UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO DE CIÊNCIAS, TECNOLOGIAS E SAÚDE DO CAMPUS ARARANGUÁ
CURSO DE GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

Pedro Menezes Pimenta

Análise do Método dos Elementos Finitos para a Equação de Poisson

Araranguá
2023

Pedro Menezes Pimenta

Análise do Método dos Elementos Finitos para a Equação de Poisson

Trabalho de Conclusão de Curso do Curso de Graduação em Engenharia de Computação submetido ao Centro de Ciências, Tecnologias e Saúde do Campus Araranguá da Universidade Federal de Santa Catarina para a obtenção do título de Bacharel em Engenharia de Computação.
Orientadora: Profa. Priscila Cardoso Calegari, Dra.

Araranguá
2023

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Pimenta, Pedro Menezes
Análise do Método dos Elementos Finitos para a Equação de Poisson / Pedro Menezes Pimenta ; orientadora, Priscila Cardoso Calegari, 2023.
25 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Campus Araranguá, Graduação em Engenharia de Computação, Araranguá, 2023.

Inclui referências.

1. Engenharia de Computação. 2. Engenharia de Computação. 3. Método dos Elementos Finitos. 4. Equação de Poisson. I. Calegari, Priscila Cardoso. II. Universidade Federal de Santa Catarina. Graduação em Engenharia de Computação. III. Título.

Pedro Menezes Pimenta

Análise do Método dos Elementos Finitos para a Equação de Poisson

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do Título de Bacharel em Engenharia de Computação e aprovado em sua forma final pelo Curso de Graduação em Engenharia de Computação.

Araranguá, 1 de Dezembro de 2023.

Prof. Jim Lau, Dr.
Coordenador do Curso

Banca Examinadora:

Profa. Priscila Cardoso Calegari, Dra.
Orientadora

Prof. Antonio Carlos Sobieranski, Dr.
Avaliador
Universidade UFSC

Prof. Marcelo Zannin Da Rosa, Dr.
Avaliador
Universidade UFSC

Análise do Método dos Elementos Finitos para a Equação de Poisson

Finite Elements Method Analysis for Poisson's Equation

Pedro Menezes Pimenta [*] Priscila Cardoso Calegari [†]

2023, DEZEMBRO

Resumo

Técnicas computacionais para a resolução de Equações Diferenciais Parciais são muito importantes nos campos da matemática e da física. Uma técnica comum é a discretização do domínio em estudo, do qual sabemos apenas alguns valores em pontos distintos. Utilizando o Método dos Elementos Finitos é possível obter uma aproximação de uma função para a qual apenas se tem conhecimento dos valores de sua derivada em alguns pontos. Essa aproximação permite a utilização de malhas variadas para se obter um melhor resultado. É realizado um experimento comparativo entre a solução obtida através de malhas com diferentes configurações e métodos numéricos iterativos para a solução dos sistemas lineares. Analisamos a discretização da equação de Poisson bidimensional com diferentes malhas pelo Método de Elementos Finitos. Os resultados obtidos através da resolução do sistema linear são comparados por meio do número de iterações com base nas malhas utilizadas. Além disso, a análise de convergência do erro é apresentada com o uso de soluções manufaturadas.

Palavras-chaves: Método dos Elementos Finitos, Malhas, Métodos Iterativos.

[*]pedro.mp@grad.ufsc.br
[†]priscila.calegari@ufsc.br

Análise do Método dos Elementos Finitos para a Equação de Poisson

Finite Elements Method Analysis for Poisson's Equation

Pedro Menezes Pimenta [*] Priscila Cardoso Calegari [†]

2023, DECEMBER

Abstract

Computational techniques for solving Partial Differential Equations are very important in the fields of mathematics and physics. A common technique is to discretize the domain under study, of which we only know a few values at different points. Using the Finite Element Method, it is possible to obtain an approximation of a function for which we only know the values of its derivative at a few points. This approximation allows the use of different meshes to obtain a better result. A comparative experiment is carried out between the solution obtained using meshes with different configurations and iterative numerical methods for solving linear systems. We analyzed the discretization of the two-dimensional Poisson's equation with different meshes using the Finite Element Method. The results obtained by solving the linear system are compared by means of the number of iterations based on the meshes used. In addition, the error convergence analysis is presented using manufactured solutions.

Key-words: Finite Elements, Meshes, Iterative Methods.

[*]pedro.mp@grad.ufsc.br
[†]priscila.calegari@ufsc.br

1 Introdução

As Equações Diferenciais Parciais (EDPs) são um importante meio de estudo de áreas do conhecimento como a física, matemática e engenharia, já que fazem parte de modelos matemáticos e servem para descrever fenômenos físicos. Dentre os fenômenos físicos, campos elétricos são descritos por EDPs como a equação de Poisson. Para isso, existem métodos que utilizam medições pontuais e por meio conexões entre esses pontos são formadas malhas. Assim, conseguimos obter uma reconstrução do campo em estudo. Técnicas para esse objetivo vêm sendo desenvolvidas há muitos anos, como mostra (LIU; LI; PARK, 2022) e os trabalhos mais recentes de (ANDERSON et al., 2021) .

Uma técnica para a discretização de um campo ou superfície é o Método dos Elementos Finitos (MEF). O trabalho de (LIU; LI; PARK, 2022) apresenta um histórico do método, que se originou a partir do estudo da discretização de malhas em 1941 e que evoluiu para a resolução numérica de EDPs. Até hoje, o método é amplamente estudado em vários trabalhos detalhando seu funcionamento, feitos por (SAAD, 2003), (JOHNSON, 1988), (QUARTERONI; SALERI, 2007) e (RINCON; LIU, 2020). As bases do método foram descritas antes dos computadores modernos. Na década de 60, começou a ser utilizado para a resolução de EDPs (LIU; LI; PARK, 2022). Os computadores da época não possuiam grande capacidade de escalonamento e o método atingiu níveis muito superiores com a utilização de computadores modernos. Entretanto, ainda vem se desenvolvendo ao longo dos anos, sendo estudado com o desenvolvimento de ferramentas como em (ANDERSON et al., 2021) e em diferentes problemas de aplicação como por exemplo, escoamentos de biomecânica (LEE; GRIFFITH, 2022) e escoamentos em meios porosos (MOZOLEVSKI; MURAD; SCHUH, 2021).

O funcionamento do método consiste em discretizar o domínio por meio de uma malha, composta por elementos. Para o problema unidimemsional, o domínio é um intervalo e a malha é composta por subintervalos. Já para o problema bidimensional, o domínio mais simples é um retângulo, que pode ser decomposto em triângulos, retângulos, pentágonos, entre outras formas, e até mesmo uma combinação das anteriores. Dentre estas a mais básica é o triângulo, bastante utilizado por (SAAD, 2003) em seu livro. Após a discretização do domínio, utilizamos funções que definem a influência dos valores discretizados dentro de cada elemento. Com isso fazemos a substituição do problema original, uma EDP, em um problema discreto. Então ao invés de resolver uma EDP de forma direta, transformamos em um sistema linear que combina as funções internas de cada elemento da malha numa única solução. Assim, a solução obtida mantém sua continuidade, contudo existe um erro associado a essa aproximação. O número de equações do sistema linear e o erro estão associados à quantidade de elementos da malha. Assim, a quantidade de elementos da malha, pode demandar bastante poder computacional. Nos últimos anos surgiram ferramentas como a biblioteca *open-source, Modular Finite Element Method* - MFEM, desenvolvida por (ANDERSON et al., 2021). A ferramenta engloba o que há de mais recente no desenvolvimento do método dos Elementos Finitos.

O principal objetivo do presente trabalho é obter uma solução aproximada para a equação de Poisson por meio de Método dos Elementos Finitos. Como objetivo específico, o trabalho visa apresentar uma ferramenta para a solução de EDPs e auxiliar na formação de novos estudantes na área de Análise Numérica.

O presente artigo está organizado da seguinte maneira. A Seção 2 apresenta alguns trabalhos que tangenciam de alguma forma o desenvolvimento do presente trabalho. A fundamentação teórica, na Seção 3, apresenta a formulação do problema e abrange conceitos

que descrevem matematicamente o funcionamento do método. A Seção 4 apresenta a metodologia e explica como foi feita a discretização do domínio e principalmente, o funcionamento dos métodos numéricos utilizados para a solução dos sistemas lineares. Seguem os detalhes de desenvolvimento na Seção 5, explicando a implementação do trabalho feito na linguagem C++. A Seção 6 apresenta os resultados obtidos e discussões. Finalmente, na Seção 7 apresentamos as considerações finais e propostas de trabalhos futuros, bem como, as referências bibliográficas.

2 Trabalhos Correlatos

Existem diversos trabalhos que exploram os limites do Método dos Elementos Finitos, porém estamos mais interessados na discretização e reconstrução de uma função através da solução de uma EDP. Esse foi o trabalho de (PEDROSO, 2019), que realizou um estudo na resolução da equação de Poisson. Em seu trabalho foi analisado graficamente como a aproximação era impactada pela quantidade de elementos triangulares (triângulos retângulos) da malha. No presente trabalho, desejamos também analisar diferentes tipos de malhas e avaliar o desempenho de dois métodos numéricos utilizados na resolução do sistema linear.

O trabalho de (FISCHER et al., 2020) trouxe uma análise profunda sobre o escalonamento de métodos numéricos para a resolução de EDPs. Foi analisado de que forma um *solver* de alta performance lida com diferentes graus de liberdade para obter a solução do problema com a menor quantidade de iterações possível e seu impacto em *hardware*. Para a realização desse estudo também foi utilizada a ferramenta (ANDERSON et al., 2021).

