Simulações Numéricas 3D em Ambiente Paralelo de Hipertermia com Nanopartículas Magnéticas

Dissertação apresentada ao Programa de Pós-graduação em Modelagem Computacional, da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do grau de Mestre em Modelagem Computacional.

Orientador: Prof. D.Sc. Felipe dos Santos Loureiro

Coorientador: Prof. D.Sc. Marcelo Lobosco

Juiz de Fora

2014

Ficha catalográfica elaborada através do Programa de geração
automática da Biblioteca Universitária da UFJF,
com os dados fornecidos pelo(a) autor(a)

Reis, Ruy Freitas .
Simulações Numéricas 3D em Ambiente Paralelo de Hipertermia com Nanopartículas Magnéticas / Ruy Freitas Reis. -- 2014.
104 p. : il.

Orientador: Felipe dos Santos Loureiro
Coorientador: Marcelo Lobosco
Dissertação (mestrado acadêmico) - Universidade Federal de Juiz de Fora, ICE/Engenharia. Programa de Pós-Graduação em Modelagem Computacional, 2014.

1. Hipertermia. 2. Nanopartículas Magnéticas. 3. Computação de Alto Desempenho. 4. OpenMP. 5. CUDA. I. Loureiro, Felipe dos Santos , orient. II. Lobosco, Marcelo, coorient. III. Título.

Ruy Freitas Reis

Simulações Numéricas 3D em Ambiente Paralelo de Hipertermia com Nanopartículas Magnéticas

Dissertação apresentada ao Programa de Pós-graduação em Modelagem Computacional, da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do grau de Mestre em Modelagem Computacional.

Aprovada em 5 de novembro de 2014.

BANCA EXAMINADORA

Prof. D.Sc. Felipe dos Santos Loureiro - Orientador
Universidade Federal de Juiz de Fora

Prof. D.Sc. Marcelo Lobosco - Coorientador
Universidade Federal de Juiz de Fora

Prof. Ph.D. Webe João Mansur
Universidade Federal do Rio de Janeiro

Prof. D.Sc. Rafael Sachetto Oliveira
Universidade Federal de São João del-Rei

Prof. D.Sc. Bernardo Martins Rocha
Universidade Federal de Juiz de Fora

Dedico este trabalho primeiramente a Deus, que me possibilitou a vida e consequências dela. A minha família que nunca deixou de me amparar nesse processo evolutivo, e a todos que me acompanharam durante essa caminhada.

AGRADECIMENTOS

Agradeço, primeiramente a Deus, inteligência suprema, causa primária de todas as coisas, que me permitiu nascer, viver e me conduziu para o que sou hoje.

Agradeço, também, à minha querida família, que me dedicou amor incondicional, desde o princípio de minha existência. Lane, mãezinha querida, por ser essa pessoa maravilhosa que, em momento algum me deixou desamparado, mesmo de longe sempre me acolhendo de braços abertos e me amando de maneira sublime. José Robson, pai querido, que mesmo não entendendo direito o que faço, sempre acreditou em meu potencial, e nunca deixou de me apoiar nesta caminhada. Meu querido irmão, Robson Vitor, o qual sempre me serviu de inspiração nesta caminhada, uma pessoa maravilhosa que, com seu amplo conhecimento, me inspira a ir além. Minha querida irmã, Nágila Analy, também por todo o carinho e apoio doado durante toda minha existência.

Minha querida namorada, Juliana, por todo o carinho dedicado desde antes de entrar na faculdade, me mostrando sempre o lado bom da vida e trazendo seu doce amor para meu coração. Cada palavra sua, dita durante o tempo que nos conhecemos, contribuiu para o que sou hoje.

Há também os amigos, um presente da vida. Gostaria de agradecer, em especial, aqueles que me acompanharam nesta caminhada: Aos irmãos Guilherme e Rafael e o amigo Leandro, irmãos que conheci, os quais nunca me negaram apoio, mesmo nas horas mais difíceis; aos amigos do laboratório: Ana Amélia, Anna Cláudia, Carla, Daniele, Evelyn, Gustavo, João, Janaína, Johnny, José Humberto, Joventino, Lucas, Marcos, Rodrigo (Lemão) e Sabrina, pelos bons momentos compartilhados no "Prédio Azul", e fora também! Aos amigos de república, João Paulo e Adnan.

Aos professores do Programa de Pós-Graduação em Modelagem Computacional, que contribuíram pra minha formação. Em especial aos orientadores, Felipe, que me confiou e possibilitou a criação deste projeto, além do inegável apoio durante esta caminhada que se iniciou durante a graduação e Marcelo Lobosco, que acompanha desde o segundo período da faculdade, sempre me apoiando profissionalmente, além de sábios conselhos doados. E, a todos os funcionários que contribuíram para o funcionamento do programa.

Por fim, a todas as instituições de fomento que contribuíram para a conclusão deste projeto: FAPEMIG, CAPES, UFJF e CNPq.

*"**O que é sucesso?***

Rir muito e com frequência; ganhar o respeito de pessoas inteligentes e o afeto das crianças; merecer a consideração de críticos honestos e suportar a traição de falsos amigos; apreciar a beleza, encontrar o melhor nos outros; deixar o mundo um pouco melhor, seja por uma saudável criança, um canteiro de jardim ou uma redimida condição social; saber que ao menos uma vida respirou mais fácil porque você viveu

Isto é ter sucesso!"

(Ralph Waldo Emerson)

RESUMO

Este estudo tem como objetivo a modelagem numérica do tratamento de tumores sólidos com hipertermia utilizando nanopartículas magnéticas, considerando o modelo tridimensional de biotransferência de calor proposto por Pennes (1948). Foram comparadas duas diferentes possibilidades de perfusão sanguínea, a primeira constante e, a segunda, dependente da temperatura. O tecido é modelado com as camadas de pele, gordura e músculo, além do tumor. Para encontrar a solução aproximada do modelo foi aplicado o método das diferenças finitas (MDF) em um meio heterogêneo. Devido aos diferentes parâmetros de perfusão, foram obtidos sistemas de equações lineares (perfusão constante) e não lineares (perfusão dependente da temperatura). No domínio do tempo foram utilizados dois esquemas numéricos explícitos, o primeiro utilizando o método clássico de Euler e o segundo um algoritmo do tipo preditor-corretor adaptado dos métodos de integração generalizada da família-alpha trapezoidal. Uma vez que a execução de um modelo tridimensional demanda um alto custo computacional, foram empregados dois esquemas de paralelização do método numérico, o primeiro baseado na API de programação paralela *OpenMP* e o segundo com a plataforma CUDA. Os resultados experimentais mostraram que a paralelização em *OpenMP* obteve aceleração de até 39 vezes comparada com a versão serial, e, além disto, a versão em CUDA também foi eficiente, obtendo um ganho de 242 vezes, também comparando-se com o tempo de execução sequencial. Assim, o resultado da execução é obtido cerca de duas vezes mais rápido do que o fenômeno biológico.

Palavras-chave: Nanopartículas. Hipertermia. Biotransferência de Calor. Computação de Alto Desempenho. CUDA. *OpenMP*.

ABSTRACT

This work deals with the numerical modeling of solid tumor treatments with hyperthermia using magnetic nanoparticles considering a 3D bioheat transfer model proposed by Pennes(1948). Two different possibilities of blood perfusion were compared, the first assumes a constant value, and the second one a temperature-dependent function. The living tissue was modeled with skin, fat and muscle layers, in addition to the tumor. The model solution was approximated with the finite difference method (FDM) in an heterogeneous medium. Due to different blood perfusion parameters, a system of linear equations (constant perfusion), and a system of nonlinear equations (temperature-dependent perfusion) were obtained. To discretize the time domain, two explicit numerical strategies were used, the first one was using the classical Euler method, and the second one a predictor-corrector algorithm originated from the generalized trapezoidal alpha-family of time integration methods. Since the computational time required to solve a three-dimensional model is large, two different parallel strategies were applied to the numerical method. The first one uses the OpenMP parallel programming API, and the second one the CUDA platform. The experimental results showed that the parallelization using OpenMP improves the performance up to 39 times faster than the sequential execution time, and the CUDA version was also efficient, yielding gains up to 242 times faster than the sequential execution time. Thus, this result ensures an execution time twice faster than the biological phenomenon.

Keywords: Nanoparticles. Hyperthermia. Bioheating. High performace Computation. CUDA. OpenMP.

SUMÁRIO

1 INTRODUÇÃO **16**

1.1 Motivação **16**

1.2 Objetivos **18**

1.3 Organização **19**

2 MODELO MATEMÁTICO **20**

2.1 Formulação Diferencial da Condução de Calor **20**

2.2 Biotransferência de Calor em Tecido Vivo **24**

2.2.1 Simplificações Assumidas **24**

2.2.2 Formulação da Equação de Pennes **25**

2.3 Hipertermia com Nanopartículas Magnéticas **27**

2.3.1 Geração de Calor pelas Nanopartículas **27**

2.3.2 Aplicador do Campo Magnético **30**

2.3.3 Modelos de Fonte de Calor **30**

2.4 Perfusão Sanguínea **32**

2.4.1 Fisiologia da Arquitetura Vascular **32**

2.4.2 Modelos de Perfusão Sanguínea **36**

2.5 Formalização do Modelo Matemático **38**

3 MÉTODO NUMÉRICO **40**

3.1 A Discretização no Espaço **40**

3.1.1 Condutividade Térmica em Meio Heterogêneo **40**

3.1.2 Método das Diferenças Finitas em Meio Heterogêneo **42**

3.1.3 Modelo Discretizado **44**

3.1.4 Condições de Contorno **44**

3.1.4.1 Condições de Contorno de Dirichlet **45**

3.1.4.2 Condições de Contorno de Neumann **45**

3.2 A Discretização no Tempo **46**

3.2.1 Aplicação da Diferença Progressiva **47**

3.2.2 Aplicação do Preditor-Corretor **47**

3.3 Estabilidade do Método Numérico 50

4 A ESTRATÉGIA PARALELA 51
4.1 A API *OpenMP* 53
4.1.1 Um primeiro exemplo OpenMP 55
4.1.2 Paralelização do Método Numérico 56
4.2 A Plataforma NVIDIA CUDA 58
4.2.1 A extensão CUDA C 59
4.2.2 A arquitetura de uma GPU da geração NVIDIA Fermi 61
4.2.3 Um primeiro exemplo CUDA 63
4.2.4 Paralelização do Método Numérico 64

5 RESULTADOS 67
5.1 Estudo 1 67
5.1.1 Parâmetros do Modelo 68
5.1.2 Simulações do Estudo 69
5.1.3 Paralelização do Método 70
5.2 Estudo 2 73
5.2.1 Parâmetros do Modelo 73
5.2.2 Simulações do Estudo 74
5.2.3 Paralelização do Método 76
5.3 Estudo 3 78
5.3.1 Parâmetros do Modelo 79
5.3.2 Simulações do Estudo 80
5.3.3 Paralelização do Método 84
5.3.4 Análise do Erro no Método Preditor-Corretor 88

6 CONCLUSÕES 89
6.1 Trabalhos Futuros 91

APÊNDICES 99

LISTA DE ILUSTRAÇÕES

2.1 Elemento infinitesimal. Extraído de Jiji (2009). 20
2.2 Fluxos nos elementos infinitesimais no plano 2D. Extraído de Jiji (2009). . . . 22
2.3 Capilares. Adaptado de Jiji (2009) 26
2.4 Comportamento magnético e mecanismos de geração de calor. Adaptado de Moros (2012, p.297) 28
2.5 Exemplos de ciclos de histerese, demonstrando diferentes ciclos baseados no material (a) e intensidade do campo aplicado (b). Adaptado de Hergt *et al.* (1998). 29
2.6 Este esboço mostra o primeiro protótipo de um aplicador de campo magnético alternado para uso médico, o MFH 300F. Adaptado de Gneveckow *et al.* (2004, p.1445) 30
2.7 Evolução da temperatura aferida nos membros posteriores do rato. A seta indica a chegada ao estado estacionário. Adaptado de Salloum *et al.* (2008, p.593) 31
2.8 Imagens da velocidade do fluxo sanguíneo antes e após o transplante de células cancerígenas em rato: a) antes do transplante e b) dois, c) cinco, d) nove, e) doze, e f)dezesseis dias após o transplante. Adaptado de Ishida *et al.* (2012). 34
2.9 Mudança no fluxo sanguíneo em ratos aquecidos com banho de água em diferentes temperaturas. Adaptado de Song *et al.* (1984). 35
2.10 Mudança relativa do fluxo sanguíneo em pele, músculo e tumor em animais após aquecimento de 30 à 40 minutos. Adaptado de Song *et al.* (1984) . . . 36
2.11 Perfusão sanguínea dependente da temperatura: a)músculo; b)gordura; c)tumor. 39

3.1 Discretização do domínio tridimensional. 41
3.2 Pontos i e $i+1$ em um meio heterogêneo com o ponto fictício $i+\frac{1}{2}$. 41
3.3 Meio heterogêneo com os pontos fictícios $i+\frac{1}{2}$ e $i-\frac{1}{2}$. 43
3.4 *Stencil* no espaço do esquema numérico tridimensional de 7 pontos. 44
3.5 Domínio discretizado com o ponto fantasma. 46
3.6 *Stencil* e primeira ordem no tempo. 47

4.1 Evolução do número de transistores nos computadores de 1971 até 2011. Adaptado de TechnologyPod (2014). 52
4.2 Sistema de memória compartilhada. Adaptado de Pacheco (2011, p. 33). . . . 53
4.3 Sistema de memória compartilhada multinúcleo. Adaptado de Pacheco (2011, p. 34). 54
4.4 Processo *multithread*. Adaptado de Pacheco (2011, p. 18). 54
4.5 Número de operações com ponto flutuante comaprando CPU e GPU. Adaptado de NVIDIA (2014c). 59
4.6 Banda de memória comparando CPU e GPU. Adaptado de NVIDIA (2014c). . 59
4.7 Estrutura da CPU e GPU. Adaptado de NVIDIA (2014c). 60
4.8 Organização das *threads* em blocos de grades com a plataforma CUDA. 61
4.9 *Datasheet* de uma GPU da arquitetura Fermi. Adaptado de Wittenbrink *et al.* (2011). 62

5.1 Domínio da simulação do estudo da seção 5.1. Destaque para o tumor localizado no centro do cubo. 68
5.2 Solução do modelo (2.23) no tempo $t = 3.000$s. 69
5.3 Solução da equação (2.23) no tempo $t = 3.000$s em uma linha no eixo x. 70
5.4 Solução da Eq. (2.23) em um ponto no tecido saudável $(0,050m; 0,076m; 0,026m)$ e no tecido tumoral $(0,050m; 0,0500m; 0,050m)$ através do tempo. 71
5.5 Compartilhamento de recursos do Chip AMD Opteron Série 6200 Interlagos. Adaptado de NumericalAlgorithmsGroup (2011). 72
5.6 Aceleração do tempo de execução do código utilizando *OpenMP* em uma máquina *multicore*. 72
5.7 Domínio da simulação do estudo da seção 5.2. Destaque para o tumor localizado no centro do cubo. 74
5.8 Solução do modelo no tempo $t = 3.000s$ mostrando o volume de tecido morto no domínio. Estudo da seção 5.2 75
5.9 Gráficos da solução da equação transiente no instante $t = 3000$s. Estudo da seção 5.2. 76

5.10 Solução transiente da modelo em um ponto do domínio com tecido saudável (0,06m;0,06m;0,06m) e tumoral (0,05m;0,05m;0,05m). Estudo da seção 5.2. 77
5.11 Solução da equação transiente no instante $t = 3000$s em $x = 0,05$m e $z = 0,05$m através do eixo y. Estudo da seção 5.2. 77
5.12 Aceleração do tempo de execução do código utilizando *OpenMP* em uma máquina *multicore*. Estudo da seção 5.2. 78
5.13 Domínio do tecido vivo com um tumor no estudo da seção 5.3. 79
5.14 Solução a equação estacionária na superfície $x = 0,05$. 81
5.15 Solução da equação estacionária no plano xz em $y = 0,025$. 82
5.16 Solução da equação estacionária, comparando os modelos linear (ω_b constante) e não linear (ω_b em função da temperatura), sobre o eixo x em $y = 0,05$ e $z = 0,05$. 83
5.17 Comparativo da solução da equação transiente entre os modelos linear (ω_b constante) e não linear (ω_b em função da temperatura). Mostra a evolução no tempo no tecido tumoral em $x = 0,025$, $y = 0,05$ e $z = 0,05$, e saudável em $x = 0,025$, $y = 0,065$ e $z = 0,065$. 83
5.18 Temperatura ao longo do eixo y em $t = 3.000s$, $x = 0,025$ e $z = 0,05$, comparando os modelos linear (ω_b constante) e não linear (ω_b em função da temperatura). 84
5.19 Solução da equação transiente em $t = 3000$s em alguns planos zy. 85
5.20 Impacto da variação do tamanho do bloco. Adaptado da *Cuda Occupancy Calculator* (NVIDIA, 2014a). 86
5.21 Comparativo entre a primeira e a segunda iteração do erro relativo no método preditor-corretor durante a simulação. 88

LISTA DE TABELAS

3.1 Métodos membros da família trapezoidal generalizada. 48

5.1 Parâmetros do modelo no estudo 1. 69
5.2 Parâmetros dos pontos de injeção no estudo 1. 69
5.3 Parâmetros do modelo no estudo 2. 74
5.4 Parâmetros dos pontos de injeção no estudo 2. 74
5.5 Parâmetros do modelo no estudo da seção 5.3. 80
5.6 Parâmetros dos pontos de injeção no estudo da seção 5.3. 80
5.7 Tempo de execução e aceleração dos primeiros 250s segundos de simulação de tratamento de hipertermia. 87
5.8 Tempo de execução e aceleração da simulação total de um tratamento de hipertermia. 87

A.1 Fórmulas de Diferenças Finitas 100

LISTA DE ABRAVIAÇÕES E SIGLAS

API	*Application Programming Interface*
CPU	*Central Processing Unit*
CUDA	*Compute Unified Device Architecture*
DCC	Departamento de Ciência da Computação
FP	*Float Point*
FPU	*Float Point Unit*
GPU	*Graphic Process Unit*
Int	*Integer*
MDF	Método das Diferenças Finitas
MEC	Método dos Elementos de Contorno
MEF	Método dos Elementos Finitos
MVF	Método dos Volumes Finitos
MPI	*Message Passing Interface*
NUMA	*Non-Uniform Memory Access*
OpenMP	*Open Multi-Processing*
RAM	*Random Access Memory*
SAR	*Specific Absorption Rate*
UFJF	Universidade Federal de Juiz de Fora
UMA	*Uniform Memory Access*

1 INTRODUÇÃO

1.1 Motivação

Segundo a Organização Mundial da Saúde (OMS, 2014) o câncer é a segunda doença que mais mata no mundo. A título de exemplo pode-se citar que em 2008 quase 14%, ou 7,6 milhões de mortes, foram causadas por ele. Existem mais de 100 tipos de câncer, e, além disto, pode-se desenvolver câncer em mais de 60 órgãos. Dentre todas as possibilidades os mais comuns são: pulmão ($12,7\%$), mama ($10,9\%$), colorretal ($9,8\%$), estômago ($7,8\%$) e próstata ($7,1\%$). De 2008 para 2012 a mortalidade causada por câncer no mundo cresceu 8%, ou seja, passou de $7,6$ milhões para $8,2$ milhões de óbitos anuais. Além disso estima-se que novos casos de câncer no período de 2007 à 2030 possam saltar de $11,3$ milhões para $15,5$ milhões.

Nos últimos anos vem sendo desenvolvidos vários métodos para tentar contornar este problema de saúde mundial. Dentre as principais técnicas de tratamento que atualmente existem, pode-se citar a quimioterapia e a radioterapia. Além disto, existem formas mais agressivas de ataque a esta doença, por exemplo, a retirada cirúrgica do tecido tumoral (Nathan, 2007).

A hipertermia foi descoberta na década de 50 (Gilchrist *et al.*, 1957), e desde então, vem se desenvolvendo tratamentos de câncer através desta técnica. A hipertermia é baseada em superaquecer o tecido até um limiar de temperatura que causa a necrose celular. Assim, pode-se destruir as células tumorais sem a necessidade de um processo cirúrgico. Atualmente, a hipertermia é muito utilizada como terapia não invasiva contra o câncer, ajudando outras técnicas, como a quimioterapia e a radioterapia. Dentre as maneiras de realizar o tratamento através de hipertermia, pode-se destacar a baseada na aplicação de nanopartículas magnéticas.

Apesar da hipertermia com nanopartículas magnéticas ter sido descoberta nos anos 50 (Gilchrist *et al.*, 1957), ela ainda está em estágio inicial de desenvolvimento para

tratamento em câncer. As nanopartículas podem ser aplicadas em tecidos tumorais irregulares e/ou profundos e quando expostas a um campo magnético alternado de baixa frequência superaquecem o local. O calor produzido por estas partículas é gerado, principalmente, por relaxação de Néelian e/ou Browniano (Rosensweig, 2002).

Esta abordagem é muito utilizada no tratamento de tumores do fígado (Matsuki *et al.*, 1994) e mama (Hilger *et al.*, 2005), o segundo tipo de câncer mais comum segundo a OMS (2014). Há, também, estudos clínicos realizados por Johannsen *et al.* (2005) que abordam testes de hipertermia com fluido magnético no tratamento do câncer de próstata. Análises histológicas dos tecidos cancerígenos mostram uma necrose parcial das células após este tipo de tratamento. Para que o tratamento tenha sucesso, deve-se elevar a temperatura acima de 43^oC no tumor inteiro, provocando assim a necrose celular, com o mínimo de dano às células da vizinhança saudável. O estudo feito por Kawai *et al.* (2005) mostra ainda que tratamentos repetidos com hipertermia matam células com câncer não somente por aquecimento, mas também por resposta imune induzida.

O modelo mais comum utilizado para equacionar a biotransferência de calor foi proposto por Pennes (1948). Este ficou bem conhecido pois obtém-se uma boa aproximação do fenômeno de transferência de calor em tecido vivo e, além disso, desde sua proposição até os dias atuais é amplamente utilizado para fazer simulações numéricas de estudos baseados em biotransferência de calor em tecidos vivos, como os estudos feitos por Salloum *et al.* (2009); Moroz *et al.* (2002); Liu e Xu (2000); Cao *et al.* (2010); Karaa *et al.* (2005) dentre outros.

Um dos parâmetros deste modelo é o calor gerado e, no caso do tratamento utilizando hipertermia, pode-se aproximar como o calor é absorvido pelo corpo. Para estimar a geração de calor causada pelo tratamento com hipertermia utiliza-se uma denominada taxa de absorção específica, ou SAR (do inglês *specific absorption rate*). A maioria dos estudos sobre o valor desta taxa é em termos de W/kg, obtidos por estudos *in vivo* ou *in vitro* utilizando tecidos vivos. Na literatura existem várias formas de representar esta geração de calor, mas pode-se destacar um modelo dependente do espaço e um modelo que depende do espaço e tempo, respectivamente propostos por Salloum *et al.* (2008) e

Mital e Tafreshi (2012).

Sempre procurando aproximar o modelo do fenômeno físico, é de interesse a simulação em domínios tridimensionais, permitindo assim a simulação de domínios com formatos não simétricos ou irregulares, como o caso de um tumor. Contudo, a resolução de modelos matemáticos de equações diferenciais parciais em domínios tridimensionais demanda grande tempo computacional, deixando a resolução destes modelos extremamente custosa. Visando diminuir o tempo para encontrar a solução do modelo, pode-se utilizar técnicas de computação paralela, seja empregando para este fim múltiplos processadores ou até mesmo com o uso de placas gráficas.

1.2 Objetivos

Este trabalho irá simular a utilização de nanopartículas magnéticas no tratamento de tumores sólidos, através de um modelo matemático de biotransferência de calor. Contudo, conforme dito anteriormente, a resolução de modelos matemáticos tridimensionais, baseados em equações diferenciais demandam longo tempo computacional, tornando necessária a utilização de técnicas de computação paralela para acelerar o tempo de simulação.

Para simular o tratamento de câncer por meio de hipertermia com nanopartículas magnéticas previamente descrito, foi implementado o modelo matemático proposto por Pennes (1948), em um domínio tridimensional, considerando as devidas adaptações necessárias. Nas simulações serão considerados cenários com tumor e músculo, e um cenário mais completo com tumor, músculo, gordura e pele.

Este trabalho analisa também a utilização dos modelo de SAR dependente somente do espaço e dependente do tempo e do espaço.

Para encontrar a solução do modelo matemático, é necessária a utilização de um método numérico que seja capaz de aproximar a solução da equação diferencial. Este trabalho irá comparar a utilização de duas técnicas explícitas para encontrar a solução numérica deste modelo, uma de ordem de convergência linear e outra quadrática no tempo, respectivamente o método de Euler e um método preditor-corretor. Em ambos os casos,

no espaço, adota-se diferença centrada, *i.e.*, um método de segunda ordem.

Em um estudo sobre hipertermia com nanopartículas magnéticas em um domínio bidimensional, foi obtida uma aceleração do tempo de execução em até 35 vezes, em uma máquina de 64 *cores* (Reis *et al.*, 2014a). Assim, visando acelerar o tempo de execução da simulação, serão utilizadas técnicas de computação paralela *multithreaded*, ou seja, baseadas na utilização de múltiplos fluxos de execução. Para a implementação na CPU foi utilizada a API de programação paralela *OpenMP* e para a implementação em GPU a plataforma de computação paralela *Compute Unified Device Architecture*, ou simplesmente CUDA, criada pela NVIDIA. Neste estudo optou-se pelas linguagens C/C++ para o desenvolvimento dos códigos, tanto para CUDA (Sanders e Kandrot, 2010; Kirk e Hwu, 2010) quanto para *OpenMP* (Pacheco, 2011).

1.3 Organização

Este trabalho foi disposto em 6 capítulos, incluindo a introdução. Os próximos três capítulos descrevem, em principal, material e métodos, bem como os conceitos fundamentais para a compreensão dos resultados obtidos. O Capítulo 2 descreve os conceitos físicos e biológicos que fundamentam os modelos matemáticos utilizados nas simulações numéricas realizadas neste trabalho. O Capítulo 3 descreve o método numérico utilizado para encontrar as soluções aproximadas dos modelos matemáticos descritos no Capítulo 2. O Capítulo 4 explica as estratégias paralelas abordadas neste trabalho, bem como alguns conceitos básicos de programação paralela para a melhor compreensão dos resultados obtidos. Após a apresentação dos modelos e técnicas aplicadas neste trabalho, o Capítulo 5 detalha os 3 estudos de casos que foram simulados neste trabalho, bem como faz a apresentação dos resultados obtidos em cada um destes estudos. Além disto, o Capítulo 5 apresenta os resultados da paralelização obtida, tanto com o emprego da API de programação paralela *OpenMP*, quanto da plataforma CUDA. Por fim, o Capítulo 6 discute o que se pode observar nos resultados do capítulo anterior.

2 MODELO MATEMÁTICO

"A peculiaridade da ciência física, com respeito a outras ciências da natureza, é que sistemas físicos tem muitos atributos quantitativos. Estes atributos quantitativos são responsáveis pela introdução de grandezas físicas que permitem a descrição quantitativa do fenômeno físico. Consequentemente o comportamento de um fenômeno, *i.e.*, sua lei, é descrita pelo relacionamento entre grandezas físicas pertinentes, *i.e.*, por uma equação."(Tonti, 2014, v.257, p.1261, tradução nossa)

2.1 Formulação Diferencial da Condução de Calor

O estudo da distribuição de temperatura em uma dada região tem grande importância não somente na área da saúde, mas também em inúmeras áreas do conhecimento, principalmente na engenharia, *e.g.*, na determinação de previsão do tempo (Vu *et al.*, 2013), no resfriamento de caldeiras e no aquecimento global (Busch *et al.*, 2012).

Para obter um modelo matemático que calcule a distribuição de temperatura em uma região qualquer, pode-se aplicar o princípio da conservação de energia em um certo volume de controle. Desta forma, tomando um elemento infinitesimal $dV = dxdydz$, conforme ilustrado na Fig. 2.1, e aplicando o princípio da conservação de energia (primeira lei da termodinâmica) conforme descrito por Jiji (2009), obtém-se a Eq. (2.1).

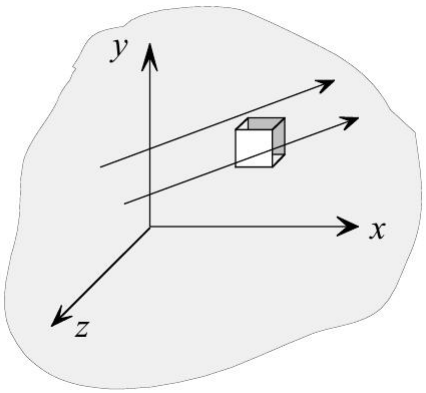


Figura 2.1: Elemento infinitesimal. Extraído de Jiji (2009).

$$\dot{E}_{entra} + \dot{E}_g - \dot{E}_{sai} = \dot{E}, \tag{2.1}$$

onde *Taxa de Energia Adicionada*, *Taxa de Energia Gerada*, *Taxa de Energia Removida* e *Taxa de Energia Alterada no Elemento* são $\dot{E}_{entra}$, $\dot{E}_g$, $\dot{E}_{sai}$ e $\dot{E}$, respectivamente. Este modelo considerará as seguintes simplificações:

1. Velocidade uniforme;
2. Pressão constante;
3. Densidade constante; e
4. Variação de energia potencial desprezível.

A energia é trocada com o elemento por condução e convecção. Os dois modos de transferência de energia são mostrados na Fig. 2.2. Nesta figura não foi considerada a coordenada z, a qual pode ser analogamente incorporada na formulação. A energia entra no elemento por condução com um fluxo q_x, q_y e q_z, nas direções dos vetores unitários $\hat{i}$, $\hat{j}$ e $\hat{k}$, respectivamente. Baseando-se no fato de que o fluxo representa a energia por unidade de área e por unidade de tempo, o fluxo deve ser multiplicado pela área normal a ele. A energia também entra no elemento através de fluxo de massa. A taxa do fluxo de massa que entra no elemento na direção do vetor $\hat{i}$ é $\rho U dydz$, onde ρ é a densidade e U a velocidade do componente na direção do vetor $\hat{i}$. A taxa de energia carregada pela massa é $\rho U \hat{h} dydz$, onde $\hat{h}$ é a entalpia por unidade de massa. As componentes correspondentes nas direções y e z são $\rho V \hat{h} dxdz$ e $\rho W \hat{h} dxdy$, onde V e W são as componentes da velocidade nas direções $\hat{j}$ e $\hat{k}$, respectivamente. Com isso pode-se expressar $\dot{E}_{entra}$ com a equação:

$$\dot{E}_{entra} = q_x dydz + q_y dxdz + q_z dxdy + \rho U \hat{h} dydz + \rho V \hat{h} dxdz + \rho W \hat{h} dxdy, \tag{2.2}$$

onde q_x, q_y e q_z são os fluxos por unidade de volume nas direções de $\hat{i}$, $\hat{j}$ e $\hat{k}$.

