MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

INPE-9558-NTC/351

INTRODUÇÃO À ENTRADA DE DADOS NO OPENDIX: FORMATOS ".DX" , ".GENERAL" E ".GRB"

Roberto Blaz
Margarete Oliveira Domingues
Odim Mendes Júnior

INPE
São José dos Campos
2003

SUMÁRIO

LISTA DE FIGURAS

Pág.

CAPÍTULO 1

Introdução

O OpenDX é um programa de computador gratuito semelhante ao Data Explorer da IBM (maiores informações podem ser obtidas na página http://www.opendx.org). Ele consiste de um pacote de ferramentas para manipular, processar, transformar, visualizar e animar dados em diversas plataformas computacionais e pode ser utilizado tambem para visualização de dados no paradigma da computação de processamento paralelo.

Esse programa de visualização possui uma forma padrão de entrada de dados, designado pela extensão **".dx"** (lê-se ponto dx). Com isso, em muitos casos, como na Meteorologia, na Oceanografia e nas Ciências Espaciais, torna–se necessário utilizar da ferramenta gráfica de importação de dados existentes no OPENDX, conhecida como DATA PROMPTER, para preparar e fazer a entrada dos dados no programa. Contudo, para tirar todo o proveito da estrutura de dados que o OPENDX permite, é necessário entender de uma forma mais detalhada os conceitos de campos, de grupos e de conectividade, afim de criar os formatos nativos do OPENDX e, dessa forma, dispensar a entrada interativa DATA PROMPTER.

O objetivo deste trabalho é contribuir essencialmente na disseminação da entrada de dados no formato **".dx"** e **".general"** do OPENDX, difundindo e facilitando a utilização desse programa multiplataforma pelos usuários interessados. Em particular, todos os exemplos e testes apresentados neste relatório foram realizados nos ambientes operacionais GNU/LINUX Red Hat 7.2 e Unix Solaris 7.

Buscou-se estruturar o texto de uma forma prática. No Capítulo 2, apresenta-se uma visão geral do OPENDX. No Capítulo 3, introduz-se o DATA PROMPTER, que é uma interface gráfica de importação de dados. No Capítulo 4, explica-se o formato de dados **".general"** . No Capítulo 5, explica-se o formato de dados **".dx"** . No Capítulo 6, por meio de um caso específico para servir de exemplo, demonstra-se o potencial de uso do programa OPENDX na visualização de dados de modelos meteorológicos de previsão de tempo. No capítulo 7, fazem-se algumas considerações gerais finais.

CAPÍTULO 2

Uma visão geral do OpenDX

OPENDX é uma ferramenta de visualização muito poderosa pela forma de lidar com dados e permitir suas visualizações. Tem como finalidades principais a visualização de informações formatadas em 2D e 3D, de forma interativa ou automática (por meio de *scripts* ou pela inclusão no código fonte de funções em linguagem C ou Fortran), e a criação de filmes de visualização temporal de uma base de dados em formato *mpeg*(<www.mpeg.org>). Uma das grandes vantagens desse recurso é sem dúvida a facilidade de programação em ambiente *visual*, conhecido pela sua sigla em inglês VPE (que é um ambiente de programação visual, "Visual Program Evinronment"). Essa ferramenta tem uma enorme gama de aplicações, desde as áreas de ciências em geral, por exemplo a meteorologia, até a indústria. Na Figura 2.1 são apresentados alguns exemplos de visualizações de dados no OPENDX.

O primeiro passo para a utilização do OPENDX é fazer com que o OPENDX reconheça os dados a serem visualizados. O uso do OPENDX de forma interativa é de certa forma fácil, pois sua interface é auto explicativa. Para ativar a janela inicial, basta digitar *dx* em um terminal *xterm*, em modo gráfico. A janela inicial é composta por oito botões que indicam: *Import Data (Importação de dados), Run Visual Programs (Executar programas visuais), Edit Visual Program (Editar programa visual), New Visual Program (Criar novo programa visual), Run Tutorial (Executar tutorial), Samples (Exemplos), Help (Ajuda)* e *Quit* (Sair) ver Figura 2.2

O primeiro botão é o *Import Data*, que possibilita importar dados no OPENDX. Quando ele é acionado, surge uma nova janela denominada de DATA PROMPTER, que permite descrever o arquivo de dados que se deseja importar (ver Figura 2.3). Maiores detalhes do DATA PROMPTER são apresentados no Capítulo 3.

O segundo botão é o *Run Visual Programs*, que serve para executar os programas visuais já elaborados para o OPENDX, os **".net"** . Ao acionar esse botão, é solicitado o nome do arquivo que contém o programa. A seguir o OPENDX inicia a execução do programa. As etapas são mostradas na Figura 2.4.

O terceiro botão é o *Edit Visual Programs*, que introduz uma nova janela para editar (VPE), como apresentada na Figura 2.5. Nessa nova janela também existe a possibilidade de execução do programa, só que neste caso, há uma maior necessidade

de memória para manter as telas gráficas do VPE e a execução do programa. Na Figura 2.6 está sendo apresentado um exemplo de um programa visual e os resultados gráficos obtidos após sua execução.

O quarto botão é o *New Visual Program*, que possibilita a criação de um novo programa visual no VPE.

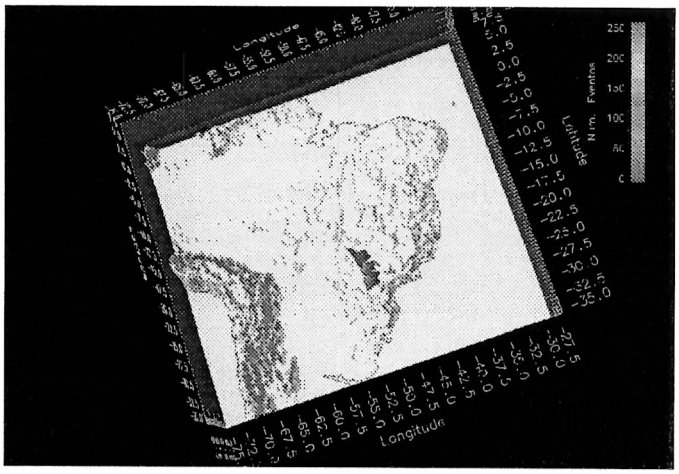
O quinto botão é o *Run Tutorial* que mostra um breve tutorial sobre o OPENDX. Na Figura 2.8 estão apresentados alguns exemplos desse tutorial.

O sexto botão é o *Samples* que dá acesso aos exemplos **".net"** do OPENDX, usualmente guardados no diretório */usr/local/dx/samples/programs/*, que acompanham o OPENDX (*dxsamples-4.1.3.tar.gz*), conforme apresentado na Figura 2.9

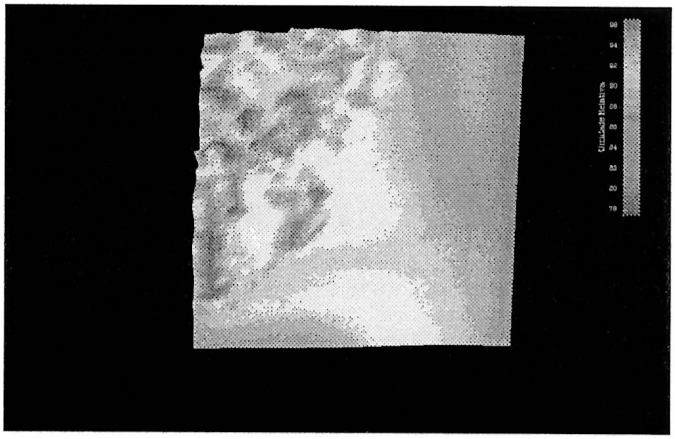
O botão *Help* aciona a descrição de cada um dos botões da janela de iniciação do OPENDX e explica como acioná-los diretamente no *xterm*. Na Figura 2.10 estão apresentados alguns exemplos dessa janela de ajuda.

O botão *Quit* finaliza o OPENDX.

(a)



(b)



(c)

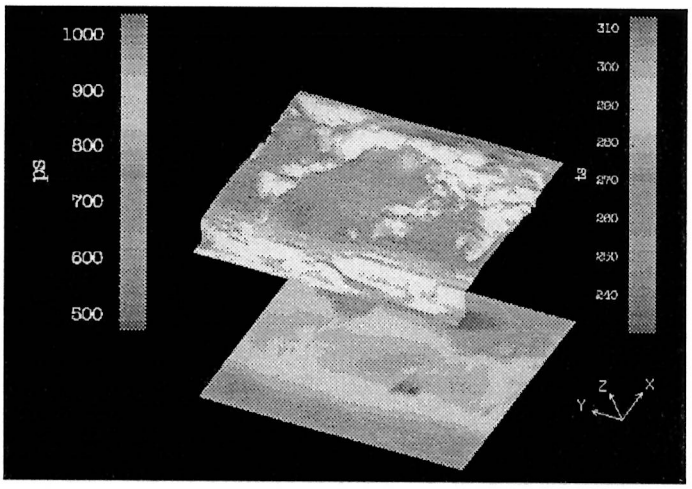


Fig. 2.1 – Exemplos de visualizações de dados no OpenDX: (a) Densidade de Descargas elétricas atmosféricas; (b) Umidade relativa à superfície; (c) Pressão e Temperatura da superfície

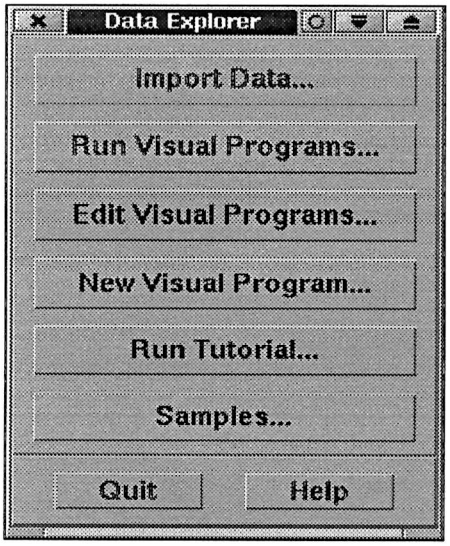


Fig. 2.2 – Janela inicial do OpenDX

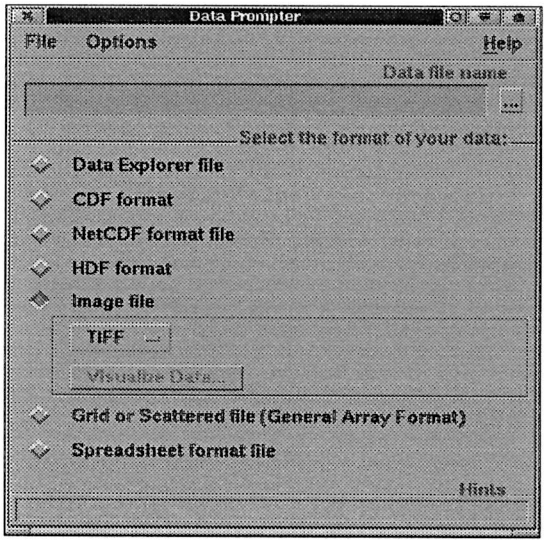


Fig. 2.3 – Data Prompter

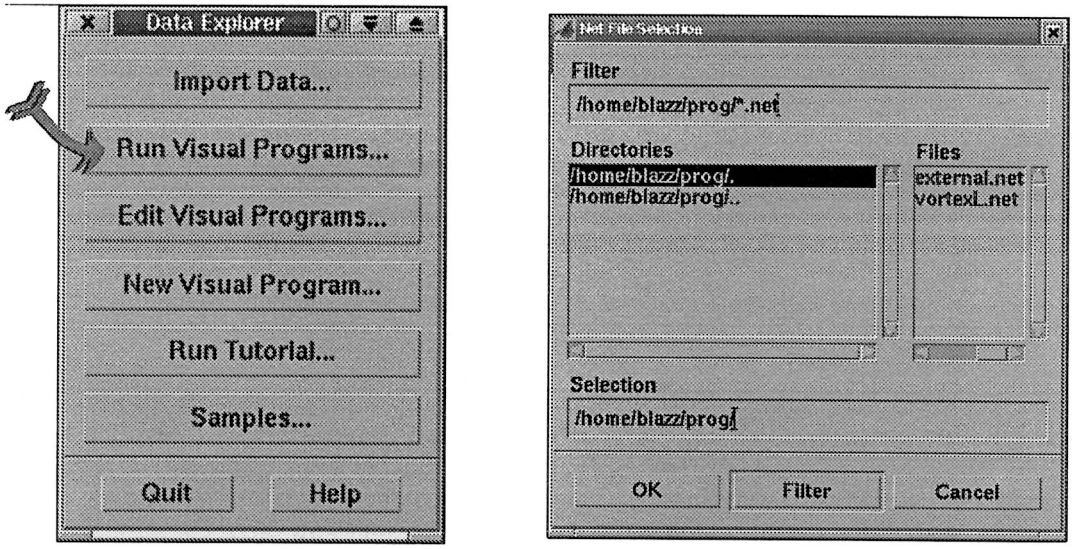


Fig. 2.4 – Execução de programas no OPENDX.

