DESENVOLVIMENTO DE UM PÓS-PROCESSADOR PARA VISUALIZAÇÃO DE RESULTADOS DE SIMULAÇÃO NUMÉRICA EM AQÜÍFEROS

JOSÉ EDUARDO QUARESMA

Dissertação apresentada à Escola de Engenharia de São Carlos, da Universidade de São Paulo, como parte dos requisitos para obtenção do título de Mestre em Hidráulica e Saneamento.

ORIENTADOR: Prof. Dr. Edson Cezar Wendland

São Carlos

2004

Ficha catalográfica preparada pela Seção de
Tratamento da Informação do Serviço de Biblioteca -
EESC/USP

Q1d Quaresma, José Eduardo
Desenvolvimento de um pós-processador para
visualização de resultados de simulação numérica em
aquéferos / José Eduardo Quaresma. -- São Carlos,
2004.

Dissertação (Mestrado) -- Escola de Engenharia de
São Carlos-Universidade de São Paulo, 2004.
Área: Hidráulica e Saneamento.
Orientador: Prof. Dr. Edson Cezar Wendland.

1. Aquífero. 2. Simulação. 3. Visualização
científica. 4. OpenDX. 5. Elementos finitos.
I. Título.

FOLHA DE JULGAMENTO

Candidato: Engenheiro **JOSÉ EDUARDO QUARESMA**

Dissertação defendida e julgada em 10-09-2004 perante a Comissão Julgadora:

Prof. Dr. **EDSON CEZAR WENDLAND** (Orientador)
(Escola de Engenharia de São Carlos/USP)

Aprovado

Prof. Dr. **JACKSON ROEHRIG**
(Universidade de Ciências Aplicadas de Colônia/Alemanha)

Aprovado

Prof. Titular **WOODROW NELSON LOPES ROMA**
(Escola de Engenharia de São Carlos/USP)

Aprovado

Profa. Titular **MARIA DO CARMO CALIJURI**
Coordenadora do Programa de Pós-Graduação em
Engenharia (Hidráulica e Saneamento) e
Presidente da Comissão de Pós-Graduação

“(...) Dizem que o homem inteligente é aquele que é capaz de fazer bem. E dizem que o homem sábio é aquele que é capaz de reunir vários homens inteligentes para fazer melhor. (...)”

Mario Covas Junior (1930 - 2001)

*Aos meus avós Luciano e Izaura, por me apoiarem
nos momentos mais difíceis.*

*À minha mãe Maria José (Ziza) por me ensinar
amar e respeitar o próximo.*

*Ao meu pai exemplo de trabalho e honestidade.
Meu verdadeiro amigo.*

*Aos meus tios Plínio (in memoriam) e Maria por es-
tarem sempre presentes em nossas tristezas e alegrias.*

*Às minhas adoráveis sobrinhas Júlia e Laís, minha
alegria.*

AGRADECIMENTOS

Ao meu estimado amigo e orientador **Prof. Dr. Ing. Edson Cezar Wendland**, pela orientação segura, eficiente e paciente. Por estar sempre presente nos momentos difíceis, incentivando e acreditando na realização deste trabalho. Meu sincero reconhecimento e gratidão.

Aos Professores Dr. Rodrigo de Mello Porto e Dr. Woodrow Nelson Lopes Roma, pela amizade, sugestões na elaboração e correções deste trabalho.

À Professora Dra. Luisa Fernanda Ribeiro Reis, por sua dedicação na orientação do Programa de Aperfeiçoamento de Ensino – PAE, muito obrigado.

Ao Professor MSc. Jorge Luiz Rabelo, meu amigo. Por nossa amizade e por me apoiar nessa difícil trajetória, muito obrigado.

Ao Gerente da CETESB (Companhia de Tecnologia de Saneamento Ambiental), Professor MSc. Engenheiro José Jorge Guimarães, que teve fundamental importância na minha formação como engenheiro civil e por me apoiar como pesquisador, meus sinceros agradecimentos.

Aos meus amigos do LHC (Laboratório de Hidráulica Computacional), principalmente o MSc. José Anderson Nascimento Batista (pelos conselhos, sugestões e correções), Luis, Rogério (Kid), Marcelo, Fábio (Kerk's), Leonel, Carlos, Karine, Júlio e o Thiago.

Ao amigo Prof. Dr. Jefferson Nascimento de Oliveira por ceder os dados do modelo de fluxo do Aquífero Bauru e por suas sugestões neste trabalho.

Aos amigos Gersinho e Jorginho (República Araraquara), Giovanni e a Josiele, Ivaltemir (Tinil), Peter, Luciana, Ana Rosa, Sandra, Botari, Eudes, Marcio Salla, Guilherme Lima, Marcos Fábio, Sérgio (Ceará), Daiana.

Ao Departamento de Hidráulica e Saneamento da Escola de Engenharia de São Carlos – USP, pela estrutura fornecida para desenvolvimento deste trabalho, aos professores e funcionários, especialmente às secretárias da pós-graduação Sá, Pavi e à recepcionista Rose, pela amizade, atenção e colaboração.

Aos Professores Flávio de Freitas Castilho, Antônio Carlos Matoso, Orlando e Alberto Mantese, Marcos Alberto Ferreira da Silva, Marcelo Augusto Tavares de Alcântara, diretores, colegas, professores e funcionários da FECA – Faculdade de Engenharia Civil de Araraquara, pelos ensinamentos e amizade.

À família De Marco, Dr. Gerson Joel De Marco e a Dona Rosa Olga Paschoal De Marco (*in memorian*), pelo apoio e amizade.

Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPQ) pela bolsa de estudo concedida.

À Nossa Senhora Aparecida, minha fé.

Finalmente, agradeço a todos que, direta ou indiretamente, às vezes sem sabê-lo, colaboraram para que este trabalho fosse realizado.

SUMÁRIO

1	INTRODUÇÃO	1
2	OBJETIVO	3
2.1	OBJETIVOS ESPECÍFICOS:	3
3	REVISÃO BIBLIOGRÁFICA	4
3.1	AVALIAÇÃO COMPARATIVA DE PROGRAMAS DISPONÍVEIS	4
3.1.1	<i>Conceitos sobre Visualização Científica</i>	4
3.1.2	<i>Processo de Visualização</i>	5
3.1.3	<i>Estrutura dos Sistemas de Visualização (Arquiteturas)</i>	7
3.1.4	<i>Bibliotecas para programação de aplicações</i>	9
3.1.5	<i>Sistemas “Turnkey” (Pacotes Comerciais)</i>	10
3.1.6	<i>Geradores de Aplicação (Application Builders)</i>	14
3.2	FERRAMENTAS UTILIZADAS PARA DESENVOLVIMENTO DO VPA	17
3.2.1	<i>Uma breve introdução sobre o sistema operacional GNU/Linux</i>	17
3.2.2	<i>O ambiente de programação ideal - Glade 2.0 (GTK+ 2)</i>	18
3.2.3	<i>O compilador GCC e o debugger GDB (DDD)</i>	20
3.2.4	<i>Ferramentas para desenvolvimento no GNU/Linux</i>	21
3.2.5	<i>Metodologia para Visualização de Processos em Aquíferos</i>	22
4	DESENVOLVIMENTO DO VPA	25
4.1	ARQUIVOS BINÁRIOS	25
4.2	DESENVOLVIMENTO DA INTERFACE DO VPA	27
4.3	TIPOS DE FORMATAÇÕES UTILIZADAS PARA ESCRITA DO “VPA.DX”	32
4.4	COMPONENTES DE CADA OBJETO DO <i>FIELD</i>	33
4.5	DESENVOLVIMENTO DOS <i>SCRIPTS (PIPELINES)</i>	34
5	VISUALIZAÇÃO E DISCUSSÃO DOS RESULTADOS (UTILIZAÇÃO DO VPA)	42
5.1	INTRODUÇÃO	43
5.2	REQUISITOS PARA INSTALAÇÃO DO SPA (COM VPA)	43

5.3	SISTEMA OPERACIONAL.....	44
5.4	INSTALAÇÃO DO VPA.....	44
5.5	DEPENDÊNCIAS DE BIBLIOTECAS PARA AMBIENTES KDE E GNOME	44
5.6	INSTALAÇÃO DO <i>OPENDX</i>	45
5.7	ARQUIVOS DE CONFIGURAÇÃO PARA O VPA	46
5.8	UTILIZAÇÃO DO VPA PARA VISUALIZAÇÃO DE PROCESSOS EM AQUÍFEROS	47
5.9	VISUALIZAÇÃO DAS ISOLINHAS	52
5.10	VISUALIZAÇÃO DA DIREÇÃO DOS GRADIENTES	53
5.11	VISUALIZAÇÃO DE GRADIENTES POR INTENSIDADE DE CORES RGB	54
5.12	VISUALIZAÇÃO DA MALHA.....	54
5.13	VISUALIZAÇÃO DE SUPERFÍCIES	55
6	CONCLUSÃO.....	63
7	REFERÊNCIAS BIBLIOGRÁFICAS.....	64
	ANEXO A	71
	ANEXO B.....	74
	ANEXO C	81

LISTA DE FIGURAS

FIGURA 1 – ANÁLISE CIENTÍFICA E <i>FEEDBACK</i> COM O AUXÍLIO DA VISUALIZAÇÃO.	6
FIGURA 2 – DIFERENTES TIPOS DE MALHAS.	23
FIGURA 3 – PROCESSO DE LEITURA E ESCRITA PARA VISUALIZAÇÃO DOS RESULTADOS.	25
FIGURA 4 – DESENVOLVIMENTO DO VPA COM AUXÍLIO DO AMBIENTE DE PROGRAMAÇÃO <i>GLADE 2</i>	28
FIGURA 5 – CRIAÇÃO DE UM “ <i>SIGNALS</i> ” PARA O OBJETO.	28
FIGURA 6 – CONFIGURAÇÃO DOS PARÂMETROS DO PROJETO.	30
FIGURA 7 – CONFIGURAÇÃO DOS PARÂMETROS, PARA GERAÇÃO DOS ARQUIVOS.	30
FIGURA 8 – <i>SCRIPT</i> (COM <i>PIPELINE</i>) PARA VISUALIZAÇÃO DO PROCESSO.	37
FIGURA 9 – <i>SCRIPT</i> (<i>PIPELINES</i>) PARA VISUALIZAÇÃO DAS ISOLINHAS.	37
FIGURA 10 – <i>SCRIPT</i> (<i>PIPELINES</i>) PARA VISUALIZAÇÃO DA DIREÇÃO DOS GRADIENTES.	38
FIGURA 11 – <i>SCRIPT</i> (<i>PIPELINES</i>) PARA VISUALIZAÇÃO DA INTENSIDADE DE GRADIENTES POR CORES RGB.	38
FIGURA 12 – <i>SCRIPT</i> (<i>PIPELINES</i>) PARA VISUALIZAÇÃO DA MALHA HÍBRIDA.	39
FIGURA 13 – <i>SCRIPT</i> (<i>PIPELINE</i>) PARA VISUALIZAÇÃO DE SUPERFÍCIES (MÓDULO 1).	39
FIGURA 14 – <i>SCRIPT</i> (<i>PIPELINE</i>) PARA VISUALIZAÇÃO DE SUPERFÍCIES (MÓDULO 2).	40
FIGURA 15 – <i>SCRIPT</i> (<i>PIPELINE</i>) PARA VISUALIZAÇÃO DE SUPERFÍCIES (MÓDULO 3).	40
FIGURA 16 – <i>SCRIPT</i> (<i>PIPELINE</i>) PARA VISUALIZAÇÃO DE SUPERFÍCIES (MÓDULO 4).	41
FIGURA 17 – <i>SCRIPT</i> (<i>PIPELINE</i>) PARA VISUALIZAÇÃO DE SUPERFÍCIES (MÓDULO 5).	41
FIGURA 18 – INTERFACE GRÁFICA DO VPA.	42
FIGURA 19 – DEPENDÊNCIAS DO EXECUTÁVEL DO VPA.	45
FIGURA 20 – INSTALAÇÃO DO <i>OPENDX</i> COM O AUXÍLIO DO GERENCIADOR DO <i>YAST2</i>	46
FIGURA 21 – EXECUÇÃO DO VPA EM UM TERMINAL <i>XTERM</i>	47
FIGURA 22 – INTERFACE DO VPA COM OS RESULTADOS DA SIMULAÇÃO.	48
FIGURA 23 – SELEÇÃO DO ITEM PARA VISUALIZAÇÃO DO PARÂMETRO.	49
FIGURA 24 – MAPEAMENTO DE VALORES ESCALARES POR MEIO DE UMA TABELA DE CORES.	50
FIGURA 25 – CONDUTIVIDADE HIDRÁULICA DO MODELO DE ESCOAMENTO DO AQUÍFERO BAURU.	51
FIGURA 26 – CARGA HIDRÁULICA DO MODELO DE ESCOAMENTO DO AQUÍFERO BAURU.	51
FIGURA 27 – SELEÇÃO DO ITEM PARA VISUALIZAÇÃO DAS ISOLINHAS, CONTORNO E POÇOS.	52

FIGURA 28 – VISUALIZAÇÃO DAS ISOLINHAS DE CARGA HIDRÁULICA DO AQUÍFERO BAURU, NA CIDADE DE SÃO JOSÉ DO RIO PRETO.	52
FIGURA 29 – SELEÇÃO DO ITEM PARA VISUALIZAÇÃO DA DIREÇÃO DOS GRADIENTES.	53
FIGURA 30 – VISUALIZAÇÃO DA DIREÇÃO DOS GRADIENTES DA CARGA HIDRÁULICA DO AQUÍFERO BAURU, NA CIDADE DE SÃO JOSÉ DO RIO PRETO.	53
FIGURA 31 – SELEÇÃO DO ITEM PARA VISUALIZAÇÃO DA INTENSIDADE DOS GRADIENTES.	54
FIGURA 32 – SELEÇÃO DO ITEM PARA VISUALIZAÇÃO DA MALHA.	54
FIGURA 33 – VISUALIZAÇÃO DOS GRADIENTES DE CARGA HIDRÁULICA, POR INTENSIDADE DE CORES RGB.	55
FIGURA 34 – VISUALIZAÇÃO DA SUPERFÍCIE PIEZOMÉTRICA.	56
FIGURA 35 – MALHA DO MODELO DE ESCOAMENTO DO AQUÍFERO BAURU.	56
FIGURA 36 – ZOOM DOS ELEMENTOS DA MALHA (TRIÂNGULOS E QUADRILÁTEROS).	57
FIGURA 37 – MENU PARA ESCOLHA DO “VIEW CONTROL”.	58
FIGURA 38 – JANELA DO “VIEW CONTROL”.	58
FIGURA 39 – ROTAÇÃO DA SUPERFÍCIE PARA VISUALIZAÇÃO DOS CONES DE DEPRESSÃO.	58
FIGURA 40 – VISUALIZAÇÃO DA REGIÃO DE MAIS CONCENTRAÇÃO DE POÇOS.	59
FIGURA 41 – VISUALIZAÇÃO DA ÁREA ONDE ESTÃO CONCENTRADOS A MAIORIA DOS POÇOS.	60
FIGURA 42 – JANELA PARA CONFIGURAÇÃO DOS EIXOS (“AUTOAXES”).	61
FIGURA 43 – JANELA INICIAL DO <i>OPENDX</i>	75
FIGURA 44 – JANELA PARA ABERTURA DO ARQUIVO “VPA.DX”.	76
FIGURA 45 – JANELA PARA SELEÇÃO DO ARQUIVO “VPA.DX”.	76
FIGURA 46 – VISUALIZAÇÃO DA SUPERFÍCIE DO TERRENO.	77
FIGURA 47 – MENU PARA ESCOLHA DO “VIEW CONTROL”.	78
FIGURA 48 – JANELA DO “VIEW CONTROL”.	78
FIGURA 49 – JANELA PARA CONFIGURAÇÃO DO EIXOS (“AUTOAXES”).	80

LISTA DE ABREVIATURAS E SIGLAS

API	-	<i>Application Programming Interface</i>
DDD	-	<i>Data Display Debugger</i>
GCC	-	<i>GNU Compiler Collection</i>
GDB	-	<i>GNU Debugger</i>
GIMP	-	<i>GNU Image Manipulation Program</i>
GKS	-	<i>Graphical Kernel System</i>
GNOME	-	<i>GNU Network Object Model Environment</i>
GNU	-	<i>GNU is not Unix</i>
GPL	-	<i>General Public License</i>
GUI	-	<i>Graphic User Interface</i>
IDE	-	<i>Integrated Development Environmental</i>
KDE	-	<i>The K Desktop Environmental</i>
PHIGS	-	<i>Programmer's Hierarchical Interactive Graphics System</i>
POO	-	<i>Programação Orientada a Objetos</i>
RAD	-	<i>Rapid Application Developer</i>
RPM	-	<i>Red Hat Package Manager</i>
SPA	-	<i>Simulação de Processos em Aquíferos</i>
ViSC	-	<i>Visualization in Scientific Computing</i>
VPA	-	<i>Visualizador de Processos em Aquíferos</i>
XML	-	<i>Extensible Markup Language</i>

Resumo

QUARESMA, J. E. (2004). Desenvolvimento de Um Pós-Processador para Visualização de Resultados de Simulação Numérica em Aquíferos. São Carlos, 2004. 112 p. Dissertação (Mestrado) – Escola de Engenharia de São Carlos, Universidade de São Paulo.

Este trabalho apresenta o desenvolvimento de um programa visualizador de processos em aquíferos intitulado VPA, como módulo integrante de um pacote de programas computacionais aplicados ao gerenciamento de recursos hídricos subterrâneos, denominado SPA (Simulação de Processos em Aquíferos). Esse visualizador foi desenvolvido para sistema operacional *SuSE Linux* (Wall et al., 1999), utilizando como ambientes gráficos o *KDE* e o *GNOME*.

O ambiente de programação *Glade 2* foi escolhido para programação e desenvolvimento do visualizador. Este ambiente de programação em C é distribuído nos âmbitos de *open source development* (*GNU*), e serve como complemento da interface gráfica com o usuário *GUI* (*Graphic User Interface*). Em conjunto com esta linguagem de programação foi usado o *Data Explorer IBM OpenDX*, distribuído também nos termos de *open source development*, como ferramenta para visualização de simulações em duas e três dimensões.

Assim, é possível visualizar os resultados gerados pela simulação de processos de escoamento de fluidos e transporte de poluentes no subsolo, por meio dos módulos de pré e pós-processamento do pacote SPA, na forma de dados binários. O módulo desenvolvido representa uma alternativa robusta e econômica para a visualização científica de processos em aquíferos.

Palavras Chave: (1) Aquífero. (2) Simulação. (3) Visualização Científica. (4) *OpenDX*. (5) Elementos Finitos.

Abstract

QUARESMA, J. E. (2004). Post-Processor Development for Numerical Simulation Result Visualization in Aquifers. São Carlos, 2004. 112 p. Dissertação (Mestrado) – Escola de Engenharia de São Carlos, Universidade de São Paulo.

