Gabriel Christiano da Silva Alves

Resolução Numérica da Equação de Poisson pelo Método dos Elementos Finitos

Rio Grande, Rio Grande do Sul, Brasil

Fevereiro, 2025

Gabriel Christiano da Silva Alves

Resolução Numérica da Equação de Poisson pelo Método dos Elementos Finitos

Trabalho de Conclusão de Curso, Matemática Aplicada Bacharelado, submetido por Gabriel Christiano da Silva Alves junto ao Instituto de Matemática, Estatística e Física da Universidade Federal do Rio Grande.

Universidade Federal do Rio Grande - FURG

Instituto de Matemática, Estatística e Física - IMEF

Curso de Matemática Aplicada Bacharelado

Orientador: Prof.ª Dr.ª Juliana da Silva Ricardo Nunes

Coorientador: Prof. Dr. Igor Oliveira Monteiro

Rio Grande, Rio Grande do Sul, Brasil

Fevereiro, 2025

Gabriel Christiano da Silva Alves

Resolução Numérica da Equação de Poisson pelo Método dos Elementos Finitos

Trabalho de Conclusão de Curso, Matemática Aplicada Bacharelado, submetido por Gabriel Christiano da Silva Alves junto ao Instituto de Matemática, Estatística e Física da Universidade Federal do Rio Grande.

Trabalho aprovado. Rio Grande, 4 de fevereiro de 2025

Documento assinado digitalmente
gov.br JULIANA DA SILVA RICARDO NUNES
Data: 06/02/2025 12:34:48-0300
Verifique em https://validar.iti.gov.br

Prof.ª Dr.ª Juliana da Silva Ricardo Nunes
(Orientador - FURG)

Documento assinado digitalmente
gov.br IGOR OLIVEIRA MONTEIRO
Data: 06/02/2025 12:41:30-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Igor Oliveira Monteiro
(Coorientador - FURG)

Documento assinado digitalmente
gov.br ANDRE MENEGHETTI
Data: 06/02/2025 13:33:40-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. André Meneghetti
(Avaliador - FURG)

Documento assinado digitalmente
gov.br GUILHERME JAHNECKE WEYMAR
Data: 06/02/2025 19:50:00-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Guilherme Jahnecke
(Avaliador - UFPEL)

Rio Grande, Rio Grande do Sul, Brasil
Fevereiro, 2025

Dedico este trabalho a Deus, que me concedeu forças, sabedoria e paciência para concluir esta jornada. Sou eternamente grato por todas as bênçãos que recebi.

Agradecimentos

Primeiramente, agradeço a Deus, por me dar força, sabedoria e perseverança para enfrentar todos os desafios ao longo desta jornada.

Ao meu pai, Isaac de Mello Alves, minha eterna gratidão. Seu apoio incondicional, confiança e amor foram fundamentais para que eu seguisse em frente, com determinação e foco. Ele sempre foi uma fonte de inspiração, me ensinando a importância da disciplina, do trabalho árduo e da resiliência diante das dificuldades. Cada passo que dei foi apoiado pelo seu exemplo de força e dedicação. Gostaria também de expressar minha profunda gratidão à minha mãe, Miriã Cristiani Velho da Silva Alves, que, embora tenha falecido quando eu era muito pequeno, sempre foi uma fonte de inspiração em minha vida. Mesmo sem tê-la conhecido de forma plena, o legado dela me acompanhou, especialmente nos momentos de estudo, quando me lembro do esforço que ela também fez para alcançar seus objetivos. Sempre pensei nela como uma referência de superação e força, o que me motivou a seguir em frente.

Agradeço também à minha tia, Ana Luíza (Bá), que sempre esteve ao meu lado, oferecendo carinho e apoio emocional nos momentos mais difíceis. O amor e a dedicação de minha família foram um verdadeiro alicerce para mim, me sustentando ao longo dessa caminhada.

À minha orientadora, Professora Juliana, e ao coorientador, Professor Igor , sou imensamente grato pela paciência, dedicação e valiosos ensinamentos. Suas orientações foram indispensáveis para a realização deste trabalho e para meu crescimento acadêmico e pessoal.

À FURG e aos professores do IMEF, agradeço pelo ambiente acadêmico acolhedor e pelo suporte durante a minha formação, que me proporcionaram as bases necessárias para a realização deste estudo.

Por fim, gostaria de expressar minha gratidão aos amigos que fiz ao longo do curso. Cada um de vocês contribuiu de maneira única para minha jornada, seja com risadas, boas conversas ou pelo apoio constante. A amizade e o companheirismo de todos tornaram essa experiência ainda mais especial e inesquecível. Agradeço por tudo o que compartilhamos e aprendemos juntos.

A todos, meus mais sincero agradecimento.

Resumo

Neste trabalho, estuda-se a aplicação do Método dos Elementos Finitos (MEF) para resolver a equação de Poisson em um domínio unidimensional. O estudo do MEF é fundamental para a resolução de equações diferenciais parciais (EDPs) que, muitas vezes, são difíceis de resolver de forma analítica. Dessa forma, surgiu a motivação para a escolha deste tema, que está na importância de utilizar métodos numéricos eficientes, como o MEF, para solucionar equações diferenciais parciais, que muitas vezes não podem ser resolvidas de forma analítica devido à sua complexidade. A partir da formulação variacional da equação de Poisson, foi realizada a implementação computacional do MEF no programa GNU Octave, detalhando todas as etapas envolvidas no processo. Com o objetivo de observar, para diferentes soluções analíticas, como a solução numérica se aproxima da solução analítica à medida que a malha de discretização é refinada, ou seja, conforme aumentamos a quantidade de nós na malha, temos uma melhor visualização e eficácia do método. Com base nos resultados obtidos, foi possível analisar a eficácia da implementação do MEF utilizando o programa GNU Octave.

Palavras-chaves: Método dos Elementos Finitos, Equação de Poisson, Formulação Variacional, GNU Octave, Problema Unidimensional, Quadratura de Gauss Legendre.

Lista de ilustrações

Lista de tabelas

Sumário

Introdução

As equações diferenciais parciais (EDPs) são a base para modelar vários fenômenos. No entanto, a complexidade destas equações, na maioria das vezes, impede que soluções analíticas diretas sejam encontradas. Logo, para resolver equações que apresentam esse comportamento complexo, o ideal é buscar soluções numéricas que aproximem da solução do problema.

Com a evolução dos problemas de Matemática, Física e Engenharia, houve a necessidade de buscar outros métodos de resolução. Em meados do século XVIII, surge a teoria do método numérico, conhecido como o Método dos Elementos Finitos (MEF). Nessa ocasião, segundo Lotti et al. (2006), "quando Gauss propôs a utilização de funções de aproximação para a solução de problemas matemáticos", essa proposta foi gradualmente aprimorada e, com o tempo, que se tornou uma das abordagens mais amplamente utilizadas para a solução de EDPs complexas.

Dentro do desenvolvimento do MEF, um ponto importante foi a criação do Método de Ritz, proposto pelo matemático e físico suíço Walter Ritz em 1909. De acordo com Leissa (2005),"Em 1908 e 1909, Walter Ritz publicou dois artigos que demonstraram completamente um procedimento direto para resolver problemas de valor de contorno e autovalor numericamente, com qualquer grau de exatidão desejado, também usando funcionais de energia". Um trabalho teórico que serviu como base para o MEF.

Com o avanço da tecnologia e o desenvolvimento dos computadores na década de 50, a era digital iniciou uma nova etapa na resolução de problemas numéricos. A capacidade de realizar cálculos complexos, que até então eram feitos manualmente, passou a ser possível de forma rápida e eficiente, o que possibilitou a implementação prática de métodos como o MEF, que se tornou uma ferramenta viável para resolver problemas complexos em diversas áreas.

Na década de 70, o MEF passou a ser usado em problemas de mecânica dos fluidos, o que fez com que ele se tornasse ainda mais relevante. Desde então, o MEF tem se mostrado um método eficaz e adaptável, sendo cada vez mais aplicado para resolver vários tipos de EDPs em diferentes áreas Brunow (2017).

O MEF é uma técnica numérica que divide o domínio do problema em pequenas partes, chamadas de elementos finitos Becker, Carey e Oden (1981). No caso bidimensional, por exemplo, esse domínio pode ser decomposto em sub-regiões no formato triangular, quadrilateral. A coleção de elementos finitos é conhecida como Malha de elementos finitos.

Segundo Calle, Devloo e Gomes (2004), o nome de MEF está associado ao método

de Galerkin com funções aproximantes contínuas e polinomiais por partes. Este método é muito eficaz para lidar com equações diferenciais complicadas, principalmente quando não conseguimos encontrar uma solução analiticamente.

De acordo com Oden, Carey e Becker (1981), "A generalidade e a riqueza dos conceitos são razões para o seu notório sucesso em uma vasta gama de problemas". Entre os problemas que podem ser aplicados o método estão: aerodinâmica Hashimoto, Suzuki e Nakamura (1984), mecânica de fluidos Teixeira e Awruch (2001), eletromagnetismo Augustyniak e Usarek (2016), análise estrutural Souza et al. (2018), navegação cirúrgica Eltes et al. (2021), e entre outros. Na matemática, por exemplo, podemos utilizar o método para a construção de soluções aproximadas para problemas de valor de contorno. Em destaque aparecem no desenvolvimento da teoria, os matemáticos Richard Courant, Walther Ritz e Boris Galerkin.

Dentre as EDPs, temos a equação de Poisson, que é uma equação elíptica frequentemente utilizada como ponto de partida para o estudo e aplicação do MEF. A equação de Poisson é aplicada em várias áreas, como potencial eletrostático Schnitzer e Lambrakis (1991), dinâmica dos fluidos Sterza et al. (2020) e geofísica Ince et al. (2020), entre outras.

Neste trabalho, seguimos a abordagem proposta por Johnson (1987), utilizando o MEF para resolver a equação de Poisson em um domínio unidimensional. A implementação será realizada por meio do programa GNU Octave, com o objetivo de mostrar como a solução numérica se aproxima da solução analítica à medida que aumentamos a quantidade de nós na malha. A seguir, este trabalho está organizado da seguinte forma: No Capítulo 1, apresentamos a fundamentação matemática necessária para compreender o MEF, incluindo definições e teoremas relevantes. No Capítulo 2, abordamos a teoria do MEF e, em seguida, sua aplicação ao problema de Poisson, incluindo a formulação variacional, a construção da matriz de rigidez e do vetor de carga, bem como a resolução do sistema linear resultante. O Capítulo 3 apresenta uma introdução ao GNU Octave e descreve brevemente o ambiente de programação utilizado para a implementação do MEF. Já no Capítulo 4, detalhamos a implementação do MEF no Octave, incluindo o desenvolvimento dos códigos utilizados e a análise de um problema teste. No Capítulo 5, resolvemos exemplos práticos com diferentes soluções analíticas, permitindo visualizar a relação entre as soluções numéricas obtidas e as soluções analíticas, evidenciando como a solução numérica se aproxima da solução analítica à medida que a malha de discretização é refinada. Finalmente, no Capítulo 6, apresentamos as conclusões deste trabalho e sugerimos possíveis direções para trabalhos futuros.

Com isso, esperamos proporcionar uma compreensão clara do MEF e sua eficácia na resolução de EDPs, além de demonstrar a importância da implementação prática e da visualização dos resultados.

1 Fundamentação Matemática

1.1 Breve Introdução

Neste capítulo, apresentam-se algumas definições matemáticas que são essenciais, obtidas de livros de referência na área, como os de Johnson (1987), Farlow (1993), Anton e Rorres (2001), Iório (2005), Burden e Faires (2008), Lima (2009), Thomas, Weir e Hass (2012), Lima (2014), Benitez (2017). Essas definições fornecem o alicerce teórico necessário para a compreensão dos conceitos que serão utilizados ao longo deste trabalho, servindo como base para o desenvolvimento das ideias e técnicas apresentadas.

1.2 Equação Diferencial Parcial

Definição 1.2.1. Equação Diferencial Parcial - EDP:

Uma EDP é uma equação que envolve duas ou mais variáveis independentes $x, y, z, t, \ldots$ e as derivadas parciais de uma função (variável dependente) $u = u(x, y, z, t, \ldots)$. De maneira mais precisa, uma EDP em n variáveis independentes $x_1, \ldots, x_n$ é uma equação da forma

$$F\left(x_1, \ldots, x_n, u, \frac{\partial u}{\partial x_1}, \ldots, \frac{\partial u}{\partial x_n}, \frac{\partial^2 u}{\partial x_1^2}, \ldots, \frac{\partial^2 u}{\partial x_1 \partial x_n}, \ldots, \frac{\partial^k u}{\partial x_n^k}\right) = 0,$$

onde $x = (x_1, \ldots, x_n) \in \Omega$, Ω é um subconjunto aberto de $\mathbb{R}^n$, F é uma função dada e $u = u(x)$ é a função que queremos determinar.

Definição 1.2.2. Classificação de Equações Diferenciais Parciais (EDPs):

As EDPs são classificadas de acordo com vários critérios. A classificação é um conceito importante, pois a teoria geral e os métodos de solução geralmente se aplicam apenas a uma classe específica de equações. Seis classificações básicas são apresentadas a seguir:

1. **Ordem da EDP**: A ordem de uma EDP é a ordem da derivada parcial mais alta na equação. Por exemplo, nas equações:

$$u_t = u_x, \quad \text{(primeira ordem)}$$

$$u_t = u_{xx}, \quad \text{(segunda ordem)}$$

2. **Número de Variáveis**: O número de variáveis independentes é o número de variáveis que a função desconhecida depende. Por exemplo:

$$u_t = u_{xx} \quad \text{(duas variáveis: } x \text{ e } t\text{)}$$

3. **Linearidade**: Uma EDP é linear se a função desconhecida e suas derivadas aparecem de forma linear, ou seja, sem produtos ou potências das variáveis. Um exemplo de EDP linear de segunda ordem em duas variáveis é:

$$Au_{xx} + Bu_{xy} + Cu_{yy} + Du_x + Eu_y + Fu = G,$$

Onde A, B, C, D, E, F são coeficientes constantes.

4. **Homogeneidade**: A equação 3 é chamada homogênea se o lado direito da equação for identicamente zero para todas as variáveis dependentes. Se o lado direito não for zero, a equação é não homogênea.

5. **Tipos de Coeficientes**: Se os coeficientes A, B, C, D, E e F na equação 3 são constantes, então a equação é dita ter coeficientes constantes. Caso contrário, os coeficientes podem variar com as variáveis independentes.

6. **Três Tipos Básicos de Equações Lineares**: As EDPs lineares podem ser classificadas em três tipos principais:

   a) **Equações Parabólicas:** $B^2 - 4AC = 0$.

   b) **Equações Elípticas:** $B^2 - 4AC < 0$.

   c) **Equações Hiperbólicas:** $B^2 - 4AC > 0$.

Definição 1.2.3. Condições de Contorno:

Existem diferentes tipos de condições de contorno, entre elas:

- **Condição de Dirichlet**: Especifica o valor da função na fronteira do domínio. É expressa por:

$$u(x) = g(x) \quad \text{para} \quad x \in \partial\Omega$$

Onde $u(x)$ é a solução da equação diferencial parcial (EDP), e $g(x)$ é uma função dada que define o valor da solução na borda $\partial\Omega$ do domínio Ω.

- **Condição de Neumann**: Define o valor da derivada normal da função na fronteira do domínio, ou seja, estabelece a taxa de variação da solução ao longo da fronteira. A equação é dada por:

$$\frac{\partial u}{\partial n}(x) = h(x) \quad \text{para} \quad x \in \partial\Omega$$

Onde $\frac{\partial u}{\partial n}(x)$ é a derivada normal de u na borda $\partial\Omega$, e $h(x)$ é uma função dada que define a taxa de variação da solução na borda do domínio.

Definição 1.2.4. Equação de Poisson:

Dado o domínio Ω e $f : \Omega \to \mathbb{R}$ uma função contínua, a equação de Poisson é dada por:

$$-\triangle u(x) = f(x), \quad x \in \Omega \tag{1.1}$$

Onde $\triangle$ é o laplaciano, que representa a soma das segundas derivadas de $u(x)$ em relação às suas variáveis espaciais.

1.3 Análise Real

Definição 1.3.1. Função de Classe C^0:

Seja $f : D \to \mathbb{R}$ uma função com domínio $D \subseteq \mathbb{R}$. Então, f é dita ser de classe $C^0(D)$ se for uma função contínua.

Definição 1.3.2. Função Contínua:

Uma função $f : X \to \mathbb{R}$, definida no conjunto $X \subset \mathbb{R}$, diz-se contínua no ponto $a \in X$ quando, para todo $\epsilon > 0$ dado arbitrariamente, pode-se obter $\delta > 0$ tal que, se $x \in X$ e $|x - a| < \delta$, impliquem $|f(x) - f(a)| < \epsilon$. Em símbolos, f contínua no ponto a significa:

$$\forall \epsilon > 0,\ \exists \delta > 0; x \in X,\ |x - a| < \delta \Rightarrow |f(x) - f(a)| < \epsilon.$$

Definição 1.3.3. Função Derivável:

Sejam $f : X \to \mathbb{R}$ e $a \in X \cap X'$. A derivada da função f no ponto a é o limite

$$f'(a) = \lim_{x \to a} \frac{f(x) - f(a)}{x - a} = \lim_{h \to 0} \frac{f(a + h) - f(a)}{h}.$$

Bem entendido, o limite acima pode existir ou não. Se existir, diz-se que f é derivável no ponto a. Quando existe a derivada $f'(x)$ em todos os pontos $x \in X \cap X'$, diz-se que a função $f : X \to \mathbb{R}$ é derivável no conjunto X e obtém-se uma nova função $f' : X \cap X' \to \mathbb{R}, x \mapsto f'(x)$, chamada a função derivada de f. Se f' é contínua, diz-se que f é de classe C^1.

Outras notações para a derivada de f no ponto a são

$$Df(a), \quad \frac{df}{dx}(a) \quad \text{e} \quad \left.\frac{df}{dx}\right|_{x=a}.$$

Teorema 1.3.1. Teorema Fundamental do Cálculo:

Seja $f : I \to \mathbb{R}$ uma função contínua no intervalo I. As seguintes afirmações a respeito de uma função $F : I \to \mathbb{R}$ são equivalentes:

1. F é uma integral indefinida de f, isto é, existe $a \in I$ tal que $F(x) = F(a) + \int_a^x f(t)\,dt$, para todo $x \in I$.
2. F é uma primitiva de f, isto é, $F'(x) = f(x)$ para todo $x \in I$.

