GOVERNO DO ESTADO
DO RIO DE JANEIRO

UENF
Universidade Estadual do Norte Fluminense Darcy Ribeiro

CCT
LENEP

PROJETO: PROGRAMAS DE VISUALIZAÇÃO TRIDIMENSIONAL DE DADOS GEOFÍSICOS (GEO3D)

Pesquisador: Marco Antônio Rodrigues de Ceia
NOV/2008

Sumário

1) INTRODUÇÃO

Os dados geofísicos são geralmente adquiridos em diversos pontos de medidas (estações), ao longo de uma linha (perfil) com direção determinada. Este tipo de arranjo das estações de medida é comumente empregado em levantamentos de dados de geo-radar (GPR), de sísmica de reflexão e de eletro-resistividade, por exemplo. A miniaturização dos componentes eletrônicos tem tornado os equipamentos geofísicos cada vez mais leves e fáceis de operar, o que aliado a capacidades de armazenamento de dados cada vez maiores, tem propiciado o aumento do número de perfis de dados medidos nos levantamentos de campo. Com isto, a demanda pela utilização de recursos que possam exibir representações tridimensionais do conjunto dados obtidos também tem aumentado. A visualização tridimensional (3D) de dados geofísicos pode ser decisiva para a interpretação desses dados, que podem ser utilizados numa grande variedade de aplicações que envolvam o imageamento da sub-superfície, como por exemplo, a localização de objetos enterrados, como tubulações e cabos, a análise estratigráfica para fins geológicos, o mapeamento de plumas de contaminação em estudos ambientais ou o mapeamento de aqüíferos subterrâneos.

2) METODOLOGIA

A geofísica de exploração tem por objetivo gerar imagens da sub-superfície através de medidas diretas ou indiretas das propriedades físicas das rochas. Geralmente as medidas diretas são realizadas em testemunhos oriundos de poços, enquanto que as medidas indiretas inferem a influência das propriedades físicas das rochas na propagação de ondas ou na variação dos valores dos campos físicos observados. Os principais métodos geofísicos para exploração são os métodos sísmicos, métodos potenciais e os métodos elétricos e eletromagnéticos.

Entre os métodos elétricos podemos destacar a eletro-resistividade a qual utiliza correntes, geradas pelo acoplamento de uma fonte de tensão elétrica com o solo, de modo que sabendo-se o valor desta tensão aplicada, o valor da corrente medido em uma parte do perfil e as distâncias (espaçamentos) entre os eletrodos de injeção e de medida de corrente, pode-se determinar a resistividade em um ponto da sub-superfície. Utilizando-se diferentes espaçamentos é possível mapear a variação da resistividade em profundidade. E a execução destas medidas em várias estações ao longo de um perfil irá mostrar também a variação lateral da resistividade (Telford et al., 1990). A utilização de algoritmos de modelagem direta ou inversa irá gerar uma imagem geo-elétrica da sub-superfície. Entre as diversas aplicações deste método estão: o mapeamento de aqüíferos subterrâneos e de plumas de contaminação. Já o método geo-radar também conhecido como GPR ("*Ground Penetrating Radar*") é um método geofísico eletromagnético, que serve para localizar e mapear alvos em sub-superfície até profundidades de, no máximo, algumas dezenas de metros. Suas aplicações vão desde as áreas da geofísica, geologia, engenharia civil, meio ambiente, e outras áreas (Annan, 1992). Este método usa campos eletromagnéticos nas freqüências entre 10 e 1000 MHz para detectar objetos naturais ou culturais no subsolo, baseando-se na reflexão de ondas planas eletromagnéticas (EM) em estruturas geológicas e feições anômalas presentes em sub-superfície, as quais apresentam um contraste de propriedades elétricas (condutividade e permissividade). Estas ondas são emitidas por uma antena transmissora colocada na superfície, parte delas são refletidas em estruturas em sub-

superfície e captadas por uma antena receptora também disposta na superfície. Uma série de medidas realizadas ao longo de uma linha, quando representadas lado a lado, fornecem uma imagem de alta resolução, tanto lateral quanto vertical, sobre uma seção ao longo do perfil, conhecida como radargrama. Quanto maior o contraste nas propriedades elétricas entre o alvo e o meio encaixante, mais nítida será sua representação no radargrama.

Os programas desenvolvidos por Ceia (2004), em sua tese de doutorado, que foram usados para a visualização tridimensional (3D) de dados GPR, serão o ponto de partida para o conjunto de programas proposto (GEO3D). Estes programas que foram elaborados em MATLAB permitiam a importação de perfis GPR 2D paralelos e isométricos, no formato DT1, a interpolação 3D desses perfis e a visualização 3D dos perfis interpolados.

O MATLAB é uma ferramenta de programação de alto nível, que pode ser usada em várias plataformas como Windows, LINUX e IRIX (MATLAB, 2004). O agrupamento de um variado conjunto de bibliotecas (funções e sub-rotinas) num mesmo produto é um dos pontos fortes desta linguagem, uma vez que facilita sua inclusão nos programas criados pelos usuários. As versões mais atuais contam com um núcleo formado pelo interpretador de comandos ("*workspace*") e um editor de textos para que se possa escrever o programa. Recursos de assistência ("*help*" e "*debugger*") também são disponíveis. O programa pode rodar diretamente no interpretador de comandos sem a necessidade de compilação. Contudo, utilizando-se o módulo de compilação também é possível gerar aplicativos independentes ("*stand-alone*").

Os programas de Ceia (2004), foram aperfeiçoados de modo a permitir a importação de dados geofísicos, nos formatos DT1 e em ASCII. O GEO3D cria uma matriz volumétrica que abrange todos os dados do conjunto de perfis utilizados. Essa matriz terá um valor constante em cada célula (elemento da matriz) e posteriormente o GEO3D associará o valor de cada informação presente nos perfis a células da matriz. O programa também permitirá a interpolação 3D dos dados geofísicos e a gravação da matriz volumétrica num arquivo de dados em disco. Um fluxograma do GEO3D é mostrado na Figura 1. Os recursos de visualização 3D da linguagem de programação MATLAB serão utilizados de maneira a permitir a visualização dos perfis em diagramas de "fence", em "timeslices" e em cubos de dados.

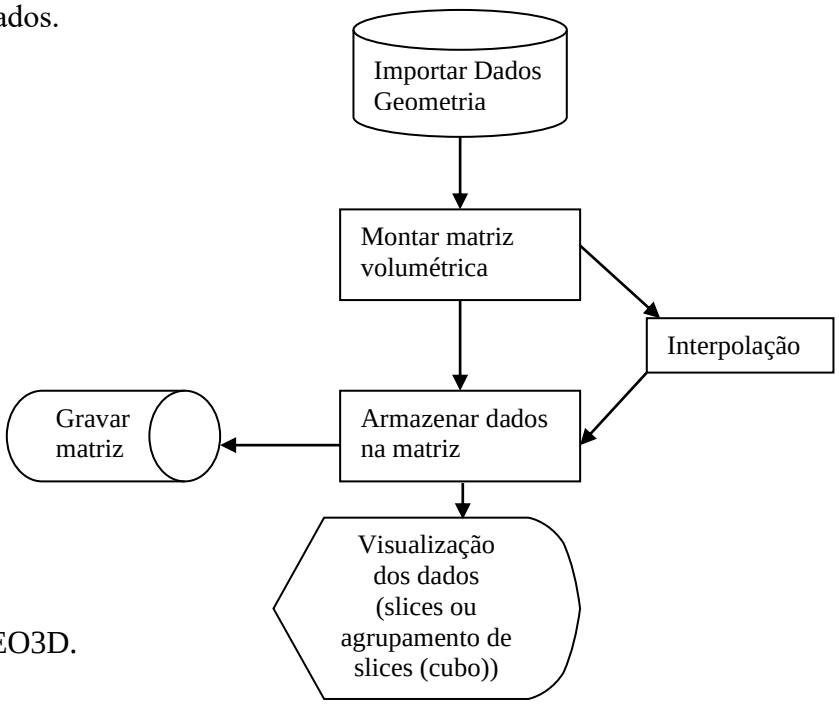

Figura 1 – Fluxograma do GEO3D.

3 – PROGRAMAS

3.1 – PACOTE GEO3D

O GEO3D é um conjunto de programas elaborados em MATLAB para visualização tridimensional de dados geofísicos. A listagem completa dos programas está no anexo A. O programa GPR3D permite importar um conjunto de dados GPR no formato dt1, utilizado pelos equipamentos fabricados pela Sensors & Software.

A etapa inicial consiste em preparar um arquivo de opções (OPT) onde constem as informações descritas abaixo:

- Arquivo que contêm a listagem dos arquivos de dados ⇨ slicefiles.fil
- Número de Traços/Estações ⇨ 37
- Espaçamento entre Traços/Estações ⇨ 0.25
- Espaçamento entre Perfis ⇨ 2
- Quantidade de Arquivos de dados ⇨ 11

Este arquivo OPT é um arquivo de texto (ASCII) que pode ser editado em qualquer editor de texto como o "*notepad*" (Windows) ou o "*gedit*" (LINUX). Da mesma maneira, o arquivo FIL também pode ser editado em qualquer dos editores citados.

O programa GPR3D irá configurar a geometria através dos espaçamentos fornecidos, ler cada perfil e armazenar os dados numa matriz volumétrica e a interpola. Os resultados são as matrizes tridimensionais xi,yi,zi e vi. As 2 primeiras relativas a geometria do levantamento realizado na superfície e a terceira pode ser tempo de trânsito (GPR/sísmica), intervalo AB/2 (resistividade) ou mesmo profundidade (no caso de modelos ou seções em profundidades). A última matriz representa os valores do parâmetro geofísico relativo ao método, como por exemplo: amplitude do campo elétrico no caso do GPR ou a resistividade no caso do método eletro-resistivo.

Para alterar o tamanho do intervalo de interpolação, tem que se alterar a variável "grat" no programa gpr3d. Este intervalo será 1/grat. De modo que para obtermos 1 valor interpolado entre 2 valores medidos, grat terá que ser 2. grat=1 significa que não há interpolação. Menores valores do intervalo de interpolação podem resultar em um maior detalhe da imagem, mas estes valores têm que ser usados de forma criteriosa, de forma a não gerar falsos resultados. A diminuição do intervalo também implica no aumento das matrizes 3D e conseqüentemente irá requerer mais espaço de memória.

Uma listagem do conjunto de programas é descrita na Tabela 1. Eles devem ser executados a partir da linha de comando (*workspace*). A figura 2 mostra um exemplo relativo ao GPR3D. Basta digitar o nome do programa e teclar ENTER. A tela inicial do GPR3D é mostrada na Figura 3. O programa pergunta pelo arquivo de opções (OPT).

O programa PSLICE permite visualizar "*slices*" nas direções x, y e z. No caso de dados de GPR, estes podem ser visualizados em diagramas do tipo cerca "*fence*" (x,y) ou em "*timeslices*" (z).

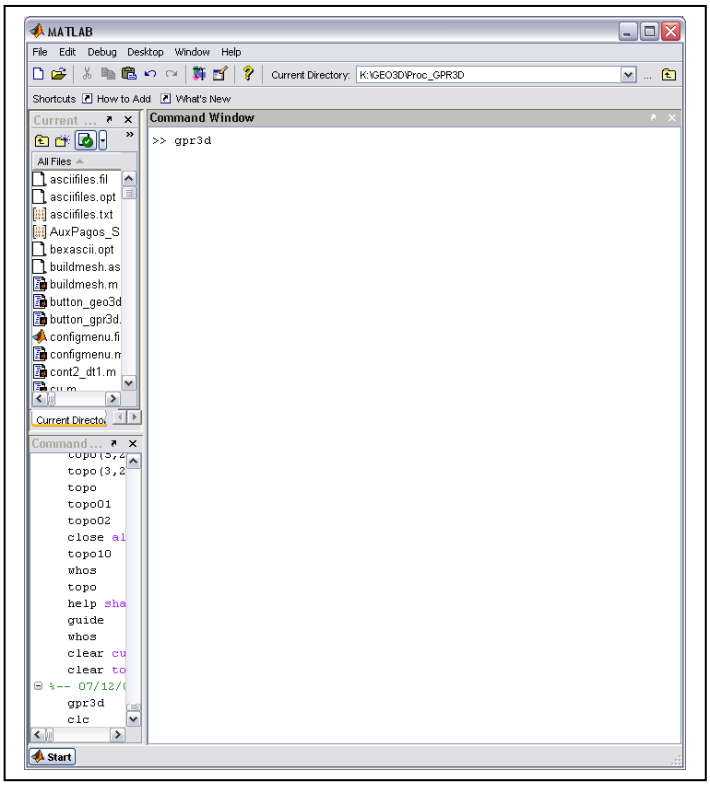

Figura 2 – Ilustração das janelas do MATLAB. Para rodar o GPR3D, basta digitar este nome na linha de comando e pressionar a tecla ENTER.

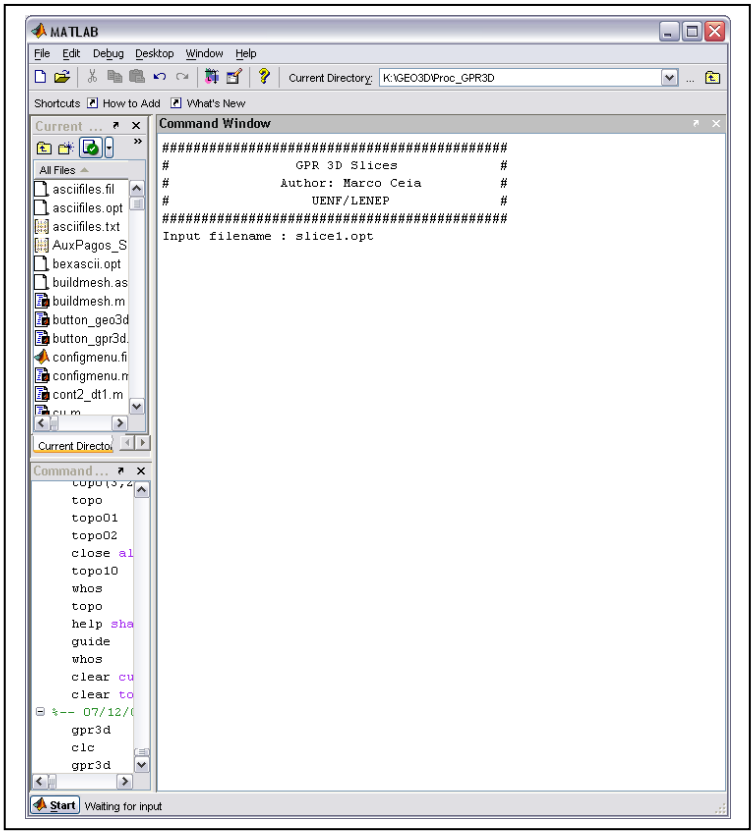

Figura 3 – Ilustração do início do programa GPR3D.

