UNIVERSIDADE ESTADUAL DE LONDRINA

FABIO TAKESHI MATSUNAGA

UM SISTEMA WEB PARA RESOLUÇÃO DE PROBLEMAS ENVOLVENDO EQUAÇÕES DIFERENCIAIS PARCIAIS PELO MÉTODO DAS DIFERENÇAS FINITAS

Londrina
2014

FABIO TAKESHI MATSUNAGA

UM SISTEMA WEB PARA RESOLUÇÃO DE PROBLEMAS ENVOLVENDO EQUAÇÕES DIFERENCIAIS PARCIAIS PELO MÉTODO DAS DIFERENÇAS FINITAS

Dissertação de Mestrado apresentado à Universidade Estadual de Londrina como parte dos requisitos para obtenção do título de Mestre em Ciência da Computação.

Orientador: Prof. Dr. Jacques Duílio Brancher.
Co-orientadora: Profa. Dra. Neyva Maria Lopes Romeiro.

Londrina
2014

Catalogação elaborada pela Divisão de Processos Técnicos da Biblioteca Central da Universidade Estadual de Londrina

Dados Internacionais de Catalogação-na-Publicação (CIP)

M434s Matsunaga, Fabio Takeshi.
Um sistema web para resolução de problemas envolvendo equações diferenciais parciais pelo método das diferenças finitas / Fabio Takeshi Matsunaga. – Londrina, 2014.
87 f. : il.

Orientador: Jacques Duílio Brancher.
Coorientador: Neyva Maria Lopes Romeiro.
Dissertação (Mestrado em Ciência da Computação) – Universidade Estadual de Londrina, Centro de Ciências Exatas, Programa de Pós-Graduação em Ciência da Computação, 2014.
Inclui bibliografia.

1. Simulação (Computadores) – Teses. 2. Equações diferenciais parciais – Soluções numéricas – Teses. 3. Geração numérica de malhas – Análise numérica – Teses. 4. Arquitetura de redes de computador – Teses. I. Brancher, Jacques Duílio. II. Romeiro, Neyva Maria Lopes. III. Universidade Estadual de Londrina. Centro de Ciências Exatas. Programa de Pós-Graduação em Ciência da Computação. IV. Título.

CDU 519.61-7

FABIO TAKESHI MATSUNAGA

UM SISTEMA WEB PARA RESOLUÇÃO DE PROBLEMAS ENVOLVENDO EQUAÇÕES DIFERENCIAIS PARCIAIS PELO MÉTODO DAS DIFERENÇAS FINITAS

Dissertação de Mestrado apresentado à Universidade Estadual de Londrina como parte dos requisitos para obtenção do título de Mestre em Ciência da Computação.

BANCA EXAMINADORA

_____

Prof. Dr. Jacques Duílio Brancher
UEL – Londrina - PR

_____

Prof. Dr. Bruno Bogaz Zarpelão
UEL – Londrina – PR

_____

Prof. Dr. Evandro Bacarin
UEL – Londrina – PR

_____

Prof. Dr. Eliandro Rodrigues Cirilo
UEL – Londrina – PR

Londrina, 13 de Fevereiro de 2014.

Dedico este trabalho aos meus familiares, amigos
e colegas profissionais que me acompanharam durante esta caminhada,
pois sem eles eu não teria forças e motivação para eu poder
realizar meus sonhos e objetivos.

AGRADECIMENTOS

Ao professor e orientador Jacques Duílio Brancher pelas orientações, pela paciência, pelas revisões da dissertação e pela oportunidade de me ingressar no programa de mestrado e pelas ideias oferecidas, as quais foram primoridais para a execução deste projeto de pesquisa e para o meu crescimento pessoal e profissional.

À professora e co-orientadora Neyva Maria Lopes Romeiro pelos ensinamentos dos métodos numéricos, os quais foram fundamentais para a execução do sistema desenvolvido. Também ao professor Eliandro Rodrigues Cirilo que conduziu o grupo de estudos dos métodos matemáticos em conjunto com a prof. Neyva e prof. Paulo Natti, o que nos motivou o desenvolvimento do projeto.

À dra. Miroslava Rakocevic que teve uma parte da contribuição na minha formação profissional e pessoal e em algumas produções científicas durante o período do mestrado.

Aos colegas de mestrado, Armando, Mariana, Fábio S., Estevan, Felipe A., Fernando, Ramon e Felipe S. pela amizade durante o projeto e pelas ótimas trocas de experiências e conhecimentos e por terem me ajudado na revisão da dissertação. Também aos colegas Roberto, Carolina, Matheus, Murilo, Deryk, Paulo, Rafael, Elisa, Fernanda F.

Fernanda O., Gilberto, Marcos, Eduardo, Luiz F., Anderson, Bruno, Wagner, Márcio e outros pela companhia durante essa caminhada.

Ao Jose Luiz Vilas Boas pelo auxílio no desenvolvimento de algumas interfaces e funcionalidades do sistema web e pelas reuniões produtivas do nosso projeto. Acredito que sem a nossa dedicação e confiança mútua, o trabalho não teria chegado ao estado atual em que se encontra.

Aos professores Rodolfo Barros, Wesley Attrot, Evandro Bacarin e Paulo

Negri pelas aulas ministradas e pelos ensinamentos que me motivaram no prosseguimento da formação acadêmica. Tambem as secretárias do DC, Rosana Reis e Valdete Matos pela atenção e pela disponibilidade em tempo integral de fornecer salas e laboratórios para a condução do projeto.

Aos meus amigos pelo companheirismo e pelas boas experiências vivenciadas, além de me darem forças e sempre estarem direta ou indiretamente juntos comigo neste projeto de vida.

À Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pelo fornecimento da bolsa de estudos, a qual foi um fomento para a condução do meu projeto, das produções científicas e para a minha viagem à conferência de Temuco, Chile.

“É difícil dizer que é impossível,
pois a fantasia de ontem á a esperança de hoje
e a realidade de amanhã.”
– Robert H. Goddard

MATSUNAGA, Fabio Takeshi. **Um sistema web para resolução de problemas envolvendo equações diferenciais parciais pelo método das diferenças finitas**. 2014. 87 f. Dissertação (Mestrado) - Universidade Estadual de Londrina, Londrina. 2013.

RESUMO

A discretização por diferenças finitas é um método amplamente utilizado na solução de diversos problemas reais envolvendo simulações numéricas e equações diferenciais parciais. Simulações em geral são compostos de três passos principais: pré-processamento, processamento e pósprocessamento, os quais são operações custosas e complexas para problemas mais refinados. Houve uma motivação para o uso de ferramentas computacionais para simulações, entretanto muitos destes são restritos à instalação de um software local e na limitação de processamento, como nos softwares desktop e dispositivos móveis. É possível também explorar a junção de tecnologias web com uso de equações diferenciais parciais para genericamente resolver problemas reais. O objetivo deste trabalho foi usar tecnologias web para resolver problemas estacionários e dependentes do tempo envolvendo equações diferenciais parciais através do método das diferenças finitas. As equações aplicadas no sistema web foram a de Laplace, da condução do calor 1D e 2D, da onda 1D e do transporte 1D e 2D em coordenadas cartesianas e a equação da energia 2D em coordenadas generalizadas. O sistema consiste em um web service de comunicação cliente-servidor, um banco de dados para o armazenamento de informações dos usuários, requisições e resultados, um módulo de processamento de cálculos (back-end) e uma interface gráfica para visualização das malhas discretizadas (front-end). O serviço pode ser executado em diversos dispositivos e plataformas, desde computadores pessoais a dispositivos móveis, uma vez que os cálculos são feitos em um servidor remoto, exibindo os resultados ao usuário em tempo de execução. O sistema proposto mostrou-se eficiente na resolução de problemas 1D e 2D, conseguindo efetuar simulação de diversos fenômenos físicos desejados pelo usuário. A principal contribuição deste trabalho é a disponibilidade de uma aplicação web para pesquisadores resolverem problemas reais descritos por meio de equações diferenciais em geometrias regulares e generalizadas. Além disso, foi possível disponibilizar uma arquitetura web aberta para a inclusão de outros tipos de sistemas de coordenadas e para resolução de diversos tipos de equações diferenciais parciais.

Palavras-chave: Back-end. Banco de dados. Coordenadas generalizadas. Front-end. Visualização de malhas.

MATSUNAGA, Fabio Takeshi. **A web system for solving real problems involving partial differential equations by finite difference discretization method**. 2014. 87 f. Dissertation (Master's Degree) - State University of Londrina, Londrina. 2013.

ABSTRACT

Numerical simulations such as partial differential equations resolution are applied to different knowledge areas, and methods as the finite difference discretization have been widely applied to solve real problems. Simulations are generally composed of three steps: pre-processing, processing and post-processing, which are expensive and complex operations for large problems, due to involvement with linear algebra operations. There is a motivation for the use of computational tools for simulations, however many of these are restricted to a local software installation and limiting processing, as in the desktop softwares and mobile devices. It is possible to explore the junction of web technologies with the use of partial differential equations to solve a range of real problems. The aim of this study was to use web-based technologies to develop a web system to solve stationary and time-dependent partial differential equations. The equations applied to the web service was the Laplace equation, 1D/2D heat conduction, 1D wave propagation and 1D/2D transport equation in Cartesian coordinates and the 2D energy equation in generalized coordinates, solved by the finite difference discretization method. The system consists of web service for client-server communication, a database for user information, requests and results storage, a module of calculation processing (front-end), a graphical interface for visualization of discretized mesh (back-end). The service can be executed on devices regardless their performance, such as personal computers and mobile devices, since the high cost calculation are made on a remote server while the results are displayed on the screen at runtime. The proposed system was effective in solving 1D and 2D real problems, being able to perform simulation of physical phenomena demanded by the user. The main contributions of this work are the availability of a web application researchers solve real problems described by differential equations in regular and generalized geometries. Moreover, it was possible to provide an open web architecture for including other types of coordinate systems and to solve others partial differential equations.

Keywords: Back-end. Database. Front-end. Generalized coordinates. Heat conduction. Mesh visualization.

LISTA DE FIGURAS

Figura 4.1 – Malha computacional e linhas x e h 33
Figura 5.1 – Principais módulos do sistema web 38
Figura 5.2 – Arquitetura do sistema web 39
Figura 5.3 – Mapa de tecnologias computacionais utilizadas para o desenvolvimento do sistema 40
Figura 5.4 – Exemplos de algumas árvores sintáticas e suas respectivas expressões: função matemática (A) e condição intervalar (B) 42
Figura 5.5 – MER do banco de dados de comunicação cliente-servidor e armazenamento de resultados e informações 43
Figura 5.6 – Diagrama do web service e as mensagens WSDL trocadas durante o processamento 46
Figura 5.7 – Utiliação de uma mesma cadeia de cores para diferentes intervalos de valores, tanto para positivos como (A) para negativos (B) 48
Figura 5.8 – Interface do gerador de malhas desenvolvido em um navegador web (A) e um dispositivo móvel (B) 49
Figura 6.1 – Saídas para equação do calor 1D na forma de malhas visuais 53
Figura 6.2 – Saídas em algumas instâncias de tempo do problema 2 da equação do calor 2D 54
Figura 6.3 – Saídas em algumas instâncias de tempo do problema 3 da equação do calor 2D 55
Figura 6.4 – Saídas em algumas instâncias de tempo da equação da onda 1D. As escalas são definidas de acordo com o tamanho da área de plotagem, em que o gráfico é ajustado 56
Figura 6.5 – Saídas em algumas instâncias de tempo do teste da equação do transporte 2D. 57 57
Figura 6.6 – Saídas obtidas no Scilab na equação do calor 1D (A) e 2D (B) 58
Figura 6.7 – Exemplo de malha gerada 1, em coordenadas generalizadas utilizando as métricas de transformações 59
Figura 6.8 – Exemplo de malha gerada 3 (figura ’garrafa’), em coordenadas generalizadas utilizando as métricas de transformações 60
Figura 6.9 – Saídas da execução de uma simulação da equação de energia 2D em uma geometria generalizada em várias intâncias de tempo 61

Figura 6.10 – Malha 2D da geometria do Lago Igapó gerada pelo sistema web em diferentes níveis de detalhamento com malhas 40x20 (A) e 100x50 (B), em comparação com o domínio físico do problema, obtido do Google Maps (C). A dimensão do problema está medido em Km 62
Figura 6.11 – Visualização da simulação do transporte de poluentes em diferentes instâncias de tempo (A-C) em comparação com o problema original (D) – obtido de [1] 64
Figura 6.12 – Comparação de malhas refinadas entre dois dispositivos e sistemas operacionais: (A) dispositivo móvel e (B) web browser convencional. 64
Figura 6.13 – Visualização da geração de malhas (A) e resolução de uma EDP em uma instância de tempo (B) do sistema web acessado de um dispositivo móvel. 64
Figura 6.14 – Sistema de buscas de problemas resolvidos..... 65
Figura 6.15 – Consulta de um problema resolvido, com a abertura dos parâmetros utilizados..... 65

LISTA DE SIGLAS E ABREVIATURAS

BLAS	Basic Linear Algebra Subprograms
CAD	Computer Aided Architecture
CPU	Central Processing Unit
CSV	Comma Separated Values
D	Diagonal Principal
DDF	Discretização por Diferenças Finitas
DL	Diagonal Inferior
DU	Diagonal Superior
E/S	Entrada/Saída
EDP	Equação Diferencial Parcial
GPGPU	General-Purpose Graphical Processing Unit
GPU	Graphical Processing Unit
HTML	HyperText Markup Language
ID	Número de Identificação
IDA	Incremental Dynamics Analysis
IDE	Integrated Development Environment
Java	SE Java Standard Edition
JAX-WS	Java API for XML Web Services
JSF	Java Server Faces
JSP	Java Server Pages
LAPACK	Linear Algebra PACKage
MEF	Método dos Elementos Finitos
MER	Modelo Entidade-Relacionamento
MVC	Model Viewer Controller
PaaS	Platform as a Service
OOCSMP	Object Oriented Continuous Simulation Language
PVI	Problemas de Valores Iniciais
PVC	Problemas de Valores de Contorno
SaaS	Software as a Service
SCG	Sistemas de Coordenadas Generalizadas
SOAP	Simple Object Access Protocol
SQL	Structured Query Language

TI	Tecnologia da Informação
WSDL	Web Services Description Language
WYSIWYG	What You See Is What You Get
XML	eXtensible Markup Language

SUMÁRIO

1 INTRODUÇÃO 14

2 REVISÃO BIBLIOGRÁFICA 18
2.1 FERRAMENTAS DESKTOP E MÉTODOS DE SOLUÇÃO DE EDPS 18
2.2 TECNOLOGIAS WEB E SERVIÇOS EM NUVEM 19
2.3 MOBILIDADE DOS SIMULADORES 21
2.4 SOLUÇÕES PARA PROBLEMAS REAIS 22

3 DIFERENÇAS FINITAS EM COORDENADAS CARTESIANAS 24
3.1 EQUAÇÃO DA CONDUÇÃO DO CALOR 1D E 2D 25
3.2 EQUAÇÃO DA PROPAGAÇÃO DA ONDA UNIDIMENSIONAL 28
3.3 EQUAÇÃO DA ADVECÇÃO-DIFUSÃO 1D E 2D 29

4 DIFERENÇAS FINITAS EM COORDENADAS GENERALIZADAS 31
4.1 GERAÇÃO DE MALHAS EM COORDENADAS GENERALIZADAS 32
4.2 EQUAÇÃO DA ENERGIA 2D EM COORDENADAS GENERALIZADAS 35

5 ARQUITETURAS E MÓDULOS DO SISTEMA WEB 38
5.1 TECNOLOGIAS E IMPLEMENTAÇÕES UTILIZADAS 40
5.2 MÓDULO DE PROCESSAMENTO DE CÁLCULOS 41
5.3 MÓDULO DE BANCO DE DADOS 43
5.4 MÓDULO WEB SERVICE – COMUNICAÇÃO FRONT-END E BACK-END 45
5.5 INTERFACE GRÁFICA COM USUÁRIO E VISUALIZAÇÃO DE MALHAS 47
5.5.1 Coordenadas Cartesianas 47
5.5.2 Coordenadas Generalizadas 49

6 RESULTADOS E DISCUSSÕES 52
6.1 TESTE NAS COORDENADAS GENERALIZADAS 56

7 CONCLUSÕES E TRABALHOS FUTUROS 67
7.1 TRABALHOS FUTUTOS 68

REFERÊNCIAS 69

APÊNDICES 74

APÊNDICE I - Descrição da interface gráfica do gerador de malhas em coordenadas generalizadas. 75

ANEXOS 76

ANEXO I - Produções científicas no tema da dissertação em congressos aceitos como qualis na área de Ciência da Computação, entitulados de: 77

1 INTRODUÇÃO

O processo de simulações computacionais teve uma grande ascensão nos últimos anos, sendo um fomento para diversos grupos de pesquisadores. A principal motivação para isso é que este processo é a base para a modelagem de problemas reais de diversas áreas do conhecimento, como a matemática, física, química e engenharias. Ademais, o uso de simulações permite que situações reais sejam avaliadas e analisadas com um custo menor, mesmo que de um modo mais simplificado e com diversos níveis de detalhamento.

A base de uma grande parte das simulações são as equações diferenciais parciais (EDP), as quais têm substituído as descrições de muitos fenômenos através de fórmulas matemáticas tradicionais [2]. Uma vez que a solução analítica das EDPs é de difícil cálculo, a resolução deve ser feita por meio da aplicação de métodos numéricos que discretizam essas equações, através da definição da distribuição espacial e temporal do problema. Os mais conhecidos e utilizados são as diferenças finitas, volumes finitos e elementos finitos, os quais são bastante utilizados na modelagem através de EDPs [3, 4].

As simulações de problemas reais tornam-se complexas e inviáveis sem o uso de métodos computacionais, devido ao envolvimento com operações de álgebra linear [5]. Dentre os métodos numéricos citados, o método da discretização por diferenças finitas (DDF) é considerado o mais simples de implementar, umas vez que este é totalmente baseado em operações algébricas. Os fenômenos e problemas podem ser descritos simbolicamente através da associação das variáveis e parâmetros globais de equações, facilitando as operações algébricas e a descrição das EDPs [2, 4].

Para quaisquer tipos de simulações, existem três etapas importantes a serem ponderadas: o pré-processamento, o processamento e o pós-processamento. O pré-processamento consiste na preparação dos dados iniciais, na descrição do domínio espacial e da geometria a ser modelada, bem como os parâmetros de entrada iniciais, como a definição das condições iniciais e das condições de contorno de um problema. A ideia dessa etapa é também ter informações para dividir um modelo geométrico 2D ou 3D em diversos elementos analíticos denominados nós (sub-processamento), para realizar operações em cada um deles.

A etapa do processamento propriamente dito é a simulação da EDP sobre a malha computacional discretizada, em que são utilizados diversos métodos numéricos para conhecer a solução do problema em cada nó específico do domínio discreto. É neste processo que pode-se aplicar diferentes EDPs, como a equação da condução do calor, da propagação da onda, do transporte e de Navier-Stokes. Para este processo, questões como refinamento de malhas para uma simulação mais precisa e de alto nível de detalhamento tem sido considerados [6], bem como uma grande variedade de procedimentos para geração e refinamento de malhas [7, 8] e o melhoramento do desempenho dos mesmos [9].

A etapa do pós-processamento consiste extração dos resultados parciais e finais da simulação, para eventuais análises de dados e a visualização gráfica dos resultados. Para o caso de resolução de EDPs utilizando DDF, a etapa do pós-processamento permite visualizar os resultados graficamente em diferentes perspectivas, além de permitir que o usuário realize manipulações em diversos pontos de nós para melhorar a precisão dos resultados [10, 11].

Em geral, as simulações de EDPs, assim como outros métodos numéricos, são complexas e de alto custo computacional, o que fomenta o desenvolvimento de diversas ferramentas computacionais e interativas. Grande et al [12] por exemplo, utilizaram tecnologias multimídia de gravação de vídeos para visualização de interação de ondas eletromagnéticas, através do método das diferenças finitas. Já outros autores [13, 14] investiram no uso de dispositivos móveis para proporcionar maior praticidade e mobilidade na execução de diversos tipos de simulação envolvendo EDPs.

Serviços e aplicações locais que utilizam ferramentas interativas, tais como vídeos e dispositivos móveis, necessitam de alguns requisitos ideais e o desempenho das execuções dependem das configurações de *hardware* do dispositivo. Dentre os requisitos, incluem uma grande capacidade de memória, uma configuração de *hardware* ideal e instalações de *softwares* locais. Outros desafios existentes nos simuladores são encontradas em design [15] ou respostas apropriadas dos processamentos de cálculos [16].

Questões como a acessibilidade de serviços em simuladores numéricos têm sido tratados nos últimos anos [17]. Essas ideias surgiram devido à necessidade dos simuladores terem seus códigos-fontes reescritos e modificados para cada caso experimental e para serem adaptados para um determinado *hardware*. Entretanto, quando um serviço é portado em dispositivos móveis, as execuções podem sofrer restrições de desempenho, pois sofrem de limitações de processamento.

Diante do cenário observado, tornou-se crescente a motivação no uso de tecnologias *web* para o desenvolvimento de diversas aplicações de modo geral, isto é, o processo de "weblização"[10]. A primeira razão é que a transferência do conhecimento e do aprendizado é prático, eficiente e permite a combinação e reutilização de diversas rotinas. Outra é que os métodos numéricos utilizam um grande volume de dados, o que exige banco de dados e servidor remoto para o armazenamento de informações, os quais são constantemente mantidos e facilmente acessíveis de qualquer lugar e em qualquer momento [18].

As tecnologias *web* se caracterizam também pelo controle e atualização de versão de uma maneira automática e transparente, sem ter nenhuma inconveniência aos usuários. Existe também uma facilidade de *deployments*, isto é, disponibilizar um determinado sistema para o seu uso global. Além disso, *softwares* e serviços não necessitam ser reinstalados caso ocorra alguma avaria no computador.

Os sistemas *web* junto com o uso da computação em nuvem (*cloud computing*), estão se tornando uma tendência em alta na indústria de Tecnologia de Informação

[19]. Estes sistemas também vem sendo utilizadas em diversas aplicações científicas na área matemática, física, química e engenharias [20]. Considerando as aplicações científicas, alguns sistemas baseados em *web* para soluções de diversos problemas reais específicos foram desenvolvidos [21, 22, 11].

O estado da arte de diversos serviços encontra-se em aplicações remotas que utilizam recursos de *hardwares* e *softwares* advindos de servidores remotos [23]. Contudo, são poucas soluções existentes baseadas em *web* que tiveram como foco a resolução de EDPs e consequentemente, para simulação de diversos fenômenos descritos por esse modelo, sendo que a maioria focou no desenvolvimento de um modelo *web* para um caso específico [24, 11]. Além disso, a portabilidade e a acessibilidade destas ferramentas de simulação são outras questões importantes a serem tratadas [25].

Considerando-se os argumentos expostos anteriormente, o objetivo do presente trabalho é desenvolver um sistema que usa tecnologias *web* para solução de problemas 1D e 2D, estacionários e dependentes do tempo, envolvendo EDPs. O método numérico utilizado para resolver estas equações é o método DDF em coordenadas cartesianas 1D e 2D e generalizadas 2D. Neste último caso, foi considerado também um método de geração de malhas computacionais em coordenadas generalizadas. Para isto, foi desenvolvido um *front-end* de interface com o usuário, um *web service* de comunicação em tempo de execução entre o cliente e o servidor e um *back-end* de processamento dos cálculos escalável em servidores remotos.

Este trabalho espera contribuir no ramo da Ciência da Computação, da matemática e física computacional, buscando fornecer serviços necessários para que pesquisadores e cientistas da área possam resolver problemas reais de várias grandezas temporais e espaciais. Além disso, espera-se que seja possível tornar o conhecimento adquirido de cada usuário acessível por todos. Para isso, não basta ter somente um algoritmo que realize todos os cálculos, mas também um meio de acompanhar o avanço das tecnologias computacionais, como os serviços fornecidos pela *Internet* e dispositivos portáteis, os quais estão tornando-se bastante populares entre os usuários devido a sua praticidade.

Outra contribuição esperada é que o sistema proposto minimize as limitações dos simuladores de serem executados em dispositivos que não tenham um alto desempenho no processamento. Com isso, o trabalho pode mostrar também um modelo de uma arquitetura aberta e normalizada para a simulação de diversos tipos de métodos numéricos, tornando um sistema flexível para adaptação e ajuste em variadas situações reais, independente do método numérico a ser aplicado. Dessa forma, pode-se demonstrar um meio para ampliar o estado da arte da área dos serviços fornecidos pela *web* em conjunto com a matemática/física computacional.

Este trabalho está organizado da seguinte forma:

- No Capítulo 2 será apresentada a revisão bibliográfica e trabalhos relacionados aos métodos das diferenças finitas e simulações em sistemas baseados em *web*;

- No Capítulo 3 são apresentadas as EDPs utilizadas neste trabalho, bem como as suas discretizações por diferenças finitas;
- No Capítulo 4 serão apresentados os métodos de geração de malhas e o método DDF em coordenadas generalizadas;
- No Capítulo 5 será dada uma descrição detalhada dos principais módulos e da arquitetura do sistema *web*;
- No Capítulo 6 são apresentados os resultados obtidos;
- No Capitulo 7 serão apresentadas as conclusões deste trabalho e as propostas para trabalhos futuros.

2 REVISÃO BIBLIOGRÁFICA

Diversos autores desenvolveram algoritmos para resolução de EDPs, a partir dos quais foram explorados implementações e melhoramento dos métodos numéricos. Paralelo à isso, outros autores exploraram ferramentas computacionais para execução de simulações, desde sistemas *web* até serviços móveis. Considerando-se isso, esta seção apresenta uma revisão de literatura sobre o estado da arte no ramo de simulação numérica e na solução de problemas reais com o uso das tecnologias computacionais mencionadas, para depois contextualizar o presente trabalho com a contribuição dos trabalhos revisados.

2.1 FERRAMENTAS DESKTOP E MÉTODOS DE SOLUÇÃO DE EDPS

Dentre as aplicações *desktop* mais recentes e relevantes no ramo das simulações numéricas, pode-se citar o trabalho de Reimer e Cheviakov [26]. Os autores desenvolveram um simulador em Matlab, um *software* voltado para área de computação científica e métodos numéricos. O simulador serviu para resolver a equação de Poisson através das diferenças finitas, com controle de refinamento de malhas. A vantagem da aplicação é que ela explora outros tipos de sistemas de coordenadas em 2 e 3 dimensões espaciais, tais como as cilíndricas e esféricas, além das retangulares.

Outras pesquisas desenvolveram aplicações de EDPs, que vão além do desenvolvimento de sistemas de simulação. Questões como a estabilidade numérica durante o processo de simulação, erros numéricos e taxas de convergências também fundamentam resultados mais precisos. Han e Dai [27], por exemplo, focaram no desenvolvimento de novos esquemas de soluções por diferenças finitas para resolver estas questões de estabilidades numéricas.

A vantagem do modelo desenvolvido por Han e Dai [27] é que ele considera diversos tipos de condições de contorno, tais como Neumann e Dirichlet. Este novo esquema realiza um ajuste da localização dos pontos no interior da malha os quais são próximos às condições de contorno. A metodologia aplicada por eles é a combinação do esquema de discretização de Crank-Nicholson com a extrapolação de Richardson.

A criação de novos modelos numéricos para resolução de EDPs pode ser auxiliada com o uso de rotinas e sub-rotinas existentes. O reaproveitamento destes, os quais estão disponíveis em diversas bibliotecas e pacotes, pode ser um passo para a otimização de modelos matemáticos. Amodio e Settanni [28] propuseram uma rotina em Matlab denominada HOFid_UP baseada em diferenças finitas que resolve problemas de perturbações de segunda ordem.

A rotina HOFid_UP consegue resolver diversos problemas lineares e não-

lineares, uma vez que o código-fonte é baseado no método *upwind* generalizado para diferenças finitas de ordens superiores. A partir disso, verificações de desempenho demonstraram em uma aproximação dos resultados próximas à precisão de máquina. Apesar da precisão adquirida, o código-fonte pode ser otimizado e comparado com outras linguagens de programação.

Alguns autores também utilizaram os recursos do *software* Matlab, tais como Iftode [29], que propôs uma implementação de equações de Maxwell para a propagação de ondas bidimensionais. Como a solução analítica desta equação para o caso 2D é desconhecida e de difícil cálculo, o método utilizado nesta implementação foi a DDF. Considerando-se isto, o uso do *software* foi eficiente para a simulação do fenômeno, uma vez que o Matlab mostrou-se bastante versátil na implementação de métodos matriciais.

Um trabalho mais recente [30] desenvolveu um *software* para simulação de efeitos eletromagnéticos utilizando a aproximação por DDF. O algoritmo foi programado utilizando o ambiente Scilab, similar ao Matlab porém de licença GPL (*General Public License*). A vantagem do *software* desenvolvido é que os resultados das simulações conseguem ser próximos aos dos experimentos laboratoriais, além de explorar geometrias 2D e 3D, estas sendo difíceis de serem simuladas experimentalmente.

2.2 Tecnologias Web e Serviços em Nuvem

As aplicações locais citadas na seção 2.1 mostraram-se eficazes na modelagem de problemas por meio de simulações numéricas. Entretanto, elas são restritas no sentido que o usuário deve ter um *software* específico instalado em sua máquina, normalmente de caráter acadêmico e científico, tais como o Matlab, Scilab e o Octave. Estas aplicações locais muitas vezes dependem de um dispositivo com requisitos ideais para simulações de problemas de grande porte.

Para superar as limitações de processamento e do acesso aos serviços, diversos sistemas baseados em *web* foram desenvolvidos. Um dos primeiros que utilizaram abordagens *web* para simulações numéricas foi de Afonseca et al. [31]. Nesta pesquisa, os autores propuseram um serviço para resolver EDPs 2D, cuja visualização de malhas gráficas é realizada utilizando OOCSMP (*Object Oriented Continuous Simulation Language*), uma linguagem de orientação à objetos própria para resolução de EDPs.

O foco do desenvolvimento do OOCSMP foi o uso de conceitos de orientação a objetos para representação de componentes de simulação em diversas classes e o uso da computação distribuída para dividir o problema em diversas partes menores. Porém, para este trabalho, não foram considerados requisitos como a interoperabilidade, que é a capacidade de um sistema se comunicar com outro de uma forma transparente. Além disso, a reusabilidade, que é o uso de componentes em diferentes contextos, também deixou de ser considerada.

Mais recentemente, Ari e Muhtaroglu [21] desenvolveram um serviço em nuvem (*Software as a Service* – SaaS) de análise linear e não-linear de estruturas mecânicas utilizando o método dos elementos finitos. O foco deste SaaS foi a solução de problemas

estáticos e foi adotado um esquema de escalonamento inteligente em ambientes CPU *multi-core* e de computação distribuída. Este esquema foi adotado para seleção dinâmica de parâmetros, o que ocasionou uma melhoria de desempenho de cerca de oito vezes, comparado com ferramentas de elementos finitos tradicionais.

Xiao-Ping e Ru-Fu [15] focaram no *design* ideal e na análise remota de elementos finitos de produtos de formatos complexos, usando a metodologia de computação em nuvem. Neste trabalho, os autores propuseram um *web service* de quatro camadas, baseados em recursos de conhecimento distribuído, com o uso de um *software* de análise comercial. Resultados mostraram que o serviço proporciona uma modelagem local e rápida através da análise remota e exposição local.