Teorema 1.3.2. Regra da Substituição:

Se $u = g(x)$ for uma função derivável cuja imagem é um intervalo I e f for contínua em I, então:

$$\int f(g(x))g'(x)\,dx = \int f(u)\,du. \tag{1.2}$$

Teorema 1.3.3. Integração por Partes:

Se $f, g : [a, b] \to \mathbb{R}$ têm derivadas contínuas, então:

$$\int_a^b f(x)g'(x)\,dx = [f(x)g(x)]_a^b - \int_a^b f'(x)g(x)\,dx. \tag{1.3}$$

Definição 1.3.4. Funcional Linear

Seja V um espaço linear sobre os reais $\mathbb{R}$, então uma aplicação $f : V \to \mathbb{R}$ é chamada de funcional linear em V, se:

$$f(cv_1 + v_2) = cf(v_1) + f(v_2), \quad \forall v_1, v_2 \in V,\, c \in \mathbb{R}. \tag{1.4}$$

1.4 Álgebra Linear

Definição 1.4.1. Espaço Vetorial

Definição: Um espaço vetorial v sobre o conjunto dos números reais $\mathbb{R}$ (ou o conjunto dos números complexos $\mathbb{C}$) consiste de um conjunto v munido da operação $+$ satisfazendo os seguintes axiomas:

- $(\mathbf{A_0})$ Para $V, W \in v$, temos que $V + W \in v$ (fechamento).
- $(\mathbf{A_1})$ Para $V, W \in v$, temos que $V + W = W + V$ (comutatividade).
- $(\mathbf{A_2})$ Para $U, V, W \in v$, temos que $U + (V + W) = (U + V) + W$ (associatividade).
- $(\mathbf{A_3})$ Existe um elemento $0 \in V$ tal que $0 + V = V + 0 = V$ (existência do neutro).
- $(\mathbf{A_4})$ Para qualquer $V \in v$, existe um elemento $-V \in v$ tal que $V + (-V) = (-V) + V = 0$ (existência do inverso).

Existe uma multiplicação de elementos v por números de $\mathbb{R}$ (ou de $\mathbb{C}$) satisfazendo os axiomas:

- ($\mathbf{M_0}$) Para λ em $\mathbb{R}$ (ou em $\mathbb{C}$), e $V \in v$, λV é um elemento de v (fechamento).
- ($\mathbf{M_1}$) Se 1 é a identidade da multiplicação em $\mathbb{R}$ (ou em $\mathbb{C}$), então $1V = V$ qualquer que seja V em v (existência da identidade para a multiplicação por escalar).
- ($\mathbf{M_2}$) Para $\lambda_1, \lambda_2 \in \mathbb{R}$ (ou em $\mathbb{C}$), e $V \in v$, tem-se que $\lambda_1(\lambda_2 V) = (\lambda_1\lambda_2)V$ (associatividade).
- ($\mathbf{AM_1}$) Para λ em $\mathbb{R}$ (ou em $\mathbb{C}$), e $V, W \in v$, tem-se que $\lambda(V + W) = \lambda V + \lambda W$ (distributividade).
- ($\mathbf{AM_2}$) Para λ_1, λ_2 em $\mathbb{R}$ (ou em $\mathbb{C}$), e $V \in v$, tem-se que $(\lambda_1 + \lambda_2)V = \lambda_1 V + \lambda_2 V$ (distributividade).

Definição 1.4.2. Subespaço Vetorial

Seja E um espaço vetorial. Um **subespaço vetorial** (ou simplesmente um **subespaço**) de E é um subconjunto $F \subset E$ com as seguintes propriedades:

1. $0 \in F$;
2. Se $u, v \in F$, então $u + v \in F$;
3. Se $v \in F$, então, para todo $\alpha \in \mathbb{R}$, $\alpha v \in F$.

Definição 1.4.3. Matriz:

Conjunto de números reais (ou complexos) dispostos em forma de tabela, isto é, distribuídos em m linhas e n colunas, sendo m e n números naturais N = {1,2,3,...}.

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix}.$$

Notação: $A = (a_{ij})_{m \times n}$, onde:

- a_{ij} - elemento genérico da matriz A,
- i - índice que representa a linha do elemento a_{ij},
- j - índice que representa a coluna do elemento a_{ij},
- $m \times n$ - ordem da matriz. Lê-se "m por n".

Definição 1.4.4. Produto Interno:

Um ***produto interno*** em um espaço vetorial real V é uma função que associa cada par de vetores a um número real $(\mathbf{u}, \mathbf{v})$ a cada par de vetores $\mathbf{u}$ e $\mathbf{v}$ em V de tal maneira que os seguintes axiomas são satisfeitos para quaisquer vetores $\mathbf{u}$,$\mathbf{v}$, e $\mathbf{w}$ de V e qualquer escalar l:

1. $(\mathbf{u}, \mathbf{v}) = (\mathbf{v}, \mathbf{u})$ **[Axioma de simetria]**
2. $(\mathbf{u} + \mathbf{v}, \mathbf{w}) = (\mathbf{u}, \mathbf{w}) + (\mathbf{v}, \mathbf{w})$ **[Axioma de aditividade]**
3. $(l\mathbf{u}, \mathbf{v}) = l(\mathbf{u}, \mathbf{v})$ **[Axioma de homogeneidade]**
4. $(\mathbf{v}, \mathbf{v}) \geq 0$ **[Axioma de positividade]**
   e $(\mathbf{v}, \mathbf{v}) = 0$ se, e somente se, $\mathbf{v} = \mathbf{0}$

Um espaço vetorial real com um produto interno é chamado de ***espaço com produto interno real***.

Definição 1.4.5. Produto Interno em $C[a, b]$:

Sejam $\mathbf{f} = f(x)$ e $\mathbf{g} = g(x)$ duas funções contínuas em $C[a, b]$. O produto interno $(\mathbf{f}, \mathbf{g})$ é definido por

$$(\mathbf{f}, \mathbf{g}) = \int_a^b f(x)g(x)\,dx. \tag{1.5}$$

Definição 1.4.6. Transformação Linear:

Se $T : V \to W$ é uma função de um espaço vetorial V em um outro espaço vetorial W, então T é chamada de **transformação linear** de V em W se, para quaisquer vetores $\mathbf{u}$ e $\mathbf{v}$ em $\mathbf{V}$ e qualquer escalar c, valem:

1. $T(\mathbf{u} + \mathbf{v}) = T(\mathbf{u}) + T(\mathbf{v})$
2. $T(c\mathbf{v}) = cT(\mathbf{v})$

Definição 1.4.7. Matriz Quadrada:

Matriz Quadrada é aquela cujo número de linhas é igual ao número de colunas $(m = n)$. Exemplos:

$$\begin{bmatrix} 1 & -2 & 0 \\ 3 & 0 & 1 \\ 4 & 5 & 6 \end{bmatrix}.$$

No caso de matrizes quadradas $A_{m\times m}$, costumamos dizer que A é uma matriz de ordem m.

Definição 1.4.8. Matriz Diagonal:

Uma matriz quadrada na qual todas as entradas fora da diagonal principal são iguais a zero é chamada de **matriz diagonal**.

Definição 1.4.9. Transposta de uma Matriz:

Se A é uma matriz $m \times n$ qualquer, então a **transposta de** A, denotada por A^T, é definida como a matriz $n \times m$ que resulta da permutação das linhas com as colunas de A; ou seja, a primeira coluna de A^T é a primeira linha de A, a segunda coluna de A^T é a segunda linha de A, e assim por diante.

Definição 1.4.10. Matrizes Simétricas:

Uma matriz quadrada A é chamada de **simétrica** se $A = A^T$.

Definição 1.4.11. Matriz Definida Positiva:

Se a matriz simétrica $A_{n\times n}$ tem a propriedade $x^\top Ax > 0$ para todo vetor $x \in \mathbb{R}^n$, exceto $x = 0$, então a forma quadrática $x^\top Ax$ é dita definida positiva e A é uma matriz definida positiva.

Definição 1.4.12. Matriz Singular:

Dada uma matriz quadrada A, se pudermos encontrar uma matriz B de mesmo tamanho tal que $AB = BA = I$, então diremos que A é invertível e que B é uma inversa de A. Se não puder ser encontrada uma tal matriz B, então diremos que A é não-invertível ou singular.

Teorema 1.4.1. Toda matriz $A_{n\times n}$ positiva definida é inversível e sua inversa é também definida positiva.

Definição 1.4.13. Sistema Linear:

Sejam m e n números naturais maiores ou iguais a 1 ($m, n \geq 1$). Um sistema linear S, de m equações com n incógnitas (ou simplesmente sistema linear $m \times n$) é um conjunto de m equações lineares, cada uma delas com n incógnitas, consideradas simultaneamente. Denotamos um tal sistema linear como segue:

$$S = \begin{cases} \alpha_{11} \cdot x_1 + \alpha_{12} \cdot x_2 + \cdots + \alpha_{1n} \cdot x_n = \beta_1 \\ \alpha_{21} \cdot x_1 + \alpha_{22} \cdot x_2 + \cdots + \alpha_{2n} \cdot x_n = \beta_2 \\ \quad \vdots \\ \alpha_{i1} \cdot x_1 + \alpha_{i2} \cdot x_2 + \cdots + \alpha_{in} \cdot x_n = \beta_i \\ \quad \vdots \\ \alpha_{m1} \cdot x_1 + \alpha_{m2} \cdot x_2 + \cdots + \alpha_{mn} \cdot x_n = \beta_m \end{cases}$$

Vamos denotar a i-ésima dessas equações ($1 \leq i \leq m$) por eq(i).

Definição 1.4.14. Método de Galerkin:

O método de Galerkin aproxima o Espaço de Dimensão Infinita por um Espaço de Dimensão Finita e, com isso, obtemos um Problema Variacional Discreto. Ao escolher uma base para esse espaço de dimensão finita $v_k \in V_n$, transformamos o Problema Variacional Discreto em um Sistema Linear de Equações Algébricas, conforme apresentado abaixo:

$$A_{n\times n}U = F.$$

Aqui se destacam dois casos para a escolha da Base:

1. **Base Global** (Método de Galerkin Original): a matriz A é densa (cheia).
2. **Base Local** (Método de Elementos Finitos): a matriz A é esparsa. Usa menos memória e facilita a obtenção da solução numérica.

1.5 Cálculo Numérico

Nesta seção, apresentamos algumas definições de cálculo numérico, as quais podem ser encontradas no livro de *Análise Numérica* de Burden e Faires (2008).

Definição 1.5.1. Regra do Ponto Médio:

$$\int_a^b f(x)\,dx \approx (b-a)\cdot f\left(\frac{a+b}{2}\right). \tag{1.6}$$

Definição 1.5.2. Regra do Trapézio:

$$\int_a^b f(x)\,dx \approx \frac{b-a}{2}\left[f(a)+f(b)\right]. \tag{1.7}$$

Definição 1.5.3. Regra de Simpson:

$$\int_a^b f(x)\,dx \approx \frac{b-a}{6}\left[f(a)+4f\left(\frac{a+b}{2}\right)+f(b)\right]. \tag{1.8}$$

Definição 1.5.4. Quadratura de Gauss-Legendre:

$$\int_a^b f(x)\,dx \approx \frac{b-a}{2}\sum_{i=1}^{n} c_i f\left(\frac{b-a}{2}x_i+\frac{a+b}{2}\right). \tag{1.9}$$

As constantes c_i e as raízes dos polinômios de Legendre são necessárias para a regra da quadratura. A Tabela 1 lista os valores para $n = 2$, 3 e 4.

Tabela 1 – Tabela de Gauss-Quadratura para 2, 3 e 4 pontos

Número de pontos	$\mathbf{x_i}$	$\mathbf{c_i}$
2 pontos	$-0,577350$	1
	$0.,77350$	1
3 pontos	$-0,774597$	$0,555556$
	0	$0,888889$
	$0,774597$	$0,555556$
4 pontos	$-0,861136$	$0,347855$
	$-0,339981$	$0,652145$
	$0,339981$	$0,652145$
	$0,861136$	$0,347855$

2 METODOLOGIA

2.1 Breve Introdução

Neste capítulo será apresentada a teoria do Método dos Elementos Finitos (MEF), que se baseia na discretização do domínio do problema em pequenas sub-regiões, chamadas de elementos finitos. Em seguida, detalha-se a aplicação do MEF ao problema de Poisson, abordando tanto a formulação variacional quanto a de minimização, destacando a equivalência na formulação variacional e de minimização do problema, e a equivalência entre essas formulações. A geração dos gráficos apresentados neste capítulo foi realizada utilizando o software GNU Octave.

2.2 O Método dos Elementos Finitos

Inicialmente, o estudo do problema unidimensional no MEF serve como base para a compreensão do método. A partir desse estudo, podemos estender o conceito para problemas em dimensões maiores. Para exemplificar a discretização do domínio em duas dimensões no MEF, observe a Figura 1 de Souza (2003). Consiste em dividir o domínio do problema em sub-regiões que podem ser no formato triangular ou quadrilateral. No exemplo abaixo, o domínio bidimensional é dividido em triângulos.

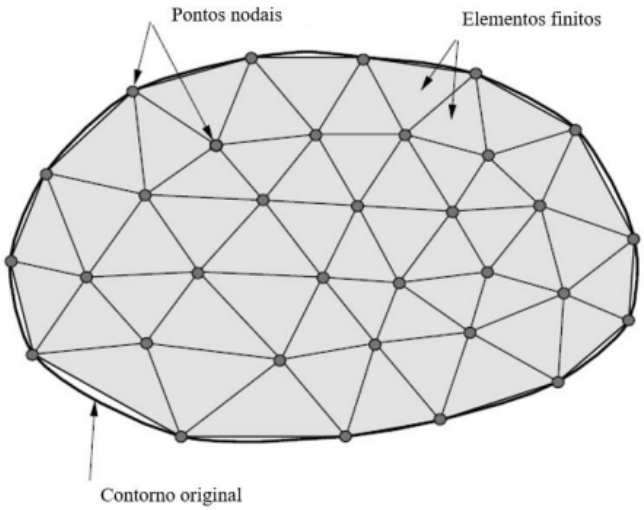


Figura 1 – Malha com sub-regiões triângulares.

Fonte: Remo Magalhães de Souza , 2003

As sub-regiões que surgem são chamadas de "elementos finitos". Na figura também é possível observar que estas regiões estão conectadas por pontos, chamados nós ou pontos nodais. Quando pensamos no conjunto dos elementos finitos, formamos a chamada malha. A malha possui um papel fundamental no método, pois quanto mais elementos, mais precisos podem ser os resultados. No método, substituímos um problema de valor de contorno por uma formulação variacional, o que nos leva, junto com a discretização do domínio, a um sistema de equações lineares para resolver numericamente o problema, como será mostrado a seguir.

2.3 Problema de Poisson - Caso Unidimensional

Seguindo a abordagem apresentada no livro de Johnson (1987), o problema de Poisson em uma dimensão é dado por

$$\begin{cases} -u''(x) = f(x) & \text{para } 0 < x < 1, \\ u(0) = u(1) = 0, \end{cases} \tag{D}$$

Onde $f(x)$ é uma função dada, e as condições de contorno $u(0) = u(1) = 0$ correspondem à condição de Dirichlet. O intervalo considerado é $[0, 1]$.

Vamos mostrar que o problema de valor de contorno, Equação (D) ,pode ser resolvido de forma equivalente nas formulações Variacional (V) e de Minimização (M) que são destacadas a seguir. Com isso, ao mostrar a equivalência entre (D), (V) e (M), destacamos as diferentes maneiras de abordar a solução do mesmo problema matemático. Conceitos que apresentaremos abaixo.

Considere o espaço linear V que é definido como o conjunto das funções v que são contínuas em $[0, 1]$, tais que v' é contínua por partes e limitada em $[0, 1]$, e $v(0) = v(1) = 0$.

Notação do espaço V:

$$V = \left\{ \begin{array}{l} \quad v \text{ é contínua em } [0,1], \\ v \mid v' \text{ é contínua por partes e limitada em } [0,1], \\ \quad v(0) = v(1) = 0 \end{array} \right\}. \tag{2.1}$$

Considere o funcional linear $F : V \to \mathbb{R}$ que é definido como:

$$F(v) = \frac{1}{2}(v', v') - (f, v) \tag{2.2}$$

onde $(\cdot, \cdot)$ representa o produto interno em V.

Teorema 2.3.1. Considere o problema de valor de contorno (D). São equivalentes:

(1) u é solução do problema de valor de contorno (D).