Para executar o PSLICE é preciso usar a sintaxe:

pslice(xi,yi,zi,vi,[inl],[crossl],[timeslice])

Onde [inl],[crossl] e [timeslice] são os intervalos a serem plotados.

Para ilustração, vamos usar um conjunto de dados GPR obtidos em afloramentos turbidíticos da Bacia de Almada (Ceia,2004). É um conjunto de dados com 11 perfis, com espaçamento entre traços de 0,25 m e espaçamento entre perfis de 2 m. Estes dados foram importados com o GPR3D e interpolados (grat=2 para y e grat=1 para x e z). Para plotarmos "*slices*" em x=5 m e o perfil em y=10 m, podemos usar:

pslice(xi,yi,zi,vi,[5],[10],[]) .

O resultado é mostrado na figura 4. "*Inline*" significa a direção dos perfis adquiridos (x) e "*crossline*" a direção transversal a estes perfis (y). Neste caso a o eixo vertical é o tempo de trânsito.

Note que os valores têm que ser definidos dentro dos []. Se nenhum valor estiver definido dentro desses [], significa que nenhum "*slice*" será plotado nesta direção.

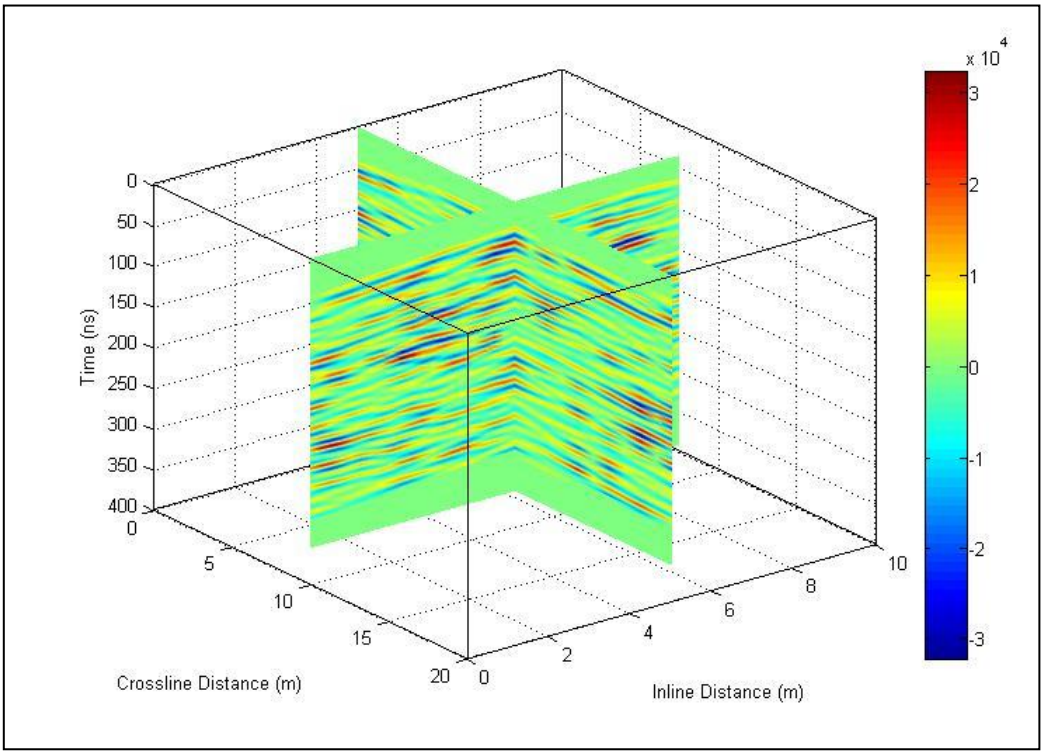

Figura 4 – Exemplo de um diagrama do tipo "*fence*" gerado com o PSLICE, a partir de dados de GPR obtidos na Bacia de Almada.

Para gerar uma "*timeslice*", por exemplo, em 100 ns, podemos usar:

pslice(xi,yi,zi,vi,[],[],[100])

O resultado é mostrado na figura 5. Desta forma a partir da interpolação 3D de perfis 2D, construímos um horizonte temporal. Mais de uma "*timeslice*" pode ser exibida de uma só vez como mostra a Figura 6.

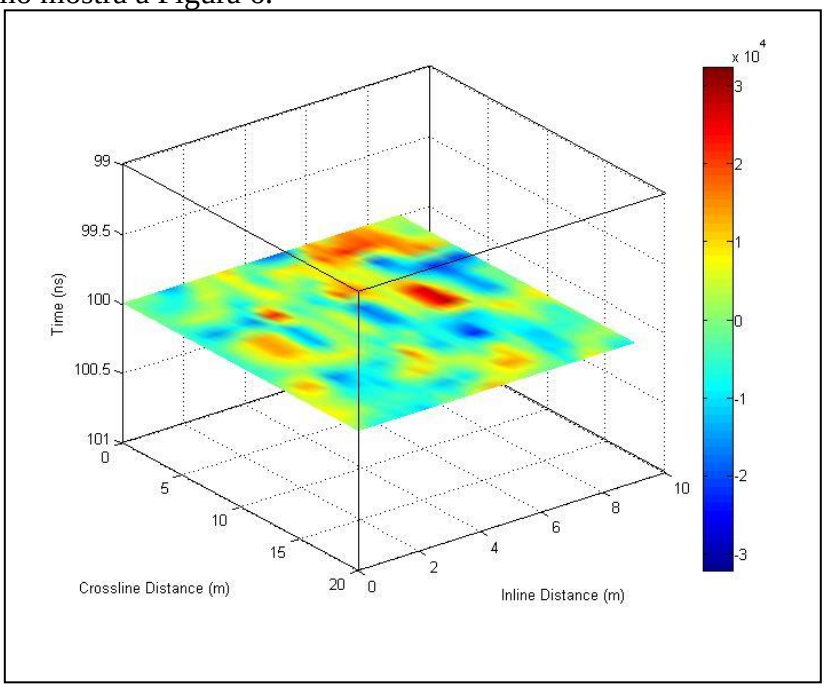

Figura 5 – "*Timeslice*" gerada pelo PSLICE. T=100 ns.

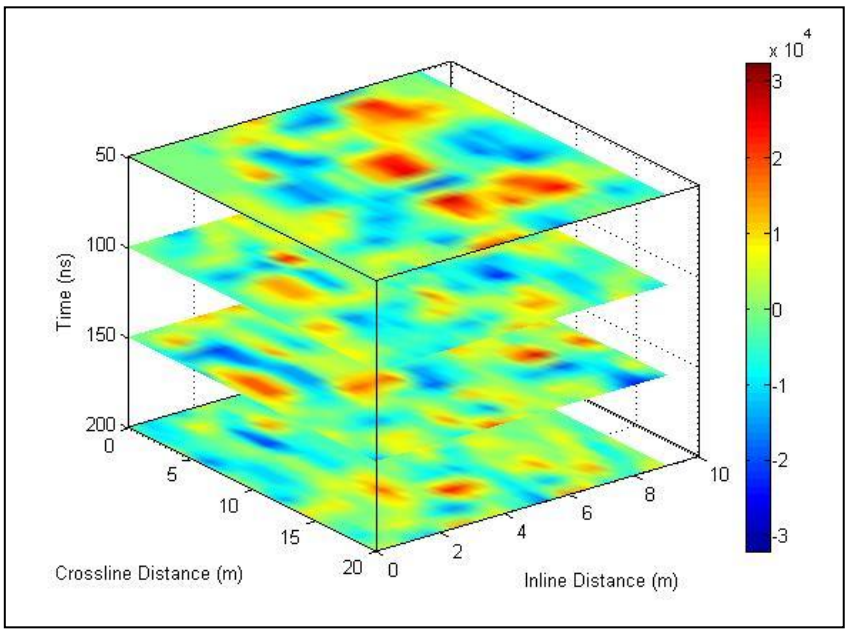

Figura 6 - Múltiplas "*timeslices*" gerada pelo PSLICE. T=50,100,150 e 200 ns. pslice(xi,yi,zi,vi,[],[],[50 100 150 200])

O PSLICE também pode exibir combinações de "*slices*" em diferentes direções, como mostrado pela Figura 7. O agrupamento destes "*slices*" pode produzir um cubo 3D, como mostrado na Figura 8.

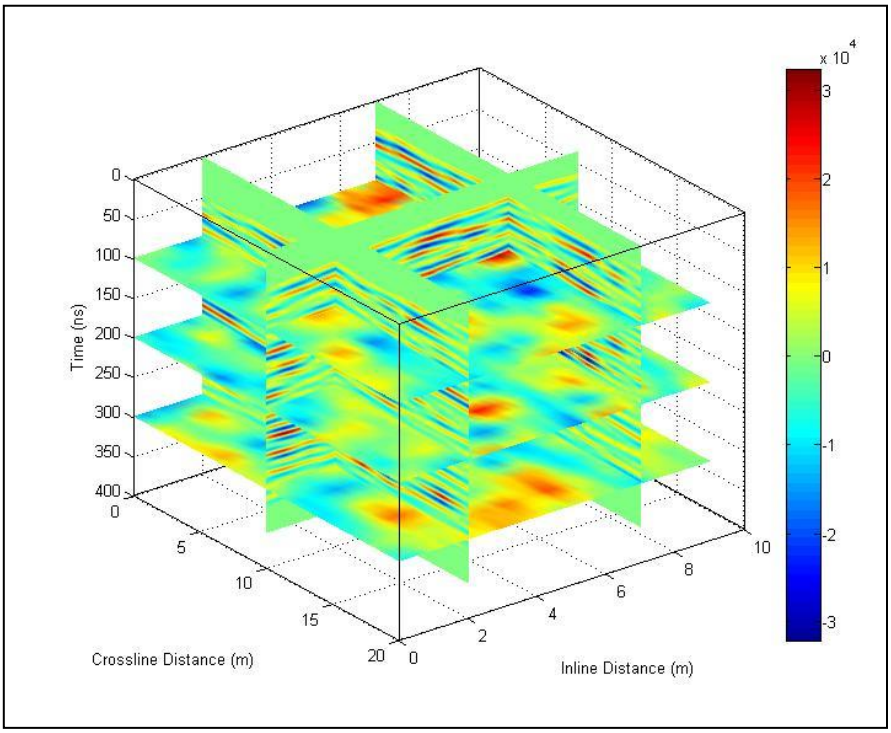

Figura 7 – "*Slices*" em diferentes direções. pslice(xi,yi,zi,vi,[2 7],[10],[100 200 300])

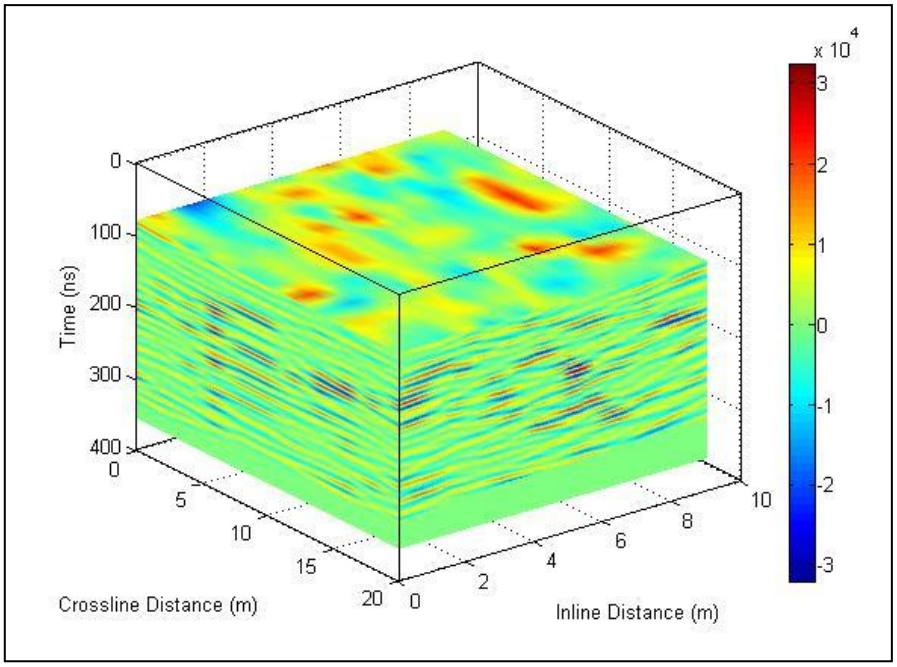

Figura 8 – Cubo 3D gerado pelo PSLICE através de um agrupamento de "*slices*". pslice(xi,yi,zi,vi,[0:3],[],[80:400])

O comando save2mat permite salvar a matriz num arquivo *.MAT. Basta usar a sintaxe:

saveg2mat(filename,xi,yi,zi,vi)

Os arquivos *.MAT podem ser carregados com o programa loadgmat, usando-se a sintaxe:

loadgmat(filename,xi,yi,zi,vi)

Obs: **Não** colocar a extensão ".MAT" no final.

O programa GEO3D importa dados em formato ASCII e faz o mesmo que o GPR3D. A Figura 9 mostra um exemplo de visualização de um modelo geo-elétrico sintético.

Para selecionar um intervalo de amplitudes a ser visualizada pode se usar o script famprange. Desta forma os valores da matriz interpolada (**vi**) que estiverem fora do intervalo de amplitudes são assumidos como NaN e uma nova matriz (**avi**) é gravada. Então ao se utilizar o programa PSLICE para plotar "*slices*" da matriz **avi**, os valores fora do intervalo selecionado não serão exibidos. A Figura 10 mostra um exemplo para o mesmo conjunto de dados mostrado na Figura 9. É preciso re-configurar a escala de cores manualmente, através da opção *colormap editor* do MATLAB.

