1 Formulação do problema

Dados $f : [0,1] \to \mathbb{R}$ e constantes reais positivas α e β, determine $u : [0,1] \to \mathbb{R}$ tal que

$$\begin{cases} -\alpha u_{xx}(x) + \beta u(x) = f(x), \quad x \in]0,1[, \\ u(0) = u(1) = 0. \end{cases} \tag{1}$$

Exemplos de solução exata para o problema em (1)

- Ex. 1. Se $\alpha = 1$, $\beta = 0$ e $f(x) = 8$, então $u(x) = -4x(x-1)$.
- Ex. 2. Se $\alpha = \beta = 1$ e $f(x) = x$, então $u(x) = x + \frac{e^{-x} - e^{x}}{e - e^{-1}}$.
- Ex. 3. Se $\alpha = \beta = 1$ e $f(x) = -2\pi^2 \cos(2\pi x) + \sin^2(\pi x)$, então $u(x) = \sin^2(\pi x)$.

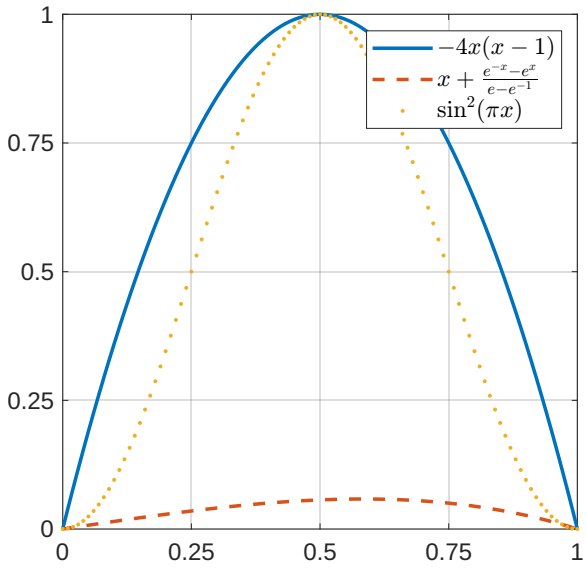

Figura 1: Solução do problema (1) para α, β e f definidos nos exemplos 1, 2 e 3.

2 Problema aproximado - via método de Galerkin

Dados $f : [0,1] \to \mathbb{R}$ e constantes reais positivas α e β, determine $u_h \in V_m = [\varphi_0, \varphi_1, \ldots, \varphi_{m-1}]$ tal que

$$\alpha \int_0^1 \frac{du_h}{dx}(x)\frac{dv_h}{dx}(x)dx + \beta \int_0^1 u_h(x)v_h(x)dx = \int_0^1 f(x)v_h(x)dx, \quad \forall v_h \in V_m. \tag{2}$$

2.1 Formulação matricial

Tomando $u_h(x) = \sum_{j=0}^{m-1} c_j \varphi_j(x)$ e $v_h = \varphi_i$, para $i = 0, 1, \ldots, m-1$, em (2), temos a forma matriz-vetor do problema aproximado, isto é, determinar um vetor $c \in \mathbb{R}^m$ tal que

$$Kc = F,$$

em que, para $i, j \in \{0, 1, \ldots, m-1\}$,

$$K_{i,j} = \alpha \int_0^1 \frac{d\varphi_i}{dx}(x)\frac{d\varphi_j}{dx}(x)dx + \beta \int_0^1 \varphi_i(x)\varphi_j(x)dx \quad \text{e} \quad F_i = \int_0^1 f(x)\varphi_i(x)dx.$$

3 Função base linear por partes

Dado $m \in \mathbb{N}$ fixo, considere $0 = x_0 < x_1 < \cdots < x_m < x_{m+1} = 1$ uma partição uniforme de $[0,1]$, ou seja, o diâmetro de cada elemento é dado por

$$h = 1/(m+1).$$

Para cada $i \in \{0, 1, \ldots, m-1\}$, defina

$$\varphi_i(x) = \begin{cases} \dfrac{x - x_i}{h}, & \forall x \in [x_i, x_{i+1}], \\ \dfrac{x_{i+2} - x}{h}, & \forall x \in [x_{i+1}, x_{i+2}], \\ 0, & \forall x \notin [x_i, x_{i+2}]. \end{cases} \quad \text{e} \quad \frac{d\varphi_i}{dx}(x) = \begin{cases} \dfrac{1}{h}, & \forall x \in]x_i, x_{i+1}[, \\ -\dfrac{1}{h}, & \forall x \in]x_{i+1}, x_{i+2}[, \\ 0, & \forall x \notin]x_i, x_{i+2}[. \end{cases}$$