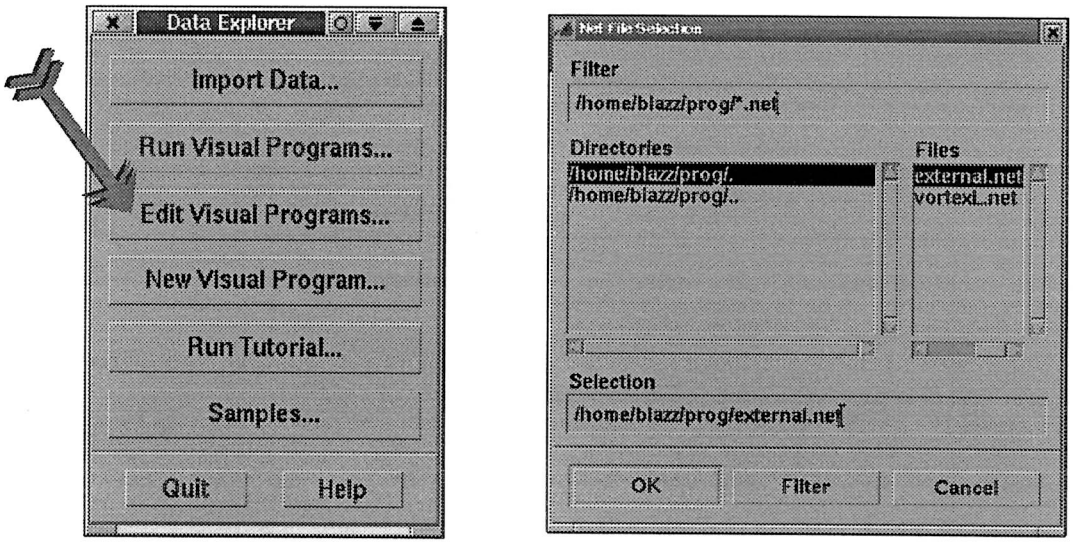
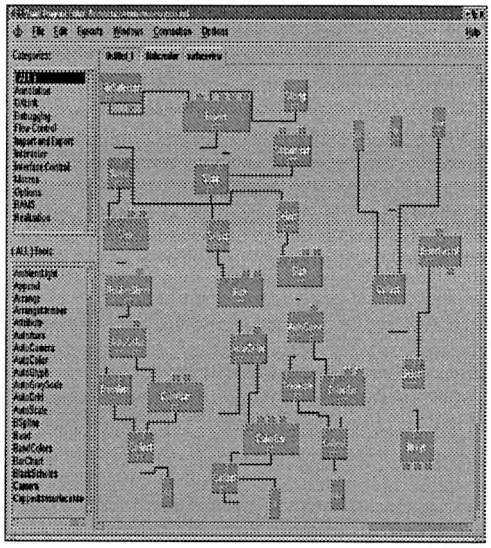


Fig. 2.5 – Ambiente de edição de programas visuais no OPENDX.

(a)VPE



(b)Visualização dos dados

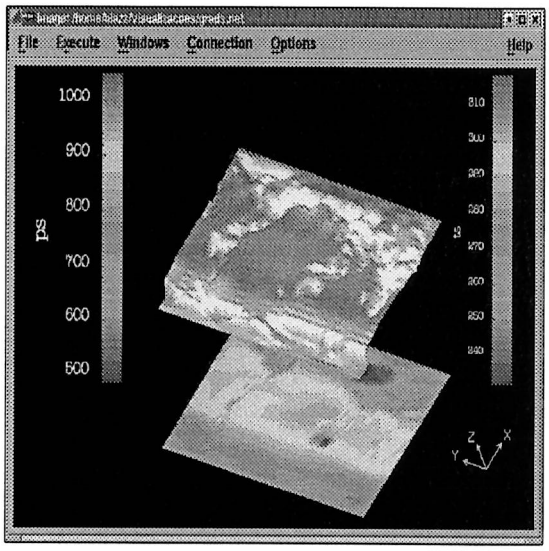


Fig. 2.6 – Programa visual no OPENDX. Adaptação de um arquivo grads (grib)

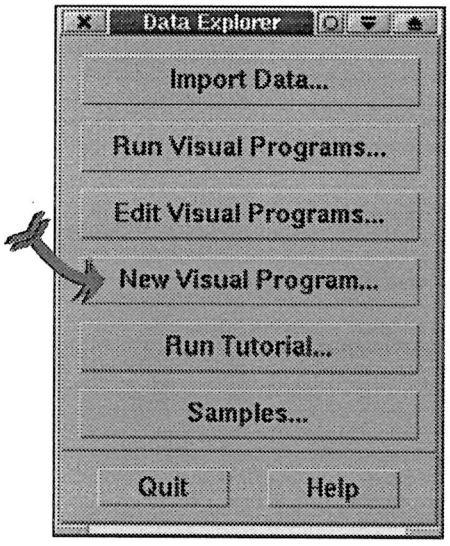
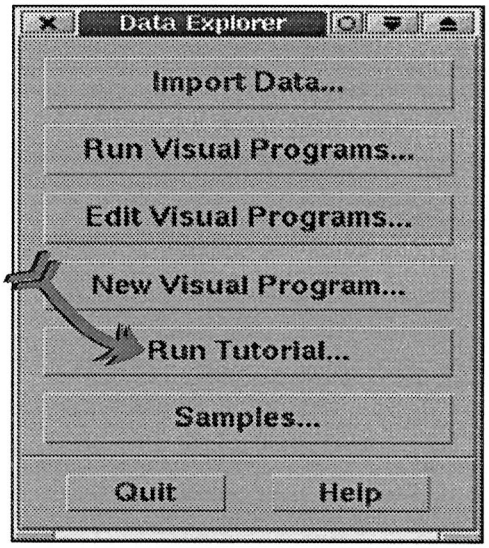


Fig. 2.7 – Exemplos de um Novo Programa Visual

(a)OpenDX



(b)Tutorial

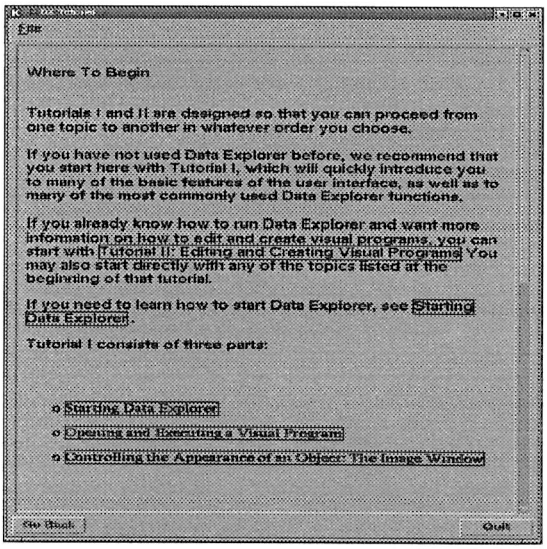
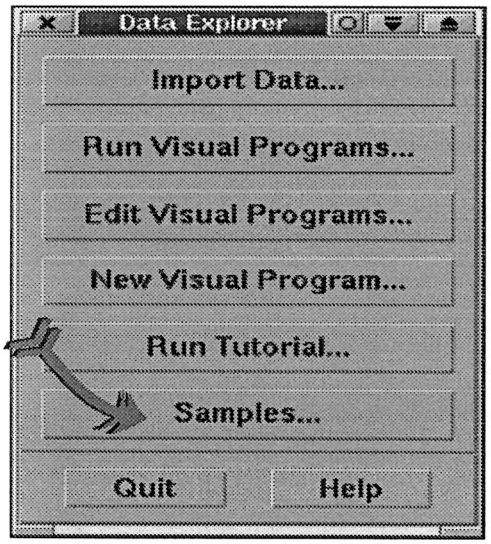


Fig. 2.8 – Tutorial do OpenDX.



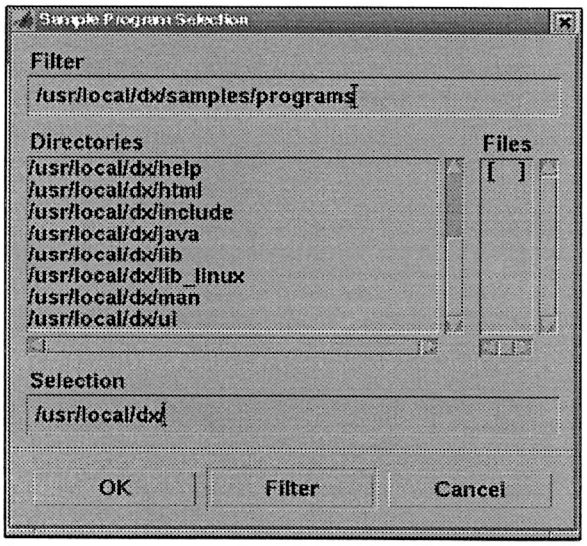


Fig. 2.9 – Auxílio por meio de exemplos no OpenDX.

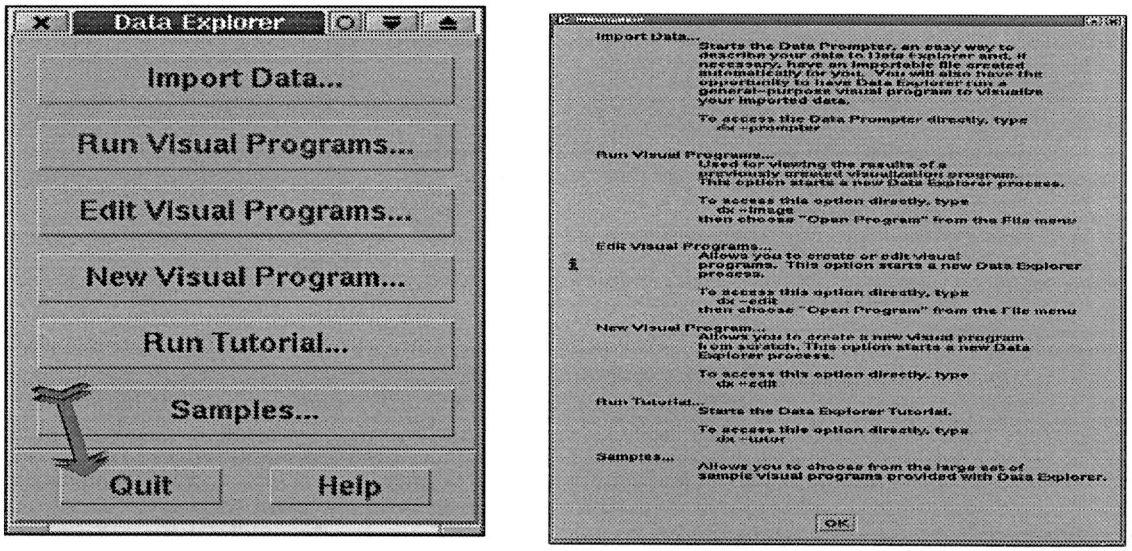


Fig. 2.10 – Ajuda interativa no OpenDX.

CAPÍTULO 3

"Data Prompter": *Interface Gráfica de Importação de Dados*

Os dados nativos do OPENDX são os **".dx"** , que serão discutidos no Capítulo 5. Uma das maneiras de se entrar com dados no OPENDX é importar esses dados de um arquivo de dados pré-existente nos formatos CDF, netCDF, HDF e em um formato geral de dados matriciais (**".general"**).

O OPENDX tem disponível uma interface gráfica, DATA PROMPTER , que importa os dados do usuário para o OPENDX . Este tipo de entrada de dados é feito por meio das seguintes etapas:

1) Acionar o DATA PROMPTER — No terminal *xterm*, em ambiente gráfico, digitar dx e escolher o menu **Import Data**, ou diretamente no terminal *xterm*, também em ambiente gráfico, digitar **dx -prompter.**(Figura 3.1)

2) Na janela do DATA PROMPTER — Existe dois botões: (a) **File**, que permite a seleção do arquivo de dados; e (b) **Options**, que permite abrir uma janela de mensagem de dados. Abaixo desses botões existe uma janela para a entrada do nome do seu arquivo de dados, à direita desse campo existe um botão com o título "..." para visualização do diretório padrão de busca de dados. O DATA PROMPTER permite importar e visualizar os seguintes tipos de arquivo (ver Figura 3.1):

   - Arquivos no formato **".dx"** ;
   - Arquivos no formato CDF;
   - Arquivos no formato netCDF;
   - Arquivos no formato HDF;
   - Arquivos de imagem no formato TIFF, MIFF, GIF, RGB, R+G+B, YUV;
   - Arquivos com dados em grade regularmente espaçadas (Grid) ou em grade irregularmente espaçadas (*scattered*)
   - Planilha de dados.