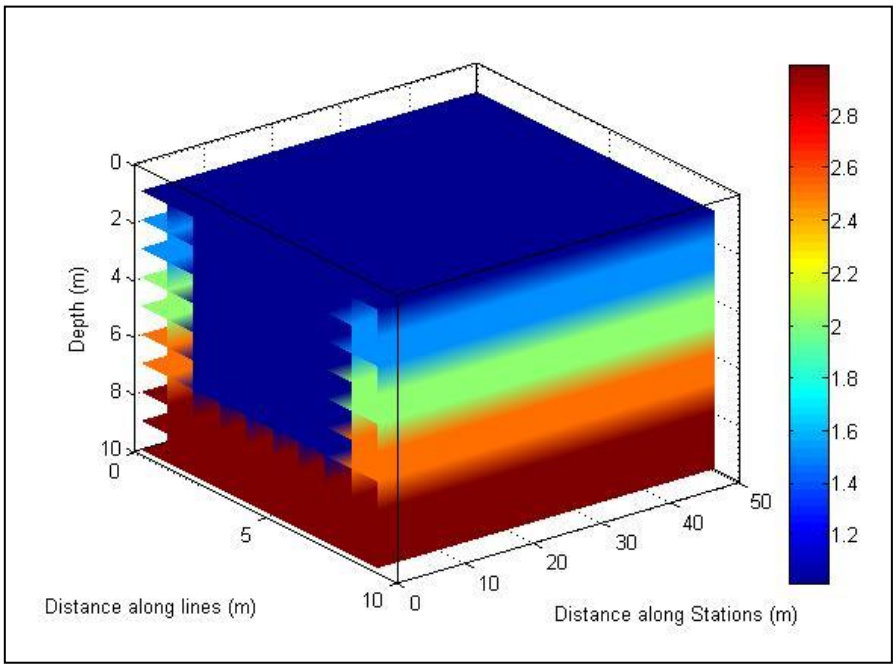

Figura 9 – "*Slices*" de um modelo geoelétrico sintético. A escala de cores representa log(Resistividade). O conjunto de dados foi "interpolado" com o GEO3D (grat=1).

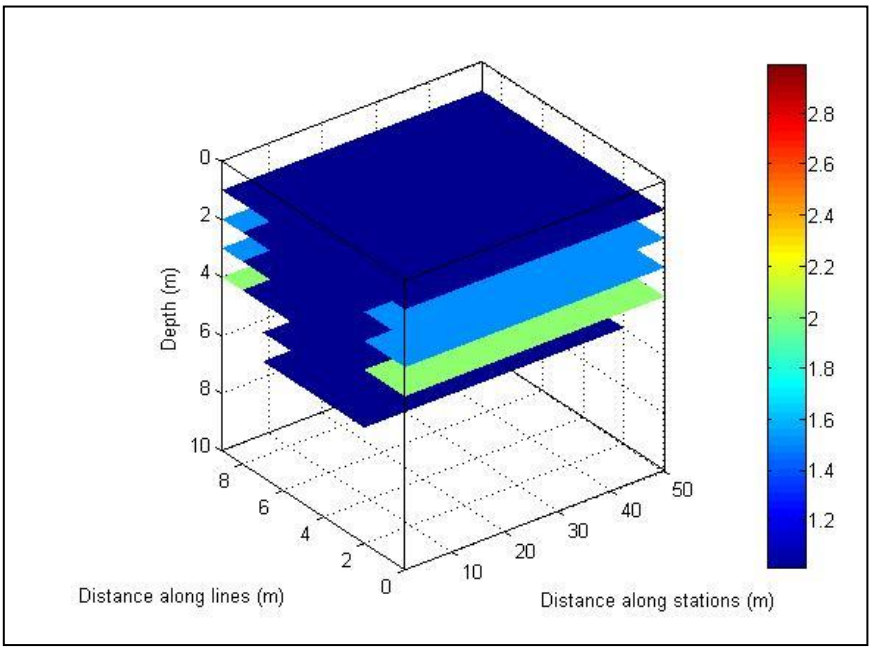

Figura 10 – Conjunto de dados da Figura 9 submetidos a uma seleção de intervalo de amplitude. Neste caso a amplitude é log(Resistividade) e o intervalo selecionado foi 1-2.

GPR3D	Lê os dados em formato dt1, monta matriz volumétrica e faz interpolação 3D.
GEO3D	Lê os dados em formato ASCII, monta matriz volumétrica e faz interpolação 3D.
pslice	Plota "*slices*" dos dados interpolados (matriz vol.).
saveg2mat	Armazena os dados interpolados (matriz vol.).
loadgmat	Carrega os dados interpolados (matriz vol.).
famprange	Exibe "slices" somente com valores de amplitude dentro de um intervalo selecionado.

Tabela 1 – Lista do conjunto de programas GEO3D

3.2 – "*Graphical User Interface*" (GUI) GEO3D

Neste projeto também foi desenvolvida uma interface gráfica (GUI) para o conjunto de programas GEO3D. A Figura 11 mostra a tela inicial da interface, a qual pode ser acessada digitando geo3dinit na janela de comandos do MATLAB e pressionando ENTER.

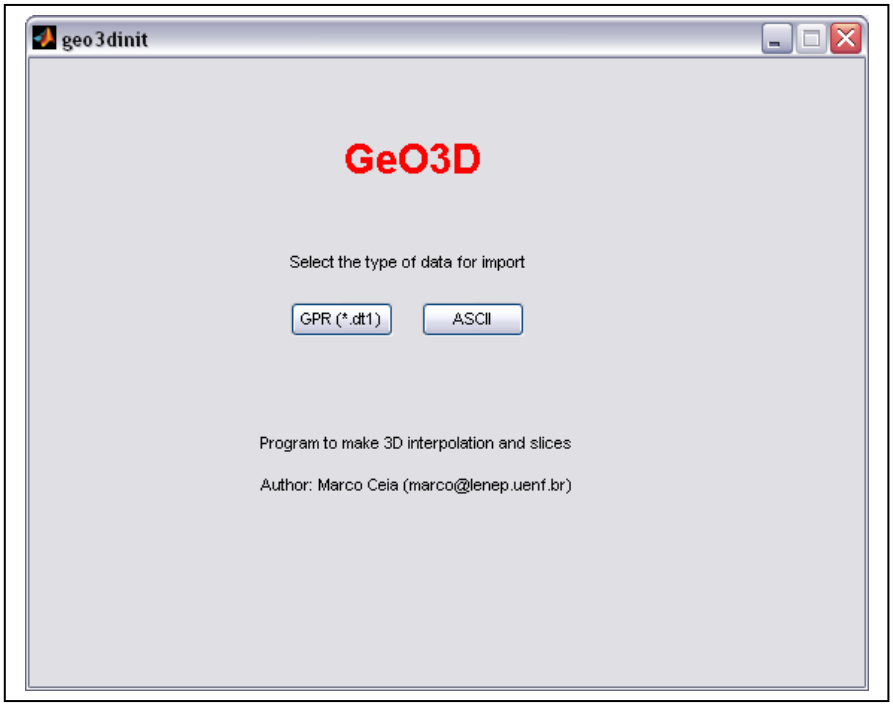

Figura 11- Tela inicial da interface gráfica do GEO3D.

Esta tela inicial permite ao usuário escolher qual o tipo de conjuntos de dados a serem lidos. Se o tipo escolhido for o formato dt1, aparece uma tela como a mostrada na Figura 12.

Existem dois modos para entrada dos dados a serem usados para plotar os "*slices*". O primeiro consiste em carregar um arquivo OPT e a partir daí o programa lerá os arquivos de dados listados no arquivo FIL correspondente. A opção EDIT OPT permite que se altere o arquivo OPT selecionado.

Uma vez carregado, pode-se configurar os intervalos de interpolação (GRID STEPS) e depois realizar esta interpolação 3D, pressionando-se a tecla INTERPOLATE. A partir daí a matriz vi com os dados interpolados estará pronta. Esta matriz poderá ser salva em disco usando-se a opção SAVE TO MAT.

O segundo modo de entrada dos dados que serão usados para plotar os "slices" é justamente carregar um arquivo MAT salvo anteriormente, usando-se a opção LOAD MAT a qual abre uma janela (Figura 13) que pergunta qual o nome do arquivo MAT a ser carregado.

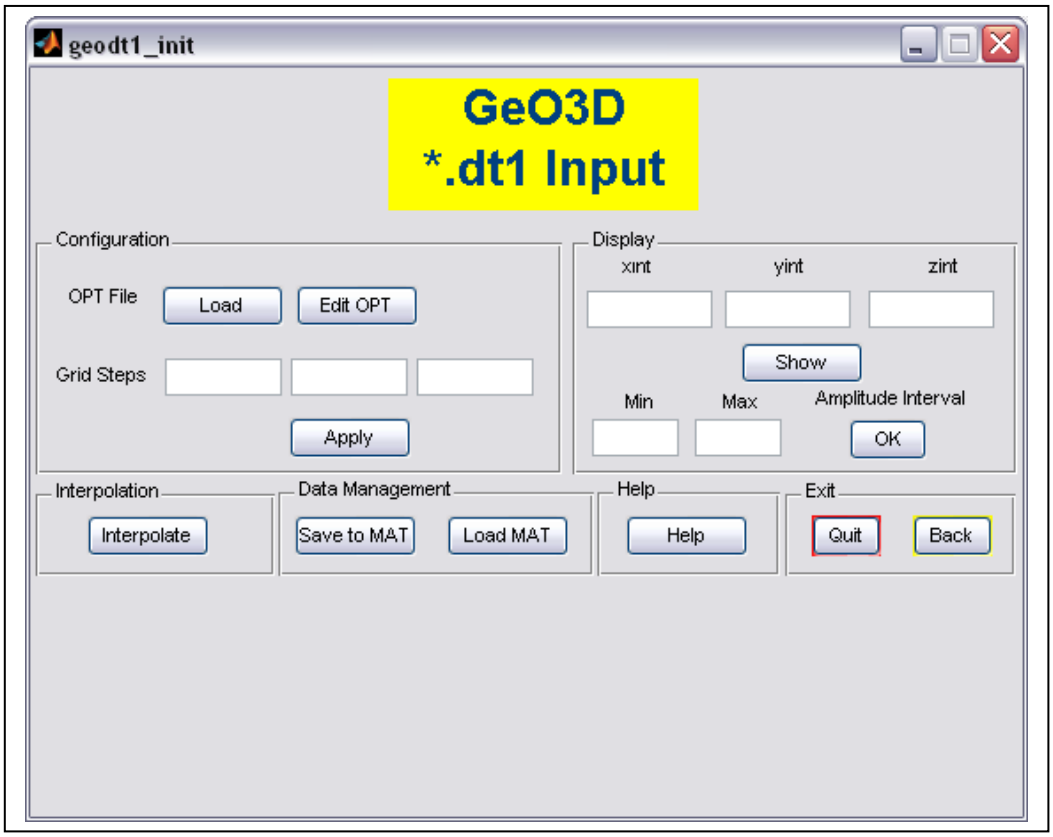

Figura 12 – Tela da interface gráfica para arquivos no formato dt1.

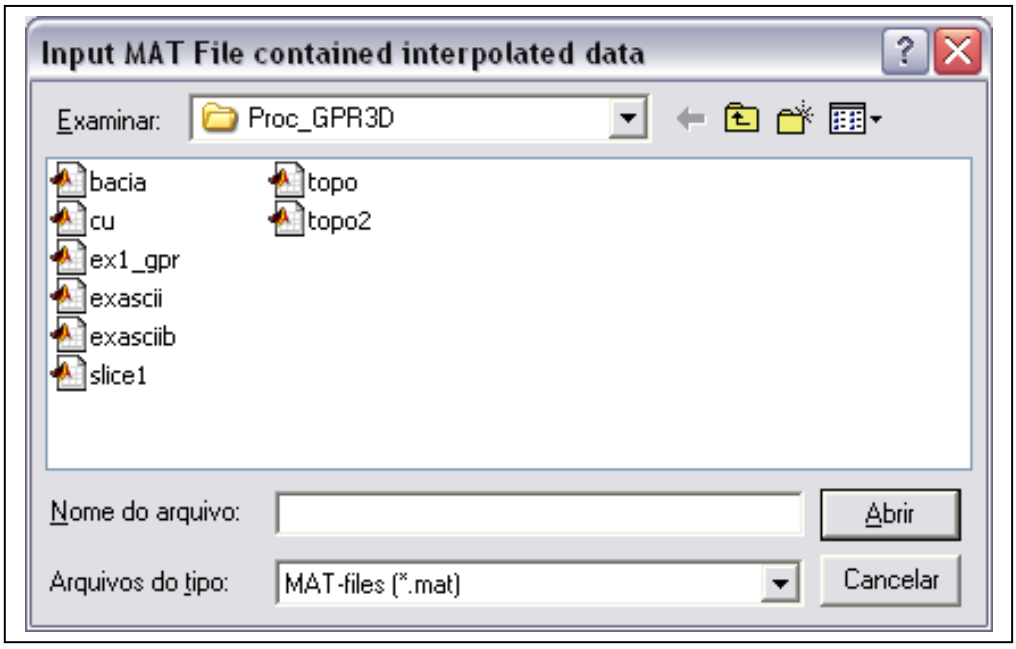

Figura 13 – Tela para seleção do arquivo MAT a ser carregado.

Para plotar os "*slices*", deve-se primeiro configurar os intervalos xint, yint e zint com as "*slices*" que queremos plotar. Por exemplo xint=[], yint=[] e zint=[100 200].

