UNIVERSIDADE FEDERAL DE JUIZ DE FORA
FACULDADE DE ENGENHARIA
BACHARELADO EM ENGENHARIA COMPUTACIONAL

Guilherme Martins Couto

Aceleração de simulações de arritmias cardíacas através de métodos numéricos baseados no esquema *Alternating Direction Implicit* e computação paralela

Juiz de Fora
2023

Guilherme Martins Couto

Aceleração de simulações de arritmias cardíacas através de métodos numéricos baseados no esquema *Alternating Direction Implicit* e computação paralela

Trabalho de Conclusão de Curso apresentado à Faculdade de Engenharia da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do grau de Bacharel em Engenharia Computacional.

Orientador: Prof. Dr. Rodrigo Weber dos Santos

Coorientador: Prof. Dr. Joventino de Oliveira Campos

Juiz de Fora

2023

Ficha catalográfica elaborada através do Modelo Latex do CDC da UFJF
com os dados fornecidos pelo(a) autor(a)

Martins Couto, Guilherme.

Aceleração de simulações de arritmias cardíacas através de métodos numéricos baseados no esquema *Alternating Direction Implicit* e computação paralela / Guilherme Martins Couto. – 2023.

34 f. : il.

Orientador: Rodrigo Weber dos Santos

Coorientador: Joventino de Oliveira Campos

Trabalho de Conclusão de Curso (Graduação) – Universidade Federal de Juiz de Fora, Faculdade de Engenharia. Bacharelado em Engenharia Computacional, 2023.

1. Eletrofisiologia Computacional. 2. Simulação de Arritmias. 3. Computação de Alto Desempenho. I. Weber dos Santos, Rodrigo, orient. II. de Oliveira Campos, Joventino, coorient. III. Título.

Guilherme Martins Couto

Aceleração de simulações de arritmias cardíacas através de métodos numéricos baseados no esquema *Alternating Direction Implicit* e computação paralela

Trabalho de Conclusão de Curso apresentado à Faculdade de Engenharia da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do grau de Bacharel em Engenharia Computacional.

Aprovado em 13 de dezembro de 2023

BANCA EXAMINADORA

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

Prof. Dr. Joventino de Oliveira Campos - Coorientador
Universidade Federal de Juiz de Fora

Prof. Dr. Bernardo Martins Rocha
Universidade Federal de Juiz de Fora

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

AGRADECIMENTOS

Recentemente foi meu aniversário e me senti a pessoa mais amada do mundo. Estive cercado das pessoas que eu amo e que me fazem bem. Nada me deixa mais feliz do que esses momentos em que é possível estar junto. Sei que a vida com vocês será sempre repleta de amor.

Gostaria de começar agradecendo a minha família, em especial aos meus pais, Fátima e Gustavo, por toda a confiança que depositaram em mim desde que me entendo por gente e pela convicção inabalável de que a educação é o caminho. Foi em nossa casa e com vocês que aprendi a sonhar. Muito obrigado, eu amo demais vocês.

Minha irmã Lívia é a pessoa que mais amo no mundo e com quem moro em Juiz de Fora há quatro anos. Obrigado, Li, pelo carinho mais sincero e puro que já tive. Sem você, eu nada seria.

Sou um devoto das minhas amizades. Encho-me de otimismo quando penso que o futuro em que viverei está sendo pensado e construído por pessoas tão brilhantes e inspiradoras. Que sorte é ter vocês por perto. Muito obrigado, amigos, por fazerem de quem eu sou. Carrego e cultivo um pouco de cada um nos meus pensamentos e no meu coração.

A Universidade me fez conhecer pessoas incríveis e me fez ter experiências transformadoras que eu não poderia deixar de citar. Agradeço ao pessoal que esteve comigo no Garra - Cursinho Popular, foram anos muito bons. Aos meus colegas de Fisiocomp, também deixo meu muito obrigado pela rotina, pelas trocas e pelas risadas. Por fim, gostaria de agradecer pelo apoio e dedicação dos meu orientadores Rodrigo e Joventino. Sempre que estou com vocês, aprendo muito e fico mais motivado percebendo o quanto ainda há para aprender.

Cota zero

Stop.
A vida parou
ou foi o automóvel?

(Carlos Drummond de Andrade, 1930, *Alguma Poesia*)

Só eu posso pensar se Deus existe
Só eu posso chorar quando estou triste
Eu cá com meus botões de carne e osso
Eu falo e ouço
Eu penso e posso

Eu posso decidir se vivo ou morro
Porque sou vivo, vivo pra cachorro
E sei
Que cérebro eletrônico nenhum me dá socorro
Em meu caminho inevitável para a morte

(Gilberto Gil, 1969, Trecho da música *Cérebro Eletrônico*)

RESUMO

A arritmia cardíaca é uma condição relacionada a alterações no ritmo dos batimentos do coração. Ela acontece quando os estímulos elétricos não se propagam corretamente pelo miocárdio e pode levar à morte súbita. A modelagem computacional pode ser uma importante aliada na identificação de arritmias graves, além de possibilitar testes virtuais personalizados para o paciente. Nesse contexto, o monodomínio, modelo matemático que descreve a propagação elétrica no tecido cardíaco, é amplamente utilizado. Resolvê-lo, no entanto, ainda é uma tarefa complexa e demorada. Este trabalho apresenta estratégias numéricas e computacionais para acelerar as simulações de arritmias ventriculares. São comparados três métodos numéricos baseados no esquema *Alternating Direction Implicit* (ADI): *Operator Splitting* ADI, *Second-Order Semi-Implicit* ADI e *Mixed-Order Semi-Implicit* ADI. As implementações são paralelizadas usando a biblioteca *OpenMP*, e quatro métricas são analisadas: a janela de vulnerabilidade, usada para avaliar o risco de arritmia no coração; a velocidade de propagação do estímulo; o erro numérico; e o tempo de execução. Além disso, apresenta-se uma abordagem alternativa com uma implementação paralela para GPUs usando a biblioteca *CUDA* que é comparada ao simulador *MonoAlg3D*.

Palavras-chave: eletrofisiologia computacional; simulação de arritmias; computação de alto desempenho.

ABSTRACT

Cardiac arrhythmia is a condition related to changes in the rhythm of the heartbeat. It happens when electrical stimuli do not propagate correctly through the myocardium and can lead to sudden death. Computational modeling can be an important ally in identifying serious arrhythmias, in addition to enabling personalized virtual tests for the patient. In this context, the monodomain, a mathematical model that describes electrical propagation in cardiac tissue, is widely used. Solving it, however, is still a complex and time-consuming task. This work presents numerical and computational strategies to accelerate simulations of ventricular arrhythmias. Three numerical methods based on the Alternating Direction Implicit (ADI) scheme are compared: Operator Splitting ADI, Second-Order Semi-Implicit ADI and Mixed-Order Semi-Implicit ADI. The implementations are parallelized using the OpenMP library, and four metrics are analyzed: the vulnerability window, used to assess the risk of heart arrhythmia; the speed of propagation of the stimulus; the numerical error; and execution time. Furthermore, an alternative approach is presented with a parallel implementation for GPUs using the CUDA library that is compared to the MonoAlg3D simulator.

Keywords: computational electrophysiology; simulation of arrhythmias; high performance computing.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Segundo o relatório da *American Heart Association*, em 2019, as doenças cardiovasculares (DCVs) foram a principal causa de mortes no mundo, sendo responsáveis por 27% do total de óbitos (1). Em diversos países, dentre as DCVs, a mais letal é a morte súbita cardíaca (MSC) (1), quando há uma parada inesperada da função do órgão. A fibrilação ventricular é o principal tipo de arritmia associado a MSC (2). Isso mostra a importância de estudar essa condição e de aprimorar as práticas clínicas atuais, a fim de salvar vidas por meio de uma estratificação mais precisa dos riscos de cada pessoa. Contudo, atualmente, para se entender a fundo e escolher o melhor tratamento para um quadro de arritmia, ainda são necessários exames altamente invasivos, sendo, o mais comum, o estudo eletrofisiológico.

