UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA MECÂNICA
INSTITUTO DE GEOCIÊNCIAS

# LabSis: Um ambiente para desenvolvimento de aplicações sísmicas em Matlab.

*Autor:* CRISTIANO DA SILVA MARCOLINO
*Orientador:* PROF. DR. MARTIN TYGEL
*Co-orientador:* PROF. DR. RODRIGO PORTUGAL

10/04

UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA MECÂNICA
INSTITUTO DE GEOCIÊNCIAS

# LabSis: Um ambiente para desenvolvimento de aplicações sísmicas em Matlab.

*Autor:* CRISTIANO DA SILVA MARCOLINO
*Orientador:* PROF. DR. MARTIN TYGEL
*Co-orientador:* PROF. DR. RODRIGO PORTUGAL

Curso: Ciências e Engenharia de Petróleo

Dissertação de mestrado apresentada à Subcomissão de Pós-Graduação Interdisciplinar de Ciências e Engenharia de Petróleo (FEM e IG), como requisito para a obtenção do título de Mestre em Ciências e Engenharia de Petróleo.

Campinas, 2004
SP - Brasil

## FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DA ÁREA DE ENGENHERIA - BAE - UNICAMP

Marcolino Cristiano

M333L LabSis: Um ambiente para desenvolvimento de aplicações sísmicas em Matlab. / Cristiano da Silva Marcolino – Campinas, SP:[ s.n.], 2004.

Orientadores: Martin Tygel; Rodrigo Portugal.

Dissertação (mestrado) – Universidade Estadual de Campinas, Faculdade de Engenharia Mecânica e Instituto de Geociências.

1. Geofísica. 2. Matlab (Programa de computador). 3. Método sísmico de reflexão. I. Tygel, Martin. II. Portugal, Rodrigo. III. Universidade Estadual de Campinas, Faculdade de Engenharia Mecânica. IV. Instituto de Geociencias. V. Título.

UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA MECÂNICA
INSTITUTO DE GEOCIÊNCIAS

DISSERTAÇÃO DE MESTRADO

# LabSis: Um ambiente para desenvolvimento de aplicações sísmicas em Matlab.

*Autor:* CRISTIANO DA SILVA MARCOLINO
*Orientador:* PROF. DR. MARTIN TYGEL

Banca Examinadora:

Prof. Dr. Martin Tygel, Presidente
UNICAMP / IMECC

Profa. Dra. Maria Cristina Cunha
INICAMP / IMECC

Prof. Dr. Jessé Costa
UFPa / CG



Campinas, 18 de Outubro de 2004

Aos meus Pais e a minha amada Celina.

# Agradecimentos

Ao meu orientador Prof. Dr. Martin Tygel, pela oportunidade de desenvolver este projeto, bem como pela ajuda e dedicação na elaboração deste trabalho.

Ao meu co-orientador Prof. Dr. Rodrigo Portugal, pela ajuda na elaboração do software e pelas sugestões e correções nesta dissertação.

Ao Cepetro, pelo auxílio fornecido durante este curso.

Aos meus pais, por me ajudarem nos momentos em que eu necessitei.

À Schlumberger, por permitir minhas viagens e ausências necessárias para o término deste trabalho.

À minha esposa Celina, que me apoiou todo o tempo, mesmo nos momentos em que não pude estar ao seu lado por estar me dedicando a este trabalho, qualquer forma agradecimento feito aqui seria injusta, porque seria com certeza incompleta.

# Resumo

MARCOLINO, Cristiano da Silva. LabSis Um Ambiente para Desenvolvimento de Aplicações Sísmicas em Matlab.

Campinas Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas, 2004. 120p. Dissertação (Mestrado)

O pacote computacional Matlab é uma ferramenta de uso generalizado no meio acadêmico pelas suas vantagens de programação simples e direta e uso fácil de gráficos e visualizações, permitindo rapidamente implementações iniciais de algoritmos e procedimentos em uma série de aplicações. Em contrapartida às facilidades operacionais, os programas Matlab não possuem a eficiência computacional exigidas das linguagens de programação propriamente ditas (tais como Fortran e C, por exemplo). Tais propriedades fazem com que o Matlab seja, por excelência, um pacote de obtenção de "primeiras versões", dedicadas a testes em "problemas pequenos". Numa segunda etapa, os programas Matlab devem ser submetidos aos procedimentos de praxe da engenharia de software, incluindo a mudança de linguagem de programação para uso final em problemas práticos. Tal característica explica porque o Matlab seja tão utilizado na academia, em particular no ensino e elaboração de dissertações e teses.

No caso específico do Laboratório de Geofísica Computacional da Unicamp, uma variedade de programas Matlab foi desenvolvida, visando aplicações ao ensino e a pesquisa de métodos de processamento de dados geofísicos, com ênfase aos métodos sísmicos. Devido aos focos específicos e sem muita conexão entre si, os programas foram desenvolvidos sem uma unidade de concepção, resultando na dificuldade de sinergia e utilização dos programas por um público mais amplo ou mesmo por outros alunos e usuários do próprio Laboratório.

O LabSis, desenvolvido nesta dissertação, surge como um pacote integrador destas funções, utilizando as ferramentas gráficas do Matlab para criar uma interface simplificada e intuitiva ao usuário. O LabSis é formado por "funções casca", as quais fazem a ponte entre os algoritmos originais e o usuário. O uso destas funções casca libera o programador da tarefa de alterar as funções externas que compõem o LabSis, mantendo assim a filosofia dos autores dos programas originais. O fato de ter sido escrito totalmente em Matlab, torna o Labsis um software de código aberto, permitindo a qualquer programador a introdução de novas funções e programas. Uma vez que não é compilado em nenhum sistema específico, podendo assim ser executado em qualquer sistema onde o Matlab esteja instalado, torna o LabSis um software multi-plataforma.

Construído para ser um pacote que englobe funções presentes e futuras, o LabSis contém, em sua versão atual, algoritmos (simples) de modelagem por traçado de raios, aproximação de Born e integral Kirchhoff, análise de velocidades NMO, transformada $\tau - p$ e análise de variação de amplitude com afastamento (AVO), migração Kirchhoff em profundidade e demigração Kirchhoff. O dado pode ser a qualquer momento visualizado através de uma ferramenta de "plotagem" de dados sísmicos. O programa permite ao usuário trabalhar com vários dados sísmicos ao mesmo tempo, sendo possível alternar entre eles a qualquer momento. O programa é totalmente gráfico, liberando o usuário de recorrer à linha de comando. No entanto essa opção existe, sendo útil para o caso de sucessivas repetições com ligeira variação de parâmetros.

O LabSis é integrado com o pacote InterSis, um software também desenvolvido no Laboratório de Geofísica Computacional da Unicamp, e que consiste de uma interface gráfica para programas de modelagem de dados sísmicos. Com auxílio do InterSis, é possível gerar um modelo geológico e exportá-lo para o LabSis onde o mesmo é utilizado nas suas várias funções. Uma outra possibilidade é a utilização do InterSis para a modelagem de dados sísmicos e transferi-los para o LabSis para tarefas de processamento ou imageamento. A importação de dados no formato Seismic Unix (SU), bem conhecido na comunidade geofísica acadêmica e profissional, faz com que o LabSis possa se comunicar sem dificuldades com o mundo externo, permitindo a utilização de dados gerados por outros softwares.

LabSis é um software didático, desenvolvido primordialmente para o ensino e a pesquisa, com o objetivo de tornar possível o entendimento e a verificação, na prática, de conceitos teóricos expostos em sala de aula. Tais características fazem com que o LabSis seja um atraente pacote para ser utilizado em cursos de graduação e pós-graduação. Por ser um programa leve, o LabSis não requer grandes exigências de máquina (a não ser que o dado utilizado assim o exija). Finalmente, o caráter integrador do LabSis permite sua utilização como plataforma unificada paras várias aplicações, em particular na área de modelagem e imageamento de dados sísmicos.

# Abstract

MARCOLINO, Cristiano da Silva. LabSis An Environment for Development of Seismic Applications in Matlab.

Campinas Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas, 2004. 120p. Master Thesis

The Matlab package is a tool of widespread use in the academic environment, because of its advantages in simple direct programming, graphs and visualization tools. It allows initial implementations of algorithms and procedures very quickly in a series of applications. As a counterpar to the above good qualities, Matlab programs do not exhibit the computational efficiency that is found in typical programming languages (such as Fortran and C), as required for "final production codes". Such properties make the Matlab a package for "prototype codes". On a later stage, Matlab programs can be submitted to standard software engineering procedures, that contemplate a more adequate programming language for final use in practical problems. This characteristic of Matlab illustrates why it is so widely used in academia, especially for teaching and research purposes.

In the specific case of the Laboratory of Computational Geophysics at Unicamp, a variety of Matlab programs have been developed in the last few years, mainly in the area of seismic data processing. Due to their very specific focus and lack of a common interface, the programs did not benefit from any conceptual unity that would allow more widespread application, even for users of the Laboratory.

LabSis, developed in this thesis, appears as an integrator package of these functions, using

the graphic tools of Matlab to create a simplified and intuitive interface for the user. LabSis is composed as a series of "wrapper functions", which make the bridge between the original algorithms and the final user. The employment of these wrapper functions frees the programmer from the task of altering the external functions that compose LabSis, maintaining the author's original program philosophy. The fact of being totally written in Matlab turns LabSis software an open source application, allowing any user to introduce new functions and programs. Since LabSis it is not compiled in any specific system (namely, it can be executed on any system where Matlab is installed), it also a multi-platform software.

Built to be a package to include present and future functions, LabSis contains, in the current version, programs designed for modelling (using ray tracing, Born and Kirchhoff methods), NMO velocity analysis, computation of $\tau - p$ transforms, amplitude versus offset (AVO) analysis, Kirchhoff true-amplitude migration and demigration .

Visualization of results is always available by means of a tool that plots seismic data. The program allows the user to work simultaneously with several data sets, switching between them at any moment. The program is a graphical user interface (GUI) application. The user does not need to use command lines, however, that option exists, being useful for the case of successive repetitions with small variation of parameters.

LabSis is integrated with the InterSis package, a software also developed at the Laboratory of Computational Geophysics of Unicamp, that consists of a graphic interface for seismic data modelling programs. With the aid of InterSis, it is possible to generate a geological model and export it to LabSis. Another possibility is to use InterSis to produce synthetic seismograms and transfer the datasets to LabSis for processing or imaging tasks. The possibility to import data in the Seismic Unix (SU) format, enables LabSis to communicate with the external world, allowing the use of data generated by other softwares.

LabSis is a didactic software, specifically developed for teaching and research, with the aim of verifying in practice, many theoretical concepts exposed in the classroom. Such characteristics

make LabSis attractive to Undergraduate and Graduate Programs that have geophysical data processing among their topics of interest. LabSis has not heavy requirements of computational speed or memory, unless the volume of data used demands it. LabSis integrated structure, makes possible its use as a small development an communication platform to a wide range of users.

# Conteúdo

# Lista de Figuras

# Nomenclatura

## Letras Latinas

| | |
|---|---|
| $c$ | Velocidade da onda no meio acima do refletor |
| $c_+$ | Velocidade de propagação abaixo do refletor |
| $D$ | Dado sísmico |
| $\mathbf{e}$ | Ponto espalhador localizado no refletor |
| $F$ | Assinatura da fonte |
| $h$ | Afastamento médio entre a fonte e o receptor. |
| $ID$ | Dado demigrado |
| $IM$ | Seção migrada |
| $\hat{\mathbf{n}}$ | Vetor normal |
| $R$ | Coeficiente de reflexão óptico geométrico |
| $u$ | Campo de onda espalhado monocromático |
| $V$ | Volume do espalhamento |
| $v_{NMO}$ | Velocidade NMO. |
| $v_{RMS}$ | Velocidade RMS. |
| $v_0$ | Velocidade da camada superficial. |
| $W_M$ | Função peso de amplitude verdadeira |
| $\mathbf{x}_G$ | Posição do receptor. |
| $\mathbf{x}_S$ | Posição da fonte. |

## Superescritos

$^0$ Ponto inicial.

$^*$ Ponto ótimo.

$^T$ Matriz transposta.

$\beta$ Ângulo de emergência do raio normal.

## Letras Gregas

$\alpha$ Pertubação no campo vagarosidade

$\beta$ Ângulo de emergência do raio normal.

$\sum_R$ Superfície refletiva

$\phi$ Fase

$\tau$ Tempo de transito

$v$ Velocidade de propagação média

$\omega$ Freqüência

## Siglas

ANP Agência Nacional de Petróleo.

CMP Ponto médio comum (*commom midpoint*).

CO Afastamento comum (*commom offset*).

CR Receptor comum (*commom receiver*).

CS Tiro comum (*commom shot*).

LGC Laboratório de Geofísica Computacional

NIP *Normal Incident Point.*

NMO *Normal MoveOut.*

RMS *Root Mean Square.*

ZO Afastamento nulo (*Zero Offset*).

# Capítulo 1

# Introdução

As atividades de pesquisa e desenvolvimento nas mais variadas áreas de atuação demandam de forma cada vez mais acentuada o apoio de pacotes computacionais capazes de executar um grande fluxo de tarefas e integrados por interfaces gráficas da melhor qualidade. Estas características possibilitam o melhor gerenciamento e utilização da vasta gama de informações e metodologias disponíveis nos diversos meios. Este é, sem dúvida, o caso da área de processamento e imageamento de dados geofísicos, em particular da sísmica voltada para a exploração e monitoramento de reservatórios de petróleo.

Um dos pacotes computacionais capazes de atender às exigências acima é o MATLAB. Este pacote ganhou vasta aceitação no meio acadêmico por possibilitar de maneira eficiente, integrada e, sobretudo, gráfica, o desenvolvimento dos mais variados algoritmos com o demandado, por exemplo, em simulações computacionais para modelagem e construção de sismogramas sintéticos, métodos de imageamento e inversão de atributos a partir de sismogramas, etc.

## 1.1 MATLAB

O pacote computacional MATLAB é uma ferramenta de uso generalizado no meio acadêmico pelas suas vantagens de programação simples e direta e uso fácil de gráficos e visualizações. Isto

permite que algoritmos e procedimentos em uma série de aplicações sejam rapidamente implementados e testados. Embora os programas MATLAB não possuam a eficiência computacional exigida das linguagens típicas de programação (por exemplo, Fortran e C), as boas propriedades acima fazem do MATLAB uma ótima opção para a obtenção de protótipos, dedicadas a testes em aplicações de pequeno porte. Numa segunda etapa, os programas MATLAB devem ser submetidos aos procedimentos de praxe da engenharia de software, incluindo a mudança de linguagem de programação para uso final em problemas práticos. Tal característica do MATLAB explica porque o MATLAB seja tão utilizado na academia, em particular no ensino e elaboração de dissertações e teses.

O MATLAB não estava disponível até o final dos anos 80, e até então, Fortran era a linguagem escolhida para computação científica. Ainda que C também fosse uma possibilidade, a sua falta de flexibilidade para se trabalhar com números complexos era uma desvantagem considerável. Por outro lado, o Fortran não tinha algumas das vantagens da linguagem C, como estruturas, apontadores e alocação dinâmica de memória.

O MATLAB evoluiu do pacote LINPACK, conhecido pelos programadores Fortran como uma robusta coleção de ferramentas para álgebra linear (Margrave (2001)). Entretanto o MATLAB também introduziu uma linguagem de programação nova, orientada por matrizes e um ambiente interativo que utiliza objetos gráficos. Essas características ofereceram vantagens suficientes para os usuários, os quais observaram um aumento significativo na sua produtividade sobre os ambientes mais tradicionais. Desde então, o MATLAB tem originado um grande número de ferramentas, tanto comerciais quanto livres, excelentes gráficos 2D e 3D, extensões orientadas a objetos e, tudo isso com uma interface interativa.

É claro que as linguagens C e Fortran também evoluíram. A linguagem C avançou para C++, o Fortran evoluiu para Fortran 90. O diferencial do MATLAB, em contraposição ao Fortran e C, foi a sua concepção como um pacote completo. Por exemplo, a inclusão no MATLAB de uma ferramenta gráfica utilizando a própria linguagem, é o principal benefício. Isso significa que

programas MATLAB fazendo uso de gráficos se tornaram um padrão para uma ampla gama de usuários. Neste contexto, vale observar a habilidade do MATLAB em mostrar de forma gráfica conjuntos de dados através de um *breakpoint* no compilador. Essas são vantagens práticas que podem ser muito importantes quando a visualização se torna um atributo essencial ao desenvolvimento.

A sintaxe vetorial do MATLAB conduz a um código mais conciso do que a maioria das outras linguagens. Por exemplo, para igualar uma matriz à transposta de outra basta usar o comando `A=B'`, o que é muito mais fácil que fazer em outras linguagens. Por exemplo, em FORTRAN77, a mesma operação é especificada pelas instruções:

```
do i=1,n
    Do j=1,m
        A(i,j)=B(j,i)
    endo
endo
```

É também freqüentemente argumentado que C e Fortran são mais eficientes que MATLAB e, portanto mais ajustáveis a intensivas tarefas computacionais (Margrave (2001)). Isto é verdade e pode representar uma real desvantagem na utilização do MATLAB. Porém, em muitos casos incluindo especialmente as aplicações acadêmicas, o que realmente importa é a eficiência de todo o processo científico desde a origem da idéia, passando pela implementação bruta e testes, chegando à uma forma adequada, pelo menos a problemas pequenos (protótipos). Neste sentido, o MATLAB é muito mais eficiente para o processo como um todo, uma vez que os ambientes gráficos interativos, um grande conjunto de ferramentas e um rigoroso sistema de teste de erros levam a um rápido sistema de construção de protótipos para grandes algoritmos.

Linguagens tradicionais como C e Fortran foram originadas em uma época em que os computadores possuíam o tamanho de salas e os recursos bem mais limitados, o que sempre exigiu bastante do programador humano. Sua sintaxe específica levou à eficiência de locações memória

para melhorar a velocidade computacional, o que era essencial naquela época. Entretanto, os computadores evoluíram, sendo hoje bem mais poderosos e baratos. Desta forma, pode fazer mais sentido deixar tarefas mais complexas para o computador e liberar o programador humano para trabalhar em um nível mais elevado. O MATLAB é uma linguagem de alto nível, que libera o programador de uma série de procedimentos de caráter técnico e rotineiro, para que o mesmo possa se concentrar em resolver o problema real.

## 1.2 LabSis

No Laboratório de Geofísica Computacional de Unicamp, como subproduto dos vários trabalhos realizados, uma variedade de programas MATLAB foi desenvolvida na área de métodos de processamento de dados sísmicos. Entretanto, devido aos focos específicos, os programas foram gerados sem uma unidade de concepção, resultando na dificuldade de sinergia e utilização dos programas por um público mais amplo ou mesmo por outros alunos.

Tal situação motivou a concepção do pacote LabSis, escrito em MATLAB, como sendo uma plataforma integradora, visando o mais amplo aproveitamento dos vários programas. Essas considerações justificam porque o LabSis foi escrito em MATLAB e não numa linguagem mais tradicional como Fortran ou C, ou mais moderna como C++ ou Java. Vale salientar que no LAbSis, as funções já existentes em MATLAB não foram reescritas ou sequer alteradas, com vistas a manter as filosofias específicas dos programas.

Nesta dissertação, desenvolvemos o pacote LabSis, destinado a unificar uma série de programas existentes no Laboratório de Geofísica Computacional da Unicamp, bem como suportar a introdução de novos programas dentro da mesma concepção unificadora. Com isto, esperamos contribuir para que os programas já existentes e futuros sejam utilizados de forma mais fácil e melhor por um público mais amplo acadêmico e profissional.

O LabSis é um software constituído de vários módulos que englobam diferentes passos do processamento sísmico. O objetivo deste capítulo é introduzir os principais conceitos teóricos

abordados pelo software.

Nos capítulos que se seguem, procuraremos descrever a concepção e os diversos módulos que compõem o LabSis, ilustrando a exposição com vários exemplos. Esta dissertação é composta de uma introdução e sete capítulos, além de um Apêndice de caráter mais técnico.

No Capítulo 2, descrevemos de maneira sucinta os mais importantes passos da chamada seqüência de processamento de dados sísmicos, enfatizando os passos para os quais o pacote LabSis, objeto deste trabalho, já possui um algoritmo incorporado.

No Capítulo 3 descrevemos a motivação, filosofia, fatores que levaram a construção deste software e porque ele não foi construído anteriormente. Também são apresentados os desafios encontrados para a construção do pacote.

No Capítulo 4 apresentamos o pacote InterSis, também desenvolvido no Laboratório de Geofísica Computacional da Unicamp, e que atua como interface gráfica para a introdução de modelos geológicos, execução de programas de modelagem e visualização de resultados. Também comentamos sobre sua integração com o LabSis.

No Capítulo 5, um dos mais importantes desta dissertação, descrevemos detalhadamente os aspectos computacionais envolvidos no LabSis, suas principais características, incluindo limitações e dificuldades, mostrando, sob o ponto de vista interno, o caminho que foi seguido para confecção do software. E explicamos também, em uma importante seção, como introduzir uma nova função ou programa no pacote.

No Capítulo 6 são descritos os módulos existentes atualmente no LabSis, com ênfase à sua utilização, servindo de manual para usuários iniciantes.

O Capítulo 7 é dedicado à apresentação de uma série de exemplos ilustrativos de aplicação

do pacote, ressaltando sua integração com o InterSis .

O Capítulo 8 apresenta as conclusões da dissertação com alguma propostas para desenvolvimentos futuros.

Finalmente, um Apêndice lista os nomes dos diretórios e pastas criados no LabSis, com os respectivos autores de cada função.

# Capítulo 2

# O método sísmico de reflexão

O método sísmico de reflexão, ou simplesmente, a sísmica de reflexão, consiste, basicamente, em obter informações da subsuperfície através da emissão de ondas produzidas por fontes artificiais em superfície e posterior registro dessas ondas em receptores, também em superfície. As ondas emitidas pelas fontes propagam-se em subsuperfície, sendo transmitidas e refletidas por interfaces que separam camadas de rochas de diferentes propriedades geológicas. As ondas observadas nos receptores são provenientes de reflexões nessas interfaces.

A perturbação do meio para geração de ondas é realizada pelas fontes sísmicas, que podem ser terrestres e marítimas. Para o uso em terra, utilizam-se explosivos ou vibradores mecânicos junto ao solo. Para o uso em mar, utilizam-se os chamados canhões de ar ("air-guns").

Após uma onda ser gerada pela fonte, ela se propaga pelo meio onde encontra camadas de diferentes propriedades (litologias) originando, entre outros efeitos, reflexões e transmissões. As interfaces que separam camadas de rochas de diferentes propriedades (essencialmente densidade e velocidades de propagação), dão origem à reflexão de parte da energia sísmica, a qual retorna à superfície e é registrada nos receptores espalhados em superfície para este fim.

No método de reflexão sísmica, os levantamentos sísmicos consistem de sucessivos "tiros" registrados por diferentes conjuntos de receptores. Os dados sísmicos consistem dos registros dos

receptores, guardadas as posições dos tiros e do conjunto de receptores associados ao tiro. O processo de levantamento de dados sísmicos é denominado de aquisição sísmica. No chamado processamento sísmico, são considerados subconjuntos especiais dos dados sísmicos, denominados seções sísmicas. As seções sísmicas consistem de registros provenientes de pares fonte e receptor organizados segundo uma dada disposição, denominada configuração sísmica.

No presente trabalho, consideramos a chamada situação 2D na qual as fontes e receptores situam-se em uma mesma linha sísmica e se considera que, para efeitos de propagação, o meio em subsuperfície pode ser considerado um plano vertical abaixo da linha sísmica.

## 2.1 Arranjos Sísmicos

Listamos abaixo algumas das configurações mais utilizadas no processamento sísmico. Para simplificar a exposição, consideramos que a linha sísmica é horizontal, não sendo considerados os efeitos topográficos característicos dos levantamentos terrestres.

### 2.1.1 Configuração de afastamento constante (CO)

A configuração de afastamento constante consiste do deslocamento, ao longo do perfil de levantamento (no caso 2D, uma linha sísmica) de pares de fonte e receptor, para os quais a distância entre a fonte e o receptor é mantida fixa. A distância entre um par fonte e receptor é denominado afastamento, daí, o nome de configuração de afastamento comum ou "common offset"(CO). A Figura 2.1 mostra um levantamento CO em uma linha sísmica horizontal na situação 2D num modelo constituído por uma camada homogênea (velocidade e densidade constantes) acima de um refletor horizontal.

### 2.1.2 Configuração de ponto médio comum (CMP)

A configuração de ponto médio comum, ou "common midpoint" (CMP) é constituída de pares fonte e receptor dispostos simetricamente a um ponto médio fixo. Este ponto é denominado

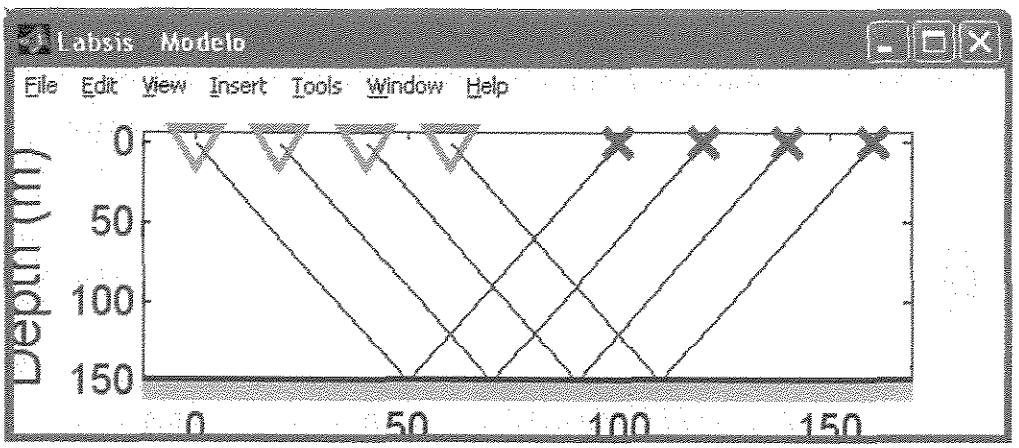


Figura 2.1: Configuração de afastamento constante (CO).

ponto médio comum (CMP).

No caso de um modelo de camadas homogêneas (isto é, cada camada possui parâmetros elásticos constantes) e separadas por interfaces planas e horizontais, todos os pares fonte e receptor da configuração CMP iluminam o mesmo ponto em profundidade. Por este motivo, no passado a configuração CMP era denominada configuração em ponto de profundidade comum, ou "common depth point" (CDP). Desta forma a configuração CMP tem o efeito de amostrar diversas vezes um mesmo ponto em sub-superfície para afastamentos diferentes. A Figura 2.2 mostra um levantamento CMP em uma linha sísmica horizontal na situação 2D num modelo constituído por uma camada homogênea.

A configuração CMP é a mais utilizada no processamento de dados sísmicos pois, mesmo para os casos em que o modelo geológico não corresponde à situação de camadas homogêneas horizontais, os pares fonte e receptor iluminam ainda uma mesma pequena região em subsuperfície.

Tendo em vista a propriedade de um conjunto CMP amostrar aproximadamente uma mesma região em profundidade, faz sentido a consideração da soma, com a devida correção de tempo devida aos vários afastamentos, de todos os registros e associar o resultado ao ponto CMP. O resultado do procedimento seria o de simular a resposta do meio no caso de par fonte e receptor

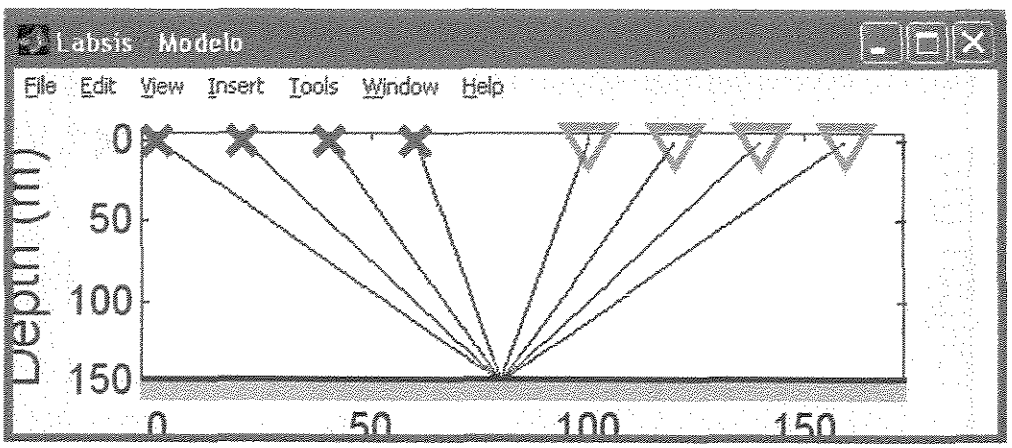


Figura 2.2: Configuração de ponto médio comum (CMP).

coincidente no CMP. O processo acima descrito é denominado empilhamento (ou "stack") e a correção que se deve aplicar a cada par fonte e receptor devido ao seu afastamento é denominada correção NMO ("normal move out").

Aplicando-se o procedimento acima a conjuntos CMP que se deslocam-se ao longo da linha sísmica, obtém-se a chamada seção empilhada NMO, a qual apresenta um significativo aumento na relação sinal/ruído em relação às ondas refletidas. Maiores detalhes sobre a correção NMO e sua aplicação ao empilhamento serão fornecidos mais adiante.

### 2.1.3 Configuração de Tiro Comum (CS)

Na configuração de tiro comum, ou "common shot" (CS), os registros em um conjunto de receptores são originados por um único tiro (uma única fonte). Uma seção CS consiste de um único tiro registrado em toda uma linha de receptores. Para cobrir uma grande área em subsuperfície com eventos de reflexão, as locações dos tiros e dos receptores são transladadas (ou avançam posições) em uma mesma distância, o arranjo sendo repetido ao longo da linha. Esses experimentos são repetidos até que se haja uma cobertura julgada suficiente da região em subsuperfície que se quer iluminar.

A Figura 2.3 mostra um levantamento CS em uma linha sísmica horizontal na situação 2D

num modelo constituído por uma camada homogênea acima de um refletor horizontal.

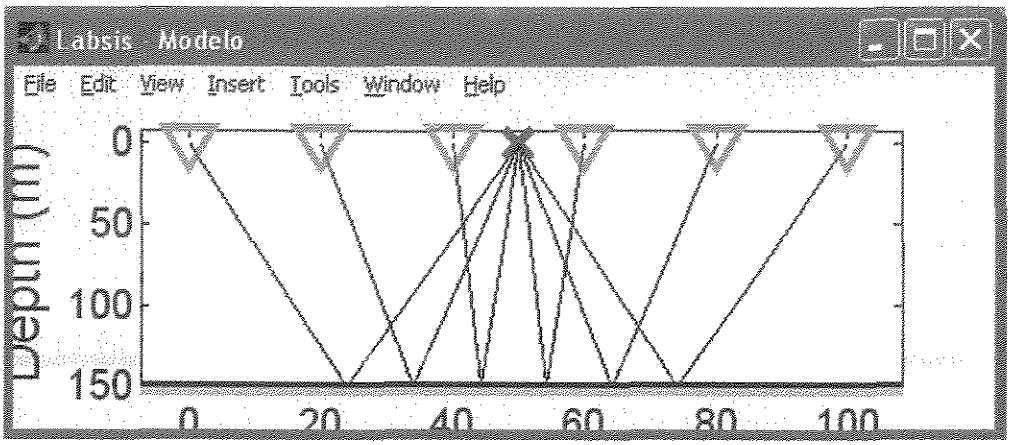


Figura 2.3: Configuração de tiro comum (CS).

### 2.1.4 Configuração de Afastamento Nulo (ZO)

A configuração de afastamento nulo, ou "zero-offset" (ZO), é definida como aquela em que os pares fonte e receptor ficam localizados na mesma posição, isto é, sem afastamento entre ambos. Esse tipo de configuração não pode ser realizado na prática. Com efeito, se o receptor e a fonte ficarem na mesma posição, na detonação da fonte, o receptor será destruído e, conseqüentemente não haverá registro do sinal.

A Figura 2.4 mostra um levantamento (fictício) em uma linha sísmica horizontal na situação 2D num modelo constituído por uma camada homogênea acima de um refletor horizontal.

Tendo em vista sua grande utilidade no processamento, a chamada seção ZO, isto é, aquela que seria obtida dessa situação fictícia, é simulada através do processamento (empilhamento) de dados em configuração CMP.

## 2.2 Modelagem Sísmica

A chamada modelagem sísmica computacional tem por objetivo simular a propagação de ondas em subsuperfície conhecidos o modelo geológico (especificações das densidades e velocidades

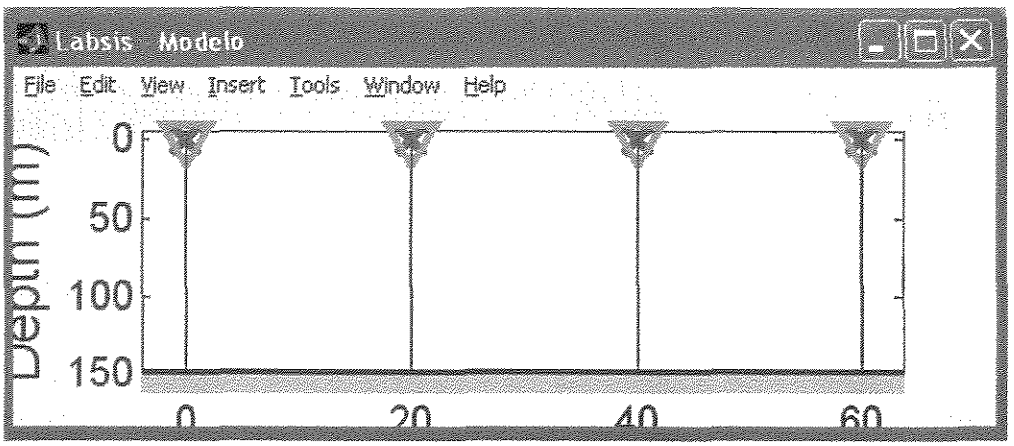


Figura 2.4: Configuração de afastamento nulo (ZO).

das camadas, forma das interfaces, etc.) , bem como os chamados parâmetros de aquisição sísmica (posições das fontes e receptores, especificação dos pulsos das fontes, etc.). O objetivo da modelagem é construir "sismogramas sintéticos", isto é as seções sísmicas correspondentes ao modelo geológico e o levantamento considerados. A modelagem sísmica é uma poderosa ferramenta para a interpretação sísmica e uma parte essencial dos algoritmos de inversão sísmica. Outra importante aplicação da modelagem sísmica é a avaliação dos parâmetros de aquisição sísmica, visando projetar um levantamento que otimize a iluminação da região alvo.

A formulação matemática da modelagem sísmica consiste de um sistema de equações diferenciais parciais (geralmente equações da onda acústicas, elásticas, visco-elásticas, etc.) acompanhadas de condições de contorno (por exemplo, comportamento nas interfaces e bordas do modelo) e condições iniciais (caracterização da emissão de energia pela fonte, tempo de propagação requerido, etc.). Por exemplo, na modelagem acústica e na ausência de fontes internas, o sistema de equações diferenciais que expressa a resposta de um modelo geológico a um campo de ondas incidente, é constituído de equações da onda do tipo:

$$\rho(\boldsymbol{x})\nabla\cdot\left(\frac{1}{\rho(\boldsymbol{x})}\nabla u(\boldsymbol{x},t)\right)-\frac{1}{c^2(\boldsymbol{x})}\frac{\partial^2 u(\boldsymbol{x},t)}{\partial t^2}=0. \tag{2.1}$$

Estas equações devem ser satisfeitas em cada camada, com condições de contorno nas interfaces que separam as camadas. Na equação acima, $u(x,t)$ representa a pressão em um ponto $x$ da camada e tempo $t$, sendo $\rho(x)$ e $c(x)$ a densidade e a velocidade acústica na camada.