(2) u é solução do problema na formulação variacional (V), que é

$$u \in V \text{ tal que } (u', v') = (f, v), \quad \forall\, v \in V. \tag{V}$$

(3) u é solução do problema na formulação de minimização (M), que é

$$u \in V \text{ tal que } F(u) \leq F(v), \quad \forall\, v \in V. \tag{M}$$

Demonstração. Primeiro vamos mostrar que (D) é equivalente a (V).
Supondo que u é solução de (D). Consideramos a equação diferencial de Poisson em (D):

$$-u'' = f. \tag{2.3}$$

Multiplicamos a equação (2.3) por uma função teste arbitrária $v \in V$ definido em (2.1) e integrando em $[0, 1]$, temos o produto interno:

$$-(u'', v) = (f, v).$$

Ou equivalente, a definição (1.5) no intervalo $[0, 1]$,

$$-\int_0^1 u''v\, dx = \int_0^1 fv\ dx = (f, v).$$

Agora, vamos integrar o lado esquerdo da igualdade, usando o teorema de integração por partes (1.3):

$$\int_0^1 u''v\, dx = u'v|_0^1 - \int_0^1 u'v'\, dx.$$

Aplicando ao nosso intervalo $[0, 1]$, temos:

$$\int_0^1 -u''v\, dx = -u'v|_0^1 + \int_0^1 u'v'\, dx.$$

Como $v \in V$, temos os valores de v nos contornos $v(0) = v(1) = 0$. Aplicando no termo $-u'v|_0^1$, ele se anula. Portanto, obtemos:

$$\int_0^1 -u''v\, dx = \int_0^1 u'v'\, dx = (u', v').$$

Segue que

$$\int_0^1 u'v'\, dx = \int_0^1 fv\, dx,$$

ou seja, pela definição (1.5), segue que

$$(u', v') = (f, v).$$

Logo a u satisfaz (V), $\forall v \in V$. Aqui foi mostrado que (D) $\implies$ (V). Agora, vamos mostrar que

- (V) $\Leftrightarrow$ (M):

Primeiro, vamos mostrar que o problema (V) equivale a (M).
Suponha que u é uma solução para o problema variacional (V) e considere $v \in V$. Como u e $v \in V$, podemos definir $w = v - u \in V$. Aplicando o funcional linear definido na Equação (2.2) a $v = w + u$, temos

$$F(v) = F(u + w) = \frac{1}{2}(u' + w', u' + w') - (f, u + w).$$

Utilizando as propriedades de produto interno,

$$F(u + w) = \frac{1}{2}[(u', u') + 2(u', w') + (w', w')] - [(f, u) + (f, w)]. \tag{2.4}$$

Simplificando a expressão:

$$F(v) = F(u + w) = \frac{1}{2}(u', u') + (u', w') + \frac{1}{2}(w', w') - (f, u) - (f, w).$$

Como u é solução do problema (V) , e $w \in V$, pela Equação V

$$(u', w') = (f, w).$$

Então substituindo em (2.4), ficamos com a equação

$$F(v) = F(u + w) = \frac{1}{2}(u', u') + \frac{1}{2}(w', w') - (f, u).$$

Também temos pela definição do funcional que

$$F(u) = \frac{1}{2}(u', u') - (f, u)$$

e pela propriedade do produto interno que

$$(w', w') \geqslant 0.$$

Portanto u é solução de (M), ao desprezar

$$(w', w') \geqslant 0,$$

podemos concluir

$$F(v) \geq F(u), \forall v \in V.$$

Agora vamos mostrar a recíproca que se u é solução de (M) então é de (V):

(M) diz que u minimiza F, isto é

$$F(u) \leq F(v), \quad \forall v \in V.$$

Então, consequentemente

$$F(u) \leq F(u + \epsilon v).$$

Seja

$$g(\epsilon) = F(u + \epsilon).$$

Portanto, aplicando a definição do funcional, temos:

$$g(\epsilon) = F(u + \epsilon v) = \frac{1}{2}[(u' + \epsilon v', u' + \epsilon v')] - (f, u + \epsilon v).$$

Utilizando as propriedades de produto interno,

$$g(\epsilon) = \frac{1}{2}(u', u') + \epsilon(u', v') + \frac{\epsilon^2}{2}(v', v') - (f, u) - \epsilon(f, v).$$

Derivando $g(\epsilon)$ em relação a ϵ, obtemos:

$$g'(\epsilon) = (u', v') + \epsilon(v', v') - (f, v).$$

Fazendo $\epsilon = 0$, temos:

$$g'(0) = (u', v') - (f, v). \tag{2.5}$$

Note que $g(0) = F(u)$ e $F(u)$ é o menor valor real que F assume $\forall v \in V$. Assim, $g(0)$ é mínimo e, portanto , $g'(0)$ deve ser zero, isto é, $g'(0) = 0$. Isso implica que em 2.5

$$(u', v') = (f, v), \forall\ v \in V.$$

- (V) $\implies$ (D):

Para finalizar, supondo que u é solução do problema variacional (V), então pela definição de produto interno aplicado no intervalo [0,1] (1.5), temos

$$\int_0^1 u'v'\,dx - \int_0^1 fv\,dx = 0 \quad \forall v \in V. \tag{2.6}$$

Integrando o primeiro termo do lado esquerdo da equação com o teorema de integração por partes:

$$\int_0^1 u'v'\,dx = u'v|_0^1 - \int_0^1 u''v\,dx.$$

Como $v(0) = v(1) = 0$, o termo de borda $u'v|_0^1$ se anula. Portanto

$$\int_0^1 u'v'\,dx = -\int_0^1 u''v\;dx. \tag{2.7}$$

Usando a Equação na (2.7) na expressão (2.6), temos:

$$-\int_0^1 u''v\,dx - \int_0^1 fv\,dx = 0.$$

Reescrevendo,

$$-\int_0^1 (u'' + f)v\,dx = 0.$$

Como $v' \neq 0$ e u'' é contínua, concluímos que:

$$u'' + f = 0 \quad \text{ou} \quad -u'' = f.$$

Portanto, u satisfaz o problema (D).

Com esse resultado mostramos que os problemas (D),(V) e (M) são equivalentes. □

Teorema 2.3.2. Se u é solução de (V), então a solução é única.

Demonstração. Mostraremos que uma solução para (V) é única. Suponha que u_1 e u_2 são ambas soluções do problema variacional (V), então

$$(u_1', v') = (f, v)$$

e

$$(u_2', v') = (f, v).$$

Subtraindo as equações e aplicando a propriedade de produto interno, obtemos:

$$(u_1' - u_2', v') = 0.$$

Aplicando a definição de produto interno em $[0, 1]$, temos

$$\int_0^1 [u_1'(x) - u_2'(x)]\;\; v'(x)dx = 0.$$

Escolhemos $v = u_1 - u_2$, temos $v' = u_1' - u_2'$ e substituindo na integral, obtemos:

$$\int_0^1 (u_1'(x) - u_2'(x))(u_1'(x) - u_2'(x))\,dx = \int_0^1 [u_1'(x) - u_2'(x)]^2\,dx = 0.$$

Como o integrando $(u_1' - u_2')^2$ é não-negativo e a integral é zero, isso implica que

$$u_1' - u_2' = 0.$$

Segue que

$$u_1 - u_2 = C,$$

onde C é uma constante de integração.
Aplicando as condições de contorno $u_1(0) = u_2(0) = 0$ e $u_1(1) = u_2(1) = 0$, segue que a constante $C = 0$. Assim, concluímos que

$$u_1 = u_2.$$

Isso prova que a solução é única. □

Na próxima seção apresentaremos como usar o MEF para encontrar uma solução numérica para o problema de Poisson. Ao final, para aplicarmos o método, precisamos resolver um sistema de equações lineares.

2.4 Aproximação da função solução do problema de Poisson via MEF usando funções contínuas e lineares por partes

Nesta subseção, continuaremos seguindo a abordagem do livro do Johnson (1987) e vamos aplicar o MEF usando funções de uma variável contínuas e lineares por partes. O objetivo é dividir o intervalo contínuo em subintervalos menores e aproximar a solução da equação de Poisson com funções contínuas e lineares por partes em cada subintervalo, garantindo que a solução de elementos finitos (numérica) seja contínua em todo o domínio e atenda às condições de contorno do PVC de Poisson.

Dado o intervalo $[0, 1]$, considere a partição $0 = x_0 < x_1 < \cdots < x_M < x_{M+1} = 1$ desse intervalo. Seja V_h subespaço vetorial de dimensão finita de V, em que se $v \in V_h$ então:

1. $v \in C^0([0, 1])$;
2. $v(0) = v(1) = 0$;
3. v é linear em cada subintervalo $I_i = [x_i, x_{i+1}]$, $i \in \{0, 1, \ldots, M\}$ da partição de [0,1].

Além disso, definimos as funções de base (ou função chapéu) $\phi_j \in V_h$, como:

$$\phi_j(x_i) = \begin{cases} 1 & \text{se } i = j, \\ 0 & \text{se } i \neq j, \end{cases} \tag{2.8}$$

onde i e j variam de 1 a M.

Portanto, a função ϕ_j é uma função linear por partes e contínua que assume o valor 1 no nó x_j e 0 em outros nós x_i ($i \neq j$). Assim, cada função ϕ_j representa uma função linear em todos os subintervalos $I_i = [x_i, x_{i+1}]$ da partição de [0,1], garantindo a continuidade nos pontos onde os subintervalos se encontram.

A partição do intervalo $[0, 1]$ em subintervalos I_j e as funções ϕ_j podem ser visualizadas na Figura 2.

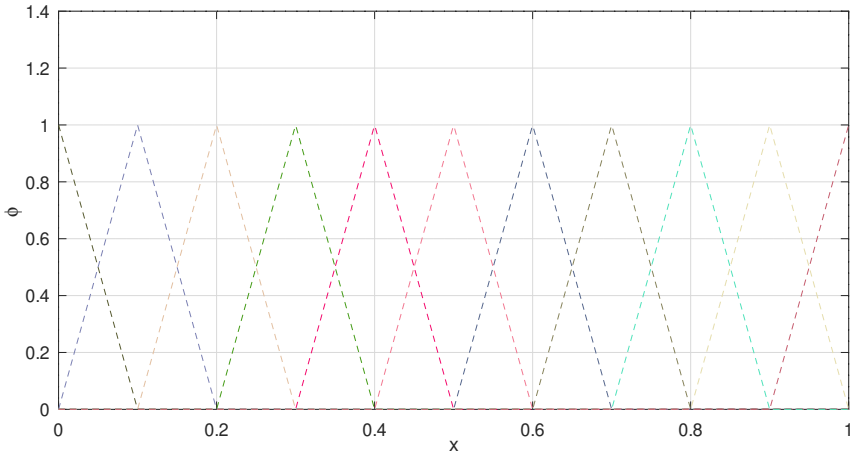


Figura 2 – Função chapéu.

Fonte: Autor

A Figura 2 ilustra como o intervalo é dividido em subintervalos e como as ϕ_j são construídas em cada subintervalo.

Para descrever uma função qualquer $v \in V_h$, podemos escolher os valores $\eta_j = v(x_j)$ nos nós x_j, onde $j = 0, \ldots, M$. Assim, cada função $v \in V_h$ pode ser escrita como uma combinação linear das funções de base ϕ_i, conforme a Equação 2.9.

$$v(x) = \sum_{i=1}^{M} \eta_i \phi_i(x). \tag{2.9}$$

Como as funções $\{\phi_j\}_1^M$ são linearmente independentes e geram V_h. Elas formam uma base para esse espaço, logo ϕ_i é um elemento da base de V_h.

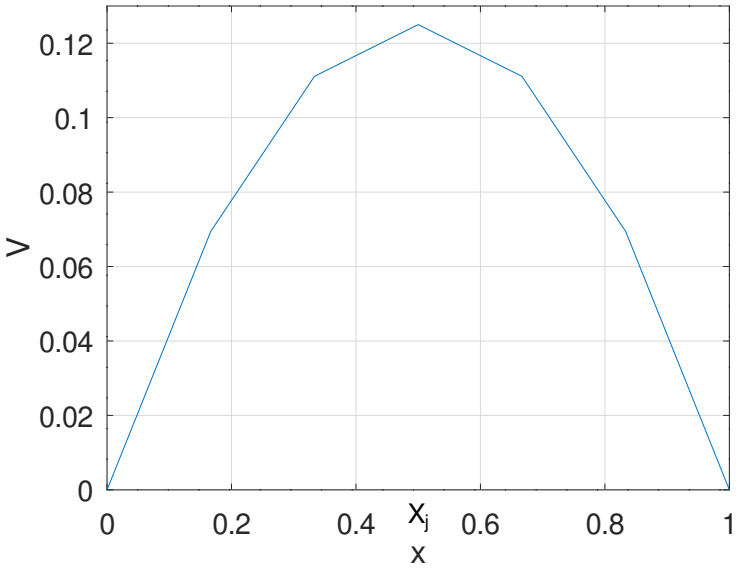


Figura 3 – Exemplo de uma função $v \in V_h$.

Fonte: Autor

Na Figura 3, apresentamos um exemplo de uma função que pertence ao espaço V_h, ilustrando o uso de polinômios de grau 1 da função chapéu.

2.5 Formulação Variacional do Problema

Com base no que Johnson (1987) sugere, agora podemos reformular o problema de Poisson unidimensional dado em (D), utilizando a abordagem do MEF.

Vamos utilizar o Método de Galerkin, onde a formulação variacional (V) do problema (D) é reformulado restringindo do espaço V para o espaço V_h, ou seja , precisamos encontrar

$$u_h \in V_h,$$

tal que

$$(u_h', v') = (f, v) \quad \forall\, v \in V_h. \tag{Vh}$$

Suponhamos que $u_h \in V_h$ satisfaz a formulação Vh acima, então a aproximação por elementos finitos é dada por

$$(u_h', \phi_j') = (f, \phi_j), \quad j = 1, \ldots, M, \tag{2.10}$$

onde ϕ_j foi definida na seção anterior. Usando a base $\{\phi_i\}$ pode-se reescrever u_h como combinação linear de ϕ_i:

$$u_h(x) = \sum_{i=1}^{M} u_i \phi_i(x) \quad \text{com} \quad u_i = u_h(x_i). \tag{2.11}$$

Logo, substituindo na Equação 2.10 e utilizando propriedade de produto interno, temos

$$\sum_{i=1}^{M} u_i(\phi_i', \phi_j') = (f, \phi_j), \quad j = 1, \ldots, M. \tag{2.12}$$

Assim a Equação 2.12 é um sistema de equações lineares com incognitas u_i e coeficientes (ϕ_i', ϕ_j'), e pode ser reescrita como um sistema linear na forma matricial como:

$$A\mathbf{u} = \mathbf{b}, \tag{2.13}$$

com M equações em M incógnitas $u_1, \ldots, u_M$ onde o vetor $u = (u_1, u_2, \ldots, u_n)$ é a solução. A matriz A é chamada de matriz de rigidez e b de vetor de carga. Colocando na forma matricial, temos

$$A = \begin{pmatrix} (\phi_1', \phi_1') & (\phi_1', \phi_2') & \cdots & (\phi_1', \phi_M') \\ (\phi_2', \phi_1') & (\phi_2', \phi_2') & \cdots & (\phi_2', \phi_M') \\ \vdots & \vdots & \ddots & \vdots \\ (\phi_M', \phi_1') & (\phi_M', \phi_2') & \cdots & (\phi_M', \phi_M') \end{pmatrix},$$

onde as entradas de A são da forma $a_{ij} = (\phi_i', \phi_j')$. E os vetores u e b, são da forma:

$$\mathbf{u} = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_M \end{pmatrix};$$

$$\mathbf{b} = \begin{pmatrix} (f, \phi_1) \\ (f, \phi_2) \\ \vdots \\ (f, \phi_M) \end{pmatrix}.$$

Agora, precisamos resolver o sistema linear para determinar o vetor $\mathbf{u}$, que contém os coeficientes u_i das funções base ϕ_i. Esses coeficientes são usados para construir a solução aproximada u_h, que é uma combinação linear das funções base ϕ_i, conforme definido em 2.11.

2.6 Calculando os elementos da matriz A

Seguindo a abordagem apresentada por Johnson (1987) , como foi visto o método Galerkin gerou um sistema de equações linear. Vamos analisar as propriedades da matriz encontrada A, mas antes precisamos entender o comportamento das funções de interpolação ϕ_j que estão sendo utilizadas.

Temos

$$\phi_j(x) = \begin{cases} \frac{x-x_{j-1}}{x_j-x_{j-1}}, & x \in I_{j-1}; \\ \frac{x-x_{j+1}}{x_j-x_{j+1}}, & x \in I_j; \\ 0, & \text{caso contrário.} \end{cases} \tag{2.14}$$

Logo derivando em [0,1],

$$\phi'_j(x) = \begin{cases} \frac{1}{x_j-x_{j-1}}, & x \in I_{j-1}; \\ \frac{1}{x_j-x_{j+1}}, & x \in I_j; \\ 0, & \text{caso contrário.} \end{cases} \tag{2.15}$$

Como ilustração do comportamento da função ϕ_j é apresentado o gráfico na Figura 4.

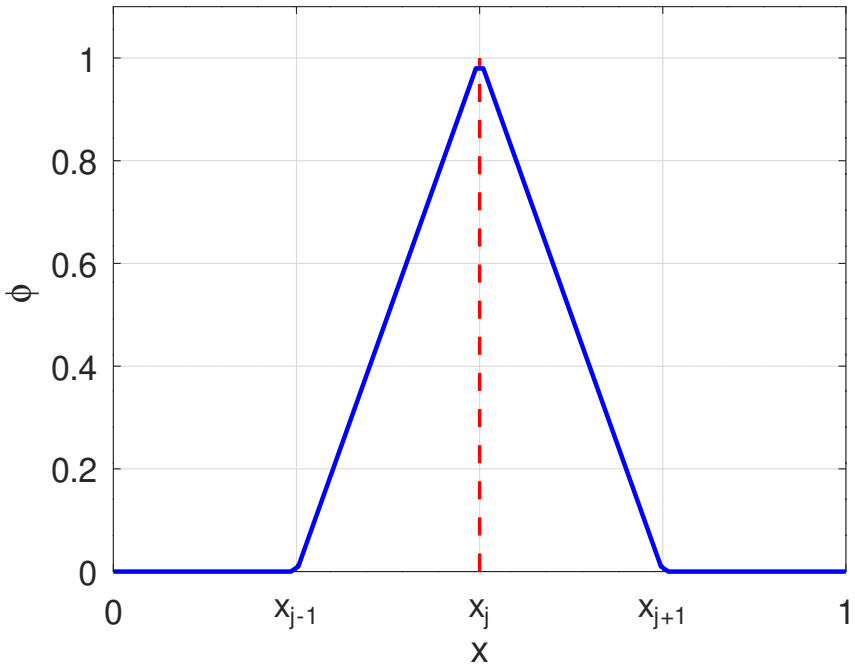


Figura 4 – Função base.

Fonte: Autor

Usando a fórmula 2.15 pode-se chegar em uma expressão mais detalhada para as entradas da matriz A. Com esse detalhamento, será possível ver que temos uma matriz tridiagonal e simétrica (definições que temos no capítulo 1). Para isso, calculando o valor do (ϕ'_j, ϕ'_j) no intervalo $I_{j-1} = [x_{j-1}, x_{j+1}]$, temos

$$(\phi'_j, \phi'_j) = \int_{x_{j-1}}^{x_j} \left(\phi'_j(x)\right)^2 \, dx + \int_{x_j}^{x_{j+1}} \left(\phi'_j(x)\right)^2 \, dx.$$

No intervalo $I_{j-1} = [x_{j-1}, x_j]$, sendo $h_j = x_j - x_{j-1}$, temos:

$$\phi'_j(x) = \frac{1}{h_j}.$$

Logo resulta

$$\int_{x_{j-1}}^{x_j} \left(\phi_j'(x)\right)^2 \, dx = \int_{x_{j-1}}^{x_j} \left(\frac{1}{h_j}\right)^2 \, dx = \frac{1}{h_j^2} \cdot (x_j - x_{j-1}) = \frac{1}{h_j^2} \cdot h_j = \frac{1}{h_j}.$$

E no intervalo $I_j = [x_j, x_{j+1}]$, sendo $h_{j+1} = x_{j+1} - x_j$, temos:

$$\phi_j'(x) = -\frac{1}{h_{j+1}}.$$

E com isso,

$$\int_{x_j}^{x_{j+1}} \left(\phi_j'(x)\right)^2 \, dx = \int_{x_j}^{x_{j+1}} \left(\frac{-1}{h_{j+1}}\right)^2 \, dx = \frac{1}{h_{j+1}^2} \cdot (x_{j+1} - x_j) = \frac{1}{h_{j+1}^2} \cdot h_{j+1} = \frac{1}{h_{j+1}}.$$

Portanto,

$$(\phi_j', \phi_j') = \frac{1}{h_j} + \frac{1}{h_{j+1}},$$

para $j = 1, \ldots, M$.

