

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

Simulador de Traço Sísmico - VERSÃO 1.0  
TRABALHO DA DISCIPLINA PROGRAMAÇÃO PRÁTICA

RAMON CRESPO DIOGO  
SÁVIO DA SILVEIRA JANUÁRIO

MACAÉ - RJ  
MARÇO - 2015

# Sumário

<b>1</b>	<b>Introdução</b>	<b>1</b>
1.1	Escopo do problema . . . . .	2
1.2	Objetivos . . . . .	2
<b>2</b>	<b>Especificação</b>	<b>3</b>
2.1	Especificação do software - requisitos . . . . .	3
2.1.1	Nome do produto e componentes . . . . .	3
2.1.2	Especificação . . . . .	3
2.1.3	Requisitos funcionais . . . . .	4
2.1.4	Requisitos não funcionais . . . . .	4
2.2	Casos de uso do software . . . . .	4
2.3	Diagrama de caso de uso geral do software . . . . .	5
2.4	Diagrama de caso de uso específico do software . . . . .	5
<b>3</b>	<b>Elaboração</b>	<b>6</b>
3.1	Análise de domínio . . . . .	6
3.2	Formulação teórica . . . . .	6
3.2.1	Funções <i>Wavelet</i> . . . . .	7
3.2.2	Convolução: . . . . .	9
3.2.3	Impedância Acústica . . . . .	10
3.2.4	Coeficiente de Reflexão . . . . .	11
3.3	Identificação de pacotes . . . . .	11
<b>4</b>	<b>AOO – Análise Orientada a Objeto</b>	<b>13</b>
4.1	Diagramas de classes . . . . .	13
4.1.1	Dicionário de classes . . . . .	15
4.2	Diagrama de seqüência – eventos e mensagens . . . . .	15
4.2.1	Diagrama de seqüência geral . . . . .	15
4.3	Diagrama de comunicação – colaboração . . . . .	16
4.4	Diagrama de máquina de estado . . . . .	17
4.5	Diagrama de atividades . . . . .	17

---

<b>5</b>	<b>Projeto</b>	<b>20</b>
5.1	Projeto do sistema . . . . .	20
5.2	Projeto orientado a objeto – POO . . . . .	22
5.3	Diagrama de componentes . . . . .	24
5.4	Diagrama de implantação . . . . .	24
<b>6</b>	<b>Implementação</b>	<b>27</b>
6.1	Código fonte . . . . .	27
<b>7</b>	<b>Teste</b>	<b>41</b>
7.1	Testes . . . . .	41
7.1.1	Tipo de entrada . . . . .	42
7.1.2	Teste usando Chapéu Mexicano . . . . .	42
7.1.3	Teste usando Morlet . . . . .	42
<b>8</b>	<b>Documentação</b>	<b>52</b>
8.1	Documentação do usuário . . . . .	52
8.1.1	Como rodar o software . . . . .	52
8.2	Documentação para desenvolvedor . . . . .	53
8.2.1	Dependências . . . . .	53
8.2.2	Documentação usando doxygen . . . . .	53

# Capítulo 1

## Introdução

Neste capítulo iremos abordar uma estratégia que tem sido utilizada na fase de exploração de petróleo, o traço sísmico sintético.

A estimativa da resposta sísmica de um intervalo litológico atravessado por poços é uma das pedras fundamentais de um bom trabalho de interpretação sísmica. Com tal propósito, antes de se iniciar a interpretação propriamente dita, utiliza-se construir o sismograma sintético, que corresponde ao resultado da modelagem da resposta sísmica em determinada área, normalmente para incidência vertical (embora o sismograma possa e deva ser gerado, considerando-se as variações decorrentes do afastamento fonte-receptor que também estão presentes no dado sísmico real) através do uso de informações de velocidade e densidade medidas em perfis de poços. Sua construção exige ainda a estimativa de um pulso sísmico que pode ser matemático (por exemplo um pulso de ricker com 20 Hz), ou estimado a partir dos dados sísmicos disponíveis. Tal prática serve a vários propósitos, dos quais podemos citar os mais importantes:

Durante a fase de processamento sísmico, por exemplo, sua construção pode ajudar na avaliação da eficiência de etapas como a deconvolução ou o tratamento de amplitudes;

Na interpretação será utilizado para correlação da litologia atravessada pelo poço (em profundidade) com sua expressão no dado sísmico real, e conseqüente identificação do comportamento sísmico de interfaces litológicas (se uma determinada interface corresponde a um coeficiente de reflexão positivo ou negativo e definição da sua intensidade);

Investigação dos limites de resolução do dado sísmico, isto é, se o dado sísmico disponível possibilitará ao intérprete individualizar a reflexão do topo e da base de determinado alvo litológico;

Investigar a convenção de polaridade de um dado sísmico cuja convenção seja desconhecida;

Se na área estão disponíveis vários poços, com respectivos perfis de velocidade e densidade, o interprete poderá construir vários sismogramas, para avaliar a variabilidade da sismofície associada à determinada interface. Isso o ajudará durante a interpretação, a admitir tal variabilidade durante o processo de rastreamento das interfaces;

Em processos como o da inversão do dado sísmico, para estimativa da função impedância a partir dos dados de amplitude, a construção do sismograma será necessária para estimativa da wavelet e calibração do processo de inversão;

Se a correlação do sismograma sintético com o dado sísmico for satisfatória, o perfil sônico (de velocidades) usado na sua construção pode ser usado (através da integração dos tempos de trânsito em cada intervalo) para construção de uma relação tempo-profundidade que permitirá ao intérprete traduzir toda a litologia atravessada pelo poço, na sua posição em tempo na seção sísmica;

A classificação de sismofácies características de determinados intervalos litoestratigráficos também se vale da construção de sismogramas sintéticos e correlação das eletrofácies com a resposta sísmica do intervalo.

No presente trabalho, é proposto a criação de um software orientado a objeto em C++ que gere o traço sísmico sintético a partir de dados da formação que possibilita correlação com a sísmica real.

## 1.1 Escopo do problema

O software irá determinar a impedância, resistividade e assim gerar o traço sísmico sintético utilizando o método da convolução com auxílio das funções Wavelet de Morlet ou Ricker. Há varias utilizações para o software uma delas é que com o traço sísmico sintético, durante a fase de processamento sísmico, por exemplo, sua construção pode ajudar na avaliação da eficiência de etapas como a deconvolução ou o tratamento de amplitudes;

## 1.2 Objetivos

Os objetivos deste trabalho são:

- Objetivo geral:
  - Desenvolver um programa para a gerar um traço sísmico sintético de uma formação arbitrária a partir de dados da formação com espessura da camada, velocidade de propagação da onda acústica e a densidade.
- Objetivos específicos:
  - Permitir ao usuario entrar com dados característicos da formação.
  - Disponibilizar métodos de entrada de dados a partir de arquivos do tipo texto.
  - Desenvolver um software que receba dados reais ou arbitrários que calcule a impedância e reflectividade de cada camada, a partir de onde será gerado um traço sísmico por convolução com uma wavelet indicada pelo o usuário.

# Capítulo 2

## Especificação

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

### 2.1 Especificação do software - requisitos

#### 2.1.1 Nome do produto e componentes

<b>Nome</b>	Simulador para a geração de um traço sísmico sintético
<b>Componentes principais</b>	Gerar um modelo sintético para caracterização do reservatório
<b>Missão</b>	Auxiliar geofísicos e engenheiros a caracterizar um reservatório e possibilitar comparação com dados geofísicos e de sísmica de poços reais.

#### 2.1.2 Especificação

O objetivo deste projeto é construir um software para fins acadêmicos que será livre, licença GNU/GPL v2.0, multiplataforma e orientado a objeto em C++, de modo a facilitar futuras modificações. Este será em modo texto, pois há pouca interação do usuário com o software. Inicialmente, o usuário definirá os parâmetros de entrada (velocidade acústica, densidade e profundidade de cada camada) utilizando um arquivo de texto (.txt ou .dat) ou informando ao programa manualmente. Usando essas informações o software irá calcular a impedância acústica, a refletividade, em seguida, irá convolver a refletividade podendo escolher a função wavelet, assim podendo gerar o traço sísmico sintético. Os resultados serão exibidos na forma de gráficos usando o gnuplot e impressos na tela

podendo, se desejado pelo usuário, salvar em disco na opção de um arquivo de texto (.txt e .dat).

### 2.1.3 Requisitos funcionais

<b>RF-01</b>	O usuário deverá ter liberdade para escolher o tipo de entrada de parâmetros.
<b>RF-02</b>	Deve permitir o carregamento de arquivos criados pelo software.
<b>RF-03</b>	Deve permitir a escolha da função <i>wavelet</i> de Morlet ou Chapéu Mexicano.
<b>RF-04</b>	O usuário terá disponível os seus resultados em gráfico e em exportados como texto.

### 2.1.4 Requisitos não funcionais

<b>RNF-01</b>	O programa deverá ser multi-plataforma, podendo ser executado em <i>Windows</i> , <i>GNU/Linux</i> ou <i>Mac</i> .
---------------	--

## 2.2 Casos de uso do software

A Tabela 2.1 apresenta um caso de uso que descreve um ou mais cenários de uso do software, exemplos de uso, como o sistema interage com usuários externos (atores) e algumas das etapas a serem seguidas, representando uma seqüência típica de uso do programa.

Nome do caso de uso:	Geração do traço sísmico.
Resumo/ descrição:	Geração do traço sísmico sintético de uma formação arbitrária.
Etapas:	1. Criar objeto simulador.
	2. Criar objeto de Propriedade do Sistema.
	3. Calcular Impedância.
	4. Calcular Refletividade.
	5. Convolver refletividade com <i>wavelet</i> .
	6. Gerar gráfico.
	7. Analisar resultado.
Cenários Alternativos:	Inserir dados fora da realidade do sistema simulado.

## 2.3 Diagrama de caso de uso geral do software

O diagrama de caso de uso geral da Figura 2.1 mostra o usuário acessando os sistemas de ajuda do software, calculando a área de uma função ou analisando resultados.

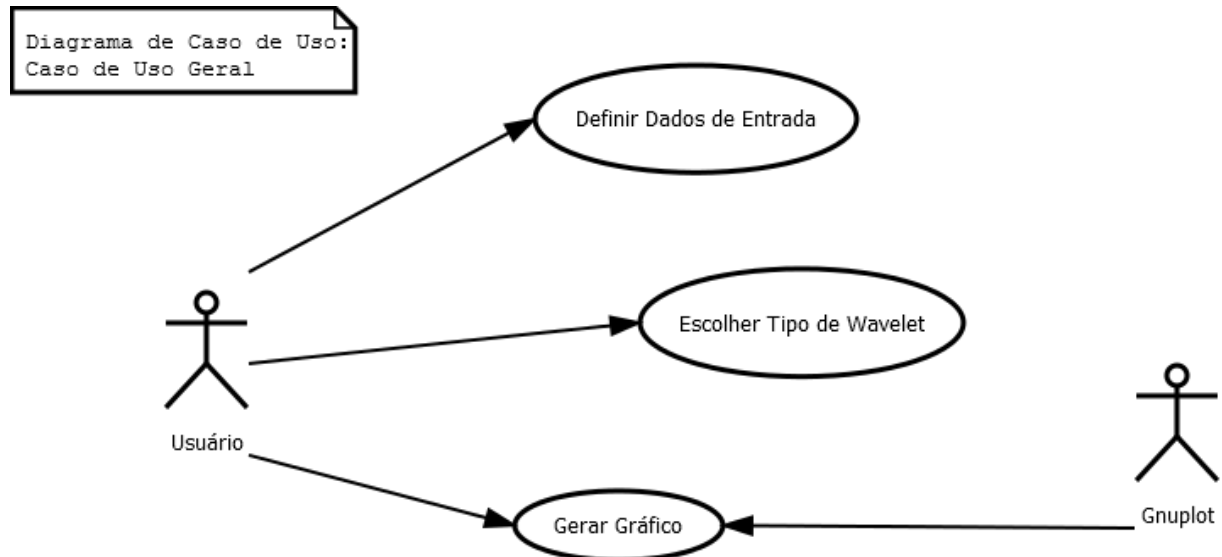


Figura 2.1: Diagrama de caso de uso – Caso de uso geral

## 2.4 Diagrama de caso de uso específico do software

A Figura 2.2 representa o caso de uso específico para o usuário gerando o arquivo de entrada, no qual ele define os parâmetros que serão utilizados pelo programa.

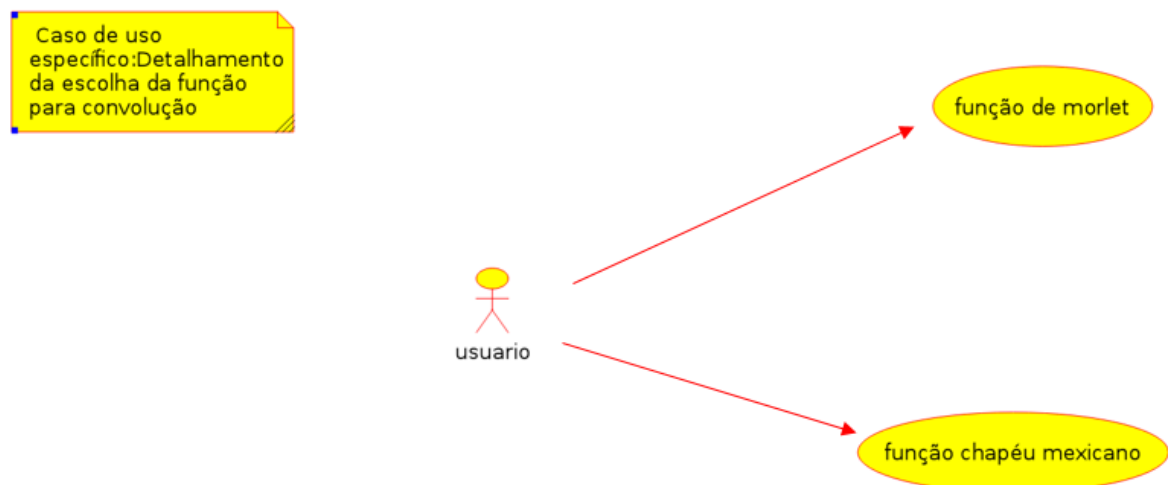


Figura 2.2: Diagrama de caso de uso específico: Detalhamento da escolha da função para convolução



# Capítulo 3

## Elaboração

Neste capítulo iremos apresentar o processo que levou a criação do software, como: Entrevistas, pesquisas na bibliografia e elaboração de diagramas que nos mostram os conhecimentos que seriam necessários para tal.

### 3.1 Análise de domínio

A identificação de pacotes possui grande importância, pois demonstra as relações entre as diferentes partes do sistema, como classes, componentes, subclasses, interfaces, nós, colaborações e casos de uso.

Após estudo dos requisitos/especificações do sistema, algumas entrevistas com profissionais acadêmicos como o professor Fernando Moraes, estudos na biblioteca do LENEP e disciplinas do curso foi possível identificar nosso domínio de trabalho:

- O pacote “Banco de Dados”, contendo as Propriedades físicas das camadas, isto é, velocidade acústica, densidade e espessura que são valores obtidos em medidas experimentais;
- O pacote “Cálculo do traço sísmico sintético”, que contém os executáveis específicos para a geração do traço sísmico, incluindo as respectivas constantes utilizadas nas equações.
- O pacote “Biblioteca Numérica”, em cujo conteúdo está o método numérico necessário para o uso da convoluções que serão utilizados no programa.
- O pacote “Gráficos” que inclui o acesso ao Gnuplot.

### 3.2 Formulação teórica

Apresentaremos nesta seção o embasamento teórico para a criação e execução deste projeto de engenharia.

### 3.2.1 Funções *Wavelet*

Graças a capacidade de decompor as funções tanto no domínio da frequência quanto no domínio do tempo, as funções *wavelet* são ferramentas poderosas de processamento de sinais, muito aplicadas na compressão de dados, eliminação de ruído, separação de componentes no sinal, identificação de singularidades, detecção de auto-similaridade, e muito mais.

