Um Software para Exploração de Resultados do Sistema LEVSOFT

Rodrigo Hideki Tanaka[1], Angelo Passaro[2], Nancy Mieko Abe[2]

[1]Universidade do Vale do Paraíba/Faculdade de Ciências da Computação, São José dos Campos - SP, hidekit@uol.com.br
[2]Instituto de Estudos Avançados / Laboratório de Engenharia Virtual, São José dos Campos - SP, nancy@ieav.cta.br

Resumo - Este trabalho apresenta um módulo para a exploração gráfica de resultados obtidos da simulação de problemas pelos métodos de elementos finitos (MEF) e dos volumes finitos (MVF) bidimensionais. O módulo apresenta recursos de visualização em forma de gráficos bidimensionais e tridimensionais, isolinhas, representação de campos por mapas de cores e por vetores e outros recursos de visualização gráfica. O sistema foi desenvolvido em linguagem C++, com programação orientada a objetos, e utiliza recursos do pacote gráfico OpenGL. A estrutura das classes reutilizáveis para a geração e apresentação de gráficos desenvolvidas neste trabalho são apresentadas.

Palavras-chave: Visualização gráfica, método dos elementos finitos, análise de resultados, CAE
Área do Conhecimento: I - Ciências Exatas e da Terra

Introdução

A simulação de fenômenos e o projeto de dispositivos por computador são cada vez mais comuns em laboratórios de pesquisa e empresas devido aos avanços das tecnologias de informação. Tanto o projeto de um componente mecânico sujeito a determinadas condições físicas (por exemplo, velocidade de rotação, "stress mecânico") como a análise da eficiência da blindagem magnética utilizada nos celulares pode ser realizada por meio de simulações em um computador. Esta ferramenta computacional é conhecida como CAE.

O software LEVSOFT foi desenvolvido no Laboratório de Engenharia Virtual (LEV) do Instituto de Estudos Avançados, um dos Institutos do Centro Técnico Aeroespacial (CTA), para apoio á pesquisa, visando a análise de comportamento de dispositivos ou de fenômenos físicos por meio de simulações [1].

Atualmente, o LEVSOFT permite a simulação de vários fenômenos físicos e a análise de fibras ópticas, de componentes de óptica integrada baseados em efeitos eletroópticos, magnetoópticos e termoópticos, de dispositivos de microondas, assim como análises estacionárias (eletrostática, magnetostática e eletrocinética) e no domínio do tempo (transferencia de calor). O LEVSOFT é composto por seis módulos: Modelagem Geométrica, que permite desenhar um corte de um dispositivo em estudo usando recursos construtivos e operações sobre primitivas; Modelagem Física, que permite associar ao modelo geométrico do dispositivo propriedades físicas de materias, fonte e condições de contorno [2]; Módulo de Elementos Finitos, que gera a malha de elementos finitos utilizando o método de Delaunay [3],[4]; Módulo de Resolução, o que monta e resolve o sistema de equações algébricas oriundas do MEF nas equações que regem o estudo; Módulo de Refinamento Auto-Adaptativo, que gera a malha automaticamente e a refina em regiões críticas e Módulo de Exploração de Resultados.

O módulo de exploração de resultados tem um papel importante no sistema pois permite mostrar a grande quantidade de dados obtidos por meio dos cálculos computacionais realizados para resolver um determinado problema utilizando recursos visuais. Dentre esses recursos podem ser citados gráficos, representação por meio de isolinhas, representação por mapas de cores ou por vetores. Os recursos de visualização gráfica favorecem uma melhor compreensão dos resultados, facilitando a identificação de pontos críticos no estudo do problema (fenômeno). Este trabalho apresenta o módulo de exploração de resultados do sistema LEVSOFT em sua versão orientada a objetos, que apresenta os recursos visuais citados anteriormente, permite a visualização dos resultados em várias janelas ao mesmo tempo, e a análise de estudos que apresentem mais de uma variável de estado.