Os arquivos com dados em grade regularmente espaçadas ou em grade irregularmente espaçadas e os em planilha de dados precisam ser descritos detalhadamente no

DATA PROMPTER. Quando se trabalha com dados em grade regularmente ou irregularmente espaçadas, algumas informações complementares devem ser passadas ao Data Prompter para que seus dados sejam importados corretamente (ver Figura 3.3). Uma das primeiras informações que deve ser passada é qual o tipo de grade que compõe seus dados. O OPENDX tem como opções os seguintes tipos de grade:

Grade Regular— Todo o espaçamento é regular independentemente da direção. Neste caso há necessidade de informar se os dados estão em blocos ou colunas, se eles contém ou não um passo de tempo, e de quantas variáveis eles são compostos.

Parcialmente Regular—Há a necessidade de informar se os dados estão em blocos ou colunas, se eles contém ou não um passo de tempo, e de quantas variáveis eles são compostos.

Grade regular deformada—Em que determinados pontos o espaçamento é regular. Nesse tipo de grade, além das opções já descritas, as dimensões dos dados devem ser usadas .

Grade dispersa de dados—um conjunto de dados irregularmente espaçados.

Para que os dados possam ser importados corretamente, mais algumas informações devem ser passadas ao DATA PROMPTER. Acionando o botão *Describe Data*, uma nova janela será aberta para que os dados sejam detalhados (Figura 3.4). Nessa nova janela as seguintes opções podem ser encontradas:

1) *Data file* (arquivo de dados): Janela de definição de arquivo .

2) *Header* (cabeçalho): Essa opção deve ser usada quando houver um cabeçalho para seus dados. Ativando essa opção, o *menu pop-up* ao lado poderá ser utilizado para definir-se onde termina o cabeçalho e começam os dados. Existem três opções:

   - *Of Bytes* (em bytes): Quantos bytes devem ser desprezados até o início dos dados;
   - *Of Lines* (em linhas): Quantas linhas devem ser desprezadas até o início dos dados;

- *String Marker* (Conjunto de caracteres): Qual palavra determina o início dos dados.

3) *Gride Size* (Tamanho da Grade) e ♯ *of points* (Número de Pontos) :

   a) *Gride Size* é usado quando os dados tem conexões em malha, ao ser acionado o campo de texto ao lado fica ativo para entrada da quantidade de dimensões dos dados e os campos que não forem usados podem permanecer em branco.

   b) ♯ *of points* é usado quando os dados são compostos por pontos separados. O campo de texto ativa-se ao entrar com o número de pontos.

   Essas opções não podem ser usadas simultaneamente.

4) *Data Format* (formato dos dados): O formato de dados pode ser de dois tipos:

   - *American Standart Code for Information Interchange(ASCII)*: Tipo texto;
   - *Binary (Binário/IEEE)*: Quando aciona-se essa opção, um novo *menu pop-up* é ativado com outras duas opções:
     a) Primeiro byte significativo está à direita *(Most Significant Byte First)*
     b) Primeiro byte significativo está à esquerda *(Least Significant Byte First).*

5) *Data order (Orden dos Dados)*: Especifica-se a forma como os dados irão variar, podendo variar em colunas ou em linhas.

6) *Field Interleaving* (Forma de Distribuição dos Dados): Nessa opção é informado a maneira como os dados estão organizados, podendo ser em blocos ou colunas. Se os dados estão organizados em blocos, ao se acionar essa opção um novo *menu pop-up* pode ser acessado ao lado para se informar como devem ser interpretados os dados do bloco, em vetores $(x_0y_0, x_1y_1, \cdots, x_ny_n)$ ou como componentes $(x_0, x_1, \cdots, x_n, y_0, y_1, \cdots, y_n)$.

7) *Series* (Distribuição em Séries): Utiliza-se essa opção quando os dados são compostos de uma seqüência de campos. Quando isso acontece, três caixas de texto *(n, start e delta)* são ativadas para se informar o número de elementos no arquivo de dados *(n)* , o número do primeiro membro da série *(start)* e a variação de posições entre os elementos*(delta).*

8) *Series interleaving* (Distribuição das Séries): Ao se ativar a janela *Series*, a janela *Series interleaving* também é ativada para que se possa definir séries compostas por mais de um campo $(F_1, F_2, \cdots)$ e qual a organização desses campos:

   - *Series members together* : $(F_1^{s0}, F_2^{s0}, F_1^{s1}, F_2^{s1}, \cdots, F_1^{sn}, F_2^{sn})$
   - *Fields together*: $(F_1^{s0}, F_1^{s1}, F_2^{s0}, F_2^{s1}, \cdots, F_n^{s0}, F_n^{s1})$

9) *Series separator* (Separador das Séries): Apenas é usado se os dados que compõem os membros da série são separados por: bytes, linhas ou caracteres delimeadores.

10) *Grid positions* (Posicionamento na Grade): Usado apenas se as posições de malha não estão especificadas no arquivo. Nesta categoria existem três opções:

   - *Completely regular* (Completamente regular): As dimensões da grade são todas regulares. Esta opção fixa em regular o *menu pop up* ao lado da caixa de texto que contém as origens e os deltas das grades em regular.
   - *Partially regular* (Pacialmente regular): As dimensões das grades são parcialmente regulares. Esta opção habilita o *menu pop up* ao lado das caixas de texto que contém as origens e os deltas das grades para que se possa escolher entre regular e irregular.
   - *Explicit position list* (Lista de posições explícitas): As posições estão completamente irregulares, então deve-se especificar essas posições na caixa de texto ao lado do *menu pop up*.

11) Ao lado direito do Prompter existem as opções de campos mais usados que podem ser modificados ou criados pelo usuário, as opções são as seguintes:

   - *Field list* (Lista de Campo) : Lista de campos que define os campos mais usados com suas respectivas atribuições. Para alterar entre um campo e outro basta usar o botão *Move field*. A definição padrão é *f*ield0.
   - *Field name* (Nome do campo): Aparece o nome do campo em uso.
   - *Type* (Tipo): Onde deve-se especificar os tipos de dados dos campos; *float, byte, unsigned byte, signed byte, short, unsigned short, signed short, int, unsigned int, signed int, double* e *string*. Esses tipos são os mesmos utilizados na linguagem *C*. O livro de Kernighan e Ritchie (1990) é uma referência básica dessa linguagem.

- *Structure* (Estrutura): Seleciona a estrutura dos dado que compõem os campos: *scalar, 1-vetor, 2-vetor, 3-vetor, 4-vetor, 5-vetor, 6-vetor, 7-vetor, 8-vetor* e *9-vetor.*
- *Dependency* (Dependência): Determina como os dados estão relacionados entre si. A dependência entre os dados pode ser do tipo:
  - *positions* (posição)
  - *connections* (conexão)
- *Layout* (Esquema): Esta opção ativa-se somente quando as opções *Field interleaving, columnar,* e o *Data format, ASCII,* estão selecionados. Esta opção mostra a forma como estão localizados os elementos dos dados em uma linha de texto. Existem duas caixas de texto ao lado desse item. Devem ser informados quantos caracteres devem ser pulados antes que comece a ser lido os dados dos campos *(skip)* e também a largura do campo que se deve ler *(width).*
- *Block* (Blocos): Esta opção é ativada quando *Field interleaving (block)* e *Data format (ASCII)* estão ativos e explica a localização de um item de dados em uma linha ou campo de texto. Existem três caixas de texto ao lado dessa opção onde se deve informar quantos caracteres devem ser pulados antes que se comece a leitura dos dados do campo *(skip)*, especificar quantos elementos devem ser lidos após o pulo *(♯elem)*, e especificar a largura do componente de dados *(width).*

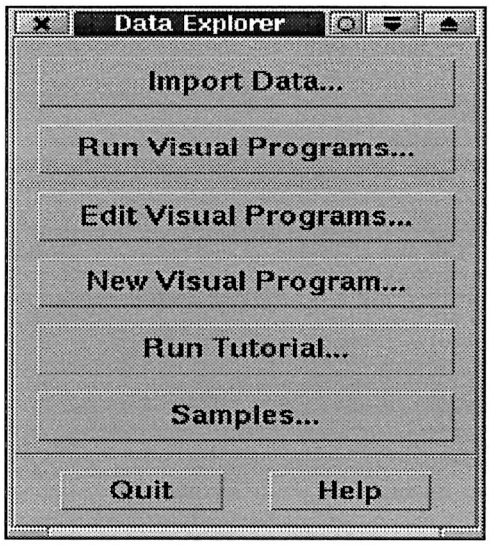
12) Existem ainda quatro botões que servem para adicionar um campo a lista *(add)*, modificar o campo *(modify)*, apagar o campo da lista *(delete)* e inserir um novo campo *(insert).*

13) *Record separator* (Separador de Registros): Este botão torna-se acessível quando *Field interleaving (block)* está ativo e serve para especificar um separador entre blocos ou registros no arquivo de dados.

Se o arquivo está no formato de planilha (*Spreadsheet Format File*), ou seja, colunas de dados relacionadas entre si, são necessárias informações se os dados estão organizados em tabela ou em matrizes e qual caracter define a separação das colunas (Figura 3.5). Organização de tabela significa que os dados serão tratados como um campo único de dados em que cada coluna será um componente específico de dados. Já na organização em matriz, assume–se que essa matriz é bidimensional com linhas

e colunas especificando as duas dimensões e que todos os dados nas colunas são do mesmo tipo.

(a)OpenDX



(b)Data Prompter

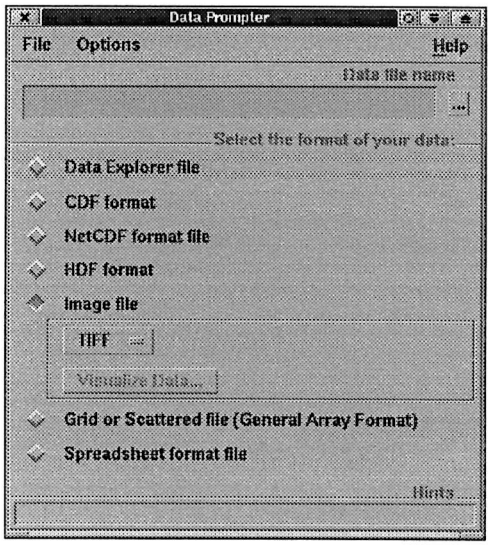


Fig. 3.1 – Janela gráfica inicial do a) OpenDX , e b) Data Prompter .