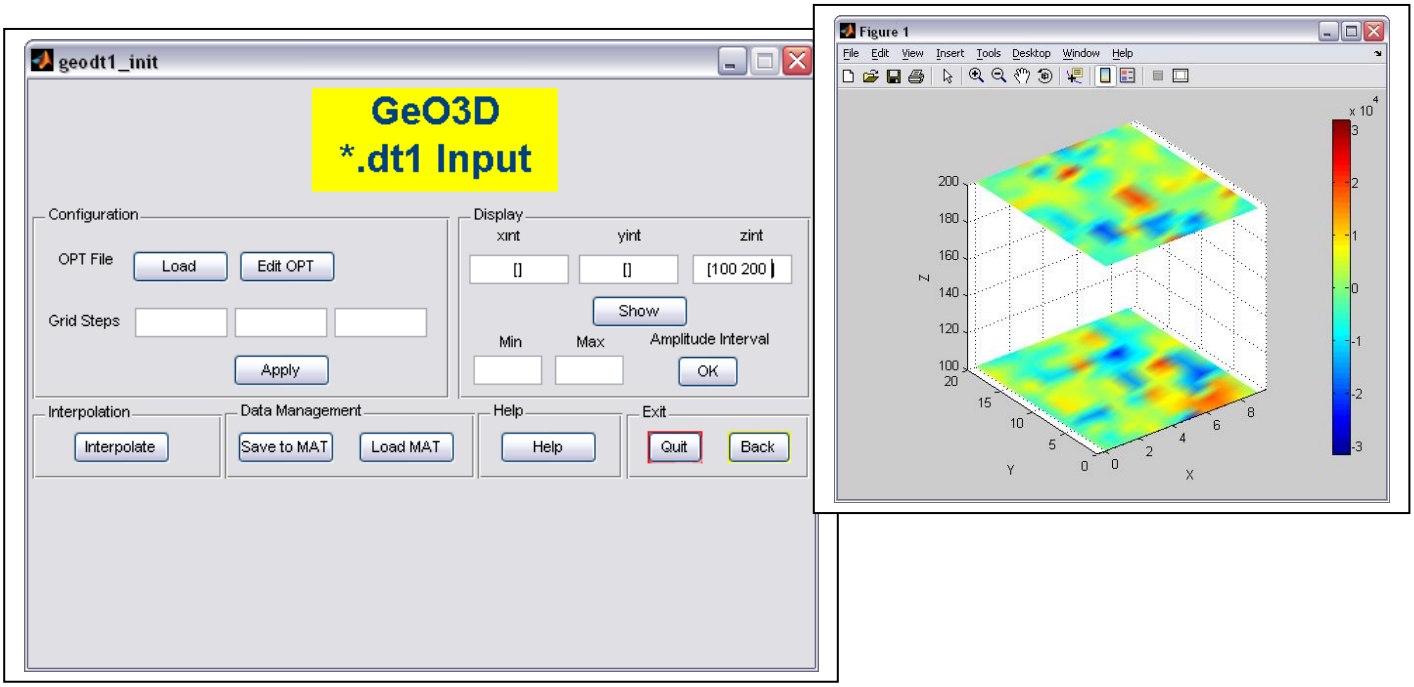

Figura 14 – Painel Esquerdo: Exemplo de configuração dos intervalos xint,yint e zint. Painel Direito: Tela Exibida ao se pressionar a tecla SHOW.

Pode-se selecionar também um intervalo de amplitudes a serem plotadas, usando-se as opções de AMPLITUDE INTERVAL. Preenchendo-se os campos min e max e pressionado-se a tecla OK.

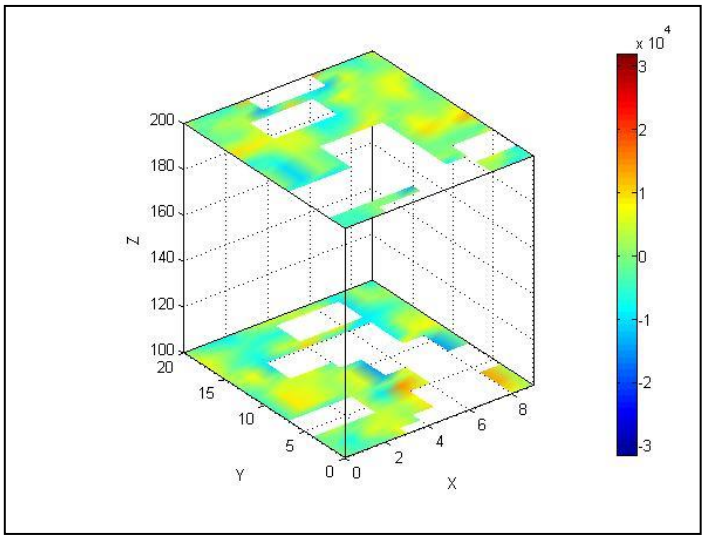

Figura 15 – "Slices" 100 e 200 do exemplo anterior com intervalo de amplitudes selecionado de –15000 a 15000.

A tecla HELP ao ser pressionada exibe uma tela com informações das opções do GUI, tal qual como um guia rápido ("*quick start*"). A tecla BACK volta para a tela inicial (Figura 11). E a tecla QUIT sai do programa.

A tela com a opção de entrada de dados no formato ASCII (figura 16), é parecida com a do formato DT1, e funciona da mesma maneira.

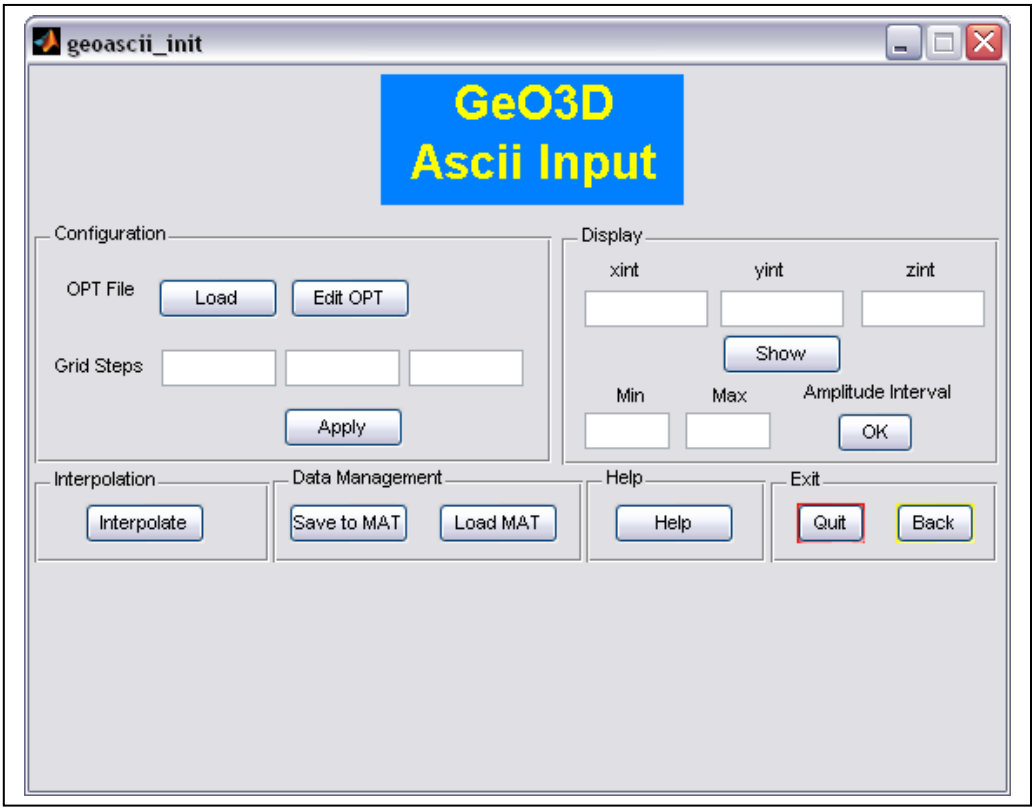

Figura 16 - Tela da interface gráfica para arquivos no formato ASCII.

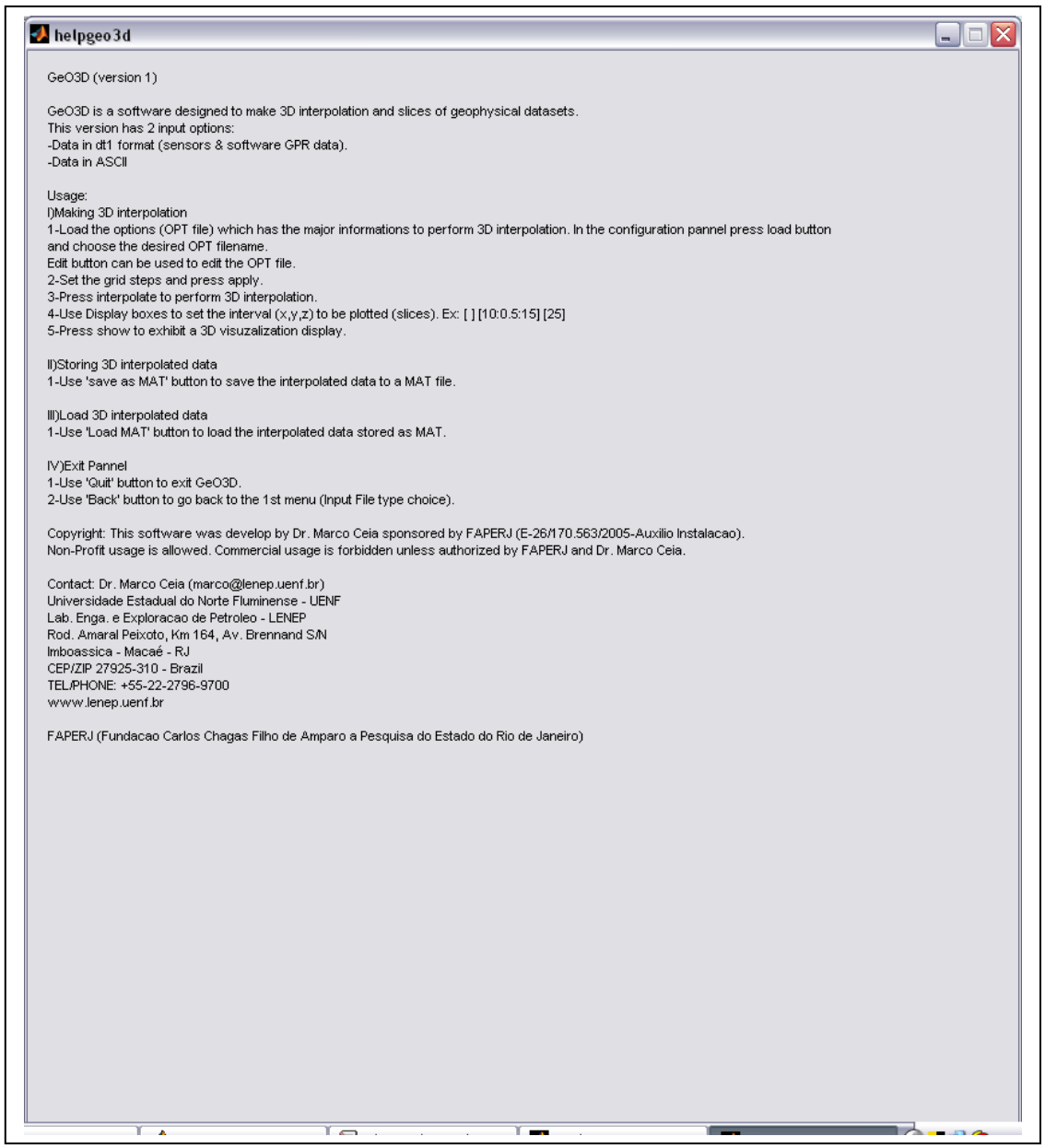

Figura 17 – Tela exibida ao pressionar-se a tecla HELP.

4)USANDO OS RECURSOS DO MATLAB

O MATLAB têm disponíveis vários recursos extremamente úteis para as demandas de visualização 3D. A Figura 18 mostra alguns botões para acessar os recursos mais utilizados, como por exemplo: comandos de rotação, edição e zoom.

4.1)Rotação

Pressionando-se o botão de rotação pode-se girar a figura livremente. Isto possibilita observar a figura de vários ângulos e desta forma auxilia o geofísico na interpretação de estruturas sub-superficiais. A Figura 19 ilustra um exemplo de rotação.

4.2)Zoom

Este comando permite que se amplie (“*zoom in*”) ou reduza (“*zoom out*”) o tamanho da imagem exibida.

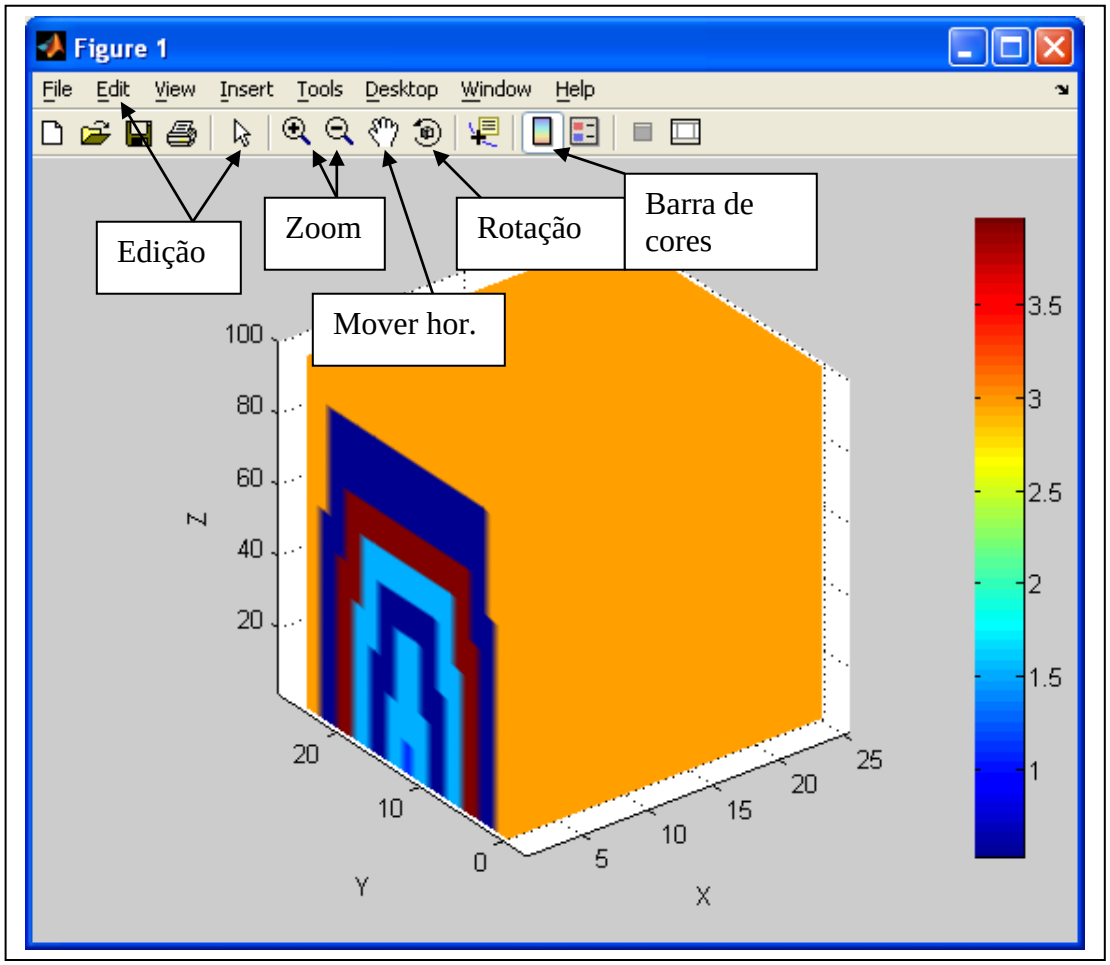

Figura 18 – Atalhos para ativação dos principais recursos do MATLAB.

