

Método de Elementos Finitos

Pedro H A Konzen

24 de setembro de 2018

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 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, 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

Capa	i
Licença	ii
Prefácio	iii
Sumário	iv
1 Método de elementos finitos em 1D	1
1.1 Interpolação e projeção	1
1.1.1 Interpolação	2
1.1.2 Projeção L^2	10
1.2 Problema modelo	15
1.2.1 Formulação fraca	15
1.2.2 Formulação de elementos finitos	16
1.2.3 Estimativa <i>a priori</i>	19
1.3 Condições de contorno	23
1.3.1 Condições de Dirichlet	23
1.3.2 Condições de Neumann	26
1.3.3 Condições de Robin	33
1.4 Sistemas de equações	37
1.5 Malha adaptativa	40
Respostas dos Exercícios	41
Referências Bibliográficas	42
Índice Remissivo	43

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

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

Observamos que dado $v \in P_1(I)$, temos que v é unicamente determinada pelos valores $\alpha_0 = v(x_0)$ 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

$$\varphi_j(x_i) = \begin{cases} 1 & , i = j, \\ 0 & , i \neq j \end{cases}, \quad (1.2)$$

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.

$$v(x) = \alpha_0\varphi_0(x) + \alpha_1\varphi_1(x). \quad (1.3)$$

Além disso, observamos que

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

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 $\mathcal{I} = \{l_0 = x_0, x_1, \dots, x_n = l_1\}$ e, portanto, com n subintervalos $I_i = [x_{i-1}, x_i]$ de comprimento $h_i = x_i - x_{i-1}$, $i = 1, 2, \dots, n$. Na malha \mathcal{I} definimos o seguinte espaço das funções lineares por partes

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

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

$$\varphi_j(x_i) = \begin{cases} 1 & , i = j, \\ 0 & , i \neq j \end{cases}, \quad (1.6)$$

com $i, j = 0, 1, \dots, n$. Podemos verificar que

$$\varphi_i(x) = \begin{cases} (x - x_{i-1})/h_i & , x \in I_i, \\ (x_{i+1} - x)/h_{i+1} & , x \in I_{i+1}, \\ 0 & , \text{noutros casos} \end{cases} \quad (1.7)$$

veja, Figura 1.1. É notável que $\phi_i(x)$ 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** $\pi : C^0(I) \rightarrow P_1(I)$ por

$$\pi f(x) = f(x_0)\varphi_0(x) + f(x_1)\varphi_1(x). \quad (1.8)$$

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 $f(x) = 3 \sin(2\pi x)$ no espaço $P_1([1/4, 3/4])$. Neste caso

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

Com o [FENiCS](#), podemos computar a função interpolada πf com o seguinte código:

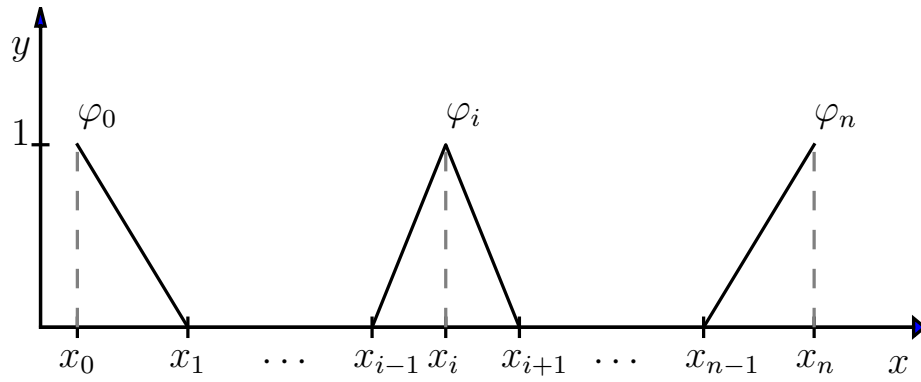


Figura 1.1: Base nodal para o espaço das funções lineares por parte.

