

# Método de Elementos Finitos

Pedro H A Konzen

7 de dezembro de 2023

# Licença

Este trabalho está licenciado sob a Licença Atribuição-CompartilhaIgual 4.0 Internacional Creative Commons. Para visualizar uma cópia desta licença, visite [http://creativecommons.org/licenses/by-sa/4.0/deed.pt\\_BR](http://creativecommons.org/licenses/by-sa/4.0/deed.pt_BR) ou mande uma carta para Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.

# Prefácio

Nestas notas de aula são abordados temas introdutórios sobre o método de elementos finitos para a simulação de equações diferenciais. Como ferramenta computacional de apoio didático, apresentam-se códigos em [Python](#) com suporte da biblioteca [FEniCSx](#).

Agradeço a todos e todas que de modo assíduo ou esporádico contribuem com correções, sugestões e críticas. :)

Pedro H A Konzen

# Conteúdo

# Capítulo 1

## Problemas Unidimensionais

### 1.1 Interpolação e Projeção

Seja dado um intervalo  $I = [x_0, x_1] \subset \mathbb{R}$ ,  $x_0 \neq x_1$ . O **espaço vetorial das funções lineares** em  $I$  é definido por

$$P_1(I) := \{v : v(x) = c_0 + c_1x, x \in I, c_0, c_1 \in \mathbb{R}\}. \quad (1.1)$$

Observamos que dado  $v \in P_1(I)$ , temos que  $v$  é unicamente determinada pelos valores

$$\begin{aligned} \alpha_0 &= v(x_0), \\ \alpha_1 &= v(x_1). \end{aligned} \quad (1.2)$$

Como consequência, existe exatamente uma única função  $v \in P_1(I)$  para quaisquer dados valores  $\alpha_0$  e  $\alpha_1$ . Desta observação, introduzimos a chamada **base nodal** (base lagrangiana<sup>1</sup>)  $\{\varphi_0, \varphi_1\}$  para  $P_1(I)$ , definida por

$$\varphi_j(x_i) = \begin{cases} 1 & , i = j, \\ 0 & , i \neq j \end{cases}, \quad (1.3)$$

com  $i, j = 0, 1$ . Consulte a Figura ??.

---

<sup>1</sup>Consulte mais em [Notas de Aula: Matemática Numérica I: Interpolação de Lagrange](#).

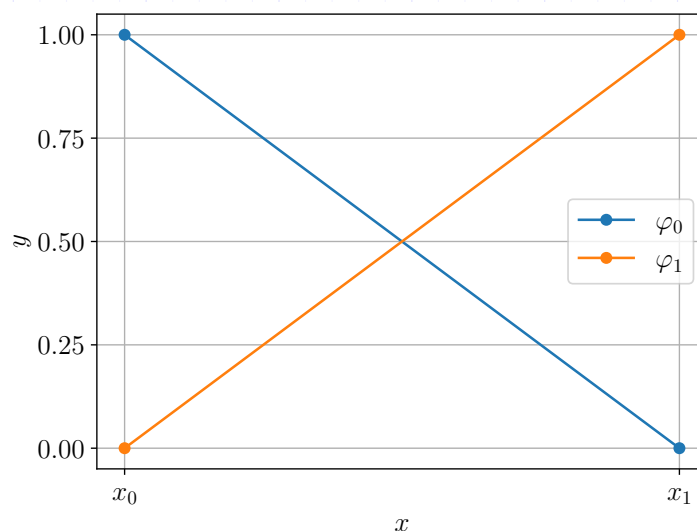


Figura 1.1: Base nodal para o espaço  $P_1([x_0, x_1])$ .

Com esta base, toda função  $v \in P_1(I)$  pode ser escrita como uma combinação linear das funções  $\varphi_0$  e  $\varphi_1$  com coeficientes  $\alpha_0$  e  $\alpha_1$  (**graus de liberdade**), i.e.

$$v(x) = \alpha_0 \varphi_0(x) + \alpha_1 \varphi_1(x). \quad (1.4)$$

Além disso, observamos que

$$\varphi_0(x) = \frac{x - x_1}{x_0 - x_1}, \quad (1.5)$$

$$\varphi_1(x) = \frac{x - x_0}{x_1 - x_0}. \quad (1.6)$$

Uma extensão do espaço  $P_1(I)$  é o **espaço das funções lineares por partes**. Dado  $I = [l_0, l_1]$ ,  $l_0 \neq l_1$ , consideramos uma partição (**malha**) de  $I$  com  $n + 1$  pontos

$$\mathcal{I} = \{l_0 = x_0, x_1, \dots, x_n = l_1\} \quad (1.7)$$

e, portanto, com  $n$  subintervalos  $I_i = [x_{i-1}, x_i]$  de comprimento (**tamanho da malha**)  $h_i = x_i - x_{i-1}$ ,  $i = 1, 2, \dots, n$ . Na malha  $\mathcal{I}$  definimos o seguinte **espaço das funções lineares por partes**

$$V_h := \{v : v \in C^0(\mathcal{I}), v|_{I_i} \in P_1(I_i), i = 1, 2, \dots, n\}. \quad (1.8)$$

Observamos que toda função  $v \in V_h$  é unicamente determinada por seus valores nodais  $\{\alpha_i = v(x_i)\}_{i=0}^n$ . Reciprocamente, todo conjunto de valores nodais  $\{\alpha_i\}_{i=0}^n$  determina unicamente uma função  $v \in V_h$ . Desta observação, temos que os **valores nodais determinam os graus de liberdade** com a base nodal  $\{\varphi_j\}_{j=0}^n$  para  $V_h$  definida por

$$\varphi_j(x_i) = \begin{cases} 1 & , i = j, \\ 0 & , i \neq j \end{cases}, \quad (1.9)$$

com  $i, j = 0, 1, \dots, n$ . Ou seja, temos que

$$v(x) = \sum_{j=0}^n \alpha_j \varphi_j(x). \quad (1.10)$$

Podemos verificar que

$$\varphi_i(x) = \begin{cases} (x - x_{i-1})/h_i & , x \in I_i, \\ (x_{i+1} - x)/h_{i+1} & , x \in I_{i+1}, \\ 0 & , \text{noutros casos} \end{cases} \quad (1.11)$$

consulte, Figura ???. É notável que  $\varphi_i(x)$  tem suporte compacto  $I_i \cup I_{i+1}$ .

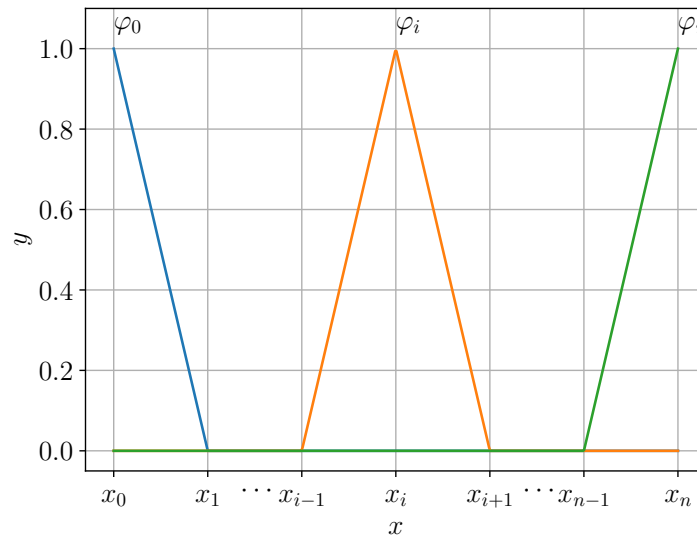


Figura 1.2: Base nodal para o espaço das funções lineares por parte.

### 1.1.1 Interpolação

[[tag:revisar]]

Interpolação é uma técnica de aproximação de funções. Dada uma função contínua  $f$  em  $I = [l_0, l_1]$ , definimos o **operador de interpolação linear**  $\pi : C^0(I) \rightarrow V_h$  por

$$\pi f(x) = \sum_{j=0}^n f(x_j) \varphi_j(x) \quad (1.12)$$

Observamos que  $\pi f$  é igual a  $f$  nos nodos  $x_j$ ,  $j = 0, 1, 2, \dots, n$ .

**Exemplo 1.1.1.** A Figura ?? ilustra a interpolação da função  $f(x) = 3 \sin(2\pi x)$  no espaço de elementos finitos  $V_h$  das funções lineares por partes com 5 células.

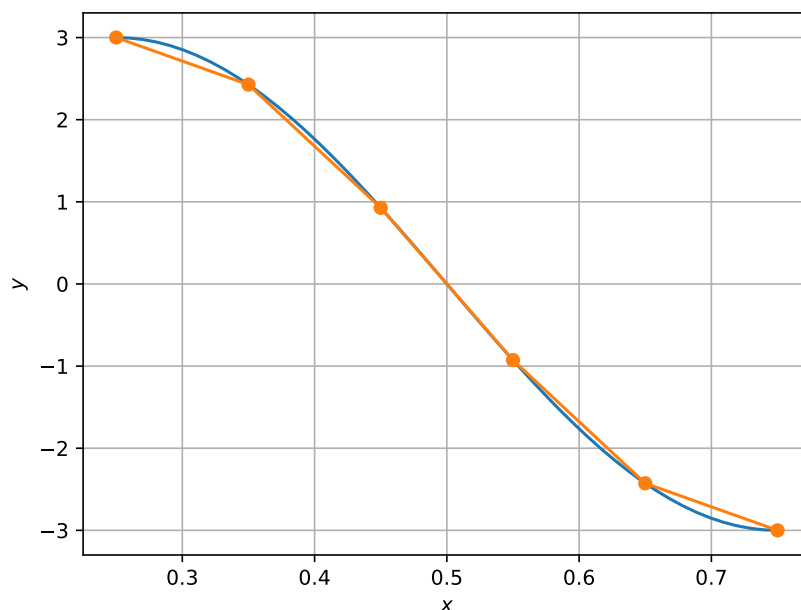


Figura 1.3: Interpolação linear de  $f(x) = 3 \sin(2\pi x)$  no espaço de elementos finitos  $V$ .

Código 1.1: mefld\_interp\_lin

```
1 from dolfinx import fem, mesh
```

Notas de Aula - Pedro Konzen \*/\* Licença CC-BY-SA 4.0



```

2 import ufl
3 import numpy as np
4 from mpi4py import MPI
5 import matplotlib.pyplot as plt
6
7 # malha
8 l0 = 0.25
9 l1 = 0.75
550 10 domain = mesh.create_interval(MPI.COMM_WORLD,
11                                nx = 5,
12                                points = [10, 11])
13 x = ufl.SpatialCoordinate(domain)
500 14
15 # espaço
16 V = fem.FunctionSpace(domain, ('P', 1))
17
450 18 # fun
19 def fun(x, mod):
20     return 3.*mod.sin(2.*mod.pi*x)
400 21
22 x = ufl.SpatialCoordinate(domain)
23 f_expr = fem.Expression(fun(x[0], ufl),
24                          V.element.interpolation_points())
350 25
26 # interpolação
27 pif = fem.Function(V)
300 28 pif.interpolate(f_expr)

```

Agora, vamos buscar medir o erro de interpolação, i.e.  $f - \pi f$ . Para tanto, podemos usar a norma  $L^2$  definida por

$$\|v\|_{L^2(I)} = \left( \int_I v^2 dx \right)^{1/2}. \quad (1.13)$$

Lembramos que valem a **desigualdade triangular**

$$\|v + w\|_{L^2(I)} \leq \|v\|_{L^2(I)} + \|w\|_{L^2(I)} \quad (1.14)$$

e a **desigualdade de Cauchy-Schwarz**<sup>2</sup>

$$\int_I vw \, dx \leq \|v\|_{L^2(I)} \|w\|_{L^2(I)}, \quad (1.15)$$

<sup>2</sup>Também conhecida como desigualdade de Cauchy–Bunyakovsky–Schwarz. Augustin-

para qualquer funções  $v, w \in L^2(I)$ .

**Proposição 1.1.1.** (Erro da interpolação linear) O interpolador  $\pi f : C^0(I) \rightarrow P_1(I)$  satisfaz as estimativas

$$\|f - \pi f\|_{L^2(I)} \leq Ch^2 \|f''\|_{L^2(I)}, \quad (1.16)$$

$$\|(f - \pi f)'\|_{L^2(I)} \leq Ch \|f''\|_{L^2(I)}, \quad (1.17)$$

onde  $C$  é uma constante e  $h = x_1 - x_0$ .

*Demonstração.* Denotemos o erro de interpolação por  $e = f - \pi f$ . Do teorema fundamental do cálculo, temos

$$e(y) = e(x_0) + \int_{x_0}^y e'(x) dx, \quad (1.18)$$

onde  $e(x_0) = f(x_0) - \pi f(x_0) = 0$ . Daí, usando a desigualdade de Cauchy-Schwarz (??), temos

$$e(y) = \int_{x_0}^y e' dx \quad (1.19)$$

$$\leq \int_{x_0}^y |e'| dx \quad (1.20)$$

$$\leq \int_I 1 \cdot |e'| dx \quad (1.21)$$

$$\leq \left( \int_I 1^2 dx \right)^{1/2} \left( \int_I e'^2 dx \right)^{1/2} \quad (1.22)$$

$$= h^{1/2} \left( \int_I e'^2 dx \right)^{1/2}, \quad (1.23)$$

donde

$$e(y)^2 \leq h \int_I e'^2 dx = h \|e'\|_{L^2(I)}^2. \quad (1.24)$$

Então, integrando em  $I$  obtemos

$$\|e\|_{L^2(I)}^2 = \int_I e^2(y) dy \leq \int_I h \|e'\|_{L^2(I)}^2 dy = h^2 \|e'\|_{L^2(I)}^2, \quad (1.25)$$

Louis Cauchy, 1789 - 1857, matemático francês. Viktor Yakovlevich Bunyakovsky, 1804 - 1889, matemático Russo. Karl Hermann Amandus Schwarz, 1843 - 1921, matemático alemão.

ou seja, temos a seguinte desigualdade

$$\|e\|_{L^2(I)} \leq h\|e'\|_{L^2(I)}. \quad (1.26)$$

Agora, observando que  $e(x_0) = e(x_1) = 0$ , o **teorema de Rolle**<sup>3</sup> garante a existência de um ponto  $\tilde{x} \in I$  tal que  $e'(\tilde{x}) = 0$ , donde do teorema fundamental do cálculo e da desigualdade de Cauchy-Schwarz, segue

$$e'(y) = e'(\tilde{x}) + \int_{\tilde{x}}^y e'' dx \quad (1.27)$$

$$= \int_{\tilde{x}}^y e'' dx \quad (1.28)$$

$$\leq \int_I 1 \cdot |e''| dx \quad (1.29)$$

$$\leq h^{1/2} \left( \int_I e''^2 \right)^{1/2}. \quad (1.30)$$

Então, integrando em  $I$ , obtemos

$$\|e'\|_{L^2(I)}^2 \leq h^2 \|e''\|_{L^2(I)}^2, \quad (1.31)$$

a qual, observando que  $e'' = f''$ , equivale a segunda estimativa procurada, i.e.

$$\|(f - \pi f)'\|_{L^2(I)} \leq Ch \|f''\|_{L^2(I)}. \quad (1.32)$$

Por fim, de (??) e de (??), obtemos a primeira estimativa desejada

$$\|f - \pi f\|_{L^2(I)} \leq Ch^2 \|f''\|_{L^2(I)}. \quad (1.33)$$

□

Vamos, agora, generalizar o resultado da Proposição ?? para a interpolação no espaço  $V_h$  das funções lineares por parte.

O seguinte resultado fornece uma estimativa do erro de interpolação em relação ao tamanho  $h_i$  de cada elemento da malha.

<sup>3</sup>Michel Rolle, 1652 - 1719, matemático francês.

