Bruno Gouvêa de Barros

# Simulações computacionais de arritmias cardíacas em ambientes de computação de alto desempenho do tipo Multi-GPU

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. Rodrigo Weber dos Santos

Coorientador: Prof. D.Sc. Sergio Alonso Munoz

Coorientador: Prof. D.Sc. Marcelo Lobosco

Juiz de Fora

2013

Gouvêa de Barros, Bruno.
Simulações computacionais de arritmias cardíacas em ambientes de computação de alto desempenho do tipo Multi-GPU / Bruno Gouvêa de Barros. -- 2013.
95 p. : il.

Orientador: Rodrigo Weber dos Santos
Coorientador: Marcelo Lobosco
Dissertação (mestrado acadêmico) - Universidade Federal de Juiz de Fora, Faculdade de Engenharia. Programa de Pós-Graduação em Modelagem Computacional, 2013.

1. Eletrofisiologia cardíaca. 2. Equações diferenciais. 3. Multi-GPU. 4. Arritmia cardíaca. I. dos Santos, Rodrigo Weber, orient. II. Lobosco, Marcelo, coorient. III. Título.

de Barros, Bruno Gouvêa

Simulações computacionais de arritmias cardíacas em ambientes de computação de alto desempenho do tipo Multi-GPU/Bruno Gouvêa de Barros. – Juiz de Fora: UFJF/MMC, 2013.

XIII, 95 p.: il.; 29, 7cm.

Orientador: Rodrigo Weber dos Santos

Coorientador: Sergio Alonso Munoz

Coorientador: Marcelo Lobosco

Dissertação (mestrado) – UFJF/MMC/Programa de Modelagem Computacional, 2013.

Referências Bibliográficas: p. 82 – 91.

1. Eletrofisiologia cardíaca. 2. Equações diferenciais. 3. Multi-GPU. 4. Arritmia cardíaca. I. dos Santos, Rodrigo Weber *et al.*. II. Universidade Federal de Juiz de Fora, MMC, Programa de Modelagem Computacional.

Bruno Gouvêa de Barros

**Simulações computacionais de arritmias cardíacas em ambientes de computação de alto desempenho do tipo Multi-GPU**

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 25 de Fevereiro de 2013.

BANCA EXAMINADORA

---

Prof. D.Sc. Rodrigo Weber dos Santos - Orientador
Universidade Federal de Juiz de Fora

---

Prof. D.Sc. Sergio Alonso Munoz - Coorientador
PTB-Berlin

---

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

---

Prof. D.Sc. Philippe Olivier Alexandre Navaux
Universidade Federal do Rio Grande do Sul

---

Prof. D.Sc. Saul de Castro Leite
Universidade Federal de Juiz de Fora

*Dedico este trabalho a toda a minha família, a meus amigos, a meus professores e em especial a meus orientadores.*

## AGRADECIMENTOS

Mais um capítulo da minha vida chega ao fim no dia de hoje, termino ele mais preparado e ainda mais motivado para continuar crescendo profissionalmente, por isso gostaria de agradecer a todos que me apoiaram e estiveram do meu lado, pois foram fundamentais para mais essa conquista. Gostaria de agradecer primeiramente a minha mãe que me deu todo apoio, amor, incentivo e sempre esteve presente para o que eu precisasse, sem ela eu nunca teria chegado onde cheguei. Agradeço também a toda minha família e todos os meus amigos que sempre me apoiaram e me deram força nos momentos mais difíceis. Em especial gostaria de agradecer ao meu orientador Rodrigo que além do conhecimento transmitido, sempre me apoiou e me incentivou. Gostaria de agradecer também aos meus coorientadores Lobosco e Sergio, aos demais professores e aos colegas do Fisiocomp, todos vocês foram indispensáveis em minha formação.

## RESUMO

Os modelos computacionais tornaram-se ferramentas valiosas para o estudo e compreensão dos fenômenos da eletrofisiologia cardíaca. No entanto, a elevada complexidade dos processos biofísicos e o nível microscópico de detalhes exigem complexos modelos computacionais. Aspectos-chave da eletrofisiologia cardíaca, tais como condução lenta e bloqueio de condução tem sido tema de pesquisa de muitos estudos, uma vez que estão fortemente relacionados à arritmia cardíaca. No entanto, ao reproduzir estes fenômenos os modelos necessitam de uma discretização sub-celular para a solução das equações diferenciais e uma condutividade eléctrica do tecido não uniforme e heterogênea. Devido aos elevados custos computacionais de simulações que reproduzem a microestrutura fina do tecido cardíaco, estudos prévios têm considerado experimentos de tecido de pequenas dimensões e têm utilizados modelos simples de células cardíacas. Neste trabalho, desenvolvemos um modelo (modelo microscópico) da eletrofisiologia cardíaca que capta a microestrutura do tecido cardíaco usando uma discretização espacial muito fina ($8\mu m$) e utilizamos um modelo celular moderno e complexo baseado em Cadeias de Markov para a caracterização da estrutura e dinâmica dos canais iônicos. Para lidar com os desafios computacionais, o modelo foi paralelizado usando uma abordagem híbrida: a computação em *cluster* e GPGPUs (*General-purpose computing on Graphics Processing Units*). Nossa implementação paralela deste modelo, utilizando uma plataforma multi-GPU, foi capaz de reduzir os tempos de execução das simulações de mais de 6 dias (em um único processador) para 21 minutos (em um pequeno *cluster* de 8 nós equipado com 16 GPUs). Além disso, para diminuir ainda mais o custo computacional, foi desenvolvido um modelo discreto equivalente ao modelo microscópico. Este novo modelo foi paralelizado usando a mesma abordagem do modelo microscópico e foi capaz de executar simulações que demoravam 21 minutos em apenas 65 segundos. Acreditamos que esta nova implementação paralela abre caminho para a investigação de muitas questões em aberto associadas à natureza complexa e discreta da propagação dos potenciais de ação no tecido cardíaco.



**Palavras-chave:** Eletrofisiologia cardíaca. Equações diferenciais. Multi-GPU. Arritmia cardíaca.

## ABSTRACT

Computer models have become valuable tools for the study and comprehension of the complex phenomena of cardiac electrophysiology. However, the high complexity of the biophysical processes and the microscopic level of details demand complex mathematical and computational models. Key aspects of cardiac electrophysiology, such as slow conduction, conduction block and saltatory effects have been the research topic of many studies since they are strongly related to cardiac arrhythmia. However, to reproduce these phenomena the numerical models need to use sub-cellular discretization for the solution of the PDEs and nonuniform, heterogeneous tissue electric conductivity. Due to the high computational costs of simulations that reproduce the fine microstructure of cardiac tissue, previous studies have considered tissue experiments of small or moderate sizes and used simple cardiac cell models. In this work we develop a cardiac electrophysiology model (microscopic model) that captures the microstructure of cardiac tissue by using a very fine spatial discretization ($8\mu m$) and uses a very modern and complex cell model based on Markov Chains for the characterization of ion channel's structure and dynamics. To cope with the computational challenges, the model was parallelized using a hybrid approach: cluster computing and GPGPUs (General-purpose computing on graphics processing units). Our parallel implementation of this model using a Multi-GPU platform was able to reduce the execution times of the simulations from more than 6 days (on a single processor) to 21 minutes (on a small 8-node cluster equipped with 16 GPUs). Furthermore, in order to decrease further the computational cost we have developed a discrete model equivalent to the microscopic one. This new model was also parallelized using the same approach as the microscopic model and was able to perform simulations that took 21 minutes to be executed in just 65 seconds. We believe that this new parallel implementation paves the way for the investigation of many open questions associated to the complex and discrete propagation nature of action potentials on cardiac tissue.



**Keywords:** Cardiac electrophysiology. Differential equations. Multi-GPU. Cardiac arrhythmia.

## SUMÁRIO

# LISTA DE ILUSTRAÇÕES

## LISTA DE TABELAS

## LISTA DE SIGLAS

**GPU** *Graphics Processing Unit*

**CPU** *Central Processing Unit*

**GPGPU** *General-purpose computing on Graphics Processing Unit*

**EDP** Equação Diferencial Parcial

**EDO** Equação Diferencial Ordinária

**GHK** Goldman-Hodgkin-Katz

**BDK** Bondarenko

**MPI** *Message Passing Interface*

**CUDA** *Compute Unified Device Architecture*

**SMP** *Symmetric Multi-Processing*

**DP** Desvio Padrão

**PL** Propagação Longitutinal (ao longo das fibras do tecido cardíaco)

**PT** Propagação Transversal (no sentido transversal à direção das fibras do tecido cardíaco)

**MVF** Método dos Volumes Finitos

**MEF** Método dos Elementos Finitos

**MDF** Método das Diferenças Finitas

**CFL** Courant-Friedrichs-Lewy

**RL** Rush-Larsen

**API** *Application Programming Interface*

**RAM** *Random Access Memory*

**GB** *Gigabyte*

# 1 INTRODUÇÃO

As doenças cardíacas são responsáveis por um terço de todas as mortes no mundo [1]. A deformação mecânica do coração está diretamente relacionada à eletrofisiologia cardíaca. Portanto, o conhecimento da eletrofisiologia cardíaca é essencial para compreender muitos aspectos do comportamento fisiológico e fisiopatológico do coração [2]. Os processos biofísicos que estão envolvidos em muitas arritmias, que são disfunções que afetam o padrão excitatório do coração, podem ser estudados através da modelagem do coração. Os modelos computacionais da eletrofisiologia cardíaca [3, 4] se tornaram ferramentas valiosas para o estudo e compreensão de tais fenômenos complexos, pois permitem que diferentes informações adquiridas a partir de diferentes escalas físicas e experimentos possam ser combinados afim de gerar uma imagem melhor da funcionalidade do sistema como um todo. Não surpreendentemente, a alta complexidade dos processos biofísicos se traduz em complexos modelos matemáticos e computacionais. Modelos cardíacos modernos são descritos por sistema de equações diferenciais parciais (EDPs) não-lineares que podem resultar em um problema com milhões de incógnitas.

Modelos matemáticos para eletrofisiologia celular são um componente chave da modelagem cardíaca. Eles servem tanto como ferramentas de investigação independentes, para investigar o comportamento de miócitos cardíacos individuais, quanto como um componente essencial da simulação do tecido e do órgão baseado nos modelos do bidomínio ou do monodomínio [4]. Os modelos celulares podem ser escritos como um sistema genérico de equações diferenciais ordinárias (EDOs) não-lineares e podem variar em complexidade desde simples modelos fenomenológicos [5] (baseados em duas variáveis) para modelos complexos que descrevem um grande número de processos fisiológicos detalhados [6] (baseados em 40 até 80 variáveis diferenciais). Modelos simples se concentram na geração do potencial de ação (PA), que se propaga de célula para célula gerando uma onda elétrica que se propaga pelo coração. Modelos complexos levam em conta não somente a geração do PA, mas também descrevem como esse fenômeno está relacionado à homeostase cardíaca e levam em conta também diferentes componentes sub-celulares, tais como os canais iônicos da membrana celular. Os avanços da genética, biologia molecular e dos experimentos da eletrofisiologia têm fornecido novos dados e informações relacionadas

com a estrutura e função dos canais iônicos. Modelos baseados em Cadeias de Markov têm sido cada vez mais utilizados para descrever tanto a função quanto a estrutura dos canais iônicos. Estes modelos baseados em Cadeias de Markov permitiram simulações de anomalias estruturais devido a doenças genéticas e aos efeitos de fármacos em canais iônicos [7, 8, 9]. Infelizmente, estes modernos modelos de miócitos cardíacos apresentam desafios diferentes tanto para os métodos numéricos, devido à natureza *stiff* das EDOs introduzidas pelas Cadeias de Markov, quanto para a computação de alto desempenho, devido ao tamanho dos problemas, uma vez que o número de variáveis diferenciais sobe de um par para quase uma centena[10].

No nível do tecido o modelo do bidomínio [4] é considerado como sendo a descrição mais completa da atividade elétrica. Este sistema não-linear de EDPs pode ser simplificado para o modelo conhecido como modelo do monodomínio, que pode ser menos preciso, mas também é menos exigente computacionalmente do que o modelo bidomínio. Infelizmente, as simulações de grande escala, tais como os resultantes da discretização de um coração inteiro, continuam a ser um grande desafio computacional. Além disso, os aspectos chave da eletrofisiologia cardíaca, tais como a condução lenta, bloqueio da condução e efeitos saltatórios ou efeitos dente de serra exigem uma discretização sub-celular para a solução das EDPs e uma condutividade elétrica do tecido não uniforme e heterogênea. Estes aspectos da eletrofisiologia cardíaca estão fortemente relacionados com a arritmia cardíaca, a reentrada, a fibrilação ou desfibrilação, e têm sido o tema de pesquisa de muitos estudos [11, 12, 13, 14, 15, 16, 17, 18, 19, 20].

No entanto, a necessidade de uma discretização sub-celular para a solução das EDPs e uma condutividade elétrica do tecido não uniforme e heterogênea têm impedido o estudo dos fenômenos acima mencionados em simulações do tecido cardíaco em grande escala. Além disso, devido aos elevados custos computacionais associados às simulações destes modelos microscópicos de tecidos cardíacos, trabalhos anteriores adotaram modelos simples de miócitos, em vez dos modernos modelos baseados em Cadeias de Markov [6, 10].

Neste trabalho, apresentamos uma solução para este problema, com base em plataformas Multi-GPU (*clusters* equipados com unidades de processamento gráfico), que permitem simulações rápidas de modelos microscópicos do tecido combinados com modelos modernos e complexos dos miócitos. A solução é baseada na fusão de dois diferentes ambientes de computação de alto desempenho que já foram utilizadas em trabalhos

anteriores para modelagem cardíaca [21, 22, 23, 24]: Agregados de computadores; e Unidade de processamento gráfico de propósito geral ou GPGPU (do inglês *General-purpose computing on graphics processing units*) [25, 26, 27, 28, 29, 30]. Desenvolvemos um modelo bidimensional que é baseado em um trabalho anterior de Spach e colaboradores [11, 17] que leva em conta a microestrutura do tecido cardíaco, a distribuição heterogênea das junções comunicantes, *gap junctions* (ver figura 1.1), e uma discretização de 8 $\mu$m. Este modelo microscópico de tecido foi combinado com o modelo de Bondarenko e colaboradores [6] que é um modelo moderno e complexo de miócitos baseado em em Cadeias de Markov. Nossa implementação paralela deste modelo, utilizando uma plataforma multi-GPU foi capaz de reduzir os tempos de execução das simulações de mais de 6 dias (em um único processador) para 21 minutos (em um pequeno *cluster* de 8 nós equipado com 16 GPUs - 2 GPUs por nó). Como resultado, usando esta implementação paralela fomos capazes de simular a formação de ondas em espiral, uma forma de atividade reentrante auto-sustentada fortemente associada à arritmia cardíaca. Tanto quanto é do nosso conhecimento, esta é a primeira vez que as ondas em espiral são simuladas utilizando um modelo cardíaco que inclui a microestrutura do tecido cardíaco e utiliza um modelo moderno e complexo de miócito.

O tecido cardíaco é formado por células excitáveis chamadas miócitos. Os miócitos são distribuídos de forma heterogêneas e estão ligados por *gap junctions*. A matriz extracelular, *gap junctions* não condutoras, vasos sanguíneos e células não excitáveis incorporados no tecido, podem perturbar o acoplamento das *gap junctions* entre os miócitos [31]. A presença de ligações não-condutoras no tecido cardíaco pode afetar fortemente a propagação de ondas e aumenta o risco de arritmias [31].

A origem da arritmia é, naturalmente, devido a múltiplos fatores, envolvendo uma interação entre o acoplamento das *gap junctions*, a excitabilidade da membrana e a arquitetura das células e dos tecidos [32]. Diversas doenças estão associadas à mudanças na distribuição de *gap junctions* e estas alterações levam a um aumento de incidência de episódios de arritmia cardíaca.

Com objetivo de compreender e estudar o fenômeno da arritmia, em particular fenômenos de reentrada, geramos centenas de cenários através de um método estocástico para perturbar a distribuição das *gap junction*, ou seja, para perturbar o acoplamento entre os miócitos. Porém devido a esta natureza estocástica do método e a necessidade

de gerar muitos cenários, um grande número de simulações se torna necessária, e com isso o custo computacional mesmo diminuído pela versão paralela se torna muito alto.

Por isso, com o objetivo de diminuir ainda mais o alto custo computacional, desenvolvemos um modelo discreto equivalente ao modelo microscópico, calculando a condutividade célula a célula do modelo microscópico. Este modelo discreto mostrou-se computacionalmente muito menos custoso que o modelo microscópico, uma vez que simulações como a geração de ondas em espiral, que antes demoravam em torno de 5 horas (em um pequeno *cluster* de 8 nós equipado com 16 GPUs - 2 GPUs por nó), demoram apenas alguns minutos utilizando o modelo discreto paralelo.

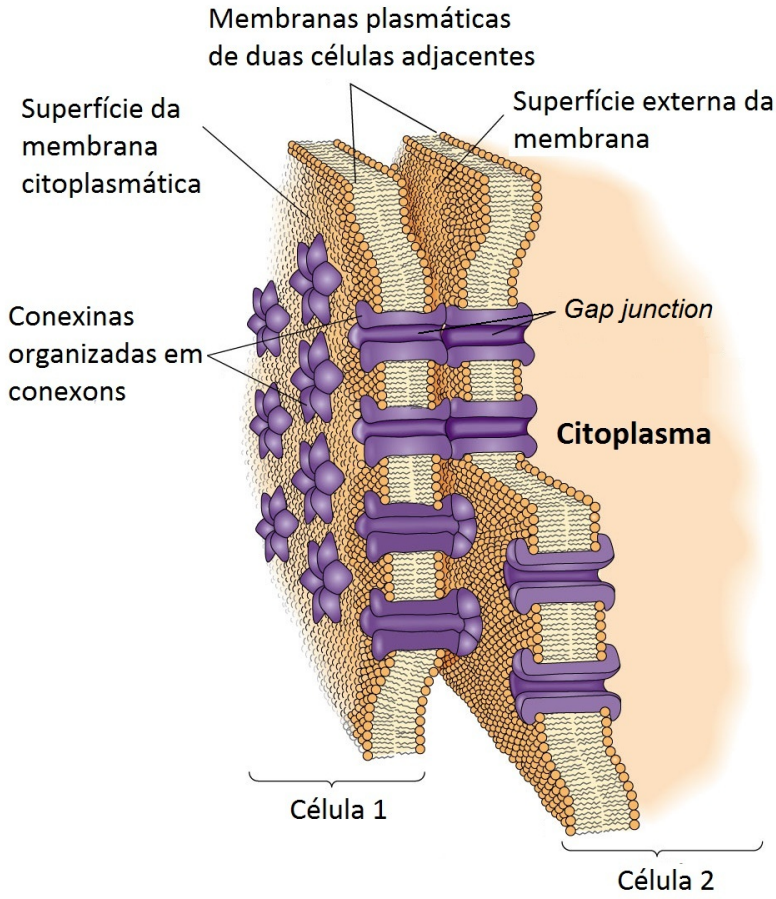


Figura 1.1: Microestrutura do tecido cardíaco onde podemos ver o citoplasma, a membrana celular e as *gap junctions* conectando duas células. Adaptado de [33].

# 2 MODELAGEM DA ELETROFISIOLOGIA CARDÍACA

## 2.1 Introdução

Uma arritmia é uma desordem que envolve o sistema elétrico do coração. Tal desordem leva a batimentos cardíacos que são muito rápidos, muito lentos, ou apenas irregulares e podem resultar em sintomas de fadiga, tonturas, desmaios, palpitações cardíacas, e até mesmo parada cardíaca súbita [34]. Estes problemas elétricos do coração são diferentes dos envolvendo as artérias e músculos do coração.

Eletrofisiologia cardíaca, em termos simples, é o estudo dos impulsos elétricos do coração. É uma sub-especialidade da Cardiologia dedicada exclusivamente ao tratamento de arritmias.

Este capítulo apresenta conceitos básicos sobre a eletrofisiologia cardíaca e sobre a modelagem matemática de células e tecidos cardíacos.

## 2.2 O Coração e o Potencial de Ação

O coração humano tem a função de gerar batimentos rítmicos, em média 70 vezes por minuto, 24 horas por dia, durante 80 anos ou mais, o que resulta em cerca de 3 bilhões de contrações da musculatura cardíaca que devem ocorrer sem falhas.

O coração é uma bomba eletromecânica, e para bombear o sangue, necessita de um estímulo elétrico. Estes estímulos são entregues a cada célula cardíaca (células especializadas eletricamente ativas) através de uma sofisticada rede de distribuição composta por nervos especiais chamado de sistema de condução, permitindo então a contração do coração.

Qualquer perda de atividade elétrica, até mesmo por alguns segundos, resulta em síncope, e a perda de atividade elétrica por alguns minutos pode resultar em morte [35].

O coração é dividido em duas partes, a direita e a esquerda, cada uma delas composta por um átrio e um ventrículo.

Os átrios têm a função de servir como reservatório e via de entrada do sangue para os ventrículos, além disso funcionam como uma bomba fraca, que auxiliam a impulsionar o sangue para o ventrículo. Os ventrículos, por sua vez, fornecem a principal força para propelir o sangue através das circulações pulmonar e periférica[36].

Como podemos observar na figura 2.1, o sangue passa pelo coração duas vezes antes de completar um ciclo completo. Primeiramente, o átrio direito recebe o sangue desoxigenado do corpo e o leva para o ventrículo direito que bombeia-o para os pulmões, esta parte é conhecida como circulação pulmonar. Lá o sangue libera gás carbônico e é oxigenado, retorna ao coração para o átrio esquerdo, que então o leva para o ventrículo esquerdo, onde finalmente é bombeado para o resto do corpo, esta etapa é a circulação periférica.

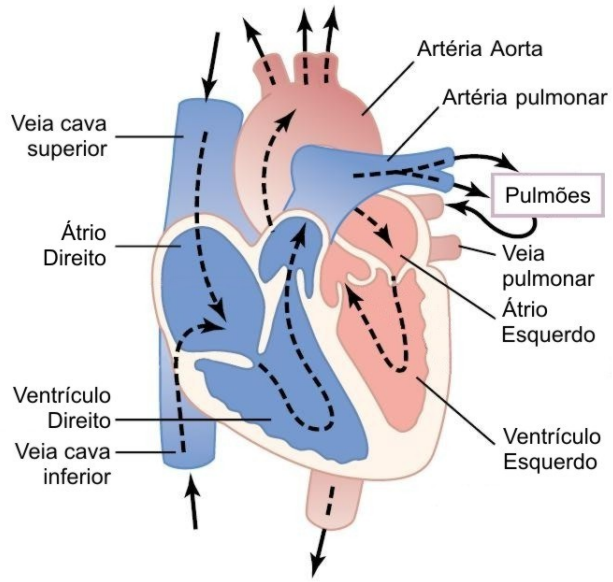


Figura 2.1: Fluxo sanguíneo. Adaptado de [36].

O tecido muscular cardíaco é composto por células, chamadas de miócitos, que pertencem a uma classe especial de células conhecidas como excitáveis. Os miócitos cardíacos possuem potencial de ação característico.

Os miócitos cardíacos se organizam ligando-se uns aos outras em série, na forma de fibras musculares estriadas [37].

Como podemos observar na figura 2.2 os discos intercalados, ou discos Z, são na verdade membranas celulares que separam células individuais umas das outras.

Nessas regiões em que as membranas celulares se fundem, é onde se formam as *gap junctions*. Essas *gap junctions* são muito permeáveis e possibilitam a difusão de íons e outras pequenas moléculas de uma célula para outra. Desta maneira, o tecido muscular

cardíaco é um sincício de muitas células interligadas.

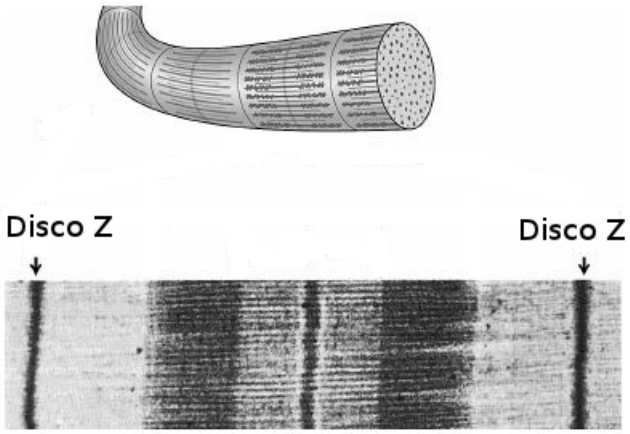


Figura 2.2: Figura mostrando os discos Z de um sarcômero.

Quando células cardíacas são excitadas eletricamente, sofrem uma variação temporal do seu potencial transmembrânico, que é a diferença de potencial através da membrana, causadas por correntes iônicas que cruzam os canais iônicos. Este fenômeno denominado potencial de ação se propaga para as outras células através das *gap junctions* e serve como ativador do processo de contração das células.

A figura 2.3 mostra o potencial de ação cardíaco destacando as fases de cada região, a polaridade da célula, e correntes de entrada e saída de íons em cada fase do potencial de ação.

