Método de Elementos Finitos

Pedro H A Konzen

24 de agosto de 2018

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

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

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

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

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** $\pi : C^0(I) \to P_1(I)$ 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 $f(x) = 3\,\mathrm{sen}(2\pi x)$ 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:

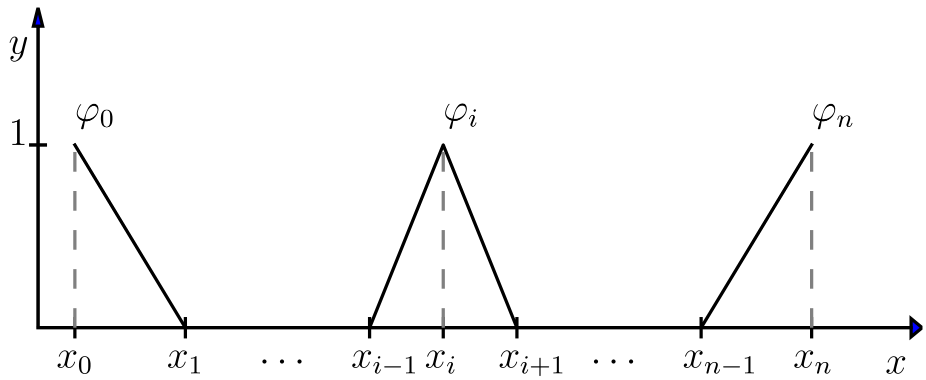

Figura 1.1: Base nodal para o espaço das funções lineares por parte.

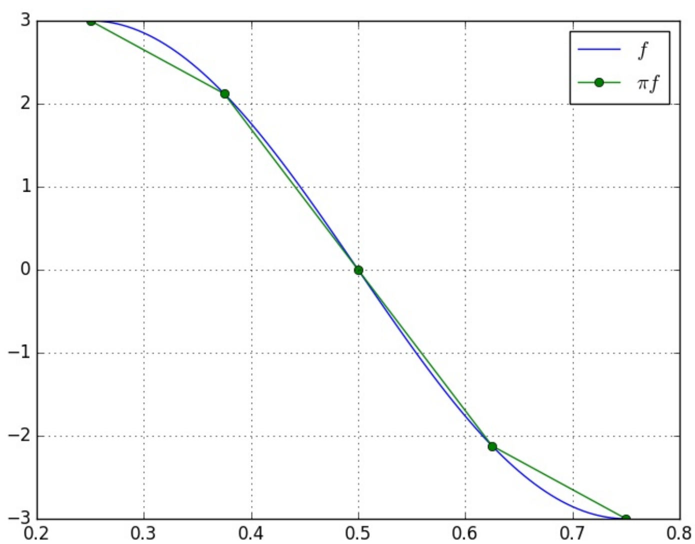

Figura 1.2: Interpolação linear de $f(x) = 3\,\mathrm{sen}(2\pi x)$ no espaço $P_1([1/4, 3/4])$.

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

```
import matplotlib.pyplot as plt

# malha
mesh = IntervalMesh(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

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

Lembramos que valem as desigualdades triangular

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

e a de Cauchy-Schwarz

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

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,

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

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

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

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 (1.23) 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 $f(x) = 3\,\mathrm{sen}(2\pi x)$ 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)

n=5
for k in np.arange(1,n+1):
```

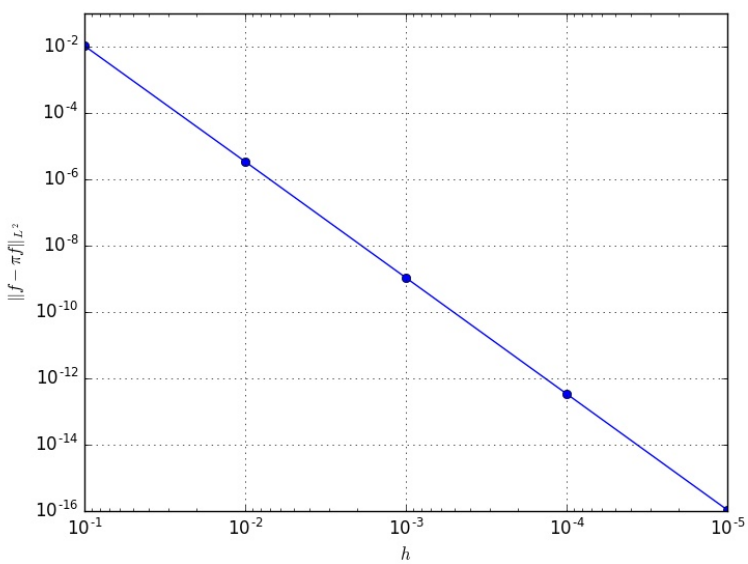

Figura 1.3: Erro de interpolação de $f(x) = 3\,\mathrm{sen}(2\pi x)$ no espaço $P_1([0,h])$.