São várias as abordagens existentes na literatura destinadas a resolver as equações da onda e condições de contorno nas interfaces, que caracterizam a modelagem. Conforme Carcione et al. (2002), essas abordagens podem, a grosso modo, ser classificadas em três principais categorias *métodos diretos*, *métodos integrais* e *métodos de traçado de raios*.

### 2.2.1 Métodos diretos

Nesta abordagem, o modelo geológico é aproximado por uma malha numérica, isto é, o modelo é discretizado em um número finito de pontos. As equações diferenciais parciais, incluindo as correspondentes condições de contorno, são resolvidas por métodos discretos, por exemplo o método de diferenças finitas e o método de elementos finitos. Os métodos diretos podem ser muito precisos quando uma malha suficientemente fina é utilizada.

Também são incluídos na categoria de métodos diretos os chamados métodos espectrais, os quais procuram expressar a solução do problema através de uma série de funções especiais (por exemplo, séries de Fourier-Bessel, séries de polinômios de Tchebicheff, etc.). Uma boa propriedade dos métodos diretos é a geração de instantâneos (ou "snapshots") que mostram o campo de ondas em um determinado tempo e que podem ser de valia na interpretação dos resultados. Além disso, os métodos diretos representam a totalidade do campo de ondas produzido pelas equações diferenciais e condições de contorno, levando a resultados mais realistas. Uma desvantagem dos métodos diretos, é a sua grande demanda de esforço computacional (tempo de processamento, memória, etc.), o que os torna inviáveis em várias situações (por exemplo modelamentos grandes conjuntos de dados 3D). Neste sentido os métodos de integrais e de traçado de

raios são bem mais econômicos e atraentes.

### 2.2.2 Métodos integrais

Os métodos integrais são baseados em representações integrais das soluções que formulam a modelagem. Esses métodos são geralmente baseados no chamado princípio de Huygens, através do qual o campo de ondas observado nos receptores pode ser considerado como uma superposição de "campos de ondas secundários" originados por distribuições de fontes pontuais em volumes ou superfícies. Os métodos integrais são, em geral, mais restritivos em suas aplicações do que os métodos diretos, exigindo modelos mais simples. Mesmo assim eles são bastante úteis em várias situações, pois permitem análises mais aprofundada, assintótica ou analítica que conduz a um maior entendimento dos fenômenos de propagação que a abordagem numérica dos métodos diretos. Apresentamos de maneira bem sucinta dois dos métodos integrais mais utilizados na modelagem sísmica, a saber, os métodos de Born e de Kirchhoff.

**Modelagem Born**

A modelagem Born é um método integral baseado na chamada aproximação de Born, bem conhecida na teoria das perturbações. Partindo em um modelo dado, modelo de referência, onde a propagação de ondas tem solução conhecida, a modelagem de Born procura simular a propagação em um modelo perturbado, isto é, caracterizado por parâmetros com pequenas diferenças em relação aos parâmetros correspondentes do modelo de referência. Um exemplo do modelagem de Born para o caso da equação acústica é apresentado em Novais (1998). Geralmente, considera-se uma certa região do modelo original, a chamada região perturbada, onde os parâmetros variam em comparação aos parâmetros do modelo de referência. Esta variação, porém, é suposta pequena, ou seja, os parâmetros perturbados não diferem muito dos não perturbados.

A modelagem Born é realizada através de uma integral de volume sobre a região perturbada,

cujo integrando contém um coeficiente de espalhamento. Fisicamente, a modelagem Born pode ser interpretada como proveniente de uma superposição de fontes secundárias pontuais (difratores) distribuídas pelo volume perturbado. Esta representação satisfaz a condição de reciprocidade (i.e., se ambas as posições de fonte receptor forem trocadas, a resposta não se altera).

Com descrito, por exemplo, em Novais (1998), a modelagem Born que simula a resposta em um receptor, $x_G$ devido a uma fonte em $x_S$, pode ser formulada pela integral:

$$u_B(x_S, x_G, \omega) = \omega^2 F(\omega) \int_V d^3x \frac{\alpha(x)}{c(x)} a(x, x_S, x_G) e^{i\omega\phi(x, x_S, x_G)} , \qquad (2.2)$$

onde, $u_B(x_S, x_G, \omega)$ representa o campo de onda monocromático de freqüência $\omega$ observado, $c(x)$ é a velocidade da onda no meio de referência, $\alpha(x)$ a velocidade de propagação perturbada e $V$ o volume do espalhamento, que é a região onde $\alpha(x)$ é não nulo. A função $F(\omega)$ representa a transformada de Fourier do pulso de entrada na freqüência $\omega$. O fator $a(x)$ representa o produto das amplitudes dos raios que ligam a fonte ao ponto em profundidade $x$ e deste ao receptor. Ambos esses raios supõe espalhamento geométrico do tipo fonte pontual. Finalmente, a $\phi(x_S, x_G, x)$ representa a fase, isto é o tempo de percurso total da fonte $x_S$ ao ponto "espalhador" $x$ e deste até o receptor $x_G$. Note que só há contribuição na região onde $\alpha(x)$ é não nulo, isto é, na região perturbada.

**Modelagem Kirchhoff**

A modelagem Kirchhoff, também um método integral, representa o campo de ondas observado, não como uma integral de volume (Born), mas como uma (ou várias) integrais de superfície ao longo dos refletores em subsuperfície. Esta integral de superfície é denomi-

nada na literatura sísmica de integral de Kirchhoff-Helmholtz (ver, por exemplo, Tygel et al. (1999b)).

O campo de ondas originado de uma fonte pontual e refletido em um refletor suave sobreposto por um meio acústico pode ser descrito pela integral de Kirchhoff-Helmholtz. Esta integral está baseada na idéia de que a reflexão do campo de ondas no receptor é uma superposição de campos de onda produzidos por pontos difratores (fictícios) distribuídos na interface. Estes difratores, chamados fontes secundárias de Huygens, são supostos fontes isoladas que não interagem entre si (suposição de espalhamento simples), excitados pelo campo incidente. O campo de ondas observado é dado pela superposição de todas as contribuições das ondas de Huygens. Na aproximação de alta freqüência, a amplitude das fontes de Huygens é determinada tratando o campo incidente no refletor, como uma onda plana que é refletida pela tangente plana do refletor. A Integral de Kirchhoff-Helmholtz resultante, descreve o campo de ondas refletido no receptor por uma integração ponderada sobre todas as fontes de Huygens ao longo do refletor.

A integral de Kirchhoff-Helmholtz é largamente utilizada para obter um modelo preciso das reflexões primárias em modelos compostos por camadas suaves (refletores).

Para o caso de uma fonte pontual em $\boldsymbol{x}_S$ e registrado em $\boldsymbol{x}_G$, a modelagem Kirchhoff para o caso de uma interface refletora, também no domínio da freqüência, é dada por (Tygel et al. (1999b))

$$u_K(\boldsymbol{x}_S, \boldsymbol{x}_G, \omega) = i\omega F(\omega) \int_\Sigma d\boldsymbol{x}\; R(\boldsymbol{x}_S, \boldsymbol{x}) a(\boldsymbol{x}_S, \boldsymbol{x}_G, \boldsymbol{x}) \mathcal{O}(\boldsymbol{x}_S, \boldsymbol{x}_G, \boldsymbol{x}) e^{i\omega\phi(\boldsymbol{x}_S, \boldsymbol{x}_G, \boldsymbol{x})}\,, \tag{2.3}$$

onde, $u_K(\boldsymbol{x}_S, \boldsymbol{x}_G, \omega)$ é campo monocromático de freqüência $\omega$) observado, $a(\boldsymbol{x}_S, \boldsymbol{x}_G, \boldsymbol{x})$ e $\phi(\boldsymbol{x}_S, \boldsymbol{x}_G, \boldsymbol{x})$, são como definidos na fórmula de modelagem de Born. Além disso, $R(\boldsymbol{x}_S, \boldsymbol{x})$

é o coeficiente de reflexão de onda plana, supondo a frente de onda associada ao raio que parte de $x_S$ e o refletor em $x$. Finalmente, $\mathcal{O}(x_S, x_G, x)$, denominado fator de obliquidade, representa uma função peso que é máxima no ponto de reflexão, $x_R$, determinado por $x_S$ e $x_G$ e se atenua quando o ponto $x$ no refletor se afasta de $x_R$. A Figura 2.5 ilustra a geometria da integral de Kirchhoff-Helmholtz.

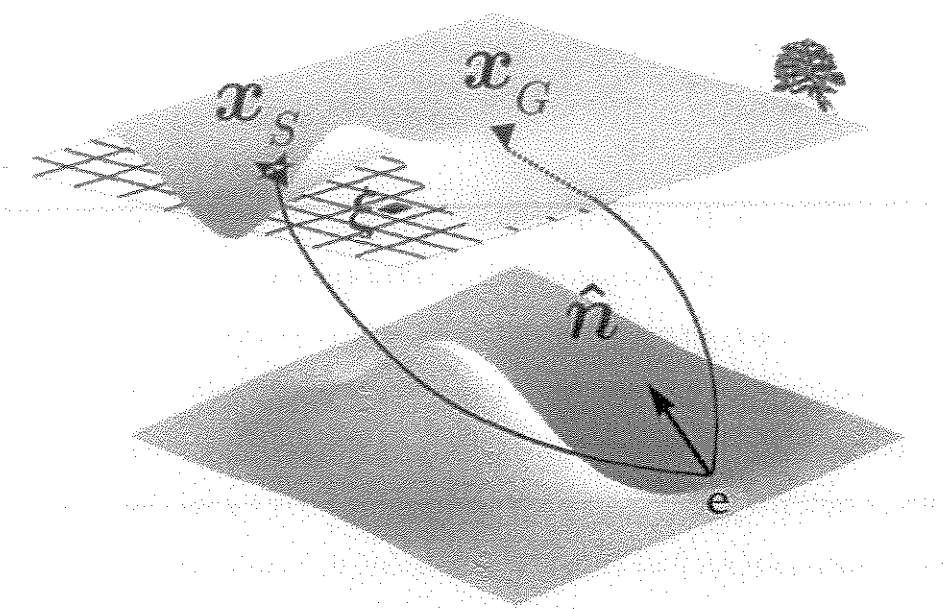


Figura 2.5: Geometria da integral de Kirchhoff-Helmholtz utilizada na modelagem Kirchhoff (adaptado de Jaramillo and Bleistein (1999)).

### 2.2.3 Traçado de raios

O método de traçado de raios tenta resolver as equações diferenciais parciais da modelagem através de aproximações assintóticas das soluções, obtidas pela chamada teoria dos raios. Um raio representa um caminho preferencial onde a energia se propaga de uma fonte a um receptor. Ao invés de considerar o campo de ondas na sua totalidade, a teoria dos raios procura expressar este campo através de uma superposição de "eventos"

$$u_{Total}(x,t) = \sum_{n=1}^{N} A_n(x) F(t - \tau_n(x)). \tag{2.4}$$

Na equação acima, cada evento é caracterizado por um tempo de trânsito, $\tau_n(x)$, e uma amplitude, $A_n(x)$, que dependem apenas da posição do receptor, $x$, considerando conhecido o campo incidente (tipicamente, o campo produzido por uma fonte pontual de posição dada). A função, $F(t)$, comum a todos os eventos que constituem a representação da solução dada pela equação (2.4), é suposta conhecida e representa o pulso da fonte incidente. Uma observação importante é que o pulso, $F(t)$, tem largura pequena, "isolando", desta forma os eventos individuais que constituem o campo de ondas.

A equação (2.4) contém, de forma implícita, uma escolha de eventos (no caso $N$ eventos) que devem representar o campo de ondas observado. Esta escolha é realizada a priori pelo modelador, baseado nas especificidades do problema. Para a obtenção dos tempos de trânsito e amplitudes dos eventos, que são as incógnitas na representação do campo modelado através da teoria dos raios, deve-se substituir a equação (2.4) no sistema de equações da onda que modela a propagação. Tendo em vista a linearidade das equações da onda (ver, por exemplo, o caso das ondas acústicas do tipo (2.1)), a substituição pode ser feita através de um evento de cada vez.

A substituição do evento

$$u(x,t) = A(x) F(t - \tau(x)), \tag{2.5}$$

na equação acústica (2.1) conduz à chamada equação iconal

$$|\nabla \tau(x)|^2 = \frac{1}{c^2(x)}. \tag{2.6}$$

a qual, ao lado de condições de contorno adequadas (tipicamente, a posição e a direção inicial de propagação), determina a trajetória do raio através de um sistema de equações diferenciais ordinárias, geralmente resolvido numericamente. Este sistema é denominado *sistema de traçado de raios*. A determinação das trajetórias dos raios e dos tempos de trânsito é referido como o *problema cinemático* da teoria dos raios (ver Červený (2001)).

Uma vez obtido o raio, procede-se a uma segunda etapa, que consiste da determinação da amplitude, $A(\boldsymbol{x})$, ao longo do raio. Esta determinação é referida como o *problema dinâmico* da teoria dos raios. A amplitude $A(\boldsymbol{x})$ ao longo do raio satisfaz à chamada equação de transporte (ver Červený (2001))

$$\boldsymbol{\nabla} \cdot \left( \frac{A^2(\boldsymbol{x})}{\rho(\boldsymbol{x})} \boldsymbol{\nabla} \tau(\boldsymbol{x}) \right) = 0. \tag{2.7}$$

Ao lado de condições de contorno adequadas (por exemplo a especificação de atributos da fonte, tais como padrão de radiação, etc.), a equação de transporte determina as amplitudes ao longo do raio através de um sistema de equações diferenciais ordinárias, geralmente resolvido numericamente. Este sistema é denominado *sistema de traçado dinâmico de raios* (ver Červený (2001)).

Existem, basicamente, dois principais tipos de traçado de raios (a) *traçado de raio de valor inicial* e (b) *traçado de raio de valor de fronteira*.

No traçado de raio de valor inicial, a direção do raio é conhecida em algum ponto do raio (ou determinada, por exemplo, a partir de uma superfície inicial). A posição do ponto e a direção do raio neste ponto constituem as condições iniciais do sistema de traçado de raios, os quais determi-

nam de forma única o raio como solução.

No traçado de raio de valor de fronteira, a direção do raio não é conhecida "a priori" em nenhum de seus pontos. Ao invés disso, outras condições são fornecidas, tais como, por exemplo, a especificação dos ponto inicial e final do raio, Červený (2001). Neste caso específico, falamos de "traçado de raio por dois pontos". Como um outro exemplo, sabemos como calcular a direção inicial do raio em todos os pontos em alguma superfície inicial, e estamos procurando o raio que passa por um ponto fixo específico, fora da superfície inicial.

A explicitação dos sistemas de equações diferenciais ordinárias que resolvem os problemas cinemático e dinâmico da teoria dos raios está fora do escopo deste trabalho. Recomendamos ao leitor interessado a referência Červený (2001) para a abrangente descrição da teoria dos raios e suas aplicações.

Finalizamos este parágrafo com algumas observações. A teoria dos raios produz soluções aproximadas (assintóticas) do problema de modelagem, que tem validade na vizinhança de eventos de interesse (como reflexões primárias, múltiplas, difrações, etc.). As modelagens obtidas pela teoria dos raios são muito úteis pela economia de esforço computacional e pelo foco que podem dar a eventos específicos de interesse, os quais podem ser depois comparados aos eventos observados nos dados reais. Especialmente para grandes e complexos modelos tridimensionais, onde o esforço computacional é enorme, os métodos de traçado de raios podem ser a única alternativa viável. Uma outra vantagem é que, devido à sua eficiência computacional, os métodos de traçado de raios são bastante utilizados em situações onde são necessárias modelagens sucessivas, como, por exemplo, na atualização de modelos de velocidade para melhor explicar dados observados.

## 2.3 Análise de Velocidades

A análise de velocidades é uma etapa crítica no processamento sísmico, uma vez que possibilita o empilhamento (*stacking*, a saber uma seção de afastamento nulo simulada que é a primeira imagem obtida do processamento sísmico. O empilhamento sísmico é realizado através de curvas de tempos de trânsito, denominadas *curvas de sobretempo* ou de *moveout*. A mais importante destas curvas é a chamada *curva de sobretempo normal* ou *normal moveout (NMO)*, a qual, aplicada a um conjunto de pares fonte e receptor na configuração de ponto médio comum (*CMP gather*), é dada por:

$$t^2(h) = t_0^2 + \frac{4h^2}{v_{NMO}^2} \,. \tag{2.8}$$

Na equação acima, $t_0$ representa o tempo de reflexão ao longo do raio normal (isto é, o raio refletido com afastamento nulo) a partir do CMP. O tempo $t_0$ é escolhido pelo intérprete, estando o mesmo associado a um evento de reflexão de interesse. A variável $h$ designa o meio afastamento entre um par fonte e receptor no *CMP gather*, sendo portanto também uma quantidade conhecida. Finalmente, $v_{NMO}$, denominado *velocidade NMO* representa o parâmetro desconhecido, objeto da análise de velocidades para sua determinação. A obtenção de $v_{NMO}$ se dá através da otimização da coerência (verossimilhança ou *semblance*) de energia resultante do empilhamento dos dados sísmicos segundo a curva NMO (2.8). O processo de determinação de $v_{NMO}$ através de coerência de empilhamento é denominado análise de velocidades. Este processo clássico é padrão em praticamente todas as seqüências de processamento sísmico.

O ajuste de hipérbole pode ser realizado usando-se pacotes que permitem graficamente sua visualização sobre conjuntos de tiro ou CMP visando à identificação direta do valor de propagação da onda refletida. Esta operação pode ser utilizada como uma primeira aproximação durante os trabalhos de interpretação pois, como as condições do meio natural se distanciam das condições

ideais que definem a hipérbole, esse ajuste, freqüentemente, irá depender muito da sensibilidade do intérprete, podendo assim levar a erros interpretativos. É, no entanto, de extrema importância numa primeira análise de conjuntos CMP e na análise de ruído. Observe que quanto maior o afastamento, maior será a correção NMO que deverá ser aplicada ao traço sísmico. Essas diferenças devem ser calculadas e corrigidas dos traços sísmicos para remover a influência do afastamento. Após essa operação, os traços podem somados para se obter o traço empilhado em uma determinada posição CMP (Figura 2.6).

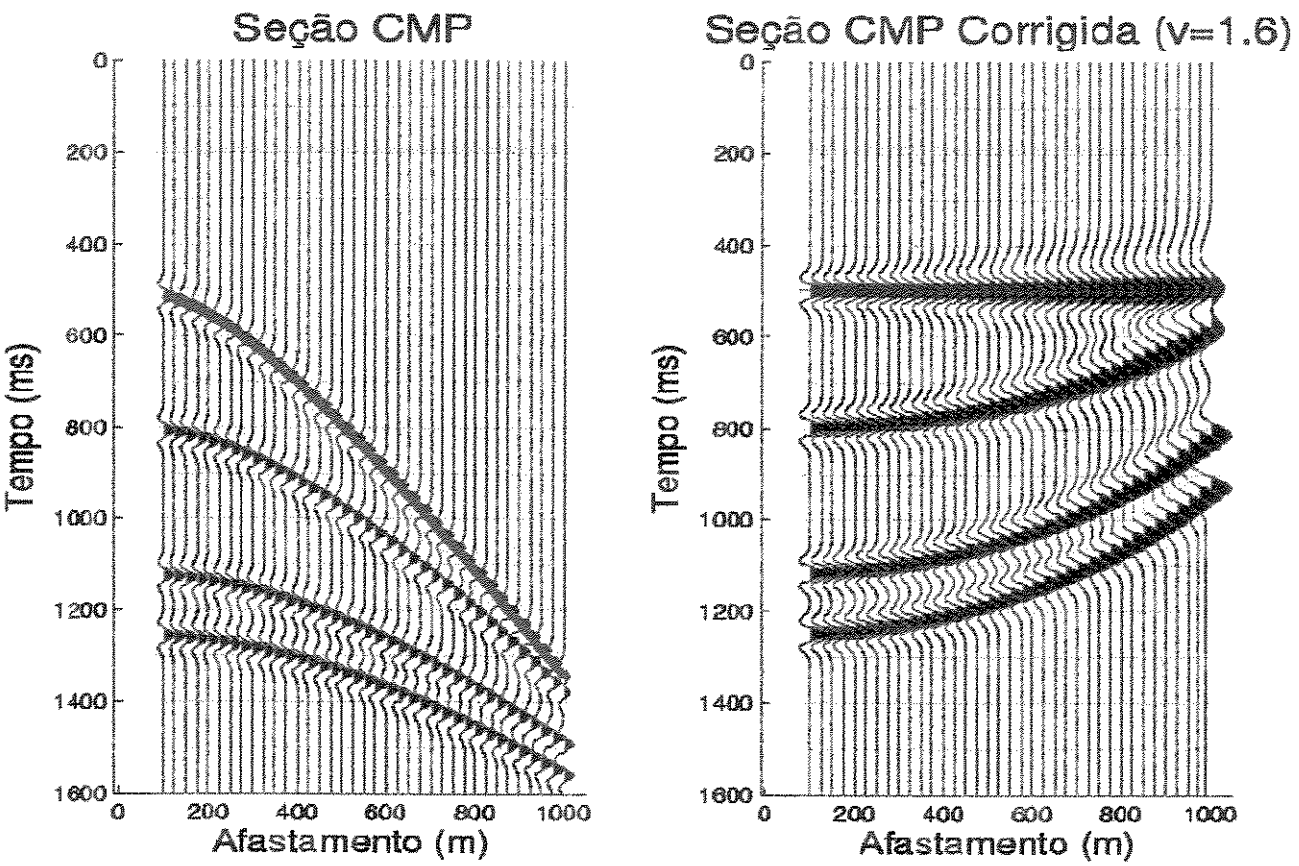


Figura 2.6: Conjunto de traços CMP pré empilhamento, antes e após a correção NMO, onde a velocidade aplicada é a que melhor representa a reflexão assinalada em vermelho.

Para a escolha da melhor velocidade NMO, busca-se no conjunto (*gather*) CMP aquela que melhor reduz a hipérbole de chegadas das reflexões a uma reta horizontal. Os pacotes de processamento, assim como o LabSis, permitem a aplicação de um painel de correções relativos a dife-

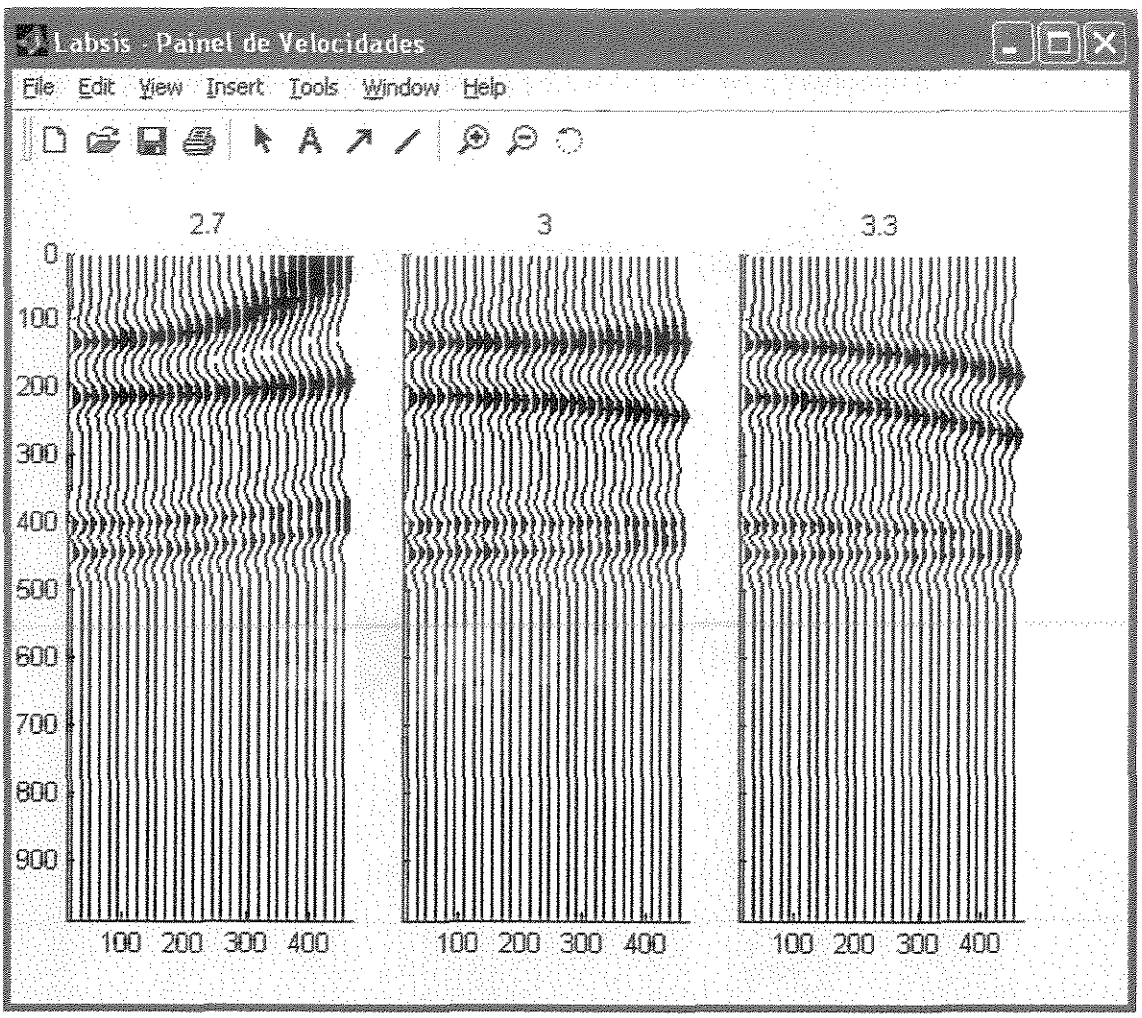


Figura 2.7: Painéis CVS (*constant velocity stacks*) com diferentes valores, onde a velocidade correta é a aplicada no painel 2.

rentes valores de velocidades NMO que variam segundo um pequeno incremento. A visualização dos conjuntos CMP, ou de parte de seções empilhadas, obtidos para cada valor correção, permite a escolha da velocidade a ser adotada (Figura 2.7).

Como resultado da correção NMO sobre os traços sísmicos, observa-se porém, uma distorção (estiramento) do pulso sísmico (figura 2.8). Este efeito indesejável da correção NMO, conhecido como *estiramento NMO* ou *NMO stretching*, aumenta com o afastamento, sendo mais acentuado nas pequenas profundidades. Na sísmica rasa, o estiramento NMO pode se tornar bastante severo.

Como conseqüência do estiramento NMO, é aplicado um *silenciamento* (ou *muting*) dos traços de afastamentos mas longos para permitir uma análise de velocidades e empilhamentos mais precisos.

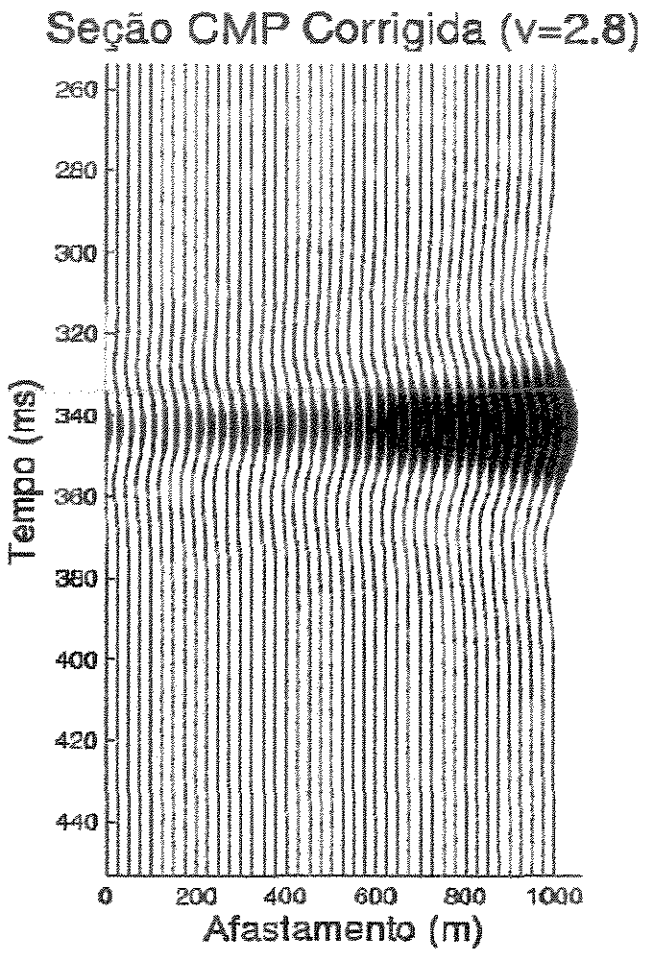


Figura 2.8: Ilustração do efeito de estiramento do traço (*stretching*) quando aplicada a correção NMO

Este é um importante aspecto onde se distingue o processamento sísmico convencional do processamento de sísmica rasa. Enquanto no primeiro são permitidos *stretching* de até 100%, no segundo essa porcentagem pode ser sensivelmente menor, chegando a 15%, Miller (1992), o que obviamente vai depender da cobertura utilizada, uma vez que diminui sensivelmente a relação sinal/ruído para os eventos mais rasos em razão dos traços empilhados.

## 2.4 Imageamento Sísmico

A chamada teoria unificada do imageamento sísmico apresentada por Hubral et al. (1996) e Tygel et al. (1996) provê uma metodologia geral para resolver uma larga variedade de problemas encontrados na obtenção de imagens e inversão de amplitudes através do processamento sísmico. As componentes chaves da teoria unificada consistem da utilização individual ou em cadeia, de transformações de migração e demigração de tipo Kirchhoff e em amplitude verdadeira. As operações de migração e demigração, bem como seus encadeamentos, são dados por integrais ponderadas. Na migração Kirchhoff, temos uma integral ponderada ao longo de curvas de difração. Na demigração Kirchhoff, temos uma integral de empilhamento ao longo de isócronas. Ambas essas integrais podem ser realizadas também por espalhamento ao invés de empilhamento. Maiores detalhes sobre estes processos podem ser encontrados em Hubral et al. (1996) e Tygel et al. (1996).

A migração em profundidade é um importante processo de imageamento consistindo na transformação de seções sísmicas observadas em tempo, para as correspondentes seções em profundidade. O processo de imageamento que objetiva a transformação inversa à migração é a demigração, a qual parte da seção migrada em profundidade e a leva de volta à correspondente seção em tempo. Em termos assintóticos (isto é, considerando eventos bem caracterizados pela teoria dos raios) a demigração é o processo inverso à migração ela desfaz os efeitos de migração que são usados ao ir de tempo para profundidade e vice-versa, quando o mesmo modelo de velocidades e a mesma configuração são utilizadas. A seguir descrevemos sucintamente as integrais ponderadas que realizam a migração e a demigração Kirchhoff como utilizadas na teoria unificada do imageamento sísmico.

**Migração Kirchhoff** A migração Kirchhoff consiste do empilhamento de dados no domínio do tempo, de tal maneira que, quaisquer reflexões possivelmente pertencentes a dado ponto em profundidade, $P$, escolhido arbitrariamente, são somadas e atribuídas a este ponto. A migração Kirchhoff pode ser representada na forma de uma integral de empilhamento (ver, por exemplo, Tygel et al. (1996))

$$V(P) = \int d\boldsymbol{\xi}\, W_M(\boldsymbol{\xi}; P)\, \partial_t^g D(\boldsymbol{\xi}, t)\Big|_{t=\tau(\boldsymbol{\xi};P)} \tag{2.9}$$

Na integral acima (que pode ser uma integral simples, no caso 2D, ou uma integral dupla, no caso 3D), $V(P)$ representa o resultado da migração no ponto em profundidade $P = P(x, z)$, sendo $D(\boldsymbol{\xi}, t)$ a seção sísmica a ser migrada. O parâmetro $g$ será 1 para caso 3D e $1/2$ para o caso 2D. O parâmetro $\boldsymbol{\xi}$, especifica o traço na seção sísmica. Este parâmetro pode ser um escalar, no caso de uma única linha sísmica característico de dados sísmicos 2D, ou um vetor sendo de duas componentes, no caso de localização areal, característica de dados sísmicos 3D. Em verdade, o parâmetro $\boldsymbol{\xi}$ pode ser entendido como a variável que especifica o par fonte e receptor $S = S(\boldsymbol{\xi})$ e $G = G(\boldsymbol{\xi})$ que constitui o traço sísmico. A variável $t$ especifica a amostra temporal do dado sísmico no traço. A notação $\partial_t D$ indica que o traço deve ser derivado em relação ao tempo para ser utilizado na operação de migração. No caso 3D, a operação $\partial_t D$ consiste da derivada partial simples de $D(\boldsymbol{\xi}, t)$ em relação ao tempo. No caso 2D, a operação correspondente é a chamada derivada meio, obtida, por exemplo, aplicando-se um filtro de tipo raiz quadrada da freqüência na transformada de Fourier do traço em relação ao tempo (ver detalhes em Tygel et al. (1996)). A função

$$\tau(\boldsymbol{\xi}; P) = T(S(\boldsymbol{\xi}), P) + T(G(\boldsymbol{\xi}), P) \tag{2.10}$$

representa a curva de empilhamento (no caso 3D esta curva é, na verdade uma superfície) da migração. Denominada *curva de difração* do ponto em profundidade, $P$, e associada aos traços $\boldsymbol{\xi}$ da seção sísmica, a curva $\tau(\boldsymbol{\xi}; P)$ consiste da soma dos tempos de percurso $T(S(\boldsymbol{\xi}), P)$, ao longo do raio que liga a fonte $S(\boldsymbol{\xi})$ a $P$, e $T(G(\boldsymbol{\xi}), P)$, que liga o receptor

$G(\xi)$ ao mesmo ponto. Finalmente, a função $W_M(\xi; P)$ designa o peso que deve ser associado a cada amostra empilhada de modo a produzir uma migração em amplitude verdadeira. Amplitude verdadeira significa que amplitude do resultado da migração teve o efeito de espalhamento geométrico automaticamente removido pelo processo de migração. A expressão do peso e o detalhamento de como o mesmo interfere no resultado da migração pode ser encontrada em Tygel et al. (1996).

Uma das principais vantagens da migração Kirchhoff é que os eventos de reflexão que são migradas no processo não precisam ser identificadas pelo usuário. O processo de integração (ou empilhamento) é aplicado independentemente do número e localizações desses eventos, produzindo a imagem em profundidade. Por outro lado, a operação de migração necessita de um modelo de velocidades que deve ser fornecido pelo usuário para o cálculo da curva de empilhamento e do peso da migração.

Embora freqüentemente chamada de tal, a migração não pode ser considerada um processo inverso à modelagem. Na realidade, a migração é uma operação adjunta para modelagem, Santos et al. (1999). A migração não é projetada para reconstruir o modelo de subsuperfície original, mas sim para localizar suas interfaces refletoras, provendo coeficientes de reflexão dependentes do ângulo como medidas de amplitude na seção migrada. A recuperação dos parâmetros físicos necessários para modelagem direta é feita por um processo adicional, chamado inversão.

