Método de Elementos Finitos

Pedro H A Konzen

31 de janeiro de 2019

Licença

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, faço uso de códigos em python com suporte da biblioteca FEniCS.

Agradeço aos(às) estudantes e colegas que assiduamente ou esporadicamente contribuem com correções, sugestões e críticas em prol do desenvolvimento deste material didático.

Pedro H A Konzen

Sumário

Capítulo 1

Método de elementos finitos em 1D

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

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

Observamos que dado $v \in P_1(I)$, temos que v é unicamente determinada pelos valores $\alpha_0 = v(x_0)$ e $\alpha_1 = v(x_1)$. Como consequência, existe exatamente uma única função $v \in P_1(I)$ para quaisquer dados valores α_0 e α_1. Desta observação, introduzimos a chamada base nodal $\{\varphi_0, \varphi_1\}$ para $P_1(I)$, definida por

$$\varphi_j(x_i) = \left\{\begin{array}{ll} 1 & ,i = j, \\ 0 & ,i \neq j \end{array}\right. , \tag{1.2}$$

com $i{,}j = 0{,}1$. Com esta base, toda função $v \in P_1(I)$ pode ser escrita como uma combinação linear das funções φ_0 e φ_1 com coeficientes α_0 e α_1 (**graus de liberdade**), i.e.

$$v(x) = \alpha_0\varphi_0(x) + \alpha_1\varphi_1(x_1). \tag{1.3}$$

Além disso, observamos que

$$\varphi_0(x) = \frac{x_1 - x}{x_1 - x_0}, \quad \varphi_1(x) = \frac{x - x_0}{x_1 - x_0}. \tag{1.4}$$

Uma extensão do espaço $P_1(I)$ é o espaço das funções lineares por partes. Dado $I = [l_0, l_1]$, $l_0 \neq l_1$, consideremos uma partição (**malha**) de I com $n+1$ pontos $\mathcal{I} = \{l_0 = x_0, x_1, \ldots, x_n = l_1\}$ e, portanto, com n subintervalos $I_i = [x_{i-1}, x_i]$ de comprimento $h_i = x_i - x_{i-1}$, $i = 1, 2, \ldots, n$. Na malha $\mathcal{I}$ definimos o seguinte espaço das funções lineares por partes

$$V_h := \{v : \; v \in C^0(\mathcal{I}),\, v|_{I_i} \in P_1(I_i),\, i = 1{,}2, \ldots ,n\}. \tag{1.5}$$

Observamos que toda função $v \in V_h$ é unicamente determinadas por seus valores nodais $\{\alpha_i = v(x_i)\}_{i=0}^n$. Reciprocamente, todo conjunto de valores nodas $\{\alpha_i\}_{i=0}^n$ 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

$$\varphi_j(x_i) = \left\{ \begin{array}{ll} 1 & , i = j, \\ 0 & , i \neq j \end{array} \right. , \tag{1.6}$$

com $i{,}j = 0{,}1, \ldots ,n$. Podemos verificar que

$$\varphi_i(x) = \left\{ \begin{array}{ll} (x - x_{i-1})/h_i & , x \in I_i, \\ (x_{i+1} - x)/h_{i+1} & , x \in I_{i+1}, \\ 0 & , \text{noutros casos} \end{array} \right. \tag{1.7}$$

veja, Figura 1.1. É notável que $\varphi_i(x)$ tem suporte compacto $I_i \cup I_{i+1}$.

1.1.1 Interpolação

A interpolação é uma das técnicas de aproximação de funções. Dada uma função contínua f em $I = [x_0, x_1]$, definimos o **operador de interpolação linear** $\pi : C^0(I) \to P_1(I)$ por

$$\pi f(x) = f(x_0)\varphi_0(x) + f(x_1)\varphi_1(x). \tag{1.8}$$

Observamos que πf é igual a f nos nodos x_0 e x_1.

Exemplo 1.1.1. A Figura 1.2 ilustra a interpolação da função $f(x) = 3\,\text{sen}(2\pi x)$ no espaço $P_1([1/4, 3/4])$. Neste caso

$$\pi f(x) = f\left(\frac{1}{4}\right)\frac{3/4 - x}{1/2} + f\left(\frac{3}{4}\right)\frac{x - 1/4}{1/2}. \tag{1.9}$$

Com o FENiCS, podemos computar a função interpolada πf com o seguinte código:

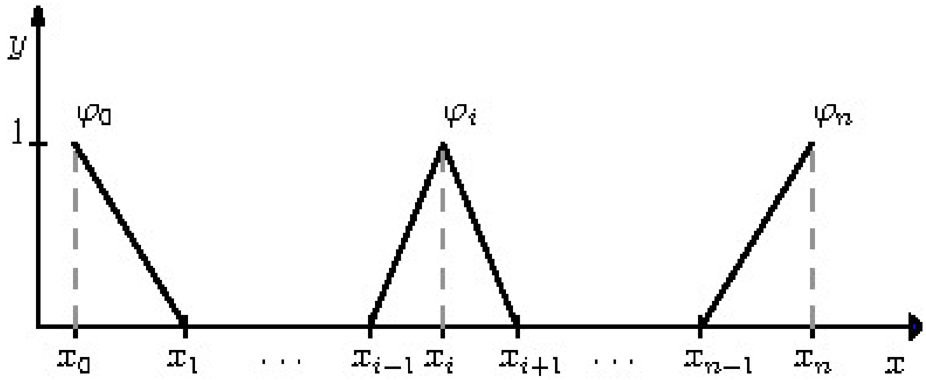

Figura 1.1: Base nodal para o espaço das funções lineares por parte.

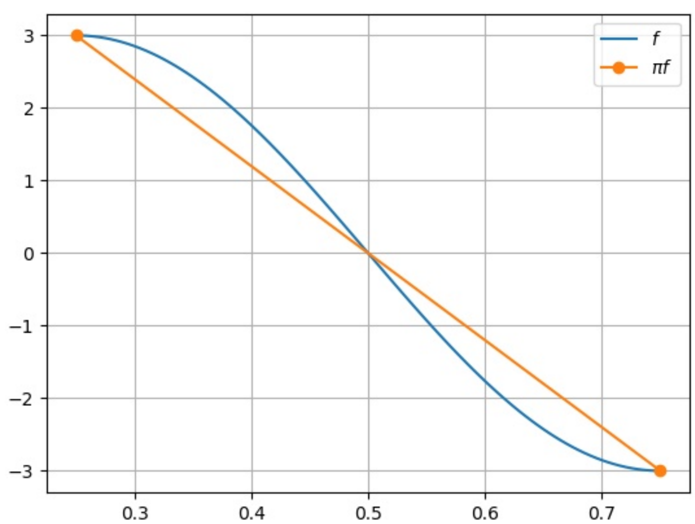

Figura 1.2: Interpolação linear de $f(x) = 3\,\mathrm{sen}(2\pi x)$ no espaço $P_1([1/4, 3/4])$.

```
from __future__ import print_function, division
from fenics import *
import numpy as np
```

```
import matplotlib.pyplot as plt

# malha
mesh = IntervalMesh(1,0.25,0.75)

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

# funcao
f = Expression('3*sin(2*pi*x[0])',
                   degree=10)

# interpolacao
pif = interpolate(f,V)

# grafico
xx = IntervalMesh(100,0.25,0.75)
plot(f,mesh=xx,label="$f$")
plot(pif,mesh=mesh,
     marker='o',label="$\pi f$")
plt.legend(numpoints=1)
plt.grid('on')
plt.show()
```

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

$$\|v\|_{L^2(I)} = \left(\int_I v^2\,dx\right)^{1/2}. \tag{1.10}$$

Lembramos que valem a **desigualdade triangular**

$$\|v + w\|_{L^2(I)} \leq \|v\|_{L^2(I)} + \|w\|_{L^2(I)} \tag{1.11}$$

e a **desigualdade de Cauchy-Schwarz**[1]

$$\int_I vw\,dx \leq \|v\|_{L^2(I)}\|w\|_{L^2(I)}, \tag{1.12}$$

[1] Também conhecida como desigualdade de Cauchy–Bunyakovsky–Schwarz. Augustin-Louis Cauchy, 1789 - 1857, matemático francês. Viktor Yakovlevich Bunyakovsky, 1804 - 1889, matemático Russo. Karl Hermann Amandus Schwarz, 1843 - 1921, matemático alemão.

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

Proposição 1.1.1. (Erro da interpolação linear) O interpolador πf satisfaz as estimativas

$$\|f - \pi f\|_{L^2(I)} \leq Ch^2 \|f''\|_{L^2(I)}, \tag{1.13}$$
$$\|(f - \pi f)'\|_{L^2(I)} \leq Ch \|f''\|_{L^2(I)}, \tag{1.14}$$

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

$$e(y) = e(x_0) + \int_{x_0}^{y} e'(x)\, dx, \tag{1.15}$$

onde $e(x_0) = f(x_0) - \pi f(x_0) = 0$. Daí, usando a desigualdade de Cauchy-Schwarz (1.12), temos

$$\begin{aligned}
e(y) &= \int_{x_0}^{y} e'\, dx &(1.16)\\
&\leq \int_{x_0}^{y} |e'|\, dx &(1.17)\\
&\leq \int_I 1 \cdot |e'|\, dx &(1.18)\\
&\leq \left(\int_I 1^2\, dx\right)^{1/2} \left(\int_I e'^2\, dx\right)^{1/2} &(1.19)\\
&= h^{1/2} \left(\int_I e'^2\, dx\right)^{1/2}, &(1.20)
\end{aligned}$$

donde

$$e(y)^2 \leq h \int_I e'^2\, dx = h\|e'\|^2_{L^2(I)}. \tag{1.21}$$

Então, integrando em I obtemos

$$\|e\|^2_{L^2(I)} = \int_I e^2(y)\, dy \leq \int_I h\|e'\|^2_{L^2(I)}\, dy = h^2 \|e'\|^2_{L^2(I)}, \tag{1.22}$$

ou seja, temos a seguinte desigualdade

$$\|e\|_{L^2(I)} \leq h\|e'\|_{L^2(I)}. \tag{1.23}$$

Agora, observando que $e(x_0) = e(x_1) = 0$, o **teorema de Rolle**[2] 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

$$\begin{aligned}
e'(y) &= e'(\tilde{x}) + \int_{\tilde{x}}^{y} e''\,dx && (1.24)\\
&= \int_{\tilde{x}}^{y} e''\,dx && (1.25)\\
&\leq \int_I 1\cdot|e''|\,dx && (1.26)\\
&\leq h^{1/2}\left(\int_I e''^2\right)^{1/2}. && (1.27)
\end{aligned}$$