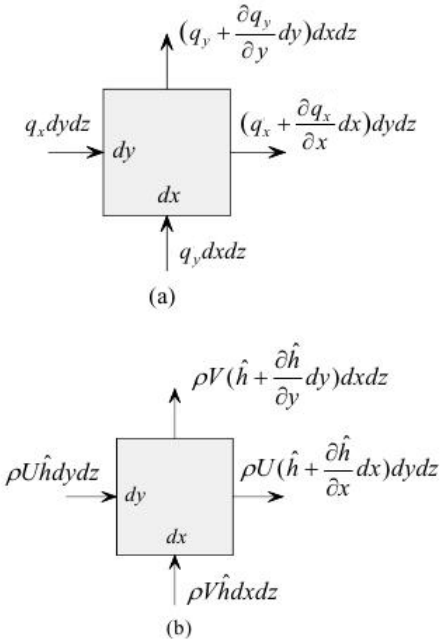


Figura 2.2: Fluxos nos elementos infinitesimais no plano 2D. Extraído de Jiji (2009).

A energia gerada $\dot{E}_g$ é dada pela equação:

$$\dot{E}_g = Qdxdydz. \tag{2.3}$$

A energia que deixa o elemento é construída usando-se expansão por série de Taylor (conforme Apêndice B). A formulação é dada pela Eq. (2.4) (note que U, V e W são constantes).

$$\begin{aligned}\dot{E}_{sai} = (q_x + \frac{\partial q_x}{\partial x}dx)dydz + (q_y + \frac{\partial q_y}{\partial y}dy)dxdz + (q_z + \frac{\partial q_z}{\partial z}dz)dxdy + \\ \rho U(\hat{h} + \frac{\partial \hat{h}}{\partial x}dx)dydz + \rho V(\hat{h} + \frac{\partial \hat{h}}{\partial y}dy)dxdz + \rho W(\hat{h} + \frac{\partial \hat{h}}{\partial z}dz)dxdy.\end{aligned} \tag{2.4}$$

A energia trocada dentro do elemento $\dot{E}$ resulta em:

$$\dot{E} = \rho\frac{\partial \hat{u}}{\partial t}dxdydz, \tag{2.5}$$

onde $\hat{u}$ é a energia interna por unidade de massa e t é o tempo. Substituindo as Eqs. (2.2)

a (2.5) na Eq. (2.1), obtém-se a equação:

$$-\nabla \cdot \vec{q} - \rho \vec{v} \cdot \nabla \hat{h} + Q = \rho \frac{\partial \hat{u}}{\partial t}, \tag{2.6}$$

onde $\vec{v} = (U, V, W)^T$, ∇ é o operador gradiente e $\nabla \cdot ()$ é o operador divergente.

A entalpia $\hat{h}$ é definida pela equação:

$$\hat{h} = \hat{u} + \frac{P}{\rho}, \tag{2.7}$$

onde P é a pressão, assumida como constante. Substituindo $\hat{u} = \hat{h} - \frac{P}{\rho}$ na Eq. (2.6) e rearranjando os termos, obtém-se a equação:

$$-\nabla \cdot \vec{q} + Q = \rho \left(\frac{\partial \hat{h}}{\partial t} + \vec{v} \cdot \nabla \hat{h} \right). \tag{2.8}$$

O próximo passo é expressar o fluxo e a entalpia em termos da temperatura T. A lei de Fourier de condução, dada pela Eq. (2.9), relaciona o fluxo de calor com o gradiente da temperatura:

$$\vec{q} = -k \nabla T, \tag{2.9}$$

onde k é a condutividade térmica.

A mudança da entalpia em pressão constante é dada pela equação:

$$d\hat{h} = c_p dT, \tag{2.10}$$

onde c_p é o calor especifico sob pressão constante. A substituição das Eqs. (2.9) e (2.10) na Eq. (2.8) resulta em:

$$\nabla \cdot k \nabla T + Q = \rho c_p \left(\frac{\partial T}{\partial t} + \vec{v} \cdot \nabla T \right). \tag{2.11}$$

Se for desconsiderado o termo convectivo na Eq. (2.11), obtém-se a seguinte equação:

$$\nabla \cdot k \nabla T + Q = \frac{\partial T}{\partial t} \rho c_p. \tag{2.12}$$

2.2 Biotransferência de Calor em Tecido Vivo

A Eq. (2.12) descreve a transferência de calor em um meio qualquer, porém o objetivo deste trabalho é simular a transferência de calor em tecido vivo. Então, esta seção irá descrever algumas simplificações e adaptações que foram consideradas por Pennes para descrever este fenômeno.

A complexa natureza da transferência de calor em tecido vivo dificulta a modelagem. Algumas suposições e simplificações devem ser adotadas para tratar deste problema, porém capturando as características essenciais. Esta seção apresenta uma resumida formulação de uma equação que determina a distribuição de temperatura em tecido vivo. Essa modelagem se baseou na perfusão sanguínea, propriedades térmicas e simplificação da geometria vascular.

2.2.1 Simplificações Assumidas

A equação de Pennes é baseada principalmente em quatro simplificações. São elas:

1. **Equilíbrio Local**: A principal troca de calor entre o sangue e o tecido acontece entre os vasos capilares e arteríolas[1] de fornecimento de sangue aos capilares e as vênulas[2] de drenagem. Então, toda a transferência de calor entre as pré-arteríolas e pós-vênulas e os tecidos são desconsideradas.

2. **Perfusão Sanguínea**: O fluxo sanguíneo nos pequenos capilares é assumido como sendo isotrópico. Consequentemente isso desconsidera o efeito da direção do fluxo sanguíneo.

3. **Arquitetura Vascular**: As grandes veias sanguíneas na vizinhança dos vasos capilares não desempenham nenhum papel de troca de energia entre tecido e sangue capilar. Consequentemente este modelo não considera a geometria vascular local.

[1] As arteríolas são os últimos segmentos das ramificações dos vasos arteriais e apresentam diâmetro inferior a $0,1$mm.

[2] Veia de pequeno calibre ($0,2$ a 1 mm) que estabelece a ligação entre os capilares e as veias de maior calibre.

4. **Temperatura do Sangue**: Assume-se que o sangue que alcança as arteríolas, suprindo os vasos capilares, está na temperatura interna do corpo T_a. A temperatura do sangue T_a entra pelas vênulas e arteríolas e, instantaneamente, equilibra com a temperatura T do tecido local. Baseado nestas suposições, Pennes modela o efeito sanguíneo como sendo uma fonte ou sorvedouro isotrópico de calor, o qual é proporcional a taxa de fluxo sanguíneo e a diferença de temperatura entre a temperatura central do corpo T_a e o tecido local T. Neste modelo, o sangue originalmente submetido a uma temperatura T_a não perde ou ganha energia conforme circula nos vasos sanguíneos do corpo.

2.2.2 Formulação da Equação de Pennes

O elemento de tecido mostrado na Fig. 2.3 é grande o suficiente para ser saturado com arteríolas, vênulas e capilares, mas pequeno se comparado com a dimensão da região considerada.

A taxa de energia adicionada ao elemento é devido a condução e convecção. Aqui o componente convectivo é eliminado e substituído pela energia adicionada devido a perfusão sanguínea. O modo mais simples de considerar este efeito é tratá-lo como energia gerada $\dot{E}_g$. Chamam-se Q_b e Q_m a taxa de energia adicionada ao sangue por unidade de volume do tecido e a taxa de energia metabólica produzida por unidade de volume do tecido, respectivamente. Então a Eq. (2.3) se torna:

$$\dot{E}_g = Q dx dy dz = (Q_m + Q_b) dx dy dz. \tag{2.13}$$

Para formular Q_b considere os elementos mostrado na Fig. 2.3. De acordo com as simplificações adotadas, o sangue entra no elemento de tecido vivo a uma temperatura interna corporal T_a, e instantaneamente com o tecido de temperatura T. Então:

$$Q_b = \rho_b c_b \dot{\omega}_b (T_a - T(\vec{x}, t)), \tag{2.14}$$

onde c_b, $\dot{\omega}_b$ e ρ_b são o calor específico do sangue, taxa de fluxo sanguíneo volumétrico por

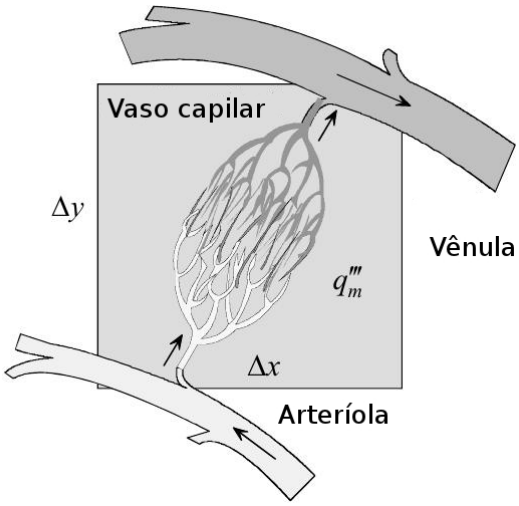


Figura 2.3: Capilares. Adaptado de Jiji (2009)

.

unidade de volume de tecido e densidade do sangue, respectivamente.

Substituindo a Eq. (2.14) em (2.13), obtêm-se:

$$Q = Q_m + \rho_b c_b \dot{\omega}_b (T_a - T(\vec{x}, t)). \tag{2.15}$$

Substituindo-se a Eq. (2.15) na equação de condução de calor (2.12), obtém-se a seguinte equação, conhecida como Equação de Pennes:

$$\nabla \cdot k \nabla T(\vec{x}, t) + \rho_b c_b \dot{\omega}_b (T_a - T(\vec{x}, t)) + Q_m = \rho c \frac{\partial T(\vec{x}, t)}{\partial t}, \tag{2.16}$$

onde c, k e ρ são o calor específico do tecido, condutividade térmica do tecido e densidade do tecido, respectivamente.

Tomando $t \to \infty$, pode-se obter a equação no estado estacionário, dada por:

$$\nabla \cdot k \nabla T(\vec{x}) + \rho_b c_b \dot{\omega}_b (T_a - T(\vec{x})) + Q_m = 0. \tag{2.17}$$

2.3 Hipertermia com Nanopartículas Magnéticas

2.3.1 Geração de Calor pelas Nanopartículas

A maior vantagem do tratamento do câncer utilizando hipertermia com nanopartículas magnéticas é a capacidade de realizar tratamento em tecidos profundos *e.g.*, mama, próstata e fígado, de uma forma não invasiva, através da injeção de nanopartículas e posterior exposição das mesmas a um campo magnético alternado. As nanopartículas podem ser injetadas principalmente de duas formas: intravenosa, usando uma solução biocompatível, ou diretamente no tumor alvo, através de agulhas (Moros, 2012; Minkowycz *et al.*, 2012).

A geração de calor em materiais magnéticos submetidos a um campo magnético H alternado pode ser, de modo geral, dividida em 3 regimes: pela corrente de Foucault, histereses em estruturas multidomínio (nanoescala e maiores) e por relaxações em domínio único (nanopartículas superparamagnéticas) (Hergt *et al.*, 1998). Estes comportamentos magnéticos podem ser observados na Fig. 2.4.

Uma vez que as nanopartículas são injetadas, pode-se aplicar um campo magnético alternado de baixa frequência (aproximadamente $100kHz$). As nanopartículas magnéticas começam a gerar calor devido a histereses e mecanismos de relaxação de Néelian e Browniano. A relaxação browniana envolve a rotação do momento das partículas para alinhá-las com o campo magnético externo, o que causa calor pela fricção das mesmas. A relaxação de Néelian envolve a rotação dos momentos internos das partículas individualmente, o que também gera calor por fricção. A histerese é uma característica de substâncias magnéticas que, na presença de um campo magnético, imantam (alinham o *spin*), porém quando o campo é retirado a substância não se desmagnetiza completamente, muito menos de forma instantânea. Quando aplicado um campo magnético alternado, formam-se os ciclos de histereses, que variam conforme a potência do campo e a substância. A Fig. 2.5 apresenta os gráficos do campo magnético H *versus* indução magnética M.

Um dos principais desafios da modelagem deste fenômeno é a quantificação do calor gerado pela interação entre as nanopartículas e o campo magnético. Este calor gerado

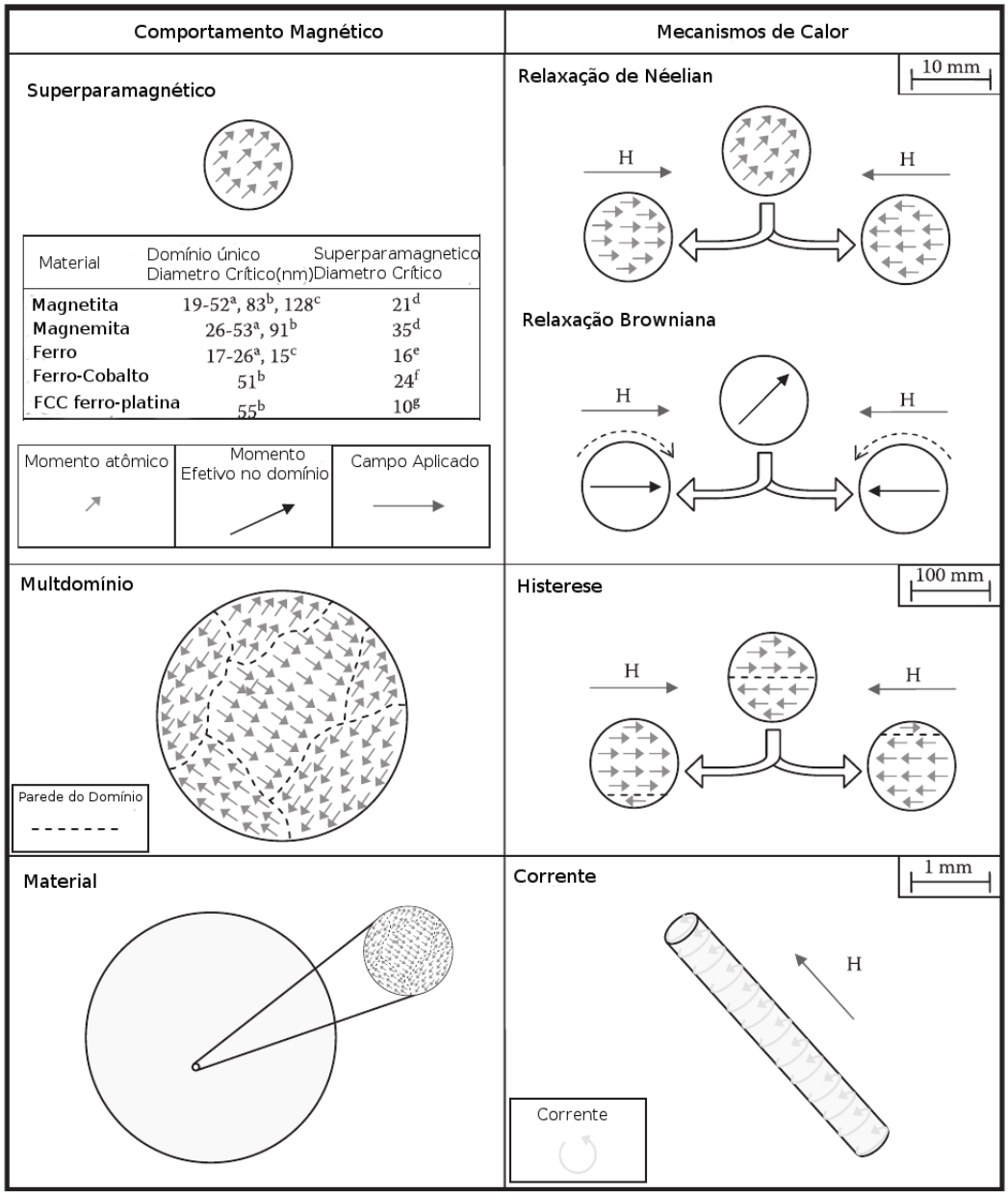


Figura 2.4: Comportamento magnético e mecanismos de geração de calor. Adaptado de Moros (2012, p.297)

.

será representado por uma taxa de absorção específica, chamada SAR (do inglês *specific absorption rate*), a qual calcula a energia dissipada pelas nanopartículas.

Um grande número de estudos *in vitro* demonstrou a efetividade das nanopartículas magnéticas, na presença de um campo magnético alternado, para o superaquecimento de células para um nível citotóxico. Estudos feitos por Jordan *et al.* (1999) mostraram que tratamentos entre 5 e 120 minutos a uma temperatura de 43^oC a 48^oC produziram necrose celular. Além disso, o tratamento com óxido férrico também apresentou um tipo

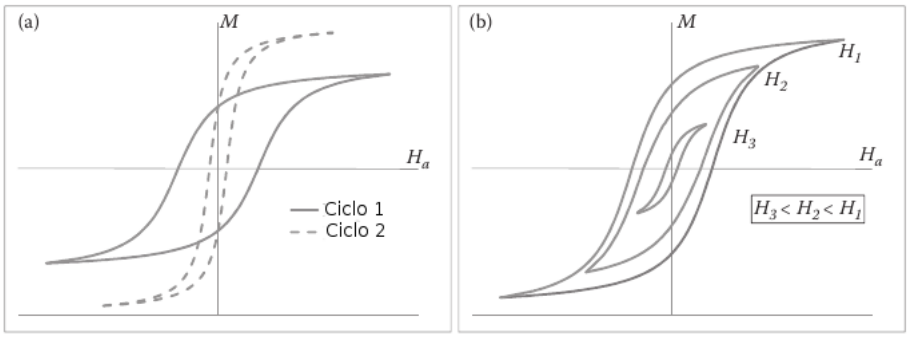


Figura 2.5: Exemplos de ciclos de histerese, demonstrando diferentes ciclos baseados no material (a) e intensidade do campo aplicado (b). Adaptado de Hergt *et al.* (1998).

de sensibilização após os primeiros 60 minutos, que diminui a taxa de sobrevivência das células. É especulado que isto possa ser resultado de uma ruptura da membrana celular ou dano nas organelas, devido ao aquecimento causado pelas nanopartículas.

Em um estudo separado, observou-se que as concentrações de ferro de $1mM$ não produzem efeitos citotóxicos a 37^oC, mas tornam-se tóxicos a temperaturas de 43^oC (Freeman *et al.*, 1990).

Impulsionados pelos resultados *in vitro*, alguns grupos de pesquisa continuaram com estudos *in vivo* em animais, e obtiveram resultados promissores nas últimas 5 décadas.

A hipertermia com fluidos magnéticos também foi utilizada para tratamento em ratos de tumores do tipo glioblastoma multiforme e de próstata (Jordan *et al.*, 2006; Johannsen *et al.*, 2005). Além disso, o estudo *in vivo* mostrou o potencial do tratamento com hipertermia como uma terapia que pode ser combinada com radioterapia em carcinoma de próstata (Johannsen *et al.*, 2006).

Dentre as nanopartículas existentes, óxido ferromagnético, Fe_3O_4, e magnetita, $\gamma - Fe_2O_3$, são as nanopartículas mais estudadas (Moroz *et al.*, 2002). Estas nanopartículas possuem o tamanho de aproximadamente $10 - 40nm$ e são largamente utilizadas devido às suas respectivas biocompatibilidades e capacidade de alta geração de calor, segundo Salloum *et al.* (2008).

2.3.2 Aplicador do Campo Magnético

Outra importante ferramenta neste tipo de tratamento é o aplicador do campo magnético. É necessária uma forma segura e eficaz para aplicar o campo magnético externo para ativar as nanopartículas implantadas. Dentre as preocupações necessárias pode-se citar a uniformidade do campo, conforto do paciente, e a capacidade de realizar o tratamento em todo o corpo. Segundo Gneveckow *et al.* (2004), atualmente o único aplicador para uso clínico é o *NanoActivator* da empresa *MagForce Nanotechnologies AG* (MagForce, 2014), cujo esboço é apresentado na Fig. 2.6.

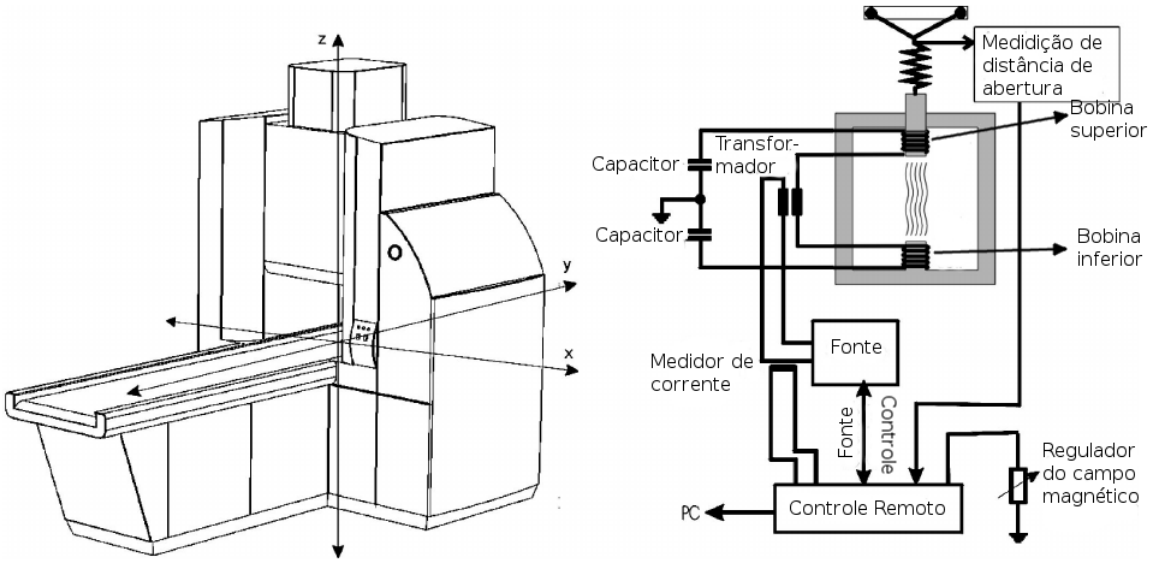


Figura 2.6: Este esboço mostra o primeiro protótipo de um aplicador de campo magnético alternado para uso médico, o MFH 300F. Adaptado de Gneveckow *et al.* (2004, p.1445)

.

Os pacientes são deitados horizontalmente na cama do aplicador, e deslizados na direção do eixo-y do aparelho. O aplicador gera um campo magnético alternado de 100kHz entre os dois polos magnéticos, criando um campo magnético aproximadamente cilíndrico de diâmetro 20 cm entre os polos. O campo magnético depende linearmente da corrente aplicada à solenoide, e pode ser ajustado de 2 até 18 kA/m.

2.3.3 Modelos de Fonte de Calor

Neste trabalho se representará a geração de calor devido ao tratamento por hipertermia por uma função denominada SAR. Para o tipo de partícula e força do campo magnético

utilizado neste tipo de procedimento, a distribuição espacial do calor gerado pelo tratamento é dada por esta taxa de absorção.

Um recente estudo experimental *in vivo* feito por Salloum *et al.* (2008) e executado em membros posteriores de ratos mostrou que a taxa de absorção em volta do ponto de injeção pode ser aproximada por uma curva de distribuição gaussiana:

$$Q_r(\vec{x}) = \sum_{i=1}^{M} A_i . e^{\frac{-r_i^2}{r_{0,i}^2}}, \tag{2.18}$$

onde M é o número de pontos de injeções de nanopartículas no tumor, r_0 o raio de abrangência da hipertermia, A a taxa de geração de calor máxima e r a distância do ponto de injeção.

Esta aproximação foi baseada no estudo inverso da evolução da temperatura. A Fig. 2.7 representa a evolução da temperatura, aferida a cada 2 segundos por um termômetro termoelétrico, nos membros posteriores do rato.

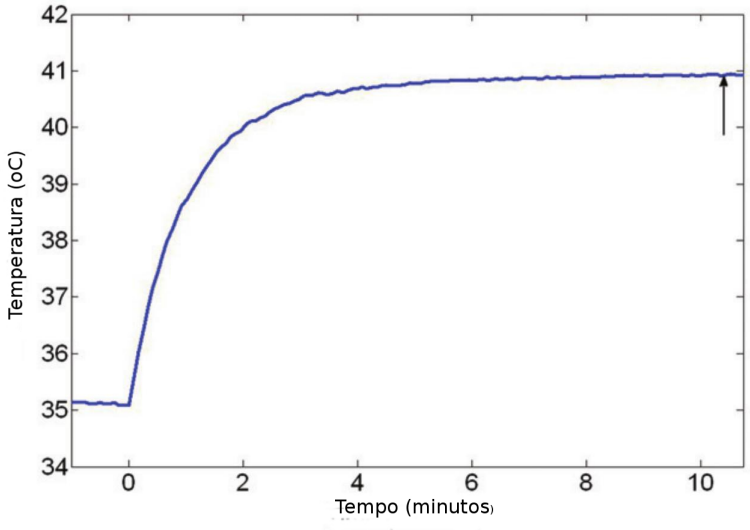


Figura 2.7: Evolução da temperatura aferida nos membros posteriores do rato. A seta indica a chegada ao estado estacionário. Adaptado de Salloum *et al.* (2008, p.593)

.

Recentemente, Mital e Tafreshi (2012) incluiram um perfil temporal na Eq. (2.18) para representar o decaimento do calor durante o tratamento através de hipertermia, ficando

a expressão final do SAR da seguinte forma:

$$Q_r(\vec{x},t) = \sum_{i=1}^{M} A_i e^{-r_i^2/r_{0_i}^2} e^{-t/\tau}, \tag{2.19}$$

onde τ é uma constante temporal que controla a taxa de decaimento.

2.4 Perfusão Sanguínea

2.4.1 Fisiologia da Arquitetura Vascular

Não se sabe se células malignas são mais sensíveis ao calor quando comparadas às células normais, porém o que pode ser demonstrado é que tecidos neoplásicos[3] podem sofrer danos à temperaturas inócuas aos tecidos normais (Corry *et al.*, 1982; Onishi *et al.*, 1990). Além disso, durante a aplicação de calor, a temperatura do tumor cresce mais se comparada a das células normais dos tecidos circundantes (Storm *et al.*, 1979). Acredita-se que o calor é controlado, principalmente, pela dissipação causada pelo fluxo sanguíneo durante a aplicação de calor. Portanto a temperatura superior no tecido tumoral é atribuída ao vagaroso fluxo sanguíneo (Song *et al.*, 1984).

Durante o estágio inicial do crescimento tumoral, as células do tumor são mantidas por nutrientes fornecidos pelo sistema vascular local. Conforme o tumor cresce, as vênulas adjacentes da arquitetura vascular se tornam dilatadas e tortuosas. Subsequentemente, as células endoteliais[4] do sistema vascular local alterado começam a se proliferar, talvez por influência de fatores de angiogênese[5] das células tumorais (Yang, 2012).

Acredita-se que o padrão vascular é característico de cada tipo de tumor, apesar de algumas variações neste padrão serem encontradas dependendo do estágio de evolução em um mesmo tipo de tumor. Quanto mais o tumor se desenvolve, mais capilares são formados, na maioria dos casos de forma senoidal, *i.e.*, um canal fino com formato irregular. Além disto, frequentemente, entre as lacunas das células endoteliais, as células

[3]Tecido neoplásico é aquele formado por células neoplásicas, *i.e*, célula que, por algum mecanismo, teve seu código genético alterado a ponto de perder sua função biológica característica.

[4]Célula endotelial é uma célula de morfologia achatada que cobre o interior dos vasos sanguíneos, principalmente dos capilares.

[5]Fatores de angiogênese são substâncias que estimulam o crescimento vascular.

neoplásicas salientam-se no canal dos capilares, obstruindo a perfusão sanguínea (Peterson *et al.*, 1969).

Uma vez que é criada a arquitetura vascular no tumor e o sangue começa a fluir das artérias primárias para o tumor, a massa tumoral cresce progressivamente. Apesar de crescer o número e o tamanho da nova vasculatura, elas continuam sendo alimentadas pelos mesmos vasos sanguíneos primários. Isto faz a demanda de sangue exceder a capacidade das artérias primárias, resultando na redução da pressão arterial. Por outro lado, a pressão extravascular cresce devido a rápida proliferação das células tumorais em um espaço limitado. Então, essa limitação da difusão de oxigênio e dos nutrientes para as células, causada pela estagnação resultante do excesso de pressão extravascular, é uma das causas de necrose nos tumores (Thomlinson e Gray, 1955). Vale enfatizar que o fluxo sanguíneo varia de acordo com o tipo e estágio de crescimento tumoral, ou ainda, o tamanho do tumor. Ademais, a perfusão sanguínea no tumor não é homogênea, *i.e.*, não ocorre da mesma forma em todas as regiões do tecido tumoral (Falk, 1978; Ishida *et al.*, 2012). Aparentemente o fluxo sanguíneo em células neoplásicas é geralmente menor que a correspondente em células normais, porém exceções podem existir. Na Fig. 2.8 é possível observar a evolução do fluxo sanguíneo a medida que cresce o melanoma em um rato. Inicialmente são criados novos capilares, que contribuem para o aumento na velocidade do fluxo sanguíneo. No décimo sexto dia observa-se o inicio da redução do fluxo em algumas regiões. Além disto é possível observar a não-homogeneidade do fluxo sanguíneo no tecido tumoral.