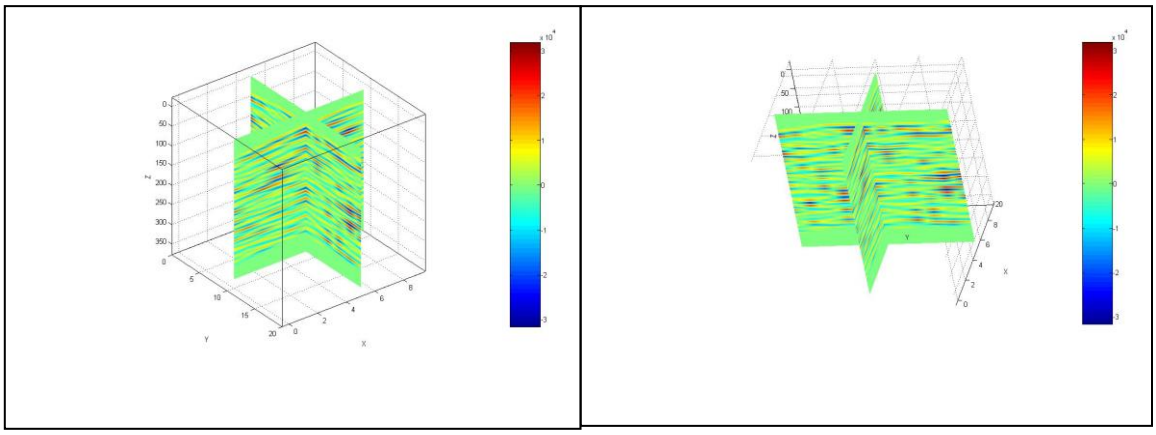

Figura 19 – Ilustração mostrando o efeito da rotação na figura dos dados GPR.

4.3) Edição

Este recurso permite que se edite parâmetros do eixo como: espaçamento, título, cor do eixo (Figura 20). Também permite editar a escala de cores (Figura 21), linhas 3D e uma série de outras propriedades.

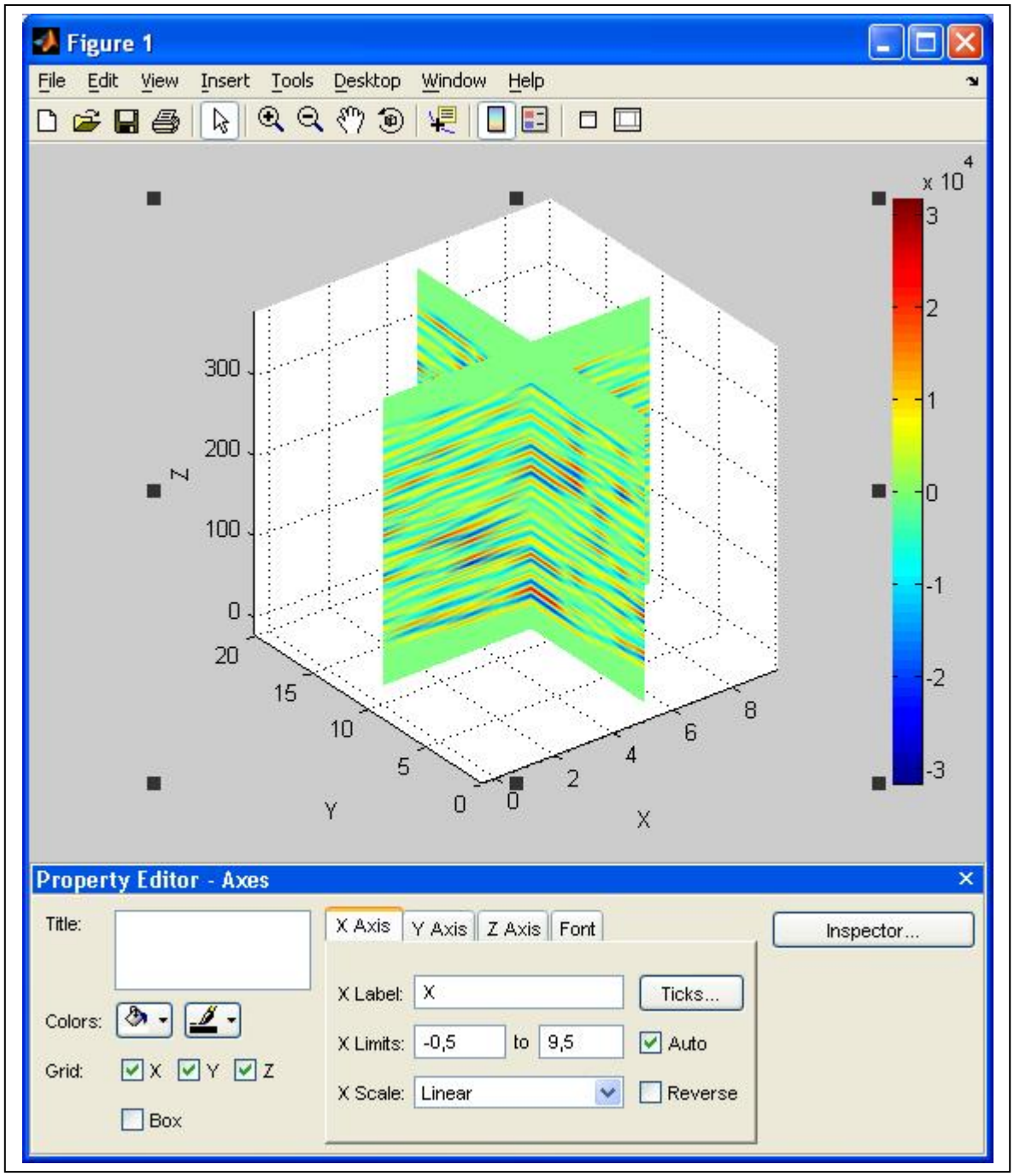

Figura 20 – Ilustração da tela do editor de propriedades.

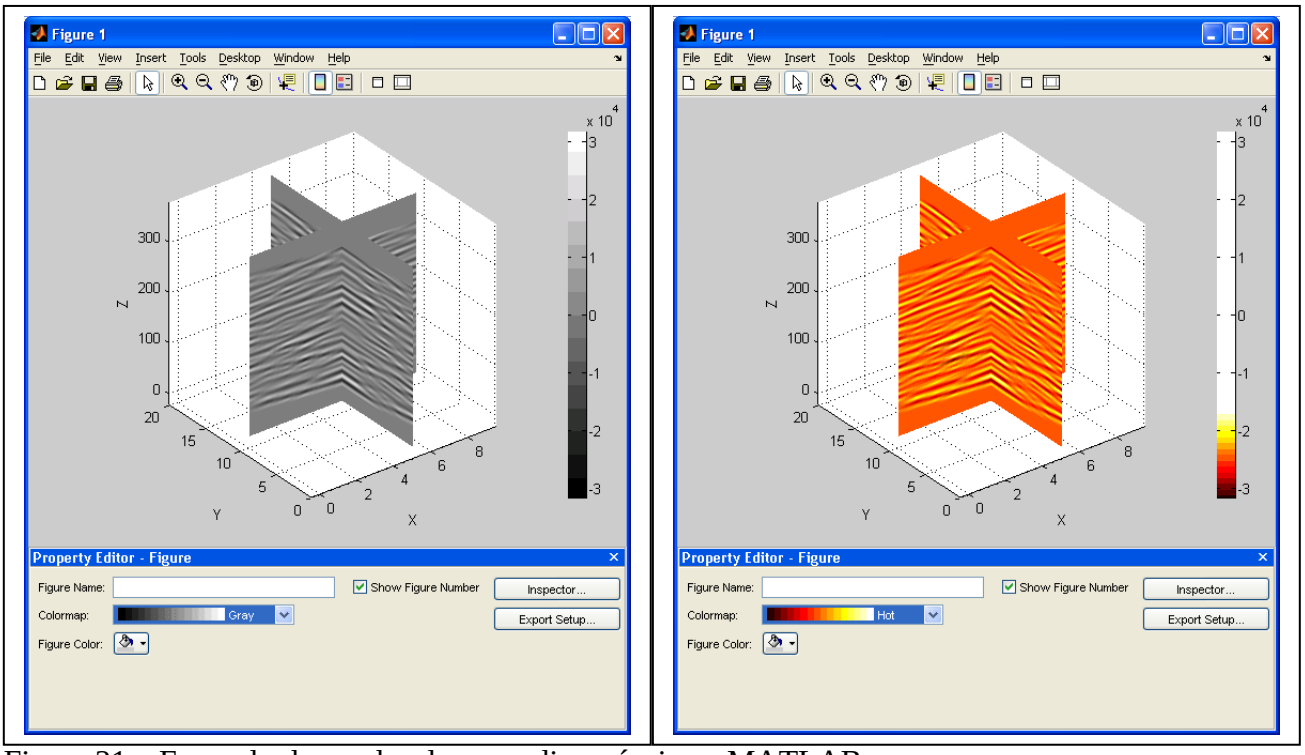

Figura 21 – Exemplo de escalas de cores disponíveis no MATLAB.

5) LIMITAÇÕES

A principal limitação dos programas GEO3D é que a entrada deve ser de perfis com o mesmo tamanho. Por isso, às vezes têm se excluir alguns traços/estações dos perfis. Outra limitação é que ainda são poucas as mensagens de erro. Há uma limitação de memória também. Foi realizado um teste com o conjunto de dados de GPR da Bacia de Almada, utilizado na Figura 20, de modo a plotar "*timeslices*" de 50 ns a 250 ns, com intervalo variável ([50:intz:250]). O máximo que se conseguiu foi intz=0.01, no qual foram ocupado 5046336 bytes da memória, só com os dados. Mesmo utilizando as opções de otimização da memória disponíveis no MATLAB.

6) CONCLUSÃO

O desenvolvimento dos programas de visualização tridimensional de dados geofísicos (GEO3D), permitirá o acesso de pesquisadores, estudantes e profissionais das áreas de geociências a um "*software*" gratuito e de fácil operação, que terá impacto em suas interpretações dos dados geofísicos e com isso irá contribuir para um imageamento mais realístico da sub-superfície. Desta forma, essas imagens e interpretações irão facilitar a tomada de decisões que impliquem na perfuração da sub-superfície, seja em situações de forma a evitar que esta cause danos em tubulações enterradas, ou na prospecção de água subterrânea, ou em obras que visem evitar que uma determinada pluma de contaminação atinja um aqüífero utilizado para prover água para a população.

O fato dos programas terem o código aberto permite aos pesquisadores a inserção destes códigos em seus próprios programas, facilitando a observação de imagens 4D, de simulações de fluxo e escoamento em meios porosos, ou mesmo dos resultados de

algoritmos de processamento 3D. Sendo assim, estes programas poderão beneficiar o desenvolvimento de "*softwares*" que necessitem deste tipo de representação gráfica.

Esperamos que as novas versões venham a melhorar o desempenho ao superar as limitações observadas.

7) REFERÊNCIAS

ANNAN, A.P.,1992. *Ground Penetrating Radar Workshop Notes*: Sensors & Software Inc., Canada.

CEIA, M.A.R, 2004. *Estudo dos Afloramentos Turbidíticos da Bacia de Almada (BA) com imagens de Geo-Radar (GPR)*. Tese de Doutorado. Universidade Estadual do Norte Fluminense/ Lab. de Engenharia e Exploração de Petróleo. Macaé, RJ. 178 pp.

MATLAB, 2004. Mathworks Website. www.mathworks.com

TELFORD, W.M., GELDART, L.P., SHERIFF, R.E., 1990. *Applied Geophysics. Second Edition*. Cambridge University Press.

8) AGRADECIMENTOS

Agradeço a FAPERJ pelo financiamento deste trabalho, resultante do auxílio-instalação anteriormente mencionado, ao LENEP/UENF pela infra-estrutura que possibilitou a realização deste trabalho e ao Prof. Jadir da Conceição Silva (*in memoriam*) (UFRJ) por suas valiosas dicas e suporte.