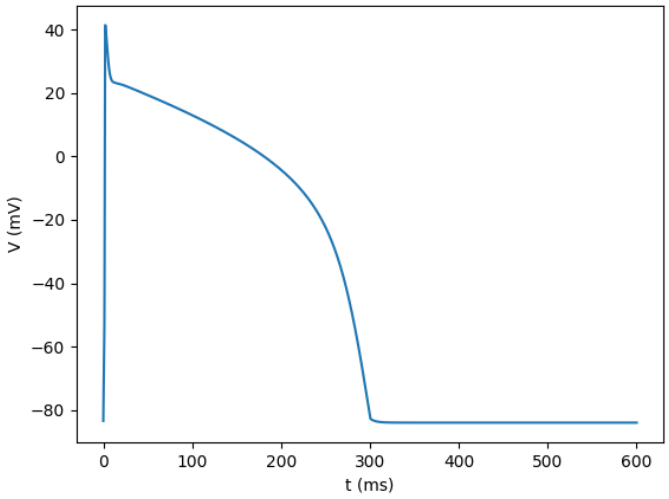
Com o avanço das tecnologias médicas e com a expansão da medicina de precisão na cardiologia, diagnósticos e tratamentos estão passando a ser mais detalhados e personalizados para as especificidades de cada paciente. A modelagem computacional, nesse cenário, já é uma importante aliada. Com o passar do tempo, a perspectiva é de que os modelos, construídos e ajustados a partir de dados de exames pouco invasivos, como eletrocardiograma e imagens de ressonância magnética, sejam ainda mais utilizados para simular o comportamento eletrofisiológico do coração. Naturalmente, algumas barreiras surgem entre o desenvolvimento e a aplicação efetiva de uma nova ferramenta. Hoje, um dos mais relevantes desafios a ser superado é o longo tempo gasto para se realizar uma simulação completa (3).

O coração é um órgão vital que tem a função de bombear sangue para o resto do corpo, seguindo um determinado ritmo, para que não falte oxigênio em nenhuma parte. Para realizar essa tarefa, o músculo cardíaco, formado por cardiomiócitos, precisa se contrair, diminuindo o volume da cavidade dos ventrículos e expulsando o sangue que tem dentro. Essa mecânica só acontece, pois há uma estrutura, na região do átrio direito, chamada nó sinoatrial (NSA), que gera um sinal elétrico que estimula, em uma pessoa saudável, a contração dos átrios, em um primeiro momento, e, em seguida, dos ventrículos. Ou seja, o NSA é o responsável por manter a frequência dos batimentos cardíacos. Quando, por algum motivo, o coração perde essa sincronia, ocorre uma arritmia que pode ser letal.

A membrana dos cardiomiócitos tem canais que permitem a passagem de íons do meio intracelular para o extracelular e vice-versa. Sem interferência, ela se mantém em um potencial de repouso. Entretanto, se houver um estímulo, ocorre uma rápida despolarização e gera-se um potencial de ação, que faz com que as células se contraiam. Um sistema de equações diferenciais ordinárias (EDOs) é capaz de representar o comportamento elétrico da célula, uma vez que a condutância das estruturas proteicas da membrana varia com a voltagem e com o tempo.

A Figura 1 mostra o potencial de ação simulado de uma célula do ventrículo, que, ao receber um estímulo, sai de seu potencial de repouso negativo para um valor positivo, fase chamada de despolarização. A célula retorna, então, ao seu potencial de repouso novamente, de forma mais lenta, na chamada repolarização.

Figura 1 - Potencial de ação de uma célula do ventrículo



Fonte: Elaborada pelo autor (2023).

As células do coração ligam-se umas às outras através de junções *gap*. Essa conexão forma o tecido cardíaco e permite a propagação elétrica. Esta, por conta da distribuição das junções e do formato de bastonete dos cardiomiócitos, acontece de maneira anisotrópica (4). O modelo do bidomínio leva em consideração os espaços intracelular e extracelular para modelar esse fenômeno (5). Isso, porém, torna-o bastante complexo, já que é preciso trabalhar com dois tensores de condutividade e duas equações diferenciais parciais (EDPs).

A fim de simplificar essa dinâmica microscópica, aplica-se técnicas de homogeneização para suavizar as equações que descrevem a propagação elétrica. Portanto, quando os tensores de condutividade intracelular e extracelular são proporcionais, ou quando a resistência extracelular pode ser desprezada, é possível considerar apenas o espaço intracelular no problema, que passa a ser descrito com apenas uma EDP pelo modelo do monodomínio.

Quando, por algum motivo, um estímulo passa pela mesma área do tecido mais de uma vez, diz-se que ocorreu reentrada, que está associada à arritmia, uma vez que aconteceu uma excitação prematura. A onda de propagação, nessa condição, forma uma espiral, que pode ser sustentável ou não (6). Geralmente, quando há um infarto do miocárdio, que é o tecido cardíaco, aparecem cicatrizes, chamadas de fibroses, nas áreas lesionadas. Elas afetam a propagação do estímulo elétrico e aumentam significativamente

os riscos de arritmia (7).

Este trabalho, portanto, tem como objetivo acelerar as simulações de arritmias ventriculares através de estratégias numéricas e técnicas computacionais. Para isso, inicialmente, expõe-se a formulação matemática do problema que envolve o acoplamento do modelo de FitzHugh-Nagumo (FHN) (8), adaptado para reproduzir o comportamento de células do coração, ao modelo do monodomínio. O resultado é um sistema que inclui uma EDP parabólica de reação-difusão e duas EDOs não lineares. Em seguida, *Operator Splitting* ADI (OS_{ADI}), *Second-Order Semi-Implicit* ADI (SSI_{ADI}) e *Mixed-Order Semi-Implicit* ADI (MOSI_{ADI}), três métodos numéricos baseados no esquema ADI, são apresentados. Eles, paralelizados com a biblioteca *OpenMP*, são comparados levando em conta quatro métricas: a janela de vulnerabilidade, que captura a suscetibilidade do coração ao surgimento de arritmias (9), a velocidade de propagação do estímulo, o erro numérico e o tempo de execução. Por fim, uma abordagem de paralelização do OS_{ADI} utilizando *CUDA* é proposta e comparada com o *MonoAlg3D*, um simulador já consolidado (10).

2 FORMULAÇÃO MATEMÁTICA

O comportamento eletrofisiológico do tecido cardíaco é descrito por um sistema de equações diferenciais que é fruto do acoplamento do modelo celular de FitzHugh-Nagumo, com uma adaptação, ao modelo do monodomínio.

2.1 MODELO DO MONODOMÍNIO

O monodomínio descreve, através de uma equação de reação-difusão, a propagação do estímulo elétrico pelo miocárido. Isso ocorre, pois as células que o formam estão conectadas pelas junções *gap*, que permitem propagação elétrica entre células vizinhas. Neste trabalho, como o foco é o estudo dos métodos numéricos, considera-se, por simplificação, um domínio isotrópico. Assim, tem-se a seguinte formulação:

$$\beta \left[C_m \frac{\partial V}{\partial t} + I_{ion}(V, \boldsymbol{\eta}) \right] = \nabla \cdot \boldsymbol{\sigma} \nabla V + I_{est}, \tag{2.1}$$

$$\frac{\partial \boldsymbol{\eta}}{\partial t} = \mathbf{f}(V, \boldsymbol{\eta}), \tag{2.2}$$

$$\mathbf{n} \cdot \boldsymbol{\sigma} \nabla V = 0, \quad \text{em } \partial\Omega, \tag{2.3}$$

