Visualização Científica dos Resultados de Métodos Híbridos de Wavelets + Diferenças Finitas.

RELATÓRIO DE PROJETO DE INICIAÇÃO CIENTÍFICA – 1° ANO
(PIBIC-INPE/CNPq)
— Distribuição Restrita / Versão em desenvolvimento —

Roberto Blaz (CPTEC/LMO, Bolsista PIBIC-INPE/CNPq)
E-mail: blazz@cptec.inpe.br

Dra. Margarete Oliveira Domingues (CPTEC/LMO/INPE, Orientador)
E-mail: margaret@cptec.inpe.br

Laboratório Associado de Meteorologia e Oceanografia
Centro de Previsão de Tempo e Estudos Climáticos
Maio de 2002

RESUMO

Em Hidrodinâmica são comuns as situações em que o escoamento apresenta estruturas com variações bruscas. Para uma representação adequada dessas estruturas são necessárias grades muito refinadas. Em muitos casos, essas estruturas possuem uma localização espacial reduzida em comparação com o domínio do escoamento. Por essa razão, há um interesse por métodos numéricos adaptativos que usem subgrades mais refinadas apenas nessas regiões, e subgrades menos refinadas nas regiões em que o escoamento seja mais suave. Este trabalho tem como desafio visualizar dados de um desses modelos de adaptabilidade espacial, conhecido como WDF, de forma a minimizar os esforços computacionais de reconstrução dessas subgrades em sua grade regular mais refinada. Os dados do programa WDF são setorizados em blocos de dados contendo sua posição relativa a grade mais refinada e seu nível de refinamento. Foi escolhido o pacote de visualização científica OpenDX para realizar tal tarefa. O OpenDX é um poderoso pacote de visualização mutidimensionais de alta qualidade e portabilidade, de código aberto e de distribuição gratuita. Ele possui recursos de manipular, processar, transformar, criar processos de visualização e animar dados em diversas plataformas. Para que o OpenDX fosse capaz de entender essa estrutura de dados, foi programada a entrada de dados num formato próprio do OpenDX, de forma que os diversos blocos de dados pudessem ser reposicionados na malha original de acordo com a sua posição e refinamento. Após esse procedimento foi desenvolvido um programa de visualização desses dados no ambiente gráfico do OpenDX. Com tal procedimento foi possível visualizar o escoamento de interesse sem a reconstrução da malha mais refinada e sem perda de informação visual. O domínio do uso dessa ferramenta abre perspectivas de emprego dessa ferramenta em outras aplicações e a criação de novos tipos de visualizações com outros dados de modelos numéricos, como os de previsão de tempo.

SUMÁRIO

LISTA DE FIGURAS

CAPÍTULO 1

INTRODUÇÃO

Em Hidrodinâmica são comuns as situações em que o escoamento apresenta estruturas com variações bruscas. Para uma representação adequada dessas estruturas são necessárias grades muito refinadas. Em muitos casos, essas estruturas possuem uma localização espacial reduzida em comparação com o domínio do escoamento. Por essa razão, há um interesse por métodos numéricos adaptativos que usem subgrades mais refinadas apenas nessas regiões, e subgrades menos refinadas nas regiões em que o escoamento seja mais suave.

Os dados visualizados nesse projeto vem dos modelos de adaptabilidade espacial conhecido como WDF Domingues (2001). O problema envolvido é o de minimizar os esforços computacionais e evitar a reconstrução das subgrades menos refinadas na grade regular mais refinada na hora de efetuar-se a visualização dos dados.

Foi escolhido o pacote de visualização científica OpenDX para realizar a tarefa de visualizar tais malhas, pois ele possui recursos de manipular, processar, transformar, criar processos de visualização e animar dados. A história do OpenDX começou em 24 de maio de 1999, quando a IBM proprietária do código fonte do Data Explorer, liberou parte desse código a um grupo de colaboradores. Esse grupo desenvolveu um pacote de visualização de dados a partir desse código fonte. Esse pacote de visualização tornou-se mais tarde o OpenDX <http://www.opendx.org>. Essa ferramenta é gratuita, multiplataforma, possui código aberto e tem como maior colaborador David Thompson.

O objetivo científico do trabalho proposto é estender as facilidades do poderoso pacote gratuito de visualização científica OpenDX a resultados numéricos de modelos e dados meteorológicos. Isso resulta no benefício de rápida interpretação de resultados, possibilita a aplicação a outros fenômenos pertinentes; e facilita a utilização dessas informações pela comunidade.

CAPÍTULO 2

MALHAS ADAPTÁVEIS

O método das híbrido wavelets+diferenças finitas foi sugerido por Holmström (1997) e desenvolvido por Domingues (2001); Domingues et al. (2002). Por um lado, está um esquema tradicional de diferenças finitas em malha uniforme. Por outro lado está a discretização com adaptação de funções por valores pontuais em malha irregular. A análise wavelet contribui na construção de tais malhas e na construção de um esquema de interpolação adaptativa para fazer a conecção entre os ambientes uniforme e não uniforme, utilizando as técnicas de análise wavelet interpolatória. Para o refinamento local da malha utilizam-se os coeficientes wavelet significativos como indicadores das regiões de pouca suavidade da função representada.

Nesse tipo de técnica, a estrutura da malha adaptativa resultante é totalmente dependente da função analizada, podendo apresentar uma composição bem heterogênea: esparsa em regiões de suavidade e densa nas regiões de pouca suavidade. Esse fato aumenta a complexidade da estrutura de dados, principalmente em dimensões superiores. Por isso, para o caso bidimensional, são preferidas malhas que sejam localmente regulares, em blocos, como as visualizadas neste trabalho. Isso facilita as operações de acesso, amazenamento e dos cálculos, com o custo de se ter um maior número de pontos nas malhas.

Os coeficientes wavelet do caso bidimensional também podem ser interpretados como erros de interpolação. Portanto, em princípio, a metodologia pontual se aplica também ao caso bidimensional. No entanto, a topologia das malhas irregulares bidimensionais podem aumentar a complexidade de armazenamento, de busca e de cálculos. Por isso, são geralmente preferidas aquelas malhas que sejam localmente regulares, em blocos, como as visualizadas neste trabalho.

No caso bidimensional são utilizadas malhas definidas por partições de um retângulo $[0, L] \times [0, D]$. No nível menos refinado, é assumido que $\mathcal{X}^0$ é a malha com $N_x \times N_y$ pontos obtida pela partição regular do retângulo, com espaçamento $h_x = \frac{L}{N_x}$ na direção de x e $h_y = \frac{D}{N_y}$ na direção de y. Isto é

$$\mathcal{X}^0 = \{\gamma = (kh_x, \ell h_y), 0 \leq k < N_x, 0 \leq \ell < N_y\}.$$

Por refinamento diádico, têm–se as malhas regulares

$$\mathcal{X}^j = \left\{\gamma = (kh_x^j, \ell h_y^j), 0 \leq k < N_x^j, 0 \leq \ell < N_y^j\right\},$$

em que $N_x^j = 2^j N_x$, $N_y^j = 2^j N_y$, e $h_x^j = 2^{-j} h_x$ e $h_y^j = 2^{-j} h_y$. Desta maneira, tem–se uma sequência de malhas encaixadas $\mathcal{X}^j \subset \mathcal{X}^{j+1}$, em que $\mathcal{X}^{j+1}$ é obtida incluindo pontos médio em $\mathcal{X}^j$.

Considerando a raiz $\mathcal{B}_{(0,0)}^0 = \mathcal{X}^0$, após $\mathcal{J}$ gerações tem–se uma árvore quaternária completa de $\mathcal{J}+1$ níveis. Em cada nível $0 \leq j \leq \mathcal{J}$ existem 2^{2j} blocos (nós), cujos pontos diretores integram um conjunto que denominamos $\mathcal{I}^j$.

Inicia-se a construção da malha $\mathcal{M} = \mathcal{M}_\epsilon$ a partir do bloco inicial $\mathcal{B} = \mathcal{X}^0$ e faz-se $\imath_\epsilon(\mathcal{B}) = 1$. A seguir, calcula-se o indicador $\imath_\epsilon$ de seus filhos. Se o indicador for nulo em todos os quatro casos, então o bloco $\mathcal{B}$ é considerado folha e integra-se à malha $\mathcal{M}$. Caso contrário, apenas são consiredados folha, e integrados à malha, aqueles blocos filhos com indicador nulo. Nos demais blocos, com indicador igual a um, repete-se o processo de teste até alcançar um nível em que todos os blocos são considerados folha. Na prática, no entanto, pode ser necessário antecipar a interrupção deste processo de construção devido a limitações físicas da máquina, em cujo caso, estabelece-se um número máximo de blocos presentes na malha. Esse procedimento é resumido nos Algoritmos 2.1 e 2.2. Esse esquema de construção é apresentado na Figura 2.1.

Algorithm 2.1 – Construção de Malhas: $Constr(\mathcal{B}, \mathcal{F}, \epsilon)$.

