Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Bacharelado em Ciência da Computação

Simulador de Eletrofisiologia Cardíaca Utilizando o PETSc em Ambientes Paralelos

Lucas Marins Ramalho de Lima

JUIZ DE FORA
AGOSTO, 2025

Simulador de Eletrofisiologia Cardíaca Utilizando o PETSc em Ambientes Paralelos

Lucas Marins Ramalho de Lima

Universidade Federal de Juiz de Fora

Instituto de Ciências Exatas

Departamento de Ciência da Computação

Bacharelado em Ciência da Computação

Orientador: Joventino de Oliveira Campos

Coorientador: Bernardo Martins Rocha

JUIZ DE FORA

AGOSTO, 2025

Simulador de Eletrofisiologia Cardíaca Utilizando o PETSc em Ambientes Paralelos

Lucas Marins Ramalho de Lima

MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DO INSTITUTO DE CIÊNCIAS EXATAS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA, COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM CIÊNCIA DA COMPUTAÇÃO.

Aprovada por:

Joventino de Oliveira Campos
Doutor em Modelagem Computacional

Bernardo Martins Rocha
Doutor em Modelagem Computacional

Rodrigo Weber dos Santos
Doutor em Matemática

Ruy Freitas Reis
Doutor em Modelagem Computacional

JUIZ DE FORA
15 DE AGOSTO, 2025

À Sociedade Republicana Democrática da Escada e à memória do nosso amigo Rafael. Foi com vocês que aprendi a amar a matemática.

Resumo

A simulação computacional da eletrofisiologia cardíaca é uma ferramenta de grande potencial clínico e científico, cujo avanço é limitado pelo alto custo computacional. A utilização de computação paralela é, portanto, indispensável, e bibliotecas de software como o PETSc (Portable, Extensible Toolkit for Scientific Computation) fornecem a infraestrutura para tal. Este trabalho apresenta uma avaliação da biblioteca PETSc como plataforma para o desenvolvimento e análise de desempenho de protótipos de simuladores baseados no Método dos Elementos Finitos (MEF). Através de uma metodologia incremental, foram implementados três problemas de complexidade crescente: a equação de Poisson, para verificação de código; o modelo monodomínio, como principal caso de uso; e a equação do calor não linear, para validação da extensibilidade para problemas eletromecânicos do coração; A análise quantitativa focou na convergência do erro, no impacto de pré-condicionadores (Jacobi e ILU), na escalabilidade paralela e na comparação entre o refinamento de malha e o aumento da ordem dos elementos. Os resultados validaram a corretude da implementação e revelaram um claro *trade-off* de desempenho nos pré-condicionadores, cuja eficácia se mostrou dependente da escala do problema: enquanto o baixo custo computacional do Jacobi o tornou mais rápido no cenário 2D, a poderosa redução de iterações do ILU foi decisiva para sua superioridade no teste 3D de larga escala. A principal descoberta foi que o aumento da ordem dos elementos se mostrou uma estratégia computacionalmente mais eficiente que o refinamento da malha para atingir a acurácia desejada nos testes realizados. Conclui-se que o PETSc é uma ferramenta robusta e flexível para o aprimoramento de simuladores de eletrofisiologia cardíaca.

Palavras-chave: Eletrofisiologia cardíaca, modelagem do coração, computação paralela, computação distribuída, eficiência computacional, computação de alto desempenho.

Abstract

Computational simulation of cardiac electrophysiology is a tool of great clinical and scientific potential, whose advancement is limited by high computational cost. The use of parallel computing is therefore indispensable, and software libraries such as PETSc (Portable, Extensible Toolkit for Scientific Computation) provide the necessary infrastructure. This work presents an evaluation of the PETSc library as a platform for the development and performance analysis of simulator prototypes based on the Finite Element Method (FEM). Through an incremental methodology, three problems of increasing complexity were implemented: the Poisson equation, for code verification; the monodomain model, as the main use case; and the nonlinear heat equation, to validate the extensibility for cardiac electromechanical problems. The quantitative analysis focused on error convergence, the impact of preconditioners (Jacobi and ILU), parallel scalability, and the comparison between mesh refinement and increasing the element order. The results validated the correctness of the implementation and revealed a clear performance trade-off in the preconditioners, whose effectiveness proved to be dependent on the problem's scale: while the low computational cost of Jacobi made it faster in the 2D scenario, the powerful iteration reduction of ILU was decisive for its superiority in the large-scale 3D test. The main finding was that increasing the element order proved to be a more computationally efficient strategy than mesh refinement to achieve the desired accuracy in the performed tests. It is concluded that PETSc is a robust and flexible tool for the enhancement of cardiac electrophysiology simulators.

Keywords: Cardiac electrophysiology, heart modeling, parallel computing, distributed computing, computational efficiency, high-performance computing.

Agradecimentos

Gostaria de expressar minha profunda gratidão a todas as pessoas que contribuíram para a realização deste trabalho.

Agradeço imensamente ao meu orientador, Professor Joventino Campos, e ao meu coorientador, Professor Bernardo Martins, pela condução e pelos ensinamentos ao longo desta jornada. A paciência e o conhecimento de ambos foram indispensáveis para a concretização desta monografia.

Agradeço com carinho aos meus amigos de coração, aos meus pais e, em especial, à minha companheira Laura. Cada palavra de incentivo, cada gesto de compreensão e todo o apoio de todos vocês foram imprescindíveis para que eu conseguisse chegar até aqui.

Por fim, agradeço a todas as instituições e recursos que possibilitaram a realização desse trabalho, em especial à Fapemig (APQ-02445-24 e APQ-02752-24) e ao CNPQ (423278/2021-5 e 310722/2021-7).

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Abreviações

API	Interface de Programação de Aplicações
CG	Gradiente Conjugado
CNPQ	Conselho Nacional de Desenvolvimento Científico e Tecnológico
CPU	Unidade Central de Processamento
DCC	Departamento de Ciência da Computação
DMPLEX	Gerenciamento de Malhas não Estruturadas
DoF	Graus de Liberdade
EDO	Equação Diferencial Ordinária
EDP	Equação Diferencial Parcial
Fapemig	Fundação de Amparo à Pesquisa do Estado de Minas Gerais
GMRES	Resíduo Mínimo Generalizado
GPU	Unidade de Processamento Gráfico
ILU	Fatoração ILU Incompleta
KSP	Métodos de Subespaço de Krylov
MEF	Método dos Elementos Finitos
MMS	Método das Soluções Manufaturadas
MPI	Message Passing Interface
MVF	Método dos Volumes Finitos
PETSc	Portable, Extensible Toolkit for Scientific Computation
PetscDS	Sistema Discreto
PETSCFE	Abstração para Elementos Finitos do PETSc
SNES	Solucionadores de Equações Não Lineares Escaláveis
TS	Time Stepper
UFJF	Universidade Federal de Juiz de Fora

1 Introdução

A modelagem computacional da eletrofisiologia cardíaca é um campo interdisciplinar que engloba conhecimentos da biologia, medicina, matemática aplicada e ciência da computação para simular o comportamento elétrico de um coração (TRAYANOVA *et al.*, 2024). Nesse processo, a propagação de estímulos elétricos no tecido cardíaco é representada através de modelos matemáticos e computacionais que reproduzem o comportamento complexo das correntes iônicas e das estruturas intracelulares, além da propagação elétrica de uma célula para outra.

Tais modelos possuem um alto custo computacional, pois exigem uma refinada discretização do domínio tanto no espaço quanto no tempo (NIEDERER *et al.*, 2011b). Portanto, para que simulações em escala de órgão, como um coração humano completo, sejam viáveis, é imperativo que a implementação desses modelos seja eficiente, otimizando o uso dos recursos computacionais disponíveis.

Para superar essa barreira computacional, é necessário combinar a aplicação de métodos numéricos eficientes e a programação paralela, particularmente na resolução dos grandes e esparsos sistemas de equações que emergem da discretização do modelo matemático (VINCENT *et al.*, 2015). Nessa abordagem, as duas estratégias se complementam: enquanto os métodos numéricos buscam reduzir a complexidade ou o número de operações necessárias para a solução, a programação paralela atua na distribuição do custo computacional, dividindo tarefas de montagem, fatoração e solução desses sistemas lineares em múltiplos núcleos de processamento, permitindo que as operações sejam realizadas mais rapidamente.

Para implementar essa abordagem paralela de forma robusta e eficiente, a comunidade científica dispõe de bibliotecas de software especializadas. Dentre elas, destaca-se o Portable, Extensible Toolkit for Scientific Computation (PETSc), um pacote de código aberto projetado para a solução numérica de problemas científicos complexos em arquiteturas de alto desempenho (BALAY *et al.*, 2021). A biblioteca é projetada em torno de uma infraestrutura de computação paralela, sobre a qual implementa um robusto con-

junto de ferramentas numéricas: resolvedores (solvers) para sistemas lineares e não lineares, integradores temporais e uma ampla seleção de pré-condicionadores. Essa arquitetura modular torna o PETSc uma base sólida e amplamente utilizada para o desenvolvimento de simuladores avançados, como os de eletrofisiologia cardíaca (PLANK *et al.*, 2021).

A redução no tempo de execução das simulações é, portanto, o fator-chave que torna esta tecnologia viável para a aplicação clínica (TRAYANOVA *et al.*, 2024). Simulações que demandam dias ou semanas são impraticáveis no apoio a decisões médicas. Atingir um desempenho computacional que permita a criação de modelos cardíacos customizados em uma janela de tempo útil é fundamental para o avanço da medicina personalizada, o desenvolvimento de novas terapias e o aprimoramento de dispositivos. Além do benefício clínico direto, um ganho de performance também acelera o ciclo de pesquisa e desenvolvimento, diminuindo a necessidade de ensaios in vivo e, consequentemente, os custos e dilemas éticos associados.

1.1 Contextualização

Ao longo de seu desenvolvimento, a modelagem computacional da eletrofisiologia cardíaca consolidou-se como uma poderosa ferramenta para a investigação científica e clínica. Inicialmente, as simulações foram cruciais para formular e testar hipóteses sobre os mecanismos fundamentais da função cardíaca. Com o avanço da área, tornaram-se uma plataforma para avaliar, *in silico*, o impacto de compostos farmacológicos no tecido cardíaco (AMANFU; SAUCERMAN, 2011). Mais recentemente, demonstraram seu valor clínico ao otimizar a precisão de procedimentos como a ablação por cateter, guiando a terapia para melhorar a taxa de sucesso em pacientes com arritmias ventriculares pós-infarto (CAMPOS *et al.*, 2019). O sucesso dessas aplicações é diretamente dependente da sofisticação e da eficiência dos simuladores utilizados.

Um exemplo de simulador amplamente adotado pela comunidade científica internacional é o MonoAlg3D (OLIVEIRA *et al.*, 2017). Baseado no método dos volumes finitos (MVF), ele está apto para simulações que utilizam geometrias complexas de pacientes, geradas a partir de imagens médicas. Sua estratégia de computação de alto desempenho foca na máxima otimização de recursos em um nó computacional único, combinando o

paralelismo em CPUs multi-core (via OpenMP) com a aceleração massiva de cálculos em GPUs.

Outra plataforma de referência, o openCARP (PLANK *et al.*, 2021), emprega o método dos elementos finitos (MEF) para lidar com a complexidade de geometrias cardíacas realistas. Seu poder computacional é fundamentado na biblioteca PETSc, que serve como base para sua arquitetura de computação distribuída via MPI (MESSAGE PASSING INTERFACE FORUM, 2015). Ao ser construído sobre o PETSc, o openCARP herda todo o seu ecossistema, incluindo uma vasta coleção de solucionadores e estruturas de dados otimizadas para ambientes de larga escala. Atualmente, seu foco principal é a eletrofisiologia, mas o projeto está em expansão para incorporar modelos de mecânica cardíaca.

Um terceiro exemplo, o simulador Cardiax (CAMPOS *et al.*, 2018), desenvolvido na Universidade Federal de Juiz de Fora (UFJF), ilustra o foco no desafio do acoplamento multifísico da eletromecânica cardíaca. O simulador já utiliza a biblioteca PETSc, porém seu uso atual se concentra na etapa de resolução dos sistemas de equações lineares e não lineares. Contudo, as estruturas de dados e a montagem das matrizes do método dos elementos finitos ainda não foram migradas para o ecossistema paralelo do PETSc. Essa característica faz com que, na prática, a implementação seja serial, pois a montagem da matriz — uma etapa computacionalmente intensiva — não explora o paralelismo. Nesse contexto, a integração completa com as estruturas de dados do PETSc representa uma clara oportunidade para a paralelização integral do simulador, otimizando não apenas a resolução, mas também a etapa de montagem do sistema.

As versões recentes do PETSc (BALAY *et al.*, 2021) incorporam funcionalidades que se estendem além da resolução de sistemas de equações. A biblioteca agora oferece suporte para a implementação de métodos de elementos finitos, incluindo o gerenciamento de malhas não estruturadas em ambientes distribuídos (através do DMPlex). Essa capacidade permite que etapas computacionalmente intensivas da simulação, como a montagem de matrizes e vetores, sejam executadas em paralelo utilizando as estruturas de dados nativas da biblioteca. Adicionalmente, o desenvolvimento do PETSc inclui suporte para aceleração em GPUs, indicando um caminho para futuras otimizações. Este conjunto de

funcionalidades oferece os mecanismos necessários para a modernização e paralelização de simuladores existentes, como o Cardiax, que ainda não exploram tais recursos.