A entrada de sódio durante da fase B implica em um potencial de membrana positivo em relação ao potencial de equilíbrio para o potássio, desta forma, canais transitórios de sódio se abrem causando uma corrente de saída deste íon, auxiliando na repolarização da célula e contribuindo para o pico da fase C. Porém esses canais transitórios rapidamente se inativam. Durante a fase D, ou platô, há uma corrente despolarizante de cálcio, cuja entrada do íon cálcio estimula a liberação do cálcio armazenado no retículo sarcoplasmático da célula, ocasionando um aumento do nível de cálcio no meio intracelular e promovendo a contração do cardiomiócito. Durante o repouso, a concentração de cálcio é mantida baixa no interior da célula por meio da recaptura ativa do cálcio pelo retículo sarcoplasmático e pela atuação de um trocador sódio-cálcio na membrana celular. A entrada de cálcio nesta fase é contrabalançada principalmente por uma corrente repolarizante de potássio. Os canais de cálcio se inativam com o tempo, enquanto mais canais retificadores de potássio se abrem. Um maior número de canais abertos de potássio resulta na repolarização da célula (fase E). Todo o processo, entre a abertura dos canais

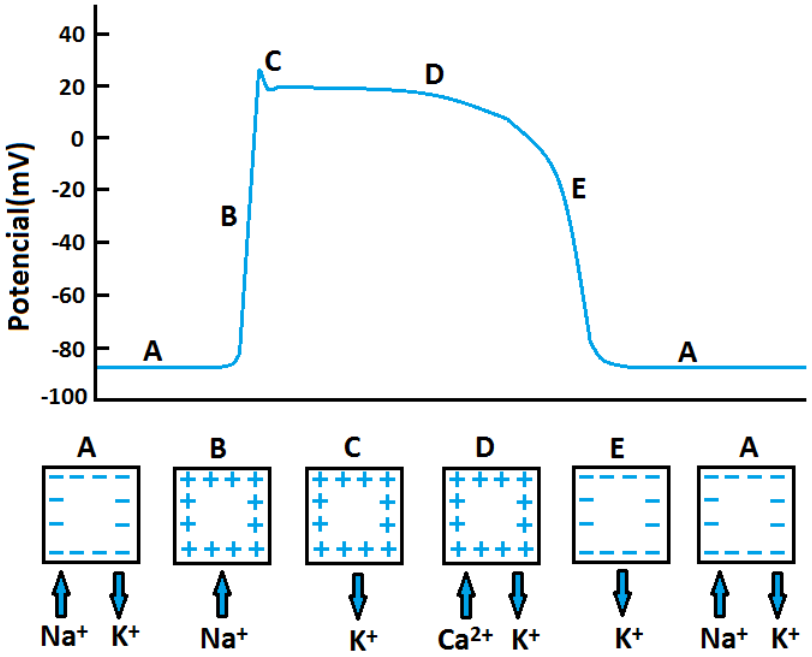


Figura 2.3: Potencial de ação cardíaco passando pelas fases: (A) Repouso, (B) Despolarização, (C) Repolarização Inicial, (D) Platô, (E) Repolarização e (A) Repouso. As correntes iônicas relacionadas com a mudança de potencial em cada fase são mostradas na parte inferior da figura.

de sódio e a repolarização, dura algumas centenas de milissegundos, porém a duração do potencial de ação varia para cada espécie. A cada potencial de ação, o cardiomiócito ganha íons de sódio e perde íons de potássio. No entanto, a manutenção da homeostase (princípio que estabelece que, dentro de determinados limites, o organismo tende a manter a estabilidade do meio interno, apesar das variações no meio externo) intracelular é garantida pelo mecanismo de bomba $Na^+ - K^+$, que retira três íons de sódio da célula para cada dois íons de potássio que entram, atuando também no processo de repolarização celular.

O nódulo sino-atrial, ou sinusal, é uma estrutura anatômica do coração responsável pela geração dos impulsos cardíacos normais. A condução do impulso é dependente da direção, ocorrendo mais rápido no sentido longitudinal (ao longo das células) do que no transversal, pois os cardiomiócitos, com formato longo e fino, são conectados por um maior número de *gap junctions* nas pontas.

Uma característica importante do potencial de ação é o período refratário, caracterizado como o período de tempo no qual a célula não pode ser reexcitada. O período refratário acompanha o potencial de ação na membrana. Tem como efeito limitar

a frequência de potenciais de ação, além de promover a unidirecionalidade da propagação do potencial de ação. O período refratário divide-se em absoluto e relativo. No absoluto, qualquer estímulo para gerar potencial de ação é inútil, pois os canais de sódio estão inativos. No relativo, alguns destes canais já estarão ativos, mas nem todos. Com isso já é possível gerar potenciais de ação, mas as correntes injetadas precisam ter valores maiores do que o usual para que isso ocorra. Durante esse período, o limiar de corrente para geração de um potencial de ação fica acima do valor normal, indo de um valor muito grande no início do período até o valor normal no fim dele. A transição entre os dois períodos ocorre aproximadamente quando a repolarização do potencial de ação atinge o potencial limiar excitatório. Nas células miocárdicas, o período refratário é estendido por um platô, que é mantido pelo influxo de íons cálcio na célula. Esse alargamento do período refratário permite um maior descanso destas células, além de participar na sincronização dos batimentos.

Caso ocorra um atraso suficientemente grande na propagação de uma corrente em uma determinada área do tecido gerado por alguma anormalidade (presença de ligações não condutoras), algumas regiões podem sair do período refratário e podem acabar sendo reexcitadas, gerando uma padrão desordenado na propagação que pode vir a resultar no fenômeno da arritmia de reentrada [38].

## 2.3 Membrana Celular

As células são envoltas por uma membrana que é uma estrutura que serve como uma barreira separando o meio intracelular do meio extracelular, ou seja, a membrana controla o fluxo das substâncias que entram e saem do citoplasma.

Esta membrana é constituída de uma bicamada fosfolipídica como mostra a figura 2.4, onde cada lipídio (figura 2.5) é composto por duas extremidades hidrofóbicas, ligadas a uma extremidade hidrofílica[39].

Devido a estas características anfipática (uma extremidade hidrofílica e a outra hidrofóbica)[40] em um meio aquoso, as extremidades hidrofóbicas do lipídio repelidas pela água se voltam para dentro da bicamada enquanto a extremidade hidrofílica se volta para fora, constituindo assim uma barreira natural que impede a passagem de moléculas carregadas[41].

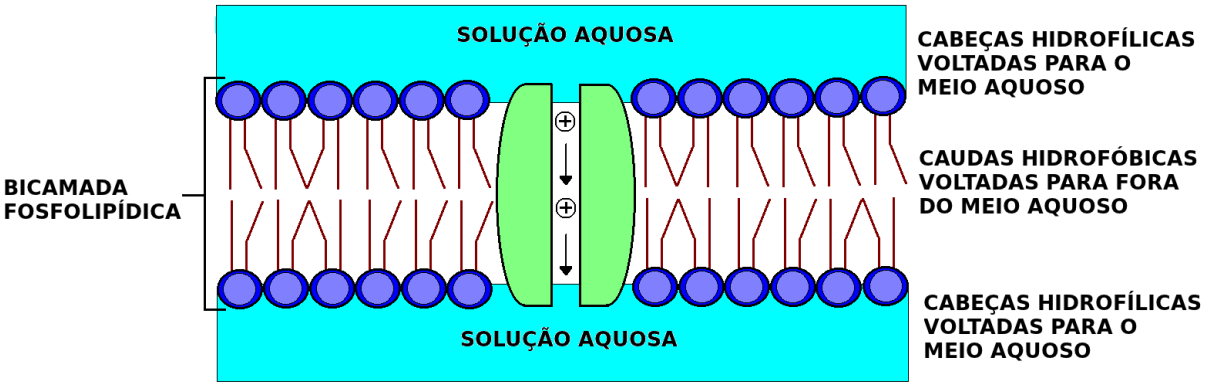


Figura 2.4: Membrana plasmática.

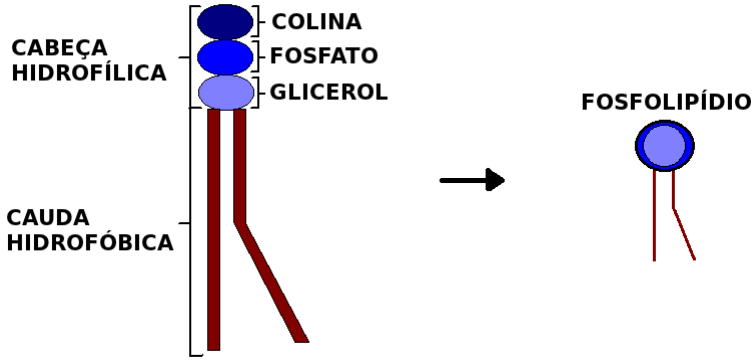


Figura 2.5: Fosfolipídio.

Além da bicamada fosfolipídica, a membrana também é formada por pequenas frações de açúcares e por proteínas, que formam os canais iônicos, que são estruturas responsáveis por controlar o fluxo dos íons através da membrana.

Ambos os meios intracelular e extracelular são soluções aquosas compostas por sais dissolvidos, principalmente $NaCl$ e $KCl$, os quais se dissociam em íons $Na^+$, $K^+$ e $Cl^-$ [42].

A diferença nas concentrações desses íons gera a diferença de potencial através da membrana.

## 2.4 Difusão e Lei de Fick

A difusão é um movimento aleatório que depende da energia térmica de um sistema de partículas e da diferença de concentração entre duas regiões, de modo que o fluxo de partículas de uma região de maior concentração para outra de menor concentração pode

ser entendido como difusão simples ou transporte passivo de partículas.

Quando um sistema apresenta uma diferença no número de moléculas por unidade de volume (concentração), dentro e fora da membrana, se apresenta um gradiente de concentração que é proporcional ao número de partículas que atravessam a membrana por unidade de tempo. Podemos definir uma área unitária perpendicular à direção de difusão, onde a constante de proporcionalidade está dada pelo coeficiente de difusão das moléculas que atravessam a membrana [35]. Esse processo é conhecido como Lei de Fick e sua representação matemática é dada pela equação 2.1:

$$J_F = -D\nabla c \tag{2.1}$$

onde $\nabla c$ é o gradiente de concentração do íon $c$, $D$ é o coeficiente de difusão do meio, $J_F$ é o fluxo dos íons.

## 2.5 Potencial de Equilíbrio de Nernst

A diferença de cargas entre os meios intracelular e extracelular gera uma diferença de potencial que por sua vez gera um campo elétrico, e este campo elétrico gera um força, fazendo com que os íons se desloquem no sentido oposto ao da difusão, corforme representado pela equação 2.2:

$$J_P = -m\frac{z}{|z|}c\nabla V \tag{2.2}$$

onde $m$ é a mobilidade do íon, $z$ é a carga do íon, $|z|$ é a valência do íon, $c$ é a concentração do íon, e $\nabla V$ é o gradiente do potencial elétrico.

Quando os dois fluxos se igualam um equilíbrio é alcançado, conforme ilustrado pela equação 2.3.

$$J = J_F + J_P = 0 \tag{2.3}$$

A mobilidade iônica está relacionada ao coeficiente de difusão[43]. Esta relação foi determinada por Einstein [44] e é dada pela equação 2.4:

$$m = D\frac{|z|F}{RT} \tag{2.4}$$

onde $F$ é a constante de Faraday, $R$ é a constante ideal dos gases, e $T$ é a temperatura absoluta.

Utilizando a relação de Einstein na equação 2.3 chegamos à seguinte expressão:

$$-D\left(\nabla c + \frac{zF}{RT}c\nabla V\right) = 0 \tag{2.5}$$

Simplificando a equação para o caso unidimensional de um canal iônico de comprimento $L$, chegamos na equação 2.6:

$$\frac{dc}{dx} + \frac{zF}{RT}c\frac{dV}{dx} = 0 \tag{2.6}$$

Dividindo os dois lados da equação por $c$, e integrando a expressão com $x = 0$ a $x = L$ temos a equação 2.7:

$$\int_{c(x=0)}^{c(x=L)} \frac{1}{c}dc + \int_{V(x=0)}^{V(x=L)} \frac{zF}{RT}dV = 0 \tag{2.7}$$

calculando as integrais finalmente chegamos a equação 2.8:

$$\ln(c)|_{c(x=0)}^{c(x=L)} = -\frac{zF}{RT}\left[V(x=L) - V(x=0)\right] \tag{2.8}$$

Com isso, considerando o potencial transmembrânico como a diferença entre os potenciais dos meios intracelular e extracelular, $V_m = V_i - V_e$, obtemos a seguinte equação:

$$V_{eq} = \frac{RT}{zF}\ln\left(\frac{c_e}{c_i}\right) \tag{2.9}$$

onde $c_e$ e $c_i$ são as concentrações extracelular e intracelular do íon $c$. O potencial $V_{eq}$, para qual o fluxo é zero, é denominado potencial de equilíbrio de Nernst.

## 2.6 Modelo Elétrico para a Membrana

A membrana, que funciona como um isolante, é formada por duas camadas de lipídeos que separam os meios condutores intra e extracelular. Portanto, a membrana atua como um capacitor como podemos ver na figura 2.6.

A capacitância de um capacitor é medida pelo quociente da quantidade de carga

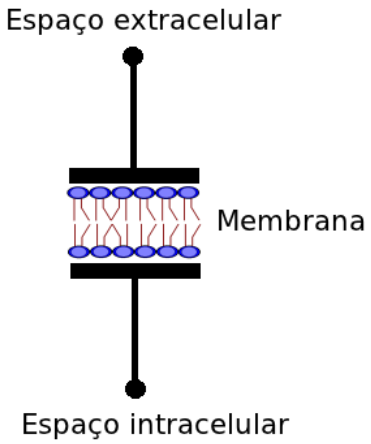


Figura 2.6: Membrana como um capacitor.

armazenada pela diferença de potencial que existe entre as placas:

$$C_m = \frac{Q}{V} \tag{2.10}$$

onde $C_m$ é a capacitância, $Q$ é a carga, e $V$ é a diferença de potencial.

Um modelo de circuito elétrico simplificado para a membrana plasmática pode ser encontrado na figura 2.7. Neste modelo assume-se que a membrana funciona como um capacitor em paralelo com um resistor, que não necessariamente tem características lineares. Este resistor representa as correntes iônicas que atravessam a membrana. Aplicando a lei dos nós de Kirchoff obtemos a seguinte equação:

$$C_m \frac{dV}{dt} + I_{ion} = 0 \tag{2.11}$$

onde $V = V_i - V_e$.

## 2.7 Canais Iônicos

Canais iônicos são poros formados por proteínas que ajudam a estabelecer e controlam a diferença de potencial através da membrana plasmática da célula, permitindo assim o fluxo seletivo de íons. Quando o potencial transmembrânico é diferente do potencial de Nernst (equação 2.9), uma corrente de íons passa por estes canais.

A condutância destes canais aos íons depende do potencial transmembrânico, já que mudanças neste potencial geram alterações na conformação das proteínas que os compõem

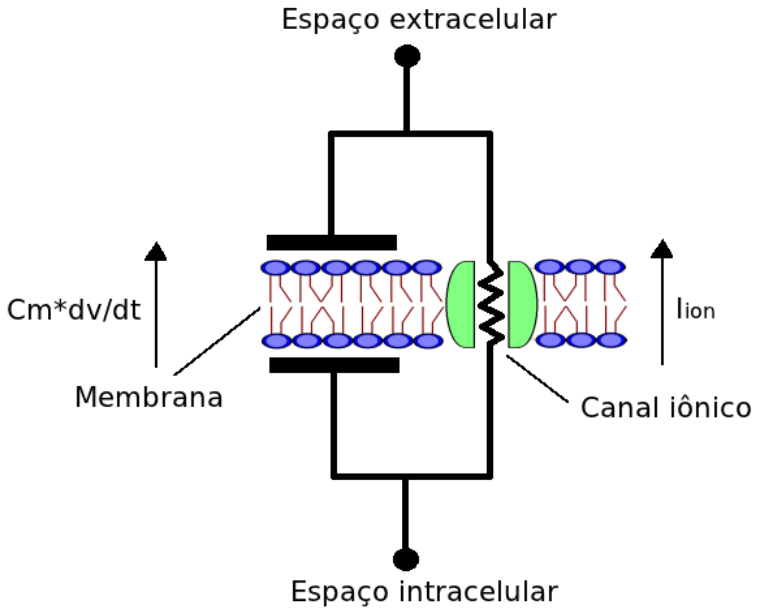


Figura 2.7: Modelo elétrico para a membrana.

e mudam a permeabilidade do canal[40].

Dois modelos serão apresentados para modelar a relação entre as correntes iônicas e o potencial transmembrânico.

O primeiro é conhecido como modelo linear, pois a relação entre corrente e voltagem neste modelo é linear, e é dado pela equação:

$$I = G(V - V_{eq}) \tag{2.12}$$

onde G é a condutância do canal dada pelo inverso da resistência. A determinação deste parâmetro pode ter diferentes abordagens, podendo ser considerado uma constante, função do tempo, do potencial elétrico ou até mesmo de concentrações iônicas[40].

O segundo modelo para a relação entre a corrente iônica e o potencial transmembrânico é a equação não-linear de Goldman-Hodgkin-Katz (GHK), que é deduzido a partir da hipótese de que o campo elétrico é constante na membrana [45], e é dado pela equação:

$$I = P_S \frac{z^2F^2}{RT} V \frac{c_i - c_e \exp(\frac{-zFV}{RT})}{1 - \exp(\frac{-zFV}{RT})} \tag{2.13}$$

onde $P_S = \frac{D}{L}$ é a permeabilidade da membrana ao íon considerado, e $\mathbf{c_i}$ e $\mathbf{c_e}$ são as concentrações interna e externa deste íon.

Tanto no modelo linear quanto no modelo GHK a corrente iônica é igual a zero quando o potencial transmembrânico é igual ao potencial de Nernst do íon em questão. Estes modelos representam correntes em canais isolados e abertos.

No entanto, nas células, esses canais aparecem em conjuntos e são formados por subunidades que podem estar abertas ou fechadas, de maneira que só é possível fluir corrente em um canal quando todas as subunidades estiverem abertas.

Podemos representar esta característica assumindo que um determinado canal pode estar em um estado aberto $O$ ou em um estado fechado $C$ , e que a taxa de transição de um estado para o outro é uma função do potencial transmembrânico $V_m$.

A equação abaixo representa a transição dos estados fechado $C$ e aberto $O$.

$$C \underset{\beta}{\overset{\alpha}{\rightleftharpoons}} O \tag{2.14}$$

Sendo $X$ a quantidade de canais abertos, podemos definir a variação de $X$ no tempo da seguinte maneira:

$$\frac{dX}{dt} = \alpha(V_m)(1 - X) - \beta(V_m)X \tag{2.15}$$

onde $\alpha(V_m)$ é a taxa de transição do estado fechado $C$ para o estado aberto $O$ e $\beta(V_m)$ é a taxa de transição do estado aberto para fechado.

Como mencionado anteriormente os canais são formados por subunidades. Considerando que um canal tem $n$ subunidades independentes e lembrando que um canal só está aberto quando todas subunidades estão abertas, e considerando todas as subunidades iguais, podemos dizer que a probabilidade deste canal estar aberto é dada por:

$$O = X^n \tag{2.16}$$

onde $X$ é dado pela equação 2.15.

Para o caso das subunidades não serem consideradas iguais e terem diferentes taxas de transição para os estado aberto $O$ e fechado $C$, podemos deduzir a equação de probabilidade de abertura da seguinte forma: dado um canal com $m$ subunidades do tipo $X$ com taxas $\alpha_X$ e $\beta_X$ e $n$ subunidades $Y$ com taxas $\alpha_Y$ e $\beta_Y$, a equação resultante

é apresentada pela equação 2.17:

$$O = X^m Y^n \tag{2.17}$$

Esta equação pode ser estendida de forma análoga para quaisquer números de subunidades diferentes.

Utilizando a equação 2.17 para abertura dos canais e um modelo de corrente linear, podemos definir a equação para a corrente iônica como:

$$I = G_{max} O (V - V_{eq}) \tag{2.18}$$

onde $G_{max}$ é a condutância máxima que é alcançada quando todos os canais estão no estado aberto.

## 2.8 Modelo Celular de Hodgkin e Huxley

Com o objetivo de reproduzir o potencial de ação de axônios gigantes de lula, o modelo de Hodgkin-Huxley foi proposto na década de 50 por Hodgkin e Huxley [46].

Embora não tenha sido desenvolvido para células cardíacas, este é um importante modelo pois foi utilizado como base para o desenvolvimento de diversos outros modelos [40].

Como podemos observar na figura 2.8 este modelo é composto por três correntes, uma corrente de sódio $I_{Na}$, uma corrente de potássio $I_K$, e uma corrente de fuga $I_L$ que representa as outras correntes agrupadas, de forma que:

$$I_{ion} = I_{Na} + I_K + I_L. \tag{2.19}$$

As equações para cada uma das correntes são:

$$I_{Na} = g_{Na} m^3 h (V - V_{Na}) \tag{2.20}$$

$$I_K = g_K n^4 (V - V_K) \tag{2.21}$$

$$I_L = g_L (V - V_L) \tag{2.22}$$

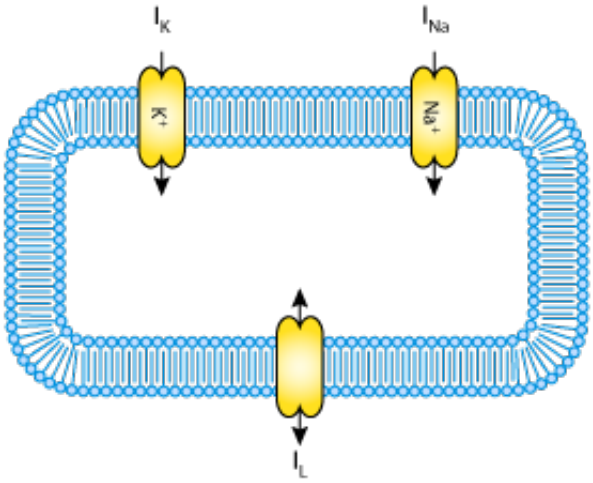


Figura 2.8: Circuito do modelo de Hodgkin e Huxley. Figura adaptada de [47].

onde $g_{Na}$, $g_K$, e $g_L$ são as condutâncias máximas para cada um dos canais, $V_{Na}$, $V_K$, e $V_L$ são os potenciais de equilíbrio de Nernst para cada um dos canais, $m$, $h$ e $n$ são as probabilidades de cada subunidade dos canais iônicos estar aberta. O canal de sódio é composto por três subunidades $m$ e uma $h$, o canal de potássio é composto por quatro subunidades $n$, e a corrente de fuga não possui subunidades. As subunidades estão de acordo com a equação 2.15.

Substituindo as correntes na equação 2.11 obtemos a seguinte equação:

$$-C_m \frac{dV}{dt} = g_{Na} m^3 h (V - V_{Na}) + g_K n^4 (V - V_K) + g_L (V - V_L) + I_{ext} \tag{2.23}$$

onde $I_{ext}$ é uma corrente externa aplicada.

## 2.9 Modelo Celular de Bondarenko

O modelo de Bondarenko e colaboradores (BDK) foi o primeiro modelo apresentado para os miócitos ventriculares de rato [6]. Ao todo, o modelo possui 41 Equações Diferenciais Ordinárias (EDOs) que representam as correntes iônicas.

O termo que representa a corrente iônica $I_{ion}$ neste modelo consiste na soma de 15

correntes transmembrânicas (figura 2.9), e é dado pela equação abaixo:

$$I_{ion} = I_{CaL} + I_{p(Ca)} + I_{NaCa} + I_{Cab} + I_{Na} + I_{Nab} + I_{NaK}$$
$$+I_{Kto,f} + I_{Kto,s} + I_{K1} + I_{Ks} + I_{Kur} + I_{Kss} + I_{Kr} + I_{Cl,Ca}, \quad (2.24)$$

onde:

- $I_{CaL}$ é a corrente de cálcio $Ca^{2+}$ do tipo L,
- $I_{p(Ca)}$ é a corrente máxima da bomba de cálcio($Ca2$),
- $I_{NaCa}$ é o trocador sódio $Na^+$ e cálcio $Ca^{2+}$,
- $I_{Na}$ é a corrente rápida de sódio $Na^+$,
- $I_{Nab}$ e $I_{Cab}$ são as correntes de fundo de sódio $Na^+$ e cálcio $Ca^{2+}$,
- $I_{NaK}$ é a bomba $Na^+/K^+$,
- $I_{Kto,s}$ e $I_{Kto,f}$ são a inativação lenta e rápida da corrente $K+$ transiente de saída,
- $I_{K1}$ é a corrente $K^+$ retificadora de entrada,
- $I_{Kr}$ e $I_{Ks}$ são as correntes rápidas e lentas de $K^+$,
- $I_{Kur}$ é a corrente $K^+$ de retificação ultra rápida,
- $I_{Kss}$ é a corrente $K+$ do estado estacionário e
- $I_{Cl,Ca}$ é a corrente cloro-cálcio.