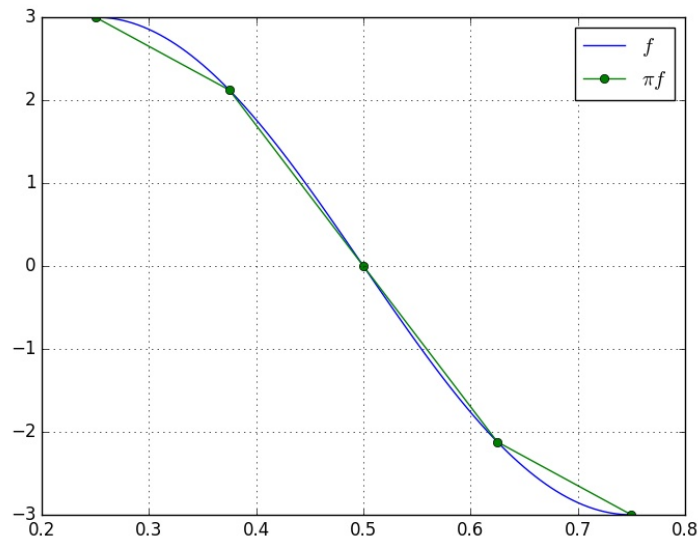


Figura 1.2: Interpolação linear de $f(x) = 3 \sin(2\pi x)$ no espaço $P_1([1/4, 3/4])$.

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

```

import matplotlib.pyplot as plt

# malha
mesh = IntervalMesh(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()

```

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

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

Lembramos que valem as desigualdades triangular

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

e a de Cauchy-Schwarz¹

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

¹Também 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

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

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

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

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

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

$$e(y) = \int_{x_0}^y e' dx \quad (1.16)$$

$$\leq \int_{x_0}^y |e'| dx \quad (1.17)$$

$$\leq \int_I 1 \cdot |e'| dx \quad (1.18)$$

$$\leq \left(\int_I 1^2 dx \right)^{1/2} \left(\int_I e'^2 dx \right)^{1/2} \quad (1.19)$$

$$= h^{1/2} \left(\int_I e'^2 dx \right)^{1/2}, \quad (1.20)$$

donde

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

Então, integrando em I obtemos

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

ou seja,

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

Agora, observando que $e(x_0) = e(x_1) = 0$, o teorema de Rolle² 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

$$e'(y) = e'(\tilde{x}) + \int_{\tilde{x}}^y e'' dx \quad (1.24)$$

$$= \int_{\tilde{x}}^y e'' dx \quad (1.25)$$

$$\leq \int_I 1 \cdot |e''| dx \quad (1.26)$$

$$\leq h^{1/2} \left(\int_I e''^2 \right)^{1/2}. \quad (1.27)$$

Então, integrando em I , obtemos

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

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

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

Por fim, de (1.23) e de (1.29), obtemos a primeira estimativa desejada

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

□

Exemplo 1.1.2. A Figura 1.3 mostra a evolução do erro na norma L^2 da interpolação de $f(x) = 3 \sin(2\pi x)$ no espaço $P_1([0, h])$ para $h = 10^{-5}, 10^{-4}, \dots, 10^{-1}$. 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)
```

²Michel Rolle, 1652 - 1719, matemático francês.

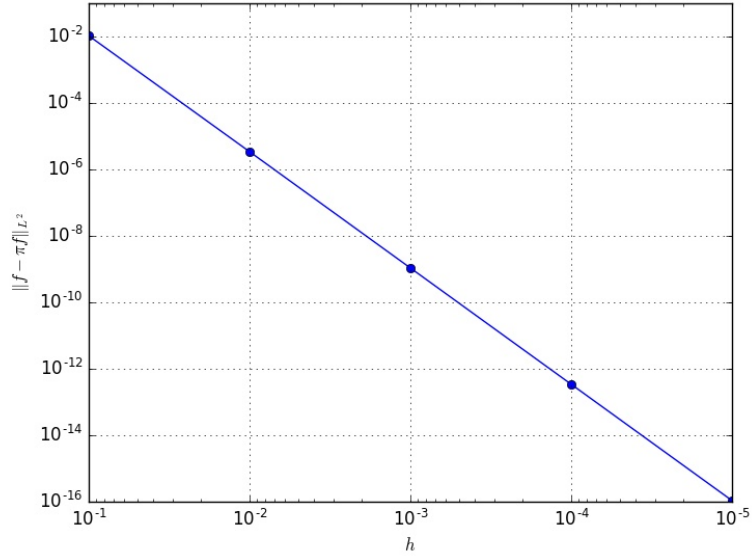


Figura 1.3: Erro de interpolação de $f(x) = 3\sin(2\pi x)$ no espaço $P_1([0, h])$.

```
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 $\pi : C^0(I) \rightarrow V_h$ na malha \mathcal{I} de I por

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

Exemplo 1.1.3. A Figura 1.4 ilustra a interpolação 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 pontos).