3.1 Notação global/local e elemento padrão da base linear

Os gráficos das funções φ_i, para $i = 0, 1, \ldots, m-1$, podem ser vistos na Figura 2. Na Figura 3, apresentamos a notação global, a notação local e o elemento padrão referente à base linear por partes.

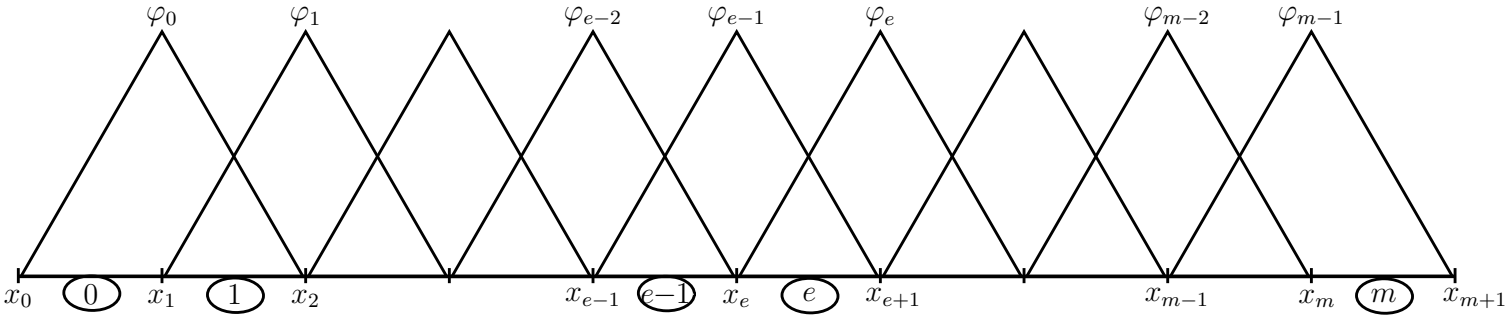

Figura 2: Função base linear por partes.

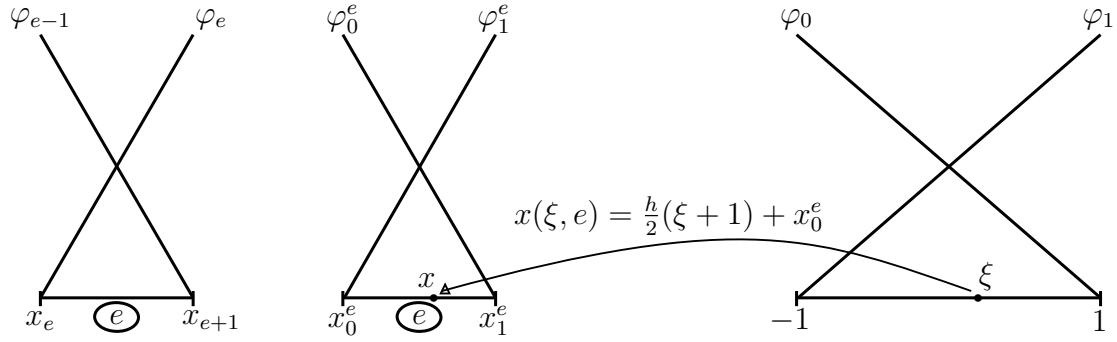

Figura 3: Elemento global, local e padrão referente a base linear.

