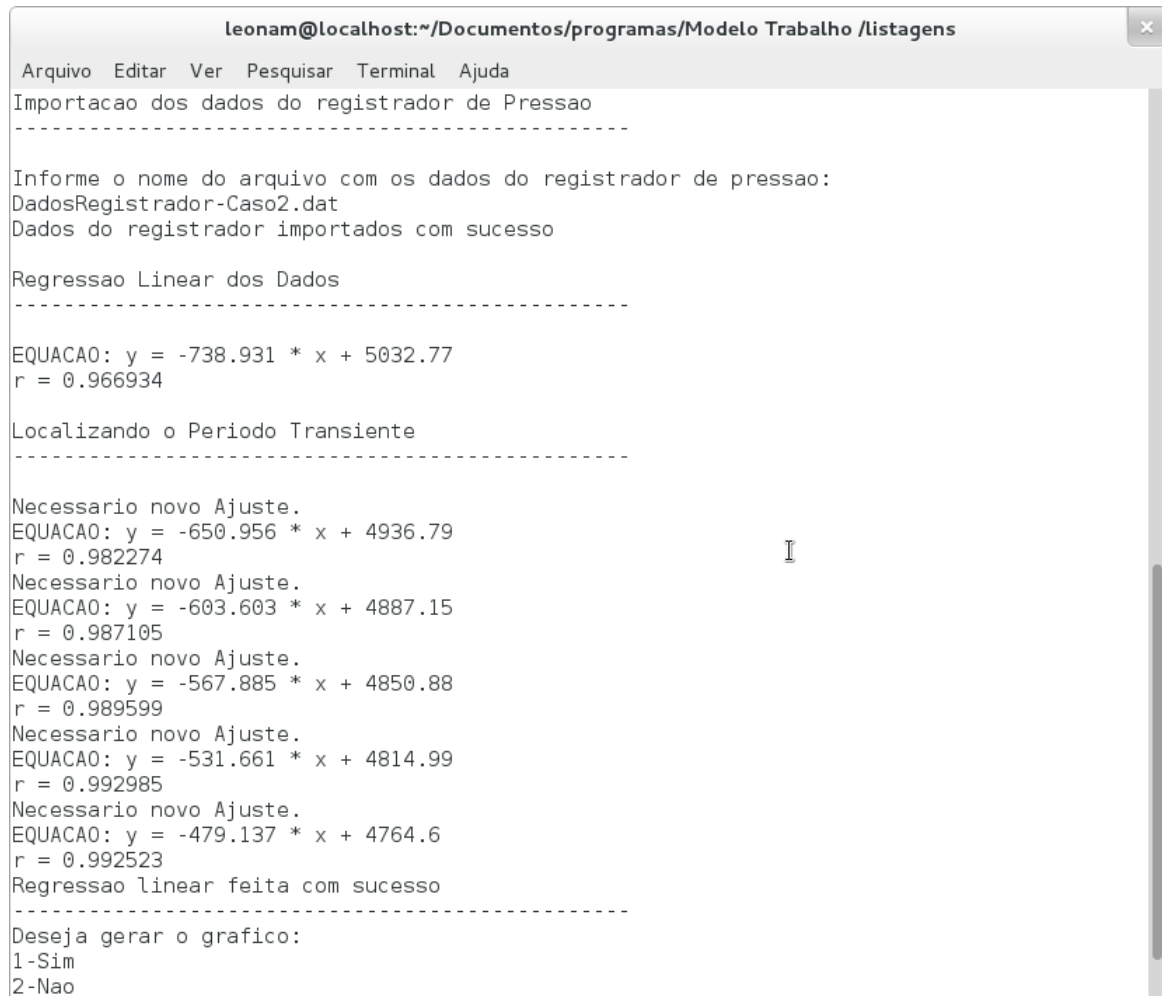


```
leonam@localhost:~/Documentos/programas/Modelo Trabalho /listagens
Arquivo Editar Ver Pesquisar Terminal Ajuda
PROGRAMA PARA CALCULO DE PARAMETROS DE RESERVATORIO POR TESTES DE PRESSAO

1-Rodar o Programa
2-Sair

1
Entrada de Dados do teste de pressao realizado
-----
Qual o sistema de unidades utilizado para informar os parametros:
1 - Americano (Oilfield)
2 - Brasileiro (Petrobras)
3 - Sistema Internacional
3
Informe a porosidade da rocha reservatorio:
.14
Informe a altura do reservatorio:
7.0
Informe o fator volume-formacao do fluido:
1.126
Informe a viscosidade do fluido:
.55
Informe a compressibilidade total (fluido+rocha):
.000016
Informe a vazao de producao:
988.0
Informe o tempo de producao:
295.7
Informe a pressao no poço:
709.0
Informe o raio do poço:
.5
Entrada de Dados Finalizada
```