**Demigração Kirchhoff** Como discutido em Hubral et al. (1996) e Tygel et al. (1996), em um sentido assintótico existe uma operação inversa à integral de migração de Kirchhoff. Essa inversa tem a mesma estrutura da integral da migração Kirchhoff, desta vez aplicada à uma seção migrada em profundidade. A demigração de Kirchhoff empilha ao longo de uma curva (ou superfície) os dados de uma seção migrada em profundidade. A demigração Kirchhoff consiste do empilhamento de dados no domínio da profundidade, de tal maneira

que, quaisquer eventos de reflexão possivelmente pertencentes a um dado ponto no domínio do tempo, $Q$, escolhido arbitrariamente, são somados e atribuídos a este ponto. Conforme Tygel et al. (1996), a integral de demigração de Kirchhoff pode ser escrita na forma

$$U(Q) = \int d\boldsymbol{x}\, W_D(\boldsymbol{x}; Q) \partial_z^g V(\boldsymbol{x}, z)\Big|_{z=\zeta(\boldsymbol{x};Q)} . \tag{2.11}$$

na integral de demigração acima (que pode ser uma integral simples, no caso 2D, ou dupla no caso 3D), $U(Q)$, denota o dado demigrado na posição $Q = Q(\boldsymbol{\xi}, t)$, sendo $V(\boldsymbol{x}, z)$ a seção migrada que é objeto da demigração. O parâmetro $g$ será 1 para caso 3D e $1/2$ para o caso 2D. A notação $\partial_z V$ indica que o traço da seção migrada deve ser derivado em relação à coordenada $z$ para ser utilizado na operação de migração (repare que a derivada em relação a $z$ na demigração desempenha o papel da derivada em relação a $t$ na migração). No caso 3D, a operação $\partial_z D$ consiste da derivada parcial simples de $V(\boldsymbol{x}, z)$ em relação a $z$. No caso 2D, a operação correspondente é a chamada derivada meio, obtida, por exemplo, aplicando-se um filtro de tipo raiz quadrada da freqüência na transformada de Fourier do traço em relação a $z$ (ver detalhes em Tygel et al. (1996)). A função $\zeta(\boldsymbol{x}; Q)$ representa a curva de empilhamento (no caso 3D esta curva é, na verdade uma superfície) da demigração. Denominada *isócrona de demigração* do ponto, $Q$, no domíno do tempo, e associada aos traços, $\boldsymbol{x}$ da seção migrada original, a curva $\zeta(\boldsymbol{x}; Q)$ consiste da inversa da curva de difração $\tau(\boldsymbol{\xi}; P)$ no seguinte sentido dado o ponto $Q = Q(\boldsymbol{\xi}, t)$ no domínio do tempo, determine os pontos $P = P(\boldsymbol{x}, z)$ em profundidade tais que a soma dos tempos de percurso $T(S(\boldsymbol{\xi}), P)$, ao longo do raio que liga a fonte $S(\boldsymbol{\xi})$ a $P$, e $T(G(\boldsymbol{\xi}), P)$, que liga o receptor $G(\boldsymbol{\xi})$ ao mesmo ponto sejam iguais ao tempo dado $t$ referente a $Q$. Finalmente, a função $W_D(\boldsymbol{x}; Q)$ designa o peso que deve ser associado a cada amostra empilhada de modo a produzir uma demigração em amplitude verdadeira. A expressão do peso e seu significado no contexto da realização da demigração em amplitude verdadeira pode ser encontrada em Tygel et al. (1996).

O conceito de amplitude verdadeira em conexão com a demigração Kirchhoff vem diretamente da definição de migração Kirchhoff. Como a demigração é o processo inverso à migração, ela desfaz as mudanças anteriores que a migração fez nos dados. Isto implica que a demigração deve mover os refletores de volta às curvas (ou superfícies) de tempo de reflexão. A demigração em amplitude verdadeira deve ainda reintroduzir automaticamente no resultado, o espalhamento geométrico, suposto ausente nas amplitudes da seção migrada em profundidade original (isto é, o a seção migrada de entrada era suposta em amplitude verdadeira).

Uma característica particularmente atraente da operação de demigração é que, qualquer rotina desenvolvida para migração Kirchhoff de amplitude verdadeira, pode ser facilmente modificada para executar a demigração de amplitude verdadeira. De fato, a similaridade estrutural dos conceitos de migração e demigração, constituem uma parte significante da teoria unificada do imageamento de reflexão sísmica de Hubral et al. (1996) e Tygel et al. (1996).

Finalizamos esta seção com algumas observações breves sobre a teoria unificada do imageamento sísmico descrita em Hubral et al. (1996) e Tygel et al. (1996). Uma importante contribuição desta teoria consiste no encadeamento das transformações de migração e migração (em qualquer ordem) para a resolução de uma vasta gama de problemas de imageamento. Dois desses problemas ocupam posição de destaque, a saber a *transformação de configuração* e a *remigração*. Na transformação de configuração, uma seção sísmica original, dada em uma certa configuração (por exemplo seção de afastamento comum em um afastamento $h_1$ dado), em uma outra seção sísmica (simulada), que corresponderia aos mesmos dados sísmicos, desta vez dispostos em uma diferente configuração (por exemplo em afastamento nulo). Na remigração, uma seção migrada em profundidade segundo um modelo de velocidades, é transformada numa outra seção migrada (simulada) oriunda de um diferente modelo de velocidades.

A transformação de configuração é realizada pela aplicação encadeada (ou *em cascata*) de

uma migração em profundidade dos dados sísmicos em sua configuração original, seguida de uma demigração em tempo na configuração desejada. As duas transformações podem ser reduzidas por procedimentos matemáticos a uma única integral de tipo Kirchhoff, com curva de empilhamento e pesos de amplitude verdadeira específicos. O processo de transformação de uma seção de afastamento comum (não nulo) para a sua correspondente (simulada) em afastamento nulo é denominado *migração para afastamento nulo* (MZO). Este processo pode ser de bastante valia no processamento sísmico.

A transformação de remigração é realizada, por sua vez, pela aplicação em cascata de uma demigração aplicada à seção migrada original, seguida de uma migração em profundidade da seção em tempo obtida, segundo um modelo de velocidades desejado. A remigração tem por objetivo a obtenção de seções migradas oriundas da atualização de modelos de velocidade. Da mesma forma que no problema de transformação de configuração, também a remigração pode ser reduzida a uma única integral de tipo Kirchhoff, com curva de empilhamento e pesos de amplitude verdadeira específicos.

# Capítulo 3

# LabSis

## 3.1 Motivação

Na Unicamp, mais especificamente em grupos de trabalho do Depto. de Matemática Aplicada e do Depto. de Geologia e Recursos Naturais, foram desenvolvidas, de maneira descentralizada, várias funções em Matlab que se destinam ao processamento, modelagem e ao imageamento sísmico. Em geral, cada função cumpre o papel de resolver um determinado problema específico da sísmica, fazendo com que a comunicação entre tais funções seja prejudicada por uma falta de padronização, com relação à formatação de dados e, entrada e saída de parâmetros.

Outro problema, era o fato de que nas funções existentes, a entrada de dados era por linha de comando, introduzindo-se vários parâmetros a cada vez que era necessário executar uma determinada função. Para que se completasse certas operações, era necessário rodar vários programas diferentes e formatar várias vezes o dado. Destas dificuldades, surge a necessidade da implementação do LabSis, que cumpre portanto, este papel integrador entre estas funções, agregando facilidade ao seu uso, através de um ambiente gráfico baseado em janelas, bem como, aumentando o potencial de uso perante toda a comunidade de pesquisa em sísmica da Unicamp.

O LabSis traz um ambiente de integração para o desenvolvimento de funções aplicadas à resolução de problemas da sísmica. O programa foi projetado pensando tanto na integração das

funções existentes, quanto nas funções que podem vir a serem desenvolvidas. Usando-se funções casca, pode-se introduzir novas funções sem alterar o conteúdo das rotinas existentes. Para tanto, basta-se colocar uma entrada de menu para a nova função e, construir uma função casca que faça o link entre o usuário e a nova função. Sendo assim, os usuários, tanto professores quanto alunos, podem inserir suas contribuições, as quais são resultados de suas pesquisas, fazendo com que o pacote cresça em conteúdo e integração e, fazendo com que tais contribuições sejam devidamente aproveitadas.

O fato de que o LabSis tenha sido escrito em Matlab, o torna multi-plataforma pois, o Matlab é suportado em Microsoft Windows e sistemas Unix. Sendo que, o LabSis foi devidamente testado em MS Windows e em GNU Linux, tendo obtido igual funcionalidade e performance em ambas as plataformas.

O LabSis é um software didático, desenvolvido especificamente para o ensino e a pesquisa facilitando o entendimento de conceitos teóricos complexos. O software é ideal para ser utilizado em cursos de graduação e pós-graduação. Por ser um programa leve, não exige uma máquina robusta, a não ser que o dado utilizado, assim o exija. Por ser um pacote integrado, as chances de distribuição são maiores do que as funções separadas, atingindo uma gama maior usuários e permitindo assim, uma melhor divulgação dos programas.

## 3.2 Porque o LabSis não foi construído antes

Para se organizar o conjunto de funções de diferentes localidades e autores, bem como para unificar a formatação de dados de entrada e saída, houve necessidade de um agente unificador que realizasse tal função. Além disso, apesar do Matlab ser uma linguagem amigável, a sua programação visual não é um procedimento trivial, na verdade, pode ser bem complexa. Para tanto é necessário um conhecimento mais aprofundado em programação Matlab, para a construção de programas utilizando suas ferramentas gráficas.

Os conceitos teóricos envolvidos nas funções utilizadas são bem abrangentes e complexos por tratarem-se de assuntos distintos como modelagem e migração. Para isso, é necessário alguém com conhecimentos prévios na área.

## 3.3 O que mudou com a introdução do LabSis

Com o LabSis, para se modelar um dado e depois migrar esse dado, basta entrar uma vez com os parâmetros do dado e, depois selecionar algumas entradas de menu para executar as funções desejadas. Com as funções antigas, esse procedimento era feito através de linhas de comando, dessa forma, para modelar um dado sísmico por traçado de raios, deve-se executar a função `raio` da seguinte maneira:

```
dado_sismico = raio(opg,opr,kk,x,z,xs,zs,xr,zr,c1,c2,t,tma);
```

para visualizar o dado modelado:

```
wigbplot(dado_sismico,t,xr);
```

para migrar o dado modelado:

```
dado_migrado = migra(y,h,t,dado_sismico,v,x,z);
```

e depois, para visualizar o dado migrado:

```
wigbkern(real(dado_migrado),z,x,1);
axis([min(x) max(x) min(z)max(z)]);
ylabel('Profundidade (m)','fontsize',14);
```

Com o LabSis, todos esses procedimentos ficaram englobados em único programa e, interface com o usuário ficou bem mais agradável, com um programa totalmente visual. Agora há uma maior integração entre as várias funções existentes. LabSis também foi projetado levando-se em conta funções futuras, para que possam ser adicionadas facilmente.

Com as funções separadas, os dados processados ficavam armazenados na memória, em variáveis do Matlab. Com o LabSis, os dados podem ser salvos em disco, para que o processamento possa ser continuado posteriormente, ou refeito quando necessário.

Outra funcionalidade introduzida com o LabSis foi a integração com outros softwares da área. Sendo o principal desses o InterSis, um software do grupo de geofísica computacional do Instituto de Matemática e Estatística e Computação Científica da Unicamp. Assim, é possível construir um modelo geológico no InterSis (que tem ferramentas mais poderosas para isso) e, importá-lo para o LabSis onde pode ser usado para modelagem sísmica. No LabSis também existe a possibilidade de importar dados no formato SU (Seismic Unix). Assim, um dado pode ser modelado no InterSis e importado no formato SU para o LabSis, onde este pode ser migrado.

## 3.4 Desafios encontrados

Para a construção do LabSis, apesar das funções básicas já existirem, tornou-se necessário conhecer a teoria a ser utilizada. Isso foi necessário para que se soubesse exatamente como tais funções funcionavam. A teoria envolvida mostrou-se complexa, isso por ter uma grande abrangência de assuntos.

Em geral, as funções existentes eram de difícil utilização, com uma interface não amigável, sendo um trabalho extra compreender o funcionamento de cada rotina e, qual a melhor forma de implementa-la ao pacote. As várias rotinas disponíveis estavam distribuídas em uma grande variedade de áreas e assuntos, portanto nem todas seriam apropriadas para o pacote, por tratarem temas tão distintos. Assim, o passo seguinte foi decidir qual seria a abordagem do programa e, quais funções seriam relevantes.

A introdução das funções casca resolveu uma grande variedade de problemas. Inicialmente, elas foram introduzidas como uma solução para se resolver a questão dos direitos dos autores das

funções. Como não queria-se alterar o conteúdos dos programas utilizados, optou-se pela criação desse tipo de funções, que seriam uma ponte entre as rotinas utilizadas e o LabSis. Tais funções também foram úteis na determinação da maneira como o programa trabalharia internamente com as variáveis, novamente servindo de ponte na ligação entre as funções utilizadas. Outro aspecto ligado às funções casca, é a entrada e saída de dados, a parte visual não era o problema, mas sim como a comunicação entre as funções introduzidas seria feita. Novamente, o problema foi resolvido como link as funções casca.

Um estudo mais aprofundado em programação visual em Matlab foi necessário, antes que os primeiros passos pudessem ser dados. Para a comunicação com o InterSis, teve-se primeiro que chegar a um acordo com o autor do programa, para que um formato de arquivo fosse padronizado, ficando assim como problema a ser resolvido a importação desse padrão de arquivo. Para importar dados sísmicos de saída do InterSis, optou-se pelo formato Seismic Unix (SU), por esse ser o único formato de saída desse programa. Para isso, foi necessário conhecer a estrutura deste formato binário, e pesquisas e testes ajudaram a resolver os problemas computacionais.

# Capítulo 4

# Intersis

O LabSis foi criado com a intenção de ser um programa capaz de trabalhar em conjunto com o InterSis. No entanto pode não ficar claro o papel do InterSis. Para que esse assunto se torne claro, esse capítulo é dedicado ao InterSis, explicando todas as suas principais características e funcionalidades. A maior parte deste capítulo foi baseado na tese de mestrado que resultou na construção do InterSis, de Ochoa (2003).

O InterSis foi criado utilizando uma biblioteca gráfica chamada GTK+, a qual faz parte do projeto GNU. Além dessa biblioteca, ele também faz uso das bibliotecas padrões de C e SU, para codificação e manipulação de dados sísmicos. O uso dessas bibliotecas visa a integração de várias ferramentas computacionais, fazendo do InterSis um pacote bastante completo e, solucionando uma série de dificuldades que se apresentam no dia-a-dia na modelagem sísmica. Uma vantagem adicional é que, desta forma, faz-se com que a modelagem, em cada uma de suas etapas, seja realizada de forma rápida segura e padronizada.

A modelagem sísmica, realizada através da utilização de pacotes de modelagem de caráter acadêmico e de livre acesso público, é a principal vantagem do InterSis. Em geral, a realização da modelagem inclui, além do programa específico, o uso de outros pacotes ao longo do processo. As etapas estão divididas basicamente em pré e pós-modelagem. Etapas pré-modelagem incluem a especificação do modelo e dos parâmetros de aquisição, por exemplo. Já as etapas pós-modelagem,

são relacionadas à visualização do resultado obtido ou à preparação desse resultado para processos subseqüentes. O InterSis tem a capacidade de trabalhar com esses dados e formatá-los para sua entrada no modelador e para o uso do resultado em outras aplicações.

O ambiente gráfico do InterSis inclui ferramentas necessárias para o fluxo da modelagem de forma modular e amigável, sendo que o usuário pode especificar as informações iniciais para a modelagem, escolher o modelador e utilizar ferramentas para análise dos dados sísmicos. O InterSis utiliza uma biblioteca livre, GTK+, que possui os recursos necessários para construir esse tipo de interface amigável e, solucionar problemas de ambiente gráfico. A utilização desse tipo de biblioteca faz com que o uso e distribuição do software, livremente, seja possível.

A execução de processos no InterSis é dividida em duas categorias: processos internos e externos. Os processos internos utilizam a linguagem C, permitindo a realização de operações internas e a comunicação com os pacotes externos (modeladores, Seismic Unix, sistema operacional). A linguagem do código fonte é a linguagem C, facilitando a comunicação entre a interface e seus processos. Os processos externos constituem principalmente a execução de processos no sistema operacional (Linux), os quais invocam comandos em terminais do sistema operacional, e são facilitados pelo InterSis, que gera arquivos de lote (scripts) para execução dessas funções. Uma vantagem extra do InterSis, também presente no LabSis, é a reprodutibilidade de resultados, já que os arquivos gerados ficam disponíveis ao usuário em sua área de trabalho e podem ser executados diretamente no terminal.

## 4.1 O Modelador utilizado pelo InterSis - Seis88

O programa Seis88 foi o primeiro modelador com o qual foi aplicado o InterSis. A motivação para o seu uso foi a alta qualidade científica, sendo o mais completo e reconhecido por parte da comunidade acadêmica. Embora existam versões mais recentes, o Seis88 foi escolhido por ser muito utilizado e de uso livre. Por ser um programa desenvolvido há bastante tempo e sem suporte, o

Seis88 possui limitações impostas pela linguagem em que foi implementado, *FORTRAN77*. Algumas das limitações foram redefinidas para evitar maiores restrições ao InterSis. Isso quer dizer que o código fonte do programa sofreu modificações, dentre as quais destacam-se:

**Número máximo de receptores:**o número máximo de receptores por tiro era de 99, a saber, representava o limite superior para um inteiro de 2 dígitos. Este valor foi incrementado para 999, correspondente a 3 dígitos e, amplamente suficiente para nossas aplicações.

**Tamanho da malha de velocidade:** A malha do modelo de velocidades, era de um tamanho fixo de 50x50. A mudança realizada, permite que a especificação de malha de tamanho variável seja especificada por inteiros variáveis de 1x1 até 99999 x 99999.

**Arquivos de saída:** As informações de cinemática e dinâmica dos raios traçados pelo Seis88 eram misturadas com a informação das trajetórias, em um único arquivo de texto formatado. Atualmente, as informações são disponíveis em arquivos diferentes.

**Criação de sismograma e diagrama de trajetória de raios:** Através da solução no item anterior, as informações foram separadas em diferentes arquivos, o que permite a elaboração do sismograma no InterSis e, o uso da informação de trajetória de raios de forma rápida e recursiva. Essa solução tem um impacto muito importante para o usuário pois, a partir dela foi possível a exploração de utilidades não previstas no pacote original, tais como: a escolha de pulsos para a fonte e a construção de um mapa de cobertura (*fold*).

**Especificação do modelo sísmico:** Dentro do modelador Seis88, a real especificação das interfaces era oriunda de um interpretação interna e invisível ao usuário. O InterSis permite que a construção das interfaces é realizada de forma gráfica e direta pelo usuário. Em seguida, essa interface é automaticamente parametrizada e formatada como requerida no Seis88.

## 4.2 Limitações Computacionais do InterSis

Algumas limitações não puderam ser contornadas pelo InterSis, e são classificadas abaixo:

1. O número máximo de interfaces é de 20, incluindo a superfície de aquisição e a base do modelo. Esse é o número de interfaces aceito pelo Seis88.

2. A construção das interfaces é feita por meio de *splines* cúbicas e/ou linhas poligonais, limitando o uso de interfaces mais complexas.

3. O InterSis suporta somente os modeladores Seis88 (traçado de raios) e fd2d (diferenças finitas).

4. Modelos de extrema complexidade (lentes, zonas isoladas, etc) não são aceitos.

5. Apenas modelos 2D e 2.5D são considerados.

6. Modelos de velocidade e densidade estão limitados a funções com variação linear (gradientes constantes) em qualquer direção.

7. Aquisições são consideradas somente na forma de tiro comum, outras aquisições podem ser geradas através de reordenações de conjuntos de aquisições de tiro comum.

8. Os parâmetros de construção são considerados como padrão e são:
   Dimensões do Modelo:
   x : [0.5-99.0]km
   z : [0.5-99.0]km
   Tiros : [1-1000]
   Receptores : [1-1000]
   Gap : [2-20]
   Tempo de Registro : [0.1-10.00]s

## 4.3 Módulos do InterSis

O InterSis é uma interface gráfica projetada para interagir com outros pacotes de modelagem sísmica de livre uso, permitindo a pré-modelagem que consiste em: definir modelos geológicos, definir pulsos de fonte, aquisições, etc. E, também permite a pós-modelagem, como: visualização e manejo de resultados, avaliação dos parâmetros do modelo geológico, parâmetros de aquisição, pulso, amostragem, etc.

Por esse motivo, o InterSis é dividido em 6 módulos, sendo 1 módulo principal e cinco módulos auxiliares, os quais serão vistos a seguir.

### 4.3.1 Módulo Principal: Especificações Gerais do Modelo

Este módulo permite ao usuário especificar as características gerais do modelo. Alguns desses parâmetros são: dimensões, número de camadas, aspectos relacionados com os modeladores a serem utilizados e características físicas. Outros parâmetros a serem especificados são descritos abaixo:

**Velocidades:** Devem ser especificadas em cada camada as velocidades de onda P e onda S, sendo que o InterSis considera somente três tipos de variação de velocidades de onda P no interior de cada camada.

**Densidade:** Pode ser especificada de três formas, como: densidade constante, função linear da velocidade de onda P, padrão Seis88 (relação linear).

### 4.3.2 Módulo para Construção de Modelos Geológicos

Este módulo conta com uma área gráfica e múltiplas opções para as especificações de modelos multi-camadas 2D. As interfaces são interpretadas de acordo com as especificações do modelador Seis88. Assim é possível criar interfaces formadas por linhas poligonais e/ou curvas suaves.

Neste módulo, pode-se determinar também, a chamada *assinatura* ou *código do raio* (para o uso com o modelador de traçado de raios).

### 4.3.3 Módulo de Parâmetros de Aquisição

Neste módulo faz-se as especificações do tipo de aquisição e a geometria da mesma. Existem 3 tipos de aquisição disponíveis neste módulo: Multicobertura, Afastamento Nulo (Z0) e Perfilagem de Poço. A descrição de cada tipo de aquisição está a seguir:

**Multicobertura:** Várias fontes e receptores são arbitrariamente distribuídos sobre a linha sísmica. Pode-se utilizar a geometria regular de receptores/pontos de tiro, onde a distribuição de receptores/pontos de tiro pode ser fixada, e geometria irregular de receptores/pontos de tiro, onde a informação de distribuição dos mesmos é carregada por meio de um arquivo texto.

**Afastamento Nulo (Z0):** Consiste em gerar seções de afastamento nulo utilizando-se pontos regularmente espaçados.

**Perfilagem de Poço:** A localização do ponto de tiro e a localização do poço, devem ser fornecidas, assim como, a distribuição regular dos receptores dentro do poço.

### 4.3.4 Módulo para Especificação do Pulso

A escolha acurada do pulso é de vital importância para a criação de um sismograma e, é feita através do controle de suas características, como: causalidade, fase, duração e conteúdo de freqüência.

Neste módulo é possível criar pulsos analíticos, como: pulso de *Gabor*, pulso de *Berlage*, pulso de *Ricker*; spikes de banda limitada, importar e criar arbitrariamente pulso.

A análise do pulso no domínio do tempo e da freqüência, também pode ser feita neste módulo, sendo que para isso o InterSis utiliza uma biblioteca computacional do SU (Seismic Unix).

### 4.3.5 Módulo de Visualização e Análise de Dados Sísmicos

Já entrando na etapa de pós-modelagem, o InterSis tem um módulo exclusivo para visualização e análise de dados sísmicos. O InterSis é capaz de visualizar dados no formato SU, tanto produzidos internamente como dados gerados em outros pacotes.

Dentro desse módulo é possível: verificar cabeçalhos, aplicar transformadas de Fourier 1-D e 2-D, aplicar ganho e fazer adição de ruído. Auxiliando na correção de parâmetros de aquisição, devido a *alias espacial* ou *temporal*.

### 4.3.6 Módulo de Trajetória de Raios e Cobertura

Dentro desse módulo é possível reconhecer propriedades cinemáticas e dinâmicas do traçado de raios e das trajetórias dos raios, também é possível mensurar o grau de iluminação (cobertura) do refletor. Essas informações, através do InterSis, são disponibilizadas no modo visual e texto.

## 4.4 Integração com LabSis

O LabSis e o InterSis foram construídos com a intenção de se complementarem. Nunca cogitou-se a idéia de se construir dois programas, tão poderosos, sem que pudessem trocar informaçõe entre si, para continuar uma etapa ou comparar um resultado. Por esse motivo, os programadores dos dois softwares chegaram a um acordo sobre o formato de dados que seria usado para essa integração.

O InterSis tem uma poderosa ferramenta para geração de modelos estratigráficos, assim o usuário tem a opção de gerar um modelo geológico no InterSis e usá-lo no LabSis, para uma modelagem.

O usuário tem a opção de modelar esse mesmo dado no InterSis e, depois exportá-lo para LabSis, no formato SU, para comparar os resultados dos diferentes métodos. Ambos os dados podem ser imageados ou processados, usando-se as ferramentas disponíveis no LabSis.

# Capítulo 5

# Características Computacionais

## 5.1 Principais Características Computacionais do LabSis

A principal característica do LabSis é o fato de ser um pacote integrador de várias funções diferentes. Para facilitar tal tarefa, seu principal recurso é poder fazer isso sem a necessidade de usar a linha de comandos, usando uma linguagem orientada a objetos, totalmente visual, Math-Works Inc.. Permitindo assim, o uso do software de maneira intuitiva e versátil

LabSis foi desenvolvido em Matlab 6.5, Release 13, para Microsoft Windows e testado em Matlab 6.1, Release 12, para Linux, executado em RedHat 7.3 com kernel 2.4.18. Por não ser um programa que necessita ser compilado, o LabSis não teve problemas para ser executado em ambas as plataformas, e sem a necessidade de alterações computacionais.

O LabSis é um software multivariável, isso significa que, ele pode trabalhar com vários dados sísmicos ao mesmo tempo. Essa característica dá ao usuário uma mobilidade maior para trabalhar com modelagem, processamento e imageamento, sem a necessidade de sair e entrar no programa ou limpar memória para trabalhar com um novo dado. Essa característica facilita o reprocessamento do dado por diferentes algoritmos, mantendo o dado original na memória para comparação.

Todas as funções do LabSis com funcionalidades semelhantes são agrupadas por módulos.

Esses módulos podem ser acessados através de uma hierarquia de menus e submenus que ficam na barra de ferramentas da janela, assim como a maioria dos programas gráficos. Esses menus estão agrupados pelos seguintes módulos: Dado, Modelo, Modelagem, Processamento e Imageamento (maiores detalhes sobre as funcionalidades de cada um desses módulos podem ser encontrados no Capítulo 6).

No Matlab, interfaces visuais trabalham com uma quantidade muito grande de variáveis. Cada objeto adicionado em uma janela, associado a uma determinada função, necessita de uma variável na memória para ser chamada durante a execução. O LabSis lida com isso limpando, uma a uma, todas as variáveis que se tornam desnecessárias após desempenharem o seu papel. Assim, quando o programa é encerrado, as únicas variáveis remanescentes na memória do Matlab, são as que contém o dado sísmico em si, e outras que possam a ser úteis, como as referentes a modelos geológicos e cabeçalhos de arquivos importados.

O LabSis permite a criação de dados sísmicos usando vários arranjos diferentes, sendo que as possíveis limitações residem apenas nas funções externas utilizadas. Assim, o LabSis permite ao usuário trabalhar com os seguintes arranjos: CO (*common offset*), CMP (*common middle*), CS (*common shot point*) e ZO (*zero offset*).

Para salvar os arquivos em disco, o LabSis trabalha com arquivos *.mat*, um formato de arquivos binários, de precisão dupla, usados pelo Matlab. Sua vantagem é que eles podem ser criados em uma máquina para depois serem lidos pelo Matlab, em outra máquina, com um formato de ponto flutuante totalmente diferente. Essa operação é feita mantendo toda a precisão que os diferentes formatos permitem.

## 5.2 Dificuldades Computacionais

É sabido que a comunicação entre o programador e o usuário é muitas vezes complicada, pois o que pode parecer simples na mente do programador, é algumas vezes obscuro para o usuário e vice-versa. Um comando pode parecer simplório para o usuário pois executa uma tarefa simples, porém o algoritmo subjacente pode conter um alto grau de sofisticação.

Dessa forma, esse capítulo visa mostrar alguns detalhes da programação, mostrando os principais desafios computacionais encontrados e como eles foram solucionados.

### 5.2.1 A variável interna *s*

O LabSis foi projetado para permitir uma total integração entre as diferentes funções e o usuário. Determinar qual seria formato interno do LabSis, ou seja, como todas as informações ficariam armazenadas, foi desafio inicial, mas sua solução, se tornou a base de todo o programa.

Para o formato interno das variáveis do LabSis, foi usada uma variável, do tipo *structure array*, chamada de *s*. Essa variável *s* contém todos os parâmetros necessários para rodar a maior parte das funções do LabSis, sem que o usuário tenha que digitá-los novamente, sendo apenas de uso interno do programa. Não é necessário alterá-la, via linha de comando, apesar de ser possível e permitido. A seguir tem-se um exemplo dos campos da variável *s*:

```
s =

          data: [100x200 double] O dado sísmico em si      [nxsxnt]
        domain: 'time'           Domínio                    string
          type: 'seis'           Tipo                       string
           wav: 'ricker'         Tipo da wavelet            string
aquisitiontype: 'co'             Tipo de aquisição          string
         xsini: -800             Coordenada x inicial da fonte
```

```
     dxs: 15                Intervalo em x para a fonte
ntracos: 100                Número de traços sísmicos
     nxs: 100               Número de pontos do vetor xs
      xs: [1x100 double]    Coordenadas x da fonte      [1xnxs]
   zsini: 0                 Coordenada z inicial da fonte
     dzs: 0                 Intervalo em z para a fonte
     nzs: 100               Número de pontos do vetor zs
      zs: [1x100 double]    Coordenadas z da fonte      [1xnzs]
   xrini: -700              Coordenada x inicial dos receptores
     dxr: 15                Intervalo em x entre os receptores
     nxr: 100               Número de pontos do vetor xr
      xr: [1x100 double]    Coordenadas x dos receptores [1xnxs]
   zrini: 0                 Coordenada z inicial dos receptores
     dzr: 0                 Intervalo em pontos do vetor zr
     nzr: 100               Número de pontos do vetor zr
      zr: [1x100 double]    Coordenadas z dos receptores [1xnzs]
    tini: 1                 Tempo de tiro inicial
      dt: 20                Intervalo de amostragem
      nt: 100               Número de pontos do vetor t
       t: [1x200 double]    Vetor t com os tempos       [1xnzs]
     tma: 144               Tamanho da wavelet
    nome: 'seção 52'        Nome para a variável s atual string
       h: [1x100 double]    Vetor com os offsets    [1xntraços]
```

### 5.2.2 A ferramenta Guide

GUIs (*Graphical User Interface*) é uma interface para o usuário, construída em linguagem Matlab, utilizando objetos gráficos, como botões, caixas de texto, sliders e menus. Em geral, o significado destes objetos já é conhecido pela maioria dos usuários. Por exemplo, quando movemos um slider, um valor é alterado, ou quando pressionamos um botão OK, nossos ajustes são aplicados, e uma caixa de diálogo é fechada. É claro que, para alavancar essas funcionalidades, o programador deve ser sensato em como usar essas ferramentas integradas.

Aplicações que contem GUIs, são geralmente mais fáceis de se aprender e utilizar, pois não há necessidade se conhecer os comandos que estão na programação, ou como eles funcionam. O resultado de uma particular ação do usuário, vai depender de quão clara e limpa é a interface construída. Por essa razão, a maneira de como o programador usa aplicações GUIs pode determinar o sucesso ou o fracasso do programa.

Matlab implementa GUIs como sendo um objeto *figure* em janelas, contendo vários tipos de outros objetos. Pode-se programar cada objeto para desempenhar uma determinada ação quando ativado pelo usuário. Adicionalmente, o programador pode salvar e executar o seu GUI quando julgar necessário. Com a intenção de facilitar essas tarefas, a MathWorks criou o GUIDE, um ambiente para desenvolvimento de interfaces gráficas para o usuário, que acompanha a distribuição Matlab.

O GUIDE, apresentado na figura 5.1, foi primeiramente construído para ser uma ferramenta de *layout*. Entretanto, o GUIDE também gera um arquivo com extenção .m, o qual contém o código para inicialização do GUI. Esse arquivo providencia uma estrutura para implementação dos *callbacks* (funções que executam uma determinada ação quando o usuário ativa componentes no GUI). O GUIDE é uma ferramenta que auxilia na construção de interfaces gráficas em Matlab, permitindo adicionar objetos de sua interface de maneira fácil e interativa, bastando selecionar o objeto desejado em uma lista e arrastá-lo para a janela da interface que está sendo construída.

Assim como é possível escrever um arquivo .m que contenha todos os comandos para organizar um GUI, é fácil usar o GUIDE para organizar os componentes interativamente e gerar os dois tipos de arquivos necessários para salvar e executar o GUI, são eles:

- Arquivo .fig - contém uma descrição completa da figura GUI e de todos os seus objetos *children* (sub-objetos que podem ser colocados em GUI como gráficos), assim como os valores de todas as propriedades dos objetos.

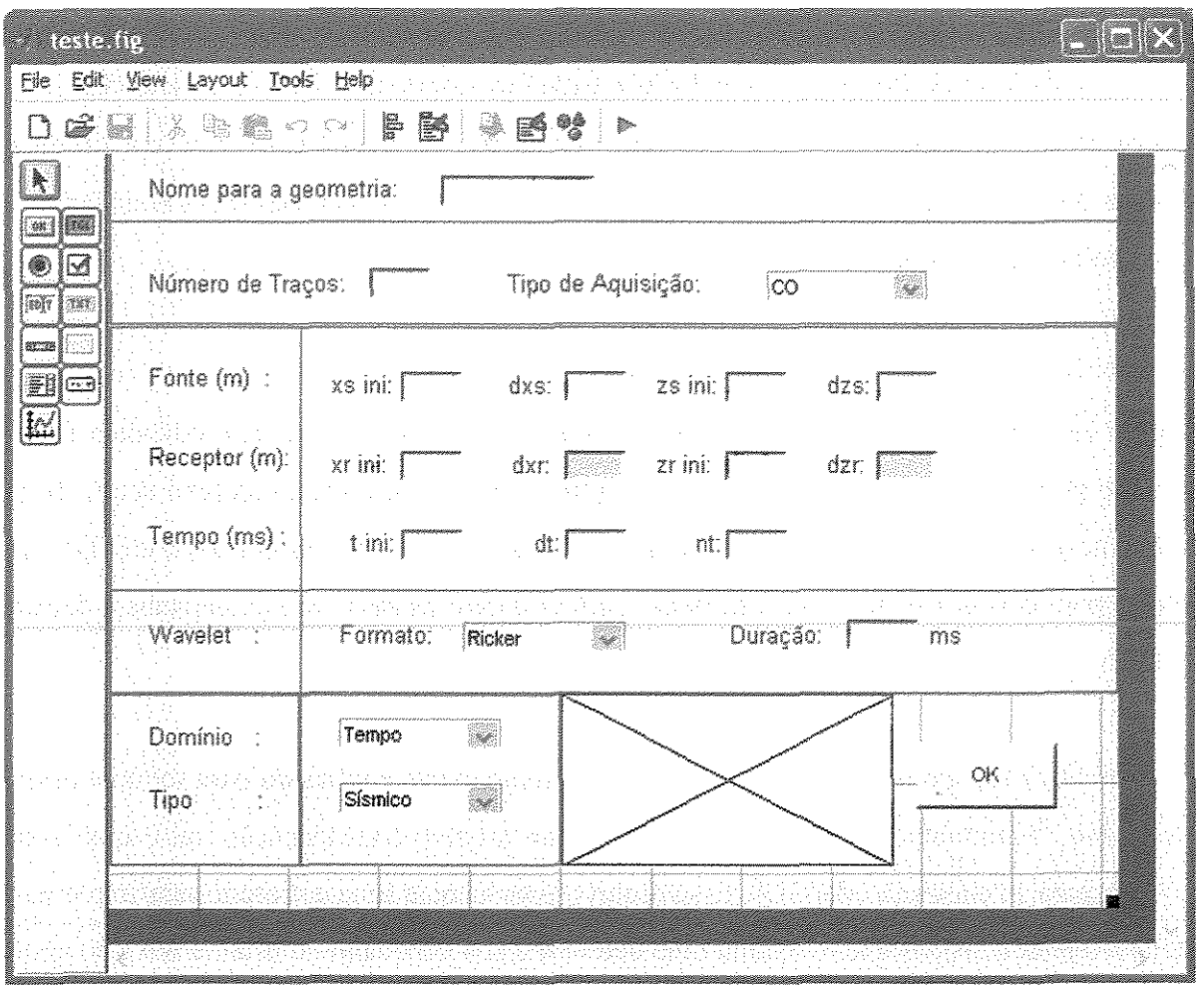


Figura 5.1: Ferramenta GUIDE, usada para a criação de interfaces gráficas em Matlab

- Arquivo .m - contém as funções que executam e controlam o GUI e seus *callbacks*, que são definidos como sub-funções. Esses arquivos .m são referenciados na documentação do Matlab como sendo aplicações.