**Proposição 1.1.2.** *O interpolador  $\pi f$  satisfaz as estimativas*

$$\|f - \pi f\|_{L^2(I)}^2 \leq C \sum_{i=1}^n h_i^4 \|f''\|_{L^2(I)}^2, \quad (1.34)$$

$$\|(f - \pi f)'\|_{L^2(I)}^2 \leq C \sum_{i=1}^n h_i^2 \|f''\|_{L^2(I)}^2. \quad (1.35)$$

$$(1.36)$$

*Demonstração.* Ambas desigualdades seguem da desigualdade triangular e da Proposição ???. Por exemplo, para a primeira desigualdade, temos

$$\|f - \pi f\|_{L^2(I)}^2 \leq \sum_{i=1}^n \|f - \pi f\|_{L^2(I_i)}^2 \quad (1.37)$$

$$\leq \sum_{i=1}^n C h_i^4 \|f''\|_{L^2(I_i)}^2. \quad (1.38)$$

□

### 1.1.2 Projeção $L^2$

[[tag:revisar]]

Dada uma função  $f \in L^2(I)$ , definimos o **operador de projeção  $L^2$**   $P_h : L^2(I) \rightarrow V_h$  por

$$\int_I (f - P_h f) v \, dx = 0, \quad \forall v \in V_h. \quad (1.39)$$

Como  $V_h$  é um espaço de dimensão finita, a condição (??) é equivalente a

$$\int_I (f - P_h f) \varphi_i \, dx = 0, \quad i = 0, 1, \dots, n, \quad (1.40)$$

onde  $\varphi_i$  é a  $i$ -ésima função base de  $V_h$ . Além disso, como  $P_h f \in V_h$ , temos

$$P_h f = \sum_{j=0}^n \xi_j \varphi_j, \quad (1.41)$$

onde  $\xi_j$ ,  $j = 0, 1, \dots, n$ , são  $n + 1$  incógnitas a determinar. Logo,

$$\int_I (f - P_h f) \varphi_i \, dx = 0 \quad (1.42)$$

$$\int_I f \varphi_i dx = \int_I P_h f \varphi_i dx \quad (1.43)$$

$$\int_I f \varphi_i dx = \int_I \left( \sum_{j=0}^n \xi_j \varphi_j \right) \varphi_i dx \quad (1.44)$$

$$\sum_{j=0}^n \xi_j \int_I \varphi_j \varphi_i dx = \int_I f \varphi_i dx, \quad (1.45)$$

para  $i = 0, 1, \dots, n$ .

Observamos que (??) consiste em um sistema de  $n + 1$  equações lineares para as  $n + 1$  incógnitas  $\xi_j$ ,  $j = 0, 1, \dots, n$ . Este, por sua vez, pode ser escrito na seguinte forma matricial

$$M\xi = b, \quad (1.46)$$

onde  $M = [m_{i,j}]_{i,j=0}^{n+1}$  é chamada de **matriz de massa**

$$m_{i,j} = \int_I \varphi_j \varphi_i dx \quad (1.47)$$

e  $b = (b_0, b_1, \dots, b_n)$  é chamado de **vetor de carregamento**

$$b_i = \int_I f \varphi_i dx. \quad (1.48)$$

Ou seja, a projeção  $L^2$  de  $f$  no espaço  $V_h$  é

$$P_h f = \sum_{j=0}^n \xi_j \varphi_j, \quad (1.49)$$

onde  $\xi = (\xi_0, \xi_1, \dots, \xi_n)$  é solução do sistema (??).

**Exemplo 1.1.2.** A Figura ?? ilustra a projeção  $L^2$  da função  $f(x) = 3 \sin(2\pi x)$  no espaço  $V_h$  das funções lineares por partes em uma malha uniforme do intervalo  $I = [1/4, 3/4]$  com  $n = 4$  subintervalos (5 células).

Código 1.2: ex\_mef1d\_proj.py

```
1 from dolfinx import fem, mesh
2 from dolfinx.fem.petsc import LinearProblem
3 import ufl
4 from mpi4py import MPI
```

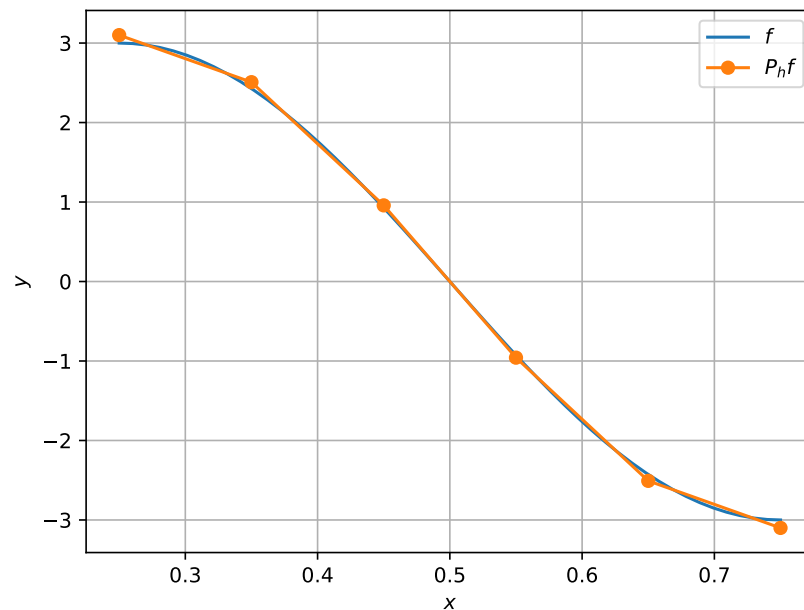


Figura 1.4: Projeção  $L^2$  de  $f(x) = 3\sin(2\pi x)$  no espaço  $V_h$  das funções lineares por partes sobre uma malha com 5 células.

```

5
6 # malha
7 l0 = 0.25
8 l1 = 0.75
9 domain = mesh.create_interval(MPI.COMM_WORLD,
10                                nx=5,
11                                points=[l0, l1])
12 x = ufl.SpatialCoordinate(domain)
13
14 # espaço
15 V = fem.functionspace(domain, ("P", 1))
16
17 # fun
18 f = 3.*ufl.sin(2.*ufl.pi*x[0])
19
20 # project f
21 u = ufl.TrialFunction(V)
22 v = ufl.TestFunction(V)

```

```

23 a = ufl.dot(u, v) * ufl.dx
24 L = ufl.dot(f, v) * ufl.dx
25 problem = LinearProblem(a, L, bcs=[])
26 Phf = problem.solve()

```

O próximo teorema mostra que  $P_h f$  é a função que melhor aproxima  $f$  dentre todas as funções do espaço  $V_h$ .

**Teorema 1.1.1.** (A melhor aproximação.) A projeção  $L^2$  satisfaz

$$\|f - P_h f\|_{L^2(I)} \leq \|f - v\|_{L^2(I)}, \quad \forall v \in V_h. \quad (1.50)$$

*Demonstração.* Dado  $v \in V_h$ , temos

$$\|f - P_h f\|_{L^2(I)}^2 = \int_I |f - P_h f|^2 dx \quad (1.51)$$

$$= \int_I (f - P_h f)(f - v + v - P_h f) dx \quad (1.52)$$

$$= \int_I (f - P_h f)(f - v) dx + \int_I (f - P_h f)(v - P_h f) dx \quad (1.53)$$

$$= \int_I (f - P_h f)(f - v) dx \quad (1.54)$$

$$\leq \|f - P_h f\|_{L^2(I)} \|f - v\|_{L^2(I)}, \quad (1.55)$$

donde segue o resultado.  $\square$

O próximo teorema fornece uma estimativa *a-priori* do erro  $\|f - P_h f\|_{L^2(I)}$  em relação ao tamanho da malha.

**Teorema 1.1.2.** A projeção  $L^2$  satisfaz

$$\|f - P_h f\|_{L^2(I)}^2 \leq C \sum_{i=1}^n h_i^4 \|f''\|_{L^2(I_i)}^2. \quad (1.56)$$

*Demonstração.* Tomando a interpolação  $\pi f \in V_h$ , temos do Teorema da melhor aproximação (Teorema ??) e da estimativa do erro de interpolação (Proposição ??) que

$$\|f - P_h f\|_{L^2(I)}^2 \leq \|f - \pi f\|_{L^2(I)}^2 \quad (1.57)$$

$$\leq C \sum_{i=1}^n h_i^4 \|f''\|_{L^2(I_i)}^2. \quad (1.58)$$

□

### 1.1.3 Exercícios

[[tag:revisar]]

**Exercício 1.1.1.** Faça um código para verificar a segunda estimativa da Proposição ?? no caso da interpolação da função  $f(x) = 3 \sin(2\pi x)$  no espaço  $P_1$  das funções lineares.

**Exercício 1.1.2.** Faça um código para verificar as estimativas da Proposição ?? no caso da interpolação da função  $f(x) = 3 \sin(2\pi x)$  no espaço  $V_h$  das funções lineares por partes.

**Exercício 1.1.3.** Faça um código para computar a projeção  $L^2$   $P_h f$  da função  $f(x) = x - \cos(x)$  no espaço  $V_h$  das funções lineares por partes em uma malha com 10 células no intervalo  $I = [0, \pi]$ . Faça o esboço dos gráficos de  $f$  e  $P_h f$  e compute o erro  $\|f - P_h f\|_{L^2(I)}$ .

## 1.2 Problema Modelo

[[tag:revisar]]

Nesta seção, discutimos sobre a aplicação do método de elementos finitos para o seguinte problema de valor de contorno: encontrar  $u$  tal que

$$-u'' = f, \quad x \in I = [0, L], \quad (1.59)$$

$$u(0) = u(L) = 0, \quad (1.60)$$

onde  $f$  é uma função dada.

### 1.2.1 Formulação Fraca

[[tag:revisar]]



A derivação de um método de elementos finitos inicia-se da formulação fraca do problema em um espaço de funções apropriado. No caso do problema (??)-(??), tomamos o espaço

$$V_0 = \{v \in H^1(I) : v(0) = v(1) = 0\}. \quad (1.61)$$

Ou seja, se  $v \in H^1(I)$ , então  $\|v\|_{L^2(I)} < \infty$ ,  $\|v'\|_{L^2(I)} < \infty$ , bem como  $v$  satisfaz as condições de contorno do problema.

A formulação fraca é, então, obtida multiplicando-se a equação (??) por uma função teste  $v \in V_0$  (arbitrária) e integrando-se por partes, i.e.

$$\int_I f v \, dx = - \int_I u'' v \, dx \quad (1.62)$$

$$= \int_I u' v' \, dx - u'(L)v(L) + u'(0)v(0) \quad (1.63)$$

$$(1.64)$$

Donde, das condições de contorno, temos

$$\int_I u' v' \, dx = \int_I f v \, dx. \quad (1.65)$$

Desta forma, o problema fraco associado a (??)-(??) lê-se: encontrar  $u \in V_0$  tal que

$$a(u, v) = L(v), \quad \forall v \in V_0, \quad (1.66)$$

onde

$$a(u, v) = \int_I u' v' \, dx \quad (1.67)$$

$$L(v) = \int_I f v \, dx, \quad (1.68)$$

são chamadas de forma bilinear e forma linear, respectivamente.

### 1.2.2 Formulação de Elementos Finitos

[[tag:revisar]]

Uma formulação de elementos finitos é uma aproximação do problema fraco (??) em um espaço de dimensão finita. Aqui, vamos usar o espaço  $V_{h,0}$  das

funções lineares por partes em  $I$  que satisfazem as condições de contorno, i.e.

$$V_{h,0} = \{v \in V_h : v(0) = v(L) = 0\}. \quad (1.69)$$

Então, substituindo o espaço  $V_0$  pelo subespaço  $V_{h,0} \subset V_0$  em (??), obtemos o seguinte problema de elementos finitos: encontrar  $u_h \in V_{h,0}$  tal que

$$a(u_h, v) = L(v), \quad \forall v \in V_{h,0}. \quad (1.70)$$

**Observação 1.2.1.** A formulação de elementos finitos não é única, podendo-se trabalhar com outros espaços de funções. No caso em que o espaço da solução é igual ao espaço das funções testes, a abordagem é chamada de método de Galerkin<sup>4</sup>.

Observemos que o problema (??) é equivalente a: encontrar  $u_h \in V_{h,0}$  tal que

$$a(u_h, \varphi_i) = L(\varphi_i), \quad i = 1, \dots, n-1, \quad (1.71)$$

onde  $\varphi_i$ ,  $i = 1, \dots, n-1$ , são as funções base de  $V_{h,0}$ . Então, como  $u_h \in V_{h,0}$ , temos

$$u_h = \sum_{j=1}^{n-1} \xi_j \varphi_j, \quad (1.72)$$

onde  $\xi_j$ ,  $j = 1, 2, \dots, n-1$ , são incógnitas a determinar. I.e., ao computarmos  $\xi_j$ ,  $j = 1, 2, \dots, n-1$ , temos obtido a solução  $u_h$  do problema de elementos finitos ??.

Agora, da forma bilinear (??), temos

$$a(u_h, \varphi_i) = a \left( \sum_{j=1}^{n-1} \xi_j \varphi_j, \varphi_i \right) \quad (1.73)$$

$$= \sum_{j=1}^{n-1} \xi_j a(\varphi_j, \varphi_i). \quad (1.74)$$

<sup>4</sup>Boris Grigoryevich Galerkin, matemático e engenheiro soviético. Fonte: [Wikipédia](#).

Daí, o problema (??) é equivalente a resolvermos o seguinte sistema de equações lineares

$$A\xi = b, \quad (1.75)$$

onde  $A = [a_{i,j}]_{i,j=1}^{n-1}$  é a matriz de rigidez com

$$a_{i,j} = a(\varphi_j, \varphi_i) = \int_I \varphi'_j \varphi'_i dx, \quad (1.76)$$

$\xi = (\xi_1, \xi_2, \dots, \xi_{n-1})$  é o vetor das incógnitas e  $b = (b_i)_{i=1}^{n-1}$  é o vetor de carregamento com

$$b_i = L(\varphi_i) = \int_I f \varphi_i dx. \quad (1.77)$$

**Exemplo 1.2.1.** Consideramos o problema (??)-(??) com  $f \equiv 1$  e  $L = 1$ , i.e.

$$-u'' = 1, \quad x \in I = [0, 1], \quad (1.78)$$

$$u(0) = u(1) = 0. \quad (1.79)$$

Neste caso, a solução analítica  $u(x) = -x^2/2 + x/2$  pode ser facilmente obtida por integração.

Agora, vamos computar uma aproximação de elementos finitos no espaço das funções lineares por partes  $V_{h,0} = \{v \in P_1(I); v(0) = v(1) = 0\}$  construído numa malha uniforme de 5 células no intervalo  $I = [0, 1]$ . Para tanto, consideramos o problema fraco: encontrar  $u \in V_0 = \{v \in H^1(I); v(0) = v(1) = 0\}$  tal que

$$a(u, v) = L(v), \quad (1.80)$$

onde

$$a(u, v) = \int_I u' v' dx, \quad L(v) = \int_I f v dx. \quad (1.81)$$

Então, a formulação de elementos finitos associada, lê-se: encontrar  $u_h \in V_{h,0}$  tal que

$$a(u_h, v_h) = L(v_h), \quad \forall v_h \in V_{h,0}. \quad (1.82)$$

A Figura ?? apresenta o esboço dos gráficos da solução analítica  $u$  e da sua aproximação de elementos finitos  $u_h$ .

