Resolução do Problema Unidimensional com Dependência Temporal, Não Linearidade e Condições de Dirichlet via o Método dos Elementos Finitos

João Victor Lopez Pereira

18 de novembro de 2024

Rio de Janeiro - RJ

Resumo:

Este documento apresenta a resolução de um sistema genérico de equações diferenciais ordinárias, com uma componente não linear, utilizando o Método dos Elementos Finitos, conforme abordado nas aulas da disciplina *Introdução ao Método dos Elementos Finitos*, ministradas pelo prof. Dr. Marcello Goulart Teixeira na Universidade Federal do Rio de Janeiro, durante o segundo semestre de 2024. Neste documento, apresentamos a resolução de uma equação do calor em uma dimensão espacial, com dependência temporal e não linearidade.

Abstract:

This document presents the solution of a generic system of ordinary differential equations, with a non-linear component, using the Finite Elements Method, as covered in the course *Introduction to the Finite Element Method*, taught by professor Dr. Marcello Goulart Teixeira at the Federal University of Rio de Janeiro, during the second half of 2024. In this document, we present the solution of a heat equation in one spatial dimension, with temporal dependence and non-linearity.

Agradecimentos:

Agradeço ao professor Marcello Goulart, a Bruno Alves, Leonardo, Hashimoto e a vários outros colegas que me ajudaram no entendimento do conteúdo necessário para a realização das contas e do Método dos Elementos Finitos como um todo.

Thanks:

I would like to thank professor Marcello Goulart, Bruno Alves, Leonardo, Hashimoto, and several other colleagues who helped me understand the necessary content for carrying out the calculations and the Finite Element Method as a whole.

Sumário

Capítulo 1

Aspectos Teóricos

1.1 Definição da Formulação Forte

Dado constantes $\alpha > 0$, $\beta \geq 0$, $T > 0$, $f(x,t)$ tal que $x \in [0,1]$ e $t \in [0,T]$ e $g(x)$ tal que g é uma função não linear, queremos encontrar $u(x,t)$ tal que:

$$\begin{cases} u_t(x,t) - \alpha u_{xx}(x,t) + \beta u(x,t) + g(u) = f(x,t) \ \forall x \in]0,1[\\ u(0,t) = u(1,t) = 0 \\ u(x,0) = u_0(x) \end{cases}$$

1.2 Transição entre a Formulação Forte e Fraca

Visto que:

$$u_t(x,t) - \alpha u_{xx}(x,t) + \beta u(x,t) + g(u) = f(x,t)$$

Podemos multiplicar ambos por lados por uma função $v(x) \in V$ tal que:

$$v(1) = v(0) = 0$$

que nos ajude a eliminar a segunda derivada de u em x:

$$u_t(x,t) - \alpha u_{xx}(x,t) + \beta u(x,t) + g(u) = f(x,t)$$

$$[u_t(x,t) - \alpha u_{xx}(x,t) + \beta u(x,t) + g(u)]v(x) = f(x,t)v(x)$$

$$u_t(x,t)v(x) - \alpha u_{xx}(x,t)v(x) + \beta u(x,t)v(x) + g(u)v(x) = f(x,t)v(x)$$

$$\int_0^1 u_t(x,t)v(x)\,dx - \int_0^1 \alpha u_{xx}(x,t)v(x)\,dx + \int_0^1 \beta u(x,t)v(x)\,dx + \int_0^1 g(u)v(x)\,dx = \int_0^1 f(x,t)v(x)\,dx$$

$$\int_0^1 u_t(x,t)v(x)\,dx - \alpha \int_0^1 u_{xx}(x,t)v(x)\,dx + \beta \int_0^1 u(x,t)v(x)\,dx + \int_0^1 g(u)v(x)\,dx = \int_0^1 f(x,t)v(x)\,dx$$