Por esses motivos, o GUIDE inicialmente mostrou-se uma ferramenta bem útil para a construção do LabSis. No entanto, nota-se, pela descrição acima, que o arquivo .m, que é um arquivo texto padrão ASCII, não contém o código do layout dos objetos contidos na interface; todas essas informações ficam guardadas arquivo .fig, que é um arquivo binário. Isso torna difícil a comunicação dinâmica com funções externas durante a execução do programa, já que todas as informações ficam indisponíveis para funções externas.

Como o LabSis é constituído de um grande número de funções, de diversos autores, essa ferramenta foi descartada, optando-se por colocar todas as informações da interface gráfica manualmente no arquivo .m. Apesar de ser uma opção mais trabalhosa (o porque disso será explicado na próxima seção), esse método permite uma integração dinâmica com funções externas e, por ser um arquivo ASCII, garante a transparência do código, facilitando o acesso para melhorias futuras a serem feitas pelo autor ou por outros programadores.

### 5.2.3 Figuras

No Matlab, a saída gráfica é direcionada para uma janela separada da janela de comando e essa janela é referenciada como uma figura. As características desta janela são controladas pelo sistema de janelas do computador e, pelas propriedades de figuras do Matlab. Funções gráficas criam automaticamente uma figura com as propriedades padrão. Se existem múltiplas figuras, será considerada como ativa a última utilizada, ou a última selecionada pelo mouse.

Novas figuras podem ser criadas através da função `figure` do Matlab. No momento da criação de uma nova figura é possível nomeá-la, de modo que operações ou objetos possam utilizar a figura correta. Esse comando também permite a introdução de vários parâmetros para a figura, como: cor, posição, tamanho, maneira que vai executar certas tarefas, informações, ícones e menus a serem mostrados, ou que tarefas serão desempenhadas quando a figura é criada ou fechada. Na próxima seção temos um exemplo simplificado dos parâmetros de criação de figuras utilizado no LabSis.

### 5.2.4 Programando com objetos gráficos uicontrol

Para a construção de um GUI pode-se usar a ferramenta GUIDE, ou objetos gráficos 'uicontrol', que são os objetos usados pelo programador para implementar a interface gráfica através dos

arquivos .m. Quando selecionados, a maioria dos objetos 'uicontrol' desempenham uma ação pré-definida pelo programador. Matlab suporta numerosos estilos de 'uicontrols', cada um construído para um propósito diferente, eles são:

- caixas de texto editáveis
- botões de rádio
- menus pop-up
- frames
- caixas de listas
- botões de click
- botões de posição
- sliders
- check Box
- textos estático

Essa ferramenta é muito mais poderosa do que o GUIDE e dá ao programador uma maior liberdade, por permitir explorar totalmente todas as opções disponíveis do objeto a qualquer momento. No entanto, a utilização de 'uicontrols' diretamente no script .m é muito mais complicada do que utilizando a ferramenta GUIDE, pois é necessário configurar manualmente todos os parâmetros do objeto, tais como coordenadas, tamanho, estilo, em que janela ele irá aparecer, etc. Por exemplo, para escrever a palavra UNICAMP em uma janela (supondo que essa janela já exista e esteja previamente configurada) deve-se fazer como no exemplo abaixo:

```
uicontrol(h_tese,'Units','normalized','Style','text',...
    'FontWeight','Bold','Position',[0.4 0.5 .2 .1],...
    'String','UNICAMP','fontsize',16,'HorizontalAlignment','left');
```

Para se criar uma figura simples para este texto, sem muita formatação, e do tamanho padrão do Matlab, faz da seguinte maneira:

```
%Cria a figura
h_tese=figure;

%poe a cor do sistema na figura
set(h_tese,'Color',get(0,'defaultUicontrolBackgroundColor'),...
    'MenuBar','none','NumberTitle','off',...
    'name','Figura para tese','DeleteFcn',['clear h_tese']);
```

Juntando-se as duas seqüências de comando acima (primeiro criando a figura e depois adicionando o texto), tem-se a janela da Figura 5.2.

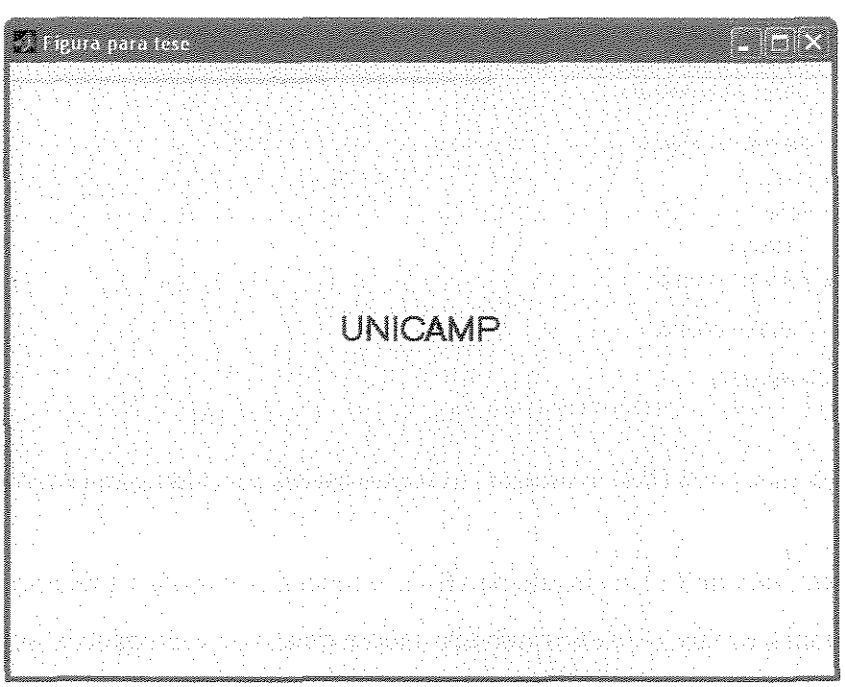


Figura 5.2: Exemplo de figura criada em Matlab usando-se objetos uicontrol

### 5.2.5 Alternando entre as variáveis da memória

LabSis permite ao usuário trabalhar com diferentes dados sísmicos, sem que seja necessário executar o programa novamente, salvar o dado em um arquivo em disco, ou copiá-lo manualmente para uma variável temporária. Para isso, basta selecionar no menu dado, dentro de uma lista que contém o nome de todos os dados sísmicos carregados na memória, o nome referente ao dado sísmico que se deseja trabalhar, a variável que está sendo usada atualmente, é assinalada com um símbolo ✓na frente de seu nome (figura 5.3), possibilitando assim, ao usuário, alternar facilmente entre diferentes dados sísmicos.

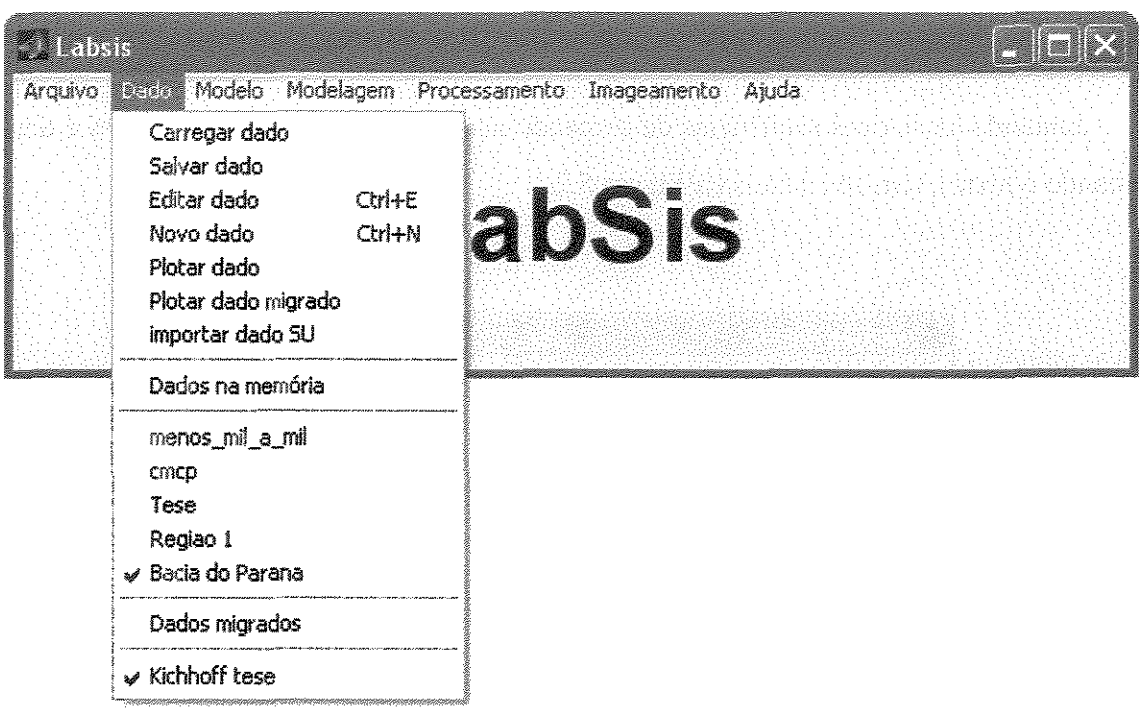


Figura 5.3: Tela inicial do LabSis mostrando os menus usados para alterar entre as variáveis da memória

Programar uma tarefa tão singela não foi um procedimento tão fácil, apesar de que a solução pode parecer bem simples. Toda vez que um dado é criado ou carregado, o seu nome aparece em uma lista no menu dado. O dado que está sendo usado atualmente fica armazenado em uma variável interna, *s*, enquanto todos os outros dados sísmicos que estão na memória, mas não estão sendo usados ficam armazenados em variáveis secundárias *s1*, *s2*, *s3*, *s4*, ..., com um limite máximo de

dez.

Essas variáveis secundárias são dados sísmicos no formato LabSis. Toda vez que o usuário desejar alterar de uma variável $X$, que atualmente está sendo utilizada, para uma variável $Y$ armazenada na memória em uma variável secundária, o dado $X$ é copiado para uma variável secundária que esteja vazia e a variável $s$ é deletada, e somente então o dado $Y$ é copiado para uma nova variável $s$. Permitindo assim uma troca entre o dado armazenado na memória e dado atualmente em uso.

A principal dificuldade, aqui reside no que para os olhos do usuário pode parecer uma operação simples. Para que o usuário saiba qual variável está sendo usada, é colocado um símbolo ✓na frente do nome deste dado, como visto na Figura 5.3. Para executar tal tarefa, todas as vezes que o usuário altera de uma variável em uso para outra residente na memória, uma função deve colocar um símbolo ✓a frente da variável que está sendo selecionada, e verificar todas as outras variáveis que estão armazenadas na memória. Como o LabSis trabalha com um total máximo de dez variáveis, todas devem ser checadas e, para isso usa-se o procedimento abaixo para cada um dos menus (total de dez):

```
menu_s1=uimenu(menu_dado,'Separator','on','visible','off',...
      'label','ss1','Callback',['s=s1;',...
      'set(menu_s1,''checked'',''on'');'...
      'set(menu_s2,''checked'',''off'');',...
      'set(menu_s3,''checked'',''off'');'...
      'set(menu_s4,''checked'',''off'');',...
      'set(menu_s5,''checked'',''off'');'...
      'set(menu_s6,''checked'',''off'');',...
      'set(menu_s7,''checked'',''off'');'...
      'set(menu_s8,''checked'',''off'');',...
      'set(menu_s9,''checked'',''off'');'...
      'set(menu_s10,''checked'',''off'');']);
```

Tal procedimento é executado toda vez que o programa se inicia, residindo na memória para

ser executado sempre que uma variável é selecionada no menu.

Quando uma nova variável é criada, ela deve ser introduzida no menu, indicando que está residente na memória e deve ser copiado para variável secundária. O procedimento é feito de maneira semelhante ao anterior. A função chamada `acertamenus` verifica, através de um contador, qual a posição de memória que está vazia e coloca o dado nesta posição. No exemplo abaixo, tem-se parte da função `acertamenus` para o caso em que a segunda posição de memória está vazia, ou seja, tem-se apenas um dado na memória, enquanto o segundo está sendo acrescentado.

```
case 2
    s2=s;
    set(menu_s2,'label',s.nome,'visible','on','checked','on');
    set(menu_s1,'checked','off');
    set(menu_s3,'checked','off');
    set(menu_s4,'checked','off');
    set(menu_s5,'checked','off');
    set(menu_s6,'checked','off');
    set(menu_s7,'checked','off');
    set(menu_s8,'checked','off');
    set(menu_s9,'checked','off');
    set(menu_s10,'checked','off');

end
```

### 5.2.6 Caixas de diálogo no LabSis

Caixas de diálogo são usadas para alertar usuários sobre alguma operação interna do programa, de modo que usuário possa tomar a decisão sobre que ação deve ser feita. No LabSis as caixas de diálogo são usadas principalmente para proteger o usuário de, inadvertidamente, provocar uma ação que leve a uma interpretação errônea sobre o que está fazendo, ou até a uma falha na execução do programa, a qual pode ocasionar uma possível perda de dados.

Por exemplo, no LabSis, quando o usuário entra com dados em caixas de texto, onde deveriam ser digitados apenas números reais, uma função valida os campos preenchidos. Esta função verifica se o usuário acidentalmente digitou caracteres ilegais para tal campo no LabSis, tais como *strings* ou qualquer outro dado que não seja um número real. Se usuário cometeu tal infração, uma caixa notificação é mostrada na tela, e a função que está sendo executada é interrompida para que o campo possa ser corrigido. No LabSis esse procedimento é feito da seguinte maneira:

```
if isnan(s.xsini)
    errordlg('Voce deve entrar apenas com valores numéricos',
    'Entrada Inválida','modal')
    break;
end
```

Cujo resultado é a caixa mostrada na Figura 5.4.

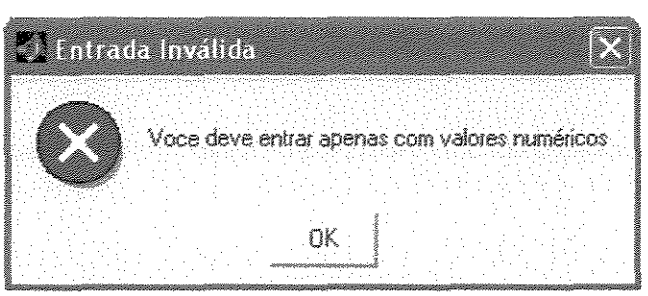


Figura 5.4: Exemplo de caixa de diálogo mostrada quando usuário entra com um valor ilegal em um campo do LabSis.

No LabSis, caixas de diálogo também são usadas para selecionar arquivos a serem salvos ou carregados, de modo que o usuário possa selecionar tal arquivo sem a necessidade de usar a linha de comando. O programador pode escolher qual o filtro de arquivos será usado e qual o título da janela. No LabSis essa operação é feita usando como base a linha de comando a seguir, que é um dos passos da função carregar.

```
if ext == 1
    [arqname,pathname] =
       uigetfile({'*.mat','Dados (*.mat)';'*.*','Todos (*.*)'},...
       'LabSis - Selecione Arquivo de Dados');
end
```

O resultado da função usando a seqüência de comandos acima, produz a janela da Figura 5.5.

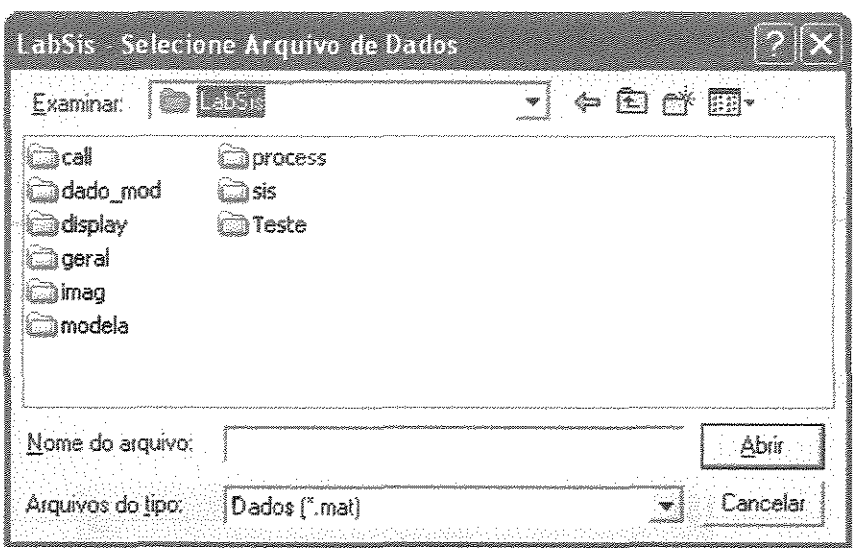


Figura 5.5: Caixa de diálogo do LabSis usada para carregar arquivos sísmicos no formato do programa.

### 5.2.7 Bloqueando Caixas de Texto Editáveis no LabSis

Na janela para a criação de uma nova geometria no LabSis, o usuário pode escolher entre os arranjos CO, CS, CMP e ZO. Os arranjos podem ser selecionados através de um menu pop-up. Cada vez que um determinado arranjo é selecionado, as caixas editáveis são bloqueadas ou não, para que a geometria correta seja inserida. Por exemplo, na Figura 5.6, quando é selecionado o arranjo *Common Shot*, onde há apenas uma fonte e vários receptores, a caixa editável correspondente ao incremento do vetor de coordenadas da fonte (dxs na figura) é bloqueada. Com a caixa bloqueada, o usuário não pode digitar nada neste campo, impedindo assim possíveis erros de entrada de dados.

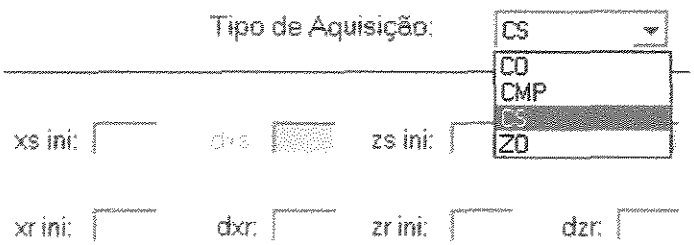


Figura 5.6: Exemplo mostrando como a seleção de geometrias bloqueia ou desbloqueia as caixas de texto editáveis no LabSis.

Essa operação torna-se possível da seguinte forma: cada vez que o usuário seleciona uma opção no menu pop-up, é executada uma função que habilita ou não a caixa de texto editável correta para o arranjo. Tal procedimento aparentemente simples, demanda uma solução algorítma considerável, como se pode observar no código abaixo.

```
val = get(pop_geom,''Value'');

⋮

    if val ==3,set(text_dxs,'enable','off');
        set(Edit_dxs,'enable','off','BackgroundColor',[.84 .81 .78]);
        set(text_dxr,'enable','on');
        set(Edit_dxr,'enable','on','BackgroundColor','white');
        set(text_dzs,'enable','off');
        set(Edit_dzs,'enable','off','BackgroundColor',[.84 .81 .78]);
        set(text_dzr,'enable','on');
        set(Edit_dzr,'enable','on','BackgroundColor','white');
    else
        set(text_dxs,'enable','on');
        set(Edit_dxs,'enable','on','BackgroundColor','white');
        set(text_dxr,'enable','off');
        set(Edit_dxr,'enable','off','BackgroundColor',[.84 .81 .78]);
        set(text_dzs,'enable','on');
```

```
        set(Edit_dzs,'enable','on','BackgroundColor','white');
        set(text_dzr,'enable','off');
        set(Edit_dzr,'enable','off','BackgroundColor',[.84 .81 .78]);
    end;
```

⋮

O exemplo mostrado acima, trata-se apenas de parte da função, sendo unicamente para o caso em que o usuário seleciona a opção CS. Um código semelhante deve ser escrito para as demais opções.

### 5.2.8 Plotagem do Modelo Geológico

Para facilitar a construção e importação de modelos geológicos no LabSis, o usuário pode visualizar, em uma janela gráfica, o modelo corrente na memória e, através do menu plotar modelo geológico. Para executar tal função, primeiramente determina-se o tamanho dos vetores de dados correspondentes a cada camada, para ajustar os valores mínimos e máximos dos eixos x e z. Através do comando em 'fill' Matlab, as camadas são preenchidas, uma a uma, a partir da superfície.

Para se determinar a cor de preenchimento da estratigrafia, foi criado um *script* que utiliza uma seqüência de dez cores. Para isso, o programa determina o número de camadas do modelo e para cada camada usa uma cor de preenchimento diferente, impedindo que a mesma cor seja repedida em uma camada próxima. Uma determinada cor só irá se repetir depois que todas as outras dez cores forem utilizadas. O código abaixo exemplifica de maneira clara o funcionamento de tal função.

```
for i=1:dadomod.ncamadas
    k=i;
    if k>10
        while k>10
```

```
            k=k-10;
        end
    end

    switch k
        case 1
            cor=amarelo;
        case 2
            cor=verde;
        case 3
            cor=vermelho;
        case 4
            cor=bege;
        case 5
            cor=branco;
        case 6
            cor=amarelo;
        case 7
            cor=verde;
        case 8
            cor=rosa;
        case 9
            cor=roxo;
        case 10
            cor=branco;
    end

    fill([dadomod.x,dadomod.x(nx:-1:1)],
       [dadomod.z(i,:),zf*ones(size(dadomod.x))],cor);
end
```

### 5.2.9 Importar Modelos Geológicos do InterSis

O InterSis salva seu modelo geológico em um arquivo texto padrão ASCII onde ficam armazenadas todas as informações referentes ao modelo geológico (número de camadas, coordenadas de interfaces, velocidades, etc). Esse arquivo não é somente uma matriz de dados, que poderia ser simplesmente lida com o comando *load -ascii* do Matlab. Esse arquivo, inicialmente continha uma série de informações de cabeçalho, os quais dificultavam a criação de um *script* para leitura em Matlab. Para isso, foi padronizado, junto ao autor do InterSis, o formato de arquivo. Abaixo segue um exemplo de formato de arquivo de modelo geológico InterSis padronizado.

```
Graphic interface for Seis88 and Virieux Finite diference method
version 0.2 Name :Synthetic model Interfaces  4   [including top
and bottom] Int ptos    x[km]    z[km]    type
  1   2
       0.00000      0.01000  SMOOTH
      14.00000      0.01000  CORNER
  2   2
       0.00000      1.04000  SMOOTH
      14.00000      1.04000  CORNER
  3   2
       0.00000      2.04000  SMOOTH
      14.00000      2.04000  CORNER
  4   2
       0.00000      4.00000  CORNER
      14.00000      4.00000  CORNER
Velocity type :GRADIENT [velocity] [vertical/gradx] [gradz] [refer
x] [refer z]
       2.30000    0.00300    0.50000    4.28615    0.28571
       2.64500    0.00900    0.50000    6.76308    1.58571
       3.04175    0.00300    0.50000    3.01538    2.81429
User :aeco  date :Wed Aug 27 21:07:39 2003
```

No arquivo listado acima encontram-se informações sobre as coordenadas dos pontos das interfaces e velocidades das camadas. Como se pode ver, o formato padrão não é uma matriz de dados simples que possa ser facilmente carregada; no entanto, este é o formato usado que mantem a compatibilidade de ambos os softwares.

Para ler o arquivo de modelo geológico usou-se os comandos em Matlab *fgets* e *fgetl*, os quais lêem uma determinada linha de um arquivo ASCII e pulam automaticamente para a linha seguinte. Esses comandos, lêem a linha como sendo um vetor de strings, incluindo todos os espaços em branco. Assim, para ler a linha 3 do exemplo acima, a mesma fica armazenada em um vetor com vinte e dois elementos (ele adiciona um espaço em branco ao final de cada linha). Entretanto, a variável que interessa ao LabSis é somente o nome do modelo, e não toda a linha. Para obter somente a parte necessária, faz-se:

```
a=fgets(fid); var.name=a(7:21);
```

O LabSis considera o número exato de caracteres da linha para armazenar na variável, dessa maneira torna-se necessário a padronização do formato do arquivo de modelo.

A partir da quinta linha do arquivo, não se pode empregar o mesmo processo usado acima, isso porque a posição das strings no vetor vai variar conforme o número de casas decimais do dado (ex. o número 1 vai ocupar apenas uma posição do vetor, mas o número 153 vai ocupar três casas decimais). Como os dados estão separados por colunas, optou-se por usar a função *isspace* do Matlab, que verifica os espaços em branco, em um vetor de *strings*. Esta função retorna 1 se a posição é um espaço em branco e 0 se não é. Assim, a função *isspace* foi usada para identificar qual parte do vetor era o dado que deveria ser transformado em um número real.

Para explicar melhor como esse procedimento é feito, toma-se como exemplo a linha seis do exemplo acima:

⋮

```
0.00000      0.01000  SMOOTH
```

⋮

Essa verificação foi feita da seguinte maneira: A linha é lida no formato de um vetor. A função verifica se a posição 1 é um espaço em branco (de fato é), caso positivo ela passa para a posição 2 e assim por diante, até chegar na posição que contenha o zero. Essa posição inicial do dado é guardada em uma variável m. A função continua verificando as outras posições até encontrar um espaço vazio (a casa logo após o ultimo zero do número 0.00000). A posição anterior a esta é a posição final deste dado. De posse destes dois valores pode-se armazenar a parte adequada do vetor em uma variável como feita no exemplo anterior. E a função repete o processo para resto da linha.

Computacionalmente, o processo acima é feito como no exemplo abaixo:

```
a=fgets(fid);
k=1;
while 1
   if isspace(a(k))==0
    break
  end
   k=k+1;
end
m=k;
while 1
    if isspace(a(m))==1
       break
    end
    m=m+1;
end
var.x(j)=str2double(a(k:m-1));
```

### 5.2.10 Mudança de Ganho na Janela de Dado Sísmico

Para plotar o dado sísmico no LabSis usando *wiggles*, foi utilizada a função *wigbkern*, que foi construída por Rodrigo Portugal. Essa função, tem como parâmetros de entrada a matriz com o dado sísmico, o vetor com os valores de tempo, o vetor com as coordenadas x dos receptores, e um parâmetro k para a escala do dado sísmico (cujo valor padrão é 1). Toda a vez que um novo dado sísmico é plotado, é usado o fator de escala padrão (1).

Para que o usuário possa mudar o ganho do dado sísmico, há um *slider* onde esse valor pode ser ajustado. Esse *slider* tem um valor mínimo de zero e um máximo de dez. Sempre que é acionado, ele coloca o valor da posição selecionada em item de texto ao lado, para facilitar o ajuste do ganho. Outra função que o slider desempenha quando acionado, é que ele plota novamente o dado na janela de sísmica usando a função *wigbkern*, mas dessa vez com o valor de ganho selecionado como entrada para o parâmetro k, na figura 5.7 tem-se o mesmo dado sísmico com diferentes ganhos.

Essa ferramenta tem a vantagem de permitir ao usuário alterar o ganho da janela de sísmica conforme sua necessidade. No entanto, como a ferramenta de mudança de ganho plota novamente o dado sísmico, usando a variável que atualmente está na memória, haverá uma limitação. Se o usuário plotar na janela 1 o dado sísmico A, em seguida mudar o dado na memória para o dado sísmico B. Caso o usuário resolva alterar o ganho da janela A, a função vai plotar novamente na janela o dado que está atualmente na memória, ou seja o dado sísmico B. Assim, ao invés de plotar novamente com um ganho diferente o dado A, o usuário estará plotando o dado B com o ganho selecionado. Esse problema pode ser resolvido alternando-se novamente a variável da memória para o dado A.

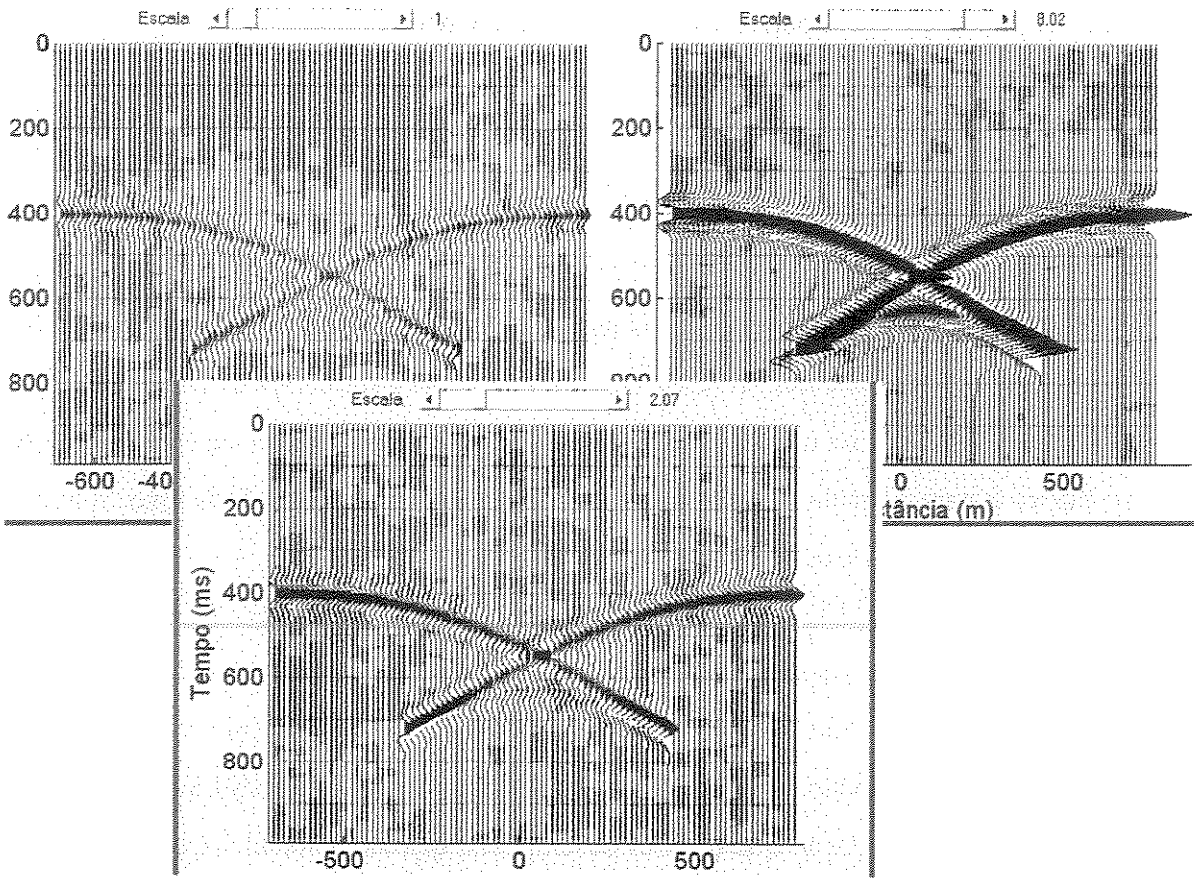


Figura 5.7: O mesmo dado sísmico com ganhos de 1,8 e 2 respectivamente.

## 5.3 Limitações Computacionais

Nesta seção consideramos as limitações computacionais atuais do LabSis, mostrando todas as características de cada limitação, bem como as suas causas.

### 5.3.1 Número máximo de variáveis na memória

Guardar na memória as variáveis de dado sísmico que estão sendo utilizadas ou, trabalhar com várias variáveis simultaneamente, exige um código mais elaborado (como explicado na Seção 5.2.5).

Por esse motivo, o LabSis trabalha com um máximo de dez variáveis para dados sísmicos simultaneamente na memória e um máximo de dez posições para variáveis de dados sísmicos migrados (totalizando vinte variáveis),além de uma posição para variável contendo os parâmetros do modelo geológico.

### 5.3.2 Ganho de amplitude na janela de sísmica

O ganho de amplitude sísmica usado no LabSis na verdade plota novamente o dado atual na memória de acordo com o ganho selecionado (maiores explicações na seção 5.2.10). Por esse motivo, ele só funciona para a variável que está selecionada. Caso haja uma janela com dado sísmico que está na memória mas não é o atualmente selecionado, haverá um erro ao tentar ajustar o ganho. Para evitar esse problema, o usuário deve alternar entre as variáveis da memória para aquela que contém o dado sísmico da janela que se deseja ajustar o ganho, selecionando, no menu dado, a variável desejada.

Outra limitação desta ferramenta é que, como ela plota o dado novamente, a operação de mudança de ganho pode se tornar uma operação muito lenta, quando se está trabalhando com dados sísmicos muito pesados, isso porque o programa executa novamente as funções para plotagem de dados sísmicos a cada mudança de ganho.

### 5.3.3 Modelo geológico

Como a intenção do LabSis é ser um software integrado com o InterSis, o módulo de construção de modelo geológico é limitado, deixando essa tarefa para o InterSis, que esse já possuí uma ferramenta mais poderosa para essa função.

Dessa maneira, no LabSis só é permitida a construções de modelos estratigráficos simples com camadas horizontais planas. Qualquer modelo mais complexo deve ser criado no InterSis e

importado no LabSis.

Pela mesma razão, a edição dos modelos geológicos através do menu *Editar modelo geológico*, só é permitida para modelos criados no LabSis, ou que obedeçam o padrão de camadas horizontais planas.

### 5.3.4 Salvar o dado LabSis

Para agilizar o processo de carregamento, o dado sísmico no formato LabSis é gravado em disco no formato binário e não ASCII. Esse dado, contém uma série de informações adicionais que impossibilitariam salvar o dado como um arquivo XYZ ASCII, com colunas devidamente delimitadas como uma matriz.

Por essa razão optou-se por salvar o dado sísmico do LabSis no formato .mat, que é um arquivo binário com formato padrão do Matlab.

### 5.3.5 Limitações impostas por funções usadas no LabSis

Como o LabSis é um pacote constituído por um conjunto de várias funções externas, criadas por programadores diferentes, além das limitações citadas acima, existem no pacote as limitações impostas pelas funções externas incorporadas ao LabSis. Tais como por exemplo:

- As funções para modelagem que utilizam algoritmos de Kirchhoff, Born e traçado de raios, aceitam qualquer forma para a interface refletora, mas no entanto são limitadas a modelos com apenas um refletor. Caso o usuário utilize um modelo com mais refletores, o LabSis envia ao usuário uma mensagem alertando que apenas a primeira camada foi considerada.

- A função CmpHoriz, para modelagem, também por traçado de raios, não tem limitação no número de camadas, podendo usar tantos refletores quanto forem necessários. No entanto,

há uma limitação quanto a forma destes refletores, pois somente são aceitos camadas planas horizontais.

## 5.4 Como introduzir uma nova função no LabSis

No decorrer desta dissertação foi mencionado constantemente a facilidade de introduzir-se uma nova função no LabSis. Deve-se entender no entanto que essa facilidade está relacionada com o nível de conhecimento em Matlab do programador. O objetivo desta seção é explicar detalhadamente o processo de introdução de uma nova função. A melhor maneira de expor essa facilidade é com um exemplo prático.