1.2 Objetivos

O objetivo geral deste trabalho é avaliar a biblioteca PETSc como uma ferramenta para o desenvolvimento de simuladores de eletrofisiologia cardíaca em ambientes de computação paralela. A análise foca em duas vertentes principais: (1) o desempenho e a escalabilidade da solução paralela e (2) a flexibilidade da biblioteca como uma base extensível para a implementação de problemas não lineares, como o acoplamento eletromecânico.

Para atingir o objetivo geral, os seguintes objetivos específicos foram definidos:

- **Desenvolver um protótipo de simulador** para a eletrofisiologia cardíaca com base no modelo monodomínio, utilizando os recursos da biblioteca PETSc.
- **Explorar a API e a estrutura de programação do PETSc** através da implementação de um problema-modelo de Poisson. Esta etapa inicial servirá como base para a compreensão e a validação das estruturas de dados, dos solucionadores e do fluxo de trabalho paralelo que serão utilizados nos objetivos subsequentes.
- **Analisar o desempenho da paralelização** em um cenário 2D, investigando o impacto de diferentes pré-condicionadores. As métricas de avaliação incluem o *speedup*, o número de iterações do solver (KSP) e a eficiência paralela.
- **Avaliar a eficiência do uso da computação paralela** para a redução do tempo de execução, mantendo a precisão dos resultados e o correto funcionamento do modelo.
- **Investigar a relação entre a ordem do elemento finito e o refinamento da malha** em um cenário 3D. O objetivo é avaliar se o uso de polinômios de aproximação de grau superior via PETScFE, pode produzir resultados de qualidade comparável a uma malha mais refinada com elementos lineares, otimizando o custo computacional.

- **Demonstrar a extensibilidade da implementação** por meio da resolução do problema não-linear análogo a mecânica cardíaca. Esta etapa valida a estrutura como uma base robusta para futuras extensões que exigem solvers não-lineares como o acoplamento eletromecânico.

A conclusão bem-sucedida destes objetivos resultará em um protótipo funcional de simulador paralelo, uma análise quantitativa de seu desempenho e uma validação de sua flexibilidade para problemas multifísicos. Coletivamente, os resultados fornecerão uma avaliação concreta do PETSc como plataforma para o desenvolvimento de simuladores cardíacos de alta performance.

2 Fundamentação Teórica

Neste capítulo, serão apresentados os fundamentos teóricos que alicerçam o desenvolvimento de um simulador protótipo. A discussão abordará os modelos matemáticos utilizados para descrever esse fenômeno, iniciando-se com a equação monodomínio para a propagação do potencial elétrico, acoplada ao modelo celular fenomenológico de Mitchell-Schaeffer (MITCHELL; SCHAEFFER, 2003). Adicionalmente, para fins didáticos e de validação, serão descritos dois problemas-modelo: a equação de Poisson e a equação do calor não linear. A discretização espacial dessas equações será realizada pelo método dos elementos finitos (MEF), cuja formulação e implementação serão detalhadas no capítulo de Metodologia.

2.1 Equação de Poisson

A Equação de Poisson é uma equação diferencial parcial elíptica utilizada para modelar diversos fenômenos físicos em estado de equilíbrio. Em sua forma geral, que considera um coeficiente de difusão D variável, a equação é dada por (2.1), cujas variáveis são descritas na Tabela 2.1.

$$-\nabla \cdot (D\nabla u) = f \quad \text{em } \Omega \tag{2.1}$$

Para que a equação tenha uma solução única, é necessário impor condições de contorno na fronteira $\delta\Omega$ do domínio. As duas condições de contorno utilizadas são:

Tabela 2.1: Descrição das variáveis da Equação de Poisson.

Variável	**Descrição**
Ω	Domínio espacial sobre o qual o problema é resolvido
u	Variável de interesse (e.g. temperatura)
D	Coeficiente de difusão do meio (constante ou uma função da posição)
f	Termo de fonte

- Condição de Dirichlet:

$$u = u_D \quad \text{em } \Gamma_D \subseteq \partial\Omega \tag{2.2}$$

- Condição de Neumann:

$$-D\frac{\partial u}{\partial n} = g \quad \text{em } \Gamma_N \subseteq \partial\Omega \tag{2.3}$$

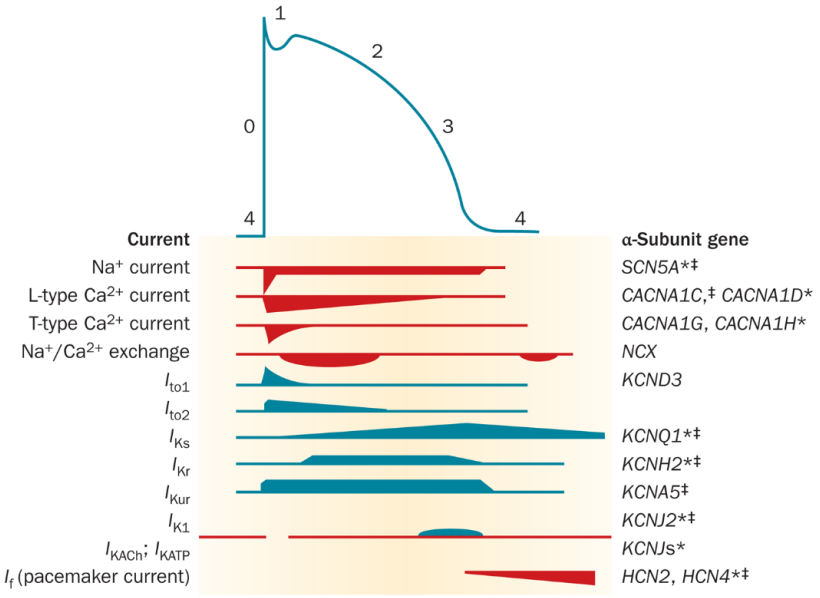
No contexto deste trabalho, a equação de Poisson é empregada como um problema-modelo para a verificação do código implementado. A principal vantagem de sua utilização reside na possibilidade de se aplicar o Método das Soluções Manufaturadas (MMS). Através do MMS, uma solução exata é predefinida, e o termo de fonte f correspondente é derivado analiticamente. Isso permite uma avaliação quantitativa e rigorosa do erro da aproximação numérica, validando a implementação do método dos elementos finitos do PETSc.

2.2 Eletrofisiologia cardíaca

O coração humano funciona como uma bomba eletromecânica, cuja eficiência depende da contração sincronizada de suas células musculares, os cardiomiócitos (KLABUNDE, 2012). Essa sincronia é orquestrada por uma onda de excitação elétrica que se propaga de forma ordenada através do tecido cardíaco. Em nível celular, esse fenômeno é governado pelo potencial de ação: uma rápida e transitória variação no potencial elétrico da membrana celular, impulsionada por fluxos de íons (Figura 2.1). Os cardiomiócitos são eletricamente acoplados por junções comunicantes (gap junctions), que permitem que a corrente iônica flua de uma célula para suas vizinhas. Esse acoplamento faz com que o tecido se comporte como um sincício funcional, onde a excitação de uma região desencadeia uma frente de onda que se propaga por toda a massa muscular, resultando na contração que impulsiona o sangue.

A propagação do estímulo elétrico no coração é um processo contínuo que emerge de eventos discretos em nível celular (KLABUNDE, 2012). A modelagem matemática desse fenômeno, portanto, requer uma abordagem que integre a escala do tecido com a escala da célula.

Figura 2.1: Relação entre correntes iônicas e formação do potencial de ação cardíaco.



Fonte: Darbar e Roden (2013).

A interação entre os cardiomiócitos e a difusão da corrente através do tecido são capturadas por uma equação de reação-difusão. O termo de difusão descreve como o potencial elétrico se espalha pelo domínio, enquanto o termo de reação modela a complexa dinâmica iônica que gera o potencial de ação em cada célula.

2.2.1 Modelo Monodomínio

O modelo monodomínio (GERARDO-GIORDA, 2016) descreve a propagação de estímulos por um tecido excitável através de uma equação diferencial parcial que define o potencial transmembrânico de cada célula em um instante de tempo. O modelo é dado por (2.4) e a descrição detalhada de cada uma de suas variáveis e parâmetros é apresentada na Tabela 2.2.

$$\chi C_m \frac{\partial V}{\partial t} = \nabla.(\boldsymbol{\sigma}\nabla V) - \chi I_{ion}(V,\eta) + I_{stim} \quad \text{em } \Omega, \tag{2.4}$$

o qual é suplementado por condições de contorno homogêneas:

$$(\boldsymbol{\sigma}\nabla V)\cdot \mathbf{n} = 0 \quad \text{em } \partial\Omega \times (0,T]. \tag{2.5}$$

Tabela 2.2: Descrição das variáveis do modelo do monodomínio

Variável	**Descrição**
V	Potencial de membrana (mV)
C_m	Capacitância da membrana celular (μF/cm^2)
$\boldsymbol{\sigma}$	Tensor de condutividade do tecido (S/cm)
$I_{ion}(V, \eta)$	Corrente iônica calculada de acordo com o modelo celular
I_{stim}	Corrente de estímulo aplicado (μA/cm^2)
η	Conjunto de variáveis de ativação (sem unidade)
T	Tempo final de simulação (ms)
χ	Razão área-volume da membrana celular (cm^{-1})

O modelo celular é acoplado ao modelo monodomínio por meio do cálculo da corrente iônica total (I_{ion}), que é obtida a partir de um sistema de equações diferenciais ordinárias (EDOs) que descreve a dinâmica dos canais iônicos e do potencial transmembrânico. Dessa forma, o comportamento eletrofisiológico celular é integrado à equação diferencial parcial do monodomínio (GERARDO-GIORDA, 2016), permitindo a simulação da propagação do potencial de ação no tecido cardíaco.

O modelo monodomínio é uma simplificação do modelo bidomínio (SUNDNES *et al.*, 2006), que descreve a atividade elétrica do tecido cardíaco por meio de duas equações diferenciais parciais, representando separadamente os potenciais intracelular e extracelular. No modelo monodomínio, essa distinção é eliminada ao assumir uma relação fixa entre as condutividades dos dois meios, resultando em uma única equação diferencial que descreve a propagação do potencial transmembrânico.

Como apresentado em (BOURGAULT; PIERRE, 2010), apesar de o modelo bidomínio capturar com maior precisão a anisotropia do meio extracelular, o modelo monodomínio consegue reproduzir de forma satisfatória os resultados observados experimentalmente, oferecendo uma alternativa computacionalmente mais eficiente para a simulação da propagação elétrica no tecido cardíaco.

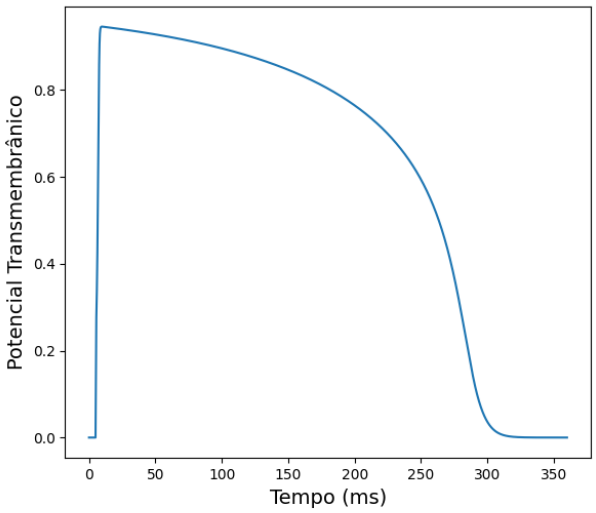
2.2.2 Modelos celulares

A célula é delimitada pela membrana plasmática, que separa o espaço intracelular do ambiente extracelular. Essa estrutura funciona como uma barreira seletiva, impedindo a passagem direta de íons entre os dois compartimentos. A diferença na concentração de íons em ambos os lados da membrana gera uma diferença de potencial, chamada de

potencial transmembrânico.

No entanto, a membrana contém proteínas especializadas, como canais iônicos e transportadores, que regulam o transporte de íons, permitindo a geração e o fluxo de correntes iônicas através da membrana. Isso faz com que o potencial transmembrânico varie no tempo, aumentando quando a célula é estimulada eletricamente e formando uma curva chamada potencial de ação (KLABUNDE, 2012) representado pelo modelo Mitchell-Shaeffer (MITCHELL; SCHAEFFER, 2003) na Figura 2.2.

Figura 2.2: Potencial de ação de célula cardíaca de acordo com o modelo Mitchell-Shaeffer



Fonte: O autor.

A modelagem matemática da atividade elétrica celular foi estabelecida pelo trabalho pioneiro de Hodgkin e Huxley (HODKIN; HUXLEY, 1952). Eles demonstraram que o comportamento dos canais iônicos da membrana poderia ser descrito por um sistema de equações diferenciais ordinárias (EDOs), criando um paradigma que fundamenta a área até hoje. Com base nesse princípio, modelos cardíacos biofisicamente detalhados foram desenvolvidos, como o de ten Tusscher (TUSSCHER; PANFILOV, 2004) e o ToR-ORd (TOMEK *et al.*, 2019). A alta fidelidade desses modelos, que podem incluir dezenas de variáveis de estado, os torna ferramentas poderosas para a investigação de mecanismos de doenças e para a farmacologia computacional. No entanto, essa complexidade também impõe um alto custo computacional, o que pode ser um fator limitante para simulações em larga escala.