In this work the development of an aquifer processes viewer program called VPA is presented. The program is integrated as a module of a computational package devoted to groundwater resources management named SPA (Simulation for Processes in Aquifers). The program was developed on *SuSE Linux* operational system (Wall *et al.*, 1999) utilizing *KDE* and *GNOME* graphical environments.

The graphic templates for the VPA graphical user interface (GUI) were taken from *Glade 2*. That graphical library is written in C programming language and is distributed under the *open source development* license (GNU). Scientific visualization routines from *Data Explorer IBM OpenDX*, also distributed under GNU license, were gathered to VPA giving to it capability to generate two- and three-dimensional graphics to numerical results.

Thus, VPA constitutes the post-processing tool for reading numerical results from binary files and constructing graphics for fluid flow and pollutant transport simulation in groundwater in the SPA environment. Such module shows itself an economic and robust alternative in scientific visualization to aquifer processes.

Keywords: (1) Aquifer. (2) Simulation. (3) Scientific Visualization. (4) *OpenDX*. (5) Finite Elements.

1 Introdução

O aproveitamento das águas subterrâneas tem aumentado significativamente nos últimos anos, não só devido à demanda, mas também em consequência da crescente degradação das águas superficiais. As águas subterrâneas constituem cerca de 30% da água doce disponível em nosso planeta (GEF, 2002). Atualmente existem cidades que são totalmente abastecidas por águas subterrâneas, e que começam a enfrentar problemas como a superexploração de aquíferos (perfuração excessiva de poços sem controle, muito próximos uns dos outros) e a contaminação das águas subterrâneas (GEF, 2003).

O alto custo de monitoramento de processos em aquíferos, tem levado ao uso freqüente de ferramentas computacionais, como forma principal de quantificar e prever os referidos processos. Neste contexto, as simulações numéricas, tornam-se uma ferramenta necessária e imprescindível na tomada de decisões relativas ao gerenciamento dos recursos hídricos subterrâneos. Com este propósito foi desenvolvido o pacote SPA (Simulação de Processos em Aquíferos). O pacote SPA (Wendland, 2003) consiste em um programa computacional baseado no método de elementos finitos destinado à simulação numérica de escoamento e transporte de solutos tridimensional, em meios porosos heterogêneos com aplicação prática na área de recursos hídricos subterrâneos. Este pacote encontra-se em desenvolvimento, e dispõe de recursos de pré e pós-processamento.

O pacote SPA foi utilizado para simulação do modelo de escoamento para o Aquífero Bauru, situado na área urbana da cidade de São José do Rio Preto (Oliveira, 2002). Esta área urbana é abastecida por poços tubulares que captam água do Aquífero Bauru. Na região encontram-se mais de 400 poços produtores cadastrados e cerca de 3.500 poços clandestinos, que retiram aproximadamente 5.000 m³/h, com base nessas informações e em dados climáticos regionais, foi construído um modelo numérico que permitiu simular toda a complexidade hidrogeológica da área urbana. Com os dados obtidos na simulação, foram feitos prognósticos para dois cenários distintos: o primeiro com diminuição da precipitação e o segundo com o aumento da área urbanizada e conseqüente impermeabilização. As inferências dessas duas hipóteses na dinâmica foram

avaliadas, sendo possível observar um colapso do sistema de captação, na região central da cidade, caso um dos prognósticos venha a ocorrer (Oliveira, 2003). Para visualização não somente desses resultados, foi desenvolvido o VPA (Visualização de Processos em Aquíferos).

O objetivo deste trabalho foi o desenvolvimento do programa VPA, como uma ferramenta para visualização de processos em aquíferos. Assim, foram descritos vários itens, cada um com sua característica: conceitos sobre visualização científica; o desenvolvimento do VPA (destacando as ferramentas utilizadas para desenvolvimento em ambiente *GNU/Linux* (Mitchell et al., 2001); a integração com SPA para leitura dos arquivos binários gerados na simulação; os programas utilizados pelo VPA, para visualização dos arquivos “.dx”; as principais ferramentas do *OpenDX* (Earnshaw, 1993; Thompson, 2001) e as visualizações dos resultados das simulações, para análise do modelo de escoamento do Aquífero Bauru.

Como resultado deste trabalho, são apresentadas diversas visualizações, servindo como ferramenta de análise para gerenciamento de recursos hídricos subterrâneos, incrementando o conhecimento das condições do modelo de escoamento para o Aquífero Bauru, na cidade de São José do Rio Preto. Assim, este programa caracteriza-se como uma ferramenta robusta para análises de resultados numéricos gerados pelo SPA.

2 Objetivo

O objetivo deste trabalho foi o desenvolvimento de um programa para visualização de processos em aquíferos (VPA), visando a análise dos resultados de simulação por elementos finitos, para o gerenciamento de recursos hídricos subterrâneos.

2.1 Objetivos específicos:

- Avaliação comparativa das principais técnicas de visualização científica e ferramentas disponíveis para desenvolvimento do VPA;
- Desenvolvimento do VPA em ambiente *SuSE Linux* (Purcell, 1997), com o auxílio de ferramentas *open source* para programação (uso do ambiente de programação *Glade 2*);
- Integração do SPA com VPA, por meio das bibliotecas *DtnLIB*, para leitura dos arquivos binários gerados pelo processamento da simulação;
- Desenvolvimento de técnicas para visualização, com o auxílio das ferramentas do *OpenDX*, possibilitando o desenvolvimento de *scripts*, para os processos de visualização, baseados em *networks* e *pipelines*.
- Elaboração do manual do VPA;
- Visualização dos resultados da simulação do modelo de escoamento do aquífero Bauru, na cidade de São José do Rio Preto.

3 Revisão Bibliográfica

3.1 Avaliação comparativa de programas disponíveis

3.1.1 Conceitos sobre Visualização Científica

A Visualização Científica existe há muito tempo, mas nem todos computadores tinham capacidade de gerar imagens bem definidas, pois a grande maioria se valia de caracteres com formas gráficas rudimentares para poder apresentar uma imagem não textual no vídeo (Alves, 2002). As aplicações de técnicas gráficas para ampliar a capacidade de interpretação de dados científicos têm sido denominadas Visualização Científica, ou Visualização em Computação (*ViSC – Visualization in Scientific Computing*). Além das áreas de ciências, a tendência de usar gráficos para interpretar informação complexa alastrou-se para inúmeras outras atividades profissionais, acadêmicas e de pesquisa.

Existem várias definições de Visualização. No seu sentido mais amplo, a palavra significa a geração de imagens mentais para organização e entendimento de um conceito, idéia ou informação. Formulações alternativas relacionam essa definição a um contexto computacional. Uma definição típica, que trata especificamente de Visualização Científica, é: “Visualização é o processo de transformar informação para uma forma visual, permitindo aos usuários observar a informação. A apresentação visual resultante permite ao cientista ou engenheiro perceber visualmente características que estão escondidas nos dados, mas que são necessárias para tarefas de exploração e análise” (Gershon, 1994). A Visualização científica estuda estratégias e algoritmos para mapear informações numéricas em representações gráficas. É importante ressaltar que o objetivo da Visualização não é entender os dados por si só, mas o fenômeno que os gerou (Minghim & Oliveira, 1997). Uma técnica de visualização envolve os seguintes passos: 1) a construção de um modelo a partir dos dados; 2) a seleção de algum mecanismo de mapeamento do modelo em um objeto de visualização abstrato, como uma imagem ou um

mapa de contornos e 3) a renderização da imagem no dispositivo de exibição gráfico. A visualização envolve, portanto, a exploração, transformação e mapeamento de dados em objetos gráficos. Os dados em ViSC podem estar definidos sobre domínios bidimensionais (2D), tridimensionais (3D) e, genericamente, multidimensionais. Em geral, eles estão organizados em uma malha (regular ou irregular), mas podem se constituir de um conjunto de pontos esparsos. As informações associadas a cada ponto podem ser um ou mais valores escalares, vetoriais ou tensoriais, e os dados podem ser constantes ou variantes no tempo. Atualmente existe disponível uma grande variedade de técnicas de visualização (Friedhoff, 1991; Keller, 1993; Roseblum, 1994; Schöeder, 1996), mas ainda não existe uma classificação abrangente que englobe todas as técnicas situando-as em uma estrutura bem definida. Numa tentativa de situar as principais técnicas em uma estrutura genérica, foi introduzida uma notação concisa, apresentada por Brodlie, que captura as principais características de um conjunto de dados. Exemplos da notação de Brodlie são apresentados no item 5.1 (Classificação de Dados para Visualização). É possível, no entanto, encontrar outras notações propostas na literatura (Bergeron, 1989; Gallop, 1994).

3.1.2 Processo de Visualização

Os sistemas de visualização devem ser usados como parte integrante do processo de investigação científica. Assim, o processo de investigação requer a criação de um modelo empírico da “realidade”, a aquisição de dados que sirvam como subsídios para verificar ou refinar o modelo e o uso desse modelo para fazer previsões. No ciclo de investigação o cientista usa o seu discernimento e conhecimento (*insight*) sobre o fenômeno para formular uma hipótese, coleta de dados, controlar um processo de computação, visualizar e interpretar os resultados e, em seguida, repetir e refinar o ciclo, como ilustrado na Figura 1.

Os sistemas de visualização também devem ser caracterizados por um procedimento interativo iniciado pelo usuário, que continua o processo de investigação à medida que recebe *feedback* do sistema. A interação realizada pelo usuário, através da interface ao qual é apresentado, é transformada logicamente de volta aos termos do fenômeno científico sob investigação, e resultados atualizados são gerados pelo sistema de visualização e apresentados ao usuário. Dessa forma, o usuário aumenta o seu conheci-

mento sobre a validade da hipótese originalmente formulada e sobre o modelo adotado para a sua investigação.

Existem duas maneiras pelas quais os dados podem ser obtidos: medições experimentais de dados ou simulações numéricas. Esses dados estão comumente distribuídos em pontos discretos. Esse fato origina um dos problemas mais relevantes em visualização que é determinar valores dos dados em pontos arbitrários. Outro problema típico do processo de visualização é que os dados de entrada geralmente contêm erros aleatórios introduzidos pelo próprio processo de aquisição.

Figura 1 - Análise científica e *feedback* com o auxílio da visualização.

Dessa forma, os dados precisam ser “filtrados” para que o erro seja removido podendo, por outro lado a filtragem, “distorcê-los”, removendo características de interesse ou incluindo informações espúrias.

Problemas relacionados à amostragem e à interpolação devem ser considerados no contexto do processo de investigação, pois são limitações a serem tratadas também nas técnicas e sistemas de visualização. É importante ainda ter em mente que esses sistemas trabalham com modelos computacionais dos fenômenos em estudo, e não com modelos matemáticos ideais (Minghim & Oliveira, 1997).

3.1.3 Estrutura dos Sistemas de Visualização (Arquiteturas)

A Visualização Científica é uma área que atualmente conta com um grande número de pesquisadores no mundo todo. A razão para esse desenvolvimento é bastante desafiadora: existe mais informação disponível para ser interpretada do que tempo hábil para fazê-lo.

O aumento do poder computacional e conseqüente desenvolvimento das técnicas numéricas (método dos elementos finitos, diferenças finitas, etc.), fizeram com que muitas áreas de pesquisa necessitassem de ferramentas gráficas para auxílio e interpretação dos dados gerados. A conversão dos dados simulados e das informações, para um formato gráfico, é denominada visualização. Essa conversão é essencial para a análise e interpretação dos resultados obtidos. A conseqüência dessas representações é a análise dos dados de forma mais clara, o que aumenta e agiliza a produtividade. Assim a visualização computacional, chamada Computação Gráfica (McCormick et al., 1987) tem merecido destaque dentro dessas atividades.

A maioria dos sistemas gráficos trabalham com bibliotecas gráficas básicas, já estabelecidas, como a *OpenGL* e suas implementações, *PEX*, *DORE* e *HOOPS*. Também estão incluídos nos sistemas gráficos, os componentes de interface final com o usuário, a base de dados gráficos, o gerenciamento da visualização e a interface de programação. Padrões gráficos, como o *GKS (Graphical Kernel System)* (ANSI, 1985a) e o *PHIGS (Programmer's Hierarchical Interactive Graphics System)* (ANSI, 1985b), tiveram importante papel na década de 80, inclusive ajudando a estabelecer o conceito do uso de padrões mesmo fora da área gráfica, tendo sido implementado em diversas plataformas. Nenhuma destas *APIs (Application Programming Interface)*, no entanto, conseguiu ter grande aceitação (Segal, 1994). A interface destinada a aplicações gráficas 2D e 3D deve satisfazer diversos critérios como, por exemplo, ser implementável em plataformas com capacidades distintas sem comprometer o desempenho gráfico do hardware e sem sacrificar o controle sobre as operações de hardware (Segal, 1996).

As principais diferenças entre eles são os recursos para configuração do sistema, da estrutura e da interface. A estrutura de um sistema de visualização inclui componentes de interface, de gerenciamento de dados para técnicas de visualização, controle de escoamento e de funções gráficas. As principais classes de sistemas, de acordo com a organização desses componentes, são:

- Sistemas “*turnkey*”: também denominados sistemas tipo “usuário-final”, possuem todos os requisitos para que o usuário gere visualizações sem necessidade de programação. São normalmente, rápidos, fáceis de usar. Devido à falta de recursos de programação, não é possível reconfigurar esses sistemas para uma visualização específica, sendo que é difícil projetá-los com um grau de generalidade suficiente para atender a um grande número de aplicações.

- Sistemas para Simulação e Visualização de Processos em Aquíferos: ao contrário dos outros sistemas para visualização são considerados como sistemas específicos e do tipo “usuário-final”; possuem módulos para geração de visualização de processos em aquíferos. Geralmente esse módulo possui todos os recursos necessários para análise dos resultados.

- Geradores de aplicação (*application builders*) com uso de programação visual: nesse tipo de arquitetura (Gallagher, 1995), o usuário tem acesso a ferramentas interativas para construir um sistema de visualização próprio. Os módulos de visualização, leitura de dados e apresentação gráfica possuem uma representação visual 2D ou 3D, que pode ser recuperada e conectada a demais elementos de visualização. A saída de um módulo pode ser conectada à entrada de outro ou vários outros (*pipelines*), formando uma rede de conexões. Essa interface é bastante potente, permitindo rápida configuração do sistema para uso em aplicações específicas. Tais sistemas são bastante difíceis de implementar, especialmente se os diferentes módulos puderem ser executados em máquinas diferentes.

- Geradores de aplicação programáveis por linguagem: um gerador de aplicação também pode ser programado por linguagem procedimental, ao invés de graficamente. Sistemas desse tipo são bastante flexíveis, se a linguagem utilizada, normalmente interpretada, inclui comandos convencionais de linguagens de programação de propósito geral. As linguagens procedimentais permitem, via de regra, um maior domínio do controle de escoamento do sistema.

O que diferencia as diversas estruturas no que diz respeito à generalidade, poder de descrição e flexibilidade. Do ponto de vista do usuário final, entretanto, um aspecto

muito importante é a forma de comunicação com os diversos módulos do sistema, bem como com os dados introduzidos e os gráficos gerados.

3.1.4 Bibliotecas para programação de aplicações

As bibliotecas representam conjuntos de rotinas na forma de *APIs* que permitem ao usuário introduzir componentes de visualização nos seus programas. Alguns exemplos são:

- *Visual3 (MIT)*: permite acesso, por meio de rotinas escritas em FORTRAN, à maioria das principais funções gráficas de visualização e de exploração de dados volumétricos.
- *CardinalVision (Wilsonville, EUA)*: fornece uma biblioteca de objetos escritos em C++ para executar funções de visualização de dados escalares, vetoriais e tensoriais associados a malhas de elementos finitos.
- *FOCUS (Visual Kinematics, EUA)*: possui um conjunto de rotinas para várias técnicas gráficas e de visualização de sólidos, fluidos e outros dados.
- *VTK (Visualization Toolkit)*: é uma biblioteca completa (Schröder, 1996), disponível a baixo custo e com código fonte, implementada para sistemas *Unix*, *Linux*, *FreeBSD* (Kochan, 2003) e também Windows®, possibilitando a programação na linguagem *Tcl/Tk* (Ousterhout, 1994). Também pode ser classificado como um *Application Builder* (gerador de aplicação).
- *OpenGL (Silicon Graphics)*: É uma biblioteca gráfica de modelagem e exibição tridimensional, bastante rápida e portátil para vários sistemas operacionais. Seus recursos permitem ao usuário criar objetos gráficos com qualidade próxima à de um *raytracer*¹, de modo mais rápido que este último, além de incluir recursos avançados

¹ *Raytracer* é um programa que gera imagens realistas de uma cena geométrica. Os objetos são representados fielmente, com reflexões, transparências e sombras que caracterizam uma cena real. O seu princípio é bastante simples, mas o resultado que produz é extraordinário. De modo geral, os *raytracers* são lentos. Isto porque o número de

de animação, tratamento de imagens e texturas. Diante das funcionalidades providas pela *OpenGL*, ela tem se tornado um padrão amplamente adotado na indústria de desenvolvimento de aplicações na área de visualização científica (Neider, 1993). Este fato tem sido encorajado também pela facilidade de aprendizado, estabilidade das rotinas e pela boa documentação disponível (Neider, 2000) e pelos resultados visuais consistentes para qualquer sistema de exibição concordante com este padrão. É também uma interface que disponibiliza um controle simples e direto sobre um conjunto de rotinas, permitindo ao programador especificar os objetos e as operações necessárias para a produção de imagens gráficas de alta qualidade. Para tanto, a *OpenGL* funciona como uma máquina de estados, onde o controle de vários atributos é realizado por meio de um conjunto de variáveis de estado que inicialmente possuem valores *default*, podendo ser alterados caso seja necessário. Por exemplo, todo objeto será traçado com a mesma cor até que seja definido um novo valor para esta variável. Por ser um padrão destinado somente à renderização² (Segal, 1994), a *OpenGL* pode ser utilizada em qualquer sistema de janelas, aproveitando-se dos recursos disponibilizados pelos diversos hardwares gráficos existentes. Este padrão é o sucessor da biblioteca gráfica conhecida como *IRIS GL*, desenvolvida pela *Silicon Graphics Inc.*, como uma interface gráfica independente de *hardware* (Kilgard, 1994). Na mesma época, foi formado o *OpenGL Architecture Review Board* (Kilgard, 1996), um consórcio independente que administra o uso da *OpenGL*, formado por diversas empresas da área gráfica.

3.1.5 Sistemas “*Turnkey*” (Pacotes Comerciais)

Em sistemas desse tipo não há necessidade de programação. Aqui se incluem sistemas de propósito geral e sistemas específicos para um domínio em particular, por exemplo, escoamento de fluidos. Alguns deles são:

cálculos a efetuar são geralmente vultosos. Também numa animação, as cenas terão de ser totalmente recalculadas, mesmo que as diferenças sejam poucas.