Pode-se notar que as tecnologias baseadas em *web* foram bastante aplicadas em simulações numéricas e que contribuem para a solução de problemas reais. O SaaS proposto por Ari e Muhtaroglu [21], apesar de ter sido desenvolvido para um caso específico, pode ser ampliado para outras aplicações relacionadas a transferência de calor, dinâmica dos fluídos, acústica e eletromagnetismo. Já o *web service* de Xiao-Ping e Ru-Fu [15] se caracteriza em um *framework* flexível e aberto para diferentes serviços.

Além das aplicações matemáticas e físicas supracitadas, serviços voltados para outras áreas científicas foram desenvolvidos. Barboza et al., por exemplo, propuseram um sistema para executar aplicativos que exigem respostas em tempo de execução, como jogos digitais, em um servidor remoto [32]. Os jogos podem ser executados em computadores pessoais, dispositivos móveis e TVs digitais, cujos desempenhos dependem exclusivamente dos requisitos do servidor remoto. O serviço proposto, apesar de ser diferente de uma aplicação de simulações numéricas, possui uma grande importância no estado da arte de sistemas *web*.

Singh et al. desenvolveram um projeto recente de um *web site* educacional para simulação do fenômeno da interferometria [5]. Neste trabalho, os autores disponibilizaram um laboratório virtual remoto em que pesquisas, experimentos físicos reais e visualização de imagens óticas, oriundas de aspectos de processamento de imagens, podem ser facilmente realizados de qualquer lugar. Este laboratório é a base para a pesquisa colaborativa através de recursos compartilhados.

Perus et al. [33] desenvolveram um sistema para simulação de curvas IDA (*Incremental Dynamics Analysis*). A aplicação *web* proposta pode ser acessada a qualquer momento de qualquer lugar, facilitando dessa forma a acessibilidade e a disponibilidade do serviço. Em contrapartida, o banco de dados é limitado a um número pré-determinado de registros de movimentos da Terra, essenciais na simulação de curvas IDA.

Juntamente com o trabalho de Singh et al. [5] citado anteriormente, o sistema de simulação de curvas IDA [33] possui métodos matemáticos (interpolação linear multidimensional), de alto custo computacional, assim como o DDF. Para isso foi utilizados um banco de dados para o armazenamento de respostas. Este banco de dados torna-se útil para futuras consultas de simulações já realizadas.

2.3 MOBILIDADE DOS SIMULADORES

Uma das questões bastante abrangentes na área das simulações é a da adaptabilidade e da disponibilidade de serviços. Por meio destas, diversas arquiteturas voltadas para serviços e aplicações móveis foram desenvolvidas nos últimos anos, tais como de Dumont e Mourlin [17]. Estes autores propuseram uma nova arquitetura paralela adaptativa para dispositivos móveis, em que cada processador realiza uma determinada operação decomposta em uma simulação numérica.

A arquitetura proposta denominada Jini [17] pode ser validada através de simulações como o cálculo de um dígito decimal de números transcendentes e operações matriciais. Suas vantagens são a flexibilidade, auto-configuração da rede e tolerância a falhas devido ao ajuste automático ao ambiente computacional. Contudo, os testes realizados limitaram-se à computação de matrizes de ordem baixa (10), devido à limitação da *hardware* que os dispositivos possuem.

Sala [13] demonstrou em sua pesquisa a importância da mobilidade de diversos sistemas de simulações e a praticidade que ela pode fornecer para os usuários, descrevendo uma arquitetura computacional para simulação de fenômenos físicos. O fenômeno simulado foi o de propagação de luz em meio linear e não-linear e mostrou-se útil tanto na área de ensino como a de pesquisa. Resultados mostraram que os processamentos eram realizados em alguns minutos e o desempenho para problemas mais complexos foi considerado aceitável.

Uma vantagem da arquitetura móvel proposta por Sala [13] é o fornecimento de uma interface ao usuário para permitir a execução de problemas envolvendo propagação de luz. Entretanto, o tempo de execução das simulações ficaram limitadas à arquitetura dos dispositivos móveis utilizadas nos testes (ARM9 e ARM11), restringindo a problemas com *grids* de no máximo 100 pontos. Dessa forma, o desempenho dos processamentos variava conforme o tipo de arquitetura utilizada, cuja velocidade de processamento era limitada entre 235 MHz e 600 MHz.

Outro exemplo do uso da mobilidade em simulações é o trabalho de Teh e Koh [14], que desenvolveram aplicativos de *smartphones* para simulação da qualidade de água, a qual é descrita através de EDPs. Os autores aproveitaram o uso destes dispositivos com *hardwares* que contribuem para um bom desempenho das execuções. O modelo utilizado para simulação da qualidade da água foi o E2ALGAE, o qual mede a qualidade de lagos, rios e estuários e diversos fenômenos, como a eutrofização em uma massa de água. O aplicativo foi testado em um curso de modelagem matemática, onde a viabilidade do mesmo foi analisada pelos próprios estudantes. A contribuição final do aplicativo móvel é a sua facilidade na coleta de dados específicos e a interpretação dos mesmos auxiliando na tomada de decisões.

2.4 Soluções para Problemas Reais

Os objetos do mundo real não podem ser modelados eficientemente utilizando coordenadas retangulares, devido à sua natureza de domínio irregular. Para isso, é necessário primeiramente aplicar metodologias de modo que este domínio possa ser modelado na forma de uma malha computacional. Com a malha gerada, pode-se usufruir de diversos métodos de discretização do domínio espacial, com algumas readaptações matemáticas.

Xu e Yang [34], por exemplo, desenvolveram um algoritmo incremental de geração de malhas, que utiliza a técnica de triangularização de pontos nas nuvens. Neste caso, os pontos são colocados mais próximos possíveis um dos outros, fazendo que o sistema faça a ligação dos pontos formando a malha final. Resultados do trabalho mostraram que o método da triangularização de malhas consegue modelar diversos objetos geométricos de modo eficiente e estável, além de lidar com uma grande nuvem de pontos dispersos.

Em uma abordagem semelhante desenvolvida por Xu e Yang [34], Notsu et al. [35] apresentaram um gerador de malhas auto-organizáveis baseados no modelo Gray Scoot. Este modelo consiste na criação de um padrão de pontos em uma região específica de parâmetro, procurando manter uma equidistância entre os pontos vizinhos. Tal modelo mostrou-se ideal para o processo de suavização da malha, através de adição e remoção local dos pontos, realizando um ajuste automático do sistema.

Tanto a metodologia de Xu e Yang [34] como o gerador de malhas de Notsu et al. [35], são propostas que se mostraram eficientes na questão de otimização de processamento. Os resultados procuraram reduzir a sobreposição de pontos e lacunas entre os nós. Além disso, os trabalhos citados podem ser ampliados para resolverem problemas mais complexos, como os casos tridimensionais.

Com relação ao refinamento de malhas computacionais, autores como Neves et al. [6] estabeleceram uma métrica com auto grau de refinamento de objetos 3D, baseado no método dos elementos finitos (MEF). Neste trabalho, os autores implementaram um código-fonte aberto para resolver EDPs envolvendo MEF através da tetraedalização automática, um processo computacionalmente custoso. Testes de desempenho mostraram a eficiência da execução do código aberto, até mesmo para objetos de volumes irregulares.

Na questão da simulação de um modelo matemático sobre uma malha computacional discretizada em coordenadas generalizadas, pode-se citar um trabalho recente que propôs uma solução numérica da equação de difusão de superfícies planas generalizadas [36]. O método numérico utilizado neste trabalho foi o de volumes finitos utilizando conceitos de métricas de transformação para mapear o domínio físico e real para o transformado. Comparações com soluções analíticas da equação de difusão e testes com casos reais mostraram a viabilidade da solução numérica proposta.

Pode-se notar que todos os métodos e algoritmos citados anteriormente são soluções ideais, podendo ser estudadas mais profundamente para serem aplicados em ambientes computacionais, como a *web*. Neste caso, pesquisas recentes [37] focaram no desenvolvimento

de *frameworks* para o armazenamento de malhas 3D, através de um controle de revisão e mesclagem de imagens no repositório (banco de dados), o que facilita as operações de leitura e escrita. Isso é realizado através de um cliente *web* (por HTML5 e WebGL), responsável pela visualização remota do conteúdo armazenado no repositório em um ambiente *web* [38].

Weng [10] propôs um *web site* denominado WebDFEA para o pós-processamento de estruturas analisadas via método dos elementos finitos. O trabalho focou na possibilidade de visualizar estas estruturas em uma página da *web*, porém com as mesmas possibilidades de manipulações gráficas e controles em um software de computação gráfica, como os do tipo CAD (*Computer Aided Design*). O desenvolvimento do WebDFEA foi realizando utilizando tecnologias HTML para a interface com o usuário e banco de dados para o armazenamento de modelos para análises de resultados.

Por fim, Walker e Chapra [11] propuseram um modelo *web* cliente-servidor voltado para simulações de propósito geral. O modelo proposto serviu para o desenvolvimento de sistemas *web* com interfaces gráficas para aplicações reais, como a modelagem da demanda de oxigênio em rios. Desta forma, mostra-se a eficiência de tecnologias *web* para uma aplicação específica fornecendo interatividade com o usuário, embora o modelo desenvolvido possa ser ampliado para acesso *offline* e armazenamento local de dados.

Diante do contexto apresentado, a questão da integração de tecnologia *web* no campo da simulação foi bastante abordada por diversos autores [24], sendo que aplicações *web* específicas foram desenvolvidas por este meio [10, 11]. Entretanto, os sistemas *web* desenvolvidos podem ser ampliados para resolução de múltiplas situações e problemas reais de modo geral, e conforme descrito, estas podem ser modeladas matematicamente através de EDPs [2]. A maioria destas são resolvidas via métodos numéricos/computacionais, como a DDF, além de envolverem passos de sub-processamentos como a geração de malhas computacionais.

Conforme pode ser visto, as EDPs foram bastante utilizados no contexto de modelagem e simulação, e diversos métodos numéricos foram aplicados para resolvê-las, especialmente o método DDF [26, 27, 4]. Independente da aplicação final, mesmo para jogos digitais [32] e serviços educacionais [5], o uso de tecnologias *web* permite que os usuários realizem somente as tarefas básicas, como a inserção dos parâmetros de entrada. Dessa forma, outros serviços, como a conversão *online* de arquivos, estão disponíveis via *web* ou na nuvem.

A junção dos conceitos mencionados sobre os métodos numéricos com aplicações *web* e suas respectivas ferramentas interativas disponíveis podem formar uma arquitetura *web* genérica que permite também a acessibilidade e a mobilidade de serviços [14]. Esta arquitetura pode ser base para sistemas que servem tanto para a solução como o armazenamento de problemas reais, os quais são propostas específicas deste trabalho.

3 DIFERENÇAS FINITAS EM COORDENADAS CARTESIANAS

Diversos problemas e fenômenos físicos são descritos e modelados através de EDPs, que são um conjunto de equações que contém funções de diversas variáveis desconhecidas junto com as respectivas derivadas parciais. Estas derivadas parciais representam a taxa de variação de uma função, que na maioria das vezes relacionam a dimensão espacial e suas derivadas com a variação no tempo. Tais EDPs também são representáveis na forma de problemas de valores iniciais (PVI) e problemas de valores de fronteira (PVC).

Neste trabalho, as EDPs a serem aplicadas no *web service* envolverão derivadas parciais de até no máximo segunda ordem e de até duas dimensões espaciais com uma temporal. Uma forma genérica de uma EDP de segunda ordem em duas dimensões é definida como $aU_{xx}+bU_{xy}+cU_{yy}+dU_x+eU_y+fU=0$, em que a, b, c, d, e e f (em que $a^2+b^2+c^2\neq 0$) são constantes ou funções das variáveis x e y, que também podem definir a linearidade ou não da EDP. Os diferentes valores dos coeficientes a, b, c, d, e e f definem diferentes tipos de EDPs. Utilizando-se do conceito de cônicas, podemos classificá-las da seguinte forma:

- Se ($b^2-ac<0$) a EDP é classificada como elíptica. Por exemplo: a equação de Laplace;
- Se ($b^2-ac=0$) a EDP é classificada como parabólica. Por exemplo: a equação da condução do calor;
- Se ($b^2-ac>0$) a EDP é classificada como hiperbólica. Por exemplo: a equação da onda e da advecção.

Todos os tipos mencionados acima serão considerados e aplicados no *web service*. Para isso, o método numérico para a resolução das EDPs foi o método DDF [39]. Este foi selecionado por ser flexível e de alta precisão, além de permitir a resolução de problemas com domínio temporal e espacial.

A ideia do método DDF é de transformar a resolução de equações diferenciais usando aproximações para as derivadas parciais. Essas aproximações consistem na discretização do domínio espacial em uma malha computacional, definindo um conjunto de pontos $x_i = x_0 + ih$ ($i = 1,2,3\ldots n$, sendo que n é o tamanho da malha) em um domínio x. Para cada um destes pontos da malha, é calculada uma aproximação de uma função e suas respectivas derivadas, utilizando de base os pontos vizinhos x_{i-1} e x_{i+1}.

Para a realização da aproximação das derivadas parciais em um determinado ponto, é utilizada a metodologia de aproximação de funções por séries de Taylor. Essa metodologia utiliza o conceito de informação em um ponto com base em seus vizinhos. Através dessa aproximação por séries de Taylor, obtém-se as fórmulas de discretização por diferenças finitas, aproximando os valores de cada nó da malha discretizada.

3.1 Equação da Condução do Calor 1D e 2D

As equações da condução do calor para os casos 1D e 2D são respectivamente definidas por:

$$\frac{\partial T}{\partial t} = K\frac{\partial^2 T}{\partial x^2}\,(x_0 \leq x \leq x_f), \tag{3.1}$$

$$\frac{\partial T}{\partial t} = K\left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}\right)\,(x_0 \leq x \leq x_f),\,(y_0 \leq y \leq y_f). \tag{3.2}$$

A definição do domínio do problema é feita através da descrição das dimensões espaciais da malha pelos intervalos $[x_0, x_f]$ e $[y_0, y_f]$ (se for o caso 2D). O tempo final de simulação dado por T_{max} define a dimensão temporal do problema. Durante o pré-processamento, as condições de contorno são dadas por $T(x_0, y, t)$, $T(x_f, y, t)$, $T(x, y_0, t)$ e $T(x, y_f, t)$, e a condição inicial dada por $T_0(x, y, 0)$.

A discretização das equações (3.1) e (3.2) consiste em definir os valores N_x, N_y e N_t, que referem-se aos números de partições espaciais em x, em y e temporais em t, respectivamente. Através destes parâmetros, são definidos os espaçamentos $\Delta x = (x_f - x_0)/(N_x - 1)$ e $\Delta y = (y_f - y_0)/(N_y - 1)$ e $\Delta t = (T_{max})/(N_t - 1)$. Já K é constante física de difusividade, e varia conforme o problema a ser aplicado, como o material utilizado.

Após a definição das partições, substituem-se as derivadas parciais de primeira ordem no tempo por equações de diferenças finitas. Estas compreendem de diversos esquemas de integração no tempo, como a diferença avançada ou progressiva (3.3) e a diferença atrasada ou regressiva (3.4). Estas são definidas por:

$$\frac{\partial T}{\partial t} = \frac{T_{i,j}^{k+1} - T_{i,j}^{k}}{\Delta t}, \tag{3.3}$$

$$\frac{\partial T}{\partial t} = \frac{T_{i,j}^{k} - T_{i,j}^{k-1}}{\Delta t}. \tag{3.4}$$

Já para a derivada de segunda ordem no espaço é utilizada o seguinte esquema da derivada centrada para ambos os casos de integração do tempo:

$$\frac{\partial^2 T}{\partial x^2} = \frac{T_{i-1,j}^{k} - 2T_{i,j}^{k} + T_{i+1,j}^{k-1}}{\Delta x^2} \quad \frac{\partial^2 T}{\partial y^2} = \frac{T_{i,j-1}^{k} - 2T_{i,j}^{k} + T_{i,j+1}^{k-1}}{\Delta y^2} \tag{3.5}$$

Considerando T_i^k a aproximação da temperatura em um ponto da malha e $\alpha = (K.\Delta t)/\Delta x^2$ e $j = 0$, obtém-se após a substituição das equações (3.3) e (3.5) em (3.2), o seguinte modelo discretizado no esquema explícito:

$$T_i^{k+1} = (1 - 2\alpha)T_i^k + \alpha(T_{i+1}^k + T_{i-1}^k). \tag{3.6}$$

Este esquema explícito calcula um valor T_i^{k+1} com base nas informações já calculadas em uma instância de tempo anterior. Similarmente, obtém-se após a substituição das equações (3.4) e (3.5), gerando o seguinte modelo discretizado no esquema implícito:

$$(1+2\alpha)T_i^k - \alpha T_{i+1}^k - \alpha T_{i-1}^k = T_i^{k-1}. \tag{3.7}$$

Nota-se claramente que o esquema implícito remete à resolução de um sistema linear $Ax = b$, uma vez que os valores em uma região discreta T_i são calculados simultaneamente. Para o caso 1D, a matriz A é tridiagonal de ordem $(N_x - 1)$ dada por:

$$A = \begin{bmatrix} (1+2\alpha) & -\alpha & 0 & \dots & 0 \\ -\alpha & (1+2\alpha) & -\alpha & \dots & 0 \\ 0 & -\alpha & (1+2\alpha) & \ddots & 0 \\ \vdots & \vdots & \ddots & \ddots & -\alpha \\ 0 & 0 & 0 & -\alpha & (1+2\alpha) \end{bmatrix}$$

Para o esquema 2D, considerando $T_{i,j}^k$ a aproximação da temperatura em um ponto da malha e $\alpha = (K.\Delta t)/\Delta x^2$, $\beta = (K.\Delta t)/\Delta y^2$, obtém-se após a substituição das equações (3.3) e (3.5) em (3.2), o seguinte modelo discretizado no esquema explícito:

$$T_{i,j}^{k+1} = (1-2\alpha-2\beta)T_{i,j}^k + \alpha(T_{i+1,j}^k + T_{i-1,j}^k) + \beta(T_{i,j+1}^k + T_{i,j-1}^k). \tag{3.8}$$

A equação (3.8) do esquema explícito é condicionalmente estável, pois se $\alpha \leq 0,5$ e $\beta \leq 0,5$ não forem satisfeitas, a simulação da distribuição da temperatura $T_{i,j}^k$ estará sujeita a instabilidades e oscilações. Similarmente, obtém-se após a substituição das equações (3.4) e (3.5), gerando o seguinte modelo discretizado no esquema implítico:

$$(1+2\alpha+2\beta)T_{i,j}^k - \alpha T_{i+1,j}^k - \alpha T_{i-1,j}^k - \beta T_{i,j+1}^k - \beta T_{i,j-1}^k = T_{i,j}^{k-1}. \tag{3.9}$$

O método implícito (3.9) é incondicionalmente estável e a garantia da estabilidade remete à resolução de um sistema linear, cuja solução é o valor de $T_{i,j}^k$ no conjunto de pontos interiores à malha. Os problemas 2D neste caso, geram sistemas com matrizes de blocos tridiagonais (formando três diagonais não-nulas) e os problemas 1D, em que $\beta = 0$, formam um sistema linear tridiagonal. Já para o caso 2D, a matriz A é composta por uma matriz de blocos tridiagonais de ordem $(N_x - 1)$, dada por:

$$A = \begin{bmatrix} B_t & B_d & O & O & \dots & O \\ B_d & B_t & B_d & O & \dots & O \\ O & B_d & B_t & B_d & \dots & O \\ O & O & B_d & B_t & \ddots & O \\ \vdots & \vdots & \vdots & \ddots & \ddots & B_d \\ O & O & O & O & B_d & B_t \end{bmatrix}$$

Cada bloco tridiagonal B_t de A é uma sub-matriz de ordem (N_y-1) e B_d uma sub-matriz diagonal da mesma ordem de B_t, dadas por:

$$B_t = \begin{bmatrix} (1+2\alpha+2\beta) & -\beta & 0 & \dots & 0 \\ -\beta & (1+2\alpha+2\beta) & -\beta & \dots & 0 \\ 0 & -\beta & (1+2\alpha+2\beta) & \ddots & 0 \\ \vdots & \vdots & \ddots & \ddots & -\beta \\ O & O & O & -\beta & (1+2\alpha+2\beta) \end{bmatrix}$$

$$B_d = \begin{bmatrix} -\alpha & O & O & \dots & O \\ O & -\alpha & O & \dots & O \\ O & O & -\alpha & \ddots & O \\ \vdots & \vdots & \ddots & \ddots & O \\ O & O & O & O & -\alpha \end{bmatrix}$$

Considerando que O é uma matriz de blocos nulas, da mesma ordem que B_t e B_d, tal sistema é resolvido através de métodos diretos, como a fatoração LU (*Lower-Up*), e métodos iterativos, como o método de Jacobi e Gauss-Seidel.

Quando não consideramos a dependência do tempo t na equação de condução do calor, ela se transforma em uma EDP denominada equação de Laplace. Esta é considerada estacionária e simula a distribuição de calor considerando um tempo ou estado final de simulação. Sua equação diferencial é ilustrada a seguir, considerando a não-intervenção de fontes externas de calor:

$$\bigtriangledown^2 T = \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0\,(x_0 \leq x \leq x_f). \tag{3.10}$$

A discretização da equação de Laplace (3.10) é obtida ao substituir as equações de diferenças finitas nas derivadas correspondentes gera o esquema explícito e implícito da EDP. Após a substituição das equações das diferenças regressivas no espaço (em x e y) e centradas no tempo, obtém-se a seguinte equação discretizada:

$$T_{i,j}^{k+1} = 2(\Delta x^2 + \Delta y^2)T_{i,j}^k + \Delta x^2 T_{i+1,j}^k + \Delta x^2 T_{i-1,j}^k + \Delta y^2 T_{i,j+1}^k + \Delta y^2 T_{i,j-1}^k. \tag{3.11}$$

Pode-se notar que todos os esquemas implícitos de discretização das equações de condução do calor descritas remetem à resulução de um sistema linear do tipo $Ax = b$. Para o *web service*, os processos de resoluções destes sistemas são resolvidos pelo módulo de processamento de cálculos utilizando bibliotecas de álgebra linear de alto desempenho, a serem descritos na seção 5.2.

3.2 Equação da Propagação da Onda Unidimensional

A equação da propagação da onda unidimensional (3.12) considera o tempo final de simulação T_{max}, as condições de contorno $U(x_0,y,t)$ e $U(x_f,y,t)$ e a condição inicial dada por $U_0(x,y,0)$. Ela é definida da seguinte forma:

$$\frac{\partial^2 U}{\partial t^2} = C\left(\frac{\partial^2 U}{\partial x^2}\right) \ (x_0 \leq x \leq x_f). \tag{3.12}$$

A malha é discretizada segundo as partições N_x (espacial em x) e N_t (temporal em t) e os espaçamentos $\Delta x = (x_f - x_0)/(N_x - 1)$ e $\Delta t = (T_{max})/(N_t - 1)$ que definem o tamanho de cada nó da malha discretizada. Além disso, junto com a constante C, deve se considerar a condição de velocidade inicial $g(x)$, denominada por $\frac{\partial U}{\partial t}(x,0)$.

A discretização da equação da onda (3.12) consiste em substituir as mesmas equações de diferenças finitas (3.3) nas derivadas de segunda ordem no tempo t e no espaço x. Em adição, deve-se substituir também a expressão que representa a condição de velocidade inicial $g(x)$ na condição inicial $U_{i,1}$. A equação da onda após o processo de discretização é ilustrada em 3.13, em que $\lambda = (C * \Delta t)/\Delta x$.

$$U_{i,j+1} = 2(1-\lambda^2)U_{i,j} + \lambda^2 U_{i+1,j} + \lambda^2 U_{i-1,j} - U_{i,j-1} \tag{3.13}$$

A equação da onda discretizada (3.13), diferentemente da condução do calor, que utiliza uma resolução de um $Ax = b$, remete à resolução de uma multiplicação matriz-vetor, cuja solução final é dada por $U_{i,j+1} = A * U_{i,j} - U_{i,j-1}$. A matriz A tridiagonal formada pelas seguintes diagonais: principal $[2(1-\lambda^2), 2(1-\lambda^2), 2(1-\lambda^2) \ldots 2(1-\lambda^2)]$, superior e inferior $[\lambda^2, \lambda^2, \lambda^2 \ldots \lambda^2]$. Neste caso, a operação de álgebra linear é definida por:

$$U_{i,j+1} = \begin{bmatrix} 2(1-\lambda^2) & \lambda^2 & O & \ldots & O \\ \lambda^2 & 2(1-\lambda^2) & \lambda^2 & \ldots & O \\ O & \lambda^2 & 2(1-\lambda^2) & \ddots & O \\ \vdots & \vdots & \ddots & \ddots & \lambda^2 \\ O & O & \lambda^2 & \ldots & 2(1-\lambda^2) \end{bmatrix} * U_{i,j} - U_{i,j-1}$$

A operação matricial descrita é de uma multiplicação de uma matriz tridiagonal com um vetor, resultando em um vetor de tamanho $N_x - 1$. O resultado desta

multiplicação é subtraído pela solução anterior da malha $U_{i,j-1}$, seja ela do estado inicial ou calculada em uma iteração anterior. Para estas operações básicas e elementares, existem bibliotecas de álgebra linear eficientes para o cálculo (veja seção 5.2).

3.3 Equação da Advecção-difusão 1D e 2D

A equação da advecção-difusão, conhecida também como a equação do transporte, no caso unidimensional também considera os parâmetros de dimensões do eixo x, T_{max}, as condições de contorno e inicial e o processo de partição da malha (espaçamento espacial e temporal para definir o tamanho da malha discretizada), definidas para a equação do calor e da onda 1D. Para o caso unidimensional, a equação de advecção-difusão é:

$$\frac{\partial U}{\partial t} = D_x \left(\frac{\partial^2 U}{\partial x^2} \right) - u \frac{\partial U}{\partial x} \ (x_0 \leq x \leq x_f). \tag{3.14}$$

Já o caso bidimensional considera os parâmetros dimensionais similares à equação da condução do calor (de tempo e espaço) e sua forma é dada por:

$$\frac{\partial U}{\partial t} = D_x \left(\frac{\partial^2 U}{\partial x^2} \right) + D_y \left(\frac{\partial^2 U}{\partial y^2} \right) - u \frac{\partial U}{\partial x} - v \frac{\partial U}{\partial y} \ (x_0 \leq x \leq x_f), (y_0 \leq y \leq y_f). \tag{3.15}$$

Além dos parâmetros temporais e espaciais, a equação de transporte considera também a velocidade média na direção x (u) e y (v) e os coeficientes de dispersão D_x e D_y, que simulam respectivamente os fenômenos da advecção e difusão. Para o caso unidimensional, quando as condições de contorno são constantes, a equação possui solução analítica descrita em [40].

A discretização da equação do transporte 1D (3.14) e 2D (3.15) pode ser efetuada utilizando diferenças centrais, progressivas e regressivas nas mais diversas derivadas parciais. A forma utilizada neste trabalho foi o uso de diferenças progressivas na derivada de primeira ordem do tempo, diferenças centrais nas derivadas de segunda ordem no espaço (x, y) e regressivas nas derivadas de primeira ordem no espaço. Com essas substituições, resulta-se na equação de transporte discretizada (3.16) em que deseja-se encontrar os valores de $U_{i,j}$ em um tempo t_k.

$$\mu U_{i,j}^k - (\sigma_x + \delta_x) U_{i-1,j}^k - \delta_x U_{i+1,j}^k - (\sigma_y + \delta_y) U_{i,j-1}^k - \delta_y U_{i,j+1}^k = U_{i,j}^{k-1} \tag{3.16}$$

Na equação (3.16), foram consideradas as seguintes notações: $\mu = 1 + \sigma_x + \sigma_y + 2\delta_x + 2\delta_y$, $\sigma_x = \frac{u\Delta t}{\Delta x}$, $\sigma_y = \frac{v\Delta t}{\Delta y}$, $\beta_x = \frac{D_x \Delta t}{\Delta x^2}$ e $\beta_y = \frac{D_y \Delta t}{\Delta y^2}$. Esta EDP discretizada remete à resulução de um sistema linear do tipo $Ax = b$. A mesma notação de matriz por blocos tridiagonais da equação do calor 2D pode ser aplicada para a equação do transporte 2D, com a diferença que esta não possui mais simetria:

$$A = \begin{bmatrix} B_t & B_s & O & O & \dots & O \\ B_i & B_t & B_s & O & \dots & O \\ O & B_i & B_t & B_s & \dots & O \\ O & O & B_i & B_t & \ddots & O \\ \vdots & \vdots & \vdots & \ddots & \ddots & B_s \\ O & O & O & O & B_i & B_t \end{bmatrix}$$

Cada bloco tridiagonal B_t de A é uma sub-matriz de ordem (N_y-1) e B_s e B_i uma sub-matriz diagonal da mesma ordem de B_t, denotadas por:

$$B_t = \begin{bmatrix} \mu & -\delta_x & O & \dots & O \\ -(\sigma_x + \delta_x) & \mu & -\delta_x & \dots & O \\ O & -(\sigma_x + \delta_x) & \mu & \ddots & O \\ \vdots & \vdots & \ddots & \ddots & -\delta_x \\ O & O & O & -(\sigma_x + \delta_x) & \mu \end{bmatrix}$$

$$B_s = \begin{bmatrix} -\delta_y & O & O & \dots & O \\ O & -\delta_y & O & \dots & O \\ O & O & -\delta_y & \ddots & O \\ \vdots & \vdots & \ddots & \ddots & O \\ O & O & O & O & -\delta_y \end{bmatrix}$$

$$B_i = \begin{bmatrix} -(\sigma_y + \delta_y) & O & O & \dots & O \\ O & -(\sigma_y + \delta_y) & O & \dots & O \\ O & O & -(\sigma_y + \delta_y) & \ddots & O \\ \vdots & \vdots & \ddots & \ddots & O \\ O & O & O & O & -(\sigma_y + \delta_y) \end{bmatrix}$$

Tal sistema é resolvido através de métodos diretos, como a fatoração LU, e métodos iterativos, como o de Jacobi e Gauss-Seidel. Os processos de resoluções destes sistemas são implementados no módulo de processamento de cálculos utilizando bibliotecas de álgebra linear de alto desempenho, a serem descritos na seção 5.2.

4 DIFERENÇAS FINITAS EM COORDENADAS GENERALIZADAS

Em casos de problemas reais, o uso de coordenadas cartesianas nem sempre leva a uma adequação ideal da fronteira do problema a ser resolvido. O domínio físico (x, y) destes problemas, na maioria dos casos, não possui uma geometria regular e bem definida. Essa geometria nem sempre coincide com o domínio da malha computacional em coordenadas cartesianas.

Existem diversos métodos matemático-computacionais para geração de malhas em geometrias irregulares, com a execução de diferentes métodos numéricos para solução destes problemas, tais como os volumes finitos [36], elementos finitos, sistemas de triangularização de nuvens de pontos [34], gerador de malhas quadrilaterais em sistemas multi-estruturais [8]. Pode-se citar também, o método das diferenças finitas com geração de malha em coordenadas generalizadas [7, 36].

O método DDF em coordenadas generalizadas consiste na discretização de um domínio físico qualquer (não necessariamente retangular) em um sistema de coordenadas que mapeia o domínio original para um transformado. O domínio transformado pode ser representado computacionalmente como uma malha regular, porém com os devidos mapeamentos realizados. Essa malha, que pode ser 2D ou 3D, pode ser resolvido utilizando-se o método DDF em diversos esquemas de discretização.

Uma das principais motivações no uso de sistemas de coordenadas generalizadas (SCG), em comparação com outros métodos citados, reside na sua menor dificuldade na implementação e solução dos problemas complexos. Existe também a possibilidade de concentrar os pontos da malha em uma região, quando uma geometria tiver muitas irregularidades, fornecendo mais flexibilidade ao gerar o modelo geométrico.