Para ser considerada uma *wavelet*, uma função tem de atender as seguintes características:

A área total sob a curva da função é 0, ou seja

$$\int_{-\infty}^{+\infty} \psi(t) dt = 0 \quad (3.1)$$

A energia da função é finita, ou seja

$$\int_{-\infty}^{+\infty} |\psi(t)|^2 dt < L \quad L \in \mathbb{N} \quad (3.2)$$

Sendo:

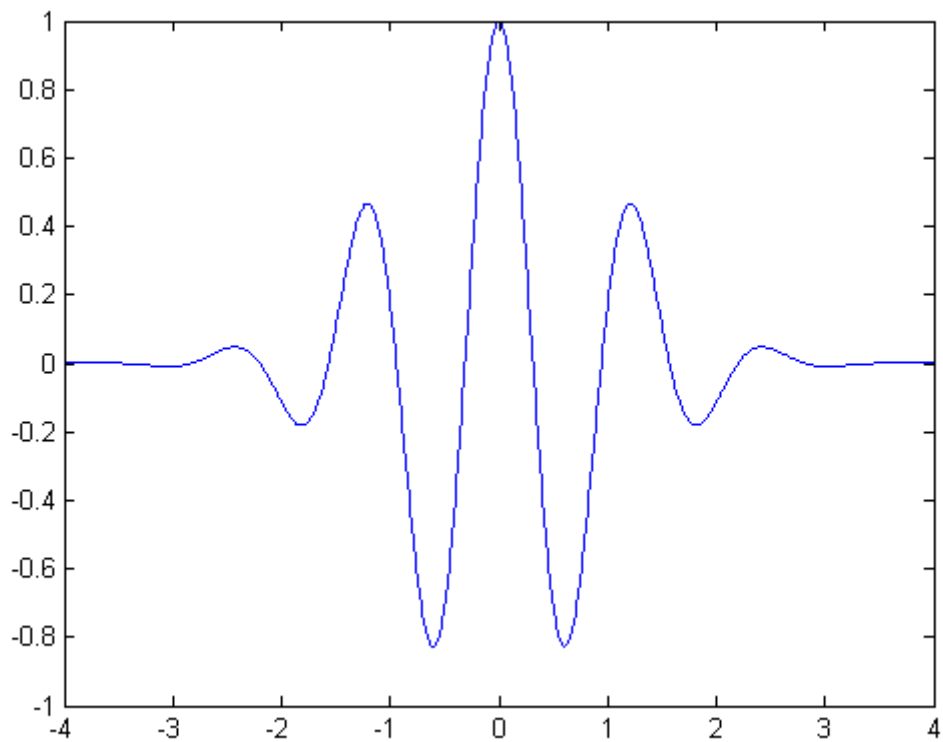
$\psi(t)$  = Uma função qualquer que atenda as características acima.

$t$  = Tempo.

Para ser utilizada na análise de sinais uma função *wavelet* precisa também de outra característica que chamamos de condição de admissibilidade, e que permite a existência da transformada inversa de *wavelet*.

Alguns exemplos de funções que atendem estas características são a função wavelet de Morlet

$$\psi(t) = \exp^{-t^2} \cos \left( \pi t \sqrt{\frac{2}{\ln 2}} \right) \approx \exp^{-t^2} \cos(2.885\pi t) \quad (3.3)$$

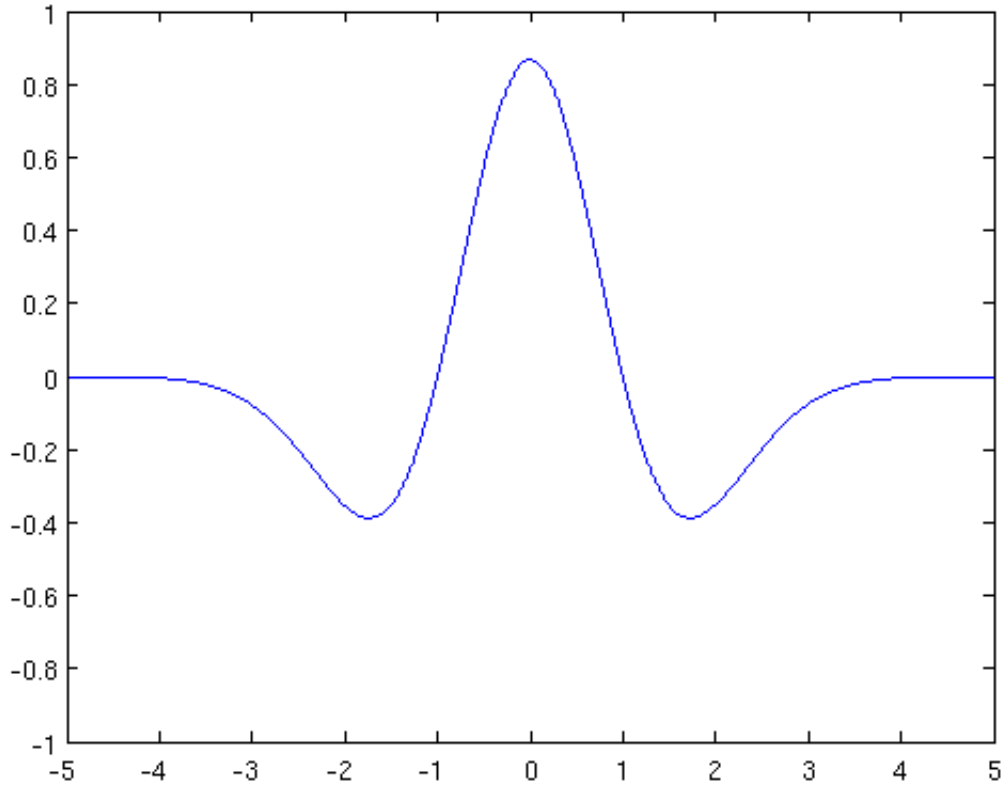
Figura 3.1: *Wavelet* de Morlet

e a curva conhecida como chapéu mexicano (do inglês *mexican hat*), definida por:

$$\psi(t) = (1 - 2t^2) \exp^{-t^2} \quad (3.4)$$

que é a segunda derivada da função Gaussiana

$$-0.5 \exp^{-t^2} \quad (3.5)$$

Figura 3.2: *Wavelet Chapéu Mexicano*

### 3.2.2 Convolução:

Em matemática, particularmente na área de análise funcional, *convolução* é um operador linear que, a partir de duas funções dadas, resulta numa terceira que mede a área subentendida pela superposição das mesmas em função do deslocamento existente entre elas. O conceito de convolução está ligado à integral de superposição na *Óptica de Fourier*, à integral de Duhamel na teoria das vibrações, ao Teorema de Borel no estudo de sistemas lineares invariantes no tempo, ao conceito de média móvel, às funções de correlação e de autocorrelação em estatística e em processamento de sinais, e a diversos conceitos usados em análise de imagens, como digitalização, alisamento, embaçamento e aberração cromática.

$$f(t) = \int_{-\infty}^{+\infty} f_1(x) f_2(t-x) dx \quad (3.6)$$

que geralmente é expressa simbolicamente como:

$$f(t) = f_1(t) * f_2(t) \quad (3.7)$$

Podemos estudar a convolução sob dois pontos de vista distintos:

- Do sinal de entrada: como cada ponto do sinal de entrada contribui para vários pontos do sinal de saída.
- Do sinal de saída: como cada ponto do sinal de saída recebeu contribuições de vários pontos do sinal de entrada.

Estas duas perspectivas são formas diferentes de analisar a mesma operação matemática, e portanto são equivalentes: a primeira fornece uma idéia conceitual da convolução, enquanto que a segunda descreve a matemática da convolução.

### 3.2.3 Impedância Acústica

A impedância acústica é definida como o produto entre a velocidade compressional e a densidade da rocha.

$$Z = \rho V \quad (3.8)$$

Sendo:

- $Z$  = Impedância acústica
- $\rho$  = Densidade da camada
- $V$  = Velocidade da onda acústica

Isso significa que a impedância acústica é uma propriedade de camada e não uma propriedade de interface como a amplitude do dado sísmico. Essa distinção faz com que a impedância acústica seja uma poderosa ferramenta a ser utilizada no processo de caracterização.

Sendo então um dado que se refere às camadas de rocha, a impedância acústica possui várias vantagens. De acordo com Latimer et al. (2000) um modelo de impedância de boa qualidade contém mais informação que o dado sísmico, pois, esse modelo possui todas as informações contidas no dado sísmico, além de possuir a informação adicional dos dados de perfis de poços. O volume de impedância acústica, dependendo do método aplicado para a sua obtenção, é o resultado da integração de dados provenientes de diferentes fontes, normalmente o dado sísmico, os dados de poços e/ou os modelos de velocidade. Assim, construir um modelo de impedância acústica é a maneira mais natural de se integrar as informações, gerando ao final do processo um modelo que pode ser compreendido por geofísicos, geólogos e engenheiros.

### 3.2.4 Coeficiente de Reflexão

Quando uma onda longitudinal atinge uma interface entre duas camadas com impedância acústica contrastante, parte desta onda vai se refletir e parte vai penetrar na segunda camada. Porém, são gerados quatro tipos de ondas longitudinal refletida, transmitida, transversal refletida e transmitida.

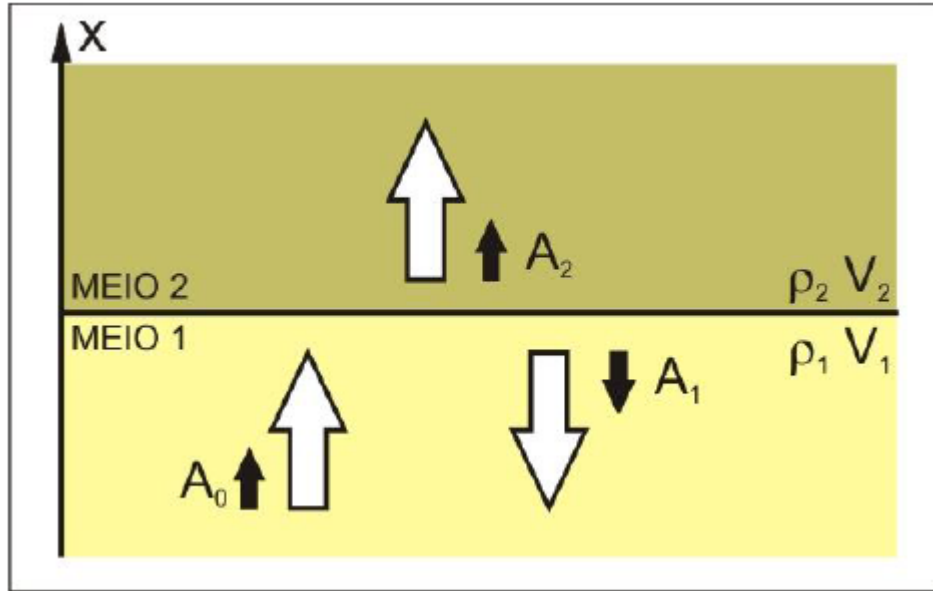


Figura 3.3: Incidência de uma onda acústica em uma interface entre rochas de impedância diferente

Se o ângulo de incidência for normal ou próximo do normal à interface, o coeficiente de reflexão será dado pela seguinte equação:

$$R = \frac{A_1}{A_0} = \frac{Z_2 - Z_1}{Z_2 + Z_1} \quad (3.9)$$

Sendo:

$R$  = Coeficiente de reflexão

$A_0$  = Amplitude da onda incidente

$A_1$  = Amplitude da onda refletida

$A_2$  = Amplitude da onda refratada

$Z_1$  = Impedância acústica da camada 1

$Z_2$  = Impedância acústica da camada 2

## 3.3 Identificação de pacotes

Logo abaixo temos o diagrama de pacotes para elaboração do software, como descrito na análise de domínio.

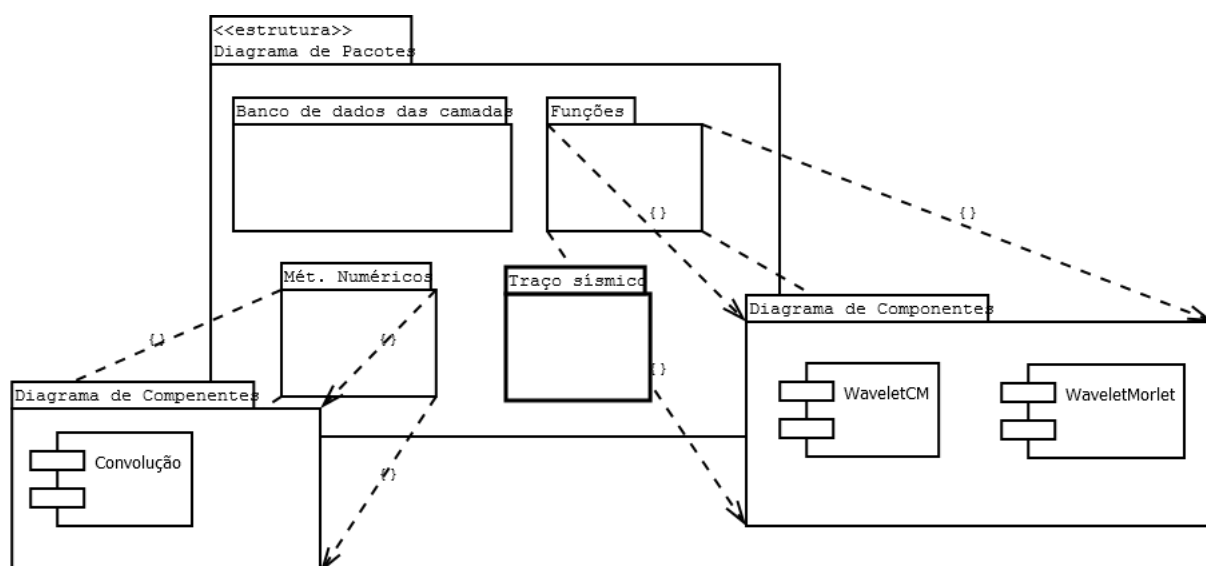


Figura 3.4: Diagrama de pacotes

# Capítulo 4

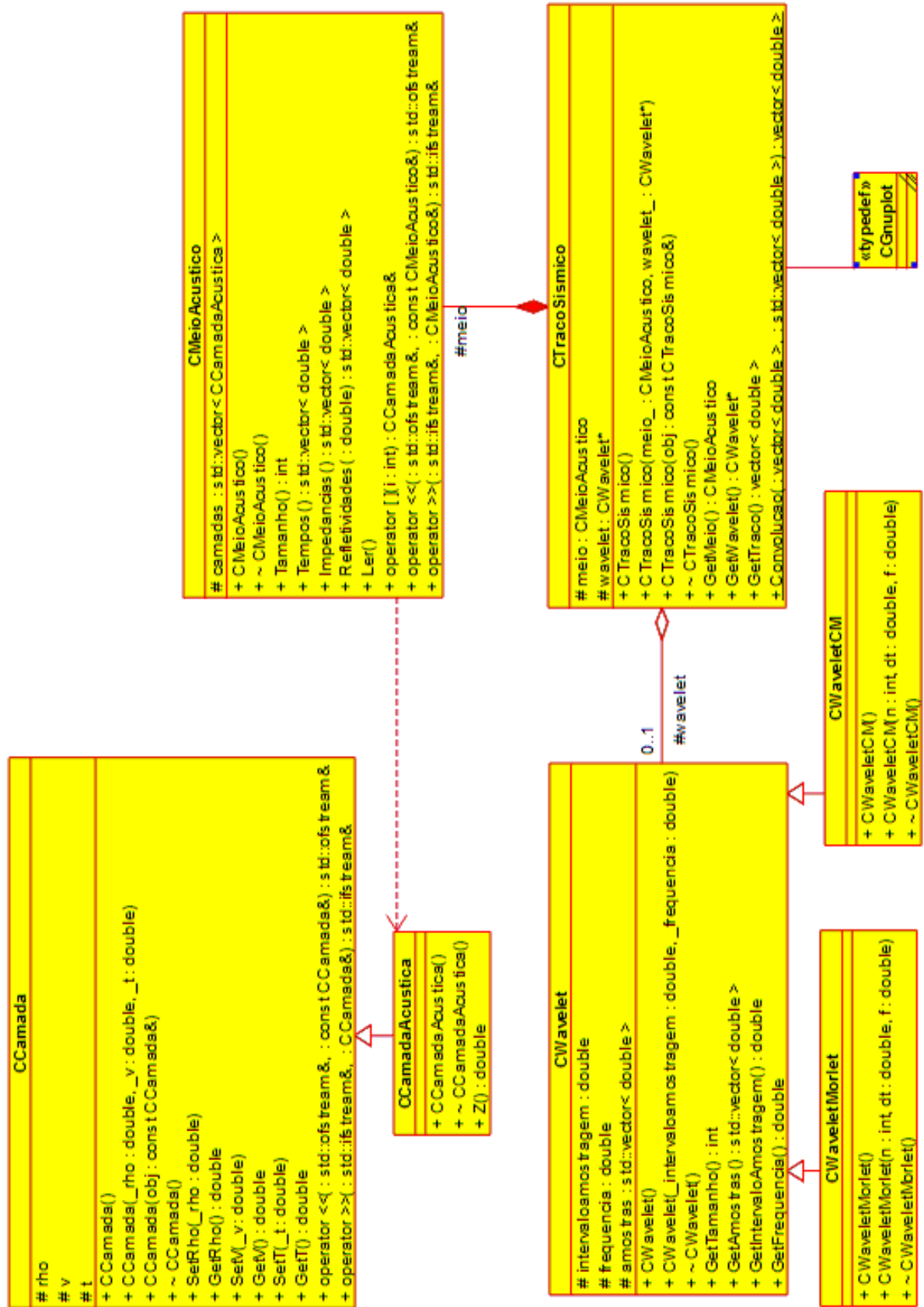
## AOO – Análise Orientada a Objeto

Nesta etapa do projeto de software, utilizam-se regras para identificar os objetos de interesse, que terão parte do mesmo, as relações entre as classes, os atributos, os métodos, as heranças e as associações. O modelo de análise deve ser conciso, simplificado e deve mostrar o que deve ser feito, sem se importar com a forma. A análise consiste em dois modelos: o estrutural (objetos e seus relacionamentos) e o dinâmico (transformações dos objetos em relação ao tempo).