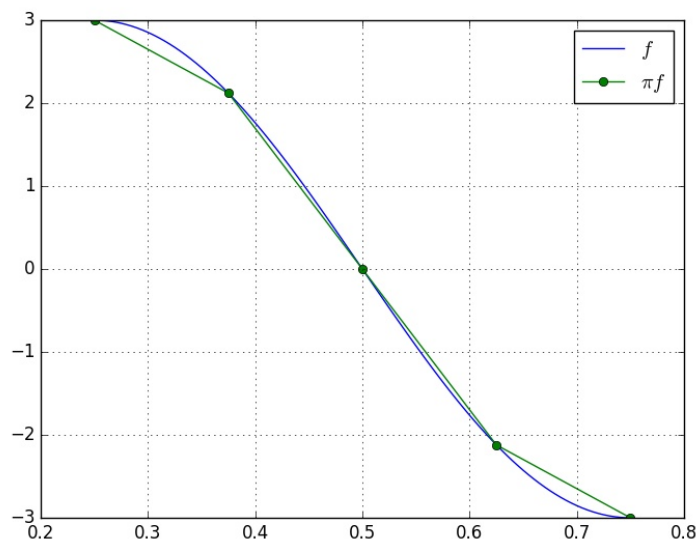


Figura 1.4: Interpolação linear de $f(x) = 3 \sin(2\pi x)$ no espaço V_h das funções lineares por partes sobre uma malha com 5 pontos.

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*

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

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

$$(1.34)$$

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

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

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

□

1.1.2 Projeção L^2

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

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

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

$$\int_I (f - P_h f) \varphi_i \, dx = 0, \quad i = 0, 1, \dots, n, \quad (1.38)$$

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

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

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

$$\int_I (f - P_h f) \varphi_i \, dx = 0 \Leftrightarrow \int_I f \varphi_i \, dx = \int_I P_h f \varphi_i \, dx \quad (1.40)$$

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

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

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

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

$$M \xi = b, \quad (1.43)$$

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

$$m_{i,j} = \int_I \varphi_j \varphi_i \, dx \quad (1.44)$$

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

$$b_i = \int_I f \varphi_i \, dx. \quad (1.45)$$

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

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

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 $f(x) = 3 \sin(2\pi x)$ 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 pontos).

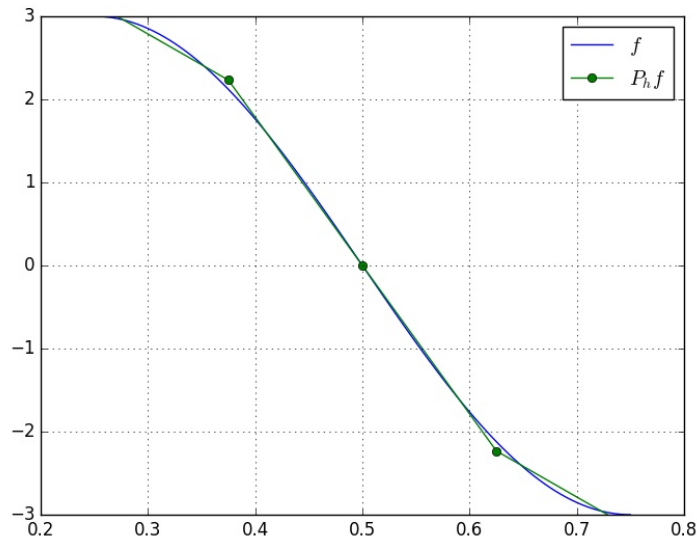


Figura 1.5: Projeção L^2 de $f(x) = 3 \sin(2\pi x)$ no espaço V_h das funções lineares por partes sobre uma malha com 5 pontos.