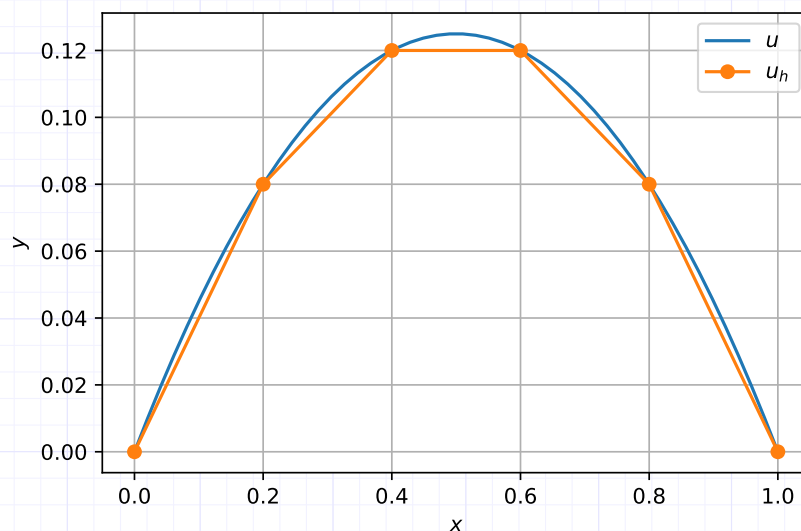


Figura 1.5: Esboço dos gráficos das soluções referentes ao Exemplo ??.

Código 1.3: ex\_mef1d\_modelo.py

```

1 from mpi4py import MPI
2
3 # malha
4 from dolfinx import mesh
5 domain = mesh.create_unit_interval(MPI.COMM_WORLD,
6                                     nx = 5)
7 # espaço
8 from dolfinx import fem
9 V = fem.functionspace(domain, ('P', 1))
10
11 # condição de contorno
12 import numpy as np
13 uD = fem.Function(V)
14 uD.interpolate(lambda x: np.full(x.shape[1], 0.))
15
16 tdim = domain.topology.dim
17 fdim = tdim - 1
18 domain.topology.create_connectivity(fdim, tdim)
19 boundary_facets = mesh.exterior_facet_indices(domain.topology)
20 boundary_dofs = fem.locate_dofs_topological(V, fdim,

```

```

21                                     boundary_facets)
650 22 bc = fem.dirichletbc(uD, boundary_dofs)
23
24 # problema mef
25 import ufl
600 26 from dolfinx import default_scalar_type
27 from dolfinx.fem.petsc import LinearProblem
28 u = ufl.TrialFunction(V)
550 29 v = ufl.TestFunction(V)
30
31 f = fem.Constant(domain, default_scalar_type(1.))
32
500 33 a = ufl.dot(ufl.grad(u), ufl.grad(v)) * ufl.dx
34 L = f * v * ufl.dx
35
36 problem = LinearProblem(a, L, bcs=[bc])
450 37 uh = problem.solve()

```

### 1.2.3 Estimativa *a Priori*

[[tag:revisar]]

Existem dois tipos de **estimativas do erro**  $e := u - u_h$ . Estimativas ***a priori***, são aquelas em que o erro é dado em relação da solução  $u$ , enquanto que nas estimativas ***a posteriori*** o erro é expresso em relação a solução de elementos finitos  $u_h$ .

**Teorema 1.2.1.** (**Ortogonalidade de Galerkin.**) A solução de elementos finitos  $u_h$  de (??) satisfaz a seguinte propriedade de ortogonalidade

$$a(u - u_h, v) := \int_I (u - u_h)' v' dx = 0, \quad v \in V_{h,0}, \quad (1.83)$$

onde  $u$  é a solução de (??).

*Demonstração.* De (??), (??) e lembrando que  $V_{h,0} \subset V_0$ , temos

$$a(u, v) = L(v) = a(u_h, v) \Rightarrow a(u - u_h, v) = 0, \quad (1.84)$$

para todo  $v \in V_{h,0}$ . □

**Teorema 1.2.2.** (**A melhor aproximação.**) A solução de elementos finitos  $u_h$  dada por (??) satisfaz a seguinte propriedade de melhor aproximação

$$\|(u - u_h)'\|_{L^2(I)} \leq \|(u - v)'\|_{L^2(I)}, \quad v \in V_{h,0}, \quad (1.85)$$

onde  $u$  é a solução de (??).

*Demonstração.* Escrevendo  $u - u_h = u - v + v - u_h$  para qualquer  $v \in V_{h,0}$  e usando a ortogonalidade de Galerkin (Teorema ??), temos

$$\|(u - u_h)'\|_{L^2(I)}^2 = \int_I (u - u_h)'(u - u_h)' dx \quad (1.86)$$

$$= \int_I (u - u_h)'(u - v + v - u_h)' dx \quad (1.87)$$

$$= \int_I (u - u_h)'(u - v)' dx + \int_I (u - u_h)'(v - u_h)' dx \quad (1.88)$$

$$= \int_I (u - u_h)'(u - v)' dx \quad (1.89)$$

$$\leq \|(u - u_h)'\|_{L^2(I)} \|(u - v)'\|_{L^2(I)}. \quad (1.90)$$

□

**Teorema 1.2.3.** (**Estimativa a priori.**) O erro em se aproximar a solução  $u$  de (??) pela solução de elementos finitos  $u_h$  dada por (??) satisfaz a seguinte estimativa *a priori*

$$\|(u - u_h)'\|_{L^2(I)}^2 \leq C \sum_{i=1}^n h_i^2 \|u''\|_{L^2(I_i)}^2. \quad (1.91)$$

*Demonstração.* Tomando  $v = \pi u$  no teorema da melhor aproximação (Teorema ??), obtemos

$$\|(u - u_h)'\|_{L^2(I)} \leq \|(u - \pi u)'\|_{L^2(I)}. \quad (1.92)$$

Daí, da estimativa do erro de interpolação (Proposição ??), temos

$$\|(u - u_h)'\|_{L^2(I)}^2 \leq C \sum_{i=1}^n h_i^2 \|u''\|_{L^2(I_i)}^2. \quad (1.93)$$

□

**Exemplo 1.2.2.** A Figura ?? apresenta o esboço da evolução do erro  $\|(u - u_h)'\|_{L^2(I)}$  da solução de elementos finitos do problema (??)-(??) para malhas uniformes com  $n = 2, 4, 8, \dots, 128$  células.

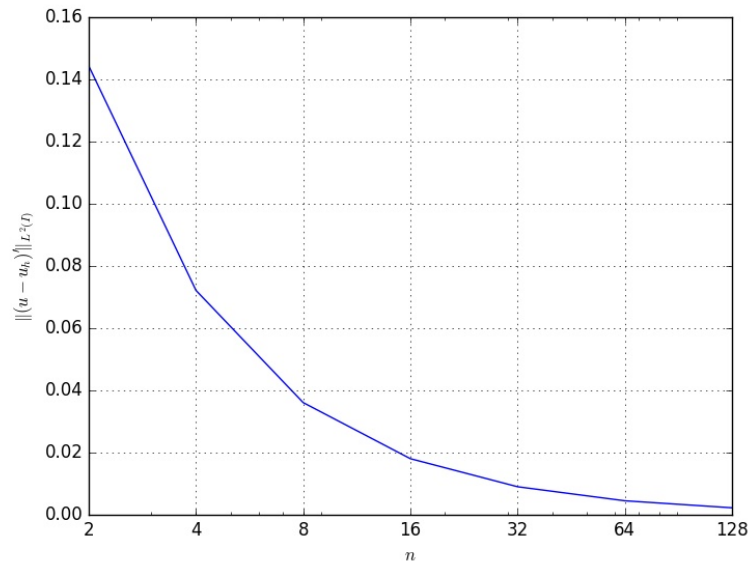


Figura 1.6: Esboço dos gráficos das soluções referentes ao Exemplo ??.

Com o [FEniCS](#), a computação do problema de elementos finitos pode ser feita com o seguinte [código](#):

```
from __future__ import print_function, division
from fenics import *
import numpy as np
import matplotlib.pyplot as plt

def boundary(x,on_boundary):
    return on_boundary

def solver(n):
    # malha
    mesh = IntervalMesh(n,0,1)

    # espaco
```

```

V = FunctionSpace(mesh, 'P', 1)

bc = DirichletBC(V, Constant(0.0), boundary)

#MEF problem
u = TrialFunction(V)
v = TestFunction(V)
f = Constant(1.0)
a = u.dx(0)*v.dx(0)*dx
L = f*v*dx

#computa a sol
u = Function(V)
solve(a == L, u, bc)

return u, mesh

#sol analitica
ua = Expression('-x[0]*x[0]/2+x[0]/2',
                degree=2)

lerrors=[]
for n in [2,4,8,16,32,64,128]:
    u, mesh = solver(n)
    e = errornorm(u,ua,norm_type='H10',mesh=mesh)
    lerrors.append(e)

plt.plot([2,4,8,16,32,64,128],lerrors)
plt.xscale('log',base=2)
#plt.yscale('log',base=2)
plt.xlabel(r"$n$")
plt.ylabel(r"$||\!(u-u_h)||\!\!_{L^2(I)}$")
plt.xlim((2,128))
plt.xticks([2,4,8,16,32,64,128],[2,4,8,16,32,64,128])
plt.grid('on')
plt.show()

```



### 1.2.4 Estimativa *a Posteriori*

[[tag:revisar]]

Aqui, vamos obter uma estimativa *a posteriori* para o erro  $e = u - u_h$  da solução de elementos finitos  $u_h$  do problema (??)-(??).

**Teorema 1.2.4.** *A solução de elementos finitos  $u_h$  satisfaz*

$$\|(u - u_h)'\|_{L^2(I)}^2 \leq C \sum_{i=1}^n \eta_i^2(u_h), \quad (1.94)$$

onde  $\eta_i(u_h)$  é chamado de elemento residual e é dado por

$$\eta_i(u_h) = h_i \|f - u_h''\|_{L^2(I_i)}. \quad (1.95)$$

*Demonstração.* Tomando  $e = u - u_h$  e usando a ortogonalidade de Galerkin (Teorema ??) temos

$$\|e'\|_{L^2(I)}^2 = \int_I e'(e - \pi e)' dx = \sum_{i=1}^n \int_{I_i} e'(e - \pi e)' dx. \quad (1.96)$$

Então, aplicando integração por partes

$$\|e'\|_{L^2(I)}^2 = \sum_{i=1}^n \int_{I_i} (-e'')(e - \pi e) dx + [e'(e - \pi e)]_{x_{i-1}}^{x_i}. \quad (1.97)$$

Daí, observando que  $e - \pi e = 0$  nos extremos dos intervalos  $I_i$  e que  $-e'' = -(u - u_h)'' = -u'' + u_h'' = f + u_h''$ , temos

$$\|e'\|_{L^2(I)}^2 = \sum_{i=1}^n \int_{I_i} (f + u_h'')(e - \pi e) dx. \quad (1.98)$$

Agora, usando as desigualdades de Cauchy-Schwarz e a estimativa padrão de interpolação (??), obtemos

$$\|e'\|_{L^2(I)}^2 \leq \sum_{i=1}^n \|f + u_h\|_{L^2(I_i)} \|e - \pi e\|_{L^2(I_i)} dx \quad (1.99)$$

$$\leq C \sum_{i=1}^n h_i \|f + u_h\|_{L^2(I_i)} \|e'\|_{L^2(I_i)} \quad (1.100)$$

$$\leq C \left( \sum_{i=1}^n h_i^2 \|f + u_h\|_{L^2(I_i)}^2 \right)^{1/2} \left( \sum_{i=1}^n \|e'\|_{L^2(I_i)}^2 \right)^{1/2} \quad (1.101)$$

$$= C \left( \sum_{i=1}^n h_i^2 \|f + u_h\|_{L^2(I_i)}^2 \right)^{1/2} \|e'\|_{L^2(I)}, \quad (1.102)$$

donde segue o resultado desejado.  $\square$

**Observação 1.2.2.** No caso da solução de elementos finitos no espaço das funções lineares por partes, temos  $u_h'' = 0$ . Logo, o elemento residual se resume em  $\eta_i(u_h) = h_i \|f\|_{L^2(I_i)}$ .

### 1.2.5 Exercícios

[[tag:revisar]]

**Exercício 1.2.1.** Obtenha uma aproximação por elementos finitos lineares por partes da solução de

$$-u'' + u = 2 \sin x, \quad \forall x \in (-\pi, \pi), \quad (1.103)$$

$$u(-\pi) = u(\pi) = 0. \quad (1.104)$$

## 1.3 Condições de Contorno

[[tag:revisar]]

Nesta seção, vamos discutir sobre soluções de elementos finitos para a equações diferencial

$$-u'' = f, \quad x \in I = [0, L], \quad (1.105)$$

com diferentes condições de contorno.

### 1.3.1 Condições de Dirichlet

[[tag:revisar]]

Consideramos o seguinte **problema com condições de contorno de Dirichlet**<sup>5</sup>: encontrar  $u$  tal que

$$-u'' = f, \quad \forall x \in I = [0, L], \quad (1.106)$$

$$u(0) = u_0, \quad u(L) = u_L, \quad (1.107)$$

com  $u_0$ ,  $u_L$  e  $f$  dados.

Tomando uma função teste  $v \in V_0 := H_0^1(I) := \{v \in H^1(I); v(0) = v(L) = 0\}$  e multiplicando-a em (??), obtemos

$$-\int_I u'' v \, dx = \int_I f v \, dx. \quad (1.108)$$

Aplicando a integração por partes, temos

$$\int_I u' v' \, dx = \int_I f v \, dx. \quad (1.109)$$

Desta forma, definimos o seguinte **problema fraco** associado: encontrar  $u \in V := \{v \in H^1(I); v(0) = u_0, v(L) = u_L\}$  tal que

$$a(u, v) = L(v), \quad \forall v \in V_0, \quad (1.110)$$

onde  $a(u, v)$  é a forma bilinear

$$a(u, v) = \int_I u' v' \, dx \quad (1.111)$$

e  $L(v)$  é a forma linear

$$L(v) = \int_I f v \, dx. \quad (1.112)$$

**Exemplo 1.3.1.** Consideramos o problema

$$-u'' = 1, \quad x \in I = [0, 1], \quad (1.113)$$

$$u(0) = 1/2, \quad u(1) = 1. \quad (1.114)$$

Sua solução analítica é  $u(x) = -x^2/2 + x + 1/2$ .

<sup>5</sup>Johann Peter Gustav Lejeune Dirichlet, 1805 - 1859, matemático alemão. Fonte: [Wikipedia](#).

Para obtermos uma aproximação de elementos finitos, consideramos o seguinte problema fraco: encontrar  $u \in V := \{v \in H^1(I); v(0) = 1/2, v(1) = 1\}$  tal que

$$a(u, v) = L(v), \quad (1.115)$$

para todo  $v \in V_0 = \{v \in H^1(I); v(0) = v(1) = 0\}$ , onde

$$a(u, v) = \int_I u' v' dx, \quad (1.116)$$

$$L(v) = \int_I f v dx. \quad (1.117)$$

Então, o problema de elementos finitos no espaço das funções lineares por partes lê-se: encontrar  $u_h \in V_h = \{v \in P_1(I); v(0) = 1/2, v(1) = 1\}$  tal que

$$a(u_h, v_h) = L(v_h), \quad (1.118)$$

para todo  $v_h \in V_{h,0} = \{v \in H^1(I); v(0) = v(1) = 0\}$ .