3 Fundamentação Teórica

O Método dos Elementos Finitos aplicado a solução de EDPs consiste em discretizar o domínio do problema, definindo elementos que compõem uma malha. Após isso definimos as funções de interpolação, que serão responsáveis por manter a continuidade da solução que queremos alcançar. Com isso, transformamos o problema contínuo em um problema discreto, dado por um sistema de equações como descreve (JOHNSON, 1988). Dessa forma, obtemos uma aproximação contínua para a solução de uma EDP a partir de alguns valores dados.

Na Seção 3.1 apresentamos a Equação de Poisson bidimensional. A Seção 3.2 apresenta a dedução do método para o problema unidimensional de acordo com a literatura. Na Seção 3.3 apresentamos o método escalonado para duas dimensões e os passos para a montagem do sistema linear, de forma a auxiliar novos estudantes.

3.1 Formulação do problema

A equação de Poisson modela matematicamente problemas como: distribuição de temperatura no caso estacionário, potencial gerado por cargas elétricas e a correção da pressão em escoamentos (CUNHA, 2000; SAAD, 2003). A equação de Poisson é dada pela equação diferencial, definida em uma região Ω do plano,

$$-\Delta u = f(x, y), \tag{1}$$

sujeita a condições de contorno, sendo $u(x, y)$ a incógnita do problema e $f(x, y)$ um dado do problema. O operador

$$\Delta := \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2},$$

é chamado Operador Laplaciano (SAAD, 2003). Vamos resolver o problema com $\Omega = [0, 1] \times [0, 1]$ e condições de contorno do tipo Dirichlet nula, ou seja, a solução $u(x, y)$ é conhecida e é igual a zero no contorno da região.

O primeiro passo para resolver a equação (1) é discretizar a região onde a equação está definida e será aproximada, ou seja, vamos definir uma malha composta por elementos. As malhas utilizadas no presente trabalho são triangulares, mais detalhes na Seção 5. As próximas seções descrevem brevemente a dedução do Método dos Elementos Finitos em regiões com uma e duas dimensões.

3.2 Método dos Elementos Finitos em uma dimensão

Para melhor entender o Método dos Elementos Finitos vamos iniciar com a equação de Poisson em uma dimensão, assim como demonstram (QUARTERONI; SALERI, 2007) em seu livro. Assim temos,

$$-u''(x) = f(x), x \in (a, b), \tag{2}$$

com $u(a) = 0$ e $u(b) = 0$ e $f(x)$ uma função conhecida. Desejamos determinar $u(x)$ a partir dos pontos conhecidos de $u''(x)$, u e f são definidas em um domínio contínuo.

Agora, multiplicamos ambos os lados da equação por uma função de teste $v \in C^1([a, b])$, uma função conhecida dada pela combinação das funções de interpolação e que se anula nos extremos do intervalo, sendo C^1 o espaço das funções com primeira derivada contínuas em $[a, b]$. Mas pelo Método de Galerkin (um dos métodos de Elementos Finitos) podemos substituir a função de teste pelas próprias funções de interpolação, facilitando o processo de resolução do sistema. E integrando por partes obtemos

$$\int_a^b u'(x)v'(x)\,dx - \left[u'(x)v(x)\right]_a^b = \int_a^b f(x)v(x)\,dx \tag{3}$$

Devemos nos atentar de que a função v se anula nos extremos $x = a$ e $x = b$. Com isso devemos determinar $u \in C^1([a, b])$ com $u(a) = 0$, $u(b) = 0$, e

$$\int_a^b u'(x)v'(x)\,dx = \int_a^b f(x)v(x)\,dx \tag{4}$$

Dessa forma, a aproximação por elementos finitos é dada por $u_h \in V_h$, tal que $u_h(a) = 0, u_h(b) = 0$, onde $V_h = \{v_h \in C^1([a, b]) : v_{h|I_j} \in \mathbb{P}_1, j = 0, ..., N\}$ é o conjunto das funções contínuas e lineares em (a, b), o conjunto $\mathbb{P}_1$ contém os polinômios de grau 1. O intervalo discretizado é composto por subintervalos I_j. Cada subintervalo é definido como um elemento que compõe o domínio da solução, e V_h^0 é o subespaço de V_h cujas funções têm valor nulo nos extremos a e b. Toda função u_h de V_h^0 admite a representação,

$$u_h(x) = \sum_{j=1}^{N} u_j \psi_j(x), \tag{5}$$

sendo x_j pontos do intervalo $[a, b]$ discretizado, $u_j = u_h(x_j)$ uma aproximação para $u(x_j)$ e $\psi_j(x)$ as funções base do espaço V_h^0, dadas por,

$$\psi_j(x) = \begin{cases} \dfrac{x - x_{j-1}}{x_j - x_{j-1}} & \text{se } x \in I_{j-1}, \\ \dfrac{x - x_{j+1}}{x_j - x_{j+1}} & \text{se } x \in I_j, \\ 0 & \text{caso contrário.} \end{cases} \tag{6}$$

Nesse caso, a função aproximadora é um polinômio de grau 1, escrito como combinação linear das funções $\psi_j(x)$. A Figura 1 apresenta o gráfico da função $\psi_j(x)$, dada por (6). Os intervalos I_{j-1} e I_j representam dois elementos distintos e o ponto x_j encontra-se na fronteira desses dois elementos.

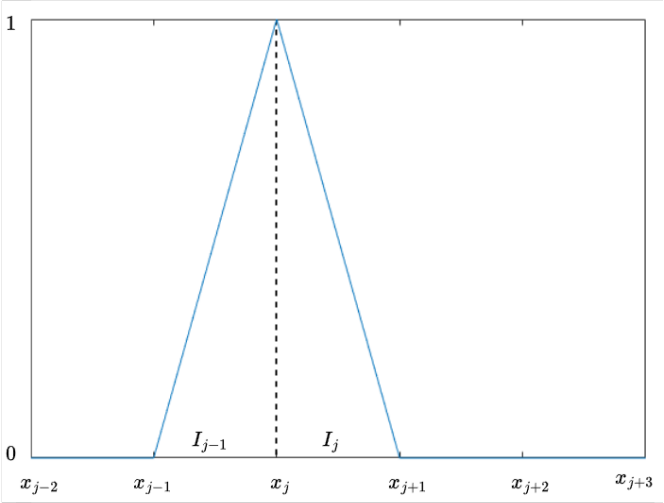


Figura 1 – Função de interpolação ψ_j. Fonte: Autor, 2023

Em cada elemento da malha, substituindo $v(x)$ por $\psi_j(x)$ em (4) temos,

$$\sum_{j=0}^{N} \int_{x_j}^{x_{j+1}} u_h'(x)\psi_j'(x)\,dx = \sum_{j=0}^{N} \int_{x_j}^{x_{j+1}} f(x)\psi_j(x)\,dx, \quad \forall \psi_j(x) \in V_h^0 \tag{7}$$

Consequentemente, o cálculo da integral na equação (7) em cada $j = 0, 1, \ldots, N$ é diferente de zero, apenas onde ψ_j não se anula, ou seja, nos intervalos (elementos) $I_{j-1} = [x_{j-1}, x_j]$ e $I_j = [x_j, x_{j+1}]$. Assim temos,

$$\int_{I_{j-1} \cup I_j} u_h'(x)\psi_j'(x)\,dx = \int_{I_{j-1} \cup I_j} f(x)\psi_j(x)\,dx, \quad j = 1, ..., N. \tag{8}$$

Agora, derivando (5) e substituindo na equação (8) e reescrevendo a equação com a separação dos elementos I_{j-1} e I_j, obtemos

$$\int_{I_{j-1} \cup I_j} u_h'(x)\psi_j'(x)\,dx = \int_{I_{j-1} \cup I_j} f(x)\psi_j(x)\,dx, \quad j = 1, ..., N. \tag{9}$$

Ainda devemos manter a junção das funções de interpolação no ponto de fronteira entre os elementos, assim obtemos o sistema linear

$$a_{j-1}u_{j-1} + b_j u_j + c_{j+1}u_{j+1} = \int_{I_{j-1}\cup I_j} f(x)\psi_j(x)\,dx \tag{10}$$

para cada $j = 1, ..., N$, sendo $a_{j-1} = \int_{I_{j-1}} \psi'_{j-1}(x)\psi'_j(x)\,dx$, $b_j = \int_{I_{j-1}\cup I_j} \psi'_j(x)\psi'_j(x)\,dx$, $c_{j+1} = \int_{I_j} \psi'_j(x)\psi'_{j+1}(x)\,dx$.

Assim, para a visualização desse sistema de equações no formato matricial, $Ax = b$, a matriz A é formada pelos coeficientes das integrais, ou seja, pelos coeficientes a_{j-1}, b_j e c_{j+1}. O vetor x contém as incógnitas u_j e o vetor b (lado direito do sistema linear) contém a o valor da integral $\int_{I_j} \psi'_j(x)\psi'_{j+1}(x)\,dx$. Portanto, resolvendo o sistema linear determinamos as aproximações u_j e obtemos a aproximação contínua para a solução $u(x)$ por meio da expressão (5).

3.3 Método dos Elementos Finitos em duas dimensões

A Seção anterior apresenta a dedução do do Método dos Elementos Finitos em uma dimensão. Para duas dimensões podemos nos basear no que foi explicado anteriormente, entretanto, para facilitar a aplicação do método em duas dimensões podemos utilizar o Método de Galerkin, escolhendo como função de teste as próprias funções de interpolação, e transformando nosso problema em um sistema linear da forma $Au = b$, em que A é a matriz de rigidez global que multiplica o vetor de coeficientes da solução aproximada u e o lado direito da equação é dado pelo vetor b. A montagem da matriz A é explicada por (AZEVEDO, 2003) em seu livro, no capítulo 8.