² **Renderização** é o processo de geração de uma imagem em uma tela a partir de uma descrição computacional. Esse processo varia enormemente, e muitas técnicas estão disponíveis para sua realização, envolvendo desde a geração de desenhos em duas dimensões (2D) até técnicas sofisticadas de geração de imagens tridimensionais.

- *FieldView (Intelligent, EUA)*: ambiente de visualização para dinâmica dos fluídos, fornece as principais funções para planos de corte, superfícies de nível e traçado de partículas, com saídas que incluem animações e vídeo.

- *Data Visualizer (Wavefront Technologies, EUA)*: ambiente para visualização de dados de elementos finitos generalizados. Permite a utilização simultânea de várias técnicas de visualização.

- *SSV (Sterling Software, EUA)*: fornece um grande conjunto de produtos e serviços de visualização, inclusive o programa *FAST (Flow Analysis Software Toolkit, University of Geórgia, EUA)*, que possui vários módulos para visualização escalar e vetorial em malhas híbridas, sob o comando do usuário, por meio de interfaces gráficas.

Outros pacotes que adotam a abordagem turnkey são *SciAn (Supercomputer Computations Res. Inst., gratuito)*, *Xdataslice (NSCA, gratuito)*, *Volvis (Stonybook, New York, gratuito)*, *GLVware (Army High Performance Comp. Res. Center, gratuito)*, *Vis-5D (Bill Hibbard, visualizador de mapas de previsão do tempo, gratuito)*, além de muitos outros, de aplicação específica ou de propósito geral (Elvins, 1994).

3.1.5.1 Sistemas para Simulação e Visualização de Processos em Aquíferos

Nestes itens serão descritos alguns pacotes comerciais, para simulação de processos em aquíferos, destacando-se as principais características para visualização dos resultados. Assim, torna-se possível uma visão geral e abrangente das principais ferramentas para visualização de processos em aquíferos.

- O *Surfer (Scientific Software Group, 2004)* é um pacote de programas comerciais para modelação espacial, que pode ser utilizado para a confecção de mapas de variáveis a partir de dados espacialmente distribuídos. Foi inicialmente desenvolvido para sistema operacional DOS e, a partir da versão 6, passou a ser desenvolvido para ambiente Windows®. Possui um módulo para visualização dos resultados da interpolação. Este módulo foi desenvolvido com o auxílio das bibliotecas *OpenGL*. Converte os dados interpolados por processos geoestatísticos (Kriging, Regressão Polinomial, etc.)

no contorno e na superfície, podendo gerar visualizações de superfícies em estrutura de arame, direção de gradientes, mapas de isolinhas e também visualização de superfície. Possui ferramentas para análise visual, tais como rotação interativa (*trackball*), tabela de cores em RGB, etc. É uma ferramenta versátil para confecção de mapas e para trabalhos específicos que utilizam visualizações de simulação geoestatística.

- O pacote Groundwater Modeling System - GMS 5.0 (Scientific Software Group, 2004) é um conjunto de ferramentas (programas específicos) para cada fase da simulação de processos em aquíferos. Seus principais módulos foram desenvolvidos para meios saturados, não saturado, incluindo módulos para calibração e visualização em 2D e 3D.

Este programa utiliza diferenças finitas, elementos finitos e método dos elementos analíticos, como método numérico para simulação. Os principais programas que fazem parte deste pacote são: MODFLOW 2000, MODPATH, MT3DMS/RT3D, SEAM3D, ART3D, UTCHEM, FEMWATER, PEST, UCODE, MODAEM e SEEP2D.

Para visualização dos processos em aquíferos este pacote utiliza a biblioteca gráfica *OpenGL*. Basicamente, este pacote integra várias ferramentas já conhecidas no meio científico, explorando a interface com o usuário e também técnicas avançadas de visualização. Assim, o GMS é um pacote que conta com uma poderosa ferramenta para visualização dos resultados gerados na simulação, sejam eles por elementos finitos, diferenças finitas e método dos elementos analíticos.

Os modelos podem ser construídos usando mapas digitais e modelo digital do terreno (MDT). Durante a construção do modelo é possível também obter uma visão rápida da apresentação dos parâmetros simulados, para compreensão do domínio e dos parâmetros a serem analisados. A visualização desses modelos podem ser em vistas, planta ou em 3D. Podem ser exploradas as ferramentas tais como a rotação interativa (*trackball*), renderização 3D, cortes transversais em qualquer lugar do domínio e as soluções de estado permanente e transiente podem ser indicadas em um formato MPEG como uma animação. A principal vantagem deste pacote é a apresentação visual dos resultados, gerados na simulação.

- O Argus ONE (Open Numerical Environment) (Scientific Software Group, 2004) é um pacote comercial para ambientes Windows®, que utiliza o SUTRA 2D (Saturated-Unsaturated Transport) como módulo para simulação de processos em

aquíferos. O SUTRA 2D é um programa para simulação em meios porosos, para modelos saturados e não saturados, de domínio público. Este programa utiliza como método numérico, diferenças finitas e elementos finitos. Este pacote utilizou as bibliotecas *OpenGL* para desenvolvimento dos ambientes CAD e GIS, utilizados para visualização dos resultados da simulação.

- FEFLOW (Scientific Software Group, 2004) é um pacote comercial para simulação de processos em aquíferos em meios saturados e não saturados, que utiliza o método de elementos finitos. Este pacote possui ferramentas robustas e sofisticadas para visualização em 2D e 3D, desenvolvidas com auxílio das bibliotecas *OpenGL*. É um dos pacotes mais completos para simulação de processos em aquíferos, para ambientes Unix, Linux e Windows®. Possui ferramentas interativas para visualização de modelos em regime permanente e transiente.

- Visual MODFLOW (Scientific Software Group, 2004) é um pacote de ferramentas para simulação de processos em aquíferos, que utiliza o método de diferenças finitas para simulação dos resultados. Basicamente, este pacote utiliza vários programas desenvolvidos pela USGS, tais como MODFLOW 2000, MODPATH, MT3D, RT3D e PEST. Para visualização dos resultados o Visual MODFLOW utiliza o **VMOD 3D-Explorer**, totalmente integrado à sua interface, que foi desenvolvido com auxílio das bibliotecas *OpenGL*. Com este módulo é possível visualizar os resultados simulados utilizando a visualização 3D com animação interativa.

- SPA (Simulação de Processos em Aquíferos) (Wendland, 2003) é um pacote computacional baseado no método de elementos finitos, destinado à simulação numérica de escoamento e transporte de solutos tridimensional em meios porosos heterogêneos. Este pacote, em desenvolvimento no Laboratório de Hidráulica Computacional (LHC) do Departamento de Hidráulica e Saneamento – EESC – USP, dispõe de recursos de pré e pós-processamento e é dividido em módulos com aplicação em diferentes etapas do processo de modelação de escoamento e transporte de poluentes em aquíferos. O SPA é composto dos seguintes programas: CMA (Construtor de Modelos de Aquíferos), VMA (Verificador de Modelos de Aquíferos), FAS (Fluxo de Água Subterrânea), TP (Transporte de Poluentes) e VPA (Visualização de Processos em Aquíferos). Para visualização dos resultados obtidos na simulação pelos módulos do SPA, foi desenvol-

vido o VPA. O VPA é um programa para visualização em 2D e 3D, que utiliza como ferramenta de visualização o *OpenDX* (desenvolvido com a biblioteca *OpenGL*). Com este programa é possível visualizar os resultados por mapeamento de cores pelo padrão RGB, isolinhas, direção de gradientes, análise por intensidade de gradientes, a malha de elementos finitos e superfícies em 3D.

3.1.6 Geradores de Aplicação (*Application Builders*)

Os geradores de aplicação permitem que o usuário crie um sistema de visualização “personalizado”. Além disso, eles admitem que usuários escrevam códigos em módulos próprios para tarefas de visualização, incluindo-os no contexto dos demais módulos pré-programados. Alguns dos mais conhecidos são:

- *AVS (Advanced Visual Systems, EUA)*: dividido em quatro módulos (um visualizador de imagens, um de geometria, um de volume e um editor de rede), possui interface visual para o projeto da rede de objetos (Upson et al., 1992). Tem sido progressivamente transportado para muitas plataformas (Herndon et al., 1994). Por outro lado, o ambiente para visualização modular como o *AVS*, tornou-se uma ferramenta popular para visualização e análise de dados científicos. *Sliders* e entradas textuais são exemplos de ferramentas de manipulação utilizadas. *Widgets 3D* são definidos como objetos que encapsulam geometria 3D e comportamento, utilizados para controlar ou exibir informações sobre objetos de aplicações 3D (Conner et al., 1992). As partes que compõem um *widget 3D* reagem aos eventos de interação com o *mouse*, depois para o *widget* e finalmente, para o objeto. Dispositivos com mais graus de liberdade vão aliviar esses problemas mas, no estágio atual, ainda apresentam problemas no uso e na resolução espacial (Herndon et al., 1994). Como solução desses problemas, já foram implementadas várias ferramentas que fazem uma ponte entre os dispositivos 2D e a interação 3D, para aumentar a usabilidade das aplicações tridimensionais nesse visualizador (AVS, 2004).

- *IRIS Explorer*: permite selecionar módulos pré-compilados numa abordagem de seleção visual simples por meio do *mouse*. Módulos foram construídos em C e FORTRAN, e são selecionados por meio de um dos componentes do programa: o *Li-*

brarian. Os outros dois componentes são um Editor de Mapas, que permite editar a rede de módulos, e o *DataScribe*, responsável pela conversão de formatos entre o Explorer e outros programas. Era fornecido gratuitamente com máquinas *Silicon Graphics*, mas atualmente é comercializado pela NAG (NAG's IRIS Explorer, 2003).

- *Khoros* (Young, 2000): inclui algumas facilidades de programação visual. Possui geradores para estender a linguagem e para adicionar novos pacotes de aplicação ao sistema. É executado em sistemas *Unix/Linux*, distribuído como *open source development* (Williams, 2002).

- *ApE* (TaraVisual, 2004): escrito na linguagem C, e originalmente desenvolvido na Universidade de Ohio para domínio público, é atualmente comercializado pela TaraVisual. Permite ao usuário incorporar os próprios módulos. Executa em várias plataformas, a custo relativamente baixo. Este programa possui interface gráfica para desenvolvimento de processos de visualização baseados em *pipelines*, sendo possível manipular os objetos em 2D e 3D. É muito usado para visualizações de MDT (Modelo Digital do Terreno).

- *IBM OpenDX (Visualization Data Explorer)* (Earnshaw, 1993): é um programa para computação gráfica utilizado para visualização científica. É desenvolvido com o auxílio das bibliotecas *OpenGL*. No ano de 1998, passou a ser distribuído gratuitamente nos âmbitos de *open source development*. Segue a mesma lógica de fluxo de dados (saídas de um módulo são entradas de outros, esses recursos são chamados de *pipeline*) dos demais nessa classe. Possui um editor visual de programas e várias ferramentas interativas para exploração dos dados. Também permite ao usuário a criação dos seus próprios módulos, que podem ser compartilhados em rede. Seu formato de arquivo para entrada de dados, é muito flexível, proporcionando ao pesquisador, várias ferramentas para o estudo visual dos resultados obtidos. Dados dificilmente estudados sem recursos de visualização podem ser explorados ao máximo usando-se as inúmeras associações possíveis de serem feitas com as diversas ferramentas presentes no *OpenDX* (IBM, SC38-0486, SC38-0496, SC38-0497, 1997). Com o *OpenDX* (Blaz, 2003), o pesquisador pode visualizar seus dados usando recursos de animação, enriquecendo ainda mais a análise dos dados, sem precisar usar a programação, o que às vezes torna a visualização complicada. Diferente de outros programas científicos, o *OpenDX* não faz

cálculos de problemas físicos, ele é um programa essencialmente de visualização. Porém, é preciso salientar que o *OpenDX* pode ser integrado a outros programas de simulação, com sua interface gráfica (Delerue, 2002) e com isso sobressai de maneira versátil e robusta (Earnshaw, 1993). Sua principal diferença está na interface gráfica e nos seus comandos para manipulação dos dados a serem visualizados. Pode-se visualizar diretamente os dados gerados por arquivos binários ou nos seguintes formatos:

- *Data Explorer File*: o formato de arquivo *.dx* é flexível, podendo representar qualquer objeto. É o formato padrão para visualizações no *OpenDX*;
- *CDF Format*: é um tipo de arquivo *standard*, ou seja, um tipo de arquivo que o *OpenDX* importa diretamente (muito usado pela: *National Space Science Data Center - NASA Goddard Space Flight Center*);
- *netCDF Format*: formato de dados muito usado no meio acadêmico. Pode ser importado diretamente pelo explorador (muito usado pela *Unidata Program Center in Boulder, Colorado*, em seus simuladores);
- *HDF Format*: formato de arquivo que importa diretamente figuras, tais como: *TIFF*, *MIFF*, *GIF* e formato *RGB*. Este formato de arquivo para visualização foi desenvolvido em pesquisas pela *National Center for Supercomputing Applications*;
- *Grid or Scattered File*: importa dados multidimensionais ou multivariáveis em série;
- *Spreadsheet Format file*: este formato de arquivo importa planilhas eletrônicas ou banco de dados (*MySQL*, *Oracle*, etc.), tipicamente usados para descrever superfícies, etc.

A geração de novas aplicações que consistem em ferramentas e programas, podem ser incorporadas ou não no *OpenDX*. Assim, contamos com uma ferramenta muito robusta para visualização científica antes inacessível.

Neste trabalho foi escolhido o *OpenDX*, como ferramenta utilizada pelo VPA para visualização de processos em aquíferos, porque possui entrada de dados de fácil programação. Assim, com o VPA pode-se visualizar os dados gerados na simulação do SPA, facilitando muitos processos de visualização, como a visualização das malhas irregulares com elemento mistos (triangulares e quadrilaterais), visualização de isolinhas, visualização da direção de gradientes, visualização de gradientes por intensidade de cores, visualização com mapa de cores RGB e a visualização de superfícies. Uma das características principais para escolha do *OpenDX* foi ele possuir seu código aberto. Com essas características foi possível desenvolver o VPA. Muitos programas e bibliotecas citadas nesse levantamento, como por exemplo *AVS*, são pacotes comerciais e não possuem código aberto para desenvolvimento de aplicações para processos em aquíferos.

3.2 Ferramentas utilizadas para desenvolvimento do VPA

3.2.1 Uma breve introdução sobre o sistema operacional GNU/Linux

A importância dos programas livres, junto ao meio acadêmico, tem aumentado nos últimos anos. Na prática, a distribuição de código-fonte de programas remonta aos primeiros sistemas produzidos na década de 60. No início da década de 80, *Richard Stallman*, preocupado com a crescente e dominante comercialização dos programas proprietários, criou o movimento *GNU* (*GNU is not Unix*), para estabelecer uma nova forma de desenvolvimento e distribuição de programas. Em 1992, o finlandês *Linus Torvald* conseguiu compilar todos os programas e ferramentas do movimento *GNU* (Loukides, 1997) em um *kernel* (núcleo do sistema operacional), o que resultou no sistema operacional *Linux*, ou seja, “*Linus for Unix*” pois o esforço era obter um padrão equivalente ou superior aos sistemas *Unix*. Enquanto o sistema operacional *Windows®* foi criado no sentido do uso “doméstico” para o “corporativo”, o *GNU/Linux* foi criado inversamente, do foco no uso “corporativo” para o “doméstico”.

A versatilidade do *GNU/Linux* aliada à sua performance operacional, tem conquistado também as áreas de pesquisa científicas como: medicina, cinematográfica, hidrogeologia, petrolífera, negócios entre outras. Baseado nos esforços de 400 mil desen-

volvedores espalhados pelos cinco continentes, em mais de 90 países, o *GNU/Linux* é o resultado de um trabalho coletivo, jamais visto na história da Humanidade. Em breve, o *GNU/Linux* atingirá a marca de 1 milhão de colaboradores, enquanto a Microsoft conta com 30 mil funcionários. Este fato é uma sentença, que dificilmente poderá ser enfrentado por qualquer empresa privada: acompanhar o ritmo de inovações tecnológicas de uma rede tão organizada, inteligente e revolucionária como a do *GNU/Linux*.

Sendo assim, o *Linux* (Schwarz, 2002) é um tipo de Unix (Lucas, 2002) e, como tal, possui todas as ferramentas básicas e avançadas de programação. Todas as distribuições de *Linux* costumam vir com um núcleo de programas. Dentre eles destacamos: 1) o comando *grep* e suas variantes, que encontram expressões regulares dentro dos arquivos; 2) os comandos *sed* e *awk* (Dougherty, 1997) que manipulam arquivos textos; 3) o compilador *GCC* (Griffith, 2002) e o *debugger GDB* (Zeller, 2000), específicos para C, C++ (Barr, 1999) e Objective C, além de Pascal, Assembler, Fortran 77 e ADA 95; 4) os editores de texto *vi* e *Emacs* (Cameron, 1999) para programação e 5) *autoconf* (MacKenzie, 2002), *automake* (MacKenzie, 2003) e *make* (Griffith; Stallman, 2002) que são utilitários para gerenciar a compilação de grupos de programas.

Vale salientar que a escolha das ferramentas depende muito dos projetos que estão em desenvolvimento. A princípio, um programador que desenvolve programas em linguagem C (Oualline, 1997) precisa dominar um compilador e *debugger*. Neste trabalho foi utilizado o compilador *GCC* e *debugger GDB* (*GNU Debugger*) utilizado pelo ambiente *DDD* (*Data Display Debugger*) (Feichas, 2003). Para edição dos códigos foram usados os editores *vi* e o *XEmacs* (Glickstein, 1997). Abaixo serão descritas as principais ferramentas utilizadas no desenvolvimento do VPA.

3.2.2 O ambiente de programação ideal - *Glade 2.0* (GTK+ 2)

O *Glade 2* é uma “mini” ferramenta *RAD* (*Rapid Application Developer*). Ao contrário das ferramentas *RAD* tradicionais, o *Glade 2* não estabelece vínculos estreitos com nenhuma linguagem de programação em particular, mas pode gerar códigos em C, C++ (Eckel, 2000), Ada 95 e Perl. Ferramentas de terceiros adicionam (extensões) suporte a Pascal, Python, Java e outras linguagens. O *Glade 2* foi utilizado para ser o construtor gráfico da interface do VPA (formulários).