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 $P_h f$ 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

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

Demonstração. Dado $v \in V_h$, temos

$$\|f - P_h f\|_{L^2(I)}^2 = \int_I |f - P_h f|^2 dx \quad (1.48)$$

$$= \int_I (f - P_h f)(f - v + v - P_h f) dx \quad (1.49)$$

$$= \int_I (f - P_h f)(f - v) dx + \int_I (f - P_h f)(v - P_h f) dx \quad (1.50)$$

$$= \int_I (f - P_h f)(f - v) dx \quad (1.51)$$

$$\leq \|f - P_h f\|_{L^2(I)} \|f - v\|_{L^2(I)}, \quad (1.52)$$

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*

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

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

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

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

□

Exemplo 1.1.5. A Figura 1.6 mostra a evolução do erro na norma L^2 da projeção de $f(x) = 3 \sin(2\pi x)$ no espaço V_h em malhas uniformes de $h = 10^{-5}, 10^{-4}, \dots, 10^{-1}$ 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))
```

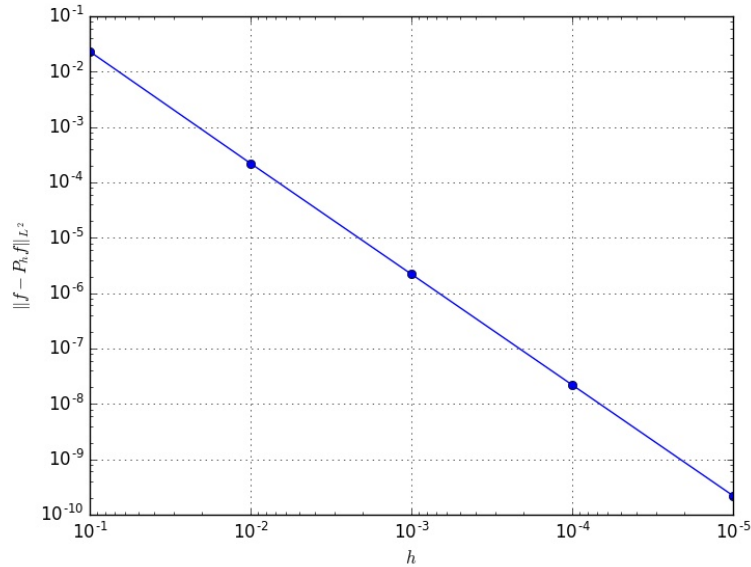


Figura 1.6: Erro de interpolação de $f(x) = 3 \sin(2\pi x)$ no espaço V_h .

Exercícios

E 1.1.1. Faça um código para verificar a segunda estimativa da Proposição 1.1.1 no caso da interpolação da função $f(x) = 3 \sin(2\pi x)$ 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 $f(x) = 3 \sin(2\pi x)$ 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 $[0, \pi]$. Faça o esboço dos gráficos de f e $P_h f$ e compute o erro $\|f - P_h f\|_{L^2}$.

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

$$-u'' = f, \quad x \in I = [0, L], \quad (1.56)$$

$$u(0) = u(L) = 0, \quad (1.57)$$

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

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

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.

$$\int_I f v \, dx = - \int_I u'' v \, dx \quad (1.59)$$

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

$$(1.61)$$

Donde, das condições de contorno, temos

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

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

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

onde

$$a(u, v) = \int_I u' v' \, dx \quad (1.64)$$

$$L(v) = \int_I f v \, dx, \quad (1.65)$$

são chamadas de forma bilinear e forma linear, respectivamente.

1.2.2 Formulação de elementos finitos

Uma formulação de elementos finitos é uma 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.

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

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

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

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 Galerkin³.

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

$$a(u_h, \varphi_i) = L(\varphi_i), \quad i = 1, \dots, n-1, \quad (1.68)$$

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

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

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

Agora, da forma bilinear (1.64), temos

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

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

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

$$A\xi = b, \quad (1.72)$$

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

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

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

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

$$b_i = L(\varphi_i) = \int_I \varphi_i dx. \quad (1.74)$$

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

$$-u'' = 1, \quad x \in I = [0,1], \quad (1.75)$$

$$u(0) = u(1) = 0. \quad (1.76)$$

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

$$a(u, v) = L(v), \quad (1.77)$$

onde

$$a(u, v) = \int_I u' v' dx, \quad L(v) = \int_I f v dx. \quad (1.78)$$

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

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

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 orthogonalidade

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

onde u é a solução de (1.63).

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

$$a(u, v) = L(v) = a(u_h, v) \Rightarrow a(u - u_h, v) = 0, \quad (1.81)$$

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

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

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

$$\|(u - u_h)'\|^2 = \int_I (u - u_h)'(u - u_h)' dx \quad (1.83)$$

$$= \int_I (u - u_h)'(u - v + v - u_h)' dx \quad (1.84)$$

$$= \int_I (u - u_h)'(u - v)' dx + \int_I (u - u_h)'(v - u_h)' dx \quad (1.85)$$

$$= \int_I (u - u_h)'(u - v)' dx \quad (1.86)$$

$$\leq \|(u - u_h)'\|_{L^2(I)} \|(u - v)'\|_{L^2(I)}. \quad (1.87)$$

□

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*

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

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

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

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

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

□

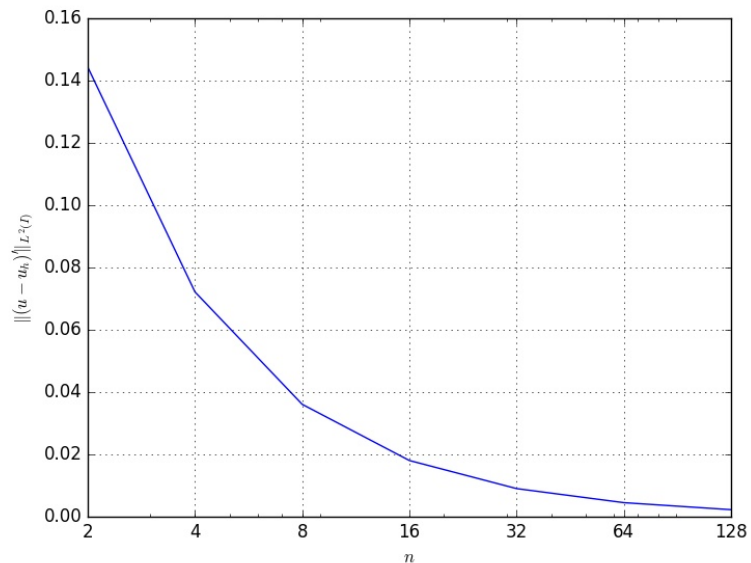


Figura 1.8: Esboço dos gráficos das soluções referentes ao Exemplo 1.2.2.

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 $n = 2, 4, 8, \dots, 128$ 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

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

return u

#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 = solver(n)
    e = errornorm(u,ua,norm_type='H10')
    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()