Os canais iônicos são representados por cadeias de Markov. Pode-se chamar de processo de Markov um sistema finito e discreto formado por estados, onde existem probabilidades de transição entre os estados. Uma cadeia de Markov é a execução repetida de um processo de Markov. Em uma cadeia de Markov o próximo estado será escolhido aleatoriamente, dependendo apenas do estado atual. A figura 2.10 mostra um diagrama de estado representando a cadeia de Markov para o canal de sódio. CNa1, CNa2, CNa3, ONa, IFNa, I1Na, I2Na, ICNa2, ICNa3 representam estados onde:

- CNa1, CNa2, CNa3 representam canal fechado,

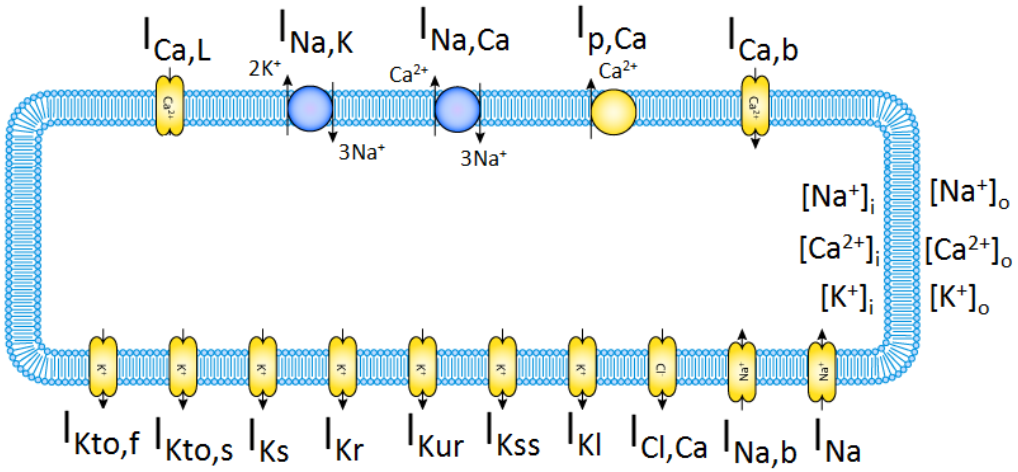


Figura 2.9: Circuito do modelo BDK. Figura adaptada de [47].

- ONa representa canal aberto,
- IFNa representa inativação rápida,
- I1Na, I2Na representam inativação intermediária,
- ICNa2, ICNa3 representam inativação

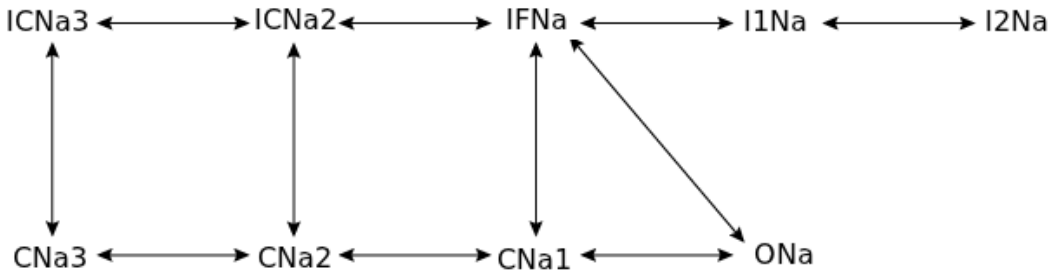


Figura 2.10: Diagrama de estado da cadeia de Markov para o canal de sódio.

## 2.10 Modelos para o Tecido

Esta seção irá descrever como as células estão interconectadas formando o tecido cardíaco pelo qual a onda elétrica se propaga, assim como discutirá as propriedades condutivas da estrutura do tecido. Para isso dois modelos serão apresentados, o Monodomínio e o Bidomínio, que diferem quanto ao número de domínios utilizados para o fluxo de corrente elétrica.

### *2.10.1 Modelo do Bidomínio*

Uma primeira abordagem para modelar o tecido trata cada célula como uma unidade separada e assim controla a variação de potencial tanto no meio intracelular quanto no meio extracelular. Entretanto, o grande número de células do coração torna essa abordagem extremamente custosa em termos de recursos computacionais.

O alto custo computacional faz com que tal abordagem seja ainda pouco explorada, sendo em seu lugar utilizadas abordagens que fazem uma aprocimação contínua do tecido. O modelo do bidomínio utiliza essa abordagem e leva este nome porque representa o tecido cardíaco como dois domínios contínuos, o meio intracelular e o meio extracelular.

O modelo do Bidomínio descreve a estrutura do tecido de uma forma homogenizada [42], na qual as equações resultantes descrevem o fenômeno de forma macroscópica, o que é bastante adequado para a maioria das situações[48].

No domínio intracelular as células estão interconectadas entre si através de *gap junctions*, enquanto no extracelular a corrente elétrica flui nos espaços entre as células.

Em cada um dos domínios define-se um potencial elétrico, e cada ponto pode ser visto como uma quantidade média sobre um pequeno volume. Assim, cada ponto no espaço contém uma fração em cada um desses meios e consequentemente dois potenciais elétricos e duas correntes.

Podemos definir $V_i$ e $i_i$ como o potencial e a corrente intracelular respectivamente e $V_e$ e $i_e$ como o potencial e a corrente extracelular.

A relação entre os potenciais e as correntes é dada pela Lei de Ohm, com isso temos:

$$i_i = -D_i \nabla V_i \tag{2.25}$$

$$i_e = -D_e \nabla V_e \tag{2.26}$$

onde $D_i$ e$D_e$ são os tensores de condutividade de cada domínio. A corrente total em um ponto qualquer $i_t = i_i + i_e$ não pode variar a não ser que fontes externas sejam introduzidas. Esta corrente é conservada, desta forma:

$$\nabla i_t = \nabla \cdot (D_i \nabla V_i + D_e \nabla V_e) = 0 \tag{2.27}$$

Em todos os pontos do domínio podemos definir um potencial transmembrânico $V_m$ e

uma corrente transmembrânica $i_m$:

$$V_m = V_i - V_e \tag{2.28}$$

$$i_m = \nabla \cdot (D_i \nabla V_i) = -\nabla \cdot (D_e \nabla V_e) \tag{2.29}$$

De acordo com o modelo da membrana, descrito previamente, a corrente transmembrânica tem uma parcela iônica e uma parcela capacitiva, desta forma podemos escrever:

$$i_m = \nabla \cdot (D_i \nabla V_i) = \beta \left( C_m \frac{\partial V_m}{\partial t} + I_{ion} \right) \tag{2.30}$$

onde $\beta$ é a relação superfície/volume da membrana, que é necessária para converter corrente transmembrânica por área em corrente transmembrânica por volume.

As Equações 2.27 e 2.30 são conhecidas como equações do bidomínio. Estas equações foram formuladas na década de 1970 por Tung e Geselowitz [49] e podem ser reescritas com uma substituição de variáveis, eliminando o potencial intracelular $V_i$ e incluindo o potencial transmembrânico $V_m$:

$$\nabla \cdot (D_i \nabla V_m) + \nabla \cdot (D_i \nabla V_e) = \beta \left( C_m \frac{\partial V_m}{\partial t} + I_{ion} \right) \tag{2.31}$$

$$\nabla \cdot (D_i \nabla V_m) + \nabla \cdot ((D_i + D_e) \nabla V_e) = 0 \tag{2.32}$$

Estas equações são a formulação mais difundida para a modelagem da propagação elétrica em tecidos cardíacos [50].

### *2.10.2 Modelo do Monodomínio*

O modelo Bidomínio é um sistema de equações diferenciais parciais de difícil resolução[48].

Entretanto, é possível simplificar o mesmo e obter um modelo mais simples, o modelo do monodomínio. O modelo do monodomínio é uma simplificação do bidomínio que considera que o meio extracelular não afeta a atividade elétrica, ou seja, $V_e = 0$. Adotando esta simplificação temos que $V_m = V_i$.

Com isso utilizando a equação 2.31 temos:

$$\nabla \cdot (D_i \nabla V_m) = \beta \left( C_m \frac{\partial V_m}{\partial t} + I_{ion} \right) \tag{2.33}$$

Outra maneira mais geral de se obter a equação do monodomínio é assumir a mesma taxa de anisotropia entre os meios intracelular e extracelular, ou seja, $D_e = \alpha D_i$ onde $\alpha$ é uma constante [40]. Mais detalhes sobre esta forma de obter a equação do monodomínio podem ser encontrados no trabalho de Rocha [48].

Esta equação não é capaz de reproduzir certos fenômenos que as equações do bidomínio conseguem, entretanto, o esforço computacional para sua resolução é bem menor. Por conta disso, esta equação é empregada em diversos trabalhos [48, 51].

# 3 COMPUTAÇÃO PARALELA

Computação paralela é uma forma de computação em que várias intruções são executadas simultaneamente [52], operando sob o princípio de que grande problemas geralmente podem ser divididos em problemas menores, que então são resolvidos concorrentemente (ou em paralelo, caso exista hardware disponível).

A computação paralela tem como principal objetivo aumentar o desempenho e reduzir o tempo para se obter o resultado da computação [53].

Como já mencionado em capítulos anteriores, modelos cardíacos que levam em conta detalhes microscópicos demandam uma discretização a nível sub-celular. Com essa necessidade de discretização a nível sub-celular, juntamente com a utilização de modelos celulares complexos, como o modelo BDK, o sistema computacional resultante torna-se extremamente custoso. Por isso é essencial a utilização de técnicas de computação paralela afim de aumentar o desempenho e reduzir o tempo das simulações.

Neste capítulo serão abordadas de modo introdutório as principais arquiteturas para computação paralela, assim como as principais ferramentas que foram utilizadas neste trabalho para o desenvolvimento das versões paralelas do código, em particular as ferramentas MPI e CUDA.

## 3.1 Arquiteturas de Computação Paralela

### *3.1.1 Arquiteturas de Memória Compartilhada*

SMP (*Symmetric Multi-Processing*) refere-se a arquitetura de computadores multiprocessadores onde dois ou mais processadores idênticos estão conectados a uma única memória principal compartilhada (figura 3.1) e são controlados por um único sistema operacional.

Os sistemas multiprocessadores mais comuns hoje usam uma arquitetura SMP. No caso de processadores multi-núcleos, a arquitetura SMP se aplica aos núcleos, tratando-os como processadores separados.

Como os multiprocessadores compartilham a memória, existe um único espaço de

endereçamento, o que significa que cada posição de memória possui um endereço único, independente do núcleo que o esteja referenciando [54].

O gargalo na escalabilidade dos SMP está no consumo de banda e energia decorrentes da interconexão de vários processadores à memória e aos discos [55].

Sistemas SMP permitem que qualquer processador trabalhe em qualquer tarefa, não importando onde que ela esteja localizada, ou seja, qualquer processador pode acessar qualquer local da memória e qualquer dispositivo de Entrada/Saída.

Com suporte do sistema operacional, sistemas SMP podem mover tarefas entre os processadores facilmente, permitindo-se balancear a carga de trabalho eficientemente.

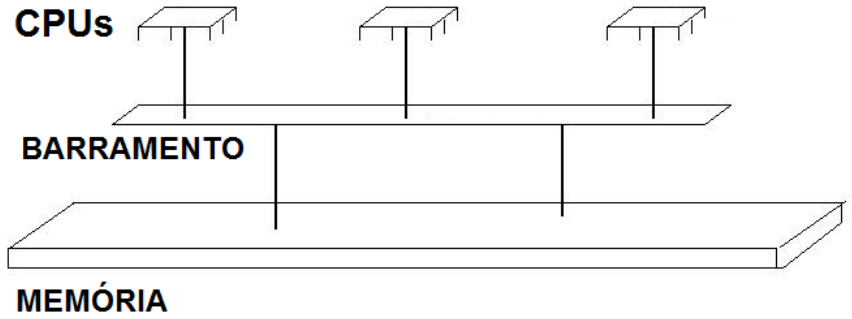


Figura 3.1: SMP - *Symmetric Multi-Processing.*

### *3.1.2 Arquiteturas de Memória Distribuída*

Na arquitetura de memória distribuida, ou multicomputadores, os processadores possuem maior independência entre si, havendo pouco ou nenhum compartilhamento de recursos. Cada nó em um multicomputador é um computador independente, com memória e disco próprios, como podemos observar na figura 3.2. Neste tipo de arquitetura o controle de paralelismo é feito pela aplicação [56].

Como cada computador possui uma memória local única, opera de forma independente e cada mudança realizada na memória local não surte efeito nas memórias dos demais computadores.

Os processadores em computadores diferentes interagem passando mensagens, por isso a rede de comunicação tem um papel importante nesta arquitetura. Quanto maior o tempo necessário para que os dados sejam trocados, maior será o impacto no tempo total de processamento.

Assim, multicomputadores tipicamente empregam redes de comunicação de baixa latência e protocolos especiais de comunicação com o objetivo de reduzir o impacto no tempo de processamento da aplicação.

Um escalonador pode ser usado para controlar a distribuicao de tarefas em um multicomputador. Cabe ao escalonador determinar quais aplicacoes executarão em quais nós. Ou seja, não se pode utilizar um nó que não tenha sido alocado à aplicação pelo escalonador.

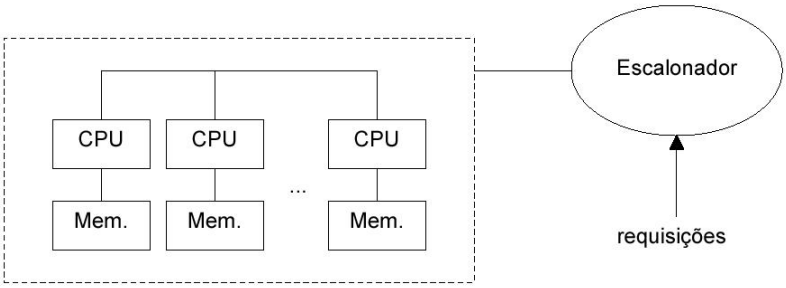


Figura 3.2: Exemplo do papel de um escalonador em um multiprocessador.

### *3.1.3 Memória Compartilhada vs. Memória Distribuída*

Esta seção fará um comparativo entre as vantagens e desvantagens das duas arquiteturas apresentadas.

Como vantagens dos sistemas que utilizam memória compartilhada podemos citar:

- menor tempo de acesso à memória, devido principalmente a proximidade física com que qualquer posição de memória se encontra em relação aos núcleos;
- são mais fáceis de programar que máquinas que se comunicam por troca de mensagens, já que a forma de programação se aproxima daquela feita em sistemas convencionais;
- maior facilidade no compartilhamento de dados entre as distintas partes de uma aplicação, visto que os mesmos já se encontram em memória e esta é compartilhada por todos os núcleos.

E como desvantagens:

- a responsabilidade do programador em sincronizar o acesso aos dados de sua aplicação;
- o menor número de núcleos de processamento que podem ser empregados simultaneamente na resolução problemas;
- o uso de um barramento de interconexão, restringindo o acesso à memória a um núcleo/processador por vez. Esta limitação pode reduzir a escalabilidade desta classe de sistemas;
- o hardware utilizado é mais complexo.

Como vantagens dos sistemas que utilizam memória distribuída podemos citar:

- facilidade para realizar o crescimento incremental do hardware. Novos computadores podem ser agregados ao sistema existente, aumentando seu poder computacional;
- alta disponibilidade. Cada computador escravo é uma unidade independente e, se algum deles falhar, não afeta os demais ou a disponibilidade inteira do sistema.

E como desvantagens:

- a latência de comunicação entre máquinas é superior, por isso há uma dependência do sistema de comunicação, que pode tornar-se um limitador do desempenho das aplicações executadas nesta arquitetura;
- a programação é mais complexa, pois a comunicação de dados entre os computadores fica a cargo do programador.

## 3.2 MPI

MPI (*Message Passing Interface*)[57] é um padrão de sub-rotinas de comunicação, que é utilizado no desenvolvimento de programas para serem executados em mais de um processador, simultaneamente. Existem várias modalidades de computação paralela, e dependendo do problema que se está tentando resolver, pode ser necessário passar informações entre os vários processadores ou nós de um *cluster*, e o MPI oferece uma infraestrutura para essa tarefa. O objetivo de MPI é prover um amplo padrão para

escrever programas com passagem de mensagens de forma prática, portátil, eficiente e flexível. Cada fabricante de computadores é responsável por desenvolver e otimizar uma biblioteca MPI para o seu ambiente paralelo de processamento [58]. Pode ser utilizado em programas FORTRAN, C ou C++.

No padrão MPI, uma aplicação é constituída por um ou mais processos que se comunicam, através de funções para o envio e recebimento de mensagens entre os processos. Inicialmente, na maioria das implementações, um conjunto fixo de processos é criado. Porém, esses processos podem executar diferentes programas. Três passos básicos devem ser seguidos para desenvolver uma aplicação MPI:

- inicialização do ambiente para comunicação;
- comunicação de dados entre processadores; e quando a comunicação for terminada,
- finalização do sistema de troca de mensagens.

Apesar de MPI ser composto por um amplo conjunto de rotinas, é possível escrever vários programas utilizando apenas seis dessas funções, que servem basicamente para: iniciar, terminar, executar e identificar processos, enviar e receber mensagens. Um exemplo de código utilizando estas funções básicas do MPI pode ser encontrado no Apêndice A.

## 3.3 GPGPU e CUDA

A GPU é a unidade de processamento gráfico, um sistema especializado em processar imagens através de uma combinação de um processador e um memória de alta velocidade (GRAM) [59]. GPGPU é um acrônimo de *General-Purpose Computation on Graphics Processing Units* (Computação de Propósito Geral na Unidade de Processamento Gráfico) e também é conhecida como GPU *Computing* (computação GPU). O objetivo da GPGPU é aproveitar o processador da placa de vídeo (GPU) para realizar tarefas de propósito geral que normalmente seriam realizadas pela CPU.

As unidades de processamento gráfico são hoje um dos sistemas multiprocessador mais poderosos em uso. Como podemos observar na figura 3.3 a quantidade de unidades de processamento presente nas GPUs é muito superior a quantidade presente nas CPUs. Por isso elas se tornaram uma opção interessante para o desenvolvimento de aplicações em

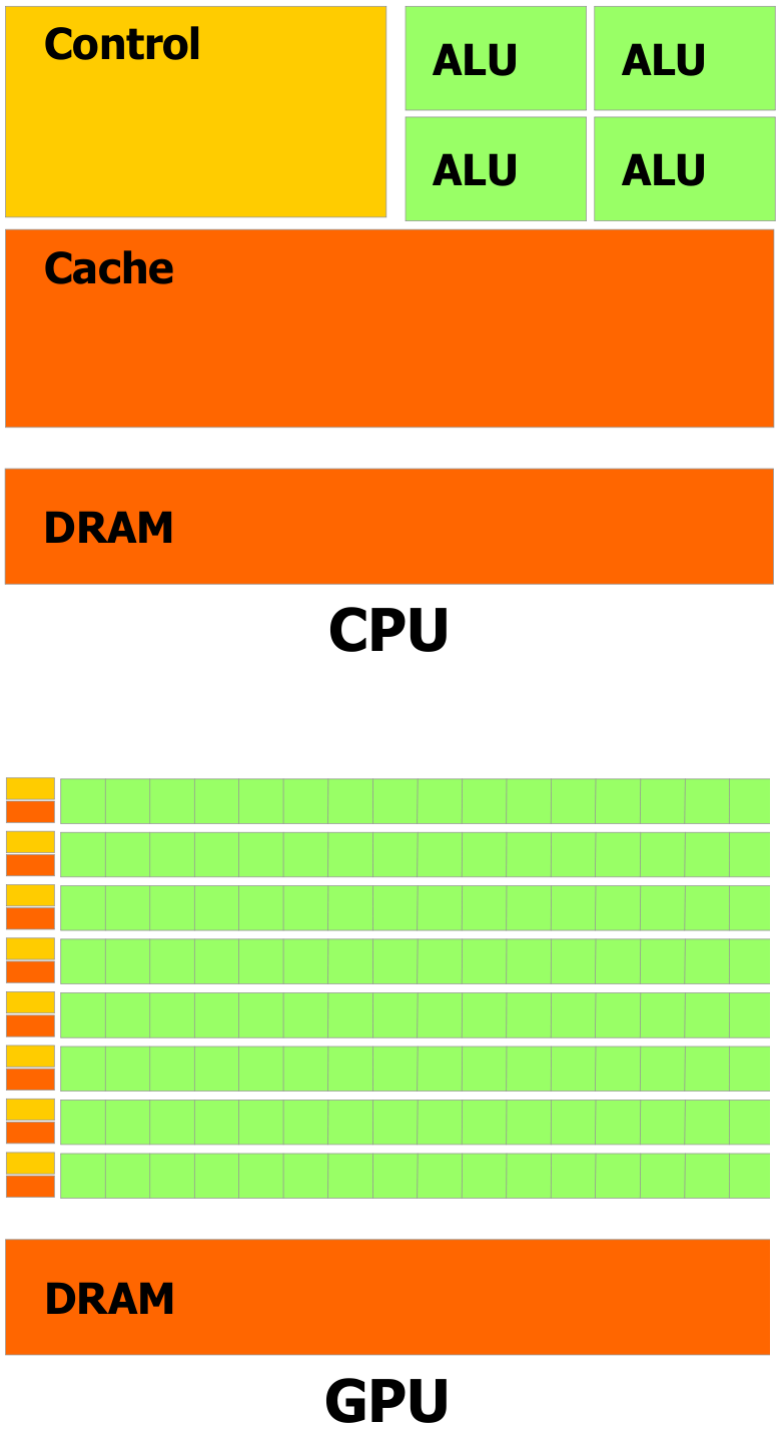


Figura 3.3: Comparação entre arquitetura de CPU e GPU. Figura adaptada de [60].

sistemas paralelos. Além disso as GPUs possuem um modelo de programação paralelo explícito e possuem um desempenho muito maior para alguns tipos de processamentos de dados quando comparados com uma CPU. Portanto, se explica o aumento do uso de GPUs para executar aplicações paralelas de alto desempenho [61].

A GPU é projetada com a responsabilidade principal de lidar com o processamento gráfico do computador. Deste modo, as funcionalidades oferecidas para o desenvolvimento de aplicações eram voltadas unicamente para o processamento e a manipulação de imagens, tornando difícil a sua aplicação para a realização de computação de propósito geral. Devido a esta dificuldade de desenvolvimento de aplicações de propósito geral na GPU foi criada uma "biblioteca" conhecida como CUDA[60] que oferece uma interface para programação em linguagens conhecidas como C e C++.

Para desenvolver um aplicativo, o programador deve criar uma função paralela chamada *kernel*. Um *kernel* é uma função C especial que pode ser chamada pela CPU, mas é executada simultaneamente na GPU por várias *threads*. Cada *thread* é executada por um processador de fluxo da GPU. Eles estão agrupados em blocos de *threads* ou apenas blocos. Os blocos podem ser de uma, duas ou três dimensões. Um conjunto de blocos de segmentos formam um *grid*, que pode ser de uma ou de duas dimensões. Quando a CPU chama o *kernel*, ela deve especificar quantos blocos e segmentos serão criados na GPU para executar o *kernel*. A sintaxe que especifica o número de *threads* que serão criadas para executar um *kernel* é formalmente conhecido como a configuração de execução e é flexível para suportar a hierarquia de *threads* do CUDA, blocos de *threads* e *grids* de blocos. Uma vez que todas as *threads* em um *grid* executam o mesmo código, um conjunto de números de identificação único é usado para distinguir as *threads* e para definir a porção apropriada dos dados que cada uma deve processar. Estas *threads* possuem duas coordenadas únicas, chamadas *blockId* e *threadId*. Os valores dessas coordenadas são atribuídos às *threads* pelo sistema de execução do CUDA. Estas duas variáveis internas podem ser acessadas dentro das funções do *kernel* e retornam os valores apropriados que identificam um bloco e uma *thread*, respectivamente. Todas as *threads* dentro de um único bloco podem sincronizar umas com as outras por meio de um operador de barreira especial, chamado *__syncthread*. Além do mais, as *threads* têm acesso a uma memória compartilhada por bloco de alta velocidade, que permite a comunicação *inter-thread*. *Threads* de blocos diferentes no mesmo *grid* somente podem coordenar suas execuções com o uso de operações atômicas

de memória global. Nenhuma suposição é feita sobre a ordem de execução dos blocos de *threads*, o que significa que um *kernel* deve executar corretamente, não importa a ordem em que os blocos são escalonados pelo hardware para serem executados.