Então, integrando em I, obtemos

$$\|e'\|^2_{L^2(I)} \leq h^2\|e''\|^2_{L^2(I)}, \tag{1.28}$$

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

$$\|(f-\pi f)'\|_{L^2(I)} \leq Ch\|f''\|_{L^2(I)}. \tag{1.29}$$

Por fim, de (1.28) e de (1.23), obtemos a primeira estimativa desejada

$$\|f-\pi f\|_{L^2(I)} \leq Ch^2\|f''\|_{L^2(I)}. \tag{1.30}$$

□

Exemplo 1.1.2. A Figura 1.3 mostra a evolução do erro na norma L^2 da interpolação de $f(x) = 3\,\text{sen}(2\pi x)$ no espaço $P_1([0,h)]$ para $h = 10^{-5}, 10^{-4}, \cdots, 10^{-1}$. Com o FENiCS, podemos computar os erros de interpolação com o seguinte código:

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

# funcao
f = Expression('3*sin(2*pi*x[0])',
               degree=10)
```

[2] Michel Rolle, 1652 - 1719, matemático francês.

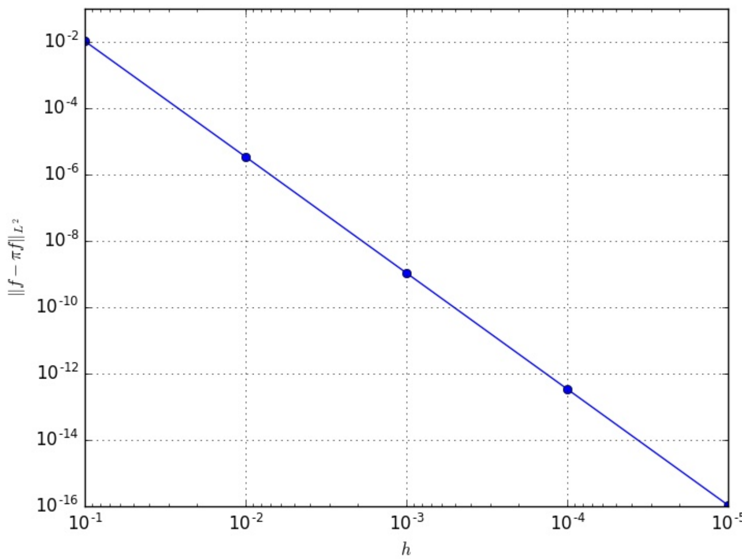

Figura 1.3: Erro de interpolação de $f(x) = 3\,\text{sen}(2\pi x)$ no espaço $P_1([0,h])$.

```
n=5
for k in np.arange(1,n+1):
    h = 10**-k
    mesh = IntervalMesh(1,0,h)
    V = FunctionSpace(mesh, 'P', 1)
    pif = interpolate(f,V)
    e = errornorm(f,pif,'L2')
    print("%d %1.0E %1.1E" % (k,h,e))
```

Vamos, agora, generalizar o resultado da Proposição 1.1.1 para a interpolação no espaço V_h das funções lineares por parte. Dada uma função contínua f em $I = [l_0, l_1]$, definimos o operador interpolador $\pi : C^0(I) \to V_h$ na malha $\mathcal{I}$ de I por

$$\pi f(x) = \sum_{i=0}^{n} f(x_i)\varphi_i(x). \tag{1.31}$$

Exemplo 1.1.3. A Figura 1.4 ilustra a interpolação da função $f(x) = 3\,\mathrm{sen}(2\pi x)$ 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 pontos).

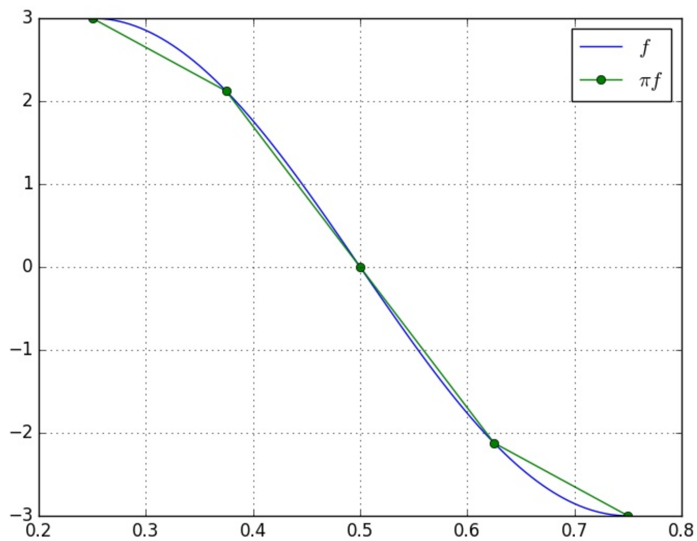

Figura 1.4: Interpolação linear de $f(x) = 3\,\mathrm{sen}(2\pi x)$ no espaço V_h das funções lineares por partes sobre uma malha com 5 pontos.

Com o FENiCS, podemos computar a função interpolada πf com o seguinte código:

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

# malha
mesh = IntervalMesh(4,0.25,0.75)

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

```
# funcao
f = Expression('3*sin(2*pi*x[0])',
                   degree=10)

# interpolacao
pif = interpolate(f,V)

# grafico
xx = IntervalMesh(100,0.25,0.75)
plot(f,mesh=xx,label="$f$")
plot(pif,mesh=mesh,
     marker='o',label="$\pi f$")
plt.legend(numpoints=1)
plt.grid('on')
plt.show()
```

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

Proposição 1.1.2. *O interpolador πf satisfaz as estimativas*

$$\|f - \pi f\|_{L^2(I)}^2 \leq C \sum_{i=1}^{n} h_i^4 \|f''\|_{L^2(I)}^2, \tag{1.32}$$

$$\|(f - \pi f)'\|_{L^2(I)}^2 \leq C \sum_{i=1}^{n} h_i^2 \|f''\|_{L^2(I)}^2. \tag{1.33}$$

$$\tag{1.34}$$

Demonstração. Ambas desigualdades seguem da desigualdade triangular e da Proposição 1.1.1. Por exemplo, para a primeira desigualdade, temos

$$\|f - \pi f\|_{L^2(I)}^2 \leq \sum_{i=1}^{n} \|f - \pi f\|_{L^2(I_i)}^2 \tag{1.35}$$

$$\leq \sum_{i=1}^{n} C h_i^4 \|f''\|_{L^2(I_i)}^2. \tag{1.36}$$

□

1.1.2 Projeção L^2

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

$$\int_I (f - P_h f) v \, dx = 0, \quad \forall v \in V_h. \tag{1.37}$$

Como V_h é um espaço de dimensão finita, a condição (1.38) é equivalente a

$$\int_I (f - P_h f) \varphi_i \, dx = 0, \quad i = 0, 1, \cdots, n, \tag{1.38}$$

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

$$P_h f = \sum_{j=0}^{n} \xi_j \varphi_j, \tag{1.39}$$

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

$$\int_I (f - P_h f)\varphi_i \, dx = 0 \Leftrightarrow \int_I f\varphi_i \, dx = \int_I P_h f \varphi_i \, dx \tag{1.40}$$

$$\Leftrightarrow \int_I f\varphi_i \, dx = \int_I \left(\sum_{j=0}^{n} \xi_j \varphi_j \right) \varphi_i \, dx \tag{1.41}$$

$$\Leftrightarrow \sum_{j=0}^{n} \xi_j \int_I \varphi_j \varphi_i \, dx = \int_I f \varphi_i \, dx, \tag{1.42}$$

para $i = 0, 1, \ldots, n$.
Observemos, agora, que (1.42) consiste em um sistema de $n+1$ equações lineares para as $n+1$ incógnitas ξ_j, $j = 0, 1, \ldots, n$. Este, por sua vez, pode ser escrito na seguinte forma matricial

$$M\xi = b, \tag{1.43}$$

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

$$m_{i,j} = \int_I \varphi_j \varphi_i \, dx \tag{1.44}$$

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

$$b_i = \int_I f \varphi_i \, dx. \tag{1.45}$$

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

$$P_h f = \sum_{j=0}^{n} \xi_j \varphi_j, \tag{1.46}$$

onde $\xi = (\xi_0, \xi_1, \dots, \xi_n)$ é solução do sistema (1.43).

Exemplo 1.1.4. A Figura 1.5 ilustra a projeção L^2 da função $f(x) = 3\,\mathrm{sen}(2\pi x)$ 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 pontos).

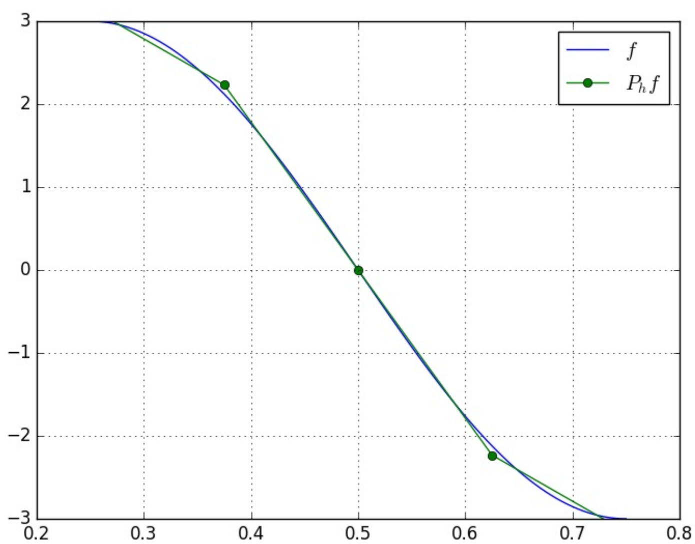

Figura 1.5: Projeção L^2 de $f(x) = 3\,\mathrm{sen}(2\pi x)$ no espaço V_h das funções lineares por partes sobre uma malha com 5 pontos.

Com o FENiCS, podemos computar $P_h f$ com o seguinte código:

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

# malha
```

```
mesh = IntervalMesh(4,0.25,0.75)

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

# funcao
f = Expression('3*sin(2*pi*x[0])',
                   degree=10)

# projecao
Phf = project(f, V)

# grafico
xx = IntervalMesh(100,0.25,0.75)
plot(f,mesh=xx,label="$f$")
plot(Phf,mesh=mesh,
     marker='o',label="$P_h f$")
plt.legend(numpoints=1)
plt.grid('on')
plt.show()
```