APÊNDICE A – PROGRAMAS GEO3D

A.1-GPR3D.m

```
%Program GPR3D
%Author: Marco Ceia (marco@lenep.uenf.br) - UENF/LENEP
%v.3 - NOV/2007
%This program reads a set of 2D GPR lines, make a 3D interpolation,
%store the data in a volume matrix and display plan slices.
%It works only with Sensors & Software datatype.
%Sintax: gpr3d (When the program ask for input options file, enter the *.opt filename and path
%Ex. slice1.opt
%
clear all
clc
disp('##############################################');
disp('#              GeO 3D Slices            #');
disp('#               DT1 Input             #');
disp('#             Author: Marco Ceia         #');
disp('#               UENF/LENEP             #');
disp('##############################################');
%
filein1=input('Input filename : ','s');%ASCII file with input options
%Input Parameters
a=textread(filein1,'%s');
filein2=char(a(1));%GPR (*.dt1) filename
ntrace=str2num(char(a(2)));%Number of traces
stepin=str2num(char(a(3))); %(Step size (offset) - Inline)
stepcross=str2num(char(a(4))); %(Step size (offset) - Crossline)
nfiles=str2num(char(a(5)));%Number of GPR files
%
if nfiles>1
   %Loading GPR filenames
   a=textread(filein2,'%s');
   for i=1:nfiles
      gprfile(i,:)=char(a(i));
   end
   %Loading first GPR file for test
   %xin = inline x-coord
   %tin = inline time
   [xin,tin,data,dt]=fscandt1(gprfile(1,:),ntrace,stepin);
   [ldata,cdata]=size(data);
   clear data;
   clear a;
   %
   %Crossline Coordinates
```

```
   %xcross = crossline x-coord
   for i=1:nfiles;xcross(i)=(i-1)*stepcross;end;
   %
   %Loading data
   for l=1:nfiles
      disp(['Reading File #',num2str(l),' :',gprfile(l,:)]);
      [xin,tin,data,dt]=fscandt1(gprfile(l,:),ntrace,stepin);
      for j=1:ldata
         for i=1:cdata
            data2(l,i,j)=data(j,i);
            x(l,i,j)=xin(i);
            y(l,i,j)=xcross(l);
            z(l,i,j)=tin(j);
         end
      end
      clear data;
   end

%Grid Setup
disp('------------------------------------------');
grat=2;%Grid Ratio
disp(['Grid Ratio = 1/',num2str(grat)]);
xmax=max(xin);xmin=min(xin);%xstp=stepin/grat;
xstp=stepin/1;
ymax=max(xcross);ymin=min(xcross);ystp=stepcross/grat;
zmax=max(tin);zmin=min(tin);zstp=dt;
disp(['Grid Cell Size X = ',num2str(xstp)]);
disp(['Grid Cell Size Y = ',num2str(ystp)]);
disp(['Grid Cell Size Z = ',num2str(zstp)]);
disp('------------------------------------------');

%Memory Cleanup
clear filein1;
clear gprfile;
clear ldata;
clear cdata;
clear i;
clear j;
clear l;

%Building Mesh
disp('Building Mesh');
[xi,yi,zi]=meshgrid(xmin:xstp:xmax,ymin:ystp:ymax,zmin:zstp:zmax);

%Memory Cleanup
clear filein2;
clear zmax;
```

```
clear zmin;
clear zstp;
clear stepin;
clear stepcross;
clear xstp;
clear ystp;
clear xmax;
clear xmin;


%Making interpolated volume (vi)
disp('3D Interpolation');
vi=interp3(x,y,z,data2,xi,yi,zi);

%Memory Cleanup
clear x;
clear y;
clear z;
clear data2;
%Consolidate workspace memory
cwd = pwd;
cd(tempdir);
pack
cd(cwd)
%

else
   disp('Not enough files available');
end
```

A.2-GEO3D.m

```
%Program Geo3Db
%Author: Marco Ceia (marco@lenep.uenf.br) - UENF/LENEP
%v.2 - NOV/2007
%This program reads a set of 2D lines, in ASCII format, makes a 3D interpolation,
%store the data in a volume matrix and display plan slices.
%It works only with Sensors & Software datatype.
%Sintax: geo3d (When the program ask for input options file, enter the *.opt filename and path
%Ex. exascii.opt
%
clear all
clc
disp('##############################################');
disp('#              GeO 3D Slices            #');
disp('#               ASCII Input             #');
```

```
disp('#             Author: Marco Ceia         #');
disp('#               UENF/LENEP               #');
disp('##########################################');
%
filein1=input('Input filename : ','s');%ASCII file with input options
%Input Parameters
a=textread(filein1,'%s');
filein2=char(a(1));%GPR (*.dat) filename
ntrace=str2num(char(a(2)));%Number of traces
stepin=str2num(char(a(3))); %(Step size (between stations) - Inline)
stepcross=str2num(char(a(4))); %(Step size (between lines) - Crossline)
nfiles=str2num(char(a(5)));%Number of data files
%
if nfiles>1
   %Loading data filenames
   a=textread(filein2,'%s');
   for i=1:nfiles
      gprfile(i,:)=char(a(i));
   end
   temp1=load(gprfile(1,:));
   data=temp1(:,:);
   [ldata,cdata]=size(data);
   clear data;
   clear a;
   dt=1;%Delta time (GPR). Step size in z.
   %Crossline Coordinates
   %xcross = crossline x-coord
   for i=1:nfiles;xcross(i)=(i-1)*stepcross;end;
   %
   %Loading data
   for l=1:nfiles
      disp(['Reading File #',num2str(l),' :',gprfile(l,:)]);
      temp1=load(gprfile(l,:));
      xin=1:cdata;tin=1:ldata;data=[temp1(:,:)];
      for j=1:ldata
         for i=1:cdata
            data2(l,i,j)=data(j,i)';
            x(l,i,j)=xin(i);
            y(l,i,j)=xcross(l);
            z(l,i,j)=tin(j);
         end
      end
      clear data;
   end

%Grid Setup
disp('------------------------------------------');
```

```
grat=1;%Grid Ratio
disp(['Grid Ratio = 1/',num2str(grat)]);
xmax=max(xin);xmin=min(xin);xstp=1;%xstp=stepin/grat;
ymax=max(xcross);ymin=min(xcross);ystp=1;%ystp=stepcross/grat;
zmax=max(tin);zmin=min(tin);zstp=dt%ystp set to be equal to dt
disp(['Grid Cell Size X = ',num2str(xstp)]);
disp(['Grid Cell Size Y = ',num2str(ystp)]);
disp(['Grid Cell Size Z = ',num2str(zstp)]);
disp('-----------------------------------------');

%Memory Cleanup
clear filein1;
clear gprfile;
clear ldata;
clear cdata;
clear i;
clear j;
clear l;

%Building Mesh
disp('Building Mesh');
[xi,yi,zi]=meshgrid(xmin:xstp:xmax,ymin:ystp:ymax,zmin:zstp:zmax);

%Memory Cleanup
clear filein2;
clear zmax;
clear zmin;
clear zstp;
clear stepin;
clear stepcross;
clear xstp;
clear ystp;
clear xmax;
clear xmin;


%Making interpolated volume (vi)
disp('3D Interpolation');
vi=interp3(x,y,z,data2,xi,yi,zi);

%Memory Cleanup
clear x;
clear y;
clear z;
clear data2;
%Consolidate workspace memory
cwd = pwd;
```

```
        cd(tempdir);
        pack
        cd(cwd)
        %

        else
           disp('Not enough files available');
end
```

A.3-PSLICE.m

```
        function [ps]=pslice(xi,yi,zi,vi,inl,crossl,timeslice)
        %Program to display GPR 3D slices
        %Author: Marco Ceia - UENF/LENEP
        %syntax: [ps]=pslice(xi,yi,zi,vi,[inl],[crossl],[timeslice])
        %xi,yi,zi must be generated through meshgrid
        %vi is the 3D GPR data generated by interp3
        %[inl] - Array with inline limits
        %[crossl] - Array with crossline limits
        %[timeslice] - Array with time limits

        disp('Ploting a slice');
        %Ploting a slice of the data
        slice(xi,yi,zi,vi,inl,crossl,timeslice);
        shading interp
        xlabel('Inline Distance (m)');
        ylabel('Crossline Distance (m)');
        zlabel('Time (ns)');
        set(gca,'ZDir','reverse');
        set(gca,'YDir','reverse');
        colormap(jet);
        colorbar;
        box on;

hold on;
```

A.4-SAVEG2MAT.m

```
        function saveg2mat(filename,xi,yi,zi,vi)
        %Program to store 3D interpolated data matrix in a MAT file
        %Author: Marco Ceia - UENF/LENEP
        %syntax: saveg2mat(filename,xi,yi,zi,vi)
        %filename - Name of MAT file where the variables will be stored
        %xi,yi,zi must be generated through meshgrid
        %vi is the 3D GPR data generated by interp3
        save(filename,'xi','yi','zi','vi');
```

```
disp(strcat('Matrix stored in: ',filename,'.mat'))
```

A.5-LOADGMAT.m

```
        function loadgmat(filename,xi,yi,zi,vi)
        %Program to load 3D interpolated data matrix stored in a MAT file
        %Author: Marco Ceia - UENF/LENEP
        %syntax: loadgmat(filename,xi,yi,zi,vi)
        %filename - Name of MAT file where the variables will be stored
        %xi,yi,zi must be generated through meshgrid
        %vi is the 3D GPR data generated by interp3
        load(strcat(filename,'.mat'),'xi','yi','zi','vi');
disp(strcat('Matrix loaded from: ',filename,'.mat'))
```

A.6-FAMPRANGE.m

```
        function [fv]=famprange(xi,yi,zi,vi,minamp,maxamp,intx,inty,intz);
        %Amplitude Selection
        %This script set a 3D matrix (cuv) in which values of the inerpolated
        %matrix (vi) outside of the specified range are set to NaN. This way slices
        %will show only the data which values are within the specified range.
        %Author: Marco Ceia (marco@lenep.uenf.br)
        %Getting max & min from vi
        maxvi=max(max(max(vi)));
        minvi=min(min(min(vi)));
        disp(strcat('Max Val of 3D Interp Data :',num2str(maxvi)));
        disp(strcat('Min Val of 3D Interp Data :',num2str(minvi)));
        disp(strcat('Max Val Selected :',num2str(maxamp)));
        disp(strcat('Min Val Selected :',num2str(minamp)));

        [vidim1,vidim2,vidim3]=size(vi);

        for ivi1=1:vidim1
           for ivi2=1:vidim2
              for ivi3=1:vidim3
                 if(vi(ivi1,ivi2,3)>=minamp & vi(ivi1,ivi2,ivi3)<=maxamp)
                    avi(ivi1,ivi2,ivi3)=vi(ivi1,ivi2,ivi3);
                 else
                    avi(ivi1,ivi2,ivi3)=NaN;
                 end
              end
           end

        end
        slice(xi,yi,zi,avi,intx,inty,intz);
        shading flat
        xlabel('X');
        ylabel('Y');
```

```
        zlabel('Z');
        colormap(jet);
        colorbar;
        box on;
axis vis3d tight
```

A.7-fscandt1.m

```
        function [x,t,data2,dt]=fscandt1(name,ntrace,step);
        %Program fscandt1
        %Marco Ceia (marco@lenep.uenf.br)
        %UENF/LENEP
        %-------------------------------------------------
        %Read pulseekko GPR data.
        %function [x,t,data2,dt]=fscandt1(name,ntrace,step)
        %x = position t = time
        %data2 = amplitude values
        %dt = number of samples / timewindow
        %name = Filename ntrace = number of traces
        %step = position increment
        %Input File
        fid=fopen(name,'r');

        %File Reading

        for i=1:ntrace
          i;
          header=fread(fid,32,'single');
          nptptr=header(3); %Number of points/trace
          data=fread(fid,nptptr,'int16');
          trace(:,i)=[header;data];
        end
        clear data;
        st=fclose(fid);
        data2=trace((33:nptptr),:);
        [ldata2,cdata2]=size(data2);
        dt=header(9)/(ldata2-1);
        spc=step;%Spacing
        for i=1:cdata2;x(i)=(i-1)*spc;end;
for j=1:ldata2;t(j)=(j-1)*dt;end;
```

APÊNDICE B – "*GRAPHICAL USER INTERFACE*" (GUI)

B.1-geo3dinit.m

```
function varargout = geo3dinit(varargin)
% GEO3DINIT M-file for geo3dinit.fig
%      GEO3DINIT, by itself, creates a new GEO3DINIT or raises the existing
%      singleton*.
%
%      H = GEO3DINIT returns the handle to a new GEO3DINIT or the handle to
%      the existing singleton*.
%
%      GEO3DINIT('CALLBACK',hObject,eventData,handles,...) calls the local
%          function named CALLBACK in GEO3DINIT.M with the given input arguments.
%
%      GEO3DINIT('Property','Value',...) creates a new GEO3DINIT or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before geo3dinit_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to geo3dinit_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Copyright 2002-2003 The MathWorks, Inc.

% Edit the above text to modify the response to help geo3dinit

% Last Modified by GUIDE v2.5 30-Nov-2007 14:57:48

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @geo3dinit_OpeningFcn, ...
                   'gui_OutputFcn',  @geo3dinit_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
```

```
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before geo3dinit is made visible.
function geo3dinit_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to geo3dinit (see VARARGIN)

% Choose default command line output for geo3dinit
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes geo3dinit wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = geo3dinit_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close
geodt1_init

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close
geoascii_init
```

B.2-geodt1_init.m

```
function varargout = geodt1_init(varargin)
% GEODT1_INIT M-file for geodt1_init.fig
%      GEODT1_INIT, by itself, creates a new GEODT1_INIT or raises the existing
%      singleton*.
%
%      H = GEODT1_INIT returns the handle to a new GEODT1_INIT or the handle
to
%      the existing singleton*.
%
%      GEODT1_INIT('CALLBACK',hObject,eventData,handles,...) calls the local
%          function named CALLBACK in GEODT1_INIT.M with the given input
arguments.
%
%       GEODT1_INIT('Property','Value',...) creates a new GEODT1_INIT or raises
the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before geodt1_init_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to geodt1_init_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Copyright 2002-2003 The MathWorks, Inc.

% Edit the above text to modify the response to help geodt1_init

% Last Modified by GUIDE v2.5 10-Dec-2007 16:20:51

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @geodt1_init_OpeningFcn, ...
                   'gui_OutputFcn',  @geodt1_init_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
```

```
else
   gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before geodt1_init is made visible.
function geodt1_init_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to geodt1_init (see VARARGIN)

% Choose default command line output for geodt1_init
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes geodt1_init wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = geodt1_init_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on selection change in popupmenu2.
function popupmenu2_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns popupmenu2 contents as cell array
%        contents{get(hObject,'Value')} returns selected item from popupmenu2

% --- Executes during object creation, after setting all properties.
function popupmenu2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

```
% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

% --- If Enable == 'on', executes on mouse press in 5 pixel border.
% --- Otherwise, executes on mouse press in 5 pixel border or over text1.
function text1_ButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to text1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --- Executes during object creation, after setting all properties.
function popupmenu4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end