Um código CUDA geralmente segue alguns passos:

1. Inicialização do dispositivo;
2. Alocação de memória da GPU. Isso pode ser feito por exemplo com a função:

   ```
   cudaMalloc((void **) &a_d, size);
   ```

   onde **&a_d** é o ponteiro para a memória alocada no dispositivo e **size** é o tamanho em *bytes* a ser alocado;
3. Transferências dos dados para o dispositivo GPU. A função:

   ```
   cudaMemcpy(a_d, a_h, size, cudaMemcpyHostToDevice)
   ```

   realiza este procedimento, copiando o valor total de **size** *bytes* de **a_h** para **a_d**;
4. Chamada ao *kernel* a ser executado na GPU simultaneamente por várias *threads*. Um exemplo de chamada de um kernel é:

   ```
   square_array <<< n_blocks, block_size >>> ( a_d, N );
   ```

   é um exemplo de chamada de um *kernel*. Detalhes da implementação desta função podem ser vistos na figura 3.4;
5. Transferências dos dados do dispositivo GPU de volta para a CPU. A função:

   ```
   cudaMemcpy(a_h, a_d, size, cudaMemcpyDeviceToHost)
   ```

   realiza este procedimento, copiando o valor total de **size** *bytes* de **a_d** para **a_h**;

A figura 3.4 mostra um exemplo de código CUDA.

A

```
// Kernel that executes on the CUDA device
__global__ void square_array( float *a, int N )
{
    int idx = blockIdx.x * blockDim.x + threadIdx.x;
    if ( idx < N )
        a[idx] = a[idx] * a[idx];
}
```

B

```
// main routine that executes on the host
int main( void )
{
    float *a_h, *a_d; // Pointer to host & device arrays
    const int N = 10; // Number of elements in arrays
    size_t size = N * sizeof( float );
    a_h = (float *)malloc( size );    // Allocate array on host
    cudaMalloc( (void **)&a_d, size ); // Allocate array on device
    // Initialize host array and copy it to CUDA device
    for ( int i = 0; i < N; i++ )
        a_h[i] = (float)i;
    cudaMemcpy( a_d, a_h, size, cudaMemcpyHostToDevice );
    // Do calculation on device:
    int block_size = 4;
    int n_blocks   = N / block_size + ( N % block_size == 0 ? 0 : 1 );
    square_array <<< n_blocks, block_size >>> ( a_d, N );
    // Retrieve result from device and store it in host array
    cudaMemcpy( a_h, a_d, sizeof( float ) * N, cudaMemcpyDeviceToHost );
    // Print results
    for ( int i = 0; i < N; i++ )
        printf( "%d %f\n", i, a_h[i] ); // Cleanup
    free( a_h );
    cudaFree( a_d );
}
```

Figura 3.4: Exemplo de código C com CUDA, para elevar ao quadrado todos os valores de um vetor. A)Código C para um núcleo CUDA. B) Código C para o *host*.

# 4 MÉTODOS

## 4.1 O Modelo Microscópico

### *4.1.1 Modelagem da microestrutura cardíaca*

Um modelo bidimensional microscópico foi desenvolvido baseado no trabalho anterior de Spach e colaboradores [11, 17] e leva em conta a microestrutura do tecido cardíaco, a distribuição heterogênea das *gap junctions* e uma discretização de 8 $\mu$m $\times$ 8 $\mu$m [62]. Um modelo básico para conexões de miócitos foi desenvolvido e é apresentado na figura 4.1. Esta unidade básica é formada pela conexão de 32 miócitos cardíacos com diferentes formas e número de células vizinhas. Os valores médios e DP (desvio padrão) para o comprimento e largura da célula são $120.9 \pm 27.8 \mu m$ e $18.3 \pm 3.5 \mu m$, respectivamente [62]. Estes valores são similares aos encontrados na literatura: comprimento = $140 \mu m$ e largura = $19 \mu m$ [63]; comprimento = $134 \mu m$ e largura = $18 \mu m$ [64]; comprimento = $100 \mu m$ e largura = $17.32 \mu m$ [20]. Em média, cada miócito se conecta a outros 6 miócitos vizinhos. O nosso modelo bidimensional considera uma profundidade homogênea $d = 8 \mu$m [11, 17].

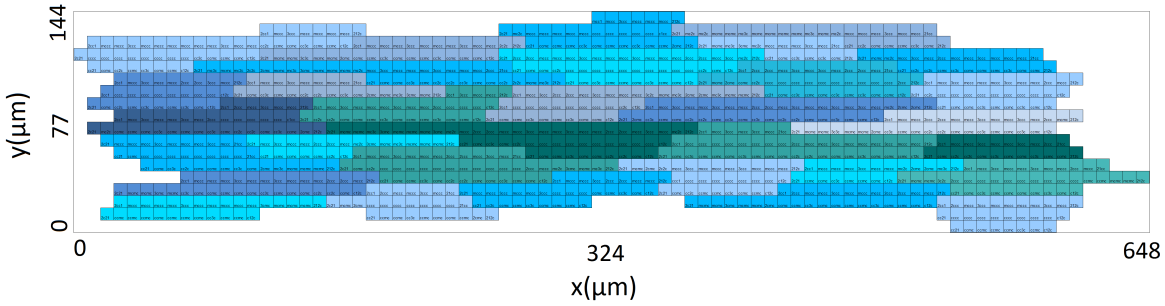


Figura 4.1: Unidade básica representando uma distribuição de miócitos cardíacos com um total de 32 células. As células são mostrados em diferentes cores alternadas ao longo do eixo $x$. A unidade de base se estende por um total de 648 $\mu m$ na direção longitudinal, contra 144 $\mu m$ na direção transversal.

Esta unidade básica foi criada de tal modo que permite a geração de tecidos maiores, através da conexão de múltiplas instâncias desta unidade. A figura 4.2 mostra como as unidades básicas são conectadas.

A figura 4.3 apresenta um exemplo de como as conexões entre os diferentes miócitos podem ser organizadas. O código foi desenvolvido de forma flexível, de modo que

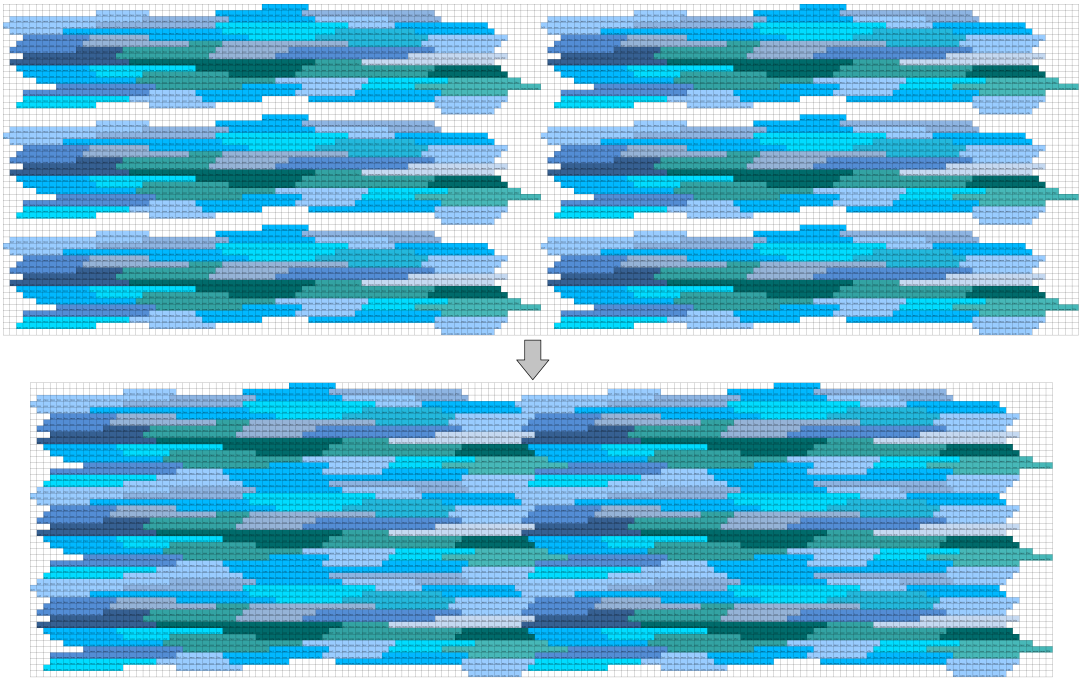

Figura 4.2: Seis unidades básicas sendo combinadas para formar um tecido maior.

permite ao usuário configurar os valores de condutividade ou condutância de cada volume discretizado $Vol_{i,j}$ (com área = h × h) para as faces norte ($\sigma_{x_{i,j+1/2}}$), sul ($\sigma_{x_{i,j-1/2}}$), oeste ($\sigma_{x_{i-1/2,j}}$) e leste ($\sigma_{x_{i+1/2,j}}$) do volume. Estes podem ser quaisquer valores não negativos. Neste trabalho, vamos definir a discretização $h$ como $8\mu$m. Além disso, com base no trabalho de Spach e colaboradores [11, 17], escolhemos apenas cinco tipos possíveis de conexões entre volumes vizinhos que são: membrana ($\sigma_m = 0.0$), citoplasma ($\sigma_c = 0.4\mu S/\mu m$), *plicate gap junction* ($G_p = 0.5\mu S$), *interplicate gap junction* ($G_i = 0.33\mu S$), e *combined plicate gap junction* ($G_c = 0.062\mu S$). Usamos $\sigma$ para condutividade e $G$ para condutância. Para as simulações apresentadas neste trabalho, a distribuição das diferentes *gap junctions* dentro dos 32 miócitos não foi gerada aleatoriamente. Em vez disso, a distribuição das *gap junctions* da unidade básica foi escolhida manualmente para reproduzir a distribuição apresentada antes em [11, 17]. Com essa configuração e valores de condutividade encontramos uma velocidade de propagação ao longo das fibras de cerca de $410\mu m/ms$ ($LP$), e de $130\mu m/ms$ no sentido transversal à direção das fibras ($TP$). Isso resulta em uma proporção $LP/TP$ de 0.32, que está próximo da proporção de propagação encontrada em [11].

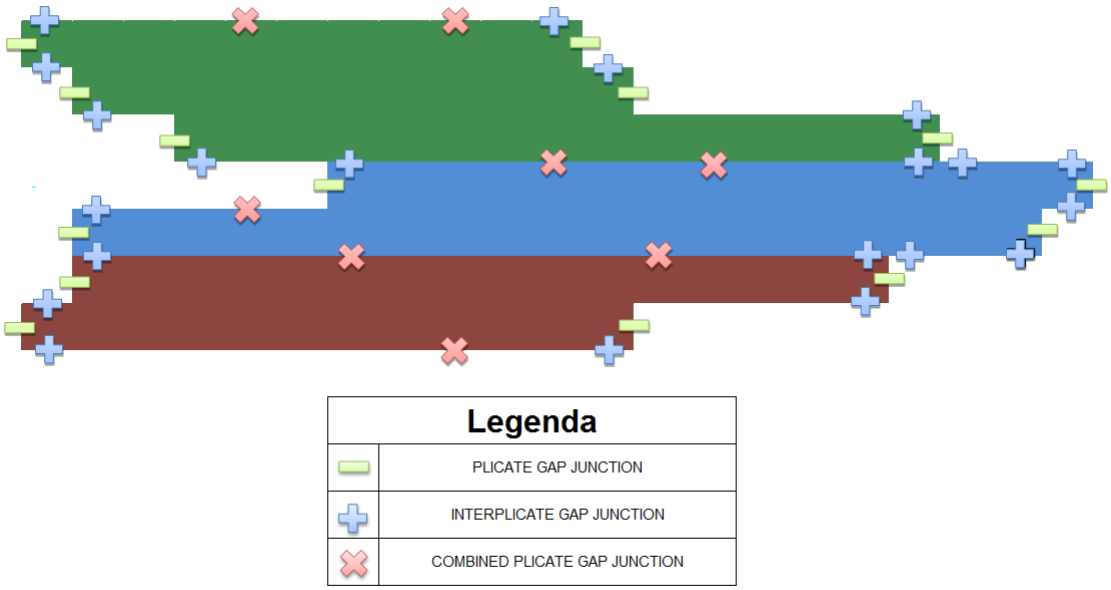


Figura 4.3: Neste trabalho, existem apenas cinco tipos possíveis de conexões entre volumes vizinhos que são: membrana, o que indica ausência de fluxo entre os volumes vizinhos; citoplasma, o que indica que os volumes vizinhos estão dentro da mesma célula; e três possíveis tipos de *gap junctions*, *plicate*, *interplicate*, e *combined plicate*. Para as simulações apresentadas neste trabalho, a distribuição das *gap junctions* da unidade básica foi escolhida manualmente para reproduzir a distribuição apresentada antes em [11, 17]. Esta figura apresenta um exemplo de como as diferentes *gap junctions* são distribuídas em três miócitos vizinhos que pertencem à unidade básica.

### *4.1.2 O modelo heterogêneo do monodomínio*

Os potenciais de ação propagam-se através do tecido cardíaco, porque o espaço intracelular das células cardíacas estão eletricamente acoplados por *gap junctions*. Neste trabalho, nós não consideramos os efeitos da matriz extracelular. Portanto, o fenômeno pode ser descrito matematicamente por uma equação diferencial parcial (EDP) do tipo reação-difusão chamada modelo do monodomínio que foi apresentada no capítulo 2, seção 2.10.2, sendo representada pelas equações:

$$\beta C_m \frac{\partial V(x,y,t)}{\partial t} + \beta I_{ion}(V(x,y,t), \boldsymbol{\eta}(x,y,t)) = \nabla \cdot (\boldsymbol{\sigma}(x,y) \nabla V(x,y,t)) + I_{stim}(x,y,t) \tag{4.1}$$

$$\frac{\partial \boldsymbol{\eta}(x,y,t)}{\partial t} = \boldsymbol{f}(V(x,y,t), \boldsymbol{\eta}(x,y,t)) \tag{4.2}$$

onde $V$ é a variável de interesse e representa o potencial transmembrânico, ou seja a diferença entre o potencial intracelular para o extracelular; $\boldsymbol{\eta}$ é um vetor de variáveis

de estado que também influenciam a geração e propagação da onda elétrica, e inclui, geralmente, a concentração intracelular de diferentes íons ($K^+$, $Na^+$, $Ca^{2+}$) e a permeabilidade dos diferentes canais iônicos da membrana; $\beta$ é a razão superfície-volume das células do coração; $C_m$ é a capacitância da membrana, $I_{ion}$ é a corrente total iônica, que é uma função de $V$ e um vetor de variáveis de estado $\boldsymbol{\eta}$, $I_{stim}$ é a corrente devida a um estímulo externo e $\boldsymbol{\sigma}$ é o tensor de condutividade do monodomínio. Assumimos que o limite do tecido é isolado, ou seja, sem fluxo no contorno ($\mathbf{n} \cdot \sigma \nabla V = 0$ em $\partial\Omega$).

Neste trabalho, o moderno e complexo modelo BDK apresentado no capítulo 2, seção 2.9, que descreve a atividade elétrica das células do ventrículo esquerdo de ratos, foi considerado para simular a cinética de $I_{ion}$ nas equações 4.1 e 4.2.

### *4.1.3 Discretização numérica no espaço e no tempo*

O método dos volumes finitos (MVF) é um método matemático utilizado para obter uma versão discreta das equações diferenciais parciais (EDPs). Este método é apropriado para simulações numéricas de vários tipos de leis de conservação (elíptica, parabólica ou hiperbólica) [65]. Tal como o método dos elementos finitos (MEF), o MVF pode ser usado em vários tipos de geometria, usando malhas estruturadas ou não estruturadas e gera esquemas numéricos robustos. O desenvolvimento do método está intrinsecamente ligado ao conceito de fluxo entre as regiões ou volumes adjacentes, isto é, que se baseia no cálculo numérico de fluxos líquidos para dentro ou para fora de um volume de controle. Para alguns problemas isotrópicos discretizados com malhas espaciais regulares, a discretização obtida com o MVF é muito semelhante à obtida com o método padrão de diferenças finitas (MDF).

Esta seção apresenta uma breve descrição da aplicação do MVF no tempo e a discretização espacial das equações heterogêneas do monodomínio. Uma informação detalhada sobre o MVF aplicado à solução do monodomínio pode ser encontrada em [66, 67].

#### 4.1.3.1 Discretização temporal

As partes de reação e difusão das equações do monodomínio foram divididas pela aplicação do operador de divisão de Godunov [68]. Assim, cada passo de tempo envolve a solução

de dois problemas distintos: um sistema não linear de EDOs

$$\frac{\partial V}{\partial t} = \frac{1}{C_m}\left[-I_{ion}(V, \boldsymbol{\eta}) + I_{stim}\right] \tag{4.3}$$

$$\frac{\partial \boldsymbol{\eta}}{\partial t} = f(V, \boldsymbol{\eta}) \tag{4.4}$$

e uma EDP parabólica

$$\beta\left(C_m \frac{\partial V}{\partial t}\right) = \nabla \cdot (\boldsymbol{\sigma} \nabla V) \tag{4.5}$$

Uma vez que a discretização espacial do nosso modelo, $h$, é extremamente pequena, a condição de Courant-Friedrichs-Lewy (condição de CFL) [69] que garante a estabilidade numérica é muito restritiva. Portanto, para a EDP foi utilizado o esquema de Euler implícito incondicionalmente estável. A derivada no tempo é mostrada na equação 4.5, que opera em $V$ e é aproximada por um esquema de Euller implícito de primeira ordem:

$$\frac{\partial V}{\partial t} = \frac{V^{n+1} - V^n}{\Delta t_p}, \tag{4.6}$$

onde $V^n$ representa o potencial transmembrânico no tempo $t_n$ e $\Delta t_p$ é o passo de tempo usado para avançar a equação diferencial parcial no tempo.

Para a discretização do sistema não-linear de EDOs notamos que a sua natureza *stiff* exige passos de tempo muito pequenos. Para os modelos simples baseados na formulação de Hodgkin-Huxley, este problema é normalmente superado utilizando o método de Rush-Larsen (RL) [70]. No entanto, para os modelos mais modernos e complexos que são altamente baseados em Cadeias de Markov, o método RL parece ser ineficaz por não permitir passos de tempo maiores durante a integração numérica. Para o caso do modelo BDK, testamos ambos os métodos, de Euler e RL, e ambos exigiram o mesmo passo tempo, $\Delta t_o = 0.0001$ ms por questões de estabilidade. Como o método de RL é mais caro por passo de tempo do que o método de Euler, neste trabalho foi utilizado o método simples de Euler explícito para a discretização das EDOs não-lineares.

No entanto, como já dito anteriormente, usamos intervalos de tempo diferentes para a solução dos dois diferentes problemas desacoplados, a EDP e as EDOs. Uma vez que utilizar um método incondicionalmente estável para a EDP, o passo de tempo $\Delta t_p$ poderia

ser muito maior do que o utilizado para a solução do sistema linear de EDOs, $\Delta t_o = 0.0001$ ms. Neste trabalho, utilizamos $\Delta t_p = 0.01$ ms, ou seja, cem vezes maior do que $\Delta t_o$. Isso não introduziu qualquer erro numérico significativo. Calculamos o erro relativo L2 do potencial transmembrânico entre uma solução que usa o mesmo passo de tempo para a EDO e para a EDP, $\Delta t_o = \Delta t_p = 0.0001$ ms, $V_{m_{ref}}$ e uma solução que usa $\Delta t_o = 0.0001$ ms e $\Delta t_p = 0.01$ ms, $V$:

$$erro = \frac{\sqrt{\sum_{i=1}^{nt}\sum_{j=1}^{nv}\left(V\left(i,j\right) - V_{m_{ref}}\left(i,j\right)\right)^2}}{\sqrt{\sum_{i=1}^{nt}\sum_{j=1}^{nv} V_{m_{ref}}\left(i,j\right)^2}}, \tag{4.7}$$

onde $nt$ é o número de passos de tempo e $nv$ é o número total de volumes discretizados. Para a simulação de um tecido de tamanho $0.5 \times 0.5$ cm durante 20 ms (estímulo no centro do tecido) o erro encontrado foi de 0.01%.

#### 4.1.3.2 Discretização espacial

O termo difusão na equação 4.5 deve ser discretizado no espaço. Para isso, vamos considerar o seguinte:

$$\boldsymbol{J} = -\boldsymbol{\sigma}\nabla V \tag{4.8}$$

onde $J$ $(\mu A/cm^2)$ expressa a densidade de fluxo de corrente intracelular e

$$\nabla \cdot \boldsymbol{J} = -I_v. \tag{4.9}$$

Nesta equação, $I_v(\mu A/cm^3)$ é uma corrente volumétrica e corresponde ao lado esquerdo da equação 4.5, servindo como base para a solução dos volumes finitos.

Para a discretização do espaço, vamos considerar uma malha uniforme bidimensional, constituída por quadriláteros regulares (chamados "volumes"). Localizado no centro de cada volume temos um nó. A quantidade de interesse $V$ é associada a cada nó da malha.

Depois de definida a geometria da malha e dividindo o domínio em volumes de controle, as equações específicas do MVF podem ser apresentadas. A equação 4.9 pode ser integrada espacialmente ao longo de um volume individual $V_{i,j}$ de tamanho $h^2\, d$, levando a:

$$\int_\Omega \nabla \cdot \boldsymbol{J} dv = -\int_\Omega I_v \, dv. \tag{4.10}$$

Aplicando o teorema da divergência temos:

$$\int_{\Omega} \nabla \cdot \boldsymbol{J} dv = \int_{\partial\Omega} \boldsymbol{J} \cdot \vec{\xi} ds, \tag{4.11}$$

onde $\vec{\xi}$ é o vetor unitário normal ao contorno $\partial\Omega$. Em seguida, temos:

$$\int_{\partial\Omega} J \cdot \vec{\xi} ds = -\int_{\Omega} I_v dv. \tag{4.12}$$

Finalmente, assumindo que $I_v$ representa um valor médio em cada um quadrilátero específico, e substituindo a equação 4.5 na equação 4.12, obtemos:

$$\beta \left( C_m \frac{\partial V}{\partial t} \right) \Bigg|_{(i,j)} = \frac{-\int_{\partial\Omega} J_{i,j} \cdot \vec{\xi} ds}{h^2\, d}. \tag{4.13}$$

Para este problema bidimensional particular, constituído por um *grid* uniforme de quadriláteros com lados $h$, o cálculo de $J_{i,j}$ pode ser subdividido como uma soma de fluxos nas faces:

$$\int_{\partial\Omega} J_{i,j} \cdot \vec{\xi} ds = (I_{x_{i+1/2,j}} - I_{x_{i-1/2,j}} + I_{y_{i,j+1/2}} - I_{y_{i,j-1/2}}) \tag{4.14}$$

onde $I_{x_{m,n}}$ e $I_{y_{m,n}}$ são calculados nas faces ($(m,n) = (i+1/2, j)$, $(i-1/2, j)$, $(i, j+1/2)$, ou $(i, j-1/2)$) como segue. Para este caso, nós definimos um valor de condutividade na face $(m,n)$, por exemplo, a condutividade intracelular ou citoplasmática, $\sigma_c$, como apresentado na seção 4.1.1. Com isso temos:

$$\begin{aligned} I_{x_{m,n}} &= -\sigma_c(m,n) \frac{\partial V}{\partial x} \Bigg|_{(m,n)} hd \\ I_{y_{m,n}} &= -\sigma_c(m,n) \frac{\partial V}{\partial y} \Bigg|_{(m,n)} hd. \end{aligned} \tag{4.15}$$

Em outra situação, nós definimos um valor de condutância para a face $(m,n)$, por exemplo, uma condutância de *gap junction* $G$, como apresentado na seção 4.1.1, assim

temos:

$$\begin{aligned} I_{x_{m,n}} &= -G(m,n)\Delta_x V\Big|_{(m,n)} \\ I_{y_{m,n}} &= -G(m,n)\Delta_y V\Big|_{(m,n)} \end{aligned} . \tag{4.16}$$

Usando diferença finita centrada temos para equação 4.15:

$$\begin{aligned} \frac{\partial V}{\partial x}\Big|_{(i+1/2,j)} &= \frac{V_{i+1,j} - V_{i,j}}{h} \\ \frac{\partial V}{\partial y}\Big|_{(i,j+1/2)} &= \frac{V_{i,j+1} - V_{i,j}}{h} \end{aligned} \tag{4.17}$$

Para a equação 4.16, temos:

$$\begin{aligned} \Delta_x V\Big|_{(i+1/2,j)} &= V_{i+1,j} - V_{i,j} \\ \Delta_y V\Big|_{(i,j+1/2)} &= V_{i,j+1} - V_{i,j} \end{aligned} \tag{4.18}$$

Equações para $\frac{\partial V}{\partial x}\Big|_{(i-1/2,j)}$, $\frac{\partial V}{\partial y}\Big|_{(i,j-1/2)}$, $\Delta_x V\Big|_{(i-1/2,j)}$ e $\Delta_y V\Big|_{(i,j+1/2)}$ podem ser obtidas analogamente.