O próximo teorema mostra que P_hf 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

$$\|f - P_hf\|_{L^2(I)} \leq \|f - v\|_{L^2(I)}, \quad \forall v \in V_h. \tag{1.47}$$

Demonstração. Dado $v \in V_h$, temos

$$\begin{align}
\|f - P_hf\|^2_{L^2(I)} &= \int_I |f - P_hf|^2\,dx \tag{1.48}\\
&= \int_I (f - P_hf)(f - v + v - P_hf)\,dx \tag{1.49}\\
&= \int_I (f - P_hf)(f - v)\,dx + \int_I (f - P_hf)(v - P_hf)\,dx \tag{1.50}\\
&= \int_I (f - P_hf)(f - v)\,dx \tag{1.51}\\
&\leq \|f - P_hf\|_{L^2(I)}\|f - v\|_{L^2(I)}, \tag{1.52}
\end{align}$$

donde segue o resultado. □

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*

$$\|f - P_h f\|^2_{L^2(I)} \leq C \sum_{i=1}^{n} h_i^4 \|f''\|^2_{L^2(I_i)}. \tag{1.53}$$

Demonstração. Tomando a interpolação $\pi f \in V_h$, temos do Teorema da melhor aproximação (Teorema 1.1.1) e da estimativa do erro de interpolação (Proposição 1.1.2) que

$$\|f - P_h f\|^2_{L^2(I)} \leq \|f - \pi f\|^2_{L^2(I)} \tag{1.54}$$

$$\leq C \sum_{i=1}^{n} h_i^4 \|f''\|^2_{L^2(I_i)}. \tag{1.55}$$

□

Exemplo 1.1.5. A Figura 1.6 mostra a evolução do erro na norma L^2 da projeção de $f(x) = 3\,\mathrm{sen}(2\pi x)$ no espaço V_h em malhas uniformes de $h = 10^{-5}, 10^{-4}, \cdots, 10^{-1}$ no intervalo $[1/4, 3/4]$.
Com o FENiCS, podemos computar os erros de projeção com o seguinte código:

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

# funcao
f = Expression('3*sin(2*pi*x[0])',
               degree=10)

n=5
for k in np.arange(0,n):
    mesh = IntervalMesh(5*10**k,0.25,0.75)
    h = mesh.hmax()
    V = FunctionSpace(mesh, 'P', 1)
    Phf = project(f,V)
    e = errornorm(f,Phf,'L2')

    print("%d %1.0E %1.1E" % (k,h,e))
```

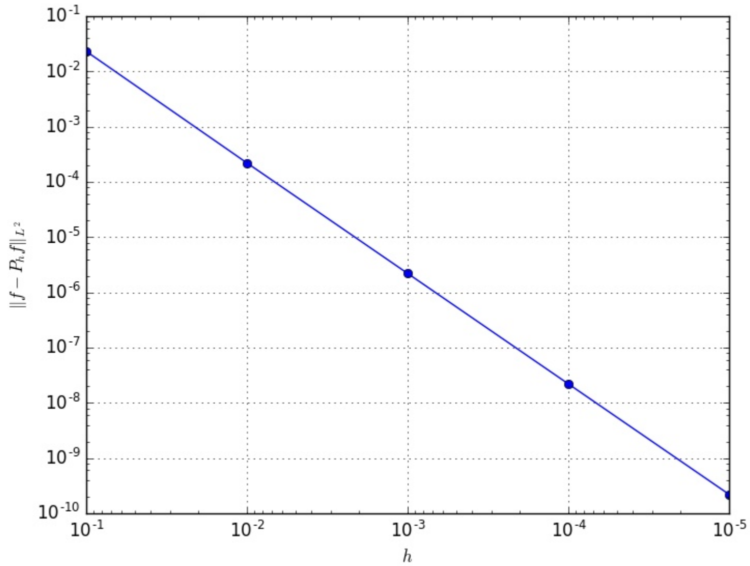

Figura 1.6: Erro de interpolação de $f(x) = 3\,\text{sen}(2\pi x)$ no espaço V_h.

Exercícios

E 1.1.1. Faça um código para verificar a segunda estimativa da Proposição 1.1.1 no caso da interpolação da função $f(x) = 3\,\text{sen}(2\pi x)$ no espaço P_1 das funções lineares.

E 1.1.2. Faça um código para verificar as estimativas da Proposição 1.1.2 no caso da interpolação da função $f(x) = 3\,\text{sen}(2\pi x)$ no espaço V_h das funções lineares por partes.

E 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

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

$$-u'' = f, \quad x \in I = [0,L], \tag{1.56}$$

$$u(0) = u(L) = 0, \tag{1.57}$$

onde f é uma função dada.

1.2.1 Formulação fraca

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 (1.56)-(1.57), tomamos o espaço

$$V_0 = \{v \in H^1(I) : \; v(0) = v(1) = 0\}. \tag{1.58}$$

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 (1.56) por uma função teste $v \in V_0$ (arbitrária) e integrando-se por partes, i.e.

$$\int_I fv\,dx = -\int_I u''v\,dx \tag{1.59}$$

$$= \int_I u'v'\,dx - u'(L)v(L) + u'(0)v(0) \tag{1.60}$$

$$\tag{1.61}$$

Donde, das condições de contorno, temos

$$\int_I u'v'\,dx = \int_I fv\,dx. \tag{1.62}$$

Desta forma, o problema fraco associado a (1.56)-(1.57) lê-se: encontrar $u \in V_0$ tal que

$$a(u,v) = L(v), \quad \forall v \in V_0, \tag{1.63}$$

onde

$$a(u,v) = \int_I u'v'\,dx \tag{1.64}$$

$$L(v) = \int_I fv\,dx, \tag{1.65}$$

são chamadas de forma bilinear e forma linear, respectivamente.

1.2.2 Formulação de elementos finitos

Uma formulação de elementos finitos é um aproximação do problema fraco (1.63) 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.

$$V_{h,0} = \{v \in V_h : \; v(0) = v(L) = 0\}. \tag{1.66}$$

Então, substituindo o espaço V_0 pelo subespaço $V_{h,0} \subset V_0$ em (1.63), obtemos o seguinte problema de elementos finitos: encontrar $u_h \in V_{h,0}$ tal que

$$a(u_h,v) = L(v), \quad \forall v \in V_{h,0}. \tag{1.67}$$

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[3].

Observemos que o problema (1.67) é equivalente a: encontrar $u_h \in V_{h,0}$ tal que

$$a(u_h,\varphi_i) = L(\varphi_i), \quad i = 1, \ldots, n-1, \tag{1.68}$$

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

$$u_h = \sum_{j=1}^{n-1} \xi_j \varphi_j, \tag{1.69}$$

onde ξ_j, $j = 1,2, \ldots, n-1$, são incógnitas a determinar. I.e., ao computarmos ξ_j, $j = 1,2, \ldots, n-1$, temos obtido a solução u_h do problema de elementos finitos 1.67.

Agora, da forma bilinear (1.64), temos

$$a(u_h,\varphi_i) = a\left(\sum_{j=1}^{n-1} \xi_j \varphi_j, \varphi_i\right) \tag{1.70}$$

$$= \sum_{j=1}^{n-1} \xi_j a(\varphi_j,\varphi_i). \tag{1.71}$$

Daí, o problema (1.67) é equivalente a resolvermos o seguinte sistema de equações lineares

$$A\xi = b, \tag{1.72}$$

[3]Boris Grigoryevich Galerkin, matemático e engenheiro soviético. Fonte: Wikipédia.

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

$$a_{i,j} = a(\varphi_j,\varphi_i) = \int_I \varphi_j'\varphi_i'\,dx, \tag{1.73}$$

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

$$b_i = L(\varphi_i) = \int_I \varphi_i\,dx. \tag{1.74}$$

Exemplo 1.2.1. Consideremos o problema (1.56)-(1.57) com $f \equiv 1$ e $L = 1$, i.e.

$$-u'' = 1, \quad x \in I = [0,1], \tag{1.75}$$

$$u(0) = u(1) = 0. \tag{1.76}$$

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 $u \in V_0 = \{v \in H^1(I);\ v(0) = v(L) = 0\}$ tal que

$$a(u, v) = L(v), \tag{1.77}$$

onde

$$a(u, v) = \int_I u'v'\,dx, \quad L(v) = \int_I fv\,dx. \tag{1.78}$$

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

$$a(u_h, v_h) = L(v_h), \quad \forall v_h \in V_{h,0}. \tag{1.79}$$

A Figura 1.7 apresenta o esboço dos gráficos da solução analítica u e da sua aproximação de elementos finitos u_h.

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
```

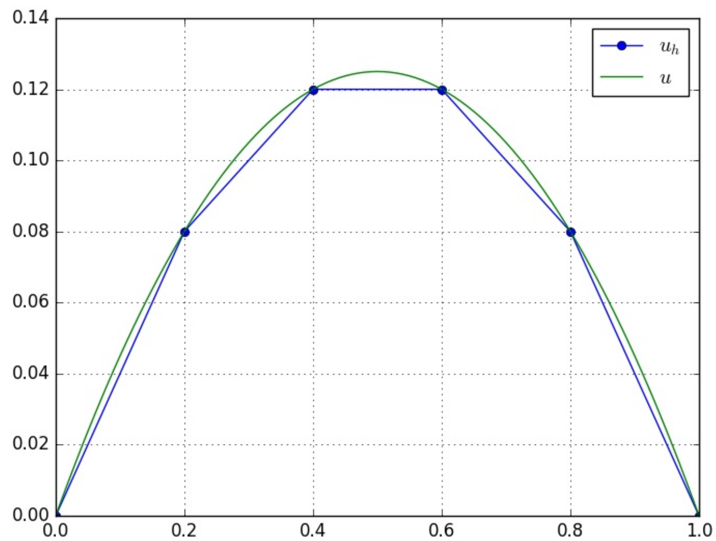

Figura 1.7: Esboço dos gráficos das soluções referentes ao Exemplo 1.2.1.

```
# malha
mesh = IntervalMesh(5,0,1)

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

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

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)

#grafico
plot(u,marker="o",label="$u_h$")

#sol analitica
ua = Expression('-x[0]*x[0]/2+x[0]/2',
               degree=2)
xx = IntervalMesh(100,0,1)
plot(ua,mesh=xx,label="$u$")
plt.legend(numpoints=1)
plt.grid('on')
plt.show()
```

1.2.3 Estimativa *a priori*

Existem dois tipos de estimativas do erro $e := u - u_h$. Estimativas *a priori*, são aqueles 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 (1.67) satisfaz a seguinte propriedade de ortogonalidade

$$a(u - u_h, v) := \int_I (u - u_h)' v' \, dx = 0, \quad v \in V_{h,0}, \tag{1.80}$$

onde u é a solução de (1.63).