Para ilustrar esse procedimento, usou-se a função *crossplot.m* de autoria de Rodrigo de Souza Portugal (IG/Unicamp). Essa função permite ao usuário, selecionar uma determinada região entre dois dados sísmicos e, fazer um *crossplot* entre os pontos, na região selecionada, mostrando uma série de informações estatísticas sobre estes pontos.

A função *crossplot.m* exigiu um trabalho um pouco maior, para ser introduzida no LabSis, do que as outras funções introduzidas anteriormente. Esse esforço deveu-se em parte, como será mostrado a seguir, pela própria estrutura do LabSis.

### 5.4.1 Executando a função

O primeiro passo é conhecer a função, analisá-la, e executá-la usando dados do LabSis como parâmetros de entrada. Para isso, basta analisar o help da função, de onde tira-se as seguintes informações:

```
>> help crossplot

 CROSSPLOT performs a crossplot of two data set
```

```
crossplot(a, b, x, t, tmin, tmax)
INPUT| DESCRIPTION                          |  TYPE   | SIZE       | DEFAULT
 a   | first data                          | matrix  | (nx x nt)  |    -
 b   | second data                         | matrix  | (nx x nt)  |    -
 x   | horizontal coordinates (cdp [m])    | array   | ( 1 x nx)  |    -
 t   | vertical coordinates (time [ms])    | array   | ( 1 x nt)  |    -
 ti  | vertical coordinate begin clip      | real    | ( 1 x 1 )  |  min(t)
 tf  | vertical coordinate end clip        | real    | ( 1 x 1 )  |  max(t)

USES
    fitselec, getline, inpolygon

>>
```

Como pode-se observar acima, em todos os parâmetros da função podem ser usados dados já pertencentes ao LabSis. Dessa maneira, usando-se duas varáveis internas no formato LabSis, s1 e s2, e que já residem na memória, pode-se fazer a seguinte correlação: a = s1.data, b= s2.data, x = s1.xr, t = s1.t, os dois parâmetros finais, ti e tf ficam a critério do usuário.

Dessa, maneira, usando-se ti = 0 e tf = 2000ms, pode-se executar a função através do comando: crossplot(s1.data, s2.data, s1.xr, s1.t, 0, 2000). O resultado pode ser conferido nas Figuras 5.8 e 5.9. Mostrando que é possível executar a função utilizando as varáveis do LabSis. Resta agora fazer uma função casca que execute o comando anterior, de uma maneira mais amigável ao usuário.

### 5.4.2 Criando a função casca

A função casca vai servir de ponte entre o usuário e a execução da função. Os parâmetros que o usuário deverá escolher serão: quais, entre as variáveis da memória, serão escolhidas e, quais os tempos, máximo e mínimo, a serem exibidos no gráfico. Para isso, é necessária uma interface gráfica onde o usuário escolherá estes parâmetros. O código desta interface gráfica deve constar

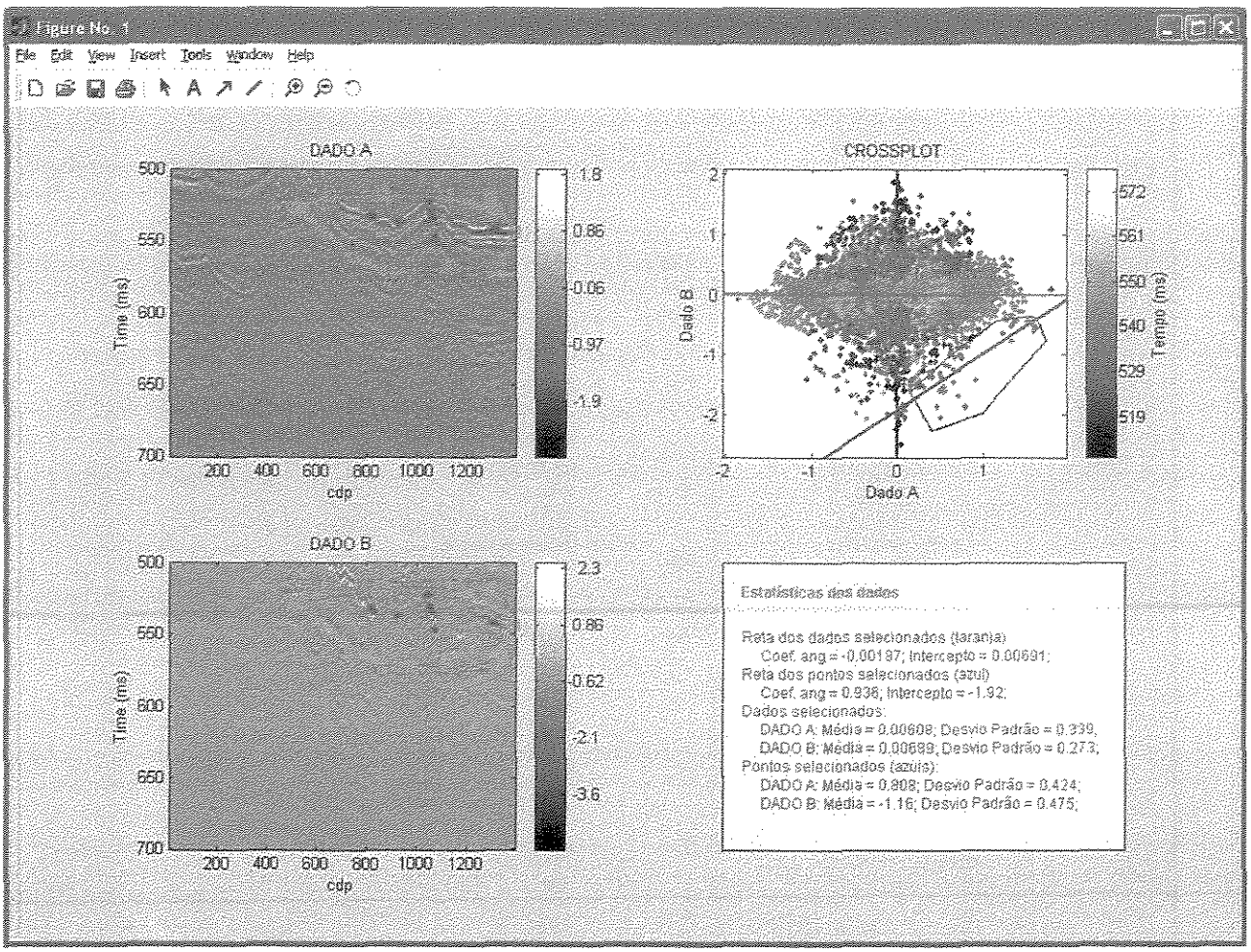


Figura 5.8: Janela 1 da função crossplot.

na função casca.

Para a interface gráfica, primeiramente deve-se criar a figura, retirar os menus padrões do Matlab que não serão utilizados, colocar o título do LabSis e colocar a cor padrão do sistema operacional na figura. Esses passos estão descritos no código abaixo:

```
% Cria a figura
h_cross=figure;

% tira o menu defaut do matlab
```

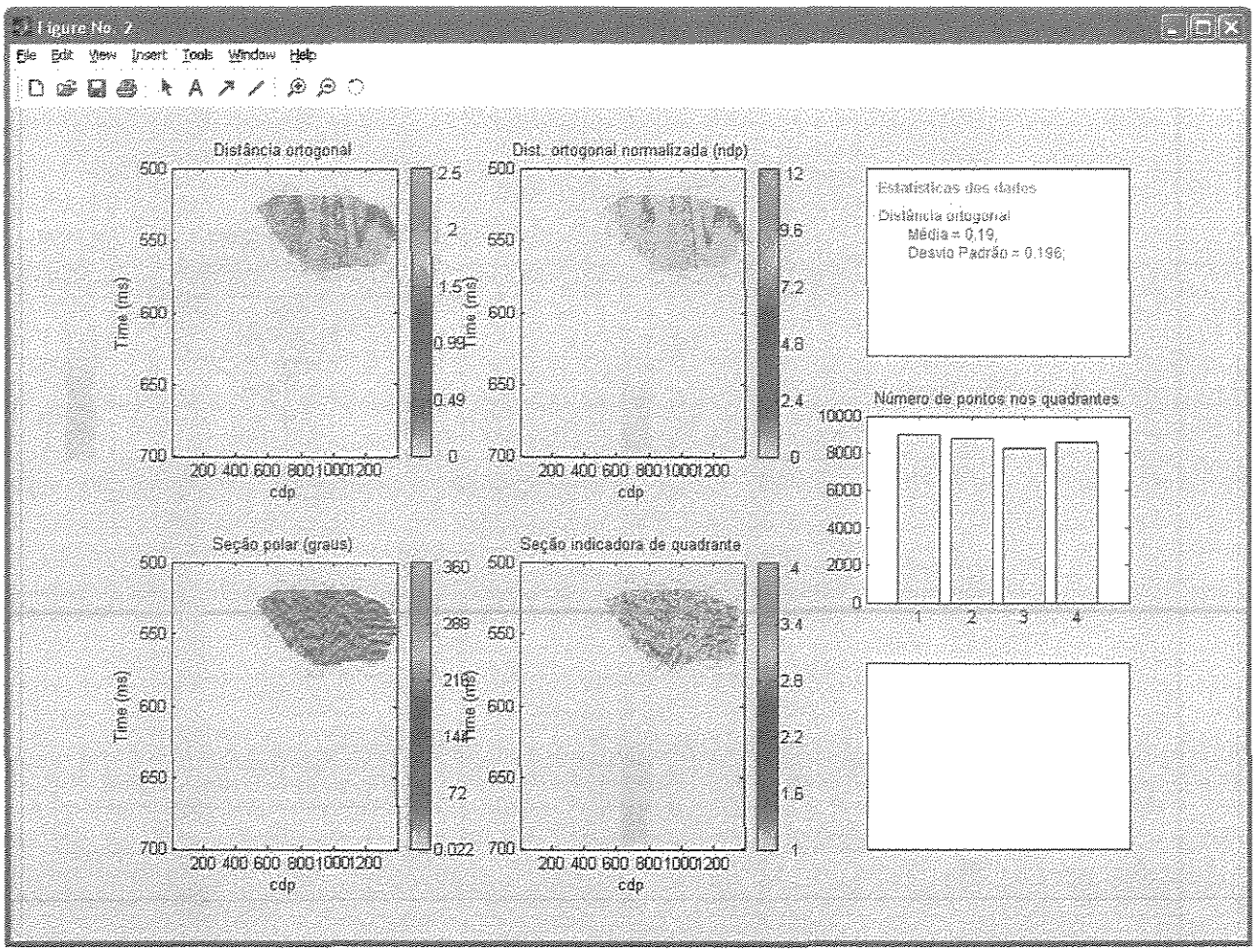


Figura 5.9: Janela 2 da função crossplot.

```
set(h_cross,'MenuBar','none','NumberTitle','off','name','Labsis -
Parâmetros para o crossplot');

% poe a cor do sistema na figura
set(h_cross,'Color',get(0,'defaultUicontrolBackgroundColor'));
```

Com isso, o Matlab cria uma figura chamada h_fig, com os parâmetros citados acima. Para melhorar a estética da janela, é interessante reduzir o tamanho padrão que o Matlab impõe a um tamanho mais aceitável. O novo tamanho e posição da janela é calculado baseado na posição e no tamanho da janela principal do LabSis, como o código a seguir:

```
%ajusta tamanho e posiçao da janela em relaçao a janela do LabSis
pos=get(h_labsis,'Position');
    x_pop=pos(1)+(pos(3)/2)-200;
    y_pop=pos(2)+(pos(4)/2)-125;
set(h_cross,'Position',[x_pop y_pop 500 350]);
```

Agora basta adicionar os objetos como textos, caixas de textos e menus, segundo o código a seguir:

```
uicontrol(h_cross,'Units','normalized', 'Style','text',...
 'Position',[0.07 0.82 .15 .05],'String','Variável 1:','fontsize',10);

pop_var1 =
uicontrol(h_cross,'Units','normalized','Style','pop',...
 'Position',[.30 .82 .2 .05],'String',nomes,'BackgroundColor','white');

uicontrol(h_cross,'Units','normalized', 'Style','text',...
 'Position',[0.07 0.62 .15 .05],'String','Variável 2:','fontsize',10);

pop_var2 =
uicontrol(h_cross,'Units','normalized','Style','pop',...
 'Position',[.30 .62 .2 .05],'String',nomes,'BackgroundColor','white');

uicontrol(h_cross,'Units','normalized', 'Style','text',...
 'Position',[0.07 0.42 .2 .05],'String','Tempo mínimo:','fontsize',10);

Edit_tmin=uicontrol(h_cross,'Units','normalized',
'Style','edit',...
    'Position',[0.32 0.42 .15 .06],'BackgroundColor','white');

uicontrol(h_cross,'Units','normalized', 'Style','text',...
 'Position',[0.07 0.22 .2 .05],'String','Tempo máximo:','fontsize',10);

Edit_tmax=uicontrol(h_cross,'Units','normalized',
```

```
'Style','edit',...
 'Position',[0.32 0.22 .15 .06],'BackgroundColor','white');

but_ok =
uicontrol(h_cross,'Units','normalized','Style','pushbutton',...
 'Position',[.7 .1 .2 .12],'String','OK');
```

O resultado do que foi construído até agora pode ser conferido pela Figura 5.10.

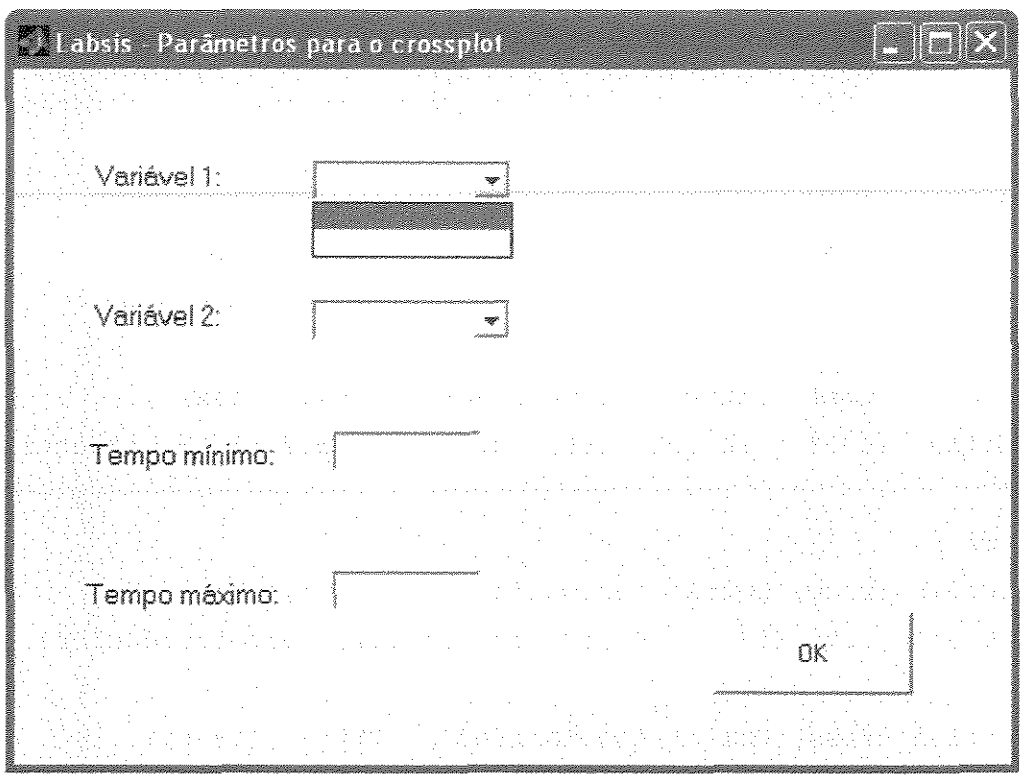


Figura 5.10: Janela gráfica contendo os objetos necessários para a execução da função crossplot.

A programação feita até agora foi apenas "maquiagem", servindo apenas para a criação da parte gráfica do programa. Resta agora tornar esta função operacional, e integrá-la com a função crossplot.

Residente na memória do LabSis, existe a variável chamada *contval*. Ela diz quantas variáveis sísmicas o LabSis tem em sua memória, ela é utilizada para adicionar, no menu dado, o número

de entradas necessárias para as variáveis na memória, além de servir para o programa, internamente, conferir se o número de variáveis sísmicas na memória, não ultrapassou o máximo permitido (maiores informações na Seção 5.2.5). De posse dessa variável, pode-se criar um vetor contendo o nome de todas as variáveis da memória da seguinte maneira:

```
for i=1:contval-1
   nomes(i)=cellstr(eval(strcat('s',int2str(i),'.nome')));
end
```

O loop acima concatena a string `s` com o valor da variável `i` e a extensão `.nome`, dessa forma, para `i=1` por exemplo tem-se `s1.nome`. O comando *eval* executa a string como se fosse um comando. O resultado é um vetor chamado nomes, contendo os nomes de todas as variáveis sísmicas existentes na memória. Note que no código das variáveis `pop_var1` e `pop_var2`, no campo string, encontra a variável nomes. Isso significa que o conteúdo do vetor nomes aparecerá na janela, sobre os menus popups em forma de uma lista.

Agora resta adicionar os *callbacks*, ou seja, as funções que serão executadas quando um dos objetos for chamado. Como essas funções são simples, elas não foram adicionadas em arquivos separados, e sim diretamente ao final do script principal, como visto abaixo:

```
set(pop_var1,'Callback',['var1=get(pop_var1,''value'');',...
   'set(Edit_tmin,''string'',',...
   'min(eval(strcat(''s'',int2str(var1),''.t''))));',...
   'set(Edit_tmax,''string'',',...
   'max(eval(strcat(''s'',int2str(var1),''.t''))));']);

set(pop_var2,'Callback',['var2=get(pop_var2,''value'');',...
   'if var1== var2,warndlg(''As variáveis 1 e 2 são iguais.'',',...
   '''Aviso!'',''modal'');end;']);
```

No código acima, tem-se o *callback* do menu `pop_var1`, onde cada vez que uma variável é selecionada, os tempos máximos e mínimos desta variável são mostrados nas caixas de texto

Edit_tmin e Edit_tmax, respectivamente. Além do *callback* do menu pop_var2, onde sempre que a mesma variável que foi selecionada, no primeiro menu popup, for também selecionada no segundo menu, uma mensagem de aviso é mostrada alertando o usuário. Para encerrar a função casca, só resta o *callback* do botão de OK, como pode-se observar no código a seguir.

```
set(but_ok,'Callback',['d1=(eval(strcat(''s'',int2str(var1),''.data'')));',...
        'd2=(eval(strcat(''s'',int2str(var2),''.data'')));',...
        'x=(eval(strcat(''s'',int2str(var1),''.xr'')));',...
        't=(eval(strcat(''s'',int2str(var1),''.t'')));',...
        'tmin=str2double(get(Edit_tmin,''string''));',...
        'tmax=str2double(get(Edit_tmax,''string''));',...
        'crossplot(d1, d2, x, t, tmin, tmax);delete(h_cross)']);
```

A função acima correlaciona os parâmetros sísmicos do LabSis com os necessários para a execução da função crossplot e, em seguida inicia esta função. Ao final, a janela é fechada através do comando `delete(h_cross)`. Neste ponto, a função está completa e integrada com o LabSis e pode ser chamada através de linha de comando. O único detalhe que resta é adicionar uma entrada de menu para a nova função, na janela principal do LabSis, que pode ser feito adicionando-se a linha que segue abaixo, ao arquivo labsis.m.

```
uimenu(menu_processamento,'Label','AVO -  Crossplot',...
'Separator','off','callback','crossplotsis;');
```

# Capítulo 6

# O Programa

Como foi mantida a intenção de não alterar qualquer uma das funções existentes para se preservar o trabalho de seus autores, optou-se por construir funções casca para cada uma das funções. Desse modo, o usuário não necessita entrar com o mesmo dado de diferentes maneiras, para as diferentes funções. O usuário entra apenas uma vez com o dado, que fica armazenado na memória do programa, no formato LabSis. Assim, quando necessário, este dado é convertido para o formado de entrada para a função solicitada, usando uma função casca específica. Um outro beneficio das funções casca é a facilidade de se introduzir novas funções. Para isso, basta (a grosso modo) construir uma função casca que transforme o padrão LabSis no padrão da função a ser introduzida. E, se necessário, construir uma interface para parâmetros de entrada adicionais.

Por ser escrito em Matlab, LabSis é um software multi-plataforma que roda em qualquer sistema que o suporte. Neste trabalho, ele foi devidamente testado em MS Windows e em Linux, mas provavelmente funcionará em qualquer sistema onde o Matlab possa a suportar. Além disso, é um programa de código aberto, permitindo melhorias futuras.

Para permitir uma melhor integração com outros softwares da área, LabSis pode importar arquivos de modelo geológico no formato InterSis, software implementado pelo Laboratório de Geofísica Computacional (IMECC) e, binários no formato SU.

Neste capítulo, é mostrado o funcionamento básico do LabSis, expondo todas as opções existentes

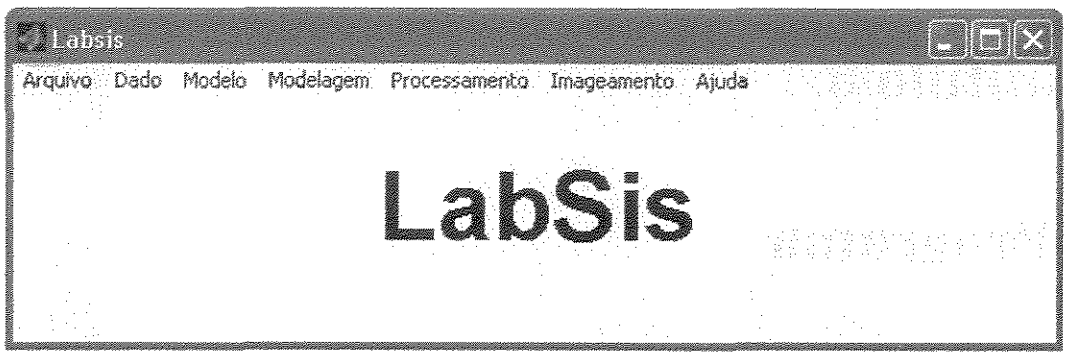


Figura 6.1: Tela inicial do LabSis

## 6.1 Módulo de dados sísmicos

Para se criar um novo dado a ser modelado, deve-se primeiramente entrar com os parâmetros geométricos que vão ficar armazenados na variável *s*, como uma espécie de cabeçalho, estes serão usados como parâmetros de entrada para as diferentes funções existentes. Para se introduzir esse parâmetros, deve-se clicar no menu 'Novo Dado', assim abre-se a janela para a introdução destes parâmetros, como visto na Figura 6.2.

Para a criação de um novo dado, deve-se primeiramente, entrar com um nome qualquer para o dado que está sendo criado. Este nome é usado apenas externamente para que o usuário possa trabalhar com diferentes dados ao mesmo tempo e alternar entre eles. Os nomes das variáveis criadas ficam armazenadas no 'menu dado', seção 'dados na memória'. A cada vez que o usuário seleciona um dado nesta seção, ele é automaticamente carregado para a variável interna *s* para que o programa possa utilizá-la.

O próximo passo é, preencher os campos restantes, onde o número de traços, corresponde ao número de traços sísmicos desejados. Para selecionar o tipo de aquisição, tem-se as opções CO,

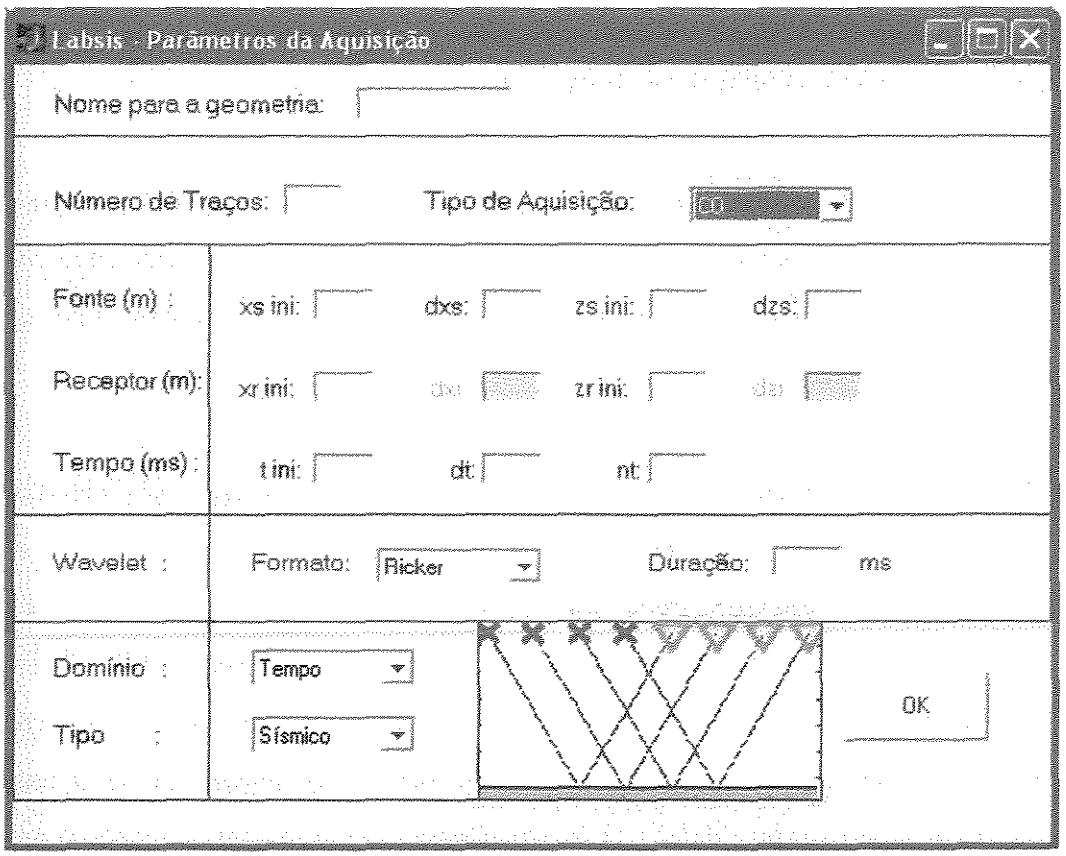


Figura 6.2: Tela para criação de novo dado.

CMP, CS e ZO, sendo que, a imagem referente ao tipo de aquisição, como no exemplo da Figura 6.2, é atualizada automaticamente mostrando um esquema do arranjo escolhido.

Para entrar com o vetor coordenadas x da fonte (xs), deve-se entrar com o valor inicial do vetor (xs ini) e o intervalo entre os pontos (dxs), assim será criado um vetor com número total de pontos, igual ao número de traços (exceto para CS) e com intervalo entre os pontos igual a dxs. O mesmo vale as coordenadas z da fonte (zs), coordenadas x dos receptores (xr) e coordenadas z dos receptores (zr). Para o tempo, o número máximo do vetor será dado pelo campo nt, o inicial por tini e o intervalo de amostragem por dt, seguindo o mesmo principio dos vetores anteriores.

Para o tipo da wavelet, pode-se escolher no menu *pop-up* entre as opções Ricker ou Gabor, essa e algumas outras opções deste tipo de menu não estão totalmente implementadas, mas estão no

menu para facilitar a introdução de funções futuras, que trabalhem com diferentes tipo de wavelets. O tamanho da wavelet é escolhido no quadro duração da wavelet.

Nos menus seguintes, pode-se escolher o domínio do dado com as opções tempo, freqüência, profundidade e Taup; e o tipo do dado, podendo ser sísmico, coerência, densidade, vagarosidade e velocidade. Com todos parâmetros preenchidos, basta clicar em OK, para que seja criado o cabeçalho do novo dado.

O usuário tem também a opção de salvar o dado atual ou carregar um dado salvo anteriormente no formato .mat, além de importar dados no formato SU.

Outra opção que facilita a operação com dados sísmicos, é o menu 'Plotar Dado'. Esse comando plota no formato *wiggle*, a matriz da variável *s.data*. O usuário pode também plotar, da mesma forma, dados migrados através do menu 'Plotar Dados Migrados', na Figura 6.3 há um exemplo de dado plotado no LabSis. O usuário também pode mudar o ganho da amplitude do dado sísmico, através do slider acima do dado.

Todo dado no formato LabSis, poder ser editado, sendo que é permitido ao usuário mudar os parâmetros estabelecidos anteriormente, para um reprocessamento. Para editar o dado da memória, basta acessar o submenu 'Editar Dado', onde será aberta uma janela como a da Figura 6.2, mas com todos os campos já preenchidos com os parâmetros do dado selecionado. Basta alterar o valor desejado e clicar no botão OK.

### 6.1.1 Importar dados do Seismic Unix

Para poder ter maior integração com outros pacotes de processamento sísmico, incluindo o InterSis, o LabSis pode importar dados sísmicos no formato Seismic Unix (SU). Tal tarefa é desempenhada pela função importasu, que é acessada através do submenu 'importar dado SU'.

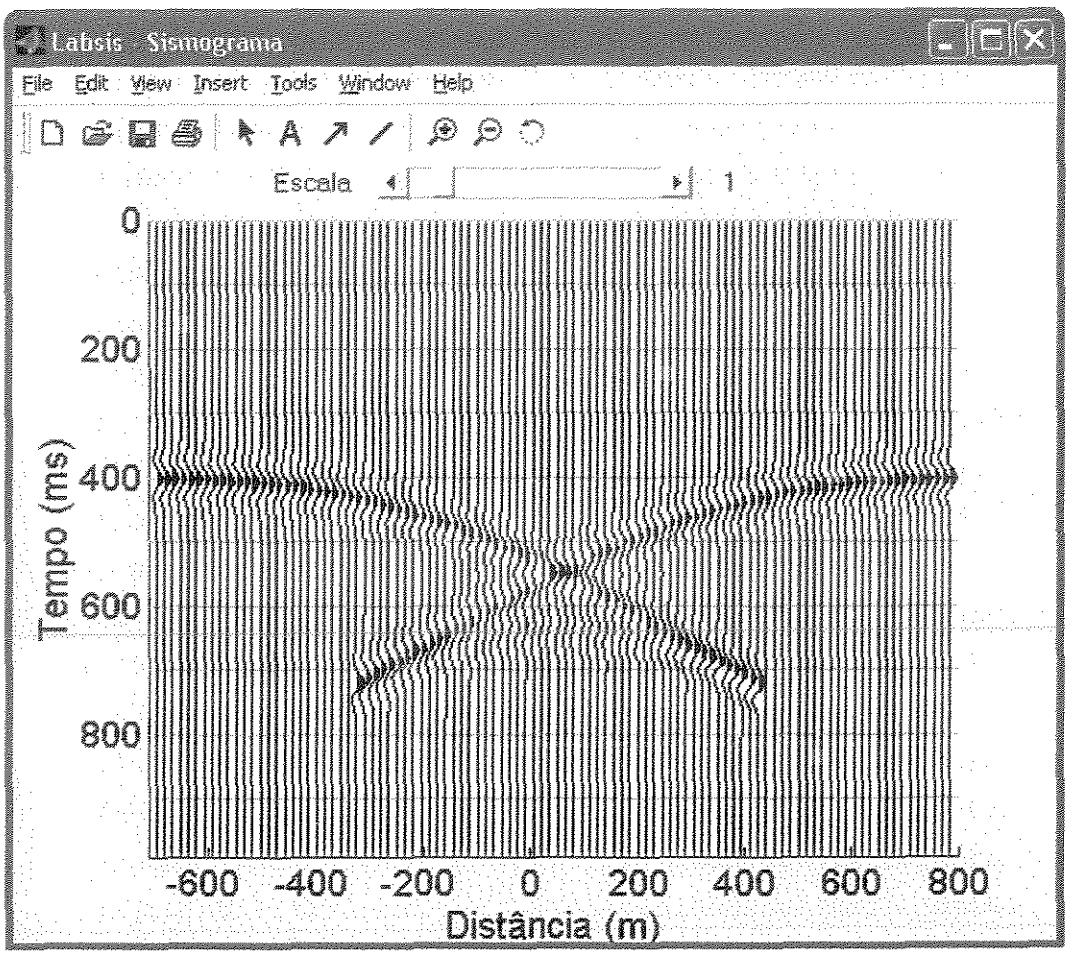


Figura 6.3: Exemplo de sismograma modelado e plotado pelo LabSis.

Essa função transforma o dado no SU para o formato LabSis, permitindo ao usuário escolher qual arranjo sísmico deseja extrair do dado.

Uma vez selecionado o dado SU, a função lê o arquivo e mostra a janela da Figura 6.4. Nesta janela são mostradas algumas informações sobre o arquivo, como: número de fontes, espaçamento entre as fontes, número de receptores, espaçamento entre os receptores e número de traços. Todas as informações do cabeçalho do arquivo ficam armazenada na variável *he*, que é uma variável do tipo *structure array*. A seguir tem-se um exemplo de variável he obtida a partir de um dado SU, gerado no InterSis, com 3600 traços.

```
he =
      tracl: [1x3600 double]        gain: [1x3600 double]
      tracr: [1x3600 double]           igc: [1x3600 double]
       fldr: [1x3600 double]           igi: [1x3600 double]
      tracf: [1x3600 double]          corr: [1x3600 double]
         ep: [1x3600 double]           sfs: [1x3600 double]
        cdp: [1x3600 double]           sfe: [1x3600 double]
       cdpt: [1x3600 double]          slen: [1x3600 double]
       trid: [1x3600 double]          styp: [1x3600 double]
        nva: [1x3600 double]          stas: [1x3600 double]
        nhs: [1x3600 double]          stae: [1x3600 double]
       duse: [1x3600 double]         tatyp: [1x3600 double]
     offset: [1x3600 double]         afilf: [1x3600 double]
      gelev: [1x3600 double]         afils: [1x3600 double]
      selev: [1x3600 double]        nofilf: [1x3600 double]
     sdepth: [1x3600 double]        nofils: [1x3600 double]
       gdel: [1x3600 double]           lcf: [1x3600 double]
       sdel: [1x3600 double]           hcf: [1x3600 double]
      swdep: [1x3600 double]           lcs: [1x3600 double]
      gwdep: [1x3600 double]           hcs: [1x3600 double]
     scalel: [1x3600 double]          year: [1x3600 double]
     scalco: [1x3600 double]           day: [1x3600 double]
         sx: [1x3600 double]          hour: [1x3600 double]
         sy: [1x3600 double]        minute: [1x3600 double]
         gx: [1x3600 double]           sec: [1x3600 double]
         gy: [1x3600 double]       timbas: [1x3600 double]
     counit: [1x3600 double]          trwf: [1x3600 double]
      wevel: [1x3600 double]       grnors: [1x3600 double]
     swevel: [1x3600 double]       grnofr: [1x3600 double]
        sut: [1x3600 double]       grnlof: [1x3600 double]
        gut: [1x3600 double]         gaps: [1x3600 double]
      sstat: [1x3600 double]        otrav: [1x3600 double]
      gstat: [1x3600 double]           d1: [1x3600 double]
      tstat: [1x3600 double]           f1: [1x3600 double]
```

```
laga: [1x3600 double]          d2: [1x3600 double]
lagb: [1x3600 double]          f2: [1x3600 double]
delrt: [1x3600 double]       ungpow: [1x3600 double]
muts: [1x3600 double]       unscale: [1x3600 double]
mute: [1x3600 double]          ntr: [1x3600 double]
  ns: [1x3600 double]         mark: [1x3600 double]
  dt: [1x3600 double]
```

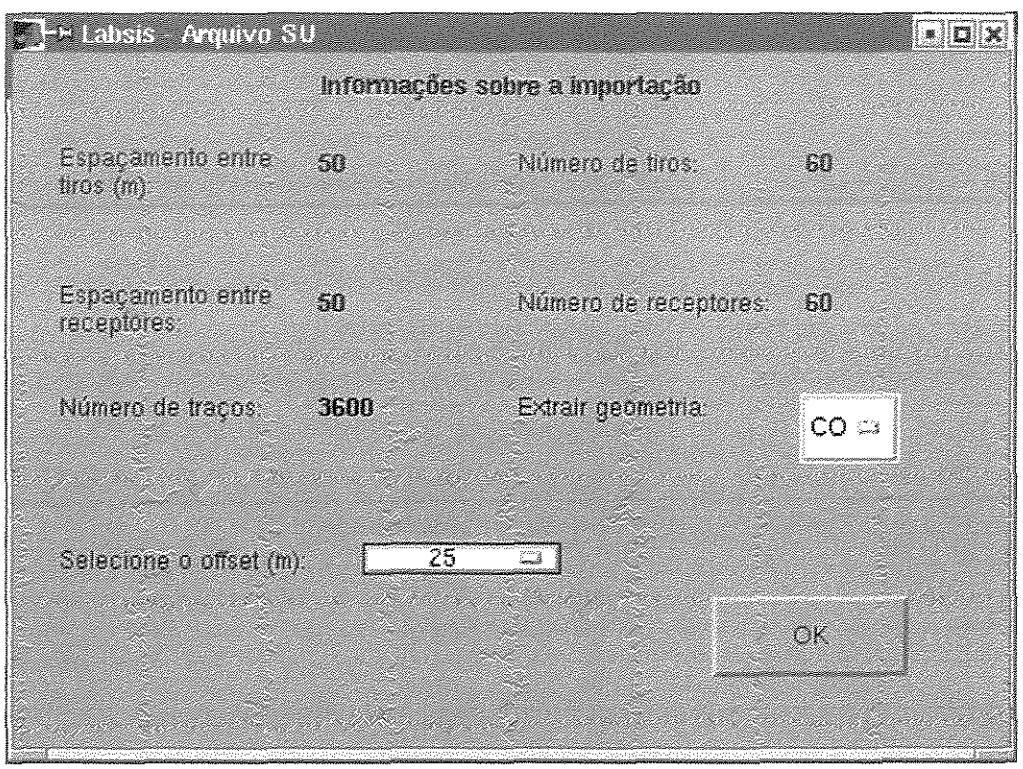


Figura 6.4: Janela com os parâmetros necessários para importar um dado no formato SU para o LabSis.