No caso bidimensional utilizaremos elementos triangulares, pois são formas mais simples de obtermos as funções de interpolação. A Figura 2 representa os elementos triangulares T_k e os pontos (x_j, y_j) (vértices de pelos menos um elemento), com $j = 1, \ldots, N$ e N número de vértices.

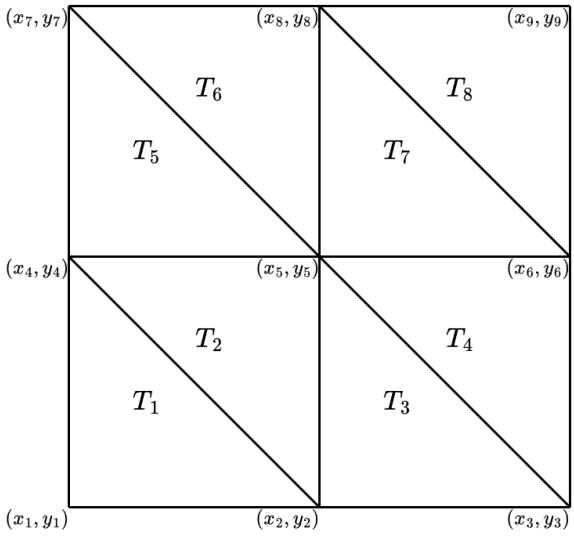


Figura 2 – Malha triangular com 8 elementos e 9 vértices. Fonte: Autor, 2023

As funções de interpolação $\psi_j(x)$ em cada um dos vértices de cada elemento triangular são planos como ilustra a Figura 3.

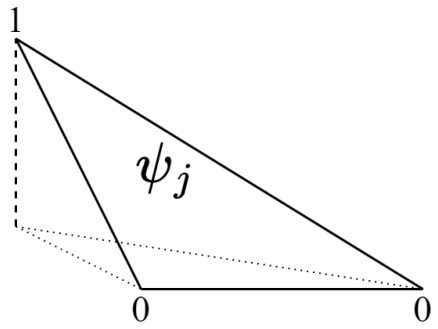


Figura 3 – Função de interpolação ψ_j num triângulo. Fonte: Autor, 2023

Sendo assim, a $\psi_j(x)$ de cada elemento deve possuir valor unitário para cada ponto x_j e ser nula nos outros pontos,

$$\psi_i(x_j, y_j) = \begin{cases} 1, & \text{se } i = j \\ 0, & \text{se } i \neq j \end{cases}$$

Podemos definir as funções de interpolação a partir dos três pontos que definem um triângulo qualquer. Por regra do Método dos Elementos Finitos nenhum dos ângulos internos do triângulo pode ser muito agudo, pois isso influenciaria na função de interpolação. Por convenção os pontos do elemento são percorridos no sentido anti-horário, pois o determinante da matriz dos pontos é positivo e equivalente ao dobro da área do triângulo. A Figura 4 apresenta um elemento triangular da malha, denotado por T_k.

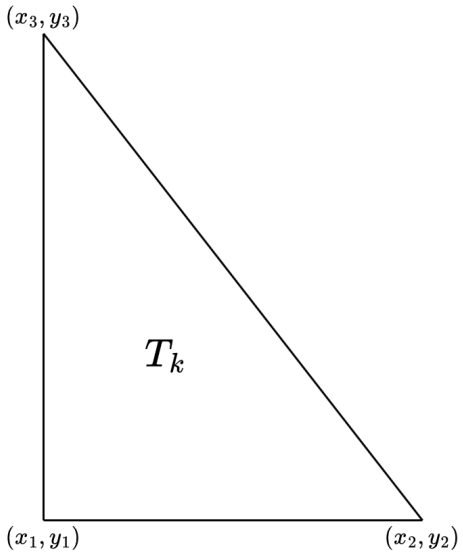


Figura 4 – Triângulo T_k. Fonte: Autor, 2023

A partir dos três pontos contidos pelo elemento, podemos definir as funções de

interpolação como

$$\psi_1(x_1, y_1) = a_1 + b_1 x_1 + c_1 y_1 \tag{11}$$
$$\psi_2(x_2, y_2) = a_2 + b_2 x_2 + c_2 y_2 \tag{12}$$
$$\psi_3(x_3, y_3) = a_3 + b_3 x_3 + c_3 y_3 \tag{13}$$

Como cada uma das funções deve possuir valor unitário em seu ponto correspondente, podemos reescrever (11) na forma matricial,

$$\begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{bmatrix} \begin{bmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

em que a matriz P contém os pontos dos vértices do triângulo, a segunda matriz contém os coeficientes da função de interpolação de cada ponto e o lado direito da equação contém a matriz identidade. Dessa forma, podemos concluir que a matriz de coeficientes é equivalente à matriz inversa da matriz P, associada aos pontos do triângulo. A matriz inversa da matriz associada aos pontos do triângulo pode ser reescrita como

$$P^{-1} = \begin{bmatrix} a_1 & a_2 & a_3 \\ \dfrac{\partial \psi_1}{\partial x} & \dfrac{\partial \psi_2}{\partial x} & \dfrac{\partial \psi_3}{\partial x} \\ \dfrac{\partial \psi_1}{\partial y} & \dfrac{\partial \psi_2}{\partial y} & \dfrac{\partial \psi_3}{\partial y} \end{bmatrix}, \tag{14}$$

onde cada coluna contém os coeficientes das funções de interpolação. Já a segunda e terceira linha contém os gradientes em x e em y, respectivamente.

Agora que podemos encontrar de forma rápida as funções de interpolação de qualquer elemento triangular. Também podemos determinar os coeficientes da matriz local de cada elemento. Para isso, basta calcular o escalar entre os vetores gradiente das funções encontradas, as componentes em x e em y dos vetores são dadas, respectivamente, pela segunda e terceira linha de P^{-1}, conforme (PEDROSO, 2019). Essa forma de determinar ψ é uma forma mais direta para elementos triangulares, não necessariamente é feita da mesma forma pela biblioteca MFEM.

$$\nabla \psi_i \cdot \nabla \psi_j, \quad i, j \in \{1, 2, 3\} \tag{15}$$

Com isso é possível perceber que a própria matriz inversa (14), possui os gradientes das funções de interpolação que desejamos. Para determinar os coeficientes da matriz local de cada elemento basta calcular o produto escalar dos gradientes em cada coluna mas excluindo a primeira linha, já que ela não faz parte do gradiente. Assim temos que a matriz local do elemento T_k, denotada por $K^{(e)}$ possui os coeficientes,

$$K^{(e)} = \begin{bmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{bmatrix} \tag{16}$$

Os índices da matriz indicam quais funções de interpolação estão sendo multiplicadas, por exemplo, o coeficiente K_{12} é o resultado escalar da multiplicação entre ψ_1 e ψ_2 como é mostrado em (15) e por consequência K_{12} é equivalente a K_{21}. A enumeração utilizada nessa explicação é apenas local, ou seja, apenas como referência num elemento qualquer. Em uma malha maior a enumeração dos pontos (vértices) deve ser levada em consideração na montagem da matriz de rigidez global A, como mostra (SAAD, 2003) em seu livro no capítulo 2. Por exemplo, se um elemento contêm os vértices 1, 5 e 7, então seus coeficientes estarão presentes nas possíveis combinações de linhas e colunas utilizando os três índices, ou seja, o conjunto de posições será $\{(1,1),(1,5),(1,7),(5,1),(7,1),(5,7),(7,5),(5,5),(7,7)\}$. Isso deverá ser feito para cada elemento. Caso outros elementos compartilhem pontos iguais, os coeficientes desse mesmo ponto serão somados na matriz global.

4 Metodologia

Nesta Seção será abordado como deve ser feita a discretização do domínio para se obter a solução da Equação de Poisson (1) pela implementação utilizada no Método dos Elementos Finitos, e como os métodos de resolução de sistemas lineares utilizados funcionam. Como descrito na Seção anterior, reduzimos o problema de aproximação para um sistema de equações lineares, após realizar a montagem desses sistemas utilizamos dois métodos numéricos para a resolução, o método do Gradiente Conjugado Pré-condicionado e o método Iterativo Estacionário Jacobi Pré-condicionado. Ambos os métodos estão presentes na biblioteca MFEM (ANDERSON et al., 2021) e foram utilizados para efeito de comparação.