Demonstração. De (1.67), (1.63) e lembrando que $V_{h,0} \subset V_0$, temos

$$a(u,v) = L(v) = a(u_h,v) \Rightarrow a(u - u_h, v) = 0, \tag{1.81}$$

para todo $v \in V_{h,0}$. □

Teorema 1.2.2. *(*A melhor aproximação) A solução de elementos finitos u_h dada por (1.67) satisfaz a seguinte propriedade de melhor aproximação

$$\|(u-u_h)'\|_{L^2(I)} \leq \|(u-v)'\|_{L^2(I)}, \quad v \in V_{h,0}, \tag{1.82}$$

onde u é a solução de (1.63).

Demonstração. Escrevendo $u - u_h = u - v + v - u_h$ para qualquer $v \in V_{h,0}$ e usando a ortogonalidade de Galerkin (Teorema 1.2.1), temos

$$\begin{align}
\|(u-u_h)'\|^2_{L^2(I)} &= \int_I (u-u_h)'(u-u_h)'\,dx \tag{1.83}\\
&= \int_I (u-u_h)'(u-v+v-u_h)'\,dx \tag{1.84}\\
&= \int_I (u-u_h)'(u-v)'\,dx + \int_I (u-u_h)'(v-u_h)'\,dx \tag{1.85}\\
&= \int_I (u-u_h)'(u-v)'\,dx \tag{1.86}\\
&\leq \|(u-u_h)'\|_{L^2(I)}\|(u-v)'\|_{L^2(I)}. \tag{1.87}
\end{align}$$

□

Teorema 1.2.3. *(*Estimativa *a priori*) O erro em se aproximar a solução u de (1.63) pela solução de elementos finitos u_h dada por (1.67) satisfaz a seguinte estimativa *a priori*

$$\|(u-u_h)'\|^2_{L^2(I)} \leq C\sum_{i=1}^n h_i^2\|u''\|^2_{L^2(I_i)}. \tag{1.88}$$

Demonstração. Tomando $v = \pi u$ no teorema da melhor aproximação (Teorema 1.2.2), obtemos

$$\|(u-u_h)'\|_{L^2(I)} \leq \|(u-\pi u)'\|_{L^2(I)}. \tag{1.89}$$

Daí, da estimativa do erro de interpolação (Proposição 1.1.2), temos

$$\|(u-u_h)'\|^2_{L^2(I)} \leq C\sum_{i=1}^n h_i^2\|u''\|^2_{L^2(I_i)}. \tag{1.90}$$

□

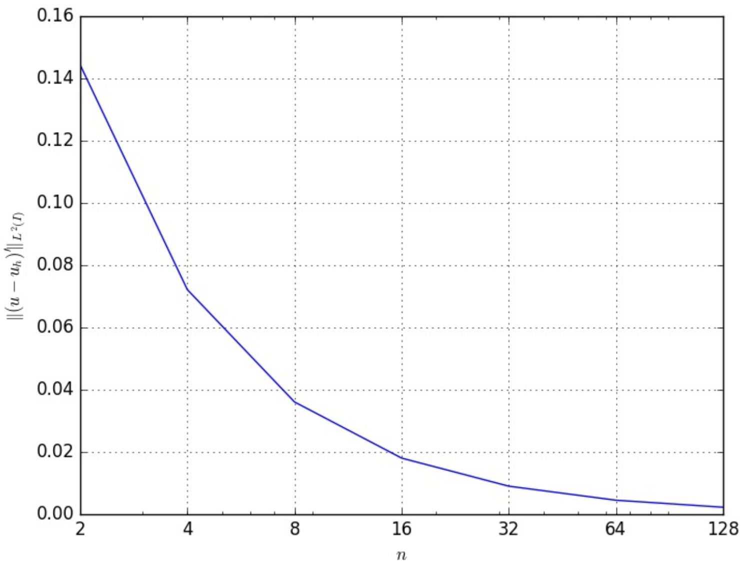

Figura 1.8: Esboço dos gráficos das soluções referentes ao Exemplo 1.2.2.

Exemplo 1.2.2. A Figura 1.8 apresenta o esboço da evolução do erro $\|(u - u_h)'\|_{L^2(I)}$ da solução de elementos finitos do problema (1.75)-(1.76) para malhas uniformes com $n = 2, 4, 8, \ldots, 128$ células.
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',basex=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*

Aqui, vamos obter uma estimativa a posteriori para o erro $e = u - u_h$ da solução de elementos finitos u_h do problema (1.56)-(1.57).

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

$$\|(u - u_h)'\|^2_{L^2(I)} \leq C \sum_{i=1}^{n} \eta_i^2(u_h), \tag{1.91}$$

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

$$\eta_i(u_h) = h_i \|f - u_h''\|_{L^2(I_i)}. \tag{1.92}$$

Demonstração. Tomando $e = u - u_h$ e usando a ortogonalidade de Galerkin (Teorema 1.2.1) temos

$$\|e'\|^2_{L^2(I)} = \int_I e'(e - \pi e)'\,dx = \sum_{i=1}^{n} \int_{I_i} e'(e - \pi e)'\,dx. \tag{1.93}$$

Então, aplicando integração por partes

$$\|e'\|^2_{L^2(I)} = \sum_{i=1}^{n} \int_{I_i} (-e'')(e - \pi e)\,dx + [e'(e - \pi e)]_{x_{i-1}}^{x_i}. \tag{1.94}$$

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

$$\|e'\|^2_{L^2(I)} = \sum_{i=1}^{n} \int_{I_i} (f + u_h'')(e - \pi e)\,dx. \tag{1.95}$$

Agora, usando as desigualdades de Cauchy-Schwarz e a estimativa padrão de interpolação (1.23), obtemos

$$\begin{align}
\|e'\|^2_{L^2(I)} &\leq \sum_{i=1}^{n} \|f + u_h\|_{L^2(I_i)} \|e - \pi e\|_{L^2(I_i)}\,dx \tag{1.96}\\
&\leq C \sum_{i=1}^{n} h_i \|f + u_h\|_{L^2(I_i)} \|e'\|_{L^2(I_i)} \tag{1.97}\\
&\leq C \left(\sum_{i=1}^{n} h_i^2 \|f + u_h\|^2_{L^2(I_i)}\right)^{1/2} \left(\sum_{i=1}^{n} \|e'\|^2_{L^2(I_i)}\right)^{1/2} \tag{1.98}\\
&= C \left(\sum_{i=1}^{n} h_i^2 \|f + u_h\|^2_{L^2(I_i)}\right)^{1/2} \|e'\|_{L^2(I)}, \tag{1.99}
\end{align}$$

donde segue o resultado desejado. □

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

Exercícios

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

$$-u'' + u = 2\operatorname{sen} x, \quad \forall x \in (-\pi, \pi), \tag{1.100}$$
$$u(-\pi) = u(\pi) = 0. \tag{1.101}$$

1.3 Condições de contorno

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

$$-u'' = f, \quad x \in I = [0, L], \tag{1.102}$$

com diferentes condições de contorno.

1.3.1 Condições de Dirichlet

Consideremos o seguinte problema com condições de contorno de Dirichlet[4]: encontrar u tal que

$$-u'' = f, \quad \forall x \in I = [0, L], \tag{1.103}$$
$$u(0) = u_0, \quad u(L) = u_L, \tag{1.104}$$

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 (1.103), obtemos

$$-\int_I u''v\,dx = \int_I fv\,dx. \tag{1.105}$$

[4]Johann Peter Gustav Lejeune Dirichlet, 1805 - 1859, matemático alemão. Fonte: Wikipedia.