O ambiente *GNOME*, por exemplo, foi construído a partir da biblioteca de componentes *GTK+ 2*, criada originalmente para a confecção do *Gimp* (GNU Image Manipulation Program). O *GTK+ 2* foi escrito em linguagem C, e portanto, não utiliza os recursos de orientação a objetos do C++, mas foi projetado de acordo com bons princípios de POO (Programação Orientada a Objetos). O objetivo era definir uma API facilmente mapeável para outras linguagens de programação. Os *bindings* (ligação entre API e linguagem) do *GTK+ 2* para C++, Java ou PHP produzem o mesmo resultado. Outra característica do *design* do *GTK+ 2* é abstrair as APIs de baixo nível do sistema operacional nas bibliotecas Glib e GDK. A primeira fornece estrutura de dados, gerenciamento de memória, manipulação de arquivos e outros serviços do sistema operacional, enquanto a segunda fornece as primitivas gráficas. Conseqüentemente, aplicações escritas em linguagens multiplataforma, como Java ou PHP, funcionam em todas as plataformas suportadas pelo *GTK+ 2* sem quaisquer modificações. A performance do *GTK+ 2* possui tanta qualidade quando as APIs gráficas ou bibliotecas de componentes nativas das plataformas suportadas, entretanto o *GTK+ 2* possui a sua própria aparência (*look-and-feel*), que é um misto de Motif e Windows®. O *GTK+ 2* fornece uma base sólida para o *Glade 2*, que possui dois modos de operação. No primeiro, a descrição da interface é escrita em um dialeto XML particular do *Glade 2* e esta descrição é utilizada para gerar código em C ou outra linguagem. O código deve ser editado para incluir as rotinas de tratamento de eventos e compilado para gerar um executável nativo. É possível voltar ao *Glade 2* para modificar a interface e gerar novamente o código GUI, sem perder a lógica de tratamento de eventos. O segundo modo de operação é mais interessante: a descrição em XML da interface é processada em tempo de execução por meio da biblioteca *LibGlade*, para gerar os componentes visuais. A *LibGlade* fornece meios de, dado o identificador de um componente qualquer, obter uma referência ao componente que pode então ser manipulada para responder a eventos ou modificar propriedades. Foi justamente a necessidade de internacionalizar aplicações que motivou este segundo modo de operação de uma aplicação GUI gerada pelo *Glade 2*. Algumas IDEs (*Integrated Development Environment*) para Unix e *GNU/Linux* (Siever, 2000), como o Anjuta, utilizam o *Glade 2* como componente de desenho de formulários, formando um conjunto mais próximo da ferramenta *RAD* estilo Windows®. O *Glade 2* tem a vantagem de minimizar o aprendizado para os desenvolvedores desktop, que não necessitam aprender XML em absoluto para usufruir da ferramenta e já ter sido validado por centenas de aplicações reais, em geral escritas para o *desktop GNOME* e *KDE* (Lozano, 2003).

3.2.3 O compilador *GCC* e o *debugger* *GDB* (*DDD*)

O *GCC* começou a ser desenvolvido no início da década de 80 para ser um compilador da linguagem C. Seu objetivo era ser a base de um projeto para substituir as versões comerciais do UNIX (Maxwell, 2002) existentes na época, as quais não tinham o código-fonte disponível. Desde então, outras linguagens foram acrescentadas: C++, Objective C, awk, Pascal e ADA. O *GCC* se tornou o melhor compilador de C do mundo, devido à colaboração de milhares de programadores.

Esse projeto depois desacelerou o desenvolvimento do *GCC*. Então, uma empresa americana (*Cygnus*) resolveu fazer um “concorrente” (chamado EGCS), também *open source*, porém ganhando dinheiro com o suporte. Graças a isso, o desenvolvimento se acelerou e o compilador, hoje, já é melhor que o *GCC* original para C++, porém não para C. Criou-se, também, o projeto do Pentium *GCC* (*PGCC*) para aproveitar a arquitetura do Pentium e, assim, compilar programas. Testes com o *PGCC* revelam uma diminuição de 30% em média, na hora de carregar programas compilados em relação ao *GCC* original. Porém, não existem programas sem erros de programação. Assim, é necessário um *debugger*. Junto com o *GCC* foi criado o *GDB*, que hoje também debuga várias linguagens além do C, como C++, Pascal, Perl e Fortran.

O *GDB* roda dentro de um terminal texto (em linha de comando). Por isso, foram desenvolvidos *front ends* gráficos para ele. Temos o *DDD*, o *xxgdb* e o *XEmacs* com *GDB* internamente, por exemplo. O *DDD* (*Data Display Debugger*) se destaca entre eles. Possui várias características ideais a um *debugger*, como leitura do valor de uma variável quando da passagem com o mouse por cima dela, amostragem de variáveis em diversos formatos, como decimal, octal e hexadecimal, visualizador de gráfico do valor de variáveis (Oliveira, 1999). O uso deste programa deve ser feito principalmente quando um programa que está sendo desenvolvido já atingiu um estágio avançado, e erros de execução se tornam impossíveis de resolver simplesmente analisando o código a “olho nú”.

3.2.4 Ferramentas para desenvolvimento no *GNU/Linux*

A compilação é o processo de transformar os comandos da linguagem C (ou outra linguagem) para comandos que o computador entende, chamados de linguagem de máquina. O processo pode ser encarado como um tradutor, convertendo o código fonte (texto) em código binário (zeros e uns).

O resultado final de uma compilação é um arquivo denominado executável que contém os comandos em linguagem de máquina que acessam arquivos do tipo: binário, acesso seqüencial e aleatório.

A link-edição é a adição ao código objeto desses comandos, que são copiados das bibliotecas às quais pertencem. O produto final da link-edição é um programa executável. Um programa, para ser gerado, depende de um (ou mais) código-objeto e bibliotecas. Além disso, o código-objeto e as bibliotecas dependem de um código fonte e dos *includes (header files)* associados a eles. O comando *make* é bastante poderoso e genérico. Ele é um dos integrantes da “caixa de ferramentas” do *Linux*. Seu principal objetivo é o de automatizar as tarefas relativas à geração de programas, desde a compilação até a instalação deles no sistema.

O *make* não serve apenas para programas escritos em linguagem C/C++, mas para virtualmente qualquer linguagem. Este utilitário foi desenvolvido para manter grupos de programas. O seu propósito é determinar automaticamente que pedaços de grandes programas precisam ser recompilados e quais programas precisam ser usados para recompilá-los.

Para usar o *make*, é necessário construir um arquivo chamado *Makefile*, que conterá as dependências entre os arquivos do programa.

Basicamente, o arquivo executável é atualizado com os novos arquivos-objetos, que por sua vez são feitos por arquivos-fontes. Criado esse arquivo, um simples comando *make* compilará, de tempos em tempos, somente os programas-fontes que tiverem sido atualizados. Assim o programa *make* checa se houve alteração em qualquer um dos módulos, a partir da data de criação do arquivo correspondente ao programa que está sendo gerado.

3.2.5 Metodologia para Visualização de Processos em Aquíferos

3.2.5.1 Classificação dos Dados para Visualização

Uma massa de dados pode ser considerada como uma entidade que contém um conjunto de valores em um certo intervalo de variação e está definido sobre algum domínio de variáveis independentes. Podemos expressar matematicamente essa entidade como uma função de múltiplas variáveis $F(x_1, x_2, \dots, x_n)$, em que n é a dimensão do domínio. Entretanto, nem sempre o conjunto de dados implica na existência de um mapeamento, podendo ser visualizado um conjunto de pontos esparsos sem qualquer função associada.

A notação de Brodlié é a mais utilizada em Visualização Científica para expressar o tipo de dados a serem visualizados. Uma entidade escalar S definida sobre um domínio de dimensão n , por exemplo, é representada pelo símbolo E_n^S ; entidades vetoriais e tensoriais, por sua vez, são denotadas pelos sobrescritos V e T . A dimensão das entidades vetoriais ou tensoriais pode ser especificada por meio de um sufixo (por exemplo, V_3). Assim, $E_2^{V_2}$ denota um conjunto de vetores de dimensão 2 definido sobre um domínio 2D. Um conjunto de pontos esparsos definido sobre um domínio 3D é denotado por E_3^P .

Essa distinção entre pontos esparsos, escalares, vetores e tensores é a base para a principal classificação das técnicas de visualização. A dimensão do domínio é usada para uma sub-classificação, e permite estabelecer distinções adicionais de acordo com a natureza do domínio. Por exemplo, a entidade pode ser definida ponto a ponto ou sobre regiões em um domínio contínuo, ou pode ser definida sobre um conjunto enumerado. Assim, a notação $E_{[n]}$ indica um domínio n -dimensional com a entidade definida sobre regiões e não sobre pontos individuais, e $E_{\{n\}}$ indica um domínio n -dimensional consistindo de um conjunto de valores enumerados. É muito comum, também, ter-se entidades que contêm mais de um conjunto de valores. Por exemplo, pode-se ter dois campos escalares (como pressão e temperatura) definidos sobre um mesmo domínio, e deseja-se mostrar ambos na mesma representação gráfica. Essa situação é representada por E_n^{mS} , indicando um conjunto de m escalares definidos sobre um domínio n -

dimensional. Existem técnicas de visualização específicas para entidades do tipo conjunto de pontos, escalares, vetores e tensores. A dimensão do domínio também é particularmente importante na definição de métodos de visualização apropriados. No caso de conjuntos de pontos, uma técnica usual é a dos bem conhecidos *scatter plots*, que podem ser mostrados no espaço 1D, 2D, ou 3D (uma “nuvem” de pontos no espaço). Em geral, um conjunto de dados possui uma certa organização que normalmente consiste de uma malha (ou grade) de células, onde os dados estão posicionados nos vértices de cada célula e valores no interior das células podem ser obtidos por interpolação. Dessa forma, a estrutura de um conjunto de dados está implicitamente definida por pontos (que associam informação geométrica aos itens de dados) e células (que associam propriedades topológicas). Os dados propriamente ditos encontram-se atribuídos aos elementos (pontos, células ou elementos) dessa estrutura. Em uma malha cartesiana, por exemplo, todos os elementos são quadrados (no caso 2D) ou cubos idênticos (no caso 3D), alinhados aos eixos principais. Na Figura 2a pode ser observado o caso 2D.

Figura 2 – Diferentes tipos de malhas.

Uma malha regular tem todos os seus elementos idênticos e alinhados aos eixos, mas esses elementos são retângulos (paralelepípedos) regulares, ao invés de quadrados

(Figura 2b). Elementos de uma malha retilínea são quadriláteros ou hexaedros alinhados aos eixos, mas não necessariamente idênticos, como ilustrado na Figura 2c. Elementos em uma malha estruturada são quadriláteros ou hexaedros não alinhados aos eixos principais, como os que aparecem em grades esféricas ou curvilíneas (a Figura 2d ilustra um caso 2D). Uma malha estruturada em blocos é um conjunto de malhas estruturadas agrupadas. Uma malha não-estruturada (Figura 2e) contém polígonos ou poliedros sem qualquer padrão explícito de conectividade. No caso 3D, as células podem ser tetraedros, hexágonos, pirâmides, etc. Uma malha híbrida é uma combinação de quaisquer dos tipos anteriores.

Em resumo, a principal distinção é que conjuntos de dados podem ser regulares (estruturados) ou irregulares (não-estruturados). Conjuntos regulares apresentam um relacionamento inerente entre os pontos aos quais estão associados os dados. Com isso, não é necessário armazenar explicitamente as posições de todos os pontos. No caso de uma malha cartesiana, por exemplo, basta manter a posição inicial, o espaçamento entre os pontos e o número total de pontos, o que permite uma representação computacional compacta. Esse não é o caso dos conjuntos irregulares que, por outro lado, têm a vantagem de serem mais flexíveis e de permitir representações adaptativas (com mais células nas regiões nas quais a informação é mais densa) (Minghim & Oliveira, 1997).

4 Desenvolvimento do VPA

4.1 Arquivos binários

Todos os programas do pacote SPA, foram desenvolvidos com o auxílio de um conjunto de bibliotecas denominadas DtnLIB. Esse conjunto de bibliotecas tem como finalidade, a integração dos programas desse pacote, tornando possível a leitura e escrita de arquivos binários.

Esses arquivos binários são criados para armazenar as informações dos parâmetros das simulações. Os programas executam as bibliotecas DtnLIB, que organiza e acessa os dados da simulação conforme sua organização. Na Figura 3 pode ser observado o processo de leitura e escrita para visualização dos resultados. Os arquivos binários, gerados pelos programas de simulação do SPA, permitem uma maior flexibilidade (leitura e escrita), entretanto exige maiores recursos de programação e depuração do código fonte para sua geração.

Figura 3 – Processo de leitura e escrita para visualização dos resultados.

Visando o acesso rápido à lista de parâmetros disponíveis, os arquivos são gerados em pares associados: o arquivo de dados propriamente dito, no qual estão armazenados os vetores de dados, e associado a um arquivo índice, o qual contém a lista dos respectivos dados disponíveis, com informações adicionais (parâmetro identificador, classe, número, valores Min./Máx., instante, etc.).

Os dados da simulação são separados em diversos arquivos binários que são identificados por 3 letras iguais, nomeados a seguir:

- O arquivo binário de controle **aaa** não é gerenciado pela DtnLIB. Este arquivo de controle contém informações para execução dos módulos do SPA.

- Informações do Modelo e do Projeto (arquivos de índices e dados do modelo):

bbb	—	arquivo de índices;
ccc	—	arquivo de dados;

- Resultados de interpolação e de simulação em regime permanente (arquivos de índices e dados, com resultados de interpolação e de simulações em regime permanente):

ddd	—	Índice;
eee	—	Dados;

- Resultados de simulações em regime transiente (arquivos de índices e dados, com resultados de simulações em regime transiente):

fff	—	Índice;
ggg	—	Dados 1 → último instante;
hhh, iii, jjj	—	Dados 2, 3, 4 → ao longo do tempo;
kkk	—	Dados 5 → subsidência superficial transiente;

- Resultados de pós-processamento (arquivos de índices e dados, com resultados de pós-processamento):

III	–	Índice;
mmm	–	Dados;

- Dados de modelação inversa (arquivos de índices e dados, com resultados de modelação inversa):

nnn	–	Índice;
ooo	–	Dados;

4.2 Desenvolvimento da Interface do VPA

O VPA foi desenvolvido com auxílio de várias técnicas computacionais. Para o desenvolvimento deste programa, foi usado o sistema operacional *SuSE Linux* (Mazzano, 2000) com os ambientes (*Desktop*) *KDE* e *GNOME*. A interface gráfica do VPA (*Front End*), os códigos e as rotinas do visualizador, foram desenvolvidos com o auxílio da ferramenta *RAD* para programação, *Glade 2*.

Na Figura 4 é demonstrada a montagem da interface do VPA, com o ambiente de programação *Glade 2*.

Esta ferramenta permite o desenvolvimento de interfaces para o usuário de forma rápida e fácil, utilizando C como linguagem de programação. Este ambiente de programação foi escolhido por utilizar a mesma linguagem de programação em que foi desenvolvida a biblioteca DtnLIB, utilizada para leitura dos resultados contidos nos arquivos binários.

Após a montagem da interface do VPA, foram atribuídos os parâmetros (“*Signals*”) para cada componente da interface. Neste caso, deve-se selecionar o objeto na interface (do VPA), e criar um “*Signals*”, na janela de propriedades dos *Widgets* do *Glade*. Ilustrada na Figura 5.

Figura 4 – Desenvolvimento do VPA com auxílio do ambiente de programação *Glade 2*.

Figura 5 – Criação de um “Signals” para o objeto.

Para geração dos arquivos principais do programa, devem ser configuradas as opções do projeto, por meio do botão “*Options*”, e clicar no botão “OK”. Serão criados então, no um diretório denominado “src”, que contém os arquivos necessários para programação da interface. Nas Figuras 6 e 7 são demonstrados como configurar as opções de projeto no *Glade*.

Os arquivos são gerados automaticamente pelo *Glade*, mas para isso, é preciso pressionar o botão “*Build*”, da janela principal do *Glade*, que cria o diretório **/home/jeq/VPA**.

Para gerar os códigos fonte, foi necessário utilizar um terminal *xterm*, e digitar o caminho do diretório **/home/jeq/VPA**. Entrando neste diretório é necessário executar o *script* “**./autogen.sh**”, que foi criado automaticamente pelo *Glade*. Executando esse *script* (*Korn Shell* (Olczak, 2000)), será criado um subdiretório denominado “src” contendo os seguintes arquivos:

- **callbacks.h**: Cabeçalhos para funções de callback;
- **callbacks.c**: Neste código são incorporadas as funções vinculadas ao objeto visual. São escritos os códigos de chamada para execução de funções DtnLIB, para gerar o arquivo de visualização, etc.;
- **interface.h**: Cabeçalhos para funções que criam a interface;
- **interface.c**: Código que contém as funções para criação da interface;
- **support.h**: Cabeçalhos para funções de suporte;
- **support.c**: Código que contém as funções de suporte;
- **main.c**: Neste código também são incorporadas as funções principais de leitura dos arquivos binários.

Dentro do diretório “src” do VPA, foi criado outro diretório denominado “read-mesh”. Neste diretório foram criados os códigos para auxiliar na leitura dos resultados armazenados em arquivos binários.

Figura 6 – Configuração dos parâmetros do projeto.

Figura 7 – Configuração dos parâmetros, para geração dos arquivos.

Para a leitura dos arquivos binários, é inicializada a biblioteca DtnLIB, que utiliza a função de cabeçalho denominada “dtnlib.h”, declarada na função “main.c”. Para leitu-

ra dos dados disponibilizados pela DtnLIB, foram desenvolvidas várias funções que foram incluídas em uma biblioteca estática denominada “**libremesh.a**”. Essa biblioteca estática foi gerada com o auxílio de um *makefile* (ferramenta para compilação descrita no item 4.4), que compila os códigos fonte das funções desenvolvidas, que utiliza a ferramenta (comando) “*ar*” do *GNU/Linux*. Nessa biblioteca foram armazenadas as funções: *d2c.c*, *elemmi.c*, *f2c.c*, *formsort.c*, *initlib.c*, *ler_infos.c*, *ler_malha.c*, *permanente.c* e *vpa2dx.c*. As principais características desses arquivos-objeto, são:

- *d2c.c*: Este código contém as funções para conversão das variáveis *double* para *char*. Quando as bibliotecas DtnLIB fazem a leitura dos arquivos binários, esse código converte os dados armazenados em formato binário (*int*, *float*, *double*, *long*, *char*, *etc.*) para caracteres, sendo assim possível a exibição na interface gráfica.
- *f2c.c*: Este código é semelhante ao “*d2c.c*”. Sua única diferença é a conversão das variáveis *float* para *char*.
- *formsort.c*: Este código contém a função para ordenação dos dados da malha;
- *initlib.c*: Neste código estão as funções para inicialização de outras funções contidas na DtnLIB e para leitura das informações das informações do modelo simulado;
- *ler_infos.c*: Esta função foi desenvolvida para ler os dados do projeto após a inicialização das funções da “*initlib.c*”;
- *ler_malha.c*: Esta função, após a leitura dos dados pela função “*ler_infos.c*”, são disponibilizados os dados sobre a malha gerada. Neste caso, as informações são exibidas na interface gráfica do usuário. São disponibilizados os seguintes dados: número de nós, lista das conexões entre os nós, e o número de elementos, que podem ser triangulares ou quadrilaterais;

- `permanente.c`: Com esta função é feita a leitura dos valores escalares, atribuídos aos nós ou aos elementos da malha gerada;
- `vpa2dx.c`: Este código estão incluídas as funções que lêem os parâmetros do modelo para visualização dos processos em aquíferos. Após a disponibilização dos dados, pelas bibliotecas DtnLIB, as funções `ler_infos`, `ler_malha` e `permanente`, disponibilizam as listas de vetores e escalares necessários para visualização dos resultados. Assim é escrito o arquivo “vpa.dx”, utilizado pelo programas (*pipelines* de visualização) do *OpenDX*.