Rearranjando e substituindo as discretizações das equações 4.6 e 4.14 em 4.13 e decompondo os operadores, tal como descrito pelas equações 4.3, 4.4 e 4.5 chegamos em:

$$\begin{aligned} (\sigma_{i+1/2,j} + \sigma_{i-1/2,j} + \sigma_{i,j+1/2} + \sigma_{i,j-1/2} + \alpha)V^*_{i,j} - \sigma_{i,j-1/2}V^*_{i,j-1} - \sigma_{i+1/2,j}V^*_{i+1,j} \\ - \sigma_{i,j+1/2}V^*_{i,j+1} - \sigma_{i-1/2,j}V^*_{i-1,j} \\ = \alpha V^n_{i,j} \end{aligned} \tag{4.19}$$

$$\begin{aligned} C_m \frac{V^{n+1}_{i,j} - V^*_{i,j}}{\Delta t_o} &= -I_{ion}(V^*_{i,j}, \boldsymbol{\eta}^n) \\ \frac{\boldsymbol{\eta}^{n+1} - \boldsymbol{\eta}^n}{\Delta t_o} &= f(\boldsymbol{\eta}^n, V^*, t) \end{aligned} \tag{4.20}$$

onde $\alpha = (\beta C_m h^2)/\Delta t_p$, $n$ é o passo atual, $*$ é um passo intermediário e $n+1$ é o próximo

passo de tempo. Além disso $\sigma$ pode representar qualquer uma das condutâncias das *gap junctions* ($G_p$, $G_i$, $G_c$) dividida pela profundidade $d$ ou qualquer valor de condutividade ($\sigma_c$, $\sigma_m$) definido para cada face do volume, tal como descrito na seção 4.1.1. Isto define as equações para cada volume finito $Vol_{i,j}$. Primeiramente, resolvemos o sistema linear associado à equação 4.19 para avançar $\Delta t_p$ no tempo e, então posteriormente, resolvemos o sistema não-linear de EDOs associado às equações 4.20 $N_o$ vezes até termos $N_o\,\Delta t_o = \Delta t_p$.

## 4.2 O Modelo Discreto

Estudos recentes sobre a natureza discreta ou descontínua da propagação do potencial de ação têm evitado os desafios computacionais que surgem dos modelos microscópicos através do desenvolvimento e utilização de modelos discretos, em que cada um dos miócitos cardíacos é representado por um único ponto (célula) em contato com os miócitos vizinhos através de condutividades discretas [71, 72]. Esta descrição tem permitido o estudo dos efeitos que distribuições aleatórias de condutividades tem na velocidade de condução e na formação de padrões de reentrada no tecido cardíaco. Os modelos discretos foram introduzidos por Keener [73] para descrever a propagação elétrica em um cabo $1D$ de $nc$ células conectadas para o caso do baixo acoplamento das *gap junctions*. Neste modelo, as células são assumidas como isopotenciais. Portanto, apenas as condutâncias das *gap junctions* são consideradas para a ligação de miócitos vizinhos, isto é, a resistência do citoplasma é considerada insignificante. Recentemente, em um trabalho de Costa [74] modelos discretos e microscópicos de um cabo $1D$ de células conectadas foram comparados.

Neste trabalho, foi desenvolvido um modelo discreto equivalente ao modelo microscópico apresentado na seção 4.1. Para isso, utilizando a microestrutura cardíaca apresentada na seção 4.1.1 e identificando os vizinhos (ligações) de cada célula, foi calculado a condutância entre cada ligação e uma nova malha foi gerada. As equações que regem a propagação dos potenciais de ação no tecido cardíaco são as mesmas utilizadas no modelo microscópico apresentado na seção 4.1.2 (equações 4.1, 4.2):

$$\beta C_m \frac{\partial V(x,y,t)}{\partial t} + \beta I_{ion}(V(x,y,t), \boldsymbol{\eta}(x,y,t)) = \nabla \cdot (\boldsymbol{\sigma}(x,y)\nabla V(x,y,t)) + I_{stim}(x,y,t)$$

$$\frac{\partial \boldsymbol{\eta}(x,y,t)}{\partial t} = \boldsymbol{f}(V(x,y,t), \boldsymbol{\eta}(x,y,t))$$

### *4.2.1 Discretização numérica no espaço e no tempo*

A discretização temporal é feita da mesma maneira que no modelo microscópico (seção 4.1.3.1). Já a discretização espacial muda. No modelo microscópico tínhamos uma malha uniforme bidimensional, constituída por quadriláteros regulares (chamados "volumes") de lado $l = h$, agora temos células inteiras constituindo cada nó da malha, e cada célula com uma geometria diferente e um diferente número de células vizinhas. Como não há alteração na discretização temporal com relação ao modelo microscópico, precisamos refazer somente a discretização espacial.

#### 4.2.1.1 Discretização espacial

O termo difusão na equação 4.5 deve ser discretizado no espaço. Para isso, vamos considerar o seguinte:

$$\boldsymbol{J} = -\boldsymbol{\sigma}\nabla V \tag{4.21}$$

onde $J$ ($\mu A/cm^2$) expressa a densidade de fluxo de corrente intracelular e

$$\nabla \cdot \boldsymbol{J} = -I_v. \tag{4.22}$$

Nesta equação, $I_v(\mu A/cm^3)$ é uma corrente volumétrica e corresponde ao lado esquerdo da equação 4.5, servindo como base para a solução dos volumes finitos.

Para a discretização do espaço, uma malha bidimensional é construída a partir da malha utilizada no modelo microscópico; cada nó da malha representa uma célula. A quantidade de interesse $V$ é associada a cada nó da malha, ou seja, a cada célula.

Para gerar esta nova malha discreta foi desenvolvida uma aplicação que tem como entrada a malha microscópica montada a partir das unidade básicas que foram apresentadas anteriormente na seção 4.1.1. A partir desta malha, a aplicação identifica cada célula, seu ponto de referência no espaço (foi utilizado o centro da célula) e seus

vizinhos, ou seja suas ligações. Uma célula é considerada vizinha de outra se ela tem ao menos uma *gap junction*. A figura 4.4B mostra a malha discreta resultante para uma unidade básica da malha microscópica.

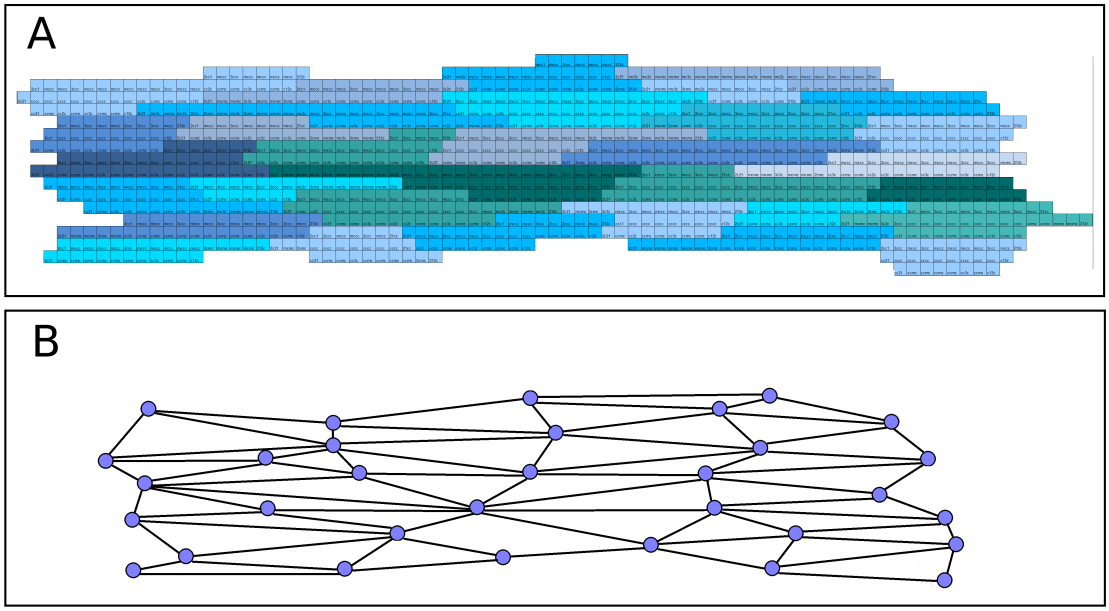


Figura 4.4: Transformação da malha microscópica A) em discreta B).

Cada célula representa um volume de tamanho $A_i d$, onde $A_i$ é a área da célula $i$ e $d$ a profundidade. Depois de definida a geometria da malha e dividindo o domínio em volumes, as equações específicas do MVF podem ser apresentadas. A equação 4.22 pode ser integrada espacialmente ao longo de um volume individual $V_i$ levando a:

$$\int_{\Omega} \nabla \cdot \boldsymbol{J} dv = -\int_{\Omega} I_v \, dv. \tag{4.23}$$

Aplicando o teorema da divergência temos:

$$\int_{\Omega} \nabla \cdot \boldsymbol{J} dv = \int_{\partial\Omega} \boldsymbol{J} \cdot \vec{\xi} ds, \tag{4.24}$$

onde $\vec{\xi}$ é o vetor unitário normal ao contorno $\partial\Omega$. Em seguida, temos:

$$\int_{\partial\Omega} J \cdot \vec{\xi} ds = -\int_{\Omega} I_v dv. \tag{4.25}$$

Finalmente, assumindo que $I_v$ representa um valor médio em cada célula, e

substituindo a equação 4.5 na equação 4.25, obtemos:

$$\beta \left( C_m \frac{\partial V}{\partial t} \right) \Bigg|_i = \frac{- \int_{\partial\Omega} J_i \cdot \vec{\xi} ds}{A_i \, d}. \tag{4.26}$$

O cálculo de $J_i$ pode ser subdividido como um somatório dos fluxos vindo dos vizinhos:

$$\int_{\partial\Omega} J_i \cdot \vec{\xi} ds = \sum_{j=1}^{n} I_{i,j} \tag{4.27}$$

onde $n$ é o número de vizinhos da célula $i$, e $j$ é o índice do vizinho. Definindo um valor de condutância para ligação de cada par $i, j$ de células, temos:

$$I_{i,j} = -G(i,j) \Delta V \Big|_{(i,j)} \tag{4.28}$$

Ainda na equação 4.28, temos que:

$$\Delta V \Big|_{(i,j)} = V_j - V_i \tag{4.29}$$

Rearranjando e substituindo as discretizações das equações 4.6 e 4.27 na equação 4.26 e decompondo os operadores, tal como descrito pelas equações 4.3, 4.4 e 4.5 chegamos em:

$$(\sum_{j=1}^{n}((G_{i,j})) + \alpha) V_i^* - (\sum_{j=1}^{n}(G_{i,j} * V_j)) = \alpha V_i^n \tag{4.30}$$

$$\begin{aligned} C_m \frac{V_i^{n+1} - V_i^*}{\Delta t_o} &= -I_{ion}(V_i^*, \boldsymbol{\eta}^n) \\ \frac{\boldsymbol{\eta}^{n+1} - \boldsymbol{\eta}^n}{\Delta t_o} &= f(\boldsymbol{\eta}^n, V^*, t) \end{aligned} \tag{4.31}$$

onde $\alpha = (\beta C_m A_i d)/\Delta t_p$, $n$ é o passo atual, $*$ é um passo intermediário e $n+1$ é o próximo passo de tempo. Isto define as equações para cada volume finito $Vol_i$. Primeiramente, resolvemos o sistema linear associado à equação 4.30 para avançar $\Delta t_p$ no tempo e, então posteriormente, resolvemos o sistema não-linear de EDOs associado às equações 4.31 $N_o$ vezes até termos $N_o \, \Delta t_o = \Delta t_p$.

### 4.2.2 Cálculo da condutância entre células vizinhas

Para a resolução do modelo discreto definido na seção anterior, precisamos ainda calcular os valores de condutância das ligações, ou seja, calcular $G_{i,j}$ da equação 4.30.

Isso foi feito aplicando o conceito de associação de resistores em paralelo à microstrutura definida na seção 4.1.1 para cada par de células vizinhas.

Da associação de resistores temos:

$$\frac{1}{R_{i,j}} = \sum_{k=1}^{n} (\frac{1}{R_{i,j,k}}) \tag{4.32}$$

onde $R_{i,j}$ é a resistência equivalente e $n$ é o total de *gap junctions* que liga a célula $i$ à célula $j$. O termo $R_{i,j,k}$ foi calculado como uma associação de resistores em série:

$$R_{i,j,k} = dist_{i,j,k} * \frac{1}{(\sigma h)} + \frac{1}{G_k} \tag{4.33}$$

onde $dist_{i,j,k}$ é o total de volumes ligados por citoplasma que a corrente precisa atravessar para ir da célula $i$ até a célula $j$ passando pela *gap junction* $k$, sendo calculado pela equação:

$$dist_{i,j,k} = (|pr_{i,x} - gp_{i,k,x}| + |pr_{i,y} - gp_{i,k,y}|) + (|pr_{j,x} - gp_{j,k,x}| + |pr_{j,y} - gp_{j,k,y}|) \tag{4.34}$$

Onde $pr_i$ é uma coordenada bidimensional representando o volume do ponto de referência da célula $i$ e $gp_{i,k}$ é uma coordenada bidimensional representando o volume que contém a *gap junction* de ligação $k$ na célula $i$. Deve-se ressaltar que estas coordenadas são índices dos volumes na malha que representa as células $i$ e $j$, portanto não tem unidade de comprimento. Dessa forma cada volume ligado por citoplasma ou por *gap junction* representa um resistor.

De forma simples o termo $dist * \frac{1}{(\sigma h)}$ representa a resistência citoplasmática total que a corrente enfrenta para ir da célula $i$ até a célula $j$, ou vice-versa. O termo $\frac{1}{G_k}$ representa a resistência da *gap junction*. Os valores de condutividade do citoplasma *sigma* e condutância $G$ para cada tipo de *gap junction* são os mesmos definidos na seção 4.1.1.

Com isso agora podemos calcular a condutância $G_{i,j}$ que é o inverso da resistência.

$$G_{i,j} = \frac{1}{R_{i,j}} \tag{4.35}$$

Para melhor entendimento deste procedimento, segue um exemplo do cálculo de $G_{i,j}$. A figura 4.5 mostra a microestrutura das duas células vizinhas utilizadas neste exemplo, bem como seus respectivos pontos de referência. Vamos considerar $i = 1$ e $j = 2$, portanto calcularemos $G_{1,2}$ ou $G_{2,1}$ já que $G_{1,2} = G_{2,1}$. Os pontos $pr1$ e $pr2$, são as coordenadas bidimensionais dos pontos de referência das células 1 e 2 respectivamente.

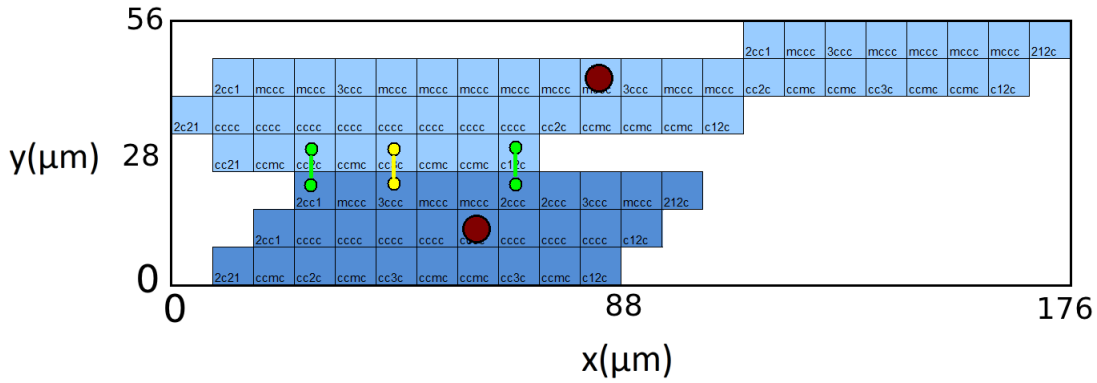


Figura 4.5: Microestrutura de duas células vizinhas, mostrando seus pontos de referência (pontos vermelhos) e suas *gap junctions* (pontos verdes e amarelos)

Os pontos que estão marcados de verde (*interplicate gap junctions*) e amarelo (*combined plicate gap junctions*) na figura 4.5 são os volumes que possuem *gap junction*.

Como podemos observar temos duas *gap junctions* do tipo *interplicate* e uma do tipo *combined plicate*. Com isso para a célula 1 temos:

$$pr_1 = (10, 5)$$

$$gp_{1,1} = (3, 3)$$

$$gp_{1,2} = (5, 3)$$

$$gp_{1,3} = (8, 3)$$

E para a célula 2:

$$pr_2 = (7, 1)$$

$$gp_{2,1} = (3, 2)$$

$$gp_{2,2} = (5, 2)$$

$$gp_{2,3} = (8, 2)$$

Como mencionado anteriormente, estes valores são os índices dos volumes que possuem *gap junction* e do volume que é ponto de referência.

Agora para cada uma das três *gap junctions* aplicamos a equação 4.34, e chegamos em:

$$dist_{1,2,1} = 14$$

$$dist_{1,2,2} = 10$$

$$dist_{1,2,3} = 6$$

O próximo passo é calcular a resistência associada a cada *gap junction* utilizando a equação 4.33, lembrando que os valores de $\sigma$ e $G$ de cada tipo de *gap junction* foram definidos na seção 4.1.1.

$$R_{1,2,1} = 6530.30k\Omega$$

$$R_{1,2,2} = 18629.03k\Omega$$

$$R_{1,2,3} = 4530.30k\Omega$$

Com estes valores agora podemos aplicar a equação 4.32 e encontrar a resistência equivalente.

$$R_{1,2} = 2338.85k\Omega$$

Finalmente aplicando a equação 4.35 chegamos em:

$$G_{1,2} = 0.000427561mS$$

## 4.3 Implementações numéricas paralelas

Simulações em grande escala, tais como as resultantes de uma discretização espacial fina de um tecido, são computacionalmente caras. Por exemplo, quando uma discretização $8\mu m$ é utilizada em um tecido de $1cm \times 1cm$, e o modelo BDK, que possui 41 variáveis diferenciais, é utilizado como modelo de células cardíacas, um total de $1250 \times 1250 \times 41 = 64,062,500$ incógnitas devem ser calculadas a cada passo de tempo. Além disso, para simular 100 ms de atividade elétrica cardíaca 64 milhões de incógnitas do sistema não linear de EDOs devem ser calculadas um milhão de vezes (com $\Delta t_o = 0.0001$ ms) e a EDP com 1.5 milhão de incógnitas deve ser computada dez mil vezes.

Para lidar com esse alto custo computacional, duas ferramentas distintas de computação paralela apresentadas no capítulo 2, seção 3, foram utilizadas em conjunto: MPI e GPGPU.

### *4.3.1 Implementação em agregados de computadores*

A implementação em **Cluster** é uma implementação paralela feita para ser executada em um agregado de computadores. A implementação em **Cluster** usa as bibliotecas PETSc [75] e MPI [57]. Ela usa um gradiente conjugado em paralelo pré-condicionado com ILU(0) (com bloco de Jacobi em paralelo) para resolver o sistema linear associado à discretização do EDP do modelo do monodomínio. Mais detalhes sobre esta implementação paralela podem ser encontrados em nossos artigos anteriores sobre o tema [21, 22, 23, 24, 62].

Para resolver os sistemas não-lineares de EDOs, o método de Euler explícito foi utilizado. Este é um problema embaraçosamente paralelo. Nenhuma dependência existe entre as soluções dos diferentes sistemas de EDOs de cada volume finito $Vol_{i,j}$. Portanto, é muito simples de implementar uma versão em paralelo do código: cada processo MPI é responsável por computar uma fração $Np$ do número total de volumes da simulação, onde $Np$ é o número de processos envolvidos na computação.

### *4.3.2 Implementação Multi-GPU*

Na implementação **Multi-GPU** decidimos manter a abordagem de agregado de computadores para a solução do sistema linear associado à discretização do EDP do modelo do monodomínio. Portanto, a implementação **Multi-GPU** também resolve a

EDP discretizada com o gradiente conjugado em paralelo pré-condicionado com ILU(0) (com bloco de Jacobi em paralelo) disponível na biblioteca PETSc.

No entanto, foram utilizadas múltiplas GPUs para acelerar a solução dos sistemas de EDOs. Esta é uma estratégia diferente das utilizadas anteriormente para as equações completas do bidomínio (EDP elíptica, EDP parabólica e sistemas de EDOs) que foram completamente implementadas em uma única GPU [30], ou para as equações completas do monodomínio (EDP parabólica e sistemas de EDOs) que foram completamente implementadas em uma única GPU [25, 26, 29].

A motivação para a escolha de uma estratégia diferente é baseada em vários motivos. Conforme apresentado em [29], o modelo do monodomínio pode ser acelerado 35 vezes utilizando uma única GPU, quando comparado com uma implementação paralela em OpenMP [76] executando em um computador *quad-core*. No entanto, este aumento de velocidade final obtido pela GPU vem de um aumento de velocidade próximo a 10 vezes para a solução da EDP e um aumento de velocidade próximo a 450 vezes para a solução dos sistemas não lineares de EDOs. Hoje em dia, como a arquitetura de vários núcleos se desenvolve rapidamente, pode-se encontrar facilmente no mercado um único computador equipado com 64 núcleos de processamento. Portanto, acreditamos que a resolução da EDP baseada em implementações paralelas nessas novas máquinas empregando bibliotecas como MPI ou OpenMP possam superar uma implementação em uma única GPU. Por outro lado, para a solução paralela dos sistemas não lineares de EDOs, uma única GPU ainda supera facilmente estes novos computadores baseados em vários núcleos. Isto nos leva a focar em implementações de GPU para as soluções paralelas de milhões de sistemas não-lineares de EDOs. Uma segunda motivação está relacionada com os precondicionadores que podem ser facilmente e eficientemente implementados para o método do gradiente conjugado em GPUs. Para as equações do bidomínio, precondicionadores geométricos *multigrid* eficientes [30] foram implementados em uma única GPU, e sofisticados precondicionadores algébricos *multigrid* [77] foram implementados em uma plataforma multi-GPU. No entanto, ambas as implementações só são viáveis para a solução do sistema linear associado à EDP elíptica das equações do bidomínio. Precondicionadores *multigrid* são muito caros e acabam sendo uma opção ineficiente para a solução da EDP parabólica, que é o tipo de EDP do modelo do monodomínio. Até agora, o precondicionador $w$-Jacobi, o mais barato, mas ineficaz

tem sido a melhor escolha para implementações GPU quando se trata da solução da EDP parabólica [29, 77]. No entanto, sabe-se bem que precondicionadores LU (ILU) incompletos combinados com Bloco de Jacobi ou métodos aditivos de Schwarz de decomposição de domínio [23] superam em muito precondicionadores do tipo Jacobi na computação em agregados para a solução da EDP do modelo do monodomínio. Este argumento favorece implementações em agregados como a melhor escolha para a solução paralela da EDP parabólica do modelo do monodomínio [78]. Finalmente, uma terceira e última motivação está relacionada com o problema particular que propomos investigar neste trabalho: modelos que revelam a microestrutura do tecido cardíaco. Outro trabalho recente apresenta uma implementação para o modelo do bidomínio em plataformas multi-GPU [79]. Ambas as EDPs e sistemas de EDOs foram implementadas em GPUs utilizando métodos explícitos, relaxamento de Jacobi e Euler explícito, respectivamente. Notamos que para o nosso modelo de tecido microscópico com discretização espacial de $8\mu m$, a utilização de um *solver* explícito e barato para a EDP seria muito ineficiente devido a graves restrições impostas pelas condições de estabilidade de CFL [69]. Portanto, mais uma vez, este argumento também favorece implementações em agregados baseadas em métodos implícitos para a solução paralela da EDP parabólica do modelo do monodomínio.

A implementação **Multi-GPU** utiliza CUDA[80] para implementar a solução numérica do modelo celular cardíaco BDK.

Como mencionado anteriormente no capítulo 2, seção 3.3, um *kernel* é uma função C especial que pode ser chamada pela CPU, mas é executada simultaneamente na GPU por várias *threads*. Dois núcleos *kernel*() foram desenvolvidos para resolver cada um dos sistemas de EDOs relacionadas com o modelo BDK. O primeiro *kernel* é responsável pelo estabelecimento das condições iniciais dos sistemas de EDOs, enquanto o segundo integra os sistemas de EDOs em cada intervalo de tempo.

Ambas as implementações do *kernel* foram otimizadas de várias maneiras diferentes. As variáveis de estado de $M$ células cardíacas foram armazenadas em um vetor chamado $SV$, cujo tamanho é igual a $M * Neq$, onde $Neq$ é o número de equações diferenciais do modelo iônico (neste trabalho, $Neq$ é igual a 41). O vetor $SV$ foi organizado de tal modo que as primeiras $M$ entradas correspondem à primeira variável de estado, as $M$ entradas seguintes correspondem à próxima variável de estado, e assim por diante. Além disso, para todos os modelos iônicos, as primeiras $M$ entradas do vetor $SV$ correspondem ao

potencial transmembrânico $V$. Durante a solução dos sistemas de EDPs, após a integração dos sistemas de EDOs, o potencial transmembrânico de cada nó deve ser passado para o *solver* do PETSC. Devido à organização de memória escolhida para o vetor $SV$, esta é uma tarefa simples desde que, como afirmado anteriormente, as primeiras $M$ entradas do vetor correspondam ao potencial transmembrânico $V$ de cada nó. Esta organização nos permite evitar operações extras de memória entre CPU e GPU, melhorando o desempenho. Outra escolha de implementação que causou um impacto positivo no desempenho foi a forma como o vetor $SV$ foi alocado. O vetor $SV$ foi alocado na memória global da GPU usando a rotina *cudaMallocPitch* que pertence à API do CUDA. Esta rotina pode preencher a alocação, a fim de garantir que os endereços de memória correspondentes de qualquer dada linha irão continuar a satisfazer os requisitos de alinhamento para as operações de coalescência executadas pelo hardware. Em suma, uma coalescência estrita requer que uma *thread* $j$ de um conjunto de $n$ *threads* tenha que acessar $u[j]$ se $u[0]$ é acessado pela *thread* 0, ou seja, todo acesso a dados tem distância fixa de $n$ elementos. Portanto, no primeiro *kernel*, para definir as condições iniciais, cada *thread* define os valores de todas as suas variáveis de estado. O *kernel* que resolve o sistema de EDOs opera de forma semelhante, ou seja, cada *thread* calcula e atualiza suas variáveis de estado e escreve em sua posição correta de memória que corresponde a suas variáveis. Além disso, o segundo *kernel* foi otimizado para utilizar as operações de memória locais tanto quanto possível.