Aplicando a integração por partes, temos

$$\int_I u'v'\,dx = \int_I fv\,dx. \tag{1.106}$$

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

$$a(u,v) = L(v), \quad \forall v \in V_0, \tag{1.107}$$

onde $a(u,v)$ é a forma bilinear

$$a(u,v) = \int_I u'v'\,dx \tag{1.108}$$

e $L(v)$ é a forma linear

$$L(v) = \int_I fv\,dx. \tag{1.109}$$

Exemplo 1.3.1. Consideremos o problema

$$-u'' = 1, \quad x \in I = [0,1], \tag{1.110}$$

$$u(0) = 1/2, \quad u(1) = 1. \tag{1.111}$$

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

$$a(u,v) = L(v), \tag{1.112}$$

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

$$a(u,v) = \int_I u'v'\,dx, \quad L(v) = \int_I fv\,dx. \tag{1.113}$$

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

$$a(u_h, v_h) = L(v_h), \tag{1.114}$$

para todo $v_h \in V_{h,0} = \{v \in H^1(I);\ v(0) = v(1) = 0\}$.
A Figura 1.9 apresenta o esboço dos gráficos da solução analítica u e da sua aproximação de elementos finitos u_h, esta construída no espaço dos polinômios lineares por partes sobre uma malha uniforme de 5 células.
Com o FENiCS, a computação do problema de elementos finitos pode ser feita com o seguinte código:

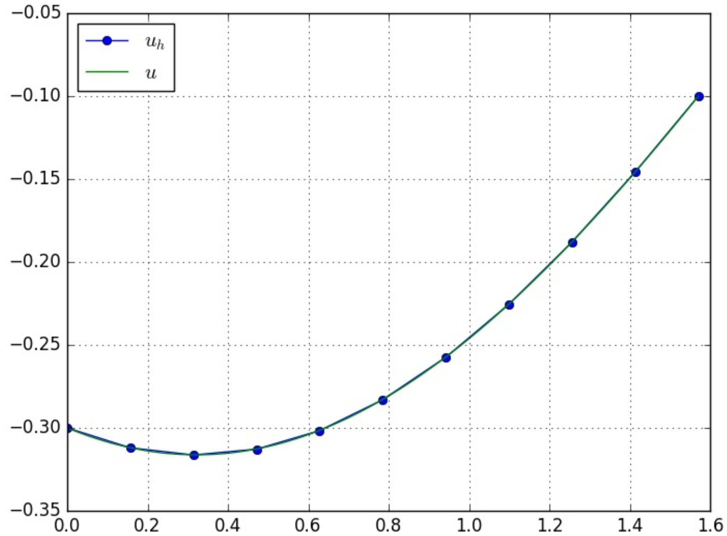

Figura 1.9: Esboço dos gráficos das soluções referentes ao Exemplo 1.3.1.

```
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(5,0,1)

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

u_D = Expression('x[0]<0.5 ? 0.5 : 1',degree=1)

def boundary(x,on_boundary):
    return on_boundary
```

```
bc = DirichletBC(V,u_D,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)

#sol analitica
ua = Expression('-x[0]*x[0]/2+x[0]+0.5',
                degree=2)

#erro L2
print("Erro L2: %1.2E\n" % errornorm(u,ua,norm_type="L2"))

plot(u,mesh=mesh,marker='o',label=r"$u_h$")
mesh = IntervalMesh(100,0,1)
plot(ua,mesh=mesh,label=r"$u$")
plt.legend(numpoints=1)
plt.show()
```

1.3.2 Condições de Neumann

Consideremos o seguinte problema com condições de contorno de Neumann[5] homogênea em $x = L$: encontrar u tal que

$$-u'' = f, \quad \forall x \in I = [0, L], \tag{1.115}$$

$$u(0) = u_0, \quad u'(L) = 0, \tag{1.116}$$

com u_0 e f dados.

[5]Carl Gottfried Neumann, 1832 - 1925, matemático alemão. Fonte: Wikipedia.

Tomando uma função teste $v \in V := \{v \in H^1(I);\ v(0) = 0\}$ e multiplicando-a em (1.115), obtemos

$$-\int_I u''v\,dx = \int_I fv\,dx. \tag{1.117}$$

Aplicando a integração por partes, temos

$$\int_I u'v'\,dx - \underbrace{u'(L)v(L)}_{u'(L)=0} + \underbrace{u'(0)v(0)}_{v(0)=0} = \int_I fc\,dx. \tag{1.118}$$

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

$$a(u,v) = L(v), \quad \forall v \in V, \tag{1.119}$$

onde $a(u,v)$ é a forma bilinear

$$a(u,v) = \int_I u'v'\,dx \tag{1.120}$$

e $L(v)$ é a forma linear

$$L(v) = \int_I fv\,dx. \tag{1.121}$$

Exemplo 1.3.2. Consideremos o problema

$$-u'' = 1, \quad x \in I = [0,1], \tag{1.122}$$

$$u(0) = 0, \quad u'(1) = 0. \tag{1.123}$$

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

$$a(u, v) = L(v), \tag{1.124}$$

para todo $v \in V$, com as formas bilinear $a(\cdot,\cdot)$ e linear $L(\cdot)$ dadas em (1.120) e (1.121).
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

$$a(u_h, v_h) = L(v_h), \quad \forall v_h \in V_h. \tag{1.125}$$

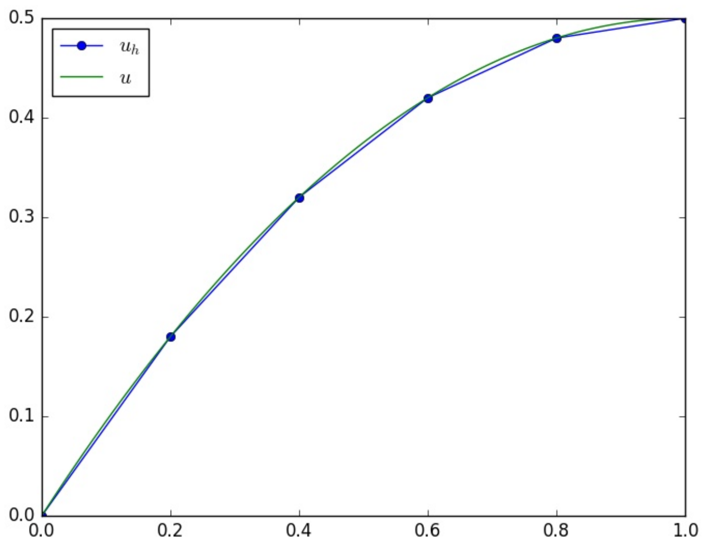

Figura 1.10: Esboço dos gráficos das soluções referentes ao Exemplo 1.3.2.

A Figura 1.10 apresenta o esboço dos gráficos da solução analítica u e da sua aproximação de elementos finitos u_h, esta construída no espaço dos polinômios lineares por partes sobre uma malha uniforme de 5 células.
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

#tolerance
tol=1e-14

# malha
mesh = IntervalMesh(5,0,1)

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

```
u_D = Constant(0.0)

def boundary(x,on_boundary):
    return near(x[0],0,tol)

bc = DirichletBC(V,u_D,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)

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

#erro L2
print("Erro L2: %1.2E\n" % errornorm(u,ua,norm_type="L2"))

plot(u,mesh=mesh,marker='o',label=r"$u_h$")
mesh = IntervalMesh(100,0,1)
plot(ua,mesh=mesh,label=r"$u$")
plt.legend(numpoints=1,loc='upper left')
plt.show()
```

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

$$-u'' = f, \quad \forall x \in I = [0, L], \tag{1.126}$$

$$u(0) = u_0, \quad u'(L) = \alpha, \tag{1.127}$$

com u_0, α e f dados.

Tomando uma função teste $v \in V := \{v \in H^1(I);\ v(0) = 0\}$ e multiplicando-a em (1.126), obtemos

$$-\int_I u''v\,dx = \int_I fv\,dx. \tag{1.128}$$

Aplicando a integração por partes, temos

$$\int_I u'v'\,dx - \alpha v(L) = \int_I fc\,dx. \tag{1.129}$$

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

$$a(u,v) - b(= L(v), \quad \forall v \in V, \tag{1.130}$$

onde $a(u,v)$ é a forma bilinear

$$a(u,v) = \int_I u'v'\,dx \tag{1.131}$$

e $L(v)$ é a forma linear

$$L(v) = \int_I fv\,dx + \alpha v(L). \tag{1.132}$$

Exemplo 1.3.3. Consideremos o problema

$$-u'' = 1, \quad x \in I = [0,1], \tag{1.133}$$
$$u(0) = 0, \quad u'(1) = 1. \tag{1.134}$$

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

$$a(u,v) = L(v), \quad \forall v \in V, \tag{1.135}$$

com

$$a(u,v) = \int_I u'v'\,dx \tag{1.136}$$

e

$$L(v) = \int_I fv\,dx + 1 \cdot v(1). \tag{1.137}$$

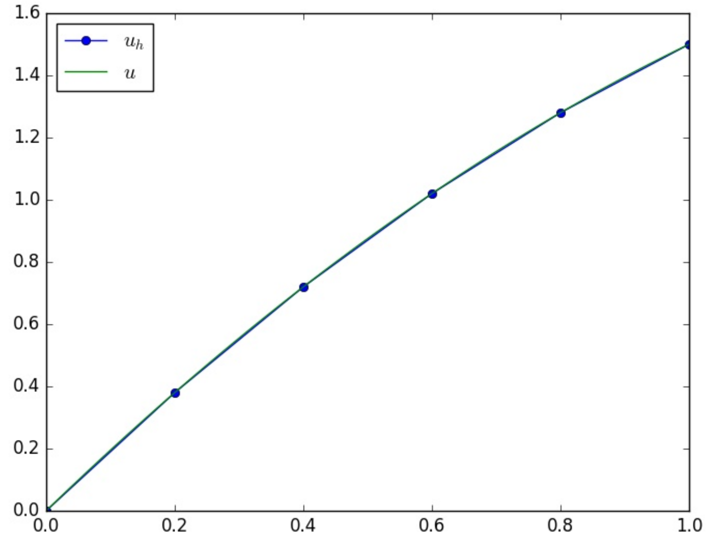

Figura 1.11: Esboço dos gráficos das soluções referentes ao Exemplo 1.3.3.