3.2 Cálculo do erro entre a solução exata e aproximada

Para medir o erro entre a solução exata e aproximada, consideraremos a norma do espaço $L^2(]0,1[)$, i.e.,

$$\mathcal{E}_h = \|u - u_h\| = \sqrt{\int_0^1 |u(x) - u_h(x)|^2 dx}.$$

Para tal, note que

$$\begin{aligned}
\mathcal{E}_h^2 &= \int_0^1 |u(x) - u_h(x)|^2 dx \\
&= \sum_{e=0}^{m} \int_{x_0^e \equiv x_e}^{x_1^e \equiv x_{e+1}} |u(x) - u_h(x)|^2 dx \\
&= \sum_{e=0}^{m} \int_{x_0^e}^{x_1^e} |u(x) - \sum_{j=0}^{m-1} c_j \varphi_j(x)|^2 dx \\
&= \int_{x_0^0}^{x_1^0} |u(x) - c_0\varphi_0(x)|^2 dx + \sum_{e=1}^{m-1} \int_{x_0^e}^{x_1^e} |u(x) - c_{e-1}\varphi_{e-1}(x) - c_e\varphi_e(x)|^2 dx + \int_{x_0^m}^{x_1^m} |u(x) - c_{m-1}\varphi_{m-1}(x)|^2 dx \\
&= \frac{h}{2}\Big[\int_{-1}^{1} |u(x(\xi,0)) - c_0\varphi_1(\xi)|^2 d\xi + \sum_{e=1}^{m-1} \int_{-1}^{1} |u(x(\xi,e)) - c_{e-1}\varphi_0(\xi) - c_e\varphi_1(\xi)|^2 d\xi + \int_{-1}^{1} |u(x(\xi,m)) - c_{m-1}\varphi_0(\xi)|^2 d\xi\Big].
\end{aligned}$$

Com isso, temos $\mathcal{E}_h$ dado por:

$$\sqrt{\frac{h}{2}\Big[\int_{-1}^{1} |u(x(\xi,0)) - c_0\varphi_1(\xi)|^2 d\xi + \sum_{e=1}^{m-1} \int_{-1}^{1} |u(x(\xi,e)) - c_{e-1}\varphi_0(\xi) - c_e\varphi_1(\xi)|^2 d\xi + \int_{-1}^{1} |u(x(\xi,m)) - c_{m-1}\varphi_0(\xi)|^2 d\xi\Big]}.$$

Atividade 1

1.1 Considerando α, β e f definidos como no exemplo 1, obtenha a solução aproximada do problema (1) via o método de Galerkin. Exiba em um mesmo gráfico as soluções aproximada e exata do problema para diversos valores de m.

1.2 Repita o processo anterior considerando α, β e f definidos como no exemplo 2.

1.3 Qual a dificuldade desse processo se considerarmos α, β e f definidos como no exemplo 3?

1.4 Considerando a mesma partição uniforme de $[0, 1]$ definida anteriormente, determine a solução aproximada do problema (1) via o método das diferenças finitas. Exiba a solução aproximada e exata para os casos de α, β e f definidos como nos exemplos 1, 2 e 3.
Dica: Para cada $i \in \{1, 2, \ldots, m\}$, considere $-\alpha\dfrac{u_{i+1} - 2u_i + u_{i-1}}{h^2} + \beta u_i = f_i$.

1.5 Exiba, em um mesmo gráfico, a solução exata e as soluções aproximadas por Galerkin e por diferenças finitas do Exemplo 1, para um mesmo valor de m.

4 Montagem das matrizes de elementos finitos de forma vetorizada

Nesta seção, continuaremos a utilizar uma partição uniforme para o intervalo $[0, 1]$. No entanto, introduziremos uma nova notação para a partição do intervalo e uma nova numeração global para as funções base lineares por partes.

Dado $n_e \in \mathbb{N}$, consideremos a partição uniforme de $[0, 1]$ como $0 = x_0 < x_1 < \cdots < x_{n_e-1} < x_{n_e} = 1$, onde o diâmetro de cada elemento é dado por $h = 1/n_e$, com n_e representando o número de elementos da malha.

Em relação à numeração global das funções base, adotaremos a convenção apresentada na Figura 4. Isso estabelece uma nova relação entre as funções locais e globais, conforme ilustrado na Figura 5.