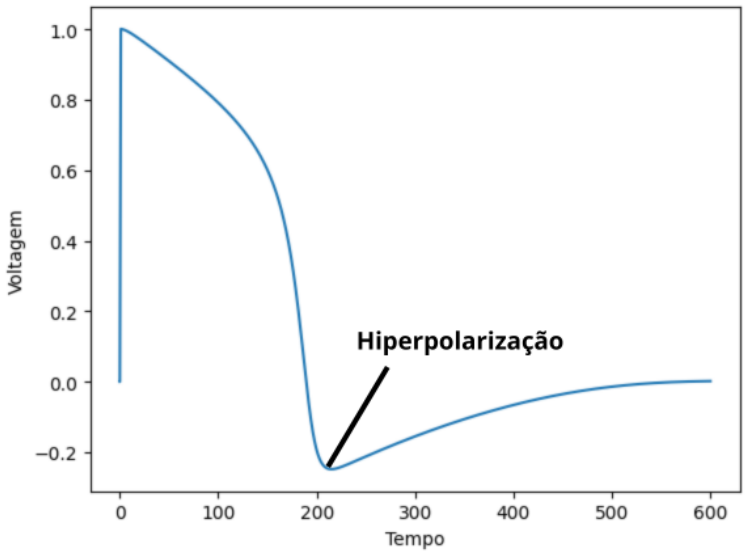
onde $\boldsymbol{\sigma}$ é o tensor de condutividade, β é a razão superfície-volume das células cardíacas, C_m é a capacitância da membrana por unidade de área, V é o potencial transmembrânico, I_{ion} é a densidade total da corrente iônica, $\boldsymbol{\eta}$ é um vetor com as variáveis de estado, I_{est} é a densidade da corrente de estímulo e $\mathbf{n}$ é o vetor unitário normal ao contorno. A Equação (2.2) representa o sistema de EDOs, que descreve a dinâmica dos cardiomiócitos (11). Já a Equação (2.3), prescreve a condição de contorno de Neumann homogênea. Ela isola o domínio, garantindo que não haja fluxo de estímulo elétrico para fora e nem para dentro.

2.2 MODELO DE FITZHUGH-NAGUMO ADAPTADO

Com apenas duas equações, o modelo de FitzHugh-Nagumo consegue representar o comportamento do potencial de ação gerado em neurônios (8). São duas variáveis, sendo uma chamada de excitação, ou rápida (v), e a outra, de recuperação, ou lenta (w).

A onda do potencial formada pelo FHN tem uma fase de hiperpolarização durante o período refratário, como mostra a Figura 2. Se o modelo fosse usado sem alterações, não seria capaz de representar as propriedades de recuperação das células do ventrículo cardíaco, que não têm essa característica (Figura 1). Isso afetaria consideravelmente os padrões de reentrada e prejudicaria as simulações.

Figura 2 - Potencial de ação gerado pelo FHN tradicional



Fonte: Elaborada pelo autor (2023).

Para contornar esse problema, em (12, 13) é proposta uma adaptação no FHN de modo que a hiperpolarização é suprimida. Assim, tem-se as seguintes equações:

$$\frac{\partial v}{\partial t} = Av\left(1 - \frac{v}{v_{th}}\right)\left(1 - \frac{v}{v_p}\right) + Bvw, \tag{2.4}$$

$$\frac{\partial w}{\partial t} = C\left(\frac{v}{v_p} - Dw\right), \tag{2.5}$$

onde v_{th} é o limite de crescimento, v_p é o pico do potencial e A, B, C e D são constantes positivas.

3 MÉTODOS NUMÉRICOS

Como as simulações foram feitas para um domínio quadrado com a mesma discretização espacial ($\Delta x = \Delta y$), os três esquemas serão apresentados levando isso em consideração. O tamanho da discretização temporal, por sua vez, é denotada por Δt.

Todos os métodos são de diferenças finitas e baseados no conceito de *Alternating Direction Implicit*. Esta estratégia propõe que, em um mesmo passo de tempo, a EDP seja fragmentada e suas derivadas calculadas de maneira implícita em cada direção separadamente.

Esse tipo de abordagem leva à necessidade de se resolver um sistema linear, que pode ser escrito como $\mathbf{MX} = \mathbf{R}$, em que $\mathbf{M}$ é uma matriz com os coeficientes vindos da aproximação da derivada de segunda ordem, $\mathbf{X}$ é uma matriz com as incógnitas do instante n e $\mathbf{R}$ é uma matriz com os respectivos lados direito do sistema no instante n. A grande vantagem do ADI está justamente no formato de $\mathbf{M}$, que, para cada direção, será tridiagonal, conforme a formulação abaixo:

$$\begin{bmatrix} 1+2\boldsymbol{\phi} & -2\boldsymbol{\phi} & \dots & \dots & 0 \\ -\boldsymbol{\phi} & 1+2\boldsymbol{\phi} & -\boldsymbol{\phi} & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & -\boldsymbol{\phi} & 1+2\boldsymbol{\phi} & -\boldsymbol{\phi} \\ 0 & \dots & \dots & -2\boldsymbol{\phi} & 1+2\boldsymbol{\phi} \end{bmatrix} \mathbf{X}^n = \mathbf{R}^n \tag{3.1}$$

com $\boldsymbol{\phi}$ sendo um parâmetro que varia entre os métodos e N o número total de pontos discretizados em uma determinada direção. A primeira e a última linhas da matriz $\mathbf{M}$ da Equação (3.1) são distintas, por conta da restrição imposta pelo contorno do dominío. Próximo às bordas, para não se utilizar um ponto de fora dos limites, seu valor é aproximado.

Com isso, torna-se possível utilizar o simples e eficiente algoritmo de Thomas (14), uma simplificação da eliminação gaussiana para matrizes tridiagonais, com complexidade $O(N)$. Em um sistema com múltiplos lados direito, como o da Equação (3.1), suas chamadas podem ser executadas em paralelo para resolvê-los individualmente. Em métodos que não fazem a separação da EDP, a matriz $\mathbf{M}$ fica mais complexa, e, como consequência, a resolução fica mais custosa.

3.1 *OPERATOR SPLITTING ALTERNATING DIRECTION IMPLICIT*

No OS_{ADI}, aplica-se um *operator splitting*, fazendo com que a reação e a difusão da Equação (2.1) sejam completamente separadas (15). Assim, a solução do monodomínio passa a depender da resolução de dois problemas:

- Reação: sistema de EDOs não lineares

$$\frac{\partial V}{\partial t} = \frac{1}{C_m}[-I_{ion}(V, \boldsymbol{\eta}) + I_{est}], \tag{3.2}$$

$$\frac{\partial \boldsymbol{\eta}}{\partial t} = \mathbf{f}(V, \boldsymbol{\eta}), \tag{3.3}$$

- Difusão: EDP parabólica

$$\beta C_m \frac{\partial V}{\partial t} = \nabla \cdot \boldsymbol{\sigma} \nabla V. \tag{3.4}$$

Neste trabalho, utiliza-se o método de Euler explícito para resolver o sistema de EDOs. Dessa forma, quando discretizadas, as Equações (3.2) e (3.3) ficam, respectivamente, como

$$\frac{V_{i,j}^{n+1} - V_{i,j}^{n}}{\Delta t} = \frac{1}{C_m}[-I_{ion}(V_{i,j}^{n}, \boldsymbol{\eta_{i,j}^{n}}) + I_{est}], \tag{3.5}$$

$$\frac{\boldsymbol{\eta}_{i,j}^{n+1} - \boldsymbol{\eta}_{i,j}^{n}}{\Delta t} = \mathbf{f}(V_{i,j}^{n}, \boldsymbol{\eta_{i,j}^{n}}). \tag{3.6}$$

A solução da Equação (3.4) é encontrada usando o OS_{ADI}, que discretiza a equação parabólica usando diferença progressiva no tempo e diferença central no espaço. Em seguida, a equação resultante é dividida em duas, onde, na primeira, as incógnitas na direção x aparecem de forma implícita, enquanto, na segunda equação, as incógnitas na direção y aparecem de forma implícita. A solução é encontrada através da resolução dos sistemas lineares:

$$-\boldsymbol{\phi}_{OS} V_{i-1,j}^{**} + (1 + 2\boldsymbol{\phi}_{OS}) V_{i,j}^{**} - \boldsymbol{\phi}_{OS} V_{i+1,j}^{**} = V_{i,j}^{*}, \tag{3.7}$$

$$-\boldsymbol{\phi}_{OS} V_{i,j-1}^{n+1} + (1 + 2\boldsymbol{\phi}_{OS}) V_{i,j}^{n+1} - \boldsymbol{\phi}_{OS} V_{i,j+1}^{n+1} = V_{i,j}^{**}, \tag{3.8}$$

em que $\boldsymbol{\phi}_{OS} = \frac{\boldsymbol{\sigma}\Delta t}{\beta C_m \Delta x^2}$ e V^* é a solução da Equação (3.5) para o passo de tempo n. Na primeira etapa, considera-se difusão apenas em x, Equação (3.7), na segunda, Equação (3.8), somente em y, como mostra a Figura 3. O OS_{ADI}, apesar de ter uma convergência de segunda ordem no espaço, no tempo, é de primeira ordem (10).