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

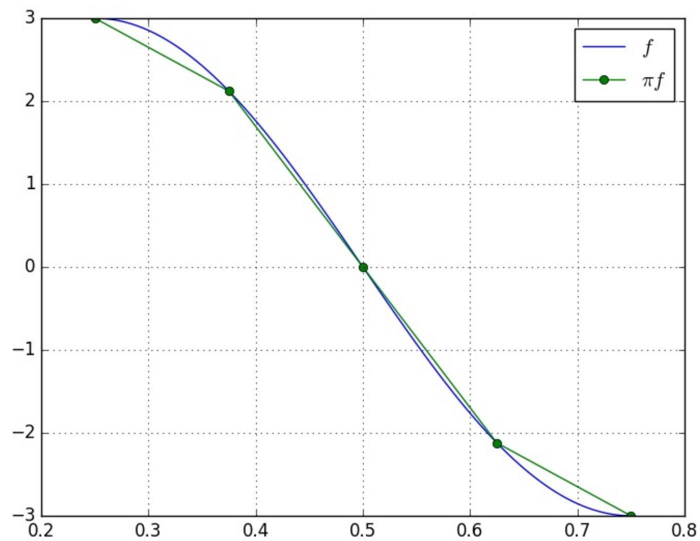

Figura 1.4: Interpolação linear de $f(x) = 3\,\text{sen}(2\pi x)$ no espaço V_h das funções lineares por partes sobre uma malha com 5 pontos.

```
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*

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

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

(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 $P_hf \in V_h$, temos

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

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

$$\begin{aligned}
\int_I (f - P_hf)\varphi_i \, dx = 0 &\Leftrightarrow \int_I f\varphi_i \, dx = \int_I P_hf\varphi_i \, dx && (1.40)\\
&\Leftrightarrow \int_I f\varphi_i \, dx = \int_I \left(\sum_{j=0}^{n} \xi_j\varphi_j\right) \varphi_i \, dx && (1.41)\\
&\Leftrightarrow \sum_{j=0}^{n} \xi_j \int_I \varphi_j\varphi_i \, dx = \int_I f\varphi_i \, dx, && (1.42)
\end{aligned}$$

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

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

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 $b = (b_0, b_1, \dots, b_n)$ 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

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

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).
Com o FENiCS, podemos computar P_hf com o seguinte código:

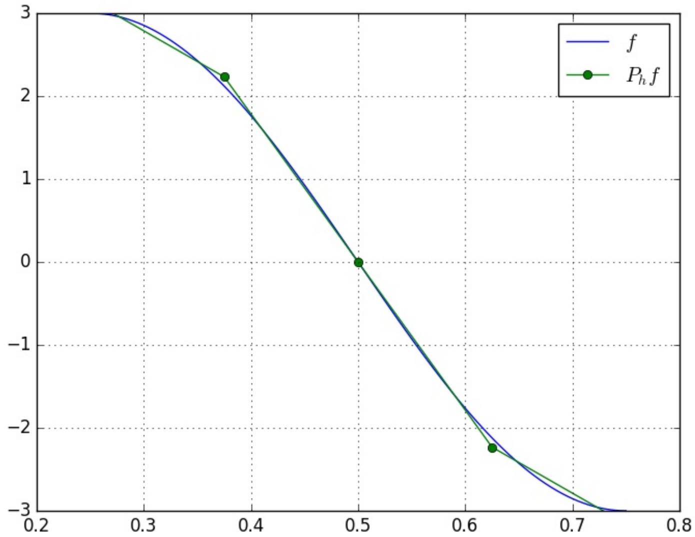

Figura 1.5: Projeção L^2 de $f(x) = 3\,\mathrm{sen}(2\pi x)$ no espaço V_h das funções lineares por partes sobre uma malha com 5 pontos.

```
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

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

Demonstração. Dado $v \in V_h$, temos

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

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*

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

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

$$\begin{aligned} \|f - P_h f\|^2_{L^2(I)} &\leq \|f - \pi f\|^2_{L^2(I)} && (1.54) \\ &\leq C \sum_{i=1}^{n} h_i^4 \|f''\|^2_{L^2(I_i)}. && (1.55) \end{aligned}$$

□

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 $[1/4, 3/4]$.