% --- Executes on selection change in Configuration.
function Configuration_Callback(hObject, eventdata, handles)
% hObject    handle to Configuration (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns Configuration contents as cell array
%        contents{get(hObject,'Value')} returns selected item from Configuration

% --- Executes on selection change in popupmenu5.
function popupmenu5_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns popupmenu5 contents as cell array
```

```
%       contents{get(hObject,'Value')} returns selected item from popupmenu5


% --- Executes during object creation, after setting all properties.
function popupmenu5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function filein1_Callback(hObject, eventdata, handles)
% hObject    handle to filein1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of filein1 as text
%        str2double(get(hObject,'String')) returns contents of filein1 as a double


% --- Executes during object creation, after setting all properties.
function filein1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to filein1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a double
```

```
%filein1=get(edit1,'string')
handles.edit1=get(hObject,'String')
%handles.edit1=filein1;
% Update handles structure
guidata(hObject, handles);

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of pushbutton2

%filein1=get(handles.edit1,'String')
[filein1, pathname, filterindex] = uigetfile('*.opt', 'Input Options (*.opt) File');
%filein1=handles.edit1;
handles.filein1=filein1;
loadoptfiledt1
handles.xin=xin;
handles.xcross=xcross;
handles.tin=tin;
handles.dt=dt;
handles.x=x;
handles.y=y;
handles.z=z;
handles.data2=data2;
% Update handles structure
guidata(hObject, handles);


% --- Executes on button press in togglebutton2.
function togglebutton2_Callback(hObject, eventdata, handles)
% hObject    handle to togglebutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of togglebutton2
filein1=handles.filein1;
edit(filein1)
guidata(hObject, handles);

function edity_Callback(hObject, eventdata, handles)
% hObject    handle to edity (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of edity as text
%        str2double(get(hObject,'String')) returns contents of edity as a double
handles.edity=(get(hObject,'String'));
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function edity_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edity (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function editz_Callback(hObject, eventdata, handles)
% hObject    handle to editz (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of editz as text
%        str2double(get(hObject,'String')) returns contents of editz as a double
handles.editz=get(hObject,'String');
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function editz_CreateFcn(hObject, eventdata, handles)
% hObject    handle to editz (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function editx_Callback(hObject, eventdata, handles)
% hObject    handle to editx (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of editx as text
%        str2double(get(hObject,'String')) returns contents of editx as a double

handles.editx=get(hObject,'String');
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function editx_CreateFcn(hObject, eventdata, handles)
% hObject    handle to editx (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

handles.editx=get(hObject,'String');
guidata(hObject, handles);


% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%Grid Steps apply button
xstp=str2num(handles.editx);
ystp=str2num(handles.edity);
zstp=str2num(handles.editz);
   xin=handles.xin;
   xcross=handles.xcross;
   tin=handles.tin;
   dt=handles.dt;
gridsetup
handles.xmin=xmin;
handles.xmax=xmax;
handles.ymin=ymin;
handles.ymax=ymax;
handles.zmin=zmin;
handles.zmax=zmax;
guidata(hObject, handles);
```

```
% --- Executes during object creation, after setting all properties.
function uipanel2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to uipanel2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

function edit7_Callback(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit7 as text
%        str2double(get(hObject,'String')) returns contents of edit7 as a double
%Xint
handles.edit7=get(hObject,'String');
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function edit7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function edit8_Callback(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit8 as text
%        str2double(get(hObject,'String')) returns contents of edit8 as a double
%Yint
handles.edit8=get(hObject,'String');
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function edit8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function edit9_Callback(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit9 as text
%        str2double(get(hObject,'String')) returns contents of edit9 as a double
%Zint
handles.edit9=get(hObject,'String');
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function edit9_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%Show button
disp('Preparing to plot 3D slices')
intx=str2num(handles.edit7);
inty=str2num(handles.edit8);
intz=str2num(handles.edit9);
xi=handles.xi;
```

```
yi=handles.yi;
zi=handles.zi;
vi=handles.vi;
%Consolidate workspace memory
cwd = pwd;
cd(tempdir);
pack
cd(cwd)
%
disp('Ploting a slice');
showslice
handles.intx=intx;
handles.inty=inty;
handles.intz=intz;
guidata(hObject, handles);

% --- Executes on key press over pushbutton4 with no controls selected.
function pushbutton4_KeyPressFcn(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
disp('Interpolate button actived')
x=handles.x;
y=handles.y;
z=handles.z;
dt=handles.dt;
xmin=handles.xmin;
xmax=handles.xmax;
ymin=handles.ymin;
ymax=handles.ymax;
zmin=handles.zmin;
zmax=handles.zmax;
data2=handles.data2;
xstp=str2num(handles.editx);
ystp=str2num(handles.edity);
zstp=str2num(handles.editz);
%Building 3D meshgrid
buildmesh
handles.xi=xi;
handles.yi=yi;
handles.zi=zi;
```

```
%3D Interpolation
interpolate
handles.vi=vi;
guidata(hObject, handles);

% --- Executes on button press in pushbutton7.
function pushbutton7_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%Quit button
clear all
close

% --- Executes on button press in pushbutton8.
function pushbutton8_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%Back button
clear all
close
geo3dinit

% --- Executes on button press in savebut.
function savebut_Callback(hObject, eventdata, handles)
% hObject    handle to savebut (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%Save to MAT button
xi=handles.xi;
yi=handles.yi;
zi=handles.zi;
vi=handles.vi;
[file2save,pathname]=uiputfile('*.mat','Save Interpolated data as a MAT file')
save(file2save,'xi','yi','zi','vi');
disp(strcat('3D Interpolated Data was stored in :',file2save))

guidata(hObject, handles);
% --- Executes on button press in loadmatbut.
function loadmatbut_Callback(hObject, eventdata, handles)
% hObject    handle to loadmatbut (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%Load MAT button
[matfilein1, pathname, filterindex] = uigetfile('*.mat', 'Input MAT File contained interpolated data');
```

```
load(matfilein1,'xi','yi','zi','vi');
handles.xi=xi;
handles.yi=yi;
handles.zi=zi;
handles.vi=vi;
maxxi=max(max(max(xi,[],1)));
maxyi=max(max(max(yi,[],1)));
maxzi=max(max(max(zi,[],1)));
minxi=min(min(min(xi,[],1)));
minyi=min(min(min(yi,[],1)));
minzi=min(min(min(zi,[],1)));

disp(strcat('X :',num2str(minxi),'-',num2str(maxxi)))
disp(strcat('Y :',num2str(minyi),'-',num2str(maxyi)))
disp(strcat('Z :',num2str(minzi),'-',num2str(maxzi)))
guidata(hObject, handles);

% --- Executes on button press in help.
function help_Callback(hObject, eventdata, handles)
% hObject    handle to help (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
helpgeo3d

function minamp_Callback(hObject, eventdata, handles)
% hObject    handle to minamp (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of minamp as text
%        str2double(get(hObject,'String')) returns contents of minamp as a double
minamp=get(hObject,'String');
handles.minamp=str2num(minamp);
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function minamp_CreateFcn(hObject, eventdata, handles)
% hObject    handle to minamp (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
```

```
end

function maxamp_Callback(hObject, eventdata, handles)
% hObject    handle to maxamp (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of maxamp as text
%        str2double(get(hObject,'String')) returns contents of maxamp as a double
maxamp=get(hObject,'String');
handles.maxamp=str2num(maxamp);
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function maxamp_CreateFcn(hObject, eventdata, handles)
% hObject    handle to maxamp (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

% --- Executes on button press in amprangeb.
function amprangeb_Callback(hObject, eventdata, handles)
% hObject    handle to amprangeb (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%Amplitude Selection
%This script set a 3D matrix (cuv) in which values of the inerpolated
%matrix (vi) outside of the specified range are set to NaN. This way slices
%will show only the data which values are within the specified range.
%Author: Marco Ceia (marco@lenep.uenf.br)
%Getting max & min from vi
xi=handles.xi;
yi=handles.yi;
zi=handles.zi;
vi=handles.vi;
minamp=handles.minamp;
maxamp=handles.maxamp;
intx=handles.intx;
inty=handles.inty;
intz=handles.intz;
```

```
maxvi=max(max(max(vi)));
minvi=min(min(min(vi)));
disp(strcat('Max Val of 3D Interp Data :',num2str(maxvi)));
disp(strcat('Min Val of 3D Interp Data :',num2str(minvi)));
disp(strcat('Max Val Selected :',num2str(maxamp)));
disp(strcat('Min Val Selected :',num2str(minamp)));

[vidim1,vidim2,vidim3]=size(vi);%vi dimensions
%Interval selection. Applying NaN to values outside the interval selected
for ivi1=1:vidim1
   for ivi2=1:vidim2
      for ivi3=1:vidim3
         if(vi(ivi1,ivi2,ivi3)>=minamp & vi(ivi1,ivi2,ivi3)<=maxamp)
            avi(ivi1,ivi2,ivi3)=vi(ivi1,ivi2,ivi3);
         else
            avi(ivi1,ivi2,ivi3)=NaN;
         end
      end
   end
end
avimax=max(max(max(avi)));
avimin=min(min(min(avi)));
disp(strcat('Max Val Founded :',num2str(avimax)));
disp(strcat('Min Val Founded :',num2str(avimin)));
if (isnan(avimax) | isnan(avimin))==1
   disp('Choose other limits to interval');
end
disp('Ploting a slice');
figure
slice(xi,yi,zi,avi,intx,inty,intz);
shading interp
xlabel('X');
ylabel('Y');
zlabel('Z');
colormap(jet);
colorbar;
box on;
axis vis3d tight
```

B.3-geoascii_init.m

```
function varargout = geoascii_init(varargin)
% GEOASCII_INIT M-file for geoascii_init.fig
%      GEOASCII_INIT, by itself, creates a new GEOASCII_INIT or raises the existing
%     singleton*.
```

```
%
%      H = GEOASCII_INIT returns the handle to a new GEOASCII_INIT or the handle to
%     the existing singleton*.
%
%     GEOASCII_INIT('CALLBACK',hObject,eventData,handles,...) calls the local
%       function named CALLBACK in GEOASCII_INIT.M with the given input arguments.
%
%       GEOASCII_INIT('Property','Value',...) creates a new GEOASCII_INIT or raises the
%     existing singleton*.  Starting from the left, property value pairs are
%     applied to the GUI before geoascii_init_OpeningFunction gets called.  An
%     unrecognized property name or invalid value makes property application
%     stop.  All inputs are passed to geoascii_init_OpeningFcn via varargin.
%
%     *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%     instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Copyright 2002-2003 The MathWorks, Inc.

% Edit the above text to modify the response to help geoascii_init

% Last Modified by GUIDE v2.5 10-Dec-2007 15:49:00

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @geoascii_init_OpeningFcn, ...
                   'gui_OutputFcn',  @geoascii_init_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before geoascii_init is made visible.
```

```
function geoascii_init_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to geoascii_init (see VARARGIN)

% Choose default command line output for geoascii_init
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes geoascii_init wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = geoascii_init_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on selection change in popupmenu2.
function popupmenu2_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns popupmenu2 contents as cell array
%        contents{get(hObject,'Value')} returns selected item from popupmenu2

% --- Executes during object creation, after setting all properties.
function popupmenu2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
```

```
end

% --- If Enable == 'on', executes on mouse press in 5 pixel border.
% --- Otherwise, executes on mouse press in 5 pixel border or over text1.
function text1_ButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to text1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --- Executes during object creation, after setting all properties.
function popupmenu4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

% --- Executes on selection change in Configuration.
function Configuration_Callback(hObject, eventdata, handles)
% hObject    handle to Configuration (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns Configuration contents as cell array
%        contents{get(hObject,'Value')} returns selected item from Configuration

% --- Executes on selection change in popupmenu5.
function popupmenu5_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns popupmenu5 contents as cell array
%        contents{get(hObject,'Value')} returns selected item from popupmenu5