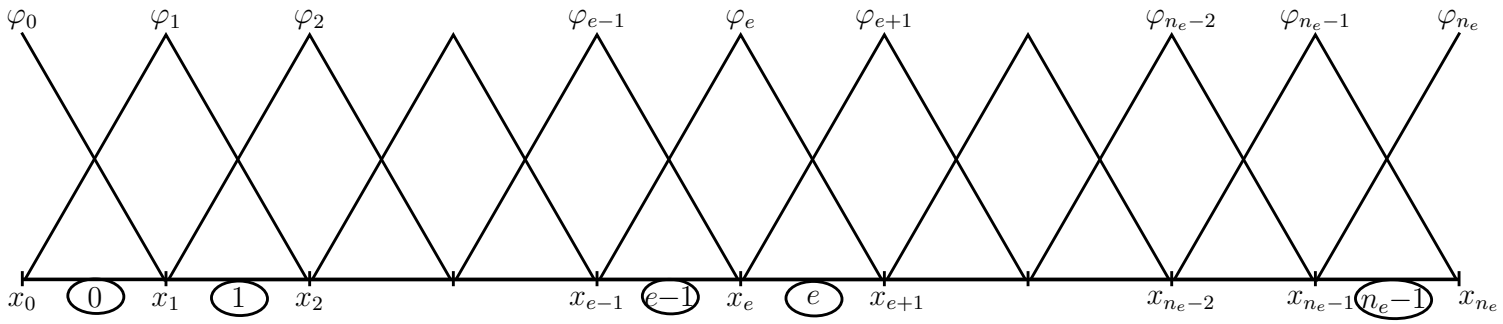

Figura 4: Função base linear por partes.

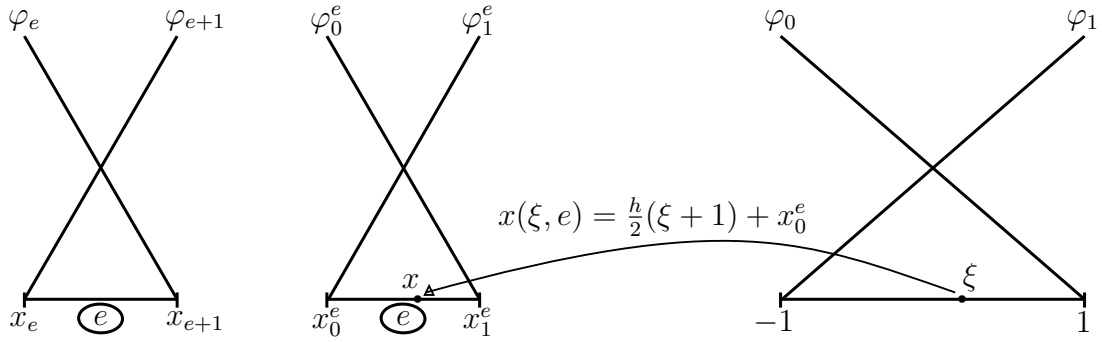

Figura 5: Elemento global, local e padrão referente a base linear.

Antes de avançarmos para a montagem das matrizes de elementos finitos de forma vetorizada, começaremos definindo uma série de matrizes e vetores que serão posteriormente utilizados no processo.
Pontos e pesos de Gauss

- P: Vetor $1 \times n_{pg}$ contendo os pontos de Gauss para a quadratura Gaussiana.
- W: Vetor $1 \times n_{pg}$ contendo os pesos para a quadratura Gaussiana.

Funções base nos pontos de Gauss

- $phiP$: Matriz $2 \times n_{pg}$ contendo as funções $\varphi_0(\xi)$ e $\varphi_1(\xi)$ calculadas nos pontos de Gauss P, i.e., $\begin{bmatrix} \varphi_0(P) \\ \varphi_1(P) \end{bmatrix}$.
- $phiPT$: Matriz $n_{pg} \times 2$, a transposta da matriz $phiP$.
- $dphiP$: Matriz $2 \times n_{pg}$ contendo $\dfrac{d\varphi_0}{d\xi}(P)$ e $\dfrac{d\varphi_1}{d\xi}(P)$.
- $dphiPT$: Matriz $n_{pg} \times 2$, a transposta da matriz $dphiP$.