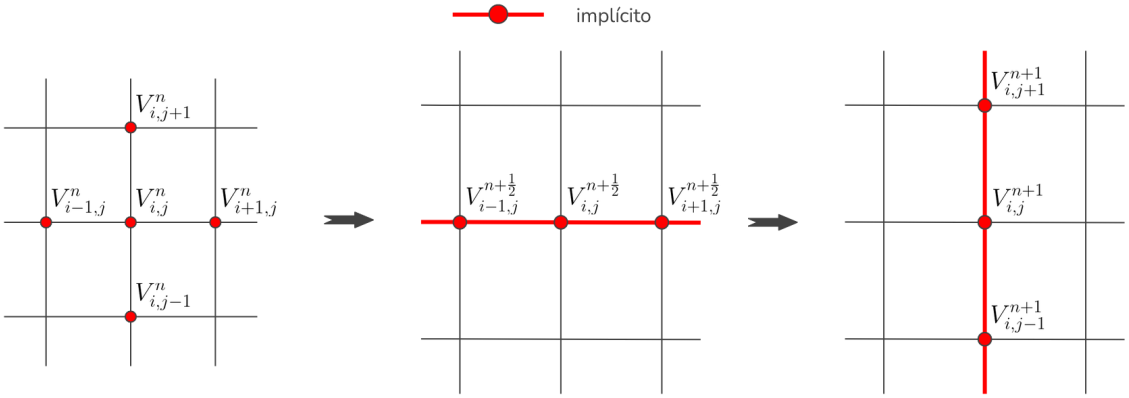
3.2 *SECOND-ORDER SEMI-IMPLICIT ALTERNATING DIRECTION IMPLICIT*

Para o SSI_{ADI}, proposto por (16), inicialmente, calcula-se, de forma explícita, uma predição V^*, a partir da Equação (2.1), e outra $\boldsymbol{\eta}^*$, vinda da Equação (2.2).

$$V^* = V^n + \frac{\Delta t}{2\beta C_m}(\nabla \cdot \boldsymbol{\sigma} \nabla V^n) + \frac{\Delta t}{2}\mathcal{R}(V^n, \boldsymbol{\eta}^n), \tag{3.9}$$

$$\boldsymbol{\eta}^* = \boldsymbol{\eta}^n + \frac{\Delta t}{2}\mathbf{f}(V^n, \boldsymbol{\eta}^n), \tag{3.10}$$

Figura 3 - Passo a passo do cálculo da difusão no OS_{ADI} mostrando em qual direção as incógnitas aparecem na forma implícita em cada etapa



Fonte: Elaborada pelo autor (2023).

considerando $\mathcal{R}$ como o termo reativo da Equação (2.1).

Assim, com os resultados das Equações (3.9) e (3.10), a solução da Equação (2.1) é dada por

$$-\boldsymbol{\phi}_{SSI}V^{n+\frac{1}{2}}_{i-1,j} + (1+2\boldsymbol{\phi}_{SSI})V^{n+\frac{1}{2}}_{i,j} - \boldsymbol{\phi}_{SSI}V^{n+\frac{1}{2}}_{i+1,j} = F^n_i, \tag{3.11}$$

$$-\boldsymbol{\phi}_{SSI}V^{n+1}_{i,j-1} + (1+2\boldsymbol{\phi}_{SSI})V^{n+1}_{i,j} - \boldsymbol{\phi}_{SSI}V^{n+1}_{i,j+1} = F^{n+\frac{1}{2}}_j, \tag{3.12}$$

onde $\boldsymbol{\phi}_{SSI} = \frac{\boldsymbol{\sigma}\Delta t}{2\beta C_m \Delta x^2}$, e os lados direito, F^n_i e $F^{n+\frac{1}{2}}_j$, representam, respectivamente,

$$F^n_i = \boldsymbol{\phi}_{SSI}V^n_{i,j-1} + (1-2\boldsymbol{\phi}_{SSI})V^n_{i,j} + \boldsymbol{\phi}_{SSI}V^n_{i,j+1} + \frac{\Delta t}{2}\mathcal{R}(V^*, \boldsymbol{\eta}^*), \tag{3.13}$$

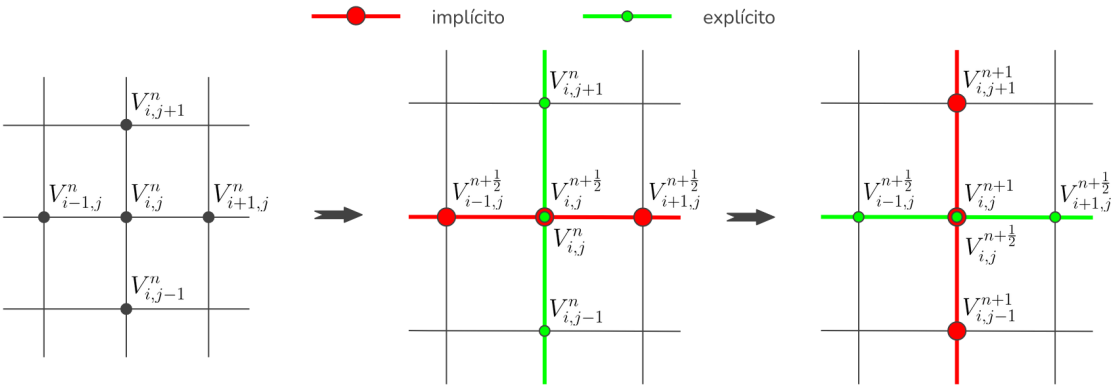
$$F^{n+\frac{1}{2}}_j = \boldsymbol{\phi}_{SSI}V^{n+\frac{1}{2}}_{i-1,j} + (1-2\boldsymbol{\phi}_{SSI})V^{n+\frac{1}{2}}_{i,j} + \boldsymbol{\phi}_{SSI}V^{n+\frac{1}{2}}_{i+1,j} + \frac{\Delta t}{2}\mathcal{R}(V^*, \boldsymbol{\eta}^*). \tag{3.14}$$

Diferentemente do OS_{ADI}, no SSI_{ADI}, a primeira metade da resolução da EDP, Equações (3.11) e (3.13), envolve tanto as incógnitas da direção x, na forma implícita, quanto as de y, na forma explícita. Em seguida, nas Equações (3.12) e (3.14), ocorre o contrário, conforme mostra a Figura 4, com as incógnitas da direção y na forma implícita e, as de x, na forma explícita.

Para garantir a convergência de segunda ordem no tempo (16), a solução das variáveis de estado, Equação (2.2), é calculada com o método Runge-Kutta de segunda ordem (17), utilizando as predições encontradas nas Equações (3.9) e (3.10).

$$\boldsymbol{\eta}^{n+1} = \boldsymbol{\eta}^n + \Delta t \mathbf{f}(V^*, \boldsymbol{\eta}^*). \tag{3.15}$$

Figura 4 - Passo a passo do cálculo da difusão no SSI_{ADI} mostrando em qual direção as incógnitas aparecem na forma implícita e explícita em cada etapa



Fonte: Elaborada pelo autor (2023).

3.3 *MIXED-ORDER SEMI-IMPLICIT ALTERNATING DIRECTION IMPLICIT*

O MOSI_{ADI} tem apenas uma diferença com relação ao SSI_{ADI} que está na atualização das variáveis de estado. Neste método, a Equação (2.2) é resolvida usando Euler explícito, sem predições.

$$\boldsymbol{\eta}^{n+1} = \boldsymbol{\eta}^n + \Delta t \mathbf{f}(V^n, \boldsymbol{\eta}^n). \tag{3.16}$$

Logo, V segue evoluindo com um método de segunda ordem, enquanto $\boldsymbol{\eta}$, com um de primeira. Isso permite com que a Equação (3.16) seja calculada antes ou, até mesmo, durante a resolução das Equações (3.9) e (3.10).