Na programação do VPA, somente nos arquivos **callbacks.c** e **main.c**, foram adicionados todos os códigos acima descritos, para chamada das funções das bibliotecas DtnLIB. No arquivo **main.c** foi declarada a biblioteca (*header files*) para leitura e escrita dos arquivos binários, denominada “**dtnlib.h**”.

Com a integração do SPA com o VPA (com o auxílio das bibliotecas DtnLIB), foi possível ler os dados, para interação com o usuário. Para visualização do arquivo “vpa.dx”, o VPA utiliza outros programas que serão descritos no próximo tópico.

4.3 Tipos de formatações utilizadas para escrita do “vpa.dx”

O arquivo gerado pelo VPA, denominado “vpa.dx” foi escrito seguindo normas que são descritas neste item.

Para visualização com os programas do *OpenDX* é preciso formatar os dados, utilizando alguns elementos básicos, tais como: pontos, posições, etc. A conexão entre essas posições é um dado relativo a cada posição ou a cada elemento formado pela conexão dos pontos. Uma estrutura de elementos finitos requer uma malha, cujos nós são indicados pelas posições, pelo tipo de elemento (triangular, quadrilateral, etc.), indicando a conectividade dos nós, e por alguns dados ligados ou ao nó ou ao elemento da malha.

O *OpenDX* denomina um conjunto mínimo de dados de *field*, sendo que para o *OpenDX* um *field* deve possuir no mínimo posições e conectividades, ou dados.

Para a criação de um *field* o *OpenDX* se utiliza de componentes, assim um *field* pode apresentar um componente posição, um componente conectividade e um componente dados (Silva, 1997).

4.4 Componentes de cada objeto do *field*

Os componentes de um *field* são denominados *objects*. A indicação de um componente se dá por meio de uma linha de cabeçalho que contém as seguintes informações:

- ***Object identifier***: pode ser um número ou um nome que serve de identificação ao objeto;
- ***Class***: indica se os dados são do tipo *array* (conjunto de dados), *series* (série de dados) ou *group* (grupo de dados);
- ***Items***: indica o número de dados que contém a lista;
- ***Data***: indica onde estão os dados, podendo estar logo a seguir da linha de cabeçalho (*follows*), em algum ponto do arquivo devendo ser fornecida a posição por meio do número de posições onde se encontra o primeiro dado, ou em outro arquivo, devendo ser indicado o *path* (caminho do arquivo) do arquivo.

Os objetos *class group* são como ponteiros para *fields*, outros grupos ou séries, assim os grupos (*groups*) são agrupamento de dados, sendo interessantes em caso de grande diversidade de dados, caso em que é aconselhável a divisão dos resultados em diversos arquivos.

Os objetos *class array* indicam que os dados formam uma lista. Os parâmetros de um objeto *class array* são:

- ***Type***: indica o tipo do objeto, podendo ser *int*, *long double*, *float*, etc.;

- *Rank*: é um índice, entre 0 e 2, no caso de 0 indica que o dado é um escalar, 1 indica que os dados são apresentados sob a forma de vetor, e 2 indica que os dados estão sob a forma de matriz;

- *Shape*: é um índice que indica a ordem de cada dado, assim um escalar (*rank* 0) não tem *shape* (formato de configuração), já no caso de um vetor (*rank* 1), *shape* indicará a ordem de cada dado do vetor, assim caso cada dado do vetor seja um escalar *shape* será 0, em caso de um vetor *shape* indicará a ordem do vetor (*shape* 3 indica um vetor de 3 índices), a indicação de *shape* para o caso de dados em forma de matriz (*rank* 2) ocorre da mesma forma, sendo agora fornecido a ordem de cada elemento da matriz.

Para exemplificar esta classificação, são fornecidos os exemplos abaixo:

- Vetor de pontos no plano (coordenadas bidimensionais): *rank* 1 *shape* 2;
- Vetor de pontos no espaço (coodenadas tridimensionais): *rank* 1 *shape* 3;
- Tensor de pressão (matriz onde cada elemento é uma matriz 3x3): *rank* 2 *shape* 3x3;

Os dados de um programa de elementos finitos são indicados por meio de três objetos: um indicando as coordenadas dos nós (*positions*), outro indicando o tipo de elemento que forma a malha (*connections*), e o terceiro é o dado referente ao nó ou a um elemento da malha (*dependency*). Assim a classe dos objetos indicados acima é do tipo *array* pois cada um dos componentes da malha forma um vetor de dados que pode ser mapeado para tornar possível a análise numérica. Quanto ao tipo das coordenadas dos nós são geralmente dos tipos *long double*, *double* ou *float*, as conectividades são do tipo *int* pois indicam o número dos nós que formaram o elemento, e o tipo dos dados será função do tipo de análise do qual este for resultado.

4.5 Desenvolvimento dos *scripts* (*pipelines*)

Para visualizar os resultados da simulação gerados pelo SPA, o VPA escreve um arquivo chamado “vpa.dx”, que utiliza os recursos e as ferramentas do *OpenDX* para ser visualizado. Assim, é necessário otimizar o processo de visualização, utilizando-se do editor visual de programas (uma das ferramentas do *OpenDX*), conhecido também como VPE (“*Visual Program Environment*”). O VPE utiliza “*Glyphs*” (ícones) que pode-se interligar formando uma rede (*networks*) de *pipelines*.

Os *scripts* desenvolvidos não são executáveis, e são diferenciados pela extensão “.net” (*network*), que acionados pelo VPA, otimizam a visualização dos processos em aquíferos. Os principais programas desenvolvidos foram:

process.net – Este *script* foi desenvolvido para visualização do parâmetro escolhido na interface do VPA.

Este script faz a representação dos resultados de uma malha com elementos mistos por meio de um mapa de cores associados a uma tabela de cores (*ColorBar*). Neste caso o módulo “*Import*” indica o nome do arquivo de dados, que neste caso é o arquivo gerado pelo VPA (“vpa.dx”).

O módulo “*AutoColor*” faz o mapeamento dos resultados por meio de cores RGB, que é utilizado também pelo “*ColorBar*”. No módulo “*Collect*” é reunido todas as informações para ser visualizadas pelo módulo “*Image*” (Figura 8).

iso.net – Este *script* foi desenvolvido para visualizar isolinhas (Figura 9). Para o desenvolvimento desse script foram utilizados os mesmos *glyphs* do exemplo anterior, o único diferente foi o “*Isosurface*”.

fluxo.net – Este *script* foi desenvolvido para visualização da direção dos gradientes, por meio de ícones no formato vetores (Figura 10).

Para representar a direção dos gradientes, foram usados os mesmos *glyphs* do *script* “*process.net*”, incorporando-se o módulo “*Gradient*” que coleta os dados em formato escalar e faz a indicação da direção com a representação de setas, geradas pelo módulo “*AutoGlyph*”.

- **grad.net** – Este *script* foi desenvolvido para visualização dos gradientes por intensidade de cores RGB (Figura 11). Para representar os gradientes por intensidade de cores RGB, foi utilizado o mesmo *script* para visualização da direção dos gradientes, subtraindo-se apenas o módulo “*AutoGlyphs*”.

- **mesh1.net** – Este *script* foi desenvolvido para visualizar a malha de elementos finitos, gerada pelo SPA, com elementos na forma de triângulos, quadriláteros (Figura 12). Para representar a malha sem os resultados, foi utilizado o *script process.net*, subtraindo-se os módulos “AutoColor” e “ColorBar”, incorporando-se os módulos “ShowConnections” que tem como função exibir a malha de elementos finitos, utilizando-se dos valores de posição e conexão. O módulo “Color” utilizado neste *script*, tem como função atribuir a malha uma cor padrão (cor preta).

- **surface.net** – Para visualização da superfície foi desenvolvido este *script* com 5 (cinco) *pipelines* (Figura 13; Figura 14; Figura 15; Figura 16 e Figura 17).

Os *scripts* 01, 02, 03, 04 e 05 estão conectados entre si, por meio dos módulos “Tramitter” e “Receiver”, formando uma “network” dos módulos, que neste caso estão com os nomes: VPA_001, VPA_002, VPA_003, VPA_004 e VPA_005.

O *script* 01, tem como finalidade, carregar os dados dos arquivo “vpa.dx”, associando-se os dados escalares ao módulo “ColorMap” (mapa de cores), que é transmitido para o módulo VPA_001 E VPA_002. No *script* 02 são recebidos os dados, já acondicionados, para o módulo “Inquire” que tem como finalidade testar se os dados são realmente escalares. Então são passos os dados para o “Transmitter” VPA_003. Caso haja erro na leitura dos dados, é abortado imediatamente a passagem de dados para o outro *script*. No *script* 03, os dados são recebidos pelos “Receivers” VPA_001 e VPA_002, e distribuído para os módulos “RubberSheet” e “Isosurface” para geração da superfície deformada com os valores das isolinhas, que são controladas pelo módulo “ScalarList” e “Reset”. Assim os módulos são computados pelo módulo “Switch” o qual repassa os dados formatados para o “Transmitter” VPA_004. No *script* 04 os dados são recebidos pelo “Receiver” VPA_001 e VPA_002. Neste *script*, os dados são repassados para os módulos “AutoGlyph” e “Sample”. Nestes módulos são acondicionadas as informações de cores sobre a superfície deformada. Portanto, no *script* 05 são recebidos todos dados para visualização, pelo módulo “Image”. Caso o usuário queira visualizar uma superfície utilizando diretamente a importação do arquivo “vpa.dx”, deve-se abrir o *OpenDX* e pressionar o botão importar dados (*Import Data*). Para maiores detalhes sobre essa visualização, consultar o manual do usuário, no anexo B. Com o desenvolvimento desses *scripts*, baseados em pipelines, foi possível a visualização dos dados simulados pelo SPA. Os testes do VPA neste caso podem ser caracterizados como os resultados da visualização, obtidos com os dados do modelo de escoamento do aquífero Bauru.

Figura 8 – Script (com pipeline) para visualização do processo.

Figura 9 – Script (pipelines) para visualização das isolinhas.

Figura 10 – Script (pipelines) para visualização da direção dos gradientes.

Figura 11 – script (pipelines) para visualização da intensidade de gradientes por cores RGB.

Figura 12 – Script (pipelines para visualização da malha híbrida.

Figura 13 – Script (pipeline) para visualização de superfícies (módulo 1).

Figura 14 – Script (pipeline) para visualização de superfícies (módulo 2).

Figura 15 – Script (pipeline) para visualização de superfícies (módulo 3).

Figura 16 – Script (pipeline) para visualização de superfícies (módulo 4).

Figura 17 – Script (pipeline) para visualização de superfícies (módulo 5).

5 Visualização e Discussão dos Resultados (Utilização do VPA)

Neste capítulo são descritos os procedimentos necessários para visualização de processos em aquíferos, como resultado deste trabalho. São analisados os resultados das visualizações para análise do comportamento do modelo de escoamento do Aquífero Bauru, na região da área urbana da cidade de São José do Rio Preto. Todas essas figuras aparecem como parte da contribuição do presente trabalho. A finalidade desta ferramenta (VPA) é servir no gerenciamento dos recursos hídricos subterrâneos, a qual demonstrou ser útil, tendo enriquecido sobremaneira a análise dos resultados. Então, são descritos os resultados obtidos com o VPA e também as formas de utilizá-los. A interface (*front end*) do VPA pode ser observada na Figura 18, com informações referentes ao modelo de fluxo do Aquífero Bauru.

Figura 18 – Interface gráfica do VPA.

5.1 Introdução

Para a interpretação dos resultados numéricos obtidos pela simulação dos módulos do SPA, foi desenvolvido o VPA. Com o VPA é possível visualizar os resultados gerados pela simulação de processos de escoamento de fluidos e transporte de poluentes no subsolo, por meio dos módulos do pacote SPA, na forma de dados binários.

Com o VPA é possível visualizar os seguintes resultados de simulação:

Dados do modelo (gerados após VMA):

- Dados da malha em 2D;
- Dados para visualização do processo;
- Dados internos;
- Parâmetros físicos;
- Geologia;

Resultados de interpolação e simulações em regime permanente (gerados após FAS):

- Resultados das equações de escoamento e de transporte;
- Resultados derivados (pós-processamento);

5.2 Requisitos para instalação do SPA (com VPA)

- Processador: Pentium® 1-4; AMD: Duron, Athlon, Athlon XP, Athlon MP; exceto 286, 386 e 486.
- Memória principal (*Main Memory*): 64 MB (memória mínima recomendável) para funcionamento das interfaces gráficas; 128 MB recomendável para funcionamento dos módulos do SPA.
- *Hard Disk* (HD): 4 GB (Para instalar o *SuSE Linux*, SPA)

- Placas de Vídeo (*Graphics cards*): Para uma melhor visualização dos módulos gráficos, usar uma distribuição *GNU/Linux* que suporte os padrões de placas gráficas.

5.3 Sistema Operacional

O VPA foi desenvolvido sistema operacional *SuSE Linux 8.x* (ou versões posteriores 32/64 bits), podendo ser executado em outras distribuições, tais como: *Red Hat*, *Debian*, *Slackware*, *Mandrake* e *Conectiva*. Também é possível utilizar os sistemas operacionais *FreeBSD* e *OpenBSD*, desde que possuam as bibliotecas GTK++ 2.0. Todas distribuições para *Linux* devem ter os ambientes KDE e GNOME.

5.4 Instalação do VPA

Para instalação do VPA, é preciso checar as dependências, conforme descritas no item 6.5, depois é necessário descompactar o arquivo “**vpa.tar**”, executando o seguinte comando: **tar -xf vpa.tar** dentro do diretório **/usr/local/SPA/**. Dentro do diretório **/usr/local/SPA/vpa** deve ser executado o seguinte comando: **make install** que instalará o executável do VPA no diretório **/usr/local/SPA/vpa/bin**. Caso ocorra algum erro na execução do VPA, ler atentamente o anexo A (Padrões de ambiente necessários).

5.5 Dependências de bibliotecas para ambientes KDE e GNOME

O VPA foi desenvolvido com o auxílio das bibliotecas GTK++ 2.0, e utiliza os ambientes KDE e GNOME. Assim, caso ocorrer algum erro na execução do VPA, deverão ser cheçadas as dependências. Para isso é necessário utilizar o comando *ldd*. Em um terminal *xterm* digitar o comando **ldd vpa**.

O comando *ldd* lista as bibliotecas dinâmicas que o VPA precisa para um correto funcionamento de sua interface gráfica. Na Figura 19 é ilustrado a execução do comando *ldd*.

```

[jeq on gandalf] /home/jeq/Desktop/VPA/src > ldd vpa
        libgtk-x11-2.0.so.0 => /usr/lib/libgtk-x11-2.0.so.0 (0x40042000)
        libgdk-x11-2.0.so.0 => /usr/lib/libgdk-x11-2.0.so.0 (0x40295000)
        libatk-1.0.so.0 => /usr/lib/libatk-1.0.so.0 (0x40303000)
        libgdk_pixbuf-2.0.so.0 => /usr/lib/libgdk_pixbuf-2.0.so.0 (0x4031c000)
        libg2c.so.0 => /usr/lib/libg2c.so.0 (0x4032f000)
        libm.so.6 => /lib/libm.so.6 (0x4034d000)
        libpangoxft-1.0.so.0 => /usr/lib/libpangoxft-1.0.so.0 (0x4036f000)
        libpango-1.0.so.0 => /usr/lib/libpango-1.0.so.0 (0x40392000)
        libpango-1.0.so.0 => /usr/lib/libpango-1.0.so.0 (0x4039f000)
        libgobject-2.0.so.0 => /usr/lib/libgobject-2.0.so.0 (0x403d2000)
        libgmodule-2.0.so.0 => /usr/lib/libgmodule-2.0.so.0 (0x40407000)
        libdl.so.2 => /lib/libdl.so.2 (0x4040b000)
        libglib-2.0.so.0 => /usr/lib/libglib-2.0.so.0 (0x4040e000)
        libgcc_s.so.1 => /lib/libgcc_s.so.1 (0x4047a000)
        libc.so.6 => /lib/libc.so.6 (0x40482000)
        libXrandr.so.2 => /usr/X11/lib/libXrandr.so.2 (0x405b8000)
        libXi.so.6 => /usr/X11/lib/libXi.so.6 (0x405bc000)
        libXext.so.6 => /usr/X11/lib/libXext.so.6 (0x405c4000)
        libXft.so.2 => /usr/X11/lib/libXft.so.2 (0x405d3000)
        libXrender.so.1 => /usr/X11/lib/libXrender.so.1 (0x405e6000)
        libfontconfig.so.1 => /usr/X11/lib/libfontconfig.so.1 (0x405ee000)
        libX11.so.6 => /usr/X11/lib/libX11.so.6 (0x40614000)
        libfreetype.so.6 => /usr/lib/libfreetype.so.6 (0x406e2000)
        /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
        libexpat.so.0 => /usr/lib/libexpat.so.0 (0x40734000)
[jeq on gandalf] /home/jeq/Desktop/VPA/src >

```

Figura 19 – Dependências do executável do VPA.

Caso houver alguma dependência de bibliotecas, acusadas durante a verificação (Figura 19), o usuário deverá contatar seu administrador de sistema para instalação das mesmas. Todas bibliotecas necessárias para execução do VPA, estão inclusas no pacote *SuSE Linux 8.x*.

5.6 Instalação do *OpenDX*

Para o funcionamento do VPA, é necessária a instalação do *OpenDX*. O *OpenDX* é distribuído nos termos de *open source development*, que consiste em um pacote de ferramentas para manipular, processar, transformar, visualizar e animar dados em diversos sistemas operacionais. Com este pacote também é possível utilizar ferramentas para processar em paralelo, quando se dispõe de grandes volumes de dados.

Para instalar o *OpenDX*, basta acessar a página www.opendx.org e fazer download dos arquivos de instalação e dos manuais que estão em formato PDF. Eles podem ser encontrados para várias distribuições *GNU/Linux* no formato RPM (*RedHat Package Manager*). Nos testes do VPA, utilizou-se a versão *OpenDX 4.3.0* para *SuSE Linux*.

Para instalação do *OpenDX*, foi utilizado o *YAST2*, com permissão de usuário *root*, para instalação dos arquivos “.rpm”.