$\forall \mathcal{B}_\gamma \in \mathcal{S}(\mathcal{B})$ **do**
if $\imath_\epsilon(\mathcal{B}_\gamma) = 0$, $\forall(\mathcal{B}_\gamma) \in \mathcal{S}(\mathcal{B})$ **then**
  $\mathcal{F} = \mathcal{F} \bigcup (B)$
  return
else
  if $\imath_\epsilon(\mathcal{B}_\gamma) = 0$, **then**
    $\mathcal{F} = \mathcal{F} \bigcup (\mathcal{B}_\gamma)$.
  else
    Apply: $Constr\left(\mathcal{B}_\gamma, \mathcal{F}, \epsilon\right)$.
  end if
end if
end do

Algorithm 2.2 – Indicador de Refinamento: $\imath_\epsilon(\mathcal{B})$

m = Level $(\mathcal{B})$-1.
if $|d_{k,\ell}^{(\alpha)\,m}| \leq \epsilon \; \forall \; d_{k,\ell}^{(\alpha)\,m} \in \mathcal{D}(\mathcal{B})$ **then**
  return 0 (*Não há coeficientes wavelet significativos*)
else
  return 1 (*Há coeficientes wavelet significativos*)
end if

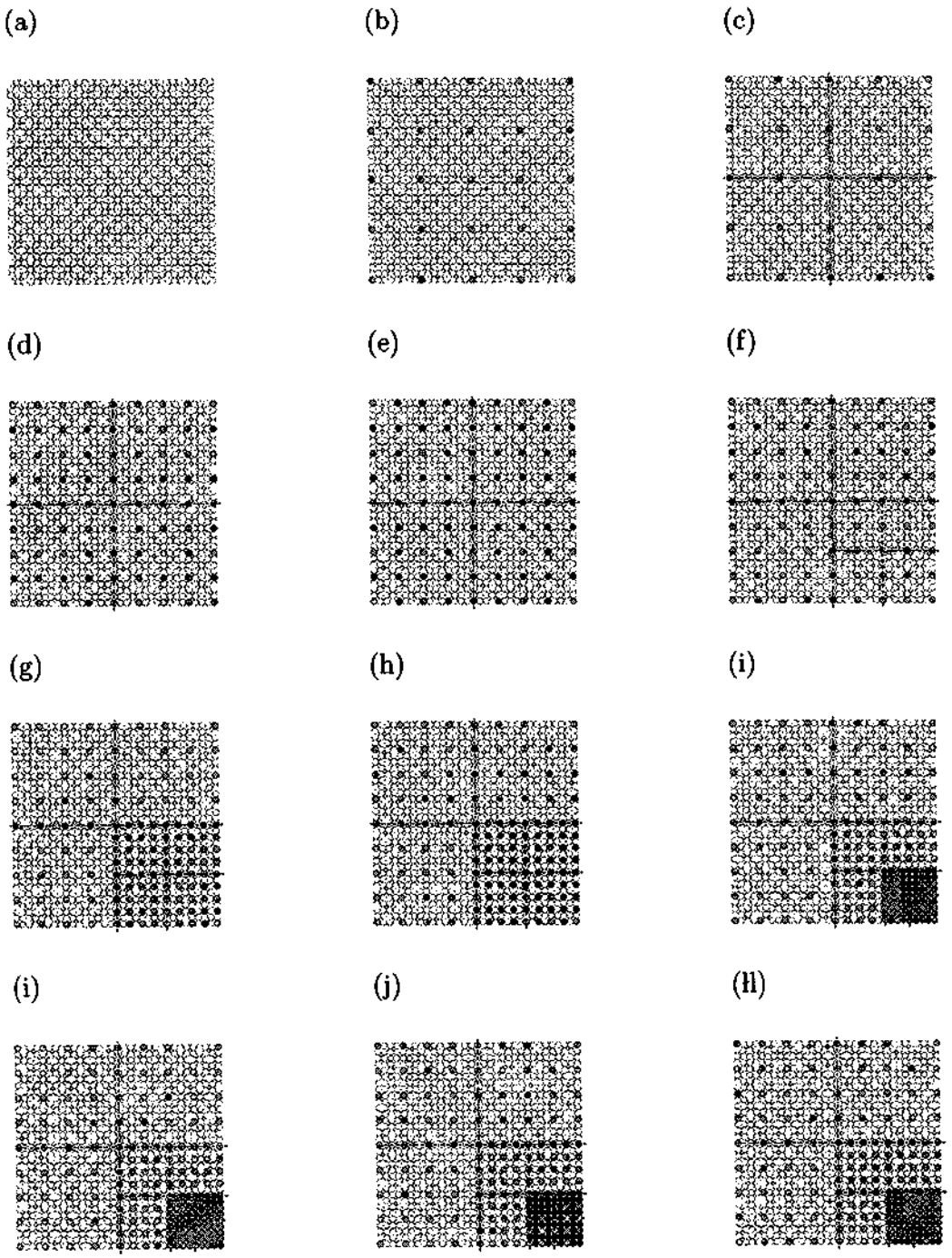

Fig. 2.1 – Esquema de construção das malhas adaptáveis

CAPÍTULO 3

OpenDX: Formato ".dx"

O formato ".dx" do OpenDX é bastante flexível. Pode-se representar, a princício qualquer objeto. Para importa-se este arquivo por meio do Data Prompter do OpenDX, deve-se apenas especificar que o arquivo é um arquivo nativo do OpenDX. Ele também pode ser importado diretamente no ambiente de programação visual do OpenDX (VPE) pela ferramenta *File Select→Import*. Como gerar os dados neste formato é o assunto brevemente tratado neste Capítulo. As informações apresentadas são baseadas nos manuais IBM (1997b,a); Thompsom et al. (2001).

3.1 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 sequência de definições de objeto, cada definição de objeto consiste de uma sequência 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 especifíico 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 seqüênciais 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 especifico 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 tem seu número de componentes especificado 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* especificão 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–se 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.