Para o produto interno (ϕ_j', ϕ_{j-1}'), temos

$$(\phi_j', \phi_{j-1}') = \int_{x_{j-1}}^{x_j} \phi_j'(x)\phi_{j-1}'(x) \, dx.$$

Usando que as derivadas de ϕ_j e ϕ_{j-1} são

$$\phi_j'(x) = \frac{1}{h_j}, \quad \phi_{j-1}'(x) = -\frac{1}{h_j},$$

e calculando a integral com elas, temos

$$(\phi_j', \phi_{j-1}') = -\int_{x_j}^{x_j} \frac{1}{h_j} \cdot \frac{1}{h_{j-1}} \, dx = -\frac{1}{h_j h_j} \cdot (x_j - x_j) = -\frac{1}{h_j h_j} \cdot h_j = -\frac{1}{h_j}.$$

Utilizando a propriedade de comutatividade do produto interno, temos

$$(\phi_j', \phi_{j-1}') = (\phi_{j-1}', \phi_j').$$

Por último tornando desnecessário mostrar (ϕ_i', ϕ_j'), pois $\phi_i'\phi_j' \equiv 0$ se $|i - j| > 1$.

$$(\phi_i', \phi_j') = 0 \text{ para } |i - j| > 1.$$

A matriz A então assume a forma tridiagonal quando $h_j = h =$ constante $\forall j$, se o espaçamento entre os nós forem constante igual h:

$$A = \frac{1}{h} \begin{bmatrix} 2 & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & \cdots & 0 \\ 0 & -1 & 2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 2 \end{bmatrix}.$$

onde os elementos são da forma

$$\begin{cases} a_{jj} = \frac{2}{h}, j = 1, ..., M; \\ a_{j,j-1} = -\frac{1}{h}, j = 2, ..., M; \\ a_{j,j+1} = -\frac{1}{h}, j = 1, ..., M-1; \\ 0, \text{caso contrário.} \end{cases}$$

Lembrando que se $v(x) = \sum_{j=1}^{M} \eta_j \phi_j(x)$ é arbitrária, outra propriedade importante da matriz é ser positiva definida 1.4.11, pois é simétrica e

$$v^\top A v = \sum_{i,j=1}^{M} \eta_i(\phi_i', \phi_j')\eta_j = \left(\sum_{i=1}^{M} \eta_i \phi_i', \sum_{j=1}^{M} \eta_j \phi_j'\right) = (v', v') > 0 \quad \forall v \in V_h \text{ e } v \neq 0. \quad (2.16)$$

Essa propriedade 2.16 é muito importante, pois de acordo com o Teorema 1.4.1 garante que a matriz é não singular, ou seja, inversível. Portanto, segue que o sistema tem solução única

$$\frac{1}{h}\begin{pmatrix} 2 & -1 & 0 & \cdots & 0 & 0 \\ -1 & 2 & -1 & \cdots & 0 & 0 \\ 0 & -1 & 2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 2 & -1 \\ 0 & 0 & 0 & \cdots & -1 & 2 \end{pmatrix} \cdot \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_{M-1} \\ u_M \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_{M-1} \\ b_M \end{pmatrix},$$

onde $h_j = \text{cte} = h \ \forall j$, e $b_i = (f, \phi_i)$ para $i = 1, \ldots, M$.

Dado uma f específica, resolvendo o sistema linear acima, temos as aproximações u_i e com isso, obtemos a aproximação contínua u_h para a solução u do PVC unidimensional de Poisson, onde foi utilizada a expressão

$$u_h(x) = \sum_{i=1}^{M} u_i \phi_i(x).$$

A solução u_h, que é formada pela combinação das funções ϕ_i, é uma aproximação da solução analítica u do problema de Poisson. Os valores u_i, que são os resultados nos pontos da malha $u_i = u_h(x_i)$ Equação 2.11, são encontrados resolvendo o sistema linear que vem da formulação variacional.

3 GNU Octave

3.1 Breve Introdução

O desenvolvimento do GNU Octave teve início por volta de 1988, com John W. Eaton sendo um dos principais desenvolvedores iniciais do projeto. Além dele, outros dois nomes importantes na criação do Octave foram James B. Rawlings, professor da Universidade de Wisconsin-Madison, e John G. Ekerdt, da Universidade do Texas, que também contribuíram significativamente para o desenvolvimento inicial do software UNESP (2023). Desde o começo, o objetivo era criar uma alternativa ao MATLAB, especialmente para quem precisava de uma ferramenta de cálculo numérico, mas sem os custos de licenciamento do programa.

O GNU Octave é uma linguagem de programação de alto nível e um ambiente de desenvolvimento voltado para a computação científica e matemática. Amplamente utilizado tanto em ambientes acadêmicos quanto industriais, o Octave foi projetado para resolver problemas numéricos em áreas como Álgebra Linear, Equações Diferenciais e Otimização, entre outras áreas. Sua interface baseada em linha de comando facilita cálculos complexos e permite a criação de scripts e funções personalizadas.

O GNU Octave é mantido por uma comunidade global de profissionais e desenvolvedores de diferentes países, o que faz com que o software esteja sempre sendo atualizado e melhorado. Ele é distribuído sob a licença GNU General Public License (GPL), o que significa que qualquer pessoa pode contribuir com o código-fonte, utilizar o programa de forma gratuita e até mesmo modificar o software conforme suas necessidades UNICAMP, 2023

3.2 Interface do GNU Octave

O GNU Octave possui uma interface intuitiva e de fácil utilização, projetada para proporcionar uma interação eficiente e sem complexidade UNESP (2023). A seguir, serão apresentados os principais componentes da interface do GNU Octave, como a janela de comando, o editor de texto e outras funcionalidades essenciais para o uso do programa.

3.2.1 Janela de Comandos

A janela de comandos é um dos componentes mais importantes do GNU Octave. Nela, é possível digitar comandos e visualizar os resultados imediatamente, o que a torna uma ferramenta prática para testar expressões ou comandos simples, sem a necessidade de

criar um script completo. Essa resposta imediata facilita a realização de cálculos rápidos e a resolução de problemas durante o desenvolvimento das funções. A Figura 5 ilustra a interface da janela de comandos:

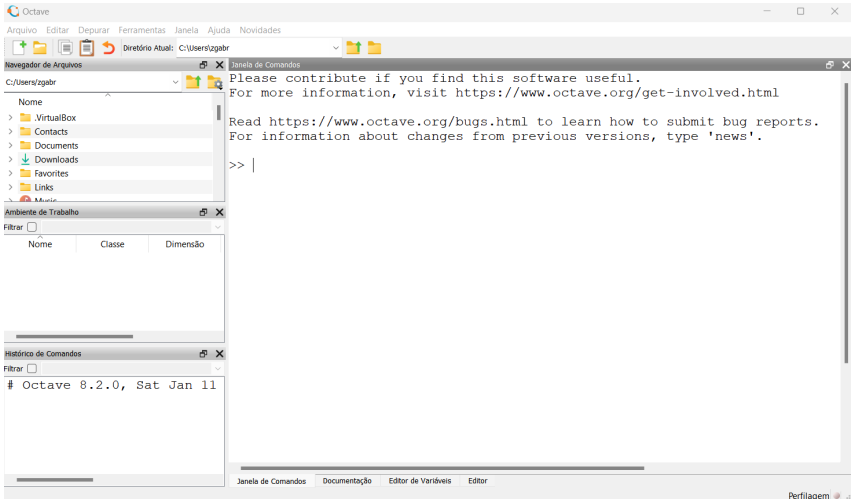


Figura 5 – Interface do GNU Octave

Fonte: Autor

3.2.2 Editor de Texto

O editor de texto do Octave é onde podemos escrever e organizar os scripts. Ele permite que criemos arquivos com a extensão .m, onde podemos colocar todo o código de forma detalhada. Podemos rodar o script direto do editor, o que torna o processo de testar e ajustar o código muito mais prático. Na Figura 6, é mostrada a interface do editor de texto:

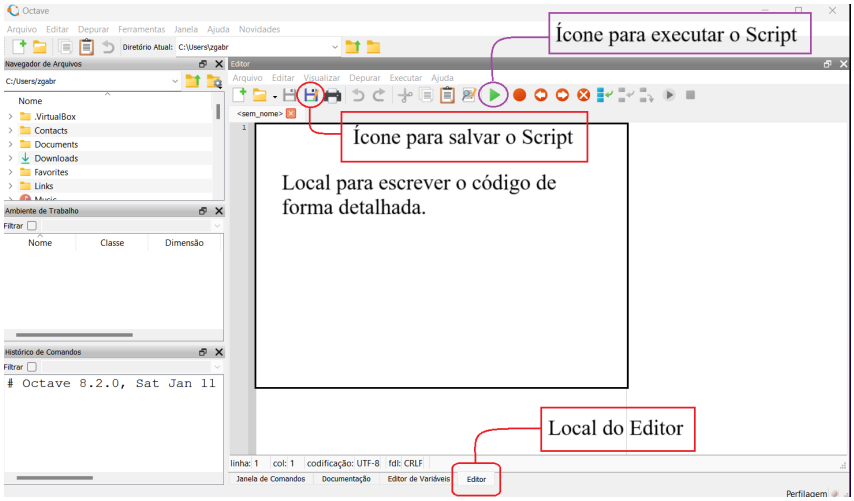


Figura 6 – Editor de scripts do GNU Octave

Fonte: Autor

3.2.3 Janela de Gráfico

O Octave tem como auxílio comandos para gerar gráficos, o que facilita a visualização dos resultados de nossos cálculos e simulações. Podemos criar gráficos 2D e 3D de forma simples, utilizando comandos como plot para representar dados de maneira clara. Isso nos ajuda muito quando queremos analisar o comportamento de uma função ou o resultado de uma simulação. Além disso, o Octave permite personalizar os gráficos, ajustando títulos, rótulos dos eixos, cores e estilos de linha, o que torna a apresentação dos dados visualmente melhor. Na Figura 7, é mostrada a interface para a janela de gráfico no Octave:

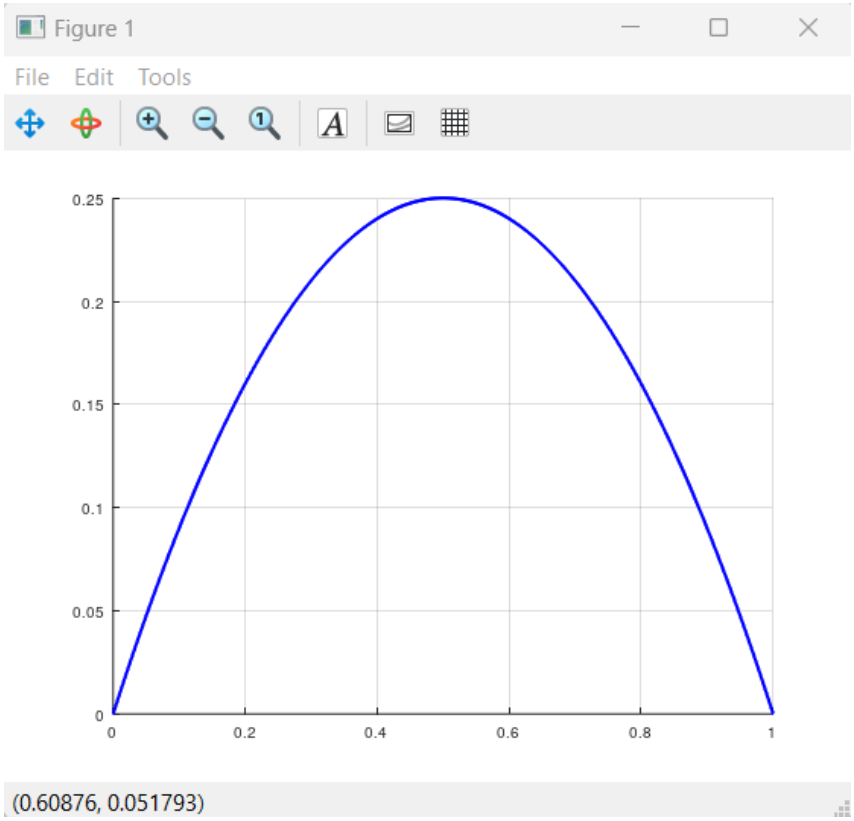


Figura 7 – Janela da figura

Fonte: Autor

O GNU Octave é um programa prático e de fácil compreensão para aqueles que estão iniciando no campo dos cálculos numéricos e simulações. Ele utiliza uma linguagem de programação simples, facilitando seu uso e entendimento. Suas características tornam o Octave uma escolha ideal para a implementação de métodos numéricos, oferecendo uma plataforma acessível e eficiente para manipulação de matrizes, resolução de sistemas lineares e visualização de resultados. Além disso, sua capacidade de criar gráficos e visualizar soluções facilita na análise mais clara dos resultados numéricos obtidos, sendo bastante útil na aplicação do MEF.

4 Implementação do Método dos Elementos Finitos no Octave

4.1 Breve Introdução

Neste capítulo, vamos ver como implementar o MEF para resolver a equação de Poisson utilizando o programa Octave, onde criaremos um código para resolver o problema de Poisson em um domínio unidimensional.

4.2 Desenvolvimento do Código

Nesta seção, mostraremos como o código foi criado para aplicar o MEF à equação de Poisson em um domínio unidimensional.

O objetivo é obter e resolver o sistema linear resultante da formulação variacional da equação de Poisson, representado pela equação (2.13), mostrando que, quanto mais refinada a malha, mais a solução numérica se aproxima da solução analítica.

Para testar a implementação do código, utilizamos os dados de um problema teste e os parâmetros necessários, apresentados na Tabela 2.

Tabela 2 – Dados e parâmetros do problema teste.

Parâmetro	**Valor**
Domínio L	$[0, 1]$
Função fonte $f(x)$	2
Condições de contorno $u(0) = u(1)$	0
Número de Nós	11
Solução Analítica $u(x)$	$x(1 - x)$

A seguir, apresentamos na Figura **??** o gráfico da solução analítica, que será usada para comparar com a solução obtida através do MEF.

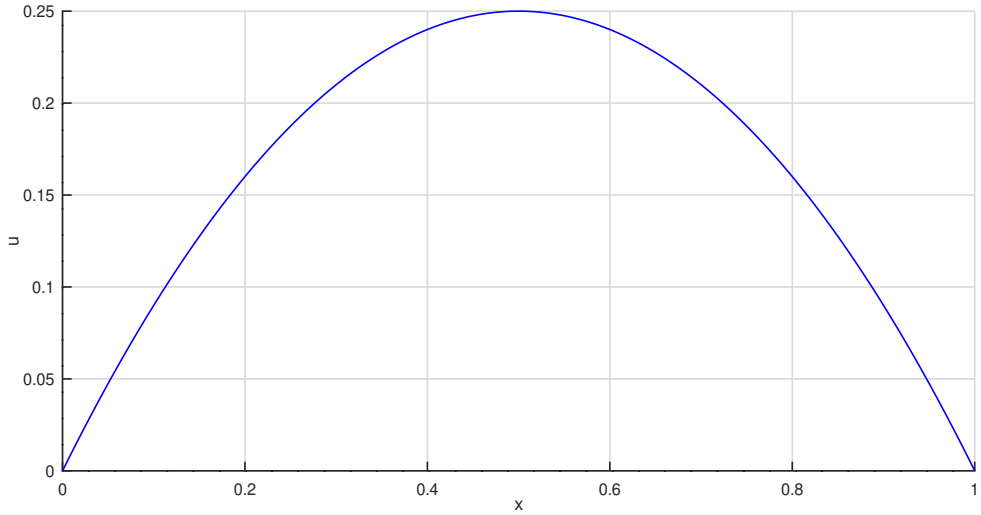


Figura 8 – Solução analítica (linha azul).

Fonte: Autor.

4.2.1 Etapas do Desenvolvimento

Dados do Problema:

O código começa criando um script inicial e colocando os dados do problema necessários para a aplicação do MEF. Serão apresentados os parâmetros do problema que devem ser informados pelo usuário na Tabela 3 abaixo:

Tabela 3 – Tabela com os dados iniciais.

Parametro	**Descrição**
L	Tamanho do intervalo do domínio
N	Número total de nós (pontos) no intervalo
n	Número de intervalos no domínio $(N-1)$
M	Número de nós internos $(N-2)$
h	Tamanho do intervalo entre os nós (L/n)
x	Vetor contendo as posições dos nós ao longo do intervalo

Temos o parâmetro L cujo valor utilizado é 1, representando o tamanho do intervalo do domínio do problema. O número de nós, N, foi estabelecido como 11 no problema teste, o que implica que o intervalo será subdividido em 11 pontos, incluindo as extremidades. Com isso, o número de intervalos, n, é calculado como N-1, ou seja, 10 intervalos entre os nós.

Em seguida, é calculado M, que corresponde ao número de nós internos, este valor é dado por N-2, ou seja, 9 nós internos. O valor de h, o tamanho do intervalo entre os nós, é calculado dividindo o comprimento total do intervalo L pelo número de intervalos n, resultando em h = 1/10 significando a resolução da malha (tamanho do elemento/ tamanho do intervalo).

Finalmente, o vetor $\mathbf{x}$ é gerado utilizando a função `linspace(0, L, N)`, que cria um vetor de N pontos igualmente espaçados entre 0 e L , representando as posições dos nós ao longo do intervalo. O vetor $\mathbf{x}$ fica como $\mathbf{x} = [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]$ e será utilizado para representar a localização dos pontos do domínio discretizado ao longo do intervalo. Na Figura 9, mostramos como ficou essa parte no script inicial:

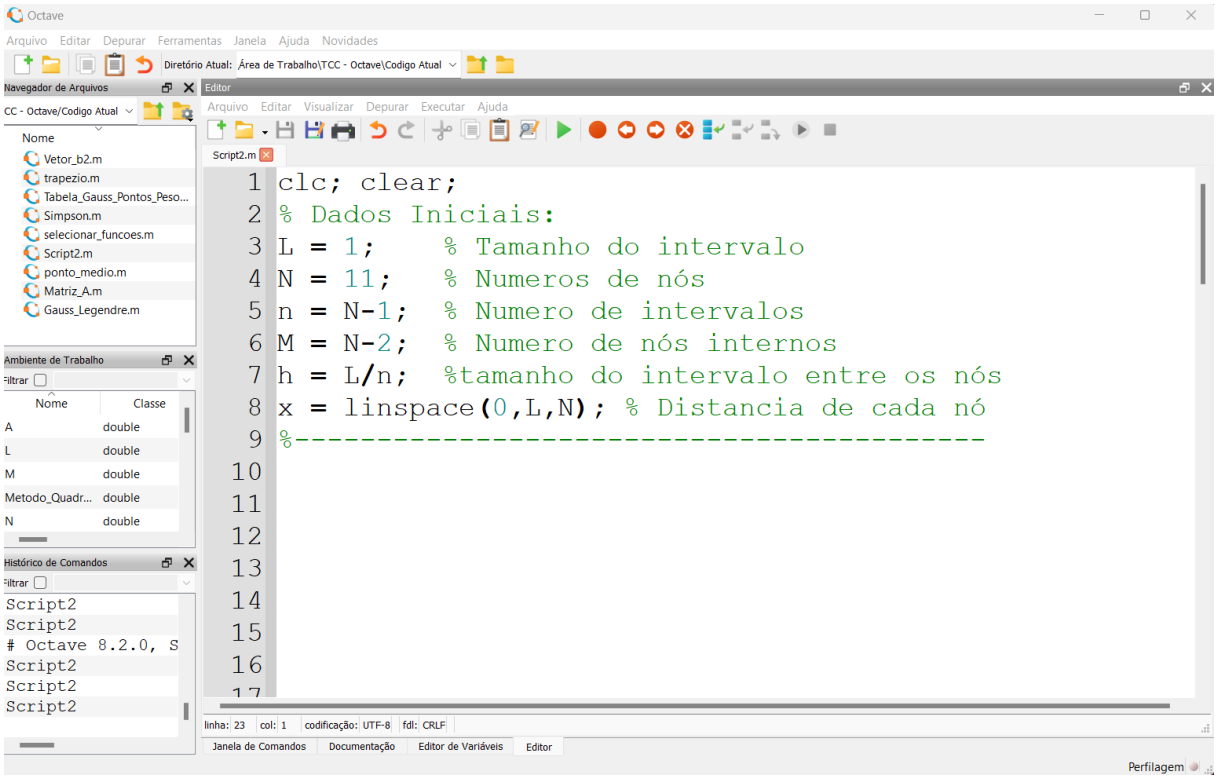


Figura 9 – Dados iniciais.