Coordenadas globais dos pontos de Gauss

- $xPTne$: Matriz $n_{pg} \times n_e$ que representa os pontos de Gauss nos elementos finitos após a transformação de variável. Em cada elemento finito, há uma função que mapeia pontos de $[-1, 1]$ para o intervalo $[x_0^e, x_1^e]$. A coluna e da matriz $xPTne$ é formada pelos valores de $x(P^T, e)$. A matriz $xPTne$ pode ser construída, de forma vetorizada (em matlab), da seguinte forma:

$$xPTne = \left[\frac{h}{2}(P^T + 1)\right]_{n_{pg} \times 1} + \left[x_0^0, x_0^1, \ldots, x_0^{n_e - 1}\right]_{1 \times n_e}.$$

Relação entre a numeração local e global das funções φ

- LG: Matriz $2 \times n_e$ que estabelece a relação entre a numeração local e global das funções base de cada elemento. Ela é definida como:

$$LG = \begin{bmatrix} 0 & 1 & \ldots & e & \ldots & n_e - 1 \\ 1 & 2 & \ldots & e+1 & \ldots & n_e \end{bmatrix}.$$

Observe que a numeração global da função local φ_a^e é dada por $LG(a, e)$.

Re-enumeração das funções base globais

- *EQ*: Vetor $1 \times (n_e + 1)$ contendo uma nova numeração para as funções base globais. As funções que servirão como base para o espaço aproximado V_m receberão numerações de 0 a $m-1$, enquanto aquelas não utilizadas receberão o número m. A definição de EQ dependerá da condição de fronteira da equação diferencial em questão. No caso do problema (1), que possui uma condição prescrita na fronteira ($u(0) = u(1) = 0$), a definição de EQ para a base linear é a seguinte:

$$EQ = [m,\ 0,\ 1, \ldots,\ m-1,\ m],$$

com $m = n_e - 1$.

Relação entre a numeração local e a numeração global após a re-enumeração

- *EQoLG*: Matriz $2 \times n_e$ que estabelece a correspondência entre a numeração local e a numeração global das funções base após o processo de re-enumeração. Essa matriz é definida como $EQ(LG)$ e, para o caso específico de EQ previamente definido, assume a forma:

$$EQoLG = \begin{bmatrix} m & 0 & \ldots & e-1 & \ldots & m-2 & m-1 \\ 0 & 1 & \ldots & e & \ldots & m-1 & m \end{bmatrix} \equiv \begin{bmatrix} n_e-1 & 0 & \ldots & e-1 & \ldots & n_e-3 & n_e-2 \\ 0 & 1 & \ldots & e & \ldots & n_e-2 & n_e-1 \end{bmatrix}.$$

4.1 Montagem vetorizada da matriz K

A matriz global K associada ao problema aproximado (2) é definida da seguinte forma:

$$K_{i,j} = \alpha \int_0^1 \frac{d\varphi_i}{dx}(x)\frac{d\varphi_j}{dx}(x)dx + \beta \int_0^1 \varphi_i(x)\varphi_j(x)dx, \quad \text{com} \quad i,\, j \in \{0, 1, \ldots, m-1\}.$$

Na versão local, a matriz K^e, que é 2×2, é expressa como:

$$K^e_{a,b} = \frac{2\alpha}{h} \int_{-1}^1 \frac{d\varphi_a}{d\xi}(\xi)\frac{d\varphi_b}{d\xi}(\xi)d\xi + \frac{\beta h}{2} \int_{-1}^1 \varphi_a(\xi)\varphi_b(\xi)d\xi, \quad \text{com} \quad a,\, b \in \{0, 1\}.$$

Resolvendo as integrais utilizando quadratura Gaussiana e reescrevendo o resultado de forma vetorizada, temos

$$\begin{aligned} K^e_{a,b} =& \frac{2\alpha}{h} \sum_{\ell=0}^{n_{pg}-1} W_\ell \frac{d\varphi_a}{d\xi}(P_\ell)\frac{d\varphi_b}{d\xi}(P_\ell) + \frac{\beta h}{2} \sum_{\ell=0}^{n_{pg}-1} W_\ell \varphi_a(P_\ell)\varphi_b(P_\ell) \\ =& \frac{2\alpha}{h} \left[W. * \frac{d\varphi_a}{d\xi}(P)\right]_{1\times n_{pg}} * \left[\frac{d\varphi_b}{d\xi}(P)^T\right]_{n_{pg}\times 1} + \frac{\beta h}{2} \left[W. * \varphi_a(P)\right]_{1\times n_{pg}} * \left[\varphi_b(P)^T\right]_{n_{pg}\times 1}. \end{aligned}$$