Os gráficos das Figs. 2.9(a) e 2.9(b) mostram o comportamento do fluxo sanguíneo quando os tecidos muscular e pele, respectivamente, são submetidos a uma banho com água quente em diferentes temperaturas. Analisando os resultados experimentais que foram obtidos, é possível inferir que para baixas temperaturas o crescimento do fluxo sanguíneo é progressivo, porém para temperaturas superiores, inicialmente o fluxo aumenta, mas decai drasticamente após um período inicial de aquecimento. De acordo com Song *et al.* (1984), o calor induzido causa mudanças na arquitetura vascular dos tecidos normais, *i.e.*, vasodilatação, crescimento do fluxo sanguíneo, bem como o aumento

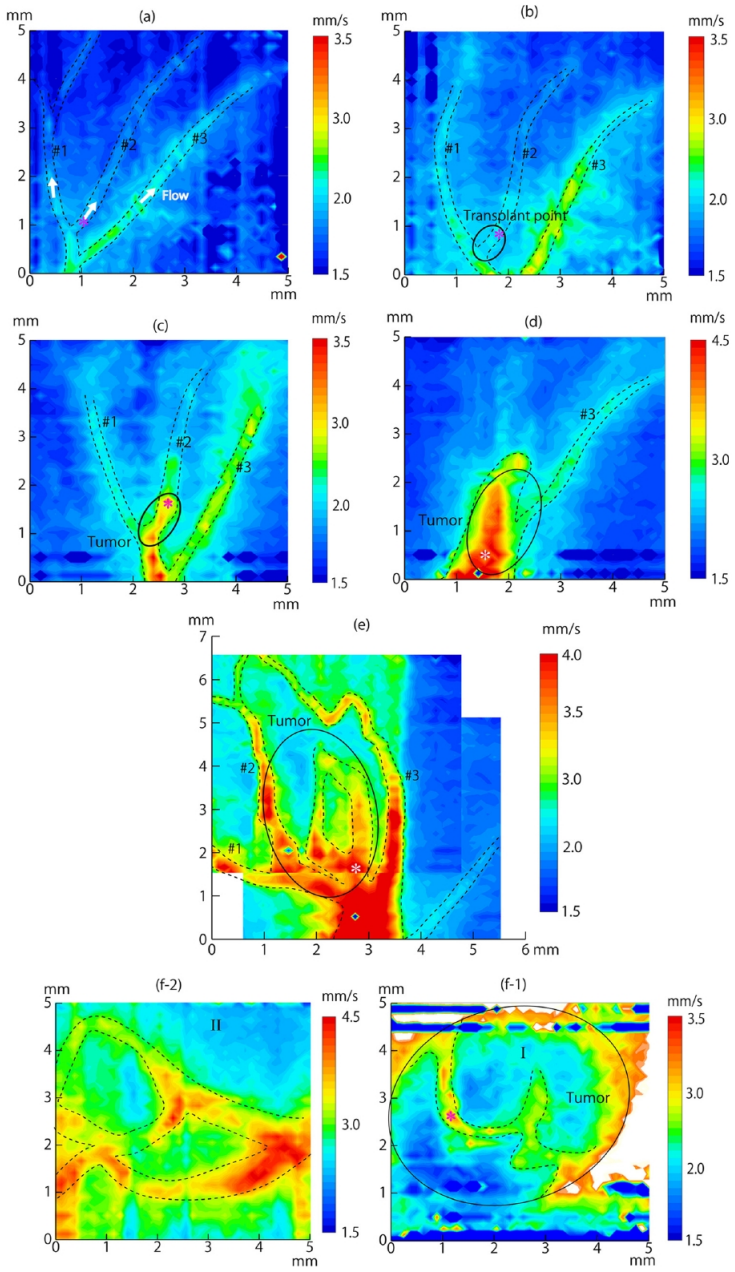
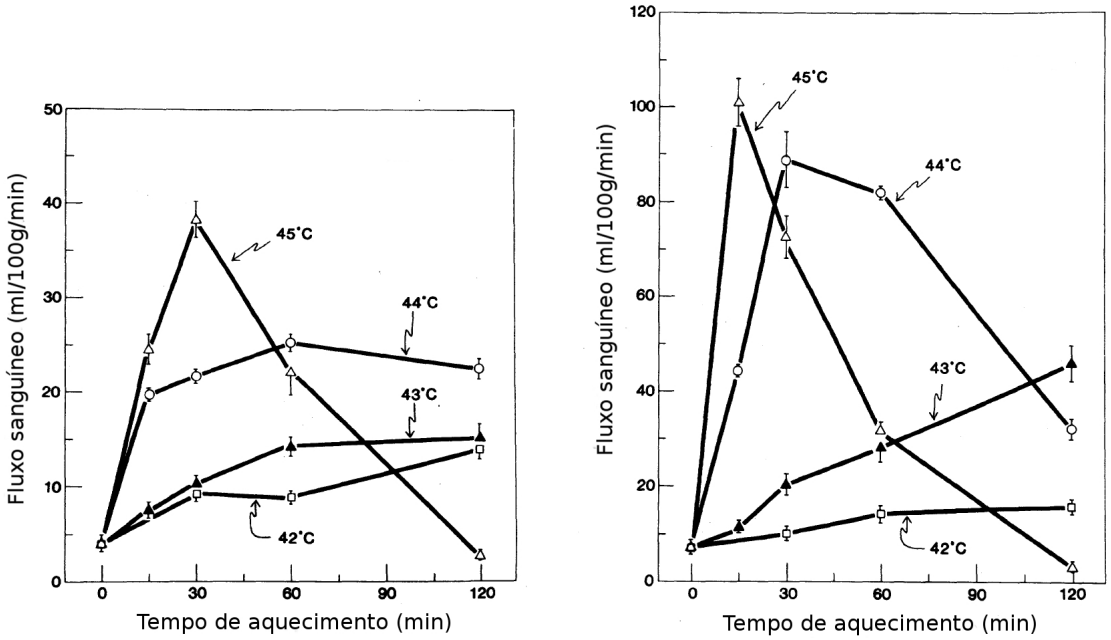


Figura 2.8: Imagens da velocidade do fluxo sanguíneo antes e após o transplante de células cancerígenas em rato: a) antes do transplante e b) dois, c) cinco, d) nove, e) doze, e f)dezesseis dias após o transplante. Adaptado de Ishida *et al.* (2012).

da permeabilidade. O extravasamento de proteínas plasmáticas, como resultado do calor induzido, aumenta a permeabilidade vascular, o que pode levar ao aumento no hematócrito[6] e viscosidade do sangue. Estas alterações podem causar o retardo na circulação sanguínea e uma eventual estagnação. Além disso, os vasos podem ser rompidos após danos mais sérios nas células endoteliais, causando hemorragia, explicando, assim,

[6]Valor ou porcentagem de glóbulos vermelhos em relação ao volume total do sangue.

os resultados observados nestas figuras.



(a) Evolução do fluxo sanguíneo no músculo.

(b) Evolução do fluxo sanguíneo na pele.

Figura 2.9: Mudança no fluxo sanguíneo em ratos aquecidos com banho de água em diferentes temperaturas. Adaptado de Song *et al.* (1984).

Conforme previamente fundamentado, o crescimento desgovernado da arquitetura vascular de um tumor pode causar sérios problemas no fluxo sanguíneo. Segundo Song *et al.* (1984), o calor induzido pode causar mudanças significativas na vasculatura tumoral diferentemente da vasculatura do tecido normal. Em experimentos realizados por este mesmo autor, o decaimento do fluxo sanguíneo pode acontecer após indução de calor à 42^oC prolongado por mais de 1h, chegando a quase zero, e gradualmente retornando ao normal após 12h. Cerca 50% de decaimento no fluxo ocorreu em aquecimento por 30 à 40 min à 41 e 42 ^oC. Reduções maiores ocorreram em aquecimentos à 43 ou 44 ^oC. A Fig. 2.10 mostra a mudança relativa no fluxo sanguíneo neste três tipos de tecido após 30 a 40 minutos de aquecimentos à temperaturas distintas.

É válido notar que a mudança de fluxo relativo do tumor foi representada por uma região, pois suas características são intrínsecas ao estágio de evolução, bem como seu tipo.

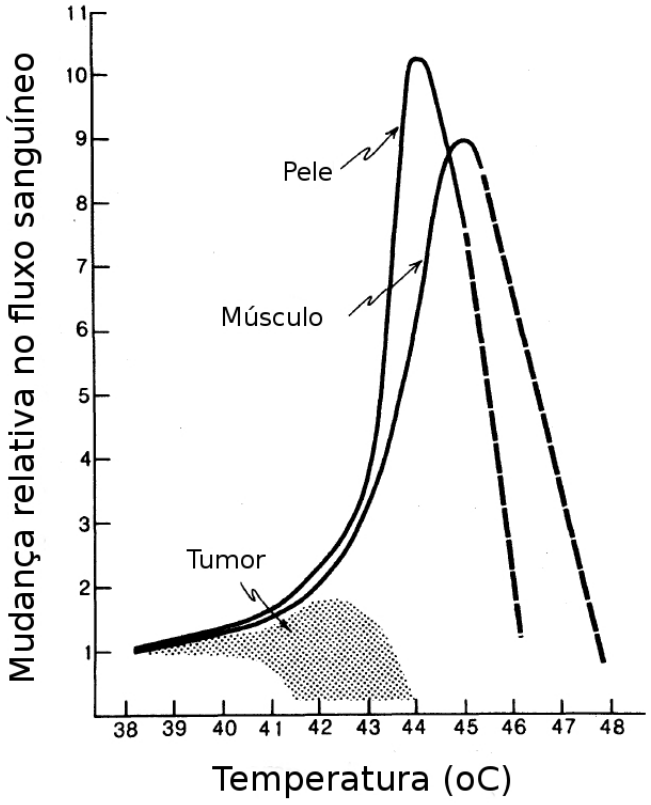


Figura 2.10: Mudança relativa do fluxo sanguíneo em pele, músculo e tumor em animais após aquecimento de 30 à 40 minutos. Adaptado de Song *et al.* (1984)

.

2.4.2 Modelos de Perfusão Sanguínea

A presente seção trata da aproximação do parâmetro de perfusão sanguínea $\omega_b = \dot{\omega}_b \rho_b$ da Eq. (2.16). Vários artigos na literatura, como Liu e Xu (2000); Salloum *et al.* (2008), sugerem a utilização de um valor constante. Já os trabalhos realizados por Gowrishankar *et al.* (2004); Lang *et al.* (1997); Erdmann *et al.* (1998) sugerem a utilização de um modelo dependente da temperatura. Então, o presente estudo analisará ambas as possibilidades.

O valor da perfusão sanguínea será diferente de acordo com o tipo de tecido, e serão consideradas as camadas de pele, gordura, músculo e tumoral. Utilizando uma função análoga às propostas por Lang *et al.* (1997) e baseados em valores das perfusões sanguíneas utilizadas por Erdmann *et al.* (1998); Liu e Xu (2000); Salloum *et al.* (2008), obteve-se as funções dadas pelas Eqs. (2.20), (2.21) e (2.22), representando, respectivamente, as perfusões sanguíneas do músculo, gordura e tumor. No caso da pele optou-se por utilizar

um valor constante $\omega_{pele} = 0,5$, segundo a literatura (Lang *et al.*, 1997).

$$\omega_{músculo} = \begin{cases} 0.45 + 3.55exp\left(- \frac{(T-45.0)^2}{12.0} \right) & T \leq 45.0^oC \\ 4.0 & T > 45.0^oC \end{cases} \tag{2.20}$$

$$\omega_{gordura} = \begin{cases} 0.36 + 0.36exp\left(- \frac{(T-45.0)^2}{12.0} \right) & T \leq 45.0^oC \\ 0.72 & T > 45.0^oC \end{cases} \tag{2.21}$$

$$\omega_{tumor} = \begin{cases} 0.833 & T < 37.0^oC \\ 0.833 - \frac{(T-37.0)^{4.8}}{5.438E+3} & 37.0 \leq T \leq 42.0^oC \\ 0.416 & T > 42.0^oC \end{cases} \tag{2.22}$$

Os gráficos das Eqs. (2.20), (2.21) e (2.22) são apresentados, respectivamente, pelas Figs. 2.11(a), 2.11(b) e 2.11(c).

Vale notar que nas Figs. 2.11(a) e 2.11(b), ou pela própria definição das Eqs. (2.20) e (2.21), que a perfusão sanguínea cresce somente até a temperatura de 45 ^oC e depois se torna constante, o que difere do gráfico da Fig. 2.10. Observe que nos dados obtidos experimentalmente o fluxo sanguíneo começa a cair quando a temperatura está acima de 44^oC na pele e 45^oC no músculo. Esta diferença se justifica pelo fato de que na modelagem feita neste trabalho não se considera que a aplicação das nanopartículas será feita no tecido saudável, mas apenas no tumoral. Então os tecidos normais não chegarão a esta temperatura, e sim em temperaturas em que não existe a queda da perfusão, sendo assim, pode ser considerada uma boa aproximação do fenômeno.

Para comparar com os resultados dos modelos dependentes da temperatura com a perfusão sanguínea constante, adotou-se uma média ponderada (*i.e.*, $\overline{\omega} = \frac{\int_{T_0}^{T_f} T\omega(T)dT}{\int_{T_0}^{T_f} TdT}$). Então, os valores de cada uma das médias para cada uma das camadas são dados por $\overline{\omega}_{músculo} = 1,87895$, $\overline{\omega}_{gordura} = 0,504908$, e $\overline{\omega}_{tumor} = 0,757981$.

2.5 Formalização do Modelo Matemático

Por fim, este processo pode ser representado matematicamente pela conhecida equação de Pennes, com algumas modificações previamente explicadas. Considere que o tecido vivo contendo o tumor é representado por um domínio aberto $\Omega \subset \mathbb{R}^3$ e $I = (0, t_f] \subset \mathbb{R}^+$ é o intervalo de tempo do tratamento, então a equação de Pennes é dada por:

$$\nabla \cdot \kappa \nabla T + \omega_b(T) c_b (T_a - T) + Q_m + Q_r = \rho c \frac{\partial T}{\partial t}, \text{ em } \Omega \times I,$$

onde $\omega_b(T) = \dot{\omega}_b(T)\rho_b$.

Além disso, para o problema ter solução única, deve-se especificar condições de fronteira apropriadas em $\Gamma = \partial\Omega$ e uma condição inicial em $t = 0$, isto é,

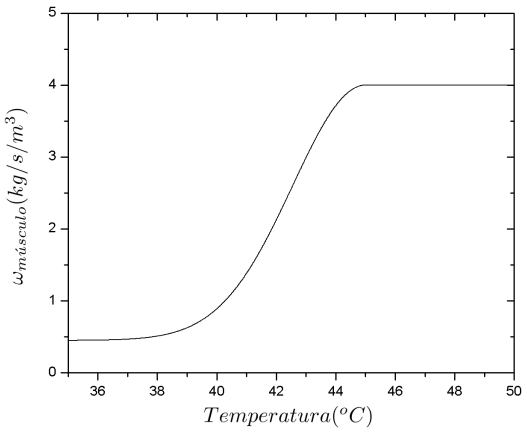
$$\alpha T + \beta \nabla T \cdot \vec{n} = f, \text{ em } \Gamma \times I,$$

$$T(\cdot, 0) = T_0, \text{ em } \Omega,$$

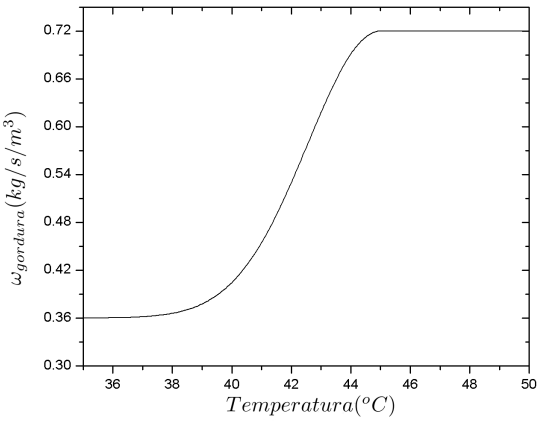
onde $f : \Omega \times I \to \mathbb{R}^+$ é a temperatura prescrita (Dirichlet) ou fluxo (Neumann ou Robin), dependendo da escolha das funções $\alpha, \beta : \Omega \to \mathbb{R}^+$, com $\vec{n}$ sendo o vetor normal à superfície.

Finalmente, juntando todas as considerações obtém-se o seguinte modelo:

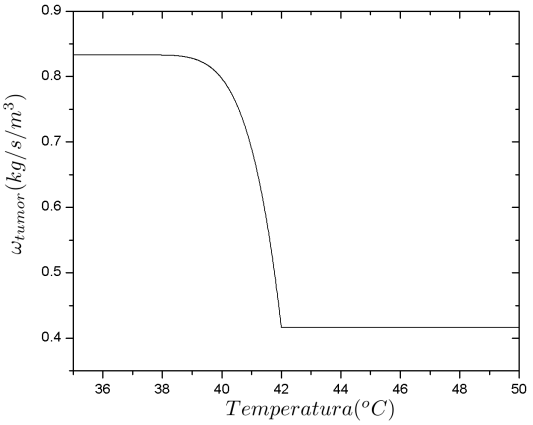
$$\begin{cases} \nabla \cdot \kappa \nabla T + \omega_b(T) c_b (T_a - T) + Q_m + Q_r = \rho c \dfrac{\partial T}{\partial t}, & \text{em } \Omega \times I \\ \alpha T + \beta \nabla T \cdot \vec{n} = f, & \text{em } \Gamma \times I. \\ T(\cdot, 0) = T_0, & \text{em } \Omega \end{cases} \tag{2.23}$$



(a) Gráfico da perfusão sanguínea do músculo dado pela Eq. (2.20)



(b) Gráfico da perfusão sanguínea da gordura dado pela Eq. (2.21)



(c) Gráfico da perfusão sanguínea do tumor dado pela Eq. (2.22)

Figura 2.11: Perfusão sanguínea dependente da temperatura: a)músculo; b)gordura; c)tumor.

3 MÉTODO NUMÉRICO

A solução de equações diferencias parciais ainda é um grande desafio para a ciência e para as engenharias. Fundamentando-se no fato de que a maioria das equações diferencias parciais não possui solução analítica, a proposição de métodos numéricos capazes de solucionar estas equações é tema de grandes linhas de pesquisa, principalmente para profissionais da área de modelagem matemática e computacional.

Este capítulo irá descrever duas estratégias que foram utilizadas na aproximação da solução do modelo dado pela Equação (2.23). Ambas as estratégias utilizam um esquema de diferenças finitas no espaço; a diferença está na marcha do tempo, onde a primeira utiliza uma diferença progressiva, *i.e.*, método de Euler, e a segunda um algoritmo preditor-corretor.

3.1 A Discretização no Espaço

Neste trabalho considera-se um domínio fechado $\Omega \cup \Gamma \subset \mathbb{R}^3$ discretizado em um conjunto de pontos regularmente espaçados $\boldsymbol{S} = \{(x_i, y_j, z_k)\,; i = 0, \ldots, I_x; j = 0, \ldots, I_y; k = 0, \ldots, I_z\}$ com I_x, I_y e I_z sendo o número de intervalos de tamanho Δx, Δy e Δz, respectivamente em cada direção, conforme ilustra a Fig. 3.1.

3.1.1 Condutividade Térmica em Meio Heterogêneo

Na Eq. (2.23) o termo $k : \Omega \to \mathbb{R}^+$ pode ser uma função descontínua. Deste modo, para garantir a continuidade do fluxo $\vec{q}$ deve-se fazer algumas considerações sobre k.

Considere o fluxo q_x, *i.e.*, o valor de $\vec{q}$ na direção $\hat{i}$, dado por:

$$q_x = -k\frac{\partial T}{\partial x}. \tag{3.1}$$

A Fig. 3.2 mostra 2 pontos consecutivos quaisquer, considere q_i o fluxo no ponto x_i,

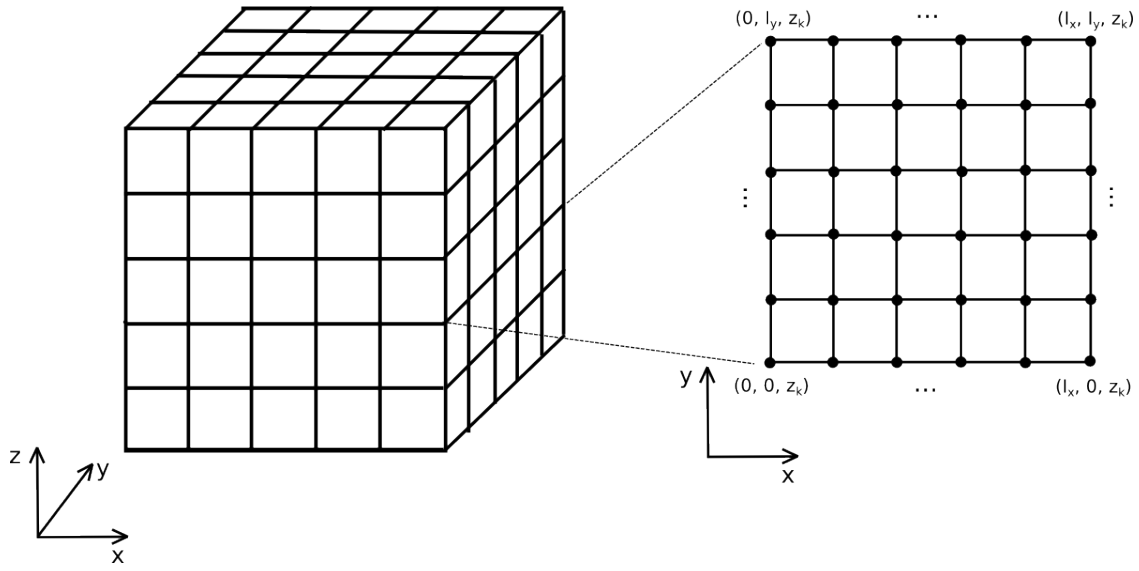


Figura 3.1: Discretização do domínio tridimensional.

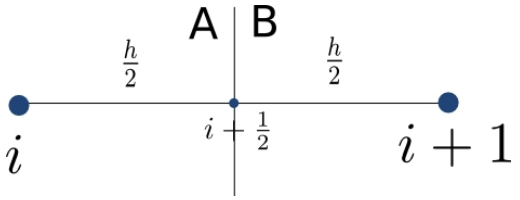


Figura 3.2: Pontos i e $i+1$ em um meio heterogêneo com o ponto fictício $i+\frac{1}{2}$.

$i+\frac{1}{2}$ um ponto fictício entre i e $i+1$, e seus respectivos fluxos laterais $q^{-}_{i+\frac{1}{2},j,k}$ e $q^{+}_{i+\frac{1}{2},j,k}$. Pode-se aproximar estes fluxos por meio de uma diferença progressiva da seguinte forma:

$$q^{-}_{i+\frac{1}{2},j,k} \approx \frac{-2k_A}{h}(T_{i+\frac{1}{2},j,k} - T_{i,j,k}) \tag{3.2}$$

e

$$q^{+}_{i+\frac{1}{2},j,k} \approx \frac{-2k_B}{h}(T_{i+1,j,k} - T_{i+\frac{1}{2},j,k}), \tag{3.3}$$

onde k_A e k_B são as condutividade térmica nos meios A e B, respectivamente.

Além disso, considere o fluxo $q_{i+\frac{1}{2}}$, aproximado pela diferença centrada no ponto $i+\frac{1}{2}$, dado por:

$$q_{i+\frac{1}{2},j,k} \approx \frac{-k}{h}(T_{i+1,j,k} - T_{i,j,k}). \tag{3.4}$$

Para garantir a diferenciabilidade e por consequência a continuidade da função, então

$q^{-}_{i+\frac{1}{2},j,k} = q^{+}_{i+\frac{1}{2},j,k} = q_{i+\frac{1}{2},j,k}$, ou seja:

$$\frac{-2k_A}{h}(T_{i+\frac{1}{2},j,k} - T_{i,j,k}) = \frac{-2k_B}{h}(T_{i+1,j,k} - T_{i+\frac{1}{2},j,k}) = \frac{-k}{h}(T_{i+1,j,k} - T_{i,j,k}). \tag{3.5}$$

Isolando o termo $T_{i+\frac{1}{2},j,k}$ da parte 1 da equação com a parte 2, e na parte 2 com a parte 3, obtém-se:

$$k(k_A + k_B)(T_{i+\frac{1}{2},j,k} - T_{i,j,k}) = 2k_A k_B(T_{i+\frac{1}{2},j,k} - T_{i,j,k}). \tag{3.6}$$

Portanto, a condutividade térmica efetiva k, calculada pela média harmônica de k_A e k_B, é dada por:

$$k = \frac{2k_A k_B}{k_A + k_B}. \tag{3.7}$$

A demonstração nas direções $\hat{j}$ e $\hat{k}$ é análoga à feita para $\hat{i}$.

3.1.2 Método das Diferenças Finitas em Meio Heterogêneo

Considerando as relações de diferenças dadas pela Tab. A.1, deseja-se obter uma relação de diferenças para discretizar o operador laplaciano, ou seja, o termo $\nabla\cdot(k\nabla T)$ da Eq. (2.23), porém considerando k não constante.

O operador laplaciano de uma função $T : \mathbb{R}^n \to \mathbb{R}$ pode ser definido como o divergente do gradiente, ou seja:

$$\nabla \cdot (\nabla T) = \sum_{i=1}^{n} \frac{\partial^2 T}{\partial x_i^2}, \tag{3.8}$$

onde, para o caso tridimensional, $n = 3$.

Considerando o eixo dado pelo vetor $\hat{i}$, $\frac{\partial^2 T}{\partial x^2}$ a derivada de segunda ordem da função T na direção $\hat{i}$ e aplicando uma diferença centrada de segunda ordem, pode-se obter a

seguinte relação de diferenças para aproximar o operador:

$$\frac{\partial^2 T}{\partial x^2} \approx \frac{T_{i+1,j,k} - 2T_{i,j,k} + T_{i-1,j,k}}{\Delta x^2}. \tag{3.9}$$

Porém, não se pode aplicar esta relação na Eq. (2.23), pois k é uma função descontínua. Neste caso, tome dois pontos fictícios no domínio $i + \frac{1}{2}$ e $i - \frac{1}{2}$, conforme a Fig. 3.3.

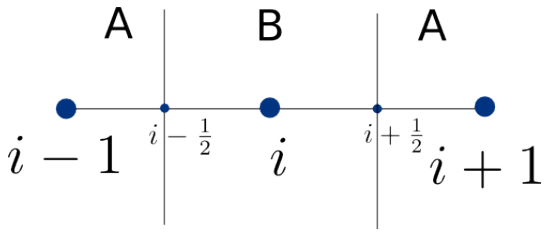


Figura 3.3: Meio heterogêneo com os pontos fictícios $i + \frac{1}{2}$ e $i - \frac{1}{2}$.

Então, pode-se calcular a diferença centrada no ponto i pelos pontos fictícios $i \pm \frac{1}{2}$ conforme ilustrado pela Eq. (3.10):

$$\frac{\partial}{\partial x}\left(k\frac{\partial T}{\partial x}\right) \approx \frac{k_{i+\frac{1}{2},j,k}\frac{\partial T_{i+\frac{1}{2},j,k}}{\partial x} - k_{i-\frac{1}{2},j,k}\frac{\partial T_{i-\frac{1}{2},j,k}}{\partial x}}{\Delta x} \tag{3.10}$$

As derivadas nos pontos $i \pm \frac{1}{2}$ podem também ser aproximadas por uma diferença centrada, porém em relação aos pontos $i \pm 1$:

$$\frac{\partial T_{i-\frac{1}{2},j,k}}{\partial x} \approx \frac{T_{i,j,k} - T_{i-1,j,k}}{\Delta x} \tag{3.11}$$

$$\frac{\partial T_{i+\frac{1}{2},j,k}}{\partial x} \approx \frac{T_{i+1,j,k} - T_{i,j,k}}{\Delta x} \tag{3.12}$$

Substituindo-se então os valores acima nos pontos fictícios da Eq. (3.10), obtém-se a relação dada pela Eq. (3.13):

$$\frac{\partial}{\partial x}\left(k\frac{\partial T}{\partial x}\right) \approx \frac{k_{i+\frac{1}{2},j,k}(T_{i+1,j,k} - T_{i,j,k}) - k_{i-\frac{1}{2},j,k}(T_{i,j,k} - T_{i-1,j,k})}{\Delta x^2} \tag{3.13}$$

Novamente, relações análogas podem ser obtidas nas direções $\hat{j}$ e $\hat{k}$. Além disto, vale enfatizar que se $k_{i+\frac{1}{2},j,k} = k_{i-\frac{1}{2},j,k}$, a Eq. (3.13) recai na Eq. (3.9).

3.1.3 Modelo Discretizado

Depois de aplicar a Eq. (3.13) nas direções $\hat{i}$, $\hat{j}$ e $\hat{k}$ da Eq. (2.23), obtém-se a discretização espacial dada pela Eq. (3.14), onde $T_{i,j,k} \approx T(x_i, y_j, z_k, t)$, sendo o operador $\Phi(T_{i,j,k}, T_{i\pm1,j,k}, T_{i,j\pm1,k}, T_{i,j,k\pm1})$ definido pela Eq. (3.15).

$$\begin{aligned} \rho_{i,j,k} c_{i,j,k} \frac{\partial T_{i,j,k}}{\partial t} = \Phi(T_{i,j,k}, T_{i\pm1,j,k}, T_{i,j\pm1,k}, T_{i,j,k\pm1}) + \\ (\omega_b(T))_{i,j,k} (\rho_b)_{i,j,k} (c_b)_{i,j,k} T_a + (Q_m)_{i,j,k} + (Q_r)_{i,j,k} \end{aligned}, \tag{3.14}$$

onde

$$\begin{aligned} &\Phi(T_{i,j,k}, T_{i\pm1,j,k}, T_{i,j\pm1,k}, T_{i,j,k\pm1}) = \\ &\tfrac{1}{\Delta x^2}\left[k_{i+\frac{1}{2},j,k}\left(T_{i+1,j,k} - T_{i,j,k}\right) - k_{i-\frac{1}{2},j,k}\left(T_{i,j,k} - T_{i-1,j,k}\right)\right] + \\ &\tfrac{1}{\Delta y^2}\left[k_{i,j+\frac{1}{2},k}\left(T_{i,j+1,k} - T_{i,j,k}\right) - k_{i,j-\frac{1}{2},k}\left(T_{i,j,k} - T_{i,j-1,k}\right)\right] + \\ &\tfrac{1}{\Delta z^2}\left[k_{i,j,k+\frac{1}{2}}\left(T_{i,j,k+1} - T_{i,j,k}\right) - k_{i,j,k-\frac{1}{2}}\left(T_{i,j,k} - T_{i,j,k-1}\right)\right] - \\ &(\omega_b(T))_{i,j,k} (\rho_b)_{i,j,k} (c_b)_{i,j,k} T_{i,j,k}. \end{aligned} \tag{3.15}$$

O *stencil* referente a este esquema numérico no espaço é detalhado na Fig. 3.4.

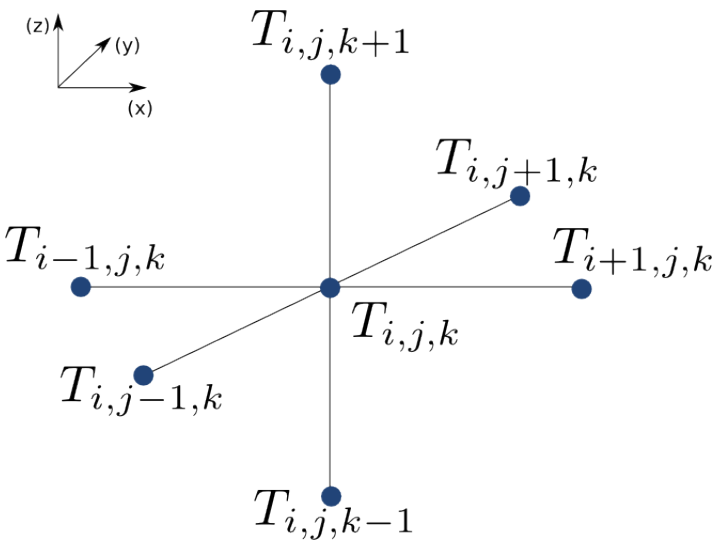


Figura 3.4: *Stencil* no espaço do esquema numérico tridimensional de 7 pontos.

3.1.4 Condições de Contorno

Em matemática, um problema de valor de contorno consiste em uma equação ou sistema de equações diferenciais munido de um conjunto de restrições sobre o contorno do domínio.

Entre os principais tipos de condições de contorno, é de interesse a este trabalho as condições de Neumann e Dirichlet.

3.1.4.1 Condições de Contorno de Dirichlet

A condição de contorno de Dirichlet, quando aplicada a uma equação diferencial parcial, especifica qual valor a solução deve tomar sobre o contorno do domínio Γ, ou seja, assumindo $\alpha = 1$ e $\beta = 0$ na Eq. (2.23), isto é, $T(\vec{x}, t) = f(\vec{x}, t)$ para $\vec{x} \in \Gamma$.

Para aplicar este tipo de condição de contorno em um esquema de diferenças finitas, deve-se atribuir o valor da condição nos pontos $(x_i, y_j, z_k) \in \Gamma$ a cada iteração do algoritmo, como pode ser observado a seguir:

$$T_{i,j,k} = f(x_i, y_j, z_k) \text{ para } (x_i, y_j, z_k) \in \Gamma, \tag{3.16}$$

onde $\Gamma = \partial\Omega$ é o contorno do domínio onde se aplica a condição de Dirichlet.

3.1.4.2 Condições de Contorno de Neumann

A condição de contorno de Neumann, quando aplicada à uma equação diferencial parcial, especifica o valor da derivada no ponto do contorno do domínio, ou seja, assumindo $\alpha = 0$ e $\beta = k$ na Eq. (2.23),isto é, $k\nabla T(\vec{x}, t) \cdot \vec{n} = f(\vec{x}, t)$ para $\vec{x} \in \Gamma$.