### 4.1 Diagramas de classes

O diagrama de classes é apresentado na Figura 4.1.





### 4.1.1 Dicionário de classes

- Classe CWavelet: Representa a função que possui capacidade de decompor funções tanto no domínio do tempo quanto da frequência. A mesma é base para classes CWaveletMorlet e CWaveletCM.
- Classe CWaveletMorlet: Esta classe herda os atributos e métodos de CWavelet e implementa a particularidade da função Morlet.
- Classe CWaveletCM: Esta classe herda os atributos e métodos de CWavelet e implementa a particularidade da função Chapéu Mexicano.
- Classe CCamada: Inclui os atributos os dados de velocidade da onda cisalhante, duplo tempo e densidade da camada.
- Classe CCamadaAcustica: Herda os atributos e métodos de CCamada e calcula os valores de impedância.
- Classe CMeioAcustico: Cria um vetor de camada acústica e gera o vetor refletividade.
- Classe CTraciSismico: Realiza uma operação linear entre o vetor refletividade e a função wavelet que pode ser do tipo Morlet ou Chapéu Mexicano, resultando em um vetor que é o traço sísmico sintético terceira, esta classe utilizará dados calculados pelas classes CMeioAcustico, CWaveletCM e CWaveletMorlet.
- Classe CGnuplot: Classe que possibilita a geração de gráficos usando o programa externo Gnuplot..

## 4.2 Diagrama de seqüência – eventos e mensagens

Um diagrama de seqüência costuma identificar o evento gerador do processo modelado, bem como o ator responsável por este evento. Além disso, ilustra como o processo acontece. Por fim, é finalizado por meio do envio de mensagens que, em geral, disparam métodos entre os objetos.

### 4.2.1 Diagrama de seqüência geral

Veja o diagrama de seqüência na Figura 4.2. Observe que inicialmente o usuário insere os dados e o simulador ler o arquivo, a seguir o simulador calcula a impedância e a refletividade da camada, logo após o cálculo o usuário escolhe o tipo de *wavelet* a ser executada a convolução e gerar o traço sísmico que depois será plotado e salvo.

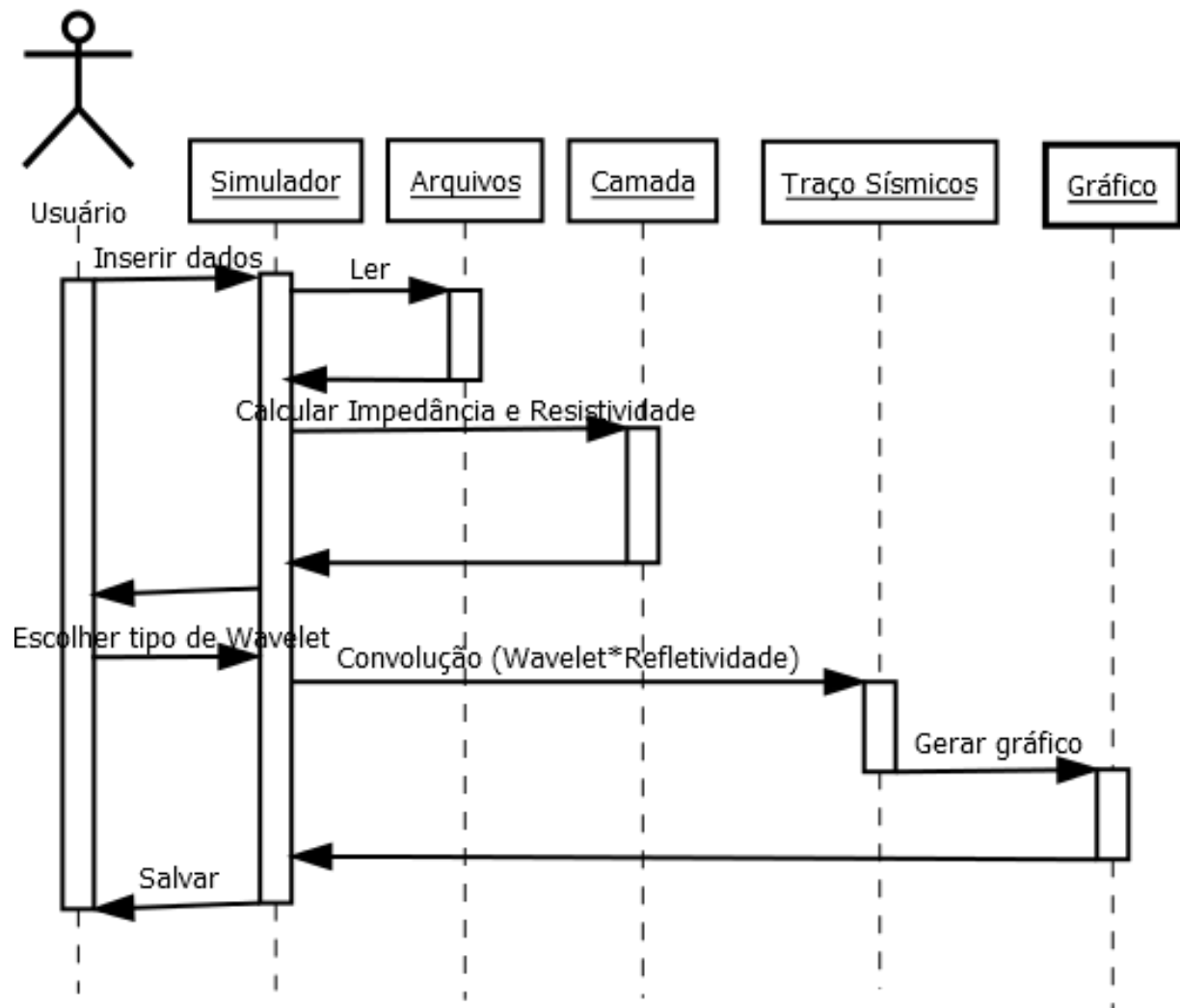


Figura 4.2: Diagrama de seqüência

### 4.3 Diagrama de comunicação – colaboração

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

Veja na Figura 4.3 o diagrama de comunicação. Observe que inicialmente o usuário entra com os dados pedindo o cálculo do traço sísmico. Feito isso são fornecidos esses dados para as demais classes efetuarem todos os cálculos e iterações necessárias para se encontrar a função do traço sísmico e gerar um gráfico da mesma.

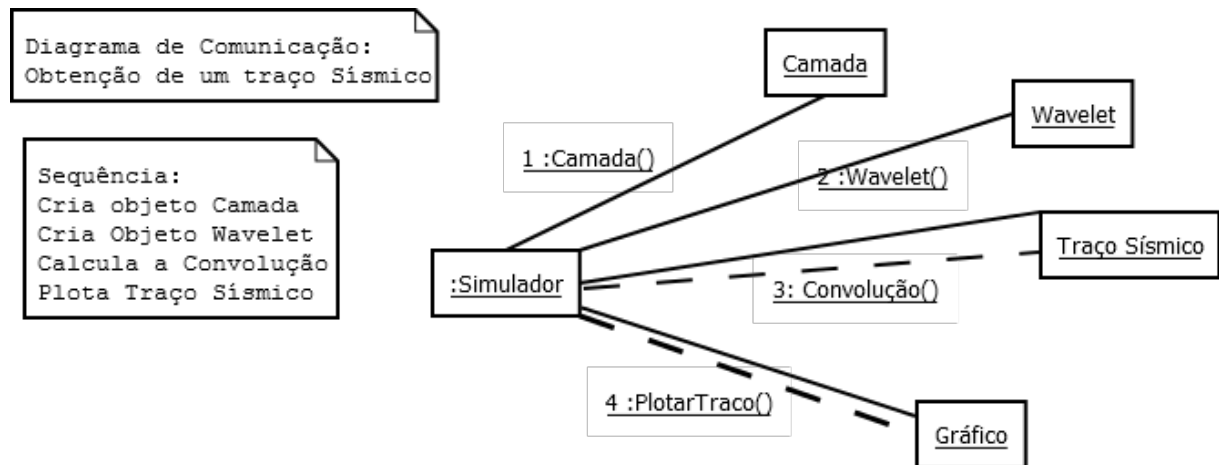


Figura 4.3: Diagrama de comunicação

## 4.4 Diagrama de máquina de estado

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

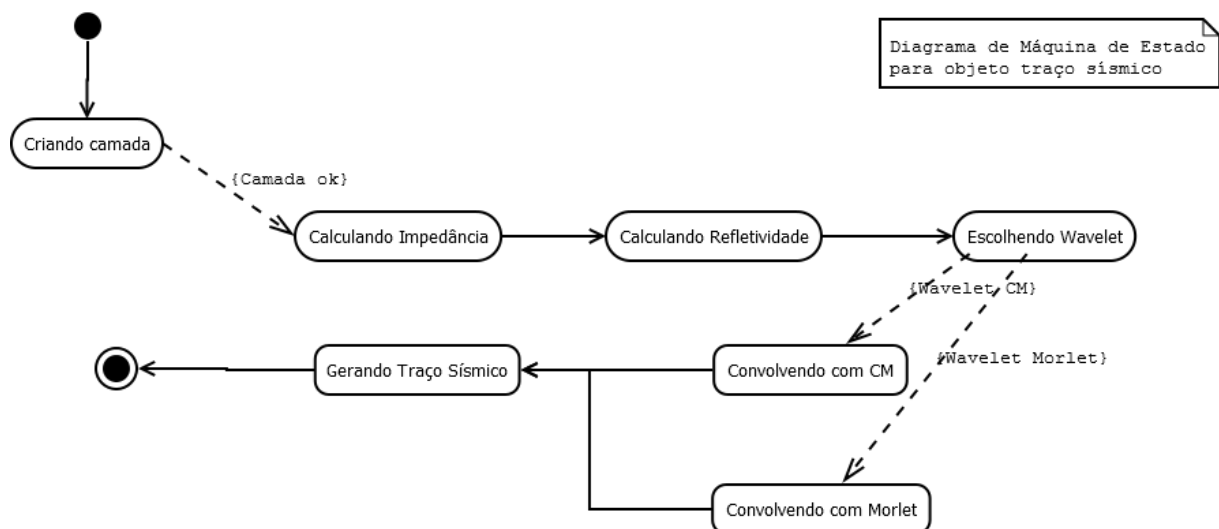


Figura 4.4: Diagrama de máquina de estado para objeto traço sísmico

## 4.5 Diagrama de atividades

Segue abaixo o diagrama de atividade com a descrição do cálculo de refletividade.4.5

Diagrama de Atividade:  
Cálculo da Refletividade

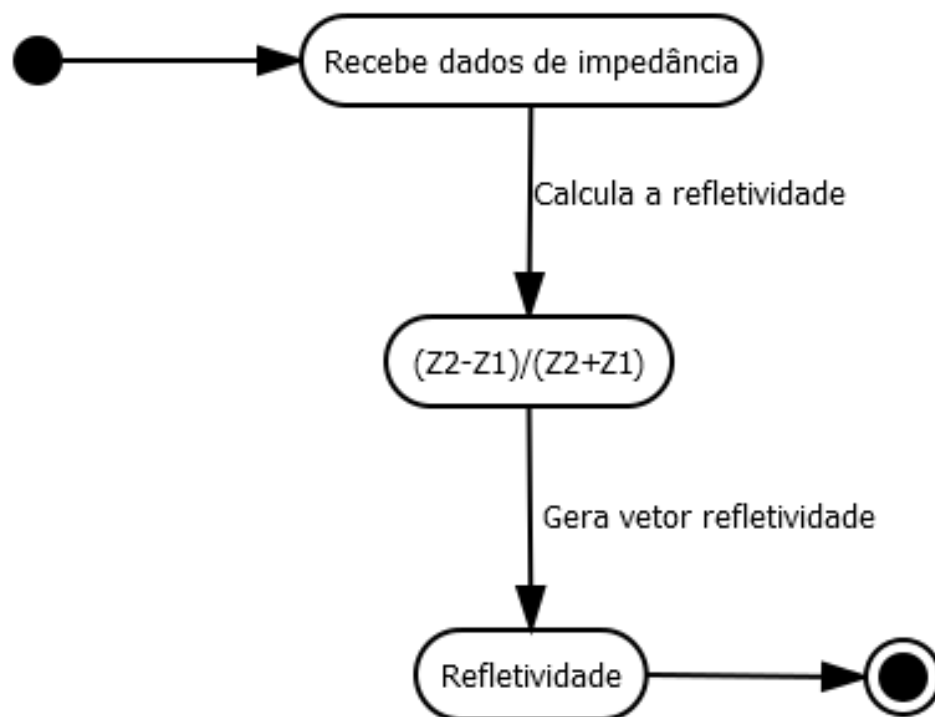


Figura 4.5: Diagrama de atividades: Cálculo da Refletividade

Segue abaixo o diagrama de atividade com a descrição do cálculo da convolução. 4.6