```

Exercícios

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

$$-u'' + u = 2 \operatorname{sen} x, \quad \forall x \in (-\pi, \pi), \quad (1.91)$$

$$u(-\pi) = u(\pi) = 0. \quad (1.92)$$

1.3 Condições de contorno

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

$$-u'' = f, \quad x \in I = [0, L], \quad (1.93)$$

com diferentes condições de contorno.

1.3.1 Condições de Dirichlet

Consideremos o seguinte problema com condições de contorno de Dirichlet⁴: encontrar u tal que

$$-u'' = f, \quad \forall x \in I = [0, L], \quad (1.94)$$

$$u(0) = u_0, \quad u(L) = u_L, \quad (1.95)$$

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 (1.94), obtemos

$$-\int_I u'' v \, dx = \int_I f v \, dx. \quad (1.96)$$

Aplicando a integração por partes, temos

$$\int_I u' v' \, dx = \int_I f v \, dx. \quad (1.97)$$

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

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

$$a(u, v) = L(v), \quad \forall v \in V_0, \quad (1.98)$$

onde $a(u, v)$ é a forma bilinear

$$a(u, v) = \int_I u' v' dx \quad (1.99)$$

e $L(v)$ é a forma linear

$$L(v) = \int_I f v dx. \quad (1.100)$$

Exemplo 1.3.1. Consideremos o problema

$$-u'' = 1, \quad x \in I = [0, 1], \quad (1.101)$$

$$u(0) = 1/2, \quad u(1) = 1. \quad (1.102)$$

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

$$a(u, v) = L(v), \quad (1.103)$$

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

$$a(u, v) = \int_I u' v' dx, \quad L(v) = \int_I f v dx. \quad (1.104)$$

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

$$a(u_h, v_h) = L(v_h), \quad (1.105)$$

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](#):