Cada processo MPI chama, durante a sua execução, os dois núcleos CUDA, uma vez que a) dois ou mais processos MPI podem ser lançados na mesma máquina, e b) cada máquina tem apenas dois dispositivos GPU. Dessa forma, os processos irão escolher qual dispositivo usar através de um esquema *round-robin*. Assim, um dispositivo GPU pode ser utilizado em simultâneo por dois ou mais processos MPI.

A decomposição linear do domínio foi utilizada na implementação paralela. O domínio do tecido foi linearmente decomposto em $Np$ subdomínios não sobrepostos (ou $Np$ Tarefas, $T_1$ a $T_{Np}$, veja figura 4.6), onde $Np$ é o número de processos MPI ou núcleos de processamento. A solução paralela da EDP é implementada via PETSc [21], com cada núcleo de processamento $p$ responsável por atualizar as variáveis associadas ao subdomínio $T_p$. No nosso ambiente computacional cada máquina ou nó tem mais núcleos de CPU (8) do que GPUs (2). Por conseguinte, para a solução das EDOs cada dispositivo GPU será responsável pelo processamento de mais de uma tarefa. As tarefas atribuídas a um nó

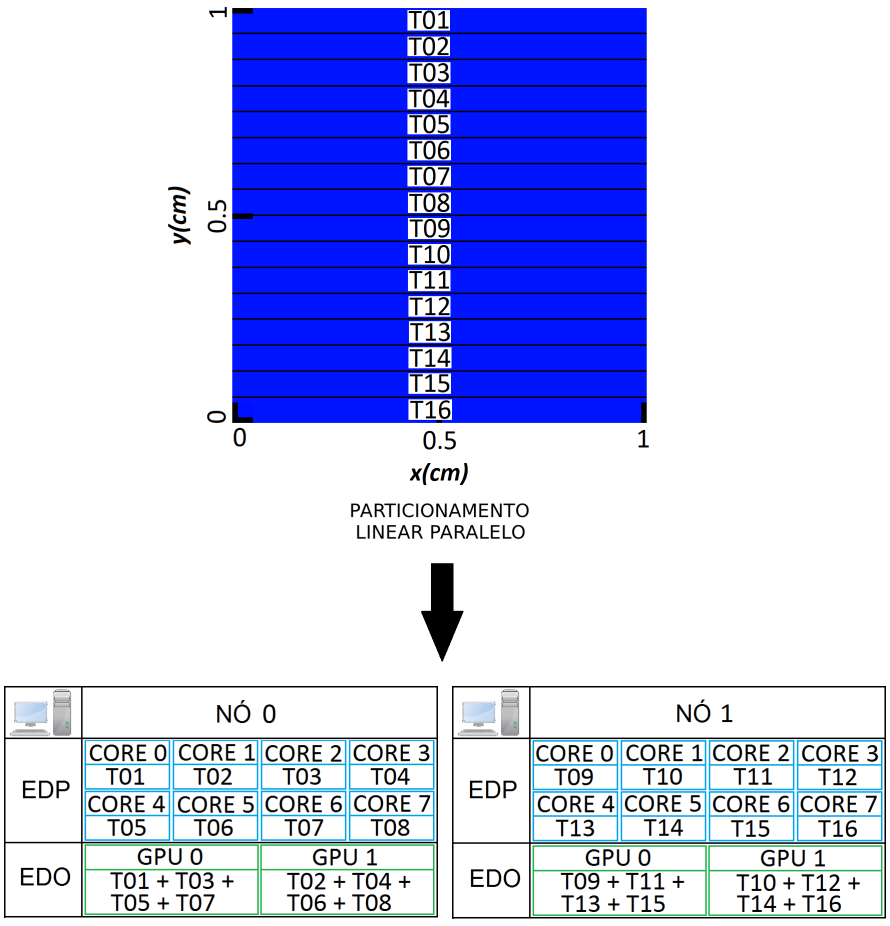


Figura 4.6: Decomposição linear paralela do tecido. Exemplo para o caso de dois nós (cada um com 8 núcleos e 2 dispositivos GPU, ou seja um total de 16 núcleos de CPU e quatro dispositivos GPU). Cada núcleo de CPU processa uma tarefa. Quatro tarefas são atribuídas a cada dispositivo GPU. Por exemplo, no Nó 0, a GPU 0 processa as tarefas $T_1$, $T_3$, $T_5$ e $T_7$, e a GPU 1 processa as tarefas $T_2$, $T_4$, $T_6$ e $T_8$.

são distribuídos para as GPUs em um esquema *round-robin*.

A figura 4.6 apresenta como o domínio do tecido será particionado para $Np = 16$ e duas máquinas (cada uma com 8 núcleos e 2 dispositivos GPU) sendo utilizadas.

Quatro tarefas são atribuídas a cada dispositivo GPU. Por exemplo, no Nó 0, a GPU 0 processa as tarefas $T_1$, $T_3$, $T_5$ e $T_7$, e a GPU 1 processa as tarefas $T_2$, $T_4$, $T_6$ e $T_8$.

Para a solução das EDOs ambas as versões sequenciais e paralelas (CUDA) do código utilizam precisão simples. Para a solução da EDP utiliza-se precisão dupla. Para o caso de simulações do monodomínio, como foi mostrado em [25], o uso de precisão simples em CUDA não afeta a precisão numérica do *solver*.

## 4.4 Experimentos de arritmia cardíaca

### *4.4.1 Reentrada em onda espiral e o Protocolo S1-S2*

Existem vários mecanismos de indução de reentrada [81, 82]. A teoria do ponto crítico introduzido por [81] explica as condições necessárias para o início de uma onda em espiral no miocárdio. A condição principal é a existência de um ponto crítico.

Podemos gerar reentrada utilizando um procedimento que explora esta característica. Primeiro aplicando um estímulo S1 na quarta parte esquerda do tecido (ver figura 4.7), com isso o potencial de ação se propaga para a direita e o tecido passa através das fases consecutivas de despolarização(excitação), platô, e repolarização antes de voltar ao repouso.

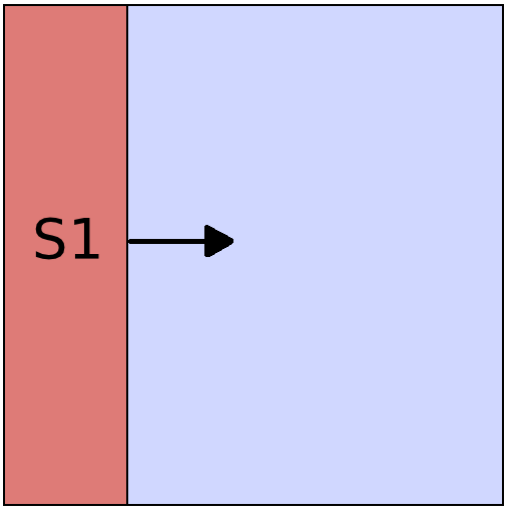


Figura 4.7: Estímulo S1.

O tecido está no período refratário durante o platô e o início da fase de repolarização.

Um estímulo S2 é aplicado no quadrante inferior esquerdo (ver figura 4.8) no momento em que parte da metade esquerda já tenha saído do período refratário mas a metade direita ainda esteja no período refratário. Isso faz com que a parte do tecido já fora do período refratário responda ao estímulo e uma nova frente de onda surja se propagando para cima. À medida que a outra parte do tecido se recupera do estímulo S1 (sai do período refratário), a frente de onda gira no sentido horário e continua a propagar ao longo das fibras na direção oposta. Este procedimento para indução de reentrada é conhecido como protocolo S1-S2.

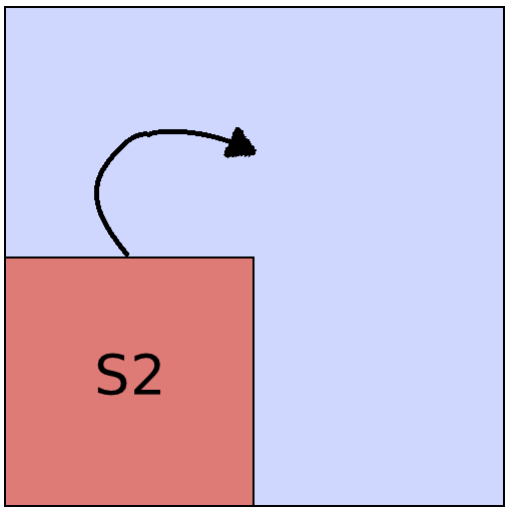


Figura 4.8: Estímulo S2.

Winfree [81] chama o período de tempo durante o qual a aplicação de um estímulo prematuro apropriado induz a reentrada de fase crítica.

Existem inúmeras outras maneiras para induzir a reentrada em um meio excitável, na próxima seção será apresentado uma outra forma de se produzir reentrada.

### *4.4.2 Remodelagem de gap junctions*

#### 4.4.2.1 Introdução

O papel das *gap junctions* como vias para a propagação ordenada de fluxo de corrente elétrica entre células, necessário para a contração sincronizada no coração saudável, nos leva a questionar o quanto as alterações da organização das *gap junctions* e da expressão de conexina pode contribuir para uma condução anormal e para a arritmia no coração humano doente [83, 84].

A origem da arritmia é, naturalmente, devido a múltiplos fatores, envolvendo uma interação entre o acoplamento da *gap junctions*, a excitabilidade da membrana e a arquitetura das células e dos tecidos [32].

Alterações relacionadas às *gap junctions* foram relatadas em um ventrículo doente: distúrbios na distribuição das *gap junctions* e níveis reduzidos de seu componente principal, a conexina 43 [85].

A perturbação da distribuição normal e ordenada da conexina 43 das *gap junctions* foi relatada pela primeira vez na zona de fronteira das cicatrizes do miocárdio ocasionadas por um infarto do miocárdio em pacientes no estágio final da doença cardíaca isquêmica [83].

Uma forma menos drástica da remodelação estrutural é associada com miocárdio hibernante no ventrículo humano [86].

O termo "miocárdio hibernante" denota uma condição em pacientes com doença arterial coronariana em que uma região do miocárdio mostra contração deficiente [87]. No "miocárdio hibernante" a quantidade total de conexina 43 é reduzida [86]. Estas observações destacaram a possibilidade de uma ligação entre alterações das conexinas 43 e a contração ventricular comprometida na doença de coração [85].

Além destes, diversos outros fatores e doenças associadas à mudanças de organização das *gap junctions*, bem como estudos que mostram um aumento na incidência de arritmias devido à estas mudanças na distribuição das *gap junctions* já foram relatados em outros trabalhos [85, 15, 83], mas não é objetivo deste trabalho aprofundar no assunto.

Devido a esta relação entre arritmia cardíaca e mudanças de organização das *gap junctions*, com o objetivo de reproduzir este aumento de incidência de arritmias, foi desenvolvido um método para remodelar as ligações entre os miócitos.

#### 4.4.2.2 Implementação

No modelo discreto duas abordagens foram utilizadas para modificar a ligação entre os miócitos. Para ambas as abordagens um parâmetro $\phi$ é definido, onde $\phi$ é um número real entre 0 e 1, que representa uma probabilidade. Na primeira abordagem, para cada célula é gerado aleatoriamente um número real entre 0 e 1, caso este número seja menor que $\phi$, a célula passa a não mais conduzir corrente elétrica para seus vizinhos, se torna isolante. Na prática, isto é feito removendo todas as suas ligações. Na segunda abordagem em vez

de cada célula ter uma probabilidade $\phi$ de se tornar isolante, é cada ligação que tem uma probabilidade $\phi$ de ser removida, isto é feito da mesma forma que na abordagem anterior, ou seja, gerando aleatoriamente um número real entre 0 e 1 e caso este número seja menor que $\phi$ a ligação é removida.

A figura 4.9A, mostra o código da implementação da primeira abordagem, e a 4.9B mostra o código da implementação da segunda abordagem.

No modelo microscópico ainda podemos implementar uma terceira abordagem, que é remover *gap junctions* de acordo com uma probabilidade $\phi$. Note que esta abordagem é diferente da segunda abordagem pois pode haver, e normalmente há, mais que uma *gap junction* entre duas células vizinhas, ou seja a remoção de uma *gap junction* não implica na remoção da ligação entre as duas células, somente diminui a capacidade de conduzir a corrente elétrica entre elas.

Segue abaixo um pseudocódigo para a implementação desta abordagem:

```
Para cada célula c faça{
    Para cada gap junction gj de c faça{
        rd = sorteia(0, 1);
        Se (rd < phi) faça{
            remove gj;
        }
    }
}
```

A

```
//Função que  sorteia um número real aleatório entre 0 e 1
//para cada célula e caso este número seja menor que o
//parâmetro prob (que representa a probabilidade de corte)
//remove todas as ligações da célula.
void cortaConexoes(vector<Cell*> *celulas, double prob){
    for (int x = 0; x < (int) celulas->size(); ++x) {

        Cell *c = celulas->at(x);

        //sorteia um número real entre 0 e 1
        double rd=(double)(rand())/(RAND_MAX);

        if((rd<prob)){

            //remove todas as ligações da célula c
            removeLigacoesCelula(c);
        }
    }
}
```

B

```
//Função que  sorteia um número real aleatório entre 0 e 1
//para cada ligação entre células e caso este número seja menor que o
//parâmetro prob (que representa a probabilidade de corte) remove a ligação
void cortaConexoes(vector<Cell*> *celulas, double prob){
    for (int x = 0; x < (int) celulas->size(); ++x) {

        Cell *c = celulas->at(x);
        int indiceCelula = x;

        for (int y = 0; y < (int) c->getVizinhos()->size(); ++y) {

            Vizinho *v =  c->getVizinhos()->at(y);
            int indiceVizinho = v->getCell();

            //condição para não sortear duas vezes cada ligação
            //já que c é vizinho de v e v também é vizinho de c
            if(indiceCelula < indiceVizinho){

                //sorteia um número real entre 0 e 1
                double rd=(double)(rand())/(RAND_MAX);

                if((rd<prob)){

                    //remove a ligação da célula c com a célula v
                    removeLigacaoCelula(c, v);
                }
            }
        }
    }
}
```

Figura 4.9: Código C++ implementado para o corte de ligações, em A) cada célula tem uma probabilidade $\phi$ de se tornar isolante (perder todas a suas ligações), já em B) é cada ligação que tem uma probabilidade $\phi$ de ser removida.

# 5 RESULTADOS

Os experimentos foram realizadas em um conjunto de 8 computadores SMP. Cada computador contém dois processadores *quad-core* Intel Xeon E5620 e 12 GB de memória RAM. Todos os nós executaram com o sistema operacional Linux na versão 2.6.18-194.17.4.el5. Os códigos foram compilados usando o gcc 4.1.2 e o CUDA 3.2. Cada nó contém dois Tesla C1060. A placa Tesla C1060 tem 240 núcleos CUDA e 4 GB de memória global.

Os valores utilizados para $\beta$ e $C_m$ foram definidos como 0.14 $cm^{-1}$ e $1.0\mu F/cm^2$, respectivamente [62]. O passo de tempo utilizado para resolver o sistema linear associado à equação 4.19 foi definido como $\Delta t_p = 0.01ms$ e para resolver o sistema não-linear de EDOs associado às equações 4.20 foi definido como $\Delta t_o = 0.0001$ ms.

Todos os testes para medir tempo foram realizados cinco vezes, os desvios padrões foram pequenos, resultando em um coeficiente de variação menor que 2% em todos os casos. O tempo médio de execução, em segundos, foi então utilizado para calcular o aumento de velocidade, definido como o tempo de execução sequencial, dividido pelo tempo de execução paralela.

## 5.1 Modelo Microscópico Paralelo

As simulações feitas com o modelo microscópico foram realizadas com uma discretização espacial de 8$\mu$m, e valores de condutividade heterogêneos, tal como descrito na seção 4.1.1. Duas configurações diferentes de tecidos cardíaco foram usadas para testar o modelo e as implementações paralelas: a simulação de $10ms$ utilizando um tecido cardíaco de $0.5cm \times 0.5cm$ de tamanho que foi estimulado no centro; e a simulação de $10ms$ utilizando um tecido cardíaco de $1.0cm \times 1.0cm$ de tamanho que foi estimulado no centro.

A figura 5.1 mostra a propagação de um estímulo central sobre o tecido de tamanho $1cm \times 1cm$ em diferentes instantes de tempo. Como esperado, macroscopicamente, a propagação parece muito suave e contínua. No entanto, ao destacar uma região menor de tamanho $1mm \times 1mm$, veja a figura 5.2, já podemos observar a natureza discreta da propagação, ou seja, a influência da microestrutura cardíaca na propagação dos potenciais

de ação.

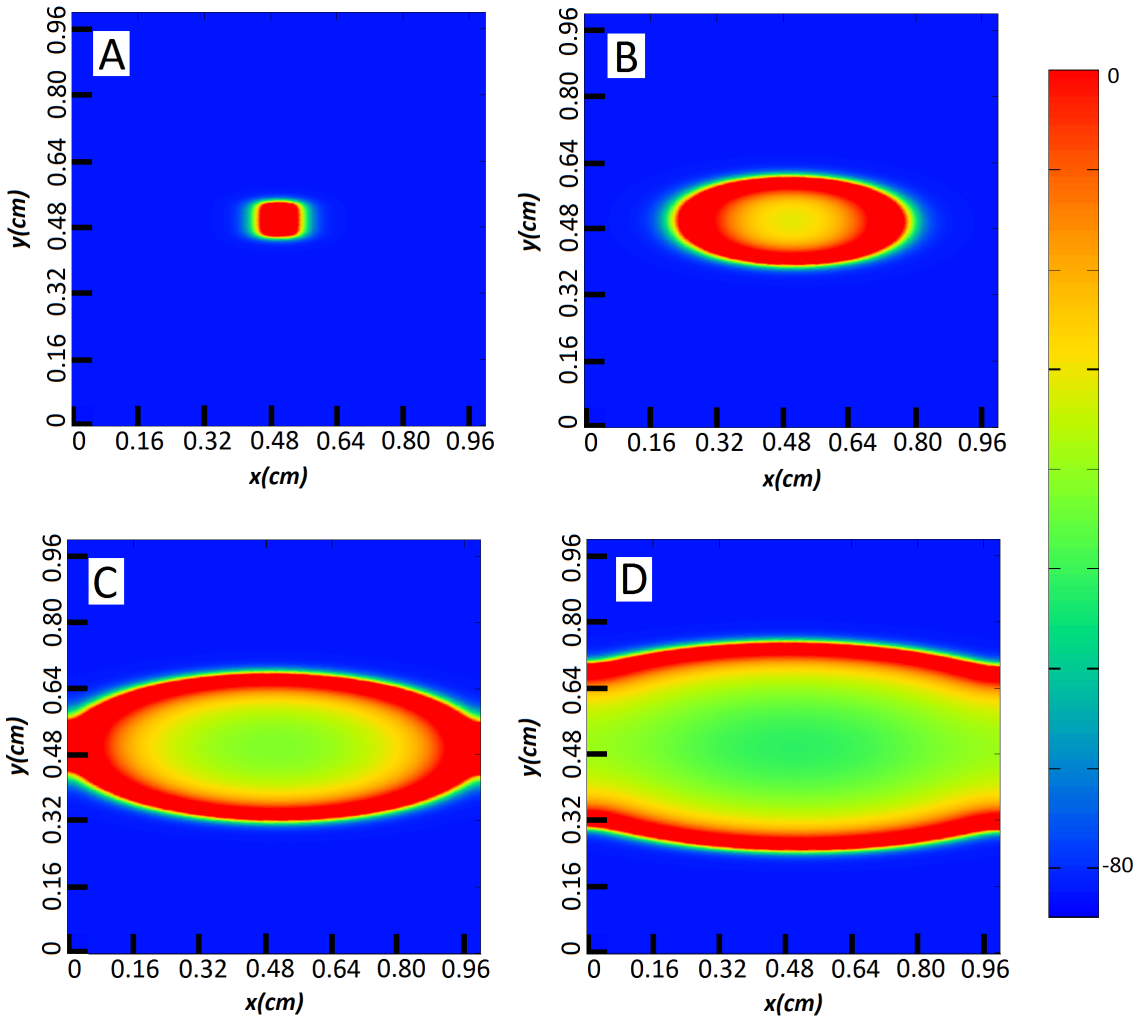


Figura 5.1: Propagação do potencial de ação (potencial transmembrânico) após um estímulo central sobre um tecido de $1cm \times 1cm$ de tamanho em diferentes instantes de tempo. A) $t = 1ms$, B) $t = 7ms$, C) $t = 13ms$ e D) $t = 19ms$.

A tabela 5.1 apresenta os tempos de execução das implementações paralelas para o experimento com um tecido quadrado de $0.5cm \times 0.5cm$. A tabela 5.2 apresenta os ganhos de velocidade para este mesmo experimento.

Como se pode observar, o tempo gasto resolvendo as EDOs é responsável por quase 90% do tempo de execução. Também pode ser observado que, embora os aumentos de velocidade obtidos com a implementação **Cluster** sejam consideráveis e quase lineares (perto de 61 com 64 núcleos), o tempo de execução total permanece elevado. Em relação a implementação **Multi-GPU** os resultados são muito melhores. Deve-se ressaltar que, embora 64 tenham sido utilizados na simulação, somente 16 dispositivos GPGPU estão disponíveis para a execução da simulação, assim 8 processadores compartilham

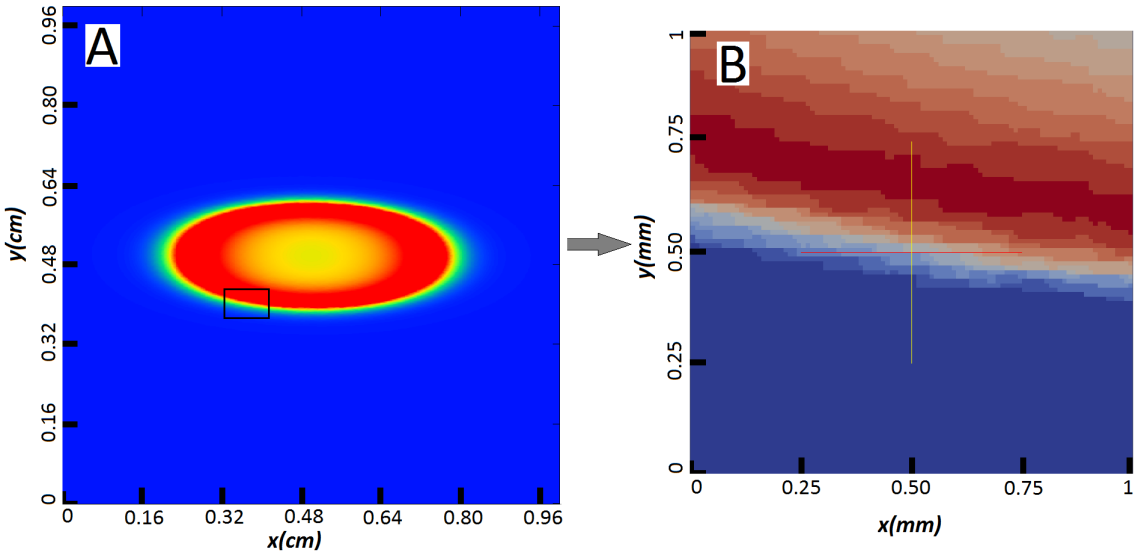


Figura 5.2: A) Potencial transmembrânico em $t = 7ms$ após um estímulo central sobre um tecido de $1cm \times 1cm$ de tamanho. B) Detalhes microscópicos que revelam a natureza discreta da propagação do potencial de ação, ou seja, a influência da microestrutura cardíaca em uma simulação com tecido de tamanho grande.

2 dispositivos GPGPU por máquina. Como se pode observar, o *speedup* (ganho de velocidade) obtido foi enorme, a simulação utilizando a implementação **Multi-GPU** em 8 máquinas foi cerca de 343 vezes mais rápida que a mesma simulação executada em um processador de núcleo único. O tempo de execução cai de 1.6 dias (usando um único núcleo de processamento) para apenas 6.7 minutos (utilizando a plataforma multi-GPU de 8 nós).