Metodologia

O módulo foi desenvolvido em linguagem C++, com programação orientada a objetos, e utiliza o pacote gráfico OpenGL para a apresentação dos modelos geométricos, físicos e de elementos finitos, além das representações gráficas que permitem a exploração de resultados.

O trabalho foi implementado utilizando-se o compilador Visual C++ e recursos da biblioteca MFC (Microsoft Foundation Classes) para a construçao da interface gráfica com o usuário.

A modelagem orientada a objetos foi realizada utilizando diagramas UML (Unified Modeling Language) [5], [6].

A construção das isolinhas e dos mapas de cores está baseada numa malha triangular obtida com a aplicação do método de Delaunay e, no caso de elementos finitos, utilizada para a resolução do problema. Em cada ponto da malha (vértice de um triângulo), são calculados os valores das variáveis de estado (incógnitas do problema).

A determinação das isolinhas é realizada a partir do número de contornos escolhido pelo usuário e pelos valores mínimo e máximo da variável a ser mapeada. Ao longo de cada isolinha o valor da variável de estado é constante e a diferença do valor da variável em uma linha de contorno e em outra linha adjacente é constante. Desta forma, em regiões em que a variável de estado apresenta grande variação no espaço, as linhas são mais concentradas e nas regiões em que a variável de estado varia pouco, as linhas são mais espaçadas.

A cada isolinha é associada uma cor de uma paleta de cores. Uma outra opção de visualização é a apresentação por faixa de valores. Neste caso, as faixas definidas por duas isolinhas são preenchidas com cores. Quanto maior o número de contornos melhor a representação do comportamento da variável.

Um algoritmo de subdivisão quadtree é utilizado para a obtenção das isolinhas [7]. O uso deste método permite um preenchimento das faixas com uma graduação de cores mais suave.

A função que implementa a subdivisão quadtree recebe um triângulo da lista de elementos finitos e verifica a qual faixa de valores pertece cada um dos valores da variável nos tręs vértices. Se a faixa de valor nos tręs vértices for a mesma, o elemento é identificado com essa faixa e será pintado com a cor associada a essa faixa. Em casos em que dois vértices estão na mesma faixa e o terceiro vértice possui valor pertencente a uma faixa acima (significando que o triângulo é atravessado por uma isolinha), a linha de contorno entre as faixas é desenhada e cada uma das partes resultantes desta divisão (um novo triângulo e um quadrilátero) serão pintados com as cores referentes as faixas a que pertencem. Se a diferença de faixa de valores for maior que 1 entre os vértices, o elemento é dividido em quatro outros triângulos e a função é chamada recursivamente até que se alcance uma das condições acima. Na Figura 1 é apresentado um esquema da aplicação do algoritmo.

Duas classes principais (CGraphic2D3DView e CGraphic2D3D) foram criadas para a construção e apresentação dos recursos de exploração.

As classes foram separadas em dois tipos: de interface e de dados. A classe de interface possui apenas funções de desenho enquanto a classe de dados é responsável por armazenar os dados a serem usados. Isso foi feito para que as classes se tornassem reutilizáveis em outros projetos.

A classe CGraphic2D3DView é a classe de interface, responsável pelas funções de desenho de isolinhas, mapa de cores e representação vetorial em 2D e também possui funções de desenho para apresentação do preenchimento em 3D. A classe CGraphic2D3DView armazena uma lista de *display lists*, um recurso oferecido pelo OpenGL para melhora do desempenho na apresentação dos objetos na tela.

A classe CGraphic2D3D é a classe de dados que armazena os dados que serão usados pela CGraphic2D3DView. Há uma associação entre as duas classes; uma classe CGraphic2D3Dview possui uma referencia para uma classe CGraphic2D3D. O relacionamento entre as classes pode ser visto na Figura 2. A classe CGraphic2D3D utiliza a função Subdivisão Quadtree para a determinação das isolinhas.