No Código 1 é apresentado um exemplo de um arquivo **".dx"** para leitura de uma matriz $5 \times 5 \times 5$.

```
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 1 – Contéudo de arquivo **".dx"**

CAPÍTULO 4

VISUALIZAÇÃO NO OpenDX

4.1 Funções representadas em malhas adaptáveis

Visualizar dados em malhas irregulares é um recurso disponível no OpenDX. Por outro lado, dada a estrutura de sub-malhas a serem representadas, verificou-se ser mais simples e menos dispendioso computacionalmente fazer visualização por meio do conjunto de sub-matrizes.

O maior desafio encontrado nesta parte do trabalho é preparar a entrada das sub-matrizes no padrão ".dx" para visualizar os dados no domínio sem precisar compor a malha refinada correspondente, e descobrir um meio de visualizar esses dados 2D de forma 3D. Basicamente cada sub-malha é representada por um trio de objetos: gridposition, gridconnection, array type.

O gridposition é o responsável por determinar quantos pontos existem na sub-malha, quais são suas coordenadas iniciais (ponto mais ao sul e mais a oeste) e seus espaçamentos na direção norte-sul e na leste-oeste.

O gridconnection define como serão conectados os pontos da malha. Neste caso se utiliza uma conexão de malha regular. Para simular a conexão entre as sub-malhas, usa-se o seguinte truque: considera-se que cada sub-malha compartilha suas fronteiras direita e superior (internas ao domínio) com suas respectivas sub-malhas vizinhas.

O objeto array type define como a matriz de dados está organizada. Neste caso ela está obedecendo uma variação do tipo linha/coluna, i.e., $(x_0, y_0)(x_0, y_1)(x_0, y_2)(x_0, y_3) \cdots (x_1, y_0)(x_1, y_1)(x_1, y_2)(x_1, y_3) \cdots$. O nome do arquivo de dados que será lido fica definido neste objeto. Após definir esses três objetos, cria-se o campo de dados a ser lido, com a instrução field. Nessa instrução informa-se quais trios de objetos estão sendo incluídos nesse campo.

No final do processamento, agrupa-se os campos na instrução group, em que o número a seguir da instrução que value determina o campo de dados definido. A função membro de formatação do arquivo ".dx" em C++ é apresentado nos Códigos 2, 3, 4, 5 e o arquivo ".dx" gerado é apresentado no Código 6.

No VPE para a representação 3D desses dados é utilizada a função **rubber sheet**. Nessa forma de visualização os dados são tratados como se uma topografia, e o padrão de cores é sobreposto ao relevo. Na Figura 4.1 é apresentado um exemplo de programa de visualização desses dados do modelo adaptável com três opções de visualização: a superfície 3D gerada, a malha com os pontos utilizados. Na Figura 4.2 estão os resultados desse programa.

4.2 Outros dados de modelos numéricos

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 interesse visualizar esses dados numéricos 4D: longitude, latitude, nível vertical e tempo para diversos campos atmosféricos, e essas grades podem estar espaçadas de forma regular ou irregular.

Neste seção é apresentado como converter dados no formato **".grb"** para efetuar visualizações OPENDX.

4.3 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 4.3. 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 ".grb" > -o <arquivo ".netCDF" > -ftype ctl

4.4 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 4.4.

No VPE faz-se uso de algumas ferramentas para informar ao OpenDX como trabalhar esses dados multidimencionais. A primeira ferramente a ser utilizada é a "Import", em que informa–se onde está o arquivo **".netCDF"** . Para que o OpenDX saiba que os dados tratam-se de dados multidimencionais a ferramenta "Slice"deve ser usada. Então, o próximo passo é separar as variáveis atmosféricas. Deve-se 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.

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 eram 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 começa 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 em associado 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 transmitíi-las para a ferramenta "Collect";

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

Exemplos da visualização descrita são mostrados na Figura 4.5 e na Figura 4.6.

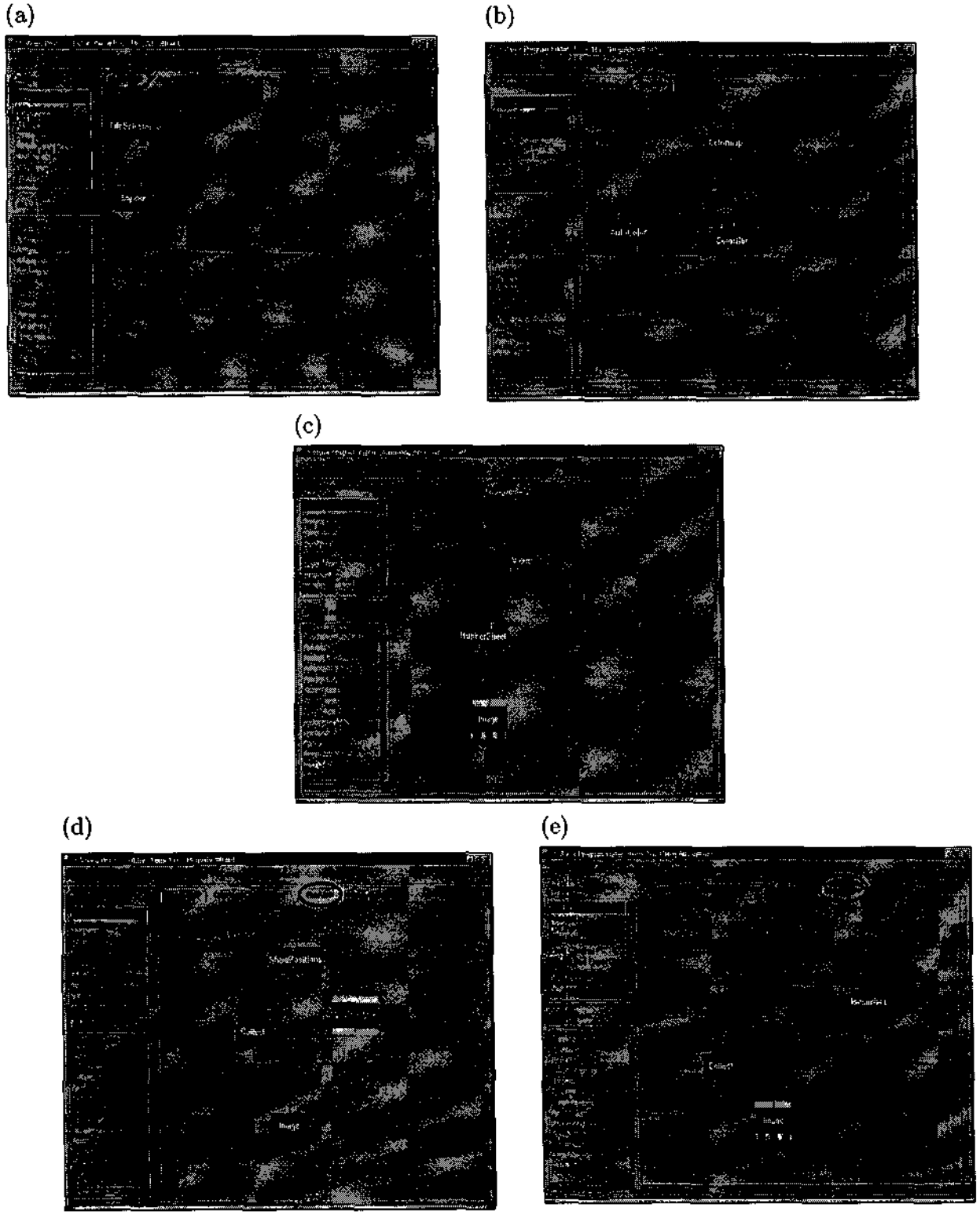

Fig. 4.1 – Exemplo de programa de visualização de dados de modelo adaptável

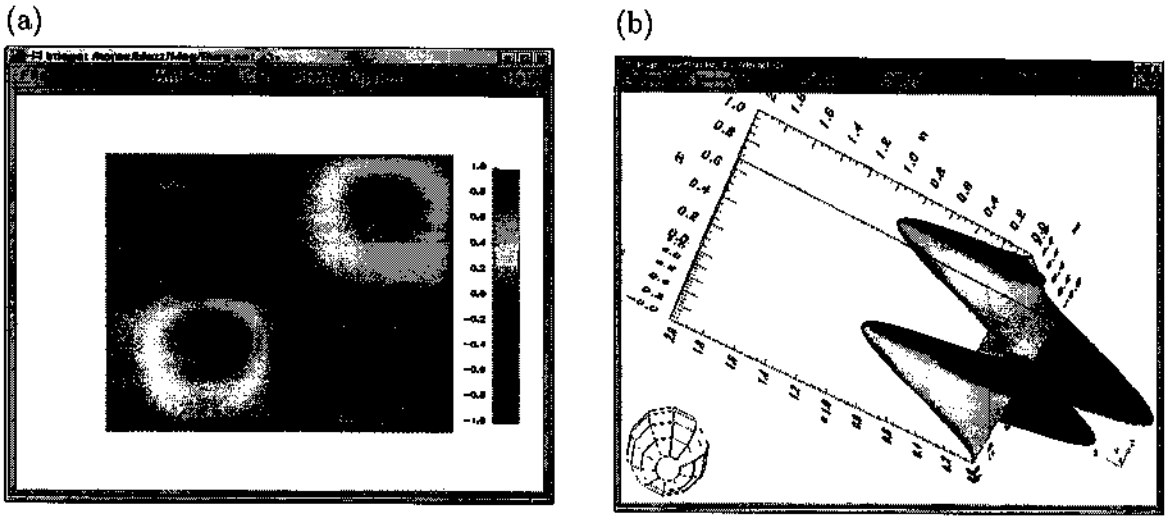

Fig. 4.2 – Visualização dos dados de modelos adaptávelveis

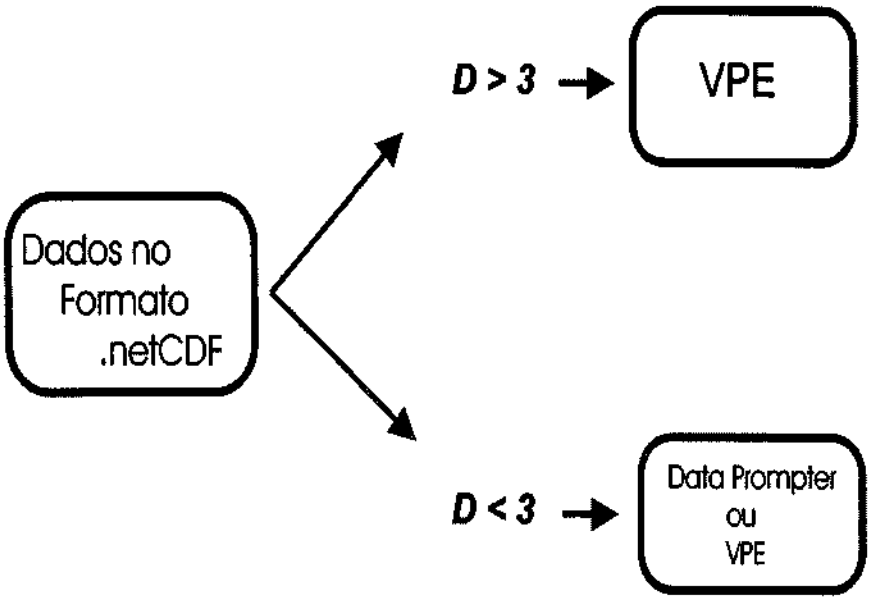

Fig. 4.3 – Esquema de importação de dados **".netCDF"**

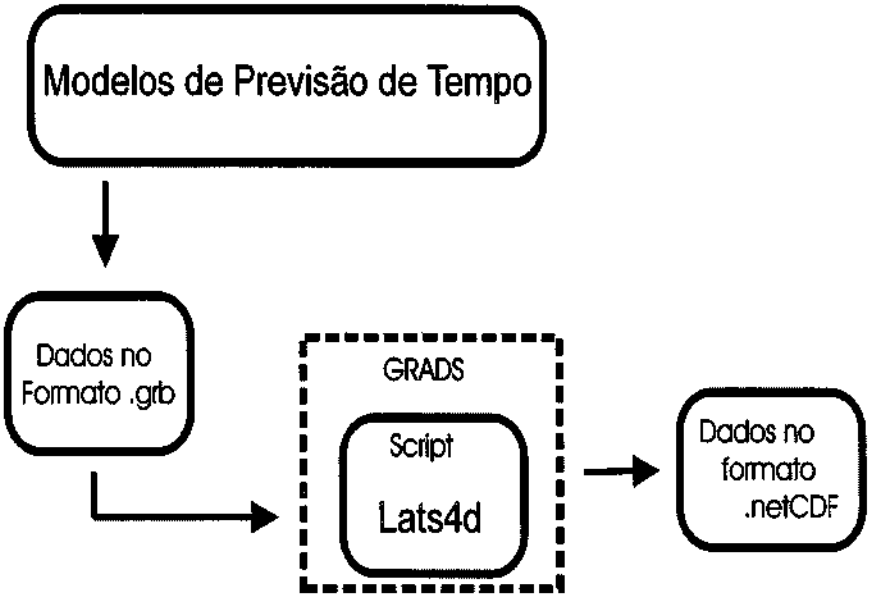

Fig. 4.4 – Esquema de transformção de dados grib

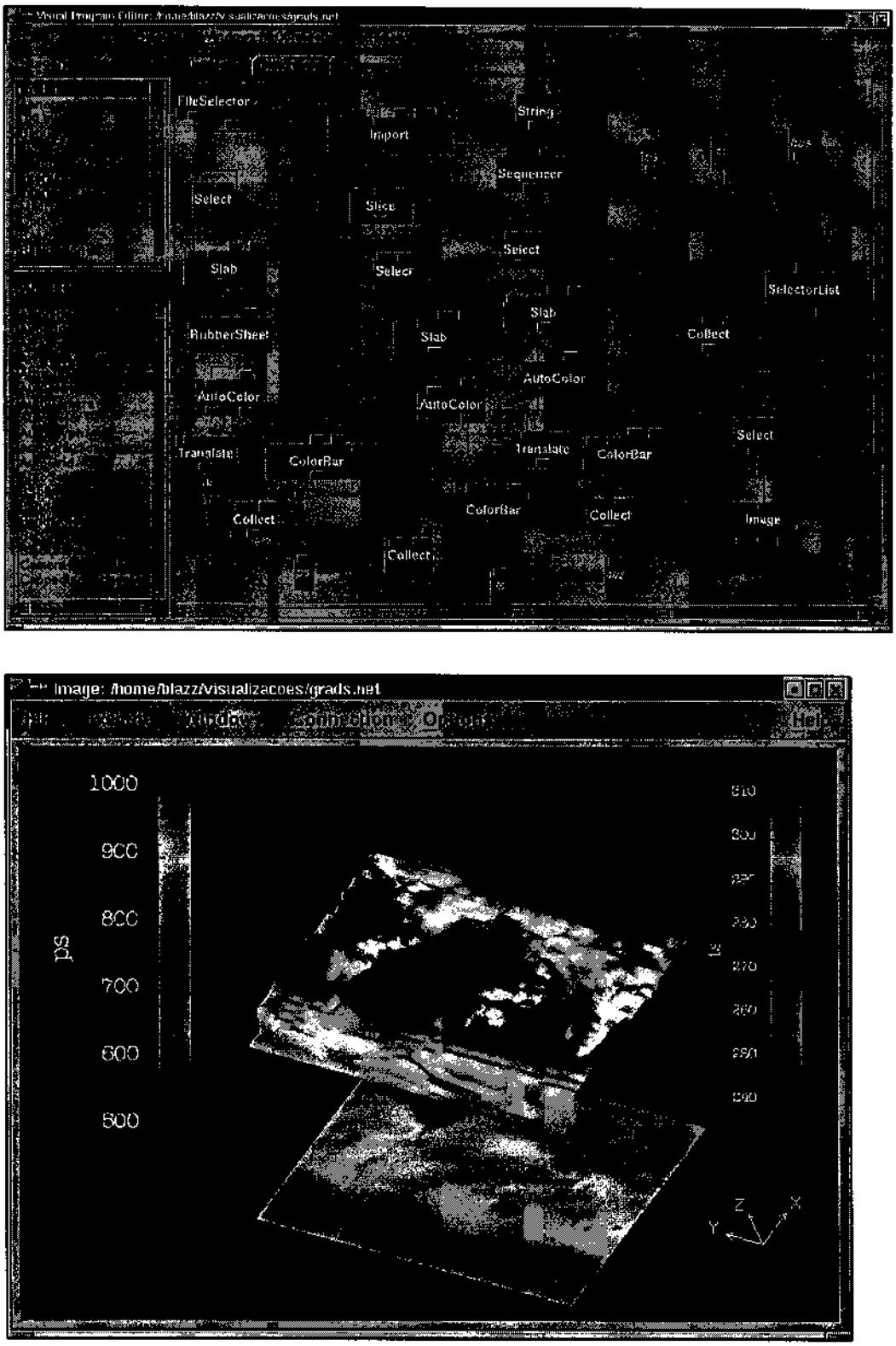

Fig. 4.5 – Programa de importação e a visualização desse dado **".netCDF"** 4D.

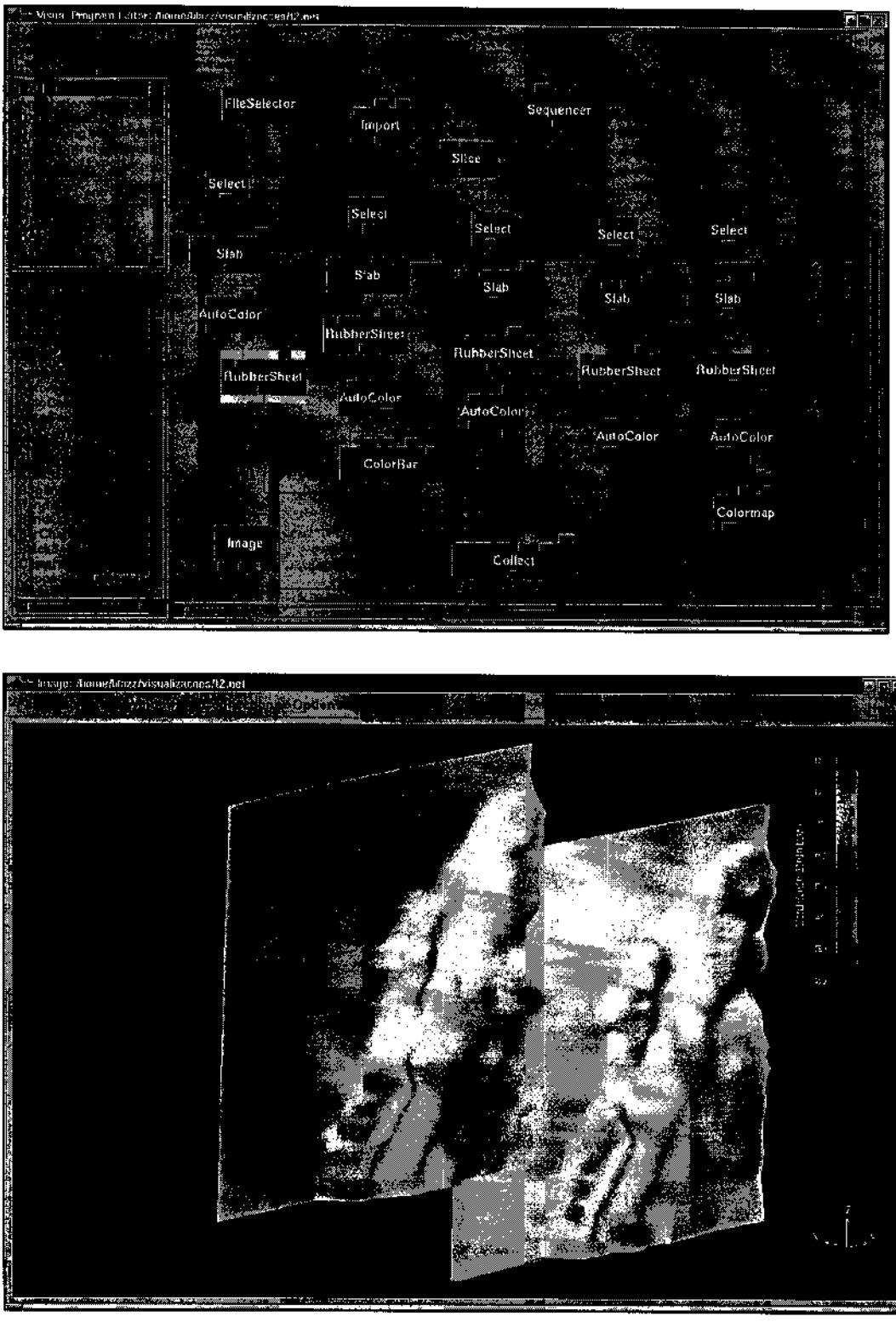

Fig. 4.6 – Exemplo das visualização do dado no formato **".netCDF"** 4D e seu programa de importação