Sabemos que dado funções f e g:

$$\int f(x)g'(x)dx = f(x)g(x) - \int f'(x)g(x)dx$$

Logo, realizando a integração por partes para eliminar a segunda derivada em x na equação:

$$\int_0^1 u_t(x,t)v(x)\,dx - \alpha \int_0^1 u_{xx}(x,t)v(x)\,dx + \beta \int_0^1 u(x,t)v(x)\,dx + \int_0^1 g(u)v(x)\,dx = \int_0^1 f(x,t)v(x)\,dx,$$

$$\int_0^1 u_t(x,t)v(x)\,dx - \alpha \left[u_x(x,t)v(x)\Big|_0^1 - \int_0^1 u_x(x,t)v_x(x)\,dx \right] + \beta \int_0^1 u(x,t)v(x)\,dx$$

$$+ \int_0^1 g(u)v(x)\,dx = \int_0^1 f(x,t)v(x)\,dx$$

$$\int_0^1 u_t(x,t)v(x)\,dx - \alpha \left[u_x(1,t)v(1) - u_x(0,t)v(0) - \int_0^1 u_x(x,t)v_x(x)\,dx \right] + \beta \int_0^1 u(x,t)v(x)\,dx$$

$$+ \int_0^1 g(u)v(x)\,dx = \int_0^1 f(x,t)v(x)\,dx,$$

$$\int_0^1 u_t(x,t)v(x)\,dx - \alpha \left[\cancel{u_x(1,t)v(1) - u_x(0,t)v(0)} - \int_0^1 u_x(x,t)v_x(x)\,dx \right] + \beta \int_0^1 u(x,t)v(x)\,dx$$

$$+ \int_0^1 g(u)v(x)\,dx = \int_0^1 f(x,t)v(x)\,dx,$$

$$\int_0^1 u_t(x,t)v(x)\,dx - \alpha \left[- \int_0^1 u_x(x,t)v_x(x)\,dx \right] + \beta \int_0^1 u(x,t)v(x)\,dx + \int_0^1 g(u)v(x)\,dx = \int_0^1 f(x,t)v(x)\,dx,$$

$$\int_0^1 u_t(x,t)v(x)\,dx + \alpha \int_0^1 u_x(x,t)v_x(x)\,dx + \beta \int_0^1 u(x,t)v(x)\,dx + \int_0^1 g(u)v(x)\,dx = \int_0^1 f(x,t)v(x)\,dx.$$

Definição de Notação

Definição: $(f, g) = \int_0^1 f(x)g(x)\,dx$

Definição: $\kappa(f, g) = \alpha \int_0^1 f_x(x)g_x(x)\,dx + \beta \int_0^1 f(x)g(x)\,dx$

Ou seja, nosso problema:

$$\int_0^1 u_t(x,t)v(x)\,dx + \alpha \int_0^1 u_x(x,t)v_x(x)\,dx + \beta \int_0^1 u(x,t)v(x)\,dx + \int_0^1 g(u)v(x)\,dx = \int_0^1 f(x,t)v(x)\,dx$$

Pode ser escrito como:

$$(u_t(t), v) + \kappa(u(t), v) + (g(u(t)), v) = (f(t), v)$$

1.3 Definição da Formulação Fraca

Dado constantes $\alpha > 0$, $\beta \geq 0$ e $T > 0$ e uma função $f(x,t)$ tal que $x \in [0,1]$, $t \in [0,T]$ e $g(x)$ uma função não linear, queremos encontrar $u(x,t) \in V$ tal que, $\forall v \in V$:

$$(u_t(t), v) + \kappa(u(t), v) + (g(u(t)), v) = (f(t), v)$$

é válido.

1.4 Problema Aproximado Totalmente Discreto via o método de Crank-Nicolson Galerkin linearizado

1.4.1 Problema Variacional no Ponto Médio

Discretizaremos o intervalo $[0,T]$ em $t_0, t_1, ..., t_N$ tal que $t_n - t_{n-1} = \tau$, $\forall n \in [0,N]$.

$$(u_t(t_{n-\frac{1}{2}}), v) + \kappa(u(t_{n-\frac{1}{2}}), v) + (g(u(t_{n-\frac{1}{2}})), v) = (f(t_{n-\frac{1}{2}}), v)$$

Tal que:

$$t_{n-\frac{1}{2}} = \frac{t_n + t_{n-1}}{2}$$

é o ponto médio do intervalo $[t_{n-1}, t_n]$.