A Subdivisão Quadtree cria uma lista de polígonos que é armazenada em uma classe denominada CQuadtreeInfo, a qual está associada ŕ CGraphic2D3D. Se o número de isolinhas a ser apresentado não muda, esta lista é utilizada para o redesenho das isolinhas, o que economiza tempo de processamento. Depois de criada a lista de polígonos, a classe CGraphic2D3DView cria uma *display list* para essa quantidade de contornos. Quando uma outra janela é criada, o processo para que o desenho do dispositivo seja apresentado na tela é mais rápido, pois a lista de polígonos já existe. Se a quantidade de linhas de contorno é modificada pelo usuário, a função Subdivisão Quadtree é chamada novamente, criando uma nova lista de polígonos. A classe CQuadtreeInfo possui uma associação com a classe CQuadtreePolygon. Esses polígonos podem ser triângulos e quadriláteros que são usadas para o desenho das isolinhas e do preenchimento.

À classe CGraphic2D3D são associadas, ainda, uma lista de pontos e uma lista de elementos finitos como pode ser visto na Figura 3. A classe Point permite armazenar uma lista de valores de variáveis de estado.

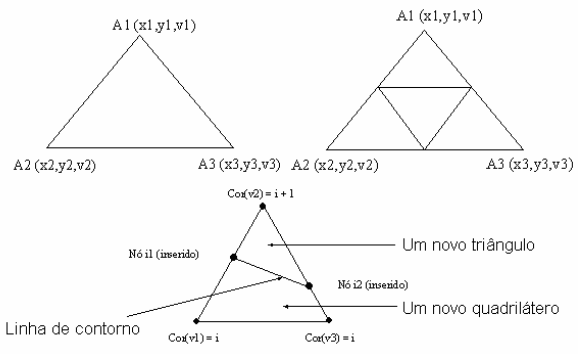

Figura 1 – Subdivisão Quadtree.

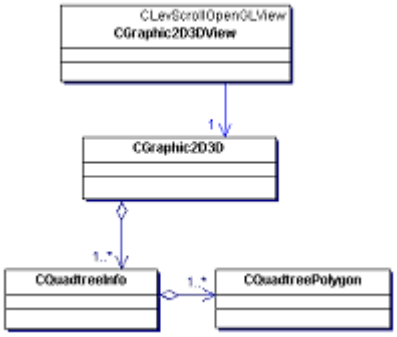

Figura 2 – Diagrama de classes de visualizações e dados.

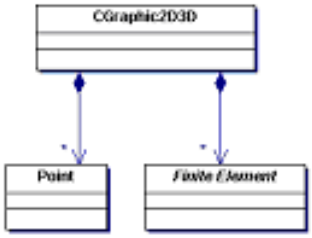

Figura 3 – Relacionamento entre a classe de dados CGraphic2D3D e as classes de pontos e de elementos finitos.

Outro recurso importante para a análise dos dados é a apresentação de gráficos xy. Para este fim foi criado um pacote gráfico 2D para apresentar este tipo de gráfico. O pacote é disponibilizado como uma biblioteca dinâmica e pode ser usado por outro software (na plataforma Win32) que necessite apresentar uma variável (y) em função de outra variável independente (x) em um gráfico bidimensional. Um outro software desenvolvido no LEV, denominado SIACRA já está utilizando a biblioteca [8].

As principais classes do pacote gráfico 2D são apresentadas na Figura 4. A classe CGraphicView é a classe responsável pela apresentação das curvas. A classe CGrafico armazena as informações das curvas (suporta até 20 curvas). A classe CAxisView é responsável pela apresentação dos eixos e suas propriedades (divisões, subdivisões, grade). A classe CEixo armazena as informações dos eixos x e y (valores máximos e mínimos, quantidade de divisões).