CAPÍTULO 5

CONCLUSÕES

Nesta fase inicial do projeto as atividades de complementação de formação academica do bolsista foram as principais principais desenvolvidas. Como parte desta atividade o bolsista estará publicando uma nota técno científica no INPE sobre algumas formas de entrada de dados no OPENDX, cujo texto é apresentado anexo.

Um dos grandes exitos desta parte inicial da pesquisa foi a melhoria na implementção da entrada de dados dos diversos blocos da malha malha adaptável no OPENDX. Ocorreram também avanços nas etapas de entendimento dos mecanismos da visualização de dados no OPENDX e algumas visualizações mais complexas já foram possíveis. Parte dos resultados obtidos nesta fase o bolsista estará apresentando em um poster na 54ª Reunião da SBPC, cujo resumo é apresentado anexo.

REFERÊNCIAS BIBLIOGRÁFICAS

Domingues, M. O. **Análise wavelet na simulação numérica de equações diferenciais parciais com adaptabilidade espacial.** Campinas, SP. Tese – IMECC/UNICAMP, outubro 2001.

Domingues, M. O.; Gomes, S. M.; Alvarez, L. M. D. Adaptative wavelet representation and differenciation on block-structed grids. **Special Issue of the PanAmerican Workshop in Applied Mathematics & Computation, Elsevier**, 2002. Submitted.

Holmström, M. **Wavelet based methods for time dependent pdes.** Tese – Uppsala University, 1997.

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>.

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

RELATÓRIO DE ATIVIDADES

A.1 Fase Atual do Projeto

As atividades desenvolvidas nesses sete meses de etapa inicial foram de introdução ao problema a ser implementado e de complementação da formação acadêmica do bolsista. Essas etapas estão de acordo com o plano de trabalho proposto. Além da parte de complementação acadêmica na parte computacional do estudante, a forma de redação e apresentação de relatórios técnicos e o início da implementacão de programas visuais no OPENDX também foram desenvolvidos. Maiores detalhes são apresentados no relatório anexo.

Para facilitar o entendimento dos processos de entrada de dados no OPENDX, procedimentos com complexidades, o bolsista dedicou-se a esse tópico, neste período de bolsa. Como resultado desse trabalho e para difundir as facilidades de entrada de dados no OPENDX por outros usuários, o bolsista está redigindo um relatório técnico sobre esse assunto. Esse relatório é apresentado anexo, em versão preliminar.

Com a familiaridade iniciada com o OPENDX nesta etapa, o bolsista já está desenvolvendo programas no ambiente de programação visual, ainda que de forma modesta.

A.2 Complementação Acadêmica do Bolsista

Foram estudados aspectos computacionais gerais e específicos para o desenvolvimento do trabalho proposto. A seguir são apresentadas em linhas gerais as etapas já realizadas.

A. Aspectos Computacionais Gerais

Foram estudados tópicos básicos para execução do projeto

1) Introdução ao sistema operacional GNU/LINUX e as implicações de licenças GNU;
2) Noções de Programação em GNU/LINUX : compiladores GNU/gcc, utilitário GNU/make, depurador GNU/gdb, ferramentas de controle de versão como o rcs, e editores de texto mais utilizados como o kwrite.

3) Noções de outras ferramentas de visualização científicas e de tratamento de imagens, em especial do GNU/gnuplot;

4) Introdução ao LaTeXe suas facilidades de editoração digital;

B. Aspectos Computacionais Específicos

1) Introdução a entrada de dados no OpenDX;

2) Introdução ao ambiente de programação visual do OpenDX;

3) Introdução a liguagem de programação visual do OpenDXe suas facilidades;

4) Introdução a linguagem de programação C

A.2.1 Outras Atividades Acadêmicas

O bolsista também participou do curso de Fortran 77 ministrado no setor de treinamento INPE para alunos da Pós Graduação em Meteorologia em novembro de 2001(certificado anexo) e do curso de Introdução ao Spring oferecido pelo SERE/OBT/INPE (http://sputnik.dpi.inpe.br:1905/col/dpi.inpe.br/lise/2001/02.21.17.49/doc/spring.htm).

A.2.2 Planos para o 2º ano do Projeto

Nesta fase dar-se-á enfase aos aspectos teóricos do projeto:

a) Rudimentos da teoria wavelet e de métodos de diferenças finitas;

b) Como são gerados os dados que serão visualizados, como se enquadra seu trabalho de Iniciação Científica no projeto de Métodos Híbridos, e qual a sua importância nesse contexto (cont.);

c) Forma de apresentação escrita de relatórios e artigos (cont.);

d) Formas de apresentação oral, uso de ferramentas disponíveis;

e) Introdução a outros problemas de visualização científica de interesse em Meteorologia (cont.).

Será estudado em detalhes a forma de entrada de dados de malhas regulares por blocos e implementados programas visuais para apresentação desses dados, como a

APÊNDICE B

CÓDIGOS GERADOS