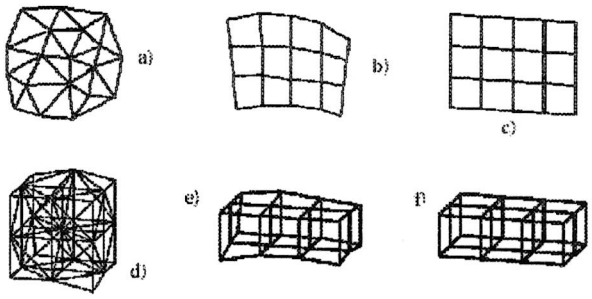


Fig. 3.2 – Tipos de Grade: a) Tetraedro 2D, b) Deformada 2D, c) Regular 2D, d) Tetraedro 3D, e) Deformada 3D e Regular 3D

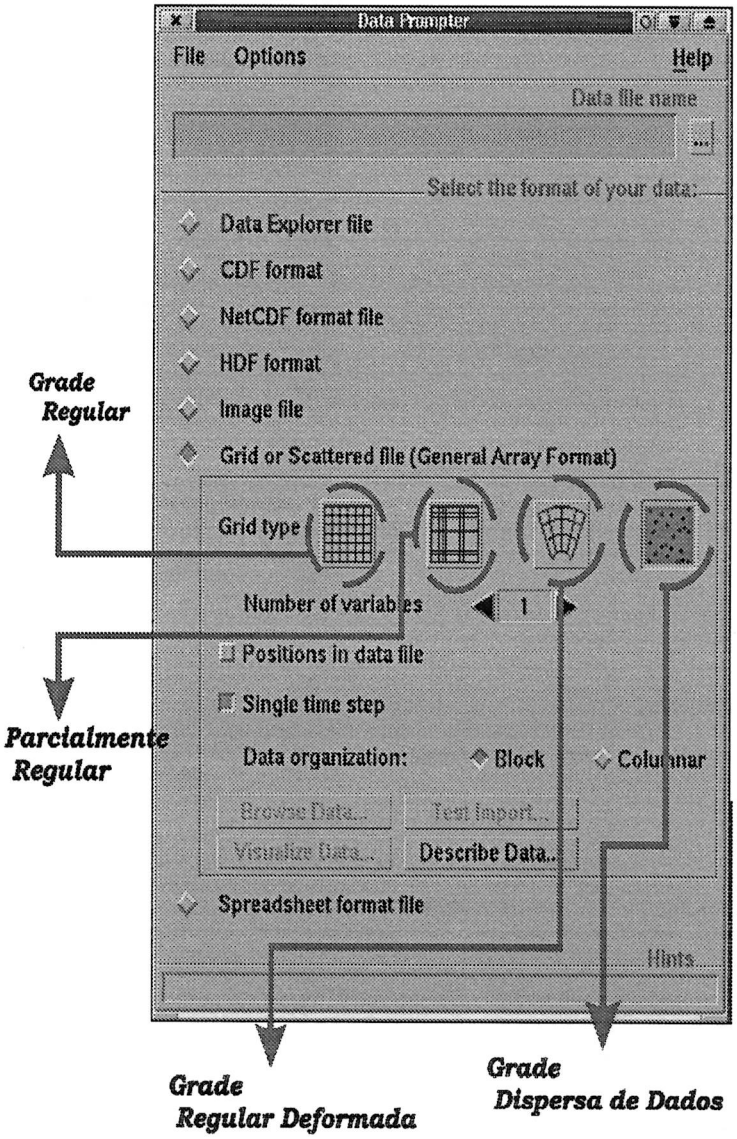


Fig. 3.3 – Janela gráfica para inclusão de dados em grades regulares e irregulares (*Grid or Scatterred File - General Array Format*).

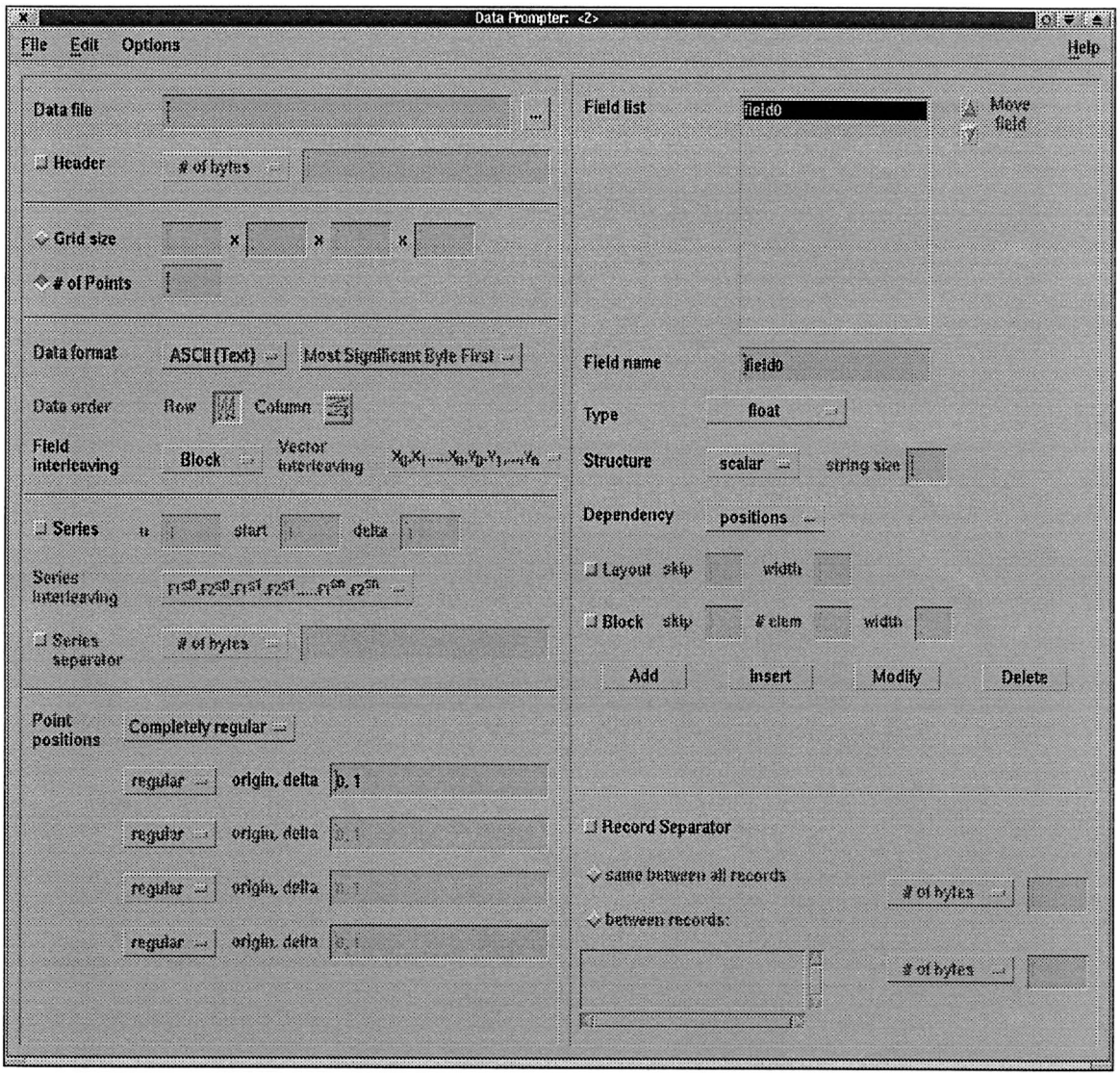


Fig. 3.4 – Janela gráfica para descrição de dados em grades regulares e irregulares (*Grid or Scatterred File - general array format*).

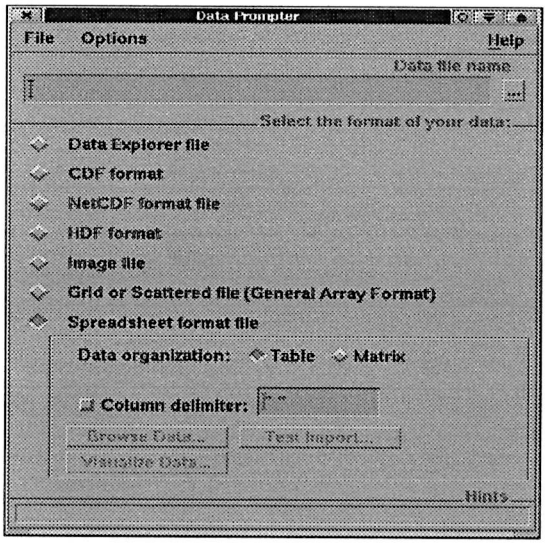


Fig. 3.5 – Janela gráfica para inclusão de dados no formato de planilha (*Spreadsheet Format File*).

CAPÍTULO 4

Formato de dados ".general"

O formato **".general"** é um formato mais indicado para usuários iniciantes.

Para preparar dados nesse formato, deve-se ter em mente algumas questões:

a) Quais as variáveis dependentes e independentes? No OPENDX, as variáveis independentes constituem as posições que compõem um campo de dados. As variáveis dependentes podem ser descritas em forma de uma grade regular ou uma lista.

b) Qual o formato dos dados: são ASCII ou binários? São ponto flutuante, inteiros, etc?

c) Os dados são dependentes das posições ou das conexões entre essas posições? (Ver Figura 4.1).

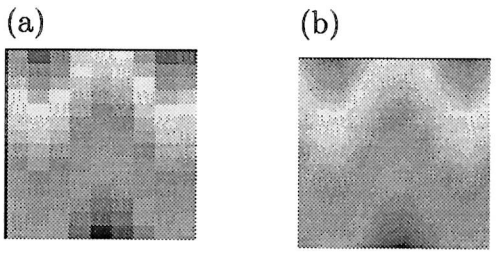


Fig. 4.1 – Dependência de dados a)Por Conexão e b)Por Posição.

d) Eles representam dados em série ou uma moldura única de dados?

e) Os dados são registros ou uma planilha? Se os dados constituem uma grade, qual a ordem dos dados na grade? Eles variam em colunas ou em linhas? Se os dados estão em formato ASCII ou binários, estes dados podem estar organizados em colunas verticais ou em blocos?

Como responder a essas questões é o assunto tratado neste Capítulo. As informações apresentadas são baseadas nos manuais da IBM (1997b,a) e de Thompsom et al. (2001).

4.1 Entendendo os arquivos ".general"

Os arquivos **".general"** utilizam um arquivo de cabeçalho para descrever a estrutura e localização dos dados a serem importados. Este cabeçalho consiste de declarações de palavras chaves que identificam características importantes desses dados, incluindo estruturas da grade, formato e tipo de dados. Este arquivo pode ser criado com um editor de textos ou com o prompter de dados que checa possíveis erros de sintaxe. Os dados importados para o OPENDX podem estar em ASCII ou binário, que são organizados em um dos dois estilos que o **".general"** suporta, que são blocos ou colunas. Dados em blocos requerem organização em registros, enquanto dados em colunas requerem organização em colunas verticais. Um esquema representativo dessas formas de armazenamento é apresentado na Figura 4.2.

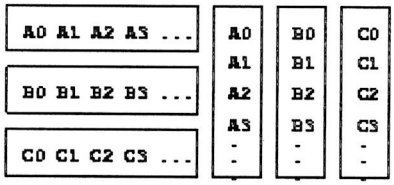


Fig. 4.2 – Exemplo de organização de dados em blocos e colunas

A seguir são apresentados exemplos de como formatar dados em blocos e em colunas, na linguagem C. Nesses exemplos, A, B e C são vetores de dados com precisão simples (*float*) com 100 elementos cada um.

a) Armazenagem em blocos na Linguagem C:

```
                                                    ⋮

int i ;
  for(i=0; i<100; i++)  printf("%10.3f",A[i]);
  for(i=0; i<100; i++)  printf("%10.3f",B[i]);
  for(i=0;i<100, i++)   printf("%10.3f",C[i]);

                                                    ⋮
```

b) Armazenagem em colunas na Linguagem C:

```
                                                    ⋮
```

```
int i ;
for (i=0; i<100; i++) printf("  %10.3f  %10.3f  %10.3f\n",A[i],B[i],C[i]);
```

⋮

Nos dois estilos, tanto blocos quanto colunas, as informações podem ser:

- Dados escalares ou vetoriais;
- Uma série temporal;
- Dados matriciais *Gridded ou Scattered.* Em dados no formato *Gridded* a estrutura de grade pode estar regular ou *warped*, mas os elementos da conexão devem ser regulares; i.e, linhas, quadrados ou cubos; Os dados *Scattered* são dados irregularmente espaçados.
- Dados dependentes da posição (associados as posições da grade) ou dependente de conexão (associada com as conexões da grade).

Os dados matriciais

0	49	0	1	0
3	59	9	9	9
1	97	21	3	0
2	78	9	9	70
45	9	0	8	4

representam uma grade regular de dados 2D. O Código 4.1.1 representa o arquivo **".general"** que contém a descrição desses dados. Com esse arquivo **".general"** pode–se importar os dados para OPENDX .

```
file = /home/prog/matriz2D.dat
grid = 5 x 5
format = ascii
interleaving = record
majority = row
field = field0
structure = scalar
type = float
dependency = positions
positions = regular, regular, 0, 1, 0, 1
end
```

Código 4.1.1 – Exemplo de um arquivo **".general"** criado a partir de uma matriz de dados 2D.

4.2 Sintaxe dos arquivos ".general" .

Os arquivos **".general"** possuem suas próprias palavras-chave. A seguir são apresentadas as principais palavras-chave utilizadas e sua estrutura de organização:

a) *Grid*: Especifica o tamanho e dimensão da grade que contém os dados.

b) *Format*: Especifica o formato e a ordem de bytes dos dados.

c) *Interleaving*: Informa ao DATA PROMPTER como é o intervalo de dados.

d) *Majority*: Especifica a organização dos dados multidimensionais que compõem o campo de dados.

e) *Field*: Especifica o nome e número de campos individuais em arquivo de dados.

f) *Structure*: Especifica a estrutura de cada campo em um arquivo de dados.

g) *Type*: Especifica o tipo de dado para cada campo.

h) *Dependency*: Especifica a dependência dos dados do campos importados.

i) *Positions*: Define as posições que compõem um campo em um arquivo de dados.