Fonte: Autor

Construção da Matriz de Rigidez A:

No código criamos uma função chamada `Matriz_A`, responsável por calcular a matriz de rigidez A no contexto do MEF para o PVC de Poisson.

Primeiramente, a função recebe dois parâmetros de entrada: o número de nós internos M (que exclui as extremidades do domínio) e o tamanho do intervalo entre os nós h. Com esses parâmetros, a função inicia a criação da matriz A.

A matriz A é inicializada com `zeros` utilizando a função `zeros(M, M)`, criando uma matriz quadrada de dimensão $M \times M$. Esse passo prepara a estrutura para o preenchimento com os valores que representam o produto interno entre as derivadas de $\phi(x)$ dos elementos no problema de MEF conforme apresentado no capitulo (2).

O preenchimento da matriz A é realizado dentro de um loop `for`, que percorre os índices de $i = 1$ até M. Sabemos que a matriz A é tridiagonal, o que significa que ela possui apenas três diagonais relevantes: a diagonal principal, a diagonal inferior e

a diagonal superior. Portanto, durante o loop, são atribuídos valores apenas para essas três diagonais, simplificando o processo de cálculo. Para cada valor de i, os seguintes elementos são atribuídos:

1. **Entrada $a_{i,i-1}$ da diagonal inferior da matriz** A: Se $i > 1$, o valor $-\frac{1}{h}$ é atribuído à entrada $a_{i,i-1}$, linha i e coluna $i-1$ na matriz A, representando a interação entre a função base ϕ_i e a função ϕ_{i-1}.

2. **Entrada $a_{i,i}$ da diagonal principal da matriz** A: O valor $\frac{2}{h}$ é atribuído à entrada $a_{i,i}$,da i-ésima linha e i-ésima coluna de A, representando a interação entre a função base ϕ_i com ela mesmo.

3. **Entrada $a_{i,i+1}$ da diagonal superior da matriz** A: Se $i < M$, o valor $-\frac{1}{h}$ é atribuído à entrada $a_{i,i+1}$, linha i e coluna $i+1$ na matriz A, representando a interação entre a função base ϕ_i e a função ϕ_{i+1}.

Esses valores preenchem a matriz de rigidez A, e a função `Matriz_A` retorna a matriz A.

```
function A = Matriz_A(M, h)
    % Fun   o para calcular a matriz A
    % M: n mero de n s internos
    % h: tamanho do intervalo entre os n s

    A = zeros(M, M); % Inicializar matriz A com zeros

    for i = 1:M
        if i > 1
            A(i, i-1) = -1/h; % Elemento inferior
        end
        A(i, i) = 2/h; % Elemento da diagonal principal
        if i < M
            A(i, i+1) = -1/h; % Elemento superior
        end
    end
end
```

Listing 4.1 – Matriz de Rigidez A

Montagem do Vetor de Carga b:

Temos que criar um script utilizando o comando `Function` sendo uma função `Vetor_b` que é responsável por calcular o vetor **b**, que corresponde ao lado direito do sistema de equações (termo independente) gerado pela formulação variacional no MEF conforme apresentado no capítulo 2. O cálculo do vetor **b** baseia-se no produto interno entre a função f, que depende do problema a ser resolvido, e a função base ϕ_j. Para calcular esse produto interno, se faz necessário utilizar um método numérico, que fornece

uma aproximação do valor da integral no produto interno. A obtenção do vetor **b** é feita de acordo com o método de quadratura que deve ser escolhido pelo usuário entre as opções implementadas.

O comando `Function` inicia criando o vetor **b** com zeros para posteriormente acrescentar os valores. Em seguida, define as funções de interpolação lineares por partes (ou funções base ϕ_i de funções "chapéu"), que são usadas como base para escrever a solução aproximada $u(x)$ dentro de cada elemento. As funções base/chapéu são obtidas usando interpolação lagrangeana e essas funções de interpolação ϕ_i são definidas para os elementos nos lados esquerdo e direito do i-ésimo nó, permitindo que as integrais sejam calculadas de forma adequada. Nos elementos restantes, a função base ϕ_i é nula.

No Código 4.2, é apresentado o código completo para o cálculo do vetor **b**.

```
function b = Vetor_b2(x, f, h, M, Metodo_Quadratura,n_Gauss)
    b = zeros(M, 1); % Inicializar o vetor b
    switch Metodo_Quadratura
        case 1 % M todo do Ponto Medio
            for i = 1:M
                i_esq = i; i_meio = i + 1; i_dir = i + 2;

                % Lado esquerdo:
                phi_esq = @(X) (X - x(i_esq)) / h;
                Lado_Esquerdo = ponto_medio(@(X) f(X)*phi_esq(X),x(i_esq),x(
                    i_meio));
                b(i) = b(i) + Lado_Esquerdo;

                % Lado direito:
                phi_dir = @(X) (X - x(i_dir)) / (-h);
                Lado_Direito = ponto_medio(@(X) f(X)*phi_dir(X),x(i_meio),x(
                    i_dir));
                b(i) = b(i) + Lado_Direito;
             end

        case 2 % M todo do Trapezio
            for i = 1:M
               i_esq = i; i_meio = i + 1; i_dir = i + 2;

                % Lado esquerdo:
                phi_esq = @(X) (X - x(i_esq)) / h;
                Lado_Esquerdo = trapezio(@(X) f(X)*phi_esq(X),x(i_esq),x(i_meio)
                    );
                b(i) = b(i) + Lado_Esquerdo;

                % Lado direito:
                phi_dir = @(X) (X - x(i_dir)) / (-h);
                Lado_Direito = trapezio(@(X) f(X)*phi_dir(X),x(i_meio),x(i_dir))
                    ;
                b(i) = b(i) + Lado_Direito;
            end

            case 3 % M todo de Simpson
```

```
            for i = 1:M
               i_esq = i; i_meio = i + 1; i_dir = i + 2;

                % Lado esquerdo:
                phi_esq = @(X) (X - x(i_esq)) / h;
                Lado_Esquerdo = Simpson(@(X) f(X)*phi_esq(X),x(i_esq),x(i_meio))
                    ;
                b(i) = b(i) + Lado_Esquerdo;

                % Lado direito:
                phi_dir = @(X) (X - x(i_dir)) / (-h);
                Lado_Direito = Simpson(@(X) f(X)*phi_dir(X),x(i_meio),x(i_dir));
                b(i) = b(i) + Lado_Direito;
          end

          case 4 % M todo de Gauss_Legendre
            for i = 1:M
               i_esq = i; i_meio = i + 1; i_dir = i + 2;

                % Lado esquerdo:
                phi_esq = @(X) (X - x(i_esq)) / h;
                Lado_Esquerdo = Gauss_Legendre(@(X) f(X).*phi_esq(X),x(i_esq),x(
                    i_meio),n_Gauss);
                b(i) = b(i) + Lado_Esquerdo;

                % Lado direito:
                phi_dir = @(X) (X - x(i_dir)) / (-h);
                Lado_Direito = Gauss_Legendre(@(X) f(X).*phi_dir(X),x(i_meio),x(
                    i_dir),n_Gauss);
                b(i) = b(i) + Lado_Direito;
            end
        otherwise
            error('Op   o inv lida. Escolha um valor de 1 a 4.');
    end
end
```

Listing 4.2 – Vetor de Carga b

Em seguida, temos como escolher qual método de quadratura queremos utilizar para a integração numérica no produto interno, e a função executa um dos seguintes casos de quadratura:

- **Método 1 - Ponto Médio:**

Neste script, criamos o código para o método do ponto médio. O script inicia com o comando function, permitindo que ele seja chamado a partir de outros scripts ou funções. Em seguida, especificamos de forma genérica a fórmula do ponto médio. Dessa maneira, obtemos o código para a quadratura numérica utilizando o método do ponto médio apresentado no Código 4.3 abaixo.

```
function pm = ponto_medio(g, a, b)

  x_medio = (a + b)/2 ;

  pm = (b-a)*g(x_medio);

  end
```

Listing 4.3 – Método do Ponto Médio

- **Método 2 Trapézio:**

Neste script, implementamos o código para o método do trapézio. O script começa com o comando function, permitindo que ele seja chamado a partir de outros scripts ou funções. Em seguida, descrevemos de forma genérica a fórmula do trapézio. Com isso, obtemos o código para a quadratura numérica através do método do trapézio apresentado no Código 4.4 abaixo.

```
function TP = trapezio(g, a, b)

  rd = (b - a)/2 ;

  TP = rd*[g(a) + g(b)];

  end
```

Listing 4.4 – Regra do Trapézio

- **Método 4 - Gauss-Legendre:**

No Código 4.5, se inicia com o comando function, garantindo que ele possa ser chamado de outros scripts ou funções. A função `Tabela_Gauss_Pontos_Pesos` irá nos fornecer os pontos e pesos necessários para a quadratura conforme o número de pontos escolhidos. A transformação do intervalo de integração é feita através de uma variável auxiliar, ajustando os pontos de quadratura do intervalo padrão $[-1, 1]$ para o intervalo $[a, b]$. A fórmula de Gauss-Legendre é então aplicada, somando os termos ajustados para calcular a integral aproximada.

```
function I = Gauss_Legendre(f, a, b, n_Gauss)
    % f: fun  o a ser integrada (handle @)
    % a, b: limites de integra  o
    % n_Gauss: n mero de pontos de quadratura

    % Passo 1: Obter os pontos de quadratura (xi) e os pesos (ci) para o n mero
        de pontos n
    [xi, ci] = Tabela_Gauss_Pontos_Pesos(n_Gauss);

    % Passo 2: Ajustar os pontos de quadratura para o intervalo [a, b]
    % Transforma  o do intervalo de quadratura [-1, 1] para [a, b]
    x_mudanca_variavel = @(t) a * (1 - t) / 2 + b * (1 + t) / 2;

    x_interv = (b-a)/2;

    % Aplicar a f rmula da quadratura de Gauss-Legendre
    I = 0;

    for j = 1:n_Gauss
        % Usar xi(j) em vez de t(j) para aplicar a transforma  o
        I = I + ci(j) * f(x_mudanca_variavel(xi(j)));
    end

    % Multiplicar pelo fator de escala (b - a) / 2
    I = I * x_interv;
end
```

Listing 4.5 – Quadratura de Gauss de Legendre

No Código 4.6, a função `Tabela_Gauss_Pontos_Pesos`, retorna os pontos de quadratura x_i e os pesos c_i para diferentes escolhas do número de pontos n_{Gauss}, que podem ser 2, 3 ou 4.

```
function [xi, ci] = Tabela_Gauss_Pontos_Pesos(n_Gauss)
   % Retorna n s (xi) e pesos (ci) para quadratura de Gauss
    switch n_Gauss
        case 2
            xi = [-0.5773502692, 0.5773502692]; % Pontos
            ci = [1, 1];                        % Pesos
        case 3
            xi = [-0.7745966692, 0, 0.7745966692];
            ci = [0.5555555556, 0.8888888889, 0.5555555556];
        case 4
            xi = [-0.8611363116, -0.3399810436, 0.3399810436, 0.8611363116];
            ci = [0.3478548451, 0.6521451549, 0.6521451549, 0.3478548451];
        otherwise
            error('N mero de pontos n  o implementado. Escolha n = 2, 3 ou 4.')
                ;
    end
end
```

Listing 4.6 – Tabela de Gauss dos Pontos e Pesos

Essa tabela é essencial para a aplicação da quadratura de Gauss-Legendre, pois fornece os valores necessários em função do número de pontos na quadratura para calcular a integral de uma função no intervalo $[-1, 1]$, que depois será transformado para qualquer intervalo desejado, como $[a, b]$. usando o teorema da regra de substituição do Cálculo 1.2. Os dados dessa tabela foram obtidos pelo livro de Análise Numérica Burden e Faires (2008).

Com isso, os métodos de quadratura do ponto médio, trapézio e Simpson foram fundamentais para a compreensão e aplicação da quadratura de Gauss-Legendre no Octave. Esses métodos básicos facilitaram o entendimento do funcionamento da quadratura de Gauss-Legendre, que foi essencial para resolver o problema no contexto do MEF. Para a execução do código, foram necessários os parâmetros específicos, como a quadratura de Gauss-Legendre e o número de 4 pontos de Gauss que foi utilizado. Com as informações necessárias, foi possível concluir a execução do script `Vetor_b` para calcular o vetor de carga $\mathbf{b}$ no sistema de equações lineares 2.13.

Resolução do Sistema Linear:

Após a aplicação no octave, precisamos resolver o sistema linear $A\mathbf{u} = \mathbf{b}$ para determinar os valores da solução u. A matriz A foi construída a partir do produto interno entre as derivadas de ϕ_i , enquanto o vetor $\mathbf{b}$ foi calculado através do produto interno entre a função fonte e a função ϕ_i. Com isso conseguimos encontrar os valores de aproximação u_i para a solução $\mathbf{u}$ do problema em cada nó x_i e, com isso, a solução numérica u_i.

A seguir, a Figura 10 apresenta os valores da matriz A, do vetor $\mathbf{b}$, e da solução $\mathbf{u}$ para os parâmetros adotados no teste de implementação.

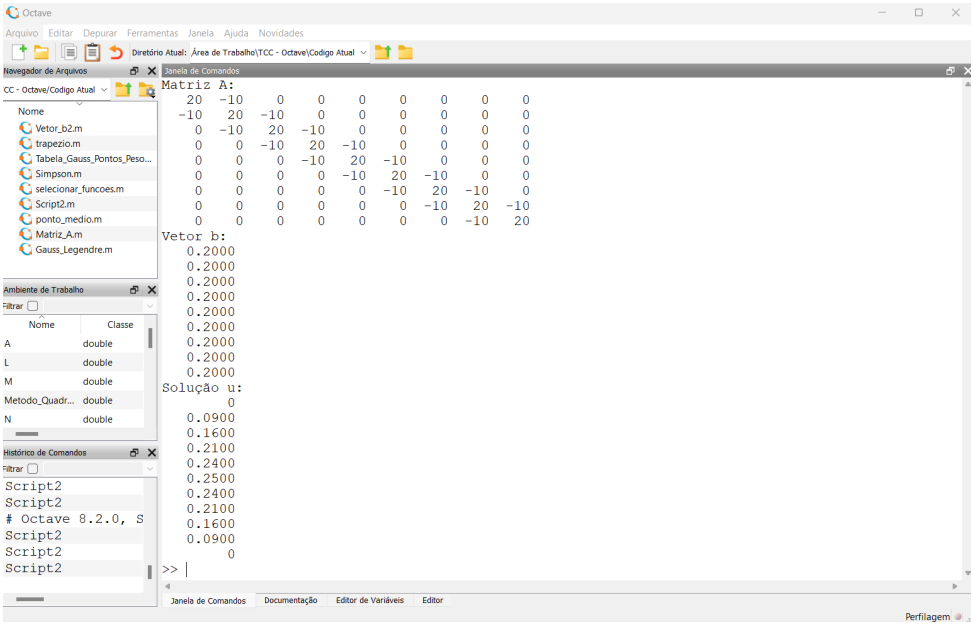


Figura 10 – Resultado da Matriz A - Vetores $\mathbf{b}$ e $\mathbf{u}$.

Fonte: Autor

Geração de Gráficos:

Após executar o código e obter os resultados, um gráfico é gerado mostrando como a solução numérica se aproxima da solução analítica. No caso, utilizamos 11 nós, como mostrado na Figura 11 abaixo.

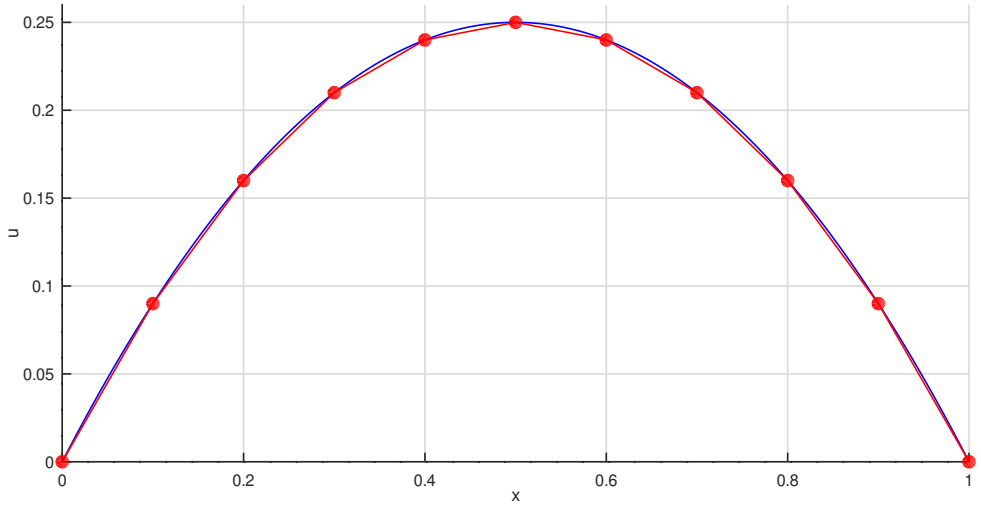
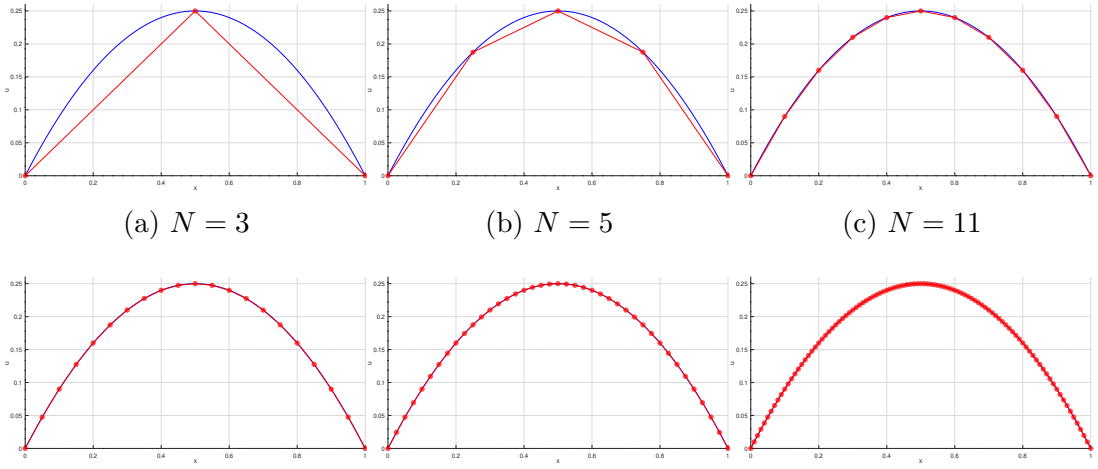


Figura 11 – Solução analítica (linha azul), solução numérica (linha vermelha) e valor da solução numérica nos nós (pontos vermelhos).