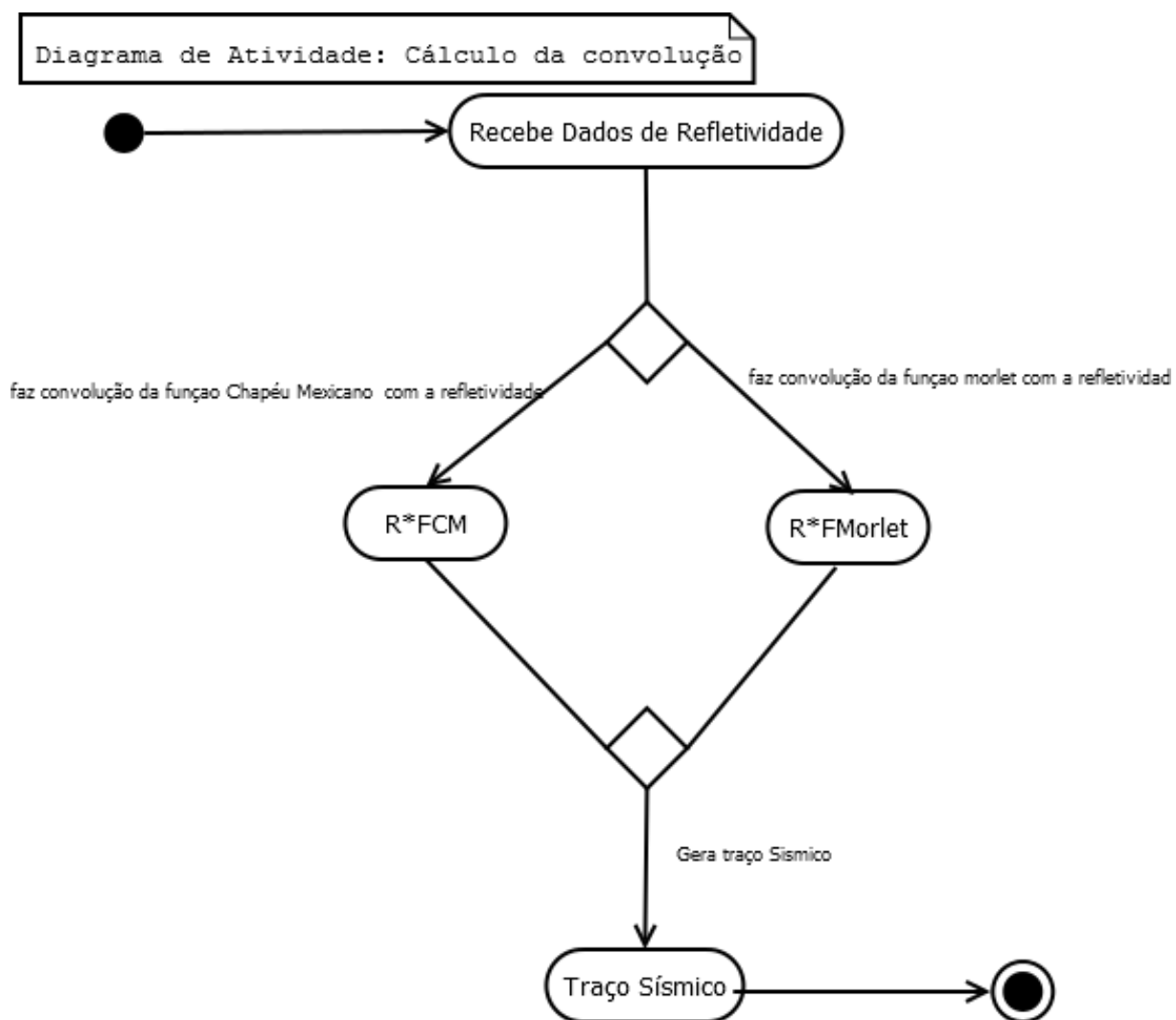


Figura 4.6: Diagrama de atividades: Cálculo da convolução

# Capítulo 5

## Projeto

Esse capítulo define o projeto do software em si, avaliando as plataformas à serem suportadas, os protocolos, recursos e interfaces utilizadas, associação à bibliotecas externas, subdivisão em hardwares, entre outros. A análise do projeto visa otimizar a estrutura do programa e otimizar os tempos de execução, memória, custos e desenvolver a estrutura dos dados.

### 5.1 Projeto do sistema

#### 1. Protocolos

- Não será necessário incluir intercomunicações entre o programa e componentes externos, uma vez que o programa terá como objetivo final salvar os resultados obtidos em um arquivo. Caberá ao usuário definir o destino posterior do arquivo (impressão, envio dos dados etc).
- Definição do formato dos arquivos de entrada pelo programa.
  - O programa terá como entrada arquivos de extensão .txt

#### 2. Recursos

- O presente programa precisará utilizar o HD, o processador, o teclado, a tela, o mouse, a memória e demais componentes internos do computador.

#### 3. Controle

- Identificação e seleção da implementação de controle, sequencial ou concorrente, baseado em procedimentos ou eventos. Implicam modificações no diagrama de execução.
  - Neste projeto o controle será sequencial.
- Identificação das condições extremas e de prioridades.

- Não se aplica.
- Identificação da necessidade de otimização. Por exemplo: prefira sistemas com grande capacidade de memória; prefira vários hds pequenos a um grande.
  - Neste projeto não ha necessidade de uso de processos de otimização. Os cálculos realizados requerem pouco espaço na memória, tanto física, quanto de processamento.
- Identificação e definição de *loops* de controle e das escalas de tempo.
  - Não se aplica.
- Identificação de concorrências – quais algoritmos podem ser implementados usando processamento paralelo.
  - Neste projeto não ha necessidade de uso de processos de processamento paralelo pois os cálculos realizados requerem pouco esforço de processamento.

#### 4. Plataformas

- Identificação das estruturas arquitetônicas comuns.
- Identificação de subsistemas relacionados à plataforma selecionada. Podem implicar em modificações no diagrama de pacotes e no diagrama de componentes
- Identificação e definição das plataformas a serem suportadas: hardware, sistema operacional e linguagem de programação.
  - O software é multiplataforma, funciona no Windows e GNU/Linux.
  - A linguagem selecionada é C++.
- Seleção das bibliotecas externas a serem utilizadas.
  - O software usa a biblioteca externa CGnuplot, que permite o acesso ao gerador de gráfico gnuplot.

#### 5. Padrões de projeto

- Normalmente os padrões de projeto são identificados e passam a fazer parte de uma biblioteca de padrões da empresa. Mas isto só ocorre após a realização de diversos projetos.
  - Não se aplica.



## 5.2 Projeto orientado a objeto – POO

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

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

### Efeitos do projeto no modelo estrutural

- Adicionar nos diagramas de pacotes as bibliotecas e subsistemas selecionados no projeto do sistema (exemplo: a biblioteca gráfica selecionada).
  - Neste projeto foi utilizado como biblioteca gráfica CGnuplot.
- Novas classes e associações oriundas das bibliotecas selecionadas e da linguagem escolhida devem ser acrescentadas ao modelo.
  - Neste projeto foi feita uma associação entre a biblioteca CGnuplot e a CTracoSismico para geração do gráfico.

### Estabelecer as dependências e restrições associadas à plataforma escolhida. Efeitos do projeto no modelo dinâmico

- Revisar os diagramas de seqüência e de comunicação considerando a plataforma escolhida.
  - Não se aplica.
- Verificar a necessidade de se revisar, ampliar e adicionar novos diagramas de máquinas de estado e de atividades.
  - Não se aplica.

### Efeitos do projeto nos atributos

- Atributos novos podem ser adicionados a uma classe, como, por exemplo, atributos específicos de uma determinada linguagem de programação (acesso a disco, ponteiros, constantes e informações correlacionadas).
  - Neste projeto o objeto gnuplot foi adicionado na classe CTracoSismico.

### Efeitos do projeto nos métodos

- Em função da plataforma escolhida, verifique as possíveis alterações nos métodos. O projeto do sistema costuma afetar os métodos de acesso aos diversos dispositivos (exemplo: hd, rede).
  - Não se aplica.
- De maneira geral os métodos devem ser divididos em dois tipos: i) tomada de decisões, métodos políticos ou de controle; devem ser claros, legíveis, flexíveis e usam polimorfismo. ii) realização de processamentos, podem ser otimizados e em alguns casos o polimorfismo deve ser evitado.
- Algoritmos complexos podem ser subdivididos. Verifique quais métodos podem ser otimizados. Pense em utilizar algoritmos prontos como os da STL (algoritmos genéricos).
  - Neste projeto os métodos refletividade da classe CMeioAcustico e convolução da classe CTracoSismico.
- Responda a pergunta: os métodos das classes estão dando resposta às responsabilidades da classe?
  - Neste projeto todos os métodos das classes estão correspondendo às responsabilidades da classe.
- Revise os diagramas de classes, de sequência e de máquina de estado.
  - Não se aplica.

### Efeitos do projeto nas heranças

- Reorganização das classes e dos métodos (criar métodos genéricos com parâmetros que nem sempre são necessários e englobam métodos existentes).
  - Não se aplica.
- Abstração do comportamento comum (duas classes podem ter uma superclasse em comum).
  - Neste projeto foi implementado o conceito de abstração do comportamento comum nas seguintes classes CWaveletMorlet e CWaveletCM, onde CWavelet é a superclasse.

- Utilização de delegação para compartilhar a implementação (quando você cria uma herança irreal para reaproveitar código). Usar com cuidado.
  - Não se aplica.

### Efeitos do projeto nas associações

- Deve-se definir na fase de projeto como as associações serão implementadas, se obedecerão um determinado padrão ou não.
- Se existe uma relação de "muitos", pode-se implementar a associação com a utilização de um dicionário, que é um mapa das associações entre objetos. Assim, o objeto A acessa o dicionário fornecendo uma chave (um nome para o objeto que deseja acessar) e o dicionário retorna um valor (um ponteiro) para o objeto correto.
  - Neste projeto foi utilizado ponteiro na classe CWavelet.
- Evite percorrer várias associações para acessar dados de classes distantes. Pense em adicionar associações diretas.
  - A classe CGnuplot foi associada na classe CTracoSismico.

## 5.3 Diagrama de componentes

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

Veja na Figura 5.1 o diagrama de componentes. A geração dos objetos depende dos arquivos de classe de extensão `.h` e `.cpp`. O subsistema ArquivoEntrada representa o arquivo que o programa importará os dados a serem manipulados. O programa executável a ser gerado depende das bibliotecas, dos arquivos desta e dos arquivos de entrada.

## 5.4 Diagrama de implantação

O diagrama de implantação é um diagrama que inclui relações entre o sistema e o hardware e deve incluir os elementos necessários para que o sistema seja colocado em funcionamento. Veja na Figura 5.2 diagrama de implantação do programa. Primeiramente, os Perfis registram a velocidade da onda nas camadas, a profundidade e a densidade das camadas, enviando os dados para um computador na superfície. Esses arquivos são compilados em formato `.txt`. O programa importa os dados desse arquivo e na sua execução precisa de um monitor para mostrar os resultados e do teclado para receber parâmetros informados pelo usuário ou cliente.

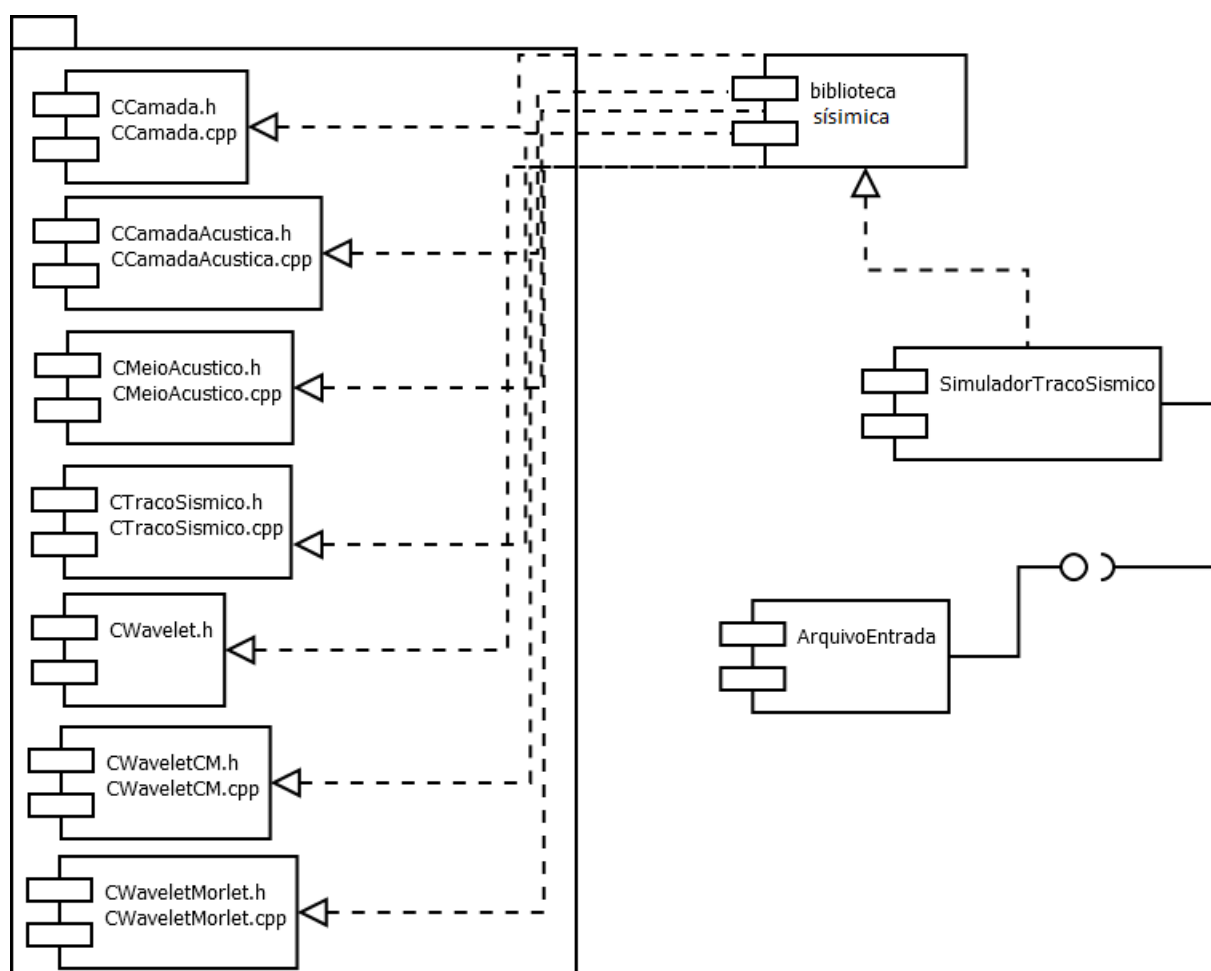


Figura 5.1: Diagrama de componentes.

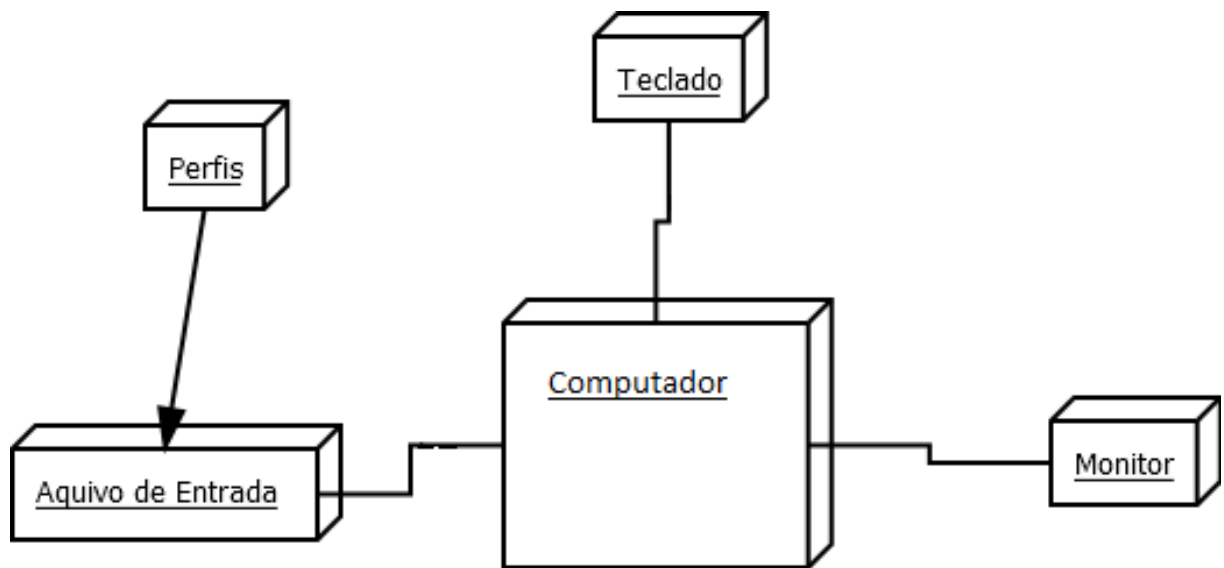


Figura 5.2: Diagrama de implantação

# Capítulo 6

## Implementação

Neste capítulo apresentaremos o código do simulador de traço sísmico sintético.