Estrutura das palavras-chave nos arquivos **".general"**

file = filename

grid = num_x *times* $num_y \times num_z \times \cdots$

or

points = n

[**block** = $skip_1, elements_1, width_1, skip_2, elements_2, width_2, \cdots, skip_f, elements_f, width_f$]

[**dependency** = $dependency_1, dependency_2, \cdots, dependency_f$]

[**field** = $name_1, name_2, \cdots, name_f$]

$$\left[\begin{array}{l} \textbf{format} = \left[\begin{array}{c} msb \\ lsb \end{array}\right] \end{array}\begin{array}{c} ascii \\ text \\ binary \\ ieee \end{array}\right]$$

$$\left[\begin{array}{l} \textbf{header} = \end{array}\begin{array}{c} bytes\ \text{n} \\ lines\ \text{n} \\ marker\ \text{"string"} \end{array}\right]$$

$$\left[\begin{array}{l} \textbf{interleaving} = \end{array}\begin{array}{c} field \\ record \\ record-vector \\ series-vector \end{array}\right]$$

[**layout** = $skip_1, width_1, skip_2, width_2, \cdots, skip_f, width_n$]

$$\left[\begin{array}{l} \textbf{majority} = \end{array}\begin{array}{c} row \\ column \end{array}\right]$$

$$\left[\begin{array}{l} \textbf{recordseparator} = \end{array}\begin{array}{cc} \textbf{bytes n} & \textbf{bytes n} \\ \textbf{lines n} & \textbf{lines n} \\ \textbf{marker}''string'' & \textbf{marker}''string'' \end{array}\ ,\cdots\right]$$

$$\left[\textbf{series} = [\text{n}, start, delta] \quad \left[, \textbf{separator} = \begin{array}{c} bytes\ \text{n} \\ lines\ \text{n} \\ marker\ \text{"string"} \end{array} \right] \right]$$

$[\textbf{structure} = structure_1, structure_2, \cdots, structure_f]$

$[\textbf{type} = type_1, type_2, \cdots, type_f]$

$$\left[\textbf{positions} = \begin{array}{l} origin_1, delta_1, \cdots, origin_d, delta_d \\ positiontype_1, positiontype_2, \cdots, positiontype_d, positon_1, position_2, \cdots, position_k \\ position_1, position_2, \cdots, position_g \end{array} \right]$$

[*end*]

4.3 Exemplos de arquivos ".general"

4.3.1 Dados escalares em uma grade regular

Dados escalares em uma grade regular também podem ser 3D. O arquivo de dados record_scalar tem um exemplo desse tipo de dados e pode ser encontrado no pacote de exemplos do OpenDX. Esses dados ilustram um campo escalar de ponto flutuante em uma grade regular importado de um arquivo de texto com extensão **".general"** como apresentado no código 4.3.1. Neste caso a origem da grade é [1 3 2], com variação 0.5, 1, e 0.75 no x, y e z respectivamente. Para gerar esse arquivo pode-se utilizar o Data Prompter. No campo de seleção de arquivos, coloca-se o nome do arquivo com as opções *Regular Grid*, *Single time step* e *Number of variables* (neste caso o número de variáveis é 1). Devem também ser acionadas o ítem *Data organization*, com a opção *Block*. No campo *Grid size* preenche-se os campos com os números 5, 8 e 6 respectivamente e o *Data format* recebe o texto ASCII. A ordem dos dados *Data order* é linha *Row* com *Grid positions*: Inserir, então a origem e a variação *origin, delta* que são 1.0, 0.5 para a primeira caixa de texto, 3.0, 1.0 para a segunda e 2.0, 0.75 para última. Após isso, salva-se o arquivo, selecionando a opção *File* e *Save as*. Este arquivo está apresentado no Código 4.3.1.

```
file = record_scalar
grid = 5 x 8 x 6
format = text
interleaving = record
majority = row
field = field0
structure = scalar
type = float
dependency = positions
positions = regular, regular, regular, 1.0, 0.5, 3.0, 1.0, 2.0, .75
end
```

Código 4.3.1 – Exemplo de um arquivo no formato **".general"** 3D

4.3.2 Dados Centralizados em Células

No exemplo a seguir, usa-se o arquivo criado (**".general"**) do exemplo anterior como base para criar um novo. Uma diferença importante é que aqui os dados do arquivo necessitam de célula centralizada (dependente de conexão), ao invés de termos um para cada posição, têm-se um para cada conexão. Inicia-se o OPENDX, seleciona-se *Import Data*, logo após *Grid or Scattered*, para então acionar o botão *Describe Data*. Na barra no topo da janela seleciona-se *File*, depois *Open* e então seleciona-se o arquivo **".general"** criado no exemplo anterior. Após isso, novamente na barra no topo da janela, seleciona-se *Options* e a seguir *Full prompter*. No campo *Data file* muda-se o nome do arquivo para record_depconnections e no campo *Dependency* muda-se a opção de *positions* para *connections*, confirmar a mudança com o botão *Modify*. Após isso, repete-se os passos para salvar um novo arquivo (**".general"**). Este arquivo está apresentado no Código 4.3.2.

4.3.3 Exclusão de Comentários nos Dados

Um arquivo pode conter comentários que podem não fazer parte de seus dados. Esses comentários não devem ser importados. Para importar somente os dados, deve-se usar a uma opção para que o OPENDX pule as informações que não necessita ao ler esse arquivo. Esse tipo de ação pode ser feito pulando-se *bytes*, linhas ou uma coleção de caracteres. Neste novo exemplo, os primeiros passos são iguais aos anteriores e o

```
file = record_depconnections
grid = 5 x 8 x 6
format = text
interleaving = record
majority = row
field = field0
structure = scalar
type = float
dependency = connections
positions = regular, regular, regular, 1, .5, 3, 1.0, 2.0, .75
end
```

Código 4.3.2 – Arquivo **".general"** de dados dependentes de conexão

nome do arquivo agora é record_withheader. O botão *Header* deve ser selecionado no *menu pop up* com a opção *of lines* e a caixa de texto deve conter o número de linhas a serem ignoradas antes do início da leitura dos dados. Agora repete-se os passos para salvar esse arquivo (**".general"**) que está aprensentado no Código 4.3.3.

```
file = record_withheader
grid = 5 x 8 x 6
format = text
interleaving = record
majority = row
header = lines 3
field = field0
structure = scalar
type = float
dependency = positions
positions = regular, regular, regular, 1, .5, 3, 1.0, 2.0, .75
end
```

Código 4.3.3 – Exemplo de um arquivo **".general"** contendo comentários

4.3.4 Modificação do Campo

Este exemplo ilustra a forma de se modificar o campo que consta em *FIELD*. Quando há mais de um tipo de campo, pode-se acessar este campo por meio do botão *modify field* ao lado da caixa *FIELD* no DATA PROMPTER. Nesta mesma caixa é possivel adicionar ou modificar um campo, sendo que ela já possui exemplos dos campo mais utilizados pelo usuário.

Como exemplo, abre-se o arquivo record_scalar, como nos exemplos anteriores, na caixa *field* com o nome $field_n$ para temperatura e confirma-se a alteração pressionando o botão *modify*. Repete-se o passo para salvar o arquivo **".general"** , representado no Código 4.3.4.

```
file = record_scalar
grid = 5 x 8 x 6
format = text
interleaving = record
majority = row
field = Temperature
structure = scalar
type = float
dependency = positions
positions = regular, regular, regular, 1, .5, 3, 1.0, 2.0, .75
end
```

Código 4.3.4 – **".general"** de um campo específico

4.3.5 Importação de dados *record* e *record-vector*

Este exemplo demonstra a forma de se importar dados com estilos *record* (os dados para cada componente do vetor são armazenados em blocos individuais) e *record-vector* (os componentes da cada vetor são armazenados consecutivamente):

a) Dados *record* – No DATA PROMPTER , selecione *File e New* na barra no topo da janela, selecionando o arquivo record_vector1, então coloca-se os valores 5 e 4 nas primeiras duas caixas de texto de *Gride Size*, na caixa *Field list* selecione o campo $field_n$, modifique *Type* para *int*, em *Structure* selecione 2-vector e pressione o botão *modify* para confirmar a modificação. Em *Vector interleaving* selecione $(x_0, x_1, \cdots, x_n, y_0, y_1, \cdots, y_n)$. Salve o arquivo **".general"** como os exemplos anteriores, o Código 4.3.5 apresenta esse arquivo.

b) Dados *record-vector* – Os passos para abrir um arquivo novo não se alteram, somente o nome do arquivo é trocado para record_vectordata2. Os passos iniciais são iguais ao anterior até a definição do *vector interleaving* em que se deve usar a opção $(x_0y_0, x_1y_1, \cdots, x_n, y_n)$. Repete–se os passos para salvar o arquivo **".general"** . O Código 4.3.6 apresenta esse arquivo.

```
file = record_vectordata1
grid = 5 x 4
format = text
interleaving = record
majority = row
field = field0
structure = 2-vector
type = int
dependency = positions
positions = regular, regular, 0, 1, 0, 1
end
```

Código 4.3.5 – Exemplo de **".general"** com dados no estilo textitrecord

```
file = record_vectordata2
grid = 5 x 4
format = text
interleaving = record-vector
majority = row
field = field0
structure = 2-vector
type = int
dependency = positions
positions = regular, regular, 0, 1, 0, 1
end
```

Código 4.3.6 – Exemplo de **".general"** com dados no estilo *record-vector*

Nos próximos exemplos tratam-se de dados de estilo *record* de *interleaving*, para dados com variáveis multiplas.

4.3.6 Grade regular deformada

Uma grade regular deformada consiste em as posições serem irregulares e as conexões regulares. No exemplo a seguir a grade é 5×4, os dados consistem de três registros, os dois primeiros contém dados do *scalar* que definem a grade. O terceiro contém valores (2-vector) que definem as posições da grade. Para importar–se esses dados no DATA PROMPTER, escolhe-se a opção *Grid or Scattered file* e seleciona-se o tipo de grade deformada, que é o terceiro botão de *Gride type*, atribui-se o número 2 a *Number of variables* e a *dimension*. Em *Data organization* confirma–se a opção *block*, após isso repete–se os passos para abrir um arquivo record_deformed. Em *Gride size*

atribui–se os valores 5 e 4. Em *Data format*, *Data order* e *Vextor interleaving* atribuir *text*, *record-vector* e $(x_0, x_1, \cdots, x_n, y_0, y_1, \cdots, y_n)$, respectivamente. Salva-se, então, o arquivo como nos exemplos anteriores. Esse arquivo é descrito no Código 4.3.7.

```
file = deformed
grid = 5 x 4
format = text
interleaving = record-vector
majority = row
field = Precipitacao, Temperatura, locations
structure = scalar, scalar, 2-vector
type = float, int, float
dependency = positions, positions, positions
end
```

Código 4.3.7 – Exemplo de um **".general"** com um dados no estilo *record-interleaving*

Existe também a possibilidade de pular informações em um bloco de dados. Para isso, no DATA PROMPTER , selecione *Gride or Scattered file*, abra o arquivo block_example como nos exemplos anteriores, ative *Header* e mude o menu de *of bytes* para *of lines*, em *Gride size* coloque os valores 5x5. Ative no *full prompter* o botão *block* e coloque o valor 17 em *skip*, 5 em *elem* e 3 para *width*. Salve o arquivo como os exemplos anteriores, o arquivo **".general"** resultante é apresentado no Código 4.3.8.