Uma das formas de se aplicar uma condição de Neumann consiste na criação de um ponto fictício chamado ponto fantasma. Este ponto ficará além do contorno conforme ilustrado na Fig. 3.5 para $\vec{n} = (1, 0, 0)^T$. O ponto fantasma será aproximado pela diferença centrada no ponto i, isto é, $(x_i, y_j, z_k) \in \Gamma$, da seguinte forma:

$$k\frac{\partial T}{\partial x} \approx k\frac{T_{i+1,j,k} - T_{i-1,j,k}}{2\Delta x} = f(x_i, y_j, z_k). \tag{3.17}$$

Isolando o ponto fantasma $T_{i+1,j,k}$ obtém-se:

$$T_{i+1,j,k} = \frac{2}{k} f(x_i, y_j, z_k)\Delta x + T_{i-1,j,k} \tag{3.18}$$

Foi escolhida a diferença centrada para aproximar o ponto fantasma, garantindo-se

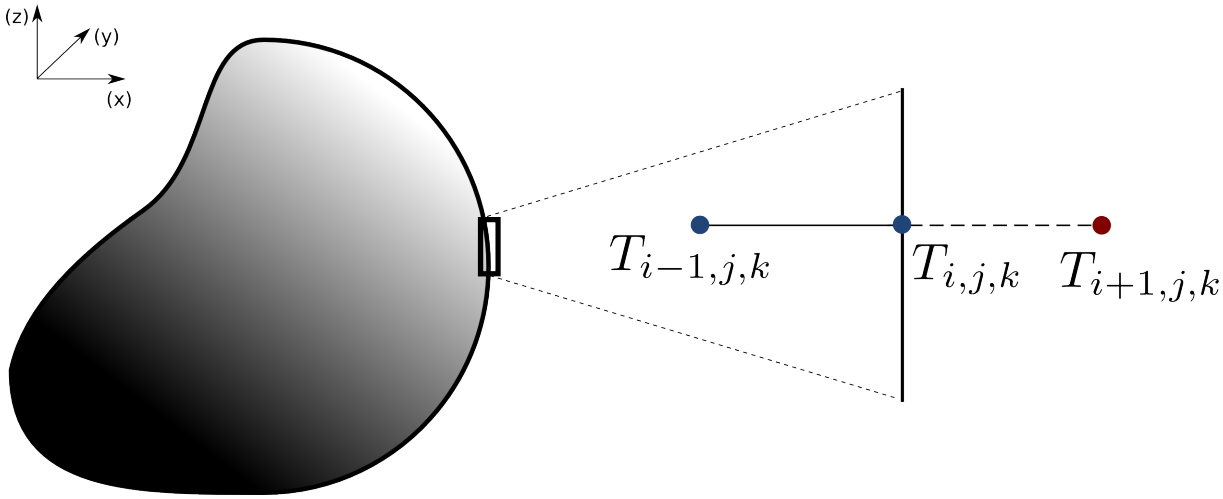


Figura 3.5: Domínio discretizado com o ponto fantasma.

assim a mesma convergência do método numérico no espaço, ou seja, segunda ordem (Tab. A.1).

Então, para aplicar o esquema de diferença finita dada pela Eq. (3.14) em uma equação que possua condição de Neumann, deve-se utilizar esta técnica em todas as direções do contorno do domínio de forma análoga ao caso unidimensional, ou seja, criando-se pontos fantasmas em todo o contorno do domínio.

3.2 A Discretização no Tempo

Neste trabalho foram implementadas duas versões de aproximação da marcha no tempo por meio de uma estratégia explícita. O domínio do tempo I é particionado em N intervalos de tempo iguais com tamanho Δt, *i.e.*, $(0, t_f] = \cup_{n=0}^{N-1}[t_n, t_{n+1}]$. A primeira versão origina-se da aplicação da diferença progressiva, porém com convergência linear. E a segunda versão é uma adaptação da família-alfa trapezoidal generalizada de métodos de integração (Hughes, 2000), que garante convergência quadrática para uma equação linear.

3.2.1 Aplicação da Diferença Progressiva

Pode-se aproximar a derivada temporal no ponto n utilizando a diferença progressiva (A.1), conforme ilustrado pela Eq.(3.19):

$$\frac{\partial T_{i,j,k}}{\partial t} \approx \frac{T_{i,j,k}^{n+1} - T_{i,j,k}^{n}}{\Delta t} \tag{3.19}$$

Deseja-se obter uma relação onde os valores da derivada no próximo instante de tempo dependam somente dos valores do passo anterior, conforme ilustrado no caso unidimensional pela Fig. 3.6.

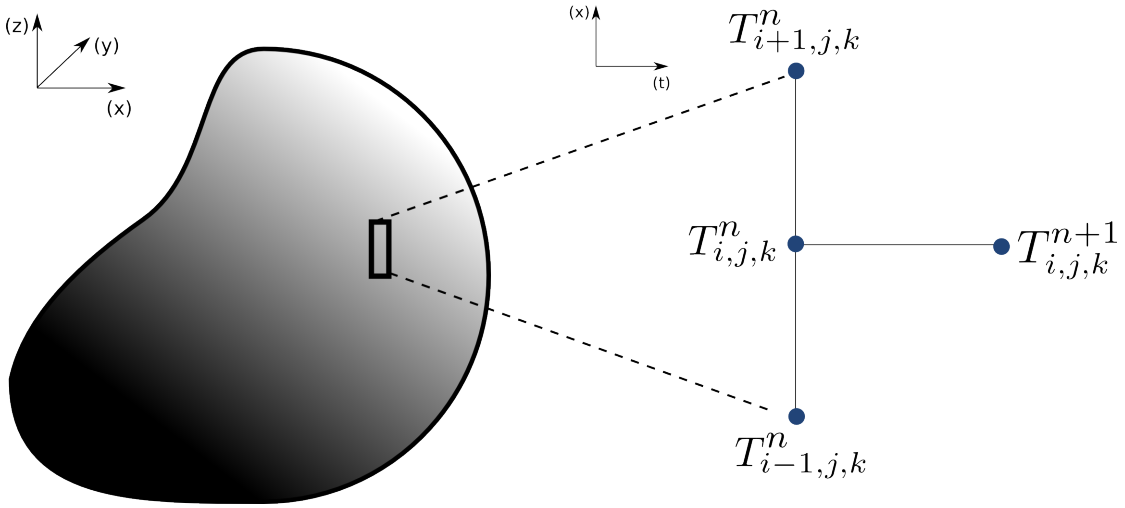


Figura 3.6: *Stencil* e primeira ordem no tempo.

Para obter uma relação explícita análoga a da Fig. 3.6 na marcha do tempo, pode-se aplicar a aproximação na derivada temporal, dada pela Eq. (3.19), na discretização espacial da Eq. (3.14), resultando assim na Eq. (3.20):

$$\begin{aligned} T_{i,j,k}^{n+1} = T_{i,j,k}^{n} + \frac{\Delta t}{\rho_{i,j,k} c_{i,j,k}} \Big[& \Phi(T_{i,j,k}^{n}, T_{i\pm1,j,k}^{n}, T_{i,j\pm1,k}^{n}, T_{i,j,k\pm1}^{n}) \\ & + (\omega_b(T))_{i,j,k}^{n} (\rho_b)_{i,j,k}^{n} (c_b)_{i,j,k}^{n} T_a + (Q_m)_{i,j,k}^{n} + (Q_r)_{i,j,k}^{n} \Big] \end{aligned} \tag{3.20}$$

3.2.2 Aplicação do Preditor-Corretor

Além do esquema utilizando diferença progressiva, pode-se aplicar uma segunda estratégia na marcha do tempo para encontrar a solução aproximada do modelo matemático. Este método é derivado da família de métodos trapezoidal generalizado. Ao aplicá-lo na

Eq. (2.23) para calcular uma aproximação no passo de tempo $n+1$, obtém-se a Eq. (3.21). Além disto, fazendo uma aproximação por série de Taylor (conforme Apêndice B) na marcha do tempo, obtém-se a Eq. (3.22), onde o termo $\frac{\partial T_{i,j,k}^{n+\alpha}}{\partial t}$ pode ser aproximado por uma ponderação com base nos respectivos valores nos passos de tempo n e $n+1$, conforme a Eq. (3.23), onde $\alpha \in [0,1]$. Esta aproximação foi adaptada de Hughes (2000, p.460). Alguns métodos membros da família trapezoidal generalizada são dados na Tab. 3.1.

$$\begin{gathered}\rho_{i,j,k}c_{i,j,k}\frac{\partial T_{i,j,k}^{n+1}}{\partial t} - \Phi(T_{i,j,k}^{n+1}, T_{i\pm1,j,k}^{n+1}, T_{i,j\pm1,k}^{n+1}, T_{i,j,k\pm1}^{n+1}) - \\ (\omega_b(T))_{i,j,k}^{n+1}(\rho_b)_{i,j,k}^{n+1}(c_b)_{i,j,k}^{n+1}T_a = (Q_m)_{i,j,k}^{n+1} + (Q_r)_{i,j,k}^{n+1}\end{gathered} \tag{3.21}$$

$$T_{i,j,k}^{n+1} = T_{i,j,k}^{n} + \Delta t\frac{\partial T_{i,j,k}^{n+\alpha}}{\partial t} \tag{3.22}$$

$$\frac{\partial T_{i,j,k}^{n+\alpha}}{\partial t} = (1-\alpha)\frac{\partial T_{i,j,k}^{n}}{\partial t} + \alpha\frac{\partial T_{i,j,k}^{n+1}}{\partial t} \tag{3.23}$$

Tabela 3.1: Métodos membros da família trapezoidal generalizada.

α	**Método**
0	Diferença progressiva; Euler explícito
0.5	Regra do trapézio; Crank-Nicolson
1	Diferença regressiva; Euler implícito

Da forma que este algoritmo está definido, se $\alpha \neq 0$ obtém-se um método implícito. Uma vez que o operador Φ tem um termo não linear (a perfusão sanguínea $\omega_b(T)$), obtém-se um sistema de equações não lineares. Uma possível saída para não necessitar resolver um sistema não linear consiste em encontrar uma aproximação para o valor de $T_{i,j,k}^{n+1}$, com base nos valores em $T_{i,j,k}^{n}$, preditando o valor, conforme a Eq. (3.24).

$$\widetilde{T}_{i,j,k}^{n+1} = T_{i,j,k}^{n} + (1-\alpha)\Delta t\frac{\partial T_{i,j,k}^{n}}{\partial t}. \tag{3.24}$$

Vale notar que as Eqs. (3.22) e (3.23) podem ser expressas em termos da Eq. (3.24):

$$T_{i,j,k}^{n+1} = \widetilde{T}_{i,j,k}^{n+1} + \alpha\Delta t\frac{\partial T_{i,j,k}^{n+1}}{\partial t}. \tag{3.25}$$

Tomando a Eq. (3.24) como aproximação inicial $\widetilde{T}_{i,j,k}^{n+1} = T_{i,j,k}^{n+1,0}$ e iterando-se r_f vezes

com a Eq. (3.25), conforme ilustrado pelo Pseudocódigo 1, obtém-se um método do tipo preditor-corretor (Hughes, 2000, p.476). Deste modo, evita-se a resolução de um sistema não linear, ou linear (dependendo do parâmetro de perfusão sanguínea adotado).

Pseudocódigo 1: Algoritmo do Preditor-Corretor.

início
  para cada $n \in I_t$ **faça**
    para $r \leftarrow 0$ ***até*** r_f **faça**
      para cada $T \in \Omega \cup \Gamma$ **faça**
        se $(r = 0)$ **então**
          $T_{i,j,k}^{n+1,r} = T_{i,j,k}^{n} + (1-\alpha)\Delta t \frac{\partial T_{i,j,k}^{n}}{\partial t}$
        fim se
        $\frac{\partial T_{i,j,k}^{n+1,r+1}}{\partial t} = \frac{1}{\rho_{i,j,k} c_{i,j,k}} \Big[\Phi(T_{i,j,k}^{n+1,r}, T_{i\pm1,j,k}^{n+1,r}, T_{i,j\pm1,k}^{n+1,r}, T_{i,j,k\pm1}^{n+1,r}) - \omega_{bi,j,k} \rho_{bi,j,k} c_{bi,j,k} T_a + (Q_m)_{i,j,k}^{n} + (Q_r)_{i,j,k}^{n} \Big]$
        $T_{i,j,k}^{n+1,r+1} = T_{i,j,k}^{n+1,0} + \alpha \Delta t \frac{\partial T_{i,j,k}^{n+1,r+1}}{\partial t}$
      fim para cada
      Checar erro : $\sqrt{\frac{\sum (T_{i,j,k}^{n+1,r+1} - T_{i,j,k}^{n+1,r})^2}{\sum (T_{i,j,k}^{n+1,r+1})^2}}$;
    fim para
  fim para cada
fim

De acordo com Hughes (2000, p.479), este algoritmo garante convergência de segunda ordem (demonstrado para o caso linear) tomando $r_f = 2$ e $\alpha = \frac{1}{2}$, diferente do método de Euler empregado na seção 3.2.1 que possui convergência linear. Além disso, para o caso de problemas não lineares, o erro pode ser checado a cada passo de tempo por meio da norma de iterações subsequentes.

Vale enfatizar que este método não é incondicionalmente estável, e segue os mesmo critérios de estabilidade do método de Euler, segundo Hughes (2000, p.479) . Devido à aproximação feita pela Eq. (3.24), mesmo tomando o parâmetro $\alpha = \frac{1}{2}$ não se obtém o método de Crank-Nicolson, o qual seria incondicionalmente estável.

3.3 Estabilidade do Método Numérico

O esquema numérico utilizado neste trabalho é um método condicionalmente estável, de modo que um critério de estabilidade deve ser satisfeito para garantir a convergência da solução numérica. Assumindo o caso linear, a análise de von Neumann revela que a estabilidade é definida pela Eq. (3.26), análogo à análise feita no Apêndice C.

$$\Delta t \left(\frac{\delta}{\Delta x^2} + \frac{\delta}{\Delta y^2} + \frac{\delta}{\Delta z^2} + \frac{\eta}{4} \right) \leq \frac{1}{2}, \tag{3.26}$$

onde $\delta = \frac{k}{\rho c}$ e $\eta = \frac{\omega_b \rho_b c_b}{\rho c}$ e, como resultado, o passo de tempo é selecionado levando em consideração o pior caso do meio.

4 A ESTRATÉGIA PARALELA

Nos últimos 60 anos houve um avanço muito grande na ciência, e um dos fatos que mais impulsionaram este rápido avanço foi a computação. Konrad Zuse (1910-1995) foi quem construiu o primeiro computador eletro-mecânico em 1941, mas somente na Segunda Guerra Mundial que começaram a ser construidos computadores com características arquitetônicas semelhantes as dos atuais. A universidade de Harvard e a IBM projetaram o computador Harvard Mark I em 1944; simultaneamente a Marinha dos Estados Unidos desenvolveu o ENIAC, que foi o primeiro computador eletrônico de propósito geral, anunciado em 1946. Desde então um dos focos da ciência passou a ser esta nova máquina. Hoje em dia muitas áreas da ciência são dependentes desta máquina para que novos desenvolvimentos científicos possam ser obtidos.

Como pode ser observado na Fig. 4.1, o número de transistores aumentou muito do ano de 1971 até 2011, e por consequência, também o poder de processamento dos computadores. Mesmo com esse grande avanço computacional ainda existem muitos problemas que não podem ser resolvidos sequencialmente pelos computadores atuais, ou ainda, que demandam muito tempo de computação, o que também torna inviável a sua execução, *e.g.*, modelagem climática, descobrimento de novos medicamentos, análise de dados. Novas estratégias de desenvolvimento de softwares foram então propostas para lidar com este problema, uma delas é a programação paralela. A programação paralela permite que sejam executados simultaneamente cálculos por diferentes núcleos (do inglês *cores*), processadores ou até mesmo computadores.

Sucintamente, a principal ideia da computação paralela é desenvolver algoritmos que realizem várias operações de forma simultânea. Hoje em dia se torna cada vez mais necessária a utilização deste tipo de técnica, visto que grande fatia dos processadores hoje lançados são multinúcleo. Portanto, ao continuar executando nesta arquitetura programas feitos para uma arquitetura anterior de núcleo único, são desperdiçados recursos de hardware e consequentemente reduz-se o desempenho que a aplicação poderia

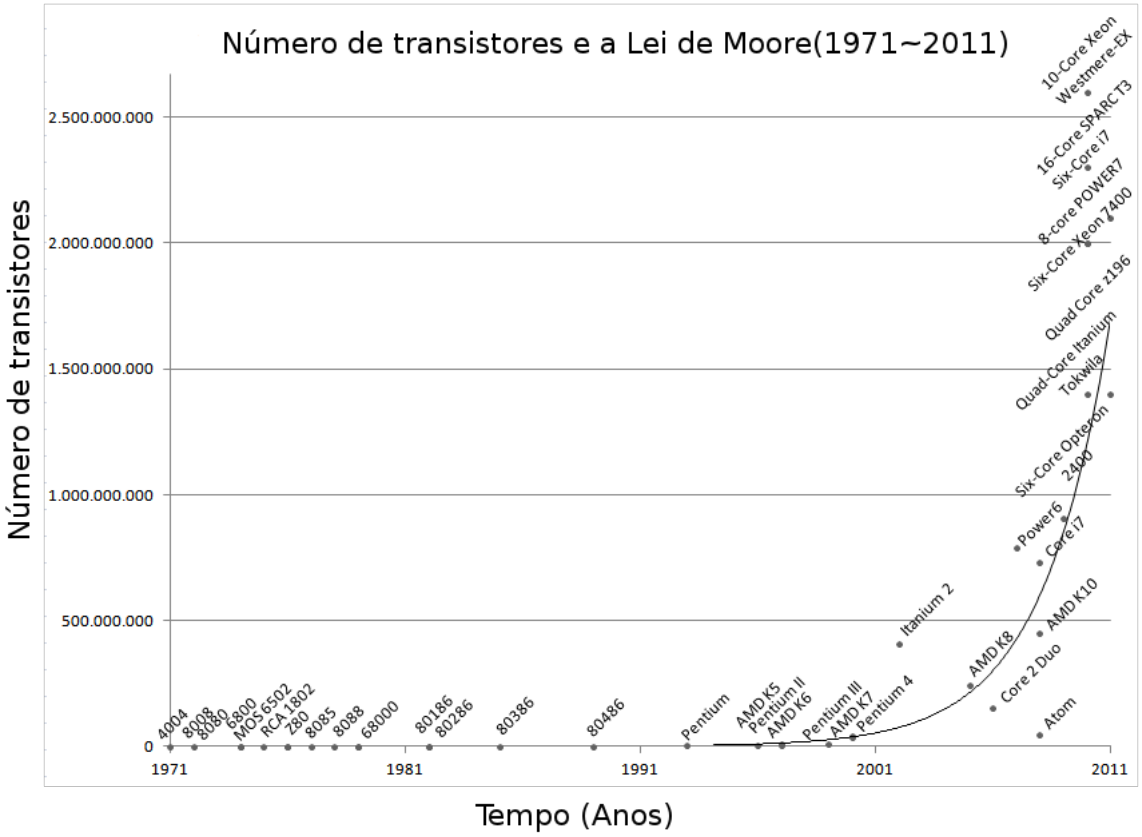


Figura 4.1: Evolução do número de transistores nos computadores de 1971 até 2011. Adaptado de TechnologyPod (2014).

potencialmente obter nesta arquitetura.

Uma das formas de se obter aceleração em processadores multinúcleo é utilizando programação com *threads*. Dentre as técnicas existentes atualmente para programar com *threads*, podem ser citadas duas APIs (*Application Programming Interface*): *POSIX Threads*, ou simplesmente PThreads, e *OpenMP* (*Open Multi-Processing*). Pode-se citar também outra API, *Message Passing Interface* (MPI), mais adequada para programação em multicomputadores, mas que também pode ser empregada em processadores multinúcleo. O MPI é um padrão para comunicação de dados através de processos paralelos (e não de *threads*) e em um ambiente de memória distribuída, diferentemente do ambiente de memória compartilhada provido por processadores multinúcleo. Dentre estas possibilidades existentes para a paralelização, este trabalho escolheu a API de programação paralela *OpenMP*.

Além das estratégias convencionais, utilizadas em unidades centrais de processamento

(CPU), outras estratégias para executar aplicações paralelas em plataformas computacionais distintas, como as placas gráficas, estão se consolidando. Dentre as alternativas disponíveis para programação em placas gráficas, este trabalho foca na utilização da plataforma *Compute Unified Device Architecture*, ou simplesmente CUDA, disponível para as placas da NVIDIA. Através desta plataforma é possível utilizar as Unidades de Processamento Gráfico de Propósito Geral (GPGPUs) disponíveis nas placas gráficas para executar cálculos de forma paralela.

4.1 A API *OpenMP*

A paralelização utilizando *threads* se baseia em um sistema de memória compartilhada de acesso uniforme, ou simplesmente UMA (*Uniform Memory Access*), conforme ilustrado na Fig. 4.2. Nesta arquitetura, o tempo de acesso à memória por cada processador é igual. Caso mais de um processador queira acessar a memória simultaneamente, potencialmente um deles deverá aguardar, visto que muitas vezes a interconexão usada para ligar as CPUs à memória é um barramento, que permite um único acesso por vez. O caso das arquiteturas multinúcleo, ou seja, com processadores que possuem mais de núcleo, é ilustrado na Fig. 4.3. Neste caso, existe adicionalmente uma potencial disputa interna, entre os núcleos, para acessar a memória.

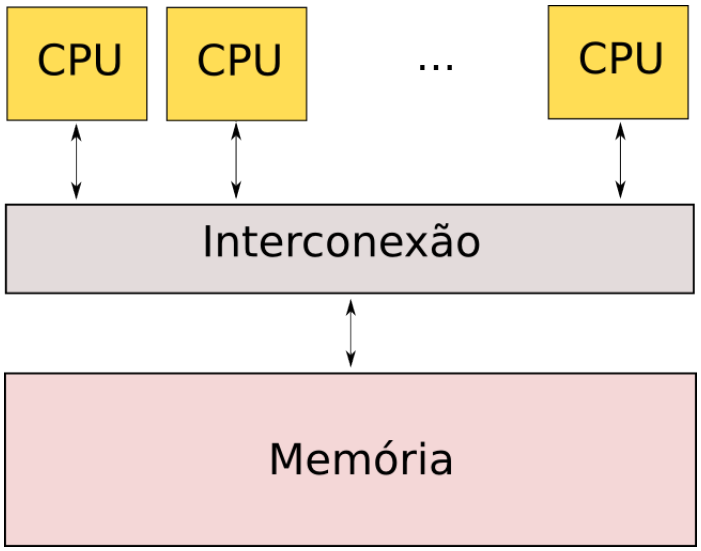


Figura 4.2: Sistema de memória compartilhada. Adaptado de Pacheco (2011, p. 33).

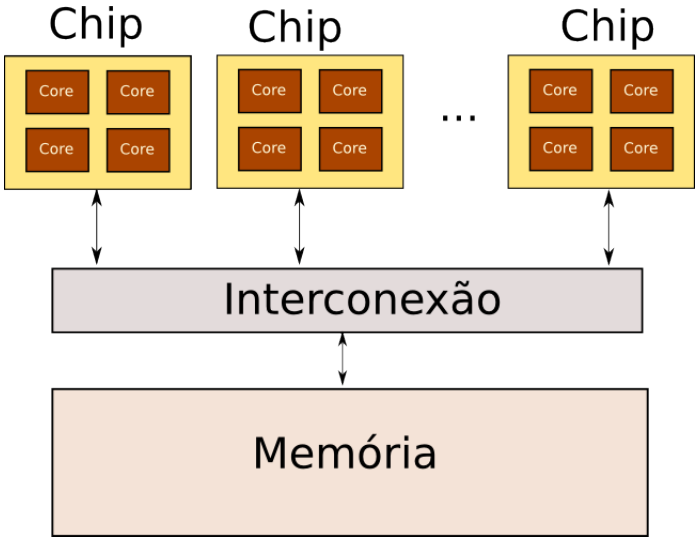


Figura 4.3: Sistema de memória compartilhada multinúcleo. Adaptado de Pacheco (2011, p. 34).

A implementação de *threads* varia de um sistema operacional para outro, mas na maioria dos casos elas estão contidas nos processos. Um processo *multithread* pode ser observado na Figura 4.4.

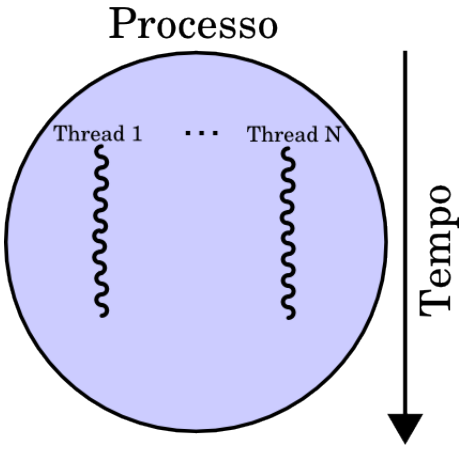


Figura 4.4: Processo *multithread*. Adaptado de Pacheco (2011, p. 18).

A paralelização com *OpenMP* baseia-se no uso de diretivas, que em C e C++ é feita com instruções especiais de pré-processamento chamadas de *pragma*. A diretiva básica para programação paralela com *OpenMP* é *# pragma omp parallel*, que cria *threads* para executarem simultaneamente o bloco de código diretamente abaixo dela.

Outra diretiva básica é a *# pragma omp for*, que realiza a distribuição das iterações de um laço, imediatamente abaixo, do tipo *for* entre as *threads*, para que sejam executadas em paralelo. A divisão das iterações na paralelização do *loop*, ou a forma com que as

threads são escalonadas, é dada pela cláusula *schedule*. Esta cláusula possui a seguinte sintaxe: *schedule (< tipo > [, < tamanho_do_chunk >])*. Os tipos de escalonamentos permitidos pelo *OpenMP* são: escalonamento estático, dinâmico ou guiado, automático, ou em tempo de execução, através dos comandos *static*, *dynamic* ou *guided*, *auto*, ou *runtime*, respectivamente. O escalonamento estático divide as iterações do *loop* de forma prévia, em tempo de compilação; o de forma dinâmica ou guiada divide o conjunto de iterações enquanto as *threads* executam, quando uma delas termina o conjunto de iterações submetido a ela, a mesma solicita ao sistema *runtime* mais trabalho; o automático é determinado pelo compilador e/ou sistema *runtime*; por fim, o escalonamento em tempo de execução é definido por meio de variáveis ambiente do sistema operacional, onde é escolhido um dos tipos previamente definidos. O tamanho do conjunto de iterações a ser atribuído a cada *thread*, conhecido como *chunk*, é opcional, podendo ou não ser definido pelo programador. Caso não seja definido, será usado um valor padrão, que varia de acordo com o tipo de escalonamento utilizado.

4.1.1 Um primeiro exemplo OpenMP

A título de exemplo, tome o algoritmo que executa uma soma vetorial descrito pelo Alg. 4.1.

```
1  #include <sdtio.h>
2
3  int main ()
4  {
5    float x[1000], y[1000], z[1000];
6    for (i=0; i<1000; i++)
7    {
8      z[i] = x[i]+y[i];
9    }
10   return 0;
11 }
```

Algoritmo 4.1: Um exemplo de soma vetorial.

Este exemplo apresenta uma programa que pode ser executado totalmente em paralelo, pois cada iteração deste *loop* pode ser executada independentemente uma da outra. Ao paralelizá-lo usando as técnicas previamente discutidas, obtém-se o Alg. 4.2.

```
1  #include <sdtio.h>
2
3  int main ()
4  {
5    float x[1000], y[1000], z[1000];
6    # pragma omp parallel for
7    for (i=0; i<1000; i++)
8    {
9      z[i] = x[i]+y[i];
10   }
11   return 0;
12 }
```

Algoritmo 4.2: Um exemplo de soma vetorial utilizando *OpenMP*.

Neste exemplo, a diretiva *# pragma omp parallel for* foi utilizada na paralelização. Esta diretiva simultaneamente cria as *threads* e distribui entre elas as iterações do laço de repetição. Por padrão, o *OpenMP* define um escalonamento estático, sendo o número de *threads* criadas em geral igual ao número de núcleos que a máquina possui, e o tamanho do *chunk* igual a $\frac{N}{numero_de_threads}$. Se for de interesse do programador alterar o tipo de escalonamento e/ou o tamanho do *chunk*, o código poderia ficar como o Alg. 4.3.

```
1  #include <sdtio.h>
2
3  int main ()
4  {
5    float x[1000], y[1000], z[1000];
6
7    # pragma omp parallel for schedule (dynamic, 2)
8    for (i=0; i<1000; i++)
9    {
10     z[i] = x[i]+y[i];
11   }
12   return 0;
13 }
```

Algoritmo 4.3: Um exemplo de soma vetorial utilizando *OpenMP*

No caso do Alg. 4.3, foi escolhido um escalonamento dinâmico com *chunk* de tamanho 2, ou seja, cada *thread* recebe como tarefa 2 iterações do *loop* e ao terminar sua execução, solicita mais 2 iterações ao sistema *runtime*.

4.1.2 Paralelização do Método Numérico

Para se obter a solução numérica da Eq. (2.23) com a API de programação paralela *OpenMP*, foi utilizado um esquema de diferença finita espacial descrito pela Eq. (3.14) e

a marcha no tempo dada pela Eq. (3.20). Este método numérico pode ser implementado de forma paralela conforme o pseudocódigo 2. A variável T é implementada por meio de uma matriz quadridimensional I_x x I_y x I_z x 2, onde I_x, I_y e I_z são os números de subdivisões do domínio nos eixos x, y e z, respectivamente. Para evitar a cópia de dados entre iterações sucessivas da marcha de tempo, adotou-se a criação da quarta dimensão na matriz T, *i.e.*, ao terminar de executar o cálculo em todos os pontos do domínio espacial, troca-se o valor das variáveis auxiliares *nova* e *antigo* com uma operação de negação unária; então o que era 1 se torna 0, e vice-versa.

Pseudocódigo 2: Implementação em paralelo com *OpenMP* do MDF explícito com Euler na marcha do tempo.

```
1  início
2      nova ← 1
3      antiga ← 0
4      # pragma omp parallel
5      para t ← t_i até t_f faça
6          #pragma omp for
7          para i ← x_i até x_f faça
8              para j ← y_i até y_f faça
9                  para k ← z_i até z_f faça
10                     T^nova_{i,j,k} = (h_t / (ρ_{i,j,k} c_{i,j,k})) [ Φ(T^antiga_{i,j,k}, T^antiga_{i±1,j,k}, T^antiga_{i,j±1,k}, T^antiga_{i,j,k±1}) −
                       ω_{bi,j,k} ρ_{bi,j,k} c_{bi,j,k} T_a + (Q_m)^n_{i,j,k} + (Q_r)^n_{i,j,k} ] + T^antiga_{i,j,k}
11                 fim para
12             fim para
13         fim para
14         nova ← não nova
15         antiga ← não antiga
16     fim para
17 fim
```

Vale destacar que o laço temporal não foi paralelizado, pois existe dependência de dados entre passos de tempos sucessivos, *i.e.*, para calcular a aproximação da solução no passo de tempo $t+1$ necessita-se dos valores calculados no passo de tempo t. Além disso, as *threads* são criadas na linha 4 para evitar o *overhead* de criação de *threads* a cada passo de tempo. Somente na linha 6 é feita a divisão de tarefas entre as *threads* previamente criadas. O comando *#pragma omp for* aparece imediatamente antes do laço do eixo x,

portanto a divisão de tarefas é feita somente neste eixo.