Código 1.4: ex\_mef1d\_dirichlet.py

```

1 from mpi4py import MPI
2
3 # malha
4 from dolfinx import mesh
5 domain = mesh.create_unit_interval(MPI.COMM_WORLD,
6                                     nx = 5)
7 # espaço
8 from dolfinx import fem
9 V = fem.functionspace(domain, ('P', 1))
10
11 # condição de contorno
12 import numpy as np
13 uD = fem.Function(V)
14 def dirichlet_bc(x):
15     y = np.full(x.shape[1], 0.5)
16     y[x[0,:] > 0.5] = 1.
17     return y
18 uD.interpolate(dirichlet_bc)
19
20 tdim = domain.topology.dim

```

```

21 fdim = tdim - 1
22 domain.topology.create_connectivity(fdim, tdim)
23 boundary_facets = mesh.exterior_facet_indices(domain.topology)
24 boundary_dofs = fem.locate_dofs_topological(V, fdim,
25                                           boundary_facets)
26 bc = fem.dirichletbc(uD, boundary_dofs)
27
28 # problema mef
29 import ufl
30 from dolfinx import default_scalar_type
31 from dolfinx.fem.petsc import LinearProblem
32 u = ufl.TrialFunction(V)
33 v = ufl.TestFunction(V)
34
35 f = fem.Constant(domain, default_scalar_type(1.))
36
37 a = ufl.dot(ufl.grad(u), ufl.grad(v)) * ufl.dx
38 L = f * v * ufl.dx
39
40 problem = LinearProblem(a, L, bcs=[bc])
41 uh = problem.solve()
42
43 # armazena para visualização (paraview)
44 from dolfinx import io
45 from pathlib import Path
46 results_folder = Path("results")
47 results_folder.mkdir(exist_ok=True, parents=True)
48 filename = results_folder / "u"
49 with io.XDMFFile(domain.comm, filename.with_suffix(".xdmf"), "w") as xdmf:
50     xdmf.write_mesh(domain)
51     xdmf.write_function(uh)

```

### 1.3.2 Condições de Neumann

[[tag:revisar]]

Consideramos o seguinte problema com condições de contorno de Neu-

mann<sup>6</sup> homogênea em  $x = L$ : encontrar  $u$  tal que

$$-u'' = f, \quad \forall x \in I = [0, L], \quad (1.119)$$

$$u(0) = u_0, \quad u'(L) = 0, \quad (1.120)$$

com  $u_0$  e  $f$  dados. Trata-se de um problema com condição de contorno de Dirichlet à esquerda e condição de contorno de Neumann<sup>7</sup> homogênea à direita.

Tomando uma função teste  $v \in V := \{v \in H^1(I); v(0) = 0\}$  e multiplicando-a em (??), obtemos

$$-\int_I u'' v \, dx = \int_I f v \, dx. \quad (1.121)$$

Aplicando a integração por partes, temos

$$\int_I u' v' \, dx - \underbrace{u'(L)v(L)}_{u'(L)=0} + \underbrace{u'(0)v(0)}_{v(0)=0} = \int_I f v \, dx. \quad (1.122)$$

Desta forma, definimos o seguinte problema fraco associado: encontrar  $u \in \tilde{V} := \{v \in H^1(I); v(0) = u_0\}$  tal que

$$a(u, v) = L(v), \quad \forall v \in V, \quad (1.123)$$

onde  $a(u, v)$  é a forma bilinear

$$a(u, v) = \int_I u' v' \, dx \quad (1.124)$$

e  $L(v)$  é a forma linear

$$L(v) = \int_I f v \, dx. \quad (1.125)$$

**Exemplo 1.3.2.** Consideramos o problema

$$-u'' = 1, \quad x \in I = [0, 1], \quad (1.126)$$

$$u(0) = 0, \quad u'(1) = 0. \quad (1.127)$$

<sup>6</sup>Carl Gottfried Neumann, 1832 - 1925, matemático alemão. Fonte: [Wikipédia](#).

<sup>7</sup>Carl Gottfried Neumann, 1832 - 1925, matemático alemão. Fonte: [Wikipédia](#).

Sua solução analítica é  $u(x) = -x^2/2 + x$ .

Podemos construir uma aproximação por elementos finitos do seguinte problema fraco associado: encontrar  $u \in V = \{v \in H^1(I); v(0) = 0\}$  tal que

$$a(u, v) = L(v), \quad (1.128)$$

para todo  $v \in V$ , com as formas bilinear  $a(\cdot, \cdot)$  e linear  $L(\cdot)$  dadas em (??) e (??).

Então, considerando elementos lineares por partes, temos o seguinte problema de elementos finitos: encontrar  $u_h \in V_h = \{v_h \in P_1(I); v_h(0) = 0\}$  tal que

$$a(u_h, v_h) = L(v_h), \quad \forall v_h \in V_h. \quad (1.129)$$

Código 1.5: ex\_mef1d\_neumann.py

```

1 from mpi4py import MPI
2
3 # malha
4 from dolfinx import mesh
5 domain = mesh.create_unit_interval(MPI.COMM_WORLD,
6                                     nx = 5)
7 # espaço
8 from dolfinx import fem
9 V = fem.functionspace(domain, ('P', 1))
10
11 # c.c. dirichlet
12 import numpy as np
13 from dolfinx.fem import dirichletbc, locate_dofs_geometrical
14 uD = fem.Function(V)
15 uD.interpolate(lambda x: np.full(x.shape[1], 0.))
16
17 def boundary_D(x):
18     return np.isclose(x[0], 0.)
19
20 dofs_D = locate_dofs_geometrical(V, boundary_D)
21 bc = dirichletbc(uD, dofs_D)
22

```

```

23 # problema mef
24 import ufl
25 from dolfinx import default_scalar_type
26 from dolfinx.fem.petsc import LinearProblem
27 u = ufl.TrialFunction(V)
28 v = ufl.TestFunction(V)
29
30 f = fem.Constant(domain, default_scalar_type(1.))
31
32 a = ufl.dot(ufl.grad(u), ufl.grad(v)) * ufl.dx
33 L = f * v * ufl.dx
34
35 problem = LinearProblem(a, L, bcs=[bc])
36 uh = problem.solve()
37
38 # armazena para visualização (paraview)
39 from dolfinx import io
40 from pathlib import Path
41 results_folder = Path("results")
42 results_folder.mkdir(exist_ok=True, parents=True)
43 filename = results_folder / "u"
44 with io.XDMFFile(domain.comm, filename.with_suffix(".xdmf"), "w") as xdmf:
45     xdmf.write_mesh(domain)
46     xdmf.write_function(uh)

```

Agora, consideramos o seguinte problema com condições de Neumann não-homogênea em  $x = L$ : encontrar  $u$  tal que

$$-u'' = f, \quad \forall x \in I = [0, L], \quad (1.130)$$

$$u(0) = u_0, \quad u'(L) = \alpha, \quad (1.131)$$

com  $u_0$ ,  $\alpha$  e  $f$  dados.

Tomando uma função teste  $v \in V := \{v \in H^1(I); v(0) = 0\}$  e multiplicando-a em (??), obtemos

$$-\int_I u'' v \, dx = \int_I f v \, dx. \quad (1.132)$$

Aplicando a integração por partes, temos

$$\int_I u' v' \, dx - \alpha v(L) = \int_I f v \, dx. \quad (1.133)$$



Desta forma, definimos o seguinte problema fraco associado: encontrar  $u \in \tilde{V} := \{v \in H^1(I); v(0) = u_0\}$  tal que

$$a(u, v) - b(v) = L(v), \quad \forall v \in V, \quad (1.134)$$

onde  $a(u, v)$  é a forma bilinear

$$a(u, v) = \int_I u' v' dx \quad (1.135)$$

e  $L(v)$  é a forma linear

$$L(v) = \int_I f v dx + \alpha v(L). \quad (1.136)$$

**Exemplo 1.3.3.** Consideramos o problema

$$-u'' = 1, \quad x \in I = [0, 1], \quad (1.137)$$

$$u(0) = 0, \quad u'(1) = 1. \quad (1.138)$$

Sua solução analítica é  $u(x) = -x^2/2 + 2x$ .

Agora, consideramos o seguinte problema fraco associado: encontrar  $u \in V = \{v \in H^1(I); v(0) = 0\}$  tal que

$$a(u, v) = L(v), \quad \forall v \in V, \quad (1.139)$$

com

$$a(u, v) = \int_I u' v' dx \quad (1.140)$$

e

$$L(v) = \int_I f v dx + 1 \cdot v(1). \quad (1.141)$$

Então, consideramos o seguinte problema de elementos finitos associado: encontrar  $u_h \in V_h = \{v_h \in P_1(I); v_h(0) = 0\}$  tal que

$$a(u_h, v_h) = L(v_h), \quad \forall v_h \in V_h. \quad (1.142)$$

Código 1.6: ex\_mefld\_neumann\_nh.py

```
1 from mpi4py import MPI
2
3 # malha
4 from dolfinx import mesh
5 domain = mesh.create_unit_interval(MPI.COMM_WORLD,
6                                     nx = 5)
7 # espaço
8 from dolfinx import fem
9 V = fem.functionspace(domain, ('P', 1))
10
11 # c.c. dirichlet
12 import numpy as np
13 from dolfinx.fem import dirichletbc, locate_dofs_geometrical
14 uD = fem.Function(V)
15 uD.interpolate(lambda x: np.full(x.shape[1], 0.))
16
17 def boundary_D(x):
18     return np.isclose(x[0], 0.)
19
20 dofs_D = locate_dofs_geometrical(V, boundary_D)
21 bc = dirichletbc(uD, dofs_D)
22
23 # problema mef
24 import ufl
25 from dolfinx import default_scalar_type
26 from dolfinx.fem.petsc import LinearProblem
27 u = ufl.TrialFunction(V)
28 v = ufl.TestFunction(V)
29
30 f = fem.Constant(domain, default_scalar_type(1.))
31 g = fem.Constant(domain, default_scalar_type(1.))
32
33 a = ufl.dot(ufl.grad(u), ufl.grad(v)) * ufl.dx
34 L = f * v * ufl.dx
35 L += g * v * ufl.ds
36
37 problem = LinearProblem(a, L, bcs=[bc])
38 uh = problem.solve()
39
```

```

40 # armazena para visualização (paraview)
41 from dolfinx import io
42 from pathlib import Path
43 results_folder = Path("results")
44 results_folder.mkdir(exist_ok=True, parents=True)
45 filename = results_folder / "u"
46 with io.XDMFFile(domain.comm, filename.with_suffix(".xdmf"), "w") as xdmf:
47     xdmf.write_mesh(domain)
48     xdmf.write_function(uh)

```

### 1.3.3 Condições de Robin

[[tag:revisar]]

Consideramos o seguinte problema com condições de contorno de Robin<sup>8</sup>: encontrar  $u$  tal que

$$-u'' = f, \quad \forall x \in I = [0, L], \quad (1.143)$$

$$u'(0) = r_0(u(0) - s_0), \quad -u'(L) = r_L(u(L) - s_L), \quad (1.144)$$

com  $r_0, r_L, s_0, s_L$  e  $f$  dados.

Tomando uma função teste  $v \in V = H^1(I)$  e multiplicando-a em (??), obtemos

$$-\int_I u'' v \, dx = \int_I f v \, dx. \quad (1.145)$$

Aplicando a integração por partes, temos

$$\int_I u' v' \, dx - \underbrace{u'(L)v(L)}_{-u'(L)=r_L(u(L)-s_L)} + \underbrace{u'(0)v(0)}_{u'(0)=r_0(u(0)-s_0)} = \int_I f v \, dx. \quad (1.146)$$

ou, mais adequadamente,

$$\int_I u' v' \, dx + r_L u(L)v(L) + r_0 u(0)v(0) = \int_I f v \, dx + r_L s_L v(L) + r_0 s_0 v(0). \quad (1.147)$$

Desta forma, definimos o seguinte problema fraco associado: encontrar  $u \in H^1(I)$  tal que

$$a(u, v) = L(v), \quad \forall v \in V, \quad (1.148)$$

<sup>8</sup>Victor Gustave Robin, 1855 - 1897, matemático francês. Fonte: [Wikipedia](#).

onde  $a(u, v)$  é a forma bilinear

$$a(u, v) = \int_I u'v' dx + r_L u(L)v(L) + r_0 u(0)v(0) \quad (1.149)$$

e  $L(v)$  é a forma linear

$$L(v) = \int_I f v dx + r_L s_L v(L) + r_0 s_0 v(0). \quad (1.150)$$

**Exemplo 1.3.4.** Consideramos o problema

$$-u'' = 1, \quad x \in I = [0, 1], \quad (1.151)$$

$$u'(0) = u(0), \quad -u'(1) = u(1) - 1. \quad (1.152)$$

Sua solução analítica é  $u(x) = -x^2/2 + 5x/6 + 5/6$ .

Aqui, tomamos o seguinte problema fraco: encontrar  $u \in V = H^1(I)$  tal que

$$a(u, v) = L(v), \quad \forall v \in V, \quad (1.153)$$

onde

$$a(u, v) = \int_I u'v' dx + u(1)v(1) + u(0)v(0) \quad (1.154)$$

e

$$L(v) = \int_I f v dx + 1 \cdot v(1). \quad (1.155)$$

Então, uma aproximação por elementos finitos lineares por partes pode ser obtida resolvendo o seguinte problema: encontrar  $u_h \in V_h = P_1(I)$  tal que

$$a(u_h, v_h) = L(v_h), \quad \forall v_h \in V_h. \quad (1.156)$$

```

1 from mpi4py import MPI
2
3 # malha
4 from dolfinx import mesh
5 domain = mesh.create_unit_interval(MPI.COMM_WORLD,
6                                     nx = 5)

```

```
7 # espaço
8 from dolfinx import fem
9 V = fem.functionspace(domain, ('P', 1))
10
11 # boundary colors
12 from dolfinx.mesh import locate_entities
13 from dolfinx.mesh import meshtags
14 boundaries = [(0, lambda x: np.isclose(x[0], 0.)),
15               (1, lambda x: np.isclose(x[0], 1.))]
16 facet_indices, facet_markers = [], []
17 fdim = domain.topology.dim - 1
18 for (marker, locator) in boundaries:
19     facets = locate_entities(domain, fdim, locator)
20     facet_indices.append(facets)
21     facet_markers.append(np.full_like(facets, marker))
22 facet_indices = np.hstack(facet_indices).astype(np.int32)
23 facet_markers = np.hstack(facet_markers).astype(np.int32)
24 sorted_facets = np.argsort(facet_indices)
25 facet_tag = meshtags(domain, fdim, facet_indices[sorted_facets], facet_markers)
26
27 # problema mef
28 import ufl
29 from dolfinx import default_scalar_type
30 from dolfinx.fem.petsc import LinearProblem
31 u = ufl.TrialFunction(V)
32 v = ufl.TestFunction(V)
33
34 f = fem.Constant(domain, default_scalar_type(1.))
35 g = fem.Constant(domain, default_scalar_type(1.))
36
37 ds = ufl.Measure('ds', domain=domain, subdomain_data=facet_tag)
38
39 a = ufl.dot(ufl.grad(u), ufl.grad(v)) * ufl.dx
40 a += u * v * ds(1) + u * v * ds(0)
41 L = f * v * ufl.dx
42 L += g * v * ds(1)
43
44 problem = LinearProblem(a, L, bcs=[])
45 uh = problem.solve()
46
```

```

47 # armazena para visualização (paraview)
48 from dolfinx import io
49 from pathlib import Path
50 results_folder = Path("results")
51 results_folder.mkdir(exist_ok=True, parents=True)
52 filename = results_folder / "u"
53 with io.XDMFFile(domain.comm, filename.with_suffix(".xdmf"), "w") as xdmf:
54     xdmf.write_mesh(domain)
55     xdmf.write_function(uh)

```

### 1.3.4 Exercícios

[[tag:revisar]]

**Exercício 1.3.1.** Considere o problema

$$-u'' + u' + 2u = -\cos(x), \quad x \in (0, \pi/2), \quad (1.157)$$

$$u(0) = -0,3, \quad u(\pi/2) = -0,1. \quad (1.158)$$

Obtenha uma aproximação por elementos finitos para a solução deste problema, empregando o espaço de elementos finitos linear sobre uma malha uniforme com 10 células. Então, compare a aproximação computada com sua solução analítica  $u(x) = 0,1(\sin(x) + 3\cos(x))$ , bem como, compute o erro  $\|u - u_h\|_{L^2}$ .

## 1.4 Malhas Auto-Adaptativas

[[tag:revisar]]

Retornemos ao problema modelo (??)-(??)

$$-u'' = f, \quad x \in I = [0, L], \quad (1.159)$$

$$u(0) = u(L) = 0. \quad (1.160)$$

A estimativa *a posteriori* dada no Teorema ??, indica que os elementos residuais  $\eta_i(u_h)$  podem ser utilizados para estimarmos a precisão da aproximação por elementos finitos. Ou seja, espera-se que quanto menores forem os elementos residuais, mais precisa é a solução por elementos finitos. Além disso,

como

$$\eta_i(u_h) = h_i \|f - u_h''\|_{L^2(I_i)}, \quad (1.161)$$

podemos reduzir  $\eta_i(u_h)$  diminuindo o tamanho da célula  $I_i$ .