Fonte: Autor

Refinando a malha com mais nós

Agora, vamos observar como a solução numérica vai se aproximando da solução analítica à medida que aumentamos o número de nós. A seguir, apresentamos na Figura 12, seis figuras com diferentes quantidades de nós, mostrando como a solução numérica se aproxima da solução analítica conforme refinamos a malha.



(a) $N = 3$ (b) $N = 5$ (c) $N = 11$

(d) $N = 21$ (e) $N = 41$ (f) $N = 101$

Figura 12 – Aumentando a resolução da malha, solução analítica (linha azul), solução numérica (linha vermelha) e valor da solução numérica nos nós (pontos vermelhos).

Fonte: Autor

Conseguimos perceber pelos gráficos que à medida que o número de nós N aumenta, a solução numérica aproxima-se cada vez mais da solução analítica. Com valores baixos de N, a solução é mais grosseira, mas com valores mais altos (como $N = 41$ ou $N = 101$), a curva fica bem suave e visualmente idêntica à solução analítica. Isso mostra como a malha com mais nós melhora a qualidade da solução numérica.

Erro máximo absoluto

Tabela 4 – Erro máximo absoluto para diferentes números de nós

Número de Nó (N)	**Erro Máximo Absoluto**
3	0.062500
5	0.015625
11	0.002500
21	0.000625
41	0.000156
101	0.000025

A Tabela 4 apresenta os valores do erro máximo absoluto para diferentes números de nós utilizados na discretização. Como esperado, observa-se que, à medida que o número de nós aumenta, o erro tende a diminuir.

5 Exemplos

5.1 Breve Introdução

Neste capítulo, serão resolvidos alguns exemplos da equação de Poisson unidimensional utilizando o MEF. A abordagem será feita a partir da formulação variacional, resultando no sistema linear $A\mathbf{u} = \mathbf{b}$, que será resolvido para obtenção da solução $\mathbf{u}$ que permite construir a solução numérica $u_h(x) = \sum_{i=1}^{M} u_i\phi_i(x)$. O código desenvolvido no Octave será utilizado para executar os cálculos, sendo fornecidos diferentes conjuntos de dados (função f) para cada exemplo. Em cada caso, serão gerados gráficos para diferentes números de nós na malha. Os exemplos apresentados têm como objetivo mostrar como a solução numérica se aproxima para a solução analítica à medida que a quantidade de nós da malha é aumentada em diferentes situações representadas pela função f.

5.2 Exemplo 1

Neste exemplo, consideramos a equação de Poisson unidimensional. O problema de Valor de Contorno (PVC) a ser resolvido é dado por:

$$\begin{cases} -u''(x) = f(x), & \text{para } 0 < x < 1, \\ u(0) = u(1) = 0, \end{cases} \tag{5.1}$$

No capítulo anterior, foi utilizada uma função fonte constante 2, o que tornou o problema mais simples de resolver. Neste exemplo, a função fonte $f(x) = 6x$, um polinômio de grau 1, resulta em uma solução analítica simples, mas mais complexa que o exemplo anterior o que torna o problema interessante adequado para a aplicação do MEF.

Dados Iniciais:

- **Domínio**: $0 \leq x \leq 1$ com comprimento $L = 1$.
- **Função Fonte**: $f(x) = 6x$.
- **Solução Analítica**: $u(x) = x(1 - x^2)$.
- **Método de Quadratura**: Utilize a quadratura de Gauss com 4 pontos.
- **Aumento do Número de Nós na Malha**: Resolva o problema com 5, 7 e 15 nós.

Solução Analítica:

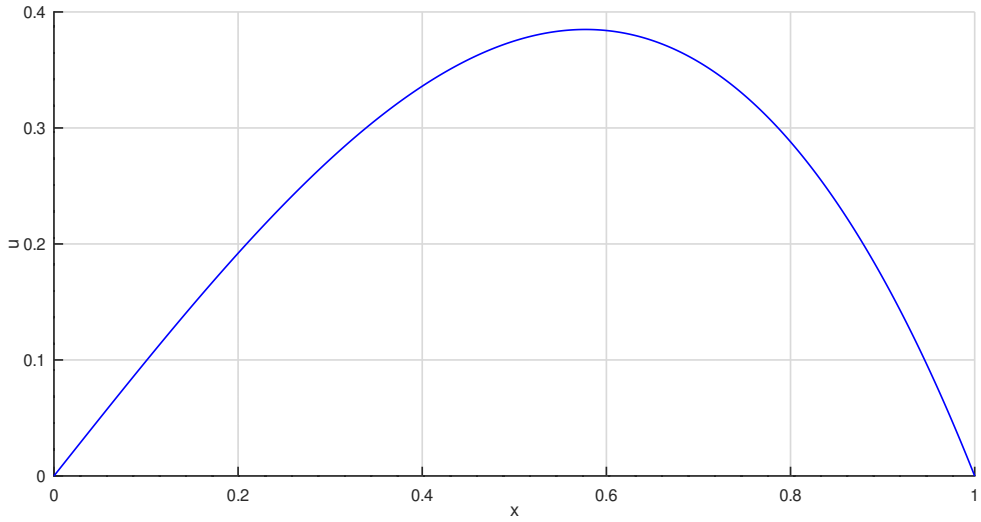


Figura 13 – Exemplo 1 - solução analítica (linha azul).

Fonte: Autor

Um ponto interessante dessa solução analítica, que a diferencia do problema teste do capítulo anterior 4, é que ela não é simétrica em relação ao centro do domínio.

5.2.1 Resultados

Solução Numérica com 5 Nós:

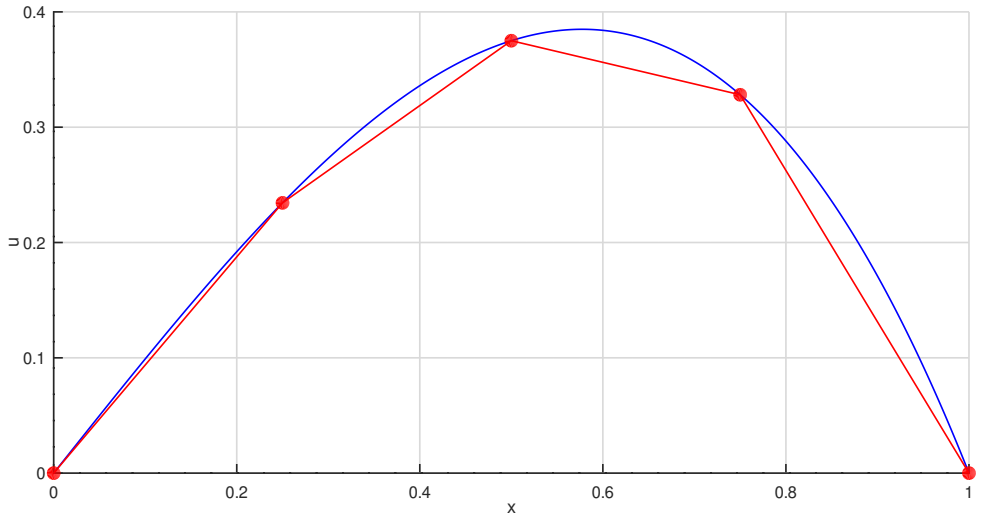


Figura 14 – Exemplo 1 - 5 nós, solução analítica (linha azul), solução numérica (linha vermelha) e valor da solução numérica nos nós (pontos vermelhos).

Fonte: Autor

Nesta Figura 14, podemos ver a solução numérica $u_h(x)$ usando 5 nós, comparada com a solução analítica $u(x) = x(1 - x^2)$. Como temos poucos nós, a aproximação

fica um pouco imprecisa, com diferenças bem visíveis em relação à curva analítica. Isso ilustra como uma malha com menos nós pode resultar em uma solução numérica de menor qualidade.

Em seguida, a Figura 15 exibe os valores obtidos da matriz A, do vetor $\mathbf{b}$ e da solução $\mathbf{u}$, referentes ao sistema linear $A\mathbf{u} = \mathbf{b}$, calculados com base no número de nós utilizado na Figura 14.

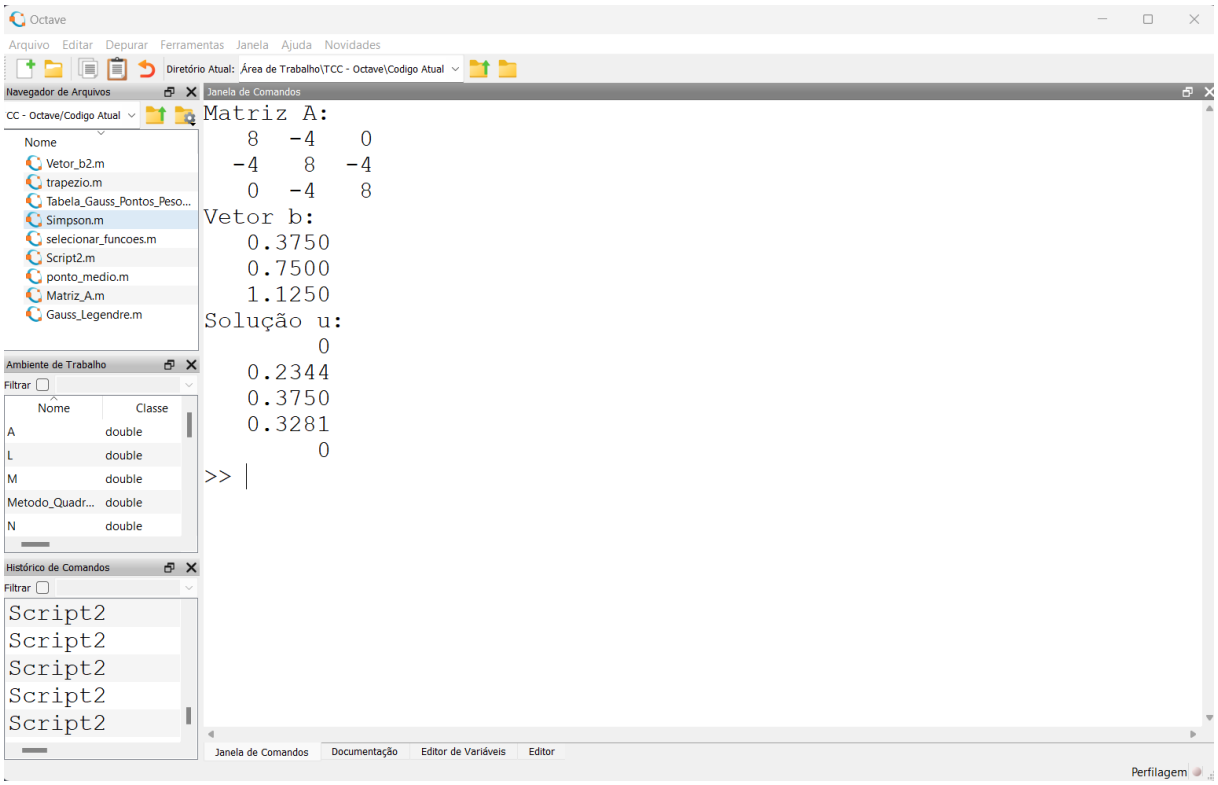


Figura 15 – Exemplo 1 - 5 nós - resultado do sistema linear.

Fonte: Autor

Na Figura 15, mostramos o resultado do sistema linear, onde estão os valores que calculamos para a Matriz A, o vetor $\mathbf{b}$ e o vetor $\mathbf{u}$ é o vetor solução que estamos buscando. A partir desses valores, conseguimos construir a solução aproximada 14 para o PVC proposto.

Solução Numérica com 7 Nós:

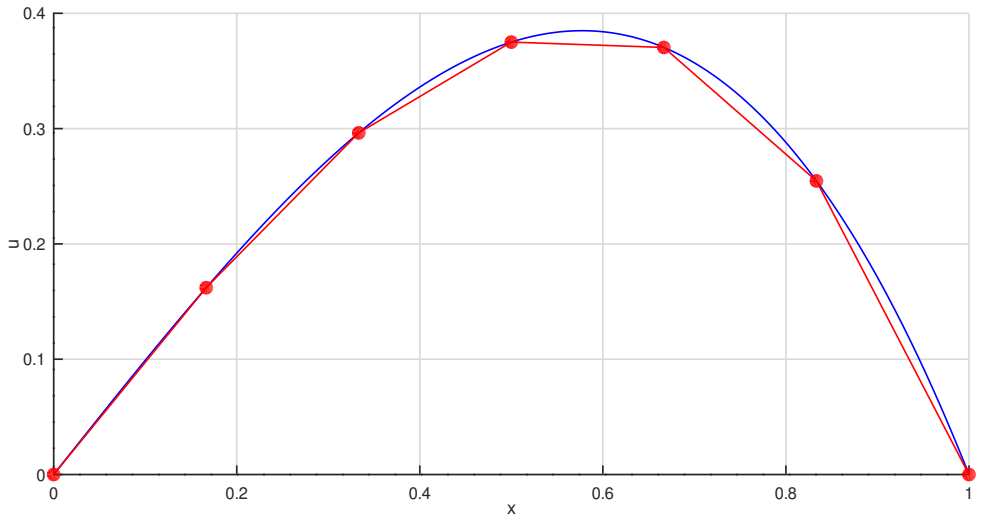


Figura 16 – Exemplo 1 - 7 nós, solução analítica (linha azul), solução numérica (linha vermelha) e valor da solução numérica nos nós (pontos vermelhos).

Fonte: Autor

Como podemos ver na Figura 16 acima, a solução numérica $u_h(x)$ com 7 nós já aproxima melhor a solução analítica. A curva ficou mais suave, e a diferença entre as duas soluções diminuiu. É possível para ver como, aumentar o número de nós, melhora bastante a qualidade da solução numérica.

Solução Numérica com 15 Nós:

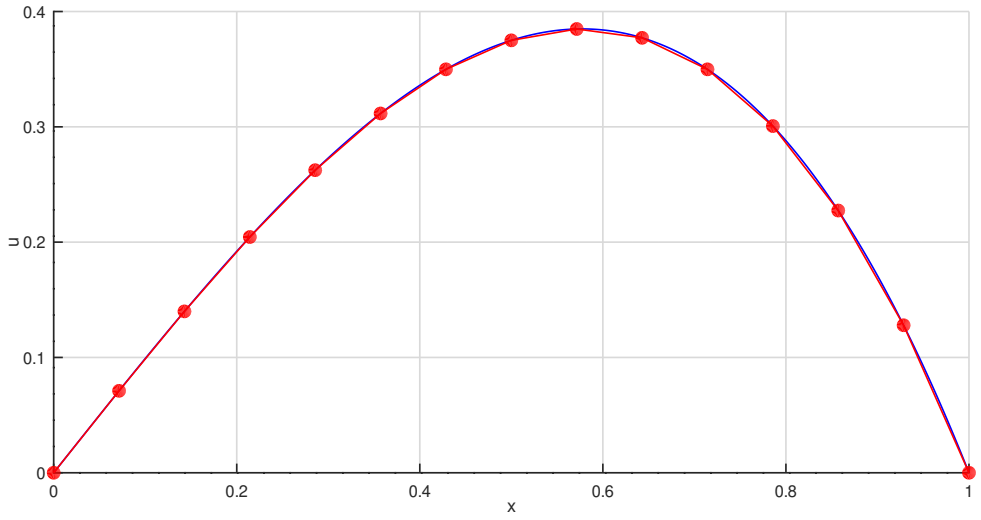


Figura 17 – Exemplo 1 - 15 nós, solução analítica (linha azul), solução numérica (linha vermelha) e valor da solução numérica nos nós (pontos vermelhos).

Fonte: Autor

Nessa última Figura 17, com 15 nós, a solução numérica $u_h(x)$ quase não se diferencia visualmente da solução analítica $u(x) = x(1 - x^2)$. Com mais pontos na malha, a

qualidade da aproximação ficou muito melhor, quase sobrepondo a curva analítica. Isso deixa claro como uma malha mais refinada faz toda a diferença na qualidade da solução numérica.

Erro máximo absoluto

Na Tabela 5, estão os valores do erro máximo absoluto para os diferentes números de nós utilizados na discretização deste exemplo. Neste caso, observa-se nas Figuras 14, 16 e 17 que, à medida que o número de nós na malha aumenta, o erro tende a diminuir.

Tabela 5 – Erro máximo absoluto para diferentes números de nós

Número de Nó (N)	**Erro Máximo Absoluto**
5	0.041039
7	0.019102
15	0.003690

5.2.2 Conclusão

Neste exemplo, mostramos como a solução numérica, obtida pelo MEF, se aproxima da solução analítica do PVC estudado. Observamos que, ao aumentar o número de nós na malha, a qualidade da solução numérica melhora significativamente, ou seja, quanto mais refinamos a malha, conseguimos nos aproximar cada vez mais da solução analítica. Esse comportamento está relacionado à diminuição do erro numérico, que tende a reduzir à medida que a malha se torna mais fina, refletindo uma maior aproximação da solução analítica.

5.3 Exemplo 2

Neste segundo exemplo, consideramos a equação de Poisson unidimensional em outra situação envolvendo a função f.O PVC a ser resolvido é dado por:

$$\begin{cases} -u''(x) = f(x), & \text{para } 0 < x < 1, \\ u(0) = u(1) = 0, \end{cases} \tag{5.2}$$

Este exemplo é mais complexo em relação ao exemplo 5.1, pois a solução analítica $u(x) = x(1 - x^3)$ é de polinômio de grau maior e quer dizer que são mais difíceis de aproximar numericamente. Isso torna a aplicação do MEF um pouco mais desafiadora.

Dados Iniciais:

- **Domínio**: $0 \leq x \leq 1$ com comprimento $L = 1$.

- **Função Fonte**: $f(x) = 12x^2$.
- **Solução Analítica**: $u(x) = x(1 - x^3)$.
- **Método de Quadratura**: Utilize a quadratura de Gauss com 4 pontos.
- **Aumento do Número de Nós na Malha**: Resolva o problema com 4, 8 e 18 nós.