Na janela da Figura 6.4 também há um menu pop-up, onde pode-se selecionar qual arranjo sísmico deseja-se extrair do dado, atualmente as opções existentes são CO e CMP. Se for selecionada a opção CO, aparece outro menu mostrando todos os offsets possíveis para o dado. Após clicar em OK, o dado será importado usando-se o offset selecionado. Se for usada a opção CMP, o dado será importado com esse arranjo, utilizando a maior cobertura possível.

## 6.2 Módulo de Modelo Geológico

Depois de ajustados os parâmetros geométricos, o próximo passo é, criar um modelo geológico para a modelagem. Assim, os parâmetros referentes aos aspectos do modelo geológico ficam armazenados em uma variável do tipo *structure array* chamada de *modgeo*. Um exemplo de variável *modgeo* se encontra a seguir:

```
modgeo =

    type: 'multicshoriz' Tipo de modelo
      nx: 501            Número de pontos do vetor coordenada x
      dx: 4              Intervalo entre os pontos de coordenada x
    xini: -1000          Valor inicial do vetor coordenada x
       x: [1x501 double] Vetor coordenada x
ncamadas: 2              Número de camadas do modelo
     esp: [2x501 double] Matriz de espessuras
     vel: [2.5000 3]     Vetor com as velocidades das camadas
    dens: [2 3]          Valores de densidades das camadas
       z: [2x501 double] Vetor com as coordenadas z das interfaces
    name: 'saliência 2'  Nome para o modelo
```

No Laboratório de Geofísica Computacional, já existe uma interface poderosa que desempenha esta função de desenhar modelos geológicos, o nome deste programa é InterSis. A intenção deste trabalho não é construir um novo programa com a mesma função de um que já a desempenha muito bem. Portanto, pelo fato de já existir tal programa para a construção de modelos geológicos,o LabSis constrói apenas modelos com camadas planas horizontais. A construção de modelos mais complexos é feita no InterSis, e importada pelo LabSis.

Para a criação de um modelo geológico com camadas planas horizontais usando o LabSis, usa-se o menu 'Modelo Geológico'. A tela que se segue, é apresentada na Figura 6.5 , onde tem-se os campos para entrada de: número de camadas, parâmetros para a criação do vetor de coordenadas x, o nome para o modelo. Após clicar em OK, outra janela é aberta, como a mostrada na Figura 6.6, onde pode-se entrar com a velocidade, a espessura e a densidade da primeira camada. Após

clicar em OK, essa janela é repetida para as camadas seguintes. Com essa estratégia de repetição de janelas, não é necessário colocar um limite para o número de camadas, já que a janela pode ficar se repetindo em um laço com o mesmo número de camadas.

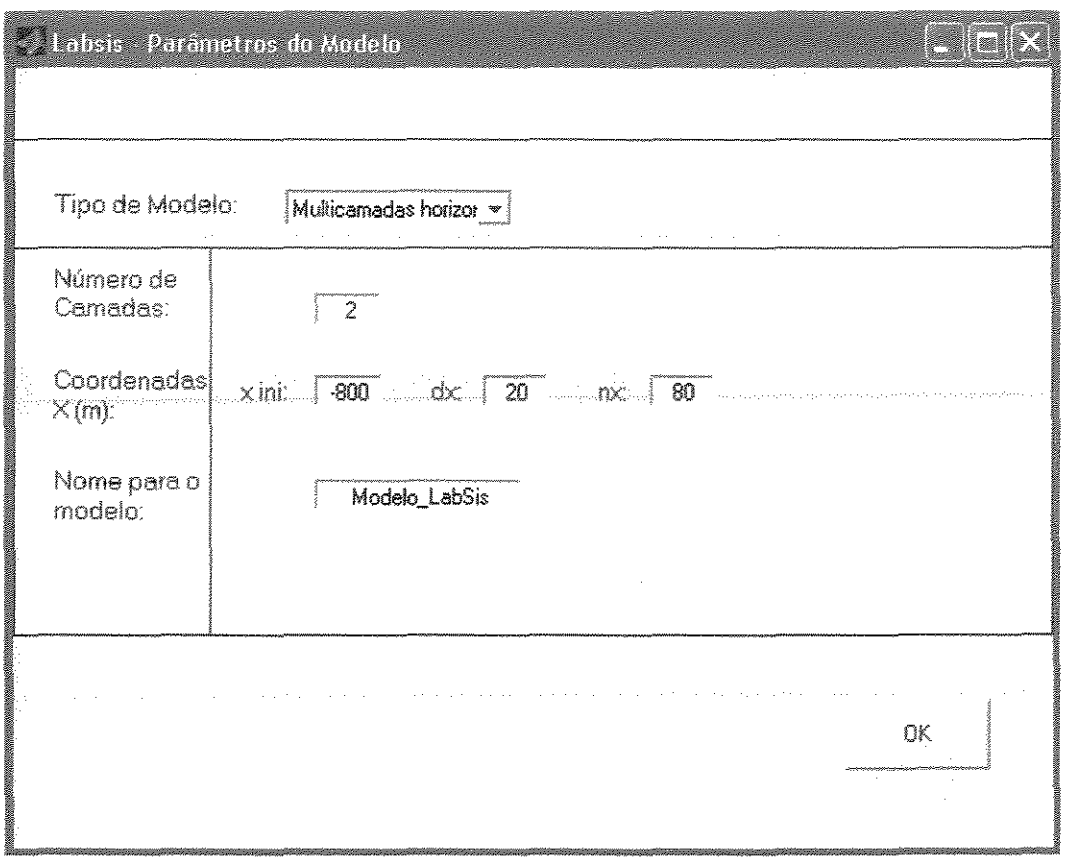


Figura 6.5: Tela para criação dos parâmetros do modelo geológico.

Uma ferramenta que auxilia na visualização do modelo geológico criado, é a opção 'Plotar Modelo Geológico', a qual desenha na tela o modelo atual, com as camadas e coordenadas (Figura 6.7).

A importação, para o LabSis, de modelos construídos no InterSis, é feita através do menu 'Importar modelo geológico InterSis'. Depois de se selecionar o arquivo, tem-se uma tela como apresentada na Figura 6.8, onde aparecem os principais parâmetros do modelo. No InterSis as coordenadas das interfaces são dadas apenas por alguns pontos de referência, no LabSis essas co-

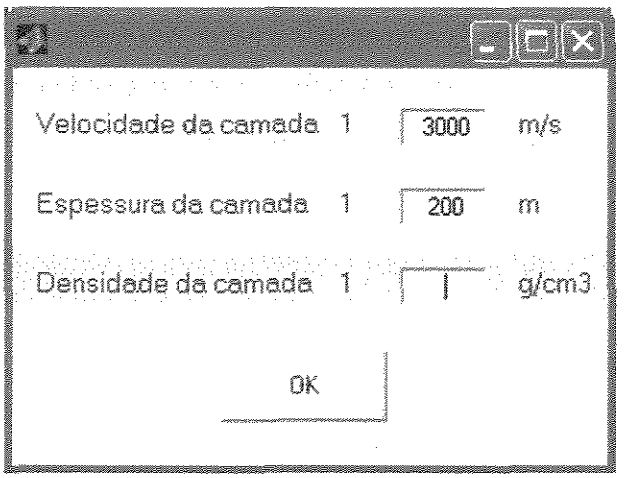


Figura 6.6: Tela para entrada dos parâmetros de cada camada.

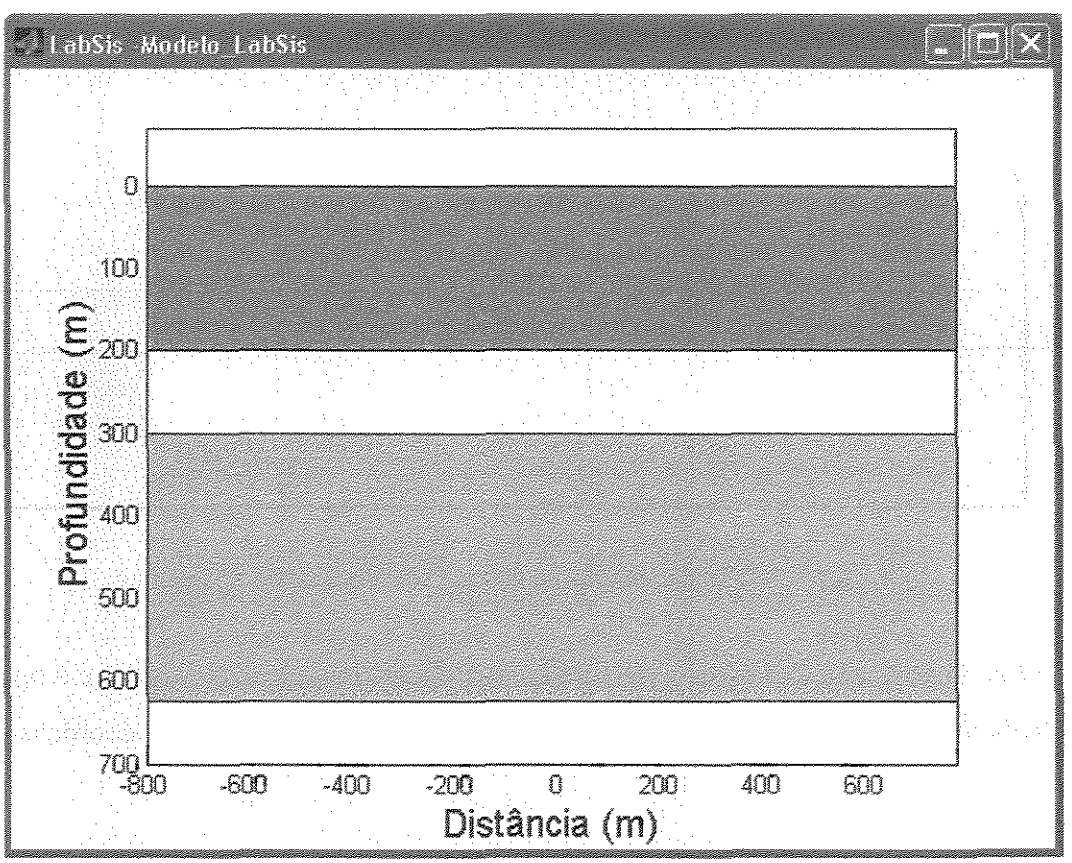


Figura 6.7: Modelo geológico com 4 camadas planas horizontais, criado no LabSis

ordenadas estão em forma de vetor. Para que essa transformação ocorra, o usuário deve entrar com o intervalo entre os pontos de x para criação do vetor a ser usado. O LabSis, automatica-

mente, escolhe um valor caso o usuário não se sinta a vontade para fazê-lo. Após clicar em OK, os parâmetros do modelo geológico são carregados na variável *modgeo* e, automaticamente plotados usando a função do LabSis para plotar modelos geológicos.

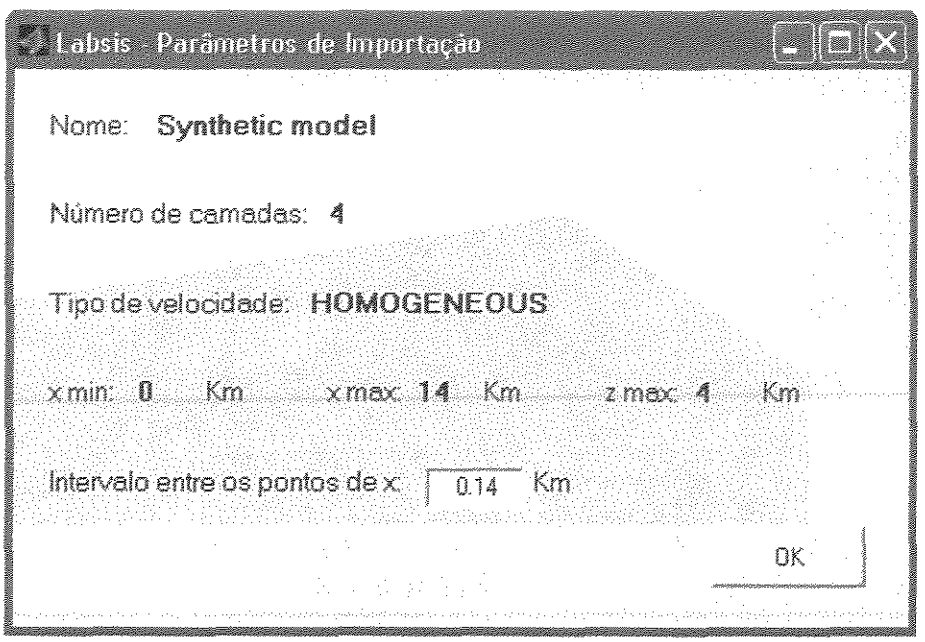


Figura 6.8: Tela de importação de modelo geológico no formato InterSis.

Após ter criado ou importado o modelo desejado, tem- se a opção de salvá-lo no disco, no formato .mat, através do menu 'Salvar Modelo Geológico'. Pode-se também carregar-lo através do menu 'Carregar Modelo Geológico'.

## 6.3 Módulo de Modelagem

As funções usadas para a modelagem, assim como, as utilizadas para o processamento e o imageamento, foram escritas por diferentes autores. O que foi feito neste caso, como já dito anteriormente, foi à construção de funções casca para tais funções, de modo que, o dado entrado anteriormente no formato interno *s*, torna-se um padrão para todas as funções. Assim, para as funções de modelagem, não é necessário entrar com nenhum parâmetro adicional, pois todos já

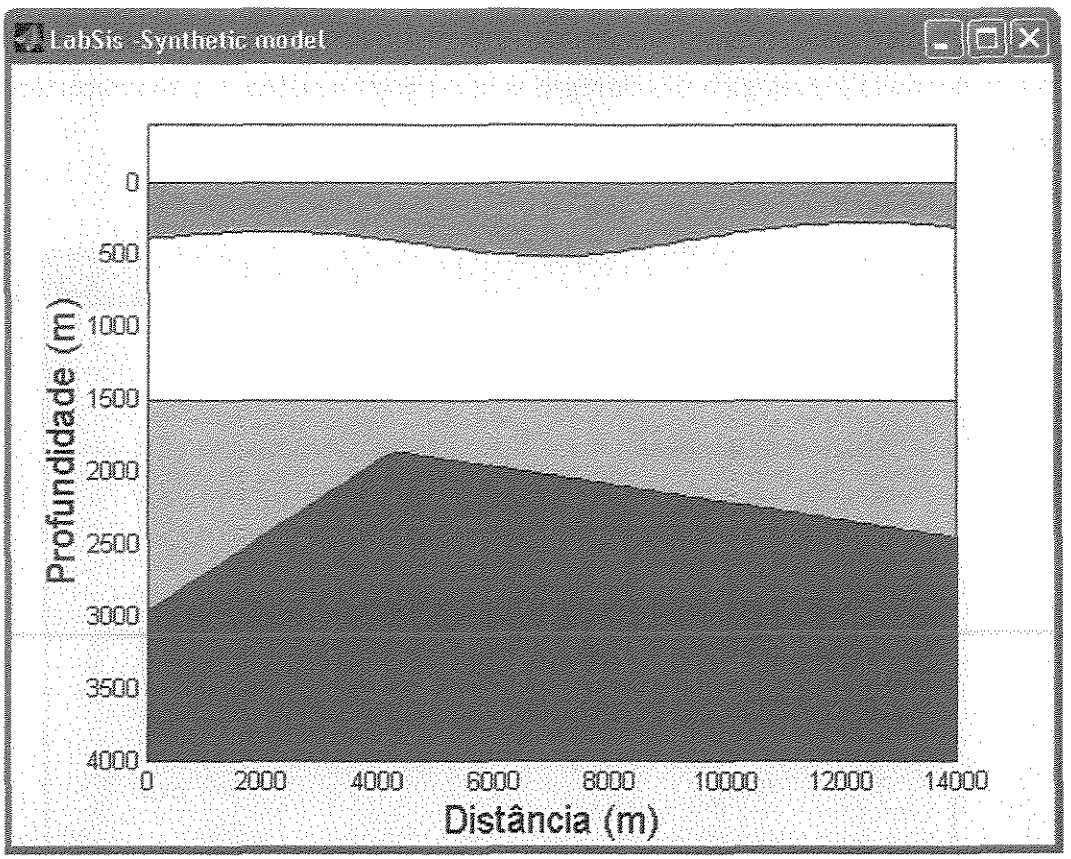


Figura 6.9: Modelo geológico construído no InterSis e Importado pelo LabSis.

estão armazenados nas variáveis *s* e *modgeo*. No LabSis existem quatro funções para modelagem, duas por traçado de raios, uma por Born e outra por Kirchhoff.

Para a modelagem por traçado de raios, existem duas funções. A primeira, é acessada através do menu 'CmpHoriz', permite uma modelagem para qualquer número de camadas planas horizontais. Após o termino da modelagem, uma janela é automaticamente aberta, com dado plotado e a matriz do sismograma fica armazenada no parâmetro *s.data.* Na Figura 6.10 há um exemplo de dado gerado usando-se o modelo geológico da Figura 6.7

A outra função para modelagem por traçado de raios é acessada através do menu 'raios'. Essa função, diferentemente da anterior, funciona para camadas inclinadas não planas, no entanto, ela só funciona para modelo de duas camadas em uma seção CO. Ao se tentar utilizar um modelo

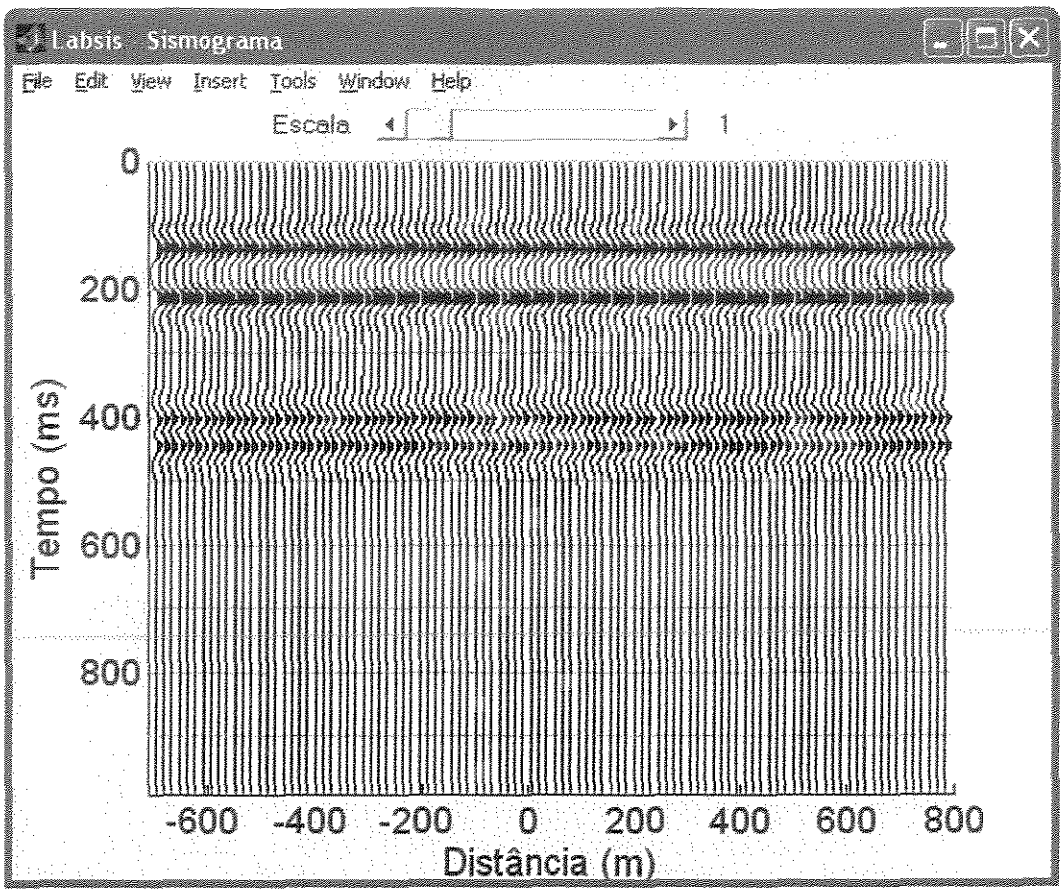


Figura 6.10: Sismograma gerado a partir da função CMPHoriz usando o modelo geológico da figura 6.7 com velocidades v1= 3000m/s, v2= 2500m/s, 3400m/s e 4000m/s

com mais de duas camadas, o usuário é alertado de que somente as duas primeiras serão consideradas, sendo que a primeira será o meio de propagação e a segunda, o refletor. Quando se utiliza essa função, ela abre duas janelas, em uma é plotado o modelo geológico utilizado, com os devidos traçados de propagação (Figura 6.11) e, em outra, o sismograma gerado (Figura 6.3).

As duas outras modelagens são feitas por Integral de Born e por Integral de Kirchhoff, que podem ser acessadas através dos menus com seus respectivos nomes. Para essas duas modelagens também não é necessário entrar com nenhum parâmetro, e assim que o usuário chama o menu, o dado é processado, armazenado no parâmetro *s.data* e o sismograma é plotado na tela. Assim como na modelagem anterior, essas funções só funcionam para modelos de duas camadas, planas

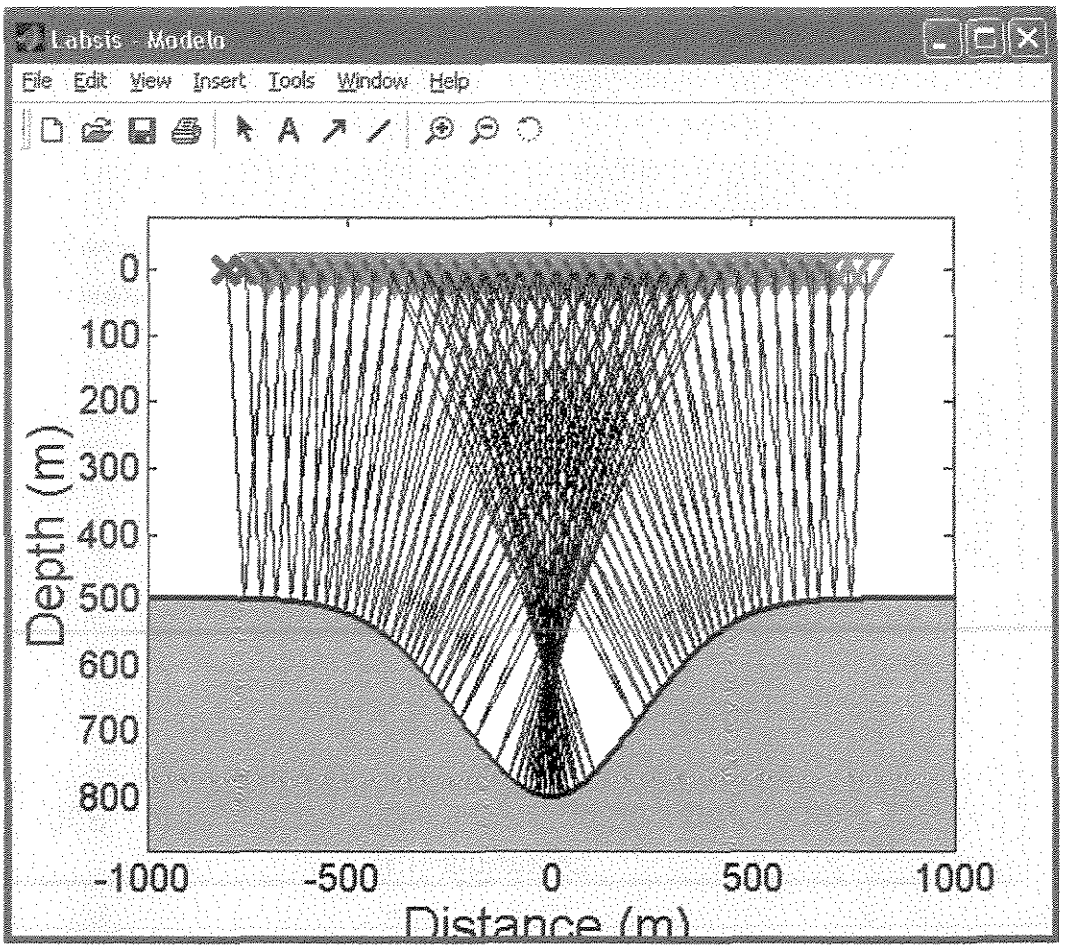


Figura 6.11: Modelo gerado pela função de traçado de raios. v1 =2500m/s v2=3000m/s.

ou não. A modelagem por Integral de Born, por exigir cálculos mais complexos, é bem mais lenta que os outras. Os resultados das modelagens por aproximação de Born e integral de Kirchhoff se encontram nas figuras 6.12 e 6.13 respectivamente.

## 6.4 Módulo de Análise de Velocidades

O módulo de análise de velocidades, é ativado através do menu *processamento* submenu *NMO -> Painel NMO*. Assim chega-se ao menu da figura 6.14. Para o uso desta função, o arranjo sísmico utilizado dever CMP, caso contrário, uma janela de mensagem aparecerá para advertir o

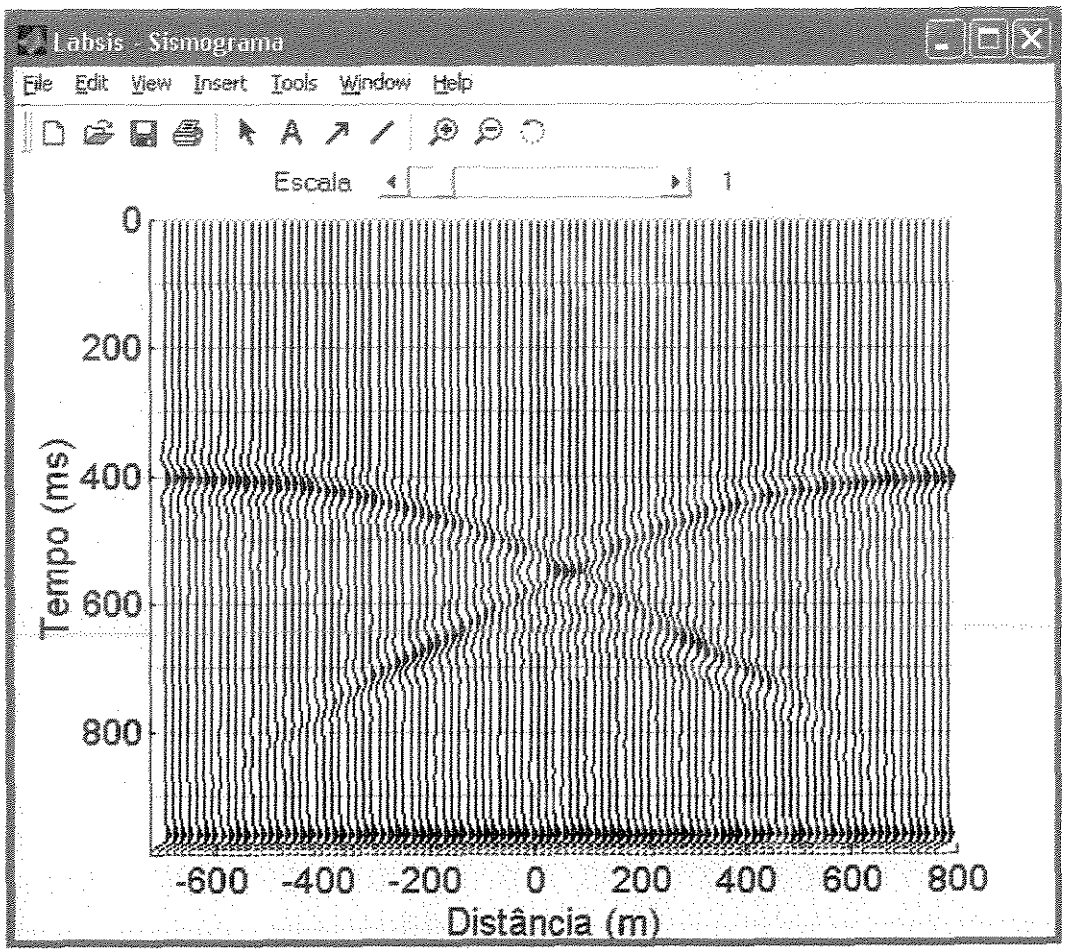


Figura 6.12: Sismograma gerado por modelagem Integral de Born usando o modelo da figura 6.11.

usuário.

Na janela da Figura 6.14 pode-se entrar com a velocidade inicial, a velocidade final e o incremento de velocidades, dessa maneira. Assim, o programa produzirá um painel com as devidas correções NMO. O número de painéis, dependerá do incremento utilizado. O painel gerado neste caso, é o que se encontra na Figura 2.7. Casa seja a habilitada a opção *Plotar o dado empilhado*, será plotado a direita de cada sismograma a soma de todas as wavelets, como mostrado na Figura 6.15.

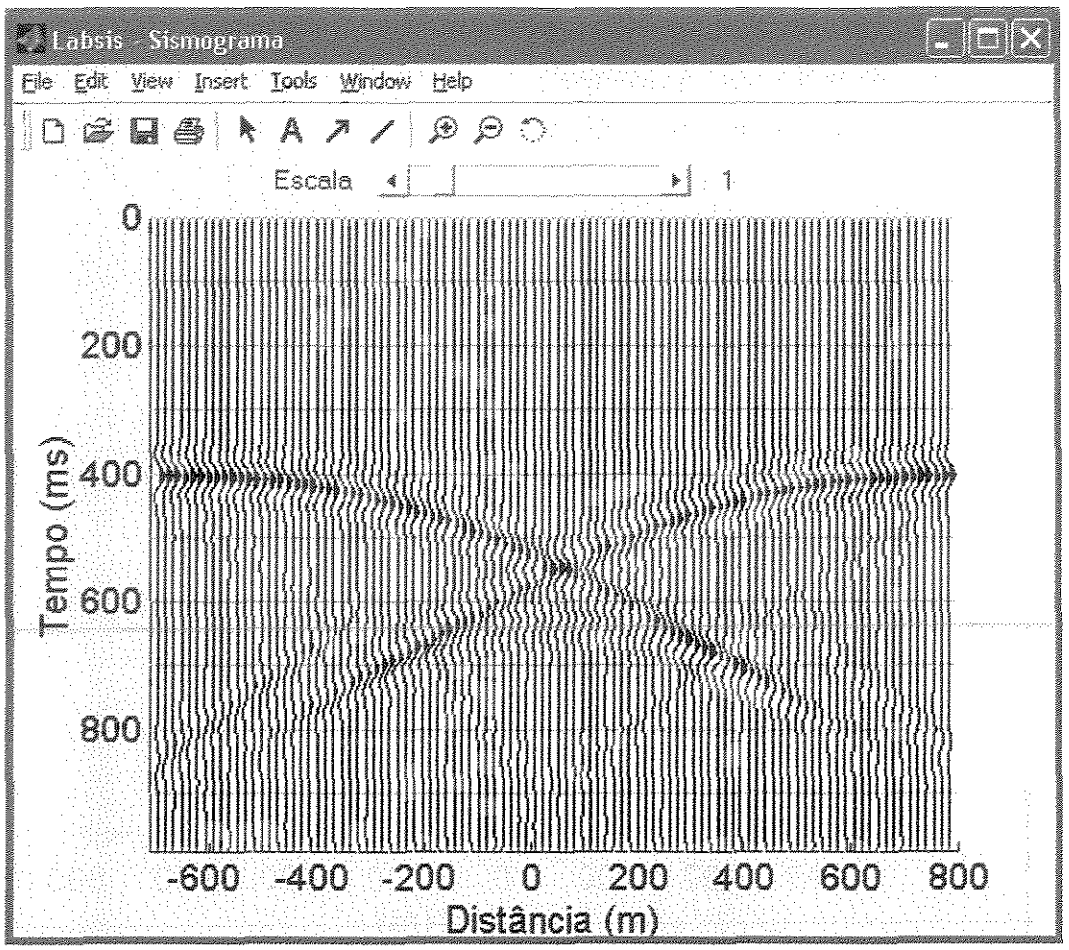


Figura 6.13: Sismograma gerado por Modelagem Integral de Kirchhoff usando o modelo da figura 6.11.