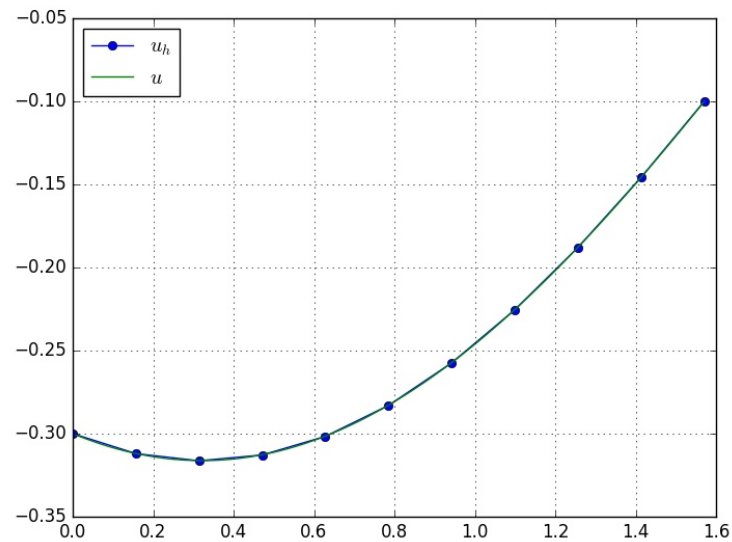


Figura 1.9: Esboço dos gráficos das soluções referentes ao Exemplo 1.3.1.

```

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)

# espacio
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 Neumann⁵ homogênea em $x = L$: encontrar u tal que

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

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

com u_0 e f dados.

⁵Carl 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 (1.106), obtemos

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

Aplicando a integração por partes, temos

$$\int_I u' v' \, dx - \underbrace{u'(L)v(L)}_{u'(L)=0} + \underbrace{u'(0)v(0)}_{v(0)=0} = \int_I f v \, dx. \quad (1.109)$$

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

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

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 1.3.2. Consideremos o problema

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

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

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

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

para todo $v \in V$, com as formas bilinear $a(\cdot, \cdot)$ e linear $L(\cdot)$ dadas em (1.111) e (1.112).

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

$$a(u_h, v_h) = L(v_h), \quad \forall v_h \in V_h. \quad (1.116)$$

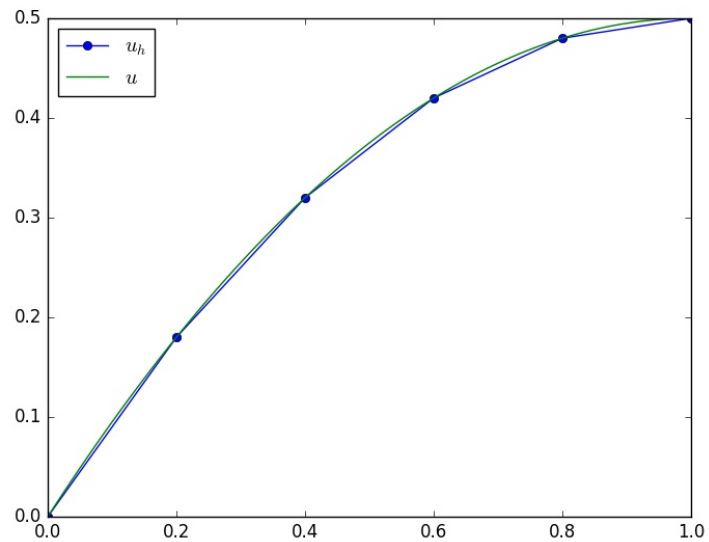


Figura 1.10: Esboço dos gráficos das soluções referentes ao Exemplo 1.3.4.

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)

# espaço
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

$$-u'' = f, \quad \forall x \in I = [0, L], \quad (1.117)$$

$$u(0) = u_0, \quad u'(L) = \alpha, \quad (1.118)$$

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 (1.117), obtemos

$$-\int_I u'' v \, dx = \int_I f v \, dx. \quad (1.119)$$

Aplicando a integração por partes, temos

$$\int_I u' v' \, dx - \alpha v(L) = \int_I f v \, dx. \quad (1.120)$$

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

$$a(u, v) - b(= L(v)), \quad \forall v \in V, \quad (1.121)$$

onde $a(u, v)$ é a forma bilinear

$$a(u, v) = \int_I u' v' \, dx \quad (1.122)$$

e $L(v)$ é a forma linear

$$L(v) = \int_I f v \, dx + \alpha v(L). \quad (1.123)$$

Exemplo 1.3.3. Consideremos o problema

$$-u'' = 1, \quad x \in I = [0, 1], \quad (1.124)$$

$$u(0) = 0, \quad u'(1) = 1. \quad (1.125)$$