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

$$a(u_h, v_h) = L(v_h), \quad \forall v_h \in V_h. \tag{1.138}$$

A Figura 1.11 apresenta o esboço dos gráficos da solução analítica u e da sua aproximação de elementos finitos u_h, esta construída no espaço dos polinômios lineares por partes sobre uma malha uniforme de 5 células.
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

#tolerance
tol=1e-14
```

```
# malha
mesh = IntervalMesh(5,0,1)

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

u_D = Constant(0.0)

def boundary_D(x,on_boundary):
    return near(x[0],0,tol)

bc = DirichletBC(V,u_D,boundary_D)

#MEF problem
u = TrialFunction(V)
v = TestFunction(V)
f = Constant(1.0)
a = u.dx(0)*v.dx(0)*dx
L = f*v*dx + 1*v*ds

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

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

#erro L2
print("Erro L2: %1.2E\n" % errornorm(u,ua,norm_type="L2"))

plot(u,mesh=mesh,marker='o',label=r"$u_h$")
mesh = IntervalMesh(100,0,1)
plot(ua,mesh=mesh,label=r"$u$")
plt.legend(numpoints=1,loc='upper left')
plt.show()
```

1.3.3 Condições de Robin

Consideremos o seguinte problema com condições de contorno de Robin[6]: encontrar u tal que

$$-u'' = f, \quad \forall x \in I = [0, L], \tag{1.139}$$
$$u'(0) = r_0(u(0) - s_0), \ -u'(L) = r_L(u(L) - s_L), \tag{1.140}$$

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 (1.139), obtemos

$$-\int_I u''v\,dx = \int_I fv\,dx. \tag{1.141}$$

Aplicando a integração por partes, temos

$$\int_I u'v'\,dx - \underbrace{u'(L)v(L)}_{-u'(L)=r_L(u(L)-s_L)} + \underbrace{u'(0)v(0)}_{u'(0)=r_0(u(0)-s_0)} = \int_I fc\,dx. \tag{1.142}$$

ou, mais adequadamente,

$$\int_I u'v'\,dx + r_L u(L)v(L) + r_0 u(0)v(0) = \int_I fc\,dx + r_L s_L v(L) + r_0 s_0 v(0). \tag{1.143}$$

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

$$a(u,v) = L(v), \quad \forall v \in V, \tag{1.144}$$

onde $a(u,v)$ é a forma bilinear

$$a(u,v) = \int_I u'v'\,dx + r_L u(L)v(L) + r_0 u(0)v(0) \tag{1.145}$$

e $L(v)$ é a forma linear

$$L(v) = \int_I fv\,dx + r_L s_L v(L) + r_0 s_0 v(0). \tag{1.146}$$

Exemplo 1.3.4. Consideremos o problema

$$-u'' = 1, \quad x \in I = [0,1], \tag{1.147}$$
$$u'(0) = u(0), \quad -u'(1) = u(1) - 1. \tag{1.148}$$

[6]Victor Gustave Robin, 1855 - 1897, matemático francês. Fonte: Wikipedia.

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

$$a(u, v) = L(v), \quad \forall v \in V, \tag{1.149}$$

onde

$$a(u, v) = \int_I u'v'\,dx + u(1)v(1) + u(0)v(0) \tag{1.150}$$

e

$$L(v) = \int_I fv\,dx + 1 \cdot v(1). \tag{1.151}$$

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

$$a(u_h, v_h) = L(v_h), \quad \forall v_h \in V_h. \tag{1.152}$$

A Figura 1.12 apresenta o esboço dos gráficos da solução analítica u e da sua aproximação de elementos finitos u_h, esta construída no espaço dos polinômios lineares por partes sobre uma malha uniforme de 5 células.

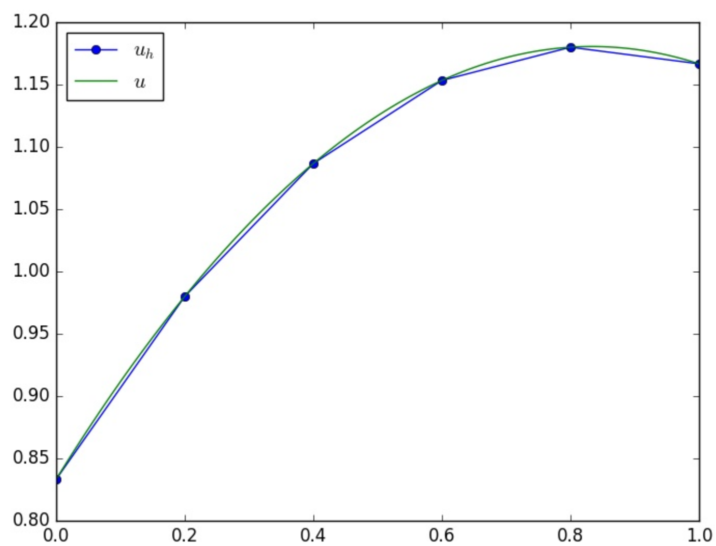

Figura 1.12: Esboço dos gráficos das soluções referentes ao Exemplo 1.3.4.

Com o FENiCS, a computação do problema de elementos finitos pode ser feita com o seguinte código:

```python
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(5,0,1)

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

#marcadores de fronteiras
boundary_markers = FacetFunction('size_t', mesh)

class BoundaryX0(SubDomain):
    def inside(self, x, on_boundary):
        return on_boundary and near(x[0], 0, tol)
bx0 = BoundaryX0()
bx0.mark(boundary_markers, 0)

class BoundaryX1(SubDomain):
    def inside(self, x, on_boundary):
        return on_boundary and near(x[0], 1, tol)
bx1 = BoundaryX1()
bx1.mark(boundary_markers, 1)

ds = Measure('ds', domain=mesh, \
                 subdomain_data=boundary_markers)

#MEF problem
u = TrialFunction(V)
v = TestFunction(V)
f = Constant(1.0)
a = u.dx(0)*v.dx(0)*dx + u*v*ds(1) + u*v*ds(0)
L = f*v*dx + 1*v*ds(1)
```

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

#sol analitica
ua = Expression('-x[0]*x[0]/2+5*x[0]/6+5./6',
                degree=2)

#erro L2
print("Erro L2: %1.2E\n" % errornorm(u,ua,norm_type="L2"))

plot(u,mesh=mesh,marker='o',label=r"$u_h$")
mesh = IntervalMesh(100,0,1)
plot(ua,mesh=mesh,label=r"$u$")
plt.legend(numpoints=1,loc='upper left')
plt.show()
```

Exercícios

E 1.3.1. Considere o problema

$$-u'' + u' + 2u = -\cos(x), \quad x \in (0, \pi/2), \tag{1.153}$$

$$u(0) = -0{,}3, \quad u(\pi/2) = -0{,}1. \tag{1.154}$$

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 $u(x) = 0{,}1(\mathrm{sen}(x) + 3\cos(x))$, bem como, compute o erro $\|u - u_h\|_{L^2}$.

1.4 Malhas auto-adaptativas

Retornemos ao problema modelo (1.56)-(1.57)

$$-u'' = f, \quad x \in I = [0,L], \tag{1.155}$$

$$u(0) = u(L) = 0. \tag{1.156}$$

A estimativa *a posteriori* dada no Teorema 1.2.4, 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

$$\eta_i(u_h) = h_i\|f - u_h''\|_{L^2(I_i)}, \tag{1.157}$$

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

$$\eta_i(u_h) > \alpha \max_{j=1,2,\ldots,n} \eta_j(u_h), \tag{1.158}$$

onde escolhemos $0 < \alpha < 1$.

Exemplo 1.4.1. Consideremos o problema

$$-u'' = e^{-100|x-\frac{1}{2}|}, \quad x \in I = [0,1], \tag{1.159}$$

$$u(0) = u(1) = 0. \tag{1.160}$$

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 (1.158) com $\alpha = 0{,}5$. A Figura 1.13 apresenta o esboço do gráfico da solução de elementos finitos na malha mais refinada. Além disso, na Tabela 1.1 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:

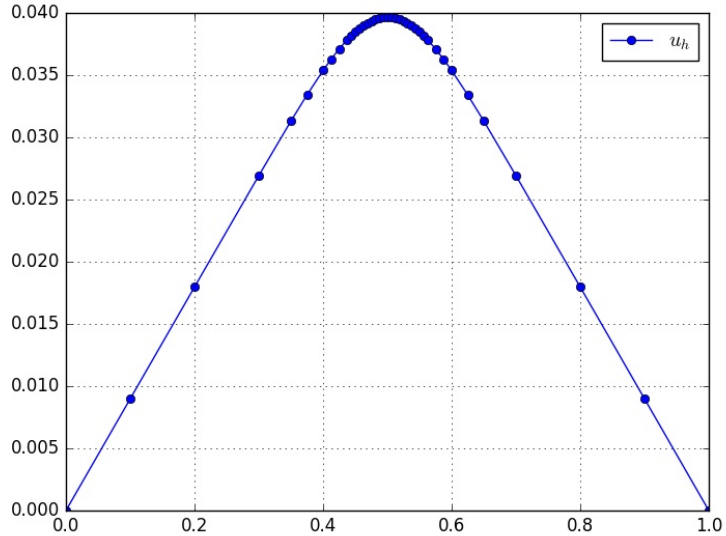

Figura 1.13: Esboço dos gráficos das soluções referentes ao Exemplo 1.4.1.

#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 1.4.1.

```
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)
```

Exercícios

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

1.5 Seleção de aplicações

1.5.1 Sistemas de equações

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

$$-u_0'' + u_1 = f_0, \forall x \in (0, L) \tag{1.161}$$

$$-u_1'' + u_0 = f_1, \forall x \in (0, L) \tag{1.162}$$

$$u_0(0) = u_{00}, \quad u_0(L) = u_{0L}, \tag{1.163}$$

$$u_1(0) = u_{10}, \quad u_1(L) = u_{1L}, \tag{1.164}$$

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

$$a(u, v) = L(v), \forall v = (v_0, v_1) \in V \times V, \tag{1.165}$$

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

$$a(u, v) = \int_I u_0' v_0' \, dx + \int_I u_1' v_1' \, dx + \int_I u_0 v_0 \, dx + \int_I u_1 v_1 \, dx \tag{1.166}$$

e a forma linear é

$$L(v) = \int_I f_0 v_0 \, dx + \int_I f_1 v_1 \, dx. \tag{1.167}$$

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

$$a(u_h, v_h) = L(v_h), \forall v_h = (v_{h0}, v_{h1}) \in V_h \times V_h, \tag{1.168}$$

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 1.5.1. Consideremos o seguinte problema de valor de contorno

$$-u_0'' + u_1 = \operatorname{sen}(x) + \cos(x), \forall x \in (-\pi, \pi) \tag{1.169}$$

$$-u_1'' + u_0 = \cos(x) - \sin(x), \forall x \in (-\pi, \pi) \tag{1.170}$$

$$u_0(-\pi) = 0, \quad u_0(\pi) = 0, \tag{1.171}$$

$$u_1(-\pi) = -1, \quad u_1(\pi) = -1. \tag{1.172}$$

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

$$a(u_h, v_h) = L(v_h), \forall v_h = (v_{h0}, v_{h1}) \in V_h \times V_h, \tag{1.173}$$

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 (1.166) e (1.167), respectivamente.

A Figura 1.14 apresenta o esboço dos gráficos das soluções analíticas $u_0(x) = \operatorname{sen}(x)$ 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.