4.1 Discretização do Domínio

A geração de uma malha deve ser analisada sobre o domínio do problema. Vamos considerar $\Omega = \{(x,y)|a \leq x \leq b, \text{ e } c \leq y \leq d\}$. Uma malha é uma partição do domínio em subregiões (elementos) T_k, de tal forma que satisfazem as condições,

$$\Omega = \bigcup_{k=1}^{\text{Nel}} T_k \text{ e } T_i \cap T_j = \varnothing, \text{ se } i \neq j,$$

sendo Nel o número de elementos (RINCON; LIU, 2020). Para discretizar o domínio podemos criar uma malha personalizada, com pontos previamente escolhidos, utilizando ferramentas de geração de malhas e depois traduzindo para um formato aceito pela biblioteca MFEM. Uma maneira mais simples é criar uma malha mais grosseira, em um editor de texto seguindo a estrutura da Figura 5, que contenha poucos elementos e então realizamos iterações de refinamento até quantos elementos julgarmos necessários. As malhas utilizadas neste trabalho foram definidas utilizando a estrutura de arquivo da Figura 5.

```
MFEM mesh v1.0

# Tipos de geometria
#POINT = 0
#SEGMENT = 1
#TRIANGLE = 2
#SQUARE = 3
#TETRAHEDRON = 4
#CUBE = 5
#PRISM = 6

# Dimensão espacial: 2 ou 3
dimension
<dimensão>

# Elementos da malha
elements
<número de elementos>
<atributo do elemento> <tipo de geometria> <índice do vértice 1> ... <índice do vértice n>
...

# Elementos da malha na fronteira
boundary
<número de elementos na fronteira>
<atributo do elemento> <tipo de geometria> <índice do vértice 1> ... <índice do vértice n>
...

# Coordenadas dos vértices
vertices
<número de vértices>
<dimensão>
<coordenada 1> ... <coordenada <dimensão> >
...
```

Figura 5 – Arquivo de malha. Fonte: Biblioteca MFEM, 2023

4.2 Sistemas Lineares

Os métodos utilizados para resolver o sistema linear do problema de Poisson bidimensional (1), foram o Gradiente Conjugado Pré-condicionado e o Método Iterativo Estacionário Jacobi Pré-condicionado. Para ambos haverá uma breve explicação de seu funcionamento e iniciam a partir de um sistema linear da forma

$$Ax = b. \tag{17}$$

Para o nosso problema, a matriz A representa a matriz de rigidez global, o vetor x armazena os coeficentes da solução aproximada de $u(x, y)$ e o vetor b corresponde aos coeficientes que dependem de $f(x, y)$.

4.2.1 Gradiente Conjugado Pré-condicionado

O Método dos Gradientes Conjugados troca o problema de resolver o sistema (17) de forma direta pelo problema equivalente de encontrar um minimizador de $F(x)$, então $\nabla F(x) = 0$, tal que

$$F(x) = \frac{1}{2}x^T Ax - b^T x,$$

assim temos

$$\nabla F(x) = Ax - b.$$

Mais detalhes sobre o método podem ser encontrados em (BURDEN; FAIRES, 2010; CUNHA, 2000; SAAD, 2003). A explicação de como ocorrem as iterações foi retirada do trabalho de conclusão de curso de (FRANGO, 2018). Para a primeira iteração é calculado o resíduo a partir de um chute inicial x_0,

$$r^{(0)} = -\nabla F(x^{(0)}) \tag{18}$$

e define a direção de busca inicial $d^{(0)}$

$$d^{(0)} = r^{(0)}, \tag{19}$$

para os próximos valores de resíduo $r^{(k)}$, de direção de busca $d^{(k)}$ e de $x^{(k)}$ é feito

$$r^{(k+1)} = r^{(k)} - \alpha^{(k)} A r^{(k)}, \tag{20}$$

$$d^{(k+1)} = r^{(k+1)} + \beta^{(k)} d^{(k)}, \tag{21}$$

$$x^{(k+1)} = x^{(k)} + \alpha^{(k)} d^{(k)}. \tag{22}$$

A escolha do coeficiente $\beta^{(k)}$ é feita de forma que as direções $d^{(k+1)}$ e $d^{(k)}$ sejam A-ortogonais. Então por definição temos que α e β são definidos da seguinte forma

$$\alpha^{(k)} = \frac{(r^{(k)})^T r^{(k)}}{(r^{(k)})^T A r^{(k)}} \tag{23}$$

$$\beta^{(k)} = -\frac{(d^{(k)})^T A r^{(k+1)}}{(d^{(k)})^T A d^{(k)}} \tag{24}$$

Assim é definido, de forma básica o funcionamento do Método do Gradiente Conjugado. A matriz A dever ser simétrica e positiva definida, ou seja, simétrica porque $A = A^T$, e positiva definida porque A é uma matriz de ordem $n \times n$ e satisfaz $x^T A x > 0$ para todos os vetores não-nulos $x \in \mathbb{R}^n$. Entretanto, neste trabalho utilizamos o mesmo método mas com um pré-condicionamento antes de iniciar a resolução do sistema. O pré-condicionamento consiste em substituirmos o sistema $Ax = b$, por outro equivalente,

$$M^{-1} A x = M^{-1} b \tag{25}$$

sendo M a matriz pré-condicionadora. Com isso espera-se que M seja tão próxima de A que M também seja simétrica e positiva definida; $M^{-1}A$ seja bem condicionado; e por consequência, a equação $Mx = b$ seja fácil de ser resolvida (CUNHA, 2000). Para que a resolução ocorra é essencial que essas três propriedades ocorram. Ao final não há muita alteração no método, a estrutura das equações permanece a mesma, com a aplicação da matriz pré-condicionadora ao resíduo, alterando as equações (19), (20), (23), (24):

$$\begin{aligned}
d^{(0)} &= M^{-1} r^{(0)} \\
\alpha^{(k)} &= \frac{r^{(k)} d^{(k)}}{d^{(k)} A d^{(k)}} \\
r^{(k+1)} &= r^{(k)} - \alpha^{(k)} A d^{(k)} \\
\beta^{(k)} &= -\frac{d^{(k)} A r^{(k+1)}}{d^{(k)} A d^{(k)}}
\end{aligned}$$

4.2.2 Método Iterativo Estacionário Jacobi Pré-condicionado

Seguindo a definição de processo estacionário (KREMER, 2009), um método iterativo é estacionário se cada aproximante da solução é obtido do anterior sempre pelo mesmo processo. O método utilizado neste trabalho foi o método de Jacobi. Consideramos novamente o sistema (17) de ordem n, com $a_{ii} \neq 0$,

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ \vdots & \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= b_n \end{aligned} \tag{26}$$

Isolando x pela diagonal em cada uma das equações temos:

$$\begin{aligned} x_1 &= \frac{1}{a_{11}}(b_1 - a_{12}x_2 - a_{13}x_3 - \cdots - a_{1n}x_n) \\ x_2 &= \frac{1}{a_{22}}(b_2 - a_{21}x_1 - a_{23}x_3 - \cdots - a_{2n}x_n) \\ &\vdots \\ x_n &= \frac{1}{a_{nn}}(b_n - a_{n1}x_2 - a_{n2}x_3 - \cdots - a_{n,n-1}x_{n-1}) \end{aligned} \tag{27}$$

Para o sistema linear (26), o método de Jacobi (CUNHA, 2000; SAAD, 2003) consiste em dada uma aproximação inicial $x^{(0)} = (x_1^{(0)}, x_2^{(0)}, \ldots, x_n^{(0)})$ e uma tolerância, gera-se uma sequência de aproximações, que converge para a solução exata, através do processo iterativo definido por:

$$\begin{aligned} x_1^{(k+1)} &= \frac{1}{a_{11}}(b_1 - a_{12}x_2^{(k)} - a_{13}x_3^{(k)} - \cdots - a_{1n}x_n^{(k)}) \\ x_2^{(k+1)} &= \frac{1}{a_{22}}(b_2 - a_{21}x_1^{(k)} - a_{23}x_3^{(k)} - \cdots - a_{2n}x_n^{(k)}) \\ &\vdots \\ x_n^{(k+1)} &= \frac{1}{a_{nn}}(b_n - a_{n1}x_2^{(k)} - a_{n2}x_3^{(k)} - \cdots - a_{n,n-1}x_{n-1}^{(k)}) \end{aligned} \tag{28}$$

Ao utilizar o pré-condicionamento no método de Jacobi, mudamos o sistema para

$$x^{(k+1)} = x^{(k)} + M(b - Ax^{(k)}), \tag{29}$$

onde a matriz M representa o pré-condicionador.

Em seu trabalho de conclusão de curso, (KREMER, 2009) também detalha o método de Jacobi, mostrando que podemos reescrever a matriz A como sendo $A = L + D + U$, em que L é a matriz triangular inferior de A, D é a matriz diagonal de A e U é a matriz triangular superior de A. Então escrevemos o sistema (17) como

$$\begin{aligned} (L + D + U)x &= b \\ Dx &= (-L - U)x + b \\ x &= D^{-1}(-L - U)x + D^{-1}b \end{aligned}$$

$$x = Tx + c$$

em que $T = D^{-1}(-L - U)$ e $c = D^{-1}b$. Com a sequência de aproximações temos:

$$x^{(k+1)} = Tx^{(k)} + c \tag{30}$$

Na prática as iterações do método não são realizadas com base em (30), pois é apenas para fins teóricos, portanto realizamos a iterações com base em (28) e para o método com pré-condicionamento utilizamos (29).

5 Experimentos Numéricos

A partir de uma biblioteca *open-source* em C++ feita para o Método dos Elementos Finitos chamada MFEM (ANDERSON et al., 2021), foi possível implementar uma aproximação da solução da equação de Poisson (1). Entretanto a implementação feita para este trabalho aceita apenas problemas com condições de contorno Dirichlet nulas. Para iniciar, partimos dos exemplos fornecidos pela biblioteca, mais precisamente do exemplo 0. Nesse exemplo o código realiza a solução da equação de Poisson com $f(x, y) = 1$ e domínio $\Omega = [0, 1] \times [0, 1]$,

$$-\Delta u = 1, \tag{31}$$

e depois forçamos os coeficientes de uma função escolhida, essa função será explicada mais adiante.