Em contrapartida a essa complexidade, existem os modelos fenomenológicos. Ao

invés de descrever cada corrente iônica individualmente, esses modelos buscam reproduzir o comportamento emergente do potencial de ação com um número reduzido de equações e parâmetros, oferecendo um balanço favorável entre a fidelidade fisiológica e a eficiência computacional.

Neste trabalho, foi adotado o modelo de Mitchell-Schaeffer (MITCHELL; SCHAEFFER, 2003). Este é um modelo de duas variáveis que descreve o potencial de membrana (u) e uma variável de "porta"(gate) (h) que representa a corrente iônica total de forma simplificada, descrita pelas equações (2.6) e (2.7), cujas variáveis são descritas na Tabela 2.3.

$$\frac{du}{dt} = \frac{-hu^2(u-1)}{\tau_{in}} - \frac{u}{\tau_{out}} \tag{2.6}$$

$$\frac{dh}{dt} = \begin{cases} \frac{1-h}{\tau_{open}}, & \text{se } u < u_{gate} \\ \frac{-h}{\tau_{close}}, & \text{se } u > u_{gate} \end{cases} \tag{2.7}$$

Tabela 2.3: Descrição das variáveis e parâmetros do modelo de Mitchell-Schaeffer.

Variável	**Descrição**
u	Potencial de membrana normalizado (adimensional)
h	Variável de "porta"que controla a corrente (adimensional)
τ_{in}	Constante de tempo para a corrente de influxo (e.g., sódio)
τ_{out}	Constante de tempo para a corrente de efluxo (e.g., potássio)
τ_{open}	Constante de tempo para a abertura da porta (ativação)
τ_{close}	Constante de tempo para o fechamento da porta (inativação)
u_{gate}	Limiar de potencial que ativa a mudança de estado da porta

2.3 Equação do Calor Não Linear

Um passo importante na evolução de um simulador de eletrofisiologia é sua extensão para modelos multifísicos de eletromecânica cardíaca. Um desafio fundamental nesses problemas é a forte não-linearidade intrínseca ao tecido cardíaco, cujas propriedades mecânicas, como a tensão e a rigidez, dependem diretamente do estado de deformação atual. Essa interdependência exige uma abordagem numérica capaz de resolver sistemas de equações não-lineares de forma robusta e eficiente a cada passo de tempo.

Para avaliar a capacidade do *framework* desenvolvido em lidar com tal desafio, foi utilizado como análogo o problema da equação do calor não linear. A escolha deste problema-modelo é estratégica por três motivos centrais. Primeiramente, sua formulação matemática replica a principal característica dos modelos mecânicos: a não-linearidade está contida em um coeficiente da equação que depende da própria solução, onde o coeficiente de difusão $D(u)$ mimetiza a dependência das propriedades do material com a solução.

Em segundo lugar, a resolução deste problema transiente não-linear requer o mesmo ferramental computacional da eletromecânica, notadamente um *solver* para sistemas de equações não-lineares, como o SNES do PETSc, acoplado a um integrador temporal. Dessa forma, a sua correta solução serve como uma validação direta de toda a infraestrutura para a simulação de problemas mais complexos.

Finalmente, ao contrário do acoplamento eletromecânico, o problema do calor permite o emprego do Método das Soluções Manufaturadas. Essa técnica possibilita a verificação quantitativa do código ao comparar a solução numérica com uma solução analítica exata, garantindo a corretude da implementação do *solver* não-linear antes de sua aplicação em cenários onde tal verificação não é factível. A equação utilizada é dada por:

$$\frac{\partial u}{\partial t} - \nabla \cdot (D(u)\nabla u) = f \tag{2.8}$$

O problema é suplementado com condições de contorno e uma condição inicial derivadas da solução exata conhecida, conforme detalhado no capítulo de Metodologia.

3 Metodologia

Este capítulo detalha a metodologia empregada para a implementação dos problemas-teste e para a avaliação da biblioteca PETSc. O foco é descrever o arcabouço computacional e matemático utilizado, desde as ferramentas de software até a formulação numérica das equações.

3.1 Método dos Elementos Finitos

O Método dos Elementos Finitos (MEF) é uma técnica numérica utilizada para encontrar soluções aproximadas para equações diferenciais (LARSON; BENGZON, 2013), cuja ideia central é discretizar um domínio contínuo em um conjunto de subdomínios menores, os elementos.

A aplicação do método se inicia com a formulação forte do problema, que é a própria equação a ser resolvida, juntamente com suas condições de contorno. Essa formulação é dita "forte" porque exige que a equação seja válida em todos os pontos do domínio e que a solução u seja suficientemente derivável ($u \in \mathcal{H}^2(\Omega)$[1]). O objetivo do MEF é transformar essa equação em uma formulação fraca, que enfraquece esses requisitos de continuidade, permitindo que se procure uma solução em um espaço de funções menos restritivo ($u \in \mathcal{H}^1(\Omega)$).

Para obter a formulação fraca, o primeiro passo é multiplicar a equação por uma função-teste (ou função de peso) v e integrar sobre todo o domínio Ω. A equação integral resultante deve ser válida para toda função-teste v no espaço escolhido.

Tomando como exemplo a equação de Poisson ($-\Delta u = f$), o processo se inicia com:

$$-\int_\Omega \Delta u v \, dx = \int_\Omega f v \, dx \quad \forall v \in \mathcal{V} \tag{3.1}$$

[1]Espaço de Sobolev de funções quadrado-integráveis cuja primeira derivada também é quadrado-integrável.

$$\mathcal{V} = \{v \in \mathcal{H}^1(\Omega); v|_{\partial\Omega} = 0\} \tag{3.2}$$

O passo seguinte e fundamental é aplicar a integração por partes ao termo da esquerda. O objetivo desta manipulação é reduzir a ordem da derivada da variável u (de segunda para primeira ordem) e transferi-la para a função-teste v.

$$\int_\Omega \nabla u \cdot \nabla v \; dx - \int_{\partial\Omega} \frac{\partial u}{\partial n} v \; ds = \int_\Omega f v \; dx \tag{3.3}$$

Esta é a formulação fraca do problema. Nota-se que como resultado da integração por partes, surge um termo de fronteira $\partial\Omega$. Este termo é crucial, pois é através dele que as condições de contorno de Neumann são aplicadas, enquanto as condições de Dirichlet são impostas diretamente no espaço de funções da solução.

Para enunciar o problema variacional de maneira generalizada, consideramos que a fronteira $\partial\Omega$ é dividida em duas partes: $\partial\Omega_D$ sobre a qual será imposta uma condição de Dirichlet, e $\partial\Omega_N$, onde uma condição de Neumann ($\nabla u \cdot \mathbf{n}|_{\partial\Omega_N} = g$ é aplicada, tal que $\partial\Omega = \partial\Omega_D \cup \partial\Omega_N$.

O problema então consiste em: encontrar uma função u que pertença ao espaço de funções $\mathcal{H}^1(\Omega)$ e que satisfaça a seguinte equação para toda função-teste $v \in \mathcal{V}$

$$a(u, v) = l(v) \tag{3.4}$$

com

$$a(u, v) = \int_\Omega \nabla u \nabla v \; dx \quad \text{e} \quad l(v) = \int_\Omega f v \; dx + g \tag{3.5}$$

A formulação variacional ainda é um problema de dimensão infinita. O passo final é discretizá-lo através do Método de Galerkin. A solução contínua u é aproximada por uma combinação linear de N funções de base (ou funções de forma) ϕ_j, definidas sobre a malha de elementos:

$$u(x) \approx u_h(x) = \sum_{j=1}^{N} U_j \phi_j(x) \tag{3.6}$$

Os coeficientes U_j são os novos valores desconhecidos. O método de Galerkin utiliza as

próprias funções de base como funções-teste, ou seja, $v = \phi_i$ para $i = 1, \ldots, N$.

Substituindo a aproximação u_h e as funções-teste ϕ_i na formulação variacional, obtém-se um sistema de N equações algébricas para as N incógnitas U_j:

$$\sum_{j=1}^{N} U_j \cdot \underbrace{\left(\int_\Omega \nabla\phi_j \cdot \nabla\phi_i \, dx \right)}_{A_{ij}} = \underbrace{\int_\Omega f\phi_i \, dx + \int_{\partial\Omega_N} g\phi_i \, ds}_{F_i} \tag{3.7}$$

Este sistema de equações é escrito na forma matricial clássica:

$$\mathbf{AU} = \mathbf{F} \tag{3.8}$$

onde $\mathbf{A}$ é a matriz de rigidez, $\mathbf{U}$ é o vetor de incógnitas e $\mathbf{F}$ é o vetor de carga. A solução deste sistema linear fornece os coeficientes que definem a solução aproximada em todo o domínio.

Uma vantagem fundamental do método dos elementos finitos é a capacidade de se controlar a precisão da solução através da escolha das funções de base, que podem ser polinômios de diferentes graus, k. A teoria do método estabelece que o erro de aproximação converge com uma ordem de $k + 1$, ou seja, o erro é proporcional a h^{k+1}, onde h é o tamanho do elemento.

3.2 Estrutura do PETSc

Uma vez estabelecida a fundamentação teórica do Método dos Elementos Finitos, o desafio subsequente é a sua tradução para um código computacional eficiente e paralelo. Para essa finalidade, este trabalho utiliza a biblioteca PETSc (Portable, Extensible Toolkit for Scientific Computation) (BALAY *et al.*, 2021), um ecossistema de software de alto desempenho para a computação científica.

O PETSc oferece acesso a uma vasta gama de resolvedores para os sistemas lineares e não lineares resultantes, além de uma abstração para a implementação do MEF. Crucialmente, toda a arquitetura da biblioteca é construída com suporte nativo para a computação paralela.

3.2.1 TimeStepper

Com exceção da equação de Poisson, os problemas abordados neste trabalho são transientes, ou seja, evoluem no tempo. Para gerenciar a integração temporal de forma robusta e modular, foi utilizada a interface TS (Time Stepper) do PETSc.

Este componente oferece acesso a uma variedade de métodos para a integração de equações diferenciais no tempo. Para este trabalho, foi selecionado o método de Euler Implícito, principalmente devido à sua alta estabilidade numérica, uma característica desejável para os problemas de eletrofisiologia. O TS automatiza o avanço da solução no tempo e se integra nativamente com os solucionadores não lineares (SNES) para resolver o sistema de equações a cada passo de tempo.

3.2.2 Solucionadores Não Lineares (SNES)

Nos problemas onde a discretização resulta em um sistema de equações não lineares, como na equação do calor não linear que será tratada nesse trabalho, é necessário empregar um solucionador apropriado. O PETSc oferece uma interface para esta tarefa, o SNES (Scalable Nonlinear Equation Solvers).

O SNES foi projetado para resolver sistemas de equações não lineares da forma geral:

$$\mathbf{F}(\mathbf{U}) = 0 \tag{3.9}$$

A base da maioria dos métodos do SNES é o método de Newton. A cada iteração i, o método de Newton lineariza o problema e resolve um sistema de equações lineares para encontrar a atualização da solução, $\Delta\mathbf{U}_i$ onde $J(\mathbf{U}_i)$ é a matriz Jacobiana do sistema:

$$\mathbf{J}(\mathbf{U}_i)\Delta\mathbf{U}_i = -\mathbf{F}(\mathbf{U}_i) \tag{3.10}$$

Após encontrar $\Delta\mathbf{U}_i$, a solução é atualizada: $U_{i+1} = U_i + \Delta U_i$. O SNES gerencia todo esse processo iterativo, incluindo estratégias como line search ou trust region para garantir a convergência.

Para a equação do calor não linear, foi utilizado o solucionador padrão do PETSc,

`SNESNEWTONLS`, que implementa o método de Newton. É importante notar que a interface SNES também pode ser utilizada como um invólucro para problemas lineares. Para os casos da equação de Poisson e do modelo monodomínio, foi utilizada a opção `-snes_type ksponly`. Essa abordagem instrui o SNES a pular o laço não linear e invocar diretamente o solucionador de sistemas lineares KSP.

3.2.3 Solucionadores Lineares KSP

O PETSc gerencia a solução de sistemas lineares através da interface KSP (Krylov Subspace Methods). Este componente oferece uma grande variedade de solucionadores diretos e iterativos, vários com suporte à execução em ambientes paralelos.

A seleção do solucionador linear para cada problema foi baseada nas propriedades matemáticas da matriz do sistema resultante pela aplicação do MEF.

Para a equação de Poisson e para o modelo monodomínio (discretizado implicitamente no tempo), o método dos elementos finitos gera um sistema de equações cuja matriz é simétrica e positiva definida. Para este tipo de sistema, o método do Gradiente Conjugado (CG) é a escolha ótima, pois garante convergência com um custo computacional e de memória mínimos.

Já para a equação do calor não linear, a solução a cada passo de tempo requer um método de Newton, que gera uma matriz Jacobiana que, em geral, não é simétrica. Na ausência da garantia de simetria, o Gradiente Conjugado não pode ser aplicado (SAAD, 2003). Portanto, foi utilizado um método mais robusto e geral, o GMRES (Generalized Minimal RESidual), que é capaz de lidar com matrizes não simétricas.

Para acelerar a convergência dos solucionadores iterativos, é comum o uso de um pré-condicionador. Neste trabalho, para avaliar o impacto do pré-condicionamento, foram testados diferentes pré-condicionadores. Como o PETSc não oferece implementação paralela do pré-condicionador ILU, foi necessário utilizar a integração do PETSc com a biblioteca externa *Hypre*, que é especializada em pré-condicionadores algébricos de alta performance.