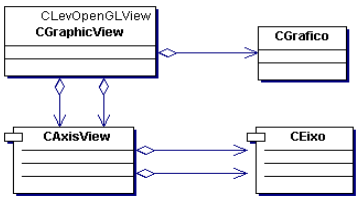

Figura 4 – Diagrama de classes do pacote Gráfico2D.

Resultados

A Figura 5 apresenta a análise de um cabo trifásico usando a visualização em mais de uma janela, com cada janela usando uma ferramenta de visualização diferente. O cabo trifásico é composto por treş condutores (cobre) de seção circular imersos em um material isolante (polietileno) também de seção circular, circundado por uma camada metálica aterrada (potencial elétrico igual a zero volts). Um dos condutores é submetido em um potencial elétrico de 100 V; o segundo a um potencial de 50 V e o terceiro está flutuando (nenhum potencial elétrico é aplicado).

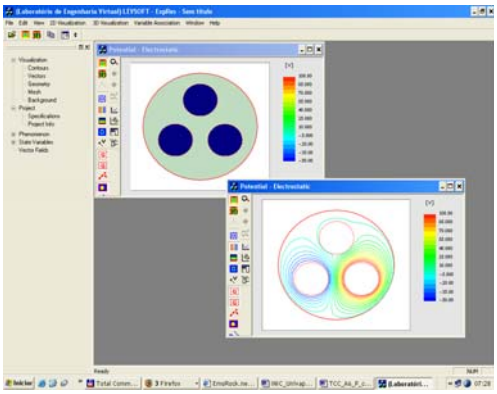

Figura 5 – Visualização de um mesmo dispositivo em duas janelas. Cada janela apresenta sua própria barra de ferramentas.

A Figura 6 apresenta o uso da ferramenta de interpolação que foi criada a partir do pacote gráfico 2D. Pode-se observar duas janelas de exploração, cada uma com uma janela adicional apresentando a interpolação dos resultados ao longo de duas linhas diferentes definidas pelo usuário. A variável de estado também pode ser apresentada em gráficos tridimensionais, Figura 7.

A Figura 8 apresenta o estudo de escoamento de um fluído por um tubo com uma aleta não porosa. Pode-se observar uma variável de estado em cada janela de visualização.

Discussão

O módulo desenvolvido permite a visualização de dados em forma de gráficos, curvas de nível, isolinhas, representação por mapas de cores e por vetores, permitindo a visualização em várias janelas do programa ao mesmo tempo. Os recursos de exploração implementados são de fácil utilização. Cada janela de visualização permite o controle de parâmetros de visualização de forma independente (tipo de representação gráfica, zoom, exploração ponto a ponto, dentre outras) por meio de uma barra de ferramentas independente. Um número ilimitado de variáveis

de estado podem ser visualizadas independentemente, facilitando a análise de fenômenos complexos.

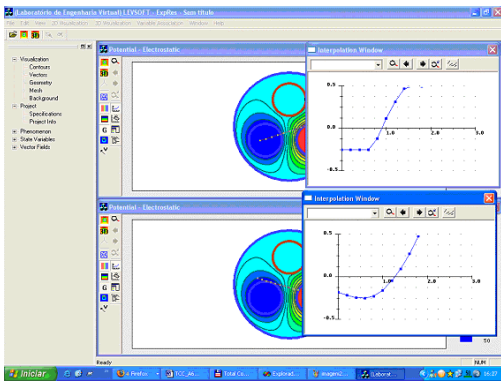

Figura 6 – Visualização do estudo usando a janela de interpolação em janelas diferentes

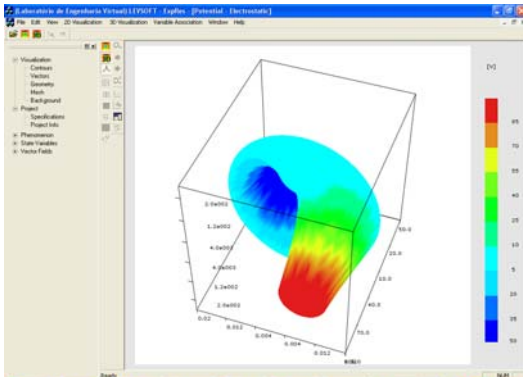

Figura 7 – Representação tridimensional do potencial elétrico em um cabo trifásico