Inicialmente fornecemos para o código o arquivo da malha a ser utilizada, utilizamos as informações da malha para obter apenas os vértices dos elementos. Por estarmos utilizando o método de Galerkin, podemos facilmente obter as funções de interpolação através dos pontos dos vértices do triângulo, facilitando a obtenção da matriz A do sistema linear para quaisquer formatos de triângulos. O arquivo de malha utilizado é da extensão .mesh, nesse arquivo estão as informações de quantas dimensões os elementos são compostos, no caso são duas dimensões, a quantidade de elementos na malha, a geometria de cada um, podendo ser triangular, quadrado ou de outro formato, e os índices dos vértices que os compõem, também contêm quais segmentos pertencem aos limites da malha e por fim, quantos vértices estão presentes e suas coordenadas.

Montamos a matriz de rigidez global a partir da malha fornecida como explicado na Seção 3.3 para obtermos a matriz A do sistema linear e o lado direito do sistema linear. Após fornecer a malha para o programa, as chamadas de função da própria biblioteca fazem a montagem da matriz de rigidez global.

Para o nosso experimento, escolhemos como solução manufaturada ,

$$u(x, y) = \text{sen}(\pi x)\text{sen}(\pi y). \tag{32}$$

Substituindo (32) em (1) obtemos,

$$f(x, y) = 2\pi^2 sen(\pi x) sen(\pi y). \tag{33}$$

Para o vetor b do sistema linear passamos os coeficientes calculados por (33) aos valores definidos pela integral do lado direito da equação (10). Para o cálculo do erro usamos um vetor com os coeficientes de (32).

As malhas criadas para os testes têm limites $0 \leq x \leq 1$ e $0 \leq y \leq 1$, das três malhas, Estru.mesh e unEstru.mesh foram cedidas pela Professora Doutora Catalina Maria Rua Alvarez e apenas traduzidas para o formato reconhecido da biblioteca MFEM (ANDERSON et al., 2021). A escolha da função deve ser feita em conjunto com as malhas utilizadas, neste caso utilizamos a função (32), pois ela é nula nas bordas das malhas escolhidas. É possível alterar a frequência de oscilação da senóide ou os limites da malha, contanto que o valor da função nos limites da malha seja zero. As imagens das malhas criadas para este trabalho são apresentadas pelas Figuras 6, 7 e 8.

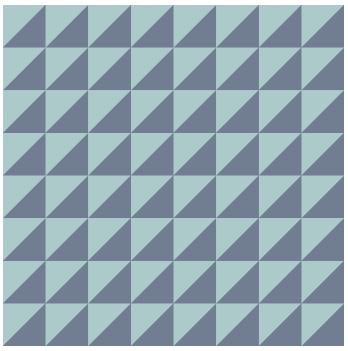


Figura 6 – Malha triang-ret.mesh composta por 128 triângulos retângulos. Fonte: Autor, 2023

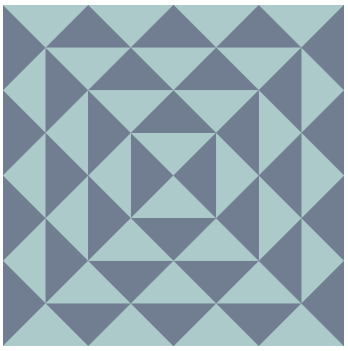


Figura 7 – Malha Estru.mesh composta por 64 triângulos retângulos. Fonte: D.ra. Catalina Maria Rua Alvarez, 2009

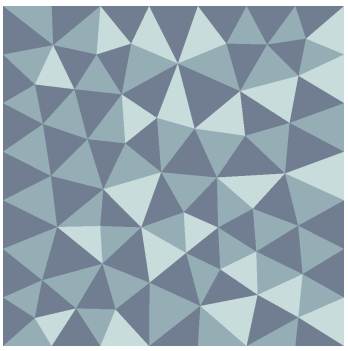


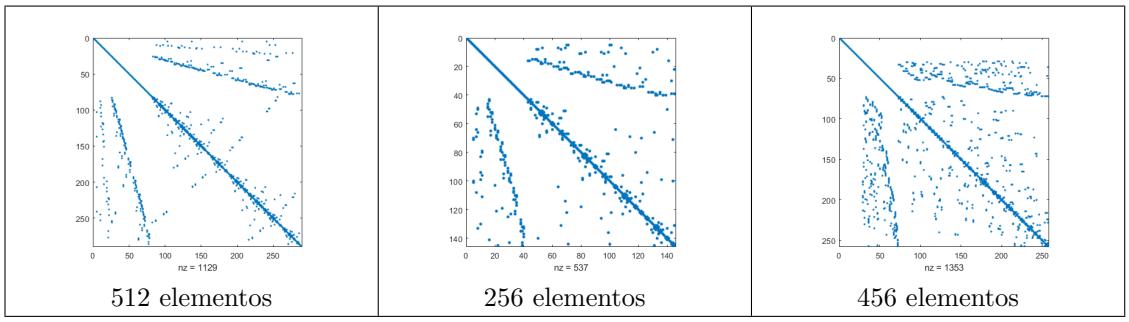
Figura 8 – Malha unEstru.mesh composta por 114 triângulos variados. Fonte: D.ra. Catalina Maria Rua Alvarez, 2009

As malhas triang-ret.mesh e Estru.mesh foram criadas com menos elementos e depois refinadas uniformemente para se obter a quantidade de elementos desejada, apenas a malha unEstru.mesh parte de maior quantidade de elementos. O refinamento uniforme é feito pela divisão de cada elemento em quatro elementos iguais, para cada elemento triangular é adicionado três novos pontos, posicionados no ponto médio de cada aresta. Isso é repetido até atingir a quantidade de elementos desejada.

Ao utilizar os métodos para a solução de sistemas lineares disponíveis na própria biblioteca MFEM (ANDERSON et al., 2021), se mostrou necessário utilizar um pré-condicionador apenas para o Método Iterativo Estacionário Jacobi descrito na Seção 4.2.2, por conta da quantidade de iterações necessárias. O mesmo pré-condicionador também foi utilizado no Método do Gradiente Conjugado Pré-condicionado descrito na Seção 4.2.1, e ao realizar testes sem o pré-condicionador os resultados quase não sofreram alterações. Entretanto o método pré-condicionado foi mantido por motivos de comparação entre os métodos.

Para o cálculo do erro, usamos a norma do máximo, ou seja, a diferença (em valor absoluto) entre solução exata e aproximada, como apresenta (BURDEN; FAIRES, 2010) em seu livro, no capítulo 7, ao comparar os erros de alguns métodos iterativos.

Também foi implementada uma função para exportar a matriz de rigidez global das malhas utilizadas num arquivo de texto, para a visualização do grau de esparsidade da matriz de rigidez global A. A biblioteca utiliza matrizes esparsas por questões de otimização. Portanto, para exportar a matriz completa foi necessário realizar uma conversão de matriz esparsa para matriz densa. Assim foi possível visualizar a esparsidade das matrizes pelo MATLAB utilizando os arquivos de texto. A Figura 9 apresenta o grau de esparsidade, ou seja, a localização dos elementos não nulos das matrizes associadas às malhas triang-ret.mesh, Estru.mesh e unEstru.mesh, respectivamente.



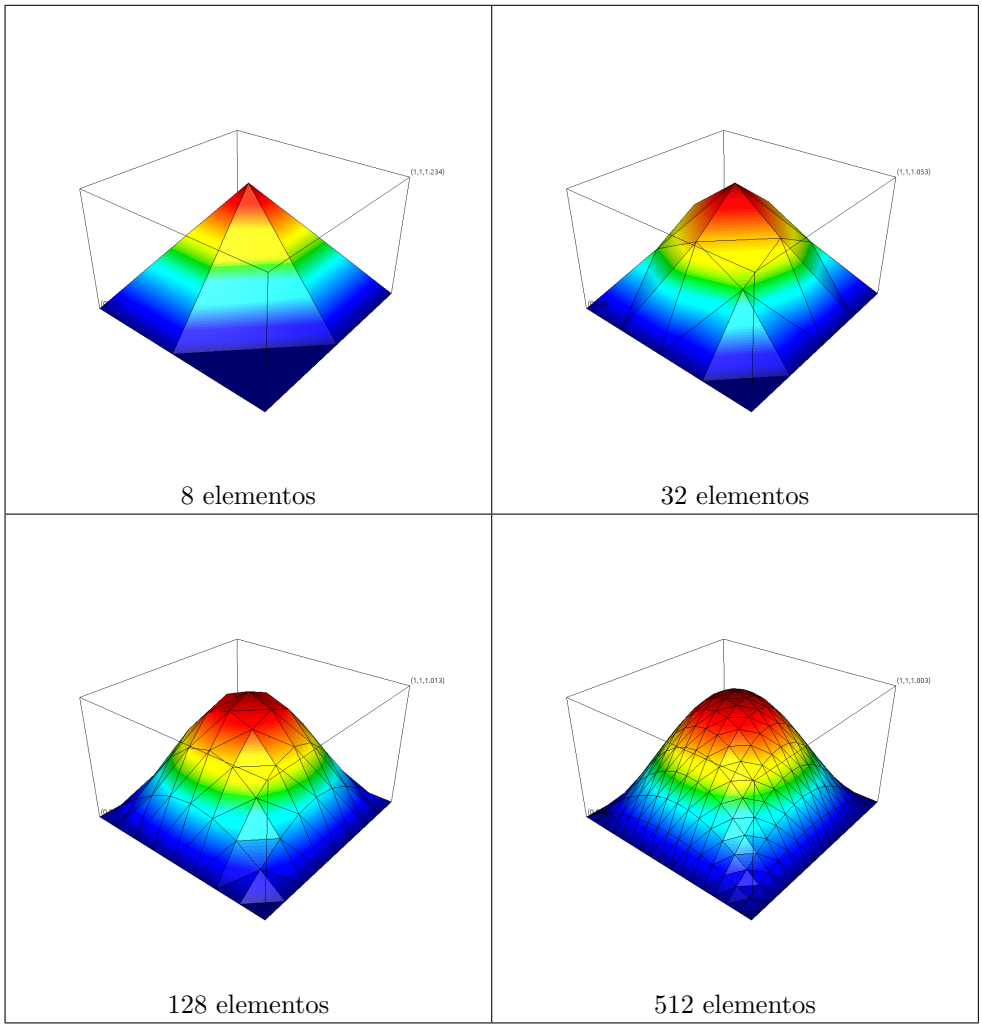
512 elementos | 256 elementos | 456 elementos

Figura 9 – Imagens dos coeficientes não nulos da matriz de rigidez global das malhas triang-ret.mesh, Estru.mesh e unEstru.mesh, respectivamente. Fonte: Autor, 2023

Os pontos iniciais correspondem aos pontos presentes nos limites da malha e esses pontos não possuem função de interpolação justamente porque seu valor já é conhecido e o efeito espelhado ocorre por causa da interação entre as funções de interpolação de pontos adjacentes. Como a enumeração dos pontos a cada refinamento é feita de forma parecida em todas as malhas o gráfico de esparsidade da matriz possui então estrutura semelhante.