Para instalação do pacote do *OpenDX*, basta clicar sobre o arquivo “dx-4.2.0-34.i586.rpm” e automaticamente abrirá o gerenciador de pacotes do *SuSE Linux* sendo possível assim a instalação e verificação das dependências.

Na Figura 20 pode ser observada a instalação do *OpenDX* com o auxílio do gerenciador de programas do *YAST2*.

Figura 20 – Instalação do *OpenDX* com o auxílio do gerenciador do *YAST2*.

5.7 Arquivos de configuração para o VPA

Durante a instalação do pacote SPA em ambiente *Linux (Unix)*, os módulos executáveis devem ser instalados no diretório: */usr/local/SPA/bin*. A variável *path* (no caso de *C-Shell*, definido no arquivo “.cshrc”) deve ser acrescentado com esse endereço. Os arquivos de configuração (*spa.opt*, *cma.kenn*) e os arquivos necessários para visualização e impressão (*.inp), devem ser instalados no diretório.

5.8 Utilização do VPA para Visualização de Processos em Aquíferos

Neste tópico será exemplificado como utilizar o VPA com as ferramentas do *OpenDX*. Após a simulação dos processos em aquíferos, são gerados os arquivos binários aaa, bbb, ccc, e outros, que são usados pelo VPA para visualizar os parâmetros disponíveis.

Neste contexto, serão exemplificadas as principais visualizações geradas pelo VPA, que neste caso são: malhas híbridas (formada por elementos triangulares e quadrilaterais), mapeamento sobre superfícies (*height-fields*), mapeamento de contorno (isolinhas), mapeamento de parâmetros por cores RGB, direção dos gradientes e intensidade dos gradientes.

Para executar o VPA, basta abrir um terminal *xterm*, em seguida digitar **usr/local/SPA/bin/vpa** (em letras minúsculas), dentro do diretório onde estão os arquivos binários gerados pela simulação, conforme exemplificado na Figura 21.

Figura 21 – Execução do VPA em um terminal *xterm*.

Dessa maneira, o VPA exibirá sua interface, que está representada na Figura 22, onde já foram lidos os arquivos binários índices para visualização do parâmetro escolhido.

Todos os exemplos de visualização, descritos neste trabalho, foram obtidos dos resultados da simulação do modelo de escoamento do aquífero Bauru, na cidade de São José do Rio Preto.

Para exemplificar, utilizaremos a visualização da condutividade hidráulica deste modelo. Neste caso, são fornecidas as seguintes informações:

Figura 22 – Interface do VPA com os resultados da simulação.

- **Informações do Projeto:** neste campo são feitas as descrições do projeto simulado: unidade de tempo igual a ano, dados da malha com número de nós igual a 14.149 e número de elementos igual a 18.131, visualizados após a simulação do módulo SITRA (Simulação de Transporte);
- **Dados do Modelo:** neste campo, o usuário poderá escolher como parâmetro, para visualização, os seguintes resultados:
 - Dados da malha em 2D;
 - Dados para visualização do processo;
 - Dados internos;
 - Parâmetros físicos;

- Geologia;

- Resultados da Simulação: neste campo, o usuário poderá escolher como parâmetro, para visualização, os seguintes resultados:

- Resultados das equações de escoamento e de transporte;

- Resultados derivados (pós-processamento);

Neste exemplo foi escolhida a condutividade hidráulica para visualização (escolhida no campo Dados do Modelo). Após a seleção feita pelo usuário, o VPA processa automaticamente um arquivo denominado “vpa.dx”, sendo necessário selecionar o tipo de visualização, por meio dos botões e menus.

Para visualizar este parâmetro, o usuário deverá pressionar o botão “VISUALIZAR PARÂMETRO” (Figura 22) ou selecionar no menu visualizar, o item “VIS Parâmetro” (Figura 23).

VIS Parâmetro	Ctrl+T
VIS Isolinhas	Ctrl+I
VIS Direção dos Gradientes	Ctrl+F
VIS Intensidade dos Gradientes	Ctrl+G
VIS Malha	Ctrl+M

Figura 23 – Seleção do item para visualização do Parâmetro.

Depois de pressionar o botão “VISUALIZAR PARÂMETRO” ou selecionar o item do menu “VISUALIZAR – VIS do Parâmetro”, será aberta automaticamente a janela do *OpenDX*, com a visualização do parâmetro escolhido, que neste caso foi a condutividade hidráulica (Figura 25). Assim o usuário poderá analisar os dados, por meio da legenda de cores (em RGB).

Neste caso, é usado o método de mapeamento por cores, onde o processo de visualização, consiste em associar dados escalares a cores, exibindo-as como indicação dos valores. Esse mapeamento é implementado por meio da indexação de uma tabela de cores (*Colorbar*).

Os valores escalares são usados como índices para a tabela, como indicado na Figura 24.

Figura 24 – Mapeamento de valores escalares por meio de uma tabela de cores.

Esta tabela é um vetor de cores (RGB), e está associada a um intervalo de variação, definido por um valor mínimo (*min*) e um valor máximo (*max*). Apesar de disponível com a ferramenta *OpenDX*, a alteração desses intervalos (*Colorbar*) não está disponível ao usuário.

Os valores escalares são mapeados nesse intervalo: para cada escalar s_i , obtém-se o índice i para uma posição na tabela.

Para gerar boas visualizações com essa técnica é preciso escolher cuidadosamente as entradas na tabela, uma tarefa nem sempre simples. A técnica é aplicável a dados escalares, definidos neste caso em domínio 2D.

Nas Figuras 25 e 26, são demonstradas as visualizações dos parâmetros: condutividade hidráulica e de carga Hidráulica, respectivamente.

Figura 25 – Condutividade Hidráulica do modelo de escoamento do Aquífero Bauru.

Figura 26 – Carga Hidráulica do modelo de escoamento do Aquífero Bauru.

5.9 Visualização das Isolinhas

Para visualização das isolinhas, do contorno (do modelo) e dos poços, o usuário após selecionar os dados nos campos “Dados do Modelo” ou “Resultados da Simulação”, deverá selecionar no menu “Visualizar” o item “VIS Isolinhas” (Figura 27). Assim, será exibida a janela com a visualização das isolinhas (Figura 28).

VIS Parâmetro	Ctrl+T
VIS <u>I</u> solinhas	Ctrl+I
VIS Direção dos Gradientes	Ctrl+F
VIS Intensidade dos <u>G</u> radientes	Ctrl+G
VIS <u>M</u> alha	Ctrl+M

Figura 27 – Seleção do item para visualização das isolinhas, contorno e poços.

Figura 28 – Visualização das isolinhas de Carga Hidráulica do Aquífero Bauru, na cidade de São José do Rio Preto.

5.10 Visualização da Direção dos Gradientes

Para visualização da direção dos gradientes, o usuário deverá seguir os seguintes passos: nos campos “Dados do Modelo” ou “Resultados da Simulação” o usuário deverá escolher o parâmetro desejado. Escolhido o parâmetro, deverá selecionar no menu “Visualizar – VIS Direção dos Gradientes” (Figura 29). Após a seleção, o VPA processa automaticamente o arquivo com os dados a serem visualizados, utilizando o *OpenDX* para visualização da direção dos gradientes (Figura 30).

Após a seleção, o VPA processa automaticamente o arquivo com os dados a serem visualizados, utilizando o *OpenDX* para visualização da direção dos gradientes (Figura 30).

VIS Parâmetro	Ctrl+T
VIS Isolinhas	Ctrl+I
VIS Direção dos Gradientes	Ctrl+F
VIS Intensidade dos Gradientes	Ctrl+G
VIS Malha	Ctrl+M

Figura 29 – Seleção do item para visualização da direção dos gradientes.

Figura 30 – Visualização da direção dos gradientes da carga hidráulica.

Na Figura 30, são apresentados os vetores que representam os gradientes de carga hidráulica. Vale salientar que o gradiente é uma definição matemática que indica a declividade de uma superfície. Dessa forma, sua representação não deve ser confundida com vetores de descarga. Em hidrogeologia, vetores de descarga são regidos pela lei de Darcy: $q = -k\nabla h$.

5.11 Visualização de Gradientes por Intensidade de Cores RGB

Para a visualização com esta opção, o usuário após selecionar os dados nos campos “Dados do Modelo” ou “Dados da Simulação”, deverá selecionar no menu “Visualizar” o item “VIS Intensidade dos Gradientes” (Figura 31). Após a seleção do item “VIS Intensidade dos Gradientes”, será exibida a janela com a visualização dos gradientes. Na Figura 33, é apresentada a visualização do mesmo parâmetro da seção anterior (direção dos gradientes da carga hidráulica), mas agora, por intensidade de cores RGB. As cores mais claras indicam gradientes mais intensos.

⚙ VIS Parâmetro	Ctrl+T
⚙ VIS Isol ^h inhas	Ctrl+I
⚙ VIS Direção dos Gradientes	Ctrl+F
⚙ VIS Intensidade dos Gradientes	Ctrl+G
⚙ VIS Malha	Ctrl+M

Figura 31 – Seleção do item para visualização da Intensidade dos Gradientes.

5.12 Visualização da Malha

O próximo passo será a visualização da malha gerada pelo método dos elementos finitos. Neste caso, a malha é irregular e híbrida (triângulos e quadriláteros como elementos da malha, apresentados na Figura 36). Para a visualização da malha, o usuário deverá selecionar no menu “VISUALIZAR – VIS Malha” (Figura 32).

⚙ VIS Parâmetro	Ctrl+T
⚙ VIS Isol ^h inhas	Ctrl+I
⚙ VIS Direção dos Gradientes	Ctrl+F
⚙ VIS Intensidade dos Gradientes	Ctrl+G
⚙ VIS Malha	Ctrl+M

Figura 32 – Seleção do item para visualização da Malha.

Figura 33 – Visualização dos gradientes de carga hidráulica, por intensidade de cores RGB.

Selecionando este item, será aberta automaticamente a janela do *OpenDX*, com a visualização da malha de elementos finitos, onde a concentração de amarelos corresponde ao refinamento da malha, feito por Oliveira (2002), nas proximidades dos poços de bombeamento.

5.13 Visualização de Superfícies

Para visualizar superfícies com o VPA, o usuário deve pressionar o botão “VISUALIZAR SUPERFÍCIES” na interface do VPA (Figura 22).

Pressionando-se esse botão, será aberta automaticamente a janela do *OpenDX*. Para ilustrar essa visualização, foi escolhido como parâmetro, a carga hidráulica. Neste caso foi escolhido como parâmetro, a condutividade hidráulica (Figura 34).

Figura 34 – Visualização da superfície piezométrica.

Figura 35 – Malha do modelo de escoamento do Aquífero Bauru.

Figura 36 – ZOOM dos elementos da malha (triângulos e quadriláteros).

Após a visualização da superfície desejada, o usuário pode inserir eixos como também rotacionar a superfície. Para isso, é necessário utilizar as ferramentas de formação do *OpenDX*. Será descrita a ferramenta “*VIEW CONTROL*” (Controle da Visualização) de acordo com os seguintes passos:

- No menu *Options* (opções), ilustrado na Figura 37, da janela de visualização, escolher o item “*VIEW CONTROL*” (Controle da Visualização). Com o “*VIEW CONTROL*”, o usuário pode, dentre outras coisas, mudar o ponto de vista, girar o objeto, aproximar e afastar o objeto. A Figura 39 ilustra a rotação da superfície piezométrica para visualização, em detalhes, dos cones de depressão.

- Para mudar o ponto de vista da visualização, o usuário deverá selecionar o botão “*SET VIEW*” na janela “*VIEW CONTROL*”, conforme demonstrado na Figura 38.

Figura 37 – Menu para escolha do “VIEW CONTROL”.

Figura 38 – Janela do “VIEW CONTROL”.

Figura 39 – Rotação da superfície para visualização dos cones de depressão.

Nesta opção pode-se selecionar 7 comandos de vista diferentes, abrangendo uma única face do objeto e para cada uma dessas 7 vistas, existe uma visão levemente oblíqua que é denominada “*off views*”. Na Figura 40, pode ser observada a visualização da região de mais concentração de poços, com o auxílio da ferramenta “*off views*”, neste caso foi utilizada a opção “*BOTTOM*”.

Figura 40 – Visualização da região de mais concentração de poços.

Observação e recomendação: quando muda-se o ponto de vista de um objeto, o programa visual é executado automaticamente, e atualizando a visualização.

- Para girar um objeto na janela de visualização, o usuário deverá clicar sobre o botão “*MODE*” da janela “*VIEW CONTROL*” (Figura 38) e mudar a opção para “*ROTATE*”. Na utilização do comando “*ROTATE*”, o usuário poderá observar no canto

inferior direito da janela, o ícone com o conjunto de eixos. Assim o usuário poderá girar o objeto em 2 e 3 dimensões.

Observação e recomendação: para que o objeto mova conforme o movimento do cursor sobre a tela, continuamente, o usuário deverá utilizar a opção “*EXECUTE ON CHANGE*” do menu “*EXECUTE*” na janela de visualização. Neste método, a imagem só é atualizada quando o botão do mouse é solto, caso não ocorra esse evento, o usuário deverá verificar se o menu “*EXECUTE*” esteja ativo.

- Para aproximar, afastar e aumentar os objetos, o usuário deverá selecionar a opção “*Zoom*” do botão “*MODE*”, da janela “*VIEW CONTROL*” (Figura 38). Assim, basta pressionar com o botão esquerdo do mouse, que aparecerá um retângulo para o enquadramento da imagem. O resultado deste procedimento será o aumento ou diminuição do objeto, de acordo com a necessidade do usuário.

Figura 41 – Visualização da área onde estão concentrados a maioria dos poços.

Observação e recomendação: caso houver problemas como o desaparecimento da imagem, basta pressionar o botão “*RESET*” da janela “*VIEW CONTROL*” (Figura 38).

Agora, serão descritas as configurações básicas do “*AUTOAXES*” do *OpenDX*. Para visualizar as dimensões do objeto, o usuário deverá selecionar no menu “*OPTIONS*” (Figura 37) a opção “*AUTOAXES*”. Será aberta uma janela, como pode ser observado na Figura 42.

Figura 42 – Janela para configuração dos Eixos (“*AUTOAXES*”).

Para visualizar os eixos, o usuário deverá ativar o botão “*AUTOAXES ENABLE*”. Habilitando o botão “*AUTOAXES*”, o usuário poderá configurar várias opções, tais co-

mo: escolher fonte (*Font*), modificar a escala do texto (*Label scale*), modificar as cores da grade (*Grid*), do eixo (*Ticks*), do plano do gráfico (*Background*).

Observações e recomendações: para visualizar melhor os eixos, o usuário deverá mudar o ponto de vista para “*DIAGONAL*”, no botão “*SET VIEW*” do “*VIEW CONTROL*” (Figura 38). Basicamente estes são os principais comandos para manipulação das ferramentas do *OpenDX*.

Este manual foi desenvolvido com o objetivo de servir como referência para usuários que necessitam visualizar dados gerados pelos módulos de simulação do SPA. Descreve passo a passo como visualizar os dados do modelo e resultados de simulação gerados pelo SPA. Para validação do VPA como ferramenta para visualização, foram usados os dados e resultados do modelo de escoamento do Aquífero Bauru, da cidade de São José do Rio Preto (Oliveira, 2002).

Assim, os exemplos discutidos neste manual demonstram a funcionalidade do VPA (com o auxílio do *OpenDX*) como ferramenta para visualização de processos em aquíferos em ambiente *SuSE Linux* (Manzano, 2000).

6 Conclusão

Com a avaliação comparativa das principais técnicas de visualização científica e também de ferramentas disponíveis para visualização de processos em aquíferos, foi possível o desenvolvimento do VPA.

Esse visualizador foi desenvolvido para ambientes *KDE* e *GNOME* e integrado ao sistema operacional *SuSE Linux*. Como ambiente de programação, o *Glade 2* mostrou-se adequado, principalmente, por utilizar a linguagem C, a mesma que foi programada a biblioteca *DtnLIB*.

A integração do SPA com o VPA foi realizada pela leitura dos resultados de simulação do SPA, que são armazenados em arquivos binários pela biblioteca *DtnLib*. Assim foi possível escrever o arquivo “vpa.dx”. Para visualização deste arquivo, foram desenvolvidos alguns scripts no editor de programação visual (*VPE*) do *OpenDX*. Esses *scripts*, baseados em *networks* e *pipelines*, mostrou-se robusto como ferramenta para construção dos diversos processos de visualização.

Como avaliação dos recursos disponibilizados pelo VPA, foram realizadas várias visualizações, obtidas dos resultados gerados do modelo de escoamento do Aquífero Bauru, na cidade de São José do Rio Preto.

Como referência para usuários interessados na visualização de dados gerados pelos módulos de simulação do SPA, foi elaborado um manual como parte integrante deste trabalho.

Com os resultados obtidos das visualizações realistas, bem como das características de utilização do VPA, conclui-se que o módulo desenvolvido apresenta-se como uma ferramenta estável, de fácil manipulação e generalista (em relação a resultados de diferentes simulações).

7 Referências Bibliográficas

Alves, W. P. (2002). Programação Gráfica em Delphi 6, São Paulo, Editora Érica Ltda., 244 p.

ANSI (1985a). Computer Graphics – Graphical Kernel System (GKS) Function Description, Technical Report, American National Standards Institute for Information Processing Systems, USA.

_____. (1985b). Computer Graphics - Graphical Kernel System (PHIGS) Function Description, Technical Report, American National Standards Institute for Information Processing Systems, USA.

AVS – Advanced Visual Systems. Disponível em: http://www.avs.com/index_wf.html. Acesso em: 10/08/2003. EUA.

Barr, M. (1999). Programming Embedded Systems in C and C++, O'Reilly & Associates Inc., 191 pp., ISBN: 1-56592-354-5.

Blaz, R.; Domingues, M. O.; Mendes, O. Jr. (2003). Introdução à entrada de dados no *OpenDX*: formatos .dx, general e grib, Technical Report INPE-9558-NTC/351, INPE, 2003.

Bodammer, Frank et al. (2002a). SuSE Linux 8.2 – Administration Guide, SuSE Linux AG., Germany, 360 p.

_____. (2002b). SuSE Linux 8.2 – User Guide, SuSE Linux AG., Germany, 360 p.

Brodie, K. W. et al. (1992). Scientific Visualization Techniques and Applications, Springer-Verlag.

Cameron, D. (1999). GNU Emacs Pocket Reference, O'Reilly & Associates Inc., 58 pp. ISBN: 1-56592-496-7.

Conner, D. B. et al. (1992). Three - Dimensional Widgets, Proceedings of the 1992 Workshop on Interactive 3D Graphics, 183-188 p.

Dougherty, D.; Robbins, A. (1997). Sed & Awk – Second Edition, O'Reilly & Associates Inc., 432 pp. ISBN: 1-56592-225-5.

Earnshaw, R. A.; Watson, D. (1993). Animation and Scientific Visualization – Tools and Applications, Academic Press Harcourt Brace & Company Publishers, 285 p.