1.4.2 Diferenças Finitas no Tempo

Seja:

$$u_t(t_{n-\frac{1}{2}}) = \frac{u(t_n) - u(t_{n-1})}{\tau} + O(\tau^2)$$

$$u(t_{n-\frac{1}{2}}) = \frac{u(t_n) + u(t_{n-1})}{2} + O(\tau^2)$$

$$u(t_{n-\frac{1}{2}}) = \frac{3u(t_{n-1}) - u(t_{n-2})}{2} + O(\tau^2)$$

Substituindo em nossa equação:

$$\left(\frac{u(t_n) - u(t_{n-1})}{\tau}, v\right) + \kappa\left(\frac{u(t_n) + u(t_{n-1})}{2}, v\right) + \left(g\left(\frac{3u(t_{n-1}) - u(t_{n-2})}{2}\right), v\right) \approx (f(t_{n-\frac{1}{2}}), v)$$

Visto que nosso sistema agora não é exato — por conta das parcelas $O(\tau^2)$ que foram desprezadas — queremos determinar $U^n \in V$ tal que $U^n \approx u(t_n)$ e seja solução da equação:

$$\left(\frac{U^n - U^{n-1}}{\tau}, v\right) + \kappa\left(\frac{U^n + U^{n-1}}{2}, v\right) + \left(g\left(\frac{3U(t_{n-1}) - U(t_{n-2})}{2}\right), v\right) = (f(t_{n-\frac{1}{2}}), v)$$

1.4.3 Problema Aproximado

Precisamos determinar $U_h^n \in V_m$ tal que:

$$\forall v_h \in V_m$$

$$\left(\frac{U_h^n - U_h^{n-1}}{\tau}, v_h\right) + \kappa\left(\frac{U_h^n + U_h^{n-1}}{2}, v_h\right) + \left(g\left(\frac{3U_h^{n-1} - U_h^{n-2}}{2}\right), v_h\right) = (f(t_{n-\frac{1}{2}}), v_h)$$

1.5 Definição do Problema Aproximado Totalmente Discreto

Dado constantes $\alpha > 0$, $\beta \geq 0$ e $T > 0$, $f(x,t)$ tal que $x \in [0,1]$ e $t \in [0,T]$, $g(x)$ uma função não linear, e $U_{0h} \in V_m$, queremos encontrar U_h tal que:

$$\forall v_h \in V_m$$

$$\left(\frac{U_h^n - U_h^{n-1}}{\tau}, v_h\right) + \kappa\left(\frac{U_h^n + U_h^{n-1}}{2}, v_h\right) + \left(g\left(\frac{3U_h^{n-1} - U_h^{n-2}}{2}\right), v_h\right) = (f(t_{n-\frac{1}{2}}), v_h)$$

1.6 Transição entre o Problema Aproximado e a Forma Matriz-vetor

Seja

$$U_h^n(x) = \sum_{j=1}^{m} c_j^n \varphi_j(x)$$

Tomando $v_h = \varphi_i$, $i \in [1, m]$:

$$\left(\frac{\left[\sum_{j=1}^m c_j^n - \sum_{j=1}^m c_j^{n-1}\right]\varphi_j}{\tau}, \varphi_i\right) + k\left(\frac{\left[\sum_{j=1}^m c_j^n + \sum_{j=1}^m c_j^{n-1}\right]\varphi_j}{2}, \varphi_i\right) + \left(g\left(\frac{\left[3\sum_{j=1}^m c_j^{n-1} - \sum_{j=1}^m c_j^{n-2}\right]\varphi_j}{2}\right), \varphi_i\right) = (f(t_{n-\frac{1}{2}}), \varphi_i)$$

$$\sum_{j=1}^{m}\left(\frac{[c_j^n - c_j^{n-1}]\varphi_j}{\tau}, \varphi_i\right) + \kappa\sum_{j=1}^{m}\left(\frac{[c_j^n + c_j^{n-1}]\varphi_j}{2}, \varphi_i\right) + \left(g\left(\frac{\sum_{j=1}^m [3c_j^{n-1} - c_j^{n-2}]\varphi_j}{2}\right), \varphi_i\right) = (f(t_{n-\frac{1}{2}}), \varphi_i)$$

$$\sum_{j=1}^{m}\left[\frac{c_j^n - c_j^{n-1}}{\tau}(\varphi_j, \varphi_i)\right] + \sum_{j=1}^{m}\left[\frac{c_j^n + c_j^{n-1}}{2}\kappa(\varphi_j, \varphi_i)\right] + \left(g\left(\frac{\sum_{j=1}^m [3c_j^{n-1} - c_j^{n-2}]\varphi_j}{2}\right), \varphi_i\right) = (f(t_{n-\frac{1}{2}}), \varphi_i)$$