```
void TMesh::PrintDataExplorer(int kmin,  int kmax, int lmin, int lmax, char *idName){

int var,block,mother,k,l, bmax, kk,ll,kp,lp,object;
char aux[140], auxdx[140];
double x,y,level,dx,dy,data;
kk =  (*this).kN;  ll =   (*this).lN;
kp=kmax + abs(kmin) +1;  lp=lmax + abs(lmin) +1;
object =0;
if( (bmax = (*this).fkblock) == 0) bmax=1;
   for(block=0;block<bmax;block++){
         level= Data[block].WhichLevel();
         dx=1.0f/(kk   * pow(2.0f,level) );
         dy= 1.0f/(ll  * pow(2.0f,level) );
          x=Data[block].WhereIsX()+kmin*dx;
          y=Data[block].WhereIsY()+lmin*dy;
          fstream out;
          sprintf(aux,"PF%d.%s.dat",block+1,idName);
          sprintf(auxdx,"MeshP.%s.dx",idName);
          out.open(aux,ios::out);
  for(k=kmin;k<=kmax;k++){
                for(l=lmin;l<=lmax;l++){
                      data=Data[mother][block].ff00[k][l];
                       if(fabs(data) < 1E-30) data=0.0f;
                       out<<data<<" ";
                       }
out<<endl;
     }
      out.close();
      if(block==0){
      (*this).DxFormatDataBegin(x, dx, y,dy, kp, lp,aux, auxdx,object);
      object+=5;
      }
  else{
       if((block==bmax-1)){
          (*this).DxFormatDataInside(x, dx, y,dy, kp,lp,aux,auxdx,object);
          object+=3;
          (*this).DxFormatDataEnd(auxdx,object);
       }
  else{
          (*this).DxFormatDataInside(x, dx, y,dy, kp,lp, aux, auxdx,object);
          object+=3;
           }
  }
}
```

Código 2 – Função membro de formatação do arquivo ".dx" em C++

```
void TMesh::DxFormatDataInside(double x0,double dx,double y0,
     double dy,int nx,int ny,char*  filedata, char* filedx, int object)
{
fstream poutdx;
int items = nx*ny;
poutdx.open(filedx,ios::app); //Para unir as diversas sub-malhas
poutdx <<endl<<"# Definition of regular positions and conections"<<endl;
poutdx << "object "<<object<<" class gridpositions counts "
       <<nx<< " "<<ny<<"  "<<endl;
poutdx <<"origin "<<x0<< " "<<y0 << "  "<<endl;
poutdx <<"delta "<<dx<< " 0  "<<endl;
poutdx <<"delta  0 "<<dy<<"  "<<endl<<endl;
poutdx<<"object "<< object+1<<" class array type float rank 0 items "
       << items<<" data file "<<  filedata<<endl<<endl;
poutdx<<"attribute \"dep\" string \"positions\" "<<endl<<endl;
poutdx<< "object "<<object+2<< " class field"<<endl;
poutdx<< "component \"positions\" value "<<object <<endl;
poutdx<< "component \"connections\" value 2 " <<endl;
poutdx<< "component \"data\" value "<< object+1 <<endl<<endl;
poutdx.close();
}
```

Código 4 – Função membro de formatação do arquivo ".dx" em C++ (cont.)

```
void TMesh::DxFormatDataEnd(char* filedx, int object)
{
fstream poutdx;
int  i,k;
poutdx.open(filedx,ios::app); //Citar os membros para visualização
poutdx<<"object 980 class group"<<endl;
poutdx<<"member 0 value  4"<<endl;
if(object>6){
 for(i=1, k=7;k<object;k+=3,i++)
       poutdx<<"member "<<i <<" value "<<k<<endl;
       }
poutdx<<"end"<<endl;
poutdx.close();
}
```

Código 5 – Função membro de formatação do arquivo ".dx" em C++ (cont.)

```
# Definição da sub-malha regular 33x33 pontos,
#    com coordenadas iniciais (x_0,y_0)=(0,0) e  dx=dy=0,015625
object 1 class gridpositions counts 33 33
origin 0 0
delta  0.015625 0
delta  0  0.015625

# Atribuindo conexões para malhas regulares
object 2 class gridconnections counts 33 33
attribute "element type" string "quads"
attribute "ref" string "positions"

#Indicando o nome do arquivo de dados,
# o formato e o número de elementos esperados
object 3 class array type float rank 0 items 1089 data file P0.dat
attribute "dep" string "positions"

#Definindo o campo. Os valores 1, 2 e 3 estão associados
# aos objetos definidos anteriormente.
object 4 class field
component "positions" value 1
component "connections" value 2
component "data" value 3

# Iniciando um novo campo
object 5 class gridpositions counts 33 33
origin 0 0.5
delta 0.0078125 0
delta  0 0.0078125

object 6 class array type float rank 0 items 1089 data file P1.dat

#Observe que a conexão dos pontos, descrita no object 2, é mantida.
object 7 class field
component "positions" value 5
component "connections" value 2
component "data" value 6

#As demais sub-malhas são representadas da mesma maneira.
...
#Após todas as sub-malhas já estarem definidas,
#cria-se o grupo, em que cada value do membro  refere-se
#ao campo (field) definido anteriormente.