Eckel, Bruce (2000). Thinking in C++ - Second Edition, Upper Saddle River - New Jersey, Prentice Hall, 636 p. Disponível no site: <http://mindview.net/Books>. Acesso em: 07/03/2003.

Elvins, T. T. (2004). Introduction to Volume Visualization: Imaging Multidimensional Scientific Data, SIGGRAPH'94, Course Notes.

Herndon, K. P.; Dam, A. Van; Gleicher, M. (1994). Workshop on the Challenges of 3D Interaction, SIGCHI Bulletin, v. 26, n. 4, 1-9 p.

Feichas, C. H. L. (2003). Usando o DDD para depurar programas, Revista do Linux, ano IV, n. 47, p. 48-49.

Gallagher, R. S. (1995). Computer Visualization, Graphics Techniques for Scientific and Engineering Analysis, CRC Press.

GEF (2002). Environmental protection and sustainable development of the Guarani aquifer system project. Report 23490, GEF/BIRD.

GEF (2003). Sistema Acuífero Guaraní GEF-BIRD-OEA perfil preliminar del proyecto piloto de gestión y protección en Ribeirão Preto, Brasil. Report Sep 19 2003, GEF/BIRD.

Glickstein, B. (1997). Writing GNU Emacs Extensions, O'Reilly & Associates Inc., 219 pp. ISBN: 1-56592-261-1.

Griffith, Arthur (2002). GCC: The Complete Reference, McGraw-Hill Companies, Inc. (USA), New York, 647 pp.

IBM Visualization Data Explorer (1997). Programmer's Reference, Third Edition – Ref. N.º SC38-0497 Disponível em: <http://www.opendx.org/support.html>. Acesso em: 07/03/03.

_____. (1997). User's Guide, Third Edition – Ref. N.º SC38-0496. Disponível em: <http://www.opendx.org/support.html>. Acesso em: 07/03/03.

_____. (1997). User's Reference, Third Edition – Ref. N.º SC38-0486. Disponível em: <http://www.opendx.org/support.html>. Acesso em: 07/03/03.

Kilgard, M. J. (1994). OpenGL and X - Part 1 - An Introduction, Technical report, Silicon Graphics Inc. – SGI.

_____. (1996). The OpenGL Utility Toolkit (GLUT) – Programming Interface API Version 3, Technical report, Silicon Graphics Inc. – SGI.

Kochan, S. G.; Wood, P. (2003). Unix Shell Programming, Third Edition, Sams Publishing, 456 pp. ISBN: 0-672-32490-3.

Loukides, M.; Oram, A. (1997). Programming with GNU Software, O'Reilly & Associates Inc., 253 pp. ISBN: 1-56592-112-7.

Lozano, F. (2003). Interfaces gráficas em XML, Revista do Linux, ano IV, n. 47, p. 54-60.

Lucas, M. (2002). *Absolute BSD – The Ultimate Guide to FreeBSD*, Publishers Group West, Berkeley, USA, 524 pp.

MacKenzie, D.; Elliston, B.; Demaille, A. (2002). *Autoconf – Creating Automatic Configuration Scripts*, version 2.57, Published by the Free Software Foundation Inc., 59 Temple Place – Suíte 330, Bonton, MA 02111-1307, USA, 232 pp.

MacKenzie, D.; Tromeey, T. (2003). *GNU Automake*, version 1.7.8, Published by the Free Software Foundation Inc., 59 Temple Place – Suíte 330, Bonton, MA 02111-1307, USA, 90 pp.

Manzano, José Augusto N. G. (2000). *SuSE Linux 6.4 – Guia Prático de Orientação*, São Paulo, Editora Érica Ltda., 240 p.

Maxwell, S. (2002). *UNIX System Administration: A Beginner's Guide*, McGraw-Hill Companies Inc., 676 pp.

McCormick, B. H.; Defanti, T. A.; Brown, M. D. (1987). *Visualization in Scientific Computing*, *Computer Graphics (special issue)*, v. 21, n. 6.

Mitchell, M.; Oldham, J.; Samuel, A. (2001). *Advanced Linux Programming*, New Riders Publishing, 368 pp., ISBN: 0-7357-1043-0.

Minghim, Rosane; Oliveira, Maria C. F. (1997). *Uma Introdução à Visualização Computacional*, XVI Jornada de Atualização em Informática, Anais / XVII Congresso da Sociedade de Computação, Editado por Cláudia Maria Bauzer Medeiros – Brasília, UnB, 494 p.

NAG's IRIS Explorer. Visualization Technical Reports.

Disponível em: <http://www.nag.co.uk/doc/TechRep/ietr.html>. Acesso em: 08/03/2003.

Neider, J.; Mason, W. and Davis, T. (1993). *OpenGL Programming Guide*, Addison Wesley.

_____. (2000). OpenGL Programming Guide, Addison Wesley.

Newham, C.; Rosenblatt, B. (1998). Learning the bash Shell, Second Edition, O'Reilly & Associates Inc., 334 pp., ISBN: 1-56592-347-2.

Olczak, A. (2000). The Korn Shell: Unix and Linux Programming Manual – Third Edition, Addison Wesley, 480 pp., ISBN: 0-201-67523-4.

Oliveira, J. N. (2002). Ferramental de gestão de águas subterrâneas para a cidade de São José do Rio Preto, SP. São Carlos, 2002. 117p. Tese (Doutorado) - Escola de Engenharia de São Carlos, Universidade de São Paulo.

Oliveira, P. H. B. (1999). As Ferramentas de Desenvolvimento do Linux, Developers' Magazine, ano IV, n. 38, p. 36-38.

Oualline, S. (1997). Practical C Programming – Third Edition, O'Reilly & Associates Inc., 504 pp., ISBN: 1-56592-306-5.

Ousterhout, J. K. (1994). Tcl and the Tk Toolkit, Addison-Wesley Publishing Company Inc., 390 pp.

Purcell, J. (1997). Linux Complete Command Reference, Red Hat Software, Inc., 1527 pp., ISBN: 0-672-31104-6.

Schröeder, Will; Martin, Ken; Lorensen, Bill (1996). The Visualization Toolkit, New Jersey, Prentice Hall, 826 p.

_____. (1997). The Visualization Toolkit - 2nd Edition – An Object-Oriented Approach to 3D Graphics, Prentice-Hall International (UK) Limited (London), 645 p.

Schwarz, M. et al. (2002). Multitool Linux – Practical Uses for Open Source Software, Addison-Wesley Publishing Company Inc., 576 pp. ISBN: 0-201-73420-6.

Scientific Software Group (2004). Environmental and Water Resource Software Catalog. Disponível em: <http://www.scientificsoftwaregroup.com>. Acesso em: 11/03/2004.

Segal, M. and Akeley, K. (1994). The OpenGL Graphics Interface, Technical report, Silicon Graphics Inc. Disponível em: http://www.opengl.org/developers/documentation/white_papers/oglGraphSys/opengl.html. Acesso em: 07/03/2003.

_____. (1996). The OpenGL Graphics Interface, Technical report, Silicon Graphics Inc. Disponível em: http://www.opengl.org/developers/documentation/white_papers/opengl/index.html. Acesso em: 07/03/03.

Siever, E. (2000). Linux in a Nutshell – Third Edition, O'Reilly & Associates Inc., ISBN: 0-596-00025-1.

Silva, E. C. (1997). Introdução ao IBM Data Explorer (IBM OpenDX), Technical Report, CENAPAD-SP, Campinas, São Paulo.

Stallman, R.; MacGrath, R.; Smith, P. (2002). GNU Make – A Program for Directing Recompilation, version 3.80, Published by the Free Software Foundation Inc., 59 Temple Place – Suíte 330, Bonton, MA 02111-1307, USA, 176 pp. ISBN: 1-882114-81-7.

Thompson, D.; Braun, J.; Ford, R. (2001). Paths to Visualization. Visualization and Imagery Solutions Inc., EUA.

Upson, C.; Faulhaber, T.; Kammins, D.; Laidlaw, D.; Schlegert, D.; Vroom, J.; Gurwitz, R.; Dam, A. Van: Student Member IEEE Computer Society, and Markus Gross, Member IEEE (1992). The Application Visualization System: A Computational Environment for Scientific Visualization – IEEE Computer Graphics and Applications, v. 09, n. 07, pp. 30-42.

U. S. Army TEC – Topographic Engineering Center (2004).

Disponível em: <http://www.tec.army.mil/TD/tvd/survey/apE.html>. Acesso em: 05/01/2004.

Young, Mark; Argiro, Danielle; Kubica, Steven (2000). Cantata: The Visual Programming Environment for the Khoros System, Khoral Research, Inc., USA, 9 p. Disponível em: <http://www.khoral.com/khoros/cantata/>. Acesso em: 07/03/03.

Wall, K.; Watson, M.; Whitis, M. (1999). Linux Programming Unleashed, Sams Publishing, 818 pp., ISBN: 0-672-31607-2.

WENDLAND, E., (2003). Desenvolvimento do pacote computacional SPA (Simulação de Processos em Aquíferos), Relatório de Pesquisa, Processo FAPESP 02/09696-3, Departamento de Hidráulica e Saneamento, Escola de Engenharia de São Carlos, Universidade de São Paulo, 78 p.

Williams, Sam (2002). Free as in Freedom – Richard Stallman’s Crusade for Free Software, O’Reilly & Associates Inc., 240 pp. ISBN: 0-596-00287-4.

Zeller, A. (2000). Debugging with DDD – User’s Guide and Reference Manual, Universität Passau, Lehrstuhl für Software-Systeme, Innstraße 33, D-94032 – Passau, Germany, 222 pp. Disponível em: <http://www.gnu.org/software/ddd/>. Acesso em: 07/03/2003.

ANEXO A

Padrões de ambiente necessários para execução do VPA

Para o bom desempenho dos módulos do SPA, devem ser definidos os seguintes padrões de ambiente:

- A variável de ambiente SPABin deve ser definida como o diretório, no qual estão armazenados os módulos executáveis do pacote SPA (normalmente: /usr/local/SPA/bin). Caso essa variável não tenha sido definida, podem ocorrer problemas quando os módulos são inicializados a partir do módulo XPLT.
- A variável de ambiente SPAGlob deve ser definida como o diretório no qual estão armazenados todos os arquivos de configuração *.inp a serem usados pelos módulos do pacote SPA (normalmente: /usr/local/SPA/config). Caso essa variável não tenha sido definida, podem ocorrer problemas durante a impressão no módulo XPLT (os arquivos *.inp são necessários) ou na utilização do módulo CMA (cma.kenn é necessário).
- A variável de ambiente SPApriv deve ser definida como o diretório (p. e. ~/config), no qual estão armazenados todos os arquivos de configuração pessoal do usuário (spa.opt, cma.kenn e plogeo.ini). Nesse diretório pode ser copiado o arquivo de parâmetros cma.kenn (do diretório global) e alterado de acordo com as necessidades do usuário. Além disso, nesse diretório são armazenados os arquivos opcionais (spa.opt e plogeo.ini) alterados pelo usuário.
- A variável de ambiente SPATerm deve ser definida de acordo com o tipo de máquina utilizada (p. e. aixterm, hpterm, dtterm e xterm), no qual todos os módulos acionados pelo XPLT podem ser iniciados.
- A variável de ambiente SPABrowser deve ser definida como um browser clássico default (p. e. netscape). O browser é acionado através do botão de ajuda.
- A variável de ambiente SPAManual deve ser definida como o diretório, no qual o manual de utilização do pacote SPA está armazenado (diretório com o arquivo index.htm), geralmente /SPA/Manual.

Assim, as variáveis de ambiente podem ser definidas como:

Em ambiente Unix ou GNU/Linux usando C-Shell (csh):

- `setenv SPABin /usr/local/SPA/bin`
- `setenv SPAGlob /usr/local/SPA/config`
- `setenv SPApriv /home/Lhc/config`
- `setenv SPATerm xterm`
- `setenv SPABrowser netscape`
- `setenv SPAManual /usr/local/SPA/Manual`

Em ambiente GNU/Linux usando o ba-Shell (bash – Burn Again Shell (Newham, 1998)):

- `export SPABin = "/usr/local/SPA/bin"`
- `export SPAGlob = "/usr/local/SPA/config"`
- `export SPApriv = "/home/Lhc/config"`
- `export SPATerm = "xterm"`
- `export SPABrowser = "netscape"`
- `export SPAManual = "/usr/local/SPA/Manual"`

Nesse caso, assume-se que os módulos executáveis estão armazenados no diretório `"/usr/local/SPA/bin"`, o manual do pacote SPA em `"/usr/local/SPA/manual"`, os arquivos de configuração global e `*.inp` em `"/usr/local/SPA/config"` e as configurações pessoais do usuário, no diretório `"/home/Lhc/config"`. Os respectivos comandos do Shell devem ser inseridos no arquivo `“.cshrc”` (csh) ou `“.bashrc”` (bash), para que elas não tenham de ser redefinidos a cada início de um terminal (Shell).

ANEXO B

Visualização de Superfícies com OpenDX

Neste anexo, serão descritos alguns métodos para visualização de superfícies. Como já descrito, o VPA escreve um arquivo nomeado “vpa.dx”. Para a visualização de superfícies, será necessário executar o arquivo diretamente no OpenDX.

Assim, serão descritas as principais ferramentas para visualização e manipulação com o OpenDX, que são o “VIEW CONTROL” (Controle da Visualização) e o “AUTOAXES” (Exibição dos Eixos). O usuário deverá seguir os seguintes passos:

- Executar o “OpenDX” no diretório onde estão os resultados (arquivo vpa.dx). Assim, o OpenDX abrirá a janela principal (Figura 43).

Figura 43 – Janela inicial do *OpenDX*.

Após carregar o OpenDX, o usuário deverá escolher o botão que tiver a opção “Import Data” (importar arquivo), onde será aberta outra janela para escolha do arquivo, que neste caso será “vpa.dx” (Figura 44).

Figura 44 – Janela para abertura do arquivo “vpa.dx”.

- Acionar a caixa “Data file name” (Figura 44), para abrir o arquivo “vpa.dx” (Figura 45).

Figura 45 – Janela para seleção do arquivo “vpa.dx”.

- Após a escolha do arquivo, serão ativados os botões “Test Import” (Testar importação dos dados) e “Visualize Data” (Visualizar dados). O usuário deverá esco-

lher o botão “Visualize Data” (Visualizar dados). A superfície será visualizada como pode ser observado na Figura 46.

Figura 46 – Visualização da superfície do terreno.

Após a visualização da superfície do terreno, o usuário pode inserir eixos como também rotacionar a superfície. Para isso, é necessário utilizar as ferramentas de formatação do OpenDX. Será descrita a ferramenta “VIEW CONTROL” (Controle da Visualização) de acordo com os seguintes passos:

- No menu Options (opções), ilustrado na Figura 47, da janela de visualização, escolher o item “VIEW CONTROL” (Controle da Visualização). Com o “VIEW CONTROL”, o usuário pode, dentre outras coisas, mudar o ponto de vista, girar o objeto, aproximar e afastar o objeto.

Figura 47 – Menu para escolha do “VIEW CONTROL”.

- Para mudar o ponto de vista da visualização, o usuário deverá selecionar o botão “SET VIEW” na janela “VIEW CONTROL” (Figura 48).

Figura 48 – Janela do “VIEW CONTROL”.

Nesta opção pode-se selecionar 7 comandos de vista diferentes, abrangendo uma única face do objeto e para cada uma dessas 7 vistas, existe uma visão levemente oblíqua que é denominada “*off views*”. Observação e recomendação: quando se muda o ponto de vista de um objeto, o programa visual é executado automaticamente, atualizando a visualização.

- Para girar um objeto na janela de visualização, o usuário deverá clicar sobre o botão “MODE” da janela “VIEW CONTROL” (Figura 48) e mudar a opção para “ROTATE”. Na utilização do comando “ROTATE”, o usuário poderá observar no canto inferior direito da janela, o ícone com o conjunto de eixos. Assim o usuário poderá girar o objeto em 2 e 3 dimensões.

Observação e recomendação: para que o objeto mova conforme o movimento do cursor sobre a tela, continuamente, o usuário deverá utilizar a opção “EXECUTE ON CHANGE” do menu “EXECUTE” na janela de visualização.

Neste método, a imagem só é atualizada quando o botão do mouse é solto. Caso não ocorra esse evento, o usuário deverá verificar se o menu “EXECUTE” está ativo. Outra observação importante é a velocidade de resposta da rotação contínua, pois varia muito de máquina para máquina.

- Para aproximar, afastar e aumentar os objetos, o usuário deverá selecionar a opção “ZOOM” do botão “MODE”, da janela “VIEW CONTROL” (Figura 48). Assim, basta pressionar com o botão esquerdo do mouse, que aparecerá um retângulo para o enquadramento da imagem. O resultado deste procedimento será o aumento ou diminuição do objeto, de acordo com a necessidade do usuário.

Observação e recomendação: caso houver problemas, como o desaparecimento da imagem, basta pressionar o botão “RESET” da janela “VIEW CONTROL” (Figura 48).

Agora serão descritas as configurações básicas do “*AUTOAXES*” do *OpenDX*. Para visualizar as dimensões do objeto, o usuário deverá selecionar no menu “*OPTIONS*” (Figura 47) a opção “*AUTOAXES*”. Será aberta uma janela (Figura 49).

Para visualizar os eixos, o usuário deverá ativar o botão “*AUTOAXES ENABLE*”. Habilitando o botão “*AUTOAXES*”, o usuário poderá configurar várias opções, tais como: escolher fonte (*Font*), modificar a escala do texto (*Label scale*), modificar as cores da grade (*Grid*), do eixo (*Ticks*), do plano do gráfico (*Background*).

Observações e recomendações: para visualizar melhor os eixos, o usuário deverá mudar o ponto de vista para “*DIAGONAL*”, no botão “*SET VIEW*” do “*VIEW CONTROL*” (Figura 48). Basicamente estes são os principais comandos para manipulação das ferramentas do *OpenDX*.

Figura 49 – Janela para configuração do Eixos (“AUTOAXES”).

Este anexo tem como o objetivo, servir como referência para usuários que necessitam visualizar dados gerados pelos módulos de simulação do SPA. Descreve passo a passo como visualizar os dados do modelo e resultados de simulação gerados pelo SPA.

Os exemplos discutidos neste anexo, ilustram a utilização do VPA. Sendo assim, é desenvolvida uma metodologia para utilização das principais ferramentas *OpenDX*, utilizando o sistema operacional *SuSE Linux*.

ANEXO C
GPL (Licença Pública Geral)

LICENÇA PÚBLICA GERAL GNU

Versão 2, junho de 1991

This is an unofficial translation of the GNU General Public License into Brazilian Portuguese. It was not published by the Free Software Foundation, and does not legally state the distribution terms for software that uses the GNU GPL – only the original English text of the GNU GPL does that. However, we hope that this translation will help Brazilian Portuguese speakers understand the GNU GPL better.