Foram testados os pré-condicionadores:

- Jacobi (nativo do PETSc) : Este pré-condicionador utiliza o inverso da matriz diago-

nal de A como o operador de pré-condicionamento. A aplicação do pré-condicionador a cada iteração consiste na solução de um sistema diagonal.

- Fatoração Incompleta LU (ILU): O método produz uma aproximação esparsa dos fatores L e U da matriz A e pode ser aplicado a matrizes não simétricas em geral.

3.2.4 Gerenciamento de Malhas não Estruturadas (DMPLEX)

Para a criação, manipulação e gerenciamento de malhas não estruturadas, o PETSc oferece a classe DMPlex. O DMPlex representa a malha através de suas entidades topológicas fundamentais: vértices (pontos), arestas (linhas), faces (superfícies) e células (volumes). A vantagem dessa representação topológica é sua generalidade. O DMPlex gerencia as conexões entre todas as suas entidades, permitindo particionar de forma robusta a malha e gerenciar a comunicação de dados em ambientes paralelos.

3.2.5 Abstração para Elementos Finitos (PETScFE)

Enquanto o DMPlex gerencia a topologia da malha e a distribuição dos dados, o componente PETScFE (PETSc Finite Element) é utilizado para gerenciar a discretização matemática do problema. O PETScFE é responsável por criar o espaço de elementos finitos sobre a malha, definindo as funções de base e as regras de quadratura necessárias para a integração numérica das formas fracas.

A implementação do método dos elementos finitos com a interface `PETScFE` é baseada na definição de funções que representam cada termo da formulação fraca do problema (KNEPLEY *et al.*, 2013). Ao invés de montar a matriz de rigidez manualmente, o usuário fornece ao PETSc as equações que descrevem os integrandos da forma fraca, e a biblioteca se encarrega de realizar a quadratura numérica e a montagem.

Isso é feito através do objeto `PetscDS` (Discrete System), que armazena a formulação fraca do problema. A principal rotina para isso é a `PetscDSSetResidual()`, que associa uma função em C a um termo específico do resíduo da equação. Por exemplo, para um problema genérico, a chamada seria:

```
PetscCall(PetscDSSetResidual(ds, 0, f0_function, f1_function));
```

Neste contexto:

- `ds` é o objeto *Discrete System* que contém a formulação matemática.
- $f0\_function$: é uma função implementada pelo usuário que calcula o integrando dos termos da forma fraca que são multiplicados pela função-teste (v). No caso da equação de Poisson, esta função é responsável por definir o lado direito do sistema ($(\int_\Omega f v \, dx)$). Para isso, ela simplesmente retorna o valor do termo de fonte (f), ou seja, neste caso, a implementação de `f0_function` é ($f_0 = f$).
- `f1_function` é outra função do usuário que calcula o termo multiplicado pela derivada da função de forma, como $\int_\Omega \nabla v \cdot (D\nabla u)\, dx$ para o poisson. Nesse caso, a função faz referência ao gradiente de u, ($f_1 = D\nabla u$)

De forma análoga, para a montagem da matriz Jacobiana, o usuário fornece as funções que descrevem as derivadas de cada termo do resíduo em relação à solução u e ao seu gradiente ∇u (KNEPLEY *et al.*, 2013). Isso é feito através da rotina `PetscDSSetJacobian()`:

```
PetscCall(PetscDSSetJacobian(ds, nf, ng, g0, g1, g2, g3));
```

Os parâmetros `nf` e `ng` são os índices que especificam, respectivamente, o campo da função-teste e o campo da variável de solução. Em problemas de campo único, como os abordados neste trabalho, existe apenas uma variável principal e, portanto, um único campo, que possui o índice 0. Dessa forma, para todos os testes realizados, a Jacobiana foi definida utilizando `nf=0` e `ng=0`.

Os quatro argumentos de função (`g0` a `g3`) representam os quatro possíveis termos da Jacobiana da forma fraca, definidos como as derivadas dos termos do resíduo (`f0` e `f1`) em relação às variáveis (u e ∇u):

$$\texttt{g0}(u, \nabla u) = \frac{\partial f_0}{\partial u} \tag{3.11}$$

$$\texttt{g1}(u, \nabla u) = \frac{\partial f_0}{\partial \nabla u} \tag{3.12}$$

$$\texttt{g2}(u, \nabla u) = \frac{\partial f_1}{\partial u} \tag{3.13}$$

$$\texttt{g3}(u, \nabla u) = \frac{\partial f_1}{\partial \nabla u} \tag{3.14}$$

Para problemas lineares e estacionários, a matriz Jacobiana é a própria matriz de rigidez, que é constante. Em problemas lineares transientes, como o modelo monodomínio com um passo de tempo implícito, a matriz do sistema linear a ser resolvida a cada passo ($\mathbf{A} = \mathbf{M}\Delta t + \mathbf{K}$) também não depende da solução e, portanto, pode ser montada uma única vez.

Para otimizar a performance nesses casos, é fundamental evitar a remontagem e fatoração desnecessária desta matriz a cada iteração. No PETSc, esse comportamento pode ser forçado utilizando a função `SNESSetLagJacobian(snes, -2)`.

Esse comando instrui o SNES a calcular a Jacobiana apenas uma vez na sua primeira iteração e reutilizar essa matriz em todos os passos de tempo subsequentes. Essa estratégia reduz o custo computacional, visto que evita a repetição de um cálculo custoso dessa matriz.

3.3 Formulações Fracas

Conforme detalhado nas seções anteriores, a implementação de um problema via PETSc e o método dos elementos finitos requer a sua formulação fraca. Esta seção se dedica a derivar essa formulação para cada um dos problemas estudados. Para cada caso, será apresentada a equação diferencial parcial governante, o desenvolvimento de sua forma fraca, as condições de contorno aplicadas e, quando relevante, a solução exata e o termo de fonte correspondente utilizados para a verificação do código.

3.3.1 Equação de Poisson

A forma forte da Equação de Poisson, considerando um coeficiente de difusão D, é dada por:

$$-\nabla \cdot (D\nabla u) = f \quad \text{em } \Omega \tag{3.15}$$

Multiplicando por uma função-teste $v \in \mathcal{V}$ e aplicando a integração por partes, chega-se à forma fraca do problema: encontrar $u \in \mathcal{H}^1(\Omega)$ tal que

$$\int_\Omega D\nabla u \cdot \nabla v \, dx = \int_\Omega f v \, dx, \quad \forall v \in \mathcal{V} \tag{3.16}$$

Para este trabalho, assume-se uma condição de contorno de Dirichlet em toda a fronteira, o que faz com que o termo de fronteira advindo da integração por partes se anule pela escolha de $v = 0$ sobre $\partial\Omega$.

Para aferir métricas de erro, será proposta uma solução exata para essa equação, e será calculado um termo fonte a partir da aplicação do Método de Soluções Manufaturadas.

Para os testes, considerou-se o domínio como o quadrado unitário $\Omega = [0,1]\times[0,1]$ e um coeficiente de difusão constante e unitário ($D = 1$). A solução exata escolhida foi:

$$u_{exata} = sen(\pi x).sen(\pi y) \tag{3.17}$$

O termo de fonte f é então calculado aplicando-se o operador $-\Delta$ (o Laplaciano, pois $D = 1$) sobre u_{exata}:

$$f = 2\pi^2(sen(\pi x).sen(\pi y) \tag{3.18}$$

A condição de contorno de Dirichlet é imposta em toda a fronteira $\partial\Omega$, utilizando-se os valores da própria solução exata, que neste caso resulta em $u = 0$ em $\partial\Omega$.

3.3.2 Modelo Monodomínio

O modelo monodomínio, apresentado anteriormente, é descrito pela equação diferencial parcial de reação-difusão, em sua forma forte dada por:

$$\chi C_m \frac{\partial V}{\partial t} = \nabla.(\sigma\nabla V) - \chi I_{ion}(V,\eta) + I_{stim} \quad \text{em } \Omega, \tag{3.19}$$

Entretanto, para aplicá-la nesse trabalho foi realizada a estratégia do *operator splitting*, separando a resolução da EDP da resolução do sistema de EDOs relacionados ao termo $I_{ion}(V,\eta)$. Essa abordagem desacopla a equação em dois subproblemas, que são resolvidos sequencialmente a cada passo de tempo Δt:

1. **Difusão**: resolve-se a EDP de difusão para o potencial da membrana

$$\chi C_m \frac{\partial V}{\partial t} = \nabla \cdot (\sigma \nabla V) \quad \text{em } \Omega \tag{3.20}$$

Para isso, é necessário multiplicar (3.20) por uma função v e aplicar a integração por partes com o fim de obter a sua formulação fraca: encontrar $u \in \mathcal{H}^1(\Omega)$ tal que

$$\int_\Omega \chi C_m \frac{\partial V}{\partial t} v, dx + \int_\Omega \sigma \nabla V \cdot \nabla v, dx = \int_{\partial\Omega} (\sigma \nabla V \cdot \mathbf{n}) v, ds \quad \forall v \in \mathcal{V} \tag{3.21}$$

O problema é então suplementado com uma condição de contorno de fluxo nulo (Neumann homogênea), que simula um tecido eletricamente isolado. Esta condição é imposta sobre a solução V em toda a fronteira $\partial\Omega$ e é definida como:

$$(\sigma \nabla V) \cdot \mathbf{n} = 0 \quad \text{em } \partial\Omega \tag{3.22}$$

A aplicação da equação (3.22) faz com que a integral de fronteira na formulação fraca (equação (3.21)) se anule, resultando na forma final utilizada neste trabalho:

$$\int_\Omega \chi C_m \frac{\partial V}{\partial t} v, dx + \int_\Omega \sigma \nabla V \cdot \nabla v, dx = 0, \quad \forall v \in \mathcal{V} \tag{3.23}$$

Para a discretização da derivada temporal, empregou-se o esquema de Euler implícito, uma abordagem de diferenças finitas. Essa funcionalidade foi provida pelo módulo *TimeStepper* da biblioteca PETSc, sendo ativada através da opção de linha de comando `-ts_type beuler`.

Em todas as simulações realizadas nesse estudo foram utilizados os seguintes parâmetros: uma capacitância de membrana de $1.0 \mu F/cm^2$ e uma condutividade de tecido isotrópica de $\sigma = 10^{-4} S/cm$.

2. **Reação**: Com a solução da difusão em cada instante de tempo, atualiza-se o estado das variáveis celulares através da resolução do sistema de EDOs do modelo celular em cada ponto da malha:

$$\frac{dV}{dt} = -\frac{1}{C_m} \left(I_{\text{ion}}(V, \eta) - I_{\text{stim}} \right) \tag{3.24}$$

Essa separação permite o uso de métodos numéricos distintos e otimizados para cada parte do problema, como um método implícito para a EDP de difusão e um explícito para o sistema de EDOs. Além da otimização numérica, essa abordagem oferece uma grande vantagem em termos de modularidade, pois a implementação do modelo celular se torna independente da solução da difusão, facilitando a integração de novos modelos sem alterar a estrutura principal do código. Essa característica local do passo de reação, onde cada ponto da malha é tratado de forma independente, também o torna um candidato ideal para futuras acelerações em arquiteturas massivamente paralelas como as GPUs. Devido a essas vantagens, a separação de operadores é uma prática padrão na eletrofisiologia computacional, sendo a abordagem adotada por simuladores consolidados como o MonoAlg3D e o openCARP.

Passo de Reação: Modelo de Mitchell-Shaeffer

O termo de reação iônica (I_{ion}) foi modelado utilizando o modelo celular fenomenológico de Mitchell-Schaeffer (MITCHELL; SCHAEFFER, 2003). Conforme justificado anteriormente, este modelo foi escolhido por sua simplicidade, representando as dinâmicas essenciais do potencial de ação com apenas duas equações diferenciais ordinárias (EDOs). Para esse trabalho, foram utilizados os parâmetros com os valores listados na Tabela 3.1.

Tabela 3.1: Parâmetros do modelo celular de Mitchell-Schaeffer utilizados nas simulações.

Parâmetro	**Valor**
τ_{in}	0.3
τ_{out}	6.0
τ_{open}	120.0
τ_{close}	150.0
u_{gate}	0.13

Conforme a estratégia de separação de operadores, o sistema de EDOs deste modelo é resolvido em cada ponto da malha após a conclusão do passo de difusão em um determinado instante de tempo. Para o seu cálculo, foi utilizado o método de Euler Explícito, devido à sua simplicidade de implementação. Entretanto, a estabilidade desse método é fortemente dependente do tamanho do passo de tempo Δt. Assim, para garantir a estabilidade numérica, foi necessário adotar um valor suficientemente pequeno ($\Delta t = 0.02\,\text{ms}$), o que, neste trabalho, também define o passo de tempo global da simulação.

Embora a estratégia de *operator splitting* permita, em princípio, a utilização de passos distintos para a EDP e as EDOs, optou-se aqui por empregar um único Δt para ambas as etapas.

Implementação no PETSc

O desacoplamento entre a solução da EDP e das EDOs, promovido pela separação de operadores, permite que a implementação de cada etapa seja feita de forma modular. A parte da difusão (EDP) é implementada através das funções de resíduo e Jacobiana do `PETScFE`, como descrito na Seção 3.2.5.