% --- Executes during object creation, after setting all properties.
function popupmenu5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

```
% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function filein1_Callback(hObject, eventdata, handles)
% hObject    handle to filein1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of filein1 as text
%        str2double(get(hObject,'String')) returns contents of filein1 as a double

% --- Executes during object creation, after setting all properties.
function filein1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to filein1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a double

%filein1=get(edit1,'string')
handles.edit1=get(hObject,'String')
%handles.edit1=filein1;
% Update handles structure
guidata(hObject, handles);

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of pushbutton2

%filein1=get(handles.edit1,'String')
[filein1, pathname, filterindex] = uigetfile('*.opt', 'Input Options (*.opt) File');
%filein1=handles.edit1;
handles.filein1=filein1;
loadoptfile
handles.xin=xin;
handles.xcross=xcross;
handles.tin=tin;
handles.dt=dt;
handles.x=x;
handles.y=y;
handles.z=z;
handles.data2=data2;
% Update handles structure
guidata(hObject, handles);

% --- Executes on button press in togglebutton2.
function togglebutton2_Callback(hObject, eventdata, handles)
% hObject    handle to togglebutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of togglebutton2
filein1=handles.filein1;
edit(filein1)
guidata(hObject, handles);

function edity_Callback(hObject, eventdata, handles)
% hObject    handle to edity (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edity as text
%        str2double(get(hObject,'String')) returns contents of edity as a double
handles.edity=(get(hObject,'String'));
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function edity_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edity (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function editz_Callback(hObject, eventdata, handles)
% hObject    handle to editz (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of editz as text
%        str2double(get(hObject,'String')) returns contents of editz as a double
handles.editz=get(hObject,'String');
guidata(hObject, handles);


% --- Executes during object creation, after setting all properties.
function editz_CreateFcn(hObject, eventdata, handles)
% hObject    handle to editz (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function editx_Callback(hObject, eventdata, handles)
% hObject    handle to editx (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of editx as text
%        str2double(get(hObject,'String')) returns contents of editx as a double

handles.editx=get(hObject,'String');
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function editx_CreateFcn(hObject, eventdata, handles)
```

```
% hObject    handle to editx (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

handles.editx=get(hObject,'String');
guidata(hObject, handles);

% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%Grid Steps apply button
xstp=str2num(handles.editx);
ystp=str2num(handles.edity);
zstp=str2num(handles.editz);
   xin=handles.xin;
   xcross=handles.xcross;
   tin=handles.tin;
   dt=handles.dt;
gridsetup
handles.xmin=xmin;
handles.xmax=xmax;
handles.ymin=ymin;
handles.ymax=ymax;
handles.zmin=zmin;
handles.zmax=zmax;
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function uipanel2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to uipanel2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

function edit7_Callback(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of edit7 as text
%        str2double(get(hObject,'String')) returns contents of edit7 as a double
%Xint
handles.edit7=get(hObject,'String');
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function edit7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function edit8_Callback(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit8 as text
%        str2double(get(hObject,'String')) returns contents of edit8 as a double
%Yint
handles.edit8=get(hObject,'String');
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function edit8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function edit9_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit9 as text
%        str2double(get(hObject,'String')) returns contents of edit9 as a double
%Zint
handles.edit9=get(hObject,'String');
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function edit9_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%Show button
disp('Preparing to plot 3D slices')
intx=str2num(handles.edit7);
inty=str2num(handles.edit8);
intz=str2num(handles.edit9);
xi=handles.xi;
yi=handles.yi;
zi=handles.zi;
vi=handles.vi;
%Consolidate workspace memory
cwd = pwd;
cd(tempdir);
pack
cd(cwd)
%
disp('Ploting a slice');
showslice
handles.intx=intx;
```

```
handles.inty=inty;
handles.intz=intz;
guidata(hObject, handles);

% --- Executes on key press over pushbutton4 with no controls selected.
function pushbutton4_KeyPressFcn(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
disp('Interpolate button actived')
x=handles.x;
y=handles.y;
z=handles.z;
dt=handles.dt;
xmin=handles.xmin;
xmax=handles.xmax;
ymin=handles.ymin;
ymax=handles.ymax;
zmin=handles.zmin;
zmax=handles.zmax;
data2=handles.data2;
xstp=str2num(handles.editx);
ystp=str2num(handles.edity);
zstp=str2num(handles.editz);
%Building 3D meshgrid
buildmesh
handles.xi=xi;
handles.yi=yi;
handles.zi=zi;
%3D Interpolation
interpolate
handles.vi=vi;
guidata(hObject, handles);



% --- Executes on button press in pushbutton7.
function pushbutton7_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
%Quit button
clear all
close

% --- Executes on button press in pushbutton8.
function pushbutton8_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%Back button
clear all
close
geo3dinit

% --- Executes on button press in savebut.
function savebut_Callback(hObject, eventdata, handles)
% hObject    handle to savebut (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%Save to MAT button
xi=handles.xi;
yi=handles.yi;
zi=handles.zi;
vi=handles.vi;
[file2save,pathname]=uiputfile('*.mat','Save Interpolated data as a MAT file')
save(file2save,'xi','yi','zi','vi');
disp(strcat('3D Interpolated Data was stored in :',file2save))
guidata(hObject, handles);
% --- Executes on button press in loadmatbut.
function loadmatbut_Callback(hObject, eventdata, handles)
% hObject    handle to loadmatbut (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%Load MAT button
[matfilein1, pathname, filterindex] = uigetfile('*.mat', 'Input MAT File contained interpolated data');
load(matfilein1,'xi','yi','zi','vi');
handles.xi=xi;
handles.yi=yi;
handles.zi=zi;
handles.vi=vi;
maxxi=max(max(max(xi,[],1)));
maxyi=max(max(max(yi,[],1)));
maxzi=max(max(max(zi,[],1)));
minxi=min(min(min(xi,[],1)));
minyi=min(min(min(yi,[],1)));
minzi=min(min(min(zi,[],1)));
```

```
disp(strcat('X :',num2str(minxi),'-',num2str(maxxi)))
disp(strcat('Y :',num2str(minyi),'-',num2str(maxyi)))
disp(strcat('Z :',num2str(minzi),'-',num2str(maxzi)))
guidata(hObject, handles);

% --- Executes on button press in help.
function help_Callback(hObject, eventdata, handles)
% hObject    handle to help (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
helpgeo3d

function minamp_Callback(hObject, eventdata, handles)
% hObject    handle to minamp (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of minamp as text
%        str2double(get(hObject,'String')) returns contents of minamp as a double
minamp=get(hObject,'String');
handles.minamp=str2num(minamp);
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function minamp_CreateFcn(hObject, eventdata, handles)
% hObject    handle to minamp (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function maxamp_Callback(hObject, eventdata, handles)
% hObject    handle to maxamp (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of maxamp as text
%        str2double(get(hObject,'String')) returns contents of maxamp as a double
maxamp=get(hObject,'String');
handles.maxamp=str2num(maxamp);
```

```
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function maxamp_CreateFcn(hObject, eventdata, handles)
% hObject    handle to maxamp (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end


% --- Executes on button press in amprangeb.
function amprangeb_Callback(hObject, eventdata, handles)
% hObject    handle to amprangeb (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%Amplitude Selection
%This script set a 3D matrix (cuv) in which values of the inerpolated
%matrix (vi) outside of the specified range are set to NaN. This way slices
%will show only the data which values are within the specified range.
%Author: Marco Ceia (marco@lenep.uenf.br)
%Getting max & min from vi
xi=handles.xi;
yi=handles.yi;
zi=handles.zi;
vi=handles.vi;
minamp=handles.minamp;
maxamp=handles.maxamp;
intx=handles.intx;
inty=handles.inty;
intz=handles.intz;

maxvi=max(max(max(vi)));
minvi=min(min(min(vi)));
disp(strcat('Max Val of 3D Interp Data :',num2str(maxvi)));
disp(strcat('Min Val of 3D Interp Data :',num2str(minvi)));
disp(strcat('Max Val Selected :',num2str(maxamp)));
disp(strcat('Min Val Selected :',num2str(minamp)));
[vidim1,vidim2,vidim3]=size(vi);%vi dimensions
%Interval selection. Applying NaN to values outside the interval selected
for ivi1=1:vidim1
```

```
    for ivi2=1:vidim2
        for ivi3=1:vidim3
            if(vi(ivi1,ivi2,3)>=minamp & vi(ivi1,ivi2,ivi3)<=maxamp)
                avi(ivi1,ivi2,ivi3)=vi(ivi1,ivi2,ivi3);
            else
                avi(ivi1,ivi2,ivi3)=NaN;
            end
        end
    end
end
avimax=max(max(max(avi)));
avimin=min(min(min(avi)));
if (isnan(avimax) | isnan(avimin))==1
    disp('Choose other limits to interval');
end
disp(strcat('Max Val Founded :',num2str(avimax)));
disp(strcat('Min Val Founded :',num2str(avimin)));
disp('Ploting a slice');
figure
slice(xi,yi,zi,avi,intx,inty,intz);
shading flat
xlabel('X');
ylabel('Y');
zlabel('Z');
colormap(jet);
colorbar;
box on;
axis vis3d tight
```

B.4-showslice.m

```
%Ploting a slice of the data
figure
box on;

slice(xi,yi,zi,vi,intx,inty,intz);
shading interp
colormap(jet);
colorbar;
xlabel('X');
ylabel('Y');
zlabel('Z');
axis vis3d ;


%hold on;
```

B.5-showslicet.m

```
%Ploting a slice of the data
figure
whos
slice(xi,yi,zi,vit,intx,inty,intz);
shading flat
colormap(jet);
colorbar;
xlabel('X');
ylabel('Y');
zlabel('Z');
axis vis3d ;

%hold on;
```

B.6-buildmesh.m

```
%Building Mesh
disp('Building Mesh');
disp(strcat('xmin :',num2str(xmin),'xstp :',num2str(xstp),'xmax :',num2str(xmax)));
disp(strcat('ymin :',num2str(ymin),'ystp :',num2str(ystp),'ymax :',num2str(ymax)));
disp(strcat('zmin :',num2str(zmin),'zstp :',num2str(zstp),'zmax :',num2str(zmax)));
[xi,yi,zi]=meshgrid(xmin:xstp:xmax,ymin:ystp:ymax,zmin:zstp:zmax);
disp('Mesh Created');
```

B.7-gridsetup.m

```
%Grid Setup
disp('-----------------------------------------');
%grat=1;%Grid Ratio
%disp(['Grid Ratio = 1/',num2str(grat)]);
xmax=max(xin);xmin=min(xin);
%xstp=stepin/grat;
ymax=max(xcross);ymin=min(xcross);
%ystp=stepcross/grat;
zmax=max(tin);zmin=min(tin);
zstp1=dt*zstp;
disp(['Grid Cell Size X = ',num2str(xstp)]);
disp(['Grid Cell Size Y = ',num2str(ystp)]);
disp(['Grid Cell Size Z x dt = ',num2str(zstp1)]);
disp([' dt = ',num2str(dt)]);
disp('-----------------------------------------');
```

B.8-interpolate.m

```
%Making interpolated volume (vi)
```

```
        disp('3D Interpolation');
vi=interp3(x,y,z,data2,xi,yi,zi);
```

B.9-helpgeo3d.m

```
        function varargout = helpgeo3d(varargin)
        % HELPGEO3D M-file for helpgeo3d.fig
        %      HELPGEO3D, by itself, creates a new HELPGEO3D or raises the existing
        %      singleton*.
        %
        %      H = HELPGEO3D returns the handle to a new HELPGEO3D or the handle to
        %      the existing singleton*.
        %
        %      HELPGEO3D('CALLBACK',hObject,eventData,handles,...) calls the local
        %          function named CALLBACK in HELPGEO3D.M with the given input
arguments.
        %
        %      HELPGEO3D('Property','Value',...) creates a new HELPGEO3D or raises the
        %      existing singleton*.  Starting from the left, property value pairs are
        %      applied to the GUI before helpgeo3d_OpeningFunction gets called.  An
        %      unrecognized property name or invalid value makes property application
        %      stop.  All inputs are passed to helpgeo3d_OpeningFcn via varargin.
        %
        %      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
        %      instance to run (singleton)".
        %
        % See also: GUIDE, GUIDATA, GUIHANDLES

        % Copyright 2002-2003 The MathWorks, Inc.

        % Edit the above text to modify the response to helpgeo3d helpgeo3d

        % Last Modified by GUIDE v2.5 25-Mar-2002 12:19:15

        % Begin initialization code - DO NOT EDIT
        gui_Singleton = 1;
        gui_State = struct('gui_Name',       mfilename, ...
                           'gui_Singleton',  gui_Singleton, ...
                           'gui_OpeningFcn', @helpgeo3d_OpeningFcn, ...
                           'gui_OutputFcn',  @helpgeo3d_OutputFcn, ...
                           'gui_LayoutFcn',  [] , ...
                           'gui_Callback',   []);
        if nargin && ischar(varargin{1})
           gui_State.gui_Callback = str2func(varargin{1});
        end

        if nargout
```

```
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before helpgeo3d is made visible.
function helpgeo3d_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to helpgeo3d (see VARARGIN)

% Choose default command line output for helpgeo3d
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes helpgeo3d wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = helpgeo3d_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
```

B.10-loadoptfile.m

```
%Input Parameters
a=textread(filein1,'%s');
filein2=char(a(1));%GPR (*.dat) filename
ntrace=str2num(char(a(2)));%Number of traces
stepin=str2num(char(a(3))); %(Step size (offset) - Inline)
stepcross=str2num(char(a(4))); %(Step size (offset) - Crossline)
nfiles=str2num(char(a(5)));%Number of GPR files
%
if nfiles>1
   %Loading Data filenames
   a=textread(filein2,'%s');
   for i=1:nfiles
```

```
            a(i)
            gprfile(i,:)=char(a(i))
          end
          temp1=load(gprfile(1,:));
          data=temp1(:,:);
          [ldata,cdata]=size(data);
          clear data;
          clear a;
          dt=1;
          %Crossline Coordinates
          %xcross = crossline x-coord
          for i=1:nfiles;xcross(i)=(i-1)*stepcross;end;
          %
          %Loading data
          for l=1:nfiles
            disp(['Reading File #',num2str(l),' :',gprfile(l,:)]);
            %[xin,tin,data,dt]=fscandt1(gprfile(l,:),ntrace,stepin);
           temp1=load(gprfile(l,:));
          %xin=[temp1(:,1)];tin=[temp1(:,2)];data=[temp1(:,3)];
         xin=(1:cdata)';tin=(1:ldata)';data=[temp1(:,:)];
            for j=1:ldata
              for i=1:cdata
                data2(l,i,j)=data(j,i);
                x(l,i,j)=xin(i);
                y(l,i,j)=xcross(l);
                z(l,i,j)=tin(j);
              end
            end
            clear data;
          end
        end
        %disp(filein1)
disp('OPT File Loaded')
```

B.11-loadoptfiledt1.m

```
        %Input Parameters
        a=textread(filein1,'%s');
        filein2=char(a(1));%GPR (*.dt1) filename
        ntrace=str2num(char(a(2)));%Number of traces
        stepin=str2num(char(a(3))); %(Step size (offset) - Inline)
        stepcross=str2num(char(a(4))); %(Step size (offset) - Crossline)
        nfiles=str2num(char(a(5)));%Number of GPR files
        %
        if nfiles>1
          %Loading GPR filenames
          a=textread(filein2,'%s');
```

```
            for i=1:nfiles
               gprfile(i,:)=char(a(i));
            end
            %Loading first GPR file for test
            %xin = inline x-coord
            %tin = inline time
            [xin,tin,data,dt]=fscandt1(gprfile(1,:),ntrace,stepin);
            [ldata,cdata]=size(data);
            clear data;
            clear a;
            %
            %Crossline Coordinates
            %xcross = crossline x-coord
            for i=1:nfiles;xcross(i)=(i-1)*stepcross;end;
            %
            %Loading data
            for l=1:nfiles
               disp(['Reading File #',num2str(l),' :',gprfile(l,:)]);
               [xin,tin,data,dt]=fscandt1(gprfile(l,:),ntrace,stepin);
               for j=1:ldata
                  for i=1:cdata
                     data2(l,i,j)=data(j,i);
                     x(l,i,j)=xin(i);
                     y(l,i,j)=xcross(l);
                     z(l,i,j)=tin(j);
                  end
               end
               clear data;
            end
         end
         %disp(filein1)
disp(strcat('OPT File Loaded : ',filein1))
```