Solução Analítica:

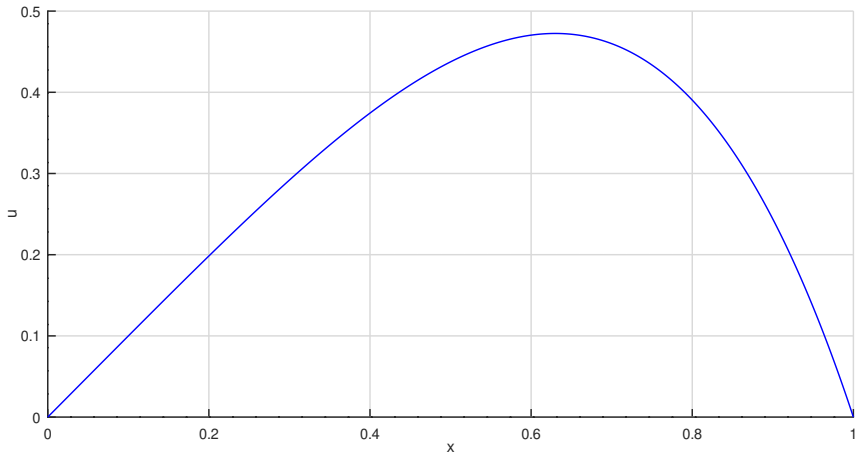


Figura 18 – Exemplo 2 - solução analítica (linha azul).

Fonte: Autor

5.3.1 Resultados

Solução Numérica com 4 Nós:

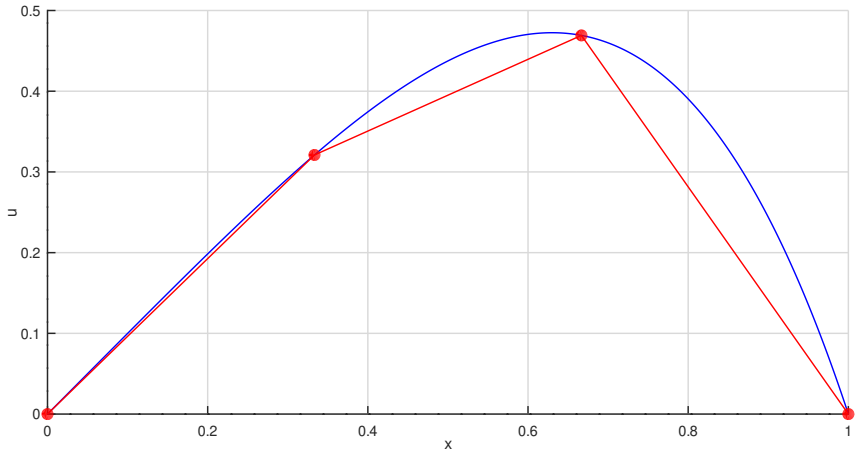


Figura 19 – Exemplo 2 - 4 nós, solução analítica (linha azul), solução numérica (linha vermelha) e valor da solução numérica nos nós (pontos vermelhos).

Fonte: Autor

Na Figura 19, comparamos a solução numérica $u_h(x)$ com 4 nós à solução analítica $u(x) = x(1 - x^3)$. Como estamos usando poucos nós, a aproximação tem uma qualidade um pouco inferior, e é possível perceber diferenças em relação à solução analítica. Já que, quanto maior o número de nós, melhor tende a ser a qualidade da solução numérica.

A seguir, a Figura 20 abaixo apresenta os elementos da matriz A, do vetor $\mathbf{b}$ e da solução $\mathbf{u}$, calculados a partir do sistema linear $A\mathbf{u} = \mathbf{b}$ para o número de nós especificado na Figura 19.

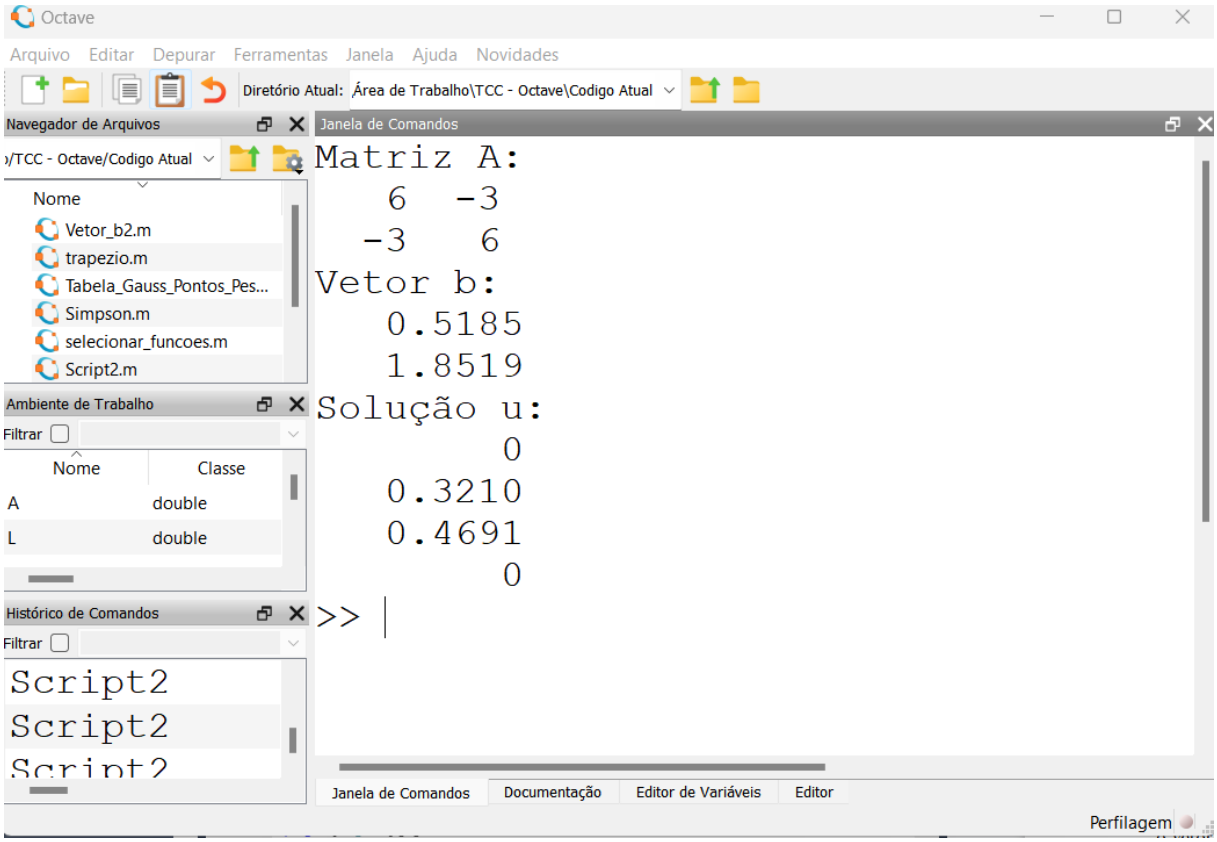


Figura 20 – Exemplo 2 - 4 nós - resultado do sistema linear.

Fonte: Autor

Na Figura 20, podemos ver o resultado do sistema linear que resolvemos. Nela, estão apresentados os valores que calculamos para a Matriz A, o vetor $\mathbf{b}$ e o vetor $\mathbf{u}$ é o vetor solução que estamos buscando. A partir desses valores, conseguimos construir a solução aproximada do nosso PVC apresentado na figura.

Solução Numérica com 8 Nós:

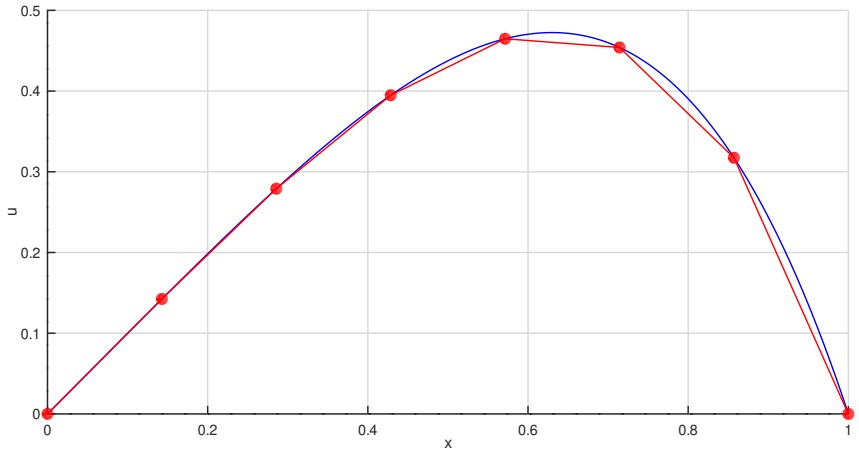


Figura 21 – Exemplo 2 - 8 nós, solução analítica (linha azul), solução numérica (linha vermelha) e valor da solução numérica nos nós (pontos vermelhos).

Fonte: Autor

Na Figura 21, comparamos a solução numérica $u_h(x)$ com 8 nós à solução analítica $u(x) = x(1-x^3)$. Com 8 nós, a aproximação já apresenta uma qualidade bem melhor em comparação com a solução obtida com 4 nós, mas ainda não chega a ser muito próxima da solução analítica em regiões de maior curvatura da solução analítica. Isso mostra como, à medida que aumentamos o número de nós, a qualidade da solução numérica tende a melhorar, embora, mesmo com 8 nós, a solução numérica ainda não seja totalmente satisfatória em relação à analítica.

Solução Numérica com 18 Nós:

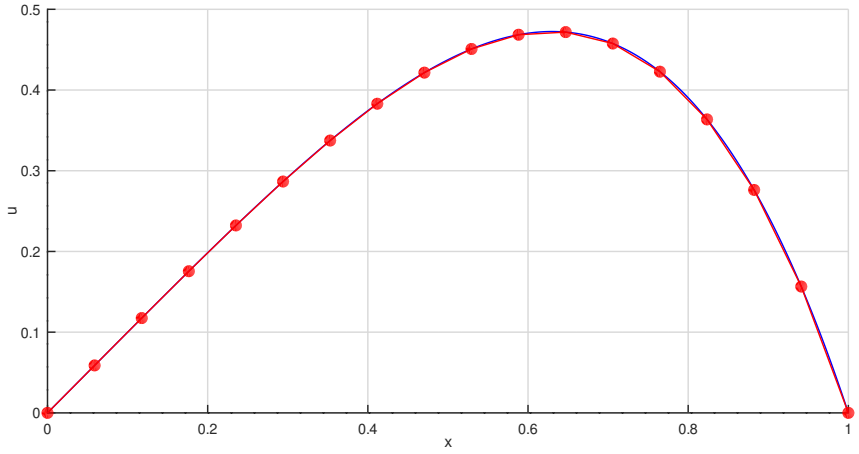


Figura 22 – Exemplo 2 - 18 nós, solução analítica (linha azul), solução numérica (linha vermelha) e valor da solução numérica nos nós (pontos vermelhos).

Fonte: Autor

Na Figura 22, com 18 nós, a solução numérica $u_h(x)$ já está bem mais próxima à solução analítica $u(x) = x(1 - x^3)$. Com mais pontos na malha, a aproximação teve uma qualidade bem melhor, quase se sobrepondo visualmente à curva analítica. Isso mostra como uma malha mais refinada realmente melhora bastante a qualidade da solução numérica.

Erro máximo absoluto

Na Tabela 6, estão apresentados os valores do erro máximo absoluto para os diferentes números de nós utilizados. Da mesma forma, observamos nas Figuras 19, 21 e 22 o erro máximo absoluto diminuindo à medida que o número de nós aumenta.

Tabela 6 – Erro máximo absoluto para diferentes números de nós

Número de Nó (N)	**Erro Máximo Absoluto**
4	0.117020
8	0.026439
18	0.004891

5.3.2 Conclusão

Neste exemplo, mostramos como a solução numérica, calculada pelo MEF, fica mais próxima da solução analítica a medida que mais nós são acrescentados a malha. Percebemos que, ao aumentar o número de nós na malha, a aproximação da solução numérica melhora bastante, ficando cada vez mais próxima da solução analítica.

5.4 Exemplo 3

Consideramos mais um novo exemplo envolvendo a equação de Poisson unidimensional. A equação a ser resolvida é novamente dada por:

$$\begin{cases} -u''(x) = f(x), & \text{para } 0 < x < 1, \\ u(0) = u(1) = 0, \end{cases} \tag{5.3}$$

Este exemplo é mais complicado do que os anteriores, pois a solução analítica $u(x) = \text{sen}(2\pi x)$. é uma função trigonométrica, ao invés de ser um polinômio. Isso torna o comportamento da solução mais complexo e exige mais cuidado ao aplicar o MEF. A função seno, por ser oscilante, pode dificultar a qualidade dos resultados numéricos.

Dados Iniciais:

- **Domínio**: $0 \leq x \leq 1$ com comprimento $L = 1$.

- **Função Fonte**: $f(x) = 4\pi^2 \text{sen}(2\pi x)$.
- **Solução Analítica**: $u(x) = \text{sen}(2\pi x)$.
- **Método de Quadratura**: Utilize a quadratura de Gauss com 4 pontos.
- **Aumento do Número de Nós na Malha**: Resolva o problema com 5, 11 e 21 nós.

Solução Analítica:

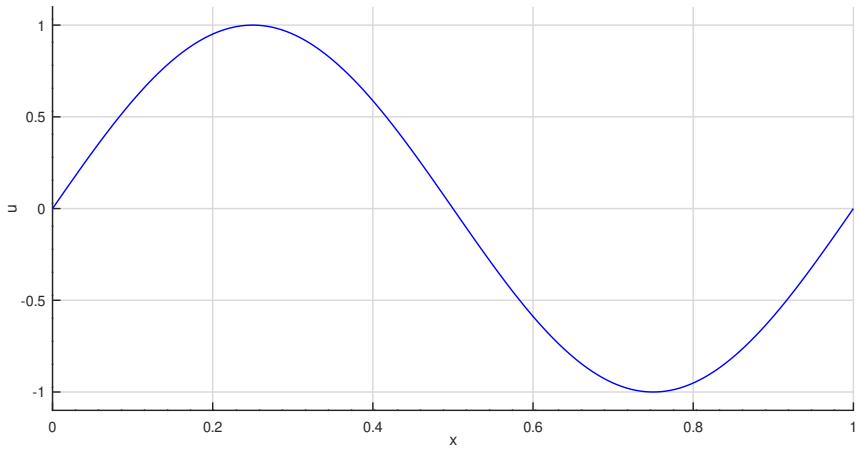


Figura 23 – Exemplo 3 - solução analítica (linha azul).

Fonte: Autor

5.4.1 Resultados

Solução Numérica com 5 Nós:

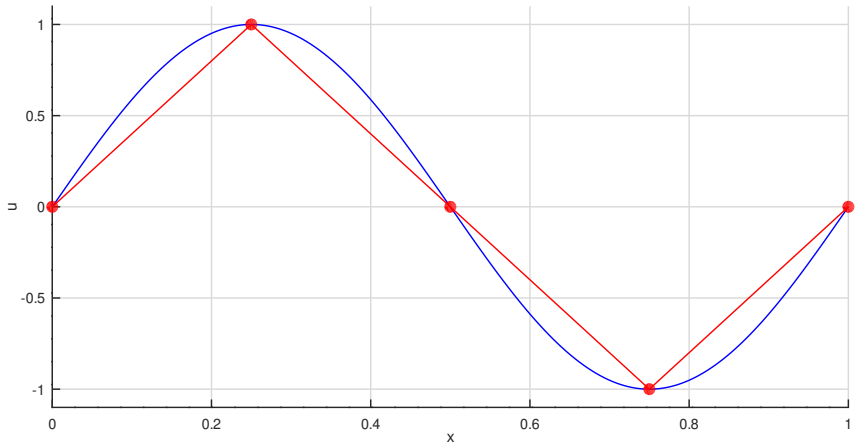


Figura 24 – Exemplo 3 - 5 nós, solução analítica (linha azul), solução numérica (linha vermelha) e valor da solução numérica nos nós (pontos vermelhos).

Fonte: Autor

Na Figura 24, podemos observar a solução numérica $u_h(x)$ com 5 nós em comparação com a solução analítica $u(x) = \text{sen}(2\pi x)$. Como utilizamos poucos nós, a qualidade da aproximação não é tão boa, e é possível perceber diferenças significativas em relação à solução analítica. Isso demonstra como uma malha com menos pontos pode resultar em uma solução de menor qualidade.

Na Figura 25, apresentamos os valores do sistema linear gerados a partir da malha utilizada, que serão analisados a seguir.

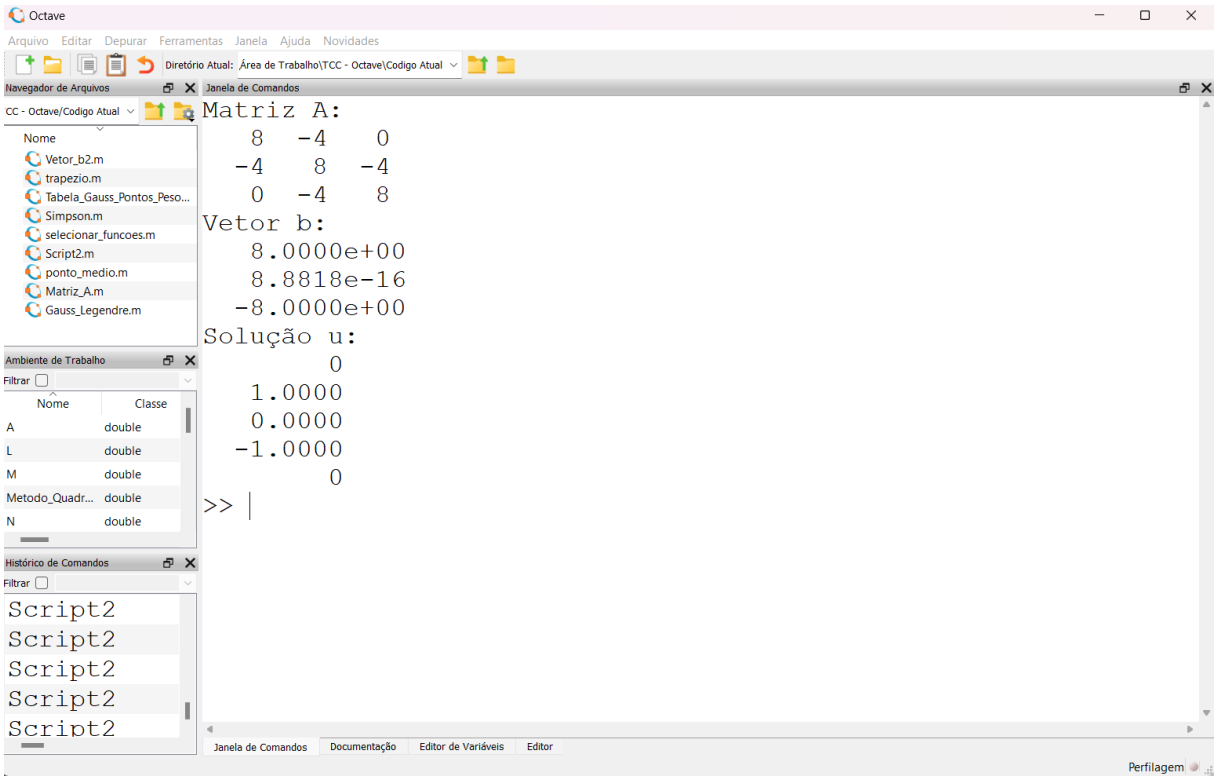


Figura 25 – Exemplo 3 - 5 nós - resultado do sistema linear.