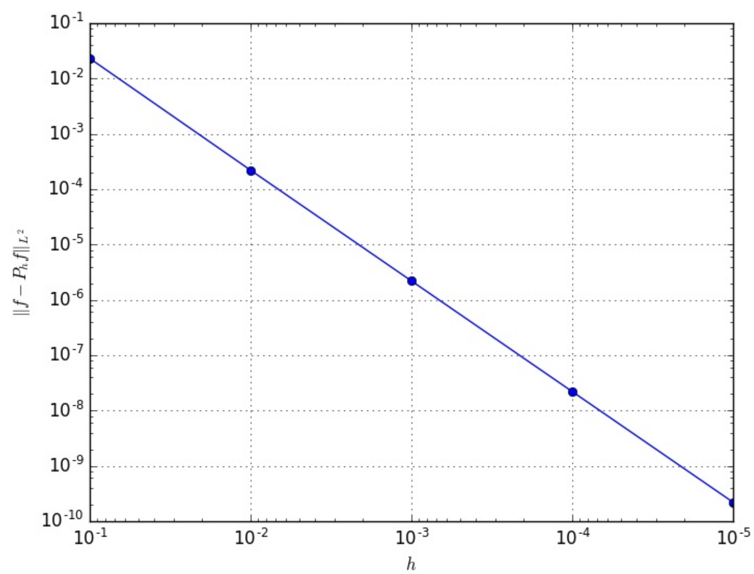

Figura 1.6: Erro de interpolação de $f(x) = 3\,\mathrm{sen}(2\pi x)$ no espaço V_h.

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))
```

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_hf 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_hf e compute o erro $\|f - P_hf\|_{L^2}$.

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

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

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

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[1].

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

$$\begin{aligned} a(u_h,\varphi_i) &= a\left(\sum_{j=1}^{n-1} \xi_j \varphi_j, \varphi_i\right) && (1.70) \\ &= \sum_{j=1}^{n-1} \xi_j a(\varphi_j,\varphi_i). && (1.71) \end{aligned}$$

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

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

[1]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 contínuas por partes $V_{h,0}$ construído numa malha uniforme de 5 células no intervalo $I = [0,1]$. Para tanto, montamos o sistema de elementos finitos (1.72), resolvemo-lo e computamos a solução com

$$u_h = \sum_{j=1}^{n} \xi_j\varphi_j, \tag{1.77}$$

onde a j-ésima função de base é

$$\varphi_j = \begin{cases} (x-x_{j-1})/(x_j-x_{j-1}) & ,x_{j-1} \leq x < x_j, \\ (x_j-x)/(x_j-x_{j+1}) & ,x_j \leq x \leq x_{j+1} \end{cases} \tag{1.78}$$

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

# malha
```

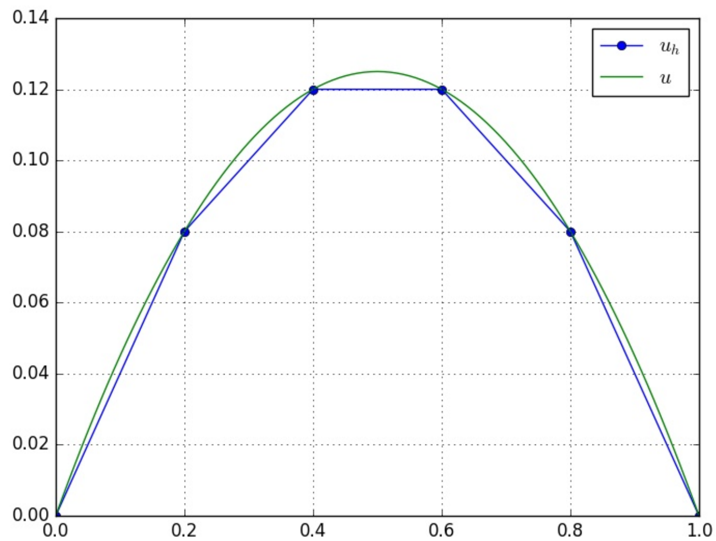

Figura 1.7: Esboço dos gráficos das soluções referentes ao Exemplo 1.2.1.

```
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 = dot(grad(u), grad(v))*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*

Em construção ...

Exercícios

Em construção ...

Resposta dos Exercícios

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