4.2 A Plataforma NVIDIA CUDA

Inicialmente, as GPUs foram criadas para acelerar gráficos e suportavam somente algumas funções prefixadas, sendo utilizadas principalmente para melhoria na execução de jogos. Alguns programadores começaram a enxergar a possibilidade de uso do poder de processamento deste *hardware* para outros fins, por exemplo, cálculos científicos. Inicialmente, tais cálculos eram realizados de uma forma pouco intuitiva, sendo necessário representá-los por meio de operações com triângulos e polígonos. Como tentativa inicial de tornar a programação de propósito geral mais simples em GPU, um grupo de pesquisadores liderado por Ian Buck desenvolveu um modelo de programação que estendia a linguagem de programação C com construções de paralelismo de dados. Uma de suas versões foi lançada para GPUs. Ao saber disto, a NVIDIA convidou este pesquisador para trabalhar em um projeto que desenvolveu a plataforma de programação paralela CUDA, lançada em 2006. Hoje em dia é possível, através desta plataforma, desenvolver códigos em linguagens como C/C++ e Fortran para serem executados diretamente na GPU (NVIDIA, 2014b).

Um comparativo entre a GPU e a CPU em relação ao número de operações de ponto flutuante por segundo e largura de banda de memória pode ser visto nas Figs. 4.5 e 4.6, respectivamente.

As GPUs oferecem um número massivo de núcleos, isto se deve ao fato de que os núcleos de processamento das GPUs são relativamente menores, quando comparados aos de CPUs. Nos núcleos das placas gráficas não é empregado nem um amplo *Instruction Set*, comparado a uma CPU, nem algoritmos de previsão de desvios condicionais, deixando a estrutura de controle mais simples. Esta característica deixa os núcleos menores, possibilitando a inserção de um número elevado de unidades de processamento na mesma placa (na ordem de centenas, ou até mesmo milhares, dependendo do modelo).

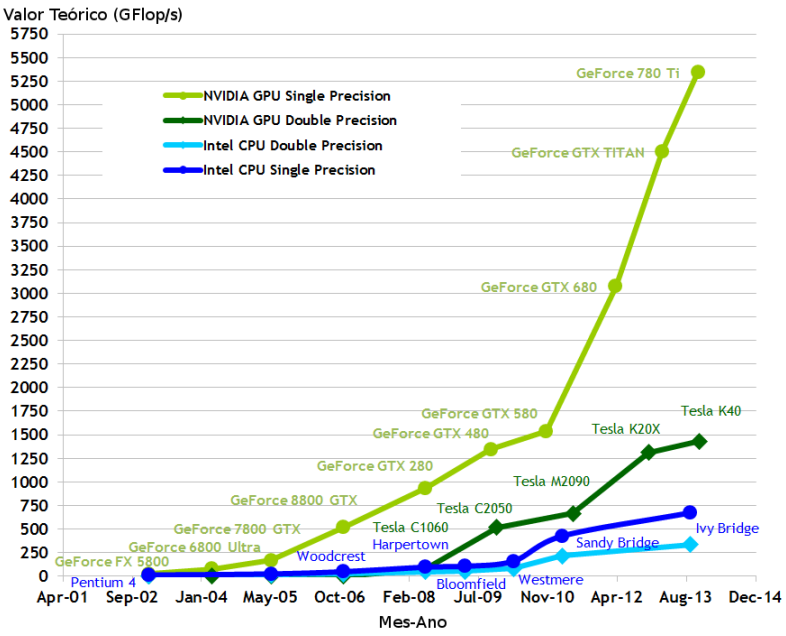


Figura 4.5: Número de operações com ponto flutuante comaprando CPU e GPU. Adaptado de NVIDIA (2014c).

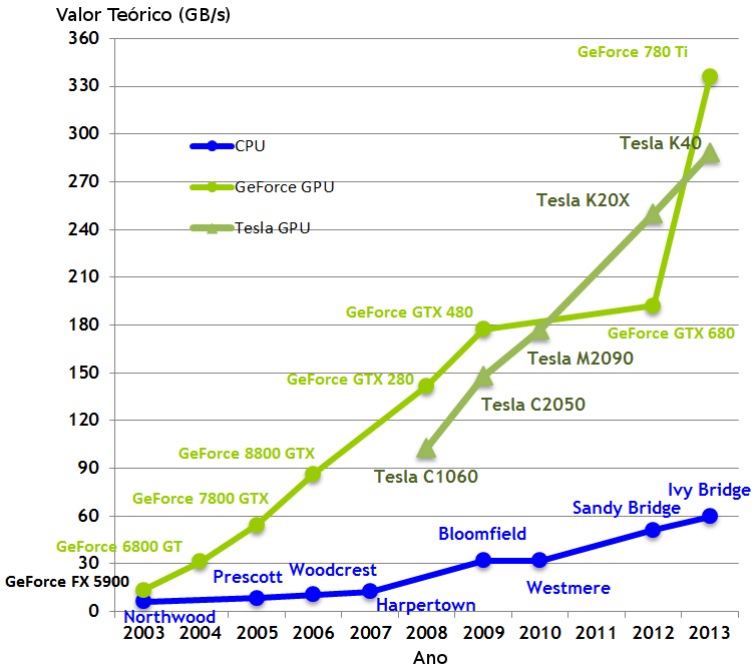


Figura 4.6: Banda de memória comparando CPU e GPU. Adaptado de NVIDIA (2014c).

4.2.1 A extensão CUDA C

A execução de códigos em uma placa gráfica da NVIDIA pode ser feita por meio de uma extensão da linguagem C, através da definição de funções conhecidas por *kernels*. Uma

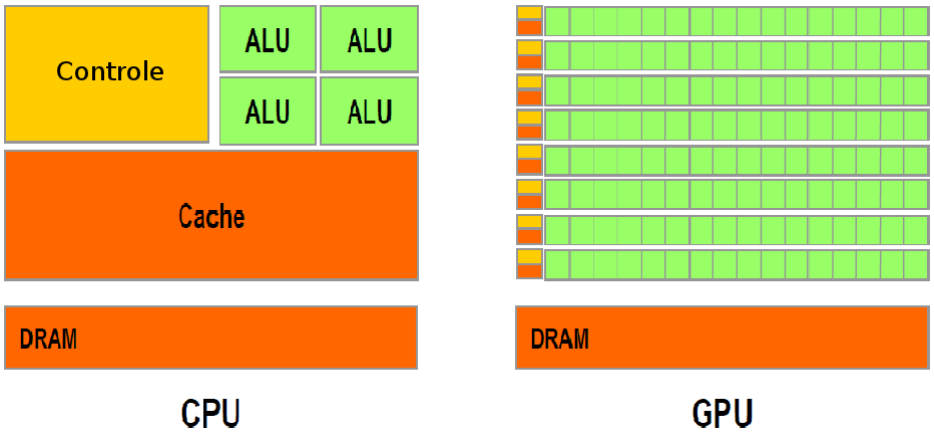


Figura 4.7: Estrutura da CPU e GPU. Adaptado de NVIDIA (2014c).

típica chamada de *kernel* pode ser exemplificada como:

```
1    myKernel<<<dimGrid, dimBlock>>>();
```

Um *kernel* é executado dentro de uma GPU por meio de *threads*, estas *threads* são alocadas por meio de grades e blocos. Uma grade é uma estrutura que aloca os blocos, os blocos são estruturas formadas por *threads*, *i.e.* cada nó da grade contém um bloco, onde cada nó do bloco é composto por uma *thread*. Nas atuais versões do *driver* CUDA permite-se a criação tanto de uma grade, quanto de um bloco tridimensional, conforme pode ser observado na Fig. 4.8. Deste modo, diferente da chamada de função padrão C, ao chamar um *kernel* deve-se informar a configuração de alocação das *threads* por meio dos parâmetros obrigatórios *dimGrid* e *dimBlock*, que especificam as dimensões da grade e dos blocos.

Uma GPU da NVIDIA basicamente possui a memória individualmente acessada pelas *threads*, chamada memória local. Além disso, também possui uma memória compartilhada, que é acessível às unidades de processamento de um mesmo *Streaming Multiprocessor*[1], e, por fim, as memórias de texturas, constantes e global, que podem ser acessadas por todas as *threads*.

Conforme dito anteriormente, ao executar um *kernel* são criadas várias *threads*, e, para melhor desempenho, preferencialmente executando a mesma instrução sobre dados

[1]Cada *Streaming Multiprocessor* (SM) é um vetor de núcleos CUDA (conhecidos também por CUDA *cores*), além de possuir unidades de funções especiais, memória compartilhada, cache dentre outros componentes

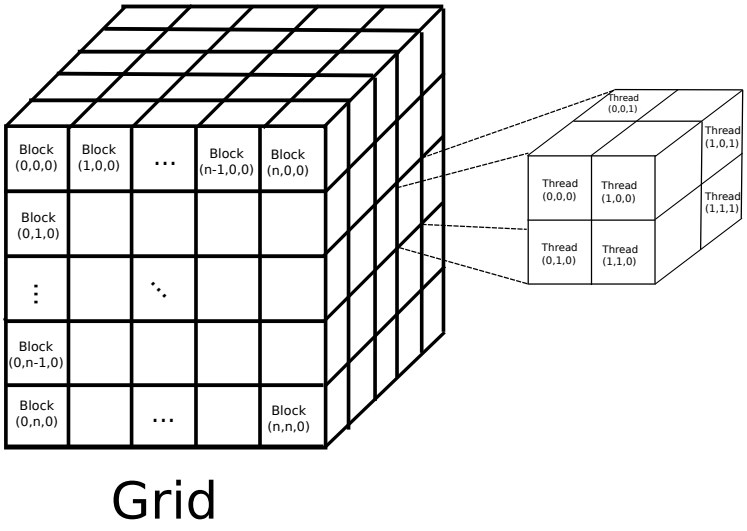


Figura 4.8: Organização das *threads* em blocos de grades com a plataforma CUDA.

distintos, ou seja, seguindo o modelo *Single Instruction Multiple Data* (SIMD). Cada *thread* é identificada unicamente pelo par ordenado *(id_bloco, id_thread)*. Um recurso muito utilizado é o programador mapear o acesso à memória global através dessa identificação.

O escalonamento das *threads* é realizado por meio de *warps*[2], assim dentro de um SM a execução ocorre em *warps* de 32 *threads*. Segundo a NVIDIA o escalonamento de uma *warp* é feito com aproximadamente zero de *overhead* dentro do hardware.

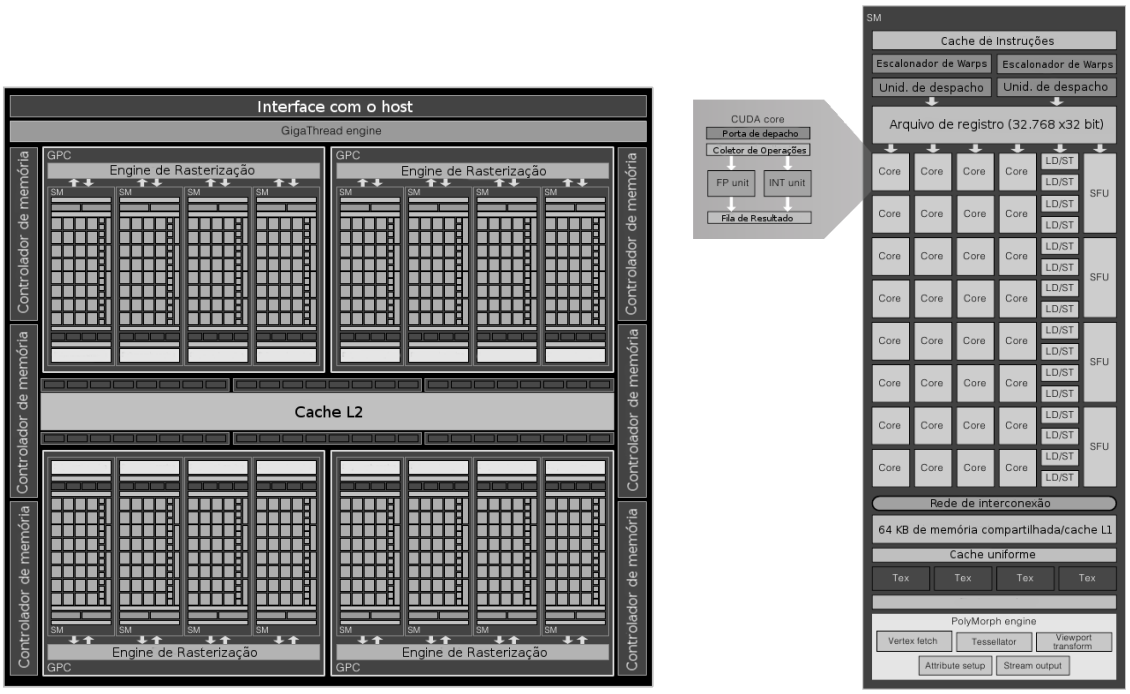
4.2.2 A arquitetura de uma GPU da geração NVIDIA Fermi

As arquiteturas de GPU já lançadas pela NVIDIA são: Tesla, Fermi, Kepler e Maxwell. Neste estudo utilizou-se a placa Tesla M2075, uma placa de vídeo da geração Fermi; por esse motivo esta seção detalhará a arquitetura desta geração.

A Fig. 4.9(a) mostra uma visão geral da arquitetura de uma placa GF100, da geração Fermi, que contêm 3 bilhões de transistores. Cada *streaming multiprocessor* (SM), ilustrado na Fig. 4.9(b), contém 32 núcleos CUDA. A estrutura de um núcleo CUDA desta geração está destacado na Fig. 4.9(b), e conforme ilustrado nesta figura, cada núcleo possui

[2] *Warp* é a unidade básica de escalonamento na GPU.

uma unidade de ponto flutuante e uma de inteiros. Em arquiteturas de GPUs anteriores era usado o padrão IEEE 754-1985 de ponto flutuante; a arquitetura desta geração utiliza o padrão IEEE 754-2008, e oferece instruções de adição e multiplicação para precisão simples quanto para dupla.



(a) Visão geral da placa

(b) Destaque para o *Streaming Multiprocessor*

Figura 4.9: *Datasheet* de uma GPU da arquitetura Fermi. Adaptado de Wittenbrink *et al.* (2011).

No caso da placa Tesla M2075, vale destacar que esta possui 14 SM de 32 núcleos CUDA cada, em um total de 448 núcleos CUDA; a frequencia de *clock* de cada núcleo é de $1,15$ GHz. Além disto, esta placa possui uma memória global de 6GB GDDR5 de $1566,00$ Mhz com suporte a ECC[3]. Quanto ao escalonamento das *threads*, o tamanho da *warp* é 32, ou seja, elas são escalonadas de 32 em 32 em cada SM. O número máximo de *threads* que esta placa permite em um mesmo bloco é de 1024.

[3] *Error Checking & Correction*: é um tipo de armazenamento de dados em computador que pode detectar e corrigir os tipos mais comuns de erro interno de corrupção de dados.

4.2.3 Um primeiro exemplo CUDA

A título de exemplo, retomando a soma vetorial serial apresentada no Alg. 4.1, pode-se implementar em CUDA a soma vetorial conforme apresentado no Alg. 4.4.

```
1  #include<iostream>
2
3  #define N 1000
4   __global__ void vectorAdd ( const int *dev_a , const int *dev_b , int *
       dev_c)
5  {
6
7        unsigned short tid = threadIdx.x ;
8
9        if ( tid < N )
10             dev_c [tid] = dev_a[tid] + dev_b[tid] ;
11
12 }
13 int main()
14 {
15   int host_a[N], host_b[N], host_c[N];
16
17   int *dev_a , *dev_b, *dev_c ;
18   cudaMalloc((void **)&dev_a , N*sizeof(int) );
19   cudaMalloc((void **)&dev_b , N*sizeof(int) );
20   cudaMalloc((void **)&dev_c , N*sizeof(int) );
21
22   for(int i = 0; i < N; i++ )
23   {
24       host_a[i] = sin(i)*sin(i);
25       host_b[i] = cos(i)*cos(i);
26   }
27
28   cudaMemcpy (dev_a , host_a , N*sizeof(int) , cudaMemcpyHostToDevice);
29   cudaMemcpy (dev_b , host_b , N*sizeof(int) , cudaMemcpyHostToDevice);
30
31   vectorAdd <<< 1, N >>> (dev_a , dev_b , dev_c ) ;
32
33   cudaMemcpy(Host_c , dev_c , N*sizeof(int) , cudaMemcpyDeviceToHost);
34
35   cudaFree (dev_a) ;
36   cudaFree (dev_b) ;
37   cudaFree (dev_c) ;
38
39   return 0;
40 }
```

Algoritmo 4.4: Um exemplo de soma vetorial utilizando CUDA.

Note que ao utilizar essa plataforma, deve-se tratar a CPU e a GPU como duas unidades diferentes. Em CUDA elas são chamadas de *host* e *device*, respectivamente. Neste exemplo da soma vetorial, primeiramente aloca-se espaço na memória tanto para o *host* (linha 15) quanto para o *device* (linhas 18-20). Em seguida é executado o

preenchimento dos valores iniciais dos vetores *host_a* e *host_b* dentro do *host*. Em seguida tais valores são copiados para os vetores correspondentes no *device*; contudo, como são dois espaços de endereçamento diferentes, usa-se a função *cudaMemcpy* para realizar a cópia (linhas 28-29). Então, tendo os vetores alocados e preenchidos, na linha 31 ocorre a chamada do *kernel vectorAdd*. Para isto, deve-se informar de que maneira serão criadas as *threads*, neste caso foi criado um *grid* de tamanho 1, contendo um bloco de tamanho 1000. A principal ideia que se segue na execução de um código dentro da GPU é dividir o trabalho entre as *threads* de forma que cada uma seja responsável por computar um passo do laço, ou o mais próximo possível disso. Neste exemplo, cada *thread* executa a soma de um elemento do vetor. Depois da execução da soma vetorial na GPU, o resultado deve ser trazido para o *host* também por meio da função *cudaMemcpy*. Por fim, a liberação da memória do *device* é feita pela função *cudaFree*.

4.2.4 Paralelização do Método Numérico

A versão em paralelo desenvolvida para ser executada na plataforma CUDA foi baseada no método numérico descrito pela discretização espacial dado pela Eq. (3.14) e o esquema explícito do tipo preditor-corretor descrito pelo Pseudocódigo 1.

A paralelização realizada em CUDA pode ser dividida em duas partes principais, a executada na CPU e a executada na GPU. A primeira está descrita no Pseudocódigo 3. Neste pseudocódigo é importante destacar a linha 7, onde acontece a chamada ao *kernel* que será executado pela GPU. Conforme previamente comentado deve-se passar, além dos parâmetros da função, também as configurações de execução na GPU, neste caso à dimensão da grade e dimensão do bloco.

O *kernel* descrito pelo Pseudocódigo 4 aplica em todo o domínio o *stencil* espacial em uma iteração do método numérico baseado no esquema preditor-corretor. Neste ponto vale destacar a necessidade de sincronização das *threads* feita na linha 8 do Pseudocódigo 3. Devido a dependência de dados existente entre as iteração do algoritmo preditor-corretor, deve ser feita uma sincronização na CPU a cada chamada do *kernel*. Com esta sincronização se garante que cada execução do *kernel* esteja calculado os valores em cada

ponto do domínio com os dados gerados na iteração predecessora.

Pseudocódigo 3: Parte executada na CPU na paralelização do MDF com preditor-corretor utilizando a plataforma CUDA.

```
1  início
2  |  nova ← 1
3  |  antiga ← 0
4  |  copiar dados da CPU para a GPU
5  |  definição de tamanho de bloco e grade
6  |  para t ← t_n até t_f faça
7  |  |  para r ← 0 até r_f faça
8  |  |  |  computeStencil<<<dimGrade,dimBloco>>>()
9  |  |  |  cudaThreadSynchronize()
10 |  |  |  nova ← não nova
11 |  |  |  antiga ← não antiga
12 |  |  fim para
13 |  fim para
14 |  copiar os resultados da GPU para a CPU
15 fim
```

Pseudocódigo 4: Parte executada na GPU (*kernel computeStencil) na paralelização do MDF utilizando a plataforma CUDA.*

```
1  início
2  |  i ← idBloco.x * dimBloco.x + idThread.x
3  |  j ← idBloco.y * dimBloco.y + idThread.y
4  |  k ← idBloco.z * dimBloco.z + idThread.z
5  |  se r = 0 então
6  |  |  T^{n+1,r}_{i,j,k} ← T^n_{i,j,k} + (Δt/2) ∂T^n_{i,j,k}/∂t
7  |  fim se
8  |  calcular ∂T^{n+1,r+1}_{i,j,k}/∂t
9  |  T^{n+1,r+1}_{i,j,k} ← T^{n+1,0}_{i,j,k} + (Δt/2) ∂T^{n+1,r+1}_{i,j,k}/∂t
10 fim
```

Cada vez que o Pseudocódigo 4 é chamado pela CPU são criadas I_x x I_y x I_z *threads*, cada uma calculando uma posição do *stencil* de acordo com sua respectiva posição na grade e no bloco. Vale destacar que neste algoritmo r é a iteração atual do preditor-corretor.

É importante enfatizar que foram implementadas todas as otimizações usuais necessárias para adequar a versão paralela do código para a arquitetura CUDA. Mais especificamente, sempre que possível, estruturas condicionais **if** foram trocadas por

operadores ternários; macros e funções *inline* foram utilizadas, ao invés de funções tradicionais; dados que são acessados mais de uma vez são armazenados nas memórias locais de cada *thread*, de modo a reduzir o uso de banda de memória.

Para maximizar a ocupação dos *stream multiprocessors*, os valores ótimos para o tamanho do bloco e da grade foram calculados com o auxílio da CUDA *Occupancy Calculator* NVIDIA (2014a). A maximização da ocupação ajuda a mascarar a latência durante algumas operações de acesso à memória. Contudo, não é possível usar os valores sugeridos pela CUDA *Occupancy Calculator* diretamente no código: alguns valores calculados para tamanho de grade e de bloco induzem a criação de mais *threads* do que o mínimo necessário para executar os cálculos. Foram então escolhidas as dimensões de bloco de forma a minimizar o valor total de *threads*, ao mesmo tempo que maximizando a ocupação. Para minimizar o número de *threads* criadas, foi desenvolvido um aplicativo auxiliar, conforme o Alg. 5. Este combina todas as possíveis dimensões do bloco que totalizam o valor dado pela CUDA *Occupancy Calculator* e apresenta as dimensões da grade e o total de *threads* que seriam criadas em cada caso. Os valores fornecidos por esta aplicação auxiliar ajudou na escolha das dimensões mínimas que produzem os melhores resultados.

Pseudocódigo 5: Aplicativo auxiliar para ajudar a escolher as dimensões do bloco.

```
1  início
2      para x ← 0 até X_f faça
3          para y ← 0 até Y_f faça
4              para z ← 0 até Z_f faça
5                  gx = (I_x+x-1)/x;
6                  gy = (I_y+y-1)/y;
7                  gz = (I_z+z-1)/z;
8                  threadsBloco = x*y*z;
9                  se threadBloco ∈ valoresCompativeis então
10                     imprime(gx, gy, gz, x, y, z, threads, gx*gy*gz*x*y*z);
11                 fim se
12             fim para
13         fim para
14     fim para
15 fim
```

5 RESULTADOS

Este trabalho foi dividido em 3 estudos de caso, os dois primeiros estudos são simulações considerando somente o tecido constituído por tumor e músculo, e o terceiro considerando pele, gordura, músculo e tumor. O primeiro e o segundo estudos diferem na função utilizada para modelar a aplicação das nanopartículas. Vale enfatizar que a morte de tecidos saudáveis nas proximidades do tumor é factível de acontecer, mas deve tentar diminui-la ao máximo, evitando assim os danos à saúde do paciente. O posicionamento dos pontos de injeção em todos os estudos foi feito de maneira qualitativa.

Além disso, este trabalho utilizou estratégias de computação de alto desempenho para acelerar o tempo de execução das simulações. Os esquemas de computação paralela aplicados foram baseadas na API de programação paralela *OpenMP* e na plataforma CUDA.

5.1 Estudo 1

Neste estudo de caso, publicado em Reis *et al.* (2014b), foram realizadas um conjunto de simulações de tratamento de câncer com hipertermia através de nanopartículas magnéticas, onde se considerou um tumor inserido na camada muscular. Além disto, foi utilizado o método numérico dado pela Eq. (3.20) para aproximar a solução do modelo matemático descrito pela Eq. (2.23), e considerando a condição inicial $T(\cdot, 0) = 37,00^oC$ em Ω; condição de Dirichlet $T = 37,00^oC$, no plano $x = 0$ e Neumann $\nabla T \cdot \vec{n} = 0,00\ W/m^2$ nos demais.

Quanto à aplicação das nanopartículas utilizou-se o modelo dado pela Eq. (2.18), *i.e.*, $Q_r(\vec{x}) = \sum_{i=1}^{M} A_i.e^{\frac{-r_i^2}{r_{0,i}^2}}$. Este modelo considera somente a dependência espacial do ponto de aplicação, ou seja, a distância r do ponto de aplicação das nanopartículas.

5.1.1 Parâmetros do Modelo

Todas as simulações neste estudo foram executadas em um domínio tridimensional. O formato e localização do tumor pode ser observado na Fig. 5.1. O tumor é formado por duas esferas de raio $0,015m$ e $0,01m$ centradas nos pontos (0,05m; 0,05m; 0,05m) e (0,04m; 0,04m; 0,04m), respectivamente.

Este domínio é dividido em dois tipos de tecido: muscular e tumoral. Considerou-se um domínio cúbico de dimensão 0,1m x 0,1m x 0,1m com 100 subdivisões em cada direção, ou seja, $\Delta x = \Delta y = \Delta z = 0,001$m. Além disto, foi considerada uma simulação de 3.000s dividida em 30.000 intervalos, ou seja, $\Delta t = 0,1$s.

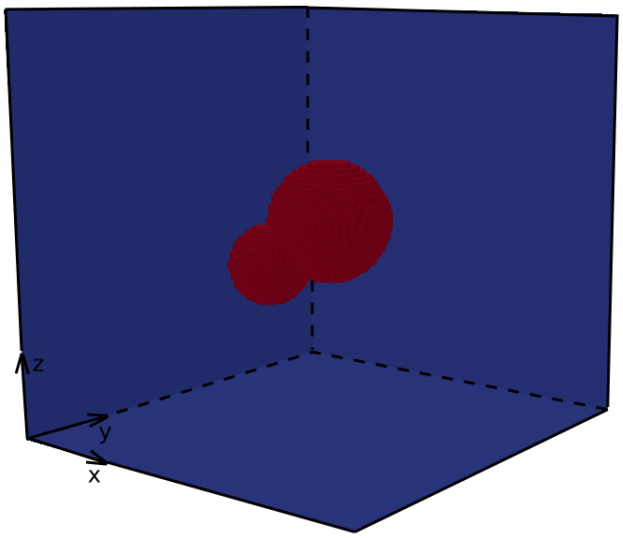


Figura 5.1: Domínio da simulação do estudo da seção 5.1. Destaque para o tumor localizado no centro do cubo.

Cada camada possui parâmetros diferentes, de acordo com as propriedades do tecido vivo. Os valores dos parâmetros são apresentados na Tab. 5.1. Este trabalho considerou o parâmetro de perfusão sanguínea dependente da temperatura, conforme as Eq. (2.20) e (2.22), referentes às perfusões sanguíneas das camadas de músculo e tumor, respectivamente.

Quanto aos pontos de aplicação das nanopartículas, foi considerado 1 ponto de aplicação, conforme a Tab. 5.2.

Tabela 5.1: Parâmetros do modelo no estudo 1.

Símbolo	**Unidade**	**Músculo**	**Tumor**
c	(J/Kg^oC)	$4.200,0$	$4.200,0$
c_b	(J/Kg^oC)	$4.200,0$	$4.200,0$
k	(W/m^oC)	$0,50$	$0,55$
ρ	(Kg/m^3)	$1.000,0$	$1.000,0$
ρ_b	(Kg/m^3)	$1.000,0$	$1.000,0$
ω_b	$(Kg/s/m^3)$	$\omega_{músculo}$	ω_{tumor}

Tabela 5.2: Parâmetros dos pontos de injeção no estudo 1.

Posição (m)	**A**	$\mathbf{r_0}$
$(0,050;\ 0,050;\ 0,050)$	$1,3$ x 10^6 W/m^3	$3,1$ x 10^{-3}m

5.1.2 Simulações do Estudo

A solução da Eq. (2.23) no tempo $t = 3.000$s, utilizando o método descrito pela Eq. (3.20) e o Pseudocódigo 2 em sua implementação, pode ser visualizada na Fig. 5.2. Este gráfico é um mapa da distribuição de temperatura, destacando algumas fatias do domínio tridimensional. Neste instante de tempo já se pode observar o estado estacionário da equação de biotransferência de calor, bem como a distribuição radial de temperatura causada pelo tratamento de hipertermia com as nanopartículas magnéticas.

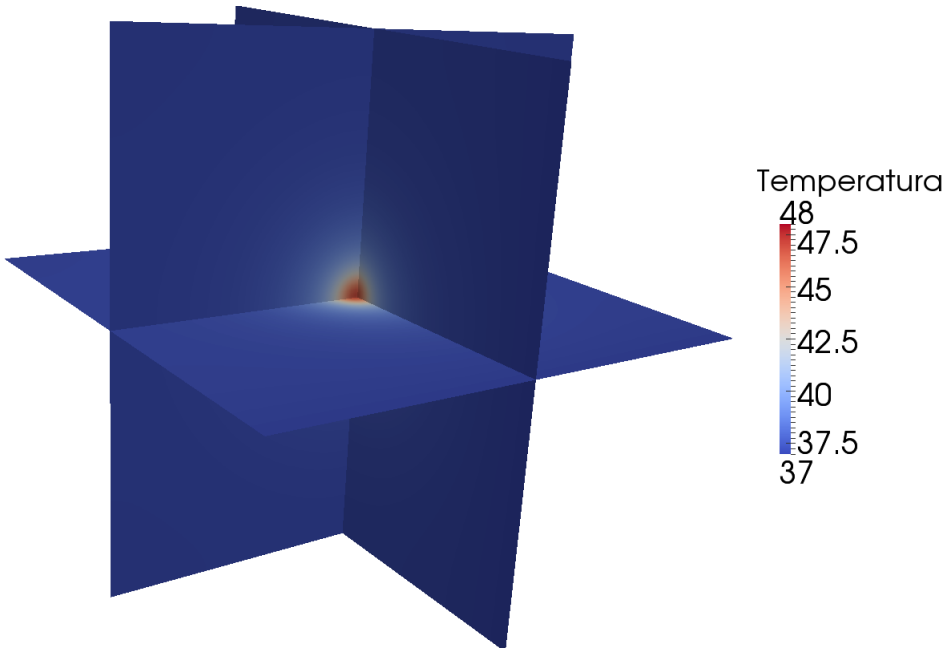


Figura 5.2: Solução do modelo (2.23) no tempo $t = 3.000$s.

A Fig. 5.3 descreve a temperatura em uma linha no eixo x, onde $y = 0,05$m e $z =$

$0,05$m, e a localização do tumor esta aproximadamente entre os pontos $x = 0,033$m e $x = 0,066$m. Nesta figura pode-se observar a elevação da temperatura no ponto de aplicação da hipertermia. Além disso, vale destacar que o tratamento por hipertermia produziu uma distribuição normal na proximidade do ponto de aplicação.