object 980 class group
member 0 value  4
member 1 value 7
member 2 value 10
member 3 value 13
```

Código 6 – Função membro de formatação do arquivo ".dx" em C++ (cont.)

Anexo

CIÊNCIAS EXATAS E DA TERRA - GEOCIÊNCIAS - METEOROLOGIA

VISUALIZAÇÕES DE RESULTADOS DE MODELOS ATMOSFÉRICOS DE PREVISÃO NÚMERICA DE TEMPO NO OPENDX.

Roberto Blaz[1]* blazz@cptec.inpe.br, Margarete Oliveira Domingues[2] (Orientador) margaret@cptec.inpe.br.

[1] Universidade Ibirapuera (UNIB), São Paulo/SP e Bolsista PIBIC-INPE/CNPq
[2] Laboratório Associado de Meteorologia e Oceanografia (LMO), Centro de Previsão do Tempo e Estudos Climáticos (CPTEC), INPE, São José dos Campos/SP , http://www.cptec.inpe.br

INTRODUÇÃO: Atualmente os centros de previsão de tempo utilizam complexos modelos numérico de tempo para auxílio às previsões de tempo e de clima. Tais modelos, em geral, possuem como saídas matrizes quadridimensionais, para diversas variáveis físicas, bastante extensas. Encontrar formas mais completas e integradoras para visualização dessas informações é uma tarefa desafiadora. Este trabalho apresenta algumas formas de visualizar esses dados na programa de computador OpenDX, baseado no DATA EXPLORER da IBM (http://www.opendx.org). Esse programa é uma ferramenta de visualização de dados poderosa, pois lida com uma representação multidimensional, permitindo o manuseio e animação dos dados. O OpenDX é constituido de módulos de programação. Algumas das suas vantagens imediatas, além do seu poder de trabalho, são ser multiplataforma e gratuito.

DADOS e METODOLOGIA: As informações utilizadas nestas visualizações são as variáveis atmosféricas quadridimensionais (x,y,z,tempo) provenientes do modelo de mesoescala MM5 (http://www.mmm.ucar.edu/mm5), tomadas para exemplificação. Esses dados estão originalmente no formato "binário" da ferramenta Grads (http://grads.iges.org/grads). A forma utilizada para que esses dados sejam introduzidos no OpenDX é convertê-los para o formato "netcdf" (http://www.unidata.ucar.edu/packages/netcdf). Esta tarefa é realizada por meio do script LATs4d do Grads (http://dao.gsfc.nasa.gov/software/grads/lats4d). Apesar do ambiente de entrada de dados do OpenDX aceitar dados no formato "netcdf", ele é incapaz de ler dados de diversas variáveis com mais de três dimensões diretamente nesse formato. Para isso é necessário desenvolver um programa de visualização desenvolvido no ambiente de programação visual do OpenDX, conhecido como VPE, de forma a definir como os dados devem ser lidos e como devem ser visualizados.

RESULTADOS: A forma de entrada dos dados, no formato "netcdf" utilizado, consiste em importar os dados diretamente com uma ferramenta do VPE chamada "import". A seguir, utilizam-se as ferramentas "slice" e "slab" para fatiar os dados em camadas de pressão atmosférica e escolher as fatias que se quer visualizar. Faz-se, então, uso da ferramenta "selector", para separar as variáveis atmosféricas, e da ferramenta "sequencer", para gerar uma animação temporal desses dados. Isso encerra a parte de entrada de dados e inicia-se o processo de desenvolver a visualização. A visualizações dos dados podem ser de diversos tipos: a)todas as variáveis atmosféricas ao mesmo tempo em uma respectiva superfície isobárica; b) uma variável nas suas diversas superfícies isobáricas; c) visualizar os dados em um determinado tempo ou em forma de animação; e d) acoplar outras ferramentas disponibilizadas pelo VPE (gradientes, mapas, topografia 3D, etc.). O OpenDX possíbilita criar um ambiente onde 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. Com este trabalho contribuiu-se também para o entendimento do programa e criação de documentação utilizando certas ferramentas do VPE. Essa reelaboração apoio-se nos manuais do DATA EXPLORER (http://www.research.ibm.com/dx/docs) e no livro "OpenDx Paths to Visualization" de D. Thompson, J. Braun e R. Ford, Visolution, de 2001.

CONCLUSÕES: O desenvolvimento da visualização de dados realizado com este trabalho abre perspectivas de utilização dessa ferramenta mais amplamente nos cotidianos acadêmico e de pesquisas nacionais. Tanto a representação de resultados numéricos como o manuseio de largas bases de dados ficam facilitados com os procedimentos organizados neste trabalho. Além de dispor de uma série de recursos e possibilitar a criação de outros, o ambiente VPE apresenta uma grande facilidade tanto na criação de visualizações quanto na velocidade com que isso pode ser implementado.

Agência Financiadora: PIBIC/CNPQ

TRABALHO DE INICIAÇÃO CIENTÍFICA

MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

CERTIFICADO DE PARTICIPAÇÃO

Certificamos que ***ROBERTO BLAZ*** *concluiu, com aproveitamento, o* ***"CURSO FORTRAN 77",*** *organizado e ministrado pelo INPE, no período de 19 a 26 de novembro de 2001.*

São José dos Campos, 17 de dezembro de 2001.

José Fernando Sanches da Silva
Chefe do Serviço de Treinamento e Desenvolvimento

José Eduardo Zaccarelli
Coordenador de Recursos Humanos

MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

INTRODUÇÃO A ENTRADA DE DADOS NO OPENDX: FORMATOS ".DX" , ".GENERAL" E ".GRB"

Roberto Blaz
Margarete Oliveira Domingues
Odim Mendes Jr

RELATÓRIO TÉCNICO

INPE
São José dos Campos
Maio de 2002

SUMÁRIO

LISTA DE FIGURAS

Pág.

CAPÍTULO 1

Introdução

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

Essa ferramenta de visualização possui uma forma padrão de entrada de dados os **".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 . No entanto, 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 para criar os formatos nativos do OPENDX e, assim, dispensar a entrada interativa DATA PROMPTER .

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