As informações dos nós discretizados em um SCG utilizam malhas estruturadas, fazendo com que os pontos destas sejam facilmente co-localizados, uma vez que o domínio transformado é facilmente mapeado na geometria do problema (físico). Com isso, tal método é considerado flexível e com um tempo de execução eficiente durante a geração da malha, devido aos elementos analíticos (nós) ordenados, fazendo com que a aplicação de um método numérico seja otimizada.

A solução do método DDF em SGC remete também à resolução de operações complexas, como os sistemas lineares do tipo $Ax = b$. Para se chegar às soluções destes sistemas, foi utilizado o método de Gauss-Seidel [41, 42]. Sua escolha deveu-se exclusivamente pelo fato de que o mesmo é bastante simples de ser implementado, ainda que não seja o mais eficiente. Outros métodos, tais como Jacobi, que é iterativo e a Fatoração LU, que é um método direto, também podem ser utilizados com sucesso.

4.1 GERAÇÃO DE MALHAS EM COORDENADAS GENERALIZADAS

O processo de geração de malhas através de SCG inicialmente consiste em definir os pontos (x,y) do contorno da geometria. Os pontos são interpolados pelo método de *splines* lineares paramétricos, para que seja possível ter conhecimento de todos os pontos do contorno da geometria. O método de interpolação consiste em determinar uma equação da reta para cada par de pontos consecutivos (x_i,y_i) e (x_{i+1},y_{i+1}):

$$y = y_i + \frac{y_{i+1}-y_i}{x_{i+1}-x_i}(x-x_i) \tag{4.1}$$

$$x = x_i + \frac{x_{i+1}-x_i}{y_{i+1}-y_i}(y-y_i) \tag{4.2}$$

Após a definição do contorno da geometria, deve-se efetuar a transformação das coordenadas cartesianas do domínio físico, como foi feita em [7] e [36]. Esta é realizada através de um mapeamento de geometrias descritas por sistema cartesiano (x,y) para o sistema generalizado (ξ,η), conhecido como domínio transformado ou plano computacional. Neste último, as dimensões são fixas de modo que cada um de seus pontos seja mapeado para um ponto do domínio físico equivalente.

Para realizar a geração de malhas, deve-se inicialmente especificar as dimensões da malha N (número de linhas em ξ) e M (número de linhas em η). Neste trabalho, assumiu-se uma malha uniforme, em que a distância entre cada nó da malha no domínio transformado é $\Delta\xi = \Delta\eta = 1$. O método para geração de malhas computacionais aplicado foi do tipo elíptico, descrito pelas equações governantes:

$$\begin{cases} \xi = \xi(x,y), \\ \eta = \eta(x,y) \end{cases} \tag{4.3}$$

As linhas ξ e η descrevem as malhas discretizadas, cujos pontos são as intersecções entre essas linhas, que são determinadas através das equações governantes (4.3). Para resolver as equações, inicialmente deve-se definir em quais condições de fronteiras Γ as coordenadas (x,y) pertencem. Neste caso, as linhas ξ tem início na região Γ_1 e fim na região Γ_3, e as linhas η tem início na região Γ_2 e fim na região Γ_4, conforme pode ser visualizado na Figura 4.1.

A partir da Figura 4.1, pode-se notar que todos os pontos do domínio do plano físico são mapeados para um ponto do plano transformado $\xi(x,y)$ e $\eta(x,y)$. Esse mapeamento é realizado pelas métricas de transformações, denominadas por $\alpha = x_\eta^2 + y_\eta^2$, $\gamma = x_\xi^2 + y_\xi^2$ e $\beta = x_\xi x_\eta + y_\xi y_\eta$ e o Jacobiano $J = (x_\xi y_\eta - x_\eta y_\xi)^{-1}$, que são encarregados de fazer as compensações em virtude da mudança do sistema de coordenadas. Estas métricas são aplicadas

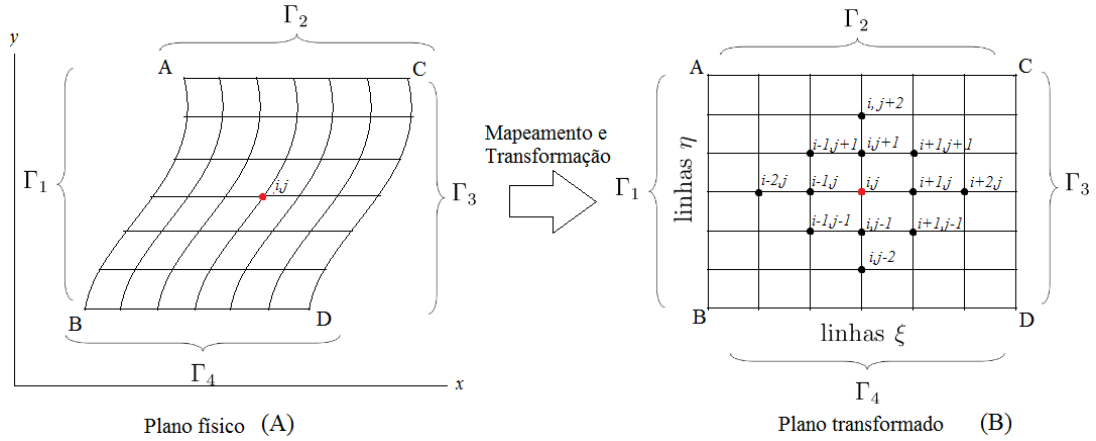


Figura 4.1 – Malha computacional e linhas ξ e η.

na equação governante de geração de malhas (4.4), que calcula um ponto genérico da malha ϕ descrito pela sua posição (x,y):

$$\alpha\phi_{\xi\xi} + \gamma\phi_{\eta\eta} - 2\beta\phi_{\xi\eta} + \frac{1}{J}(P\phi_{\xi} + Q\phi_{\eta}) = 0. \tag{4.4}$$

De 4.4, considera-se que x_ξ e y_ξ são as derivadas parciais de x e y em ξ e x_η e y_η são as derivadas parciais de x e y em η. P e Q são termo-fontes utilizados para realizar atrações entre as linhas na direção ξ e η respectivamente. A equação governante (4.4) calcula cada ponto no interior malha para a posição x e para a posição y. Para isso, todas as derivadas parciais são aproximadas via equações de diferenças finitas, considerando as diferenças centrais:

$$\phi_{\xi\xi} = \frac{\phi_{i+1,j} - 2\phi_{i,j} + \phi_{i-1,j}}{\Delta\xi^2} \tag{4.5}$$

$$\phi_{\eta\eta} = \frac{\phi_{i,j+1} - 2\phi_{i,j} + \phi_{i,j-1}}{\Delta\eta^2} \tag{4.6}$$

$$\phi_{\xi\eta} = \frac{\phi_{i+1,j+1} - \phi_{i-1,j+1} + \phi_{i-1,j-1} - \phi_{i+1,j-1}}{4\Delta\xi\Delta\eta} \tag{4.7}$$

$$\phi_{\xi} = \frac{\phi_{i+1,j} - \phi_{i-1,j}}{2\Delta\xi} \tag{4.8}$$

$$\phi_{\eta} = \frac{\phi_{i,j+1} - \phi_{i,j-1}}{2\Delta\eta} \tag{4.9}$$

$$x_{\xi} = \frac{x_{i+1,j} - x_{i-1,j}}{2} \tag{4.10}$$

$$x_{\eta} = \frac{x_{i,j+1} - x_{i,j-1}}{2} \tag{4.11}$$

$$y_\xi = \frac{y_{i+1,j} - y_{i-1,j}}{2} \tag{4.12}$$

$$y_\eta = \frac{y_{i,j+1} - y_{i,j-1}}{2} \tag{4.13}$$

Através da substituição das equações (4.5 – 4.13) e considerando uma malha estruturada em que $\Delta\xi = \Delta\eta = 1$, gera-se o esquema implícito para a geração de malhas computacionais. Tal sistema foi resolvido utilizando o método iterativo de Gauss-Seidel, cuja solução é o valor de cada ponto da malha x e y, definido por:

$$\begin{aligned}\phi_{i,j} = \frac{1}{A_p}(A_e\phi_{i+1,j} + A_w\phi_{i-1,j} + A_n\phi_{i,j+1} + A_s\phi_{i,j-1} + A_{ne}\phi_{i+1,j+1} + A_{se}\phi_{i+1,j-1} \\ + A_{nw}\phi_{i-1,j+1} + A_{sw}\phi_{i-1,j-1}).\end{aligned} \tag{4.14}$$

A equação discretizada de geração de malhas (4.14) remete à resolução de um sistema linear do tipo $Ax = b$ com oito diagonais não-nulas, uma vez que considera-se oito pontos vizinhos em um ponto (i, j) (Figura 4.1). A matriz A é composta por uma matriz de blocos tridiagonais de ordem $(N-1)$, dada por:

$$A = \begin{bmatrix} B_t & B_s & O & O & \dots & O \\ B_i & B_t & B_s & O & \dots & O \\ O & B_i & B_t & B_s & \dots & O \\ O & O & B_i & B_t & \ddots & O \\ \vdots & \vdots & \vdots & \ddots & \ddots & B_s \\ O & O & O & O & B_i & B_t \end{bmatrix}$$

Cada bloco tridiagonal B_t de A é uma sub-matriz de ordem (M-1). B_t, B_i e B_s são sub-matrizes diagonais da mesma ordem de B_t, dadas por:

$$B_t = \begin{bmatrix} A_p & -A_e & 0 & \dots & 0 \\ -A_w & A_p & -A_e & \dots & 0 \\ 0 & -A_w & A_p & \ddots & 0 \\ \vdots & \vdots & \ddots & \ddots & -A_e \\ 0 & 0 & 0 & -A_w & A_p \end{bmatrix}$$

$$B_s = \begin{bmatrix} -A_n & -A_{ne} & 0 & \dots & 0 \\ -A_{nw} & -A_n & -A_{ne} & \dots & 0 \\ 0 & -A_{ne} & -A_n & \ddots & 0 \\ \vdots & \vdots & \ddots & \ddots & -A_{ne} \\ 0 & 0 & 0 & -A_{ne} & -A_n \end{bmatrix}$$

$$B_i = \begin{bmatrix} -A_s & -A_{se} & 0 & \dots & 0 \\ -A_{sw} & -A_s & -A_{se} & \dots & 0 \\ 0 & -A_{sw} & -A_s & \ddots & 0 \\ \vdots & \vdots & \ddots & \ddots & -A_{se} \\ 0 & 0 & 0 & -A_{sw} & -A_s \end{bmatrix}$$

$$O = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \ddots & 0 \\ \vdots & \vdots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

A solução do problema de coordenadas generalizadas fica completamente solucionado. Conforme citado anteriormente, o método de Gauss-Seidel pode ser utilizado com sucesso. Os parâmetros de solução da equação de geração de malhas (4.14), com este método são: $A_p = 2\alpha + 2\gamma$, $A_e = \alpha + \frac{P}{2J^2}$, $A_w = \alpha - \frac{P}{2J^2}$, $A_n = \gamma + \frac{Q}{2J^2}$, $A_s = \gamma - \frac{Q}{2J^2}$, $A_{ne} = -\frac{\beta}{2}$, $A_{se} = \frac{\beta}{2}$, $A_{nw} = \frac{\beta}{2}$ e $A_{sw} = -\frac{\beta}{2}$.

4.2 Equação da Energia 2D em Coordenadas Generalizadas

A EDP aplicada na malha discretizada em coordenadas generalizadas foi a equação geral da energia 2D. Ela foi baseada na equação geral da conservação 2D, e dependendo da sua parametrização, pode descrever outras, como a de Navier-Stokes, a de transporte e a da distribuição da temperatura [43] e pode ser descrita da seguinte forma:

$$\frac{\partial Q}{\partial t} + \frac{\partial E}{\partial x} + \frac{\partial F}{\partial y} = s. \tag{4.15}$$

Para esta equação (4.15), Q, E e F descrevem parâmetros temporais e espaciais em x e em y respectivamente. Para este trabalho, a EDP dada em 4.15 foi parametrizada para tornar-se a equação do transporte 2D, a qual descreve fenômenos de condução de energia de acordo com o movimento do fluído. Considerando $\sigma = k(\rho C_p)^{-1}$, em que k é a condutividade térmica, ρ a massa específica e C_p o calor específico, α, β e γ as métricas de transformações da geração da malha junto com o Jacobiano das transformações J, a EDP do transporte 2D é descrita da seguinte forma:

$$\begin{aligned} &\left(\frac{1}{J\sigma}\right)\frac{\partial T}{\partial \tau} - J\alpha\frac{\partial^2 T}{\partial \xi^2} - J\gamma\frac{\partial^2 T}{\partial \eta^2} + \rho u\frac{\partial T}{\partial \xi} + \rho v\frac{\partial T}{\partial \eta} \\ &= \frac{\partial}{\partial \xi}\left(J\alpha\frac{\partial T}{\partial \xi}\right) + \frac{\partial}{\partial \eta}\left(J\gamma\frac{\partial T}{\partial \eta}\right) - 2\frac{\partial}{\partial \xi}\left(J\beta\frac{\partial T}{\partial \eta}\right) \end{aligned} \tag{4.16}$$

Na equação 4.16, o valor de ρ é constante, pois ele não varia significativamente de acordo com a temperatura. As fontes externas de produção de energia

também foram desprezadas $(s = 0)$. Os valores de u e v descrevem o movimento do fluído nas direções x e y respectivamente, geralmente calculados a partir da equação de Navier-Stokes [3]. Para o presente trabalho, utilizou-se a denotação $u = v = 0$, que descreve um caso de um fluído parado.

Além dos parâmetros descritos, outros são definidos durante o pré-processamento. Considerando Ω o domínio do interior da malha, para efetuar a simulação é necessário definir:

- O tempo máximo de simulação (T_{max}) e o número de instâncias de tempo (N_t) de comprimento $\Delta t = (T_{max})/(N_t - 1)$
- Condição inicial $T(\xi, \eta, 0)$ em Ω;
- Condição de contorno de Dirichlet $T(\xi, \eta, t)$ em $\partial\Omega_0 \times [0, T_{max}]$, onde $\partial\Omega_0$ é a região do contorno ou bordo;
- Condição de contorno de Neumann $\frac{\partial T}{\partial n}$ ou $\bigtriangledown T.\overrightarrow{n} = \overline{q}$ em $\partial(\Omega - \Omega_0) \times [0, T_{max}]$, onde $\overrightarrow{n}$ é a normal exterior, $\overline{q}$ o fluxo de energia e $\partial(\Omega - \Omega_0)$ o contorno de Ω isolado. As definições dos fluxo de calor para cada Γ são dadas por:
  - Em Γ_1: $\frac{\partial T}{\partial \xi} = -\overline{q} \rightarrow T_{i,j} = (4/3)T_{i,j+1} - (1/3)T_{i,j+2} + (2/3)\overline{q}$;
  - Em Γ_2: $\frac{\partial T}{\partial \eta} = -\overline{q} \rightarrow T_{i,j} = -(1/3)T_{i-2,j} + (4/3)T_{i-1,j} + (2/3)\overline{q}$
  - Em Γ_3: $\frac{\partial T}{\partial \xi} = \overline{q} \rightarrow T_{i,j} = -(1/3)T_{i,j+2} + (4/3)T_{i,j+1} + (2/3)\overline{q}$
  - Em Γ_4: $\frac{\partial T}{\partial \eta} = \overline{q} \rightarrow T_{i,j} = (4/3)T_{i+1,j} - (1/3)T_{i+2,j} + (2/3)\overline{q}$

No caso da condição de contorno de Neumann que descreve o fluxo de energia sobre a borda, diferentes valores de $\frac{\partial T}{\partial n} = k$ definem as propriedades de distribuição de energia. Neste caso, se $k < 0$ o contorno aquece, se $k > 0$ o contorno esfria e se $k = 0$ o contorno é isolado.

Conforme foi realizado nas EDPs da condução do calor, da propagação da onda e do transporte em coordenadas cartesianas, a EDP da energia 2D em coordenadas generalizadas é discretizada aproximando as derivadas parciais através das equações de diferenças finitas:

$$\left(\frac{1}{J\sigma}\frac{\partial T}{\partial \tau}\right)_p = \frac{1}{J_p\sigma}\frac{T_p - T_0}{\Delta\tau} \tag{4.17}$$

$$\left(J\alpha\frac{\partial^2 T}{\partial \xi^2}\right)_p = J_p\alpha_p\left(T_e - 2T_p + T_w\right) \tag{4.18}$$

$$\left(J\alpha\frac{\partial^2 T}{\partial \xi^2}\right)_p = J_p\alpha_p\left(T_e - 2T_p + T_w\right) \tag{4.19}$$

$$\left(\rho u\frac{\partial T}{\partial \xi}\right)_p = 0.5(\rho u)(T_e - T_w) \tag{4.20}$$

$$\left(\rho v\frac{\partial T}{\partial \eta}\right)_p = 0.5(\rho v)(T_n - T_s) \tag{4.21}$$

$$\left(\frac{\partial}{\partial \xi}\left(J\alpha\frac{\partial T}{\partial \xi}\right)\right)_p = 0.25(J_e\alpha_e - J_w\alpha_w)(T_e - T_w) \tag{4.22}$$

$$\left(\frac{\partial}{\partial \eta}\left(J\gamma\frac{\partial T}{\partial \eta}\right)\right)_p = 0.25(J_n\gamma_n - J_s\gamma_s)(T_n - T_s) \tag{4.23}$$

$$\left(2\frac{\partial}{\partial \xi}\left(J\beta\frac{\partial T}{\partial \eta}\right)\right)_p = 0.5(J_e\beta_e)(T_{ne} - T_{se}) - 0.5(J_w\beta_w)(T_{nw} - T_{sw}) \tag{4.24}$$

Através da substituição das equações (4.17 – 4.24) em (4.16), e considerando uma malha estruturada em que $\Delta\xi = \Delta\eta = 1$, gera-se o esquema implícito para a equação da energia. Tal esquema remete à resolução de um sistema linear do tipo $Ax = b$ com oito diagonais não-nulos, uma vez que consideram-se oito pontos vizinhos na região (i, j) – Figura 4.1. Este sistema foi resolvido utilizando o método iterativo de Gauss-Seidel, cuja solução é o valor $T_{i,j}$ em cada ponto específico da malha, definida por:

$$\begin{aligned}\frac{1}{A_p}(&A_eT_{i+1,j} + A_wT_{i-1,j} + A_nT_{i,j+1} + A_sT_{i,j-1} + A_{ne}T_{i+1,j+1} \\ &+ A_{se}T_{i+1,j-1} + A_{nw}T_{i-1,j+1} + A_{sw}T_{i-1,j-1} + A_0T_0).\end{aligned} \tag{4.25}$$

Desta equação de cálculo de transporte discretizada (4.25), tem-se os seguintes valores dos parâmetros: $A_p = \frac{1}{J_p\sigma\Delta t}$, $A_e = J_p\alpha_p + 0.25(J_e\alpha_e - J_w\alpha_w)$, $A_w = J_p\alpha_p - 0.25(J_e\alpha_e - J_w\alpha_w)$, $A_n = J_p\gamma_p + 0.25(J_n\gamma_n - J_s\gamma_s)$, $A_s = J_p\gamma_p - 0.25(J_n\gamma_n - J_s\gamma_s)$, $A_{ne} = -0.5J_e\beta_e$, $A_{se} = 0.5J_e\beta_e$, $A_{nw} = 0.5J_w\beta_w$, $A_{sw} = -0.5J_w\beta_w$ e $A_0 = \frac{1}{J_p\sigma\Delta\tau}$. Devido à semelhança da solução via Gauss-Seidel com a equação governante de geração de malhas, a mesma matriz descrita para geração de malhas pode ser utilizada para representar o sistema linear.

5 ARQUITETURAS E MÓDULOS DO SISTEMA WEB

O sistema *web* proposto é constituído de quatro módulos (Figura 5.1): 1/ um esquema de comunicação cliente-servidor, 2/ interface gráfica com o usuário, 3/ aplicação de cálculos e 4/ módulo de banco de dados. Este sistema é composto de um *front-end*, para a comunicação e interação com os clientes e de um *back-end*, responsável pela execução dos aplicativos de cada usuário solicitante. A arquitetura em detalhes está ilustrada na Figura 5.2.

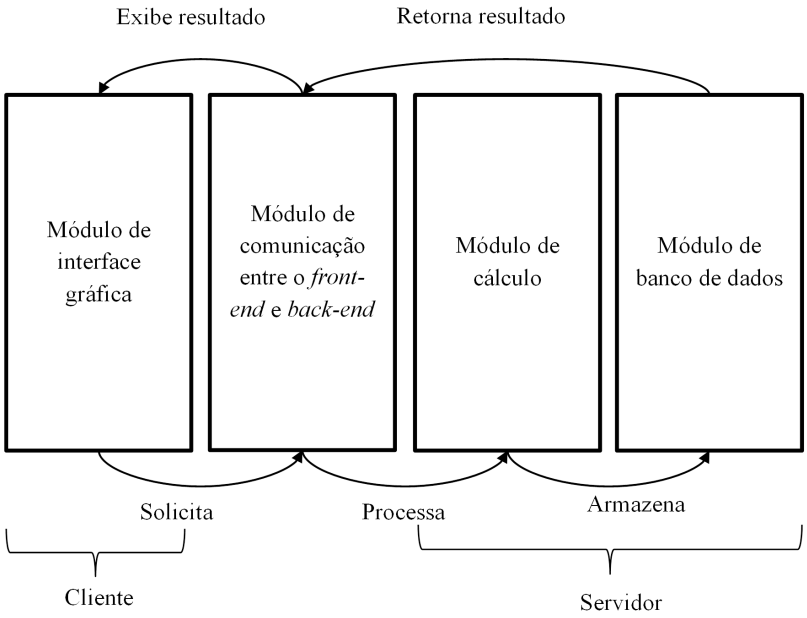


Figura 5.1 – Principais módulos do sistema *web*.

De acordo com a Figura 5.1, o web-service terá quatro módulos principais:

- Comunicação *front-end*/*back-end*: responsável por receber as solicitações do cliente e efetuar uma chamada ao servidor remoto para processar os cálculos. É o principal módulo do sistema pois realiza a integração e serve de ponte de ligação entre os demais.

- Cálculo e processamento: aplicativo do servidor remoto para realizar todas as operações das discretizações por diferenças finitas, além de armazenar os arquivos de dados dos usuários, enviar os resultados ao banco de dados e receber as solicitações do usuário.

- Gerenciamento de banco de dados: possui tabelas de armazenamento de informações relativas aos usuários, solicitações e às EDPs resolvidas.

- Interface gráfica com o usuário: parte do sistema onde o usuário realiza o *login*, a solicitação do cálculo e a visualização dos resultados da simulação e da malha computacional. Geralmente realizado em um *web browser*.

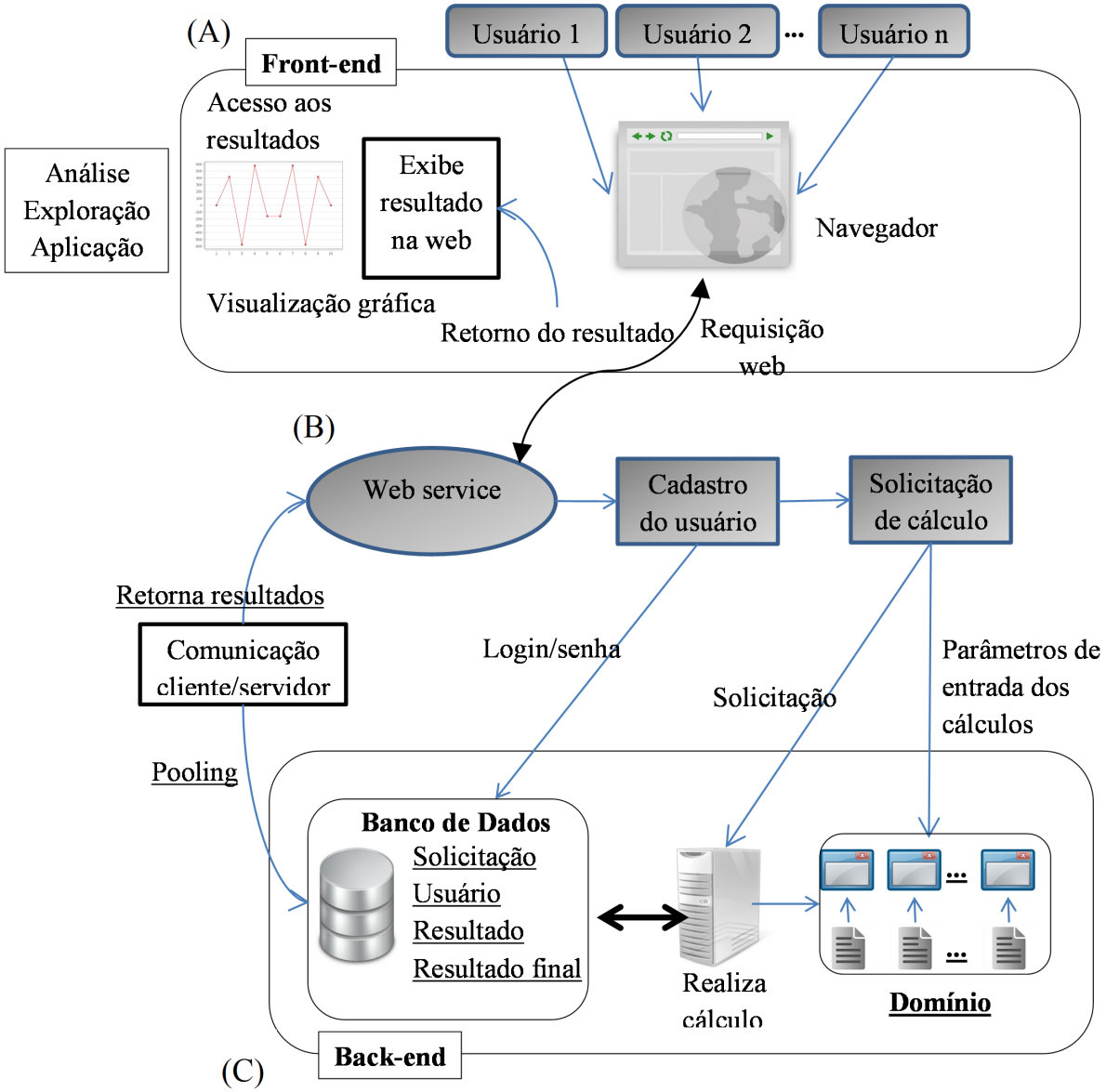


Figura 5.2 – Arquitetura do sistema *web*.

Os módulos do sistema *web* (Figura 5.1) serviram de base para a construção do sistema completo (Figura 5.2). O *front-end* consiste na interface *web* de entrada dos usuários, bem como na exibição dos resultados para análise, exploração e aplicação em um caso real (Figura 5.2A) e o *back-end* consiste no gerenciamento de banco de dados, no armazenamento de cálculos e no domínio de armazenamento do aplicativo e dos arquivos de entrada do usuário (Figura 5.2C). Para a comunicação entre o *front-end* e o *back-end* foi feito o *deployment* de um *web service* que controla os cadastros e as solicitações dos usuários e a consulta em tempo de execução dos respectivos problemas no banco de dados (Figura 5.2B).

5.1 TECNOLOGIAS E IMPLEMENTAÇÕES UTILIZADAS

O sistema *web* descrito na Figura 5.2 foi desenvolvido utilizando múltiplas tecnologias, cujo mapa pode ser visualizado na Figura 5.3. A linguagem JAX-WS (*Java API for XML Web Services* – API Java para Serviços Web XML) foi utilizada para a criação do *web service*, no ambiente de programação NetBeans IDE (Ambiente de Desenvolvimento Integrado) como aplicações em Java Servlet (Figura 5.3B). A razão da escolha desta IDE é pelo fato dela fornecer diversas ferramentas e facilidades de desenvolvimento de aplicações, sem a necessidade de instalação de recursos adicionais. Além disso, o NetBeans fornece uma gama de opções, tais como desenvolvimento de GUI no estilo WYSIWYG (*what you see is what you get*).

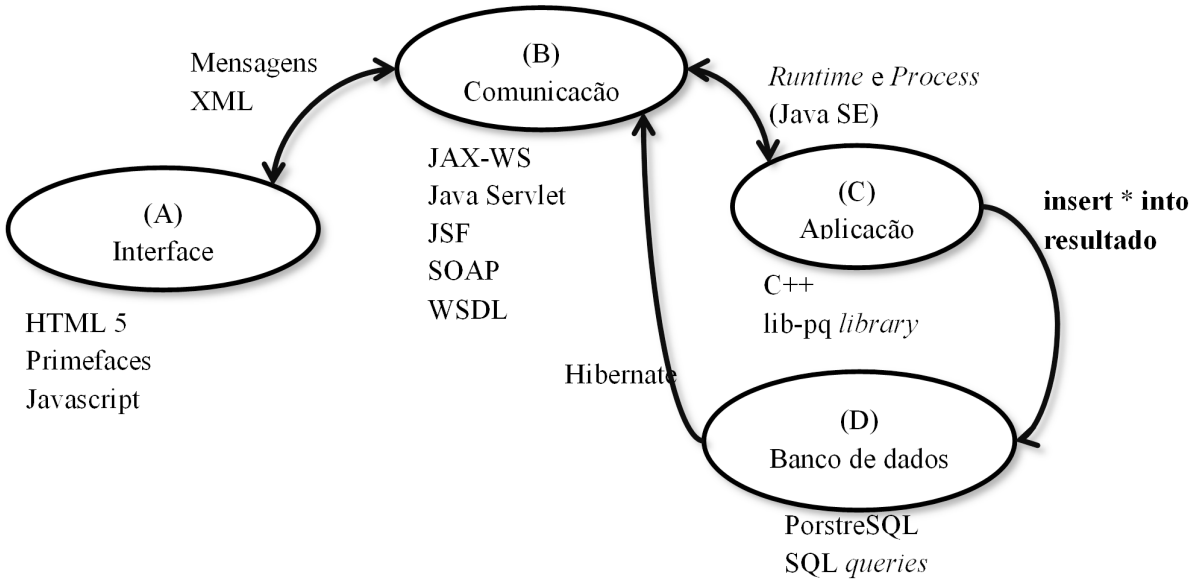


Figura 5.3 – Mapa de tecnologias computacionais utilizadas para o desenvolvimento do sistema.

O *deployment* dos serviços foi realizado no servidor GlassFish 3.1 [1], uma vez que este é um *servlet container* com um servidor de aplicação, sendo útil para integração de *web services* em uma arquitetura orientada a serviços. Já o GUI foi desenvolvido com o auxílio do Primefaces [2] (junto com o NetBeans), um *framework* útil para o desenvolvimento de páginas da *web* dinâmicas. O *framework* JSF (*Java Server Faces*) foi aplicado para realizar o controle de sessões e a conexão de MVC (*Model View Controller*) com a interface (Figura 5.3B).

Para o armazenamento e consulta de todas as informações do usuário e dos cálculos e saídas processados pelo servidor, o banco de dados PostgreSQL 9.2 foi utilizado (Figura 5.3D). O motivo da escolha deste sistema de gerenciamento de banco de dados (SGBD) deve-se ao fato dele ser robusto e *freeware*, além de utilizar uma das linguagens mais conhecidas para SGBD, a SQL (*Structured Query Language*). Outra motivação é a facilidade da sua

[1] http://glassfish.java.net/
[2] http://primefaces.org/

incorporação com diversas linguagens de programação, tendo uma interface adaptada para cada uma delas.