Do observado acima, motiva-se o seguinte algoritmo de elementos finitos com refinamento automático de malha:

1. Escolhemos uma malha inicial.
2. Iteramos:
  2. Resolvemos o problema de elementos finitos na malha corrente.
  2. Computamos  $\eta_i(u_h)$  em cada célula da malha corrente.
  2. Com base na malha corrente, Contruímos uma nova malha pelo refinamento das células com os maiores valores de  $\eta_i(u_h)$ .
  2. Verificamos o critério de parada.

Uma estratégia clássica para a escolha das células a serem refinadas é a seguinte: refina-se a  $i$ -ésima célula se

$$\eta_i(u_h) > \alpha \max_{j=1,2,\dots,n} \eta_j(u_h), \quad (1.162)$$

onde escolhemos  $0 < \alpha < 1$ .

**Exemplo 1.4.1.** Consideramos o problema

$$-u'' = e^{-100|x-\frac{1}{2}|}, \quad x \in I = [0,1], \quad (1.163)$$

$$u(0) = u(1) = 0. \quad (1.164)$$

Aqui, computamos aproximações de elementos finitos no espaço das funções lineares por partes  $V_{h,0} = \{v \in P_1(I); v(0) = v(1) = 0\}$  com sucessivos refinamentos de malha. Utilizamos uma malha inicial uniforme com 10 células e fazemos, então, 5 refinamentos sucessivos utilizando como critério de refinamento a estratégia (??) com  $\alpha = 0,5$ . A Figura ?? apresenta o esboço do gráfico da solução de elementos finitos na malha mais refinada. Além disso, na Tabela ?? temos os o número de células e o  $\eta_i(u_h)$  máximo respectivo.

Com o [FEniCS](#), a computação do problema de elementos finitos pode ser feita com o seguinte [código](#):

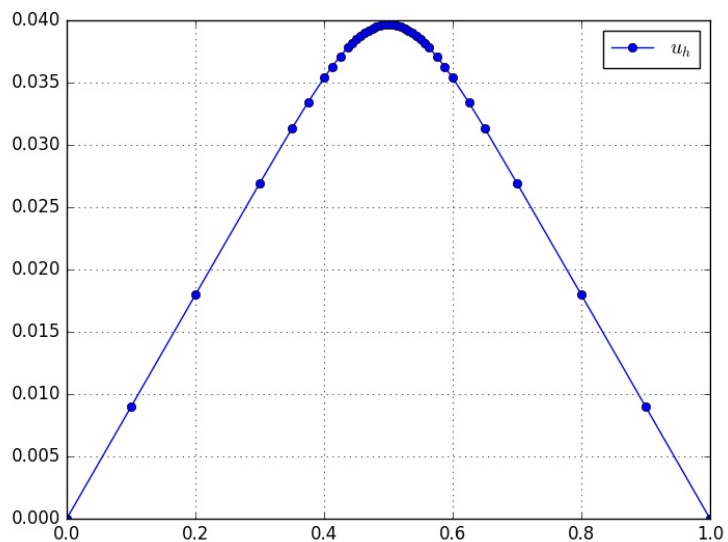


Figura 1.7: Esboço dos gráficos das soluções referentes ao Exemplo ??.

#malha	#células	$\max_i \eta_i(u_h)$
0	10	5.0E-03
1	12	2.0E-03
2	14	8.6E-04
3	22	2.9E-04
4	30	1.4E-04
5	38	6.1E-05

Tabela 1.1: Resultados referente ao Exemplo ??.

```
from __future__ import print_function, division
from fenics import *
import numpy as np
import matplotlib.pyplot as plt
```

```
# malha
mesh = IntervalMesh(10,0,1)
```

```
# espaco
```



```
V = FunctionSpace(mesh, 'P', 1)

# fonte
f = Expression('exp(-100*pow(fabs(x[0]-0.5),2))',degree=1)

# condicoes de contorno
def boundary(x,on_boundary):
    return on_boundary

#iteracoes
for iter in np.arange(6):

    #problema
    bc = DirichletBC(V,Constant(0.0),boundary)
    u = TrialFunction(V)
    v = TestFunction(V)
    a = u.dx(0)*v.dx(0)*dx
    L = f*v*dx

    #resolve
    u = Function(V)
    solve(a == L, u, bc)

    #grafico
    plt.close('all')
    xx = mesh.coordinates()[:,0]
    sorted_indices = np.argsort(xx)
    yy = u.compute_vertex_values()
    plt.plot(xx[sorted_indices],yy[sorted_indices],
             marker="o",label=r"$u_h$")
    plt.legend(numpoints=1)
    plt.grid('on')
    plt.show()

    DG = FunctionSpace(mesh, "DG", 0)
    v = TestFunction(DG)
    a = CellVolume(mesh)
    eta = assemble(f**2*v*a*dx)
```

```

# refinamento da malha
cell_markers = MeshFunction("bool", mesh, mesh.topology().dim(), False)
eta_max = np.amax(eta[:])
print(eta_max)
print("%d %d %1.1E\n" % (iter, mesh.num_cells(), eta_max))
alpha = 0.5
for i, cell in enumerate(cells(mesh)):
    if (eta[i] > alpha*eta_max):
        cell_markers[cell] = True

mesh = refine(mesh, cell_markers)
V = FunctionSpace(mesh, 'P', 1)

```

### 1.4.1 Exercícios

[[tag:revisar]]

**Exercício 1.4.1.** Use uma estratégia de sucessivos refinamentos globais para resolver o problema dado no Exemplo ???. Compare seus resultados com aqueles obtidos no exemplo.

## 1.5 Aplicação: Equação do Calor

[[tag:construcao]]

Consideramos a **equação do calor** com dadas condição inicial e condições de contorno de Dirichlet homogêneas

$$u_t = \alpha u_{xx} + f, \quad (t, x) \in (0, t_f] \times (a, b), \quad (1.165a)$$

$$u(0, x) = u_0(x), \quad x \in [a, b], \quad (1.165b)$$

$$u(t, a) = u(t, b) = 0, \quad t \in [0, t_f], \quad (1.165c)$$

onde  $f = f(t, x)$  denota uma dada fonte.

### 1.5.1 Discretização do Tempo

Consideramos os  $n_t + 1$  tempos discretos  $t^{(k)} = kh_t$ , passo no tempo  $h_t = t_f/n_t$ ,  $k = 0, 1, 2, \dots, n_t$ . Seguindo esquema  $\theta$  denotando  $u^{(k)} \approx u(t^{(k)}, x)$  e  $f^{(k)} = f(t^{(k)}, x)$ , o problema (??) pode ser aproximado pela iteração

$$\frac{u^{(k+1)} - u^{(k)}}{h_t} = \theta \left( \alpha u_{xx}^{(k+1)} + f^{(k+1)} \right) + (1 - \theta) \left( \alpha u_{xx}^{(k)} + f^{(k)} \right), \quad (1.166a)$$

$$u^{(k+1)}(a) = u^{(k+1)}(b) = 0, \quad (1.166b)$$

onde  $u^{(0)} = u_0$ .

**Observação 1.5.1.** (Esquema  $\theta$ .) O esquema  $\theta$  é uma forma robusta de escrever diferentes esquemas de discretização em uma única expressão:

- $\theta = 0$ .: Euler explícito.
- $\theta = 1$ .: Euler implícito.
- $\theta = 0.5$ : Crank-Nicolson.

Por simplificação da notação, vamos suprimir o super-índice  $k$ , denotando  $u^{(k+1)} := u$ ,  $u^{(k)} = u^0$  e similar para  $f^{(k)}$ . Com isso e rearranjando os termos, cada iteração (??) se resume ao seguinte problema de valores de contorno

$$-\alpha \theta u_{xx} + \frac{1}{h_t} u = \frac{1}{h_t} u^0 + (1 - \theta) \alpha u_{xx}^0 + (1 - \theta) f^0 + \theta f, \quad (1.167a)$$

$$u(a) = u(b) = 0. \quad (1.167b)$$

### 1.5.2 Formulação de Elementos Finitos

A formulação fraca do problema (??) consiste em: encontrar  $u \in V := H_0^1(a, b)$  tal que

$$a(u, v) = L(v), \quad \forall v \in V, \quad (1.168)$$

onde

$$a(u, v) := \int_a^b \theta \alpha u' v' dx + \int_a^b \frac{1}{h_t} uv dx, \quad (1.169)$$

$$L(v) := (1 - \theta) \int_a^b \alpha u^{0'} v' dx + \int_a^b \frac{1}{h_t} u^0 v dx + \theta \int_a^b f v dx + (1 - \theta) \int_a^b f^0 v dx \quad (1.170)$$

Então, assumindo uma malha com  $n_x$  células  $I_i = [x_i, x_{i+1}]$  de tamanho  $h_x = (b - a)/n_x$  e nodos  $x_i = a + (i - 1)h_x$ ,  $i = 0, 1, 2, \dots, n_x$ , escolhemos o espaço de elementos finitos

$$V_{h,0} := \{v \in C^0([a,b]) : v|_{I_i} \in P_1(I_i), i = 0, 1, \dots, n_x, v(a) = v(b) = 0\}. \quad (1.171)$$

Com isso, a formulação de elementos finitos do problema (??) consiste em: encontrar  $u_h \in V_{h,0}$  tal que

$$a(u_h, v_h) = L(v_h), \quad \forall v_h \in V_{h,0}. \quad (1.172)$$

**Exemplo 1.5.1.** Consideramos o seguinte problema de calor

$$u_t = u_{xx} + (\pi^2 - 1)e^{-t} \sin(\pi x), \quad (t, x) \in (0, 1] \times (0, 1), \quad (1.173a)$$

$$u(0, x) = \sin(\pi x), \quad x \in [0, 1], \quad (1.173b)$$

$$u(t, 0) = u(t, 1) = 0. \quad (1.173c)$$

```

1 from mpi4py import MPI
2 import ufl
3 from dolfinx import mesh
4 from dolfinx import fem
5 from dolfinx import default_scalar_type
6 from dolfinx.fem.petsc import LinearProblem
7
8 # parâmetros
9 tf = 1.
10 alpha = 1.
11
12 # esquema theta
13 theta = 0.5
14
15 # discretização no tempo
```

```
16 nt = 10
17 ht = tf/nt
18
19 # malha
20 domain = mesh.create_unit_interval(MPI.COMM_WORLD,
21                                     nx = 5)
22 x = ufl.SpatialCoordinate(domain)
23
24 # espaço
25 V = fem.functionspace(domain, ('P', 1))
26
27 # fonte
28 f = fem.Function(V)
29 def f(t,x):
30     return (ufl.pi**2-1.)*ufl.exp(-t)*ufl.sin(ufl.pi*x[0])
31
32 # condição de contorno
33 import numpy as np
34 uD = fem.Function(V)
35 uD.interpolate(lambda x: np.full(x.shape[1], 0.))
36
37 def boundary_D(x):
38     return np.logical_or(np.isclose(x[0], 0.),
39                           np.isclose(x[0], 1.))
40
41 dofs_D = fem.locate_dofs_geometrical(V, boundary_D)
42 bc = fem.dirichletbc(uD, dofs_D)
43
44 # mef fun.s
45 u = ufl.TrialFunction(V)
46 v = ufl.TestFunction(V)
47
48 # condição inicial
49 t = 0.
50 u0 = fem.Function(V)
51 u0.interpolate(lambda x: np.sin(np.pi*x[0]))
52
53 # fonte
54 def f(t, x):
55     return (ufl.pi**2-1.)*ufl.exp(-t)*ufl.sin(ufl.pi*x[0])
```

```

56
57 # visualização (paraview)
58 from dolfinx import io
59 from pathlib import Path
60 results_folder = Path("results")
61 results_folder.mkdir(exist_ok=True, parents=True)
62
63 # iteração no tempo
64 for k in range(nt):
65     t += ht
66
67     # forma bilinear
68     a = theta * ufl.dot(ufl.grad(u), ufl.grad(v)) * ufl.dx
69     a += u * v / ht * ufl.dx
70
71     # forma linear
72     L = (theta-1.) * ufl.dot(ufl.grad(u0), ufl.grad(v)) * ufl.dx
73     L += u0 * v / ht * ufl.dx
74     L += theta * f(t, x) * v * ufl.dx
75     L += (1.-theta) * f(t-ht, x) * v * ufl.dx
76
77     problem = LinearProblem(a, L, bcs=[bc])
78     uh = problem.solve()
79
80     # armazena para visualização (paraview)
81     filename = results_folder / f"u_{k:0>6}"
82     with io.XDMFFile(domain.comm, filename.with_suffix(".xdmf"), "w")
83         xdmf.write_mesh(domain)
84         xdmf.write_function(uh, t)
85
86     u0.x.array[:] = uh.x.array[:]

```

### 1.5.3 Exercícios

[[tag:construcao]]

## 1.6 Seleção de Aplicações

[[tag:revisar]]

Notas de Aula - Pedro Konzen \*/\* Licença CC-BY-SA 4.0

### 1.6.1 Sistemas de Equações

[[tag:revisar]]

Consideramos o seguinte problema de equações diferenciais ordinárias com valores de contorno

$$-u_0'' + u_1 = f_0, \forall x \in (0, L) \quad (1.174)$$

$$-u_1'' + u_0 = f_1, \forall x \in (0, L) \quad (1.175)$$

$$u_0(0) = u_{00}, \quad u_0(L) = u_{0L}, \quad (1.176)$$

$$u_1(0) = u_{10}, \quad u_1(L) = u_{1L}, \quad (1.177)$$

onde  $f_0, f_1, u_{00}, u_{0L}, u_{10}, u_{1L}$  são dados.

Para construirmos uma aproximação por elementos finitos podemos tomar o seguinte problema fraco associado: encontrar  $u = (u_0, u_1) \in V_0 \times V_1$  tal que

$$a(u, v) = L(v), \forall v = (v_0, v_1) \in V \times V, \quad (1.178)$$

onde  $V_0 = \{v \in H^1(I); v_0(0) = u_{00}, v_0(L) = u_{0L}\}$ ,  $V_1 = \{v_1 \in H^1(I); v_1(0) = u_{10}, v_1(L) = u_{1L}\}$ ,  $V = \{v \in H^1(I); v(0) = v(L) = 0\}$ , a forma bilinear é

$$a(u, v) = \int_I u_0' v_0' dx + \int_I u_1' v_1' dx + \int_I u_0 v_0 dx + \int_I u_1 v_1 dx \quad (1.179)$$

e a forma linear é

$$L(v) = \int_I f_0 v_0 dx + \int_I f_1 v_1 dx. \quad (1.180)$$

Então, o problema de elemento finitos associado no espaço das funções lineares por partes lê-se: encontrar  $u_h = (u_{h0}, u_{h1}) \in V_{h0} \times V_{h1}$  tal que

$$a(u_h, v_h) = L(v_h), \forall v_h = (v_{h0}, v_{h1}) \in V_h \times V_h, \quad (1.181)$$

onde  $V_{h0} = \{v_h \in P_1(I); v_{h0}(0) = u_{00}, v_{h0}(L) = u_{0L}\}$ ,  $V_{h1} = \{v_{h1} \in P_1(I); v_{h1}(0) = u_{10}, v_{h1}(L) = u_{1L}\}$ ,  $V_h = \{v_h \in P_1(I); v_h(0) = v_h(L) = 0\}$ .