O texto foi estruturado de uma forma prática. No Capítulo 2, apresenta-se uma visão geral do OPENDX . No Capítulo 3, intruduz-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"** . O Capítulo 6, demonstra, com um caso específico para exemplo, o potencial de uso do programa OPENDX na visualização de dados de modelos meteorologicos de previsão de tempo.

CAPÍTULO 2

Uma visão geral do OpenDX

OPENDX é uma ferramenta de visualização muito poderosa que tem como finalidades principais a visualização de informações formatadas em 2D e 3D, de forma interativa ou automática - por meio de *scripts* ou de 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*. Uma das grandes vantagens desse recurso é sem dúvida a facilidade de programação em ambiente *visual*, conhecido pela sua sigla em inglês VPE. Essa ferramenta tem uma enorme gama de aplicações desde meteorologia, ciências em geral, até a industria. Na Figura 2.1 são apresentados alguns exemplos de visualizações de dados no OPENDX.

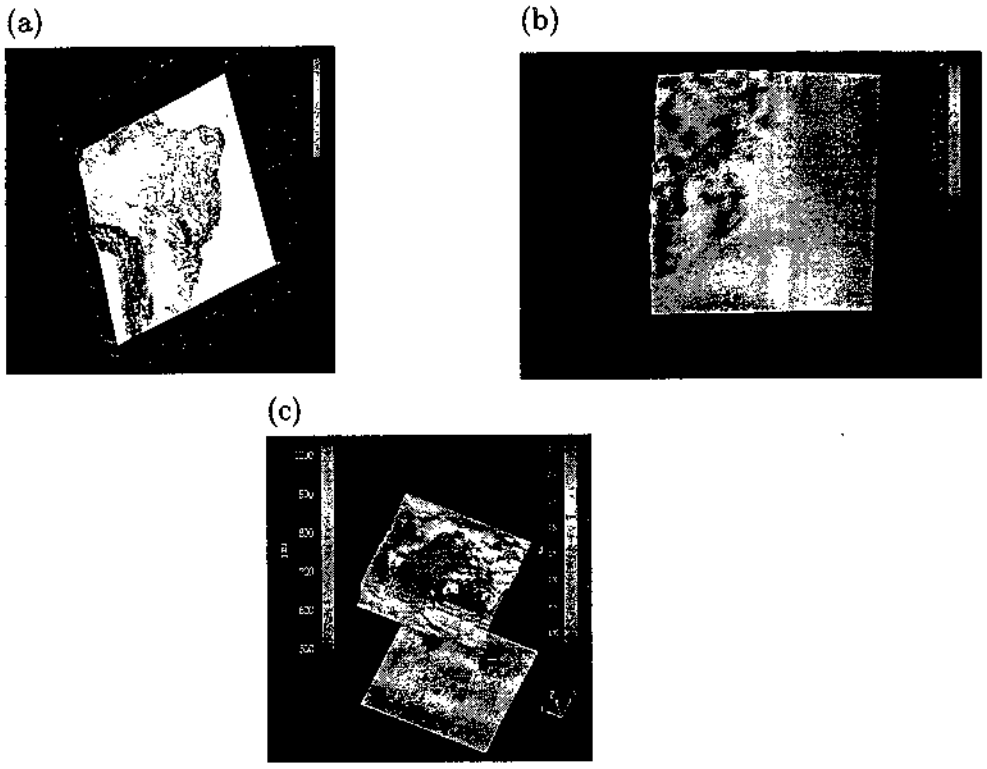

Fig. 2.1 – Exemplos de visualizações de dados no OPENDX:(a)Densidade de Descargas elétricas atmosfévicas;(b)Umidade relativa a superfície; (c)Pressão e Temperatura da superfície

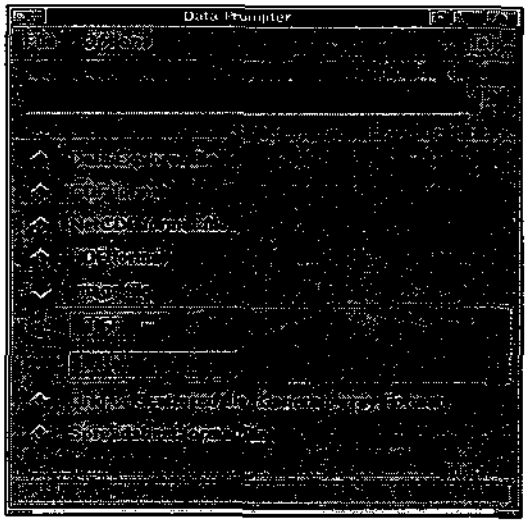

Fig. 2.3 – DATA PROMPTER

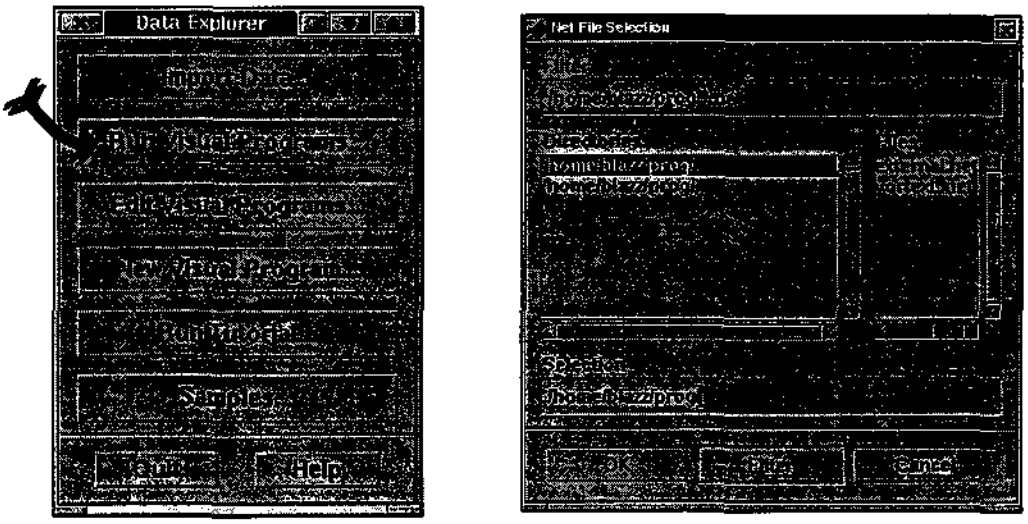

Fig. 2.4 – Execução de programas no OPENDX.

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.

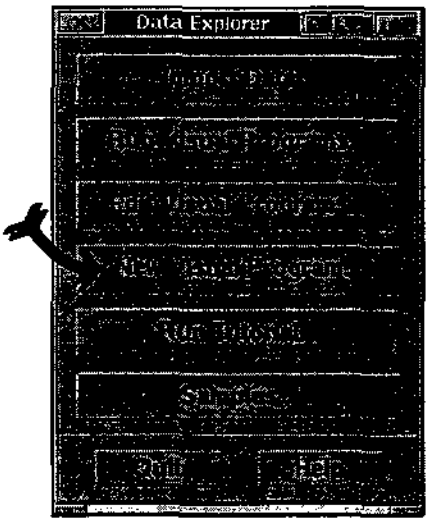

Fig. 2.7 – Exemplos de um Novo Programa Visual

(a)OPENDX (b)Tutorial

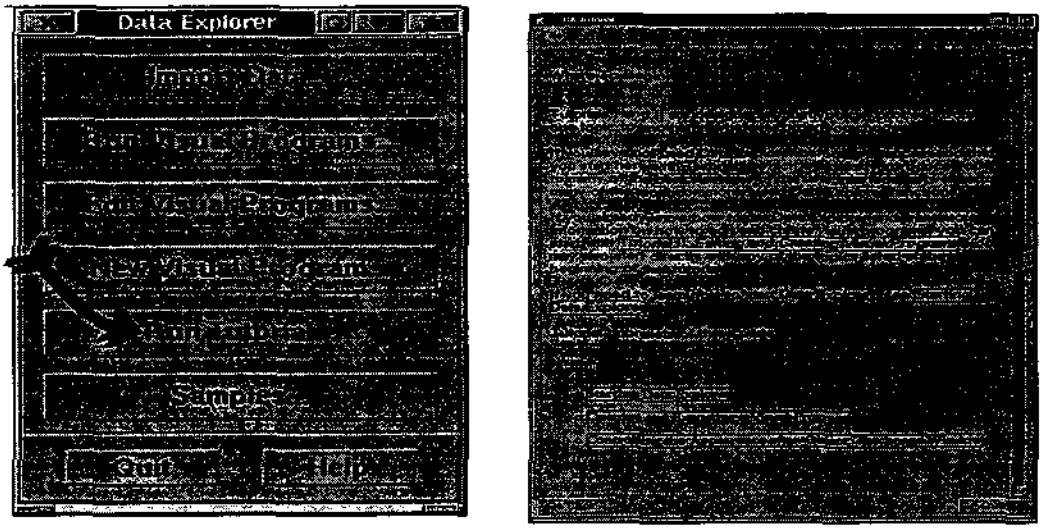

Fig. 2.8 – Tutorial do OPENDX.

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

O botão *Quit* finaliza o 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 importando esses dados de um arquivo de dados existente nos formatos: CDF, netCDF e HDF e em um formato geral de dados matriciais (**".general"**).

O OPENDX dispobiliza 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, a 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;
   - Arquivo no formato HDF;
   - Arquivo de imagem no formato TIFF, MIFF, GIF, RGB, R+G+B, YUV;
   - Arquivos com dados em grade regularmente espaçadas ou em grade irregularmente espaçadas (*scattered*)
   - Planilha de dados.

Os arquivos com dados em grade regularmente espaçadas ou em grade irregularmente espaçadas (*scattered*) e os em planilha de dados precisão ser descritos

colunas e se eles contém ou não um passo de tempo, e de quantas variáveis eles são compostos.

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

3) *Grade regular deformada*, onde em determinado ponto o espaçamento é regular; nesse tipo de grade, além das opções já descritas, as dimensões dos dados devem ser usadas .

4) *Grade dispersa de dados* é um conjunto de dados irregularmente espaçados.

Para que os dados possão ser importados corretamente mais algumas informações devem ser passadas ao DATA PROMPTER. Acionando o botão *Discribe 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 dados;
   - *Of Lines* (em linhas): Quantas linhas devem ser desprezadas até o início dados;
   - *String Marker* (Conjunto de caracteres): Qual palavra determina o início dados.

3) *Gride Size* e ♯ *of points*:

   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, os campos que não forem usados podem permanecer em branco.

   b) ♯ *of points* é usado quando os dados são compostos por pontos separados, no campo de texto que ativa-se ao entrar com o número de pontos.

10) *Grid positions*: Usado apenas se as posições de malha não estão especificados 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 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 $field_n$.
- *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 de 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*: Esta opção ativa somente quando as opções *Field interleaving, columnar,* e o *Data format, ASCII,* estão selecionados. Esta opção mostra

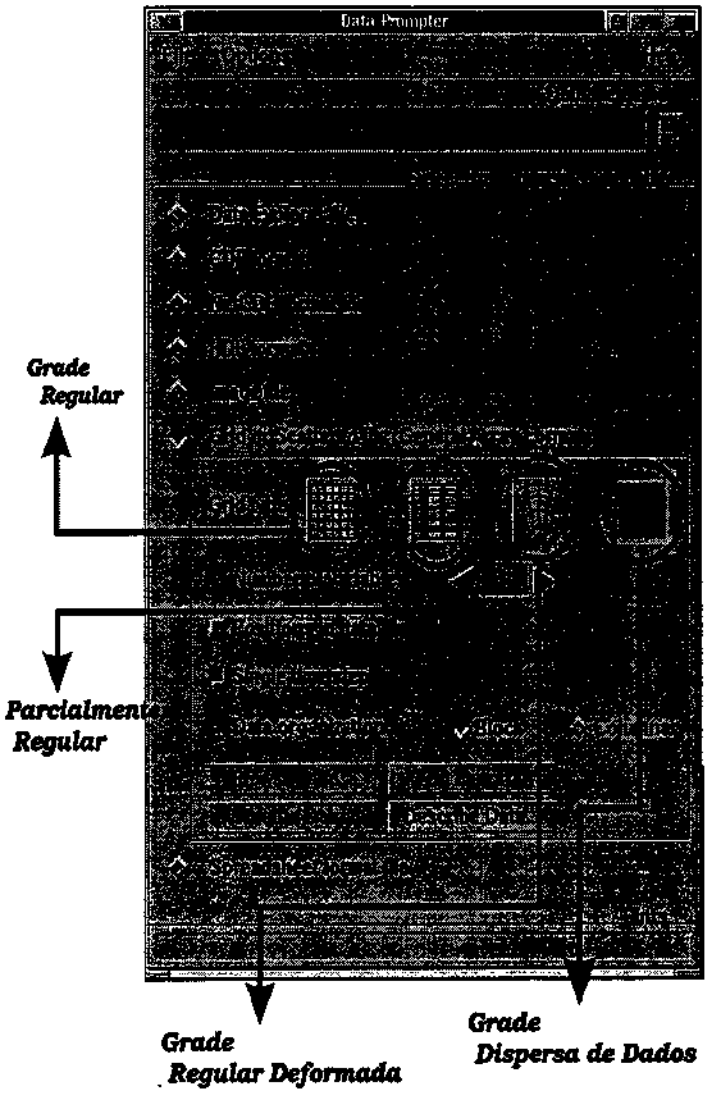

Fig. 3.3 – Janela gráfica para inclusão de dados em grades regulares e irregulares (*Grid or Scatterred File - General Array Format*).

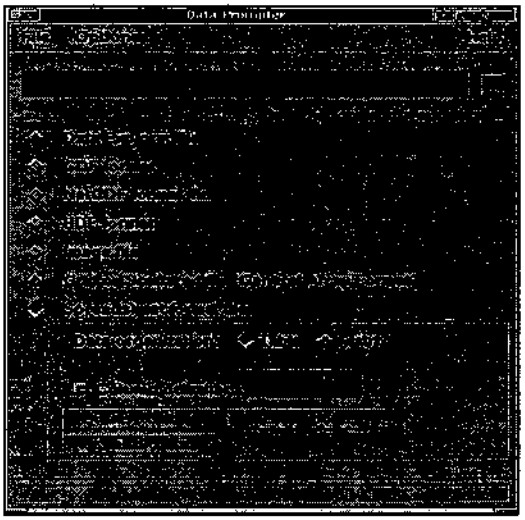

Fig. 3.5 – Janela gráfica para inclusão de dados no formato de planilha (*Spreadsheet Format File*).

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 independentes podem ser descritas em forma de uma grade regular ou uma lista.

b) Um outro aspecto a ser considerado é a dimenção das posições e componentes dos dados, e as conexões entre as posições.

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

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

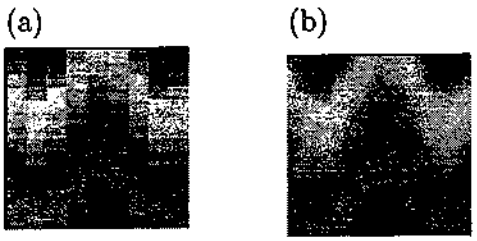

Fig. 4.1 – Dependência de dados a) Conexão e b) Posição.

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

f) 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

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.

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.

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

$$\left[\textbf{format} = \begin{bmatrix} msb \\ lsb \end{bmatrix} \begin{matrix} ascii \\ text \\ binary \\ ieee \end{matrix}\right]$$

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

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

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

$$\left[\textbf{majority} = \begin{matrix} row \\ column \end{matrix}\right]$$

$$\left[\textbf{recordseparator} = \begin{matrix} \textbf{bytes}\ \text{n} & \textbf{bytes}\ \text{n} \\ \textbf{lines}\ \text{n} & \textbf{lines}\ \text{n} \\ \textbf{marker}\,\textit{"string"} & \textbf{marker}\,\textit{"string"} \end{matrix}\right]$$

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

[**structure** = $structure_1, structure_2, \cdots, structure_f$]

[**type** = $type_1, type_2, \cdots, type_f$]

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

[*end*]

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* mudar a opção de *positions* para *connections* e confirme 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.