A comunicação com o banco de dados foi realizado utilizando o *framework* Hibernate, para também permitir uma comunicação entre o *front-end* e *back-end*, a fim de exibir os resultados da simulação em tempo de execução. Este último *framework* permite também o manuseio de informações no banco de dados de uma forma automatizada, sem a necessidade de gerar grandes e onerosos *scripts* em linguagem SQL. Estes *scripts* são encapsulados pelo Hibernate, com o intuito de deixá-lo transparente ao desenvolvedor.

A aplicação do cálculo das diferenças finitas no servidor foi desenvolvido na linguagem C/C++, uma linguagem de propósito geral que funciona em múltiplas plataformas (Figura 5.3C). Neste foi feita a integração das bibliotecas do PostgreSQL para a gravação dos resultados no banco de dados e das funções da biblioteca LAPACK (*Linear Algebra Package*[3]) e BLAS (*Basic Linear Algebra Subroutines*[4]) para a realização dos cálculos de álgebra linear de modo mais eficiente e otimizado.

A interface do gerador de malhas em coordenadas generalizadas para *web* foi implementado utilizando Javascript (Figura 5.3A). Além disso, a linguagem Javascript permite o desenvolvimento de aplicações interativas com o usuário, além de ser livre e com uma programação independente da plataforma utilizada. Para este trabalho em específico, foi utilizado também as funções da biblioteca Canvas, para o desenho de figuras geométricas e de malhas.

5.2 Módulo de Processamento de Cálculos

Os sistemas lineares provenientes do processo de discretização das EDPs do calor e do transporte 2D são resolvidas com o método da Fatoração LU, adaptado para matrizes de formatos específicos, da biblioteca LAPACK. Esta é uma biblioteca de funções de álgebra linear de alto desempenho computacional que executa operações de vetores de forma eficiente em memória compartilhada e processadores paralelos. No caso 1D utilizou-se a sub-rotina SGTSV, otimizada para sistemas tridiagonais, cujo formato de armazenamento da matriz é em três vetores (*DL* – diagonal inferior, *D* – diagonal principal e *DU* – diagonal superior).

Já para o caso 2D das EDPs do calor e da onda, foi utilizada a rotina SGBSV, otimizada para matrizes de banda de múltiplas diagonais, os quais são armazenados como um *array* para cada coluna j: $AB_{(KU+KL+i-j,j)} = A_{i,j}$ $(Max(0, j-KU) \leq i \leq Min(n, j+KL))$, em que KU é o número de diagonais superiores e KL o número de diagonais inferiores, e n a ordem da matriz banda. Tais meios de armazenamentos são para que as rotinas resolvam os sistemas lineares considerando somente os valores diferentes de zero das matrizes esparsas, ignorando os elementos nulos e aumentando sua eficiência.

[3] http://www.netlib.org/lapack
[4] http://www.netlib.org/blas/

Diferentemente das EDPs do calor e do transporte, cujas discretizações remetem à resolução de um sistema linear $Ax = b$, a EDP da onda discretizada remete à resolução de uma operação de multiplicação matriz-vetor. Para isso, foram utilizadas as rotinas SBMV (multiplicação matriz-vetor para matrizes bandas) e SAXPY (adição vetorial com multiplicação escalar), ambas da biblioteca BLAS/LAPACK. Neste caso, a operação para encontrar cada valor de $U_{i,j}$ é definida por $SAXPY(SBMV(A, U_{i,j}), U_{i,j-1}, -1)$, onde -1 indica que SAXPY realizará uma subtração de vetores.

Para a interpretação das funções das condições de contorno e do estado inicial, o aplicativo do servidor possui um analisador léxico/sintático para a geração de árvore das expressões. Estas funções podem ser constantes (árvore de um único nó), funções matemáticas $f(x,y,t)$ (condição de contorno de Dirichlet) ou funções condicionais intervalares. As funções parametrizadas pelos usuários são passadas como *string* para o aplicativo, o qual gera a árvore sintática de expressões considerando a precedência de operadores.

Para a geração da árvore sintática das expressões, inicialmente efetua-se a análise léxica, em que todos os operandos e operadores são reconhecidos e separados em um conjunto de palavras denominadas *tokens*. A partir dsses *tokens* é efetuada a análise sintática, em que efetua-se a análise da precedência de operadores. A Figura 5.4 ilustra alguns exemplos de árvores sintáticas e suas respectivas expressões de entrada.

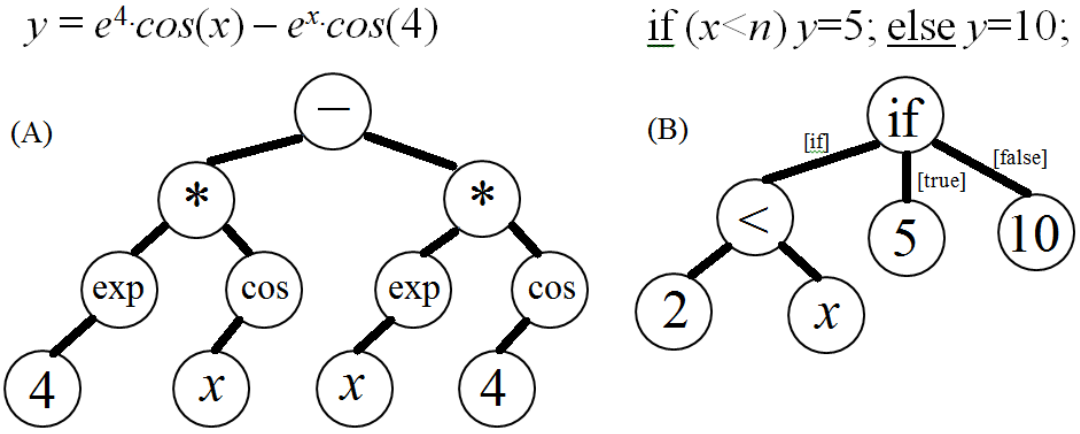


Figura 5.4 – Exemplos de algumas árvores sintáticas e suas respectivas expressões: função matemática (A) e condição intervalar (B).

A partir da árvore gerada (Figura 5.4), é feito um percurso pós-ordem para poder calcular os valores numéricos nas condições de contorno, tendo os valores de x, y ou t dependendo da situação. Na Figura 5.4A inicialmente calcula-se as expressões dos nós folhas, que servem de resultado para os nós pais, até chegar à operação final da raiz. Já na Figura 5.4B, a raiz 'if' indica uma expressão de condição, em que inicialmente verifica-se a expressão da sub-árvore esquerda de 'if', cujo retorno booleano determina a execução da expressão da sub-árvore do meio (*true*) ou direita (*false*). Este último tipo de árvore permite atribuir valores no contorno em regiões específicas.

5.3 MÓDULO DE BANCO DE DADOS

O principal meio de armazenamento de informações do usuário e dos cálculos e saídas processados pelo servidor é o banco de dados do PostgreSQL armazenado no servidor. Este banco é constituído de cinco tabelas: 'Usuário', 'Solicitação', 'Malha', 'Resultado', 'Resultado Final'.

O banco de dados, armazenado no servidor, é acessado pelo *web service*, que envia informações para o *front-end* e para o *back-end*. Este banco de dados, sendo a principal forma de comunicação cliente-servidor, armazena todas as informações e meta-informações relativas aos usuários, problemas solicitados e soluções, e é acessado tanto pelo *front-end* como pelo *back-end*. O modelo de entidade-relacionamento (MER) do banco de dados á apresentado na Figura 5.5.

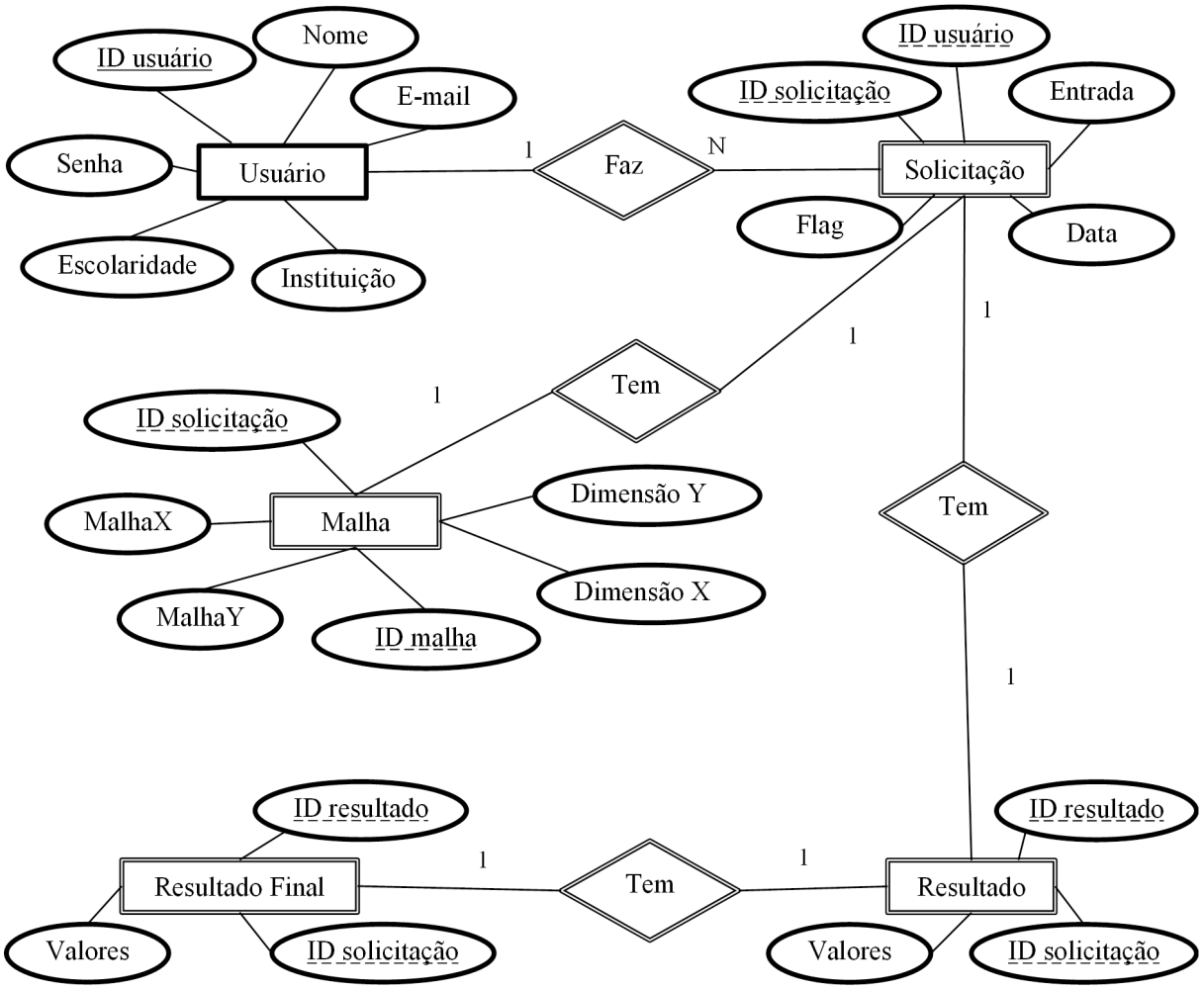


Figura 5.5 – MER do banco de dados de comunicação cliente-servidor e armazenamento de resultados e informações.

Conforme pode ser visto na Figura 5.5, cada tupla da tabela 'Usuário' é associada a várias solicitações (tabela 'Solicitação'), que contém seus resultados (tabela

'Resultado') e informações da malha (tabela 'Malha') se necessário. A tabela 'Usuário' possui as informações de todos os usuários cadastrados no sistema, como o ID, nome, e-mail, senha de acesso, instituição de ensino de origem e grau de escolaridade. Ela possui uma chave estrangeira para a tabela 'Solicitação', que contém informações de cada solicitação de cálculo.

As informações da tabela 'Solicitação' incluem os respectivos IDs, a data e a hora da solicitação, o ID do usuário solicitante e um atributo contendo as informações de todos os campos de entrada do problema separados por espaço. O tamanho desse campo de entrada, definida em forma de *string* varia conforme o problema a ser resolvido. Esta tabela também possui um campo de *flag* que determina se uma solicitação está em processo de execução, indicado por *flag*='A' (em aberto), ou se a solicitação já finalizou seu processamento, indicado por *flag*='F' (fechado).

Para casos de problemas envolvendo coordenadas generalizadas, a tabela 'Solicitação' possui uma relação com a tabela 'Malha'. Esta contém atributos como ID da malha, ID da respectiva solicitação, domínio do problema e as informações dos pontos da malha armazenadas como *varchar*, tanto para o eixo x (MalhaX) como para o eixo y (MalhaY). Todas as tuplas da tabela 'Malha' são associadas a uma solicitação do usuário, porém nem toda solicitação necessariamente possui um registro de 'Malha' associado, que é o caso de problemas em coordenadas cartesianas.

Já a tabela 'Resultado' armazena as informações dos resultados correspondentes à chave estrangeira da tabela 'Solicitação' para cada instância de tempo, contendo as informações dos IDs dos respectivos resultados. Este armazenamento é feito de modo temporário, para não comprometer o desempenho durante as consultas em tempo de execução, especialmente quando há muitos acessos simultâneos de usuários, e possui uma *trigger* que espelha seus dados para a tabela 'Resultado Final'. Este último é uma cópia da tabela 'Resultado', que é utilizada para futuras consultas de históricos de resultados de cada usuário.

Enquanto os cálculos são processados em cada instância de tempo t, seus respectivos resultados são inseridos na tabela 'Resultado' em um formato *varchar(1000000)*, permitindo o armazenamento de problemas de grande porte. Enquanto a *flag*='A' é mantida até a instância de tempo final T_{max}, um *trigger* replica os resultados já processados para a tablea 'Resultado Final'. Após o término dos cálculos, o campo da *flag* é setado como 'F', indicando o fim do processamento de uma determinada requisição ao mesmo tempo que ativa um *trigger* que remove todos as informações dos resultados da tabela 'Resultado', relacionados a um deteminado ID de solicitação.

Com o uso do esquema da *flag*, a tabela 'Resultado Final' faz com que 'Resultado' seja temporária, com o intuito de não degradar o desempenho durante as consultas e exibições em tempo de execução das simulações. A tabela 'Resultado Final' descrita acima serve de histórico de armazenamento de todos os resultados, podendo ser utilizada para futuras consultas de problemas já resolvidos. Para isso, essa tabela serve como um *cache*

das solicitações, em que um usuário realiza a solicitação de resolução de um problema e uma consulta ao banco de dados é realizada para verificar a existência do mesmo já resolvido, sendo este apresentado ao usuário.

O modelo de gerenciamento de banco de dados desenvolvido permite o compartilhamento de um conjunto de problemas e soluções já resolvidos e existentes, sem a necessidade de resolvê-los novamente, uma vez que cada problema possui sua ID e informações de entradas armazenadas em cada tupla da tabela 'Solicitação'. Esse meio de gerenciamento permite um armazenamento em *cache* de cálculos prévios, permitindo um uso econômico e otimizado dos recursos de servidor remoto. Além disso, as tabelas 'Solicitação' e 'Resultado' possuem atributos de data e hora em que o usuário fez a solicitação no *web service*, em que o usuário pode acessar problemas e soluções em um período de tempo específico.

5.4 Módulo Web Service – Comunicação Front-end e Back-end

Este módulo é considerado o principal, uma vez que ele é um componente responsável para fazer a troca de informações da entrada e dos cálculos de saídas, fornecendo uma interação em tempo de execução com o usuário. Para comunicação com o *web service*, foi utilizado o protocolo SOAP (*Simple Object Acces Protocol*), que é responsável pela troca de mensagens no formato de documentos XML (*eXtensible Markup Language*) entre o cliente e o servidor, a fim de proporcionar uma interação entre eles.

A interface da comunicação e da troca de mensagens foi descrita no formato padrão WSDL (*Web Services Description Language*), que possui diversos índices de métodos a serem requisitados por um *web service*. Os métodos vão desde a busca de informações no banco de dados, gerenciamento de informações dos usuários e das solicitações até o comando para a execução da aplicação requisitante (Figura 5.6).

O *web service* desenvolvido disponibiliza um meio para comunicação com o usuário e com o dispositivo E/S (entrada-saída) que consiste de um servidor remoto que envia e recebe dados dos processamentos. Uma vez este serviço disponível, a comunicação com o *web service* através de um GUI constrói o sistema *web* propriamente dito, junto com os demais módulos, estabelecendo uma transparência entre as aplicações, permitindo que qualquer interface possa se comunicar com o *web service*. Essa transparência é feita pelo *client wrapper* e *server wrapper*, do protocolo SOAP, para que as aplicações cliente e servidor que se comuniquem entre si através de documentos XML (Figura 5.6).

O servidor com a sua aplicação *back-end* (Figura 5.2) é a parte responsável pelo gerenciamento de acessos simultâneos de usuários e pelo armazenamento de todas as informações do banco de dados. A aplicação *front-end*, faz a leitura dessas informações quando é feita a solicitação e faz a escrita quando é feita a atualização. O próprio *back-end* também escreve essas informações no banco de dados durante a execução de uma simulação.

O *front-end* (5.2) é também responsável pelo envio das entradas fornecidas pelos usuários para o servidor, no formato de um arquivo-texto indexado com o número de

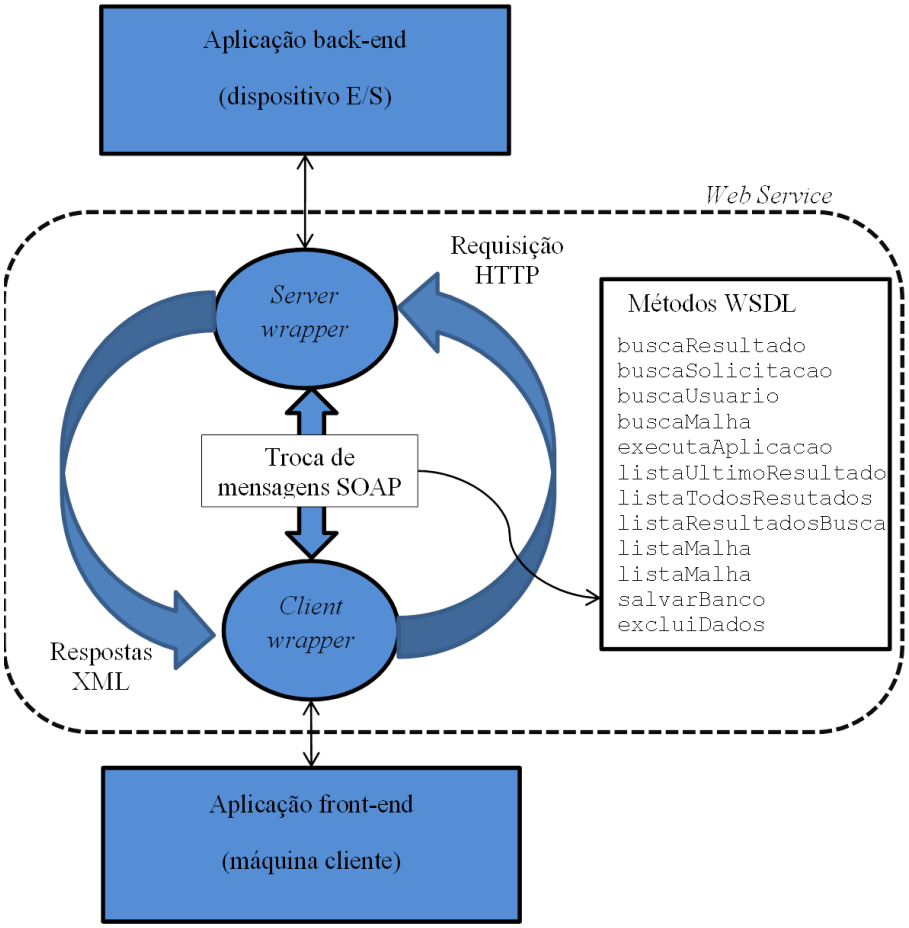


Figura 5.6 – Diagrama do *web service* e as mensagens WSDL trocadas durante o processamento.

identificação (ID) de cada usuário. Neste arquivo, são armazenados a ID do problema a ser resolvido e todos os parâmetros de entrada. A partir destas informações, o servidor realiza a leitura para fazer o processamento dos cálculos.

No *back-end*, para executar a chamada da aplicação desenvolvida em C++, foram utilizadas as classes Runtime e Process da plataforma Java SE. A classe Runtime permite retornar um objeto Process em tempo de execução associado ao aplicativo Java atual. Neste caso, o objeto Process é uma chamada do aplicativo 'xterm' [44], um emulador de terminais que permite a execução via linha de comandos de diversas aplicações em múltiplas plataformas.

Enquanto a aplicação em C++ está executando o problema, os resultados são atualizados no banco de dados. O processo de consulta e a exibição em tempo de execução dos resultados é realizado pelo componente *Poll*, um artifício presente no *framework* Prime Faces que efetua chamadas Ajax contínuas ao *web service*. O *Poll* neste caso realiza chamadas assíncronas a cada três segundos para verificar se existe atualização e inserção de novos resultados no banco de dados e efetuar a atualização na máquina cliente. Este tempo de *delay* também pode ser definido e modificado pelo próprio usuário.

Juntamente com as entradas das condições de contorno interpretadas pela

árvore sintática (Figura 5.4) e com os demais parâmetros de entrada do arquivo texto, o aplicativo no servidor calcula os resultados em cada instância de tempo. Enquanto isso, se o processo de *Poll* identificar que uma nova instância do problema foi calculada, os valores da malha de distribuição da temperatura são escritos na interface do *browser* para o usuário. Estes valores T_i^k são separado por vírgulas (1D) e de $T_{i,j}^k$ com a separação de cada coluna da malha por vírgula e de cada linha por ponto-e-vírgula (2D).

Os formatos armazenados no banco de dados são notações de vetores (problemas 1D) e matrizes (problemas 2D) que podem ser acessadas através de uma consulta no banco de dados. Os formatos armazenados são acessíveis de outras aplicações, a partir do qual, o usuário pode efetuar diversos tipos de análises numéricas, tais como erros e comparações com soluções analíticas do problema.

5.5 Interface Gráfica com Usuário e Visualização de Malhas

Após os processamentos dos cálculos, é necessária uma forma para o usuário acessar e visualizar os resultados da simulação. Isso é realizado em tempo de execução do problema que está sendo resolvido. Considerando-se isso, diferentes metodologias foram aplicadas para visualização das malhas em coordenadas cartesianas e generalizadas, além de um meio de interação com o usuário tais como consulta e acesso aos resultados e manipulação de geometrias.

5.5.1 Coordenadas Cartesianas

Para visualização dos resultados obtidos em forma de gráfico, no 1D foi utilizado um framework JAVA chamado JFreeChart. Este componente permite a criação de uma grande variedade de gráficos, mas para o projeto foi escolhido o gráfico de linhas que melhor representa a variação da temperatura. O processo que permite a visualização foi realizado da seguinte maneira:

1. Obter a String na tupla do banco de dados que contém a informação previamente calculado e exibido em um grid (DataTable do PrimeFaces)

2. A string obtida será inserida dentro de um dataset do próprio JFreeChart, chamado DefaultCategoryDataset

3. Uma vez os dados já inseridos no dataset, será executada a instrução para a plotagem do gráfico utilizando as funções da biblioteca JFreeChart.

Outra forma encontrada para visualizar os resultados obtidos no 1D foi a criação de uma cadeia de cores que vai do menor valor encontrada na tupla. Neste caso existe uma variação de cores de 15 tonalidades representando os valores positivos e outras 15 valores negativos. A forma encontrada para localizar os intervalos com suas respectivas

cores foi a divisão do maior valor positivo pela quantidade de cores positivas (15 cores) $I = MaiorValor/QtdeCores$.

Após a obtenção dos intervalos, foram feitas diversas comparações e logo depois inseridas em uma *string* juntamente com *tags* $HTML < table >, < tr > e < td >$ para a montagem de uma tabela que represente a variação das saídas das PDEs (temperatura, valor da onda e concentração). Da mesma maneira foi com os valores negativos para encontrar seus intervalos com suas respectivas cores, sendo que neste caso, o menor valor foi obtido ($I = MenorValor/QtdeCores$). A Figura 5.7 ilustra um exemplo da barra de cores gerada após a definição dos intervalos.

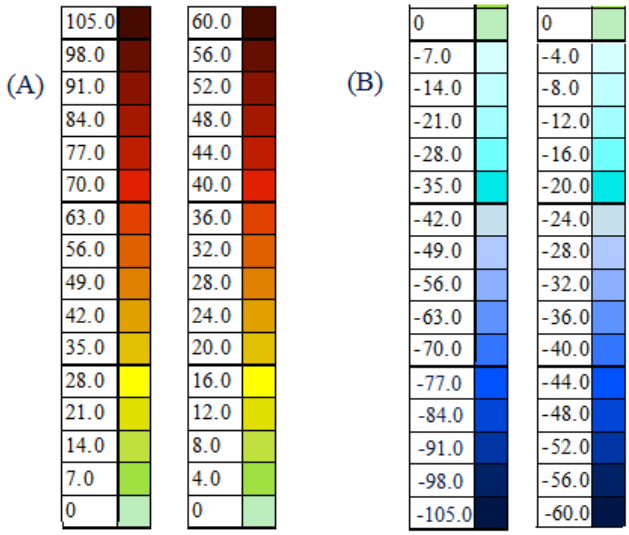


Figura 5.7 – Utiliação de uma mesma cadeia de cores para diferentes intervalos de valores, tanto para positivos como (A) para negativos (B).

Todos os valores demonstrados no código, juntamente com as *tags* HTML, foram inseridas em uma *string* para posteriormente serem interpretadas pela *tag* $< h : outputText >$ do framework JSF, como por exemplo no comando $< h : outputTextvalue =$ "$\#graficoBean.corTabela$"$escape =$ "$false$"$/>$. Para obter o gráfico 2D foi feito o mesmo processo utilizado no 1D. Encontrado os intervalos com suas respectivas cores foi dividido o maior valor positivo pela quantidade de cores positivas e negativas (Figura 5.7) para depois inserido juntamente com os comandos HTML de inserção de tabela $< table >, < tr > e < td >$.

No final do processamento, é mostrado na tela a distribuição da temperatura sobre a malha em todas as instâncias de tempo definidas, em que o usuário pode fazer o *download* das saídas no formato CSV, acessível através de aplicações que trabalhem com dados tabulados (*spreadsheets*). A partir deste arquivo, está disponível para o usuário um *script* em Scilab[5], que efetua a visualização gráfica de mudança e da densidade de distribuição da temperatura, em todos os estágios temporais.

[5] http://www.scilab.org

O modo de visualização da malha aplicado possui um meio em que o usuário pode acessar meta-informações em uma determinada região discretizada, tais como posição (x,y), condição de contorno influente e variação do valor (temperatura ou concentração) com relação ao valor do tempo anterior. Apesar do meio simples de visualização gráfica implementada no sistema *web*, a visualização dos resultados da malha discretizada podem ser realizados em qualquer dispositivo que tenha um navegador de Internet com tecnologia HTML ou Java, tais como computadores pessoais e dispositivos móveis. Isso se deve à simples configuração da malha gerada através de tags HTML.

5.5.2 Coordenadas Generalizadas

O gerador de malhas e o módulo de visualização dos resultados da simulação em coordenadas generalizadas consiste em uma interface desenvolvida em Javascript e Primefaces (Figura 5.8), junto com funções da biblioteca Canvas para o desenho das geometrias. A GUI desenvolvida recebe as entradas do usuário e mostra as saídas da geometria correspondente.

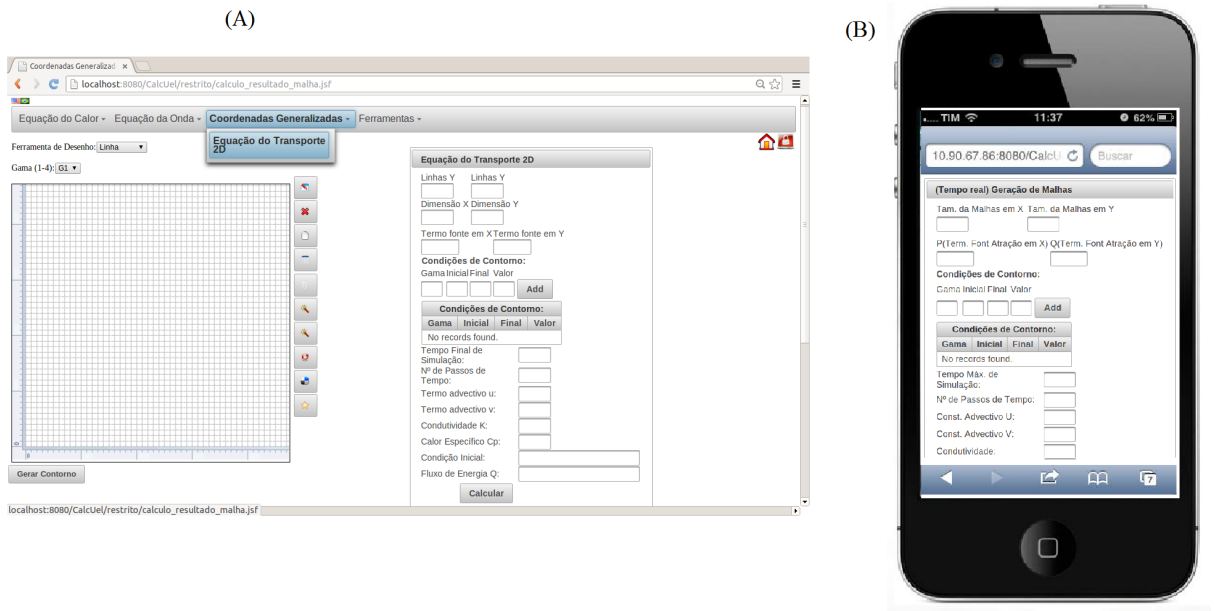


Figura 5.8 – Interface do gerador de malhas desenvolvido em um navegador *web* (A) e um dispositivo móvel (B).