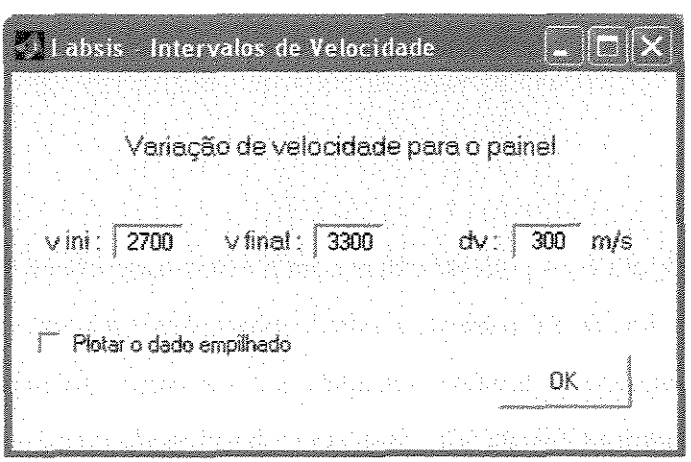


Figura 6.14: Janela para entrada dos parâmetros necessários para a construção do painel NMO

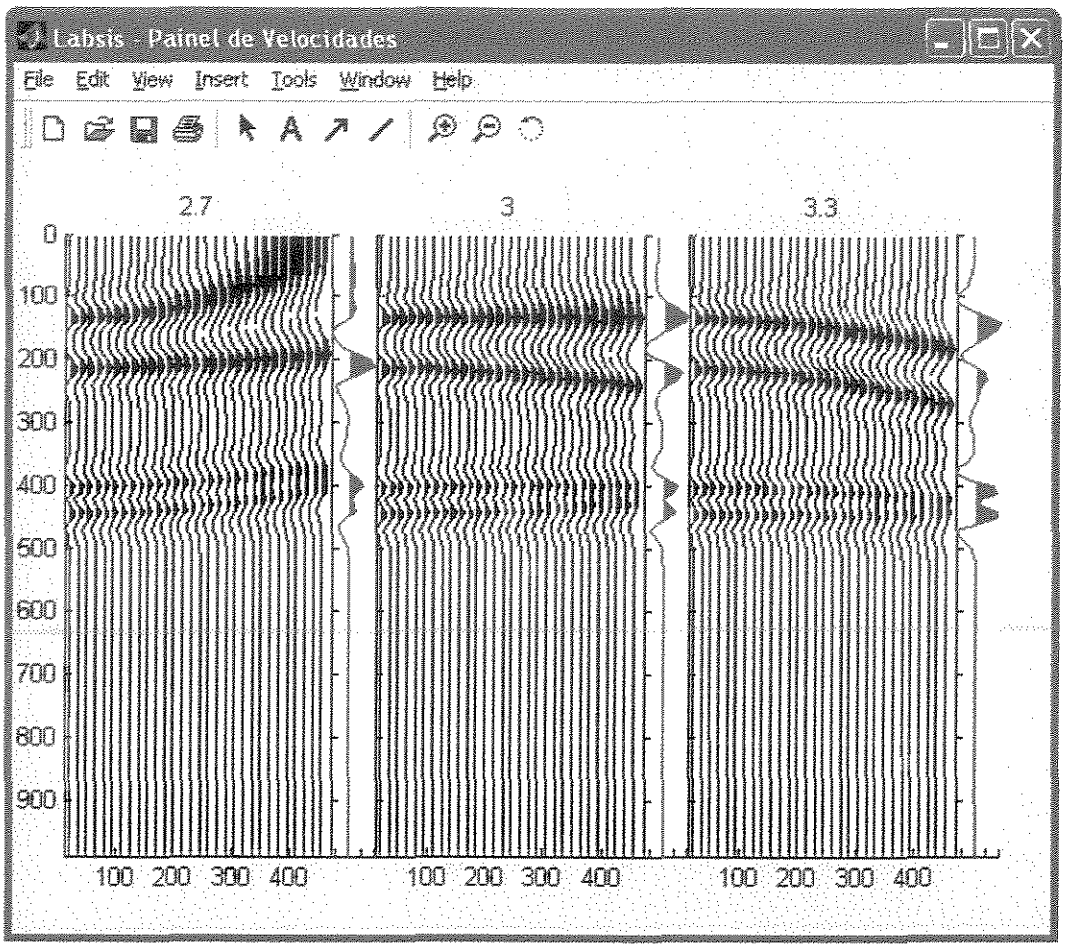


Figura 6.15: Painel de correções NMO mostrando o dado empilhado.

## 6.5 Módulo de Imageamento Sísmico

Para o imageamento, tem-se as funções de 'Migração Kirchhoff em Profundidade' e 'Demigração Kirchhoff'. Ambas as funções foram escritas para dados modelados pelas funções já descritas, com apenas duas camadas portanto, se houver mais de um refletor, a migração ou a demigração se ajustará apenas para um deles.

Assim como os dados de sismograma e os dados de modelo geológico, os dados migrados também ficam armazenados em uma variável interna especial chamada *mig*. Isso dá ao usuário a flexibilidade de manter na memória vários dados diferentes, podendo-se alternar entre eles através do menu 'Dado'. Tem-se, a seguir, um exemplo de conteúdo da variável *mig*.

```
mig =

     t: [1x100 double]      Vetor tempo
domain: 'profundidade'      Domínio
   vel: 2.5000              Velocidade para migração
  xini: -800                Coordenada x inicial
    xf: 800                 Coordenada x final
    dx: 20                  Intervalo entre os pontos do vetor x
  zini: 0                   Coordenada z inicial
    zf: 1000                Coordenada z final
    dz: 20                  Intervalo entre os pontos do vetor z
  nome: 'MIG_tese'          Nome para o dado
     x: [1x81 double]       Vetor coordenada x
     z: [1x51 double]       Vetor coordenada z
    m1: [81x51 double]      Dado migrado
    m2: [81x51 double]      Dado com dupla migração
```

A migração é acessada através do menu 'Migração Kirchhoff em Profundidade' de onde se tem acesso a janela da Figura 6.16, para entrada dos parâmetros necessários para a migração. Onde entra-se com a velocidade de migração para o refletor, os parâmetros para criação dos vetores de coordenadas x e z como nos casos anteriores e, o nome para a variável migrada. Após clicar em OK, o dado o dado é carregado na variável *mig* e, o dado da variável *mig.m1* será plotado na tela (Figura 6.17).

A demigração é acessada através do menu 'Demigração Kirchhoff', onde tem-se acesso a janela da Figura 6.18, e entra-se com a velocidade a ser usada na demigração, o valor do offset (essa função para demigração só funciona para CO), os valores para criação do vetor com as coordenadas do ponto médio, e o nome para a variável. Após se clicar em OK, o dado presente na variável *mig.m1*, atualmente na memória, será demigrado, carregado em uma variável *s* e, plotado na tela (Figura 6.19). O dado agora, volta a ter o formato original do dado modelado, e este pode

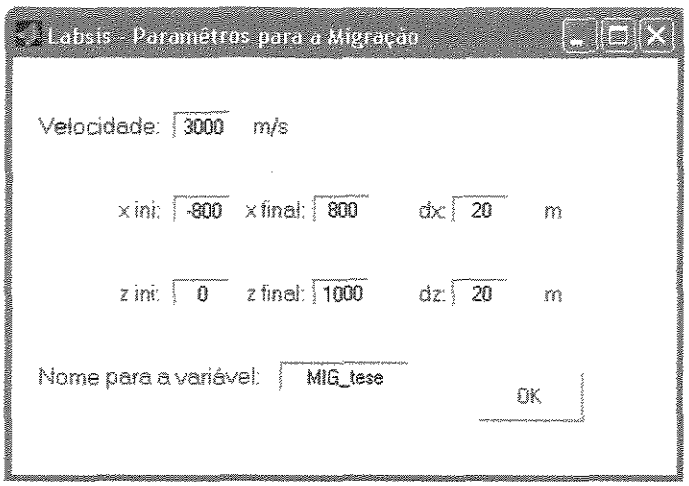


Figura 6.16: Janela para entrada dos parâmetros da Migração Kirchhoff em profundidade.

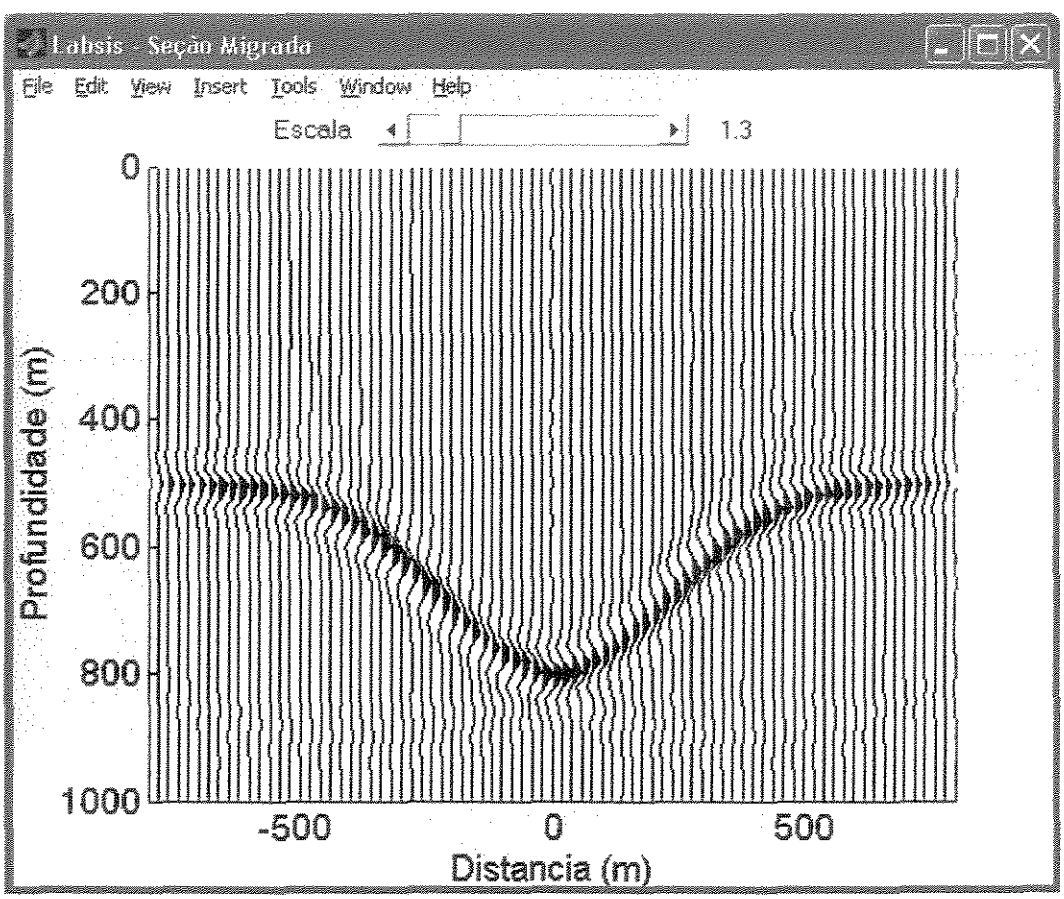


Figura 6.17: Dado da Figura 6.3, modelado-se usando o modelo geológico da Figura 6.11 e, migrado em profundidade usando as ferramentas de imageamento do LabSis.

ser editado ou migrado. O dado presente na variável *mig* é preservado.

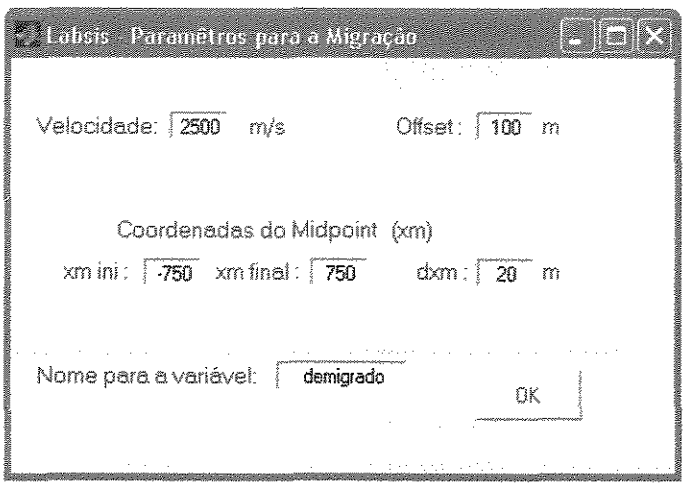


Figura 6.18: Janela para entrada dos parâmetros da Demigração Kirchhoff.

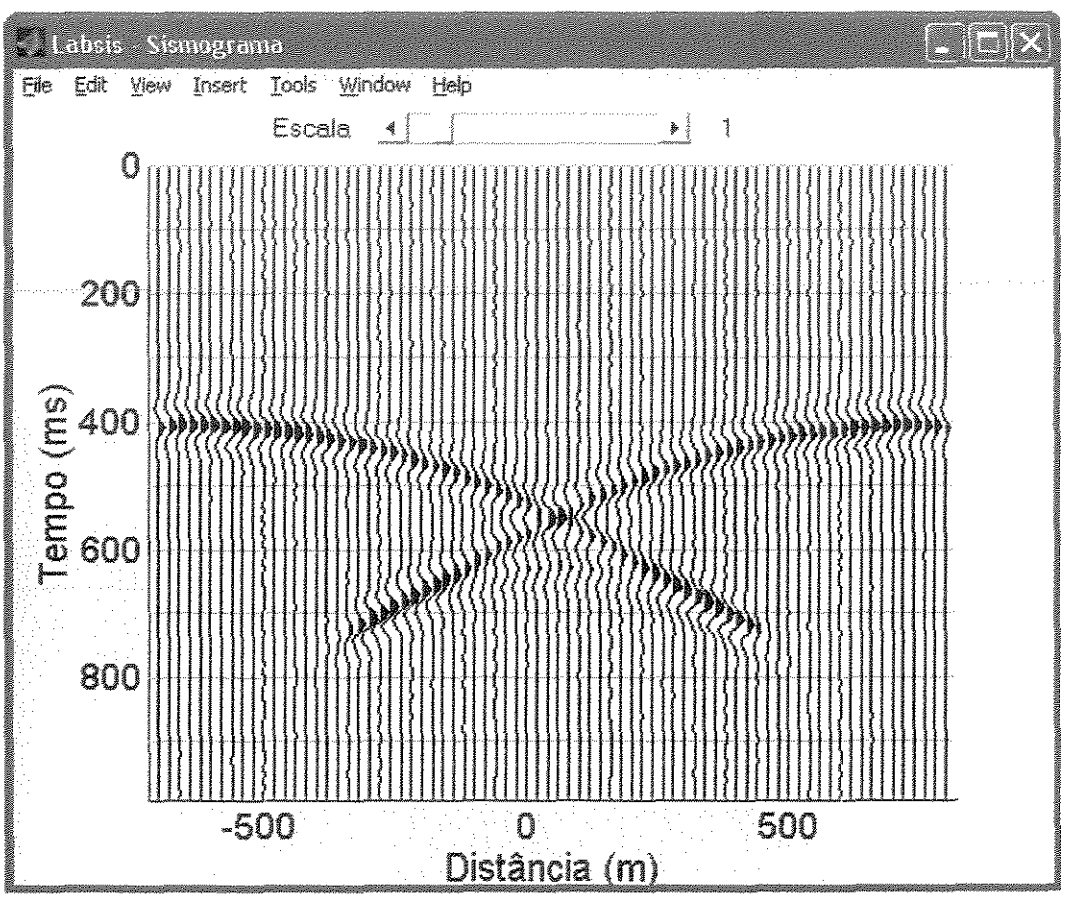


Figura 6.19: Dado da Figura 6.17, demigrado, usando-se as ferramentas de imageamento do LabSis.

# Capítulo 7

# Exemplos Ilustrativos

Este capítulo visa apresentar alguns exemplos que ilustram as capacidades do LabSis em modelar dados sísmicos, criar modelos geológicos, imagear dados sísmicos e, sua interação com o InterSis, usando diferentes situações e parâmetros. Além da possibilidade de visualizar, analisar e comparar os resultados usando ambos os softwares.

No capítulo 6 foram expostos alguns exemplos com a intenção de mostrar o funcionamento de cada menu. Neste capítulo serão analisados diferentes exemplos, e sempre que possível, compará-los com o InterSis, gerando o dado com os mesmos parâmetros nos dois programas.

Nos exemplos mostrados neste capítulo os parâmetros são expostos de maneira clara, com intenção de que o leitor possa reproduzir os resultados para comprovar-los.

## 7.1 Modelo 1: Refletor Sinclinal

O primeiro caso a ser analizado, é um modelo de refletor sinclinal criado no InterSis, o dado é modelado também no InterSis e em seguida, exportado para o LabSis. No LabSis, com o mesmo modelo geológico, modela-se o dado e em seguida ambos os dados (o modelado no LabSis e o modelado no InterSis) são migrados, demigrados e comparados.

Como primeiro passo, cria-se no InterSis o modelo geológico com um refletor sinclinal. O modelo criado tem 5000m de comprimento total e 2000m de profundidade. A velocidade da camada 1 é de 2500 m/s, e a velocidade do refletor é de 3000m/s. O grid criado no InterSis para o modelo tem um espaçamento de 50m (esse é o menor valor por esse programa). Todos esses valores foram usados como padrão para os modelos seguintes, a não ser quando mencionado de maneira diferente. Assim, construiu-se no InterSis o modelo sinclinal da figura 7.1.

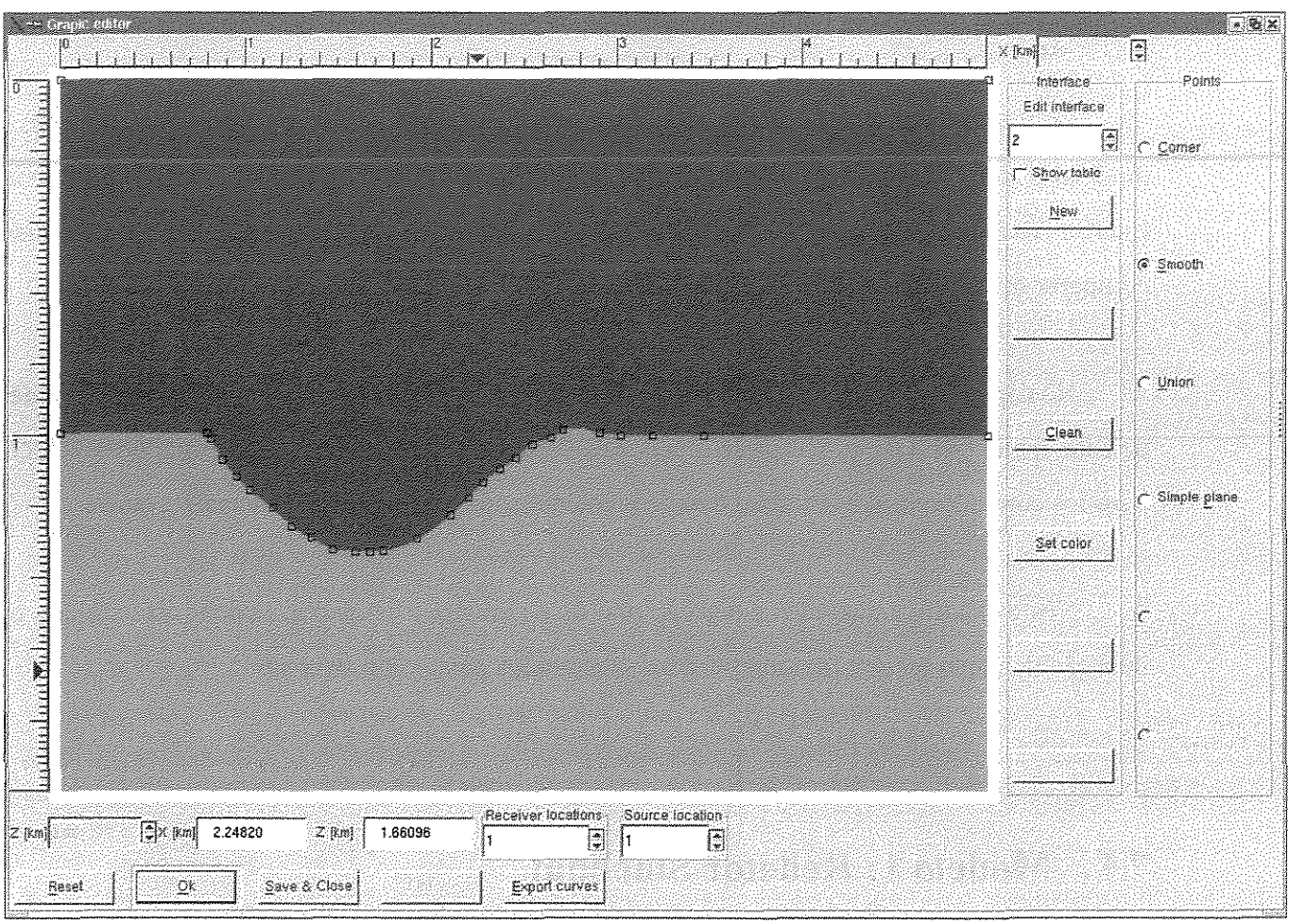


Figura 7.1: Modelo geológico de forma sinclinal criado no LabSis

O passo seguinte é modelar o dado no InterSis. Foram usados 60 receptores e 60 tiros, com um espaço entre os receptores de 50m e espaçamento entre fonte também de 50m. A fonte es-

tava na posição do receptor 5. O modelo foi gerado usando o módulo do seis88 do InterSis. Os parâmetros mencionados neste parágrafo serão os mesmos para os modelos seguintes modelados no InterSis, a não ser que seja mencionado de maneira diferente. Na figura 7.2 pode-se observar as trajetórias geradas para o primeiro e o último tiro.

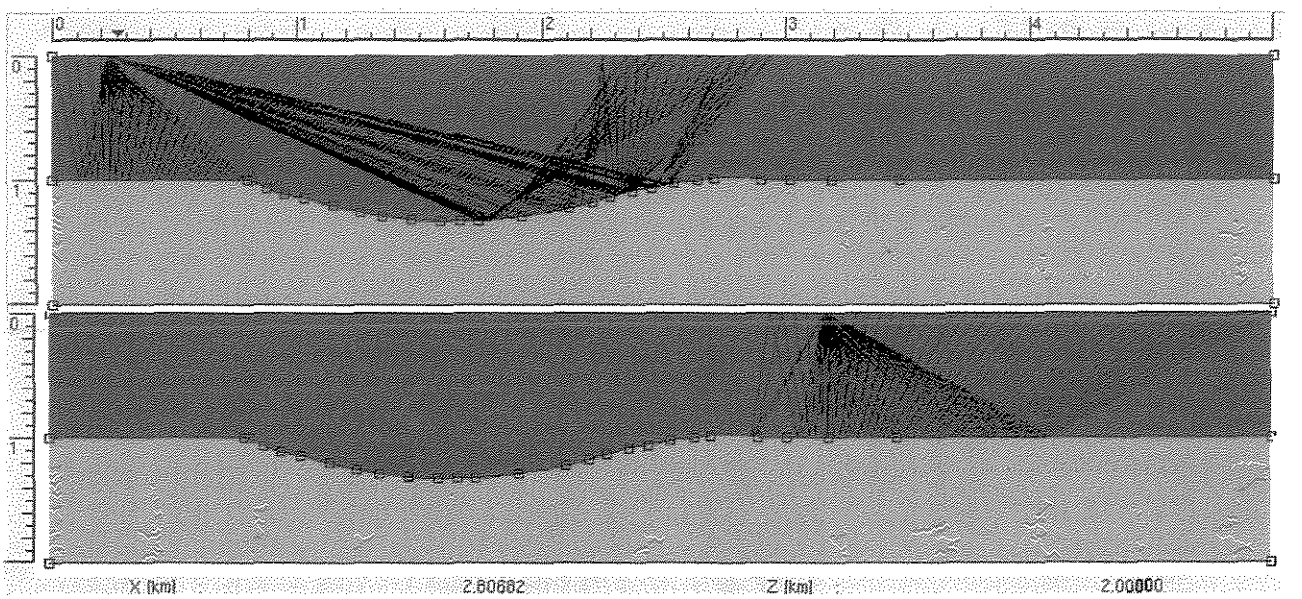


Figura 7.2: Trajetória de raios para dado sísmico modelado no InterSis para os tiros 1 e 60 respectivamente.

O dado gerado no InterSis foi importado no LabSis como um arranjo common offset, e utilizando um *offset* de 25m. Depois de importado o modelo geológico e o dado sísmico para o LabSis, foi possível modelar o dado sísmico no LabSis para comparação com o resultado do InterSis.

Os passos que serão descritos a partir de agora e até o final desta seção, serão todos executados no LabSis. Para a modelagem, usou-se um arranjo *common offset*, com 60 pares transmissor-receptor e com *offset* de 25m. Entre cada receptor havia um espaçamento de 50m e entre cada fonte havia um espaçamento também de 50m. O primeiro receptor estava na posição 200m e a primeira fonte estava na posição 225m, esses valores foram escolhidos por serem os mesmos do dado gerado no InterSis que foi importado. Esses mesmos valores foram utilizados nos modelos

seguintes. O dado foi modelado por traçado de raios, e na figura 7.3 tem-se o traçado de raios obtido no LabSis

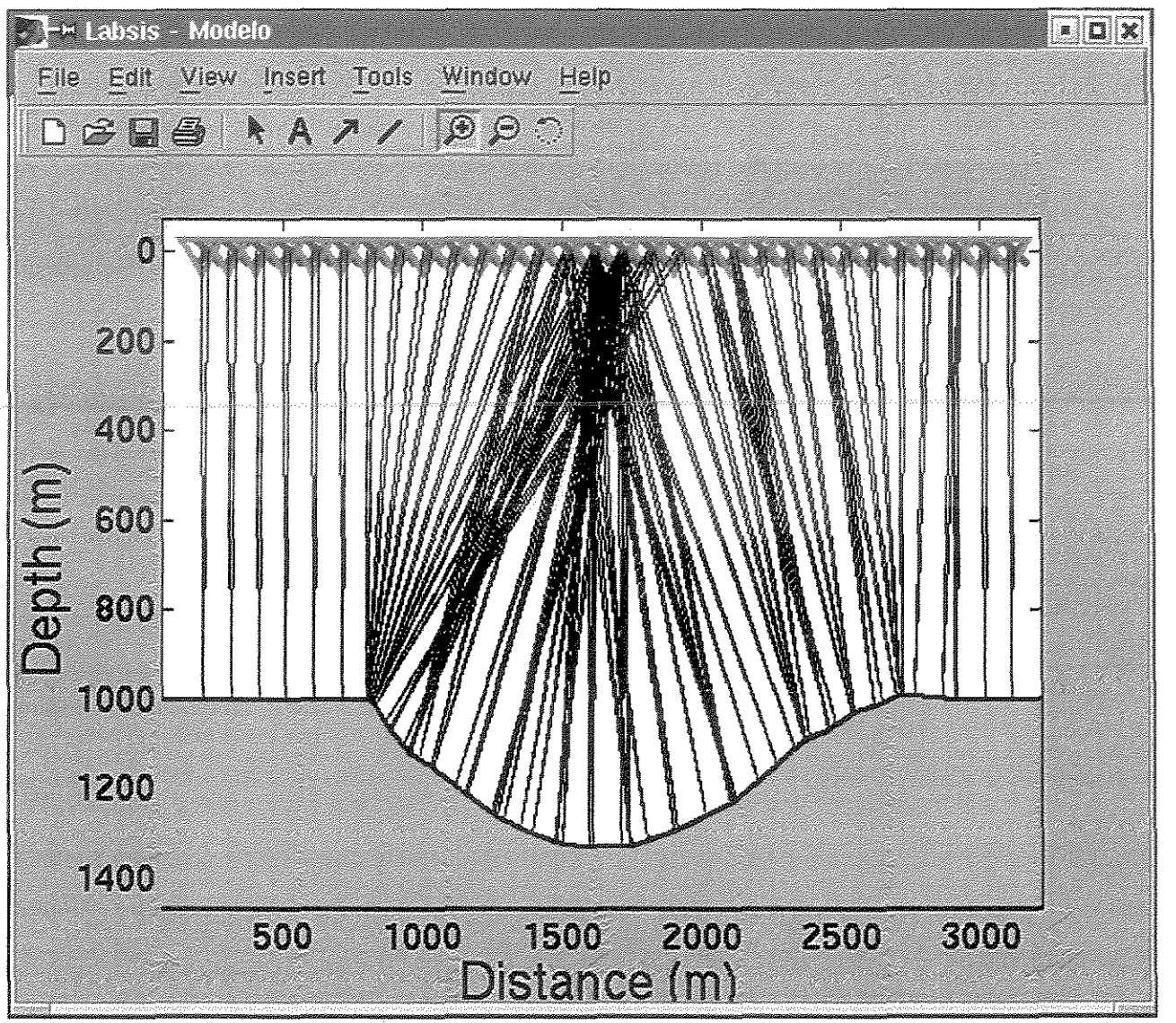


Figura 7.3: Trajetória de raios para dado sísmico modelado no LabSis usando o mesmo modelo geológico da figura 7.1.

Na figura 7.4 compara-se os resultados obtidos do dado modelado no LabSis e no InterSis. Como se pode ver, não há nenhuma grande discrepância entre os dois dados apresentados.

A seguir, ambos os dados foram migrados em profundidade, usando uma velocidade de 2500m/s e o resultado se encontra na figura 7.5. Como pode ser observado, os resultados são semelhantes, exceto que o gerado no InterSis possui um pouco mais de ruído.

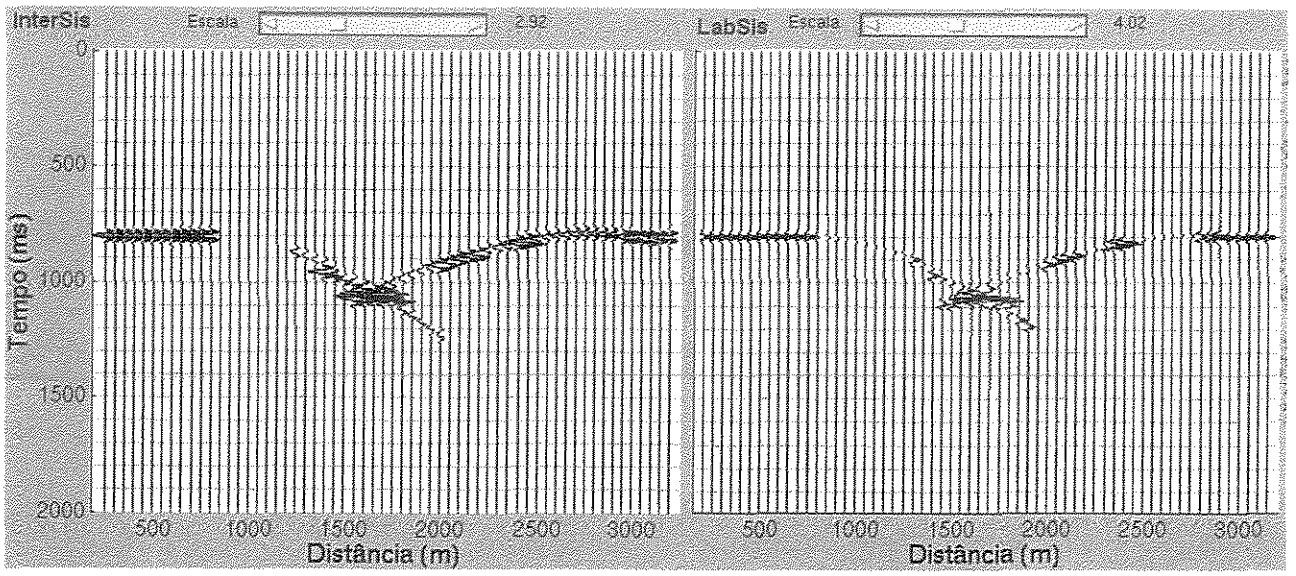


Figura 7.4: Sismogramas do dado modelado no InterSis (direita) e do dado modelado no LabSis (esquerda).

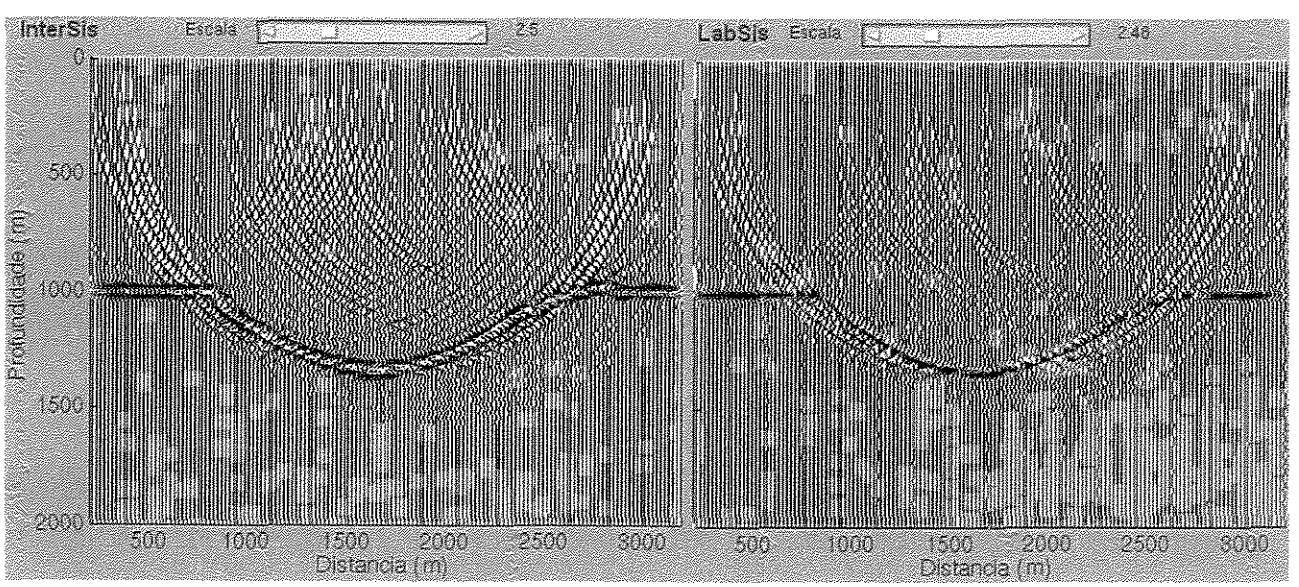


Figura 7.5: Sismogramas da figura 7.4 migrados pelo LabSis.

O próximo passo foi demigrar ambos os dados para verificar se mantém a suas características originais. O resultado desta demigração se encontra na figura 7.6. Como se vê, ambos os dados recuperaram a forma original do sismograma, apenas com um pequeno ruído no dado gerado no InterSis.

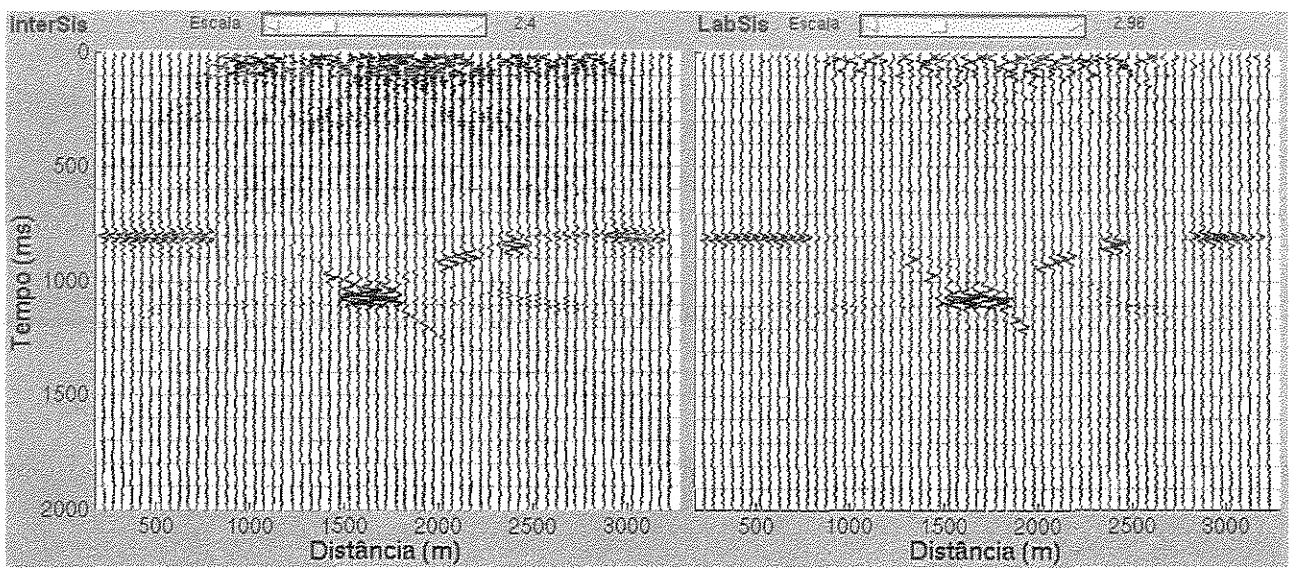


Figura 7.6: Sismogramas demigrados pelo LabSis a partir dos dados sísmicos da figura 7.5.

## 7.2 Modelo 2: Refletor Anticlinal

Nesta seção apresentamos um modelo contendo um refletor anticlinal, construído no LabSis. Os parâmetros usados para a modelagem são os mesmos do modelo anterior, sendo que a única diferença é o modelo geológico utilizado, este é apresentado na figura 7.7.