**Exemplo 1.6.1.** Consideramos o seguinte problema de valor de contorno

$$-u_0'' + u_1 = \sin(x) + \cos(x), \forall x \in (-\pi, \pi) \quad (1.182)$$

$$-u_1'' + u_0 = \cos(x) - \sin(x), \forall x \in (-\pi, \pi) \quad (1.183)$$

$$u_0(-\pi) = 0, \quad u_0(\pi) = 0, \quad (1.184)$$

$$u_1(-\pi) = -1, \quad u_1(\pi) = -1. \quad (1.185)$$

Considerando elementos lineares por partes, temos a seguinte formulação de elementos finitos: encontrar  $u_h = (u_{h0}, u_{h1}) \in V_{h0} \times V_{h1}$  tal que

$$a(u_h, v_h) = L(v_h), \forall v_h = (v_{h0}, v_{h1}) \in V_h \times V_h, \quad (1.186)$$

onde  $V_{h0} = \{v_h \in P_1(I); v_{h0}(0) = v_{h0}(L) = 0\}$ ,  $V_{h1} = \{v_{h1} \in P_1(I); v_{h1}(0) = v_{h1}(L) = -1\}$ ,  $V_h = \{v_h \in P_1(I); v_h(0) = v_h(L) = 0\}$ , com as formas bilinear e linear são dadas em (??) e (??), respectivamente.

A Figura ?? apresenta o esboço dos gráficos das soluções analíticas  $u_0(x) = \sin(x)$  e  $u_1(x) = \cos(x)$  e de suas aproximações de elementos finitos  $u_{h0}$  e  $u_{h1}$ , estas construídas no espaço dos polinômios lineares por partes sobre uma malha uniforme de 5 células.

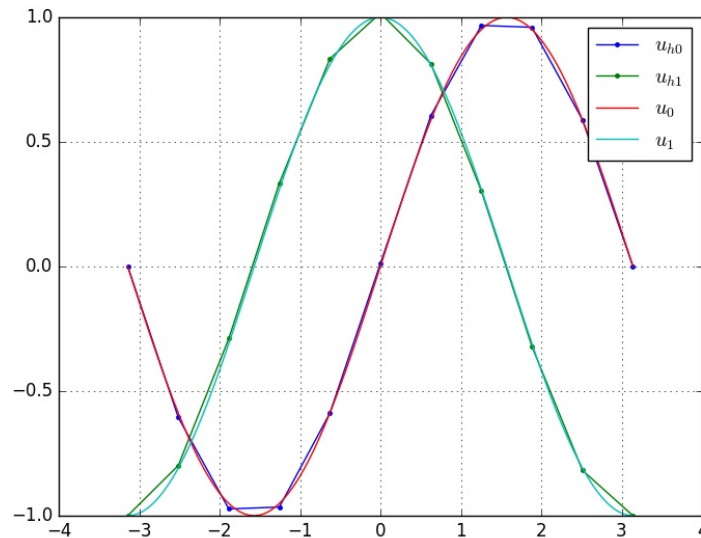


Figura 1.8: Esboço dos gráficos das soluções referentes ao Exemplo ??.

Com o [FEniCS](#), a computação do problema de elementos finitos pode ser feita com o seguinte código:

Notas de Aula - Pedro Konzen \*/\* Licença CC-BY-SA 4.0



```
from __future__ import print_function, division
from fenics import *
import numpy as np
import matplotlib.pyplot as plt

#tolerance
tol=1e-14

# malha
mesh = IntervalMesh(10,-pi,pi)

# espaco
P1 = FiniteElement('P',interval,1)
element = MixedElement([P1,P1])
V = FunctionSpace(mesh, element)

#C.C.
def boundary(x,on_boundary):
    return on_boundary

bc = [DirichletBC(V.sub(0),Constant(0.0),boundary),
      DirichletBC(V.sub(1),Constant(-1.0),boundary)]
print(bc)

#MEF problem
u = TrialFunction(V)
v = TestFunction(V)
f0 = Expression('sin(x[0]) + cos(x[0])',
                 degree=10)
f1 = Expression('cos(x[0]) - sin(x[0])',
                 degree=10)
a = u[0].dx(0)*v[0].dx(0)*dx
a += u[1]*v[0]*dx
a += u[1].dx(0)*v[1].dx(0)*dx
a -= u[0]*v[1]*dx
L = f0*v[0]*dx
L += f1*v[1]*dx
```

```
#computa a sol
u = Function(V)
solve(a == L, u, bc)

#sol analitica
u0a = Expression('sin(x[0])',
                  degree=10)
u1a = Expression('cos(x[0])',
                  degree=10)

plot(u[0],mesh=mesh,marker='.',label=r"$u_{h0}$")
plot(u[1],mesh=mesh,marker='.',label=r"$u_{h1}$")
mesh = IntervalMesh(100,-pi,pi)
plot(u0a,mesh=mesh,label=r"$u_0$")
plot(u1a,mesh=mesh,label=r"$u_1$")
plt.legend(numpoints=1)
plt.grid('on')
plt.show()
```

### 1.6.2 Exercícios

[[tag:construcao]]

## Capítulo 2

# Método de elementos finitos em 2D

### 2.1 Malha e espaço

[[tag:revisar]]

#### 2.1.1 Malha

[[tag:revisar]]

Seja  $\Omega \subset \mathbb{R}^2$  um domínio limitado com fronteira  $\partial\Omega$  suave e poligonal. Uma malha (ou triangularização)  $\mathcal{K}$  de  $\Omega$  é um conjunto de  $\{K\}$  células (ou elementos)  $K$ , tal que  $\Omega = \cup_{K \in \mathcal{K}} K$  e tal que a interseção de duas células é ou um lado, um canto ou vazio.

Classicamente as células  $K$  são escolhidas como triângulos. O comprimento do maior lado do triângulo  $K$  define o chamado **tamanho local da malha**  $h_K$ . O tamanho global da malha é definida por  $h = \max_{K \in \mathcal{K}} h_K$ .

Uma malha é dita **regular** quando existe uma constante  $c_0 > 0$  tal que  $c_K > c_0$  para todo  $K \in \mathcal{K}$ , sendo  $c_K := h_K/d_K$  e  $d_K$  o diâmetro do círculo inscrito em  $K$ . Esta condição significa que os triângulo  $K$  da malha não podem ter ângulos muito grandes nem muito pequenos. Ao longo do texto, a menos que especificado o contrário, assumiremos trabalhar com malhas regulares.

### 2.1.2 Espaço dos polinômios lineares por partes

[[tag:revisar]]

Seja  $K$  um triângulo e seja  $P_1(K)$  o espaço das funções lineares em  $K$ , i.e.

$$P_1(K) = \{v; v = c_0 + c_1x_0 + c_2x_1, (x_0, x_1) \in K, c_0, c_1, c_2 \in \mathbb{R}\}. \quad (2.1)$$

Observemos que toda função  $v \in P_1(K)$  é unicamente determinada por seus valores nodais  $\alpha_i = v(N_i)$ ,  $i = 0, 1, 2$ , onde  $N_i = (x_0^{(i)}, x_1^{(i)})$  é o  $i$ -ésimo nodo (vértice) do triângulo  $K$ . Isto segue do fato de que

$$\begin{bmatrix} 1 & x_0^{(0)} & x_1^{(0)} \\ 1 & x_0^{(1)} & x_1^{(1)} \\ 1 & x_0^{(2)} & x_1^{(2)} \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \end{bmatrix} \quad (2.2)$$

Computando o valor absoluto do determinante da matriz de coeficientes, obtemos  $2|K|$ , onde  $|K|$  denota a área de  $K$ , a qual é não nula.

Afim de usarmos os valores nodais como graus de liberdade (incógnitas), nós introduzimos a seguinte base nodal  $\{\lambda_0, \lambda_1, \lambda_2\}$  com

$$\lambda_j(N_i) = \begin{cases} 1 & , i = j, \\ 0 & , i \neq j \end{cases}, i, j = 0, 1, 2. \quad (2.3)$$

Com esta base, toda função  $v \in P_1(K)$  pode ser escrita como

$$v = \alpha_0\lambda_0 + \alpha_1\lambda_1 + \alpha_2\lambda_2, \quad (2.4)$$

onde  $\alpha_i = v(N_i)$ .

### 2.1.3 Espaço contínuo dos polinômios lineares por partes

[[tag:revisar]]

O espaço contínuo dos polinômios lineares por partes na malha  $\mathcal{K}$  é definido por

$$V_h = \{v; v \in C^0(\Omega), v|_K \in P_1(K), \forall K \in \mathcal{K}\}. \quad (2.5)$$

Observemos que toda função  $v \in V_h$  é unicamente determinada por seus valores nodais  $\{v(N_j)\}_{j=0}^{n_p-1}$ , onde  $n_p$  é número de nodos da malha  $\mathcal{K}$ . De fato, os valores nodais determinam uma única função em  $P_1(K)$  para cada  $K \in \mathcal{K}$  e, portanto, uma função em  $V_h$  é unicamente determinada por seus valores nos nodos. Agora, consideremos dois triângulos  $K_1$  e  $K_2$  compartilhando um lado  $E = K_1 \cap K_2$ . Sejam  $v_1$  e  $v_2$  os dois únicos polinômios em  $v_1 \in P_1(K_1)$  e  $v_2 \in P_1(K_2)$ , respectivamente determinados pelos valores nodais em  $K_1$  e  $K_2$ . Como  $v_1$  e  $v_2$  também são polinômios lineares em  $E$  e seus valores coincidem nos nodos de  $E$ , temos  $v_1 = v_2$ . Portanto, concluímos que toda função  $v \in V_h$  é unicamente determinada por seus valores nodais.

Afim de termos os valores nodais como graus de liberdade (incógnitas), definimos a base nodal  $\{\varphi_j\}_{j=1}^{n_p} \subset V_h$  tal que

$$\varphi_j(N_i) = \begin{cases} 1 & , i = j \\ 0 & , i \neq j \end{cases}, \quad i, j = 0, 1, \dots, n_p - 1. \quad (2.6)$$

Notemos que cada função base  $\varphi_j$  é contínua, linear por partes e com suporte somente em um pequeno conjunto de triângulos que compartilham o nodo  $N_j$ . Além disso, toda a função  $v \in V_h$  pode, então, ser escrita como

$$v = \sum_{i=0}^{n_p-1} \alpha_i \varphi_i, \quad (2.7)$$

onde  $\alpha_i = v(N_i)$ ,  $i = 0, 1, \dots, n_p$ , são os valores nodais de  $v$ .

**Exemplo 2.1.1.** A Figura ?? mostra o esboço de uma malha triangular no domínio  $D = [0, 1] \times [0, 1]$ .

Com o [FEniCS](#), podemos gerar esta malha com o seguinte código:

```
from __future__ import print_function, division
from fenics import *
import numpy as np
import matplotlib.pyplot as plt

# malha
Nx = 5
Ny = 5
mesh = UnitSquareMesh(Nx, Ny)
```

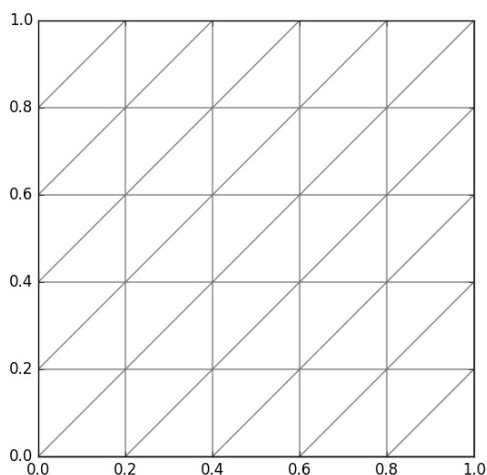


Figura 2.1: Esboço de uma malha triangular no domínio  $D = [0, 1] \times [0, 1]$ .

```
# espaço
V = FunctionSpace(mesh, 'P', 1)

# esboço da malha
plot(V.mesh())
plt.show()
```

## 2.2 Interpolação e projeção

[[tag:revisar]]

### 2.2.1 Interpolação

[[tag:revisar]]

Dada uma função contínua  $f$  em um triângulo  $K$  com nodos  $N_i$ ,  $i = 0, 1, 2$ , sua interpolação linear  $\pi f \in P_1(K)$  é definida por

$$\pi f = \sum_{i=0}^3 f(N_i) \varphi_i. \quad (2.8)$$

Logo, temos  $\pi f(N_i) = f(N_i)$  para todo  $i = 0, 1, 2$ .

Afim de determinarmos estimativas para o erro de interpolação, precisamos da chamada derivada total de primeira ordem

$$Df = \left( \left| \frac{\partial f}{\partial x_0} \right|^2 + \left| \frac{\partial f}{\partial x_1} \right|^2 \right)^{1/2}, \quad (2.9)$$

e da derivada total de segunda ordem

$$D^2 f = \left( \left| \frac{\partial^2 f}{\partial x_0^2} \right|^2 + \left| \frac{\partial^2 f}{\partial x_0 \partial x_1} \right|^2 + \left| \frac{\partial^2 f}{\partial x_1^2} \right|^2 \right)^{1/2}. \quad (2.10)$$

**Proposição 2.2.1.** (Erro da interpolação no espaço linear) A interpolação  $\pi f$  satisfaz as seguintes estimativas

$$\|f - \pi f\|_{L^2(K)} \leq Ch_K^2 \|D^2 f\|_{L^2(K)}, \quad (2.11)$$

$$\|D(f - \pi f)\|_{L^2(K)} \leq Ch_K \|D^2 f\|_{L^2(K)}. \quad (2.12)$$

*Demonstração.* Veja [?, Capítulo 4].  $\square$

**Observação 2.2.1.** A constante  $C$  depende do inverso de  $\sin(\theta_K)$  onde  $\theta_K$  é o menor ângulo de  $K$ . Desta forma, para um triângulo com  $\theta_K$  muito pequeno, as estimativas (??) e (??) perdem sentido. Este fato indica a necessidade de se trabalhar com malhas regulares.

A interpolação no espaço  $V_h$  de uma dada função  $f$  no domínio  $\Omega$  é denotada também por  $\pi f \in V_h$  e definida por

$$\pi f = \sum_{i=0}^{n_p-1} f(N_i) \varphi_i. \quad (2.13)$$

**Proposição 2.2.2.** (Erro da interpolação no espaço contínuo linear por partes) O interpolador  $\pi f \in V_h$  satisfaz as seguintes estimativas

$$\|f - \pi f\|_{L^2(\Omega)}^2 \leq C \sum_{K \in \mathcal{K}} h_K^4 \|D^2 f\|_{L^2(K)}^2, \quad (2.14)$$

$$\|D(f - \pi f)\|_{L^2(\Omega)}^2 \leq C \sum_{K \in \mathcal{K}} h_K^2 \|D^2 f\|_{L^2(K)}^2. \quad (2.15)$$

*Demonstração.* Demonstração análoga a Proposição ??.