Na sequência, são descritos os procedimentos para o usuário utilizar o gerador de malhas:

1. Dimensão do problema – o usuário especifica qual o tamanho máximo de x e y;
2. Pontos da geometria – podem ser definidos de acordo com as opções disponíveis:
   - O usuário define os segmentos de retas $[(x_0,y_0),(x_1,y_1)]$ junto com o Γ_n pertencente. O sistema realiza a ligação automática dos segmentos, em que o primeiro ponto de

um Γ_n é igual ao último de um Γ_{n-1} (para $1 \leq n \leq 3$). Se $n = 4$, o último ponto de Γ_4 é o primeiro de Γ_1;

- O usuário insere os pontos (x,y) individualmente e quando o último ponto for inserido, o sistema realiza a ligação automática de todos de acordo com o contorno Γ_n atribuído;
- O usuário faz o *upload* de um arquivo contendo os pontos da geometria. Este arquivo já separa os pontos para cada Γ_n. Neste caso, o sistema já fornece a ligação de todos os pontos simultaneamente. Independente do meio de inserção dos pontos, o formato especificado pelo Γ_n e seus respectivos conjuntos de pontos, é definido da seguinte forma:

$$\begin{bmatrix} 1 : x_{11}\ y_{11}; x_{12}\ y_{12} \\ ,1 : x_{12}\ y_{12}; x_{13}\ y_{13} \\ ,1 : x_{13}\ y_{13}; x_{21}\ y_{21} \\ 2 : x_{21}\ y_{21}; x_{22}\ y_{22} \\ 2 : x_{22}\ y_{22}; x_{23}\ y_{23} \\ 2 : x_{23}\ y_{23}; x_{31}\ y_{31} \\ 3 : x_{31}\ y_{31}; x_{32}\ y_{32} \\ 3 : x_{32}\ y_{32}; x_{33}\ y_{33} \\ 3 : x_{33}\ y_{33}; x_{41}\ y_{41} \\ 4 : x_{41}\ y_{41}; x_{42}\ y_{42} \\ 4 : x_{42}\ y_{42}; x_{43}\ y_{43} \\ 4 : x_{43}\ y_{43}; x_{11}\ y_{11} \end{bmatrix}$$

;

3. Tamanho da malha – após o término da definição dos pontos e do contorno da geometria, o usuário especifica o tamanho da malha (o qual definirá o número de linhas em ξ e em η) e então é enviado uma solicitação para o servidor remoto que fará a métrica para geração da malha computacional. Entretanto, os pontos para a aplicação em Java são armazenados em *pixel* e estes devem ser convertidos em coordenadas (x,y) para que o módulo de processamento de cálculo possa reconhecê-los;

4. Armazenamento de informações – quando todos os pontos da malha são calculados, o módulo de cálculos armazena estas informações na tabela 'Malha' do banco de dados;

5. Parâmetros de entrada da EDP – Quando os pontos da malha foram gerados, o módulo de cálculos realiza a simulação de acordo com a entrada do usuário. As entradas consistem dos seguintes parâmetros:

   - Tempo máximo de simulação T_{max};
   - Número de passos de tempo N_t;

- Condutividade do material K;
- Calor específico C_p;
- Parâmetros advectivos u e v;
- Fluxo de energia q (para definição da condição de contorno de Neumann);
- Condição inicial $T(\xi, \eta, 0)$;
- Condições de contorno de Dirichlet. Definição de condições de contorno do tipo $(\Gamma, \phi_i, \phi_f, T)_1, (\Gamma, \phi_i, \phi_f, T)_2, (\Gamma, \phi_i, \phi_f, T)_3 \ldots (\Gamma, \phi_i, \phi_f, T)_n$, em que Γ é o lado do contorno, ϕ_i e ϕ_f é o intervalo em $\partial\Omega_0 \times [0, T_{max}]$ e T o valor a ser atribuído no intervalo;

6. Após a definição dos parâmetros da equação da energia 2D, o módulo de cálculos realiza a simulação para cada instância de tempo. O processo de *Poll* faz a consulta no banco de dados para verificar a existência de novos resultados;

7. Terminado o processo, através de funções de desenho de linhas provenientes da biblioteca Canvas para realizar o desenho da malha discretizada. Este procedimento consiste nos seguintes passos:

   a) Transforma a string dos pontos da malha obtidos da tabela 'Malha' em um conjunto de vetores;

   b) Os pontos da malha calculados são lidos do banco de dados e enviados para o módulo de interface, que realiza a reconversão dos pontos em coordenadas (x, y) para pixels, de modo que a aplicação Javascript possa reconhecê-lo;

   c) Desenha as linhas ξ utilizando o recurso Path (especificação de uma linha geométrica a partir de uma posição inicial e final na tela de desenho) da biblioteca Canvas;

   d) Desenha as linhas η utilizando o recurso Path sobre as linhas ξ já plotadas, gerando o *grid* completo (malha);

   e) Realiza a distribuição dos valores sobre cada nó do grid (intersecção das linhas ξ e η), através dos recursos fillRect (ponto) e fillStyle (definição da cor dos pontos de acordo com a escala de cores). Este procedimento é repetido até o fim do problema.

O gerador de malhas da Figura 5.8, acessível tanto por um navegador *web* como por um dispositivo móvel, possui alguns esquemas de interatividade com o usuário. Dentre estes esquemas pode-se citar a adição e remoção de linhas de grade, que serve para uma visualização mais precisa, e um esquema de adição/remoção de malhas para eventuais *uploads* de arquivos de geometrias pré-definidas. Além disso, pode-se visualizar os valores numéricos para cada ponto da malha, que também estão preenchidos com uma cor estabelecida pelo esquema da cadeia de cores.

6 RESULTADOS E DISCUSSÕES

Para avaliar a viabilidade do sistema, alguns exemplos de problemas em coordenadas cartesianas e generalizadas foram resolvidos. Para isso foram simulados diversos problemas com seus respectivos parâmetros de entrada especificados na interface do sistema *web*. Os problemas testados utilizando o sistema de cálculo de EDPs por DDF em coordenadas cartesianas foram os seguintes:

- Problema 1 (Figura 6.1) - equação da condução do calor 1D
  - Constante $K = 0.8$;
  - Condições de contorno e inicial: $T(x_0,t) = 100$, $T(x_f,y,t) = 0$, $T_0(x,y,0) = 0$;
  - Parâmetros espaciais: $x_0 = 0$, $x_f = 5$, $N_x = 101$ (implica na resolução sistemas tridiagonais);
  - Parâmetros temporais: $T_{max} = 100$ e $N_t = 100$.
- Problema 2 (Figura 6.2) - equação da condução do calor 2D
  - Constante $K = 0.1$
  - Condições de contorno e inicial: $T(x_0,y,t) = 100$, $T(x_f,y,t) = 50$, $T(x,y_0,t) = 100$, $T(x,y_f,t) = 50$ e $T_0(x,y,0) = 100$;
  - Parâmetros espaciais: $x_0 = y_0 = 0$, $x_f = y_f = 4$ e $N_x = 111$ e $N_y = 111$ (implica na resolução sistemas de ordem 12100x12100);
  - Parâmetros temporais: $T_{max} = 80$ e $N_t = 5$.
- Problema 3 (Figura 6.3) - equação da condução do calor 2D
  - Constante $K = 0.1$;
  - Condições de contorno e inicial: $T(x_0,y,t) = e^y - cos(y)$, $T(x_f,y,t) = e^y * cos(4) - e^4 * cos(y)$, $T(x,y_0,t) = cos(x) - e^x$ e $T(x,y_f,t) = e^4 * cos(x) - e^x * cos(4)$, e $T_0(x,y,0) = 100$;
  - Parâmetros espaciais: $x_0 = y_0 = 0$, $x_f = y_f = 4$ e $N_x = 101$ e $N_y = 101$ (implica na resolução sistemas de ordem 10000x10000);
  - Parâmetros temporais: $T_{max} = 60$ e $N_t = 5$.
- Problema 4 (Figura 6.4) - equação da propagação da onda 1D
  - Constante $C = 1$;

  - Condições de contorno e inicial: $T(x_0,t) = 0$, $T(x_f,t) = 0$ e $T_0(x,y,0) = sen(\pi x)$;
  - Condição de velocidade inicial: $g(x) = 0$;
  - Parâmetros espaciais: $x_0 = 0$, $x_f = 2$ e $N_x = 100$ (implica na resolução sistemas de ordem 98x98);
  - Parâmetros temporais: $T_{max} = 2$ e $N_t = 100$.

- Problema 5 (Figura 6.5) - equação do transporte 2D
  - Constantes de difusividade $K_x = K_y = 0.1$;
  - Constantes advectivos $u = 1$ e $v = 0.5$;
  - Condições de contorno e inicial: $T(x_0,y,t) = 0$ e $T(x,y_0,t) = 0$ e $T_0(x,y,0) = 1$;
  - Parâmetros espaciais: $x_0 = y_0 = 0$, $x_f = y_f = 4$ e $N_x = 81$ e $N_y = 81$ (implica na resolução sistemas de ordem 6400x6400);
  - Parâmetros temporais: $T_{max} = 2$ e $N_t = 5$.

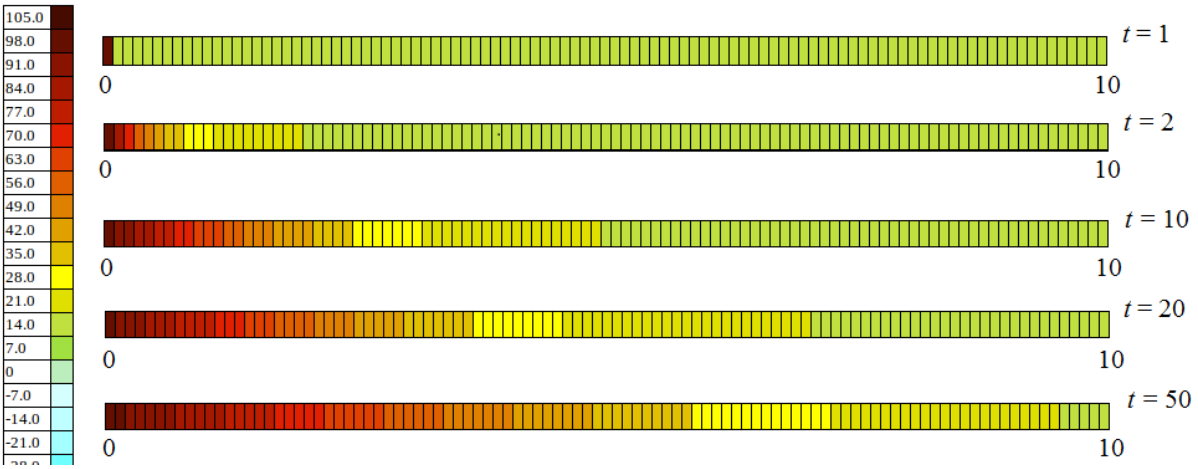


Figura 6.1 – Saídas para equação do calor 1D na forma de malhas visuais.

A Figura 6.1 descreve situações similares a uma barra de um determinado material, que submetido à uma temperatura constante inicial, uma das extremidades é submetida a uma fonte de calor. A EDP de condução do calor simula a condução dessa fonte de calor ao longo do tempo. Pode-se perceber que a condução tende a ser constante à medida que se aproxima do tempo de simulação final, característica das EDPs parabólicas.

Na Figura 6.2, com estado inicial $T_0(x,y,0) = 100$, a temperatura da superfície da barra e da placa diminui gradativamente dos contornos para o centro, devido à diminuição da temperatura ser fortemente influenciada pelas regiões próximas aos contornos, cujos valores são conhecidos e fixos. Dessa forma, a distribuição da temperatura tende a convergir a um estado final.

O problema 3 (Figura 6.3) verificou a viabilidade do usuário parametrizar condições de contorno definidas por funções matemáticas, que são interpretadas pela árvore

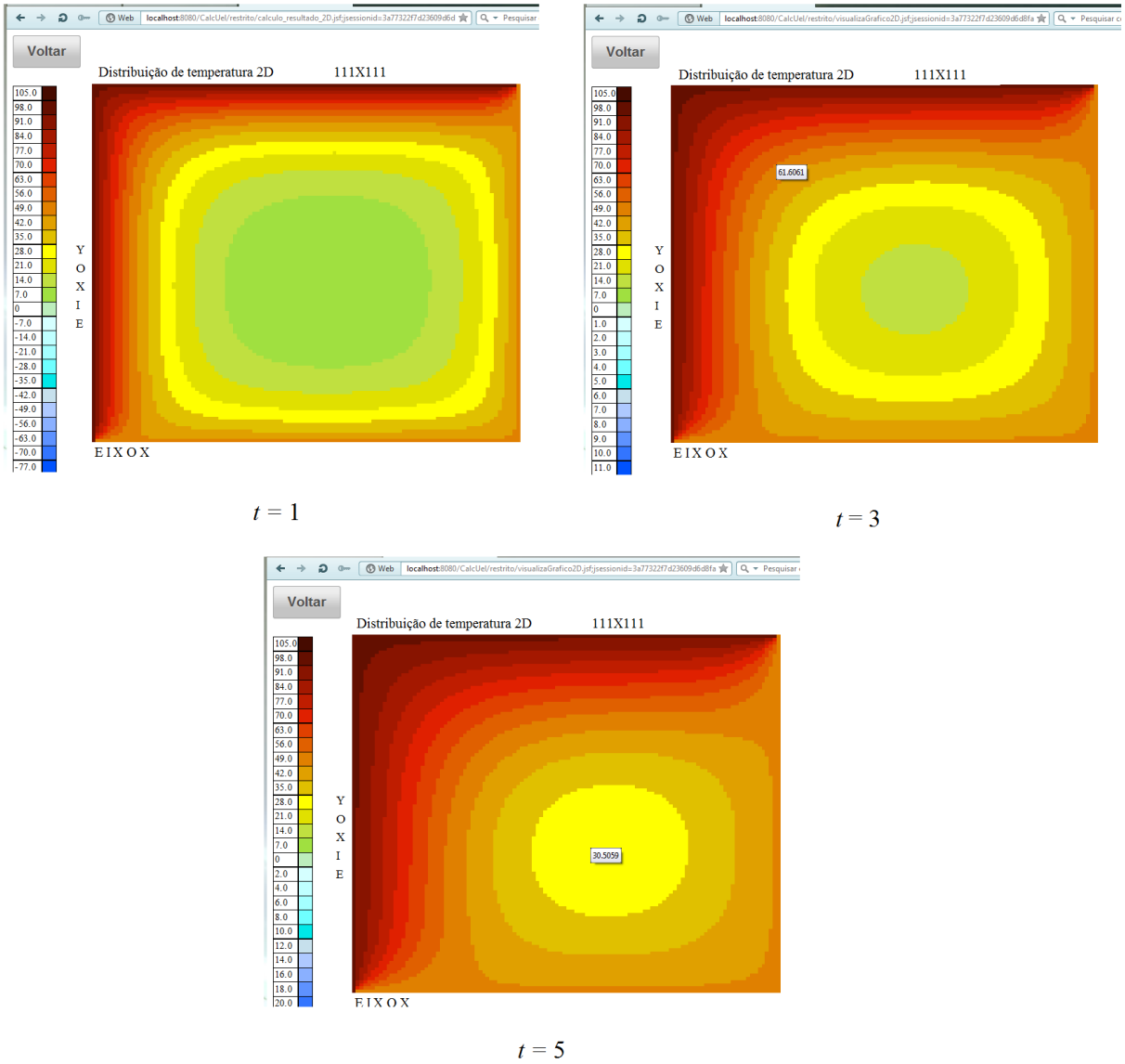


Figura 6.2 – Saídas em algumas instâncias de tempo do problema 2 da equação do calor 2D.

sintática gerada. O usuário também pode estabelecer qualquer condição de contorno de Dirichlet, sendo esta interpretado pelo analisador sintático. Esta mesma regra vale também para a condição inicial, especificada por uma função matemática dependente de x e y.

Pode-se ver na Figura 6.4 que a amplitude senoidal da onda muda periodicamente conforme o tempo, simulando a propagação de onda, sem considerar um meio ou interferência externa. O número de mudanças na exibição das respostas depende do valor de N_t, além de que o tamanho da malha N_x e N_y define o tamanho do problema. Para cada uma dessas instâncias, o servidor deverá retornar uma resposta para o cliente, fazendo com que o usuário tenha conhecimento da distribuição da temperatura na barra 1D ou na placa 2D durante o intervalo de tempo especificado.

Pode-se ver na Figura 6.5 que as constantes definidas na equação de transporte simulam ao mesmo tempo o fenômeno da difusão (similar à condução do calor) e da advecção,

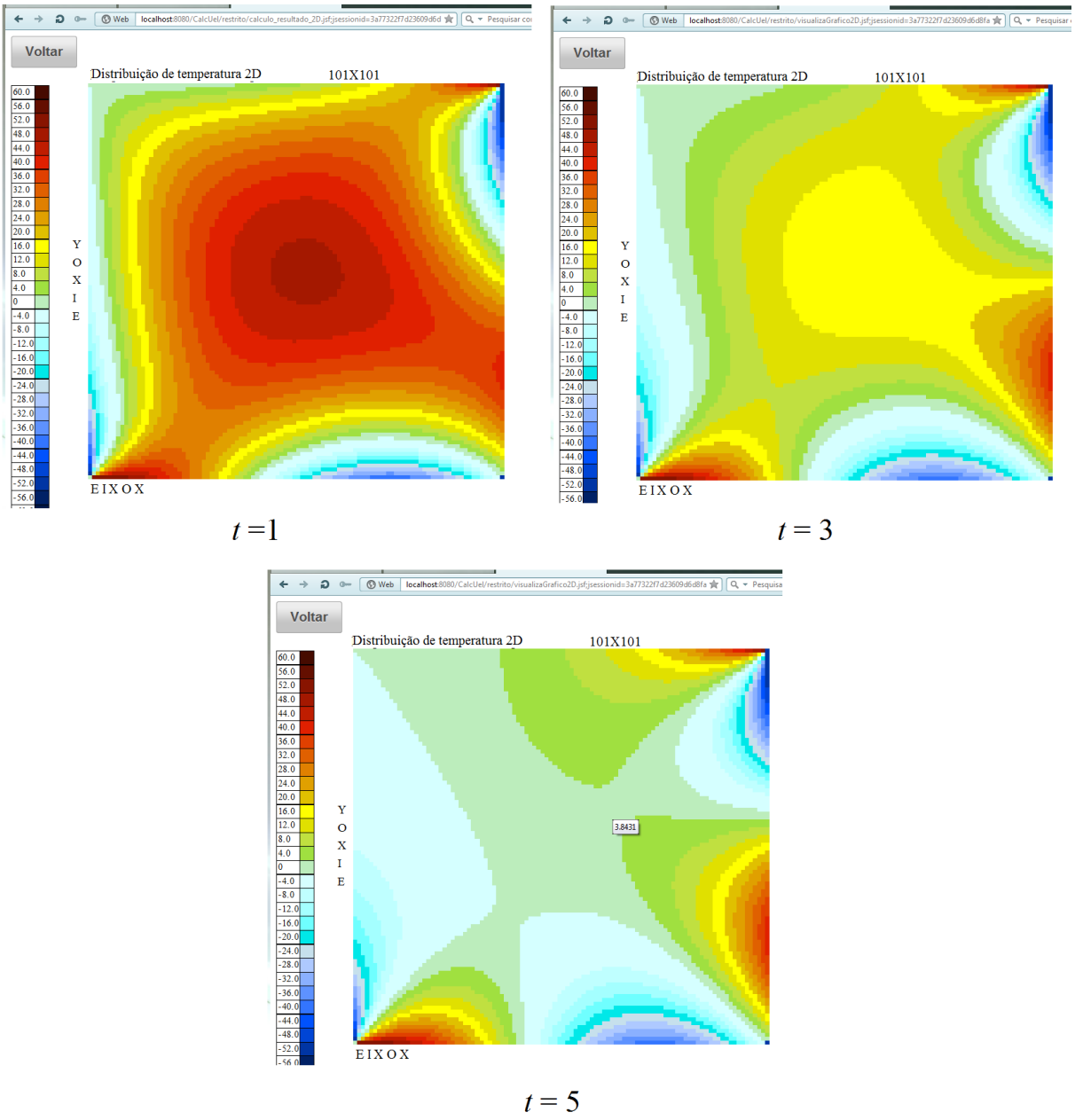


Figura 6.3 – Saídas em algumas instâncias de tempo do problema 3 da equação do calor 2D.

imposta pelas constantes u e v, em que $u > v$, fazendo com que o escoamento ocorra mais rapidamente na direção x. Os diferentes valores de N_x e N_y para o caso 2D da equação do calor e de transporte definem diferentes níveis de detalhamento, tanto na modelagem como na visualização da malha discretizada. Consequentemente, uma malha refinada define maior precisão e complexidade na simulação, deixando a visualização menos truncada.

Além da saída gráfica, é possível extrair as saídas no web browser, obtidas no formato numérico, para futuras simulações e análises em *softwares* científicos e de análise numérica. A Figura 6.6 mostra alguns exemplos de instâncias de tempo t para problemas 1D e 2D implícitos, após o processamento dos arquivos CSV no *script* em Scilab. Para o caso 1D, foram considerados $K = 0.875$, $T(x_0,t) = 0$, $T(x_f,t) = 10$ e a condição inicial $T_0(x,0) = 100$,

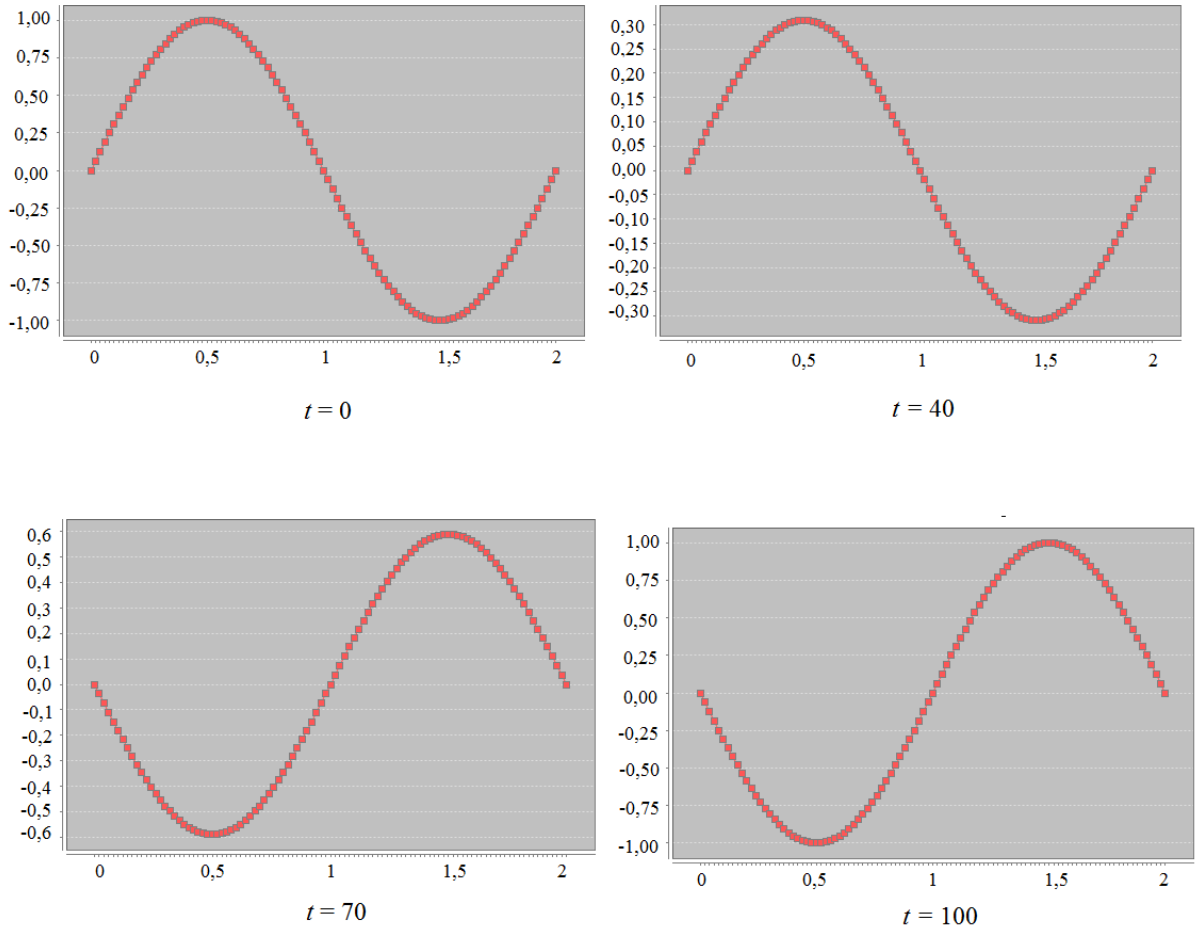


Figura 6.4 – Saídas em algumas instâncias de tempo da equação da onda 1D. As escalas são definidas de acordo com o tamanho da área de plotagem, em que o gráfico é ajustado.

tamanho da barra $x_0 = 0$ $x_f = 50$, com tamanho da malha $N_x = 2000$, implicando na resolução de sistemas tridiagonais de ordem 1998x1998.

6.1 Teste nas Coordenadas Generalizadas

Para o teste das coordenadas generalizadas, foram inseridas e geradas as malhas de algumas geometrias. Inicialmente, o usuário necessita especificar as coordenadas do plano físico que descreve a geometria. Neste caso, ele pode fazer medições manuais para obtenção dos pontos ou submeter um arquivo de dados que já contém todos os pontos pré-definidos. As Figuras 6.7 e 6.8 ilustram exemplos de malhas geradas pelo sistema visualizada na tela desenvolvida em Javascript.

A geometria da garrafa, cujo plano físico está ilustrado na Figura 6.8D, foi modelada considerando vários tamanhos de malhas – 20x20 (6.8A), 20x40 (6.8B) e 40x70 (6.8C). Os pontos do contorno da geometria foram obtidos através de medições milimétricas da silhueta do plano físico. Dessa forma, diversos objetos também podem ser modelados obtendo-se alguns pontos, uma vez que o método de interpolação por *splines* realiza a definição do

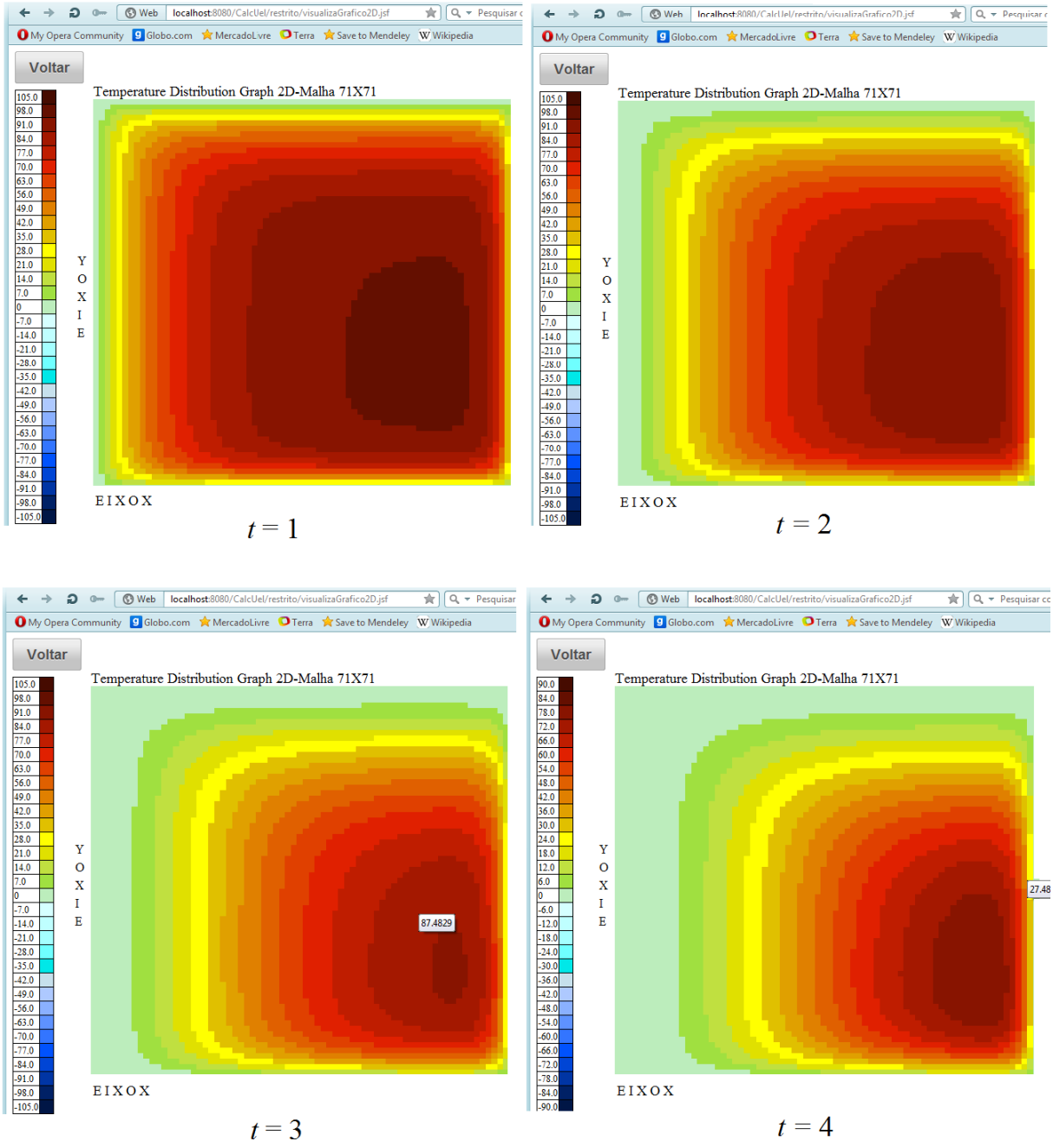


Figura 6.5 – Saídas em algumas instâncias de tempo do teste da equação do transporte 2D.

contorno total.

Pode-se notar nas Figuras 6.7 e 6.8 que foram geradas malhas estruturadas (a disposição de pontos de intersecção das linhas formam uma malha quadriculada), independente do tipo de geometria. Contanto que a geometria preferencialmente seja convexa, o gerador de malhas discretiza a geometria com menos lacunas possíveis, conforme pode ser vista na Figura 6.7, em que foram realizados atrações das linhas em uma direção considerando Q=1 (Figura 6.7B) e sem nenhuma atração (Figura 6.7A). Este modo de discretização estruturada facilita na aplicação de qualquer EDP similar ao que foi aplicado nas coordenadas cartesianas.

Após a geração da malha computacional sobre a geometria 2D, pode-se

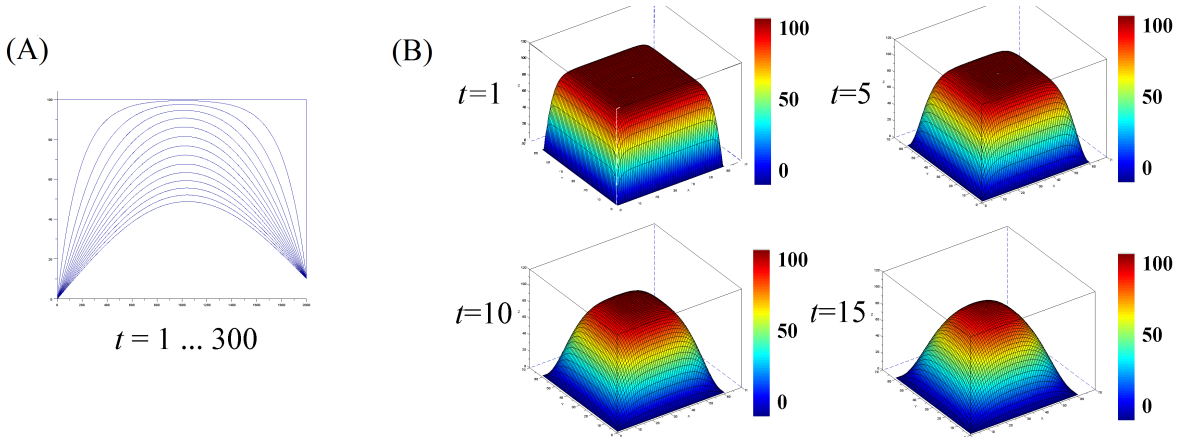


Figura 6.6 – Saídas obtidas no Scilab na equação do calor 1D (A) e 2D (B)

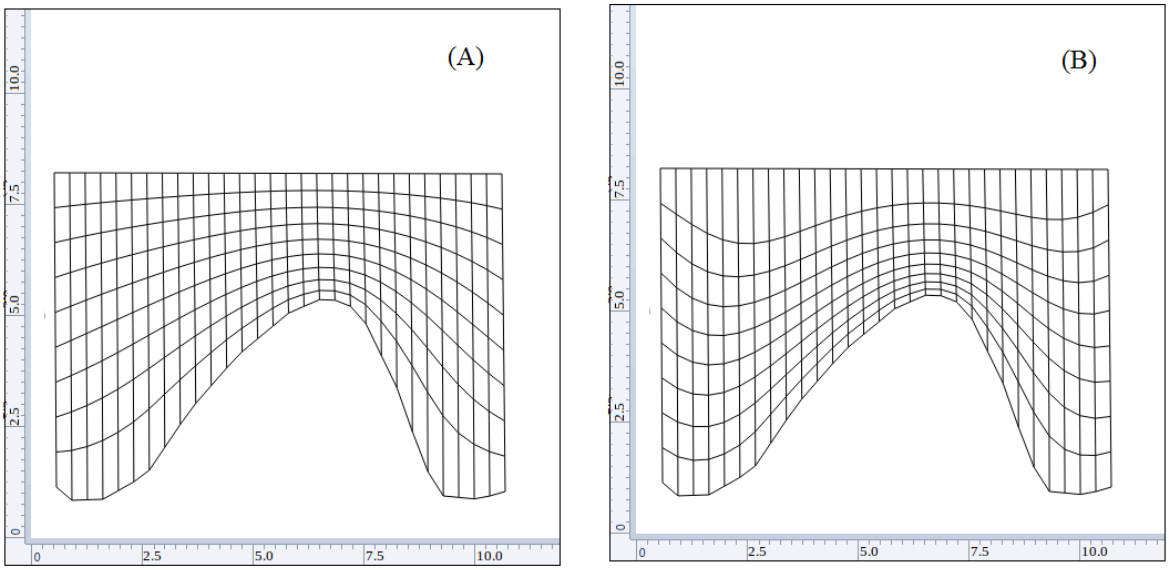


Figura 6.7 – Exemplo de malha gerada 1, em coordenadas generalizadas utilizando as métricas de transformações.

aplicar a equação da energia 2D. A Figura 6.9 ilustra um exemplo de uma simulação de uma distribuição de temperatura (energia) sobre uma garrafa. Ela representa uma situação em que uma garrafa em uma temperatura ambiente (com a condição inicial atribuída de $T(\xi,\eta,0) = 20$ graus) é apanhada com as mãos em ambas as extremidades (Figura 6.9A), sendo submetida a uma condição de contorno de Dirichlet de 40 graus atribuídas em regiões específicas de Γ_1 ($[2.5,7]$) e Γ_3 ($[2.5,7]$), além dos parâmetros $T_{max} = 10000$, $N_t = 100$, $\sigma = 1$, $u = v = 0$ (fluído parado) e $q = 0$. Como foi aplicada a metodologia de especificar cores nos pontos, as cores de tons mais quentes (vermelho), representam temperaturas maiores, e cores de tons mais frios representam temperaturas menores.