```
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

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 se importar somente os dados deve-se usar a uma opção para que o OPENDX pule as informações que não necessita-se ao ler esse arquivo, esse tipo de ação pode ser feita pulando-se *bytes*, linhas ou uma coleção de caracteres. Neste novo exemplo os primeiros passos são iguais aos anteriores e o nome do arquivo que 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 ser 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.

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, e 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_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

os passos para salvar o arquivo **".general"** , o Código 4.3.6 apresenta esse arquivo.

```
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*

Os 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*,

```
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

```
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 como 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

Seja os dados escalar múltiplos desejados dados com as seguintes caracteríticas uma matriz $4 \times 2 \times 3$ com *origin* [0 0 0] e *delta* [1 1 1] com estilo *Interliaving* e *record*. Deve-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 um arquivo ".general" como o apresentado no Código 4.3.10.

```
Umidade
    80          85          100         99          99          99
    95          91          82          95          91          98
```

```
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

Um exemplo é a grade 4 x 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 e em *Gride type* escolher *Scattered data*, em *Number of variables* selecionar o número 2, por exemplo. Aciona-se o botão *Positions in data file* e selecionar o número 2 novamente para *Dimension*. Repete-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.

CAPÍTULO 5

Formato de dados ".dx"

O formato **".dx"** de arquivo é flexível podendo-se 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 IBM (1997b,a); 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 sequência de definições de objeto, cada definição de objeto consiste de uma sequência 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 especifíico 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 seqüênciais 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 especifico de membros que são descritos pela palavra chave *member*, os

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 $(\Delta_x, \Delta_y, \Delta_z)$. Então, antes de apresentar-se 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 obtêm-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 dependentes de posição o *Object* do **".dx"** referente ao tipo de 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 deseja-se 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 agrupa-se os 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

```
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 interesse visualizar esses dados numéricos 4D: longitude, latitude, nível vertical e tempo para diversos campos atmosféricos, e 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 ".grb" > -o <arquivo ".netCDF" > -ftype ctl

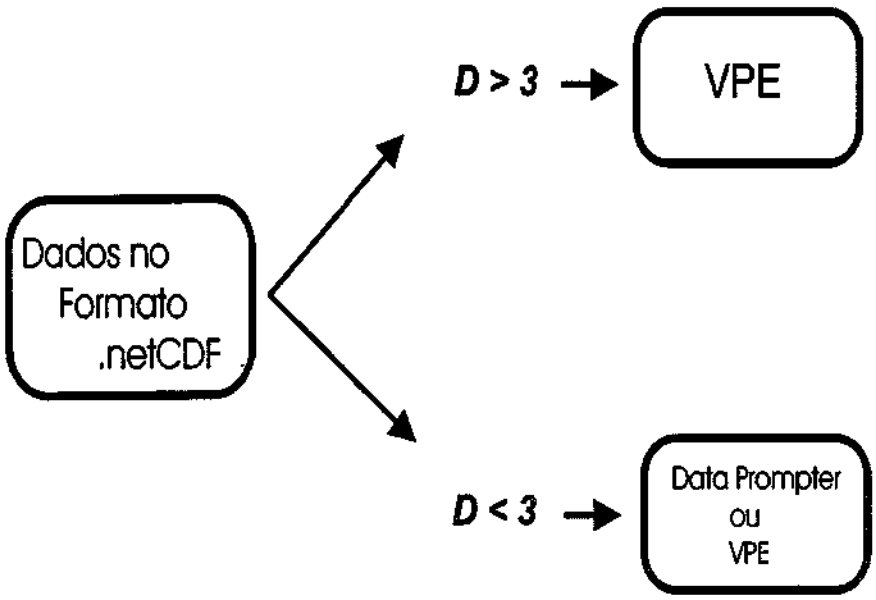

Fig. 6.1 – Esquema de importação de dados **".netCDF"**

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 eram 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 começa 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 em associado 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 transmitíl-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.

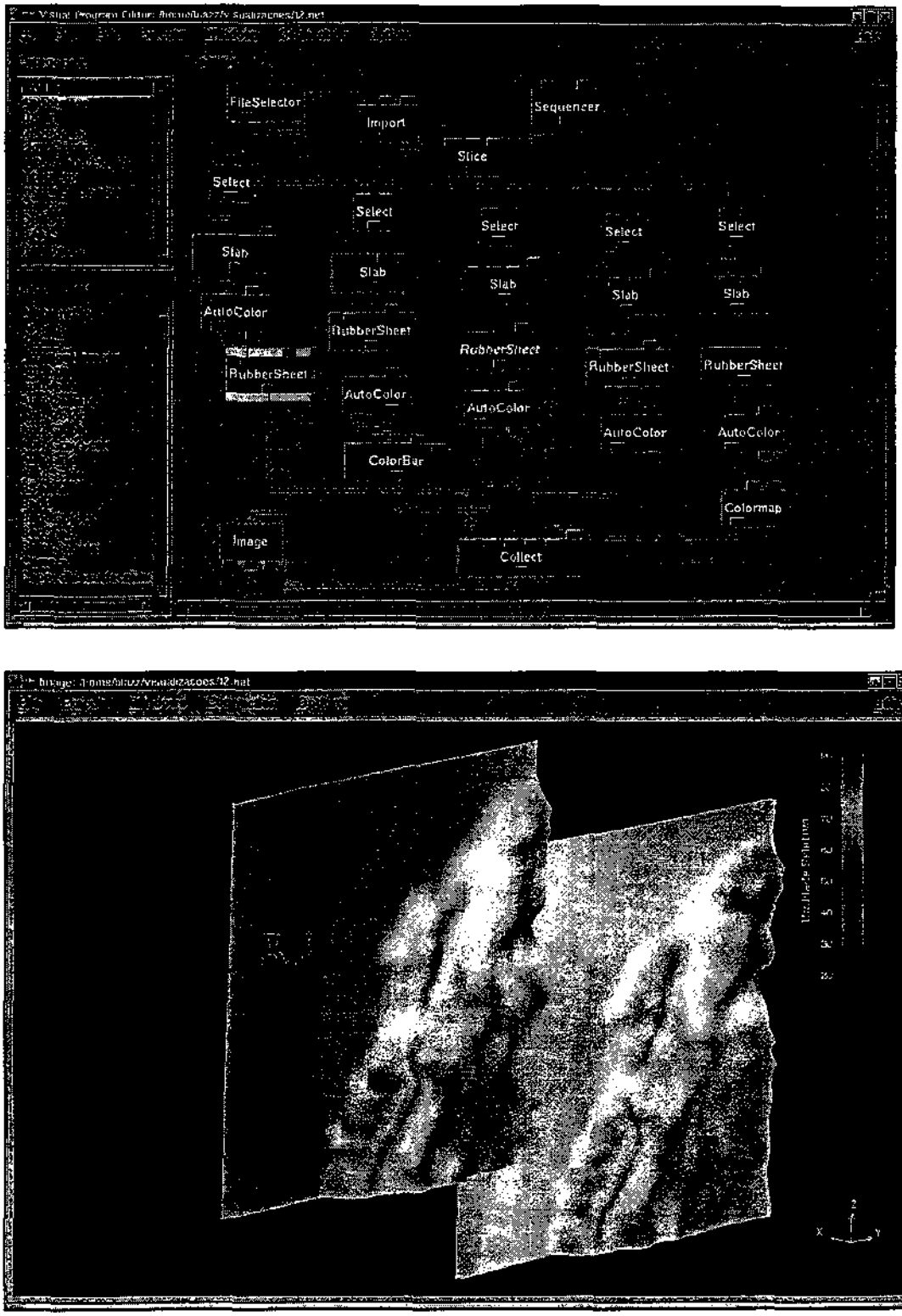

Fig. 6.4 – Exemplo das visualização do dado no formato **".netCDF"** 4D e seu programa de importação