Com o FENiCS, a computação do problema de elementos finitos pode ser feita com o seguinte código:

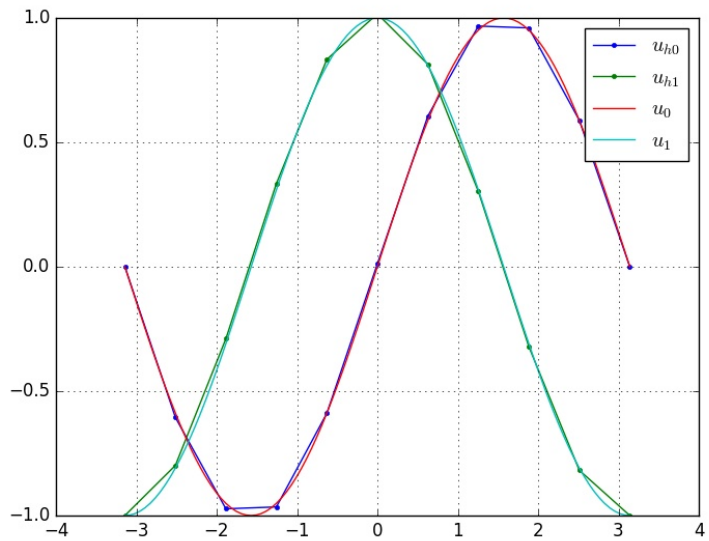

Figura 1.14: Esboço dos gráficos das soluções referentes ao Exemplo 1.5.1.

```
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()
```

Exercícios

Em construção ...

Capítulo 2

Método de elementos finitos em 2D

2.1 Malha e espaço

2.1.1 Malha

Seja $\Omega \subset \mathbb{R}^2$ um domínio limitado com fronteira $\partial\Omega$ suave e poligonal. Uma malha (ou triangularização) $\mathcal{K}$ de Ω 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 triangulo 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 circulo 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

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

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

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

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

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

$$\lambda_j(N_i) = \left\{ \begin{array}{ll} 1 & , i = j, \\ 0 & , i \neq j \end{array} \right. , \; i,j = 0,1,2. \tag{2.3}$$

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

$$v = \alpha_0\lambda_0 + \alpha_1\lambda_1 + \alpha_2\lambda_2, \tag{2.4}$$

onde $\alpha_i = v(N_i)$.

2.1.3 Espaço contínuo dos polinômios lineares por partes

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

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

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 $v_2 \in P_2(K_2)$, 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 nodas $\{\varphi_j\}_{j=1}^{n_p} \subset V_h$ tal que

$$\varphi_j(N_i) = \left\{ \begin{array}{ll} 1 & ,i = j \\ 0 & ,i \neq j \end{array} \right. ,\ i,j = 0, 1, \ldots, n_p - 1. \tag{2.6}$$

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

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

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

Exemplo 2.1.1. A Figura 2.1 mostra o esboço de uma malha triangular no domínio $D = [0, 1] \times [0, 1]$.

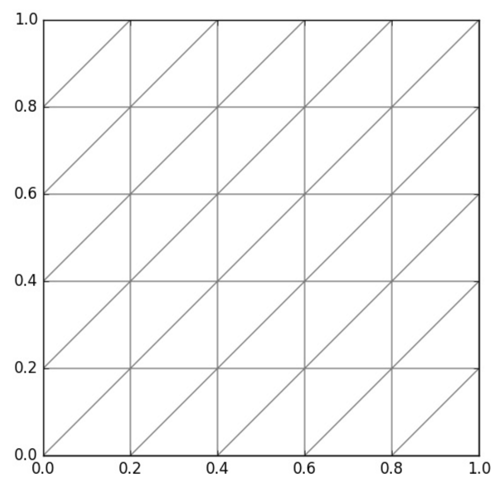

Figura 2.1: Esboço de uma malha triangular no domínio $D = [0, 1] \times [0, 1]$.

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)

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

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

2.2 Interpolação e projeção

2.2.1 Interpolação

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

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

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

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

e da derivada total de segunda ordem

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

Proposição 2.2.1. (Erro da interpolação no espaço linear) A interpolação πf satisfaz as seguintes estimativas

$$\|f - \pi f\|_{L^2(K)} \leq C h_K^2 \|D^2 f\|_{L^2(K)}, \tag{2.11}$$
$$\|D(f - \pi f)\|_{L^2(K)} \leq C h_K \|D^2 f\|_{L^2(K)}. \tag{2.12}$$

Demonstração. Veja [1, Capítulo 4]. □

Observação 2.2.1. A constante C dependo do inverso de $\operatorname{sen}(\theta_K)$ onde θ_K é o menor angulo de K. Desta forma, para um triângulo com θ_K muito pequeno, as estimativas (2.11) e (2.12) 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 Ω é denotada também por $\pi f \in V_h$ e definida por

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

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

$$\|f - \pi f\|_{L^2(\Omega)}^2 \leq C \sum_{K \in \mathcal{K}} h_K^4 \|D^2 f\|_{L^2(K)}^2, \tag{2.14}$$
$$\|D(f - \pi f)\|_{L^2(\Omega)}^2 \leq C \sum_{K \in \mathcal{K}} h_K^2 \|D^2 f\|_{L^2(K)}^2,. \tag{2.15}$$

Demonstração. Demonstração análoga a Proposição 1.1.2. □

Exemplo 2.2.1. Consideremos a função $f(x_0,x_1) = \operatorname{sen}(\pi x_0)\cos(\pi x_1)$ 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)

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

2.2.2 Projeção L^2

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

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

Analogamente a projeção em uma dimensão (veja Subseção 1.1.2), a projeção

$$P_hf = \sum_{j=0}^{n_p-1} \xi_j\varphi_j, \tag{2.17}$$

onde ξ_j satisfaz o sistema linear

$$M\boldsymbol{\xi} = \boldsymbol{b}, \tag{2.18}$$

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

$$m_{i,j} = \int_\Omega \varphi_i\varphi_j\,dx \tag{2.19}$$

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

$$b_i = \int_\Omega f\varphi_i\,dx. \tag{2.20}$$

Também, vale o resultado análogo da melhor aproximação (veja 1.1.1), i.e.

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

E, portanto, também temos a estimativa análoga para o erro de projeção (veja 1.1.2)

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

Tomando o tamanho global da malha, temos

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

Exemplo 2.2.2. Consideremos a função $f(x_0,x_1) = \operatorname{sen}(\pi x_0)\cos(\pi x_1)$ 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)

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

Exercícios

E 2.2.1. Verifique computacionalmente a Proposição 2.2.2 no caso da função $f(x_0,x_1) = \text{sen}(\pi x_0)\cos(\pi x_1)$ interpolada sobre uma malha triangular uniforme sobre o domínio $D = [0, 1] \times [0, 1]$.

E 2.2.2. Verifique computacionalmente a estimativa (2.23) no caso da função $f(x_0,x_1) = \text{sen}(\pi x_0)\cos(\pi x_1)$ projetada sobre uma malha triangular uniforme sobre o domínio $D = [0, 1] \times [0, 1]$.

2.3 Problema modelo

Nesta seção, apresentaremos a aplicação do método de elementos finitos para a equação de Poisson[1] com condições de Dirichlet[2], i.e.: encontrar u tal que

$$-\Delta u = f,\ x \in \Omega, \tag{2.24}$$
$$u = 0,\ x \in \partial\Omega, \tag{2.25}$$

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

2.3.1 Formulação variacional

A aplicação do método de elementos finitos é construída sobre a formulação fraca do problema (2.24)-(2.25). Para obtermos esta, multiplicamos (2.24) por uma função teste v em um espaço adequado V_0 e integramos no domínio Ω, i.e.

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

Então, usando a fórmula de Green[4], obtemos

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

[1]Siméon Denis Poisson, 1781 - 1840, matemático francês. Fonte: Wikipedia.

[2]Johann Peter Gustav Lejeune Dirichlet, 1805 - 1859, matemático alemão. Fonte: Wikipedia.

[3]Pierre-Simon, marquis de Laplace, 1749 - 1827, matemático francês. Fonte: Wikipedia.

[4]George Green, 1793 - 1841, matemático britânico. Fonte: Wikipedia.

Então, observando critérios de regularidade e a condição de contorno (2.25), escolhemos

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

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 (2.24)-(2.25): encontrar $u \in V_0$ tal que

$$a(u,v) = L(v),\ \forall v \in V_0, \tag{2.29}$$

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

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

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

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

2.3.2 Formulação de elementos finitos

A formulação de elementos finitos é obtida da formulação fraca (2.29) pela aproximação do espaço teste V_0 por uma espaço de dimensão finita. Tomando uma triangulação $\mathcal{K} \subset \Omega$ e considerando o espaço contínuo dos polinômios lineares por partes

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

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 (2.29): encontrar $u_h \in V_{h,0}$ tal que

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

Observemos que (2.33) é equivalente ao problema de encontrar $u_h \in V_{h,0}$ tal que

$$a(u_h,\varphi_i) = L(\varphi_i), \tag{2.34}$$

com $i = 0, 1, \cdots, n_p - 1$, 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

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

temos

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

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

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

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

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

$$A\boldsymbol{\xi} = \boldsymbol{b}, \tag{2.39}$$

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

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

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

$$b_i = L(\varphi_i). \tag{2.41}$$

Exemplo 2.3.1. Consideremos o seguinte problema de Poisson

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

$$u = 0,\ x \in \partial\Omega. \tag{2.43}$$

Na Figura 2.2 temos um esboço da aproximação de elementos finitos obtida em uma malha uniforme com 20×20 nodos. As isolinhas correspondem aos ponto tais que $u = 3 \times 10^{-1}, 2 \times 10^{-1}$, 10^{-1}, 5×10^{-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
```

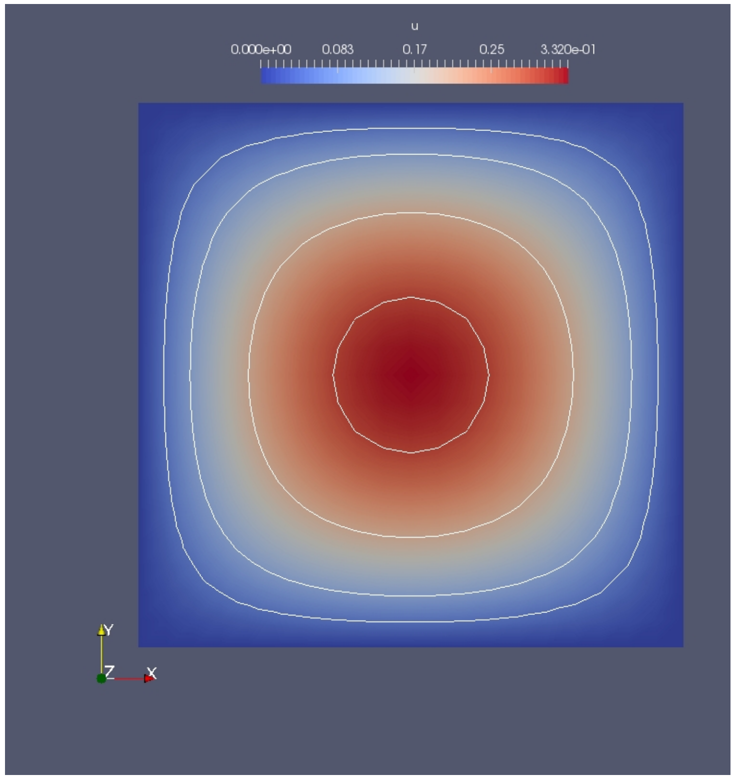

Figura 2.2: Esboço da solução de elementos finitos do problema discutido no Exemplo 2.3.1.