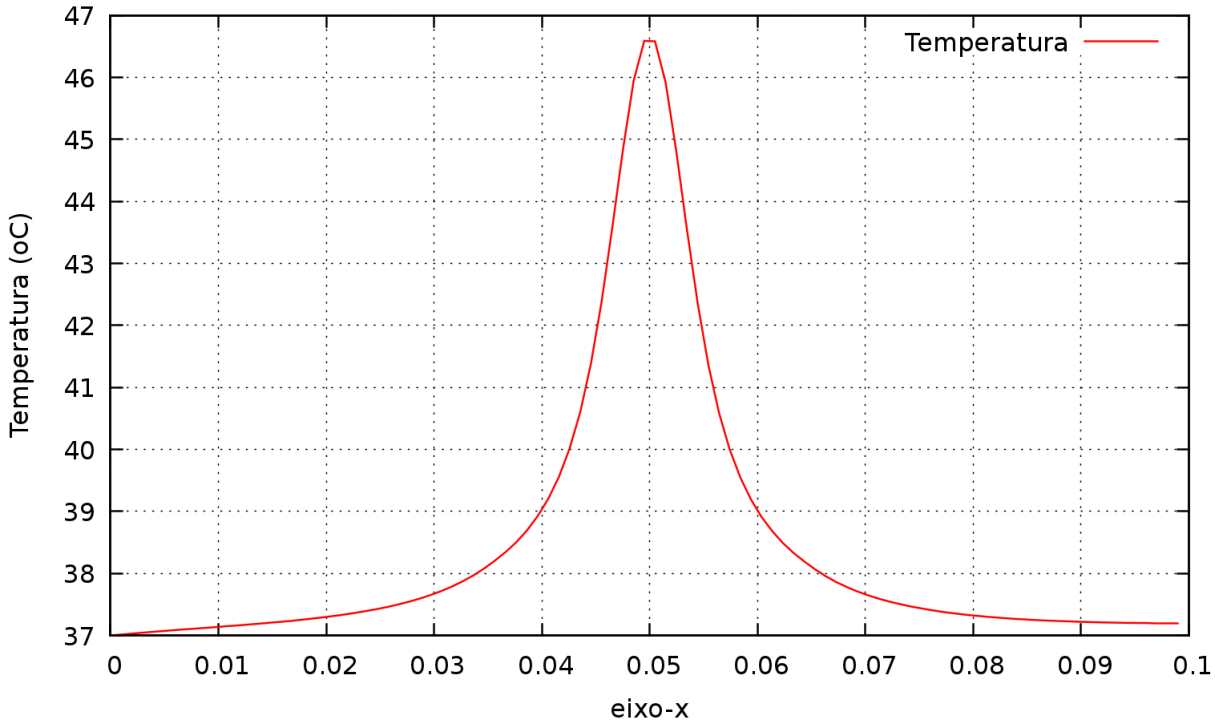


Figura 5.3: Solução da equação (2.23) no tempo $t = 3.000$s em uma linha no eixo x.

A Fig. 5.4 faz um comparativo da evolução da temperatura no tempo entre os pontos $(0,050m; 0,076m; 0,026m)$ e $(0,050m; 0,0500m; 0,050m)$, tecidos saudável e tumoral, respectivamente. Nesta figura observa-se o efeito da hipertermia durante o tempo de aplicação, além disto é importante observar que no tecido tumoral a temperatura atingiu os 43^oC, ou seja, a temperatura que causa necrose celular no tecido tumoral sem causar dano no tecido saudável.

5.1.3 Paralelização do Método

Este estudo foi executado em um computador com SMP Linux $(3.9.2 - 200)$, que possui 4 AMD Opteron 6272 CPU e 128 GB de RAM. Cada CPU possui 16 núcleos de 1400 MHz, totalizando 64 núcleos disponíveis. É válido ressaltar que neste processador cada 2 núcleos

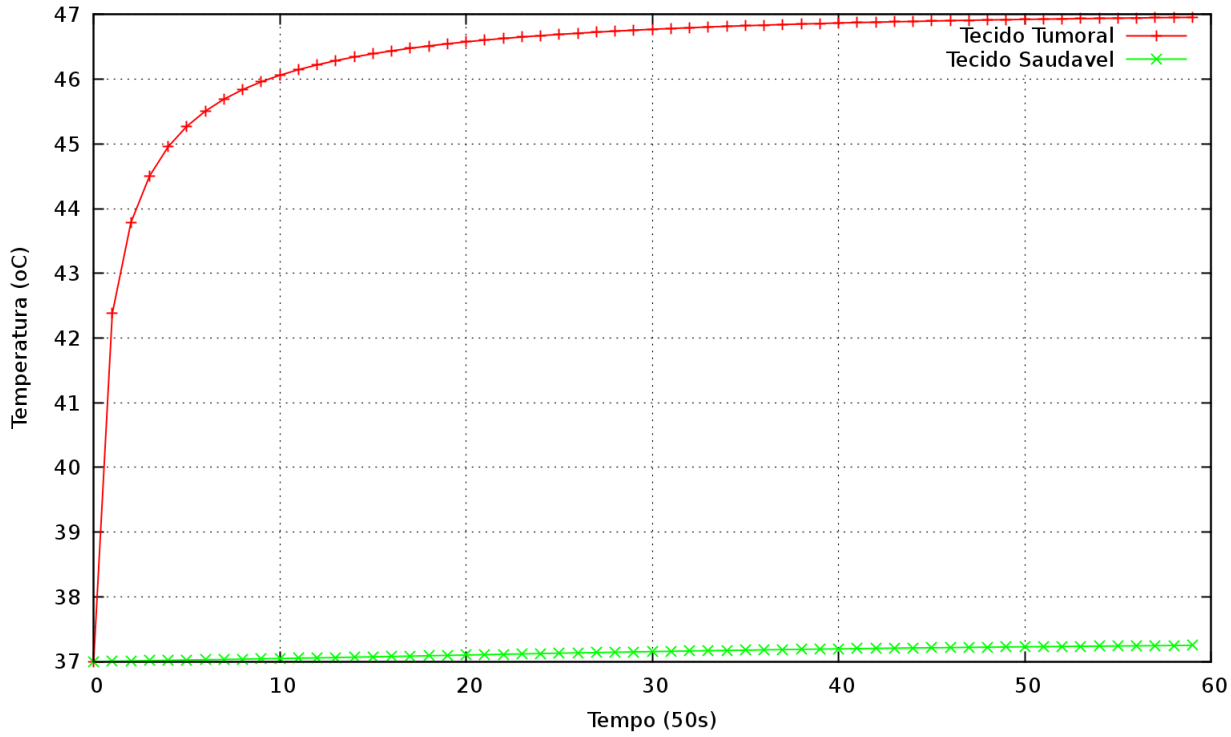


Figura 5.4: Solução da Eq. (2.23) em um ponto no tecido saudável $(0,050m; 0,076m; 0,026m)$ e no tecido tumoral $(0,050m; 0,0500m; 0,050m)$ através do tempo.

compartilham uma unidade de ponto flutuante (FPU), conforme pode ser observado na Fig. 5.5 NumericalAlgorithmsGroup (2011). Essa característica da arquitetura prejudicou o desempenho do programa. Dependendo da forma com que as *threads* são escalonadas, duas *threads* distintas podem disputar a utilização desta FPU, mesmo quando o número de *threads* é inferior ao número de FPUs disponíveis. Devido a peculiaridade da arquitetura deste computador, executou-se 3 cenários de escalonamento: estático, dinâmico com *chunk* de tamanho 1 e dinâmico com *chunk* de tamanho 2, onde o *chunk*, neste caso, é o número de linhas da matriz tridimensional que armazenam a temperatura em cada instante de tempo da simulação.

A aceleração do código obtida por meio da utilização da API de programação paralela *OpenMP* pode ser observada na Fig. 5.6. Essa aceleração foi obtida usando a média de ao menos três execuções do código e obteve um desvio padrão abaixo de $4,61$. Como pode ser observado, utilizando o escalonamento dinâmico das *threads* com *chunk* 1, obteve-se aceleração máxima acima de 39 vezes. O código que antes demandava aproximadamente

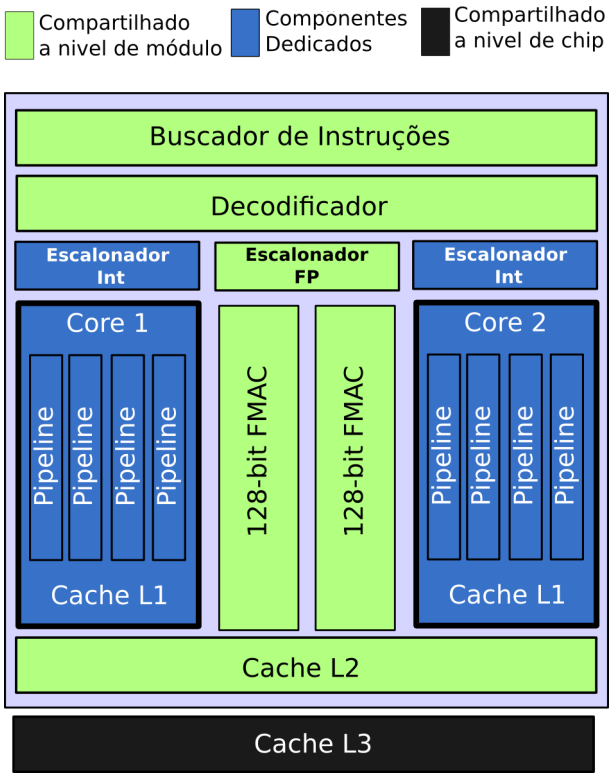


Figura 5.5: Compartilhamento de recursos do Chip AMD Opteron Série 6200 Interlagos. Adaptado de NumericalAlgorithmsGroup (2011).

3 horas, com a paralelização, passou a ser executado em apenas cerca de 5 minutos.

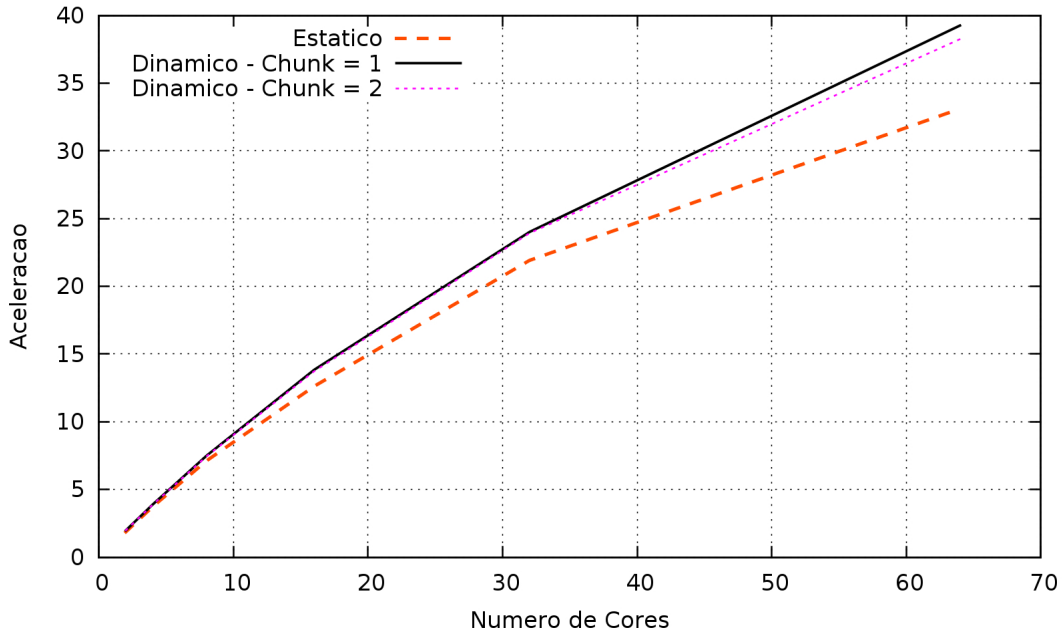


Figura 5.6: Aceleração do tempo de execução do código utilizando *OpenMP* em uma máquina *multicore*.

5.2 Estudo 2

Neste estudo de caso, foi feito um conjunto de simulações de tratamento de câncer com hipertermia por meio de nanopartículas magnéticas, onde também se considerou um tumor inserido na camada muscular. Além disto, o estudo utilizou o método numérico dado pela Eq. (3.20) para aproximar a solução do modelo matemático descrito pela Eq. (2.23). Vale ressaltar que neste estudo considerou-se a condição inicial $T(\cdot, 0) = 37,00^oC$ em Ω, condição de Dirichlet $T = 37,00^oC$, no plano $x = 0$ e Neumann $\nabla T \cdot \vec{n} = 0,00\ W/m^2$ nos demais.

Quanto à aplicação das nanopartículas, utilizou-se o modelo dado pela Eq. (2.19), *i.e.*, $Q_r(\vec{x}, t) = \sum_{i=1}^{M} A_i e^{-r_i^2/r_{0_i}^2} e^{-t/\tau}$. Este modelo considera tanto a dependência espacial do ponto de aplicação, quanto a variação no tempo.

5.2.1 Parâmetros do Modelo

Todas as simulações neste estudo foram executadas em um domínio tridimensional. O formato e localização do tumor podem ser observados na Fig. 5.7. O tumor é formado por duas esferas de raio $0,015m$ e $0,01m$ centradas nos pontos (0,05m; 0,05m; 0,05m) e (0,04m; 0,04m; 0,04m), respectivamente.

Este domínio é dividido em dois tipos de tecido: muscular e tumoral. Considerou-se um domínio cúbico de dimensão 0,1m x 0,1m x 0,1m com 100 subdivisões em cada direção, ou seja, $\Delta x = \Delta y = \Delta z = 0,001$m. Além disto, foi considerada uma simulação de 3.000s subdividida em 30.000 intervalos, ou seja, $\Delta t = 0,1$s.

O modelo considera dois tipos de camadas de tecido vivo, que possuem parâmetros diferentes. Os valores dos parâmetros são apresentados na Tab. 5.3 e foram baseados nos estudos realizados por Liu e Xu (2000). Este trabalho considerou o parâmetro de perfusão sanguínea dependente da temperatura, conforme as Eq. (2.20) e (2.22), o qual refere-se ao parâmetro de perfusão sanguínea nas camadas de músculo e tumor, respectivamente.

Além disso, quanto aos parâmetros do tratamento com as nanopartículas magnéticas, optou-se por 3 pontos de injeção, descritos pela Tab. 5.4.

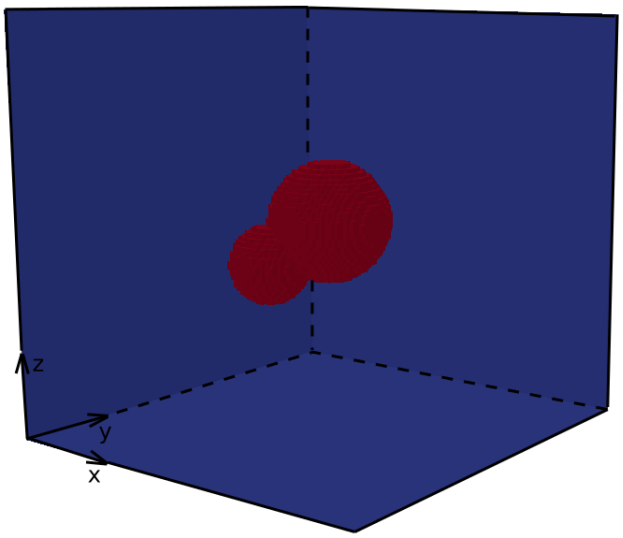


Figura 5.7: Domínio da simulação do estudo da seção 5.2. Destaque para o tumor localizado no centro do cubo.

Tabela 5.3: Parâmetros do modelo no estudo 2.

Símbolo	**Unidade**	**Músculo**	**Tumor**
c	(J/Kg^oC)	$4.200,0$	$4.200,0$
c_b	(J/Kg^oC)	$4.200,0$	$4.200,0$
k	(W/m^oC)	$0,50$	$0,55$
ρ	(Kg/m^3)	$1.000,0$	$1.000,0$
ρ_b	(Kg/m^3)	$1.000,0$	$1.000,0$
ω_b	$(Kg/s/m^3)$	$\omega_{músculo}$	ω_{tumor}

Tabela 5.4: Parâmetros dos pontos de injeção no estudo 2.

Posição (m)	**A**	$\mathbf{r_0}$	τ
$(0,050;\ 0,050;\ 0,050)$	$0,8$ x 10^6 W/m^3	$0,6$ x 10^{-3}m	30000s
$(0,045;\ 0,045;\ 0,045)$	$0,8$ x 10^6 W/m^3	$0,6$ x 10^{-3}m	30000s
$(0,040;\ 0,040;\ 0,040)$	$0,8$ x 10^6 W/m^3	$0,6$ x 10^{-3}m	30000s

5.2.2 Simulações do Estudo

Após a execução deste modelo, com os parâmetros descritos na subseção anterior, obteve-se o resultado da simulação de um tratamento por meio de hipertermia com nanopartículas magnéticas. Como já destacado, as células sofrem necrose quando a temperatura chega a aproximadamente 43^oC (Salloum *et al.*, 2008).

Todos os resultados apresentados neste estudo representam a solução da Eq. (2.23) através do método numérico explícito descrito pela Eq. (3.20) por meio do Pseudocódigo 2 e utilizando os parâmetros das Tab. 5.3 e 5.4.

A Fig. 5.8 mostra tridimensionalmente o volume de tecido que atingiu a temperatura

limite de 43^oC, ou seja, a massa de células que morreu.

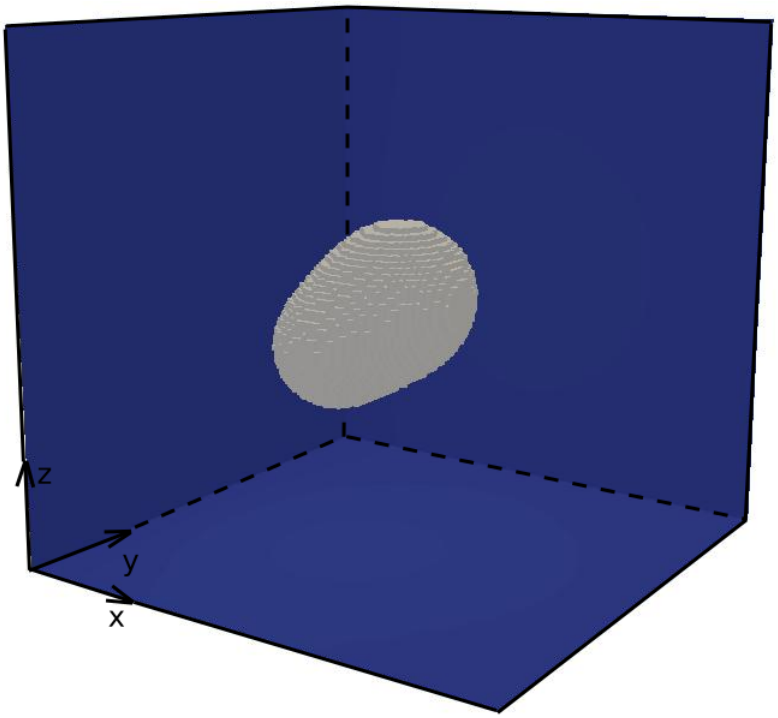
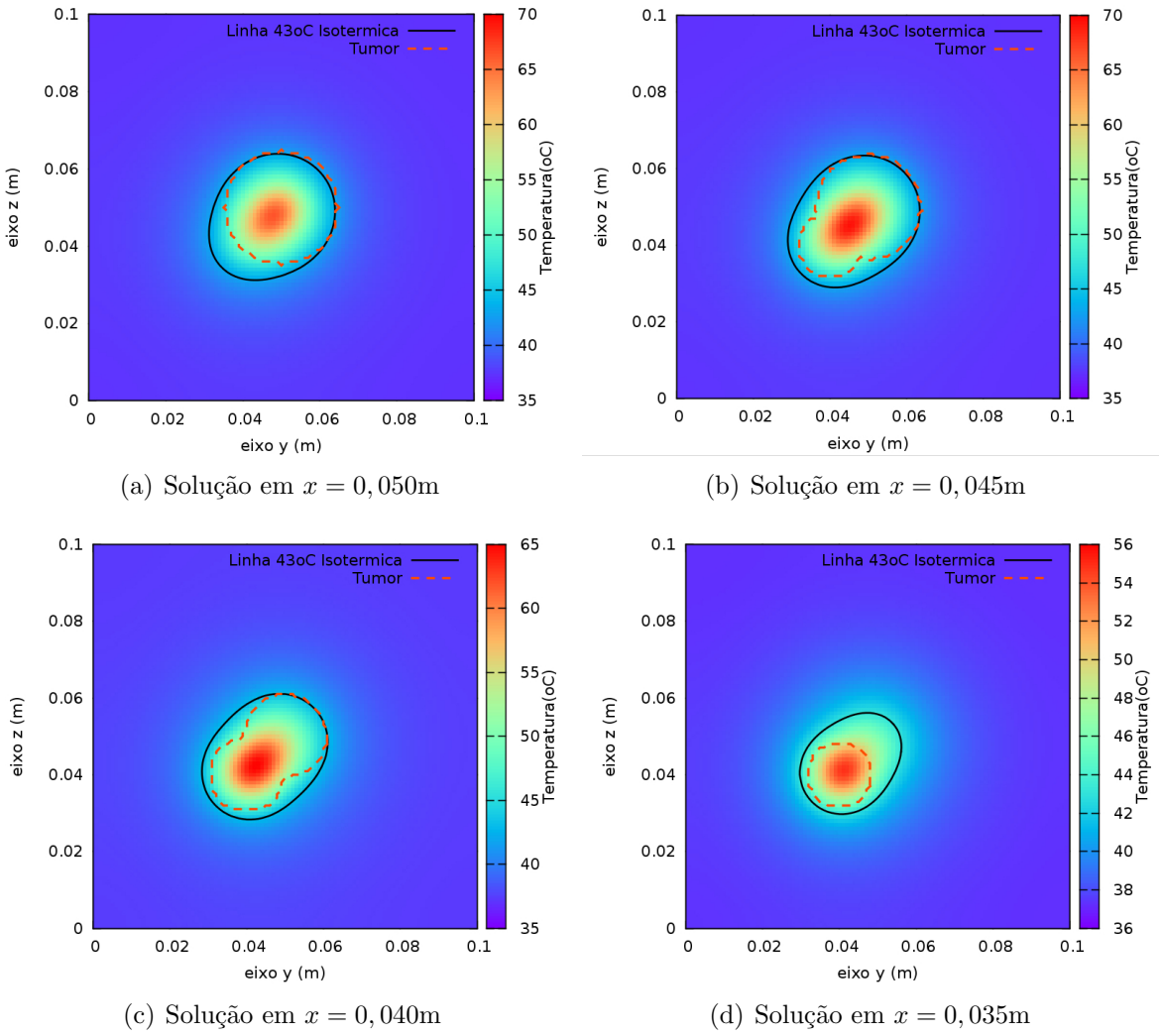


Figura 5.8: Solução do modelo no tempo $t = 3.000s$ mostrando o volume de tecido morto no domínio. Estudo da seção 5.2

As Figs. 5.9(a), 5.9(b), 5.9(c) e 5.9(d) representam a solução do modelo no instante de tempo $t = 3.000s$ nos planos $x = 0,050m$, $x = 0,045m$, $x = 0,040m$ e $x = 0,035m$, respectivamente. Nestas figuras é possível observar a linha isotérmica de $T = 43^oC$, ou seja, a temperatura onde ocorre a morte celular. Além disto, as figuras também possuem uma linha que representa a delimitação do tumor no respetivo plano do domínio.

A Fig. 5.10 representa a evolução da temperatura no tempo nos respectivos pontos. É possível observar a influência do termo temporal da Eq. (2.19), pois inicialmente a temperatura aumenta até um limiar, e a partir daí a temperatura começa a diminuir suavemente. Também pode-se observar nesta mesma figura que no ponto (0,06m;0,06m;0,06m), que representa um pedaço do domínio com tecido saudável, a temperatura não atinge o limiar de 43^oC. O contrário pode ser observado no ponto tumoral (0,05;0,05;0,05), pois este atinge e se mantêm acima de 43^oC.

A Fig. 5.11 representa a solução do modelo no instante $t = 3.000s$ em $x = 0,05$m e $z = 0,05$m através do eixo y, e a localização do tumor esta aproximadamente entre os pontos $y = 0,033$m e $y = 0,066$m. Neste gráfico é observável o perfil de temperatura



(a) Solução em $x = 0,050$m

(b) Solução em $x = 0,045$m

(c) Solução em $x = 0,040$m

(d) Solução em $x = 0,035$m

Figura 5.9: Gráficos da solução da equação transiente no instante $t = 3000$s. Estudo da seção 5.2.

no eixo y, valendo-se destacar que no ponto central ($y = 0,05$m), ponto de injeção das nanopartículas, obtém-se a temperatura mais alta, e a mesma diminui exponencialmente conforme se distancia dele.

5.2.3 Paralelização do Método

Este estudo foi executado no mesmo computador descrito no estudo 1. Devido a característica arquitetural do processador, e aos resultados apresentados no primeiro estudo, foram executados dois cenários de escalonamento: estático e dinâmico com *chunk* de tamanho 1.

Em cada um dos cenários de escalonamento das *threads* foram realizadas três

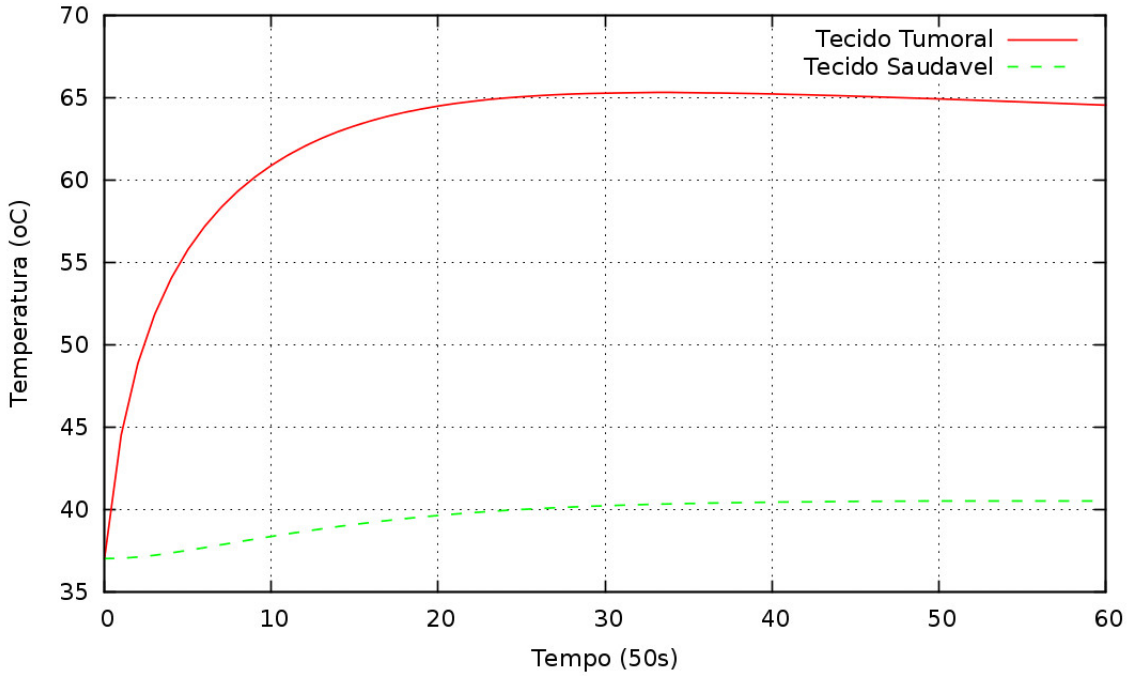


Figura 5.10: Solução transiente da modelo em um ponto do domínio com tecido saudável $(0,06\text{m};0,06\text{m};0,06\text{m})$ e tumoral $(0,05\text{m};0,05\text{m};0,05\text{m})$. Estudo da seção 5.2.

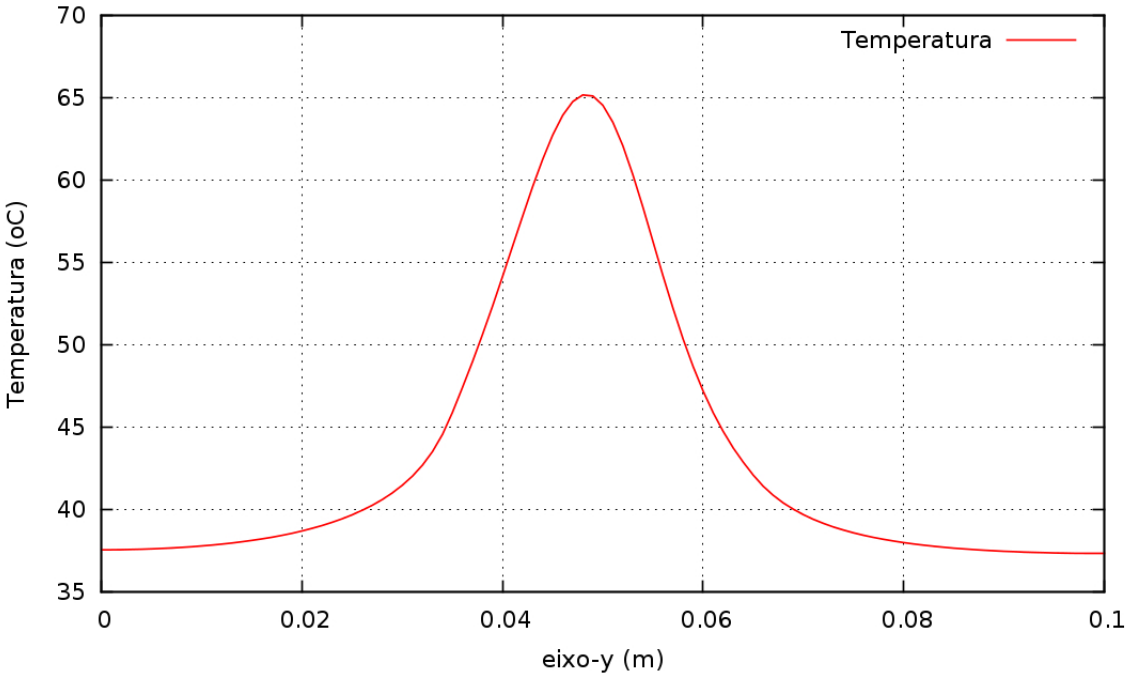


Figura 5.11: Solução da equação transiente no instante $t = 3000\text{s}$ em $x = 0,05\text{m}$ e $z = 0,05\text{m}$ através do eixo y. Estudo da seção 5.2.

execuções, obtendo-se um desvio padrão relativo máximo de $28,44\%$. Este desvio padrão foi obtido ao submeter-se o algoritmo para ser executado em 64 núcleos; nas demais simulações obteve-se desvio de no máximo $1,8274\%$. A aceleração máxima obtida neste

estudo foi de 25, 72. A Fig. 5.12 apresenta o gráfico da aceleração em função do número de núcleos utilizados. O código que antes demandava aproximadamente 3 horas e meia para ser executado sequencialmente, com a paralelização, passou a ser executado em cerca de 8, 5 minutos. Uma possível justificativa para a diferença de aceleração obtida entre este estudo e o anterior é a função Q_r escolhida. Neste estudo adotou-se a que também depende do tempo, o que demanda mais acesso à unidade de ponto flutuante do processador, gerando ainda mais disputa entre as *threads*.