Já a solução do sistema de EDOs do modelo celular é implementada em uma função auxiliar, aqui chamada de `SolveODEs`. Esta função é registrada no integrador temporal do PETSc através do comando `TSSetPostStage()`, para que seja executada sequencialmente a cada passo de tempo, logo após a etapa de difusão.

O funcionamento da rotina `SolveODEs` pode ser resumido nos seguintes passos:

1. **Acesso aos Dados:** A função obtém acesso ao vetor de solução global `V` (que contém o potencial de membrana atualizado pelo passo de difusão) e ao vetor local que armazena a variável de porta `h` do modelo de Mitchell-Schaeffer.

2. **Comunicação Paralela (Scatter):** É realizada uma operação de comunicação *scatter* (`DMGlobalToLocal`) para distribuir as porções relevantes do vetor global `V` para vetores locais em cada processo. Isso garante que cada processo tenha a informação necessária para realizar os cálculos.

3. **Cálculo Ponto a Ponto:** Em cada processo, a função itera sobre os nós da malha que lhe pertencem. Para cada nó, aplica-se um passo do método de Euler Explícito para resolver o sistema de EDOs do modelo de Mitchell-Schaeffer, atualizando os valores locais de `V` e `h`. Esta etapa é inerentemente paralelizável, pois o cálculo em um nó é independente dos outros.

4. **Atualização Global (Gather):** Após a conclusão dos cálculos locais, uma operação de comunicação reversa *gather* (`DMLocalToGlobal`) é executada para atualizar o vetor de solução global `V` com os novos valores calculados.

Dessa forma, a função `SolveODEs` executa o passo de reação da simulação, atualizando o estado celular de todo o domínio antes que o próximo passo de difusão seja calculado pelo PETSc.

Cenários de Simulação

Para avaliar o desempenho do simulador, foram definidos dois cenários de teste, representando a propagação de uma onda em tecidos 2D e 3D.

- **Onda Plana em Domínio 2D:** A simulação foi realizada em um domínio quadrado bidimensional com dimensões de $25000\mu m \times 25000\mu m$, discretizado com um espaçamento de $250\mu m$. Um estímulo elétrico $v = 1$ foi aplicado em uma faixa na lateral esquerda do domínio, definida por $x < 0.1$ cm ($1000\mu m$) como condição inicial, para gerar uma frente de onda plana.

- **Onda Radial em Domínio 3D:** O segundo cenário de simulação consistiu em dois testes tridimensionais para avaliar a acurácia e o desempenho em diferentes escalas.

  – O primeiro teste, focado na análise de convergência, foi realizado em um domínio retangular de 2.0 cm x 0.7 cm x 0.3 cm. Este cenário é uma adaptação do benchmark padrão de Niederer et al. (NIEDERER *et al.*, 2011a), com modificações para alinhar o teste aos objetivos deste trabalho. O modelo celular de ten Tusscher (TUSSCHER; PANFILOV, 2004) foi substituído pelo de Mitchell-Schaeffer, e a discretização da malha foi ajustada para os níveis de 100, 250 e 500 µm (em vez dos 200 µm originais) para garantir a existência de um ponto central comum em todas as malhas para a análise.

  – Adicionalmente, para analisar a escalabilidade paralela (*speedup*) em um problema de maior porte, foi conduzido um segundo teste. Este consistiu em uma simulação em um domínio cúbico de 4 cm de aresta (40000 µm), utilizando uma malha com discretização de 500 µm e elementos de terceira ordem (Q3), a fim de avaliar o desempenho do protótipo em um cenário computacionalmente mais intensivo.

Em ambos os casos, o estímulo elétrico foi aplicado em um canto do domínio, na região cúbica definida por $x, y, z < 1500\mu m$, para gerar uma frente de onda com propagação radial.

3.3.3 Equação do Calor Não Linear

Para validar a implementação de problemas transientes e não lineares, foi utilizado como problema-modelo a equação do calor não linear, apresentada em sua forma forte:

$$\frac{\partial u}{\partial t} - \nabla \cdot (D(u)\nabla u) = f \quad \text{em } \Omega \times (0, T] \tag{3.25}$$

A não linearidade do problema reside no coeficiente de difusão $D(u)$, que depende da própria solução u. Para os testes, foi escolhida a seguinte função para o coeficiente:

$$D(u) = 1 + u^2 \tag{3.26}$$

Para derivar a formulação fraca, a equação (3.25) é multiplicada por uma função-teste v e integrada sobre o domínio espacial Ω. Aplicando-se a integração por partes ao termo de difusão, obtém-se a seguinte forma fraca do problema: encontrar $u(t) \in \mathcal{H}^1(\Omega)$ tal que, para toda função-teste $v \in \mathcal{V}$,

$$\int_\Omega \frac{\partial u}{\partial t} v \, dx + \int_\Omega D(u)\nabla u \cdot \nabla v \, dx = \int_\Omega f v \, dx + \int_{\partial\Omega} (D(u)\nabla u \cdot \mathbf{n}) v \, ds \tag{3.27}$$

Para a verificação do código, foi novamente empregado o Método das Soluções Manufaturadas. Considerou-se o domínio como o quadrado unitário $\Omega = [0, 1] \times [0, 1]$ e a seguinte solução exata:

$$u_{\text{exata}}(x, y, t) = (1 + x + 2y).(e^{-sen(t)}) \tag{3.28}$$

O termo de fonte f é então obtido substituindo-se u_{exata} na forma forte da equação (3.25)).

$$f = (1 + x + 2y).(-e^{-t} - 10e^{-3t}) \tag{3.29}$$

Para que o problema transiente seja bem posto, a formulação fraca é suplementada por uma condição inicial e por condições de contorno, ambas derivadas diretamente da solução exata.

Neste problema de verificação, uma condição de contorno de Dirichlet é imposta sobre toda a fronteira do domínio ($\partial\Omega$). Como a condição de Dirichlet é válida em toda a fronteira, temos que $v = 0$ em $\partial\Omega$.

Essa restrição faz com que o termo de integral de fronteira, que surge da integração por partes, se anule:

$$\int_{\partial\Omega} (D(u)\nabla u \cdot \mathbf{n}) v \, ds = 0, \quad \text{pois } v = 0 \text{ em } \partial\Omega \tag{3.30}$$

Isso simplifica a formulação fraca para a forma final a ser resolvida, descrita na equação (3.31):

$$\int_{\Omega} \frac{\partial u}{\partial t} v \, dx + \int_{\Omega} D(u)\nabla u \cdot \nabla v \, dx = \int_{\Omega} f v \, dx \tag{3.31}$$

Para a aproximação desse problema foi feita uma discretização do domínio com 2500 elementos e $\Delta t = 0.1s$.

3.4 Métricas de Avaliação

Como o objetivo deste estudo é realizar uma análise quantitativa da biblioteca PETSc, é necessária a definição de um conjunto de métricas para avaliar tanto a corretude da implementação quanto seu desempenho computacional.

Para validar a corretude da implementação do método dos elementos finitos nos problemas onde uma solução exata é conhecida (via MMS), foi utilizada a norma L2 do erro. Esta métrica quantifica a diferença global entre a solução numérica (u_h) e a solução exata (u_{exata}) através da seguinte integral sobre o domínio Ω:

$$|e|_{L_2} = \sqrt{\int \Omega (u_{\text{exata}} - u_h)^2, dx} \tag{3.32}$$

A norma L2 do erro é fundamental para a realização de estudos de convergência, nos quais se verifica se a taxa de redução do erro, com o refinamento da malha, corresponde

à taxa teórica prevista pelo método.

Além disso, para avaliar a convergência dos métodos iterativos de solução linear e não linear, serão contabilizadas as iterações do KSP e do SNES. Buscando avaliar o impacto da natureza do problema e da aplicação de pré-condicionadores nos respectivos resolvedores.

Para verificar a eficiência da solução em ambientes de computação paralela, serão levantadas as métricas de *speedup* e de eficiência paralela (TROBEC *et al.*, 2018). O *speedup* mede o ganho de desempenho obtido ao utilizar p processadores em comparação com a execução serial:

$$S_p = \frac{T_1}{T_p} \tag{3.33}$$

onde T_p é o tempo de execução com p processadores. O *speedup* ideal é linear ($S_p = p$). Para normalizar essa medida, utiliza-se a Eficiência Paralela (E_p), que avalia a qualidade da paralelização:

$$E_p = \frac{S_p}{p} \tag{3.34}$$

Uma queda na eficiência com o aumento de p geralmente indica que o custo relativo da comunicação entre os processos está se tornando significativo em comparação com o trabalho computacional (TROBEC *et al.*, 2018).

3.5 Ambiente Computacional

Os experimentos de desempenho e escalabilidade foram executados em nós de um cluster cujas especificações de hardware e software são detalhadas na Tabela 3.2, garantindo assim a reprodutibilidade dos resultados.

Tabela 3.2: Especificações do Ambiente Computacional.

Componente	**Especificação**
Máquina 1	
Processadores	2 x AMD Opteron 6272 @ 2.1 GHz (Total: 32 núcleos)
Memória	31 GiB de RAM
Rede de Interconexão	Infiniband
Máquina 2	
Processadores	2 x AMD EPYC 7713 @ 3.67 GHz (Total: 128 núcleos)
Memória	503 GiB de RAM
Rede de Interconexão	Infiniband
Software	
Sistema Operacional	Rocky Linux 9.6
Compilador	GCC 13.2.0
Biblioteca MPI	MPICH 4.3.0
Biblioteca PETSc	Versão 3.23.3
Biblioteca Hypre	Versão 2.32.0

4 Resultados

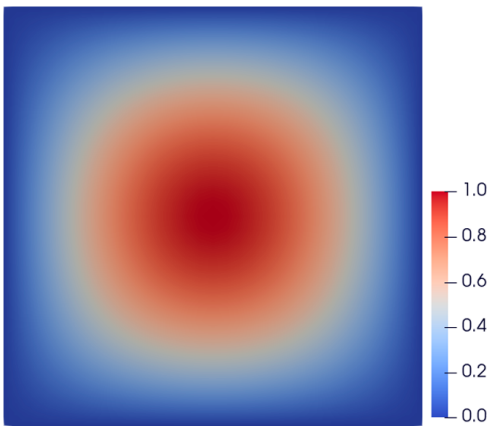
Este capítulo apresenta os resultados obtidos a partir da aplicação da metodologia descrita anteriormente. A análise está dividida por problema-teste, iniciando-se com a verificação da implementação e seguindo para os estudos de caso e performance.

4.1 Equação de Poisson

A primeira etapa da análise de resultados consiste em verificar a corretude da implementação do método dos elementos finitos. Para isso, foi utilizado o problema da equação de Poisson com uma solução manufaturada, conforme detalhado na Metodologia (Seção 3.3.1).

A Figura 4.1 apresenta a solução numérica obtida em uma malha de 32×32 elementos. Observa-se que a solução aproximada captura qualitativamente o comportamento da solução exata escolhida, com os valores máximos (em vermelho) e mínimos (em azul) localizados corretamente no domínio.

Figura 4.1: Solução aproximada da equação de Poisson.



Fonte: O autor.

Para uma validação quantitativa, foi realizado um estudo de convergência, onde a simulação foi executada em uma sequência de malhas sistematicamente refinadas (8×8, 16×16, 32×32, etc.). Foram utilizados elementos de diferentes ordens (Q1, Q2, Q3 e Q4), que correspondem ao uso de polinômios de aproximação de grau um (linear), dois

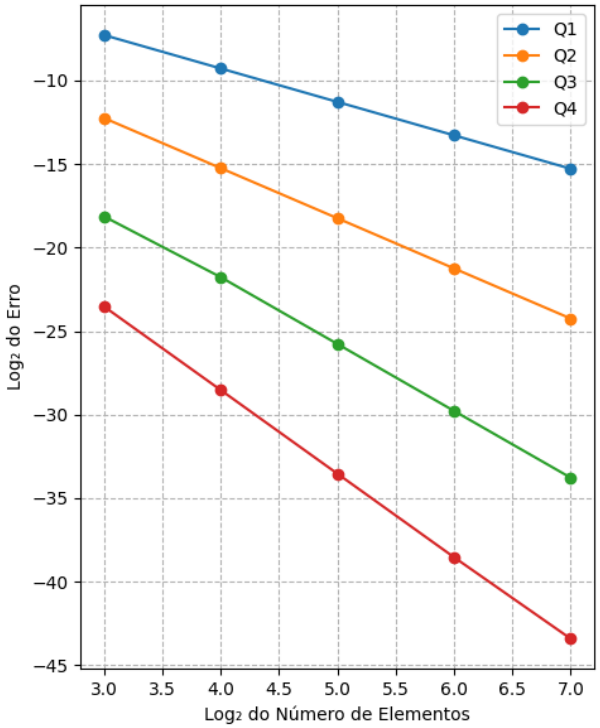
(quadrático), três (cúbico) e quatro (quártico), respectivamente. O erro entre a solução numérica e a exata foi calculado utilizando a norma L2. A Tabela 4.1 compila os resultados numéricos, incluindo a ordem de convergência do erro (p) calculada, que teoricamente deve seguir a relação $p = k + 1$.

Tabela 4.1: Resultados do estudo de convergência para a equação de Poisson com elementos Q1, Q2, Q3 e Q4.