### 6.1 Código fonte

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

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

```
#ifndef CCamada_h
#define CCamada_h

#include <fstream>

/*Esta classe representa as características da camada.
Recebe de um arquivo de entrada com os atributos densidade,
velocidade e duplo tempo (assim sabe-se a profundidade da camada)
**/

class CCamada{

protected:
//atributos
    double rho; // densidade
    double v; // velocidade da onda cisalhante
    double t; // duplo tempo

public:

    CCamada(): rho(0.0), v(0.0), t(0.0) {}

    CCamada( double _rho, double _v, double _t ): rho( _rho ), v( _v
    ), t( _t ) {} //Construtor sobrecarregado
```

```

        CCamada( const CCamada &obj ): rho( obj.rho ), v( obj.v ) , t(
            obj.t ) {}          //Construtor de cópia

//Destrutor
    virtual ~CCamada() {}

//Métodos
    void SetRho( double _rho ) { rho = _rho; } // seta o atributo
        densidade
    double GetRho() const { return rho; } // retorna o
        atributo densidade
    void SetV( double _v ) { v = _v; } // seta o atributo
        velocidade
    double GetV() const { return v; } // seta o atributo
        velocidade
    void SetT(double _t) { t = _t; } // seta o atributo tempo
    double GetT() const { return t; } // seta o atributo tempo

//sobrecarga de operador
    friend std::ofstream& operator<<(std::ofstream&, const CCamada&)
        ; //Ler do arquivo de entrada
    friend std::ifstream& operator>>(std::ifstream&, CCamada&); //
        salva no arquivo de saída
};

#endif

```

Listing 6.1: Arquivo de cabeçalho da classe CCamada.

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

```

#include "CCamada.h"

std::ofstream& operator<<(std::ofstream& fout, const CCamada& c)
{
    fout << c.rho << " " << c.v << " " << c.t;
    return fout;
};

std::ifstream& operator>>(std::ifstream& fin, CCamada& c)
{
    fin >> c.rho >> c.v >> c.t;
    return fin;
};

```

Listing 6.2: Arquivo de implementação da classe CCamada.

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

```
#ifndef CCamadaAcustica_h
#define CCamadaAcustica_h

#include "CCamada.h"

/* Esta classe recebe os atributos e os métodos
da classe CCamada e calcula a impedância
**/

class CCamadaAcustica: public CCamada //herança
{
public:

    CCamadaAcustica(){} // Construtor default
    CCamadaAcustica(double _rho, double _v, double _t):CCamada(_rho, _v,
        _t){}
    virtual ~CCamadaAcustica() {} // Destrutor
    virtual double Z() const; // Método para Cálculo
        de impedância
};

#endif
```

Listing 6.3: Arquivo de cabeçalho da classe CCamadaAcustica.

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

```
#include "CCamadaAcustica.h"

double CCamadaAcustica::Z() const
{
    return rho*v; // Cálculo da impedância = Densidade x
        Velocidade
};
```

Listing 6.4: Arquivo de implementação da classe CCamadaAcustica.

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

```
#ifndef CMeioAcustico_h
#define CMeioAcustico_h

#include <vector>
```



```

#include <fstream>
#include "CCamadaAcustica.h"
#include <iostream>

/* Esta classe representa um conjunto de camadas ordenadas em um vetor
   onde sua principal função é o calculo do vetor Refletividade
**/

class CMeioAcustico
{
protected:
    std::vector<CCamadaAcustica> camadas; // Atributo

public:
    CMeioAcustico() {} //construtor default
    ~CMeioAcustico() {} // destrutor

    // Métodos
    int Tamanho() const { return camadas.size(); }
    std::vector<double> Tempos() const;
    std::vector<double> Impedancias() const;
    std::vector<double> Refletividades(double) const;
    void Ler();

    //Sobrecarga de Operador
    CCamadaAcustica& operator[](int i) { return camadas[i]; }
    friend std::ofstream& operator<< (std::ofstream&, const
        CMeioAcustico& ); // Ler
    friend std::ifstream& operator>> (std::ifstream&, CMeioAcustico
        & ); // Salvar
};

#endif

```

Listing 6.5: Arquivo de cabeçalho da classe CMeioAcustico.

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

```

#include "CMeioAcustico.h"
#include <iostream>

using namespace std;

//Método para preencher o vetor de tempo
vector<double> CMeioAcustico::Tempos() const
{
    vector<double> tempos;

```

```

    for (int i=0; i<camadas.size(); i++)
        tempos.push_back(camadas[i].GetT());
    return tempos;
};

//Método para preencher o vetor de impedância
vector<double> CMeioAcustico::Impedancias() const
{
    vector<double> impedancias;
    for (int i=0; i<camadas.size(); i++)
        impedancias.push_back(camadas[i].Z());
    return impedancias;
};

//Calculo da Refletividade  $(z_2 - z_1) / (z_2 + z_1)$ 
vector<double> CMeioAcustico::Refletividades(double dt) const
{
    vector<double> refletividades;
    double t = 0.0;
    for (int i=1; i<camadas.size(); i++)
    {
        t += camadas[i-1].GetT();
        for (int j=0; j<t/dt; j++)
            refletividades.push_back(0);
        double r = (camadas[i].Z() - camadas[i-1].Z()) / (camadas[i].Z() +
            camadas[i-1].Z());
        refletividades.push_back(r);
    };
    t += camadas[camadas.size()-1].GetT();
    for (int j=0; j<t/dt; j++)
        refletividades.push_back(0);

    return refletividades;
};

ofstream& operator<<(ofstream& fout, const CMeioAcustico& m)
{
    for (int i=0; i<m.camadas.size(); i++)
        fout << m.camadas[i] << endl;
    return fout;
};

ifstream& operator>>(ifstream& fin, CMeioAcustico& m)
{
    while (true)
    {
        CCamadaAcustica camada;
        fin >> camada;
    }
}

```

```
        if (!fin.good())
            break;
        m.camadas.push_back(camada);
    };
    return fin;
};

void CMeioAcustico::Ler()
{
    int n;

    cout << "Quantas camadas você quer criar?\n";

    cin >> n;

    cin.get();

    for (int i = 0; i < n; i++)
    {
        double rho;
        double v;
        double t;

        cout << "Entre com o valor de densidade da camada " << i+1 << "\n";

        cin >> rho;

        cin.get();

        cout << "Entre com o valor de velocidade na camada " << i+1 << "\n";

        cin >> v;

        cin.get();

        cout << "Entre com o tempo de trânsito da onda na camada " << i+1 << "\n";

        cin >> t;

        cin.get();

        CCamadaAcustica camada(rho, v, t);
```

```

camadas.push_back(camada);

}

}

```

Listing 6.6: Arquivo de implementação da classe CMeioAcustico.

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

```

#ifndef CWavelet_h
#define CWavelet_h

#include <vector>

/* Esta classe é a classe Base das classes CWaveletMorlet e CWaveletCM,
   nela todos os atributos estão declarados onde as respectivas classes
   irão herdar e
   implementar de acordo com a particularidade de cada uma.
   */
class CWavelet
{
protected:

    //Declaração dos Atributos
    double intervaloamostragem;        // Atributo Intervalo de
        amostragem
    double frequencia;                  // Atributo frequência
    std::vector<double> amostras;      // Atributo Amostra

public:

    CWavelet() {} //construtor
    CWavelet( double _intervaloamostragem, double _frequencia )
        : intervaloamostragem( _intervaloamostragem ), frequencia(
            _frequencia ) {} //Construtor sobrecarregado
    virtual ~CWavelet() {} // destrutor

    int GetTamanho() const { return amostras.size(); } // Método que
        retorna o tamanho o vetor amostra
    std::vector<double> GetAmostras() const { return amostras; } //
        Método que retorna o vetor amostra
    virtual double GetIntervaloAmostragem() const { return
        intervaloamostragem; } // retorna o atributo intervalo amostragem
    virtual double GetFrequencia() const { return frequencia; }
        // retorna o atributo frequencia
};

#endif

```

Listing 6.7: Arquivo de cabeçalho da classe CWavelet.

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

```
#ifndef CWaveletMorlet_h
#define CWaveletMorlet_h

#include "CWavelet.h"

class CWaveletMorlet: public CWavelet
{
public:

    CWaveletMorlet() {} // Construtor
    default
    CWaveletMorlet(int n, double dt, double f); // construtor
    sobrecarregado
    virtual ~CWaveletMorlet() {} // destrutor

};

#endif
```

Listing 6.8: Arquivo de cabeçalho da classe CWaveletMorlet.

Apresenta-se na listagem 6.9 o arquivo de implementação da classe CWaveletMorlet.

```
#include "CWaveletMorlet.h"
#include <math.h>

CWaveletMorlet::CWaveletMorlet( int n, double dt, double f )
: CWavelet(dt, f)
{
    double aux = pow(2.0*M_PI*f, 2);
    double t0 = -dt*(n/2);
    for (int i=0; i<n; i++)
    {
        double t = t0 + i*dt;
        amostras.push_back(( cos( 2.0*M_PI*f*t ) - exp( -0.5*aux ))*exp(
            -M_PI*f*t*t ));
    };
};
```

Listing 6.9: Arquivo de implementação da classe CWaveletMorlet.

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

```
#ifndef CWaveletCM_h
```

```

#define CWaveletCM_h

#include "CWavelet.h"

/*Esta classe herda os atributos e métodos da classe CWavelet e
  inicializa os atributos por um construtor sobrecarregado
  **/

class CWaveletCM: public CWavelet
{

public:

    CWaveletCM() {} // construtor default
    CWaveletCM ( int n, double dt, double f ); // construtor
        sobrecarregado
    virtual ~CWaveletCM() {} // destrutor

};

#endif

```

Listing 6.10: Arquivo de cabeçalho da classe CWaveletCM.

Apresenta-se na listagem 6.11 o arquivo de implementação da classe CWaveletRicker

```

#include "CWaveletCM.h"
#include <math.h>

CWaveletCM::CWaveletCM (int n, double dt, double f )
: CWavelet(dt,f)
{
    double aux = pow( M_PI*f, 2 );
    double t0 = -dt*( n/2 );
    for ( int i=0; i<n; i++ )
    {
        double t = t0 + i*dt;
        amostras.push_back(( 1.0 - 2.0*aux*t*t )*exp( -aux*t*t ));
    };
};

```

Listing 6.11: Arquivo de implementação da classe CWaveletCM.

Apresenta-se na listagem 6.12 o arquivo com código da classe CTracoSismico.

```

#ifndef CTracoSismico_h
#define CTracoSismico_h

#include <vector>

```

```

#include "CMeioAcustico.h"
#include "CWavelet.h"
#include "CGnuplot.h"

///Esta classe executa a convolução entre o vetor de refletividades e a
função wavelet escolhida

class CTracoSismico
{
protected:
    ///atributos
    CMeioAcustico meio;
    CWavelet* wavelet;

public:
    CTracoSismico() {}///construtor default
    CTracoSismico( CMeioAcustico meio_, CWavelet* wavelet_ ): meio(
        meio_ ), wavelet( wavelet_ ) {} ///construtor sobrecarregado
    CTracoSismico( const CTracoSismico &obj ): meio( obj.meio ), wavelet
        ( obj.wavelet ) {} ///construtor de cópia
    ~CTracoSismico() {} ///destrutor

    ///métodos
    CMeioAcustico GetMeio() { return meio; }
    CWavelet* GetWavelet() { return wavelet; }
    std::vector< double > GetTraco() const;
    static std::vector<double> Convolucao(std::vector< double >, std::
        vector< double > );
};

#endif

```

Listing 6.12: Arquivo de cabeçalho da classe CTracoSismico.

Apresenta-se na listagem 6.13 o arquivo de implementação da classe CTracoSismico.

```

#include "CTracoSismico.h"

using namespace std;

///implementação da convolução
vector< double > CTracoSismico::Convolucao( vector<double> x, vector<
    double> y)
{
    vector< double > z( x.size(), 0.0 );
    int d = ( y.size() - 1 )/2;

```

```

    for (int i=d; i<y.size()-1; i++)
        for (int j=0; j<i+1; j++)
            z[i-d] += x[j]*y[i-j];

    for (int i=y.size()-1; i<x.size(); i++)
        for (int j=i+1-y.size(); j<i+1; j++)
            z[i-d] += x[j]*y[i-j];

    for ( int i=x.size(); i<x.size()+d; i++ )
        for ( int j=i+1-y.size(); j<x.size(); j++ )
            z[i-d] += x[j]*y[i-j];

    return z;
};

//implementação da convolução entre a wavelet e a refletividade,
//sendo armazenado em um vetor 'traco'
vector< double > CTracoSismico::GetTraco() const
{
    double dt = wavelet->GetIntervaloAmostragem();
    vector< double > refl = meio.Refletividades(dt);
    vector< double > wlet = wavelet->GetAmostras();
    vector< double > traco = Convolucao(refl, wlet);
    return traco;
};

```

Listing 6.13: Arquivo de cabeçalho da classe CTracoSismico.

Apresenta-se na listagem 6.14 o programa main.cpp.

```

#include "CWaveletMorlet.h"
#include "CWaveletCM.h"
#include "CMeioAcustico.h"
#include "CTracoSismico.h"

#include <fstream>
#include <iostream>
#include <vector>
#include <string>

//Este é o programa principal

using namespace std;

int main()
{
    cout << "-----Geração de Traço Sísmico\n" ;
}

```



```

cout << "\n\n Autores: Ramon Diogo e Sávio Januario \n Disciplina:
    Programação Prática em C++ \n" ;
cout << "\n\n
    ----- \n" ;

cout << "Você deseja carregar os dados manualmente ou ler do arquivo
    ? \n(1)Carregar Manualmente \n(2)Ler do arquivo\n";
int escolha; //cria variável inteira
cin >> escolha; //armazena o valor digitado na variável
cin.get();
cout << "\n\n
    ----- \n";
CMeioAcustico meio; //cria objeto do tipo CMeioAcustico
switch (escolha) //cria uma função condicional
{
    case 1:
        meio.Ler(); //executa a função Ler() da classe
            CMeioAcustico
        break;

    case 2:
        string nomearquivo1; //cria variável do tipo string
        cout << "Qual o nome do arquivo com as propriedades do
            meio acústico? \n";
        cin >> nomearquivo1; //armazena nome digitado na
            variável
        ifstream fin(nomearquivo1.c_str());
            fin >> meio; //Lê arquivo
            fin.close();
        cin.get();
        break;
}
CWavelet* wav; //cria ponteiro da classe CWavelet
int escolha1;
cout << "\n
    ----- \n";
cout << "Qual tipo de wavelet deseja utilizar?\n(1) Chapéu
    Mexicano\n(2) Morlet\n" ;
cin >> escolha1;
cin.get();

int n;
cout << "Qual o número de amostras da wavelet? \n";
cin >> n;
cin.get();

```

```

    double dt;
    cout << "Qual o intervalo em segundos de amostragem da wavelet?
        \n";
    cin >> dt;
    cin.get();

    double f;
    cout << "Qual a frequência em Hertz de pico da wavelet? \n";
    cin >> f;
    cin.get();

switch (escolha1)
{
    case 1:
        wav = new CWaveletCM( n, dt, f );
        break;
    case 2:
        wav = new CWaveletMorlet( n, dt, f );
        break;
}

    CTracoSismico traco(meio, wav); //cria objeto do tipo
        CtracoSismico inicializando-o com o objeto do tipo
        CMeioAcustico e o ponteiro da classe CWavelet como parâmetros

    string nomearquivo2; //cria variável string
    cout << "\n\a
        -----\n";
    cout << "Qual o nome do arquivo de saída do traço sísmico? \n";
    cin >> nomearquivo2; //armazena o nome digitado na variável
    cin.get();
    ofstream fout(nomearquivo2.c_str());
    vector<double> tr = traco.GetTraco(); //armazena a convolução no
        vetor 'tr'
    for (int i=0; i<tr.size(); i++)
        fout << tr[i] << endl;
    fout.close();

CGnuplot plot = CGnuplot ("lines");
plot.plotfile_x(nomearquivo2.c_str()); //plota utilizando o gnuplot

    cin.get();
    cin.get();

```

```
    return 0;  
};
```

Listing 6.14: Arquivo de implementação da função `main()`.