Tabela 5.1: Tempo médio de execução das implementações paralelas para um tecido de $0.5cm \times 0.5cm$. Os tempos de execução são apresentados em segundos.

| Implementação paralela | Núcleos | Tempo total | Tempo EDO | Tempo EDP |
|---|---|---|---|---|
| *Cluster* | 1 | 137964 | 132590 | 5264 |
| *Cluster* | 8 | 18492 | 17210 | 1262 |
| *Cluster* | 16 | 9922 | 9316 | 595 |
| *Cluster* | 32 | 4198 | 3884 | 311 |
| *Cluster* | 64 | 2283 | 2087 | 191 |
| Multi-GPU | $64 + 16$ GPUs | 401.84 | 209.4 | 187 |

Tabela 5.2: *Speedup* das implementações paralelas para um tecido de $0.5cm \times 0.5cm$. Os tempos de execução são apresentados em segundos.

| Implementação paralela | Núcleos | *Speedup* |
|---|---|---|
| *Cluster* | 8 | 7.5 |
| *Cluster* | 16 | 13.9 |
| *Cluster* | 32 | 32.9 |
| *Cluster* | 64 | 60.4 |
| Multi-GPU | 64 + 16 GPUs | 343 |

Tabela 5.3: Tempo médio de execução das implementações paralelas para um tecido de $1.0cm \times 1.0cm$. Os tempos de execução são apresentados em segundos.

| Implementação paralela | Núcleos | Tempo total | Tempo EDO | Tempo EDP |
|---|---|---|---|---|
| *Cluster* | 1 | 546507 | 523331 | 23177 |
| *Cluster* | 64 | 8934 | 8313 | 607 |
| Multi-GPU | 64 + 16 GPUs | 1302 | 682 | 611 |

Tabela 5.4: *Speedup* das implementações paralelas para um tecido de $1.0cm \times 1.0cm$. Os tempos de execução são apresentados em segundos.

| Implementação paralela | Núcleos | *Speedup* |
|---|---|---|
| *Cluster* | 64 | 61.2 |
| Multi-GPU | 64 + 16 GPUs | 420 |

A tabela 5.3 apresenta os tempos de execução das implementações paralelas para o experimento com um tecido quadrado de $1cm \times 1cm$. A tabela 5.4 apresenta os ganhos de velocidade para este mesmo experimento. Mais uma vez, os *speedups* obtidos com a implementação **Cluster** foram quase lineares (61 com 64 núcleos). Em relação à implementação **Multi-GPU** os resultados mais uma vez também foram muito melhores.

Como se pode observar, o *speedup* obtido foi enorme, a simulação utilizando a implementação **Multi-GPU** em 8 máquinas foi cerca de 420 vezes mais rápida que a mesma simulação executada em um processador de núcleo único.

O tempo de execução diminui de mais de 6 dias (usando um núcleo de processamento) para apenas 21 minutos (utilizando a plataforma multi-GPU de 8 nós). Podemos observar também que a implementação **Multi-GPU** foi quase 7 vezes mais rápida que a implementação **Cluster** quando executando em 8 computadores.

## 5.2 Simulação de reentrada em espiral utilizando protocolo S1-S2

A implementação paralela abre a possibilidade de simular formações de ondas em espirais (uma forma de atividade reentrante auto-sustentada fortemente associado com arritmia cardíaca), que demandariam grande tempo de computação para serem simuladas em uma versão sequencial do código, veja figura 5.3.

Tanto quanto é do nosso conhecimento, esta é a primeira vez que as ondas em espiral são simulados utilizando um modelo cardíaco que leva em conta a microestrutura do tecido cardíaco e um modelo de miócitos moderno e complexo. Utilizando o protocolo S1-S2, apresentado na seção 4.4.1, após algumas tentativas para encontrar a janela vulnerável correta conseguimos gerar uma onda espiral sustentada.

Para este experimento foi necessário executar o modelo por $200ms$, utilizando um tecido cardíaco de $1.0cm \times 1.0cm$. Como apresentado na tabela 5.3, a execução de $10ms$ deste modelo utilizando um tecido cardíaco de $1.0cm \times 1.0cm$, em um único processador, demorou mais de 6 dias, portanto sem a implementação **Multi-GPU** paralela, cada execução teria demorado mais de 126 dias utilizando um computador de núcleo único ou pouco mais de 4 dias usando nossa implementação **Cluster** em execução com 64 núcleos, mas sem a GPUs. Já com a implementação **Multi-GPU** paralela utilizando 8 máquinas (cada uma com 8 núcleos e 2 dispositivos GPU), cada execução demorou pouco mais de 7 horas.

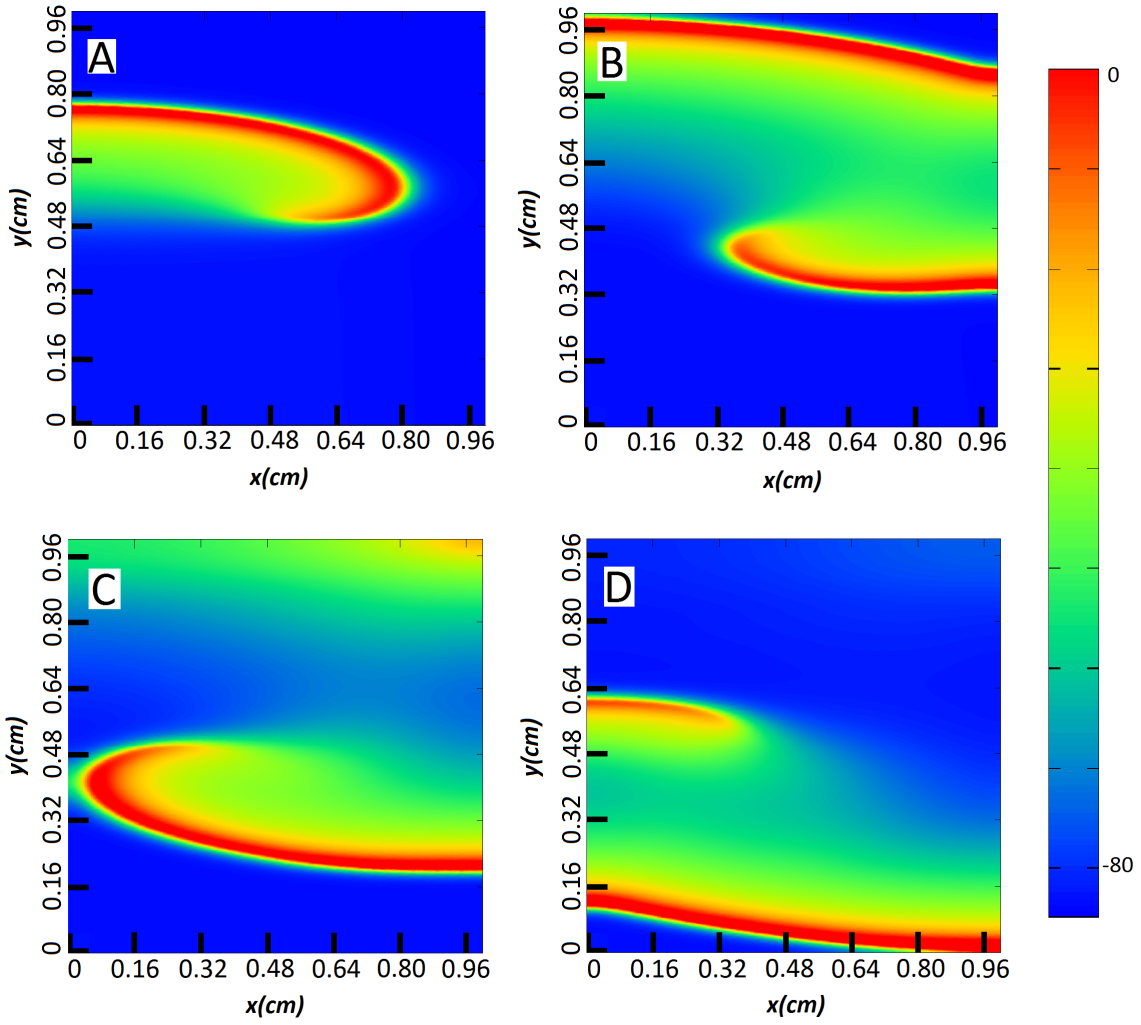


Figura 5.3: Formação de uma onda em espiral após um estímulo usando o protocolo S1-S2 em diferentes instantes de tempo : A) $t = 80ms$, B) $t = 100ms$, C) $t = 112ms$ e D) $t = 120ms$.

## 5.3 Modelo Discreto Paralelo

Para comparar o ganho de velocidade do modelo discreto com relação ao modelo microscópico, foram realizadas simulações de $10ms$ utilizando um tecido cardíaco de $1cm \times 1cm$ de tamanho que foi estimulado no centro.

A figura 5.4 mostra a propagação de um estímulo central sobre o tecido de tamanho $1cm \times 1cm$ em diferentes instantes de tempo. Este mesmo experimento também foi realizado no modelo microscópico como mostra a figura 5.2. A única diferença entre as soluções é que na solução discreta a propagação é menos suave, uma vez que no modelo discreto um único potencial transmembrânico é associado a célula, enquanto no modelo microscópico cada volume da célula possui um potencial. Ampliando uma região das figuras podemos notar essa diferença como mostra a figura 5.5.

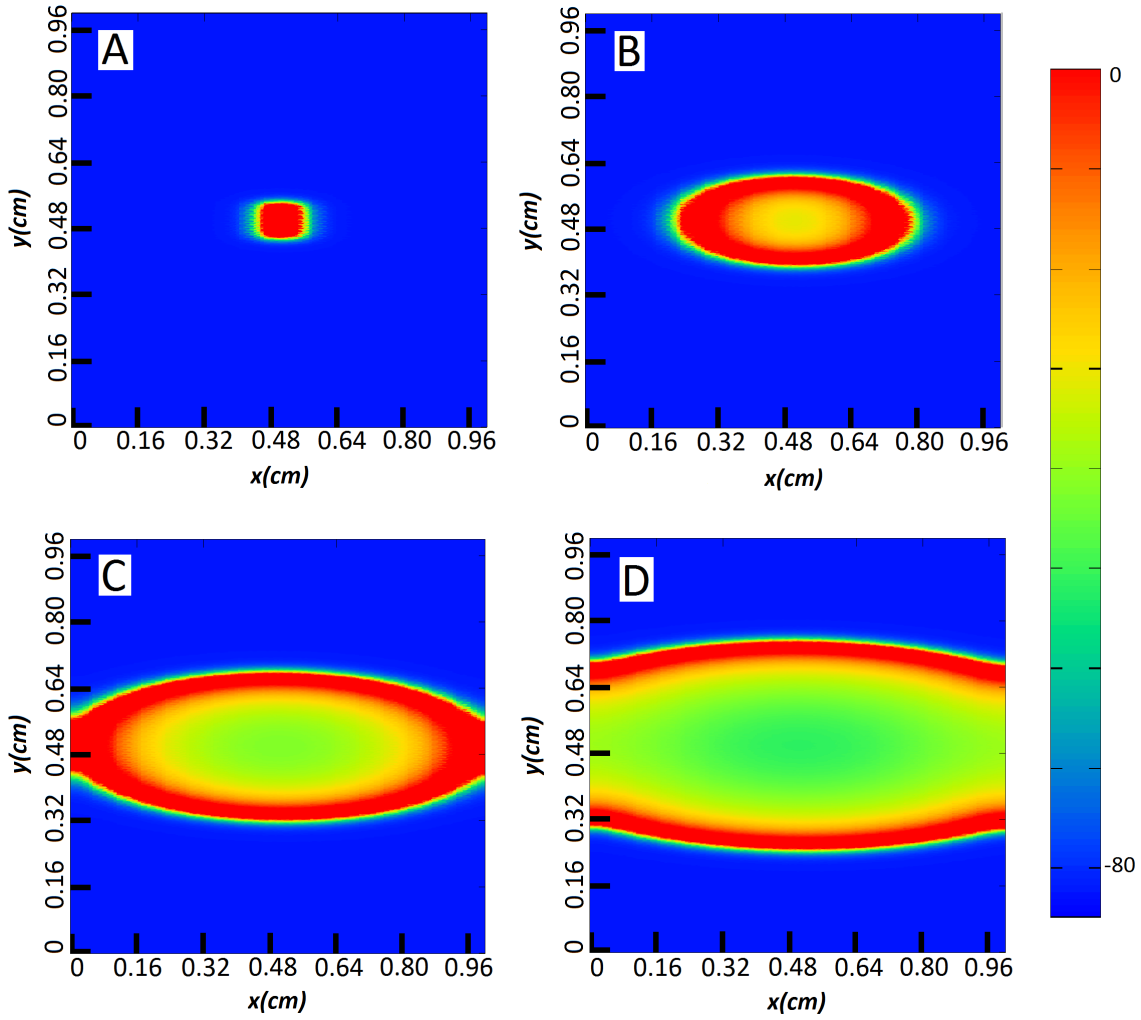


Figura 5.4: Propagação do potencial de ação (potencial transmembrânico) após um estímulo central sobre um tecido de $1cm \times 1cm$ de tamanho, utilizando o modelo discreto, em diferentes instantes de tempo. A) $t = 1ms$, B) $t = 7ms$, C) $t = 13ms$ e D) $t = 19ms$.

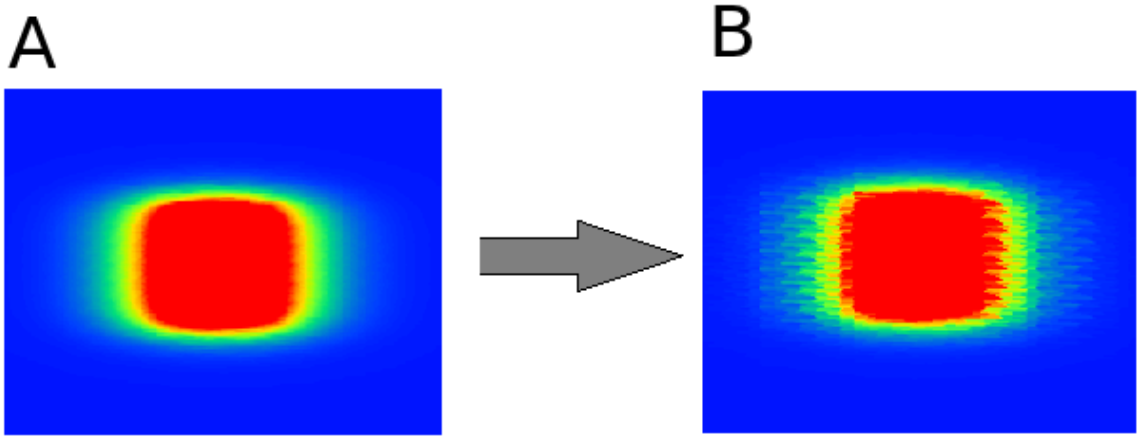


Figura 5.5: Diferença entre os modelos: A) microscópico, B) discreto.

A tabela 5.5 apresenta os tempos de execução e aceleração da implementações paralela do modelo discreto.

Tabela 5.5: Tempo médio de execução e aceleração da implementação paralela do modelo discreto. Os tempos de execução são apresentados em segundos.

| Implementação paralela | Núcleos | Tempo total | Aceleração |
|---|---|---|---|
| Sequencial Microscópico | 1 | 546507 | – |
| Multi-GPU Microscópico | 64 + 16 GPUs | 1302 | 420 |
| Multi-GPU Discreto | 64 + 16 GPUs | 65 | 8408 |

Como se pode observar, o ganho de velocidade em relação ao modelo microscópico foi muito alto, cerca de 8408 vezes mais rápida que a mesma simulação utilizando o modelo microscópico executada em um processador de núcleo único. O tempo de execução diminui de mais de 6 dias (usando um núcleo de processamento) para apenas 65 segundos.

## 5.4 Simulação de reentrada utilizando remodelagem de gap junctions

Com este modelo discreto paralelo, muito mais veloz, podemos agora realizar experimentos que exigem um grande número de simulações, como simular reentrada através da remodelagem de *gap junctions* tal como descrito na seção 4.4.2.

Para modificar a ligação entre os miócitos, foi utilizada nesse experimento a primeira abordagem apresentada na seção 4.4.2.2, onde para cada célula é gerado aleatoriamente um número real entre 0 e 1, caso este número seja menor que um parâmetro $\phi$ definido, a célula passa a não mais conduzir corrente elétrica para seus vizinhos, se torna isolante.

A geração de reentrada utilizando este método é altamente dependente deste parâmetro $\phi$, pois se ele for muito alto o potencial de ação não consegue se propagar pelo tecido, uma vez que existem muitas células isolantes no caminho. Já se o parâmetro for muito baixo há pouca alteração no propagação normal do potencial de ação e dificilmente ocorrerá reentrada.

Devido a esta sensibilidade ao parâmetro $\phi$, centenas de simulações foram realizadas para diferentes valores de $\phi$. Para cada valor de $\phi$ foram executadas 100 simulações. A tabela 5.6 mostra o valor de $\phi$ e percentagem das simulações que geraram algum tipo de reentrada com este valor.

Tabela 5.6: Percentagem de simulações que geraram algum tipo de reentrada para cada valor de $\phi$.

| $\phi$ | 0.44 | 0.45 | 0.46 | 0.47 | 0.48 | 0.49 | 0.50 | 0.51 | 0.52 | 0.53 |
|---|---|---|---|---|---|---|---|---|---|---|
| | 0% | 6% | 10% | 23% | 33% | 21% | 13% | 5% | 1% | 0% |

A figura 5.6 mostra uma simulação em que houve reentrada. Para esta simulação temos: $\phi = 0.48$; tecido de tamanho $1cm \times 1cm$; e um estímulo inicial é aplicado na borda esquerda do tecido. Podemos notar na figura que após o estímulo inicial (A) a onda se propaga pelo tecido de maneira irregular (B). Em (C) podemos observar células que não foram estimuladas pela frente de onda sendo estimuladas e em (D) ocorre uma reentrada, células que foram excitadas pelo estímulo inicial são novamente excitadas.

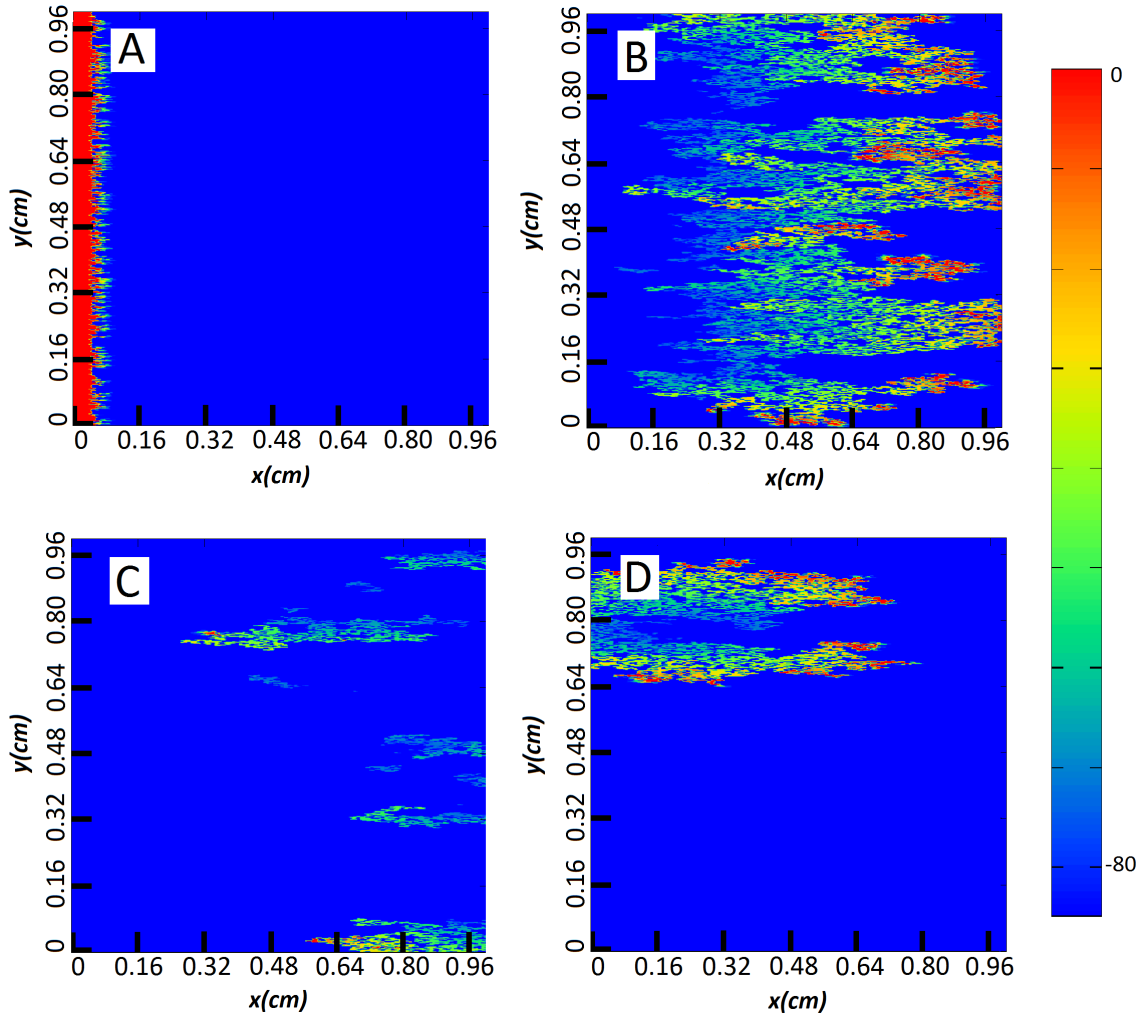


Figura 5.6: Formação de reentrada utilizando remodelagem de *gap junctions* em instantes de tempo : A) $t = 2ms$, B) $t = 44ms$, C) $t = 82ms$ e D) $t = 122ms$.

# 6 DISCUSSÃO E TRABALHOS FUTUROS

Nossos resultados mostram que a implementação **Multi-GPU** paralela descrita na Seção 4.3.2 foi capaz de acelerar significativamente a solução numérica de um modelo de electrofisiologia cardíaca que capta a microestrutura do tecido cardíaco (utilizando uma discretização espacial muito fina) e baseia-se em um modelo celular cardíaco moderno e complexo (com a formulação de Cadeia de Markov que tem sido extensivamente utilizada para a caracterização de canais iônicos, ver 2.9). *Speedups* de cerca de 420 vezes foram obtidos, reduzindo os tempos de execução de mais de 6 dias (com um núcleo de processamento) para apenas 21 minutos (utilizando a plataforma multi-GPU de 8 nós). A implementação do modelo discreto paralelo equivalente ao modelo microscópico reduziu ainda mais este tempo, para apenas 65 segundos. A implementação **Multi-GPU** híbrida apresentada neste trabalho é ainda mais atraente quando se considera que as arquiteturas de GPUs e de processadores *multicore* vão continuar a evoluir em um ritmo acelerado.

No entanto, acreditamos que a implementação paralela pode ser melhorada. Por exemplo, na implementação atual, os núcleos estão ociosos enquanto aguardam os resultados das EDOs não-lineares que estão sendo determinados pela GPU. Como trabalho futuro, pretendemos avaliar técnicas de balanceamento de carga diferentes para distribuir melhor as tarefas paralelas entre dispositivos GPU e núcleos de CPU e fazer um uso mais eficiente de todos os recursos computacionais. Outra melhoria possível está relacionada com o paralelismo multi-nível introduzido para a solução das equações do bidomínio [24] que combina o paralelismo de tarefas (via *pipeline*) e o paralelismo de dados (via decomposição de dados). Acreditamos que uma combinação similar de paralelismo de dados e de tarefa pode ser também explorada para a solução das equações do monodomínio, aumentando ainda mais a eficiência paralela de algoritmos.

Com o desenvolvimento do modelo discreto em paralelo, conseguimos reproduzir fenômenos de reentrada, relacionados à arritmia. No entanto, muitos experimentos ainda podem ser feitos, como por exemplo, realizar mais experimentos de reentrada com as outras abordagens apresentadas na seção 4.4.2.

Além disso, acreditamos que a realização de simulações comparando os modelos discreto com modelos microscópico nos proporcionará um melhor entendimento dos benefícios e limitações de cada um deles.

# 7 CONCLUSÃO

Doenças cardiovasculares estão relacionadas a um alto índice de mortalidade no mundo. Dentre as doenças cardiovasculares, as relacionadas à eletrofisiologia cardíaca estão entre as principais causas de morte. Essas disfunções elétricas são denominadas arritmias cardíacas. A modelagem computacional do coração tornou-se uma ferramenta importante no teste de novas drogas e no desenvolvimento de novos equipamentos e técnicas de diagnóstico. Porém, os modelos modernos são de grande complexidade e as simulações numéricas são computacionalmente muito custosas.

Neste trabalho desenvolvemos um modelo microscópico de eletrofisiologia cardíaca que capta a microestrutura do tecido cardíaco, utilizando uma discretização espacial muito fina. Além disso, utilizamos um modelo celular moderno e complexo baseado em Cadeias de Markov para a caracterização da dinâmica de estruturas e elementos subcelulares, como os canais iônicos.