4 PARALELIZAÇÃO EM GPU

A programação paralela em GPUs é utilizada para diminuir o tempo necessário para a resolução de um problema. Se bem feita, traz ganhos relevantes quando comparada à execução sequencial. Para isso, deve-se levar em consideração, especialmente, aspectos de *hardware*, de escalabilidade, de estrutura de dados, de balanceamento de carga e de acesso à memória. Sem essa devida atenção, os efeitos podem ser contrários ao que se espera.

Na implementação do OS_{ADI} (GPU), para o caso em questão, de um domínio quadrado de $N \times N$, todas as estruturas principais e auxiliares de duas dimensões utilizadas para armazenar as variáveis do modelo matemático foram linearizadas e transformadas em vetor. Essa abordagem, além de facilitar os processos de cópia do *host* (CPU) para o *device* (GPU) e vice-versa, também contribui para a eficiência dos acessos dentro dos *kernels*, termo relacionado às funções do código que são chamadas pela CPU, mas executadas pela GPU.

Nesse tipo de método discreto, não é possível paralelizar as iterações temporais, pois, para se calcular uma variável no instante $n+1$, é preciso saber seu valor no instante n. No domínio espacial, por sua vez, onde não há essa dependência, é viável fazer chamadas em paralelo.

A estratégia, portanto, consiste em quatro etapas principais:

- resolver o sistema de EDOs (Equações (3.5) e (3.6));
- resolver a primeira parte do ADI (Equação (3.7));
- reorganizar o vetor com os valores de V;
- resolver a segunda parte do ADI (Equação (3.8)).

No *kernel* responsável pela resolução do sistema de EDOs, Algoritmo 1, cada *thread* acessa um elemento do domínio, totalizando N^2 chamadas.

Algoritmo 1: *kernel* `resolveEDOs`

```
1 i ← blockDim.x * blockIdx.x + threadIdx.x
2 se i < N * N então
3     Va ← V[i]
4     Wa ← W[i]
5     V[i] ← Va + Δt * (LadoDireitoV(Va, Wa) + Estímulo(i, N))
6     W[i] ← Wa + Δt * LadoDireitoW(Va, Wa)
```

Para se realizar as etapas que envolvem o ADI, utiliza-se a implementação do algoritmo de Thomas para resolver sistemas tridiagonais, proposta em (18), para problemas com múltiplos lados direito. A abordagem ainda inclui uma pré-fatorização que é feita, apenas uma vez, no *host*.

São N *threads* para N chamadas do Thomas em cada fase do ADI. O acesso à memória global, para ser otimizado, é feito de maneira coalescida, com as *threads* de um mesmo *warp* acessando valores em posições próximas. Portanto, o vetor de entrada da função deve estar devidamente organizado para isso.

O Algoritmo 2 descreve o funcionamento do *kernel* que implementa o Thomas para GPU. Os vetores *la*, *lb* e *lc* representam, respectivamente, a diagonal inferior, principal e superior, já pré-fatorizadas, da matriz do sistema linear. O vetor d, por sua vez, armazena os múltiplos lados direito desse mesmo sistema. Ao final da execução, d é sobrescrito e passa a ter os valores da solução.

Algoritmo 2: *kernel* `resolveThomas`

```
1  linhaAtual ← blockDim.x * blockIdx.x + threadIdx.x
2  linhaAnterior, linhaSeguinte
3  i ← 0
4  se linhaAtual < N então
5      d[linhaAtual] ← d[linhaAtual] / lb[i]
6      para i ← 1 até N − 1 faça
7          linhaAnterior ← linhaAtual
8          linhaAtual ← linhaAtual + N
9          d[linhaAtual] ← (d[linhaAtual] - la[i] * d[linhaAnterior]) / lb[i]
10     d[linhaAtual] ← d[linhaAtual]
11     para i ← N − 2 até 0 faça
12         linhaSeguinte ← linhaAtual
13         linhaAtual ← linhaAtual - N
14         d[linhaAtual] ← d[linhaAtual] - lc[i] * d[linhaSeguinte]
```

O terceiro *kernel* da implementação do OS_{ADI} (GPU), Algoritmo 3, é simples e direto. Sua função é pegar a saída da primeira parte do ADI e reorganizá-la para ser uma entrada válida, pensando no acesso coalescido, para as chamadas em paralelo do Thomas da segunda parte.

Algoritmo 3: *kernel* `mapeamento`

```
1 k ← blockDim.x * blockIdx.x + threadIdx.x
2 se k < N * N então
3     i ← k / N
4     j ← k % N
5     saída[j + i * N] ← entrada[i + j * N]
```

5 SIMULAÇÃO COMPUTACIONAL

Os métodos foram implementados em *C* e, para comparação entre eles, paralelizados com *OpenMP*. Além disso, para a comparação da paralelização em GPU, o OS_{ADI} foi paralelizado com *CUDA*. Cada configuração de simulação foi executada cinco vezes com precisão dupla para os valores reais. As análises de dados foram feitas com *Python*. O computador utilizado para as simulações estava equipado com um processador Intel® Core™ i5-12600 e uma placa de vídeo Nvidia RTX 4070.

O conjunto de parâmetros utilizados para o modelo matemático é apresentado na Tabela 1.

Tabela 1 – Parâmetros do modelo matemático

Parâmetro	Valor
A	$1.5\ \Omega^{-1}\text{cm}^{-2}$
B	$4.4\Omega^{-1}\text{cm}^{-1}$
C	0.012
D	1
v_{th}	13mV
v_p	100mV
$\boldsymbol{\sigma}$	$1.2 \cdot 10^{-3}\Omega^{-1}\text{cm}^{-1}$
β	10^3cm^{-1}
C_m	10^{-3}mFcm^{-2}

Fonte: Elaborada pelo autor (2023).

Para estratificar ainda mais as análises, considerou-se a resolução do monodomínio dividida em duas partes (10). Assim, para o OS_{ADI}, as Equações (3.5) e (3.6) são referentes à primeira parte, e as Equações (3.7) e (3.8), à segunda. No SSI_{ADI}, por outro lado, a primeira parte é mais extensa, com as Equações (3.9), (3.10) e (3.15). Na segunda, as Equações (3.11) e (3.12) é que são resolvidas. Por fim, no MOSI_{ADI}, a diferença para o SSI_{ADI} está apenas na primeira parte, em que a Equação (3.16) entra no lugar da Equação (3.15).

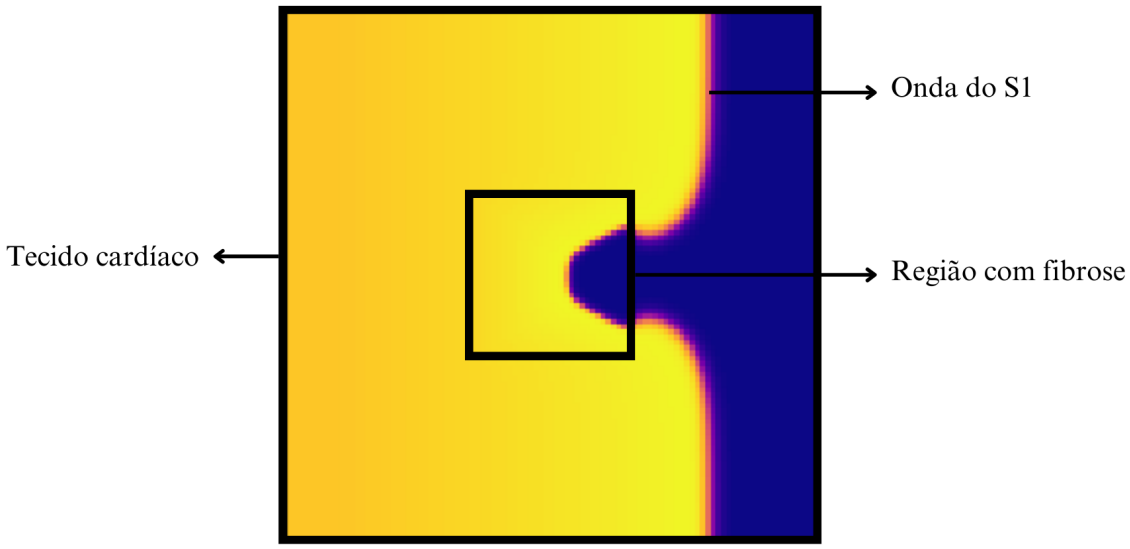
5.1 COMPARAÇÃO ENTRE OS MÉTODOS NUMÉRICOS

As simulações computacionais para comparação entre os métodos numéricos consideraram um tecido cardíaco de 2 cm $\times$2 cm com uma área quadrada fibrótica entre $[0.7, 0.7] \times [1.3, 1.3]$. Nela, a condutividade vale 20% do valor original, $\boldsymbol{\sigma}_{fib} = 0.2\boldsymbol{\sigma}$.