A simulação da Figura 6.9 ilustra mostra a condução do calor sobre a superfície da garrafa ao longo do tempo. Portanto, o sistema mostra-se viável para resolver problemas e situações reais em um ambiente *web*, principalmente no caso em que situações

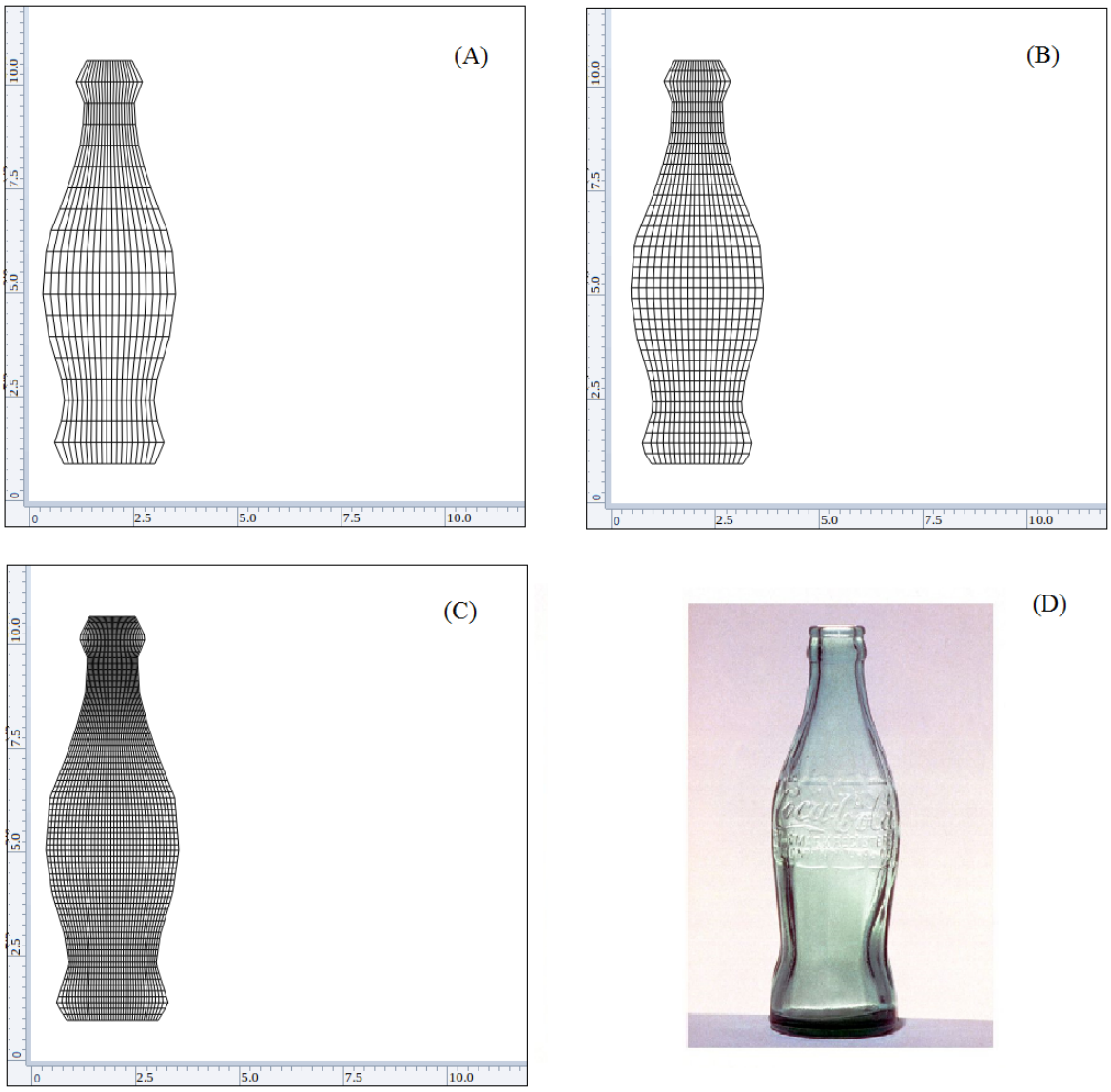


Figura 6.8 – Exemplo de malha gerada 3 (figura 'garrafa'), em coordenadas generalizadas utilizando as métricas de transformações.

reais não são geralmente descritas com uma geometria retangular, e sim com uma geometria generalizada. Para este caso, situações como o movimento de fluídos, fontes de calor externos, troca de energia entre o meio e o condutor e parâmetros advectivos foram desconsiderados.

Diante das simulações realizadas, o sistema proposto é útil na modelagem e simulação de diversas situações do mundo real. Os problemas podem ser relacionados, por exemplo, à perturbações globais e questões de ecossistemas. Um exemplo de problema foi modelado (Figura 6.10), em que a geometria 2D do Lago Igapó I foi descrita com o gerador de malhas para *web*, para em seguida realizar a simulação de possíveis fenômenos (Figura 6.11), como o transporte de poluentes sobre a superfície do lago [1, 3]. Os parâmetros foram os seguintes: $T_{max} = 27480$ segundos, $N_t = 100$, $\sigma = 0.055$ e tamanhos da malha $M = 100$ e $N = 50$, condição de contorno $(1, 2.8, 3.2, 1000)_1$, condição inicial $T(\xi, \eta, 0) = 0$ e $q = 0$.

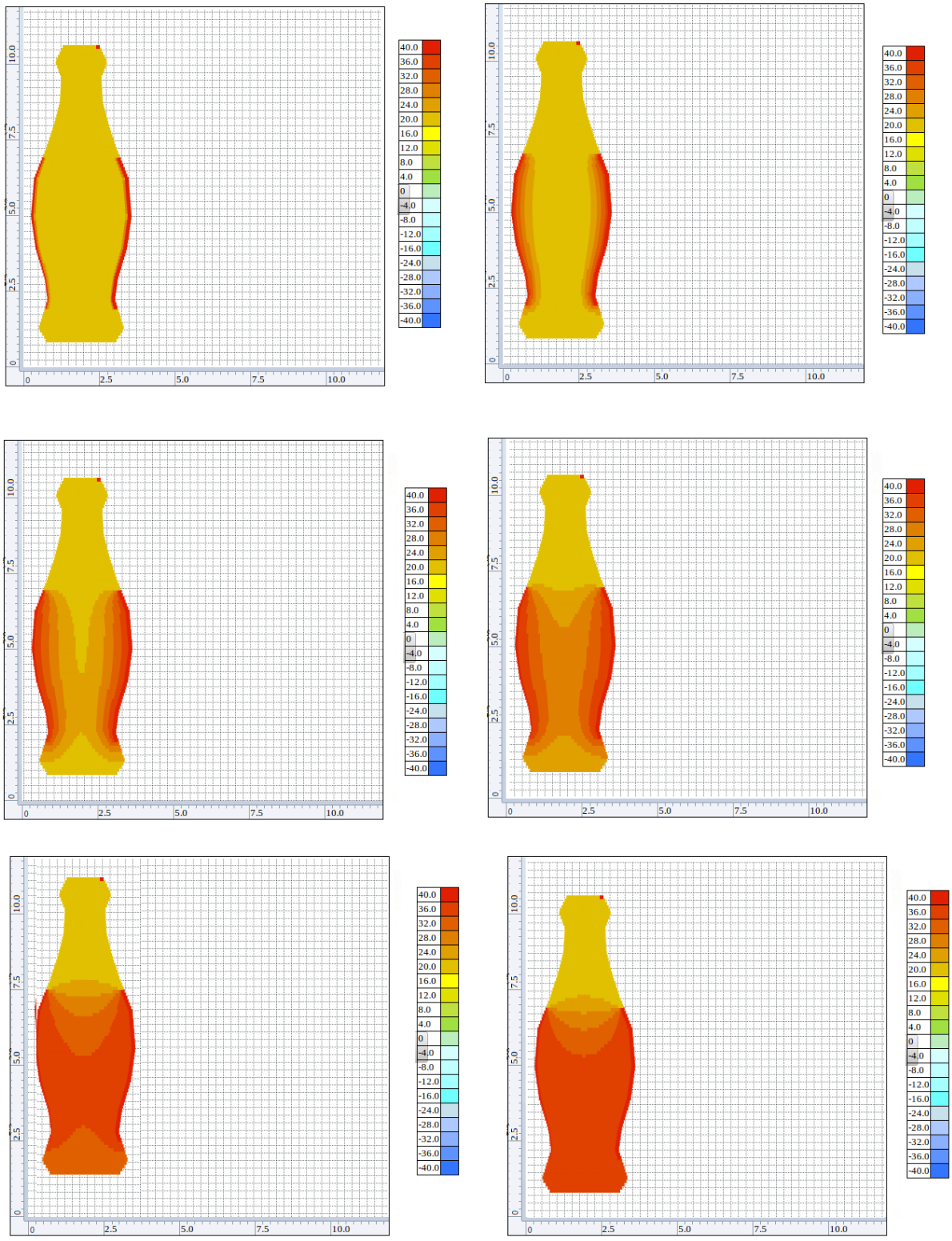


Figura 6.9 – Saídas da execução de uma simulação da equação de energia 2D em uma geometria generalizada em várias intâncias de tempo.

Na Figura 6.10, pode-se notar que foi possível tanto gerar a malha de superfícies 2D complexas, para efetuar a simulação de um fenômeno natural (Figura 6.11). Entretanto, a simulação do escoamento realizado por Romeiro et al. [1, 3] envolve o uso de

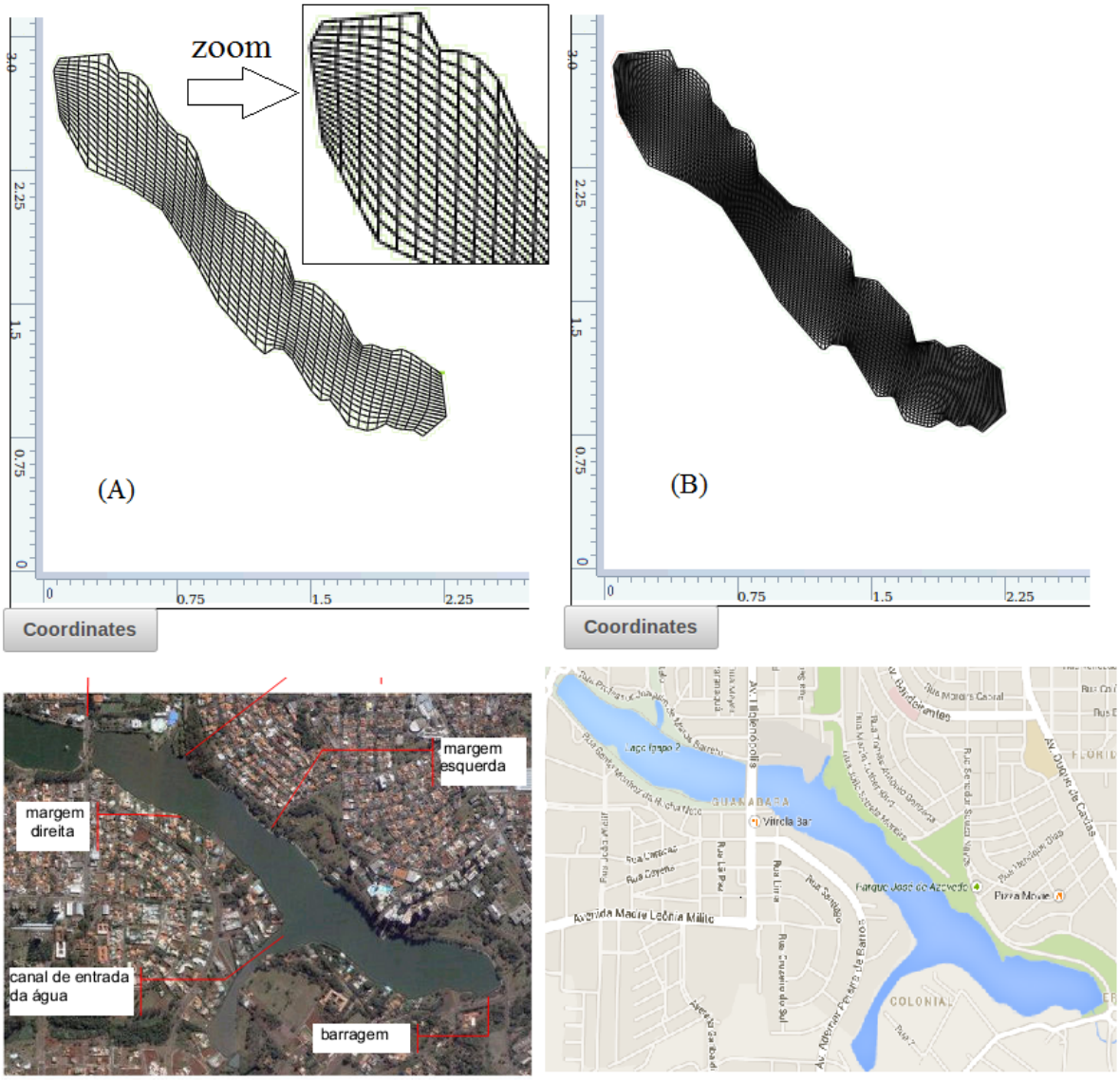


Figura 6.10 – Malha 2D da geometria do Lago Igapó gerada pelo sistema *web* em diferentes níveis de detalhamento com malhas 40x20 (A) e 100x50 (B), em comparação com o domínio físico do problema, obtido do Google Maps (C). A dimensão do problema está medido em Km.

campos de velocidades variáveis sobre o domínio inteiro, estes calculados através de equações de Navier-Stokes, além da simulação via equação de transporte. Até o presente momento, o nosso sistema *web* está limitado à equação de transporte 2D com um campo de velocidade constante, assumindo que $u = v = 1$, o que gera um efeito de escoamento mais uniforme sem efeitos de vórtice, comparado com o trabalho original (Figura 6.11D).

O problema de transporte de poluentes sobre o Lago Igapó demonstra um problema de grandeza temporal e espacial maior do que o da transferência de calor sobre a garrafa. Independente da dimensão espacial, o usuário pode fornecer os pontos da geometria de diversas formas, seja efetuando medições milimétricas ou até mesmo obter dados e imagens de

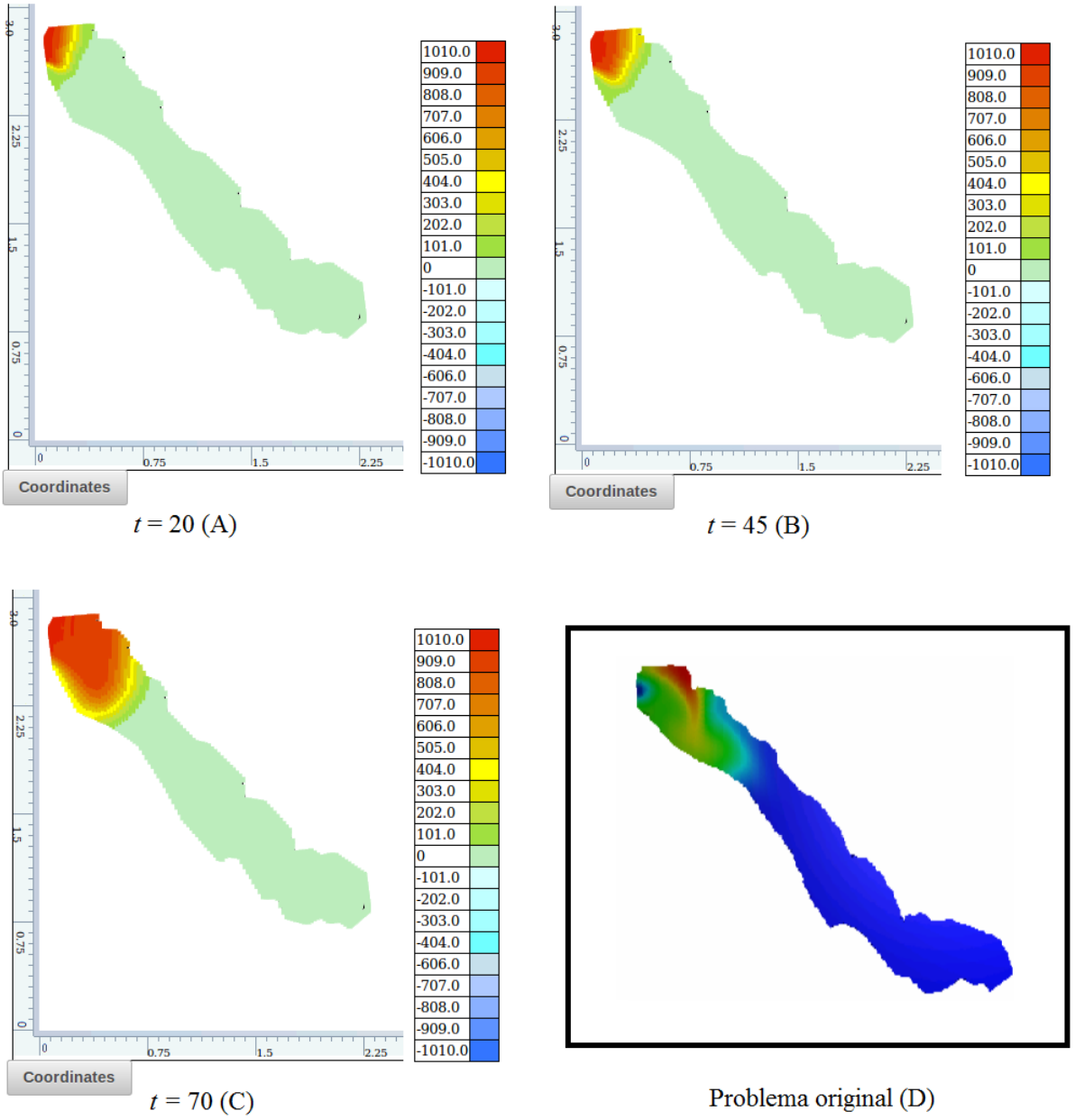


Figura 6.11 – Visualização da simulação do transporte de poluentes em diferentes instâncias de tempo (A-C) em comparação com o problema original (D) – obtido de [1].

satélites para obter uma informação mais realistas, caso seja um problema geograficamente de grande porte. Os meios de localização geoespacial de pontos já foram abordados em pesquisas [45], e podem ser uma ferramenta útil de geração de contornos da geometria, em conjunto com procedimentos e algoritmos de processamentos de imagens (detecção de bordas e contornos) para ser integrado no sistema.

Com a arquitetura implantada, pode-se realizar a inclusão de diversos tipos de EDPs, como a equação de Navier-Stokes, para permitir o cálculo de campos de velocidades variáveis. Independente do tipo de equação a ser aplicada, a arquitetura do sistema *web* proposto está aberto para a modificação de cada módulo implantado, para adaptar o sistema para resolver

outros tipos de problemas. É possível também ampliar o *web service* com outros métodos, o que pode servir para o usuário selecionar o modelo desejado. Dessa forma, o usuário realiza o estudo do fenômeno, além verificar quais tipos de esquemas de discretização se adequam melhor, permitindo uma análise mais aprofundada na etapa do pós-processamento [46].

Um exemplo de adaptação dos módulos e reusabilidade das rotinas já ocorreu durante o próprio desenvolvimento do sistema *web* proposto, em que a mesma arquitetura foi utilizado para resolver EDPs em dois tipos de situações: primeiro em coordenadas cartesianas para depois resolver problemas envolvendo coordenadas generalizadas. A modificação de alguns módulos necessários permitiu ampliar a primeira versão do sistema [47] para resolver um conjunto diferente de EDPs, em que foi necessário criar a tabela das informações dos pontos da malha no módulo do banco de dados e alterar o módulo de interface gráfica para permitir a visualização das geometrias. Para isso foi preciso também integrar a aplicação em Javascript na camada de interface gráfica com o usuário (página HTML), além de acrescentar os métodos e algoritmos de geração de malhas.

Através da modificação e adaptação dos módulos na inclusão das equações necessárias, uma utilidade prática resultante das simulações é a possibilidade de propor medidas para a melhoria da qualidade da água de um lago, embora seja possível a simulação de outros problemas reais. Isso é realizado através da análise remota dos poluentes de água. Dessa forma, é notável a aplicabilidade do *web service* proposto no auxílio de tomadas de decisões em situações de perturbações globais, visto que os serviços acessíveis remotamente já foram utilizados em diversas questões ambientais, como na análise da qualidade de água [14] e na análise de respostas sísmicas [48].

A questão da escolha do método numérico apropriado no pré-processamento na solução de EDPs vem sendo tratada há bastante tempo. Esta escolha depende do modo como se pretende resolver uma equação e a formato como se deseja obter os resultados. Para este trabalho, o método DDF foi aplicado para resolver EDPs em diferentes sistemas de coordenadas, uma vez que eles têm se mostrado ideais. Para o caso das coordenadas generalizadas, foi utilizado o mesmo método tanto para a geração de malhas como a aplicação da EDP sobre a malha, embora com propósitos diferentes.

Uma propriedade fundamental do sistema proposto é que ele pode ser executada em qualquer dispositivo, mesmo os móveis (Figuras 6.12 e 6.13), onde o usuário necessita somente de um navegador *web* com Java RE instalado e acesso à *Internet*. Dessa forma, ele não necessita ter conhecimento das fórmulas e recursos que estão executando os cálculos e do local de armazenamento de todas as informações. O desempenho das execuções e do uso de recursos computacionais depende exclusivamente dos requisitos do servidor, uma vez que este possui funções otimizadas para realização de sistemas lineares, que são processos de alto custo computacional especialmente para problemas de grande porte.

A máquina cliente, responsável por acessar a interface geral do sistema *web*, realiza somente as tarefas de conexão ao *web service* e consultas frequentes na tabela

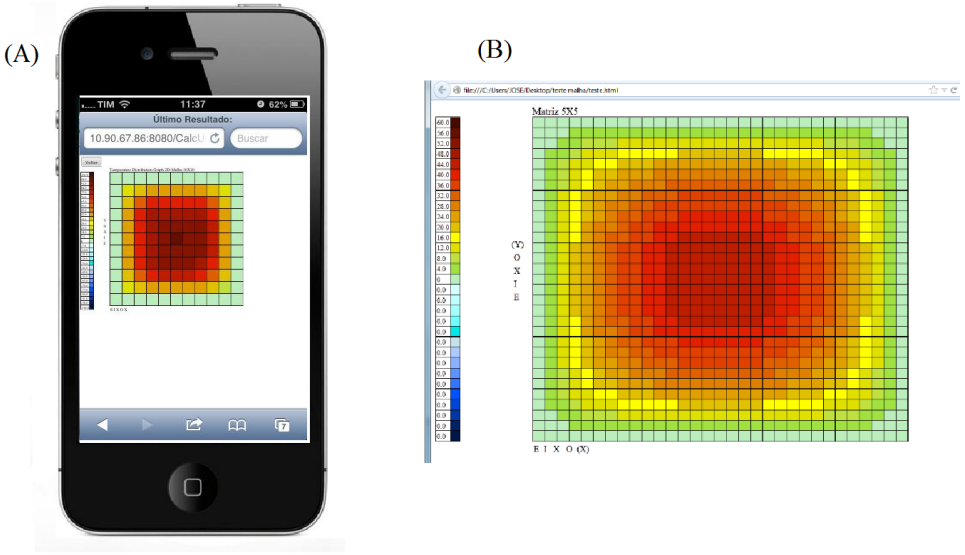


Figura 6.12 – Comparação de malhas refinadas entre dois dispositivos e sistemas operacionais: (A) dispositivo móvel e (B) web browser convencional.

Figura 6.13 – Visualização da geração de malhas (A) e resolução de uma EDP em uma instância de tempo (B) do sistema *web* acessado de um dispositivo móvel.

'Resultado' no banco de dados para exibir o resultado. O usuário entra somente com os parâmetros e entrada e realiza as interações com o gerador de malhas, sobre o qual ele pode descrever a geometria pelo dispositivo móvel. Pode-se perceber que as simulações de diferentes sistemas de coordenadas, seja cartesiana (Figuras 6.12) ou generalizadas (Figuras 6.13), são exibidas na tela dos dispositivos.

Na resolução por DDF, as etapas de pré-processamento e pós-processamento são similares entre todos os EDPs, diferenciando-se somente na etapa do processamento

propriamente dito. Além do mais, o uso de uma discretização estruturada permite que todos os resultados (pontos da malha e todas as instâncias de resultados) sejam armazenados da mesma forma no banco de dados, independente da equação aplicada, o que facilita a consulta de um histórico de problemas já resolvidos pelo próprio usuário e/ou por outros (Figura 6.14). A partir da lista obtida dessa busca, pode-se expandir cada um deles, consultando seus resultados em todas as instâncias de tempo (Figura 6.15).

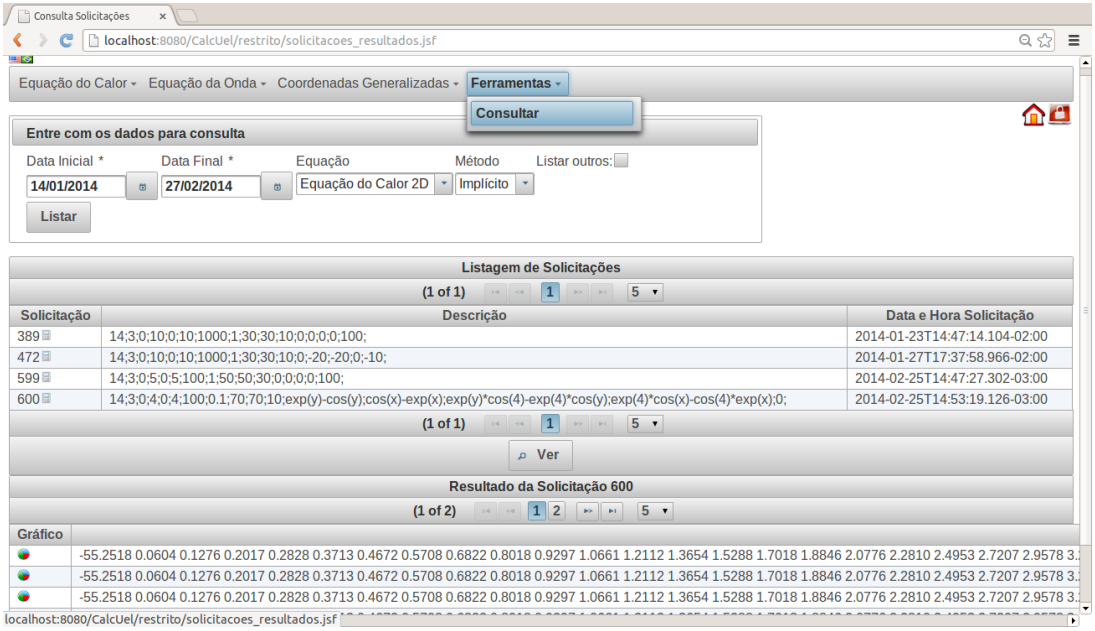


Figura 6.14 – Sistema de buscas de problemas resolvidos.

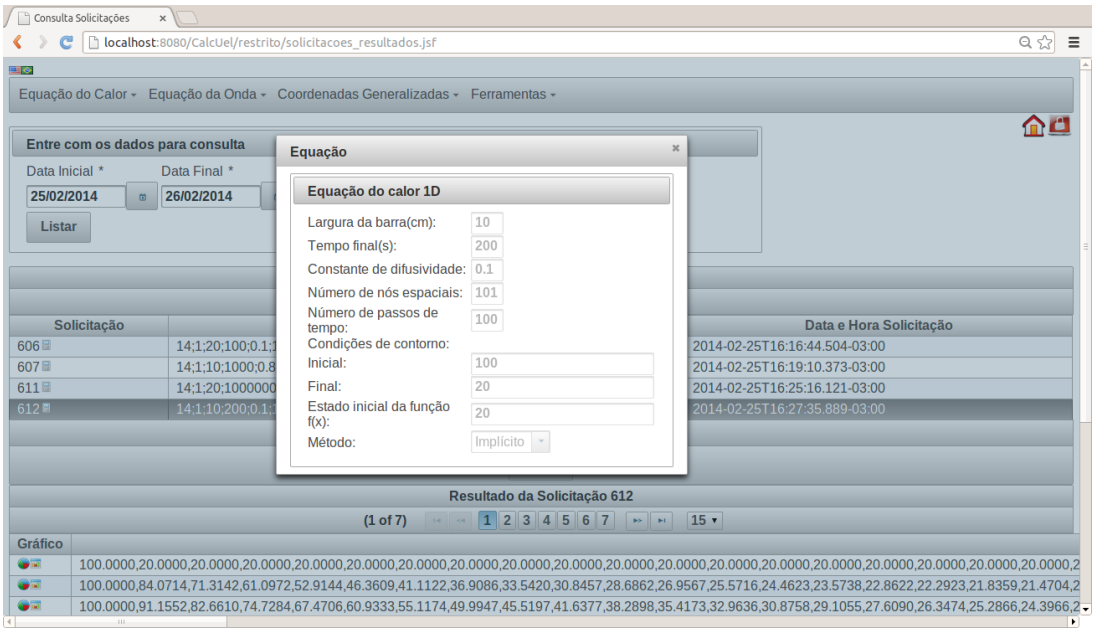


Figura 6.15 – Consulta de um problema resolvido, com a abertura dos parâmetros utilizados.