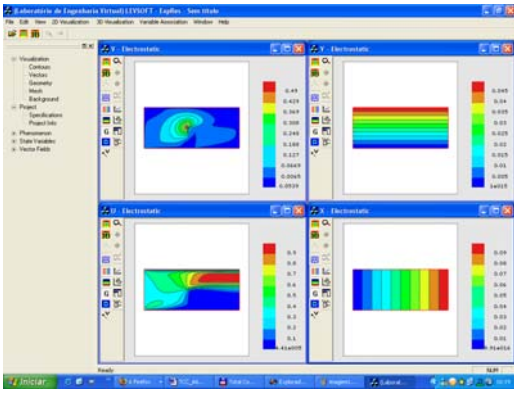

Figura 8 – Visualização de várias variáveis de estado

Conclusão

Neste trabalho foi desenvolvido um módulo de exploração gráfica de resultados obtidos da simulação de fenômenos físicos e projeto de dispositivos. Um dos objetivos do trabalho era o desenvolvimento de componentes reutilizáveis, neste contexto, pode-se citar o desenvolvimento do pacote gráfico 2D, o qual está sendo utilizado tanto no módulo de exploração de resultados como em outros softwares desenvolvidos no LEV. Outro componente reutilizável é a classe que permite processar as informações necessárias para o desenho das isolinhas, mapa de cores e representação vetorial. Esta classe, a partir das informações de coordenadas de pontos, da malha triangular e valores de variável de estado associadas, constrói, utilizando o algoritmo de subdivisão quadtree, toda a informação necessária para a apresentação gráfica dos resultados.

Agradecimentos

O trabalho foi parcialmente financiado pela FAPESP, processo nº 03/06888-1.

Referências

[1] ABE, N. M., et. al. 'Um Sistema de Software para Análise de Dispositivos e Componentes de Óptica Integrada, Fibras Ópticas e Microondas', In: *Anais do V Congresso Brasileiro de Eletromagnetismo* - CBMAG 2002, CD-ROM, Gramado, RS.

[2] SANTOS, F.J.R. Desenvolvimento de um Módulo de Software para Modelagem Geométrica e Física de Problemas Estáticos e Ópticos para a Aplicação do MEF, Trabalho de Graduação Interdisciplinar do Curso de Ciencias da Computação, Universidade Braz Cubas, Mogi das Cruzes, 2003.

[3] MAGALHÃES, G. M.; PASSARO, A.; ABE, N. M. Geração de Malha de Delaunay Orientada a Objetos, In: Workcomp 2000 – Workshop de Computação, São José dos Campos, SP, Out.2000, Anais, p.73-79.

[4] MAGALHÃES, G. M. Geração de Malha Auto-Adaptativa Bidimensional para a Equação de Poisson, Trabalho de Graduação Interdisciplinar do Curso de Ciencias da Computação, Universidade Braz Cubas, Mogi das Cruzes, 2001, 80 p.

[5] MATOS, A. V. UML: Prático e descomplicado. Editora Érica, 2002.

[6] DEITEL, H. M; DEITEL, P. J. C++ Como Programar. Editora. Bookman, 3º Edição, 2001.

[7] WANG, K. P. and BRUCH, J. C. JR. Adaptive Quadtree Subdividing Contour Plotting Algorithms, Engineering Computations, Vol.12, p.545-553, 1995.

[8] MACIEL, C., Trabalho de Graduação Interdisciplinar do Curso de Engenharia da Computação, Universidade Braz Cubas, Mogi das Cruzes, 2004.