□

**Exemplo 2.2.1.** Consideremos a função  $f(x_0, x_1) = \sin(\pi x_0) \cos(\pi x_1)$  definida no domínio  $D = [0, 1] \times [0, 1]$ . O seguinte código computa a interpolação de  $f$  no espaço  $V_h$  sobre uma malha triangular uniforme.

```
from __future__ import print_function, division
from fenics import *
import numpy as np
import matplotlib.pyplot as plt

# malha
Nx = 5
Ny = 5
mesh = UnitSquareMesh(Nx, Ny)

# espaco
V = FunctionSpace(mesh, 'P', 1)

# funcao
f = Expression('sin(pi*x[0])*cos(pi*x[1])', degree=3)

# interpolacao
pif = interpolate(f, V)

# exportando em vtk
vtkfile = File('pif.pvd')
vtkfile << pif
```

### 2.2.2 Projeção $L^2$

[[tag:revisar]]

A projeção  $L^2$  no espaço  $V_h$  de uma dada uma função  $f \in L^2(\Omega)$  é denotada por  $P_h f \in V_h$  e definida por

$$\int_{\Omega} (f - P_h f) v \, dx = 0, \quad \forall v \in V_h. \quad (2.16)$$

Analogamente a projeção em uma dimensão (veja Subseção ??), a projeção



$$P_h f = \sum_{j=0}^{n_p-1} \xi_j \varphi_j, \quad (2.17)$$

onde  $\xi_j$  satisfaz o sistema linear

$$M \boldsymbol{\xi} = \mathbf{b}, \quad (2.18)$$

onde  $M = [m_{i,j}]_{i,j=0}^{n_p-1}$  é a matriz de massa com

$$m_{i,j} = \int_{\Omega} \varphi_i \varphi_j \, dx \quad (2.19)$$

e  $\mathbf{b} = (b_1, b_2, \dots, b_{n_p-1})$  é o vetor de carga com

$$b_i = \int_{\Omega} f \varphi_i \, dx. \quad (2.20)$$

Também, vale o resultado análogo da melhor aproximação (veja ??), i.e.

$$\|f - P_h f\|_{L^2(\Omega)} \leq \|f - v\|_{L^2(\Omega)}, \quad \forall v \in V_h. \quad (2.21)$$

E, portanto, também temos a estimativa análoga para o erro de projeção (veja ??)

$$\|f - P_h f\|_{L^2(\Omega)}^2 \leq C \sum_{K \in \mathcal{K}} h_K^4 \|D^2 f\|_{L^2(K)}^2. \quad (2.22)$$

Tomando o tamanho global da malha, temos

$$\|f - P_h f\|_{L^2(\Omega)} \leq C h^2 \|D^2 f\|_{L^2(K)}. \quad (2.23)$$

**Exemplo 2.2.2.** Consideremos a função  $f(x_0, x_1) = \sin(\pi x_0) \cos(\pi x_1)$  definida no domínio  $D = [0, 1] \times [0, 1]$ . O seguinte código computa a projeção de  $f$  no espaço  $V_h$  sobre uma malha triangular uniforme.

```
from __future__ import print_function, division
from fenics import *
import numpy as np
import matplotlib.pyplot as plt
```

```

# malha
Nx = 5
Ny = 5
mesh = UnitSquareMesh(Nx,Ny)

# espaco
V = FunctionSpace(mesh, 'P', 1)

# funcao
f = Expression('sin(pi*x[0])*cos(pi*x[1])',degree=3)

# interpolacao
pif = project(f,V)

# exportando em vtk
vtkfile = File('pif.pvd')
vtkfile << pif

```

### 2.2.3 Exercícios

[[tag:revisar]]

**Exercício 2.2.1.** Verifique computacionalmente a Proposição ?? no caso da função  $f(x_0, x_1) = \sin(\pi x_0) \cos(\pi x_1)$  interpolada sobre uma malha triangular uniforme sobre o domínio  $D = [0, 1] \times [0, 1]$ .

**Exercício 2.2.2.** Verifique computacionalmente a estimativa (??) no caso da função  $f(x_0, x_1) = \sin(\pi x_0) \cos(\pi x_1)$  projetada sobre uma malha triangular uniforme sobre o domínio  $D = [0, 1] \times [0, 1]$ .

## 2.3 Problema modelo

[[tag:revisar]]

Nesta seção, apresentaremos a aplicação do método de elementos finitos

para a equação de Poisson<sup>1</sup> com condições de Dirichlet<sup>2</sup>, i.e.: encontrar  $u$  tal que

$$-\Delta u = f, \quad x \in \Omega, \quad (2.24)$$

$$u = 0, \quad x \in \partial\Omega, \quad (2.25)$$

onde  $\Delta = \partial^2/\partial x_0^2 + \partial^2/\partial x_1^2$  é o operador de Laplace<sup>3</sup> e  $f$  é uma função dada.

### 2.3.1 Formulação variacional

[[tag:revisar]]

A aplicação do método de elementos finitos é construída sobre a formulação fraca do problema (??)-(??). Para obtermos esta, multiplicamos (??) por uma função teste  $v$  em um espaço adequado  $V_0$  e integramos no domínio  $\Omega$ , i.e.

$$-\int_{\Omega} \Delta u v \, dx = \int_{\Omega} f v \, dx. \quad (2.26)$$

Então, usando a fórmula de Green<sup>4</sup>, obtemos

$$\int_{\Omega} \nabla u \cdot \nabla v \, dx - \int_{\partial\Omega} n \cdot \nabla u v \, ds. \quad (2.27)$$

Então, observando critérios de regularidade e a condição de contorno (??), escolhemos

$$V_0 := \{v \in H^1(\Omega) : v|_{\partial\Omega} = 0\}. \quad (2.28)$$

Lembramos que  $H^1(\Omega) = \{v : \|v\|_{L^2(\Omega)} + \|\nabla v\|_{L^2(\Omega)} < \infty\}$ .

Com isso, temos o seguinte problema fraco associado a (??)-(??): encontrar  $u \in V_0$  tal que

$$a(u, v) = L(v), \quad \forall v \in V_0, \quad (2.29)$$

<sup>1</sup>Siméon Denis Poisson, 1781 - 1840, matemático francês. Fonte: [Wikipedia](#).

<sup>2</sup>Johann Peter Gustav Lejeune Dirichlet, 1805 - 1859, matemático alemão. Fonte: [Wikipedia](#).

<sup>3</sup>Pierre-Simon, marquis de Laplace, 1749 - 1827, matemático francês. Fonte: [Wikipedia](#).

<sup>4</sup>George Green, 1793 - 1841, matemático britânico. Fonte: [Wikipedia](#).

onde  $a(u, v)$  é chamada de forma bilinear e definida por

$$a(u, v) := \int_{\Omega} \nabla u \cdot \nabla v \, dx \quad (2.30)$$

e  $L(v)$  é chamada de forma linear e definida por

$$L(v) := \int_{\Omega} f v \, dx. \quad (2.31)$$

### 2.3.2 Formulação de elementos finitos

[[tag:revisar]]

A formulação de elementos finitos é obtida da formulação fraca (??) pela aproximação do espaço teste  $V_0$  por um espaço de dimensão finita. Tomando uma triangulação  $\mathcal{K} \subset \Omega$  e considerando o espaço contínuo dos polinômios lineares por partes

$$V_h := \{v : v \in C^0(\Omega), v|_K \in P_1(K) \, \forall K \in \mathcal{K}\}, \quad (2.32)$$

assumimos também o subconjunto  $V_{h,0} := \{v \in V_h : v|_{\partial\Omega} = 0\}$ .

Com isso, temos o seguinte problema de elementos finitos associado (??): encontrar  $u_h \in V_{h,0}$  tal que

$$a(u_h, v_h) = L(v_h), \, \forall v_h \in V_{h,0}. \quad (2.33)$$

Observemos que (??) é equivalente ao problema de encontrar  $u_h \in V_{h,0}$  tal que

$$a(u_h, \varphi_i) = L(\varphi_i), \quad (2.34)$$

com  $i = 0, 1, \dots, n_p - 1$ , onde  $\{\varphi_i\}_{i=0}^{n_i-1}$  é a base nodal de  $V_{h,0}$  e  $n_i$  é o número de funções bases (igual ao número de nodos internos da triangulação  $\mathcal{K}$ ). Ainda, como

$$u_h = \sum_{j=0}^{n_i-1} \xi_j \varphi_j, \quad (2.35)$$

temos

$$a(u_h, \varphi_i) = a \left( \sum_{j=0}^{n_i-1} \xi_j \varphi_j, \varphi_i \right) \quad (2.36)$$

$$= \sum_{j=0}^{n_i-1} \xi_j a(\varphi_j, \varphi_i). \quad (2.37)$$

Com isso, o problema de elementos finitos é equivalente a resolver o seguinte sistema linear

$$\sum_{j=0}^{n_i-1} \xi_j a(\varphi_j, \varphi_i) = L(\varphi_i), \quad i = 0, 1, \dots, n_i - 1, \quad (2.38)$$

para as incógnitas  $\xi_j$ ,  $j = 0, 1, \dots, n_i - 1$ . Ou, equivalentemente, temos sua forma matricial

$$A\xi = \mathbf{b}, \quad (2.39)$$

onde  $A = [a_{i,j}]_{i,j=0}^{n_i-1}$  é chamada de **matriz de rigidez** com

$$a_{i,j} = a(\varphi_j, \varphi_i) \quad (2.40)$$

e  $\mathbf{b} = (b_0, b_1, \dots, b_{n_i-1})$  é o vetor de carga com

$$b_i = L(\varphi_i). \quad (2.41)$$

**Exemplo 2.3.1.** Consideremos o seguinte problema de Poisson

$$-\Delta u = 100x_0(1-x_0)x_1(1-x_1), \quad x \in \Omega := (0,1) \times (0,1), \quad (2.42)$$

$$u = 0, \quad x \in \partial\Omega. \quad (2.43)$$

Na Figura ?? temos um esboço da aproximação de elementos finitos obtida em uma malha uniforme com  $20 \times 20$  nodos. As isolinhas correspondem aos pontos tais que  $u = 3 \times 10^{-1}, 2 \times 10^{-1}, 10^{-1}, 5 \times 10^{-2}$ .

Com o **FEniCS**, podemos computar a solução deste problema com o seguinte código:

```
from __future__ import print_function, division
from fenics import *
import numpy as np
import matplotlib.pyplot as plt

# malha
```

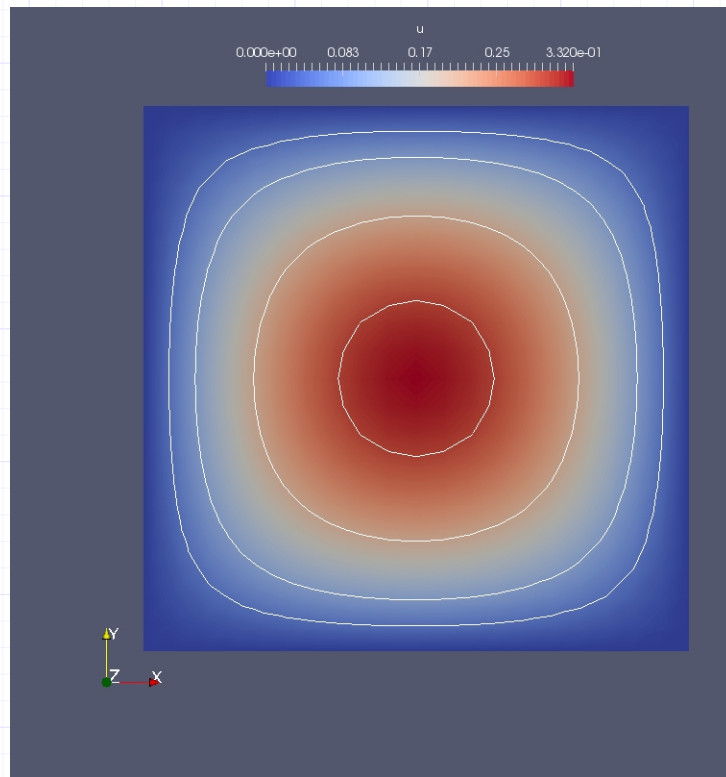


Figura 2.2: Esboço da solução de elementos finitos do problema discutido no Exemplo ??.

```

Nx = 20
Ny = 20
mesh = UnitSquareMesh(Nx,Ny)

# espaco
V = FunctionSpace(mesh, 'P', 1)

# cond. contorno
def boundary(x,on_boundary):
    return on_boundary

bc = DirichletBC(V,Constant(0.0),boundary)

```

```

# f
f = Expression('100*x[0]*(1-x[0])*x[1]*(1-x[1])', degree=4)

# MEF problem
u = TrialFunction(V)
v = TestFunction(V)
a = dot(grad(u), grad(v))*dx
L = f*v*dx

#computa a sol
u = Function(V)
solve(a == L, u, bc)

# exportando em vtk
vtkfile = File('u.pvd')
vtkfile << u

```

### 2.3.3 Exercícios

[[tag:revisar]]

**Exercício 2.3.1.** Compute uma aproximação de elementos finitos para o seguinte problema

$$-\Delta u = 10, x \in (0, 1) \times (0, 1) \quad (2.44)$$

$$u(x, 0) = 0, 0 \leq x \leq 1, \quad (2.45)$$

$$u(1, y) = 0, 0 \leq y < 1, \quad (2.46)$$

$$u(x, 1) = 1, 0 \leq x \leq 1, \quad (2.47)$$

$$u(0, y) = 1, 0 < x \leq 1. \quad (2.48)$$

## 2.4 Fundamentos da análise de elementos finitos

[[tag:revisar]]

### 2.4.1 Existência e unicidade

[[tag:revisar]]

**Teorema 2.4.1.** (Matriz positiva definida) A matriz de rigidez é positiva definida.

*Demonstração.* A matriz de rigidez  $A = [a(\varphi_j, \varphi_i)]_{i,j=0}^{n_i-1}$  é obviamente simétrica. Além disso, para todo  $\xi \in \mathbb{R}^{n_i}$ ,  $\xi \neq 0$ , temos

$$\xi^T A \xi = \sum_{i,j=0}^{n_i-1} \xi_j a(\varphi_j, \varphi_i) \xi_i \quad (2.49)$$

$$= \sum_{i,j=0}^{n_i-1} \xi_j \int_{\Omega} \nabla \varphi_j \cdot \nabla \varphi_i \, dx \, \xi_i \quad (2.50)$$

$$= \int_{\Omega} \nabla \left( \sum_{j=0}^{n_i-1} \xi_j \varphi_j \right) \cdot \nabla \left( \sum_{i=0}^{n_i-1} \xi_i \varphi_i \right) \, dx \quad (2.51)$$

$$= \left\| \nabla \left( \sum_{j=0}^{n_i-1} \xi_j \varphi_j \right) \right\|_{L^2(\Omega)}^2. \quad (2.52)$$

Portanto,  $\xi^T A \xi \geq 0$  e é nulo se, e somente se,  $v = \sum_{j=0}^{n_i-1} \xi_j \varphi_j$  for constante. Como  $v \in V_{h,0}$ , temos que  $v$  constante implica  $v \equiv 0$ , mas então  $\xi = 0$ , o que é uma contradição. Logo,  $\xi^T A \xi > 0$  para todo  $\xi \in \mathbb{R}^{n_i}$ ,  $\xi \neq 0$ .  $\square$

**Teorema 2.4.2.** (Existência e unicidade) O problema de elementos finitos (??) tem solução única.

*Demonstração.* O problema de elementos finitos (??) se resume a resolver o sistema linear  $A\xi = \mathbf{b}$ . Do Teorema ??, temos que  $A$  é uma matriz definida positiva e, portanto, invertível. Daí segue, imediatamente, que o problema (??) tem solução única.  $\square$

### 2.4.2 Estimativa a priori do erro

[[tag:revisar]]



**Teorema 2.4.3.** (Ortogonalidade de Galerkin) A solução  $u_h$  do problema de elementos finitos (??) satisfaz

$$a(u - u_h, v_h) = 0, \forall v_h \in V_{h,0}, \quad (2.53)$$

onde  $u$  é a solução do problema fraco (??).

*Demonstração.* Segue, imediatamente, do fato de que  $V_{h,0} \subset V_0$  e, portanto,

$$a(u, v_h) = L(v_h), \forall v_h \in V_{h,0}, \quad (2.54)$$

bem como

$$a(u_h, v_h) = L(v_h), \forall v_h \in V_{h,0}. \quad (2.55)$$

□

**Definição 2.4.1.** (Norma da energia.) Definimos a norma da energia por

$$|||v||| := \left( \int_{\Omega} \nabla v \cdot \nabla v \, dx \right)^{1/2} = \|\nabla v\|_{L^2(\Omega)}, \quad (2.56)$$

para todo  $v \in V_0$ .