4.3.7 Uma série temporal

Este exemplo ilustra uma grade 5×5 em que os dados são dependentes de conexão e o estilo é *record* e contêm uma série temporal com sete passos. Para importar esse tipo de dado no DATA PROMPTER seleciona-se *Grid or Scattered file*, em *Grid type* seleciona-se grade regular e desativa-se *Single time step*. Ativa-se o botão *block*, pressiona-se *Describe Data* para que os dados possam ser detalhados. A seguir são apresentados os dados.

```
# Time-Series Data
Time Step 1
12  9 14  1 10 16  7 20
19  6 11 15 18  8 13 17
Time Step 2
```

```
file = /usr/dx/samples/data/block_example.data
grid = 5 x 5
format = text
interleaving = record
majority = row
header = lines 1
field = field0
structure = scalar
type = int
dependency = positions
block = 17, 5, 3
positions = regular, regular, 0, 1, 0, 1
end
```

Código 4.3.8 – Exemplo de um **".general"** desprezando informações em um certo bloco

```
12  9  1 21 10 16  7  1
19  6 11 15 18  8 13 17
⋮

Time Step 7
12  9 14  1 10 16  7 20
19  6 11 15 18  8 13 17
```

Ativa-se o botão *Header* e altera-se o menu de *of bytes* para *String marker*. Na caixa de texto ao lado desta opção, digita-se *Time Step* para que o OPENDX saiba que são esses caracteres que ele deve ignorar antes de começar a ler os dados. Com isso uma linha a cada duas linhas é ignorada. Entra-se, então, com o valor 5 nas duas primeiras caixas de texto de *Gride Size*, ativa-se o botão *Series* com o valor 7, em *n*. Deixa-se *start* e *delta* sem alteração, ativa-se *Series separator* selecionando-se *of lines* no *menu pop up*. Digita-se o valor 1 na caixa de texto ao lado desta opção, muda-se *Dependency* para *connections*, repete–se os passos para salvar um arquivo. O arquivo gerado é apresentado no Código 4.3.9:

4.3.8 Dados escalares múltiplos

Sejam os dados escalar múltiplos os dados desejados, com as seguintes caracteríticas: uma matriz $4 \times 2 \times 3$ com *origin* [0 0 0] e *delta* [1 1 1] com estilo *Interleaving* e *record*. Devem-se criar novos campos para o *field list* com os dados que compõem seus campos, por exemplo umidade relativa, pressão e temperatura. Esses dados geram

```
file = record_series
grid = 5 x 5
format = text
interleaving = record
majority = row
header = marker ''Time Step''
series = 7, 1, 1, separator = lines 1
field = field0
structure = scalar
type = float
dependency = connections
positions = regular, regular, 0, 1, 0, 1
end
```

Código 4.3.9 – Arquivo **".general"** com dado com uma série temporal

um arquivo **".general"** como o apresentado no Código 4.3.10.

```
Umidade
    80          85          100         99          99          99
    95          91          82          95          91          98
Pressao
    1019        1021        1018        1018        1019        1024
    1020        1023        1019        1023        1018        1023
Temperatura
    20          26          27          20          28          23
    25          23          20          20          22          20
```

```
file = myrecord_multiscalar
grid = 4 x 2 x 3
format = text
interleaving = record
majority = row
header = lines 1
field = Umidade, Pressao, Temperatura
recordseparator = lines 1
end
```

Código 4.3.10 – Arquivo **".general"** com dado escalares múltipos

Se os dados são dependentes das conexões, o **".general"** apresentará o

Código 4.3.11.

```
file = myrecord_multiscalar
grid = 4 x 2 x 3
format = text
interleaving = record
majority = row
header = lines 1
field = Umidade, Pressao, Temperatura
recordseparator = lines 1
end
```

Código 4.3.11 – Arquivo **".general"** com dados escalares múltiplos dependentes de conexão

Para descrever o campo de velocidade do vento 2D decomposto nas componentes zonal e meridional, usa–se o *field list*, *Structure* opção *vector-2*, que significa que as componentes x são listados primeiro e seguidas pelas componentes y.

Um exemplo é a grade 4×3 apresentada a seguir:

```
Umidade
   80        85        100       99        99        99
   95        91        82        95        91        98
 Velocidade do Vento
   0         12        16        20        17        10
   17        13        8         1         7         4

   9         10        17        11        2         3
  18         3         18        5         10        0
Temperatura
   20        26        27        20        28        23
   25        23        20        20        22        20
```

O arquivo **".general"** para descrever esses dados está apresentado no Código 4.3.12.

4.3.9 Dados irregularmente espaçados

Neste caso deve-se optar por *Grid or Scattered file* no DATA PROMPTER, em *Gride type* escolher *Scattered data*, e em *Number of variables* selecionar o número 2, por exemplo. Aciona-se o botão *Positions in data file* e seleciona-se o número 2 novamente para *Dimension*. Repetem-se os passos de abertura de um arquivo dos exemplos anteriores selecionando-se o arquivo *record_deformed*, entrando com os valores 5 e 4 para *Gride size*, confirme para *Data Format*, *Data order* e *Vector interleaving*, respectivamente *text*, *Row* e $(x_0y_0, x_1y_1, \cdots, x_ny_n)$. Salvar o arquivo **".general"** descrito no Código 4.3.13.

```
file = myrecord_scalarvector1
grid = 4 x 3
format = text
interleaving = record
majority = row
header = lines 1
field = Umidade, Velocidade, Temperatura
structure = scalar, 2-vector, scalar
recordseparator = lines 1, lines 0, lines 1
end
```

Código 4.3.12 – Arquivo **".general"** com dados com um campo com vetor bidimencional

```
file = record_deformed
points = 20
format = text
interleaving = record-vector
field = precipitacao, temperature, locations
structure = scalar, scalar, 2-vector
type = float, int, float
dependency = positions, positions, positions
end
```

Código 4.3.13 – Arquivo **".general"** com dado em uma grade irregularmente espaçada

CAPÍTULO 5

Formato de dados ".dx"

O formato **".dx"** de arquivo é flexível, podendo representar qualquer objeto. Para se importar este arquivo por meio do DATA PROMPTER, deve-se apenas especificar que o arquivo é um arquivo nativo do OPENDX. Ele também pode ser importado diretamente no VPE pela ferramenta *File Select→Import*. Como gerar os dados neste formato é o assunto tratado neste Capítulo. As informações apresentadas são baseadas nos manuais da IBM (1997b,a) e de Thompsom et al. (2001).

5.1 Entendendo os arquivos ".dx"

Os arquivos **".dx"** podem conter uma seção de cabeçalho, uma seção de dados ou ambas. A seção de cabeçalho consiste de uma seqüência de definições de objeto, cada definição de objeto consiste de uma seqüência de cláusulas que começam pela palavra chave *object*. A seção de dados é um conjunto de itens de dados que são especificados em formato texto ou binário pelo objeto *data* nos vários objetos de arranjo, texto e binário podem ser misturados na mesma seção de dados. As palavras chaves que compõem as cláusulas dos arquivos **".dx"** são apresentados a seguir.

***object*:** Todo objeto tem uma classe e uma identificação numérica ou nome do arquivo que contém esse objeto, o objeto é introduzido por meio de uma palavra chave *object* que especifica a quantidade de objetos ou sua classe (*class*). Um número ou o nome do objeto é usado para referir-se a esse objeto.

Um objeto pode ser importado por meio de seu nome, simplesmente atribuindo esse nome a palavra chave *variable*. Todos os objetos podem ter qualquer número de atributos que serão especificados pela palavra chave *attribute*. O valor de um atributo é um objeto. O valor pode ser especificado por um nome ou uma lista de nomes usando-se a palavra chave *string* ou por um número usando a palavra chave *number* ou ainda por uma referência de objeto.

***series object*:** É uma subclasse de *object* em que cada membro tem além de seu índice uma posição de série em ponto flutuante *(float)*.

***multigrid objects*:** É uma subclasse de *object* em que cada membro tem o

mesmo número de dados e tipos de conexões. Este recurso é usado para representar um campo como primitivo. Os campos podem ser espacialmente disjuntos ou podem se sobrepor. Esses objetos podem conter um número especifíco de membros que são definidos pela palavra chave *member*. Os objetos podem ser declarados como um objeto ou como um número dentro de um arquivo. Os números dos membros devem ser seqüenciais começando pelo número 0 sem nenhum intervalo na numeração.

***composite field objects*:** É uma outra subclasse de *objects*, nesta classe cada membro tem o mesmo número de dados e tipo de conexões. Os campos podem ser espacialmente disjuntos, com os limites das posições exatamente iguais. Esses objetos podem conter um número específico de membros que são descritos pela palavra chave *member*, os objetos podem ser especificados como um objeto ou como um número dentro de um arquivo; o números de membro devem ser seqüênciais começando pelo número 0 sem nenhum intervalo na numeração.

***field objects*:** Os objetos de campo têm seus números de componentes especificados por meio da palavra chave *componentes*, o valor das componentes são especificados como um objeto específico ou um número do arquivo corrente ou ainda um número em outro arquivo.

***constant arrays objects*:** Define que todos os elementos do arranjo têm o mesmo valor.

***gridpositions*:** Utiliza-se para representar n grades dimensionais de pontos geometricamente regulares. A forma da grade (número de pontos em cada dimensão) é especificada por uma lista de números seguido da palavra chave *counts*; o número de elementos nesta lista determina a dimensão da grade. A origem da grade pode ser especificada pela palavra–chave *origin* — que lista o número da origem da coordenada. Quando *origin* não é especificado por definição ela é 0. A palavra–chave *origin* pode ser seguida por n vezes a palavra *delta*, uma para cada uma das dimensões definidas. Os últimos valores associados a palavra *delta* especificarão como é a variação da dimensão.

***product array object*:** É usado quando o arranjo é uma seqüência linear de pontos no espaço, que começa numa origem qualquer (especificada pela

palavra chave *origin*) e que por sua vez é separada pela palavra *delta*. A dimensão do espaço é determinada pelo número especificado para *origin*.

***gridconnections*:** Esta palavra–chave é utilizada para representar n conexões de uma grade regular cúbica, o n corresponde a dimensão da grade. Se a grade está separada por um campo composto, a palavra–chave *meshoffsets* também deve ser utilizada para definir como é o posicionamento desse campo em relação a grade.

***path array objects*:** Esta opção codifica a regularidade linear das conexões. Sua utilização e similar ao do *gridconnections*.

end clause: Indica o fim da seção de cabeçalho.

5.2 Criando arquivos ".dx"

O programa externalfilter.c, em linguagem C, que acompanha o OPENDX , converte um arquivo de dados regularmente espaçado contendo uma matriz de dados 3D num arquivo **".dx"** , que pode ser diretamente lido no OPENDX . Nesse trabalho os autores introduziram algumas modificações na forma de entrada de dados e na documentação desse programa e o renomearam com MyExternalFilter.c. Este programa é apresentado no Código 5.2.1. O programa MyExternalFilter.c é compilado pelo GNU/gcc v.2.95, com o comando

gcc MyExternalFilter.c -o MyExternalFilter

e para executá-lo utiliza-se o seguinte comando:

MyExternalFilter <Arquivo de Leitura> <Arquivo de saída ".dx" > .

No Código 5.2.1 é necessário informar a quantidade de elementos N_x, N_y, N_z em cada uma da direções x, y, z, a origem dos dados x_0, y_0, z_0 e a variação dos valores desses elementos na matriz 3D $(\Delta_x, \Delta_y, \Delta_z)$. Antes de apresentar os elementos da grade, deve–se introduzir essas informações da seguinte forma

$$\begin{array}{ccc} N_x & N_y & N_z \\ x_0 & y_0 & z_0 \\ \Delta_x & \Delta_y & \Delta_z \end{array}$$

e a seguir os valores dos dados no ponto de grade. Por exemplo, seja $N_x = N_y =$

$N_z = 5$, $x_0 = 1.0$, $y_0 = 3.0$, $z_0 = 2.0$ e $\Delta_x = 0.5\,\Delta_y = 0.3\,\Delta_z = 0.8$, então o arquivo de dados tem a seguinte organização:

```
5 5 5
1.0 3.0 2.0
0.5 0.3 0.8
1.0 1.0 1.0 1.0 1.0
1.0 2.0 2.0 2.0 1.0
1.0 3.0 3.0 3.0 1.0
1.0 2.0 2.0 2.0 1.0
1.0 1.0 1.0 1.0 1.0
1.0 2.0 2.0 2.0 1.0
⋮
```

Após esses dados serem convertidos pelo programa MyExternalFilter, obtém–se o arquivo **".dx"** , apresentado no Código 5.2.2.

5.3 Exemplos do ".dx"

5.3.1 Dados Escalares em Grade Regular

Estes exemplos são uma adaptação de alguns exemplos **".general"** , dados anteriormente, para o formato **".dx"** .

a) Quando os dados são dependentes de posição, o *Object* do **".dx"** referente ao tipo de dado é apresentado no Código 5.3.1.

b) Quando os dados são dependentes da conexão, os *Object* do **".dx"** alterados são apresentado no Código 5.3.2.