Malha	**Q1 (k=1)**		**Q2 (k=2)**		**Q3 (k=3)**		**Q4 (k=4)**	
	Erro L2	**p**	**Erro L2**	**p**	**Erro L2**	**p**	**Erro L2**	**p**
8x8	6.41E-03	-	2.05E-04	-	4.46E-06	-	8.23E-08	-
16x16	1.60E-03	2.00	2.50E-05	2.99	5.10E-07	4.00	8.00E-09	5.00
32x32	4.01E-04	1.99	3.13E-06	3.00	3.20E-08	3.99	2.50E-10	5.00
o height64x64	1.00E-04	2.00	3.92E-07	3.00	2.00E-09	4.00	7.81E-12	5.00
128x128	2.50E-05	2.00	4.90E-08	3.00	1.25E-10	4.00	2.44E-13	5.00

A Figura 4.2 visualiza esses dados em um gráfico log-log do erro L2 em função do número de elementos utilizados para discretizar o domínio. Neste tipo de gráfico, a inclinação da reta corresponde à ordem de convergência.

Figura 4.2: Gráfico de convergência do erro da equação de Poisson.



A análise dos resultados confirma que a implementação está correta. A Tabela 4.1 mostra que a ordem de convergência (p) calculada para os elementos Q1 aproxima-se de 2, enquanto para os elementos Q2 aproxima-se de 3 e assim por diante. Este

comportamento está em excelente concordância com a taxa teórica esperada de $p = k+1$, para um elemento Q_k, validando quantitativamente o código desenvolvido.

É importante ressaltar que a mudança entre os diferentes graus de elemento foi realizada através de uma única opção de linha de comando (`-petscspace_degree`), sem qualquer alteração no código ou na malha, o que demonstra a flexibilidade fornecida pela infraestrutura do PETSc.

4.2 Modelo Monodomínio

Esta seção apresenta os resultados obtidos com o protótipo de simulador utilizando o monodomínio. Conforme detalhado no capítulo de Metodologia (Seção 3.3.2), foi implementado um solucionador para a equação de reação-difusão utilizando a biblioteca PETSc.

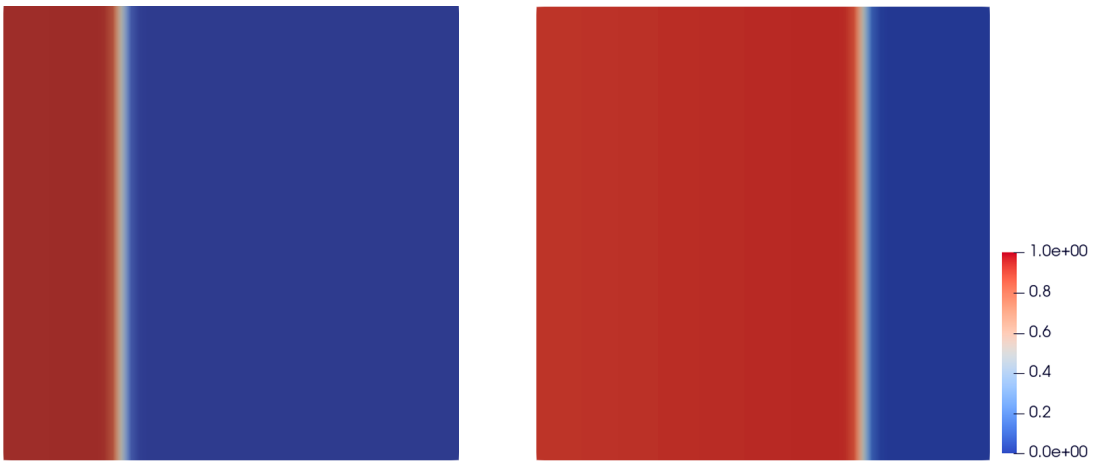
4.2.1 Onda Plana

O primeiro cenário de teste avalia a simulação de uma onda plana no domínio 2D. A análise se inicia com uma validação qualitativa do protótipo, para verificar se o comportamento físico do fenômeno é reproduzido corretamente, seguida por uma investigação quantitativa do desempenho do solucionador e da escalabilidade paralela.

Primeiramente, a Figura 4.3 apresenta a validação visual do simulador. Os instantâneos da simulação (Figura 4.3a e Figure 4.3b) demonstram a propagação de uma frente de onda uniforme, como esperado. Nessas imagens, vemos uma graduação de cor que indica o potencial transmembrânico das células, variando de 0 (em azul) até 1 (em vermelho).

Adicionalmente, o gráfico do potencial de ação para o ponto central do domínio (Figura 4.4) confirma que o simulador reproduz com sucesso a morfologia característica do modelo de Mitchell-Schaeffer.

Para analisar o impacto dos pré-condicionadores na convergência do método do Gradiente Conjugado, a simulação foi executada com três configurações distintas: sem pré-condicionador, com o pré-condicionador Jacobi (nativo do PETSc) e com ILU (via Hypre).



(a) t = 0.4 ms (b) t = 1.2 ms

Figura 4.3: Propagação da onda plana no domínio 2D.

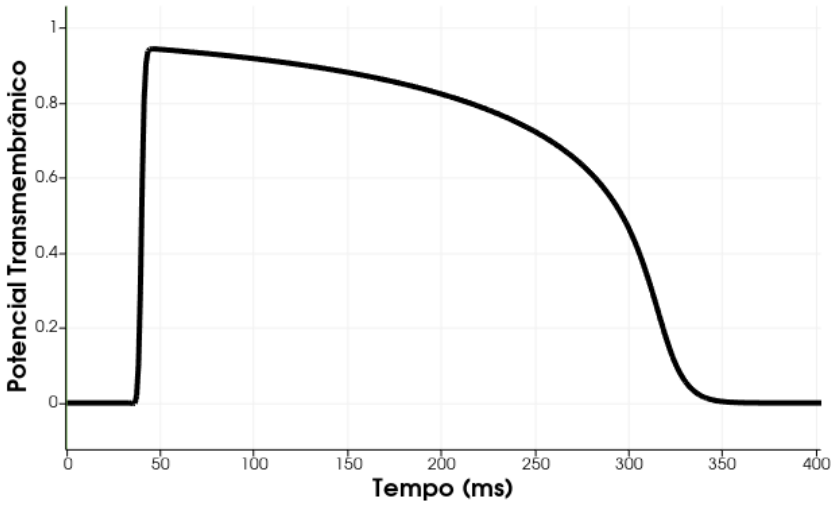


Figura 4.4: Potencial de ação de ponto no centro do domínio.

Esta análise foi repetida tanto para elementos finitos lineares (Q1) quanto quadráticos (Q2). A Tabela 4.2 resume o número médio de iterações do KSP necessárias para a convergência a cada passo de tempo em cada cenário.

Os resultados na Tabela 4.2 demonstram que o pré-condicionador ILU oferece a maior redução do número de iterações para ambos os tipos de elementos. Além disso, nota-se que a mudança de grau polinomial não apresenta grande impacto no número de iterações dos pré-condicionadores.

Para analisar a escalabilidade da implementação, foi realizado um teste de escalabilidade forte na Máquina 1 (Tabela 3.2). A metodologia consistiu em fixar o tamanho do problema em $25mm \times 25mm$, discretizado com 50×50 elementos, e variar o número de processadores. Para garantir a robustez estatística dos resultados, cada cenário foi

Tabela 4.2: Análise de pré-condicionadores com base no número de iterações do KSP

Elemento	**Pré-condicionador**	**Nº Médio de Iterações**
Q1 (Linear)	Nenhum	17.20
	Jacobi	7.71
	ILU (Hypre)	1.99
Q2 (Quadrático)	Nenhum	18,97
	Jacobi	7.54
	ILU (Hypre)	1.99

executado cinco vezes, e o tempo de execução reportado corresponde à média aritmética destes testes, com o desvio padrão permanecendo inferior a $1,5\%$ da média em todos os casos. As métricas de *speedup* e eficiência foram então calculadas com base neste tempo médio. Os testes foram realizados para todas as configurações de pré-condicionador e tipo de elemento, a fim de avaliar não apenas o desempenho geral, mas também o impacto de cada componente na performance paralela.

Tabela 4.3: Resultados de tempo e *speedup* para o caso 2D (Elementos Q1).

Nº de	**ILU (Hypre)**		**Jacobi**		**Nenhum**	
Proc.	**Tempo (s)**	**Speedup**	**Tempo (s)**	**Speedup**	**Tempo (s)**	**Speedup**
1	3055.32 ± 45.78	1.0	3161.86 ± 47.43	1.0	3387.93 ± 50.82	1.0
2	1773.79 ± 23.93	1.72	1686.35 ± 22.88	1.87	1659.32 ± 22.10	2.00
4	923.99 ± 10.16	3.31	939.06 ± 10.40	3.37	932.42 ± 10.25	3.63
6	689.40 ± 7.23	4.43	695.57 ± 7.31	4.55	694.87 ± 7.30	4.88
8	553.20 ± 5.81	5.52	536.42 ± 5.63	5.89	557.91 ± 5.87	6.07
10	462.98 ± 4.58	6.60	458.05 ± 4.52	6.90	474.21 ± 4.71	7.14
12	410.54 ± 4.09	7.44	393.14 ± 3.93	8.04	421.34 ± 4.21	8.04

Tabela 4.4: Resultados de tempo e *speedup* para o caso 2D (Elementos Q2).

Nº de	**ILU (Hypre)**		**Jacobi**		**Nenhum**	
Proc.	**Tempo (s)**	**Speedup**	**Tempo (s)**	**Speedup**	**Tempo (s)**	**Speedup**
1	5917.69 ± 88.75	1.0	6243.62 ± 93.65	1.0	6709.95 ± 100.65	1.0
2	3324.55 ± 46.54	1.78	3280.76 ± 45.93	1.90	3429.03 ± 48.10	1.96
4	1923.05 ± 21.15	3.08	1729.40 ± 19.10	3.61	1907.22 ± 20.55	3.52
6	1369.51 ± 15.42	4.32	1261.40 ± 14.20	4.95	1360.68 ± 15.28	4.93
8	1080.27 ± 12.42	5.48	1055.58 ± 12.12	5.91	1103.74 ± 12.65	6.08
10	917.26 ± 10.10	6.45	881.43 ± 9.74	7.08	939.83 ± 10.38	7.14
12	804.97 ± 8.45	7.35	766.76 ± 8.05	8.14	824.86 ± 8.65	8.13

A análise dos resultados de escalabilidade, apresentados nas Tabelas 4.3 e 4.4, revela um importante balanço (*trade-off*) entre o custo de cada pré-condicionador, seu poder de convergência e sua eficiência em ambiente paralelo.

Observando os tempos de execução absolutos, nota-se um resultado significativo:

para ambos os tipos de elemento (Q1 e Q2), o pré-condicionador Jacobi apresentou o menor tempo de solução na maioria das configurações. Embora a análise de iterações tenha mostrado que o ILU é muito superior na redução do número de passos do KSP, o custo computacional de aplicar este pré-condicionador mais complexo a cada iteração foi, para este problema, maior do que o benefício obtido. O Jacobi, por sua vez, representa um meio-termo eficaz, onde seu baixo custo de aplicação compensa um número maior de iterações em comparação com o ILU.

Embora a configuração sem pré-condicionador tenha sido a mais lenta em tempo de execução absoluto, ela demonstrou o comportamento de escalabilidade um pouco melhor para a maioria dos casos, especialmente no caso Q2. Isso ocorre porque este método possui o menor overhead de comunicação e sincronização por iteração, fazendo com que a adição de processadores resulte em um ganho de velocidade mais linear.

Finalmente, ao comparar os resultados entre os elementos Q1 e Q2, percebe-se que as simulações com elementos quadráticos (Q2) apresentaram uma escalabilidade ligeiramente superior. Este comportamento é esperado, pois a maior carga computacional dos elementos Q2 ajuda a "mascarar" a latência da comunicação, melhorando a razão entre computação e comunicação, o que resulta em uma eficiência paralela mais alta.

Em suma, para o problema da onda plana 2D neste ambiente, a estratégia de usar um pré-condicionador leve como o Jacobi ofereceu a melhor performance global.

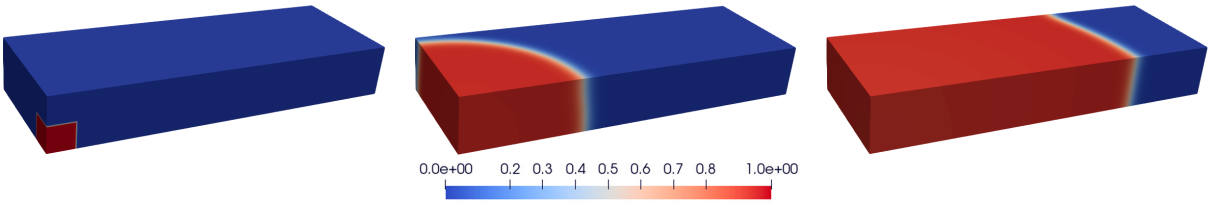
4.2.2 Domínio 3D

O segundo cenário de simulação eleva a complexidade para um domínio tridimensional, reproduzindo o experimento definido por um benchmark de eletrofisiologia (NIEDERER *et al.*, 2011a) com algumas modificações, descritas na seção (3.3.2). O objetivo principal desta análise é comparar a acurácia de elementos finitos de diferentes ordens (Q1, Q2 e Q3) na captura do tempo de ativação em pontos distantes da origem do estímulo.