O primeiro estímulo (S1) começa em $t_{S1} = 0$ ms, dura 2 ms, é aplicado na região $[0, 0.2] \times [0, 2]$ e propaga-se da esquerda para direita. Sua velocidade é calculada através do tempo gasto para chegar ao extremo oposto.

A Figura 5 mostra o tecido cardíaco, a onda do S1 e a região fibrótica.

Figura 5 - Domínio computacional com descrição do tecido, onda do S1 e fibrose



Fonte: Elaborada pelo autor (2023).

A discretização espacial foi $\Delta x = \Delta y = 0.005$ cm, e o tempo total de simulação foi $T = 320$ ms. Os métodos numéricos foram avaliados com diferentes passos de tempo, de $\Delta t = 0.01$ ms até $\Delta t = 0.16$ ms.

Para induzir uma arritmia e poder avaliar a janela de vulnerabilidade do coração, em diferentes tempos, um segundo estímulo (S2) é aplicado. Sua duração também é de 2 ms e ocorre em $[0, 1] \times [0, 1]$.

Como não se tem uma solução analítica para o monodomínio, o resultado de referência utilizado para as comparações é proveniente do SSI_{ADI} com um $\Delta t = 0.001$ ms. O erro relativo em porcentagem (ERP) serviu como métrica para avaliar a precisão dos métodos.

$$ERP = 100 \times \frac{\sqrt{\sum_{i=1}^{N_x} \sum_{j=1}^{N_y} \left(v_{i,j} - \hat{v}_{i,j}\right)^2}}{\sqrt{\sum_{i=1}^{N_x} \sum_{j=1}^{N_y} \hat{v}_{i,j}^2}}, \tag{5.1}$$

em que $\hat{v}$ é a referência, e N_x, N_y são o número de discretizações no espaço em cada direção.

5.2 COMPARAÇÃO DA PARALELIZAÇÃO EM GPU

Para a comparação com o *MonoAlg3D*, utilizou-se, como base, as mesmas configurações descritas na Seção 5.1, mas com algumas mudanças. O domínio permaneceu o mesmo mostrado na Figura 5, porém sem a região fibrótica. Além disso, como o foco era apenas a

análise do tempo gasto por cada parte, o erro numérico e a janela de vulnerabilidade não foram medidos. A arritmia, portanto, foi induzida sempre com o mesmo $t_{S2} = 120$ ms.

A duração total foi reduzida para $T = 220$ ms, mateve-se um $\Delta t = 0.02$ ms fixo e variou-se, somente, o tamanho da discretização espacial, $\Delta x = \Delta y = [0.02, 0.01, 0.002, 0.001]$, levando, respectivamente, a um total de 10 mil, 40 mil, 1 milhão e 4 milhões de elementos.

No *MonoAlg3D*, que utiliza volumes finitos, a malha computacional foi gerada por uma função padrão e a difusão foi calculada através do método de Gradiente Conjugado com tolerância de 10^{-6}.

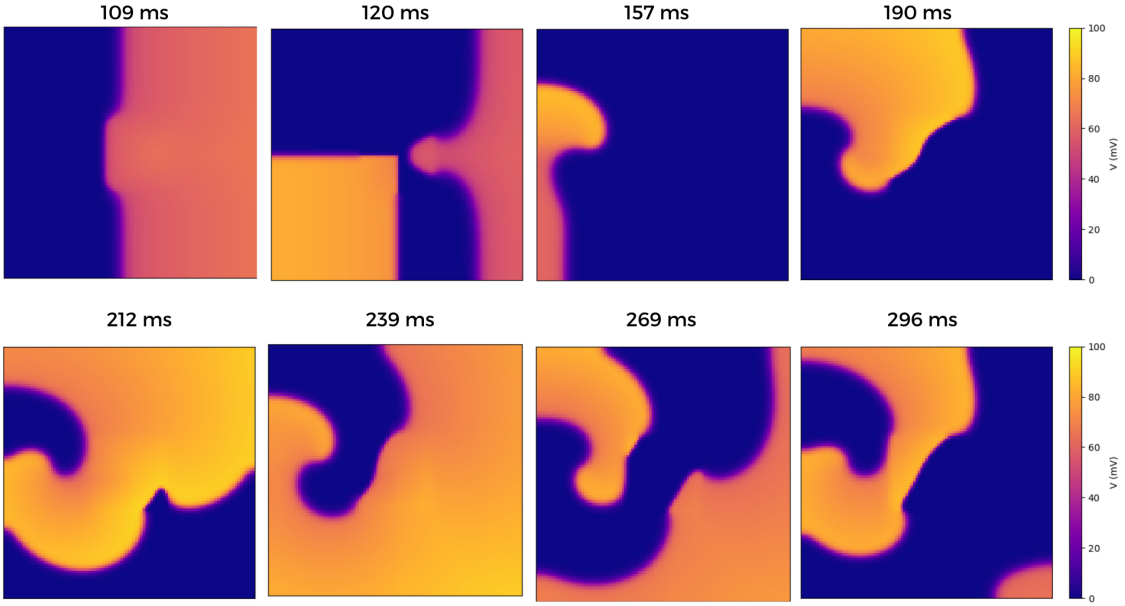
6 RESULTADOS E DISCUSSÕES

Os resultados estão divididos em duas partes, uma para a comparação entre os métodos numéricos e outra para comparação da implementação em GPU com o simulador *MonoAlg3D*. São levadas em consideração as configurações de simulação descritas na Seção 5.

6.1 COMPARAÇÃO ENTRE OS MÉTODOS NUMÉRICOS

A Figura 6 mostra, ao longo de diferentes instantes de tempo, a simulação reproduzindo uma arritmia cardíaca com a formação de uma espiral autossustentável. Neste caso, o S2 foi aplicado no instante $t_{S2} = 120$ ms e seguiu as demais configurações descritas na Seção 5.1.

Figura 6 - Formação e propagação da espiral em diferentes instantes de tempo



Fonte: Elaborada pelo autor (2023).

6.1.1 Janela de Vulnerabilidade

Pela Tabela 2, é possível perceber que os três métodos conseguiram capturar a mesma janela de vulnerabilidade. Nota-se que o OS_{ADI} foi o mais estável de todos, conseguindo executar a simulação para todos os Δt. Não é difícil perceber que ele é incondicionalmente estável, pois seus sistemas lineares são montados apenas a partir de esquemas implícitos. Por outro lado, tanto o SSI_{ADI} quanto o MOSI_{ADI} têm a presença do

termo reativo no lado direito dos sistemas lineares, exposto nas Equações (3.13) e (3.14), o que condiciona a estabilidade dos métodos. Nos casos em que não foi possível gerar a simulação da arritmia devido à instabilidade numérica, um hífen foi colocado nas tabelas.