```
# malha
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)

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

Exercícios

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

$$-\Delta u = 10,\ x \in (0,1) \times (0,1) \tag{2.44}$$
$$u(x{,}0) = 0,\ 0 \leq x \leq 1, \tag{2.45}$$
$$u(1{,}y) = 0,\ 0 \leq y < 1, \tag{2.46}$$
$$u(x{,}1) = 1,\ 0 \leq x \leq 1, \tag{2.47}$$
$$u(0{,}y) = 1,\ 0 < x \leq 1. \tag{2.48}$$

2.4 Fundamentos da análise de elementos finitos

2.4.1 Existência e unicidade

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

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

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

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

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

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

Portanto, $\boldsymbol{\xi}^T A\boldsymbol{\xi} \geq 0$ 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 $\boldsymbol{\xi} = 0$, o que é uma contradição. Logo, $\boldsymbol{\xi}^T A\boldsymbol{\xi} > 0$ para todo $\boldsymbol{\xi} \in \mathbb{R}^{n_i}$, $\boldsymbol{\xi} \neq 0$. □

Teorema 2.4.2. (Existência e unicidade) O problema de elementos finitos (2.33) tem solução única.

Demonstração. O problema de elementos finitos (2.33) se resume a resolver o sistema linear $A\boldsymbol{\xi} = \boldsymbol{b}$. Do Teorema 2.4.1, temos que A é uma matriz definida positiva e, portanto, invertível. Daí segue, imediatamente, que o problema (2.33) tem solução única. □

2.4.2 Estimativa *a priori* do erro

Teorema 2.4.3. (Ortogonalidade de Galerkin) A solução u_h do problema de elementos finitos (2.33) satisfaz

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

onde u é a solução do problema fraco (2.29).

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

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

bem como

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

□

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

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

para todo $v \in V_0$.

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

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

Demonstração. Observando que $u - u_h = u - v_h + v_h - u_h$ e usando a ortogonalidade de Galerkin (Teorema 2.4.3), temos:

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

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

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

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

$$= |||u - u_h|||^2 |||u - v_h|||. \tag{2.62}$$

□

Teorema 2.4.5. (Estimativa *a priori* do erro.) A solução u_h do problema de elementos finitos (2.33) satisfaz

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

Demonstração. O resultado segue do Teorema da melhor aproximação (Teorema 2.4.4) e da estimativa do erro de interpolação (Proposição 2.2.2), pois

$$\begin{align} |||u - u_h|||^2 &\leq |||u - \pi u|||^2 \tag{2.64}\\ &= \|D(u - \pi u)\|^2_{L^2(\Omega)} \tag{2.65}\\ &\leq C \sum_{K \in \mathcal{K}} h_K^2 \|D^2 u\|^2_{L^2(\Omega)}. \tag{2.66} \end{align}$$

□

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

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

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

$$\begin{align} \|v\|^2_{L^2(\Omega)} &= \int_\Omega v^2\,dx \tag{2.68}\\ &= -\int_\Omega v^2 \Delta\phi\,dx. \tag{2.69} \end{align}$$

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

$$\begin{align} \|v\|^2_{L^2(\Omega)} &= -\int_{\partial\Omega} v^2 n \cdot \nabla\phi\,ds + \int_\Omega \nabla v^2 \cdot \nabla\phi\,dx \tag{2.70}\\ &= \int_\Omega 2v\nabla v \cdot \nabla\phi\,dx \tag{2.71}\\ &\leq \sup_{x\in\Omega} |\nabla\phi| \|v\|_{L^2(\Omega)} \|\nabla v\|_{L^2(\Omega)}. \tag{2.72} \end{align}$$

□

Com a desigualdade de Poincaré e da estimativa *a priori* do erro (Teorema 2.4.5), temos

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

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 (2.33) satisfaz

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

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

$$-\Delta\phi = e,\ \forall x\in\Omega \tag{2.75}$$
$$\phi = 0,\ \forall x\in\partial\Omega. \tag{2.76}$$

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

$$\|e^2\|_{L^2(\Omega)} = -\int_\Omega e\Delta\phi\,dx \tag{2.77}$$
$$= \int_\Omega \nabla e\cdot\nabla\phi\,dx - \int_{\partial\Omega} e\,n\cdot\nabla\phi\,ds \tag{2.78}$$
$$= \int_\Omega \nabla e\cdot\nabla(\phi-\pi\phi)\,dx \tag{2.79}$$
$$\leq \|\nabla e\|_{L^2(\Omega)}\|\nabla(\phi-\pi\phi)\|_{L^2(\Omega)}. \tag{2.80}$$

Da estimativa *a priori* (2.73) (que segue do Teorema 2.4.5) temos

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

Agora, da regularidade elíptica $\|D^2\phi\|_{L^2(\Omega)} \leq C\|\Delta\phi\|_{L^2(\Omega)}$ [2] e da estimativa do erro de interpolação (Proposição 2.2.2), temos

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

Então, temos

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

□

Exemplo 2.4.1. Consideremos o seguinte problema de Poisson

$$-\Delta u = -2(x_0^2 - x_0) - 2(x_1^2 - x_1),\ x\in\Omega := (0,1)\times(0,1), \tag{2.84}$$
$$u = 0,\ x\in\partial\Omega. \tag{2.85}$$

Tabela 2.1: Erros de aproximações por elementos finitos referente ao problema dado no Exemplo 2.4.1.

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

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 2.1 mostra os valores dos erros $\|u - u_h\|_{L^2(\Omega)}$ para diferentes valores de h.

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)

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

2.4.3 Estimativa *a posteriori*

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

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

Demonstração. Veja [4]. □

Teorema 2.4.9. (Estimativa *a posteriori*) A solução u_h do problema de elementos finitos (2.33) satisfaz

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

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

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

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

$$\begin{align}
|||e|||^2 &= \|\nabla e\|^2_{L^2(\Omega)} \tag{2.89}\\
&= \int_\Omega \nabla e\cdot\nabla e\,dx \tag{2.90}\\
&= \int_\Omega \nabla e\cdot\nabla(e-\pi e)\,dx. \tag{2.91}
\end{align}$$

Nesta última equação, temos usado a ortogonalidade de Galerkin (Teorema 2.4.3). Daí, temos

$$\begin{align}
\int_\Omega \nabla e\cdot\nabla(e-\pi e)\,dx &= \sum_{K\in\mathcal{K}}\int_K \nabla e\cdot\nabla(e-\pi e)\,dx \tag{2.92}\\
&= \sum_{K\in\mathcal{K}} -\int_K \Delta e(e-\pi e)\,dx \nonumber\\
&+ \int_{\partial K} n\cdot\nabla e(e-\pi e)\,ds, \tag{2.93}\\
&= \sum_{K\in\mathcal{K}}\int_K (f+\Delta u_h)(e-\pi e)\,dx \nonumber\\
&+ \int_{\partial K\setminus\partial\Omega} n\cdot\nabla e(e-\pi e)\,ds, \tag{2.94}
\end{align}$$

uma vez que $-\Delta e|_K = f + \Delta u_h|_K$ e, ambos, e e πe se anulam em $\partial\Omega$.
Para computarmos o segundo termo do lado direito da ultima equação, observamos que o erro em lado E recebe contribuições dos dois elementos $K^\pm$ que compartilham E. Com isso, temos

$$\begin{align}
\int_{\partial K^+\cap\partial K^-} n\cdot\nabla e(e-\pi e)\,ds = \int_E &(n^+\cdot\nabla e^+(e^+-\pi e^+) \nonumber\\
&+ n^-\cdot\nabla e^-(e^- - \pi e^-))\,ds, \tag{2.95}
\end{align}$$

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, ∇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

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

Com isso, temos

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

onde $\mathcal{E}_I$ é o conjunto dos lados interiores na triangularização $\mathcal{K}$. Logo, retornando a (2.94), obtemos

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

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.

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

$$\leq \|f + \Delta u_h\|_{L^2(\Omega)} C h_K \|De\|_{L^2(\Omega)} \tag{2.100}$$

Para estimarmos as contribuições dos lados, usamos a desigualdade do Traço [4]

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

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

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

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

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

Daí, a estimativa segue das (2.100) e (2.104). □

Resposta dos Exercícios

E 1.2.1. Código FENiCS.

E 1.3.1. Código.

E 1.4.1. Código.

Referências Bibliográficas

[1] S.C. Brenner and L.R. Scott. *The mathematical theory of finite element methods*. Springer, 2008.

[2] L.C. Evans. *Partial Differential Equations*. Graduate studies in mathematics. American Mathematical Society, 1998.

[3] H. P. Langtangen and A. Logg. *Solving PDEs in Python*. Springer, 2017.

[4] M.G. Larson and F. Bengson. *The Finite Element Method: Theory, Implementation, and Applications*. Springer, 2013.

Índice Remissivo