5.3.2 Dados Escalares em Grade Regular com Múltiplos Campos

Quando se deseja introduzir grades regulares com múltiplos campos, primeiro introduz-se os dados, neste caso umidade e pressão, como apresentado no Código 5.3.3, e a seguir agrupam-se os campos como apresentado no Código 5.3.4.

```
# Definindo o primeiro campo ( umidade )
Object 1 class gridpositions counts 4 2 3
   origin 0.000000 0.000000 0.000000
   delta 1.000000 0.0 0.0
   delta 0.0 1.000000 0.0
   delta 0.0 0.0 1.000000
Object 2 class gridconnections counts 4 2 3
     attribute "ref" string "positions"
     attribute "elemente type" string "cubes"

Object 3 class array type float rank 0 itens file umidade.dat
Attribute "dep" string "positions"
Object 4 class field
component "positions" 1
component "connections" 2
component "data" 3

# Definindo o segundo campo ( pressao )
Object 5 class gridpositions counts 4 2 3
⋮
Object 6 class gridconnections counts 4 2 3
⋮
Object 7 class array type float rank 0 itens 24 file pressao.dat
⋮
Attribute "dep" string "positions"
Object 8 class field
component "positions" 5
component "connections" 6
component "data" 7
⋮
```

Código 5.3.3 – Parte de um arquivo "**.dx**" com múltiplos campos

```
#Montando a estrutura dos campos
Object 13 class group
member 0 value 4
member 1 value 8
⋮
```

Código 5.3.4 – Continuação do arquivo **".dx"** com múltiplos campos

```
#include<stdio.h>
#include<stdlib.h>
int  main(int argc , char argv*[]) {

FILE *pontin , *pontout;
char arqleitura[30] , arqgrava[30];
int numx, numy, numz, i, j, k;
     float originx, originy, originz ;
     float deltax, deltay, deltaz;
     float datavalue;

if (argc <3){
        printf("Programa nao foi acionado de forma correta\n");
        printf("Usar: %s <Arquivo de Leitura> <Arquivo de saída>\n", argv[0]);
        return 0 ;
        }
strcpy (arqleitura,argv[1]);
strcpy (arqgrava,argv[2]);
pontin=fopen(arqleitura,"r");
pontout=fopen(arqgrava,"wt");
fscanf(pontin, "%d %d %d", &numx, &numy, &numz);
fscanf(pontin, "%f %f %f", &originx, &originy, &originz);
fscanf(pontin, "%f %f %f", &deltax, &deltay, &deltaz);
fprintf(pontout,"bject 1 class gridpositions counts %d %d %d\n", numx,numy,numz);
fprintf(pontout,"   origin %f %f %f\n" , originx, originy, originz);
fprintf(pontout,"   delta %f 0.0 0.0\n", deltax);
fprintf(pontout,"   delta 0.0 %f 0.0\n", deltay);
fprintf(pontout,"   delta 0.0 0.0 %f\n", deltaz);
fprintf(pontout,"Object 2 class gridconnections counts %d %d %d\n", numx, numy, numz);
fprintf(pontout,"     attribute \"ref\" string \"positions\" \n");
fprintf(pontout,"     attribute \"elemente type\" string \"cubes\" \n");
fprintf(pontout,"Object 3 class array type float rank 0 itens %d data follows \n" ,
                numx*numy*numz);
for (i=0 ; i<numx ; i++) {
for (j=0 ; j<numy ; j++) {
for (k=0 ; k<numz ; k++) {
          fscanf(pontin, "%f", &datavalue);
          fprintf(pontout,"%f     ", datavalue);
         }
         fprintf(pontout,"\n", datavalue);
       }

      }
fprintf(pontout,"\n");
fprintf(pontout,"   attribute \"dep\" string \"positions\" \n");
fprintf(pontout,"Object 4 class field \n");
fprintf(pontout,"    component \"positions\" 1\n");
fprintf(pontout,"    component \"connections\" 2\n");
fprintf(pontout,"    component \"data\" 3\n");
fclose(pontin);
fclose(pontout);
  }
```

Código 5.2.1 – Programa MyExternalFilter.c

```
Object 1 class gridpositions counts 5 5 5
   origin 1.000000 3.000000 2.000000
   delta 0.500000 0.0 0.0
   delta 0.0 0.300000 0.0
   delta 0.0 0.0 0.800000
Object 2 class gridconnections counts 5 5 5
     attribute "ref" string "positions"
     attribute "elemente type" string "cubes"
Object 3 class array type float rank 0 itens 125 data follows
1.000000     1.000000     1.000000     1.000000     1.000000
1.000000     2.000000     2.000000     2.000000     1.000000
1.000000     3.000000     3.000000     3.000000     1.000000
⋮
Attribute "dep" string "positions"
Object 4 class field
component "positions" 1
component "connections" 2
component "data" 3
```

Código 5.2.2 – Arquivo **".dx"** gerado pelo programa Myexternalfilter.c

```
⋮

Attribute "dep" string "positions"
Object 4 class field
component "positions" 1
component "connections" 2
component "data" 3
```

Código 5.3.1 – Parte de arquivo **".dx"** ilustrando a dependência de posição

```
⋮

Attribute "dep" string "connections"
Object 4 class field
component "positions" 1
component "connections" 2
component "data" 3
```

Código 5.3.2 – Parte de um arquivo **".dx"** ilustrando a dependência de conexão

CAPÍTULO 6

Formato de dados ".grb"

Em Meteorologia, é comum os modelos de previsão numérica de tempo armazenarem suas saídas de dados no formato **".grb"** . Esse formato pode ser lido facilmente pela ferramenta GRADS (http://grads.iges.org/grads). Essa ferramenta é muito utilizada em Meteorologia, pois é desenvolvida para efetuar manipulação e visualização de dados meteorológicos. Entretanto sua eficiência de visualização de dados 3D e 4D é limitada. Em geral, é de grande interesse visualizar esses dados numéricos 4D: longitude, latitude, nível vertical e tempo para diversos campos atmosféricos, sendo que essas grades podem estar espaçadas de forma regular ou irregular.

Neste capítulo é apresentado como converter dados no formato **".grb"** para efetuar visualizações OpenDX .

6.1 Criando arquivos ".netCDF" a partir de aquivos ".grb"

O ambiente de entrada de dados do OpenDX aceita dados no formato **".netCDF"** (http://www.unidata.ucar.edu/packages/netcdf). Então utiliza-se um conversor conhecido com LATs4d, que transforma arquivos **".grb"** em arquivos **".netCDF"** , conforme o esquema da Figura 6.1. Esse conversor é um script do GRADS e encontra-se no endereço http://dao.gsfc.nasa.gov/software/grads/lats4d. A sintaxe utilizada na conversão é

lats4d -nc -i <arquivo ".ctl"> -o <arquivo ".netCDF" > -ftype ctl

6.2 Importação de arquivos ".netCDF"

Apesar do ambiente de entrada de dados do OpenDX aceitar dados no formato "netcdf", ele é incapaz de ler dados de diversas varíáveis com mais de três dimensões diretamente nesse formato. Esse é o caso dos modelos de previsão de tempo, como o MM5 (http://www.mmm.ucar.edu/mm5) e o ETA (http://www.cptec.inpe.br), em que as variáveis atmosféricas são 4D (x,y,z,t). Então, é necessário desenvolver um programa de visualização no OpenDX de forma a definir como os dados devem ser lidos e como devem ser visualizados, como apresentado na Figura 6.2.

No VPE, faz-se uso de algumas ferramentas para informar ao OpenDX como trabalhar esses dados multidimensionais. A primeira ferramente a ser utilizada é

a "Import", em que se informa onde está o arquivo **".netCDF"** . Para que o OPENDX saiba que os dados referem-se a dados multidimensionais, a ferramenta "Slice" deve ser usada. O próximo passo é separar as variáveis atmosféricas. Deve-se então utilizar a ferramenta "Slab" , para que os dados não sejam misturados na visualização. A ferramenta "Selector" também pode ser utilizada para selecionar a variável atmosférica. Utilizando a função "Selector" para cada variável, pode-se visualizá-las separadamente ou ao mesmo tempo em planos diferentes, unido as diversas visualizações com a ferramenta "Collect". Em ambos os caso a ferramenta "Image" gera a imagem da visualização.

A ferramenta "Sequencer" serve para gerar uma animação temporal desses dados. Isso encerra a parte de entrada de dados e inicia-se o processo de desenvolver a visualização.

6.2.1 Visualizações dos dados

As visualizações dos dados podem ser de diversos tipos:

- Todas as variáveis atmosféricas ao mesmo tempo em uma respectiva superfície isobárica;
- Uma variável nas suas diversas superfícies isobáricas;
- Visualizar os dados em um determinado tempo ou em forma de animação;
- Acoplar outras ferramentas disponibilizadas pelo VPE (gradientes, mapas, topografia 3D, etc.)

Assim pode-se criar um ambiente em que o usuário tem total liberdade para visualizar e manipular esses dados, escolhendo como quer visualizá-los ou ainda se quer girá-los adequadamente e/ou animá-los, criando assim um ambiente interativo e facilmente manipulável para auxiliar o entendimento conjuntos de dados complexos.

Os passos da visualização desses dados 4D na ferramenta VPE são apresentados a seguir.

a) Entrada com os dados por meio da ferramenta "Import";

b) Informação da dimensão dos dados por meio da ferramenta "Slice";

c) Aplicação das ferramentas "Select" e "Slab" para escolher a variável atmosférica de interesse e quais camadas a serem visualizadas. Como esses dados são compostos de várias variáveis, opta-se por usar um "Slab" e um "Selector" para cada uma delas;

d) Animação temporal desse dados com a ferramenta "Sequencer".

Terminada a fase de importação de dados e inicia-se a fase de visualização propriamente dita:

a) Para gerar padrões de falsas cores, utiliza-se a ferramenta "Autocolor";

b) Para destacar uma variável de forma 3D, utiliza-se a ferramenta "RubberSheet";

c) Para criar o padrão de cores associada a uma barra de conversão, utiliza-se o "ColorBar";

d) Para organizar o programa visual, utiliza-se as ferramentas "Receive" e "Transmiter", para receber as visualizações de cada variável atmosférica e transmiti-las para a ferramenta "Collect";

e) Para visualizar as imagens coletas, utiliza-se a ferramenta "Image".

Exemplos da desse tipo de visualização estão nas Figuras 6.3 e 6.4.