6 Resultados e Discussões

Para iniciar a apresentação dos resultados obtidos, começaremos pelas imagens das aproximações utilizando a malha triang-ret.mesh em 3D. As imagens foram obtidas utilizando a ferramenta GLVis, uma recomendação dos criadores da biblioteca MFEM. A ferramenta de visualização serviu para obter as imagens das malhas, das soluções e mapas topográficos. A Figura 10 apresenta a solução aproximada em malhas com diferentes quantidades de elementos. As malhas da Figura 6 e 7 são malhas são uniformes estruturadas, elementos de mesmo tamanho e a medida que diminuímos o tamanho dos elementos pela metade, a quantidade de elementos quadriplica. Já a Figura 8 apresenta uma malha não uniforme.



8 elementos | 32 elementos

128 elementos | 512 elementos

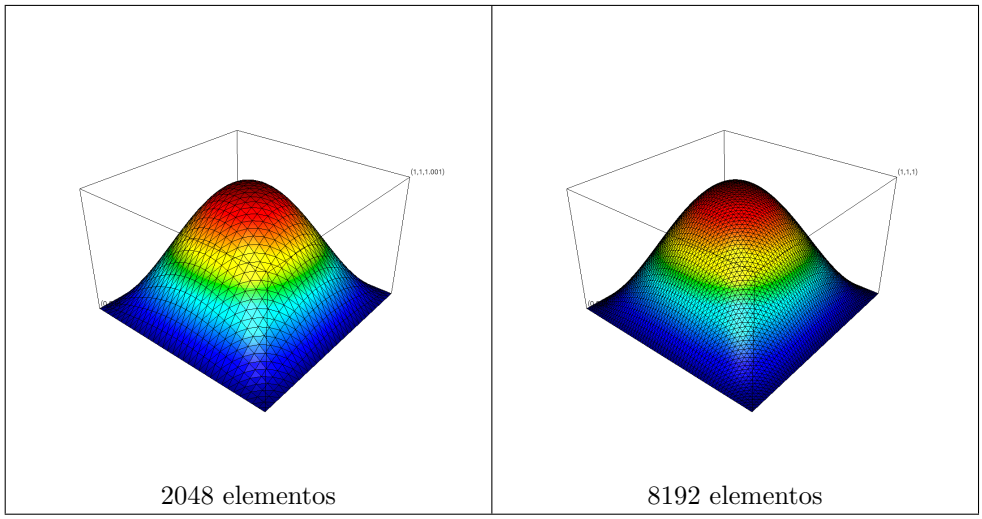


Figura 10 – Imagens da aproximação da função utilizando a malha triang-ret.mesh. Fonte: Autor, 2023

A Tabela 1 apresenta a análise de convergência do método. A primeira coluna apresenta a medida característica da malha, usamos a diagonal do elemento triangular da malha triang-ret.mesh pois é a maior medida. A segunda coluna apresenta o erro, obtido com o cálculo da norma do máximo. A terceira coluna ordem mostra que a medida que diminuímos o tamanho dos elementos da malha, pela metade, o método converge para a ordem esperada, nesse caso 2. Isso ocorre pois avaliamos o $\log_2$ da razão entre dois erros consecutivos. Neste experimento, com mais de 2048 elementos já é possível obter uma aproximação aceitável. Por volta de 8192 elementos a aproximação, pela visualização do gráfico, já é suficiente. Acima disso, o custo computacional não compensa o retorno obtido. Contudo isso deve ser analisado separadamente para outros problemas e outras malhas.

h	Erro	Ordem
$\frac{\sqrt{2}}{2}$	0,233701	-
$\frac{\sqrt{2}}{4}$	0,0530293	2,13980229760317
$\frac{\sqrt{2}}{8}$	0,0129507	2,03375962534964
$\frac{\sqrt{2}}{16}$	0,00321897	2,00835904288814
$\frac{\sqrt{2}}{32}$	0,000803564	2,00211429402292
$\frac{\sqrt{2}}{64}$	0,00020081	2,00058181732132
$\frac{\sqrt{2}}{128}$	5,02E-05	2,00003448542882

Tabela 1 – Tabela de verificação numérica da ordem do método utilizando a malha triang-ret.mesh. Fonte: Autor, 2023

As Tabelas 2, 3 e 4 apresentam, além do erro, uma comparação entre os dois

métodos para a solução de sistemas lineares Gradiente Conjugado Pré-condicionado (PCG - Preconditioned Conjugate Gradient) e método Iterativo Estacionário Jacobi (SLI - Stationary Linear Iterative) com as três malhas utilizadas. São apresentados o número de iterações que cada método executou e o tempo decorrido.

Os testes foram realizados em uma máquina virtual com sistema operacional Ubuntu. O processador utilizado foi um Intel Core i7-8700 com 6 núcleos, e 8GB de memória RAM.

Elementos	Erro (PCG)	Tempo (ms)	Iterações	Erro (SLI)	Tempo (ms)	Iterações
8	0,233701	0	2	0,233701	0	2
32	0,0530293	0	6	0,0530289	0	18
128	0,0129507	2	11	0,0129497	2	68
512	0,00321897	19	19	0,00321785	24	265
2048	0,000803564	254	32	0,000801212	320	1057
8192	0,00020081	4014	60	0,000199664	4380	4225
32768	5,02E-05	84937	114	4,90E-05	92924	16900

Tabela 2 – Tabela de comparação do erro utilizando a malha triang-ret.mesh e os dois solvers. Fonte: Autor, 2023

Elementos	Erro (PCG)	Tempo (ms)	Iterações	Erro (SLI)	Tempo (ms)	Iterações
4	0,644934	0	2	0,644934	0	2
16	0,0483114	0	3	0,0483108	0	11
64	0,0368201	0	7	0,0368208	0	36
256	0,0151014	5	12	0,0151024	6	136
1024	0,0052067	64	21	0,00520788	81	534
4096	0,00165696	1004	38	0,00165812	1135	2119
16384	0,000502921	16197	71	0,000504057	17729	8458

Tabela 3 – Tabela de comparação do erro utilizando a malha Estru.mesh e os dois solvers. Fonte: Autor, 2023

Elementos	Erro (PCG)	Tempo (ms)	Iterações	Erro (SLI)	Tempo (ms)	Iterações
114	0,0408172	1	10	0,0408161	2	56
456	0,0111453	17	19	0,0111443	21	211
1824	0,0030611	214	35	0,00306002	236	817
7296	0,000833829	3235	70	0,000832715	3519	3249
29184	0,000225499	62469	140	0,000224379	67525	12983

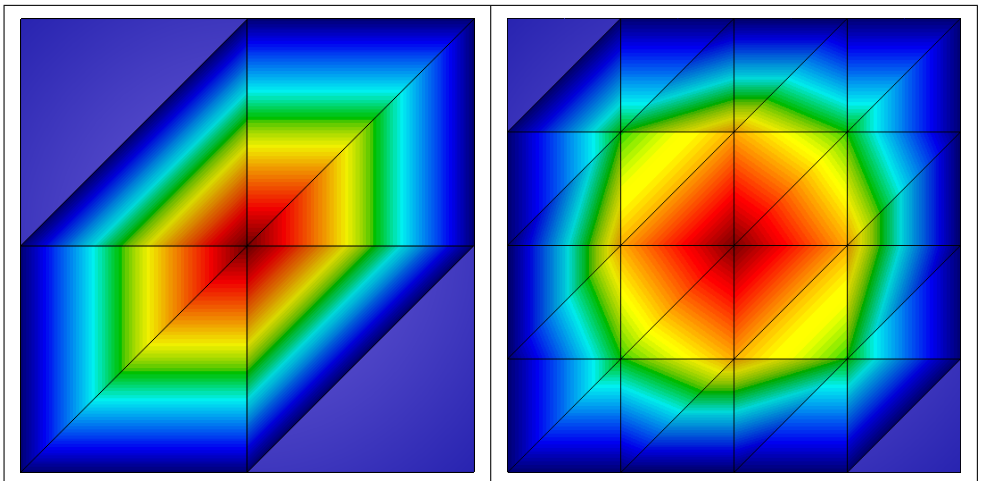
Tabela 4 – Tabela de comparação do erro utilizando a malha unEstru.mesh e os dois solvers. Fonte: Autor, 2023

O método do grandiente conjugado com pré-condicionamento se mostrou extremamente mais eficiente ao lidar com malhas com maior número de elementos. E em nenhum momento necessitou de mais de 140 iterações para obter uma boa aproximação. A maior malha utilizada foi com 32768 elementos utilizando a composição da malha triang-ret.mesh partindo de dois elementos e realizando 7 iterações de refinamento. A diferença entre

os erros obtidos pelos métodos utilizados foi de $1,2E-6$, apenas $2,31\%$, resultado em porcentagem do absoluto da razão entre os erros dos dois métodos. O SLI realmente conseguiu um valor de erro um pouco menor, contudo precisou de pelo menos 148 vezes mais iterações. Ao aumentarmos o número de elementos da malha o PCG praticamente dobrava a quantidade de iterações, enquanto o SLI aumentava em aproximadamente 4 vezes.