Tabela 2 – Janela de vulnerabilidade, em milisegundos, para cada método com Δt variando de 0.01 ms a 0.16 ms

Δt (ms)	OS_{ADI}	SSI_{ADI}	MOSI_{ADI}
0.01	[119, 132]	[119, 132]	[119, 132]
0.02	[119, 132]	[119, 132]	[119, 132]
0.04	[119, 132]	[119, 132]	[119, 132]
0.08	[119, 132]	-	-
0.16	[119, 132]	-	-

Fonte: Elaborada pelo autor (2023).

6.1.2 Velocidade do Estímulo

Em cada simulação, foi medida a velocidade de propagação do estímulo S1. O valor de referência utilizado é de 0.603763 m/s. Os resultados, para cada método, estão na Tabela 3.

Tabela 3 – Velocidade de S1, em metros por segundo, para cada método com Δt variando de 0.01 ms a 0.16 ms

Δt (ms)	OS_{ADI}	SSI_{ADI}	MOSI_{ADI}
0.01	0.598007	0.603824	0.603824
0.02	0.592495	0.603622	0.604027
0.04	0.582147	0.603217	0.603217
0.08	0.563910	-	-
0.16	0.533175	-	-

Fonte: Elaborada pelo autor (2023).

A partir da análise da Tabela 3, percebe-se que o SSI_{ADI} e o MOSI_{ADI}, quando não apresentaram instabilidade, alcançaram resultados próximos ao valor de referência. Para $\Delta t = 0.04$ ms, em que os três métodos foram estáveis, observa-se erros de 3.58%, 0.09%, and 0.09% para o OS_{ADI}, o SSI_{ADI} e o MOSI_{ADI}, respectivamente.

6.1.3 Erro Numérico

Com o mesmo $t2$, isto é, com a mesma espiral sendo gerada, foi medido o erro numérico de cada método para diferentes Δt. A Tabela 4 mostra, em porcentagem, os erros relativos encontrados.

Tabela 4 – Erros relativos, em porcentagem, para cada método com Δt variando de 0.01 ms a 0.16 ms

Δt (ms)	OS_{ADI}	SSI_{ADI}	$MOSI_{ADI}$
0.01	7.7938	0.0836	0.5844
0.02	13.4978	0.1678	1.1528
0.04	23.3787	0.3242	2.2308
0.08	38.5614	-	-
0.16	61.1982	-	-

Fonte: Elaborada pelo autor (2023).

Da Tabela 4, é possível perceber que o SSI_{ADI} e o $MOSI_{ADI}$ têm uma precisão consideravelmente maior do que o OS_{ADI}. Para o $\Delta t = 0.04$ ms, por exemplo, o SSI_{ADI} foi 72 vezes melhor, e o $MOSI_{ADI}$, 10 vezes. Mesmo não sendo tão acurado quanto o SSI_{ADI}, o $MOSI_{ADI}$ alcançou bons resultados, com erros baixos, menores do que 3% em todas as simulações que foram executadas até o fim.

6.1.4 Tempo de Execução

Para a avaliação dos tempos de execução, foi considerada a performance de cada método executando com 6 *threads* em paralelo. A Tabela 5 mostra o tempo total de execução para cada método com diferentes valores para o passo de tempo.

Tabela 5 – Tempo total de execução, em segundos, para cada método com Δt variando de 0.01 ms a 0.16 ms

Δt(ms)	OS_{ADI}	SSI_{ADI}	$MOSI_{ADI}$
0.01	16.999	25.142	24.583
0.02	8.517	12.589	12.289
0.04	4.267	6.307	6.173
0.08	2.148	-	-
0.16	1.076	-	-

Fonte: Elaborada pelo autor (2023).

A análise da Tabela 5 permite constatar que o OS_{ADI} foi o método mais rápido para todos os Δt. Em contrapartida, a Tabela 4 também mostra que ele teve os maiores erros. Com isso, pode-se afirmar que o OS_{ADI} é ideal em cenários que tenham flexibilidade com relação à acurácia e demandem que as simulações sejam concluídas no menor tempo possível.

O $MOSI_{ADI}$, como mostra a Tabela 5, foi ligeiramente mais rápido do que o SSI_{ADI} em todos os casos testados. Essa observação é promissora, pois para modelos celulares mais complexos, com mais variáveis de estado, a tendência é que o $MOSI_{ADI}$ amplie ainda mais

essa vantagem, já que sua forma de atualizar $\boldsymbol{\eta}$ é mais simples e direta, não dependendo da predição.

A Tabela 6, por sua vez, apresenta os tempos de execução de cada uma das partes descritas na Seção 5, separadamente.

Tabela 6 – Tempo de execução de cada parte, em segundos, para cada método com Δt variando de 0.01 ms a 0.16 ms

	OS_{ADI}		SSI_{ADI}		$MOSI_{ADI}$	
Δt(ms)	1ª	2ª	1ª	2ª	1ª	2ª
0.01	3.497	13.672	7.738	17.704	7.121	17.695
0.02	1.761	6.843	3.879	8.988	3.557	8.910
0.04	0.968	3.477	1.951	4.499	1.812	4.539
0.08	0.569	1.763	-	-	-	-
0.16	0.253	0.873	-	-	-	-

Fonte: Elaborada pelo autor (2023).

Nota-se, pela Tabela 6, que, em todos os casos, para todos os métodos, o tempo de execução da segunda parte foi maior que o da primeira. Isso aconteceu, porque o FHN adaptado tem apenas duas variáveis. Modelos mais complexos do ventrículo, como Iyer-Mazhari-Winslow (19), ten Tusscher et al. (20) ou ToR-ORd (21), têm dezenas de variáveis. Como mostrado em (10), o convencional é que a primeira parte leve mais tempo.

Na Tabela 7, estão os *speedups*, fatores de aceleração, de cada método comparados com os seus respectivos tempos de execução sequencial para o caso com $\Delta t = 0.01$ ms. Os resultados mostram que, apesar das diferenças, todos os três obtiveram valores bem próximos do número de *threads* utilizado, indicando, assim, uma boa paralelização.

Tabela 7 – *Speedups* para cada método com diferentes números de *threads* comparando com o tempo de execução sequencial

Threads	OS_{ADI}	SSI_{ADI}	$MOSI_{ADI}$
2	1.94	1.95	1.98
4	3.78	3.74	3.73
6	5.41	5.31	5.29

Fonte: Elaborada pelo autor (2023).

6.2 COMPARAÇÃO DA PARALELIZAÇÃO EM GPU

A Tabela 8 mostra os tempos de execução de cada parte para a implementação do OS_{ADI} tanto na CPU, executando em paralelo com 6 *threads*, quanto na GPU. Os tempos do *MonoAlg3D* foram medidos e disponibilizados pelo próprio simulador.

Tabela 8 – Tempo de execução de cada parte, em segundos, para diferentes quantidades de elementos

	OS_{ADI} (CPU)		OS_{ADI} (GPU)		*MonoAlg3D*	
Elementos	1ª	2ª	1ª	2ª	1ª	2ª
10000	0.111	0.345	0.070	0.994	0.323	1.756
40000	0.336	1.306	0.159	2.047	0.651	1.860
1000000	25.435	68.061	2.225	10.311	13.008	12.234
4000000	138.333	314.601	8.665	26.512	77.642	91.754

Fonte: Elaborada pelo autor (2023).

Como o escopo do que é medido pelo *MonoAlg3D* não é exatamente o mesmo dos OS_{ADI}, a comparação entre eles acaba não sendo tão justa. Ainda assim, é possível levantar algumas observações interessantes a partir da análise da Tabela 8. Como já era esperado, as implementações que envolvem a GPU superam significativamente os resultados da CPU para os maiores casos, como aponta a Tabela 9.

Nota-se, também, que o *MonoAlg3D*, por ser mais robusto e voltado para simulações mais complexas, acaba não conseguindo ter um ganho de desempenho tão expressivo quanto o OS_{ADI} (GPU). Outra diferença entre eles, que mostrou ter uma relevância significativa, é o método de resolver a EDP parabólica. Os resultados indicam uma vantagem do ADI com Thomas em relação ao Gradiente Conjugado. Neste contexto e com os tempos que se têm, o OS_{ADI} (GPU) foi quase 9 vezes mais rápido do que o *MonoAlg3D* na primeira parte e 3 vezes na segunda, para o caso com 4 milhões de elementos.