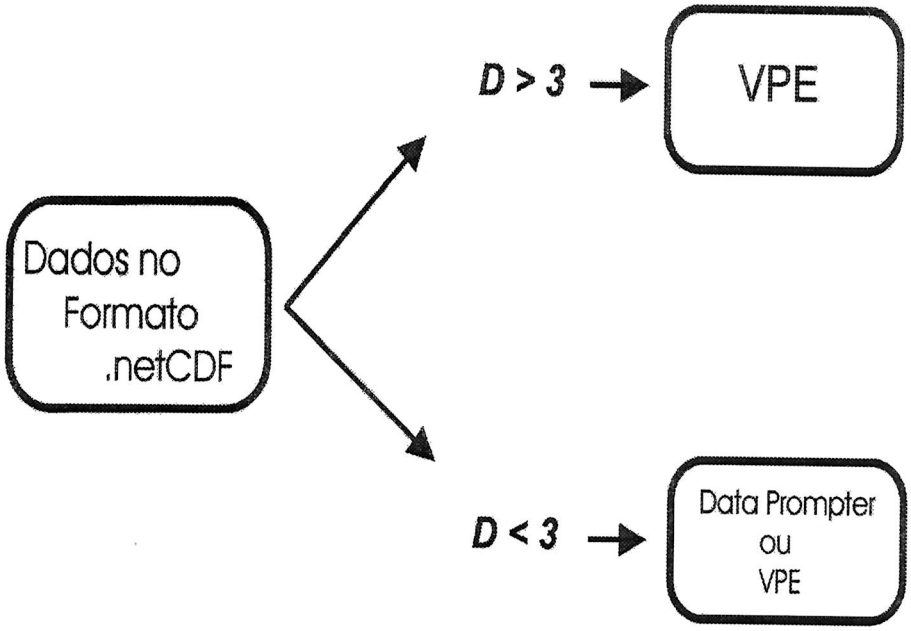


Fig. 6.1 – Esquema de importação de dados **".netCDF"**

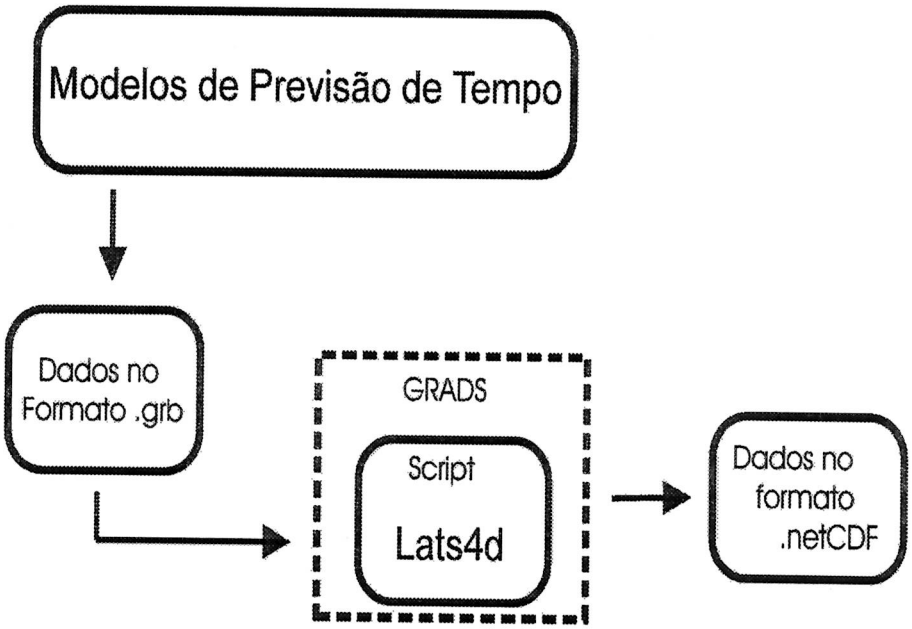


Fig. 6.2 – Esquema de transformação de dados grib

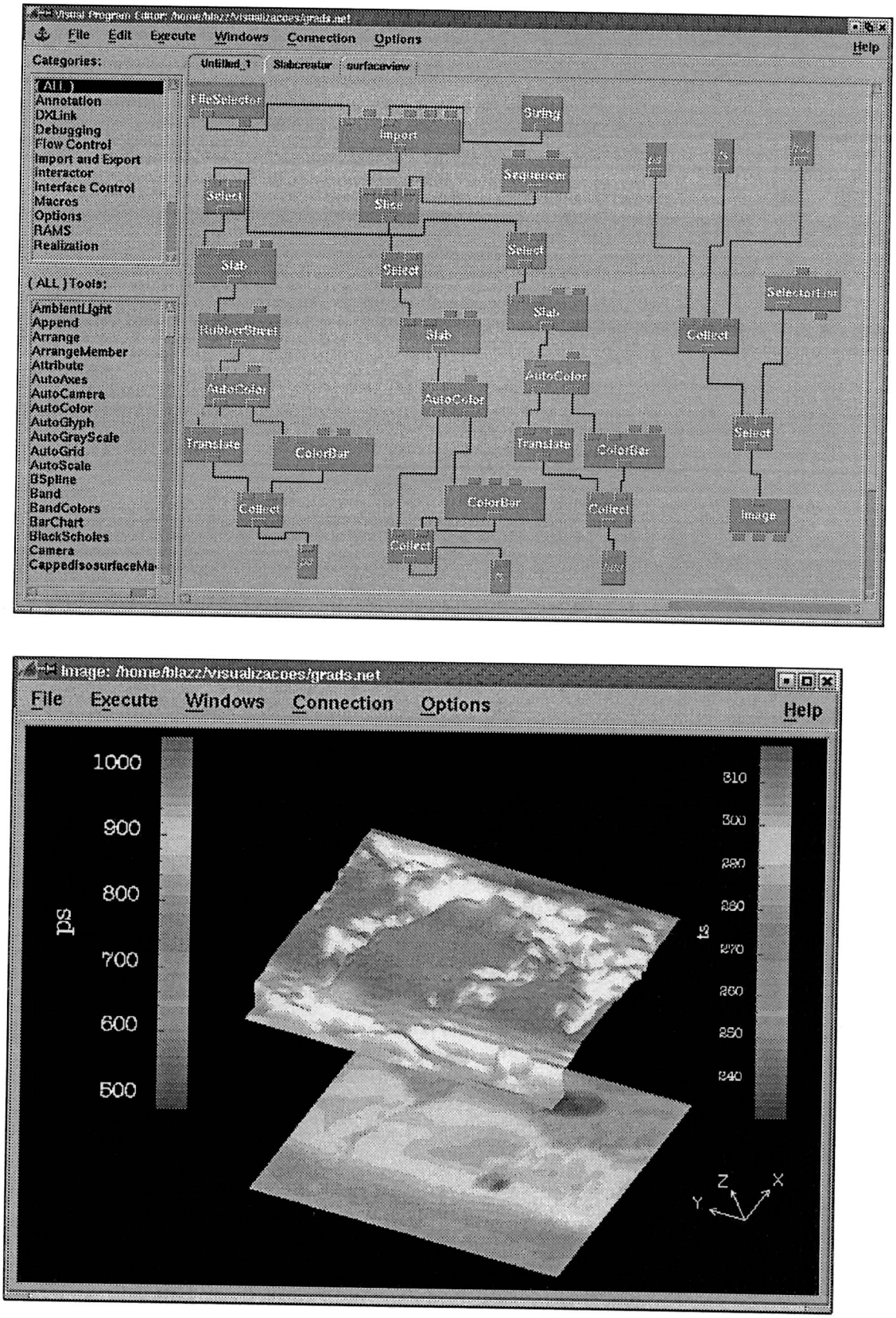


Fig. 6.3 – Programa de importação e a visualização desse dado **".netCDF"** 4D.

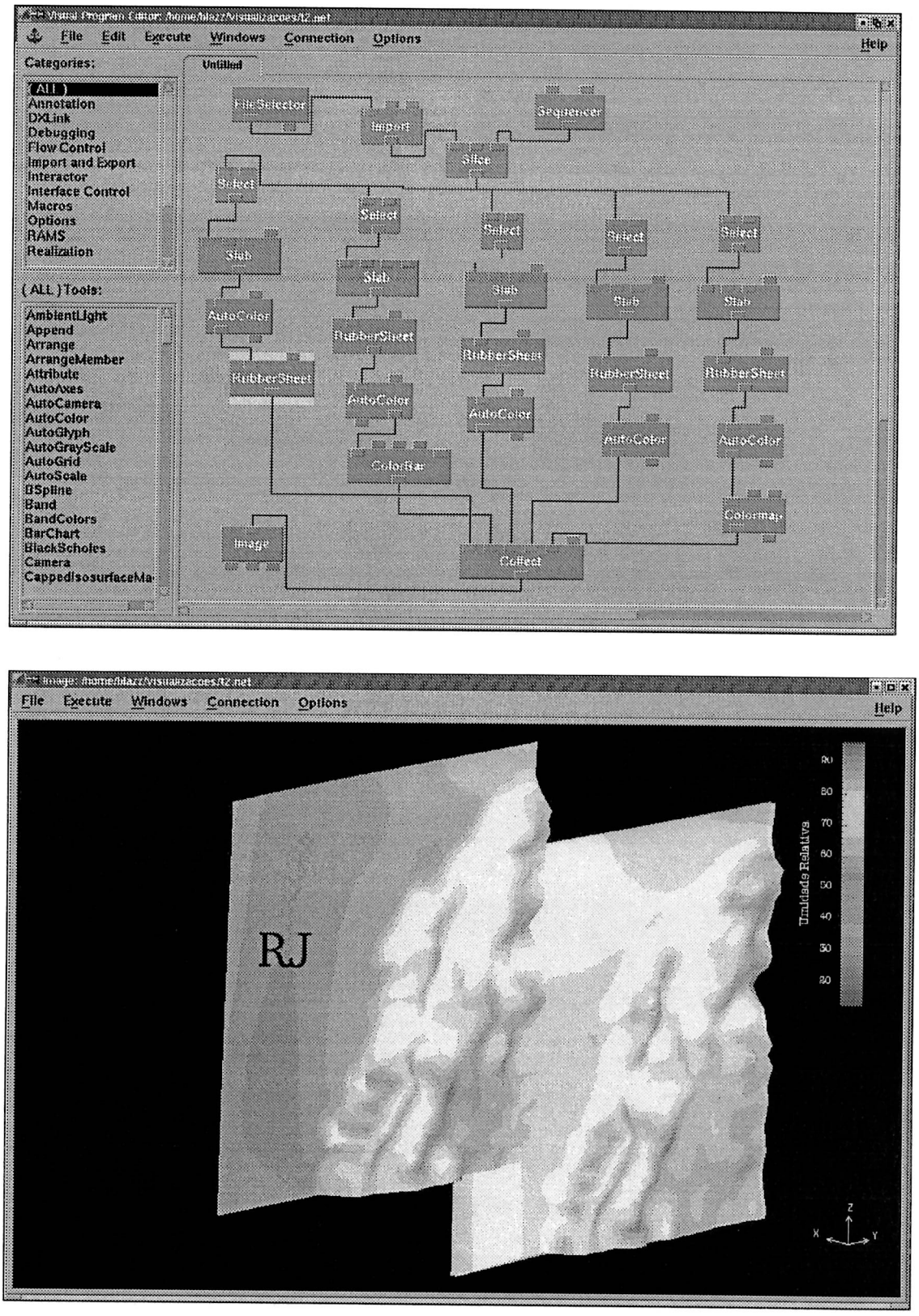


Fig. 6.4 – Exemplo das visualização do dado no formato **".netCDF"** 4D e seu programa de importação

CONCLUSÃO

Como constatado ao longo da leitura deste trabalho, o OPENDX requer um treinamento melhor dirigido de acesso aos dados para uma visualização. No entanto, fornece uma interface amigável ao usuário geral, que, após compreender as suas ferramentas, pode fazer visualizações de qualidade e sofisticadas. Esse texto cumpre seu objetivo básico de abrir perspectivas de utilização do OPENDX, ferramenta gratuita e de código aberto, mais amplamente no cotidiano dos meios acadêmicos e de pesquisas nacionais.

REFERÊNCIAS BIBLIOGRÁFICAS

IBM. **Visualization data explorer. programer 's reference.** SC38-0486-03. IBM, Estados Unidos, maio 1997a. <http://www.opendx.org>.

——. **Visualization data explorer. quickstart guide.** SC34-3262-02. IBM, Estados Unidos, setembro 1997b. <http://www.opendx.org>.

Kernighan, B. W.; Ritchie, D. M. **A linguagem de programação padrão ANSI.** Rio de Janeiro: Campus, 1990.

Thompsom, D.; Braun, J.; Ford, R. **Paths to visualization.** Estados Unidos: Visualization and Imagery Solutions,Inc, 2001. 207 p. <http://www.vizsolutions.com>.

APÊNDICE A

Etapas de Instalação do OpenDX

O OpenDX está disponível para várias plataformas. Na plataforma UNIX, as instruções contidas nas páginas do OpenDX são suficientes. Não ocorreram problemas na instalação em uma Sun com o Solaris 7. Por outro lado no caso do GNU/LINUX (Red Hat 6.2 e/ou Conectiva 6.1), percebeu-se que cada micro, na verdade, é um universo distinto de qualquer outro; embora possa parecer terem a mesma estrutura eletrônica e os mesmos pacotes computacionais. Outrossim, em GNU/Linux, a instalação depende da experiência da pessoa que está instalando. Para muitos PCs, a instalação da versão **4.10** é simples e ocorre da seguinte maneira:

a) Instala-se inicialmente os arquivos: bzip2-0.9.5d-1cl (que provê o libbz2.so.0 necessário ao ImageMagick) libstdc++-2.9.0-14cl (necessário para o OpenDx funcionar) libstdc++-2.95.1-2.10.0-3 (que complementa as bibliotecas).

b) Instala-se então o OpenDx. A seguir, para que o OpenDx funcione de qualquer lugar, é conveniente estabelecer um link simbólico para o comando de execução "dx". Para isso executar o seguinte comando: cp -l /usr/local/dx/bin/dx /usr/bin.

c) Neste ponto, se tudo"tiver transcorrido tranquilamente", é so testar o OpenDx com um dos exemplos contidos na pasta dxsamples-4.0.8. O comando IMPORT do OpenDx não encontra esse diretório diretamente, sendo necessário utilizar o comando FILESELECTOR para selecionar o diretório corretamente.

Caso haja placas controladoras de vídeo fora do comum, é possível que seja necessário instalar duas bibliotecas que lidam com o ambiente gráfico:

ibGL.so — que está contida no jogo Quake

libglide2x.so — requerida em placas de vídeo com acelerador

Elas devem ser copiadas nos diretórios:

/lib/libGL.so

/usr/lib/libglide2x.so

A instalação da versão **4.13** no Conectiva 7.0 foi mais simples:

a) Instala–se o OpenMotif conforme as instruções da página www.openmotif.org.

b) Instala–se a versão rpm do OPENDX conforme as instruções da página www.opendx.org.

c) Instala–se a versão rpm da documentação do OPENDX conforme as instruções da página www.opendx.org.

d) Instala–se outros exemplos de programas em OPENDX da Universidade de Cornell, ver página <http://www.tc.cornell.edu/Services/Vis/dx>.

A versão RedHat 7.2 não necessita da instalação do OpenMotif, pois esta versão já vem com LessMotif instalado.

ÍNDICE REMISSIVO

Symbols

C

D

F

G

S

V