Para lidar com os desafios computacionais, o modelo foi paralelizado usando uma abordagem híbrida: a computação em agregados de computadores e GPGPUs, e um modelo equivalente discreto, muito menos custoso computacionalmente, foi desenvolvido.

Diferentes experimentos foram realizados neste trabalho para os testes de desempenho. Nós mostramos que em todos os casos, a nossa aplicação multi-GPU paralela foi capaz de reduzir significativamente os tempos de execução das simulações, além disso mostramos também que houve um enorme ganho de velocidade com a utilização do modelo discreto.

Com esse aumento de desempenho proporcionado pelo modelo discreto em conjunto com a implementação multi-GPU, simulamos diferentes cenários que reproduziram arritmias cardíacas do tipo reentrada. Acreditamos que esta nova implementação abre caminho para a investigação de muitas questões em aberto associadas à natureza complexa das arritmias cardíacas.

# REFERÊNCIAS

[1] WHO, "World Health Organization", http://www.who.int/, 2010.

[2] SACHSE, F. B., *Computational cardiology: modeling of anatomy, electrophysiology, and mechanics*. v. 2966. Springer Verlag, 2004.

[3] HODGKIN, A., HUXLEY, A., "A quantitative description of membrane current and its application to conduction and excitation in nerve", *Journal of Physiology*, v. 117, pp. 500–544, 1952.

[4] PLONSEY, R., "Bioelectric sources arising in excitable fibers (ALZA lecture)." *Ann Biomed Eng*, v. 16, n. 6, pp. 519–46, 1988.

[5] ALIEV, R. R., PANFILOV, A. V., "A simple two-variable model of cardiac excitation", *Chaos, Solitons and Fractals*, v. 7, n. 3, pp. 293 – 301, 1996.

[6] BONDARENKO, V. E., SZIGETI, G. P., BETT, G. C., KIM, S. J., RASMUSSON, R. L., "Computer Model of Action Potential of Mouse Ventricular Myocytes", *American Journal of Physiology - Heart and Circulatory Physiology*, v. 287, pp. H1378–H1403, 2004.

[7] CLANCY, C. E., RUDY, Y., "Linking a genetic defect to its cellular phenotype in a cardiac arrhythmia", *Nature*, v. 400, pp. 566–569, 1999.

[8] BRENNAN, T., FINK, M., RODRIGUEZ, B., "Multiscale modelling of drug-induced effects on cardiac electrophysiological activity", *European Journal of Pharmaceutical Sciences*, v. 36, n. 1, pp. 62 – 77, 2009.

[9] CLANCY, C. E., ZHU, Z. I., RUDY, Y., "Pharmacogenetics and anti-arrhythmic drug therapy: a theoretical investigation", *American Journal of Physiology - Heart and Circulatory Physiology*, v. 292, n. 1, pp. H66–H75, January 2007.

[10] IYER, V., MAZHARI, R., WINSLOW, R. L., "A Computational Model of the Human Left-Ventricular Epicardial Myocyte", *Biophysical Journal*, v. 87, pp. 1507–1525, 2004.

[11] SPACH, M. S., HEIDLAGE, J. F., "The Stochastic Nature of Cardiac Propagation at a Microscopic Level: Electrical Description of Myocardial Architecture and Its Application to Conduction", *Circulation Research*, v. 76, n. 3, pp. 366–380, 1995.

[12] JACQUEMET, V., HENRIQUEZ, C. S., "Loading effect of fibroblast-myocyte coupling on resting potential, impulse propagation, and repolarization: insights from a microstructure model", *American Journal of Physiology - Heart and Circulatory Physiology*, v. 294, n. 5, pp. H2040–H2052, May 2008.

[13] HUBBARD, M. L., YING, W., HENRIQUEZ, C. S., "Effect of gap junction distribution on impulse propagation in a monolayer of myocytes: a model study", *Europace*, v. 9, n. suppl 6, pp. vi20–vi28, 2007.

[14] KLÉBER, A. G., RUDY, Y., "Basic Mechanisms of Cardiac Impulse Propagation and Associated Arrhythmias", *Physiological Reviews*, v. 84, n. 2, pp. 431–488, 2004.

[15] JONGSMA, H. J., WILDERS, R., "Gap Junctions in Cardiovascular Disease", *Circulation Research*, v. 86, n. 12, pp. 1193–1197, 2000.

[16] WANG, Y., RUDY, Y., "Action potential propagation in inhomogeneous cardiac tissue: safety factor considerations and ionic mechanism", *American Journal of Physiology - Heart and Circulatory Physiology*, v. 278, n. 4, pp. H1019–H1029, 2000.

[17] SPACH, M. S., BARR, R. C., "Effects of Cardiac Microstructure on Propagating Electrical Waveforms", *Circulation Research*, v. 86, n. 2, pp. e23–e28, 2000.

[18] SHAW, R. M., RUDY, Y., "Ionic Mechanisms of Propagation in Cardiac Tissue", *Circulation Research*, v. 81, n. 5, pp. 727–741, 1997.

[19] STINSTRA, J., MACLEOD, R., HENRIQUEZ, C., "Incorporating Histology into a 3D Microscopic Computer Model of Myocardium to Study Propagation at a Cellular Level", *Annals of Biomedical Engineering*, v. 38, pp. 1399–1414, 2010, 10.1007/s10439-009-9883-y.

[20] ROBERTS, S. F., STINSTRA, J. G., HENRIQUEZ, C. S., "Effect of Nonuniform Interstitial Space Properties on Impulse Propagation: A Discrete Multidomain Model", *Biophysical Journal*, v. 95, n. 8, pp. 3724–3737, 2008.

[21] DOS SANTOS, R. W., PLANK, G., BAUER, S., VIGMOND, E. J., "Parallel multigrid preconditioner for the cardiac bidomain model." *IEEE Trans Biomed Eng*, v. 51, n. 11, pp. 1960–8, 2004.

[22] PLANK, G., LIEBMANN, M., DOS SANTOS, R. W., VIGMOND, E. J., HAASE, G., "Algebraic Multigrid Preconditioner for the Cardiac Bidomain Model", *IEEE Trans Biomed Eng*, v. 54, pp. 585–96, 2007.

[23] DOS SANTOS, R. W., PLANK, G., BAUER, S., VIGMOND, E. J., "Preconditioning Techniques for the Bidomain Equations", *Lecture Notes In Computational Science And Engineering*, v. 40, pp. 571–580, 2004.

[24] XAVIER, C., OLIVEIRA, R., VIEIRA, V. F., DOS SANTOS, R. W., MEIRA, W., "Multi-Level Parallelism for the Cardiac Bidomain Equations", *International Journal of Parallel Programming*, v. 37, pp. 572–592, 2009, 10.1007/s10766-009-0110-0.

[25] ROCHA, B. M., CAMPOS, F. O., AMORIM, R. M., PLANK, G., SANTOS, R. W. D., LIEBMANN, M., HAASE, G., "Accelerating cardiac excitation spread simulations using graphics processing units", *Concurrency and Computation: Practice and Experience*, v. 23, n. 7, pp. 708–720, 2011.

[26] ROCHA, B. M., CAMPOS, F. O., PLANK, G., DOS SANTOS, R. W., LIEBMANN, M., HAASE, G., "Simulations of the Electrical Activity in the Heart with Graphic Processing Units", In: WYRZYKOWSKI, R., DONGARRA, J., KARCZEWSKI, K., WASNIEWSKI, J. (eds), *Parallel Processing and Applied Mathematics*, v. 6067, pp. 439–448, *Lecture Notes in Computer Science*, Springer Berlin, Heidelberg, 2010.

[27] AMORIM, R. M., ROCHA, B. M., CAMPOS, F. O., DOS SANTOS, R. W., "Automatic code generation for solvers of cardiac cellular membrane dynamics in GPUs". In: *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*, pp. 2666–2669, 2010.

[28] AMORIM, R. M., HAASE, G., LIEBMANN, M., DOS SANTOS, R. W., "Comparing CUDA and OpenGL implementations for a Jacobi iteration". In: *High Performance Computing Simulation, 2009. HPCS '09. International Conference on*, pp. 22 –32, june 2009.

[29] OLIVEIRA, R. S., ROCHA, B. M., AMORIM, R. M., CAMPOS, F., MEIRA, W., TOLEDO, E., DOS SANTOS, R. W., "Comparing CUDA, OpenCL and OpenGL Implementations of the Cardiac Monodomain Equations", In: WYRZYKOWSKI, R., DONGARRA, J., KARCZEWSKI, K., WASNIEWSKI, J. (eds), *Parallel Processing and Applied Mathematics*, v. 7204, pp. 111–120, *Lecture Notes in Computer Science*, Springer Berlin, Heidelberg, 2012.

[30] AMORIM, R. M., DOS SANTOS, R. W., "Solving the cardiac bidomain equations using graphics processing units", *Journal of Computational Science*, , n. 0, 2012.

[31] ALONSO, S., BÄR, M., "Reentry near the percolation threshold in a heterogeneous discrete model for cardiac", *Physikalisch-Technische Bundesanstalt*.

[32] SHAW, R. M., RUDY, Y., "Ionic Mechanisms of Propagation in Cardiac Tissue: Roles of the Sodium and L-type Calcium Currents During Reduced Excitability and Decreased Gap Junction Coupling", *Circulation Research*, v. 81, n. 5, pp. 727–741, 1997.

[33] JEFF HARDIN, GREGORY PAUL BERTONI, L. J. K., *Becker's World of the Cell (8th Edition)*. Benjamin Cumming, 2011.

[34] MEDICINE, J. H., "The Johns Hopkins Arrhythmia Service", http://www.hopkinsmedicine.org/.

[35] GOLDMAN LEE; AUSIELLO, D., *Cecil - Tratado de Medicina Interna*. Elsevier / Medicina Nacionais, 2005.

[36] GUYTON, A., *Tratado de Fisiologia Médica*. 9th ed. Guanabara Koogan, 1991.

[37] FAWCETT, D., MCNUTT, N., "The Ultrastructure of the cat Myocardium", *The Journal of Cell Biology*, v. 42, pp. 1–45.

[38] ANTZELEVITCH, C., "Basic Mechanisms of reentrant arrythmias", *Current Opinion in Cardiology*, v. 16, pp. 1–7, 2001.

[39] CAMPOS, F. O., *Modelagem computacional da eletrofisiologia cardíaca: o desenvolvimento de um novo modelo para células de camundongos e a avaliação de novos esquemas numéricos*, Master's Thesis, UFJF - Programa de Pós-Graduação Modelagem Computacional, 2008.

[40] DE OLIVEIRA, B. L., *Modelagem quantitativa da eletromecânica do tecido cardíaco humano*, Master's Thesis, UFJF - Programa de Pós-Graduação Modelagem Computacional, 2011.

[41] COSTA, C. M., *Modelagem da microestrutura de Tecdios cardíacos*, Master's Thesis, UFJF - Programa de Pós-Graduação Modelagem Computacional, 2011.

[42] KEENER, J., SNEYD, J., *Mathematical Physiology*. Springer, 1998.

[43] ASHCROFT, N., MERMIN, N., *Solid state physics*. Saunders College, 1976.

[44] EINSTEIN, A., "Über die von der molekularkinetischen Theorie der Wärme geforderte Bewegung von in ruhenden Flüssigkeiten suspendierten Teilchen", *Annalen der Physik*, v. 322, pp. 549–560, 1905.

[45] KEENER, J., SNEYD, J., *Mathematical Physiology*. Springer, 1998.

[46] HODGKIN, A. L., HUXLEY, A. F., "A Quantitative Description of Membrane Current and Its Application to Conduction and Excitation", *The Journal of Physiology*, v. 117, pp. 500–557, 1952.

[47] CELLML, "CellML. Biology, math, data, knowledge", http://www.cellml.org.

[48] ROCHA, B., *Um modelo de acoplamento eletromecânico do tecido cardíaco através do método dos elementos finitos*, Master's Thesis, UFJF - Programa de Pós-Graduação Modelagem Computacional, 2008.

[49] TUNG, L., *A bi-domain model for describing ischemic myocardial dc potentials.*, Ph.D. Thesis, Massachusetts Institute of Technology - Dept. of Electrical Engineering and Computer Science, 1978.

[50] HENRIQUEZ, C. S., "Simulating the electrical behavior of cardiac tissue using the bidomain model", *Crit Rev. Biomed. Eng.*, v. 21, pp. 1–77, 1993.

[51] KELDERMANN, R. H., NASH, M. P., GELDERBLOM, H., WANG, V. Y., PANFILOV, A. V., "Electro-Mechanical Wavebreak in a Model of the Human Left Ventricle", *American Journal of Physiology - Heart and Circulatory Physiology*, v. 299, 2010.

[52] ALMASI, G. S., GOTTLIEB, A., "Highly parallel computing", *IBM Systems Journal*, v. 29, pp. 165–166, 1990.

[53] ALMASI, G., GOTTLIEB, A., *Highly Parallel Computing*. 2nd ed. Benjamin/Cummings, 1994.

[54] WILKINSON, B., ALLEN, M., *Parallel Programming: Techniques and Applications Using Networked Workstations and Parallel*. 2nd ed. Prentice Hall, 2004.

[55] ET. AL., L. J. K., "Trends in Multi-Core DSP Plataforms", *IEEE Signal Processing Magazine, Special Issue on Signal Processing on Platforms with Multiple Cores*, v. 26, pp. 38–49, 2009.

[56] GILBERT KALB, R. M., *Massively Parallel, Optical, and Neural Computing in the United States*. Moxley, 1998.

[57] GROOP, W., LUSK, E., "User's guide for mpich, a portable implementation of MPI." http://www.mcs.anl.gov/ lusk/oldpapers/mpich-guide/paper.html, 1994.

[58] DE CARVALHO, V. F. P., *Otimização de um cluster de alto desempenho para o uso do programa PGENESIS em simulações biologicamente plausíveis em larga-escala de sistemas neurais*, Master's Thesis, USP - Programa de Pós-Graduação em Física Aplicada à Medicina e Biologia, 2007.

[59] BOGGAN, S., PRESSEL, D. M., "GPUs: An Emerging Platform for General-Purpose Computation", Publicação Eletrônica - http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA471188, 2007.

[60] NVIDIA, *NVIDIA CUDA Programming Guide 2.0*. NVIDIA, 2008.

[61] GRAÇA, G. D., DEFOUR, D., "Implementation of float-float operators on graphics hardware". In: *In 7th conference on Real Numbers and Computers, RNC7*, pp. 23–32, 2006.

[62] BARROS, B. G., OLIVEIRA, R. S., JR., W. M., LOBOSCO, M., DOS SANTOS, R. W., "Simulations of Complex and Microscopic Models of Cardiac Electrophysiology Powered by Multi-GPU Platforms", *Computational and Mathematical Methods in Medicine*, v. 2012, pp. 13, 2012.

[63] GERDES, A. M., KELLERMAN, S. E., MOORE, J. A., MUFFLY, K. E., CLARK, L. C., REAVES, P. Y., MALEC, K. B., MCKEOWN, P. P., SCHOCKEN, D. D., "Structural remodeling of cardiac myocytes in patients with ischemic cardiomyopathy", *Circulation*, v. 86, pp. 426–30, 1992.

[64] TRACY, R. E., SANDER, G. E., "Histologically Measured Cardiomyocyte Hypertrophy Correlates with Body Height as Strongly as with Body Mass Index", *Cardiology Research and Practice*, v. 2011, pp. 1–10, 2011.

[65] EYMARD, R., GALLOUËT, T., HERBIN, R., "Finite volume methods", *Handbook of numerical analysis*, v. 7, pp. 713–1018, 2000.

[66] HARRILD, D. M., HENRIQUEZ, C. S., "A finite volume model of cardiac propagation", *Annals of biomedical engineering*, v. 25, n. 2, pp. 315–334, 1997.

[67] COUDIERE, Y., PIERRE, C., TURPAULT, R., "A 2d/3d finite volume method used to solve the bidomain equations of electrocardiology". In: *Algoritmy, Conference on Scientific Computing, Slovakia, pp. 1-10*, 2009.

[68] SUNDNES, J., *Computing the electrical activity in the heart*. Springer Verlag, 2006.

[69] STRIKWERDA, J. C., *Finite difference schemes and partial differential equations*. Society for Industrial Mathematics, 2004.

[70] RUSH, S., LARSEN, H., "A practical algorithm for solving dynamic membrane equations", *Biomedical Engineering, IEEE Transactions on*, , n. 4, pp. 389–392, 1978.

[71] ALONSO, S., BAR, M., PANFILOV, A. V., "Effects of reduced discrete coupling on filament tension in excitable media", *Chaos: An Interdisciplinary Journal of Nonlinear Science*, v. 21, n. 1, pp. 013118, 2011.

[72] ALONSO, S., BAR, M., PANFILOV, A. V., "Negative Tension of Scroll Wave Filaments and Turbulence in Three-Dimensional Excitable Media and Application in Cardiac Dynamics", *Bulletin of Mathematical Biology*, v. 10587, pp. 2–12.

[73] KEENER, J. P., "The effects of gap junctions on propagation in myocardium: a modified cable theory." *Ann N Y Acad Sci.*, v. 591, pp. 257–77, 1990.

[74] COSTA, C. M., DOS SANTOS, R. W., "Limitations of the homogenized cardiac Monodomain model for the case of low gap junctional coupling". In: *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*, pp. 228 –231, 31 2010-sept. 4 2010.

[75] BALAY, S., BUSCHELMAN, K., EIJKHOUT, V., GROPP, W. D., KAUSHIK, D., KNEPLEY, M. G., MCINNES, L. C., SMITH, B. F., ZHANG, H., "PETSc users manual", http://www.mcs.anl.gov/petsc/petsc-3.3/docs/manual.pdf, 2004.

[76] DAGUM, L., MENON, R., "OpenMP: an industry standard API for shared-memory programming", *Computational Science Engineering, IEEE*, v. 5, n. 1, pp. 46 –55, jan-mar 1998.

[77] NEIC, A., LIEBMANN, M., HOETZL, E., MITCHELL, L., VIGMOND, E. J., HAASE, G., PLANK, G., "Accelerating Cardiac Bidomain Simulations Using Graphics Processing Units", *Biomedical Engineering, IEEE Transactions on*, v. 59, n. 8, pp. 2281 –2290, aug. 2012.

[78] VIGMOND, E. J., DOS SANTOS, R. W., PRASSL, A. J., DEO, M., PLANK, G., "Solvers for the cardiac bidomain equations", *Progress in biophysics and molecular biology*, v. 96, n. 1-3, pp. 3–18, 2008.

[79] NIMMAGADDA, V. K., AKOGLU, A., HARIRI, S., MOUKABARY, T., "Cardiac simulation on multi-GPU platform", *The Journal of Supercomputing*, v. 59, pp. 1360–1378, 2012.

[80] KIRK, D. B., HWU, W. W., *Massively Parallel Processors: A Hands-on Approach*. Morgan Kaufmann, 2010.

[81] WINFREE, A. T., *When Time Breaks Down: The Three-Dimensional Dynamics of Electrochemical Waves and Cardiac Arrhythmias*. Princeton University Press, 1987.

[82] ROTH, B. J., KRASSOWSKA, W., "The induction of reentry in cardiac tissue. The missing link: How electric fields alter transmembrane potential", *Chaos: An Interdisciplinary Journal of Nonlinear Science*, v. 8, n. 1, pp. 204–220, 1998.

[83] J. H. SMITH, C. R. GREEN, N. S. P. S. R. N. J. S., "Altered patterns of gap junction distribution in ischemic heart disease. An immunohistochemical study of human myocardium using laser scanning confocal microscopy." *American Journal of Pathology*, v. 139, pp. 801–821, 1991.

[84] C. R. GREEN, N. J. S., "Distribution and role of gap junctions in normal myocardium and human ischaemic heart disease", *Histochemistry*, v. 99, pp. 105–120, 1993.

[85] SEVERS, N. J., COPPEN, S. R., DUPONT, E., YEH, H.-I., KO, Y.-S., MATSUSHITA, T., "Gap junction alterations in human cardiac disease", *Cardiovascular Research*, v. 62, n. 2, pp. 368–377, 2004.

[86] KAPRIELIAN, R. R., GUNNING, M., DUPONT, E., SHEPPARD, M. N., ROTHERY, S. M., UNDERWOOD, R., PENNELL, D. J., FOX, K., PEPPER, J., POOLE-WILSON, P. A., SEVERS, N. J., "Downregulation of Immunodetectable Connexin43 and Decreased Gap Junction Size in the Pathogenesis of Chronic Hibernation in the Human Left Ventricle", *Circulation*, v. 97, n. 7, pp. 651–660, 1998.

[87] CAMICI, P. G., WIJNS, W., BORGERS, M., DE SILVA, R., FERRARI, R., KNUUTI, J., LAMMERTSMA, A. A., LIEDTKE, A. J., PATERNOSTRO, G., VATNER, S. F., "Pathophysiological Mechanisms of Chronic Reversible

Left Ventricular Dysfunction due to Coronary Artery Disease (Hibernating Myocardium)", *Circulation*, v. 96, n. 9, pp. 3205–3214, 1997.

# APÊNDICE A - Exemplo de Código MPI

```
/* Este programa soma todas os elementos de um vetor usando paralelismo MPI
   .
 * O processo raiz envia uma parte do vetor para cada processo filho.
 * Em seguida cada processo calcula a soma da porção da matriz atribuída
 * à  eles, e os processos filhos enviam suas somas parciais de volta para
    o
 * processo raiz, que calcula a soma global.
 */

#include <stdio.h>
#include <mpi.h>

#define max_rows 100000
#define send_data_tag 2001
#define return_data_tag 2002

int array[max_rows];
int array2[max_rows];

main(int argc, char **argv) {
  long int sum, partial_sum;
  MPI_Status status;
  int my_id, root_process, ierr, i, num_rows, num_procs,
  an_id, num_rows_to_receive, avg_rows_per_process,
  sender, num_rows_received, start_row, end_row, num_rows_to_send;

  // inicia o MPI e cria os processos.

  ierr = MPI_Init(&argc, &argv);

  root_process = 0;

```

```
// descobre o ID do processo e quantos processos foram iniciados.

ierr = MPI_Comm_rank(MPI_COMM_WORLD, &my_id);
ierr = MPI_Comm_size(MPI_COMM_WORLD, &num_procs);

if(my_id == root_process) {

  // processo raiz

  printf("Entre com a quantidade de números a serem somados: ");
  scanf("%i", &num_rows);

  if(num_rows > max_rows) {
    printf("Números demais.\n");
    exit(1);
  }

  avg_rows_per_process = num_rows / num_procs;

  // inicializa um vetor

  for(i = 0; i < num_rows; i++) {
    array[i] = i + 1;
  }

  // distribui parte do vetor para cada processo filho

  for(an_id = 1; an_id < num_procs; an_id++) {
    start_row = an_id*avg_rows_per_process + 1;
    end_row   = (an_id + 1)*avg_rows_per_process;

    if((num_rows - end_row) < avg_rows_per_process)
      end_row = num_rows - 1;

    num_rows_to_send = end_row - start_row + 1;

    ierr = MPI_Send( &num_rows_to_send, 1 , MPI_INT,
        an_id, send_data_tag, MPI_COMM_WORLD);
```

```
70          ierr = MPI_Send( &array[start_row], num_rows_to_send, MPI_INT,
                an_id, send_data_tag, MPI_COMM_WORLD);
        }

        // calcula a soma dos valores da parte do vetor que lhe foi atribuído
75
        sum = 0;
        for(i = 0; i < avg_rows_per_process + 1; i++) {
          sum += array[i];
        }
80
        printf("Soma %i calculada pelo processo raiz\n", sum);

        /* coleta a soma parcial de cada processo filho e imprime,
         * depois calcula a soma global e imprime*/
85
        for(an_id = 1; an_id < num_procs; an_id++) {

          ierr = MPI_Recv( &partial_sum, 1, MPI_LONG, MPI_ANY_SOURCE,
              return_data_tag, MPI_COMM_WORLD, &status);
90
          sender = status.MPI_SOURCE;

          printf("Soma parcial %i retornada pelo processo %i\n", partial_sum,
              sender);

95        sum += partial_sum;
        }

        printf("A soma global é: %i\n", sum);
      }
100
      else {

        // processo escravo

105     ierr = MPI_Recv( &num_rows_to_receive, 1, MPI_INT,
            root_process, send_data_tag, MPI_COMM_WORLD, &status);
```

```
        ierr = MPI_Recv( &array2, num_rows_to_receive, MPI_INT,
            root_process, send_data_tag, MPI_COMM_WORLD, &status);

        num_rows_received = num_rows_to_receive;

        // calcula a soma dos valores da parte do vetor que lhe foi atribuído

        partial_sum = 0;
        for(i = 0; i < num_rows_received; i++) {
          partial_sum += array2[i];
        }

        // envia a soma parcial para o processo raiz

        ierr = MPI_Send( &partial_sum, 1, MPI_LONG, root_process,
            return_data_tag, MPI_COMM_WORLD);
      }
      ierr = MPI_Finalize();
    }
```

mpi.cpp