

Método de Elementos Finitos

Pedro H A Konzen

21 de agosto 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 Fundamentos preliminares	1
1.1.1 Interpolação	2
1.1.2 Exercícios	8
Respostas dos Exercícios	10
Referências Bibliográficas	11
Índice Remissivo	12

Capítulo 1

Método de elementos finitos em 1D

1.1 Fundamentos preliminares

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

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

Podemos computar a função interpolada πf e o gráfico da Figura 1.2 no Python com o seguinte código:



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

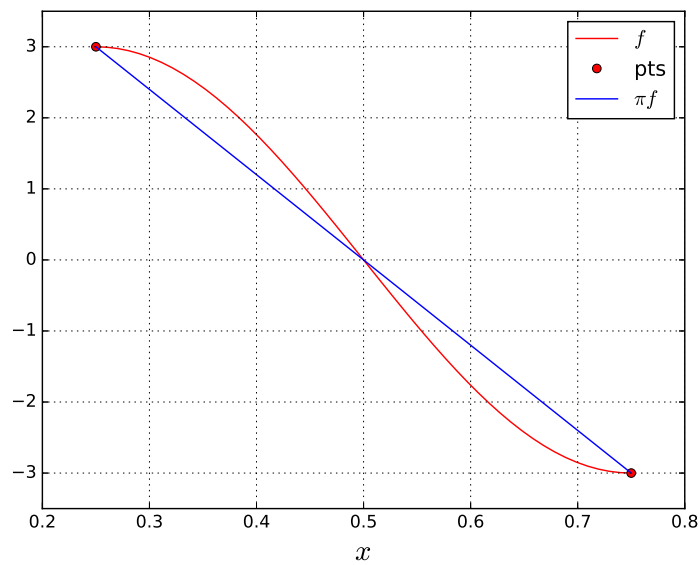


Figura 1.2: Interpolação linear de $f(x) = 3 \sin(2\pi x)$ no espaço $P_1([1/4, 3/4])$.

```
from __future__ import division
import numpy as np
from scipy import interpolate
```

```

import matplotlib.pyplot as plt

#fun obj
f = lambda x: 3*np.sin(2*np.pi*x)

#intervalo
x0=1/4
x1=3/4

#interp
pif = interpolate.interp1d([x0,x1],[f(x0),f(x1)])

#grafico
xpts = np.linspace(x0,x1)
plt.plot(xpts,f(xpts),color="red",label="$f$")
plt.plot([x0,x1],[f(x0),f(x1)],
         color="red",linestyle="",
         marker="o",label="pts")
plt.plot(xpts,pif(xpts),
         color="blue",label=r"$\pi f$")
plt.grid("on")
plt.xlabel(r"$x$",fontsize=20)
plt.ylim((-3.5,3.5))
plt.legend(numpoints=1)
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-Schwarz

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

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

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

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

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

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 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

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

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

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

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

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

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

□

Vamos, agora, generalizar este resultado 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 1.1.2. A Figura 1.3 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). Podemos computar a função interpolada πf e o gráfico da Figura 1.3 no Python com o seguinte código:

```
from __future__ import division
import numpy as np
from scipy import interpolate
import matplotlib.pyplot as plt
```

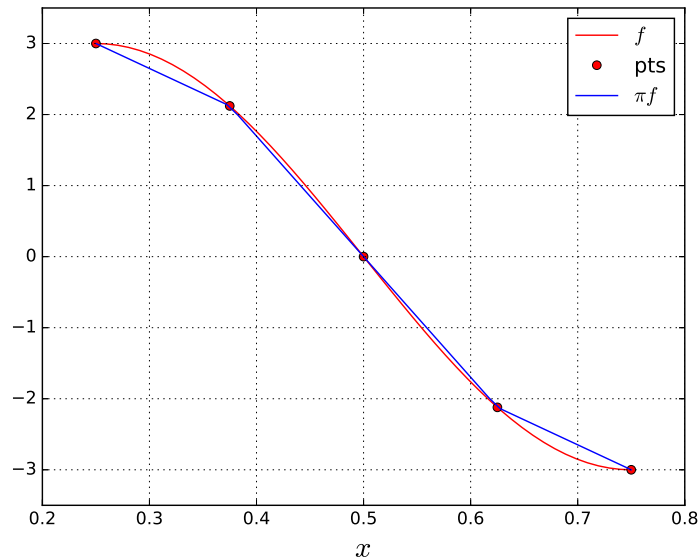


Figura 1.3: Interpolação linear de $f(x) = 3 \sin(2\pi x)$ no espaço V_h das funções lineares por partes sobre uma malha com 5 pontos.

```
#fun obj
f = lambda x: 3*np.sin(2*np.pi*x)

#malha
n=5
l0=1/4
l1=3/4
xx = np.linspace(l0,l1,n)

#interp
pif = interpolate.interp1d(xx,f(xx))

#grafico
xpts = np.linspace(l0,l1)
plt.plot(xpts,f(xpts),color="red",label="$f$")
plt.plot(xx,f(xx),
```

```

        color="red",linestyle="",
        marker="o",label="pts")
plt.plot(xpts,pif(xpts),
        color="blue",label=r"$\pi f$")
plt.grid("on")
plt.xlabel(r"$x$",fontsize=20)
plt.ylim((-3.5,3.5))
plt.legend(numpoints=1)
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

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

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

□

Em construção ...

1.1.2 Exercícios

E 1.1.1. Faça um código para verificar as estimativas 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 $f(x) = 3 \operatorname{sen}(2\pi x)$ no espaço V_h das funções lineares por partes.

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

graus de liberdade, [1](#)
malha, [2](#)
operador
 interpolação linear, [2](#)