# Capítulo 7

## Teste

Neste capítulo apresentaremos um teste do Simulador do Traço Sísmico com dados gerados especificamente para este software.

### 7.1 Testes

O primeiro grupo de dados é apresentado a seguir que foram inseridos no arquivo `entradataeste.txt` para leitura do programa. 7.1.

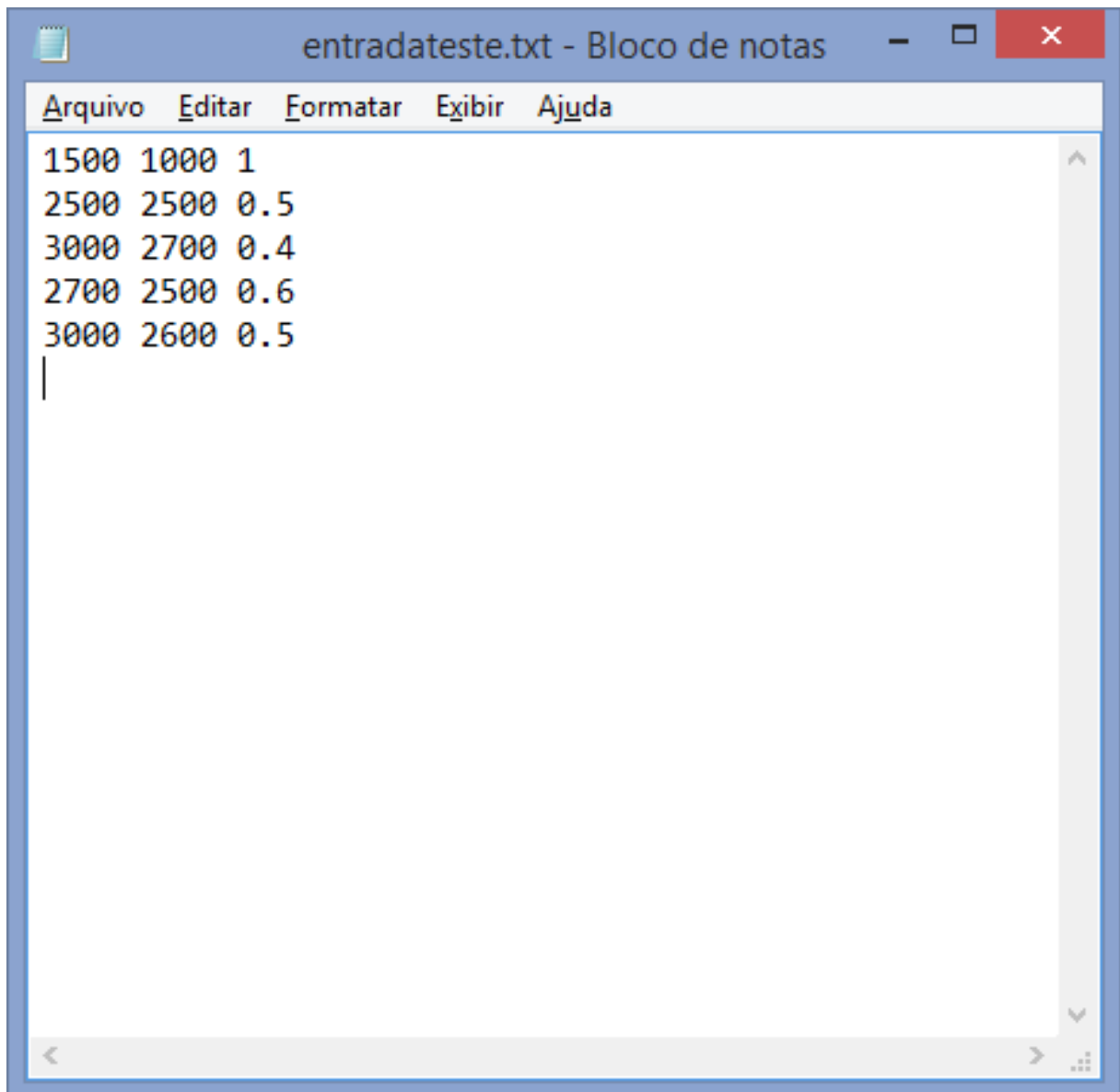


Figura 7.1: Tela mostrando o arquivo de entrada de dados

### 7.1.1 Tipo de entrada

A resposta para usuário que deseja com os dados via arquivo de disco:

Usuário informa o nome do arquivo de entrada:

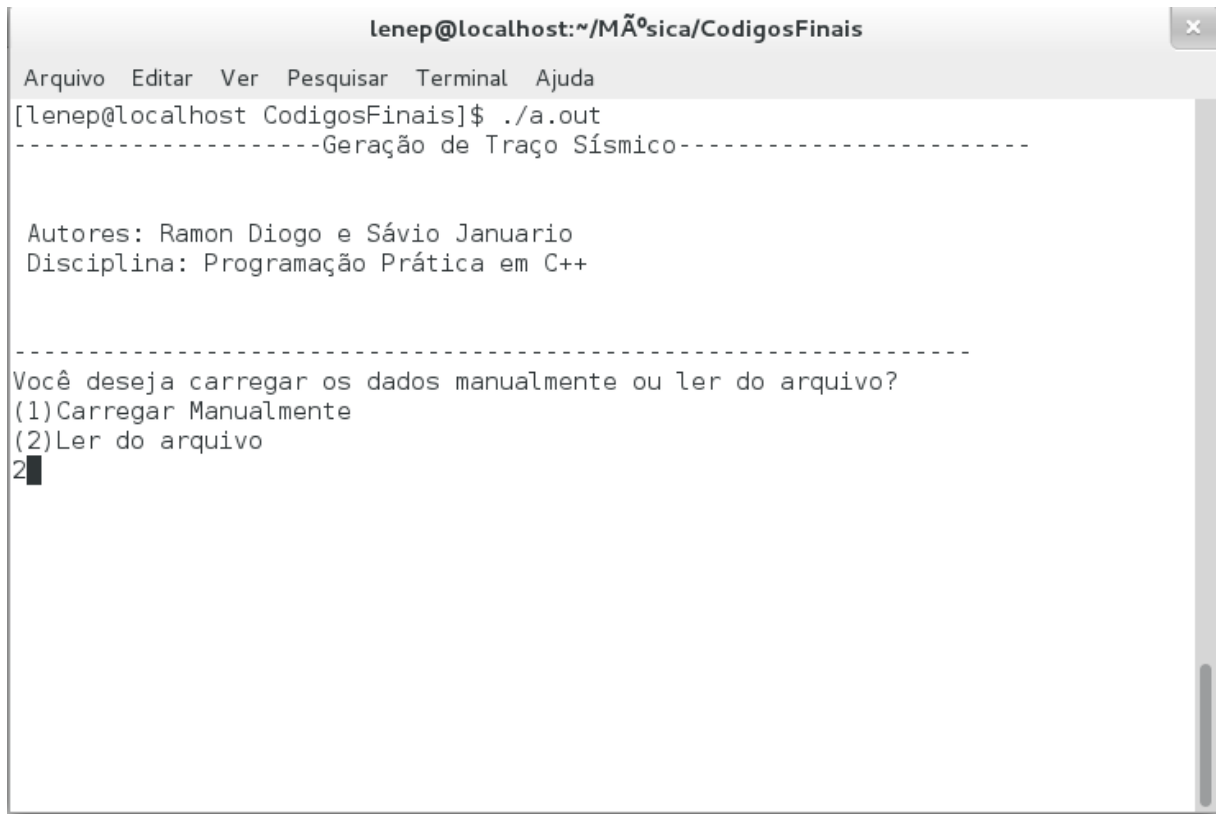
A resposta para usuário que deseja entrar manualmente com os dados:

Entrando com os valores manualmente:

### 7.1.2 Teste usando Chapéu Mexicano

### 7.1.3 Teste usando Morlet

A resposta do programa é apresentada a seguir:

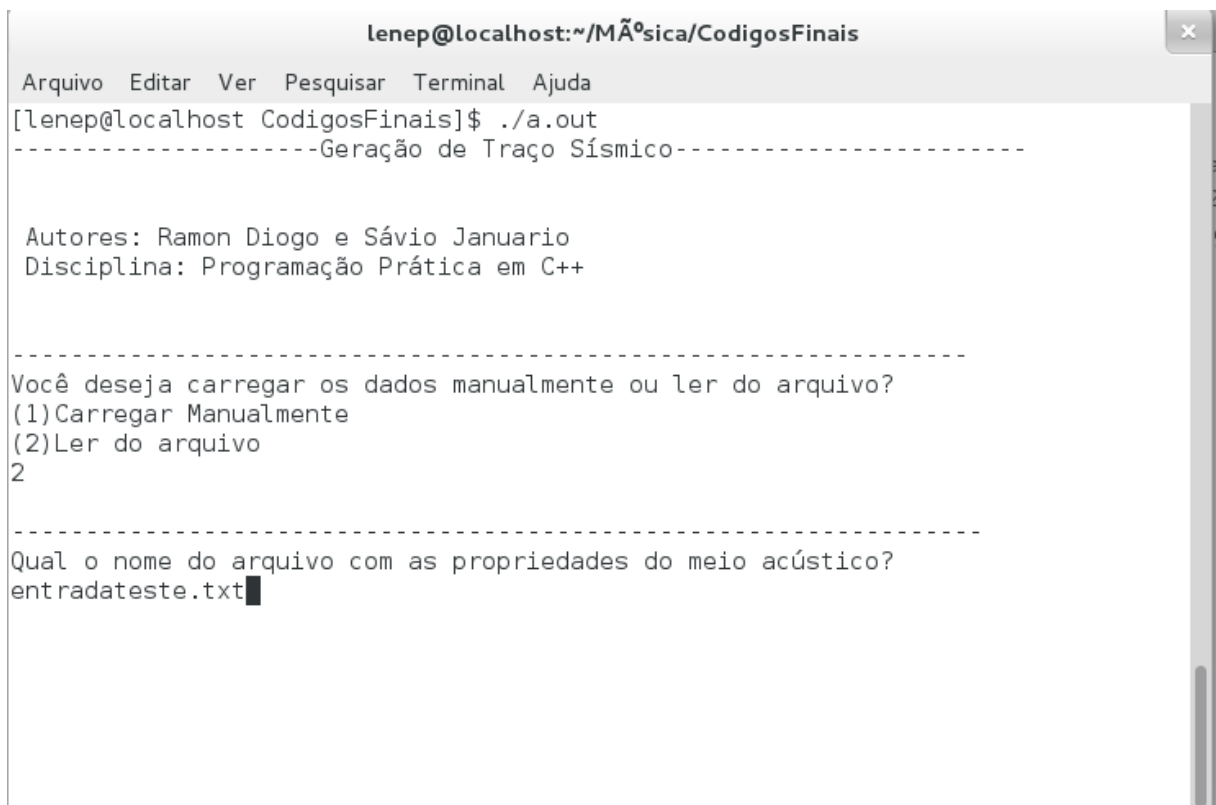


```
lenep@localhost:~/MÃsica/CodigosFinais
Arquivo Editar Ver Pesquisar Terminal Ajuda
[lenep@localhost CodigosFinais]$ ./a.out
-----Gerao de Trao Ssmico-----

Autores: Ramon Diogo e Svio Januario
Disciplina: Programao Prtica em C++

-----
Voc deseja carregar os dados manualmente ou ler do arquivo?
(1)Carregar Manualmente
(2)Ler do arquivo
2
```

Figura 7.2: Escolher o tipo de entrada de dados: Arquivo de disco

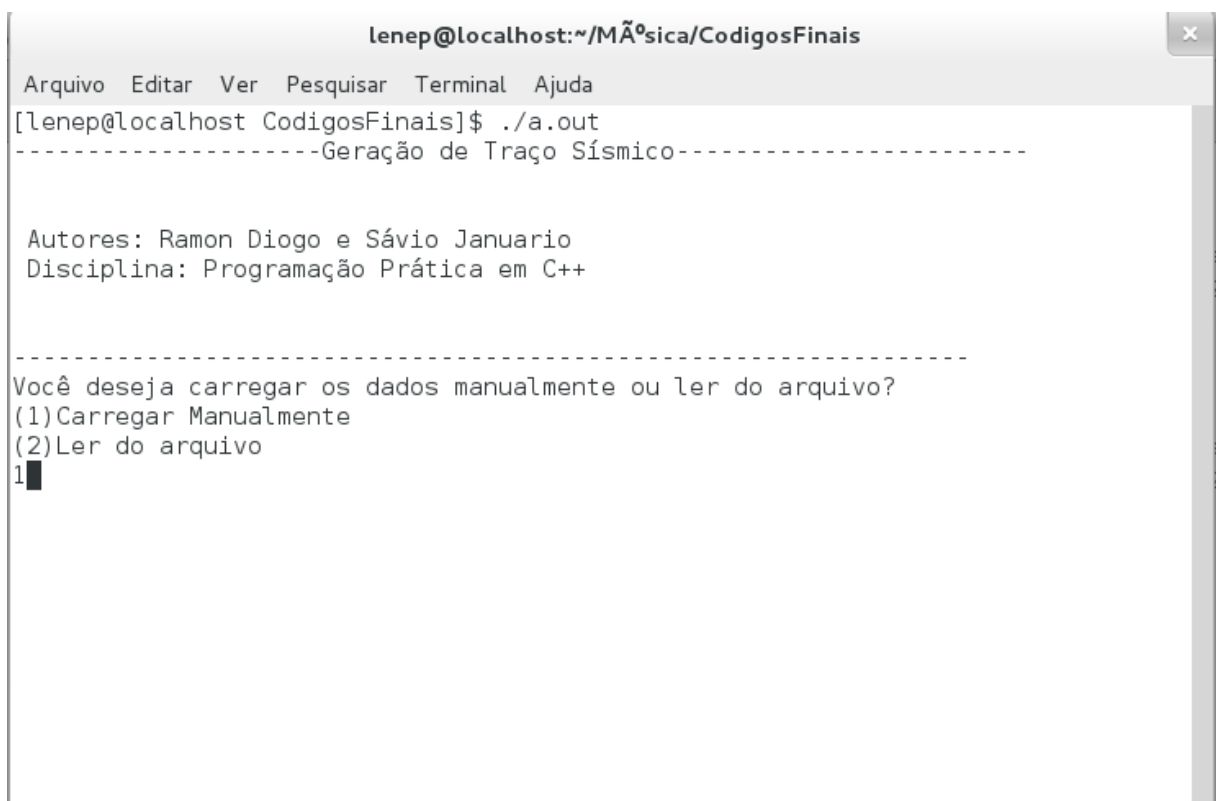


```
lenep@localhost:~/MÃsica/CodigosFinais
Arquivo Editar Ver Pesquisar Terminal Ajuda
[lenep@localhost CodigosFinais]$ ./a.out
-----Gerao de Trao Ssmico-----

Autores: Ramon Diogo e Svio Januario
Disciplina: Programao Prtica em C++

-----
Voc deseja carregar os dados manualmente ou ler do arquivo?
(1)Carregar Manualmente
(2)Ler do arquivo
2
-----
Qual o nome do arquivo com as propriedades do meio acstico?
entradataeste.txt
```