**Teorema 2.4.4.** (Melhor aproximação.) A solução  $u_h$  do problema de elementos finitos satisfaz

$$|||u - u_h||| \leq |||u - v_h|||, \forall v_h \in V_{h,0}. \quad (2.57)$$

*Demonstração.* Observando que  $u - u_h = u - v_h + v_h - u_h$  e usando a ortogonalidade de Galerkin (Teorema ??), temos:

$$|||u - u_h|||^2 = \int_{\Omega} \nabla(u - u_h) \cdot \nabla(u - u_h) \, dx \quad (2.58)$$

$$= \int_{\Omega} \nabla(u - u_h) \cdot \nabla(u - v_h) \, dx + \int_{\Omega} \nabla(u - u_h) \cdot \nabla(v_h - u_h) \, dx \quad (2.59)$$

$$= \int_{\Omega} \nabla(u - u_h) \cdot \nabla(u - v_h) \, dx \quad (2.60)$$

$$= \|\nabla(u - u_h)\|_{L^2(\Omega)}^2 \|\nabla(u - v_h)\|_{L^2(\Omega)}^2 \quad (2.61)$$

$$= |||u - u_h|||^2 |||u - v_h|||. \quad (2.62)$$

□

**Teorema 2.4.5.** (Estimativa *a priori* do erro.) A solução  $u_h$  do problema de elementos finitos (??) satisfaz

$$\|u - u_h\|^2 \leq C \sum_{K \in \mathcal{K}} h_K^2 \|D^2 u\|_{L^2(K)}^2. \quad (2.63)$$

*Demonstração.* O resultado segue do Teorema da melhor aproximação (Teorema ??) e da estimativa do erro de interpolação (Proposição ??), pois

$$\|u - u_h\|^2 \leq \|u - \pi u\|^2 \quad (2.64)$$

$$= \|D(u - \pi u)\|_{L^2(\Omega)}^2 \quad (2.65)$$

$$\leq C \sum_{K \in \mathcal{K}} h_K^2 \|D^2 u\|_{L^2(\Omega)}^2. \quad (2.66)$$

□

Para obtermos uma estimativa na norma  $L^2(\Omega)$ , podemos usar a desigualdade de Poincaré.

**Teorema 2.4.6.** (Desigualdade de Poincaré.) Seja  $\Omega \subset \mathbb{R}^2$  um domínio limitado. Então, existe uma constante  $C = C(\Omega)$ , tal que

$$\|v\|_{L^2(\Omega)} \leq C \|\nabla v\|_{L^2(\Omega)}, \quad \forall v \in V_0. \quad (2.67)$$

*Demonstração.* Se  $\Omega$  tem contorno suficientemente suave, então existe  $\phi$  tal que  $-\Delta \phi = 1$  em  $\Omega$  com  $\sup_{x \in \Omega} |\nabla \phi| < C$ . Com isso, temos

$$\|v\|_{L^2(\Omega)}^2 = \int_{\Omega} v^2 dx \quad (2.68)$$

$$= - \int_{\Omega} v^2 \Delta \phi dx. \quad (2.69)$$

Agora, usando o Teorema de Green e a desigualdade de Cauchy-Schwarz, obtemos

$$\|v\|_{L^2(\Omega)}^2 = - \int_{\partial \Omega} v^2 n \cdot \nabla \phi ds + \int_{\Omega} \nabla v^2 \cdot \nabla \phi dx \quad (2.70)$$

$$= \int_{\Omega} 2v \nabla v \cdot \nabla \phi dx \quad (2.71)$$

$$\leq \sup_{x \in \Omega} |\nabla \phi| \|v\|_{L^2(\Omega)} \|\nabla v\|_{L^2(\Omega)}. \quad (2.72)$$

□

Com a desigualdade de Poincaré e da estimativa *a priori* do erro (Teorema ??), temos

$$\|u - u_h\|_{L^2(\Omega)} \leq C \|u - u_h\| \leq Ch \|D^2 u\|_{L^2(\Omega)}, \quad (2.73)$$

onde  $h = \max_{K \in \mathcal{K}} h_K$ . Entretanto, esta estimativa pode ser melhorada.

**Teorema 2.4.7.** (Estimativa ótima *a priori* do erro.) A solução  $u_h$  do problema de elementos finitos (??) satisfaz

$$\|u - u_h\|_{L^2(\Omega)} \leq Ch^2 \|D^2 u\|_{L^2(\Omega)}. \quad (2.74)$$

*Demonstração.* Seja  $e = u - u_h$  o erro e  $\phi$  a solução do problema dual (ou problema adjunto)

$$-\Delta \phi = e, \quad \forall x \in \Omega \quad (2.75)$$

$$\phi = 0, \quad \forall x \in \partial\Omega. \quad (2.76)$$

Então, usando a fórmula de Green, a ortogonalidade de Galerkin e, então, a desigualdade de Cauchy-Schwarz, temos

$$\|e^2\|_{L^2(\Omega)} = - \int_{\Omega} e \Delta \phi \, dx \quad (2.77)$$

$$= \int_{\Omega} \nabla e \cdot \nabla \phi \, dx - \int_{\partial\Omega} e n \cdot \nabla \phi \, ds \quad (2.78)$$

$$= \int_{\Omega} \nabla e \cdot \nabla (\phi - \pi \phi) \, dx \quad (2.79)$$

$$\leq \|\nabla e\|_{L^2(\Omega)} \|\nabla (\phi - \pi \phi)\|_{L^2(\Omega)}. \quad (2.80)$$

Da estimativa *a priori* (??) (que segue do Teorema ??) temos

$$\|\nabla e\|_{L^2(\Omega)} \leq Ch \|D^2 u\|_{L^2(\Omega)}. \quad (2.81)$$

Agora, da regularidade elíptica  $\|D^2 \phi\|_{L^2(\Omega)} \leq C \|\Delta \phi\|_{L^2(\Omega)}$  [?] e da estimativa do erro de interpolação (Proposição ??), temos

$$\|\nabla (\phi - \pi \phi)\|_{L^2(\Omega)} \leq Ch \|D^2 \phi\|_{L^2(\Omega)} \leq Ch \|\Delta \phi\|_{L^2(\Omega)} \leq Ch \|e\|_{L^2(\Omega)}. \quad (2.82)$$

Então, temos

$$\|e\|_{L^2(\Omega)}^2 \leq Ch \|D^2 u\|_{L^2(\Omega)} Ch \|e\|_{L^2(\Omega)}. \quad (2.83)$$

□

**Exemplo 2.4.1.** Consideremos o seguinte problema de Poisson

$$-\Delta u = -2(x_0^2 - x_0) - 2(x_1^2 - x_1), \quad x \in \Omega := (0, 1) \times (0, 1), \quad (2.84)$$

$$u = 0, \quad x \in \partial\Omega. \quad (2.85)$$

A solução analítica deste problema é  $u(x) = (x_0^2 - x_0)(x_1^2 - x_1)$ . Aqui, obtemos aproximações por elementos finitos  $u_h$  usando uma malha triangular uniforme  $n \times n$  nodos, i.e.  $h = 1/n$ . A Tabela ?? mostra os valores dos erros  $\|u - u_h\|_{L^2(\Omega)}$  para diferentes valores de  $h$ .

Tabela 2.1: Erros de aproximações por elementos finitos referente ao problema dado no Exemplo ??.

#nodos	$h$	$\ u - u_h\ _{L^2(\Omega)}$
$10 \times 10$	$1e-1$	$9.29e-4$
$20 \times 20$	$5e-2$	$2.34e-4$
$100 \times 100$	$1e-3$	$9.40e-6$

Com o [FEniCS](#), podemos computar a solução deste problema e o erro na norma  $L^2$  com o seguinte código:

```
from __future__ import print_function, division
from fenics import *
import numpy as np
import matplotlib.pyplot as plt

# malha
Nx = 100
Ny = 100
mesh = UnitSquareMesh(Nx, Ny)

# espaco
V = FunctionSpace(mesh, 'P', 1)

# cond. contorno
def boundary(x, on_boundary):
    return on_boundary

bc = DirichletBC(V, Constant(0.0), boundary)
```

```

# f
f = Expression('-2*(x[1]*x[1]-x[1])-2*(x[0]*x[0]-x[0])',degree=2)

# MEF problem
u = TrialFunction(V)
v = TestFunction(V)
a = dot(grad(u), grad(v))*dx
L = f*v*dx

#computa a sol
u = Function(V)
solve(a == L, u, bc)

# sol. analitica
ua = Expression('x[0]*(x[0]-1)*x[1]*(x[1]-1)',degree=4)

# erro norma L2
erro_L2 = errornorm(ua, u, 'L2')
print("||u-u_h||_L2 = %1.2E\n" % erro_L2)

# exportando em vtk
vtkfile = File('u.pvd')
vtkfile << u

```

### 2.4.3 Estimativa *a posteriori*

[[tag:revisar]]

Para obtermos uma estimativa *a posteriori* vamos precisar da chamada desigualdade do traço.

**Teorema 2.4.8.** (Desigualdade do traço) Seja  $\Omega \subset \mathbb{R}^2$  um domínio limitado com fronteira  $\partial\Omega$  convexa e suave. Então, existe uma constante  $C = C(\Omega)$ , tal que para qualquer  $v \in V$  temos

$$\|v\|_{L^2(\partial\Omega)} \leq C \left( \|v\|_{L^2(\Omega)}^2 + \|\nabla v\|_{L^2(\Omega)}^2 \right)^{1/2}. \quad (2.86)$$

*Demonstração.* Veja [?]. □

**Teorema 2.4.9.** (Estimativa *a posteriori*) A solução  $u_h$  do problema de elementos finitos (??) satisfaz

$$|||u - u_h|||^2 \leq C \sum_{K \in \mathcal{K}} \eta_K^2(u_h), \quad (2.87)$$

onde o elemento residual  $\eta_K(u_h)$  é definido por

$$\eta_K(u_h) = h_K \|f + \Delta u_h\|_{L^2(K)} + \frac{1}{2} h_K^{1/2} \|[n \cdot \nabla u_h]\|_{L^2(\partial K \setminus \partial \Omega)}. \quad (2.88)$$

Aqui,  $[n \cdot \nabla u_h]_K$  denota o salto na derivada normal de  $u_h$  nos lados interiores dos elementos de  $\mathcal{K}$ . Além disso, lembremos que  $\Delta u_h = 0$ .

*Demonstração.* Denotando  $e := u - u_h$  o erro entre a solução do problema forte e a solução de elementos finitos, temos

$$|||e|||^2 = \|\nabla e\|_{L^2(\Omega)}^2 \quad (2.89)$$

$$= \int_{\Omega} \nabla e \cdot \nabla e \, dx \quad (2.90)$$

$$= \int_{\Omega} \nabla e \cdot \nabla (e - \pi e) \, dx. \quad (2.91)$$

Nesta última equação, temos usado a ortogonalidade de Galerkin (Teorema ??). Daí, temos

$$\int_{\Omega} \nabla e \cdot \nabla (e - \pi e) \, dx = \sum_{K \in \mathcal{K}} \int_K \nabla e \cdot \nabla (e - \pi e) \, dx \quad (2.92)$$

$$= \sum_{K \in \mathcal{K}} - \int_K \Delta e (e - \pi e) \, dx + \int_{\partial K} n \cdot \nabla e (e - \pi e) \, ds, \quad (2.93)$$

$$= \sum_{K \in \mathcal{K}} \int_K (f + \Delta u_h) (e - \pi e) \, dx + \int_{\partial K \setminus \partial \Omega} n \cdot \nabla e (e - \pi e) \, ds, \quad (2.94)$$

uma vez que  $-\Delta e|_K = f + \Delta u_h|_K$  e, ambos,  $e$  e  $\pi e$  se anulam em  $\partial \Omega$ .

Para computarmos o segundo termo do lado direito da última equação, observamos que o erro em lado  $E$  recebe contribuições dos dois elementos  $K^\pm$  que compartilham  $E$ . Com isso, temos

$$\int_{\partial K^+ \cap \partial K^-} n \cdot \nabla e (e - \pi e) \, ds = \int_E (n^+ \cdot \nabla e^+ (e^+ - \pi e^+))$$

$$+ n^- \cdot \nabla e^-(e^- - \pi e^-) ds, \quad (2.95)$$

onde utilizamos a notação  $v^\pm = v|_{K^\pm}$ . Lembremos que o erro  $e$  é contínuo e, portanto,  $(e^+ - \pi e^+)|_E = (e^- - \pi e^-)|_E$ . Ainda,  $\nabla u$  é contínuo, logo  $(n^+ \cdot \nabla u^+ + n^- \cdot \nabla u^-)|_E = 0$ . Entretanto,  $\nabla u_h|_E$  não é geralmente contínuo, sendo apenas constante por partes. Assim sendo e denotando o salto  $[n \cdot \nabla u_h] := (n^+ \cdot \nabla u_h^+ + n^- \cdot \nabla u_h^-)$ , temos

$$\begin{aligned} \int_E (n^+ \cdot \nabla e^+(e - \pi e) + n^- \cdot \nabla e^-(e - \pi e)) ds \\ = - \int_E [n \cdot \nabla u_h](e - \pi e) ds. \end{aligned} \quad (2.96)$$

Com isso, temos

$$\sum_{K \in \mathcal{K}} \int_{\partial K \setminus \partial \Omega} n \cdot \nabla e(e - \pi e) ds = - \sum_{E \in \mathcal{E}_I} \int_E [n \cdot \nabla u_h](e - \pi e) ds, \quad (2.97)$$

onde  $\mathcal{E}_I$  é o conjunto dos lados interiores na triangularização  $\mathcal{K}$ . Logo, retornando a (??), obtemos

$$\begin{aligned} |||e|||^2 &= \sum_{K \in \mathcal{K}} \int_K (f + \nabla u_h)(e - \pi e) dx \\ &\quad - \frac{1}{2} \int_{\partial K \setminus \partial \Omega} [n \cdot \nabla u_h](e - \pi e) ds. \end{aligned} \quad (2.98)$$

Nos resta, agora, estimarmos estes dois termos do lado direito.

A estimativa do primeiro, segue da desigualdade de Cauchy-Schwarz seguida da estimativa padrão do erro de interpolação, i.e.

$$\int_K (f + \Delta u_h)(e - \pi e) dx \leq \|f + \Delta u_h\|_{L^2(\Omega)} \|e - \pi e\|_{L^2(\Omega)} \quad (2.99)$$

$$\leq \|f + \Delta u_h\|_{L^2(\Omega)} Ch_K \|De\|_{L^2(\Omega)} \quad (2.100)$$

Para estimarmos as contribuições dos lados, usamos a desigualdade do Traço [?]

$$\|v\|_{L^2(\Omega)}^2 \leq C \left( h_K^{-1} \|v\|_{L^2(K)}^2 + h_K \|\nabla v\|_{L^2(\Omega)}^2 \right). \quad (2.101)$$

Com esta, a desigualdade de Cauchy-Schwarz e a estimativa padrão do erro de interpolação, temos

$$\int_{\partial K \setminus \partial \Omega} [n \cdot \nabla u_h](e - \pi e) ds \leq \|[n \cdot \nabla u_h]\|_{L^2(\partial K)} \|e - \pi e\|_{L^2(\partial K)} \quad (2.102)$$

$$\leq \| [n \cdot \nabla u_h] \|_{L^2(\partial K)} C \left( h_K^{-1} \| e - \pi e \|_{L^2(K)}^2 + h_K \| D(e - \pi e) \|_{L^2(K)}^2 \right)^{1/2} \quad (2.103)$$

$$\leq \| [n \cdot \nabla u_h] \|_{L^2(\partial K)} C h_K^{1/2} \| De \|_{L^2(K)}. \quad (2.104)$$

Daí, a estimativa segue das (??) e (??). □



# Resposta dos Exercícios

**Exercício 1.2.1.** [Código](#) FENiCS.

**Exercício 1.3.1.** [Código](#).

**Exercício 1.4.1.** [Código](#).

# Bibliografia

- [1] Brenner, S.C.; Scott, L.R.. The mathematical Theory of Finite Element Methods. Springer, 2008.
- [2] Evans, L.C.. Partial Differential Equations. 2. ed., AMS, 2010. ISBN: 978-0-8218-4974-3
- [3] Langtangen, H.P.; Logg, A.. Solving PDEs in Python. Springer, 2017. ISBN: 978-3-319-52461-0
- [4] Larson, M.G.; Bengson, F.. The Finite Element Method: Theory, Implementation, and Applications. Springer, 2013.