Fonte: Autor

Na Figura 25, podemos observar os resultados obtidos após a resolução do sistema linear. Nela, estão mostrados os valores calculados para a Matriz A, o vetor $\mathbf{b}$ e o vetor solução $\mathbf{u}$. Esses valores são fundamentais para a construção da solução aproximada do problema, pois refletem como o sistema linear foi montado a partir da discretização dos nós.

Solução Numérica com 11 Nós:

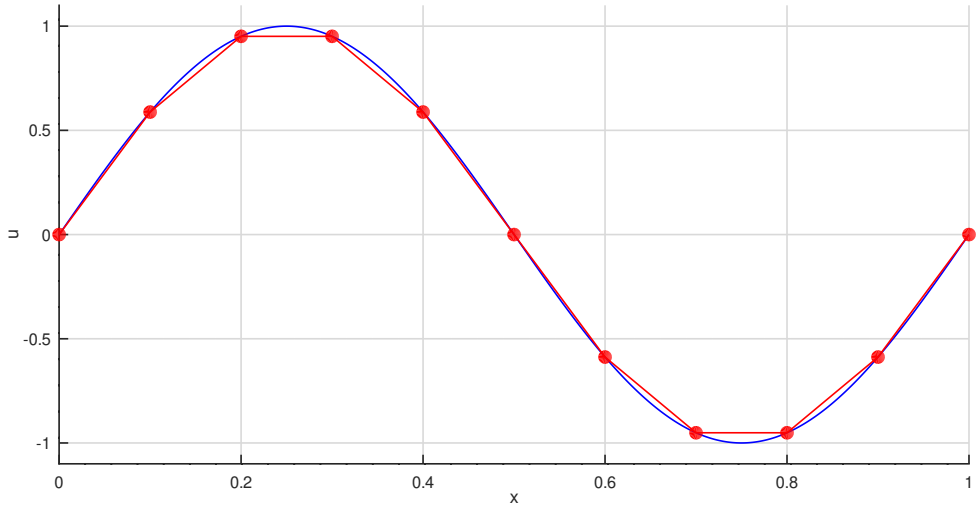


Figura 26 – Exemplo 3 - 11 nós, solução analítica (linha azul), solução numérica (linha vermelha) e valor da solução numérica nos nós (pontos vermelhos).

Fonte: Autor

Na figura 26, a solução numérica $u_h(x)$ com 11 nós já aproxima melhor a solução analítica. A curva se torna mais suave e as diferenças entre as soluções diminuem significativamente. Isso demonstra que, ao aumentar o número de nós, a aproximação da solução numérica melhora.

Solução Numérica com 21 Nós:

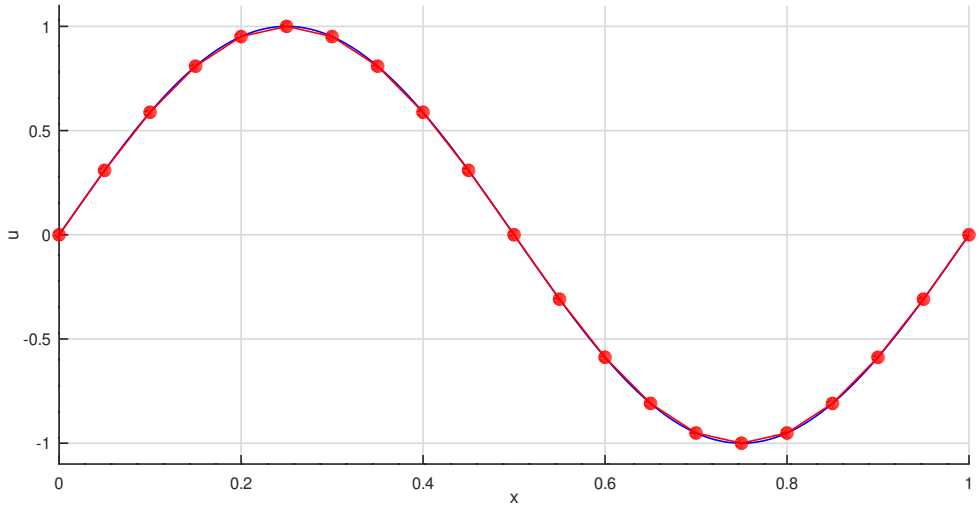


Figura 27 – Exemplo 3 - 21 nós, solução analítica (linha azul), solução numérica (linha vermelha) e valor da solução numérica nos nós (pontos vermelhos).

Fonte: Autor

Na Figura 27, com 21 nós, a solução numérica $u_h(x)$ se aproxima consideravelmente da solução analítica $u(x) = \sin(2\pi x)$. Com um maior número de pontos na malha, a

qualidade da aproximação melhorou significativamente, chegando a se sobrepor quase que completamente à curva analítica. Isso mostra de forma clara como uma malha mais refinada contribui para uma solução numérica de melhor qualidade.

Erro máximo absoluto

Na Tabela 7, são apresentados os valores do erro máximo absoluto para diferentes números de nós. Nas Figuras 24, 26 e 27, observa-se que o erro diminui com o aumento da quantidade de nós.

Tabela 7 – Erro máximo absoluto para diferentes números de nós

Número de Nó (N)	**Erro Máximo Absoluto**
5	0.210510
11	0.048943
21	0.012160

5.4.2 Conclusão

Neste exemplo, mostramos como a solução numérica, obtida utilizando o MEF, se aproxima progressivamente da solução analítica. Observamos que, à medida que aumentamos o número de nós na malha, a qualidade da solução numérica melhora consideravelmente, tornando-se cada vez mais próxima da solução analítica. Além disso, os dados fornecidos no exemplo, como a fonte $f(x) = 4\pi^2 \sin(2\pi x)$ utilizada neste caso, mostram que, embora seja possível resolver o problema analiticamente, a solução numérica pode ser bastante satisfatória. Além disso, em situações nas quais a solução analítica é difícil de encontrar ou leva muito tempo, o uso do MEF se apresenta como uma alternativa indispensável, oferecendo uma aproximação rápida e prática, especialmente quando trabalhamos com malhas mais refinadas.

6 Conclusões

Neste trabalho, foi apresentado o MEF para a resolução das equações diferenciais, com foco na equação de Poisson unidimensional. A partir da formulação variacional da equação, conseguimos obter um sistema linear utilizado para a obtenção da solução numérica.

A implementação do MEF foi realizada no programa GNU Octave, um ambiente de programação que facilita a execução de cálculos numéricos de forma eficiente. Desenvolvemos os códigos necessários para resolver o sistema linear gerado pela formulação variacional, destacando a construção da matriz A, do vetor $\mathbf{b}$ do sistema linear e dos métodos de integração numérica. Após a criação dos códigos, estes foram aplicados a exemplos com diferentes conjuntos de dados, sendo que, para cada exemplo, uma função fonte distinta foi utilizada. Os resultados mostraram como a solução numérica se aproximava da solução analítica à medida que o número de nós aumentava.

Com relação aos resultados obtidos, foi possível observar que as soluções numéricas geradas pelos códigos implementados apresentaram uma excelente aproximação às soluções analíticas, à medida que a quantidade de nós fosse aumentada, confirmando a eficácia do MEF nesse tipo de problema. Além disso, a possibilidade de analisar as soluções por meio de gráficos foi essencial para entender o comportamento da solução conforme o número de nós na malha aumentava.

Como sugestão para a continuidade deste trabalho, recomenda-se expandir o estudo para a aplicação do MEF para problemas bidimensionais. A utilização da linguagem Python seria uma excelente escolha, pois ela oferece um conjunto abrangente de ferramentas que facilitam a implementação de métodos numéricos e a visualização dos resultados. Isso permitiria abordar problemas mais complexos, proporcionando uma análise mais aprofundada e ampliando as possibilidades de aplicação do MEF.

Referências

ANTON, H.; RORRES, C. *Álgebra Linear com Aplicações*. 8. ed. Porto Alegre: Bookman, 2001. Citado na página 12.

AUGUSTYNIAK, M.; USAREK, Z. Finite element method applied in electromagnetic ndte: A review. *Journal of Nondestructive Evaluation*, Springer, v. 35, p. 1–15, 2016. Citado na página 11.

BECKER, E. B.; CAREY, G. F.; ODEN, J. T. *Finite Elements: An Introduction*. [S.l.]: Prentice-Hall, 1981. Citado na página 10.

BENITEZ, G. *Aula 18 - Método de Galerkin*. 2017. Acesso em: 23 jan. 2025. Disponível em: <https://www.professores.uff.br/gbenitez/wp-content/uploads/sites/98/2017/08/Aula_18.pdf>. Citado na página 12.

BRUNOW, R. *Curso de Introdução ao Método dos Elementos Finitos (MEF)*. 2017. Acesso em: 23 jan. 2025. Disponível em: <https://www.professores.uff.br/rbrunow/wp-content/uploads/sites/99/2017/08/Curso_MEF1.pdf>. Citado na página 10.

BURDEN, R. L.; FAIRES, J. D. *Análise Numérica*. 8. ed.. ed. São Paulo: Cengage Learning, 2008. ISBN 978-85-221-0601-1. Citado 3 vezes nas páginas 12, 19 e 45.

CALLE, J. L. D.; DEVLOO, P. R. B.; GOMES, S. M. Stabilized discontinuous galerkin method for hyperbolic equations. *Computer Methods in Applied Mechanics and Engineering*, 2004. Citado na página 10.

ELTES, P. E. et al. Development of a computer-aided design and finite element analysis combined method for affordable spine surgical navigation with 3d-printed customized template. *Frontiers in Surgery*, Frontiers Media SA, v. 7, p. 583386, 2021. Citado na página 11.

FARLOW, S. J. *Partial Differential Equations: for scientists and engineers*. New York: Dover Publications, 1993. Citado na página 12.

HASHIMOTO, M.; SUZUKI, S.; NAKAMURA, K. Application of the finite element technique to aerodynamic problems of aircraft. *Computers & structures*, Elsevier, v. 19, n. 1-2, p. 57–69, 1984. Citado na página 11.

INCE, E. S. et al. Forward gravity modelling to augment high-resolution combined gravity field models. *Surveys in Geophysics*, Springer, v. 41, p. 767–804, 2020. Citado na página 11.

IóRIO, V. *EDP: Um Curso de Graduação*. 2. ed. Rio de Janeiro: IMPA, 2005. Citado na página 12.

JOHNSON, C. *Numerical Solutions of Partial Differential Equations by the Finite Element Method*. Lund, Sweden: Studentlitteratur, 1987. ISBN 0 521 345 146. Citado 6 vezes nas páginas 11, 12, 22, 27, 29 e 30.

LEISSA, A. The historical bases of the rayleigh and ritz methods. *Journal of Sound and Vibration*, v. 287, n. 4–5, p. 961–978, 2005. Acesso em: 23 jan. 2025. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0022460X05000362>. Citado na página 10.

LIMA, E. L. *Análise Real Volume 1: Funções de uma Variável*. 10. ed. Rio de Janeiro: IMPA, 2009. Citado na página 12.

LIMA, E. L. *Álgebra Linear*. 1. ed. Rio de Janeiro: IMPA, 2014. 357 p. (Coleção Matemática Universitária). ISBN 978-85-244-0390-3. Citado na página 12.

LOTTI, R. S. et al. Aplicabilidade científica do método dos elementos finitos. *Revista Dental Press de Ortodontia e Ortopedia Facial*, SciELO Brasil, v. 11, p. 35–43, 2006. Citado na página 10.

ODEN, J. T.; CAREY, G. F.; BECKER, E. B. *Finite Elements: An Introduction*. New Jersey - USA: Prentice Hall Inc., 1981. Vol. 1. Citado na página 11.

SCHNITZER, J. E.; LAMBRAKIS, K. C. Electrostatic potential and born energy of charged molecules interacting with phosholipid membranes: Calculation via 3-d numerical solution of the full poisson equation. *Journal of theoretical biology*, Elsevier, v. 152, n. 2, p. 203–222, 1991. Citado na página 11.

SOUZA, F. et al. Análise estrutural pelo método de elementos finitos. *Revista Campo do Saber*, v. 4, n. 3, 2018. Citado na página 11.

SOUZA, R. M. de. O método dos elementos finitos aplicado ao problema de condução de calor. *Apostila, Universidade Federal do Pará, Belém*, 2003. Citado na página 21.

STERZA, R. de L. et al. Escoamento na cavidade com tampa móvel: Diferentes abordagens na resoluçao da equaçao de poisson. *Proceeding Series of the Brazilian Society of Computational and Applied Mathematics*, v. 7, n. 1, 2020. Citado na página 11.

TEIXEIRA, P. R. de F.; AWRUCH, A. M. Análise numérica de escoamentos de fluidos incompressíveis incluindo problemas com superfícies livres usando mef. *Mecánica Computacional*, n. 1, p. 30–37, 2001. Citado na página 11.

THOMAS, G. B.; WEIR, M. D.; HASS, J. *Cálculo, volume 1*. [S.l.]: Pearson Education do Brasil, 2012. Revisão técnica Cláudio Hifume Asano. Citado na página 12.

UNESP. *Tutorial de Octave para Iniciantes*. [S.l.], 2023. Acessado em: 23 jan. 2025. Disponível em: <https://ava.furg.br/pluginfile.php/790006/mod_resource/content/1/Tutorial_Octave_Unesp.pdf>. Citado na página 34.

UNICAMP. *GNU Octave*. 2023. Acessado em: 23 jan. 2025. Disponível em: <https://www.ea2.unicamp.br/ensino-digital-3/gnu-octave/>. Citado na página 34.

A Apêndice

A.1 Códigos no Octave

A.1.1 Script Inicial Completo: `Script2.m`

```
clc; clear;
% Dados Iniciais:
L = 1;     % Tamanho do intervalo
N = 4;    % Numeros de n s
n = N-1;  % Numero de intervalos
M = N-2;  % Numero de n s internos
h = L/n;  %tamanho do intervalo entre os n s
x = linspace(0,L,N); % Distancia de cada n
%-----------------------------------------
% Escolher a op   o (entre 1 e 5)
   opcao = 3; % Troque este n mero para mudar as fun   es

% Selecionar as fun   es fonte e anal tica
   [f, U] = selecionar_funcoes(opcao);
%-----------------------------------------
% Criar a Matriz A:
   A = Matriz_A(M, h);
%-----------------------------------------
% Escolher o Metodo de Quadratura
%  1 = Ponto Medio
%  2 = Trapezio
%  3 = Simpson
%  4 = Gauss_Legendre

   n_Gauss = 4; % Numero de pontos da quadratura de Gauss-Legendre
   Metodo_Quadratura = 4; % Troque este n mero para mudar a quadratura

   [xi, ci] = Tabela_Gauss_Pontos_Pesos(n_Gauss);
%-----------------------------------------
% Criar o Vetor b:

   b = Vetor_b2(x, f, h, M, Metodo_Quadratura,n_Gauss);
%-----------------------------------------
% Solu   o numerica de u:
   u = A\b;
%-----------------------------------------
% Condi   es de Contorno zero nas extremidades:
   u = [0;u;0];
%-----------------------------------------
   % Calcular o erro absoluto entre a solu   o num rica e a anal tica:

   S=[0:0.0001:1];
   U2 = interp1(x, u, S);

   erro_absoluto = abs(U2 - U(S));
   erro_maximo = max(erro_absoluto); % Erro m ximo
```

```
    % Exibir o erro:
    %disp('Erro absoluto em cada n : ');
    %disp(erro_absoluto);
    disp(['Erro m ximo: ', num2str(erro_maximo)]);

%-----------------------------------------
% Mostrar a Matriz A - Vetor b - Solu  o numerica (u):

    disp('Matriz A: ');
    disp(A);

    disp('Vetor b: ') ;
    disp(b);

    disp('Solu  o u: ');
    disp(u);

%-----------------------------------------
% Plotar o Grafico da solu  o analitica e numerica:

    figure(1)
    X=[0:0.001:1];
    hold on;
    plot( X,U(X), 'b-', 'LineWidth', 1, 'DisplayName', 'Solu  o Anal tica'); %
        Adiciona legenda para solu  o anal tica
    plot(x, u, 'r-', 'LineWidth', 1, 'DisplayName', 'Solu  o  Num rica'); %
       Adiciona linha destacada para solu  o num rica
    plot(x, u, '.', 'markersize', 20, 'color', 'red' , 'DisplayName', 'N  da
       Solu  o  Num rica'); % Adiciona ponto para solu  o num rica
    hold off;
    ylim([0, 0.5]);
    %ylim([-1.10, 1.10]); % Define os limites do eixo y para ampliar a
       visualiza  o
    %xlim([0, 0]);
    %title('Compara  o da Solu  o Anal tica com a Solu  o Numerica');
    legend show; % Exibe a legenda no gr fico
    grid on;
```

Listing A.1 – Para Escolher a Função Fonte e Analitica

A.1.2 Função Fonte e Analítica: `selecionar-funcoes.m`

```
function [f, U] = selecionar_funcoes(opcao)
    % Fun  o para selecionar fun  es fonte e anal ticas com base em uma
        op  o
    % opcao: inteiro de 1 a 5 indicando a escolha
    % Retorna f (fonte) e U (anal tica)

    switch opcao
        case 1
            f = @(x) 2;
            U = @(x) x .* (1 - x);
        case 2
            f = @(x) 6 * x;
            U = @(x) x .* (1 - x.^2);
        case 3
            f = @(x) 12 * x.^2;
            U = @(x) x .* (1 - x.^3);
        case 4
            f = @(x) 4 * pi^2 * sin(2 * pi * x);
            U = @(x) sin(2 * pi * x);
        case 5
            f = @(x) -4 * pi * cos(2 * pi * x) + 4 * pi^2 * x .* sin(2 * pi * x)
                ;
            U = @(x) x .* sin(2 * pi * x);
        otherwise
            error('Op  o inv lida. Escolha um valor de 1 a 5.');
    end
end
```

Listing A.2 – Para Escolher a Função Fonte e Analitica