Figura 7.3: O arquivo entradataeste.txt

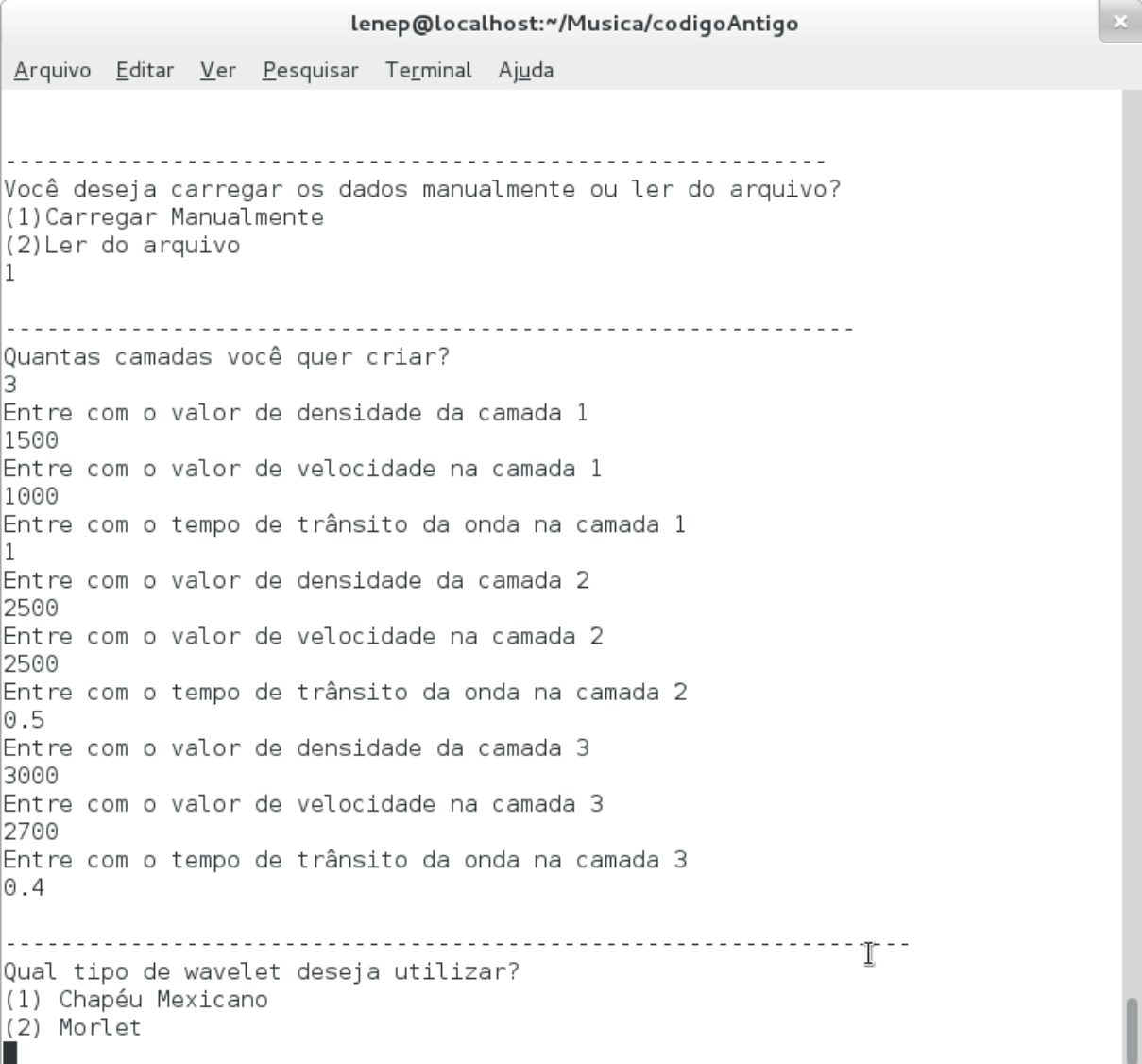


```
lenep@localhost:~/MÃsica/CodigosFinais
Arquivo Editar Ver Pesquisar Terminal Ajuda
[lenep@localhost CodigosFinais]$ ./a.out
-----Geração de Traço Sísmico-----

Autores: Ramon Diogo e Sávio Januario
Disciplina: Programação Prática em C++

-----
Você deseja carregar os dados manualmente ou ler do arquivo?
(1)Carregar Manualmente
(2)Ler do arquivo
1
```

Figura 7.4: Escolher o tipo de entrada de dados: Entrar manualmente



```
lenep@localhost:~/Musica/codigoAntigo
Arquivo Editar Ver Pesquisar Terminal Ajuda

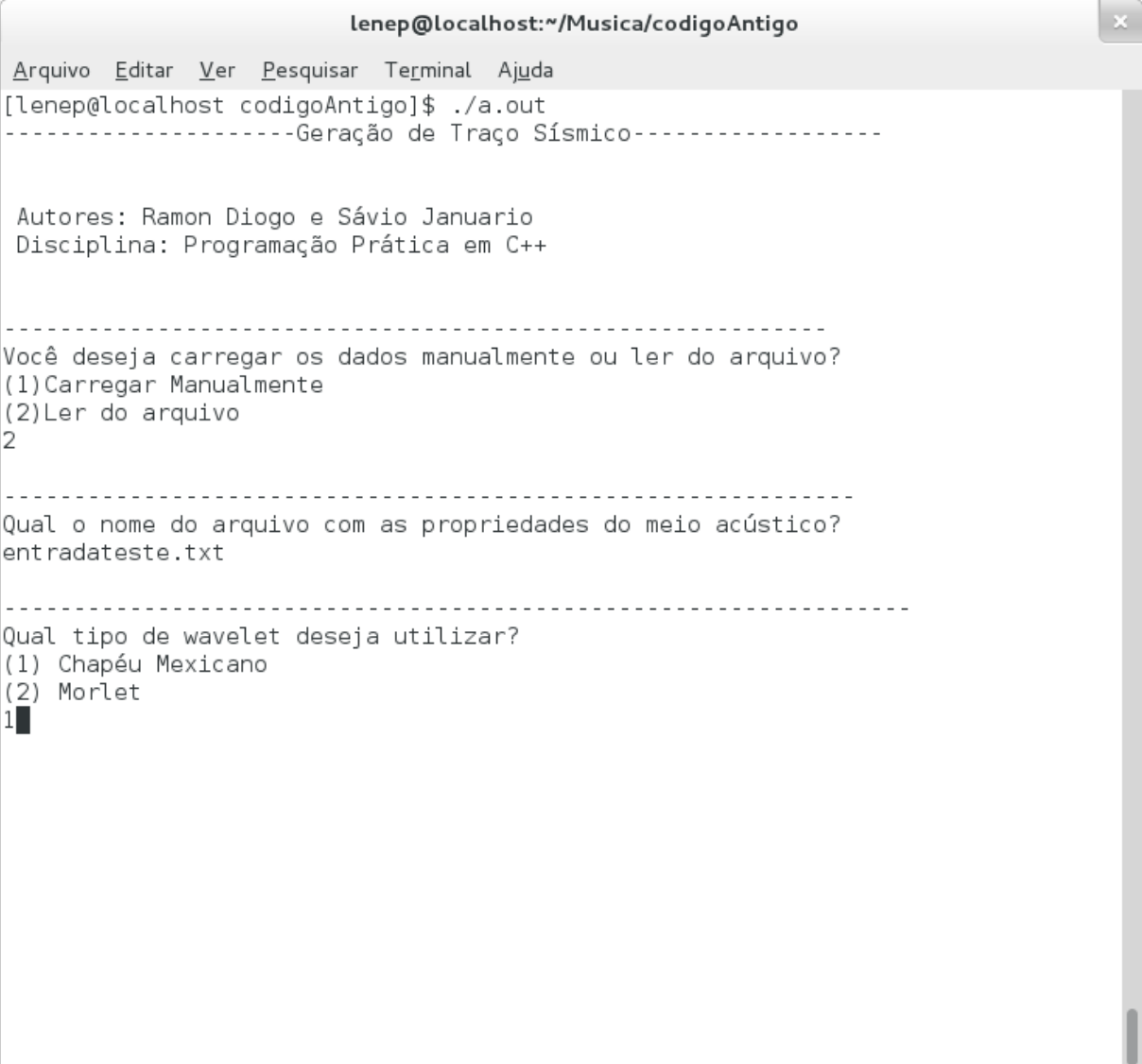
-----
Voc   deseja carregar os dados manualmente ou ler do arquivo?
(1) Carregar Manualmente
(2) Ler do arquivo
1

-----
Quantas camadas voc   quer criar?
3
Entre com o valor de densidade da camada 1
1500
Entre com o valor de velocidade na camada 1
1000
Entre com o tempo de tr  nsito da onda na camada 1
1
Entre com o valor de densidade da camada 2
2500
Entre com o valor de velocidade na camada 2
2500
Entre com o tempo de tr  nsito da onda na camada 2
0.5
Entre com o valor de densidade da camada 3
3000
Entre com o valor de velocidade na camada 3
2700
Entre com o tempo de tr  nsito da onda na camada 3
0.4

-----
Qual tipo de wavelet deseja utilizar?
(1) Chap  u Mexicano
(2) Morlet
█
```

Figura 7.5: Escolher valores de densidade, velocidade e duplo tempo para cada camada





```
lenep@localhost:~/Musica/codigoAntigo
Arquivo Editar Ver Pesquisar Terminal Ajuda
[lenep@localhost codigoAntigo]$ ./a.out
-----Geração de Traço Sísmico-----

Autores: Ramon Diogo e Sávio Januario
Disciplina: Programação Prática em C++

-----
Você deseja carregar os dados manualmente ou ler do arquivo?
(1)Carregar Manualmente
(2)Ler do arquivo
2
-----
Qual o nome do arquivo com as propriedades do meio acústico?
entradataeste.txt
-----
Qual tipo de wavelet deseja utilizar?
(1) Chapéu Mexicano
(2) Morlet
1
```

Figura 7.6: Escolher o tipo de Wavelet Chapéu Mexicano

```
lenep@localhost:~/Musica/codigoAntigo
Arquivo Editar Ver Pesquisar Terminal Ajuda
[lenep@localhost codigoAntigo]$ ./a.out
-----Geração de Traço Sísmico-----

Autores: Ramon Diogo e Sávio Januario
Disciplina: Programação Prática em C++

-----
Você deseja carregar os dados manualmente ou ler do arquivo?
(1)Carregar Manualmente
(2)Ler do arquivo
2
-----
Qual o nome do arquivo com as propriedades do meio acústico?
entradataeste.txt
-----
Qual tipo de wavelet deseja utilizar?
(1) Chapéu Mexicano
(2) Morlet
1
Qual o número de amostras da wavelet?
60
Qual o intervalo em segundos de amostragem da wavelet?
0.004
Qual a frequência em Hertz de pico da wavelet?
6
```

Figura 7.7: Informar o número de amostra, intervalo de amostragem da wavelet e frequência

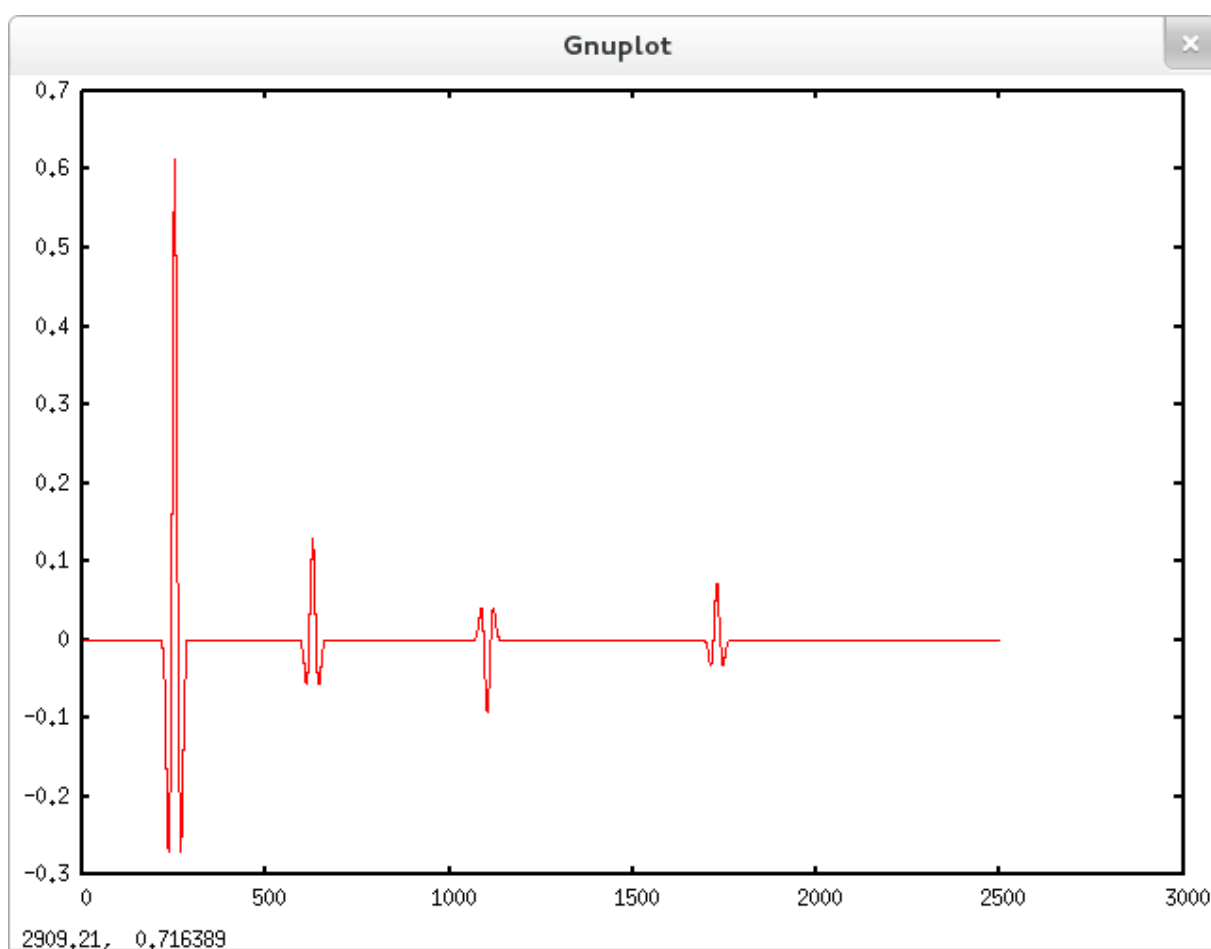
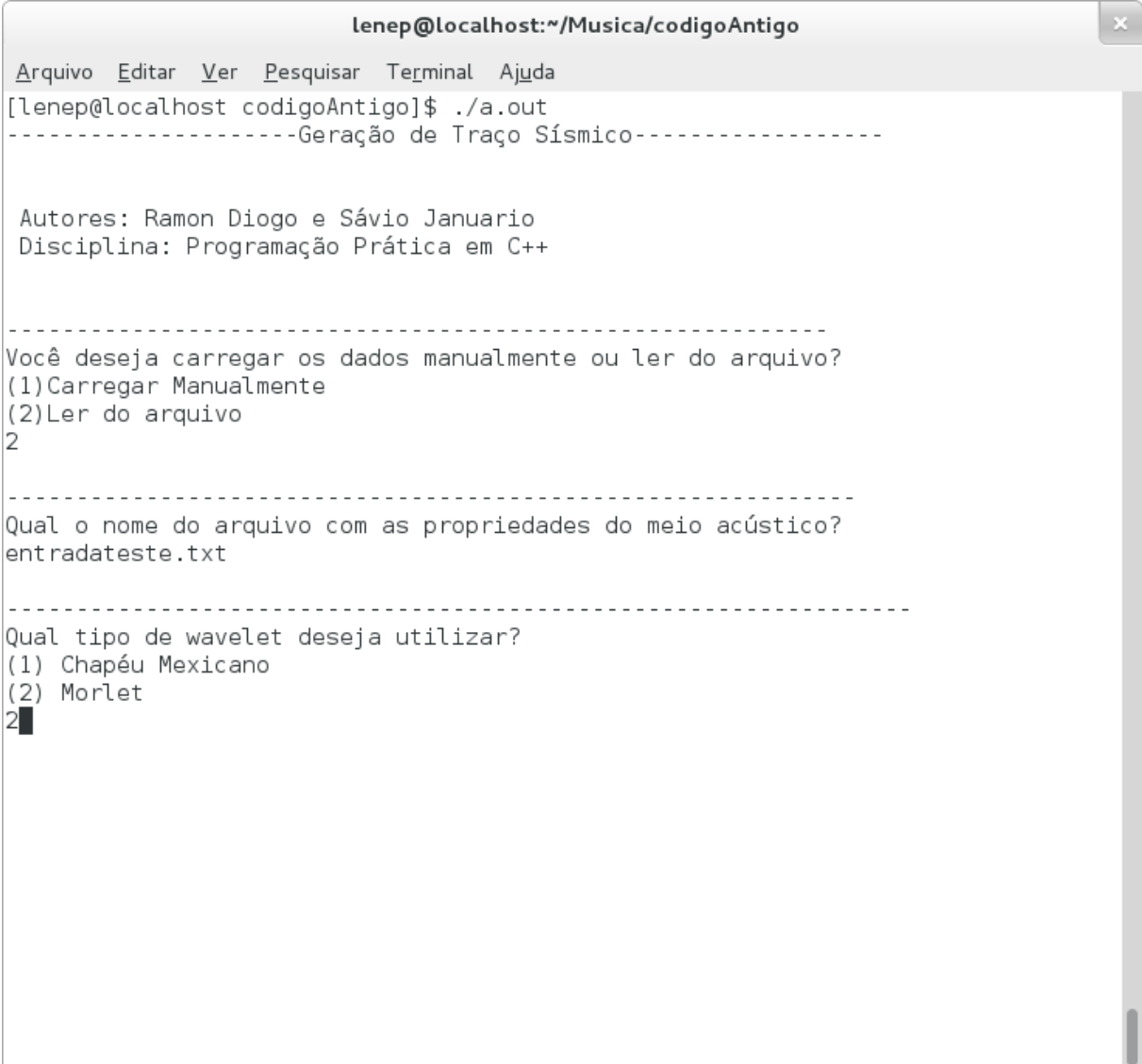