Esta é uma tradução não-oficial da Licença Pública Geral GNU (“GPL GNU”) para o português do Brasil. Ela não foi publicada pela Free Software Foundation, e legalmente não afirma os termos de distribuição de software que utiliza a GPL GNU – apenas o texto original da GPL GNU, em inglês, faz isso. Contudo, esperamos que esta tradução ajude aos que utilizam o português do Brasil a entender melhor a GPL GNU.

Copyright (C) 1989, 1991 Free Software Foundation, Inc. 675 Mass Ave,
Cambridge, MA 02139, USA.

A qualquer pessoa é permitido copiar e distribuir cópias desse documento de licença, desde que sem qualquer alteração.

Introdução

As licenças de muitos software são desenvolvidas para restringir sua liberdade de compartilhá-lo e mudá-lo. Contrária a isso, a Licença Pública Geral GNU pretende garantir sua liberdade de compartilhar e alterar software livres – garantindo que o software será livre e gratuito para os seus usuários. Esta Licença Pública Geral aplica-se à maioria dos software da Free Software Foundation e a qualquer outro programa cujo autor decida aplicá-la. Alguns outros software da FSF são cobertos pela Licença Pública Geral de Bibliotecas, no entanto, você pode aplicá-la também aos seus programas.

Quando nos referimos a software livre, estamos nos referindo a liberdade e não a preço. Nossa Licença Pública Geral foi desenvolvida para garantir que você tenha a liberdade de distribuir cópias de software livre (e cobrar por isso, se quiser); que você receba o código-fonte ou tenha acesso a ele, se quiser; que você possa mudar o software ou utilizar partes dele em novos programas livres e gratuitos; e que você saiba que pode fazer tudo isso.

Para proteger seus direitos, precisamos fazer restrições que impeçam a qualquer um negar estes direitos ou solicitar que você deles abdique. Estas restrições traduzem-se em certas responsabilidades para você, se você for distribuir cópias do software ou modificá-lo.

Por exemplo, se você distribuir cópias de um programa, gratuitamente ou por alguma quantia, você tem que fornecer aos recebedores todos os direitos que você possui. Você tem que garantir que eles também recebam ou possam obter o código-fonte. E você tem que mostrar-lhes estes termos para que eles possam conhecer seus direitos.

Nós protegemos seus direitos em dois passos: (1) com copyright do software e (2) com a oferta desta licença, que lhe dá permissão legal para copiar, distribuir e/ou modificar o software.

Além disso, tanto para a proteção do autor quanto a nossa, gostaríamos de certificar-nos que todos entendam que não há qualquer garantia nestes software livres. Se o software é modificado por alguém mais e passado adiante, queremos que seus recebedores saibam que o que eles obtiveram não é original, de forma que qualquer problema introduzido por terceiros não interfira na reputação do autor original.

Finalmente, qualquer programa é ameaçado constantemente por patentes de software. Queremos evitar o perigo de que distribuidores de software livre obtenham patentes individuais, o que tem o efeito de tornar o programa proprietário. Para prevenir isso, deixamos claro que qualquer patente tem que ser licenciada para uso livre e gratuito por qualquer pessoa, ou então que nem necessite ser licenciada.

Os termos e condições precisas para cópia, distribuição e modificação se encontram abaixo:

LICENÇA PÚBLICA GERAL GNU

TERMOS E CONDIÇÕES PARA CÓPIA, DISTRIBUIÇÃO E MODIFICAÇÃO

0. Esta licença se aplica a qualquer programa ou outro trabalho que contenha um aviso colocado pelo detentor dos direitos autorais informando que aquele pode ser distribuído sob as condições desta Licença Pública Geral. O “Programa” abaixo refere-se a qualquer programa ou trabalho, e “trabalho baseado no Programa” significa tanto o Programa em si como quaisquer trabalhos derivados, de acordo com a lei de direitos autorais: isto quer dizer um trabalho que contenha o Programa ou parte dele, tanto originalmente ou com modificações, e/ou tradução para outros idiomas. (Doravante o processo de tradução está incluído sem limites no termo “modificação”.) Cada licenciado é mencionado como “você”.

Atividades outras que a cópia, a distribuição e modificação não estão cobertas por esta Licença; elas estão fora de seu escopo. O ato de executar o Programa não é restringido e o resultado do Programa é coberto apenas se seu conteúdo contenha trabalhos baseados no Programa (independentemente de terem sido gerados pela execução do Programa). Se isso é verdadeiro depende do que o programa faz.

1. Você pode copiar e distribuir cópias fiéis do código-fonte do Programa da mesma forma que você o recebeu, usando qualquer meio, desde que você conspicua e apropriadamente publique em cada cópia um aviso de direitos autorais e uma declaração de inexistência de garantias; mantenha intactas todos os avisos que se referem a esta Licença e à ausência total de garantias; e forneça a outros recebedores do Programa uma cópia desta Licença, junto com o Programa.

Você pode cobrar pelo ato físico de transferir uma cópia e pode, opcionalmente, oferecer garantia em troca de pagamento.

2. Você pode modificar sua cópia ou cópias do Programa, ou qualquer parte dele, assim gerando um trabalho baseado no Programa, e copiar e distribuir essas modificações ou trabalhos sob os termos da seção 1 acima, desde que você também se enquadre em todas estas condições:

a) Você tem que fazer com que os arquivos modificados levem avisos proeminentes afirmando que você alterou os arquivos, incluindo a data de qualquer alteração.

b) Você tem que fazer com que quaisquer trabalhos que você distribua ou publique, e que integralmente ou em partes contenham ou sejam derivados do Programa ou de suas partes, sejam licenciados, integralmente e sem custo algum para quaisquer terceiros, sob os termos desta Licença.

c) Se qualquer programa modificado normalmente lê comandos interativamente quando executados, você tem que fazer com que, quando iniciado tal uso interativo da forma mais simples, seja impresso ou mostrado um anúncio de que não há qualquer garantia (ou então que você fornece a garantia) e que os usuários podem redistribuir o programa sob estas condições, ainda informando os usuários como consultar uma cópia desta Licença. (Exceção: se o Programa em si é interativo mas normalmente não imprime estes tipos de anúncios, seu trabalho baseado no Programa não precisa imprimir um anúncio.)

Estas exigências aplicam-se ao trabalho modificado como um todo. Se seções identificáveis de tal trabalho não são derivadas do Programa, e podem ser razoavelmente consideradas trabalhos independentes e separados por si só, então esta Licença, e seus termos, não se aplicam a estas seções quando você distribui-las como trabalhos em separado. Mas quando você distribuir as mesmas seções como parte de um todo que é trabalho baseado no Programa, a distribuição como um todo tem que se enquadrar nos termos desta Licença, cujas permissões para outros licenciados se estendem ao todo, portanto também para cada e toda parte independente de quem a escreveu.

Desta forma, esta seção não tem a intenção de reclamar direitos ou contestar seus direitos sobre o trabalho escrito completamente por você; ao invés disso, a intenção é a de exercitar o direito de controlar a distribuição de trabalhos, derivados ou coletivos, baseados no Programa.

Adicionalmente, a mera adição ao Programa de outro trabalho não baseado no Programa (ou de trabalho baseado no Programa) em um volume de armazenamento ou meio de distribuição não faz o outro trabalho parte do escopo desta Licença.

3. Você pode copiar e distribuir o Programa (ou trabalho baseado nele, conforme descrito na Seção 2) em código-objeto ou em forma executável sob os termos das Seções 1 e 2 acima, desde que você faça um dos seguintes:

a) O acompanhe com o código-fonte completo e em forma acessível por máquinas, que tem que ser distribuído sob os termos das Seções 1 e 2 acima e em meio normalmente utilizado para o intercâmbio de software; ou,

b) O acompanhe com uma oferta escrita, válida por pelo menos três anos, de fornecer a qualquer um, com um custo não superior ao custo de distribuição física do material, uma cópia do código-fonte completo e em forma acessível por máquinas, que tem que ser distribuído sob os termos das Seções 1 e 2 acima e em meio normalmente utilizado para o intercâmbio de software; ou,

c) O acompanhe com a informação que você recebeu em relação à oferta de distribuição do código-fonte correspondente. (Esta alternativa é permitida somente em distribuição não comerciais, e apenas se você recebeu o programa em forma de código-objeto ou executável, com oferta de acordo com a Subseção b acima.)

O código-fonte de um trabalho corresponde à forma de trabalho preferida para se fazer modificações. Para um trabalho em forma executável, o código-fonte completo significa todo o código-fonte de todos os módulos que ele contém, mais quaisquer arquivos de definição de "interface", mais os "scripts" utilizados para se controlar a compilação e a instalação do executável. Contudo, como exceção especial, o código-fonte

distribuído não precisa incluir qualquer componente normalmente distribuído (tanto em forma original quanto binária) com os maiores componentes (o compilador, o "kernel" etc.) do sistema operacional sob o qual o executável funciona, a menos que o componente em si acompanhe o executável.

Se a distribuição do executável ou código-objeto é feita através da oferta de acesso a cópias de algum lugar, então ofertar o acesso equivalente a cópia, do mesmo lugar, do código-fonte equivale à distribuição do código-fonte, mesmo que terceiros não sejam compelidos a copiar o código-fonte com o código-objeto.

4. Você não pode copiar, modificar, sub-licenciar ou distribuir o Programa, exceto de acordo com as condições expressas nesta Licença. Qualquer outra tentativa de cópia, modificação, sub-licenciamento ou distribuição do Programa não é válida, e cancelará automaticamente os direitos que lhe foram fornecidos por esta Licença. No entanto, terceiros que de você receberam cópias ou direitos, fornecidos sob os termos desta Licença, não terão suas licenças terminadas, desde que permaneçam em total concordância com ela.

5. Você não é obrigado a aceitar esta Licença já que não a assinou. No entanto, nada mais o dará permissão para modificar ou distribuir o Programa ou trabalhos derivados deste. Estas ações são proibidas por lei, caso você não aceite esta Licença. Desta forma, ao modificar ou distribuir o Programa (ou qualquer trabalho derivado do Programa), você estará indicando sua total aceitação desta Licença para fazê-los, e todos os seus termos e condições para copiar, distribuir ou modificar o Programa, ou trabalhos baseados nele.

6. Cada vez que você redistribuir o Programa (ou qualquer trabalho baseado nele), os recebedores adquirirão automaticamente do licenciador original uma licença para copiar, distribuir ou modificar o Programa, sujeitos a estes termos e condições. Você não poderá impor aos recebedores qualquer outra restrição ao exercício dos direitos então adquiridos. Você não é responsável em garantir a concordância de terceiros a esta Licença.

7. Se, em consequência de decisões judiciais ou alegações de infringimento de patentes ou quaisquer outras razões (não limitadas a assuntos relacionados a patentes), condições forem impostas a você (por ordem judicial, acordos ou outras formas) e que contradigam as condições desta Licença, elas não o livram das condições desta Licença. Se você não puder distribuir de forma a satisfazer simultaneamente suas obrigações para com esta Licença e para com as outras obrigações pertinentes, então como consequência você não poderá distribuir o Programa. Por exemplo, se uma licença de patente não permitirá a redistribuição, livre de "royalties", do Programa, por todos aqueles que receberem cópias direta ou indiretamente de você, então a única forma de você satisfazer a ela e a esta Licença seria a de desistir completamente de distribuir o Programa.

Se qualquer parte desta seção for considerada inválida ou não aplicável em qualquer circunstância particular, o restante da seção se aplica, e a seção como um todo se aplica em outras circunstâncias.

O propósito desta seção não é o de induzi-lo a infringir quaisquer patentes ou reivindicação de direitos de propriedade outros, ou a contestar a validade de quaisquer dessas reivindicações; esta seção tem como único propósito proteger a integridade dos sistemas de distribuição de software livres, o que é implementado pela prática de licenças públicas. Várias pessoas têm contribuído generosamente e em grande escala para os software distribuídos usando este sistema, na certeza de que sua aplicação é feita de forma consistente; fica a critério do autor/doador decidir se ele ou ela está disposto a distribuir software utilizando outro sistema, e um licenciado não pode impor qualquer escolha.

Esta seção destina-se a tornar bastante claro o que se acredita ser consequência do restante desta Licença.

8. Se a distribuição e/ou uso do Programa são restringidos em certos países por patentes ou direitos autorais, o detentor dos direitos autorais original, e que colocou o Programa sob esta Licença, pode incluir uma limitação geográfica de distribuição, excluindo aqueles países de forma a tornar a distribuição permitida apenas naqueles ou

entre aqueles países então não excluídos. Nestes casos, esta Licença incorpora a limitação como se a mesma constasse escrita nesta Licença.

9. A Free Software Foundation pode publicar versões revisadas e/ou novas da Licença Pública Geral de tempos em tempos. Estas novas versões serão similares em espírito à versão atual, mas podem diferir em detalhes que resolvem novos problemas ou situações.

A cada versão é dada um número distinto. Se o Programa especifica um número de versão específico desta Licença que se aplica a ele e a "qualquer nova versão", você tem a opção de aceitar os termos e condições daquela versão ou de qualquer outra versão publicada pela Free Software Foundation. Se o programa não especifica um número de versão desta Licença, você pode escolher qualquer versão já publicada pela Free Software Foundation.

10. Se você pretende incorporar partes do Programa em outros programas livres cujas condições de distribuição são diferentes, escreva ao autor e solicite permissão. Para o software que a Free Software Foundation detém direitos autorais, escreva à Free Software Foundation; às vezes nós permitimos exceções a este caso. Nossa decisão será guiada pelos dois objetivos de preservar a condição de liberdade de todas as derivações do nosso software livre, e de promover o compartilhamento e reutilização de software em aspectos gerais.

AUSÊNCIA DE GARANTIAS

11. UMA VEZ QUE O PROGRAMA É LICENCIADO SEM ÔNUS, NÃO HÁ QUALQUER GARANTIA PARA O PROGRAMA, NA EXTENSÃO PERMITIDA PELAS LEIS APLICÁVEIS. EXCETO QUANDO EXPRESSADO DE FORMA ESCRITA, OS DETENTORES DOS DIREITOS AUTORAIS E/OU TERCEIROS DISPONIBILIZAM O PROGRAMA "NO ESTADO", SEM QUALQUER TIPO DE GARANTIAS, EXPRESSAS OU IMPLÍCITAS, INCLUINDO, MAS NÃO LIMITADO A, AS GARANTIAS IMPLÍCITAS DE COMERCIALIZAÇÃO E AS DE ADEQUAÇÃO A QUALQUER PROPÓSITO. O RISCO TOTAL COM A QUALIDADE E DESEMPENHO DO PROGRAMA É SEU. SE O PROGRAMA SE MOSTRAR DE-

FEITUOSO, VOCÊ ASSUME OS CUSTOS DE TODAS AS MANUTENÇÕES, REPAROS E CORREÇÕES.

12. EM NENHUMA OCASIÃO, A MENOS QUE EXIGIDO PELAS LEIS APLICÁVEIS OU ACORDO ESCRITO, OS DETENTORES DOS DIREITOS AUTORAIS, OU QUALQUER OUTRA PARTE QUE POSSA MODIFICAR E/OU REDISTRIBUIR O PROGRAMA CONFORME PERMITIDO ACIMA, SERÃO RESPONSABILIZADOS POR VOCÊ POR DANOS, INCLUINDO QUALQUER DANO EM GERAL, ESPECIAL, ACIDENTAL OU CONSEQÜENTE, RESULTANTES DO USO OU INCAPACIDADE DE USO DO PROGRAMA (INCLUINDO, MAS NÃO LIMITADO A, A PERDA DE DADOS OU DADOS TORNADOS INCORRETOS, OU PERDAS SOFRIDAS POR VOCÊ OU POR OUTRAS PARTES, OU FALHAS DO PROGRAMA AO OPERAR COM QUALQUER OUTRO PROGRAMA), MESMO QUE TAL DETENTOR OU PARTE TENHAM SIDO AVISADOS DA POSSIBILIDADE DE TAIS DANOS.

FIM DOS TERMOS E CONDIÇÕES

COMO APLICAR ESTES TERMOS AOS SEUS NOVOS PROGRAMAS

Se você desenvolver um novo programa, e quer que ele seja utilizado amplamente pelo público, a melhor forma de alcançar este objetivo é torná-lo software livre que qualquer um pode redistribuir e alterar, sob estes termos.

Para isso, anexe os seguintes avisos ao programa. É mais seguro anexá-los logo no início de cada arquivo-fonte para reforçarem mais efetivamente a inexistência de garantias; e cada arquivo deve possuir pelo menos a linha de "copyright" e uma indicação de onde o texto completo se encontra.

<uma linha que forneça o nome do programa e uma idéia do que ele faz.>

Copyright (C) <ano> <nome do autor>

Este programa é software livre; você pode redistribuí-lo e/ou modificá-lo sob os termos da Licença Pública Geral GNU, conforme publicada pela Free Software Foundation; tanto a versão 2 da Licença como (a seu critério) qualquer versão mais nova.

Este programa é distribuído na expectativa de ser útil, mas SEM QUALQUER GARANTIA; sem mesmo a garantia implícita de COMERCIALIZAÇÃO ou de ADEQUAÇÃO A QUALQUER PROPÓSITO EM PARTICULAR. Consulte a Licença Pública Geral GNU para obter mais detalhes.

Você deve ter recebido uma cópia da Licença Pública Geral GNU junto com este programa; se não, escreva para a Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307, USA.

Inclua também informações sobre como contactá-lo eletronicamente e por carta.

Se o programa é interativo, faça-o mostrar um aviso breve como este, ao iniciar um modo interativo:

Gnomovision versão 69, Copyright (C) ano nome do autor O Gnomovision não possui QUALQUER GARANTIA; para obter mais detalhes digite `show w'. Ele é software livre e você está convidado a redistribuí-lo sob certas condições; digite `show c' para obter detalhes.

Os comandos hipotéticos `show w' e `show c' devem mostrar as partes apropriadas da Licença Pública Geral. Claro, os comandos que você usar podem ser ativados de outra forma que `show w' e `show c'; eles podem até ser cliques do mouse ou itens de um menu -- o que melhor se adequar ao programa.

Você também deve obter do seu empregador (se você trabalha como programador) ou escola, se houver, uma "declaração de ausência de direitos autorais" sobre o programa, se necessário. Aqui está um exemplo; altere os nomes:

Yoyodyne, Inc., aqui declara a ausência de quaisquer direitos autorais sobre o programa 'Gnomovision' (que executa interpretações em compiladores) escrito por James Hacker.

<assinatura de Ty Coon>, 1o. de abril de 1989

Ty Con, Vice-presidente

Esta Licença Pública Geral não permite incorporar seu programa em programas proprietários. Se seu programa é uma biblioteca de sub-rotinas, você deve considerar mais útil permitir ligar aplicações proprietárias com a biblioteca. Se isto é o que você deseja, use a Licença Pública Geral de Bibliotecas GNU, ao invés desta Licença.