Contudo, antes da análise quantitativa, a Figura 4.5 apresenta a validação qualitativa da simulação. Os instantâneos da simulação em três momentos distintos demonstram o comportamento esperado: um estímulo aplicado em um dos cantos do domínio gera uma frente de onda com propagação radial, que avança de forma aproximadamente

esférica até interagir com as fronteiras do bloco. Esta evolução consistente valida a correta implementação do modelo para o cenário 3D.

Figura 4.5: Evolução da frente de onda radial a partir do estímulo aplicado no canto do domínio 3D.



(a) t = 0 ms (b) t = 20 ms (c) t = 50 ms

Fonte: O autor.

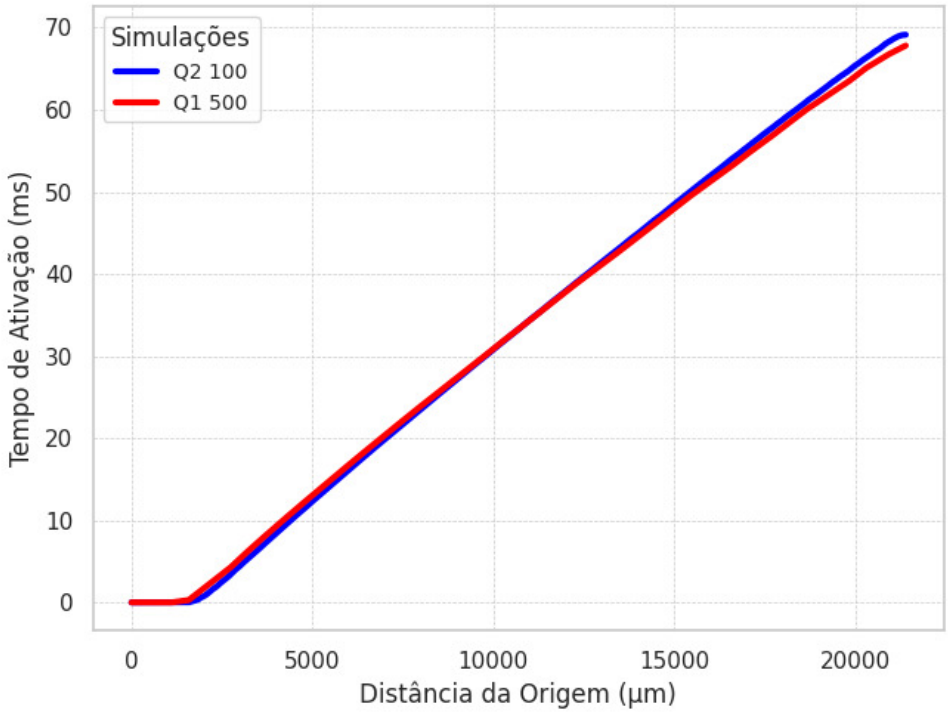
Com a validação qualitativa estabelecida, a análise quantitativa foca na comparação entre o refinamento de malha (refinamento-h) e o aumento do grau do polinômio de aproximação (refinamento-p). A expectativa teórica é que, ao refinar a malha (diminuir o tamanho h do elemento), a solução numérica convirja para o valor real. A hipótese a ser investigada é se o aumento do grau do elemento (e.g., de Q1 para Q2) pode fornecer uma acurácia similar ou superior a um refinamento de malha, porém com um custo computacional menor.

Para esta análise, a métrica de interesse foi o tempo de ativação, medido sobre pontos na diagonal do domínio. Como não há solução analítica para este problema, uma solução de referência foi gerada a partir de uma simulação com alto grau de refinamento (elementos Q2 com discretização de 100 µm). Cada simulação-teste foi executada por um tempo total de $500ms$. O erro de cada simulação-teste é então calculado em relação a essa referência como apresentado na Figura 4.6 para o caso de teste com discretização de $500\mu m$ com elementos de primeira ordem.

A Figura 4.7 ilustra a convergência do erro para as diferentes configurações testadas. Complementarmente, a Tabela 4.5 apresenta o custo computacional de cada simulação, reportado como o tempo de execução medido na Máquina 1 (Tabela 3.2).

A análise quantitativa, apresentada na Figura 4.7 e na Tabela 4.5, permite extrair conclusões importantes sobre a eficiência das estratégias de refinamento. Para os elementos Q1, observa-se a convergência esperada: ao refinar a malha, o erro no tempo de ativação diminui consistentemente, porém com um custo computacional que cresce

Figura 4.6: Comparação entre tempos de ativação na diagonal do domínio.



Fonte: O autor.

Tabela 4.5: Custo computacional e número de graus de liberdade (DoF) para as diferentes configurações no benchmark 3D.

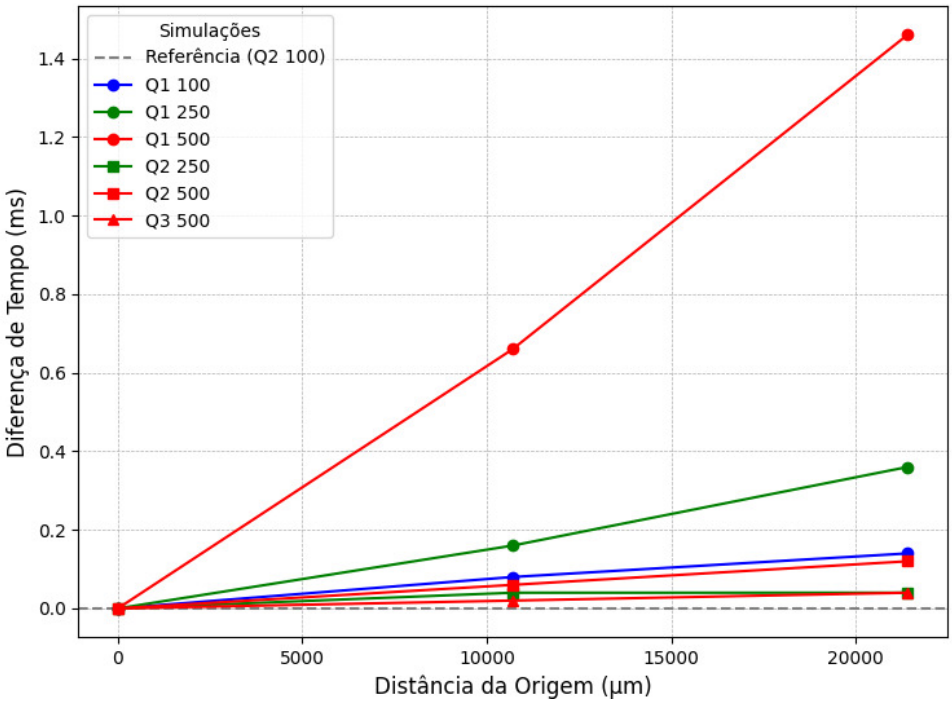
Elemento	**Discretização (μm)**	**Nº de Graus de Liberdade (DoF)**	**Tempo (s)**
Q1	500 µm	4305	316
Q2	500 µm	30537	1304
Q1	250 µm	30537	1319
Q3	500 µm	98767	6653
Q2	250 µm	229425	9058
Q1	100 µm	442281	18946
Q2	100 µm (Referência)	3448581	186117

drasticamente, chegando a superar 5 horas de execução.

O resultado mais significativo, contudo, é a comparação entre a acurácia e o custo observada na Figura 4.8. A simulação com elementos Q2 na malha de 250 µm não só alcançou um erro menor que o da simulação com Q1 na malha mais fina de 100 µm (erro de 0.04 contra 0.14 na diagonal), como o fez utilizando aproximadamente metade do tempo computacional (9058s contra 18946s).

A vantagem do refinamento-p é ainda mais evidente no caso dos elementos Q3: a simulação na malha mais grossa (500 µm) obteve uma acurácia comparável à da simulação Q2 de 250µm, mas com um custo computacional quase três vezes menor que o da melhor

Figura 4.7: Erro no tempo de ativação de elementos da diagonal para cada discretização e polinômio de aproximação.



Fonte: O autor.

simulação Q1 (6653s contra 18946s).

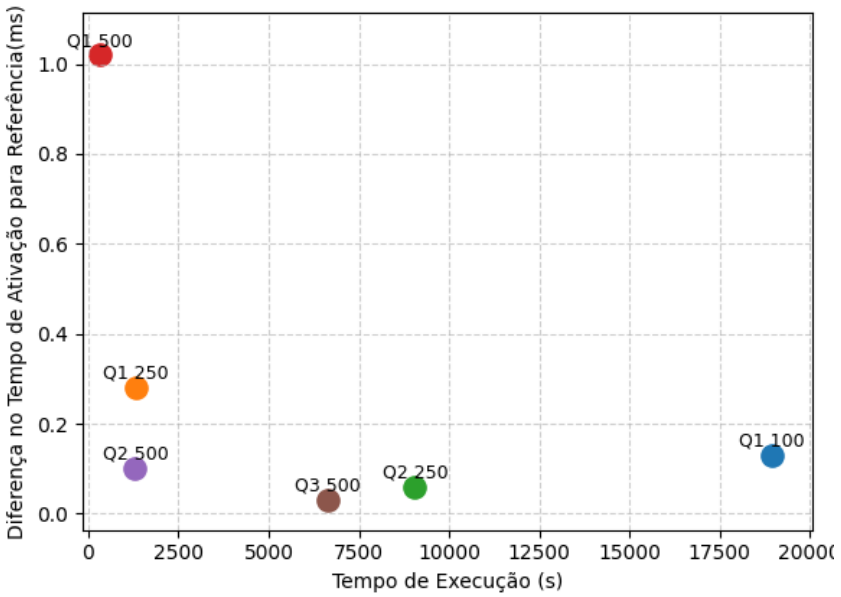
Esses resultados demonstram de forma quantitativa que, para este problema, o aumento do grau do polinômio de aproximação é uma estratégia computacionalmente mais eficiente para a obtenção de resultados acurados do que o refinamento extensivo da malha com elementos de baixa ordem.

Para avaliar o desempenho do protótipo em um cenário computacionalmente mais intensivo, foi conduzido um teste de escalabilidade com um tempo de simulação de $10ms$ em um domínio cúbico de 4 cm de aresta descrito na Seção 3.3.2, utilizando elementos de terceira ordem (Q3) de aresta $500\mu m$ (configuração de melhor custo-benefício observado na Tabela 4.5). O objetivo foi analisar a performance dos pré-condicionadores Jacobi e ILU em um problema com um número significativamente maior de graus de liberdade (13997521).

A Tabela 4.6 compila os resultados de tempo de execução, *speedup* e eficiência paralela para até 128 processadores na Máquina 2 descrita na Tabela 3.2.

Os resultados deste teste sugerem uma inversão no comportamento de desempenho observado nos cenários anteriores. Para este problema, maior e computacionalmente

Figura 4.8: Gráfico comparando a diferença no tempo de ativação em relação a solução de referência com o tempo de execução para obtenção desse resultado.



Fonte: O autor.

Tabela 4.6: Resultados de escalabilidade para o benchmark 3D em larga escala (4cm, Q3).

Nº de	**ILU (Hypre)**			**Jacobi**		
Proc. (p)	**Tempo (s)**	**Speedup**	**Efic.**	**Tempo (s)**	**Speedup**	**Efic.**
1	11449.88	1.00	1.00	81036.71	1.00	1.00
2	5753.71	1.99	1.00	40927.63	1.98	0.99
4	2862.52	4.00	1.00	21202.70	3.82	0.96
8	1567.63	7.30	0.91	11439.40	7.08	0.89
16	732.42	15.63	0.98	5799.02	13.97	0.87
32	429.14	26.68	0.83	3139.79	25.81	0.81
64	355.57	32.20	0.50	2929.85	27.66	0.43
128	262.74	43.59	0.34	2198.64	36.86	0.29

mais denso, o pré-condicionador ILU se mostrou significativamente mais rápido em tempo de execução absoluto em todas as contagens de processadores.

A razão para essa inversão fica clara ao se analisar o número médio de iterações do KSP por passo de tempo: enquanto a configuração com ILU convergiu em apenas 2.3 iterações, a com Jacobi exigiu 336.64 iterações. A complexidade do ILU, que era um overhead nos problemas menores, torna-se uma vantagem decisiva aqui. O custo de realizar mais de 300 iterações com Jacobi a cada passo de tempo se torna o principal gargalo, superando em muito o custo de aplicação do pré-condicionador ILU, que é mais robusto. Este resultado evidencia que a escolha do pré-condicionador ótimo é altamente dependente da escala e da complexidade do problema a ser resolvido.

4.3 Equação do Calor Não Linear

A última análise de caso de teste foca na capacidade do framework de resolver problemas transientes com não linearidade no termo de difusão, utilizando a equação do calor não linear como modelo. Conforme detalhado na Metodologia, este problema exige a aplicação de um solucionador para sistemas não lineares a cada passo de tempo. O objetivo desta seção é avaliar a eficiência e a robustez do solucionador SNES do PETSc para esta tarefa.