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

$$a(u, v) = L(v), \quad \forall v \in V, \quad (1.126)$$

com

$$a(u, v) = \int_I u' v' \, dx \quad (1.127)$$

e

$$L(v) = \int_I f v \, dx + 1 \cdot v(1). \quad (1.128)$$

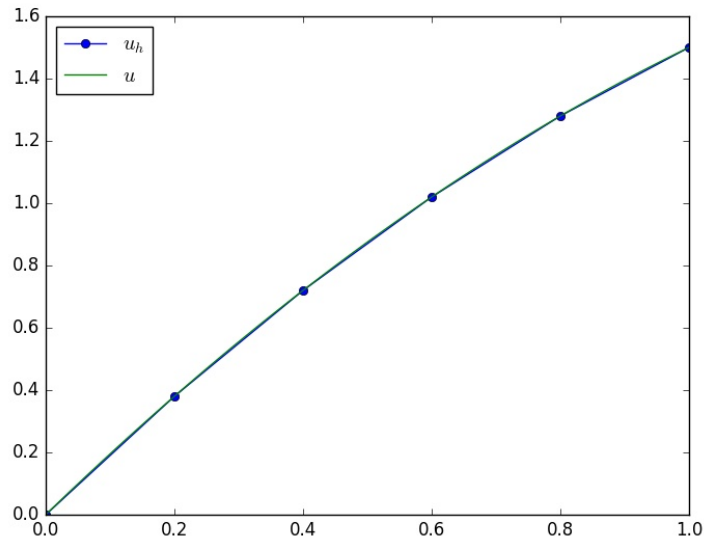


Figura 1.11: Esboço dos gráficos das soluções referentes ao Exemplo 1.3.3.

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

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

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 Robin⁶: encontrar u tal que

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

$$u'(0) = r_0(u(0) - s_0), \quad -u'(L) = r_L(u(L) - s_L), \quad (1.131)$$

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 (1.130), obtemos

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

Aplicando a integração por partes, temos

$$\int_I u' v' \, dx - \underbrace{u'(L)v(L)}_{-u'(L)=r_L(u(L)-s_L)} + \underbrace{u'(0)v(0)}_{u'(0)=r_0(u(0)-s_0)} = \int_I f v \, dx. \quad (1.133)$$

ou, mais adequadamente,

$$\int_I u' v' \, dx + r_L u(L)v(L) + r_0 u(0)v(0) = \int_I f v \, dx + r_L s_L v(L) + r_0 s_0 v(0). \quad (1.134)$$

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

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

onde $a(u, v)$ é a forma bilinear

$$a(u, v) = \int_I u' v' \, dx + r_L u(L)v(L) + r_0 u(0)v(0) \quad (1.136)$$

e $L(v)$ é a forma linear

$$L(v) = \int_I f v \, dx + r_L s_L v(L) + r_0 s_0 v(0). \quad (1.137)$$

Exemplo 1.3.4. Consideremos o problema

$$-u'' = 1, \quad x \in I = [0, 1], \quad (1.138)$$

$$u'(0) = u(0), \quad -u'(1) = u(1) - 1. \quad (1.139)$$

⁶Victor 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

$$a(u, v) = L(v), \quad \forall v \in V, \quad (1.140)$$

onde

$$a(u, v) = \int_I u'v' dx + u(1)v(1) + u(0)v(0) \quad (1.141)$$

e

$$L(v) = \int_I f v dx + 1 \cdot v(1). \quad (1.142)$$

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

$$a(u_h, v_h) = L(v_h), \quad \forall v_h \in V_h. \quad (1.143)$$

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.

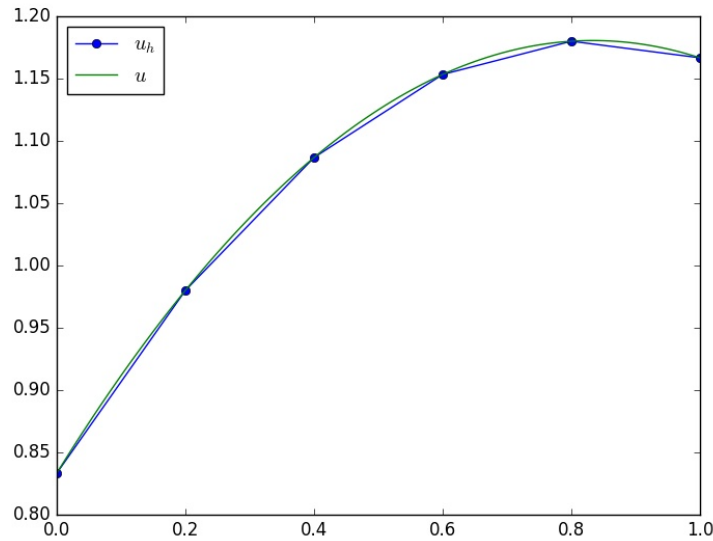


Figura 1.12: Esboço dos gráficos das soluções referentes ao Exemplo ??.

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)

#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

$$-u'' + u' + 2u = -\cos(x), \quad x \in (0, \pi/2), \quad (1.144)$$