Figura 6.7: Tela do programa mostrando a entrada de dados inicial.



```
leonam@localhost:~/Documentos/programas/Modelo Trabalho /listagens
Arquivo Editar Ver Pesquisar Terminal Ajuda
Importacao dos dados do registrador de Pressao
-----
Informe o nome do arquivo com os dados do registrador de pressao:
DadosRegistrador-Caso2.dat
Dados do registrador importados com sucesso

Regressao Linear dos Dados
-----
EQUACA0: y = -738.931 * x + 5032.77
r = 0.966934

Localizando o Período Transiente
-----
Necessario novo Ajuste.
EQUACA0: y = -650.956 * x + 4936.79
r = 0.982274
Necessario novo Ajuste.
EQUACA0: y = -603.603 * x + 4887.15
r = 0.987105
Necessario novo Ajuste.
EQUACA0: y = -567.885 * x + 4850.88
r = 0.989599
Necessario novo Ajuste.
EQUACA0: y = -531.661 * x + 4814.99
r = 0.992985
Necessario novo Ajuste.
EQUACA0: y = -479.137 * x + 4764.6
r = 0.992523
Regressao linear feita com sucesso
-----
Deseja gerar o grafico:
1-Sim
2-Nao
```

Figura 6.8: Tela do programa mostrando seleção do arquivo de entrada, e posterior regressão linear e o ajuste da curva.

```

leonam@localhost:~/Documentos/programas/Modelo Trabalho /listagens
Arquivo Editar Ver Pesquisar Terminal Ajuda
-----
Deseja gerar o grafico:
1-Sim
2-Nao

1

Parametros do Reservatorio
-----

Permeabilidade: 0.0668363 metros quadrados
Pressao Inicial: 4764.6 Pascal
Fator de Pelicula: 3.76971
Raio Efetivo: 0.0115294 metros
Queda de Pressao devido ao dano: 1569.59
Indice de Produtividade: 24.3614 %
Eficiencia de Fluxo: 61.2982 %

Informe o nome do arquivo de saida com os parametros calculados pelo simulador:
TestandoLinux
Dados Salvos com sucesso!
-----

```

Figura 6.9: Tela do programa mostrando a possibilidade de geração do gráfico, a exportação dos resultados para um arquivo de saída .dat, e a caracterização do reservatório.