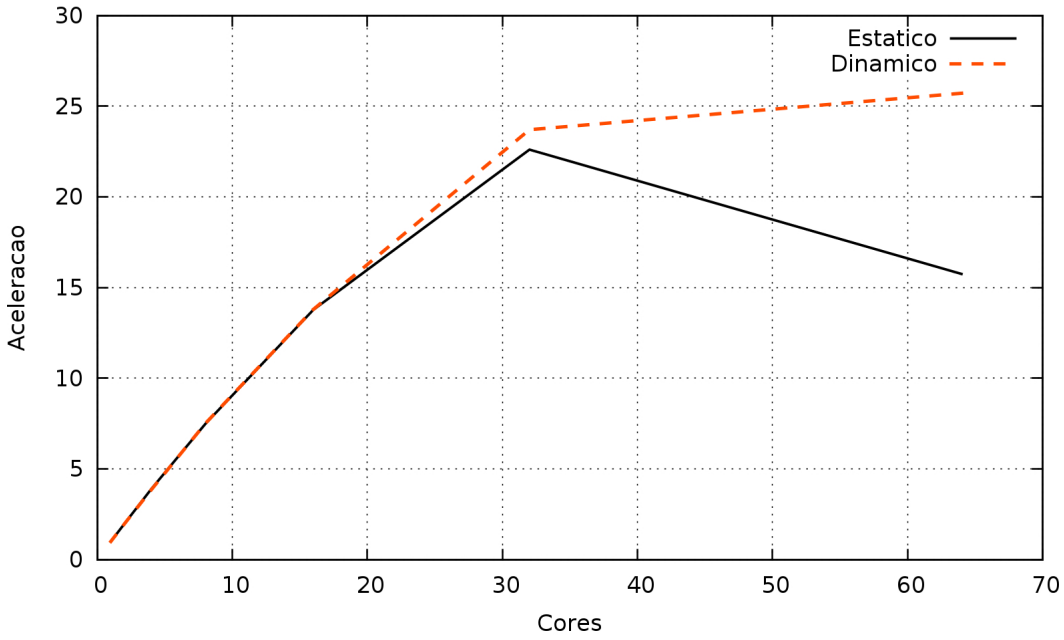


Figura 5.12: Aceleração do tempo de execução do código utilizando *OpenMP* em uma máquina *multicore*. Estudo da seção 5.2.

5.3 Estudo 3

Neste terceiro estudo de caso, foi realizado um conjunto de análises sobre a aplicação de um tratamento de câncer com hipertermia por meio de nanopartículas magnéticas, onde se consideram três camadas de tecido vivo normal, pele, gordura e músculo, além de um tumor localizado na camada muscular.

Foram realizadas duas simulações numéricas distintas, mas complementares: *(i)* A solução do modelo estacionário definido pela Eq. (2.17), e *(ii)* o modelo transiente dado pela Eq. (2.23), que descreve o tratamento de hipertermia propriamente dito.

Nas condições de contorno, foram utilizadas condição de Dirichlet $T = 37,00^oC$, no plano $x = 0$ e Neumann $\nabla T \cdot \vec{n} = 0,00\ W/m^2$ nos demais, em ambas as simulações.

5.3.1 Parâmetros do Modelo

Para realizar as simulações neste estudo de caso considerou-se um domínio retangular de lados 0,05m x 0,1m x 0,1m, conforme a Fig. 5.16 ilustra. Vale destacar que a camada muscular, gordura e pele possuem as seguintes espessuras: 38mm, 10mm e 2mm, respectivamente. Além disso, o tumor é formado por duas esferas localizadas em $(0,020m; 0,050m; 0,050m)$ e $(0,025m; 0,065m; 0,065m)$ na camada muscular do tecido de raios 0,013m e 0,010m, respectivamente

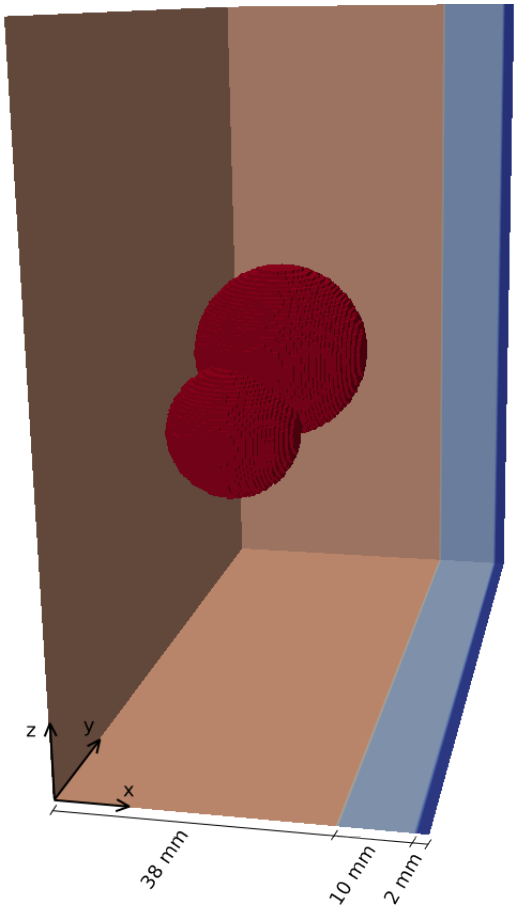


Figura 5.13: Domínio do tecido vivo com um tumor no estudo da seção 5.3.

Além do domínio, é importante mostrar quais foram os valores utilizados nos parâmetros do modelo. A Tab. 5.5 mostra os valores de cada parâmetro utilizado neste estudo. É importante relembrar que $\omega_{gordura}$, $\omega_{músculo}$ e ω_{tumor} são dados pelas Eqs. (2.21),

(2.20) e (2.22). Os valores desta tabela foram baseados em artigos da literatura (Lang *et al.*, 1997; Erdmann *et al.*, 1998; Liu e Xu, 2000; Salloum *et al.*, 2008).

Tabela 5.5: Parâmetros do modelo no estudo da seção 5.3.

Símbolo	**Unidade**	**Pele**	**Gordura**	**Músculo**	**Tumor**
Q_m	(W/m^3)	$420,0$	$420,0$	$420,0$	$4.200,0$
c	(J/Kg^oC)	$3.600,0$	$2.500,0$	$3.800,0$	$4.200,0$
c_b	(J/Kg^oC)	$4.200,0$	$4.200,0$	$4.200,0$	$4.200,0$
k	(W/m^oC)	$0,40$	$0,21$	$0,45$	$0,55$
ρ	(Kg/m^3)	$1.200,0$	$1.000,0$	$1.000,0$	$1.000,0$
ω_b	$(Kg/s/m^3)$	$0,5$	$\omega_{gordura}$	$\omega_{músculo}$	ω_{tumor}
$\overline{\omega}_b$	$(Kg/s/m^3)$	$0,5$	$0,504908$	$1,87895$	$0,757981$

No caso transiente, para modelar o tratamento de hipertermia com nanopartículas magnéticas, é importante destacar as configurações dos pontos de aplicação das nanopartículas. Os parâmetros utilizados neste estudo de caso foram os dados pela Tab. 5.6. Além disto, a função utilizada para modelar a aplicação da hipertermia é dada pela Eq. (2.18), *i.e*, $Q_r(\vec{x}) = \sum_{i=1}^{M} A_i . e^{\frac{-r_i^2}{r_{0,i}^2}}$.

Tabela 5.6: Parâmetros dos pontos de injeção no estudo da seção 5.3.

Posição (m)	**A**	$\mathbf{r_0}$
$(0,020;\ 0,055;\ 0,055)$	$0,8$ x 10^6 W/m^3	$0,6$ x 10^{-3}m
$(0,025;\ 0,045;\ 0,045)$	$0,7$ x 10^6 W/m^3	$0,6$ x 10^{-3}m
$(0,015;\ 0,040;\ 0,040)$	$0,7$ x 10^6 W/m^3	$0,6$ x 10^{-3}m

5.3.2 Simulações do Estudo

Os resultados das simulações para a parte *(i)* são apresentados nas Figs. 5.14 e 5.15. Os resultados para o modelo não linear da equação estacionária também são apresentados, para fins comparativos, na Fig. 5.16.

O mapa de distribuição de temperatura no ponto $x = 0,05$m, ou seja, na camada de pele do domínio, é apresentado pela Fig. 5.14. Este gráfico mostra a alteração metabólica causada pela existência de um tumor sobre a pele. Esta alteração do metabolismo acarreta em um ligeiro aumento da temperatura no tumor, que reflete, também, na superfície da pele. Esta alteração de temperatura pode ser utilizada, também, na detecção de tumores (Liu e Xu, 2000).

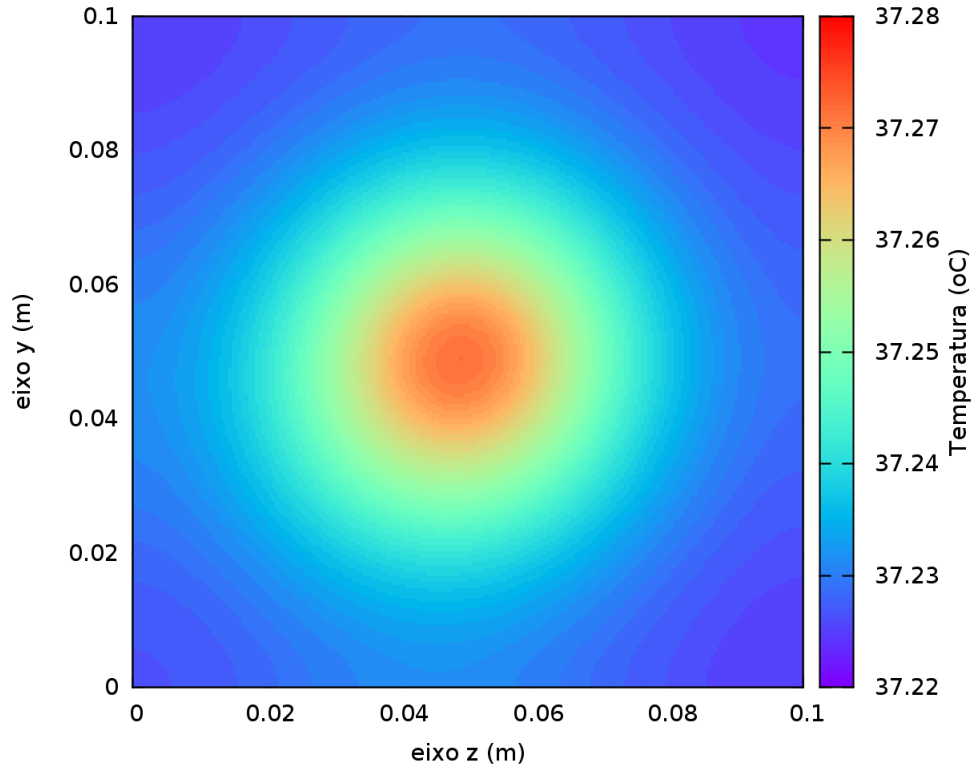


Figura 5.14: Solução a equação estacionária na superfície $x = 0,05$.

A Fig. 5.15 mostra um corte no plano xz em $y = 0,025$. Este mapa de distribuição de temperatura também mostra as alterações metabólicas causadas pelo tumor, elevando a temperatura do tecido vivo. Com base nesta figura é possível ter uma noção mais precisa da profundidade do tumor. O volume ocupado pelo tecido tumoral poderia ser aferido pela junção de vários cortes em diferentes planos do tecido.

No gráfico dado pela Fig. 5.16 pode ser observado que não houve nenhuma alteração relevante entre o modelo linear e não linear, para o caso da solução do estado estacionário. Neste caso é válido observar que, a mudança significativa nos valores das perfusões sanguíneas $\omega_{gordura}$, $\omega_{músculo}$ e ω_{tumor} ocorrem somente após a temperatura de 40^oC.

A fim de tornar a simulação da parte *(ii)* mais próxima da realidade, ao resolver o modelo dado pela Eq. (2.23) foi utilizada a solução da Eq. (2.16) como condição inicial. Conforme a Tab. 5.6, as nanopartículas foram injetadas em $(0,02m;\ 0,055m;\ 0,055m)$, $(0,025m;\ 0,045m;\ 0,045m)$ e $(0,015m;\ 0,04m;\ 0,04m)$.

Novamente, para fins comparativos, são apresentados os resultados utilizando o modelo linear e não linear. A Fig. 5.17 mostra o comparativo ao longo do tempo nos pontos

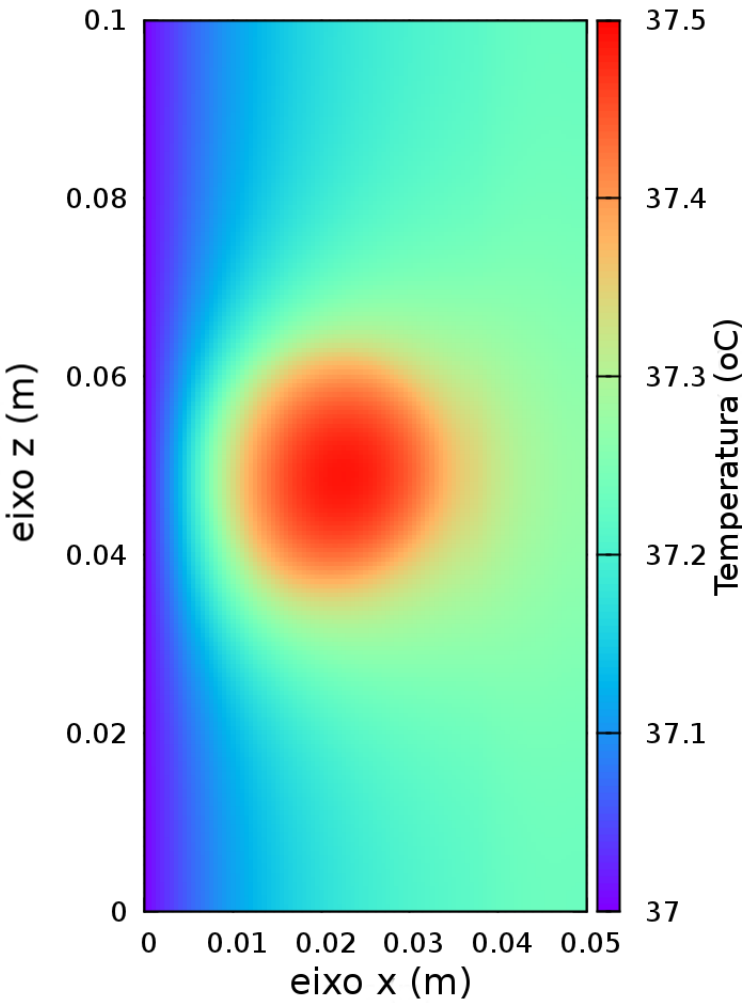


Figura 5.15: Solução da equação estacionária no plano xz em $y = 0,025$.

$(0,025m; 0.05m; 0,05m)$ e $(0,025m; 0,065m; 0,065m)$, representado pontos em tecido tumoral e saudável, respectivamente. Este gráfico mostra uma clara diferença, tanto no tecido saudável quanto no tumoral. Essa diferença acontece principalmente a partir dos 500s de simulação (tempo simulado, não o computacional).

A Fig. 5.18 mostra um corte ao longo do eixo y em $t = 3.000s$, $x = 0,025$ e $z = 0,05$, a localização do tumor está aproximadamente entre os pontos $y = 0,036$m e $y = 0,062$m. Este gráfico também compara o modelo linear e não linear, mostrando que no eixo y também há diferença entre os modelos.

Além da análise do parâmetros do modelo anteriormente realizada, é importante destacar a efetividade do tratamento, isto é, a morte das células tumorais. Neste estudo foi considerado que a temperatura de 43^oC é suficiente para causar a morte de tecidos

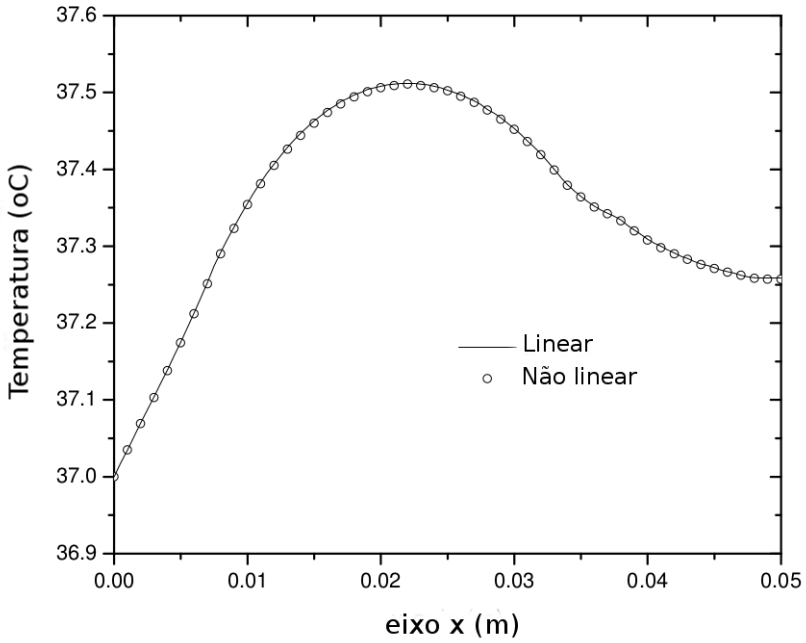


Figura 5.16: Solução da equação estacionária, comparando os modelos linear (ω_b constante) e não linear (ω_b em função da temperatura), sobre o eixo x em $y = 0,05$ e $z = 0,05$.

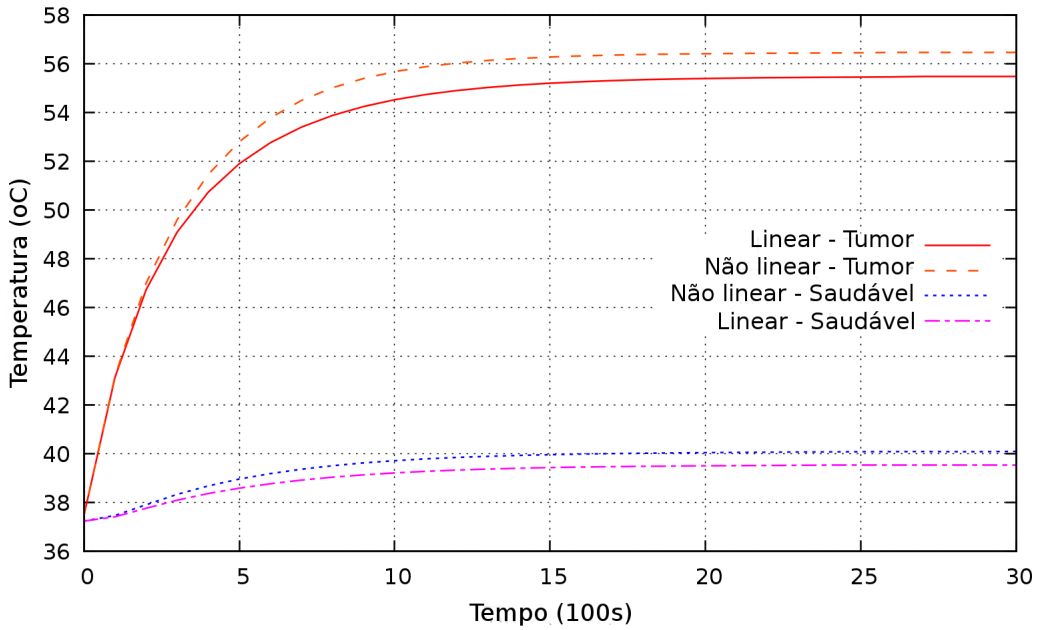


Figura 5.17: Comparativo da solução da equação transiente entre os modelos linear (ω_b constante) e não linear (ω_b em função da temperatura). Mostra a evolução no tempo no tecido tumoral em $x = 0,025$, $y = 0,05$ e $z = 0,05$, e saudável em $x = 0,025$, $y = 0,065$ e $z = 0,065$.

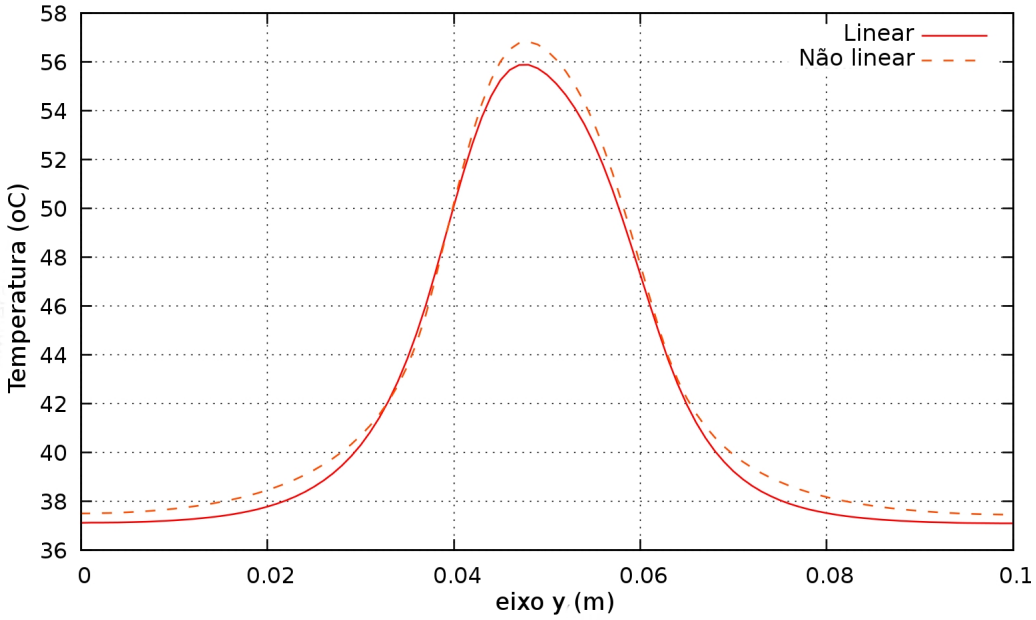


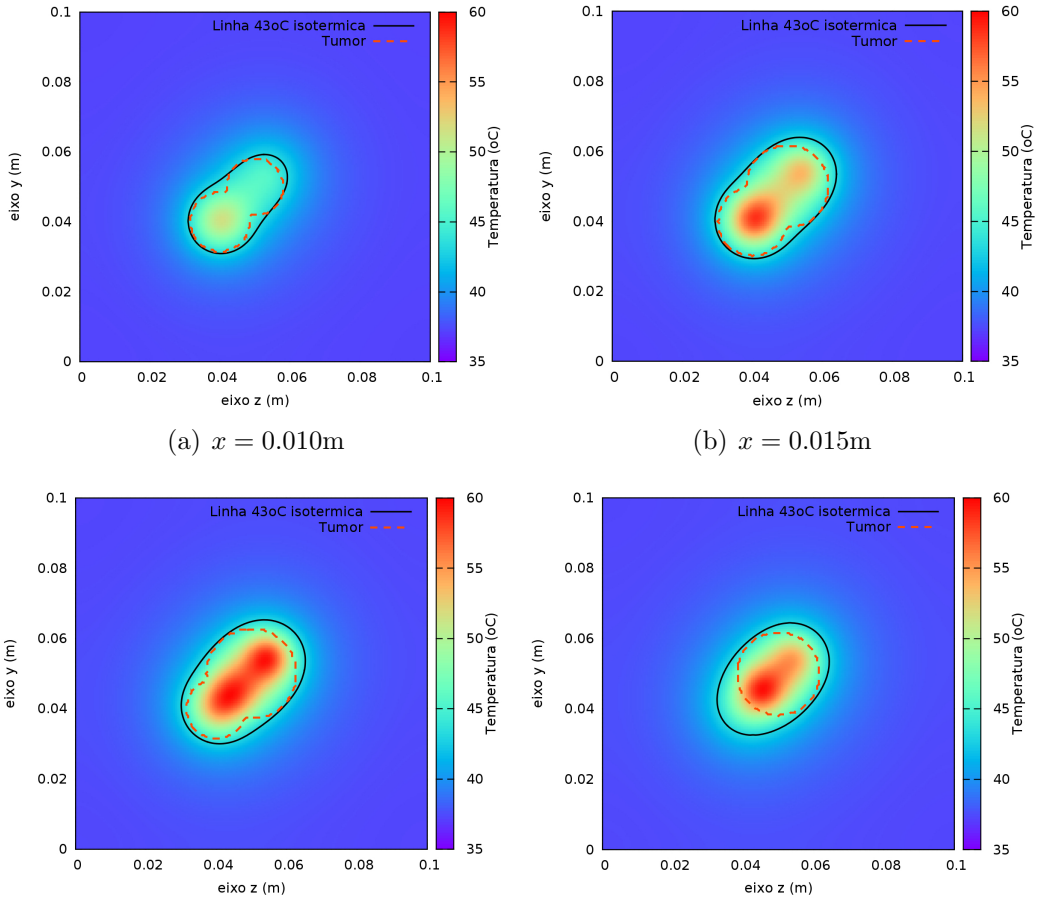
Figura 5.18: Temperatura ao longo do eixo y em $t = 3.000s$, $x = 0,025$ e $z = 0,05$, comparando os modelos linear (ω_b constante) e não linear (ω_b em função da temperatura).

vivos seja por apoptose, ou por necrose celular (Lang *et al.*, 1997; Salloum *et al.*, 2008, 2009).

Nas Figs. 5.19(a), 5.19(b), 5.19(c) e 5.19(d) são mostrados alguns cortes do tecido no eixo yz nos pontos $x = 0,010$m, $x = 0,015$m, $x = 0,020$m e$x = 0,025$m, respectivamente. Nestes cortes é possível visualizar a difusão do calor em diferentes partes do domínio simulado, bem como no tumor. Em todos os cortes do domínio a linha que simboliza a área onde se obteve a temperatura maior ou igual a 43^oC contorna a localização do tumor na respectiva posição espacial.

5.3.3 Paralelização do Método

As simulações executadas neste estudo de caso, paralelizado por meio da plataforma CUDA, foram executadas usando 3 diferentes discretizações da malha espacial, com $\Delta x = \Delta y = \Delta z = \Delta$ e $\Delta = 0,001$m, $0,0005$m, e $0,00025$m, *i.e.*, $(I_x, I_y, I_z) = (50, 100, 100)$, $(100, 200, 200)$ e $(200, 400, 400)$. Deve-se ressaltar que a utilização de mais pontos permite a simulação de tecidos maiores, do ponto de vista biológico, como também a obtenção de resultados mais precisos, do ponto de vista numérico.



(a) $x = 0.010$m (b) $x = 0.015$m

(c) $x = 0.020$m (d) $x = 0.025$m

Figura 5.19: Solução da equação transiente em $t = 3000$s em alguns planos zy.

O tamanho do bloco foi escolhido de forma a minimizar o número de *threads* e maximizar a ocupação conforme mostrado na Fig. 5.20. Então, foi criado um bloco tridimensional com $(8, 14, 4)$ *threads*, ou 448 *threads* por bloco e uma grade tridimensional com $(13, 14, 50)$. Em outras palavras, foram criadas 4.076.800 *threads* divididas em 9.100 blocos de tamanho 448.

Devido ao longo tempo computacional exigido para executar a simulação na malha refinada (*i.e.*, $\Delta = 0,00025$), os resultados da paralelização foram divididos em dois casos, o primeiro considerando os 250 primeiros segundos simulados (tempo real da simulação e não o gasto pelo computador para simular) e os 3.000s de simulação, ou seja, a simulação total. Os primeiros 250s de simulação com a malha refinada, usando a versão serial, demandaram cerca de 60h de computação, o que implica, extrapolando-se o tempo, em um total de 718h para a simulação completa. Então, torna-se inviável a execução serial

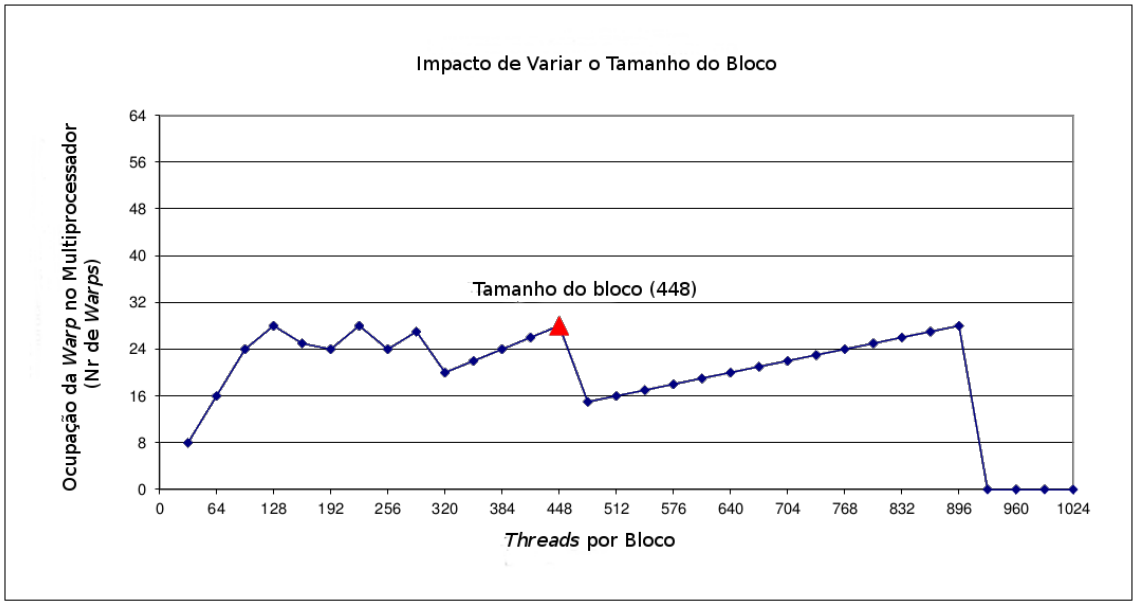


Figura 5.20: Impacto da variação do tamanho do bloco. Adaptado da *Cuda Occupancy Calculator* (NVIDIA, 2014a).

do caso refinado.

As execuções na CPU da versão sequencial da aplicação foram obtidas a partir de código compilado com as *flags* de compilação otimizada para a arquitetura do processador AMD Opeteron, *-march=bdver1* e *-O3*, utilizando-se precisão dupla. Já o desempenho da versão paralela do mesmo algoritmo, quando executada na placa de vídeo Tesla M2075, da arquitetura Fermi, é reduzido quando a precisão dupla é utilizada, em relação a mesma versão de código que utiliza precisão simples. Um eventual ganho de desempenho não é observado quando o código sequencial é executado na CPU utilizando precisão simples. Além do mais, os resultados da distribuição de temperatura do modelo simulado não apresentaram diferença significativa quando a precisão simples e dupla são utilizadas. Por estes motivos as simulações na GPU foram executadas com precisão simples, enquanto na CPU com precisão dupla.