Figura 7.8: Sáida do gráfico em arquivo de imagem



```
lenep@localhost:~/Musica/codigoAntigo
Arquivo Editar Ver Pesquisar Terminal Ajuda
[lenep@localhost codigoAntigo]$ ./a.out
-----Geração de Traço Sísmico-----

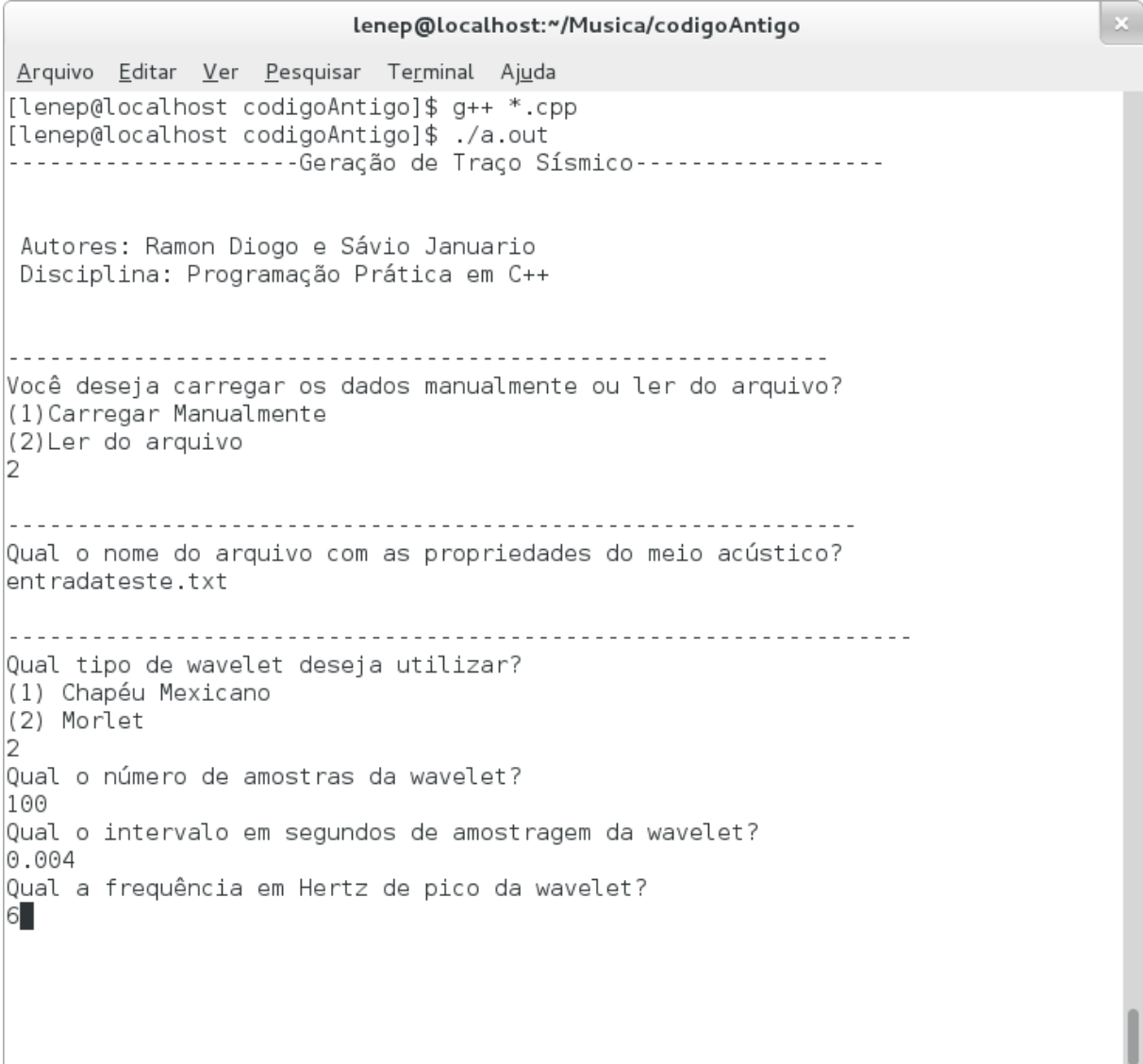
Autores: Ramon Diogo e Sávio Januario
Disciplina: Programação Prática em C++

-----
Você deseja carregar os dados manualmente ou ler do arquivo?
(1)Carregar Manualmente
(2)Ler do arquivo
2

-----
Qual o nome do arquivo com as propriedades do meio acústico?
entradateste.txt

-----
Qual tipo de wavelet deseja utilizar?
(1) Chapéu Mexicano
(2) Morlet
2
```

Figura 7.9: Escolher o tipo de wavelet Morlet



```
lenep@localhost:~/Musica/codigoAntigo
Arquivo Editar Ver Pesquisar Terminal Ajuda
[lenep@localhost codigoAntigo]$ g++ *.cpp
[lenep@localhost codigoAntigo]$ ./a.out
-----Geração de Traço Sísmico-----

Autores: Ramon Diogo e Sávio Januario
Disciplina: Programação Prática em C++

-----
Você deseja carregar os dados manualmente ou ler do arquivo?
(1)Carregar Manualmente
(2)Ler do arquivo
2
-----
Qual o nome do arquivo com as propriedades do meio acústico?
entradataeste.txt
-----
Qual tipo de wavelet deseja utilizar?
(1) Chapéu Mexicano
(2) Morlet
2
Qual o número de amostras da wavelet?
100
Qual o intervalo em segundos de amostragem da wavelet?
0.004
Qual a frequência em Hertz de pico da wavelet?
6
```

Figura 7.10: Informar o número de amostra, intervalo de amostragem da wavelet e frequência

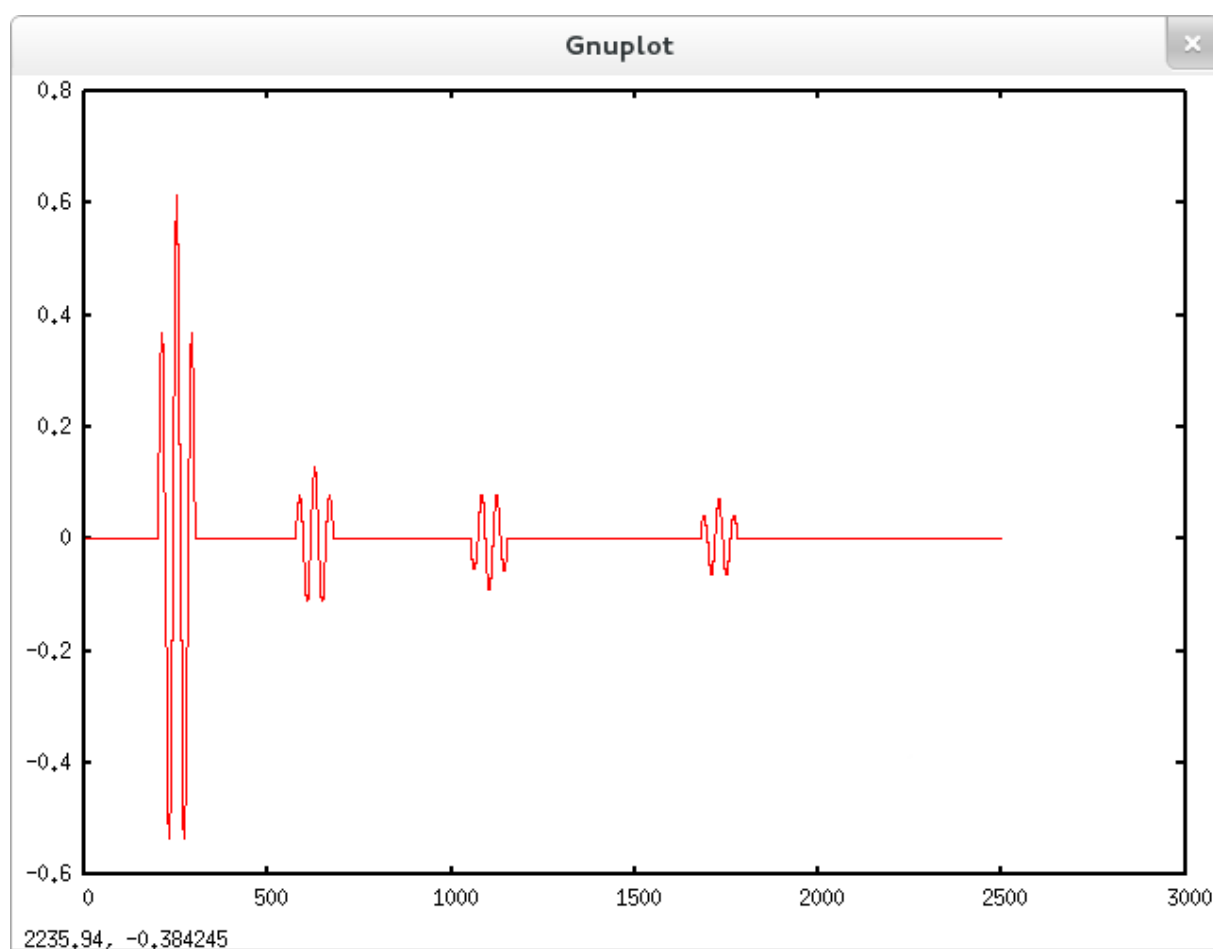


Figura 7.11: Sáida do gráfico em arquivo de imagem

# Capítulo 8

## Documentação

A presente documentação refere-se ao uso do Simulador de Traço Sísmico. Esta documentação tem o formato de uma apostila que explica passo a passo como usar o software.

### 8.1 Documentação do usuário

Descreve-se aqui o manual do usuário, um guia que explica, passo a passo a forma de instalação e uso do software desenvolvido.

#### 8.1.1 Como rodar o software

Segue abaixo o passo a passo para rodar o software:

Abrir o terminal, vá para o diretório onde está o código, compile o programa e depois execute. Logo após executar, seguir os seguintes passos:

1. Escolher o tipo de wavelet:

- (a) Para utilizar CM, escolha 1;
- (b) Para utilizar Morlet, escolha 2;

2. Informar o número de amostra da *wavelet*:

Usualmente é utilizado 60 para Chápeu Mexicano.

Usualmente é utilizado 100 para Morlet.

3. Informar o intervalo de amostragem da *wavelet*:

Usualmente é utilizado 0.004 segundos.

4. Informar o pico de frequência da *wavelet*:

Usualmente é utilizado 6 Hz.

5. Informar o nome do arquivo de entrada.

6. Informar o nome de saída.

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

## 8.2 Documentação para desenvolvedor

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

### 8.2.1 Dependências

Para compilar o software é necessário atender as seguintes dependências:

- No sistema operacional GNU/Linux::
  - Instalar o compilador `g++` da GNU disponível em <http://gcc.gnu.org>.
  - Para instalar no GNU/Linux use o comando `yum install gcc`.
- No sistema operacional Windows:
  - Instalar um compilador apropriado.
  - Recomenda-se o Dev C++ disponível em <http://dev-c.softonic.com.br/>.
- O software `gnuplot`, deve estar instalado.
  - Gnuplot está disponível no endereço <http://www.gnuplot.info/>
  - É possível que haja necessidade de setar o caminho para execução do `gnuplot`.
- O programa depende da existência de um arquivo de dados (formato `.txt`) para preencher os vetores relacionados aos parâmetros da camada (densidade, velocidade e tempo).

### 8.2.2 Documentação usando doxygen

A documentação do código do software foi feita usando o padrão JAVADOC. Depois de documentar o código, o software *doxygen* foi usado para gerar a documentação do desenvolvedor no formato *html*. O software *doxygen* lê os arquivos com os códigos (\*.h e \*.cpp) e gera uma documentação muito útil e de fácil navegação no formato *html*. Apresenta-se a seguir a imagem de hierarquia de classe mostrada na tela de saída gerada pelo software *doxygen*:



**STSS**  
Simulador de Traço Sísmico Sintético

Página Principal **Classes** Arquivos

Lista de Componentes Índice dos Componentes **Hierarquia de Classes** Componentes Membros

### Hierarquia de Classes

Esta lista de hierarquias está parcialmente ordenada (ordem alfabética):

[nível de detalhes 1 2]

▼ C CCamada	
C CCamadaAcustica	Herança
C CMeioAcustico	Esta classe representa um conjunto de camadas ordenadas em um vetor onde sua principal função é o calculo do vetor Refletividade
C CTracoSismico	Esta classe executa a convolução entre o veotr de refletividades e a função wavelet escolhida
▼ C CWavelet	
C CWaveletCM	Esta classe herda os atributos e métodos da classe CWavelet e inicializa os atributos por um construtor sobrecarregado
C CWaveletMorlet	Esta classe herda os atributos e métodos da classe CWavelet e inicializa os atributos por um construtor sobrecarregado
C Gnuplot	Classe de interface para acesso ao programa gnuplot

Gerado em Quinta, 2 de Abril de 2015 14:19:04 para STSS por [doxygen](#) 1.8.9.1

Figura 8.1: Hierarquia de classe



# Referências Bibliográficas

- [Bracewell, 2000] Bracewell, R. (2000). *The Fourier Transform And Its Applications*. McGraw-Hill, New York, 3rd edition.
- [Bueno, 2003] Bueno, A. D. (2003). *Programa de Orientada a Objeto com C++*. Novatec, São Paulo.
- [LATIMER RB, 2000] LATIMER RB, D. R. . R. (2000). *An interpreter's guide to understanding and working with seismic-derived acoustic impedance data*. The Leading Edge.

# Índice Remissivo

Análise orientada a objeto, 13

AOO, 13

Associações, 24

Assuntos, 11

atributos, 22

Casos de uso, 4

colaboração, 16

comunicação, 16

Controle, 20

Diagrama de colaboração, 16

Diagrama de componentes, 24

Diagrama de execução, 24

Diagrama de máquina de estado, 17

Diagrama de sequência, 15

Efeitos do projeto nas associações, 24

Efeitos do projeto nas heranças, 23

Efeitos do projeto nos métodos, 23

Elaboração, 6

estado, 17

Eventos, 15

Heranças, 23

heranças, 23

Identificação de pacotes, 11

Implementação, 27

métodos, 23

Mensagens, 15

modelo, 22

Plataformas, 21

POO, 22

Projeto do sistema, 20

Projeto orientado a objeto, 22

Protocolos, 20

requisitos, 3