O próximo passo consiste em montar a matriz local por completa de forma vetorizada. Seguem os detalhes:

$$\begin{aligned} K^e =& \frac{2\alpha}{h} \begin{bmatrix} W. * \frac{d\varphi_0}{d\xi}(P) \\ W. * \frac{d\varphi_1}{d\xi}(P) \end{bmatrix}_{2\times n_{pg}} * \left[\frac{d\varphi_0}{d\xi}(P)^T\ \frac{d\varphi_1}{d\xi}(P)^T\right]_{n_{pg}\times 2} + \frac{\beta h}{2} \begin{bmatrix} W. * \varphi_0(P) \\ W. * \varphi_1(P) \end{bmatrix}_{2\times n_{pg}} * \left[\varphi_0(P)^T\ \varphi_1(P)^T\right]_{n_{pg}\times 2} \\ =& \frac{2\alpha}{h} \left[W. * dphiP\right]_{2\times n_{pg}} * \left[dphiPT\right]_{n_{pg}\times 2} + \frac{\beta h}{2} \left[W. * phiP\right]_{2\times n_{pg}} * \left[phiPT\right]_{n_{pg}\times 2}. \end{aligned}$$

Após o cálculo da matriz local K^e, um procedimento comum para incorporar K^e em K é realizar o processo de acumulação usando pelo menos um "for", o loop do elemento. Segue um exemplo de pseudocódigo:

```
Inicialize a matriz global K, m × m, com zeros;
for e=0,1,...,n_e − 1 do
    for b=0,1 do
        j = EQoLG(b,e);
        for a=0,1 do
            i = EQoLG(a,e);
            if i ≃ m  &&  j ≃ m then
                K(i,j) = K(i,j) + Ke(a,b);
            end
        end
    end
end
```

Algoritmo 1: Montagem da matriz K - versão 1

Para vetorizar o processo de acumulação das matrizes locais K^e em K, precisamos eliminar a necessidade do condicional "if" no código. Para isso, a forma como definimos o vetor EQ desempenha um papel fundamental. Lembrando que atribuímos números crescentes de 0 a $m-1$ às funções φ que fazem parte da base do subespaço aproximada V_m, enquanto deliberadamente atribuímos o número m às que não fazem parte.

Devido a forma com a qual definimo EQ, podemos inicializar a matriz K com uma linha e coluna adicionais, acomodando as contribuições das funções locais que não fazem parte da base de V_m. Com isso, após concluir o processo de acumulação, basta excluir a última linha e coluna de K para retornarmos a matriz global correta. Esse processo pode ser visualizado no pseudocódigo dado a seguir.

```
Inicialize a matriz global K, (m + 1) × (m + 1), com zeros;
for e=0,1,...,n_e − 1 do
    for b=0,1 do
        j = EQoLG(b,e);
        for a=0,1 do
            i = EQoLG(a,e);
            K(i,j) = K(i,j) + Ke(a,b);
        end
    end
end
Remova a última linha e coluna da matriz K para obter a matriz global desejada;
```

Algoritmo 2: Montagem da matriz K - versão 2

Ao continuar o processo de vetorização do código, avançamos para a eliminação dos loops "for" internos, acumulando todas as entradas de K^e em K ao mesmo tempo. Isso é exemplificado no algoritmo a seguir.

```
Inicialize a matriz global K, (m + 1) × (m + 1), com zeros;
for e=0,1,...,n_e − 1 do
    Se = [K^e_{0,0}, K^e_{1,0}, K^e_{0,1}, K^e_{1,1}]^T;
    Ie = EQoLG([0, 1, 0, 1]^T, e);
    Je = EQoLG([0, 0, 1, 1]^T, e);
    K(Ie, Je) = K(Ie, Je) + Se;
end
Remova a última linha e coluna da matriz K para obter a matriz global desejada;
```

Algoritmo 3: Montagem da matriz K - versão 3

Para o passo final, ou seja, a eliminação do loop "for" que percorre os elementos, faremos uso da função *sparse*, disponível tanto em Matlab quanto Python. Para isso, precisamos montar três matrizes: I, J e S. As colunas e dessas matrizes correspondem aos vetores Ie, Je e Se usados no algoritmo anterior. Com isso, a versão final da montagem vetorizada da matriz K é a seguinte:

```
S = repmat([K^e_{0,0}, K^e_{1,0}, K^e_{0,1}, K^e_{1,1}]^T, 1, n_e);
I = EQoLG([0, 1, 0, 1]^T, :);
J = EQoLG([0, 0, 1, 1]^T, :);
K = sparse(I, J, S, m + 1, m + 1);
Remova a última linha e coluna da matriz K para obter a matriz global desejada;
```