As Tabs. 5.7 e 5.8 apresentam, respectivamente, os tempos de computação para CPU e GPU, bem como as acelerações obtidas nos primeiros 250s de simulação e na simulação completa. Todas estas simulações foram executadas ao menos 3 vezes, com um desvio padrão relativo máximo de $1,5\%$. A melhor aceleração obtida foi cerca de 242, o que significa que as 92h gastas para executar a simulação serial na CPU foram reduzidas a

23 minutos na GPU. Um fato que merece ser destacado é que o tempo de simulação exigido no computador foi duas vezes menor do que o tempo de simulação real, *i.e.*, foram simulados 50 minutos de tratamento de hipertermia em apenas 23 minutos no computador. Em cada uma das simulações executadas na GPU é criado uma *thread* por ponto do domínio, ou seja, nas malhas 50x100x100, 100x200x200 e 200x400x400 foram criadas 500.000, 4.000.000 e 32.000.000 *threads*, respectivamente.

Tabela 5.7: Tempo de execução e aceleração dos primeiros 250s segundos de simulação de tratamento de hipertermia.

Malha	**Média Tempo CPU**	**Média Tempo GPU**	**Aceleração**
50x100x100	3.167, 7s	22, 6s	140, 5
100x200x200	27.763, 9s	129, 1s	215, 1
200x400x400	215.532, 0s	12.233, 0s	176, 2

Tabela 5.8: Tempo de execução e aceleração da simulação total de um tratamento de hipertermia.

Malha	**Média Tempo CPU**	**Média Tempo GPU**	**Aceleração**
50x100x100	37.776, 2s	251, 3s	150, 3
100x200x200	331.828, 9s	1.371, 3s	242

Além das modificações descritas no capítulo anterior, foram definidos ainda o número máximo de *threads* por bloco e o número mínimo de blocos por multiprocessador, através do comando *__launch_bounds__*. Contudo, neste caso não se obteve diferença significa no tempo de execução. Além disso, a adoção das configuração de tamanho de bloco e grade que oferecem a melhor ocupação, descritas anteriormente, foram responsáveis pela redução em 17% do tempo de execução, no caso da malha intermediária. Devido a placa de vídeo utilizada (Tesla M2075) ter suporte ao ECC, foi ainda comparado o resultado da execução com o suporte a ECC habilitado e desabilitado. Com o ECC ativo o tempo de execução foi de aproximadamente 1773s, desativando-o o tempo reduziu para 1371s, ou seja, diminuiu em 29%. Este resultado foi também para o caso da malha intermediária. Por fim, o código foi compilado com todas as *flags* de otimizações usuais, *-O3* e *-arch=SM_20*.

5.3.4 Análise do Erro no Método Preditor-Corretor

Para avaliar a precisão dos resultados, a Fig. 5.21 mostra a norma do erro relativo definido por $\left(\sum_{i,j,k} \frac{\left(T_{i,j,k}^{n+1,r+1} - T_{i,j,k}^{n+1,r}\right)^2}{\left(T_{i,j,k}^{n+1,r+1}\right)^2} \right)^{1/2}$. Vale destacar que para analisar o erro utilizou-se precisão numérica dupla. Uma vez que é utilizado um algoritmo iterativo, o erro a cada instante t_{n+1} entre duas iterações sucessivas é diretamente computado e monitorado em uma simulação não linear. Como pode ser observado nesta figura, as iterações sucessivas do método preditor corretor obtém um considerável aumento na precisão dos resultados.

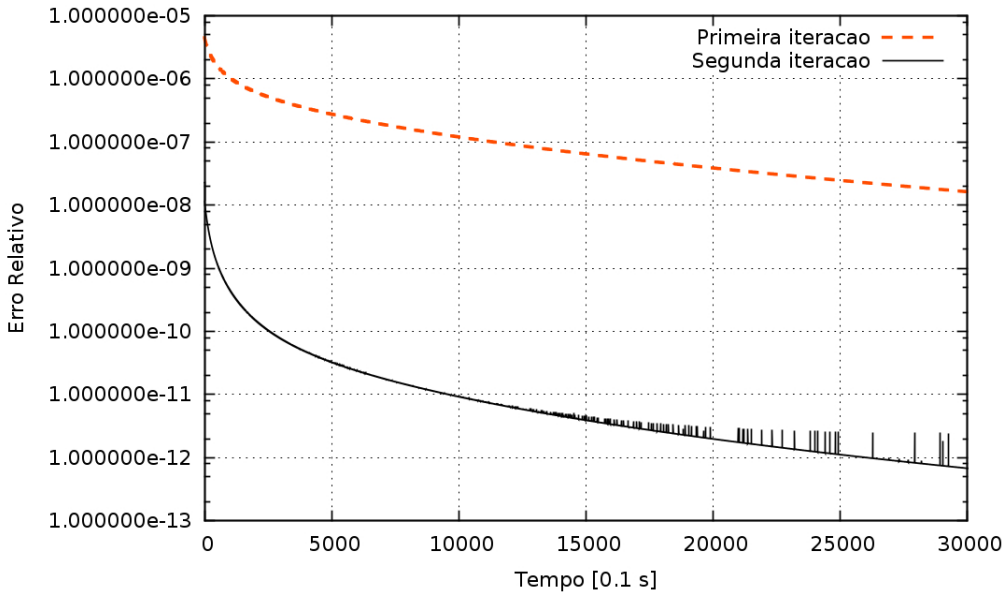


Figura 5.21: Comparativo entre a primeira e a segunda iteração do erro relativo no método preditor-corretor durante a simulação.

6 CONCLUSÕES

Este trabalho objetivou simular a utilização de nanopartículas magnéticas no tratamento de tumores sólidos, utilizando o modelo de biotransferência de calor proposto por Pennes (1948). Neste modelo matemático foi considerado um domínio tridimensional, além de diferentes cenários de aplicação das nanopartículas, considerando os tecidos de músculo, gordura e pele. Em todas as simulações se considerou um tumor inserido na camada muscular. Quanto à aplicação de nanopartículas, utilizaram-se diferentes formas de aplicação e dois modelos de taxa de absorção específica. Além disso, empregou-se, também, um modelo de perfusão sanguínea dependente da temperatura. Para encontrar a aproximação numérica do modelo matemático utilizado, empregaram-se duas estratégias numéricas na marcha do tempo, e uma discretização por diferenças finitas no espaço, baseada na clássica diferença centrada de 7 pontos, adaptada para meio heterogêneo. Quanto aos métodos aplicados na marcha do tempo, foram utilizados o Euler clássico, que possui convergência linear, e um algoritmo do tipo preditor corretor, que possui convergência quadrática. No caso onde se emprega o modelo de perfusão sanguínea dependente da temperatura, se obtém um sistema não linear. Neste caso, os métodos numéricos, aplicados na marcha do tempo, possuem a vantagem de não necessitar a resolução de um sistema de equações não lineares. Para acelerar o tempo gasto nas simulações, foram empregadas duas técnicas de computação paralela, a primeira baseada na API de programação OpenMP, e a segunda baseada na plataforma CUDA. Para fazer todas estas simulações, este trabalho foi dividido em 3 estudos.

No primeiro estudo, foi simulado este tipo de tratamento de câncer em um domínio contendo somente um tumor sólido, inserido no tecido muscular. Neste estudo foi avaliado o comportamento do modelo de geração de calor, ou SAR, com dependência espacial, analisando a variação da temperatura no espaço e no tempo. Além disso, com a utilização da estratégia de paralelização auxiliada pela API *OpenMP*, foi possível acelerar a execução da simulação em mais de 39 vezes quando comparada à versão serial, em uma máquina

de 64 núcleos. Devido a peculiaridade da arquitetura do computador, foram testados 3 cenários de escalonamento da distribuição de tarefas para as *threads*. O escalonamento dinâmico, muitas vezes, consome uma parcela do tempo computacional, portanto era esperado que o escalonamento estático obteria resultado mais eficiente, porém os melhores resultados foram obtidos com o escalonamento dinâmico das *threads*. Uma possível explicação deste acontecimento, é a disputa pela FPU acarretada pela arquitetura do processador AMD Opteron 6272.

O segundo estudo simulou o mesmo tratamento, considerando tumor e músculo, porém com a geração de calor variando no tempo e espaço. Neste caso, também foi analisada a variação da temperatura no espaço e no tempo. O comportamento da temperatura em um mesmo ponto, evoluído no tempo, inicialmente cresce, porém ao invés de estabilizar, mantém um decaimento contínuo, diferente dos resultados obtidos no estudo anterior. Este decaimento é causado pela maneira que a geração de calor foi modelada, onde se considerou um decaimento contínuo τ incluído na função Q_r. Mesmo com a variação na geração de calor, foi possível manter o limiar de temperatura que causa a morte celular. Neste estudo de caso também foi utilizada uma estratégia de paralelização auxiliada pela API *OpenMP*, porém desta vez foi obtida uma aceleração máxima de aproximadamente 26. Esta simulação foi executada na mesma máquina do estudo anterior, ou seja, com as mesmas restrições quanto ao número de FPUs disponíveis. Deve-se destacar que esta simulação demanda maior utilização da unidade de ponto flutuante, devido à escolha da função Q_r, o que pode ser uma possível explicação para a aceleração do tempo de execução ser inferior a do estudo 1.

Por fim, o último estudo também simulou o tratamento de tumores com hipertermia, porém neste caso, foi considerado um domínio contendo pele, gordura, músculo e tumor. Neste estudo se empregou um método numérico mais preciso na marcha do tempo, que no caso linear garante uma convergência quadrática. Na discretização espacial também utilizou-se a relação dada pela Eq. (3.14), ou seja, convergência quadrática no espaço. Com a aplicação deste método numérico foi possível obter convergência $O(\Delta^2, \Delta_t^2)$.

Nesta simulação, foi comparada a utilização do modelo linear com o não linear.

Esta análise revelou que a perfusão sanguínea dependente da temperatura exerce grande influência no tratamento do tumor, ou seja, na equação transiente. A temperatura variou significativamente nas regiões ao redor do ponto de aplicação das nanopartículas. Além disto, este estudo utilizou a solução no estado estacionário como condição inicial do tratamento com hipertermia. A temperatura inicial é ligeiramente mais alta devido às alterações metabólicas causadas pela existência do tumor, fornecendo uma condição inicial mais realística para o processo de hipertermia.

Este estudo também apresentou uma implementação eficiente do método numérico aplicado ao modelo de biotransferência de calor, por meio da plataforma de programação paralela CUDA. O principal objetivo foi mostrar que é possível obter um alto desempenho, equilibrando com o custo/benefício da utilização de uma placa de vídeo, para executar cálculos em paralelo. De fato, o código desenvolvido apresentou uma aceleração no tempo de execução de 242 vezes, utilizando como plataforma computacional a placa Tesla M2075, quando comparada à versão serial, que utiliza o processador AMD Opteron 6272. Esta aceleração proporcionou a execução de 50min de simulação, que antes eram executados em aproximadamente 92h, em apenas 23min, obtendo uma simulação mais rápida que o tempo real.

6.1 Trabalhos Futuros

Apesar dos bons resultados obtidos nas simulações utilizando placas gráficas, segundo o CUDA Profiler (NVIDIA, 2014c), o acesso à memória possui vazão baixa. Neste caso, pode ser sugerida a utilização de técnicas para reduzir ainda mais os custos do acesso à memória global, *e.g*, utilizar a memória compartilhada em dados comuns às *threads* de um mesmo bloco. Além disto, em todas as simulações com a plataforma CUDA foi utilizada somente uma placa de vídeo, o que poderia inviabilizar simulações com mais pontos na malha por conta de restrições relativas à quantidade total de memória disponível por placa. Para isto, podem-se empregar múltiplas GPUs, aumentando a disponibilidade de memória global, bem como oferecendo mais núcleos CUDA em uma única simulação, reduzindo potencialmente o tempo de computação. Além de técnicas

mais avançadas na computação do problema, também podem ser aplicados diferentes modelos matemáticos, comparando, por exemplo, um modelo de meio poroso (Khaled e Vafai, 2003; Amiri e Vafai, 1994) com o modelo clássico de Pennes, para a simulação de biotransferência de calor em tratamentos baseados em hipertermia. Por fim, outros métodos numéricos também podem ser empregados na solução numérica dos modelos matemáticos, por exemplo, o método das diferenças finitas generalizada (Benito *et al.*, 2001) que possui a vantagem de poder representar o domínio com uma discretização não uniforme, além de ser um método *meshless*.

Referências Bibliográficas

Amiri, A., Vafai, K., 1994. Analysis of dispersion effects and non-thermal equilibrium, non-Darcian, variable porosity incompressible flow through porous media. International Journal of Heat and Mass Transfer 37 (6), 939 – 954.

Benito, J., Ureña, F., Gavete, L., 2001. Influence of several factors in the generalized finite difference method. Applied Mathematical Modelling 25 (12), 1039 – 1053.

Busch, S., Kirillin, G., Mehner, T., 2012. Plasticity in habitat use determines metabolic response of fish to global warming in stratified lakes. Oecologia 170 (1), 275–287.

Cao, L., Qin, Q.-H., Zhao, N., 2010. An RBF-MFS model for analysing thermal behavior of skin tissues. International Journal of Heat and Mass Transfer 53, 2827–2839.

Corry, P. M., Barlogie, B., Tilchen, E. J., Armour, E. P., 1982. Ultrasound-induced hyperthermia for the treatment of human superficial tumors. International Journal of Radiation Oncology*Biology*Physics 8 (7), 1225 – 1229.

Erdmann, B., Lang, J., Seebass, M., 1998. Optimization of Temperature Distributions for Regional Hyperthermia Based on a Nonlinear Heat Transfer Modela. Annals of the New York Academy of Sciences 858 (1), 36–46.

Falk, P., 1978. Patterns of vasculature in two pairs of related fibrosarcomas in the rat and their relation to tumour responses to single large doses of radiation. European Journal of Cancer (1965) 14 (3), 237 – 250.

Freeman, M. L., Spitz, D. R., Meredith, M. J., 1990. Does heat shock enhance oxidative stress? Studies with ferrous and ferric iron. Radiation research 124 (3), 288–293.

Gilchrist, R. K., Medal, R., Shorey, W. D., Hanselman, R. C., Parrott, J. C., Taylor, C. B., 1957. Selective Inductive Heating of Lymph Nodes. Ann. Surg. 146 (4), 596–606.

Gneveckow, U., Jordan, A., Scholz, R., Brüß, V., Waldöfner, N., Ricke, J., Feussner, A., Hildebrandt, B., Rau, B., Wust, P., 2004. Description and characterization of the novel hyperthermia- and thermoablation-system MFH®300F for clinical magnetic fluid hyperthermia. Medical Physics 31 (6), 1444–1451.

Gowrishankar, T., Stewart, D., Martin, G., Weaver, J., 2004. Transport lattice models of heat transport in skin with spatially heterogeneous, temperature-dependent perfusion. BioMedical Engineering OnLine 3 (1).

Hergt, R., Andra, W., d'Ambly, C. G., Hilger, I., Kaiser, W. A., Richter, U., Schmidt, H.-G., 1998. Physical limits of hyperthermia using magnetite fine particles. Magnetics, IEEE Transactions on 34 (5), 3745–3754.

Hilger, I., Hergt, R., Kaiser, W. A., 2005. Towards breast cancer treatment by magnetic heating. Journal of Magnetism and Magnetic Materials 293 (1), 314 – 319, proceedings of the Fifth International Conference on Scientific and Clinical Apllications of Magnetic Carriers.

Holmes, M., 2007. Introduction to Numerical Methods in Differential Equations. Texts in Applied Mathematics. Springer.

Hughes, T. J., 2000. The finite element method: linear static and dynamic finite element analysis. Courier Dover Publications.

Ishida, H., Hachiga, T., Andoh, T., Akiguchi, S., 2012. In-vivo visualization of melanoma tumor microvessels and blood flow velocity changes accompanying tumor growth. Journal of Applied Physics 112 (10), –.

Jiji, L. M., 2009. Heat Conduction. Springer-Verlag Berlin Heidelberg.

Johannsen, M., Thiesen, B., Gneveckow, U., Taymoorian, K., Waldöfner, N., Scholz, R., Deger, S., Jung, K., Loening, S. A., Jordan, A., 2006. Thermotherapy using magnetic

nanoparticles combined with external radiation in an orthotopic rat model of prostate cancer. The Prostate 66 (1), 97–104.

Johannsen, M., Thiesen, B., Jordan, A., Taymoorian, K., Gneveckow, U., Waldöfner, N., Scholz, R., Koch, M., Lein, M., Jung, K., Loening, S. A., 2005. Magnetic fluid hyperthermia (MFH)reduces prostate cancer growth in the orthotopic Dunning R3327 rat model. The Prostate 64 (3), 283–292.

Jordan, A., Scholz, R., Maier-Hauff, K., van Landeghem, F., Waldoefner, N., Teichgraeber, U., Pinkernelle, J., Bruhn, H., Neumann, F., Thiesen, B., von Deimling, A., Felix, R., 2006. The effect of thermotherapy using magnetic nanoparticles on rat malignant glioma. Journal of Neuro-Oncology 78 (1), 7–14.

Jordan, A., Scholz, R., Wust, P., Schirra, H., Schiestel, T., Schmidt, H., Felix, R., 1999. Endocytosis of dextran and silan-coated magnetite nanoparticles and the effect of intracellular hyperthermia on human mammary carcinoma cells in vitro. Journal of Magnetism and Magnetic Materials 194 (1-3), 185 –196.

Karaa, S., Zhang, J., Yang, F., 2005. A numerical study of a 3D bioheat transfer problem with different spatial heating. Mathematics and Computer in simulations 68, 375–388.

Kawai, N., Ito, A., Nakahara, Y., Futakuchi, M., Shirai, T., Honda, H., Kobayashi, T., Kohri, K., 2005. Anticancer effect of hyperthermia on prostate cancer mediated by magnetite cationic liposomes and immune-response induction in transplanted syngeneic rats. The Prostate 64 (4), 373–381.

Khaled, A.-R., Vafai, K., 2003. The role of porous media in modeling flow and heat transfer in biological tissues. International Journal of Heat and Mass Transfer 46 (26), 4989 – 5003.

Kirk, D. B., Hwu, W.-m. W., 2010. Programming Massively Parallel Processors: A Hands-on Approach, primeira Edição. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Lang, J., Erdmann, B., Seebass, M., Impact of Nonlinear Heat Transfer on Temperature Control in Regional Hyperthermia. In: IEEE Trans. on Biomedical Engineering, 1997. pp. 97–73.

Liu, J., Xu, L. X., 2000. Boundary information based diagnostics on the thermal states of biological bodies. International Journal of Heat and Mass Transfer 43 (16), 2827 – 2839.

MagForce, 2014. MagForce Nanotechnologies AG. Publicação Eletrônica - http://www.magforce.de/. Último acesso em 5 de outubro de 2014.

Matsuki, H., Yanada, T., Sato, T., Murakami, K., Minakawa, S., 1994. Temperature-sensitive amorphous magnetic flakes for intratissue hyperthermia. Materials Science and Engineering: A 181?182 (0), 1366 – 1368, proceedings of the Eighth International Conference on Rapidly Quenched and Metastable Materials: Part 2.

Minkowycz, W., Sparrow, E. M., Abraham, J. P., 2012. Nanoparticle Heat Transfer and Fluid Flow. Vol. 4. CRC Press.

Mital, M., Tafreshi, H. V., 2012. A methodology for determining optimal thermal damage in magnetic nanoparticle hyperthermia cancer treatment. International Journal for Numerical Methods in Biomedical Engineering 28 (2), 205–213.

Moros, E., 2012. Physics of Thermal Therapy: Fundamentals and Clinical Applications. CRC Press.

Moroz, P., Jones, S. K., Gray, B. N., 2002. Magnetically mediated hyperthermia: current status and future directions. International Journal of Hyperthermia 18 (4), 267–284, pMID: 12079583.

Nathan, D. G., 2007. The Cancer Treatment Revolution: How Smart Drugs and Other New Therapies are Renewing Our Hope and Changing the Face of Medicine. Wiley.

NumericalAlgorithmsGroup, nov. 2011. How to make best use of the AMD Interlagos processor. Rel. tec.

NVIDIA, 2014a. CUDA Toolkit. Publicação Eletrônica - https://developer.nvidia.com/cuda-toolkit. Último acesso em 9 de setembro de 2014.

NVIDIA, 2014b. Parallel Programming and Computing Platform. Publicação Eletrônica - http://www.nvidia.com/object/cuda_home_new.html. Último acesso em 9 de setembro de 2014.

NVIDIA, 2014c. Programming Guide::CUDA Toolkit Documentation. Publicação Eletrônica - http://docs.nvidia.com/cuda/cuda-c-programming-guide. Último acesso em 9 de setembro de 2014.

OMS, 2014. Organização Mundial da Saúde. Publicação Eletroñica - http://www.who.int/. UÍltimo acesso em 6 de Junho de 2014.

Onishi, T., Machida, T., Iizuka, N., Nakauchi, K., Shirakawa, H., Masuda, F., Mochizuki, S., Tsukamoto, H., Harada, N., 1990. Influence of differences in tumor vascularity upon the effects of hyperthermia. Urological Research 18 (5), 313–318.

Ozisik, M. N., 1994. Finite Difference Methods in Heat Transfer, primeira Edição. Vol. 1. CRC Press.

Pacheco, P., 2011. An Introduction to Parallel Programming, primeira Edição. Vol. 1. MK.

Pennes, H. H., 1948. Analysis of tissue and arterial blood temperature in the restind human forearm. Journal of Applied Phisiology 1, 93–122.

Peterson, H.-I., Appelgren, K., Rudenstam, C.-M., Lewis, D. H., 1969. Studies on the circulation of experimental tumours: I. Effect of induced fibrinolysis and antifibrinolysis on capillary blood flow and the capillary transport function of two experimental tumours in the mouse. European Journal of Cancer (1965) 5 (2), 91 – 97.

Reis, R. F., Loureiro, F. S., Lobosco, M., 2014a. A Parallel 2D Numerical Simulation of Tumor Cells Necrosis by Local Hyperthermia. Journal of Physics: Conference Series

490 (1), 012138. Disponivel em: http://iopscience.iop.org/1742-6596/490/1/012138

Reis, R. F., Loureiro, F. S., Lobosco, M., Solução Numérica Paralela de um Modelo de Biotransferência de Calor Tridimensional Não Linear. In: XI SIMMEC/II EMMECOMP, 2014b. Juiz de Fora, *Anais*. Disponivel em: http://www.ufjf.br/simmec-emmcomp-2014/artigos/

Rosensweig, R., 2002. Heating magnetic fluid with alternating magnetic field. Journal of Magnetism and Magnetic Materials 252 (0), 370 – 374, proceedings of the 9th International Conference on Magnetic Fluids.

Salloum, M., Ma, R., Zhu, L., 2008. An in-vivo experimental study of temperature elevations in animal tissue during magnetic nanoparticle hyperthermia. International Journal of Hyperthermia 24 (7), 589–601.

Salloum, M., Ma, R., Zhu, L., 2009. Enhancement in treatment planning for magnetic nanoparticle hyperthermia: Optimization of the heat absorption pattern. International Journal of Hyperthermia 25 (4), 309–321.

Sanders, J., Kandrot, E., 2010. CUDA by Example, primeira Edição. Addison-Wesley Professional.

Song, C. W., Lokshina, A., Rhee, J. G., Patten, M., Levitt, S. H., 1984. Implication of blood flow in hyperthermic treatment of tumors. Biomedical Engineering, IEEE Transactions on BME-31 (1), 9–16.

Storm, F. K., Harrison, W. H., Elliott, R. S., Morton, D. L., 1979. Normal Tissue and Solid Tumor Effects of Hyperthermia in Animal Models and Clinical Trials. Cancer Research 39 (6 Part 2), 2245–2251.

TechnologyPod, 2014. Do you know what moore's law is? Publicação Eletroñica - http://www.techypod.com/2012/01/do-you-know-what-moores-law-is.html. Último acesso em 13 de agosto de 2014.

Thomlinson, R. H., Gray, L. H., 1955. The Histological Structure of Some Human Lung Cancers and the Possible Implications for Radiotherapy. British Journal of Cancer 9, 539–549.

Tonti, E., 2014. Why starting from differential equations for computational physics? Journal of Computational Physics 257, Part B (0), 1260 – 1290, physics-compatible numerical methods.

Vu, V. T., Cats, G., Wolters, L., 2013. Graphics processing unit optimizations for the dynamics of the HIRLAM weather forecast model. Concurrency and Computation: Practice and Experience 25 (10), 1376–1393.

Wittenbrink, C., Kilgariff, E., Prabhu, A., March 2011. Fermi GF100 GPU Architecture. Micro, IEEE 31 (2), 50–59.

Yang, H., 2012. Mathematical modeling of solid cancer growth with angiogenesis. Theoretical Biology and Medical Modelling 9 (1), 2.

APÊNDICE A - Fórmulas de Diferenças Finitas

Considere os pontos x_0, x_1,..., x_i, ..., x_{I_x} igualmente espaçados e com um passo de tamanho $\Delta x = x_{i+1} - x_i$, onde I_x é o número de subdivisões do domínio. O ponto η_i está localizado entre o ponto mais à esquerda e o ponto mais à direita usado na fórmula (Holmes, 2007).

Tabela A.1: Fórmulas de Diferenças Finitas

Tipo	**Fórmulas de Diferenças Finitas**	**Termo de Trunc.**
Progressiva	$f'(x_i) = \frac{f(x_{i+1}) - f(x_i)}{h} + \tau_i$	$\tau_i = -\frac{h}{2} f''(\eta_i)$
Regressiva	$f'(x_i) = \frac{f(x_i) - f(x_{i-1})}{h} + \tau_i$	$\tau_i = -\frac{h}{2} f''(\eta_i)$
Centrada	$f'(x_i) = \frac{f(x_{i+1}) - f(x_{i-1})}{2h} + \tau_i$	$\tau_i = -\frac{h^2}{6} f'''(\eta_i)$
Lateral à direita	$f'(x_i) = \frac{-f(x_{i+2}) + 4f(x_{i+1}) - 3f(x_i)}{2h} + \tau_i$	$\tau_i = -\frac{h^2}{3} f'''(\eta_i)$
Lateral à esquerda	$f'(x_i) = \frac{3f(x_i) - 4f(x_{i-1}) + 3f(x_{i-2})}{2h} + \tau_i$	$\tau_i = -\frac{h^2}{3} f'''(\eta_i)$
Centrada	$f''(x_i) = \frac{f(x_{i+1}) - 2f(x_i) + f(x_{i-1})}{h^2} + \tau_i$	$\tau_i = -\frac{h^2}{12} f''''(\eta_i)$

APÊNDICE B - Série de Taylor

A série de Taylor associada a uma função $f : \mathbb{R}^n \to \mathbb{R}$ infinitamente diferenciável (real ou complexa), definida em uma bola aberta centrada em $\vec{x_0}$ e raio r, ou seja, $B(\vec{x_0}, r)$, onde a série de potências de $f : \mathbb{R} \to \mathbb{R}$, é dada por:

$$f(x) = f(a)(x-a)^0 + \frac{f'(a)(x-a)^1}{1!} + \frac{f''(a)(x-a)^2}{2!} + \cdots + \frac{f^{(n)}(a)(x-a)^n}{n!},$$

ou, também:

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n \tag{B.1}$$

onde a é o ponto central da expansão em série de Taylor, $f^{(n)}(a)$ é a n-ésima derivada de f no ponto a e $n!$ é o fatorial de n.

No caso da série de Taylor para $f : \mathbb{R}^n \to \mathbb{R}$, tem-se:

$$f(\vec{x} + \delta\vec{x})f(\vec{x}) + \delta\vec{x} \cdot \nabla f + \frac{1}{2!}\delta\vec{x} \cdot H\delta\vec{x} + \cdots,$$

onde $\delta\vec{x} = (\delta x_1, \delta x_2, \cdots, \delta x_n)$, $\vec{x} = (x_1, x_2, \cdots, x_n)$ e H é a matriz Hessiana com componentes $H_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j}$.

APÊNDICE C - Estabilidade do Método Numérico Aplicado à Equação de Pennes

Para se obter um resultado com significado coerente, utilizado o método de diferenças finitas conforme à Eq. (3.20), deve-se estabelecer critérios de estabilidade.

Pode-se fazer no modelo de biotransferência de calor utilizado neste trabalho algo análogo ao que foi feito em Ozisik (1994, p.154). Considerando o caso unidimensional da Eq. (2.23), a forma discreta utilizando Euler (diferença progressiva) na marcha temporal e a diferença central de segunda ordem no espaço para o caso homogêneo é dada pela Eq. (C.1).

$$T_i^{n+1} = T_i^n + \frac{\Delta t}{\rho_i^n c_i^n}\left[\frac{k}{\Delta x^2}(T_{i-1}^n - 2T_i^n + T_{i+1}^n) + (\omega_b)_i^n(\rho_b)_i^n(c_b)_i^n(T_a - T_i^n) + (Q_m)_i^n + (Q_r)_i^n\right]. \tag{C.1}$$

Desconsiderando os termos de fonte e temperatura arterial, pois não influenciam no crescimento e propagação dos erros, e trocando o índice i por j para distinguir o subscrito, com $i = \sqrt{-1}$, obtém-se a Eq. (C.2).

$$T_i^{n+1} = T_i^n + r_x(T_{i-1}^n - 2T_i^n + T_{i+1}^n) - r_T T_i^n, \tag{C.2}$$

onde $r_x = \frac{k\Delta t}{\rho_i^n c_i^n \Delta x^2}$ e $r_T = \frac{(\omega_b)_i^n(\rho_b)_i^n(c_b)_i^n \Delta t}{\rho_i^n c_i^n}$.

Segundo o método de análise de estabilidade de Fourier, o termo de erro é da seguinte forma:

$$\varepsilon_j^n = \xi e^{i\beta_m \Delta x}, \tag{C.3}$$

onde $\xi = e^{\gamma \Delta t}$, $i = \sqrt{-1}$ e β_m o modo de Fourier. Conforme a definição de ξ, o termo de erro ε_j^n não irá crescer ilimitadamente conforme t aumenta. Uma vez que:

$$|\xi| \leq 1, \tag{C.4}$$

o erro também deve satisfazer a equação de diferenças dada pela Eq. (C.2). Portanto, substituindo a Eq. (C.3) em Eq. (C.2) e rearranjando os termos obtém-se:

$$\xi = 1 + r_x(e^{-i\beta_m \Delta x} + e^{i\beta_m \Delta x} - 2) - r_T,$$

o qual pode ser reescrito como:

$$\xi = 1 - 2r_x(1 - cos(\beta_m \Delta x)) - r_T, \tag{C.5}$$

uma vez que, $cos(z) = \frac{1}{2}(e^{-iz} + e^{iz})$.

Aplicando o critério de estabilidade definido pela Eq. (C.4) na Eq. (C.5) implica que:

$$-1 \leq 1 - 2r_x(1 - cos(\beta_m \Delta x)) - r_T \leq 1$$

o qual precisa satisfazer todos os valores de β_m. Majorando o valor do cosseno tem-se $(1 - cos(\beta_m \Delta x)) = 2$, então:

$$-1 \leq 1 - 4r_x - r_T \leq 1,$$

ou seja,

$$\frac{1}{2} \geq r_x + \frac{r_T}{4} \geq 0.$$

Como os termos r_x e r_T sempre são positivos, tem-se:

$$r_x + \frac{r_T}{4} \leq \frac{1}{2}.$$

Por fim, rearranjando os termos, obtém-se:

$$\Delta t\left(\frac{\delta}{\Delta x^2}+\frac{\eta}{4}\right)\leq\frac{1}{2}, \tag{C.6}$$

onde $\delta=\frac{k}{\rho_i^n c_i^n}$ e $\eta=\frac{(\omega_b)_i^n(\rho_b)_i^n(c_b)_i^n}{\rho_i^n c_i^n}$.