O termo fonte calculado através do MMS a partir da solução exata exposta na seção de metodologia implica em simulação oscilatória no tempo como demonstrado pela Figura 4.9

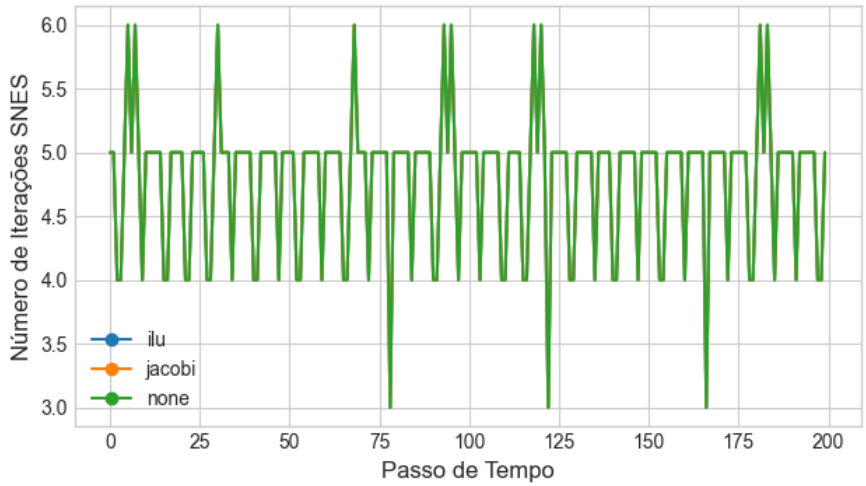
(a) t = 0 (b) t = 5 (c) t = 8.5 ms

Figura 4.9: Solução oscilatória de equação de calor não linear.

Para quantificar o desempenho, foi monitorado o número de iterações do SNES (utilizando o método de Newton com *line search*, `SNESNEWTONLS`) necessárias para atingir a convergência a cada passo de tempo da simulação. A simulação foi executada com um passo de tempo $\Delta t = 0.1s$, em uma malha bidimensional com 2500 elementos, e uma tolerância de convergência para o SNES de 10^{-8} na norma do resíduo. A Figura 4.10 apresenta a evolução deste número de iterações ao longo do tempo.

A Figura 4.10 indica a robustez do solucionador não linear. O número de iterações do SNES por passo de tempo permaneceu na faixa de 3 a 6, independentemente do precondicionador utilizado para o KSP, indicando que o método de Newton convergiu de forma rápida e estável, mesmo com a presença da não linearidade no coeficiente de difusão e uma solução oscilatória. Essa performance sugere que a abordagem do PETSc é uma base sólida para a simulação de problemas mais complexos, como os de eletromecânica cardíaca.

Figura 4.10: Número de iterações do SNES por passo de tempo para a solução da equação do calor não linear.



Fonte: O autor.

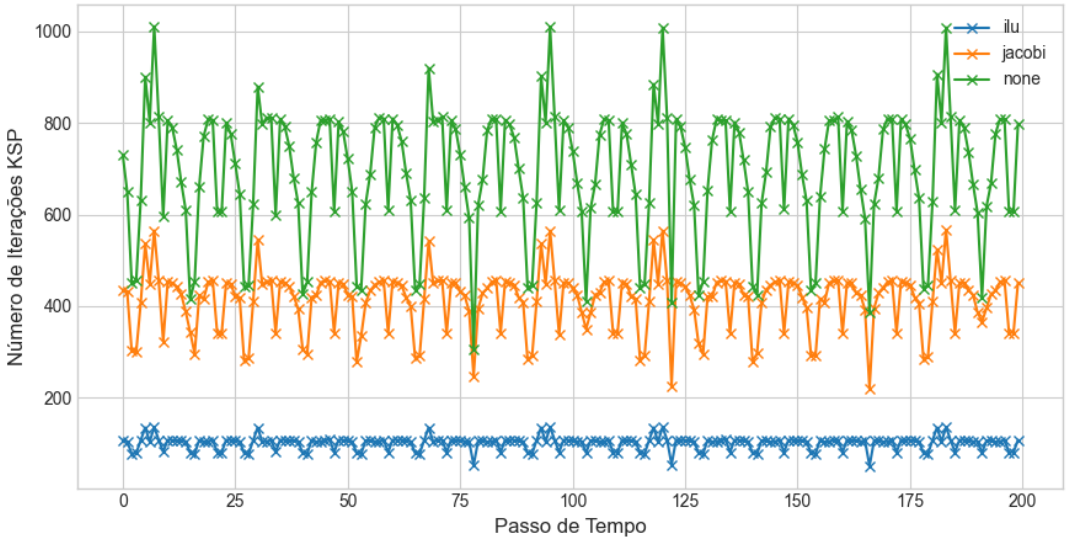
Tabela 4.7: Comparativo de desempenho dos pré-condicionadores para a equação do calor não linear.

Pré-condicionador	**Tempo de Execução (s)**	**Total de Iterações do KSP**
ILU (Hypre)	23.12	20242
Jacobi	27.99	82412
Nenhum	32.45	138717

Complementando a análise, a Figura 4.11 detalha o desempenho do solucionador linear (KSP), que é invocado a cada iteração do SNES. Observa-se que o número de iterações do GMRES não é constante, mas varia ao longo da simulação, refletindo a natureza oscilatória do problema. Em momentos de maior variação da solução, a solução do sistema se torna numericamente mais complexa, exigindo um maior número de iterações do solucionador linear. Adicionalmente, o gráfico confirma o impacto esperado dos pré-condicionadores: a configuração com ILU consistentemente exigiu o menor número de iterações, seguida pela Jacobi e, por fim, pela abordagem sem pré-condicionador (gráfico legendado com "none" em Figura 4.11), que apresentou o maior custo iterativo.

O impacto dessa redução de iterações no tempo de execução total é quantificado na Tabela 4.7. Apesar do maior custo computacional por iteração, a abordagem com o pré-condicionador ILU resultou no menor tempo de solução absoluto (23.12 s), sendo aproximadamente 17% mais rápida que a com Jacobi e 28% mais rápida que a sem pré-condicionador. Este resultado demonstra que, para problemas não lineares onde o sistema

Figura 4.11: Número de iterações do GMRES por passo de tempo para solução da equação do calor não linear com diferentes pré-condicionadores.



Fonte: O autor.

linear interno é resolvido múltiplas vezes, a escolha de um pré-condicionador robusto como o ILU é crucial para a eficiência computacional, pois o ganho obtido pela drástica redução no número total de iterações supera o seu custo de aplicação.

5 Conclusões

O presente trabalho teve como principal objetivo avaliar a biblioteca PETSc como uma ferramenta para aprimoramento de simuladores de eletrofisiologia do coração em ambientes paralelos usando o método dos elementos finitos.

Para atingir esse objetivo, primeiramente foi avaliada a implementação do DMPLEX e do PETScFE em relação à taxa de convergência teórica através da resolução da equação de Poisson com diferentes graus para os polinômios de aproximação. Os resultados mostraram concordância com as taxas esperadas descritas na literatura teórica sobre elementos finitos.

Em sequência foi analisado o impacto de dois pré-condicionadores - o Jacobi, implementado nativamente no PETSc, e a decomposição LU incompleta (ILU), implementada na biblioteca Hypre - sobre o número de iterações do método do Gradiente Conjugado resolvendo o problema da eletrofisiologia cardíaca 2D e o seu respectivo tempo de execução. Foi observado que ao aplicar o pré-condicionador ILU houve a maior redução no número de iterações e o menor tempo de execução serial, porém o Jacobi apresentou o maior *speedup* e eficiência paralela, atingindo tempos de execução inferiores com mais processadores, se comparado com o ILU.

Utilizando o protótipo de simulador de eletrofisiologia, também foi possível realizar uma análise do aumento do custo computacional ao refinar o domínio do problema ou aumentar o grau do polinômio de aproximação. Foram comparados os tempos de ativação em pontos da diagonal principal do domínio e observou-se que, utilizando polinômios de grau de aproximação maiores, é possível diminuir o erro obtido com uma malha mais grosseira e em muito menos tempo.

Adicionalmente, foi conduzido um teste de escalabilidade em um domínio 3D de larga escala, com aproximadamente 14 milhões de graus de liberdade, para avaliar os pré-condicionadores em um cenário computacionalmente mais intensivo. Neste caso, o pré-condicionador ILU demonstrou um desempenho superior em todas as métricas avaliadas. A simulação com ILU foi significativamente mais rápida em tempo de execução absoluto

e também apresentou um *speedup* mais elevado em comparação com o Jacobi. A razão para essa superioridade fica clara ao se analisar o número médio de iterações do KSP por passo de tempo: enquanto a configuração com ILU convergiu em apenas 2.3 iterações, a com Jacobi exigiu 336.64 iterações. O alto custo iterativo do Jacobi tornou-se o principal gargalo da simulação, evidenciando a eficácia do pré-condicionador ILU para problemas de grande porte.

Por fim, a robustez do framework foi confirmada com a solução da equação do calor não linear, na qual o solucionador SNES convergiu com um número baixo e estável de iterações, indicando que a ferramenta possui potencial de aprimoramento não apenas de simuladores de eletrofisiologia, mas também de problemas mecânicos e eletromecânicos do coração.

Em suma, é possível então concluir que o PETSc é uma poderosa ferramenta para o aprimoramento de simuladores de eletrofisiologia cardíaca, fornecendo uma grande gama de ferramentas e um alto grau de personalização. Além disso, vale ressaltar o observado em relação à ausência de uma configuração perfeita para todos os casos, visto que ocorre geralmente um balanço entre performance, número de iterações do método e qualidade da solução.

Reconhece-se que este estudo possui limitações, como o uso de um modelo celular fenomenológico e a exploração de um subconjunto das ferramentas disponíveis. Tais limitações, contudo, abrem caminho para trabalhos futuros. Sugere-se a aplicação da metodologia a modelos celulares biofisicamente detalhados, a utilização de malhas que representem a anatomia real do coração humano e a realização de testes incluindo análises de escalabilidade fraca. O protótipo desenvolvido e a validação do solucionador não linear servem como uma base sólida para a futura implementação do acoplamento eletromecânico, um passo importante para a criação de modelos cardíacos mais completos e realistas.

Bibliografia

AMANFU, R. K.; SAUCERMAN, J. J. Cardiac models in drug discovery and development: A review. *Critical Reviews™ in Biomedical Engineering*, 2011.

BALAY, S. *et al. PETSc Users Manual*. [S.l.], 2021. Accessed: 2025-01-23. Disponível em: ⟨https://petsc.org/release/⟩.

BOURGAULT, Y.; PIERRE, C. Comparing the bidomain and monodomain models in electro-cardiology through convergence analysis. *HAL*, 2010.

CAMPOS, F. O. *et al.* Characterizing the clinical implementation of a novel activation-repolarization metric to identify targets for catheter ablation of ventricular tachycardias using computational models. *Computers in Biology and Medicine*, 2019.

CAMPOS, J. O. *et al.* Preconditioned augmented lagrangian formulation for nearly incompressible cardiac mechanics. *Int J Numer Method Biomed Eng.*, 2018.

DARBAR, D.; RODEN, D. M. Genetic mechanisms of atrial fibrillation: impact on response to treatment. *Nat Rev Cardiol.*, 2013.

GERARDO-GIORDA, L. *An Introduction to Mathematical and Numerical Modeling of Heart Electrophysiology.* [S.l.]: Springer, Cham, 2016.

HODKIN, A. L.; HUXLEY, A. F. A quantitative description of membrane current and its aplication to conduction and excitation in nerve. *J Physiol.*, 1952.

KLABUNDE, R. E. *Cardiovascular Physiolgy Concepts*. [S.l.]: Lippincott Williams Wilkins, 2012.

KNEPLEY, M. G. *et al.* Achieving High Performance with Unified Residual Evaluation. *arXiv e-prints*, set. 2013.

LARSON, M. G.; BENGZON, F. *The finite element method: Theory, implementation, and applications.* [S.l.]: Springer, 2013.

MESSAGE PASSING INTERFACE FORUM. *MPI: A Message-Passing Interface Standard, Version 3.1*. [S.l.], 2015. Accessed: 2025-07-26. Disponível em: ⟨https://www.mpi-forum.org/docs/mpi-3.1/mpi31-report.pdf⟩.

MITCHELL, C. C.; SCHAEFFER, D. G. A two-current model for the dynamics of cardiac membrane. *Bulletin of Mathematical Biology*, 2003.

NIEDERER, S. *et al.* Verification of cardiac tissue electrophysiology simulators using an n-version benchmark. *Royal Society*, 2011.

NIEDERER, S. *et al.* Simulating human cardiac electrophysiology on clinical time-scales. *Front. Physiol.*, 2011.

OLIVEIRA, R. S. *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*, 2017.

PLANK, G.; LOEWE, A.; NEIC, A. e. a. The opencarp simulation environment for cardiac electrophysiology. *Computer Methods and Programs in Biomedicine*, 2021.

SAAD, Y. *Iterative Methods for Sparse Linear Systems*. 2nd. ed. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics (SIAM), 2003. ISBN 978-0-89871-534-7. Disponível em: ⟨https://doi.org/10.1137/1.9780898718003⟩.

SUNDNES, J. *et al. Computing the Electrical Activity in the Heart*. [S.l.]: Springer, 2006.

TOMEK, J. *et al.* Development, calibration, and validation of a novel human ventricular myocyte model in health, disease. *eLife*, 2019.

TRAYANOVA, N. A. *et al.* Computational modeling of cardiac electrophysiology and arrhythmogenesis: toward clinical translation. *American Physiological Society.*, 2024.

TROBEC, R. *et al. Introduction to Parallel Computing*. [S.l.]: Springer Cham, 2018.

TUSSCHER, K. H. T.; PANFILOV, A. V. A model for human ventricular tissue. *J Physiol Heart Circ Physiol.*, 2004.

VINCENT, K. P.; GONZALES, M. J.; GILLETTE, A. K. e. a. High-order finite element methods for cardiac monodomain simulations. *Frontier Physiology*, 2015.