$$u(0) = -0,3, \quad u(\pi/2) = -0,1. \quad (1.145)$$

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 Sistemas de equações

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

$$-u_0'' + u_1 = f_0, \forall x \in (0, L) \quad (1.146)$$

$$-u_1'' + u_0 = f_1, \forall x \in (0, L) \quad (1.147)$$

$$u_0(0) = u_{00}, \quad u_0(L) = u_{0L}, \quad (1.148)$$

$$u_1(0) = u_{10}, \quad u_1(L) = u_{1L}, \quad (1.149)$$

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

$$a(u, v) = L(v), \forall v = (v_0, v_1) \in V \times V, \quad (1.150)$$

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

$$a(u, v) = \int_I u_0' v_0' dx + \int_I u_1' v_1' dx + \int_I u_0 v_0 dx + \int_I u_1 v_1 dx \quad (1.151)$$

e a forma linear é

$$L(v) = \int_I f_0 v_0 dx + \int_I f_1 v_1 dx. \quad (1.152)$$

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

$$a(u_h, v_h) = L(v_h), \forall v_h = (v_{h0}, v_{h1}) \in V_h \times V_h, \quad (1.153)$$

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 1.4.1. Consideremos o seguinte problema de valor de contorno

$$-u_0'' + u_1 = \sin(x) + \cos(x), \forall x \in (-\pi, \pi) \quad (1.154)$$

$$-u_1'' + u_0 = \cos(x) - \sin(x), \forall x \in (-\pi, \pi) \quad (1.155)$$

$$u_0(-\pi) = 0, \quad u_0(\pi) = 0, \quad (1.156)$$

$$u_1(-\pi) = -1, \quad u_1(\pi) = -1. \quad (1.157)$$

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

$$a(u_h, v_h) = L(v_h), \forall v_h = (v_{h0}, v_{h1}) \in V_h \times V_h, \quad (1.158)$$

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 (1.151) e (1.152), respectivamente.

A Figura 1.13 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.

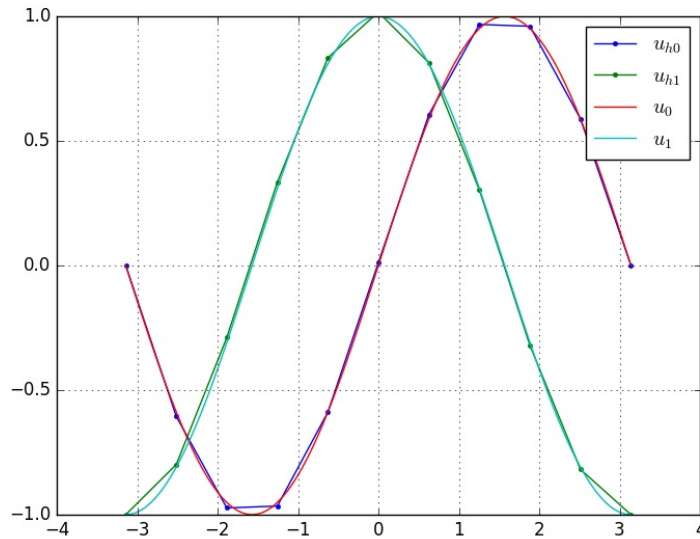


Figura 1.13: Esboço dos gráficos das soluções referentes ao Exemplo 1.4.1.

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(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 ...

1.5 Malha adaptativa

Em construção ...

Exercícios

Em construção ...

Resposta dos Exercícios

E 1.2.1. [Código](#) FENiCS.

E 1.3.1. [Código](#).

Referências Bibliográficas

- [1] Hans Petter Langtangen and Anders Logg. *Solving PDEs in Python*. Springer, 2017.
- [2] M.G. Larson and F. Bengson. *The Finite Element Method: Theory, Implementation, and Applications*. Springer, 2013.

Índice Remissivo

graus de liberdade, [1](#)

malha, [2](#)

operador

interpolação linear, [2](#)

operador de

projecção L^2 , [10](#)