Outra observação que podemos fazer é em relação a quantidade de elementos necessários para um erro aceitável. Com a malha triang-ret.mesh na Tabela 2 com 2048 elementos obtivemos um erro próximo ao erro de 16384 elementos. Com a malha Estru.mesh na Tabela 3 o erro com os mesmos 16384 elementos obteve um valor próximo ao erro de 7296 elementos com a malha unEstru.mesh na tabela 4. Assim, podemos concluir que apesar da solução não depender do formato dos elementos escolhidos, a disposição dos pontos escolhidos impactou diretamente na velocidade de obtenção de uma aproximação aceitável.

Para uma visualização das aproximações de cada malha, as Figuras 11, 12 e 13, apresentam as imagens topográficas da solução aproximada. Por questões práticas, a malha da Figura 11 é mostrada a partir de 8 elementos, ou seja, a malha inicial após uma iteração de refinamento uniforme, uma vez que na malha inicial, com 4 elementos, todos os pontos presentes fazem parte do contorno do domínio. Portanto, possuem valor nulo e a imagem topográfica seria um plano. As curvas de nível são representadas por um gradiente colorido.



8 elementos | 32 elementos

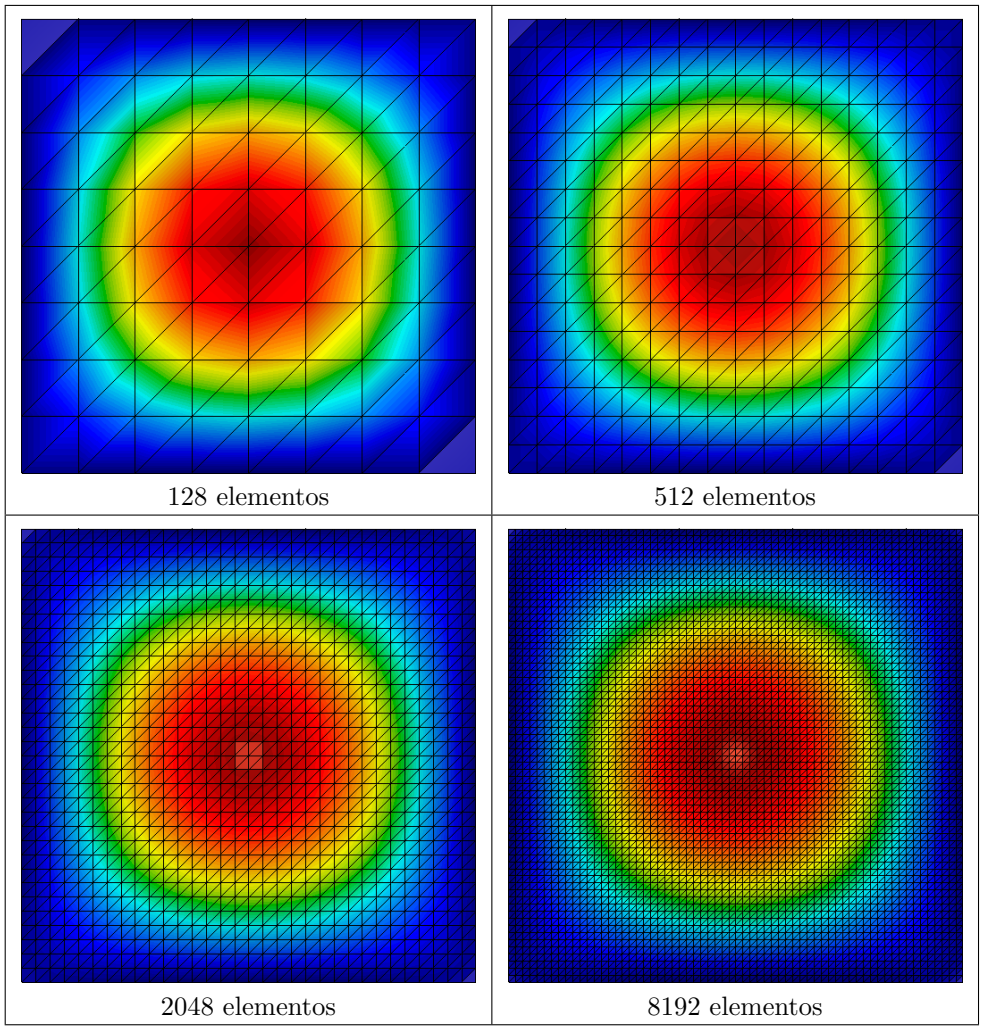


Figura 11 – Imagens topográficas da malha triang-ret.mesh com diferentes quantidades de elementos. Fonte: Autor, 2023

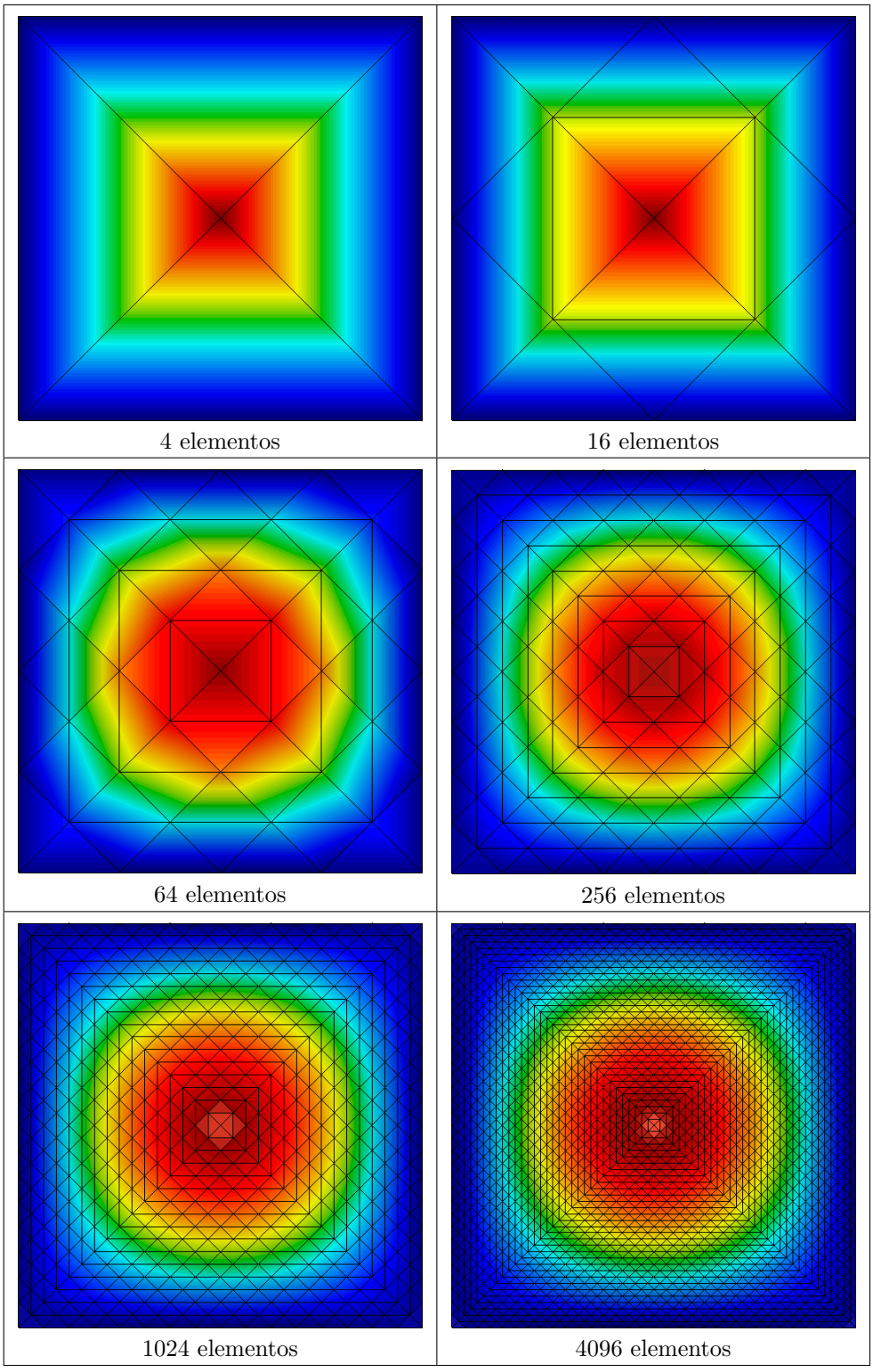


Figura 12 – Imagens topográficas da malha Estru.mesh com diferentes quantidades de elementos. Fonte: Autor, 2023