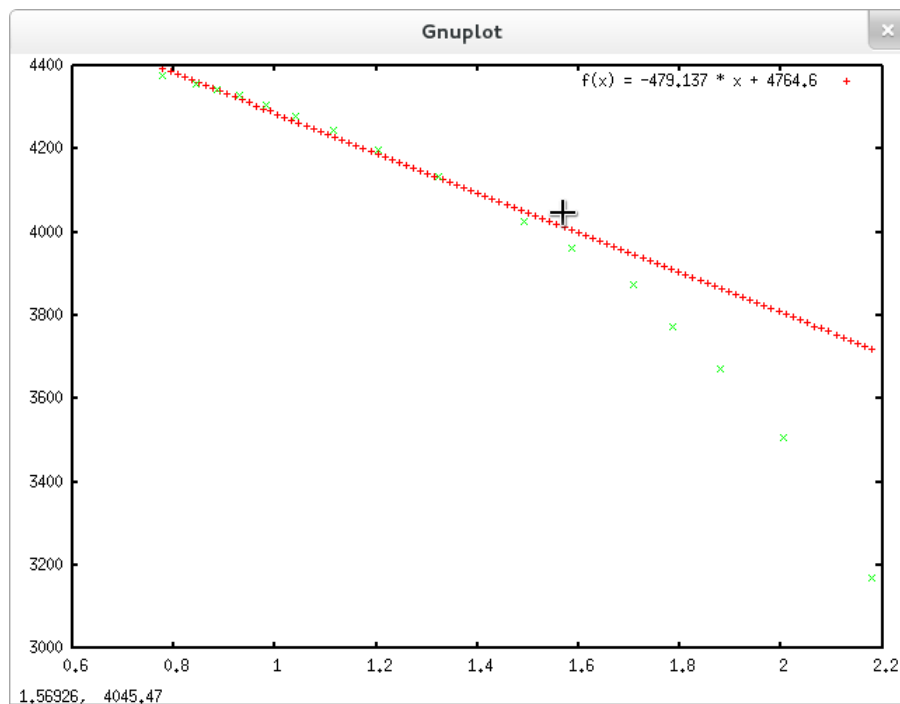


Figura 6.10: Gráfico de Pws vs. $\log[(T_p + \Delta t)/\Delta t]$ gerado pelo Gnuplot.

```

leonam@localhost:~/Documentos/programas/Modelo Trabalho /listagens
Arquivo Editar Ver Pesquisar Terminal Ajuda
Deseja variar algum parametro?
1- Sim | 2 - Nao
1
Qual parametro deseja variar?
| 1 - Porosidade | 2 - Fator Volume de Formacao | 3 - Compressibilidade | 4 - Viscosidade
|
3
Digite um valor inicial para Compressibilidade
-2
Reinforme a compressibilidade:
.000010
Digite um valor Final para Compressibilidade
.000020
Em quantos intervalos deseja dividir o Compressibilidade?
2

Calculo para Compressibilidade : 1e-05
-----
Permeabilidade: 0.0668363 metros quadrados
Pressao Inicial: 4764.6 Pascal
Fator de Pelicula: 3.53476
Raio Efetivo: 0.0145828 metros
Queda de Pressao devido ao dano: 1471.77
Indice de Produtividade: 24.3614 %
Eficiencia de Fluxo: 63.7102 %
-----
Dados Salvos com sucesso!
-----

Calculo para Compressibilidade : 1.5e-05
-----
Permeabilidade: 0.0668363 metros quadrados
Pressao Inicial: 4764.6 Pascal
Fator de Pelicula: 3.73745
Raio Efetivo: 0.0119074 metros
Queda de Pressao devido ao dano: 1556.16
Indice de Produtividade: 24.3614 %
Eficiencia de Fluxo: 61.6294 %
-----
Dados Salvos com sucesso!
-----

```

Figura 6.11: Tela do programa mostrando a possibilidade de se variar um parâmetro, e comparar os resultados, além de exportá-los para um arquivo externo. Também é destacado a ação do programa pedindo ao usuário para reinformar o valor da compressibilidade.

Capítulo 7

Documentação

A presente documentação refere-se ao uso do "Programa em C++ para avaliação de formações por dados de teste de pressão". Esta documentação tem o formato de uma apostila que explica passo a passo ao usuário como usar o programa.

7.1 Manual do Usuário

O programa calcula a permeabilidade, o fator de película, a pressão inicial, a eficiência de fluxo, o índice de produtividade, o coeficiente e o tempo de estocagem do reservatório que foi submetido ao teste de pressão (de curta ou longa duração). Para o funcionamento correto do programa, deve-se seguir os seguintes passos:

- Opcional: Para gerar o gráfico característico do reservatório: Ter instalado o software livre "Gnuplot" no computador.
- Os dados registrados no registrador de pressão do poço devem ser colocados na pasta do programa, sob qualquer nome, em formato de texto.
- Para abrir o programa no sistema operacional "Windows", simplesmente clique duas vezes em "TestePressao.exe". Caso o sistema operacional utilizado seja o "Linux", abra o terminal, selecione o caminho da pasta do programa. Digite então: "g++ *.cpp" para compilar os arquivos do programa e em seguida "./a.out".
 - Então o programa tem a interface própria para comunicação com o usuário.
 - Primeiramente, escolha se deseja rodar o programa ou sair: Digite 1 para rodar ou 2 para sair e tecle enter.
 - Escolha o sistema de unidades que o registrador de pressão trabalha: Digite 1 para o sistema americano (oilfield), 2 para o brasileiro (petrobras) ou 3 para o sistema internacional e tecle enter.
 - Informe a porosidade do reservatório e tecle enter.
 - Informe a altura do reservatório e tecle enter.
 - Informe o fator volume formação do fluido e tecle enter.
 - Informe a compressibilidade total (rocha+fluido) e tecle enter.

- Informe a vazão de produção e tecle enter.
- Informe o tempo de produção e tecle enter.
- Informe a pressão no poço e tecle enter.
- Informe o raio do poço e tecle enter.

Nesse momento, o programa fará a regressão linear dos dados importados e mostrará na tela a equação da reta no formato “ $y = a * x + b$ ”. Ele identificará o período de estocagem do reservatório para não haver erro de cálculo, pois a análise do teste de pressão deve ser feita do período transiente.

- Escolha se quer que o programa gere o gráfico com o Gnuplot: Digite 1 para sim ou 2 para não e tecle enter.

Nesse momento o programa:

- Mostrará na tela os parâmetros do reservatório, com as devidas unidades.
- Requisitará ao usuário o nome do arquivo de saída .dat com os parâmetros calculados.
- Em seguida, mostrará na tela os detalhes do período de estocagem do poço, se houver.
- Exibirá em tela a interpretação desses resultados no âmbito da exploração de petróleo.

Tecle enter e o programa perguntará se o usuário deseja variar algum parâmetro: Digite 1 para sim 2 para voltar ao início do programa.

Caso o usuário digite 1, deverá seguir as instruções:

- Selecionar o parâmetro: | 1-Porosidade | 2-Fator Volume de Formação | 3-Compressibilidade | 4-Viscosidade |.
- Selecionar o intervalo de variação, e o valor de cada variação.

Nesse momento serão gerados resultados referentes a cada valor adotado pelo parâmetro selecionado, mostrados em tela para comparação, e exportados para um arquivo .dat.

- Tecle enter e o programa fará a mesma pergunta do início: Digite 1 para rodar o programa novamente ou 2 para sair, tecle enter.

Observação: Se houver uma entrada negativa de valor (equivocada), o programa pedirá para o usuário entrar com o dado novamente.

Referências Bibliográficas

- [Adalberto Rosa, 2006] Adalberto Rosa, Renato Carvalho, D. X. (2006). *Engenharia de Reservatórios de petróleo*. Interciência, Rio de Janeiro.
- [Bueno, 2003] Bueno, A. D. (2003). *Programação Orientada a Objeto com C++ - Aprenda a Programar em Ambiente Multiplataforma com Software Livre*. Novatec, São Paulo.
- [Grossens et al., 1993] Grossens, M., Mittelbach, F., and Samarin, A. (1993). *Latex Companion*. Addison-Wesley, New York.
- [Karger, 2004] Karger, A. (2004). *O Tutorial de Lyx*. LyX Team - <http://www.lyx.org>.
- [Lee, 1982] Lee, J. (1982). *Well Testing*. SPE, New York.
- [Álvaro M. M. Peres, 2008] Álvaro M. M. Peres (2008). Teoria dos testes de pressão em poços. Rio de Janeiro. Anais do VI Encontro ENGEP.
- [LyX-Team, 2004] LyX-Team, editor (2004). *The LyX User's Guide*. LyX Team - <http://www.lyx.org>.
- [Steding-Jessen, 2000] Steding-Jessen, K. (2000). *Latex demo: Exemplo com Latex 2e*.