Algoritmo 4: Montagem da matriz K - versão final vetorizada

4.2 Montagem vetorizada do vetor F

A vetor global F associado ao problema aproximado (2) é definido como:

$$F_i = \int_0^1 f(x)\varphi_i(x)dx, \quad \text{com} \quad i \in \{0, 1, \ldots, m-1\}.$$

Em sua versão local, um vetor 2×1, temos

$$F_a^e = \frac{h}{2}\int_{-1}^1 f\big(x(\xi, e)\big)\varphi_a(\xi)d\xi, \quad \text{com} \quad a \in \{0, 1\}.$$

Para cada $a \in \{0,1\}$ fixo, utilizando quadratura Gaussiana para resolver a integral e reescrevendo o resultado de forma vetorizada, obtemos

$$\begin{aligned} F_a^e =& \frac{h}{2} \sum_{\ell=0}^{n_{pg}-1} W_\ell f\big(x(P_\ell, e)\big)\varphi_a(P_\ell) \\ =& \frac{h}{2}\Big[W. * \varphi_a(P)\Big]_{1\times n_{pg}} * \Big[f\big(x(P^T, e)\big)\Big]_{n_{pg}\times 1}. \end{aligned}$$

Consequentemente, o cálculo do vetor local F^e completo segue como:

$$\begin{aligned} F^e =& \frac{h}{2}\begin{bmatrix} W. * \varphi_0(P) \\ W. * \varphi_1(P) \end{bmatrix}_{2\times n_{pg}} * \Big[f\big(x(P^T, e)\big)\Big]_{n_{pg}\times 1} \\ =& \frac{h}{2}\Big[W. * phiP\Big]_{2\times n_{pg}} * \Big[f\big(x(P^T, e)\big)\Big]_{n_{pg}\times 1}. \end{aligned}$$

Quanto ao processo de acumulação dos vetores locais F^e em F, segue uma abordagem comumente adotada.

Inicialize o vetor global F, $m \times 1$, com zeros;
for *e=0,1,...,*$n_e - 1$ **do**
  Calcule Fe;
  for *a=0,1* **do**
    i = EQoLG(a,e);
    if $i \simeq m$ **then**
      F(i) = F(i) + Fe(a);
    end
  end
end

Algoritmo 5: Montagem do vetor F - versão 1

Seguindo uma abordagem análoga ao processo passo a passo usado na vetorização da montagem da matriz K, eliminamos o "if" e o loop "for" interno da seguinte maneira:

Inicialize o vetor global F, $(m+1) \times 1$, com zeros;
for *e=0,1,...,*$n_e - 1$ **do**
  Calcule Fe;
  $I = EQoLG(:, e)$;
  $F(I) = F(I) + Fe$;
end
Remova a última linha do vetor F para obter o vetor global correto;

Algoritmo 6: Montagem do vetor F - versão 2

Para a etapa final, isto é, remover o "for" dos elementos, basta utilizar a função *sparse*.

$S = \frac{h}{2}\Big[W. * phiP\Big]_{2\times n_{pg}} * \Big[f(xPTne)\Big]_{n_{pg}\times n_e}$;
$I = EQoLG$;
$J = ones(2, n_e)$;
$F = sparse(I, J, S, m+1, 1)$;
Remova a última linha do vetor F para obter o vetor global correto;

Algoritmo 7: Montagem do vetor F - versão final vetorizada

No Matlab, existe uma função para acumular vetores, conhecida como *accumarray*. Com esta função, como pode visto a seguir, não é necessário definir J. Não estou certo se uma função equivalente está disponível no Python.