Assim como no exemplo anterior, o dado foi modelado no InterSis e exportado para o LabSis, onde também foi modelado com os mesmos parâmetros usados no InterSis para comparação. O resultado das duas modelagens encontra-se na figura 7.8.

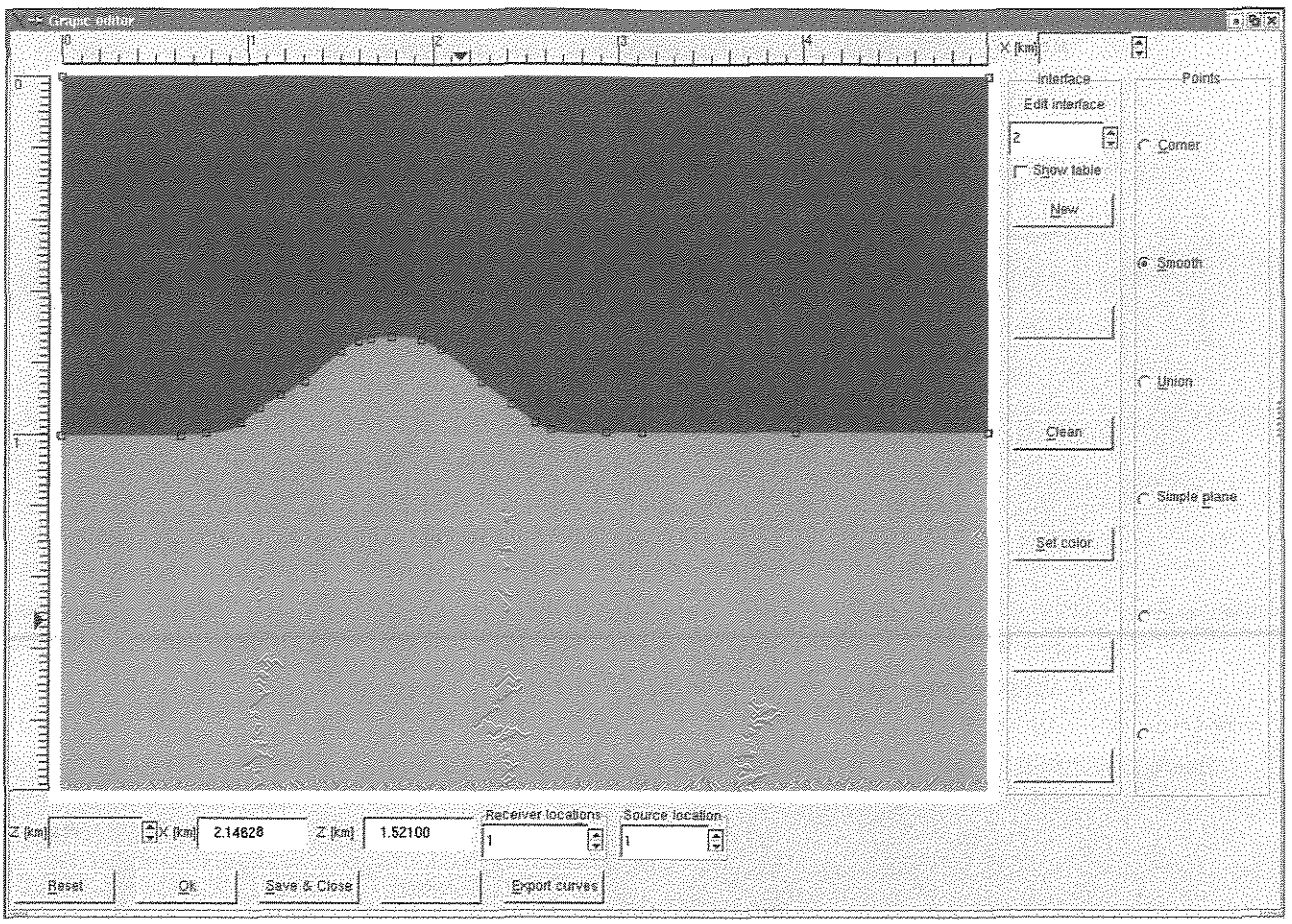


Figura 7.7: Modelo geológico com um refletor anticlinal.

Novamente, como no exemplo anterior, esses modelos foram migrados e demigrados, os resultados se encontram nas figuras 7.9 e 7.10 e tiveram qualidade semelhante a do exemplo anterior.

## 7.3 Modelo 3: Refletor em forma de Talude

Neste exemplo, foi construído um modelo geológico em forma de talude. Esse modelo geológico foi construído no InterSis com os mesmos parâmetros dos exemplos anteriores. A aquisição foi feita usando-se um arranjo common offset, como nos exemplos anteriores. Na figura 7.11 tem-se o modelo geológico com os raios sísmicos e o sismograma gerado, e na figura 7.12 tem-se o mesmo dado migrado e demigrado.

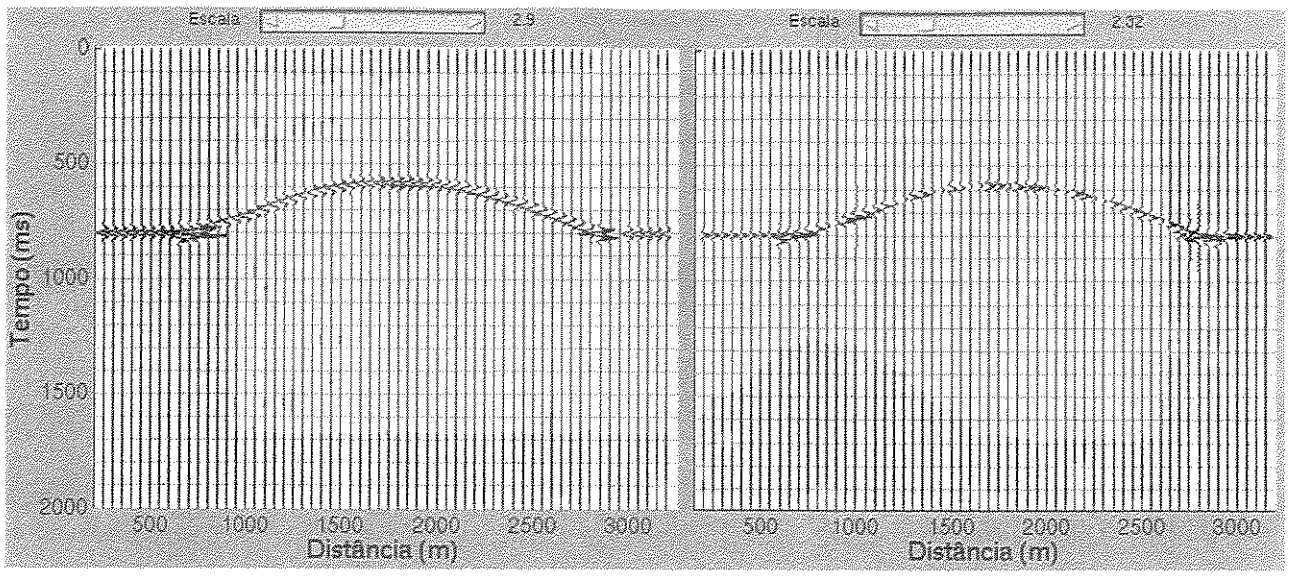


Figura 7.8: Sismogramas do dado sísmico modelado no InterSis (direita) e do dado sísmico modelado no LabSis (esquerda).

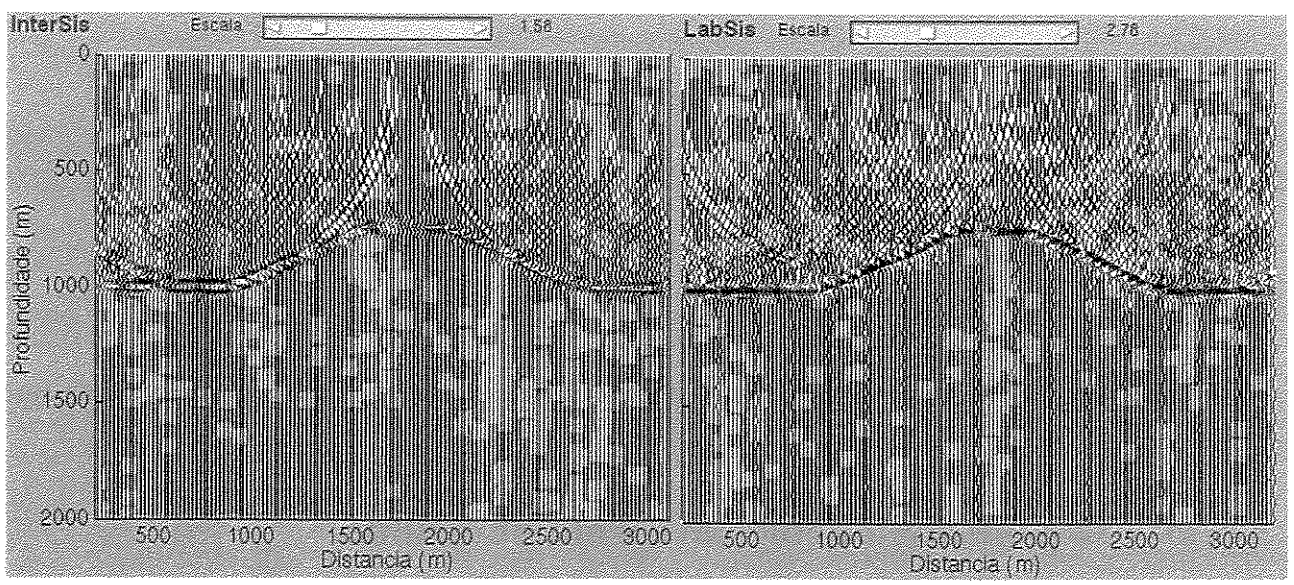


Figura 7.9: Sismogramas da figura 7.8 migrados pelo LabSis.

## 7.4 Modelo 4: Transformada Tau-P

Neste exemplo construímos um modelo de 4 camadas planas com espessuras de 150m, 200m, 150m e 200m e com velocidades de 2500m/s, 300m/s, 2700m/s e 3500m/s respectivamente. O

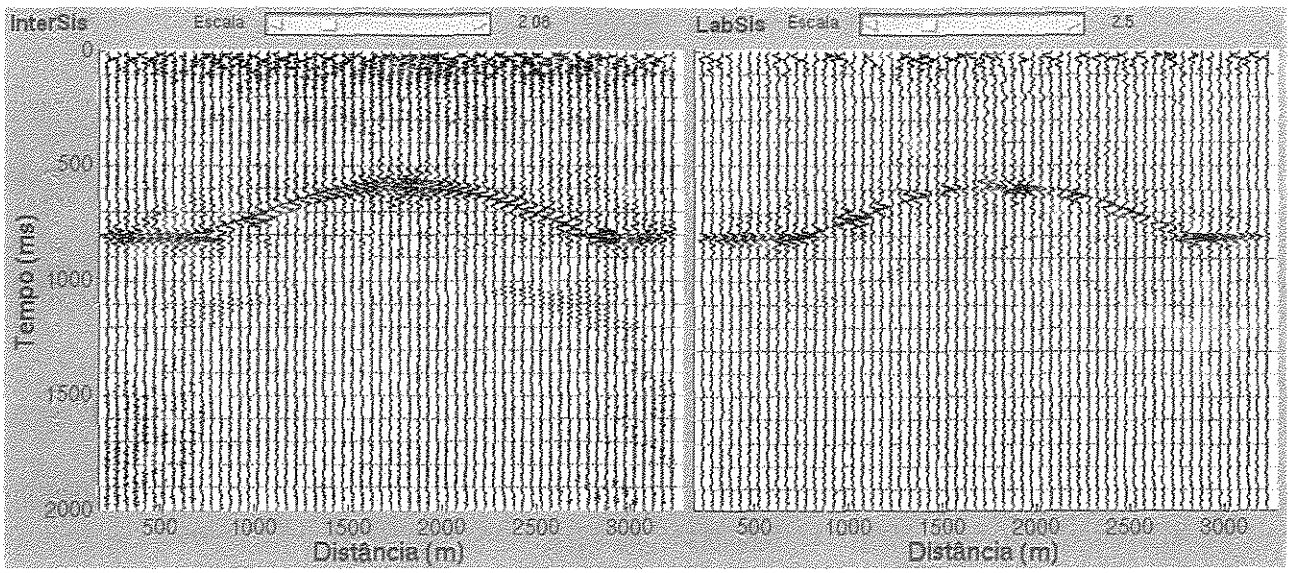


Figura 7.10: Sismogramas demigrados pelo LabSis a partir dos dados sísmicos da figura 7.9.

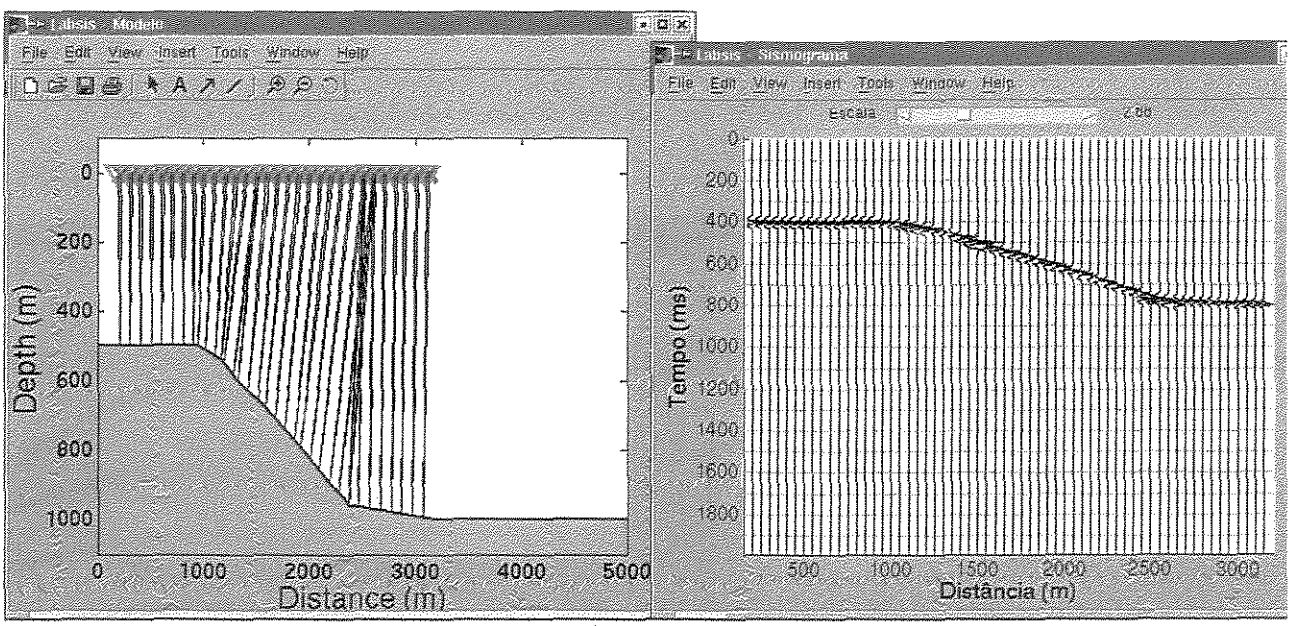


Figura 7.11: Modelo geológico em forma de talude , com o arranjo e os raios sísmicos (esquerda) e sismograma gerado a partir deste modelo (esquerda).

dado foi modelado com arranjo CS, usando 80 receptores com um espaçamento de 20m e com a fonte no centro do arranjo. Os dados foram modelados no LabSis usando-se a função CMPHoriz. O modelo geológico usado e sismograma obtido encontram-se na figura 7.13. Com o novo dado

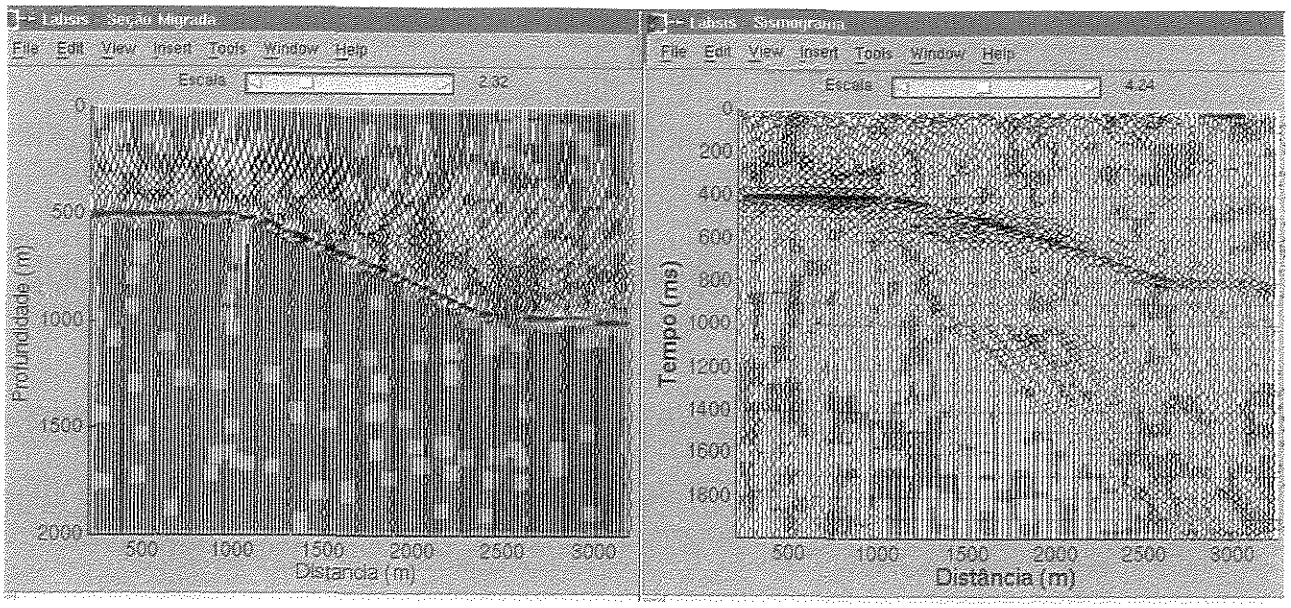


Figura 7.12: Sismograma migrado a partir do dado sísmico da figura 7.12 (esquerda) e a demigração deste sismograma (direita).

pode-se fazer a transformação Tau-P, o resultado encontra-se na figura 7.14.

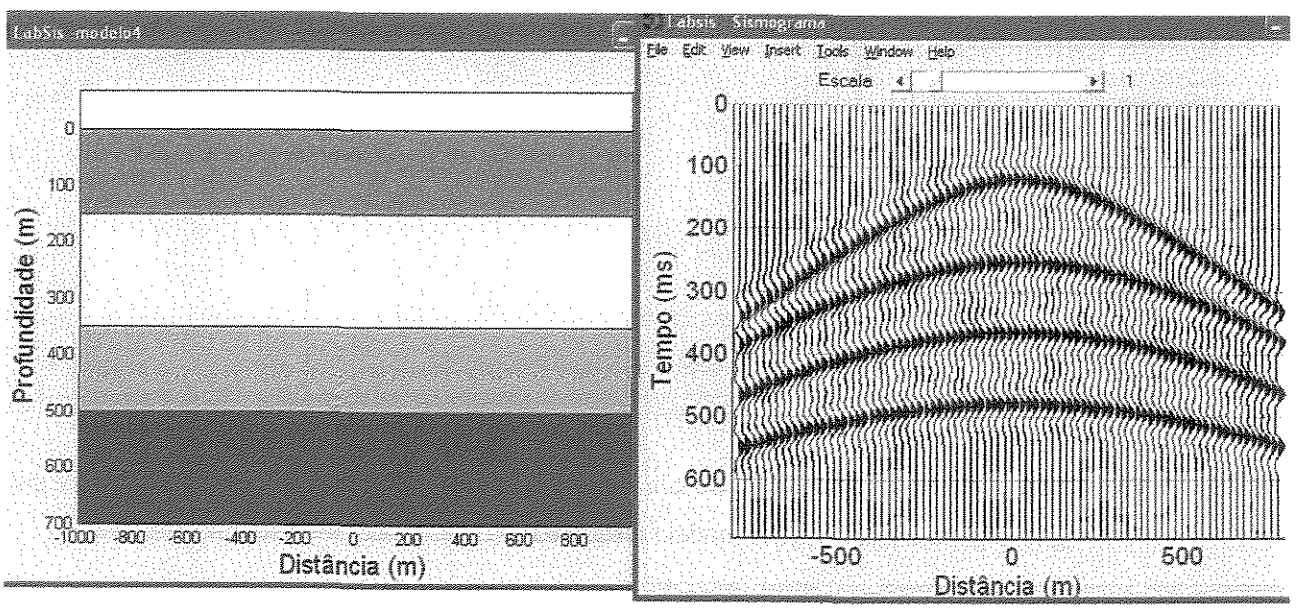


Figura 7.13: Modelo geológico e sismograma gerados no LabSis.

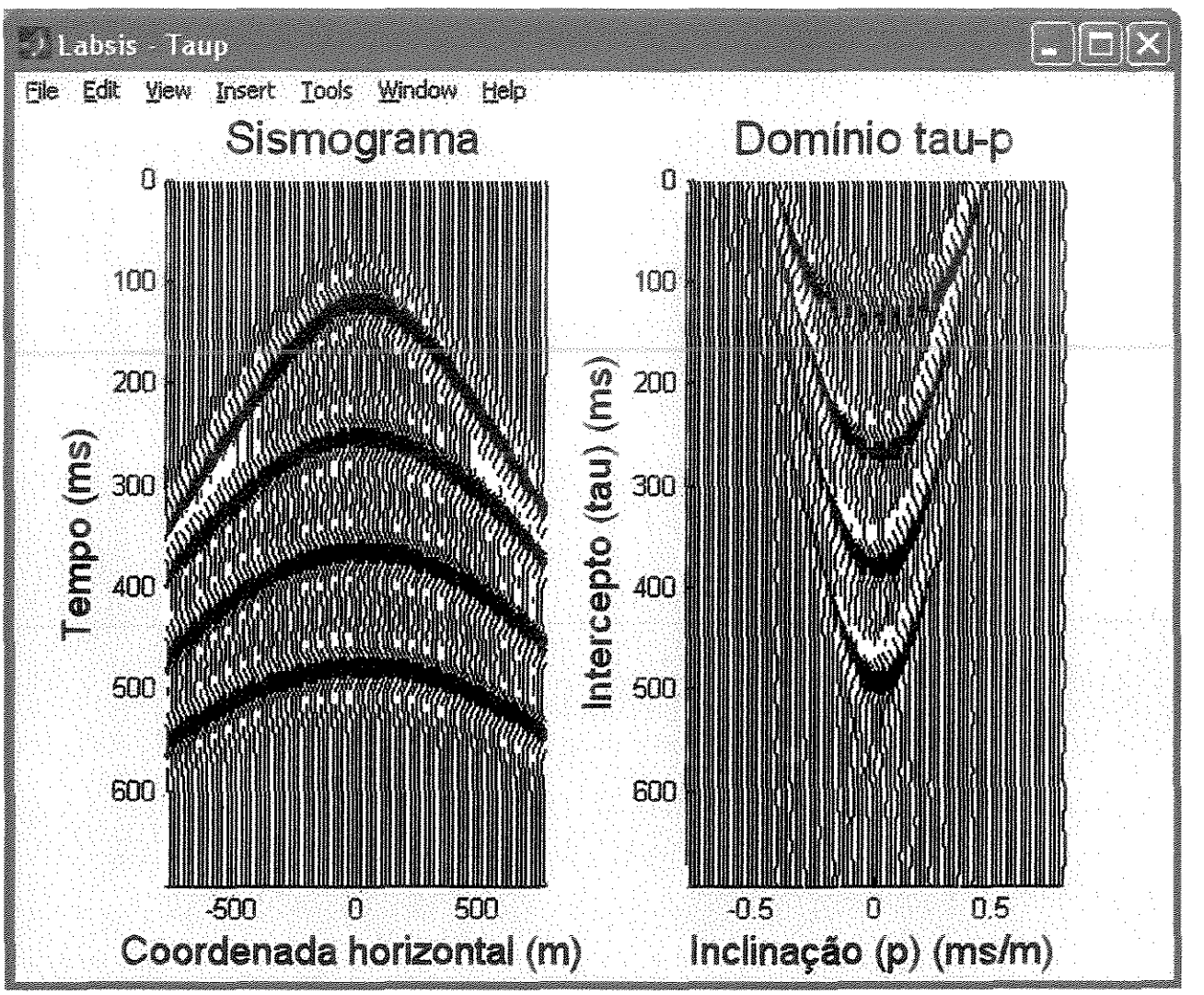


Figura 7.14: Sismograma modelado no LabSis (esquerda) e sua transformada para o domínio Tau-P (direita).

# Capítulo 8

# Conclusões

Nesta tese, foi desenvolvido o LabSis, um pacote para tratamento de dados sísmicos, usando a interface gráfica do Matlab. LabSis integra várias funções de programadores distintos, permitindo a comunicação entre elas. A existência de funções trabalhando separadamente e sem comunicação, foi a principal motivação para a criação do LabSis. Consideramos que o programa atingiu seus objetivos, com de uma interface clara e intuitiva.

Uma grande variedade de ferramentas foram incorporadas ao LabSis, dentre as quais podemos citar

- Pode-se trabalhar com várias variáveis ao mesmo tempo e, altera-las selecionando em um menu;
- Plotar dados sísmicos e, alterar o ganho através de um *slider*, na própria janela;
- Importar modelos geológicos construídos pelo InterSis
- Importar dados sísmicos no formato Seismic Unix e, conseqüentemente no formato InterSis;
- Duas funções de modelagem por traçado de raios;
- Modelagem por aproximação de Born;

- Modelagem por integral de Kirchhoff;
- Análise de velocidades NMO;
- Transformada Tau-p;
- Análise AVO;
- Migração Kirchhoff em profundidade em verdadeira amplitude;
- Demigração Kirchhoff em verdadeira amplitude;

Além destas funções, como demonstrado nesta tese, pode-se facilmente introduzir novas funções ao programa, permitindo que o LabSis possa crescer em conteúdo. A intenção deste trabalho é que ele tenha continuidade e que, a cada vez que uma nova função for criada para resolver um determinado problema, ela possa ser incorporada ao programa, para que possa ser compartilhada com outros usuários, permitindo assim o aproveitamento máximo da nova contribuição.

# Referências


Carcione, J., Herman, G., and Kroode, A., 2002, Y2k review article - seismic modeling: Geophysics, **67**, no. 4, 1304–1325.

Chapman, C. H., and Coates, R. T., 1994, Generalized Born scattering in anisotropic media: Wave Motion, , no. 19, 309–341.

Flynn, P. Formating information: A beginner's introdution to typesetting with LaTeX , março 2003. V3.2.

Gajewski, D., and Vanelle, C., 2002, Revisiting NMO again?: EAGE 64th Conference & Technical Exhibition - Florence, Italy.

Hubral, P., Schleicher, J., and Tygel, M., 1996, A unified approach to 3-D seismic reflection imaging, part 1: Basic concepts: Geophysics, no. 61, 742–758.

Iversen, E., 2001, Ray system for propagation of seismic isochorons. part 1: Isochrons rays.: 7o Congresso Internacional da Sociedade Brasileira de Geofisica.

Iversen, E., 2001, Ray system for propagation of seismic isochorons. part 2: Velocity rays.: 7 Congresso Internacional da Sociedade Brasileira de Geofisica.

Jaramillo, H., and Bleistein, N., 1999, The link of Kirchhoff migration and demigration to Kirchoff and Born modeling: Geophysics, **64**, no. 6, 1793–1805.

Liner, C. Tutorial: Concepts of normal and dip moveout:. Department of Geosciences, University of Tulsa, May 1999.

Margrave, G. F. Numerical methods in exploration seismology with algorithms in Matlab:. Department of Geology and Geophysics of University of Calgary, janeiro 2001.

MathWorks Inc. MATLAB The Language of Thechnial Computing - Crating Graphical User interfaces, version 1.

Miller, R., 1992, Normal moveout stretch mute on shallow-reflection data: Geophysics, **57**, 1502–1507.

Novais, M.,1998, Modelamentos de Kirchhoff/Born para propagação de ondas: Tese de Doutorado, Universidade de Campinas.

Novais, A., Santos, L., Tygel, M., and Ursin, B., 1997, A unified Born-Kirchhoff representation for acoustic media: Journal of Seismic Exploration.

Novais, A., Schleicher, J., Santos, L., and Tygel, M., 2003, Born, Kirchhoff, and Born-Kirchhoff modeling: a comparation: 8o Congresso Internacional da Sociedade Brasileira de Geofísica.

Ochoa, A., 2003, InterSis: Uma interface gráfica para modelamento sísmico: Tese de Mestrado, Universidade de Campinas.

Oetiker, T., Partl, H., Hyna, I., and Schlegl, E., april 1999, The not so short introduction to LaTeX2e or LaTeX2e in 87 minutes:, Version 3.7.

Pšenčik, I., Bulant, P., Červený, V., and Klimeš, L., 2001, Ray methods in the modelling of seismic wave fields: 7o Congresso Internacional da Sociedade Brasileira de Geofísica.

Santos, L., Schleicher, J., Tygel, M., and Hubral, P., 1999, Seismic modeling by demigration: Geophysics, **65**, no. 4, 1281–1289.

Santos, L., Schleicher, J., Tygel, M., and Hubral, P., 2000, Modeling, migration and demigration: The Leading Edge.

Schleicher, J., Tygel, M., Ursin, B., and Bleistein, N., 2001, The Kirchhoff-Helmholtz integral for anisotropic elastic media: Wave Motion, **34**, 353–364.

Schleicher, J., Tygel, M., and Hubral, P., 2003, Seismic true-amplitude imaging:.

Schuster, G. Basics of exploration seismology and tomography:. Geology and Geophysics Department of The University of Utah, April 2003.

Tygel, M., Schleicher, J., and Hubral, P., 1996, A unified approach to 3-D seismic reflection imaging, part 2: Theory: Geophysics, , no. 61, 759–775.

Tygel, M., Santos, L., Schleicher, J., and Hubral, P., 1999a, Kirchhoff imaging as a tool for AVO/AVA analysis: The Leading Edge, **18**, no. 8, 940–945.

Tygel, M. and Schleicher, J. and Santos, L. and Hubral, P. 1999b, The kirchhoff-helmholtz integral pair: 4th International Conference on Theoretical and Computational Acoustics.

Červený, V., 2001, Seismic ray theory: Cambridge University Press.

# Apêndice A

# Funções Utilizadas no LabSis

Neste apêndice são listadas todas as funções utilizadas pelo LabSis até a presente data. Ao lado do nome de cada função está o nome do principal autor. As funções estão agrupadas seguindo a estrutura de diretórios do LabSis.

**Diretório raiz**

| | |
|---|---|
| labsis.m | Cristiano da Silva Marcolino |
| setpath.m | Cristiano da Silva Marcolino |

**Pasta display**

| | |
|---|---|
| plotmig.m | Cristiano da Silva Marcolino |
| plotamod.m | Cristiano da Silva Marcolino |
| plotasis.m | Cristiano da Silva Marcolino |
| sliderplot.m | Cristiano da Silva Marcolino |
| wigbsis.m | Cristiano da Silva Marcolino |
| cplot.m | Rodrigo de Souza Portugal |
| wigbsect.m | Rodrigo de Souza Portugal |
| wigbkern.m | Rodrigo de Souza Portugal |
| wigbplot.m | Rodrigo de Souza Portugal |

**Pasta call**

| | |
|---|---|
| acertamenus.m | Cristiano da Silva Marcolino |
| acertamenusmig.m | Cristiano da Silva Marcolino |
| c_edita_modelo.m | Cristiano da Silva Marcolino |
| c_edita_esp_vel.m | Cristiano da Silva Marcolino |
| c_head_su.m | Cristiano da Silva Marcolino |
| c_import.m | Cristiano da Silva Marcolino |
| c_modelagem.m | Cristiano da Silva Marcolino |
| c_modelo.m | Cristiano da Silva Marcolino |
| cal_pop_dommod.m | Cristiano da Silva Marcolino |
| call_vel_esp.m | Cristiano da Silva Marcolino |
| closelabsis.m | Cristiano da Silva Marcolino |
| velocidade.m | Cristiano da Silva Marcolino |

**Pasta geral**

| | |
|---|---|
| derive.m | Lúcio Tunes dos Santos |
| eixos.m | Lúcio Tunes dos Santos |
| eixox.m | Lúcio Tunes dos Santos |
| eixoy.m | Lúcio Tunes dos Santos |
| fonte.m | Lúcio Tunes dos Santos |
| half.m | Lúcio Tunes dos Santos |
| interpol.m | Lúcio Tunes dos Santos |

**Pasta modela**

| | |
|---|---|
| bornsis.m | Cristiano da Silva Marcolino |
| cmphorizsis.m | Cristiano da Silva Marcolino |
| hoffsis.m | Cristiano da Silva Marcolino |
| kirchsis.m | Cristiano da Silva Marcolino |
| raiosis.m | Cristiano da Silva Marcolino |
| born.m | Lúcio Tunes dos Santos |
| kirch.m | Lúcio Tunes dos Santos |
| luz.m | Lúcio Tunes dos Santos |
| raio.m | Lúcio Tunes dos Santos |
| tkoco.m | Lúcio Tunes dos Santos |
| cmphoriz.m | Rodrigo de Souza Portugal |
| ricker.m | Rodrigo de Souza Portugal |
| velrms.m | Rodrigo de Souza Portugal |

**Pasta process**

| | |
|---|---|
| crossplotsis.m | Cristiano da Silva Marcolino |
| nmopanelsis.m | Cristiano da Silva Marcolino |
| taupsis.m | Cristiano da Silva Marcolino |
| velanporsis.m | Cristiano da Silva Marcolino |
| Bestfit.m | Rodrigo de Souza Portugal |
| crossplot.m | Rodrigo de Souza Portugal |
| fitselec.m | Rodrigo de Souza Portugal |
| nmopanel.m | Rodrigo de Souza Portugal |
| nmocorr.m | Rodrigo de Souza Portugal |
| orthdist.m | Rodrigo de Souza Portugal |
| taup.m | Rodrigo de Souza Portugal |
| velanpor.m | Rodrigo de Souza Portugal |

**Pasta dado_mod**

| | |
|---|---|
| altreadsegy.m | Cristiano da Silva Marcolino |
| aquisitionmanual.m | Cristiano da Silva Marcolino |
| count_struct.m | Cristiano da Silva Marcolino |
| edita_modelo.m | Cristiano da Silva Marcolino |
| editar.m | Cristiano da Silva Marcolino |
| estrutura_su.m | Cristiano da Silva Marcolino |
| header.m | Cristiano da Silva Marcolino |
| ibm2ieee.m | Cristiano da Silva Marcolino |
| importasu.m | Cristiano da Silva Marcolino |
| importintersis.m | Cristiano da Silva Marcolino |
| leintersis.m | Cristiano da Silva Marcolino |
| lesegy.m | Cristiano da Silva Marcolino |
| modelo.m | Cristiano da Silva Marcolino |
| readsu.m | Cristiano da Silva Marcolino |
| novo.m | Cristiano da Silva Marcolino |
| salvar.m | Cristiano da Silva Marcolino |
| seta_s.m | Cristiano da Silva Marcolino |

**Pasta imag**

| | |
|---|---|
| demigrasis.m | Cristiano da Silva Marcolino |
| migrasis.m | Cristiano da Silva Marcolino |
| mzosis.m | Cristiano da Silva Marcolino |
| xmigra.m | Cristiano da Silva Marcolino |
| demi.m | Lúcio Tunes dos Santos |
| migra.m | Lúcio Tunes dos Santos |
| mzo.m | Lúcio Tunes dos Santos |
| oco.m | Lúcio Tunes dos Santos |