Os *speedups* apresentados na Tabela 9 têm, como valor de comparação, os tempos de execução do OS_{ADI} executando com 6 *threads* na CPU.

Tabela 9 – *Speedups* de cada abordagem com diferentes quantidades de elementos comparando com o tempo de execução do OS_{ADI} (CPU) com 6 *threads*

	OS_{ADI} (GPU)			*MonoAlg3D*		
Elementos	1ª	2ª	Total	1ª	2ª	Total
10000	1.60	0.35	0.43	0.35	0.20	0.22
40000	2.12	0.64	0.74	0.52	0.70	0.65
1000000	11.43	6.60	7.46	1.95	5.56	3.70
4000000	15.96	11.87	12.87	1.78	3.43	2.67

Fonte: Elaborada pelo autor (2023).

Para o maior caso, o OS_{ADI} paralelizado em GPU foi praticamente 13 vezes mais rápido do que a implementação para CPU, como mostra a Tabela 9. Ainda por ela, é interessante constatar o comportamento da programação paralela em GPU. Abaixo de 1 milhão de elementos, no geral, o OS_{ADI} (CPU) foi o melhor. Percebe-se que, com relação ao OS_{ADI} (GPU), o que garante essa vantagem é a segunda parte. O *MonoAlg3D*, com as

características particulares já pontuadas, consegue superar o OS_{ADI} (CPU) a partir de 1 milhão de elementos.

7 CONCLUSÃO

Este trabalho apresentou três métodos numéricos baseados em *Alternating Direction Implicit* e técnicas de computação de alto desempenho para acelerar as simulações de arritmia cardíaca. Métricas importantes nesse tipo de estudo, como a janela de vulnerabilidade, a velocidade de propagação do estímulo, o erro numérico e o tempo de execução foram devidamente apresentadas. Dessa maneira, as comparações e discussões feitas neste trabalho contribuem para os avanços da modelagem computacional aplicada à cardiologia.

A programação paralela, tanto em CPU quanto em GPU, trouxe significativos ganhos de desempenho e mostrou-se imprescindível para que as simulações do coração possam ganhar escala e mais detalhamento.

O MOSI$_{ADI}$, apesar de não ter tido os menores erros e tempos de execução, apresentou um interessante custo-benefício. Sua forma de atualizar as variáveis de estado o torna promissor para o estudo com modelos celulares mais realistas (19, 20, 21).

Quando comparado com o tradicional método do Gradiente Conjugado, presente no simulador *MonoAlg3D*, e com a implementação em CPU, o OS$_{ADI}$ (GPU) mostrou ser o mais rápido. Ainda é necessário aprofundar as análises, avançar para três dimensões, estudar os erros numéricos, a estabilidade e afins. Porém, por ora, a perspectiva também é otimista.

7.1 TRABALHOS FUTUROS

A boa performance do OS$_{ADI}$ implementado com *CUDA* estimula a paralelização em GPUs dos demais esquemas ADI apresentados neste trabalho, a fim de investigar seus comportamentos.

Como trabalho futuros, deve-se implementar a anisotropia no modelo matemático para tornar as simulações mais realistas. Ainda nessa linha, a utilização de modelos celulares mais completos (19, 20, 21) é imprescindível.

Um outro caminho a ser percorrido é o da implementação dos métodos para simulações em 3D. Isso permitirá realizar comparações com *benchmarks* consolidados da área de eletrofisiologia (22) e aumentará a credibilidade do estudo.

REFERÊNCIAS

1 TSAO, Connie W. et al. Heart disease and stroke statistics—2023 update: a report from the American Heart Association. **Circulation**, v. 147, n. 8, p. e93-e621, 2023.

2 HUIKURI, Heikki V.; CASTELLANOS, Agustin; MYERBURG, Robert J. Sudden death due to cardiac arrhythmias. **New England Journal of Medicine**, v. 345, n. 20, p. 1473-1482, 2001.

3 NIEDERER, Steven A.; LUMENS, Joost; TRAYANOVA, Natalia A. Computational models in cardiology. **Nature Reviews Cardiology**, v. 16, n. 2, p. 100-111, 2019.

4 ROHR, Stephan. Role of gap junctions in the propagation of the cardiac action potential. **Cardiovascular research**, v. 62, n. 2, p. 309-322, 2004.

5 ROTH, Bradley J. How the anisotropy of the intracellular and extracellular conductivities influences stimulation of cardiac muscle. **journal of Mathematical Biology**, v. 30, p. 633-646, 1992.

6 CLAYTON, R. H.; ZHUCHKOVA, E. A.; PANFILOV, A. V. Phase singularities and filaments: simplifying complexity in computational models of ventricular fibrillation. **Progress in biophysics and molecular biology**, v. 90, n. 1-3, p. 378-398, 2006.

7 OLIVEIRA, Rafael Sachetto et al. Ectopic beats arise from micro-reentries near infarct regions in simulations of a patient-specific heart model. **Scientific reports**, v. 8, n. 1, p. 16392, 2018.

8 FITZHUGH, Richard. Impulses and physiological states in theoretical models of nerve membrane. **Biophysical journal**, v. 1, n. 6, p. 445-466, 1961.

9 TRAN, Diana X. et al. Vulnerability to re-entry in simulated two-dimensional cardiac tissue: effects of electrical restitution and stimulation sequence. **Chaos: An Interdisciplinary Journal of Nonlinear Science**, v. 17, n. 4, 2007.

10 SACHETTO OLIVEIRA, Rafael et al. Performance evaluation of GPU parallelization, space-time adaptive algorithms, and their combination for simulating cardiac electrophysiology. **International journal for numerical methods in biomedical engineering**, v. 34, n. 2, p. e2913, 2018.

11 KEENER, James; SNEYD, James (Ed.). **Mathematical physiology: II: Systems physiology**. New York, NY: Springer New York, 2009.

12 ROGERS, Jack M.; MCCULLOCH, Andrew D. A collocation-Galerkin finite element model of cardiac action potential propagation. **IEEE Transactions on Biomedical Engineering**, v. 41, n. 8, p. 743-757, 1994.

13 COLLI FRANZONE, Piero; PAVARINO, Luca F. A parallel solver for reaction–diffusion systems in computational electrocardiology. **Mathematical models and methods in applied sciences**, v. 14, n. 06, p. 883-911, 2004.

14 ENGELN-MÜLLGES, Giesela; UHLIG, Frank. **Numerical algorithms with C**. Springer Science & Business Media, 1996.

15 SUNDNES, Joakim et al. **Computing the electrical activity in the heart**. Springer Science & Business Media, 2007.

16 PEREIRA, Ricardo Reis. **Métodos de Diferenças Finitas para Problemas de Difusão e Reação Não Lineares**. 2019. Tese de Doutorado - Laboratório Nacional de Computação Científica, Petrópolis, 2019.

17 STRIKWERDA, John C. **Finite difference schemes and partial differential equations**. Society for Industrial and Applied Mathematics, 2004.

18 GLOSTER, Andrew et al. Efficient Interleaved Batch Matrix Solvers for CUDA. **arXiv preprint arXiv:1909.04539**, 2019.

19 IYER, Vivek; MAZHARI, Reza; WINSLOW, Raimond L. A computational model of the human left-ventricular epicardial myocyte. **Biophysical journal**, v. 87, n. 3, p. 1507-1525, 2004.

20 TEN TUSSCHER, Kirsten HWJ; PANFILOV, Alexander V. Alternans and spiral breakup in a human ventricular tissue model. **American Journal of Physiology-Heart and Circulatory Physiology**, v. 291, n. 3, p. H1088-H1100, 2006.

21 TOMEK, Jakub et al. Development, calibration, and validation of a novel human ventricular myocyte model in health, disease, and drug block. **Elife**, v. 8, p. e48890, 2019.

22 NIEDERER, Steven A. et al. Verification of cardiac tissue electrophysiology simulators using an N-version benchmark. **Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences**, v. 369, n. 1954, p. 4331-4351, 2011.