$S = \frac{h}{2}\Big[W. * phiP\Big]_{2\times n_{pg}} * \Big[f(xPTne)\Big]_{n_{pg}\times n_e}$;
$I = EQoLG$;
$F = accumarray(I(:), S(:), [m+1, 1])$;
Remova a última linha do vetor F para obter o vetor global correto;

Algoritmo 8: Montagem do vetor F - versão final vetorizada usando *accumarray*

4.3 Vetorização do cálculo do erro

Procedendo de forma análoga a subseção 3.2, para calcular o erro na norma do espaço $L^2(]0,1[)$, definido por

$$\mathcal{E}_h = \|u - u_h\| = \sqrt{\int_0^1 |u(x) - u_h(x)|^2 dx},$$

precisamos realizar o seguinte cálculo:

$$\sqrt{\frac{h}{2}\Big[\int_{-1}^{1}|u(x(\xi,0)) - c_0\varphi_1(\xi)|^2 d\xi + \sum_{e=1}^{n_e-2}\int_{-1}^{1}|u(x(\xi,e)) - c_{e-1}\varphi_0(\xi) - c_e\varphi_1(\xi)|^2 d\xi + \int_{-1}^{1}|u(x(\xi,n_e-1)) - c_{n_e-2}\varphi_0(\xi)|^2 d\xi\Big]}$$

Observe que, para cada $e \in \{1, 2, \ldots, n_e - 2\}$ fixo,

$$\begin{aligned}
&\int_{-1}^{1} \big|u\big(x(\xi,e)\big) - c_{e-1}\varphi_0(\xi) - c_e\varphi_1(\xi)\big|^2 d\xi \\
&\quad= \sum_{\ell=0}^{n_{pg}-1} W_\ell \big|u\big(x(P_\ell,e)\big) - c_{e-1}\varphi_0(P_\ell) - c_e\varphi_1(P_\ell)\big|^2 \\
&\quad= \Big[W\Big]_{1\times n_{pg}} * \left[\Big|\Big[u\big(x(P^T,e)\big)\Big]_{n_{pg}\times 1} - \big[\varphi_0(P^T),\varphi_1(P^T)\big]_{n_{pg}\times 2} * \begin{bmatrix} c_{e-1} \\ c_e \end{bmatrix}_{2\times 1}\Big| .\wedge 2\right]_{n_{pg}\times 1}.
\end{aligned}$$

Consequentemente, o valor da soma para e variando de 1 até $n_e - 2$ pode ser expresso de forma vetorizada como:

$$\begin{aligned}
&\sum_{e=1}^{n_e-2}\int_{-1}^{1} \big|u\big(x(\xi,e)\big) - c_{e-1}\varphi_0(\xi) - c_e\varphi_1(\xi)\big|^2 d\xi \\
&= sum\left(\Big[W\Big]_{1\times n_{pg}} * \left[\Big|\Big[u\big(x(P^T,1:n_e-2)\big)\Big]_{n_{pg}\times(n_e-2)} - \Big[phiPT\Big]_{n_{pg}\times 2} * \begin{bmatrix} c_0 & \ldots & c_{n_e-3} \\ c_1 & \ldots & c_{n_e-2} \end{bmatrix}_{2\times(n_e-2)}\Big| .\wedge 2\right]_{n_{pg}\times(n_e-2)}\right).
\end{aligned}$$

Observando as vetorizações acima, observe que podemos englobar o elemento 0 e o elemento $n_e - 1$ procedendo da seguinte forma:

$$\mathcal{E}_h^2 = \frac{h}{2} sum\left(\Big[W\Big]_{1\times n_{pg}} * \left[\Big|\Big[u\big(xPTne\big)\Big]_{n_{pg}\times n_e} - \Big[phiPT\Big]_{n_{pg}\times 2} * \begin{bmatrix} 0 & c_0 & \ldots & c_{n_e-3} & c_{n_e-2} \\ c_0 & c_1 & \ldots & c_{n_e-2} & 0 \end{bmatrix}_{2\times n_e}\Big| .\wedge 2\right]_{n_{pg}\times n_e}\right).$$

Nesta última expressão, resta observar que, definindo $d = \begin{bmatrix} c \\ 0 \end{bmatrix}_{(m+1)\times 1}$, podemos reescrever o resultado como:

$$\mathcal{E}_h^2 = \frac{h}{2} sum\left(\Big[W\Big]_{1\times n_{pg}} * \left[\Big|\Big[u\big(xPTne\big)\Big]_{n_{pg}\times n_e} - \Big[phiPT\Big]_{n_{pg}\times 2} * \big[d(EQoLG)\big]_{2\times n_e}\Big| .\wedge 2\right]_{n_{pg}\times n_e}\right).$$