O sistema de busca de resultados no banco de dados auxilia na consulta de problemas já resolvidos (Figura 6.14 e 6.15). O sistema de buscas, além da solicitação em

cache para o retorno de problemas que já foram resolvidos, pode ser ampliado para outros fins. Um exemplo é o uso do conceito de erros relativos [26] para que o sistema liste um conjunto de problemas similares (além do que tiver exatamente os mesmos parâmetros), cujos resultados possuem diferenças de valores que sejam menor que a tolerância de erro estabelecido. Dessa forma, o sistema busca também por soluções em que uma determinada variável não tem uma influência significativa no resultado final, sendo uma possível futura implementação da análise de sensibilidade de modelos de EDPs [49].

Com o sistema de busca, os problemas resolvidos podem ser consultados e listados de qualquer lugar e a qualquer momento, criando um sistema colaborativo de modo que pesquisadores, professores, alunos e até mesmo pessoas de outras instituições possam ter acesso à base de dados das EDPs resolvidas. É possível, portanto, o uso desse sistema como um laboratório virtual de experimentos físicos. Este tema também tem sido um tema bastante abordado em diversas pesquisas [5, 33], e contribui significativamente para a transferência de conhecimento na prática e do ensinamento moderno.

7 CONCLUSÕES E TRABALHOS FUTUROS

O objetivo proposto neste trabalho foi atingido, através de um sistema *web* que efetua simulações de diversos fenômenos físicos desejados pelo usuário. Os problemas podem ser de diversas grandezas espaciais e temporais, desde a simulação de uma condução de calor em uma garrafa durante alguns minutos até a simulação do transporte de poluentes em um lago de dimensão maior durante alguns dias. Muitos destes são simulados através de EDPs, tais como a equação de Laplace, a equação de condução do calor, a equação da onda e a equação do transporte (advecção-difusão), em geometrias regulares ou generalizadas.

Para resolver um problema, o usuário necessita apenas fornecer os parâmetros de entrada do problema a ser resolvido desejado, sendo que os processamentos são executados em um servidor remoto. Com essa aplicação, o *deployment* do serviço na *web* permite que o acesso do mesmo através de um navegador de *Internet* ou de um dispositivo móvel. Dessa forma, o serviço torna-se um sistema colaborativo de compartilhamento de soluções de diversos problemas, sendo uma ferramenta útil para a transferência de conhecimento na prática para o meio de ensino moderno e tecnológico.

O recurso utilizado para comunicação ao servidor remoto e para visualização de malhas discretizadas permite que o serviço possa ser executado em dispositivos independente de seus desempenhos. O usuário pode executar o serviço em computadores *desktops* e dispositivos móveis, tais como *tablets* e *smartphones*. Dessa forma, o dispositivo não necessita possuir recursos de *hardwares* e *softwares* potentes, desde que estes possuam um navegador com suporte à HTML, Java e acesso à Internet.

O trabalho proposto se destaca pela aplicação de problemas envolvendo equações diferenciais parciais em tempo real com uso de diferenças finitas, um método numérico considerado eficiente. Além disso, o sistema possui uma integração com banco de dados, *web services* e outras linguagens de programação para execução da aplicação. Portanto, a relevância do nosso trabalho se contextualiza em diversos serviços e aplicações científicas que estão sendo portados para a *web*, com foco em um *design* na visualização dos resultados e em ferramentas interativas.

Diante do trabalho proposto, a principal contribuição deste trabalho é a disponibilidade de uma arquitetura *web* que resolva diversos problemas descritos por meio de EDPs. Esta arquitetura é considerada interoperável por possuir um padrão aberto e permitir que o *web service* possa realizar a comunicação com o *front-end* (GUI) e o *back-end* (banco de dados e a aplicação de cálculo por DDF) de uma forma transparente, independente do sistema operacional em que estes estão sendo implantados. Este padrão aberto, estabelecido pelo modelo em camadas, permitiu uma fácil adaptação e modificação na inclusão de outros tipos de sistemas de coordenadas. Ademais, o banco de dados de armazenamento de soluções

pode ser continuamente expandido por qualquer usuário ou pesquisador, fazendo com que o sistema *web* proposto seja constituído de uma biblioteca digital de problemas reais.

7.1 TRABALHOS FUTUTOS

Como uma perspectiva, pode-se melhorar também o desempenho do processo de resolução de sistemas lineares explorando bibliotecas de funções voltadas para ambiente GPGPU (*General-Purpose Graphical Processing Unit*) e programação paralela. Dessa forma, facilita-se a execução de problemas de grande porte e equações diferenciais mais complexas.

Pretende-se realizar a inclusão de diversos tipos de EDPs, como a equação de Navier-Stokes. Essa equação permite ampliar a possibilidade de resolver uma gama maior de problemas reais, como por exemplo, através do cálculo de campos de velocidades, sendo necessário somente alterar o módulo de processamento.

Aproveitar o uso de ferramentas computacionais interativas para fornecer um viés mais educacional e didático ao sistema desenvolvido, como um meio de fornecer ensinos sobre os funcionamentos das EDPs sobre diversos fenômenos físicos. Além de ser uma ferramenta útil para pesquisadores, o sistema pode ser de grande auxílio para professores e instituições de ensinos.

Implementar a análise de sensibilidade de modelos para analisar a influência de uma variável no resultado final de uma EDP e para buscar problemas similares com resultados que possuem diferenças de valores que sejam menor que a tolerância de erro estabelecido.

Outra frente que futuramente merece uma grande atenção é o foco em resolução de problemas tridimensionais, pois é possível modificar a camada de interface *web* para permitir a simulação e visualização de problemas relacionados, especialmente na camada de interface gráfica e de processamento de cálculos. As tecnologias baseadas em web, como o WebGL, proporcionam executar aplicações gráficas 3D em um ambiente similar ao desenvolvido neste trabalho.

Referências

[1] ROMEIRO, N. M. L.; CIRILO, E. R.; NATTI, P. L. Estudo do escoamento e da dispersão de poluentes na superfície do Lago Igapó I. In: *Anais do II Seminário Nacional sobre Regeneração Ambiental de Cidades - Águas Urbanas II*. [S.l.: s.n.], 2007. p. 1–17.

[2] TONTI, E. Why starting from differential equations for computational physics? *Journal of Computational Physics*, 2014, Elsevier Inc., v. 257, p. 1260–1290, jan. 2014. ISSN 00219991.

[3] ROMEIRO, N. M. et al. Local calibration of coliforms parameters of water quality problem at Igapó I Lake, Londrina, Paraná, Brazil. *Ecological Modelling*, 2011, Elsevier B.V., v. 222, n. 11, p. 1888–1896, jun. 2011. ISSN 03043800.

[4] GUPTA, P. K. et al. Solution of the heat transfer problem in tissues during hyperthermia by finite difference-decomposition method. *Applied Mathematics and Computation*, 2013, v. 219, n. 12, p. 6882–6892, fev. 2013. ISSN 00963003.

[5] SINGH, V. et al. An educational website on interferometry. In: *2012 IEEE International Conference on Technology Enhanced Education (ICTEE)*. [S.l.: s.n.], 2012. p. 1–10. ISBN 978-1-4577-0726-1.

[6] NEVES, L. A.; MACHADO, J. M. Integrating open source codes for solid modeling and automatic mesh generation in FEM applications. In: *Proceedings of 2011 Cross Strait Quad-Regional Radio Science and Wireless Technology Conference*. [S.l.]: Ieee, 2011. p. 36–39. ISBN 978-1-4244-9792-8.

[7] CIRILO, E. R.; BORTOLI, A. L. D. Cubic Splines for Trachea and Bronchial Tubes Grid Generation. *Semina: Ciências Exatas e Tecnológicas*, 2006, v. 27, p. 147–155, 2006.

[8] LIU, Y.; XING, H. A boundary focused quadrilateral mesh generation algorithm for multi-material structures. *Journal of Computational Physics*, 2013, Elsevier Inc., v. 232, n. 1, p. 516–528, jan. 2013. ISSN 00219991.

[9] D'AMATO, J.; VéNERE, M. A CPU–GPU framework for optimizing the quality of large meshes. *Journal of Parallel and Distributed Computing*, 2013, Elsevier Inc., v. 73, n. 8, p. 1127–1134, ago. 2013. ISSN 07437315.

[10] WENG, W.-C. Web-based post-processing visualization system for finite element analysis. *Advances in Engineering Software*, 2011, Elsevier Ltd, v. 42, n. 6, p. 398–407, jun. 2011. ISSN 09659978.

[11] WALKER, J. D.; CHAPRA, S. C. A client-side web application for interactive environmental simulation modeling. *Environmental Modelling & Software*, 2014, Elsevier Ltd, v. 55, p. 49–60, maio 2014. ISSN 13648152.

[12] GRANDE, A. et al. Educational Computer Simulations for Visualizing and Understanding the Interaction of Electromagnetic Waves with Metamaterials . In: *IEEE EDUCON Education Engineering*. [S.l.: s.n.], 2010. p. 543–547. ISBN 9781424465712.

[13] SALA, F. a. Mobile phone as a platform for numerical simulation. *Computer Physics Communications*, 2012, Elsevier B.V., v. 183, n. 1, p. 26–29, jan. 2012. ISSN 00104655.

[14] TEH, S.; KOH, H. Mobile apps for water quality simulation. In: *International Conference on Teaching, Assessment and Learning for Engineering (TALE)*. [S.l.: s.n.], 2013. p. 182–187. ISBN 9781467363556.

[15] XIAO-PING, C.; RU-FU, H. Web Services Based Mechatronics Product Remote Finite Element Analysis Techniques. In: *2010 International Conference On Computer Design And Appliations (ICCDA)*. [S.l.: s.n.], 2010. v. 4, n. Iccda. ISBN 9781424471645.

[16] BENNETT, S. et al. Implementing Web 2.0 technologies in higher education: A collective case study. *Computers & Education*, 2012, Elsevier Ltd, v. 59, n. 2, p. 524–534, set. 2012. ISSN 03601315.

[17] DUMONT, C.; MOURLIN, F. A Mobile Computing Architecture for Numerical Simulation. In: *International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM'07)*. [S.l.]: Ieee, 2007. p. 68–74. ISBN 0-7695-2993-3.

[18] ISBASOIU, E. C. Numerical Calculation, Web Services and Principles for Physics Applications. In: *2010 Fourth UKSim European Symposium on Computer Modeling and Simulation*. [S.l.]: Ieee, 2010. p. 387–390. ISBN 978-1-4244-9313-5.

[19] JADEJA, Y.; MODI, K. Cloud computing-concepts, architecture and challenges. In: *2012 International Conference on Computing, Electronics and Electrical Technologies*. [S.l.: s.n.], 2012. p. 877–880. ISBN 9781467302104.

[20] VECCHIOLA, C.; PANDEY, S.; BUYYA, R. High-Performance Cloud Computing: A View of Scientific Applications. In: *2009 10th International Symposium on Pervasive Systems, Algorithms, and Networks*. [S.l.: s.n.], 2009. p. 4–16. ISBN 978-1-4244-5403-7.

[21] ARI, I.; MUHTAROGLU, N. Design and implementation of a cloud computing service for finite element analysis. *Advances in Engineering Software*, 2012, Elsevier Ltd, nov. 2012. ISSN 09659978.

[22] JORISSEN, K.; VILA, F.; REHR, J. A high performance scientific cloud computing environment for materials simulations. *Computer Physics Communications*, 2012, v. 183, n. 9, p. 1911–1919, set. 2012. ISSN 00104655.

[23] HU, L.; CHE, X.; XIE, Z. GPGPU cloud: A paradigm for general purpose computing. *Tsinghua Science and Technology*, 2013, v. 18, n. 1, 2013.

[24] BYRNE, J.; HEAVEY, C.; BYRNE, P. A review of Web-based simulation and supporting tools. *Simulation Modelling Practice and Theory*, 2010, Elsevier B.V., v. 18, n. 3, p. 253–276, mar. 2010. ISSN 1569190X.

[25] WRIEDT, T. Mie theory 1908, on the mobile phone 2008. *Journal of Quantitative Spectroscopy and Radiative Transfer*, 2008, v. 109, n. 8, p. 1543–1548, maio 2008. ISSN 00224073.

[26] REIMER, A. S.; CHEVIAKOV, A. F. A Matlab-based finite-difference solver for the Poisson problem with mixed Dirichlet-Neumann boundary conditions. *Computer Physics Communications*, 2013, v. 184, n. 3, p. 783–798, mar. 2013. ISSN 00104655.

[27] HAN, F.; DAI, W. New Higher-Order Compact Finite Difference Schemes for 1D Heat Conduction Equations. *Applied Mathematical Modelling*, 2013, n. March, mar. 2013. ISSN 0307904X.

[28] AMODIO, P.; SETTANNI, G. A finite differences MATLAB code for the numerical solution of second order singular perturbation problems. *Journal of Computational and Applied Mathematics*, 2012, v. 236, n. 16, p. 3869–3879, out. 2012. ISSN 03770427.

[29] IFTODE, C. A matlab implementation of FDTD algorithm for bidimensional layers. In: *2012 10th International Symposium on Electronics and Telecommunications*. [S.l.]: IEEE, 2012. p. 385–388.

[30] CHAVEZ-GONZALEZ, A. F. et al. Finite differences software for the numeric analysis of a non-destructive electromagnetic testing system. In: *CONIELECOMP 2013, 23rd International Conference on Electronics, Communications and Computing*. [S.l.]: IEEE, 2013. p. 82–86.

[31] ALFONSECA, M.; LARA, J. D.; VANGHELUWE, H. Web II: web-based simulation of systems described by partial differential equations. In: *Simulation Conference, 2001. Proceedings of the Winter*. [S.l.: s.n.], 2001. p. 629 – 636.

[32] BARBOZA, D. C. et al. A Simple Architecture for Digital Games on Demand Using Low Performance Resources under a Cloud Computing Paradigm. In: *2010 Brazilian Symposium on Games and Digital Entertainment*. [S.l.: s.n.], 2010. p. 33–39. ISBN 978-1-61284-391-9.

[33] PERUS, I. et al. A web-based methodology for the prediction of approximate IDA curves. *Earthquake Engineering & Structural Dynamics*, 2013, v. 42, n. April 2012, p. 43–60, 2013.

[34] XU, L.; KE-BIAO, Y. Research on triangular mesh generation algorithm of point clouds. In: *2010 2nd International Conference on Computer Engineering and Technology*. [S.l.]: Ieee, 2010. p. 638–641. ISBN 978-1-4244-6347-3.

[35] NOTSU, H.; UEYAMA, D.; YAMAGUCHI, M. A self-organized mesh generator using pattern formation in a reaction–diffusion system. *Applied Mathematics Letters*, 2013, Elsevier Ltd, v. 26, n. 2, p. 201–206, fev. 2013. ISSN 08939659.

[36] SILVA, W. P. da et al. Numerical simulation of diffusive processes in solids of revolution via the finite volume method and generalized coordinates. *International Journal of Heat and Mass Transfer*, 2009, Elsevier Ltd, v. 52, n. 21-22, p. 4976–4985, out. 2009. ISSN 00179310.

[37] DOBOS, J.; STEED, A. 3D Revision Control Framework. In: *Web3D 2013*. [S.l.: s.n.], 2012. v. 1, n. 212, p. 121–130. ISBN 9781450314329.

[38] SCHWARTZ, C. et al. WebGL-based streaming and presentation of objects with bidirectional texture functions. *Journal on Computing and Cultural Heritage*, 2013, v. 6, n. 3, p. 1–21, jul. 2013. ISSN 15564673.

[39] BURDEN, R. L.; FAIRES, J. D. *Numerical Analysis*. [S.l.]: Thomson Learning Inc., 2001.

[40] GENUCHTEN, M. T. V. Analytical solutions for chemical transport with simultaneous adsorption, zero-order production and first-order decay. *Journal of Hydrology*, 1981, v. 49, p. 213–233, 1981.

[41] TSITSIKLIS, J. N. A comparison of Jacobi and Gauss-Seidel parallel iterations. *Applied Mathematics Letters*, 1989, v. 2, n. 2, p. 167–170, 1989.

[42] YUAN, J.; ZONTINI, D. Comparison theorems of preconditioned Gauss–Seidel methods for M-matrices. *Applied Mathematics and Computation*, 2012, Elsevier Inc., v. 219, n. 4, p. 1947–1957, nov. 2012. ISSN 00963003.

[43] MALISKA, C. R. *Transferência de Calor e Mecânica dos Fluidos Computacional*. 2a. ed. Rio de Janeiro: Editora Livros Técnicos e Científicos Editora, 2004. 468 p.

[44] DICKEY, T. E. *xterm*. 2013. Disponível em: <http://invisible-island.net/xterm/xterm.html>.

[45] HU, C. et al. Geospatial Web Service for Remote Sensing Data Visualization. *2011 IEEE International Conference on Advanced Information Networking and Applications*, 2011, Ieee, p. 594–601, mar. 2011.

[46] RUSSELL, M.; PROBERT, S. FDiff3: a finite-difference solver for facilitating understanding of heat conduction and numerical analysis. *Applied Energy*, 2004, v. 79, n. 4, p. 443–456, dez. 2004. ISSN 03062619.

[47] MATSUNAGA, F. T. et al. CloudPDE: a web solver of differential equations by finite difference discretization. In: *IADIS - International Conference on Applied Computing (prelo)*. [S.l.: s.n.], 2013.

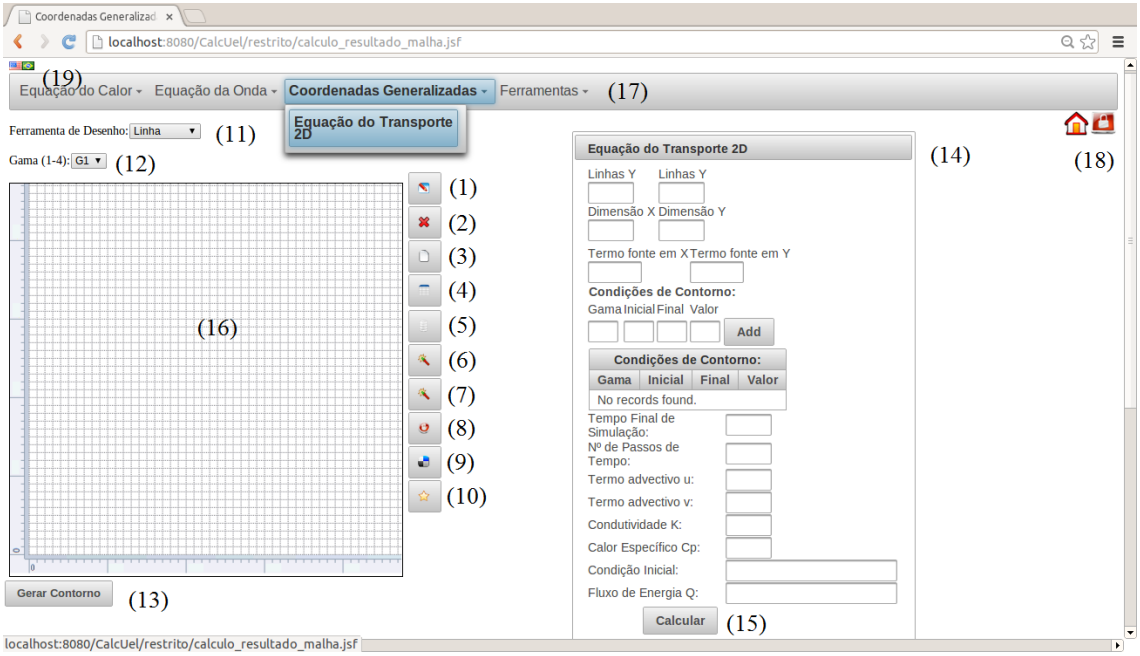
[48] YANG, Z.; LU, J.; ELGAMAL, A. A web-based platform for computer simulation of seismic ground response. *Advances in Engineering Software*, 2004, v. 35, n. 5, p. 249–259, maio 2004. ISSN 09659978.

[49] WEIBE, A. Y.; HUISINGA, W. Error-controlled global sensitivity analysis of ordinary differential equations. *Journal of Computational Physics*, 2011, v. 230, n. 17, p. 6824–6842, jul. 2011. ISSN 00219991.

APÊNDICE

APÊNDICE I

Descrição da interface gráfica do gerador de malhas em coordenadas generalizadas.



(1) Definir dimensão do problema
(2) Limpar tela
(3) Ocultar grades
(4) Mostrar grades
(5) Carregar malha de um arquivo – o usuário não necessita especificar os pontos da geometria novamente caso já tenha algum arquivo no seu histórico de problemas resolvidos
(6) Mostrar passo a passo a distribuição de energia em cada instância de tempo de um problema carregado
(7) Ativar o modo temporizador (automatização do passo (6)) – simula automaticamente um problema carregado mostrando a variação da distribuição de energia a cada n segundos
(8) Parar/continuar temporizador
(9) Mostrar/ocultar distribuição de energia – útil para visualizar somente a malha
(10) Fechar polígono – o usuário tem esta opção disponível quando for inserir a última linha da figura geométrica
(11) Opções de desenhos disponíveis: linha e pontos
(12) Especifica o contorno em que um conjunto de pontos (x,y) pertence
(13) Gerar contorno – envia o conjunto de coordenadas (x,y) que forma a geometria do problema como uma solicitação ao servidor remoto
(14) Parâmetros da EDP a ser resolvida
(15) Envia a solicitação de cálculo ao servidor remoto
(16) Tela em Javascript de desenho da geometria e do visualizador de malhas – os valores da escala da régua são calculados automaticamente (em sistemas de coordenadas (*x,y*)) após a definição da dimensão no item (1)
(17) Barra de menus
(18) Página inicial/*logoff*
(19) Definição do idioma (internacionalização do sistema *web*)

ANEXOS

ANEXO I

Produções científicas no tema da dissertação em congressos aceitos como qualis na área de Ciência da Computação, entitulados de:

- *CloudPDE: a web solver of differential equations by finite difference discretization*. Publicado em: IADIS - International Conference on Applied Computing, 2013, Texas, EUA.
- *Web Technologies for Differential Equations Solving by Finite Difference Method with Mesh Visualization Independent on Device Performance*. Publicado em: International Conference on Chilean Computer Society, 2013, Temuco, Chile.

CLOUDPDE: A WEB SOLVER OF DIFFERENTIAL EQUATIONS BY FINITE DIFFERENCE DISCRETIZATION

Fabio Takeshi Matsunaga, José Luiz Vilas Boas, Neyva Maria Lopes Romeiro
and Jacques Duílio Brancher
Departamento de Computação, Universidade Estadual de Londrina (UEL)
Rod. Celso Garcia Cid, Cx. Postal 10011, Londrina-PR Brazil

ABSTRACT

The Internet and Web technologies have been widely applied on many scientific fields, such as mathematic, physic, chemistry and engineering. These scientific applications are migrating to cloud and web services, which are becoming a new paradigm of computational applications. The aim of this work is to develop a cloud service named CloudPDE that uses the concept of cloud computing to solve problems involving partial differential equations, such as heat and wave equations, with 1 and 2 dimensions by finite difference discretization method to solve large problems. The system consists of a front-end and back-end client-server communication, a database of results and user/request information storage and a graphical user interface for outputs visualization. The CloudPDE service can be executed on devices independent on their performance, since the high-order matrixes computations and high-cost calculations required for accurate simulations are made on a remote server.

KEYWORDS

Back-end, Front-end, Linear Algebra, Scientific Applications, Web Service

1. INTRODUCTION

Cloud computing and Web Services aim to provide services which applications, platforms or infra-structures are located on remote servers and data-centers, deliver and manage services over the internet, also allowing users to access remote files and applications. These computer models have characteristics such as accessibility, scalability, coherency, virtualization and interoperability and (Moghe et al., 2012). Thus, many internet services have been developed in many scientific applications, such as mathematic, physic, chemistry and engineering.

Most specifically, in many works this technology has been focused to numerical simulations and partial differential equations solutions (Ari and Muhtaroglu, 2012; Jorissen et al., 2012). Such scientific applications often require complexes and high-cost calculation, large amount of data, high performance environment and several computational resources. Furthermore, cloud services applications have being developed for real-time interaction and communication with users (Villegas-Puyod, 2012), which requires high performance algorithms and methods.

Problems involving mathematical applications are difficulty for conventional means in classrooms, because of the extensive theory and experimental knowledge. Some developed web tools (Grande et al., 2010; Basak, 2012) for mathematical and physical phenomena visualization uses video recording and interactive tools which require a large computer memory, ideal hardware configuration and local softwares or plug-ins installations for ideal design and responses.

There are many advantages and motivations of web technologies for numerical simulations (Perus et al., 2013), such as the automatic licensing, system maintenance, upgrade and version control made from a web server for global use and facility of deployment. Furthermore, finite difference discretization method is a straightforward method and for refined mesh, it is necessary high cost linear algebra operations, such as linear systems resolution (Reimer and Cheviakov, 2013) or sparse matrix-vector multiplication and a huge amount of data requiring a remote database for information storage and server applications which are maintained and easily accessible from anywhere and anytime.

Considering the above, this paper presents the development of a service called CloudPDE to solve 1D and 2D problems solutions involving time dependent partial differential equations (PDE), such as heat conduction and wave propagation equation. The method used to solve these equations is the finite difference discretization considering Cartesian coordinates. For this purpose, a real time front-end client-server communication and a back-end to process all the calculations in remote server independent on user's machine or device performance, were developed in order to help the understanding of heat transfer phenomena and interaction with surfaces and boundary conditions.

Thus, the structure of this paper is as follows: Section 2 presents a review of literatures about related works; Section 3 presents the finite difference discretization method for 2D heat conduction equation and 1D wave propagation equation. In Section 4, we described the CloudPDE architecture to support the demand of finite difference calculation. Section 5 shows some results examples of both cases of equations. Section 6 is the conclusions and some future works.

2. LITERATURE REVIEW

Several works have been developed to solve differential equation by finite difference discretization method. Russel and Probert (2004) for example developed the FDiff3 educational software to solve heat conduction by finite difference method, aiming to facilitate the understanding of the physical phenomena. Reimer and Cheviakov (2013) developed a Matlab simulator to solve problem involving Poisson Equation, a time-independent PDE to calculate the static heat distribution, in rectangular, cylindrical and spherical coordinates, with mesh refinement control in a local application. However, many scientific applications are migrating to the Internet, providing a new paradigm of softwares and computing services.

Considering the advents of this new computing paradigm, several web and cloud services for scientific applications have been developed recently. One of the most relevant and recent, we can cite the Ari and Muhtaroglu (2012) work. On this work, the authors implanted a cloud service for linear and non-linear mechanical structures, using finite elements method to solve PDEs. In the same research line, Xiao-ping and Hu (2010) developed a web service for remote finite elements analysis focused on ideal design of complex products aiming the local fast modeling.

Another work with the same approach was developed to support scientific applications, such as a PaaS that provides some material sciences and quantum chemistry applications source-codes and benchmarks and interface tools (Jorissen et al., 2012). This work developed a platform using high performance resources and the scalability was tested in cloud clusters distributed on Amazon Web Services virtual machines.

In this same line of research, Costa et al., (2012) for example, established an information management, data analysis and processing technique to numerical simulations of natural phenomena in scientific cloud applications. The authors mentioned that an accurate simulation is performed considering a high scale simulation with more level of detail and a fine grained mesh representation of a physical domain. The main work challenges were the simulation data storage, the physical data representation and data management.

3. FINITE DIFFERENCE DISCRETIZATION METHOD

The numerical method chosen to solve PDEs was the finite difference discretization. This method has been applied to solve many PDEs in several scientific areas (Yang and Sen, 2009). The main idea is to solve PDEs using finite difference equations obtained by Taylor series to approximate the partial derivatives. Then, this method consists on spatial domain discretization in a computational mesh using the finite difference equations to approximate the values of each mesh point. The next sections will present the kinds of PDEs to be discretized and considered on this work.

3.1 One Dimensional Wave Equation

On the one dimensional wave equation (1), different from 2D heat equation, we considered the mesh dimension of x axis $[x_0,x_f]$, the final time simulation ($Tmax$), the boundaries conditions given by $u(x_0,t)$ and

$u(x_f,t)$, the initial condition $u_0(x,0)$, the initial speed condition $\frac{\partial u}{\partial t}(x,0)=g(x)$ and a C constant. The mesh is discretized according of N_x and N_t partitions, which refers to number of partitions in x and t respectively, and the spacing of each mesh node $\Delta x = (x_f\text{-}x_0)/(N_x\text{-}1)$ and $\Delta t = (T_{max})/(N_t\text{-}1)$.

$$\frac{\partial^2 u}{\partial t^2}=C^2\frac{\partial^2 u}{\partial x^2} \quad (x_0 \leqslant x \leqslant x_f) \quad (1)$$

The Equation 1 discretization consists on substitute the second order partial derivatives on time-domain and the second order partial derivative on spatial-domain finite difference equation (2), and the initial speed condition application $g(x)$ for $u_{i,1}$ (3).

$$\frac{\partial^2 u}{\partial t^2}=\frac{u_{i,j+1}-2u_{i,j}+u_{i,j-1}}{\Delta t^2} \quad \frac{\partial^2 u}{\partial x^2}=\frac{u_{i+1,j}-2u_{i,j}+u_{i-1,j}}{\Delta x^2} \quad (2)$$

$$u_{i,1}=(1-\lambda^2)u_0(x_i,0)+\frac{\lambda^2}{2}u_0(x_{i+1},0)+\frac{\lambda^2}{2}u_0(x_{i-1},0)+C\,g(x_i) \quad (3)$$

Considering $u_{i,j}$ the time-step mesh approximation in a time t_j, a spatial x_i and $\lambda=(C.\Delta t)/\Delta x$, the scheme of the discretized equation is showed in (4). It is necessary that $\lambda \leq 1$ for the stability condition and the non-oscillation during the simulation.

$$u_{i,j+1}=2(1-\lambda^2)u_{i,j}+\lambda^2 u_{i+1,j}+\lambda^2 u_{i-1,j}-u_{i,j-1} \quad (4)$$

The equation above (4) leads to a matrix-vector multiplication, where the matrix A is a tridiagonal formed of a main diagonal [$2(1\text{-}\lambda^2)$, $2(1\text{-}\lambda^2)$, $2(1\text{-}\lambda^2)$ … $2(1\text{-}\lambda^2)$], a subdiagonal and a superdiagonal formed of [λ^2 ,λ^2,λ^2…λ^2] elements, and the vector is $u_{i,j}$ elements. As we can see on (4), the discretized equation finds the t_{j+1} time step of wave propagation, then the vector resulted from the multiplication is subtracted from $u_{i,j-1}$, which is found by the initial speed condition equation (3). For these operations we used routines from BLAS/LAPACK (2013) library, which has high-performance linear algebra functions that execute vector operations efficiently on shared memories and parallel processors. The routines used to perform these operations are SBMV (matrix-vector operation for banded matrix) for tridiagonal matrix-vector multiplication and a SAXPY (vector addition with scalar multiplication) operation. for vectors subtractions. Then, the operation to find $u_{i,j+1}$ is SAXPY(SBMV(A, $u_{i,j}$), $u_{i,j-1}$, -1), where -1 indicates that the SAXPY performs a vector subtraction.