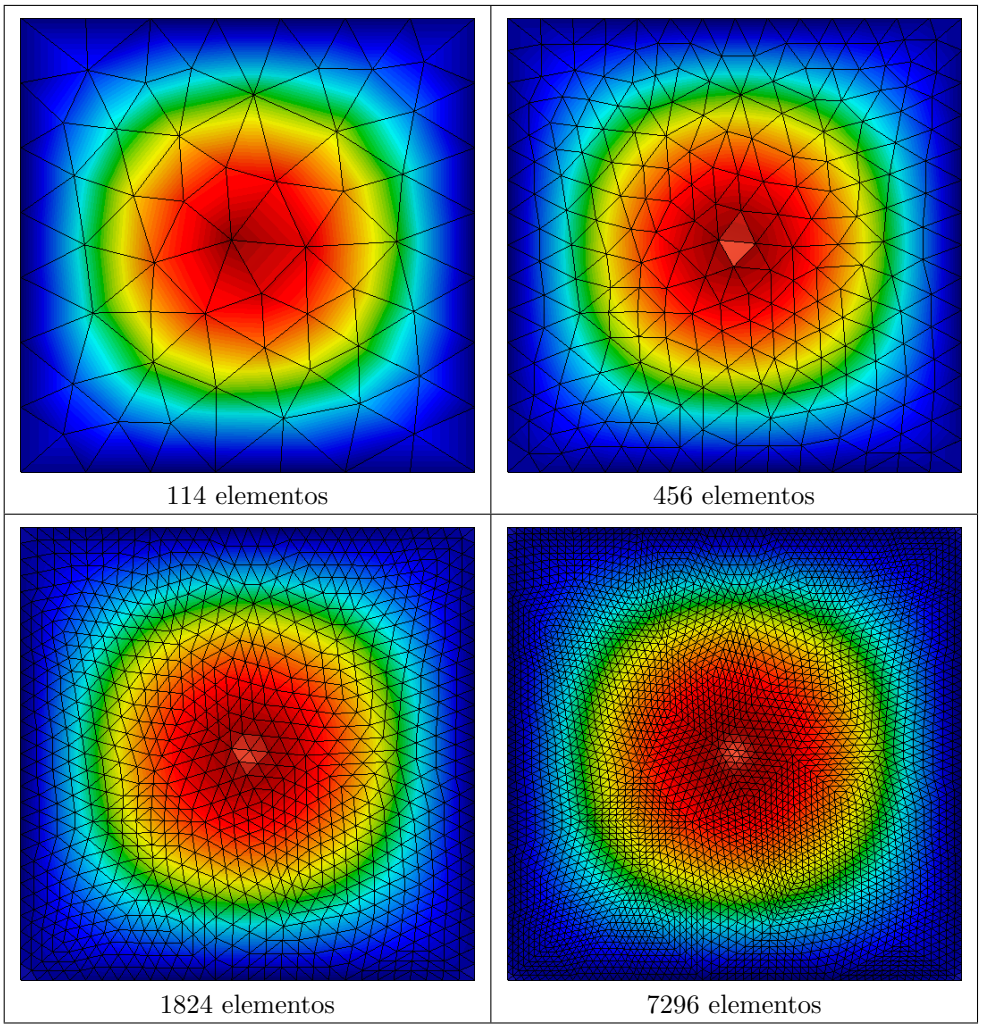


Figura 13 – Imagens topográficas da malha unEstru.mesh com diferentes quantidades de elementos. Fonte: Autor, 2023

Para cada iteração nas figuras 11, 12 e 13 é possível perceber como as curvas de nível acompanham os elementos de cada malha e como elas convergem para o formato circular dada função de aproximação ser uma senóide. Não foram mostradas todas as iterações porque a grande quantidade de elementos deixava a visibilidade do conjunto das curvas de nível e os limites dos elementos ruim.

7 Considerações Finais

Neste trabalho foi apresentado uma implementação para a solução da equação de Poisson utilizando o Método dos Elementos Finitos em malhas com elementos triangulares. Foram utilizados dois métodos numéricos iterativos diferentes para obter a solução do sistema linear. A análise de convergência do método foi feita com o uso de soluções manufaturadas. Bem como experimentos numéricos em três tipos de malhas triangulares: composta por triângulos retângulos, estruturada e não estruturada.

Em relação aos métodos numéricos utilizados para a resolução do sistema linear, o Método do Gradiente Conjugado Pré-condicionado (PCG) se mostrou extremamente mais

eficiente que o Método Iterativo Estacionário Jacobi Pré-condicionado (SLI). Através da análise do erro e da quantidade de iterações necessárias registradas para cada aproximação foi possível comprovar a eficiência do primeiro método. Verificamos que à medida que o sistema linear aumentava de tamanho, o SLI necessitava de aproximadamente 4 vezes mais iterações a cada refinamento. Portanto, o PCG obteve o melhor desempenho em todos os testes independente da malha utilizada.

Dentre as malhas fornecidas, a malha composta por triângulos retângulos (triang-ret.mesh) se mostrou mais eficiente dada sua disposição dos pontos, pois alcançou níveis de erro semelhantes às outras malhas, porém com menos elementos. Por consequência necessitou de menos iterações na resolução do sistema linear por ser um sistema menor. Entretanto, não é possível afirmar que a mesma malha será a mais eficiente para a aproximação da solução de outros problemas, ou seja, com funções diferentes da utilizada neste trabalho.

Trabalhos futuros, envolvendo problemas com outras condições de contorno e com malhas adaptativas devem ser considerados. A aproximação de soluções em malhas adaptativas seria vantajosa pois podemos ajustá-la para obter erros menores através de refinamentos localizados apenas em áreas de interesse. A biblioteca MFEM fornece suporte a esses tipos de malha, porém a estrutura do arquivo .mesh é diferente da utilizada neste trabalho, além de necessitar de ajustes de implementação.

Referências

ANDERSON, R. et al. MFEM: A modular finite element methods library. *Computers & Mathematics with Applications*, v. 81, p. 42–74, 2021. Disponível em: <https://doi.org/10.1016/j.camwa.2020.06.009>. Citado (6) vezes nas páginas [3, 4, 10, 14, 15 e 16].

AZEVEDO, A. F. M. *MÉTODO DOS ELEMENTOS FINITOS*. [S.l.: s.n.], 2003. 1–153 p. Citado na página [7].

BURDEN, R. L.; FAIRES, J. D. *Numerical Analysis*. [S.l.]: Cengage Learning, 2010. 450–495 p. ISBN 978-0-538-73351-9. Citado (2) vezes nas páginas [12 e 16].

CUNHA, M. C. C. *Métodos numéricos*. [S.l.: s.n.], 2000. ISBN 978-85-268-0877-5. Citado (3) vezes nas páginas [4, 12 e 13].

FISCHER, P. et al. Scalability of high-performance pde solvers. *The International Journal of High Performance Computing Applications*, v. 34, n. 5, p. 562–586, 2020. Disponível em: <https://doi.org/10.1177/1094342020915762>. Citado na página [4].

FRANGO, J. V. d. S. O MÉTODO DOS GRADIENTES CONJUGADOS PRÉ-CONDICIONADO POR FATORAÇÃO INCOMPLETA LU DE NÍVEL ZERO. 7 2018. Disponível em: <https://app.uff.br/riuff/bitstream/handle/1/11619/113035017%20Joseana%20Veiga%20de%20Souza%20Frango.pdf?sequence=1&isAllowed=y>. Citado na página [12].

JOHNSON, C. *Numerical solutions of partial differential equations by the finite element method*. [S.l.]: Cambridge University Press, 1988. 1–33 p. ISBN 978-05-21347-58-7. Citado (2) vezes nas páginas [3 e 4].

KREMER, I. MÉTODOS ITERATIVOS PARA SISTEMAS LINEARES. 2 2009. Disponível em: <https://repositorio.ufsc.br/xmlui/bitstream/handle/123456789/96623/Ivandra.pdf?sequence=1&isAllowed=y>. Citado na página [13].

LEE, J. H.; GRIFFITH, B. E. On the lagrangian-eulerian coupling in the immersed finite element/difference method. *Journal of Computational Physics*, v. 457, 2022. ISSN 0021-9991. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0021999122001048>. Citado na página [3].

LIU, W. K.; LI, S.; PARK, H. S. Eighty years of the finite element method: Birth, evolution, and future. *Archives of Computational Methods in Engineering*, 2022. Disponível em: <https://doi.org/10.1007/s11831-022-09740-9>. Citado na página [3].

MOZOLEVSKI, I.; MURAD, M. A.; SCHUH, L. A. High order discontinuous galerkin method for reduced flow models in fractured porous media. *Mathematics and Computers in Simulation*, v. 190, p. 1317–1341, 2021. ISSN 0378-4754. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0378475421002627>. Citado na página [3].

PEDROSO, N. M. C. M. O Método de Elementos Finitos: Aspectos da Implementação Computacional. 12 2019. Disponível em: <https://app.uff.br/riuff/bitstream/handle/1/22769/TCC_NataliaPedroso.pdf?sequence=1&isAllowed=y>. Citado (2) vezes nas páginas [4 e 9].

QUARTERONI, A.; SALERI, F. *CÁLCULO CIENTÍFICO com MATLAB e Octave*. [S.l.: s.n.], 2007. 240–270 p. ISBN 978-88-47007-17-8. Citado (2) vezes nas páginas [3 e 5].

RINCON, M. A.; LIU, I.-S. *Introdução ao Método de Elementos Finitos*. [S.l.]: Instituto de Matemática/UFRJ, 2020. 15–102 p. ISBN 978-65-86502-00-8. Citado (2) vezes nas páginas [3 e 10].

SAAD, Y. *Iterative Methods for Sparse Linear Systems*. [S.l.: s.n.], 2003. 47–108 p. ISBN 978-05-34947-76-7. Citado (6) vezes nas páginas [3, 4, 5, 10, 12 e 13].