3.2 Two Dimensional Heat Equation

On the two dimensional heat equation (5), we consider the mesh dimensions $[x_0,x_f]$ e $[y_0,y_f]$, the final time simulation ($Tmax$), the boundaries conditions given by $T(x_0,y,t)$, $T(x_f,y,t)$, $T(x,y_0,t)$ e $T(x,y_f,t)$ and the initial condition $T_0(x,y,0)$. The mesh is discretized according of N_x, N_y and N_t partitions, which refers to number of partitions in x, y and t respectively, and the spacing of each mesh node $\Delta x=(xf\text{-}x0)/(Nx\text{-}1)$, $\Delta y=(y_f\text{-}y0)/(N_y\text{-}1)$ and $\Delta t=(Tmax)/(N_t\text{-}1)$. The K is the thermal diffusivity constant (cm²/s) and the value depend on the material.

$$\frac{\partial T}{\partial t}=K\left(\frac{\partial^2 T}{\partial x^2}+\frac{\partial^2 T}{\partial y^2}\right) \quad (x_0 \leqslant x \leqslant x_f ; y_0 \leqslant y \leqslant y_f) \quad (5)$$

The Equation 1 discretization consists on substitute the first order partial derivatives on time-domain by the regressive difference equation (6) and the second order partial derivative on spatial-domain by the central difference equation (7).

$$\frac{\partial T}{\partial t} = \frac{T_{i,j}^{k} - T_{i,j}^{k-1}}{\Delta t} \quad (6)$$

$$\frac{\partial^2 T}{\partial x^2} = \frac{T_{i-1,j}^{k} - 2\,T_{i,j}^{k} + T_{i+1,j}^{k}}{\Delta x^2} \quad \frac{\partial^2 T}{\partial y^2} = \frac{T_{i,j-1}^{k} - 2\,T_{i,j}^{k} + T_{i,j+1}^{k}}{\Delta y^2} \quad (7)$$

Considering $T_{i,j}^{k}$ the time-step mesh approximation in a time t_h and a spatial point (x_i,y_j) and $\alpha = (K.\,\Delta t)/\,\Delta x^2$ and $\beta = (K.\,\Delta t)/\,\Delta y^2$, we obtain the implicit scheme of the discretized equation (8).

$$T_{i,j}^{k} = (1 + 2\alpha + 2\beta)\,T_{i,j}^{k+1} - \alpha T_{i+1,j}^{k+1} - \alpha T_{i-1,j}^{k+1} - \beta T_{i,j+1}^{k+1} - \beta T_{i,j-1}^{k+1} \quad (8)$$

The equation above (8) refers to a block tridiagonal linear system with multiple diagonals (Reimer and Cheviakov, 2013), which solution are the values of $T_{i,j}^{k}$ in a set of interior points of the mesh. These linear systems are solved by LU factorization adapted to special matrixes, provided by the LAPACK (Linear Algebra Package) library. In this case, the routine SGBSV were selected, because it is optimized to banded matrixes with multiple diagonals, which is stored as an array for each column j: $AB(KU+KL+i-j, j) = A(i,j)$ em que $Max(0,j-KU) \leq i \leq Min(n,j+KL)$, where KU is the number of superdiagonals, KL is the number of subdiagonals and n the banded matrix A order. This storage means make the SGBSV routine efficient to solving linear systems, considering only the non-zero values of matrixes.

4. CLOUD SERVICE ARCHITECTURE

This section presents a general architecture of the CloudPDE (Figure 1) composed of a front-end of client-server communication and interaction, and a back-end that executes the applications of each requester user in a remote server.

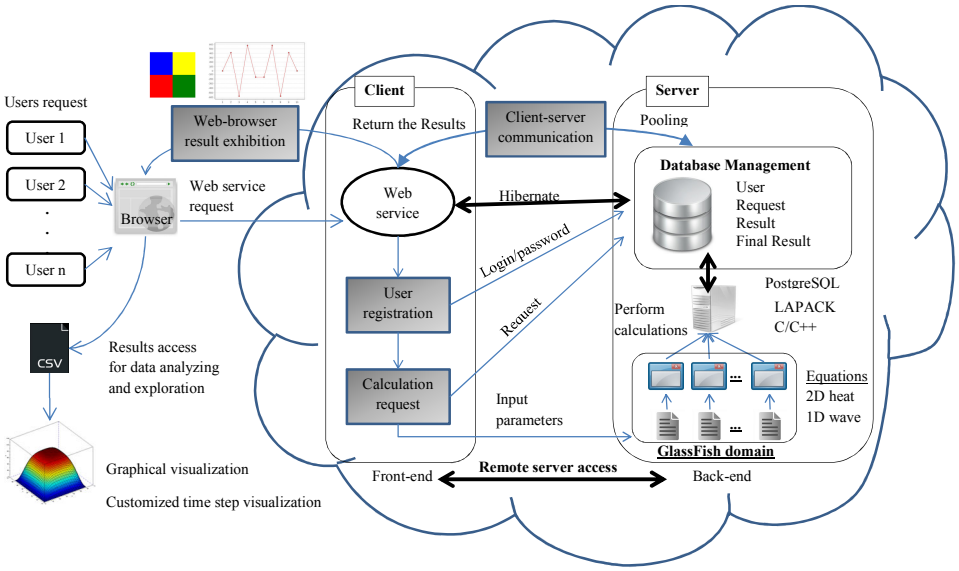


Figure 1. CloudPDE architecture (front-end, back-end and user interface) and main modules

The main modules presented on the cloud service architecture (Figure 1) are described below:

1. **User registration**: the user makes the registration and mounts the problem, entering the parameters.
2. **Calculation request**: Send the problem to the solver contained on a remote server.

3. **Client-server communication**: The server receives the problem and start solving it. For each iteration, the server sends the partial solution to the client.
4. **Web-browser result exhibition**: The client shows the results in the browser.

The client and server web services are built in NetBeans IDE as a Java Servlet application, and the deploy of them was made on GlassFish 3.1 server. The PostgreSQL 9.2 database was used as mean of calculation and output storage and queries and the client-server database communication are performed by Hibernate Framework. The finite difference server application was developed in C/C++ language with PostgreSQL integration to store the outputs on database and LAPACK integration to perform the optimized and efficient linear algebra calculations.

4.1 Front-end and Back-end

The front-end of the cloud service is responsible to information and inputs/outputs exchange between client and server, providing a real time interaction with the users. The main way to make this interaction is the database that stores all the related information (users, problems and solutions). The entity-relationship model of database is shown on Figure 2.

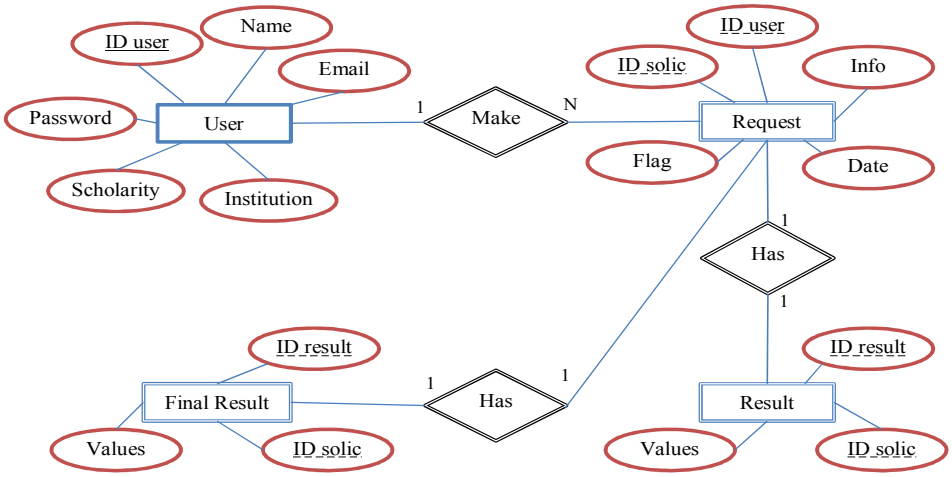


Figure 2. Entity-relationship model of client-server communication database.

As we can see on Figure 2, when a request is made, the front-end inserts the user information on 'User' table and the respective request information, with flag='A' on 'Request' table, indicating the calculation are being processed. The front-end also sends the users input to server as a text file format indexed with the user ID. This file is stored on the GlassFish server domain and is read by the back-end application, which is responsible to perform the calculations and updates the results on each time instance.

While the calculations are performed on each Δt instance, the results are inserted on 'Result' table as *varchar*(1000000) format and mirrored to 'Final Result' table performed by a trigger function until *Tmax* time and the flag='A' are maintained. When the calculation is terminated, the flag field is assigned as 'F', activating another trigger to remove all results information related to a determined ID request. This way makes the 'Result' a temporary table, aiming to not degrade the performance during the queries and real time results exhibition, especially when many simultaneous users are accessing the service.

The 'Final Result' table is a history of results of 'Result', utilized to future queries, in the case when a new user define a problem to solve, the cloud service makes a search on the database to verify their existence. If the return is true, the solved problem is presented to the users. This management database model allows the sharing of previously solved problems, existing solutions and also provides an economical use of remote server resources, so it is not necessary to solve a problem whose solution is already stored in the database. In addition, it is possible to available the solution to any users and organize the information by date of request, name, e-mail or ID of users and institutions that requested the processed problems, creating a collaborative system accessible from anywhere.

4.2 Graphical User Interface and Mesh Visualization

While the calculations are performed, the respective results instances are exhibited to the user. The results queries and real time exhibition are performed by a component called pooling, from Prime Faces framework. The pooling makes asynchronous calls every three seconds, which is considered the ideal maximum time response for real time tasks (Villegas-Puyod, 2012), to verify if there is any new line inserted in a 'Result' table, that's it, if there is a new state of the 1D bar (wave equation) or 2D surface (heat equation) in a time instance. If the pooling process detects a new line inserted, the values of discretized mesh points are written on browser on CSV (Comma Separated Values) format. On the end of processing, all the mesh points values on all t time instances ($0<t<Tmax$) for both cases of equations are available to the user.

Furthermore, there is an option where the user can view the graphics correspondent to each time instance $0<t<$Tmax. For the 1D visualization, we used the JFreeChart framework to plot line graphics to represent the temperature distribution and variation along the time. The visualization process is performed by taking the tuple of 'Result' table, which values are CSVs represented as string, inserted in a DataTable (from PrimeFaces) grid and inserted on a JFreeChart dataset.

For the 2D discretized mesh visualization, it is used the chain colors of 15 shades for positive and negative temperatures values, where the largest and the smallest values of temperature in a t is divided by 15 (number of shades), from which the interval colors is constructed and a HTML table is built to represent the surface heat distribution. In this table mesh, the user can access the temperature values on each discretized region, as well as having access to meta-information, such as (x,y) position, the influential boundary condition and the temperature variation with respect to the previous t value.

Despite the simplicity of our graphical output visualization, the visualization of discretized mesh of heat distribution can be performed on any device with HTML and Java support browsers, such as personal computers, mobile devices (tablets and smartphones). The real-time visualization of heat conduction phenomena is possible due to the simple configuration of the mesh generated by colored HTML table.

5. RESULTS

To check the viability of the system, we solved two problems. The first one is a 1D wave equation (Figure 3) with: $C = 1$, $T(x_0,t)=0$, $T(x_f,t)=0$, the initial condition $T_0(x,0) = sen(\pi x)$ and the initial speed condition $g(x)=0$. The mesh dimension specified was [0,1] and the mesh size $N_x=1000$, implying in a resolution of a tridiagonal linear system of order 998. The final time instance was $T_{max}=2$ with $N_t=100$ time steps.

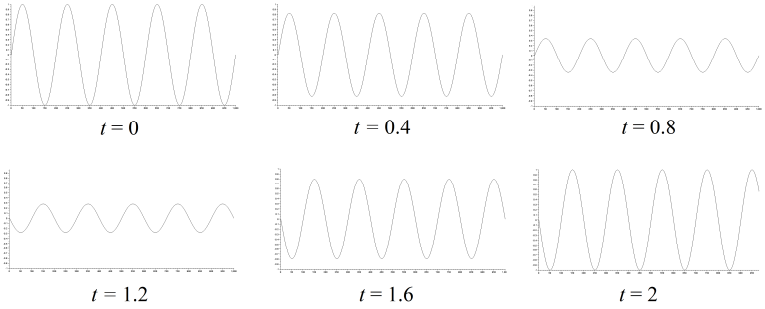


Figure 3. Line-charts of 1D wave equation outputs extracted in some time instances.

As we can see on Figure 3, the sinusoidal wave amplitude decreases over the time, until when approximately $t=1$ with the wave length constant. After that, the amplitude increases until the amplitude of the initial time step, however with displacement in x. Then, the wave equation can simulate this amplitude changing phenomenon periodically.

For the 2D heat equation (Figure 4), we considered $K=10^{-3}$, $T(x_0,y,t)=e^y-cos(y)$, $T(xf,y,t)=e^y cos(4)-e^4 cos(y)$, $T(x,y_0,t)=cos(x)-e^x$ and $T(x,y_f,t)=e^4 cos(x)-e^x cos(4)$ and the initial condition $T_0(x,y,0)=100$. The mesh dimension specified was $x=[0,4]$, $y=[0,4]$ and the mesh size $N_x=N_y=101$, which implies the resolution of a linear system of order 10,000 (matrix order of 10,000x10,000). The final time instance considered was $T_{max}=60$ and the number of time steps specified was $N_t=5$.

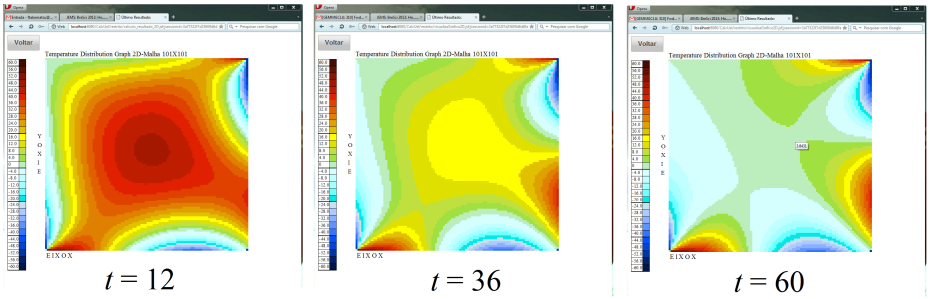


Figure 4. Mesh visualization in a web browser of 2D temperature distribution in some time instances.

On Figure 4, the mesh surface temperature decreases gradually over the time, starting from boundaries to the center, considering the initial condition of $T_0(x,y,0)=100$ (for $t=0$). This temperature decreasing is influenced by the boundary conditions and the density of temperature distribution surface has smooth transaction between mesh points. Thus, after many time instances ($t=2000$) and considering the absence of external heat sources, the surface temperature converges to a value following gradually the boundary conditions. In this case, it was necessary CloudPDE calculate the heat distribution $N_t=10$ times, that's it, on every time step the $Ax=b$ linear system was solved. The different values of mesh partitions N_x and N_y define different level of details, and consequently, the accuracy and complexity of the simulation (Figure 5) with less truncated simulation.

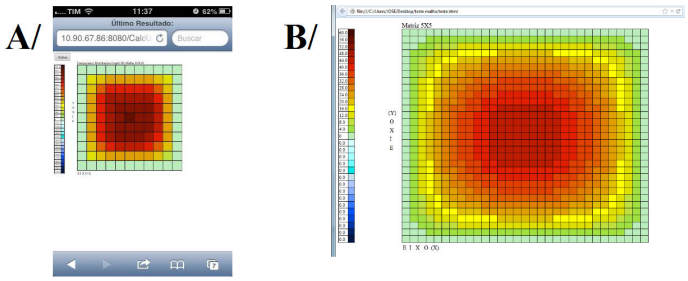


Figure 5. Mesh visualization in different devices and operating systems: A/ mobile device and a B/ conventional web browser.

We can see on both cases of equations that the CloudPDE simulated accurately physical phenomena by refined mesh definition, requiring a mesh discretization with large number of N_x and N_y (if 2D case) partitions. These refined meshes require high order and cost operations (SGBSV and SBMV). However, the service can be accessed and executed on low-performance devices, where all of discretization and matrix computations are performed in a remote server by high performance algorithms and computational resources. In addition, due to the use of HTML table to represent the heat distribution, the mesh visualization can be performed in any device with Java and HTML support, such as mobile devices (Figure 5). Moreover, the user does not need to know how the calculations are being executed either where all information are stored, even if the user has the impression that all the calculations are being processed locally. The only requirement of the user is a web browser and Internet access.

6. CONCLUSIONS

In conclusion, the CloudPDE showed able to simulate and physical phenomena (heat conduction and wave propagation). For more precise and level of detail simulating, it is necessary to define fine grained mesh representation on 1D and 2D cases, which requires a large number of numbers of partitions. Consequently, these refined meshes discretization requires high-order matrixes computations, but these calculations are performed in a cloud service hosted in a remote server, and then the service can be executed on devices

independent on their performance, such as low-performance personal computers and mobile devices. The only requirement is that the user has internet connection and a web browser.

Our work differs by solving real time PDEs using finite differences, with database, web services and linear algebra libraries integration, and differs by many desktop applications recently developed for the same purpose. The relevance of our work is given by paradigm change of scientific applications, which is migrating to Internet services. Then, the proposed service has some contributions and advantages: a collaborative system, were users can share their problems and results allowing research of problems previously solved, creating a way of practical knowledge transferring. Other is the efficient and practical learning about partial differential equations, useful on mathematics and physics phenomena teaching, individual learning and easy understanding.

The CloudPDE properties are characterized due to the real time interaction between client and server web services, the database, the user and the application of finite difference discretization. Therefore, the service developed has the potential to solve many scientific problems and in the future we can evolve our system including other PDEs, such as advection-diffusion equation, considering generalized coordinates, creating a mesh generator for these coordinates. Our work is in development, and this paper focused on the architecture implementing, then our service can be improved by implementing job scheduling to manage multiple simultaneous users and resource provisioning. In this way, our service will be available to researches to solve any kind of mathematical, physical and engineering problems.

ACKNOWLEDGEMENT

Thanks to CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nivel Superior) for the Master's scholarship to the student Fabio Takeshi Matsunaga.

REFERENCES

Ari, I. and Muhtaroglu, N., 2012. Design and implementation of a cloud computing service for finite element analysis. *Advances in Engineering Software*. In press.

Basak, H. H. and Ozansoy, K., 2012. Comparison of Traditional over WEB-Based Education: Case Study "Adobe Flash", *In Proceedings of the 2012 International Conference on Information Technology Based Higher Education and Training (ITHET)*. Istambul, Turkey, pp 1-8.

Costa, R. et al, 2012. An Analytical Data Management as a Cloud Service for Numerical Simulations. *In Proceedings of the VI Brazilian eScience Workshop*. Paraná, Brazil.

Grande. A. et al, 2010. Educational Computer Simulations for Visualizing and Understanding the Interaction of Electromagnetic Waves with Metamaterials. *In Proceedings of IEEE EDUCON Education Engineering*. Madrid, Spain, pp 543-547.

Jorissen, K. et al, 2012. A high performance scientific cloud computing environment for materials simulations. *Computer Physics Communications*, Vol. 183, No. 9, pp 1911–1919.

LAPACK – Linear Algebra PACKage. Available at http://www.netlib.org/lapack. Retrieved on 29/04/2013.

Moghe, U. et al, 2012. Cloud computing: Survey of different utilization techniques. *In Proceedings of the 2012 CSI Sixth International Conference on Software Engineering (CONSEG)*. Indore, India, pp 1-4.

Perus, I. et al, 2013. A web-based methodology for the prediction of approximate IDA curves. *Earthquake Engineering & Structural Dynamics*, Vol. 42, pp. 43-60.

Reimer, A. S. and Cheviakov, A. F., 2013. A Matlab-based finite-difference solver for the Poisson problem with mixed Dirichlet-Neumann boundary conditions. *Computer Physics Communications*, Vol. 184, No. 3, pp 783–798.

Russell, M. B. and Probert, S.D., 2004. FDiff3: a finite-difference solver for facilitating understanding of heat conduction and numerical analysis. *Applied Energy*, Vol. 79, No. 4, pp 443-456.

Villegas-Puyod, J.B., 2012. Cost effective cloud computing for real-time applications, *In Proceedings of the 2012 10th International Conference on ICT and Knowledge Engineering*. Bangkok, Thailand, pp 171-174.

Yang, L. and Sen, M. K., 2009. A new time–space domain high-order finite-difference method for the acoustic wave equation. *Journal of Computational Physics*, Vol. 228, No. 23, pp 8779-8806.

Xiao-ping, C. and Ru-fu, H. 2010. Web Services Based Mechatronics Product Remote Finite Element Analysis Techniques. *In Proceedings of the 2010 International Conference On Computer Design And Appliations (ICCDA)*. Qinhuangdao, China.

Web Technologies for Differential Equations Solving by Finite Difference Method with Mesh Visualization Independent on Device Performance

Fabio T. Matsunaga, Jose L. V. Boas, Neyva M. L. Romeiro and Jacques D. Brancher
State University of Londrina, Londrina-PR, Brazil. P.O. 10011
Email: ftakematsu@gmail.com, joseluizvilasboas@gmail.com, nromeiro@uel.br, jacques@uel.br

***Abstract*—Finite difference discretization is considered straightforward operations that use linear algebra operations, requiring devices with high-cost computational resources. The aim of this work is to use web technologies to develop a web service to enable complex simulations of 1D and 2D problems involving partial differential equations, by finite difference method. The system consists of a front-end and back-end client-server communication, a database of results and user/request storage and a graphical user interface for outputs visualization and interaction. Some contributions of the work are: practical learning tool for math and physics teaching, a collaborative system of problem sharing and a service which can be executed devices independent on their performance and platform.**

I. Introduction

Simulations of physical phenomena, such as heat transfer, wave propagation, advection-diffusion phenomena requires numerical methods, such as partial differential equations (PDEs), which are considered a straightforward operation, especially when high level of accuracy is considered [1]. This generates refined mesh and requires frequently high cost linear algebra operations, such as linear systems resolution and sparse matrix-vector multiplication. Furthermore, partial differential equations are difficult for conventional means in classrooms, because of the extensive theory and experimental knowledge. Considering that, computational and multimedia technologies, such as video recordings and interactive tools have been used in some works [2].

In addition, numerical simulations systems have used web-based technologies, which are becoming a technological trend. They are application programming interfaces (APIs), which are accessed over a network and executed on a remote server hosting a determined requested service. Web services allows simulations, which is composed by input users, model formation algorithms and procedures for output information visualization [3]. Some web tools, e.g. Adobe Flash, have many advantages compared to the traditional education method [4]. However, these web services and desktop applications still require a large computer memory, an ideal hardware configuration and local softwares installations to solve issues of ideal design or tools [5] and low responses [6].

Considering the mentioned above, many Internet services were developed to support the demand required by numerical simulations, such as an our previous work about a cloud architecture model called CloudPDE to heat transfer and wave propagation simulation [7]. However many of these systems can be improved to solve mathematical and physical learning/teaching issues of design responses, high-performance execution and the solution of any kind of PDE. Then, this paper presents the development of a web service architecture to enable 1D and 2D physical problems resolution involving stationary or time-dependent PDEs, such as Poisson, heat, and transport (advection-diffusion) equation, using finite difference discretization method. For this purpose, the web service will process all the calculations in remote server independent on device performance and operating systems, in order to help the teaching and learning of physics phenomena and the interaction between surfaces and boundary conditions.

II. Web service architecture and modules

The proposed web service is composed of three layers: a front-end of client-server communication and interaction, a back-end that executes the applications of each requester in a remote server and a user interface to results exhibitions in runtime. The main modules presented on the web service are described on the Figure 1. To develop this, the Primefaces were used to user graphical interface and the JSF (JavaServer Faces) framework were used to control sessions, develop the MVC (Model View Controller) web applications and connect it with the interface. On the server back-end, the high-cost calculations on large and accurate problems are performed by high performance linear algebra algorithms of LAPACK (Linear Algebra PACKage) library which use the advantages of shared memories and parallel processors technologies.

Another functionality contained in the web service is the expression tree interpreter of boundary and initial conditional functions. From this expression tree, a pre-order router is performed to calculate the numerical values of the expression, allowing the parametrization of different boundary conditions applied on real problems, such as Dirichlet conditions (constants represented by a single node tree), simple or complex mathematical functions (Neumann conditions) and conditional functions. For the 1D (bars) and 2D (surface) problems visualization, we used chain colors of 15 shades for positive and negative values constructed in a HTML table, representing an interval value (example on Figure 2A). The 1D and 2D results exhibition is performed in runtime, that is, on each time instance a continuous pooling call is made to the remote server. After that, the user can access some numerical analysis to validate the simulations, such as convergence test and relative error compared with analytic solutions.

The values distributions in a data table is updated during the remote calculations, creating an animated visualization of phenomena action which, time delay can be determined and modified by the user. Furthermore, on the generated mesh table, the users can interact with the service by extracting all meta-information outputs, such as influential boundary conditions and on 2D case the user can extract pieces of discretized mesh (an example of x line extraction on Figure 2B), because all of mesh information are stored on database in a matrix format. Despite the simple way of graphical visualization, the results visualization of the discretized mesh can be performed on any device that has an Internet browser with HTML or Java technology, such as personal computers and mobile devices.

All of these operations are performed on a remote server, which is responsible for the simultaneous users access management, database storage of all users – name, institution, login, and password – and respective calculations requests – date, time, input parameters (time/space domain, boundary/initial conditions, maximum time simulation and spatial/temporal partition size) – information, and all the calculated information are written on database and passed to the client. This database stores all PDEs solutions previously calculated where the user can access the problems and solutions in a specified period by database query with the date/time request, creating a collaborative system allowing teachers to access or evaluate the simulations performed by students, and even others institutions. The query option can be made by numerical values extractions, which format is set of vectors (1D problems) and matrixes (2D problems) accessible from others applications such as Matlab and Scilab, or a CSV format accessible by spreadsheets applications. This systems enable a cache service, where a problem requested is not solved again if that solution and others similar solutions with little variation is already stored on the result table.

III. Conclusion and Future Works

The web service architecture presents the property of remote equation processing allowing a wide class of client hardware to be employed in complex physical simulations and the architecture and visual interface is adequate and user-friendly for academic teaching. These characteristics become the system very useful for the learning of scientific problems useful on mathematics and physics teaching and individual learning. For more accurate simulations, the discretization method requires high-order matrixes computations, however these are performed on a remote server communicated with the user interface module. Combining the client-server communication with the mesh visualization methodology using HTML tables for both 1D and 2D problems, the results visualizations and user interactions can be executed on devices which not need to have high performance. The model visualization of the results allows the graphical visualization of problems with finite differences (whatever model discretization used), whereas the outputs are stored in a standardized way in the database. In a future perspective, the service is independent on operating system and requires only a web browser with HTML and Java support. However, the design of results visualization can be improved considering the integration of algorithms and libraries of mesh generation, mainly in 3D equations. Furthermore, the web service developed allows the easy inclusion of others PDEs and use the implemented finite difference method on generalized coordinates systems, which describes the geometry of many real objects.

References

[1] P. K. Gupta, J. Singh, K. Rai, and S. Rai, "Solution of the heat transfer problem in tissues during hyperthermia by finite difference-decomposition method," *Applied Mathematics and Computation*, vol. 219, no. 12, pp. 6882–6892, Feb. 2013.

[2] A. Grande, O. Gonzalez, J. A. Pereda, and A. Vegas, "Educational Computer Simulations for Visualizing and Understanding the Interaction of Electromagnetic Waves with Metamaterials ." in *IEEE EDUCON Education Engineering*, no. 1, 2010, pp. 543–547.

[3] I. Perus, R. Klinc, M. Dolenc, and M. Dolsek, "A web-based methodology for the prediction of approximate IDA curves," *Earthquake Engineering & Structural Dynamics*, vol. 42, no. April 2012, pp. 43–60, 2013.

[4] H. H. Basak; and K. Ozansoy, "Comparison of Traditional over WEB-Based Education: Case Study "Adobe Flash"," in *2012 International Conference on Information Technology Based Higher Education and Training (ITHET)*, 2012, pp. 1–8.

[5] C. Xiao-ping and H. Ru-fu, "Web Services Based Mechatronics Product Remote Finite Element Analysis Techniques," in *2010 International Conference On Computer Design And Appliations (ICCDA)*, vol. 4, no. Iccda, 2010.

[6] S. Bennett, A. Bishop, B. Dalgarno, J. Waycott, and G. Kennedy, "Implementing Web 2.0 technologies in higher education: A collective case study," *Computers & Education*, vol. 59, no. 2, pp. 524–534, Sep. 2012.

[7] F. T. Matsunaga, J. L. Vilas Boas, N. M. L. Romeiro, and J. D. Brancher, "CloudPDE: a web solver of differential equations by finite difference discretization," in *IADIS - International Conference on Applied Computing (prelo)*, 2013.

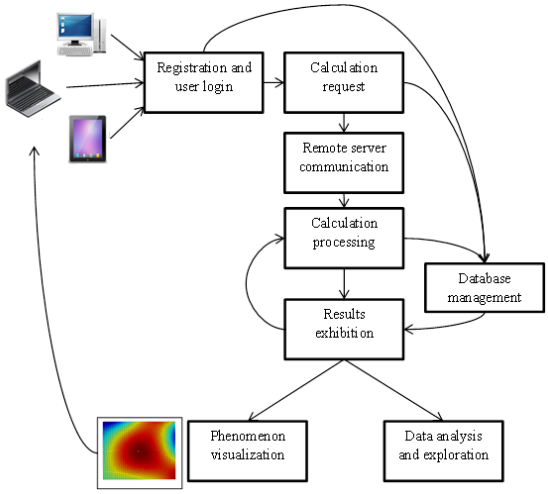


Fig. 1. Main modules of the web service

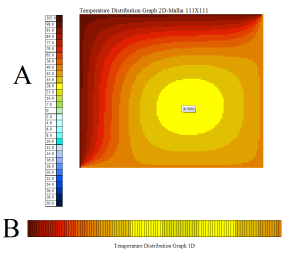


Fig. 2. Example of a mesh visualization of 2D heat distribution problem

Demais trabalhos publicados pelo autor

1. Matsunaga, F. T. ; Vilas Boas, J. L. ; Romeiro, N. M. L. ; Brancher, J. D. **Web Technologies for Differential Equations Solving by Finite Difference Method with Mesh Visualization Independent on Device Performance**. *In*: International Conference on Chilean Computer Society, 2013, Temuco, Chile. Jornadas Chilenas de Computación (*prelo*), 2013.

2. Matsunaga, F. T. ; Vilas Boas, J. L. ; Romeiro, N. M. L. ; Brancher, J. D. **CloudPDE: a web solver of differential equations by finite difference discretization**. *In*: IADIS - International Conference on Applied Computing, 2013, Texas, EUA. International Conference on Applied Computing (*prelo*), 2013.

3. Costa, E. B. B. ; Matsunaga, F. T. ; Brancher, J. D. **Análise de escalabilidade e eficiência da fatoração LU usando CPU x GPU**. *In*: XII Workshop em Desempenho de Sistemas Computacionais e de Comunicação, 2013, Maceío-AL. Anais do XXXIII Congresso da Sociedade Brasileira de Computação, 2013.

4. Matsunaga, F. T. ; Rakocevic, M. ; Brancher, J. D. **Modeling the 3D structure and rhythmic growth responses to environment in dioecious yerba-mate**. Ecological Modelling (*prelo*), 2013. DOI: `http://dx.doi.org/10.1016/j.ecolmodel.2013.10.035`

5. Matsunaga, F. T. ; Rakocevic, M. ; Brancher, J. D. **InterpolMateS1- the Module for Interpolation of Growth and Production of Yerba-Mate**. IEEE 4th International Symposium on Plant Growth Modeling, Simulation, Visualization and Applications, 2012.

6. Matsunaga, F. T. ; Rakocevic, M. ; Brancher, J. D. **Artificial neural networks in modeling of environmental time series for yerba-mate growth dynamics**. Functional-Structural Plant Modelling (FSPM), 2013.