Perceba que, ao variarmos i e j temos:

$$\begin{cases} ... + (\varphi_j, \varphi_1)\dfrac{c_j^n - c_j^{n-1}}{\tau} + \kappa(\varphi_j, \varphi_1)\dfrac{c_j^n + c_j^{n-1}}{2} + \left(g\left(\dfrac{\sum_{j=1}^m [3c_j^{n-1} - c_j^{n-2}]\varphi_j}{2} \right), \varphi_i \right) + ... = (f(t_{n-\frac{1}{2}}), \varphi_1) \\ \vdots \\ ... + (\varphi_j, \varphi_k)\dfrac{c_j^n - c_j^{n-1}}{\tau} + \kappa(\varphi_j, \varphi_k)\dfrac{c_j^n + c_j^{n-1}}{2} + \left(g\left(\dfrac{\sum_{j=1}^m [3c_j^{n-1} - c_j^{n-2}]\varphi_j}{2} \right), \varphi_k \right) + ... = (f(t_{n-\frac{1}{2}}), \varphi_k) \\ \vdots \\ ... + (\varphi_j, \varphi_m)\dfrac{c_j^n - c_j^{n-1}}{\tau} + \kappa(\varphi_j, \varphi_m)\dfrac{c_j^n + c_j^{n-1}}{2} + \left(g\left(\dfrac{\sum_{j=1}^m [3c_j^{n-1} - c_j^{n-2}]\varphi_j}{2} \right), \varphi_m \right) + ... = (f(t_{n-\frac{1}{2}}), \varphi_m) \end{cases}$$

Perceba que podemos organizar essas equações em produtos matrix-vetor tal que:

$$\begin{pmatrix} (\varphi_1, \varphi_1) & ... & (\varphi_m, \varphi_1) \\ \vdots & \ddots & \vdots \\ (\varphi_1, \varphi_m) & ... & (\varphi_m, \varphi_m) \end{pmatrix} \frac{c^n - c^{n-1}}{\tau} + \begin{pmatrix} \kappa(\varphi_1, \varphi_1) & ... & \kappa(\varphi_m, \varphi_1) \\ \vdots & \ddots & \vdots \\ \kappa(\varphi_1, \varphi_m) & ... & \kappa(\varphi_m, \varphi_m) \end{pmatrix} \frac{c^n + c^{n-1}}{2}$$
$$+ \begin{pmatrix} \left(g\left(\dfrac{\sum_{j=1}^m [3c_j^{n-1} - c_j^{n-2}]\varphi_j}{2} \right), \varphi_1 \right) \\ \vdots \\ \left(g\left(\dfrac{\sum_{j=1}^m [3c_j^{n-1} - c_j^{n-2}]\varphi_j}{2} \right), \varphi_m \right) \end{pmatrix} = \begin{pmatrix} (f(t_{n-\frac{1}{2}}), \varphi_1) \\ \vdots \\ (f(t_{n-\frac{1}{2}}), \varphi_m) \end{pmatrix}$$

Definição de Notação

Definição: $\mathcal{M} = \begin{pmatrix} (\varphi_1, \varphi_1) & ... & (\varphi_m, \varphi_1) \\ \vdots & \ddots & \vdots \\ (\varphi_1, \varphi_m) & ... & (\varphi_m, \varphi_m) \end{pmatrix}$

Definição: $\mathcal{K} = \begin{pmatrix} \kappa(\varphi_1, \varphi_1) & ... & \kappa(\varphi_m, \varphi_1) \\ \vdots & \ddots & \vdots \\ \kappa(\varphi_1, \varphi_m) & ... & \kappa(\varphi_m, \varphi_m) \end{pmatrix}$

Definição: $\mathcal{G}(c) = \begin{pmatrix} (g(\sum_{j=1}^m c_j\varphi_j), \varphi_1) \\ \vdots \\ (g(\sum_{j=1}^m c_j\varphi_j), \varphi_m) \end{pmatrix}$

Definição: $\mathcal{F}^{n-\frac{1}{2}} = \begin{pmatrix} (f(t_{n-\frac{1}{2}}), \varphi_1) \\ \vdots \\ (f(t_{n-\frac{1}{2}}), \varphi_m) \end{pmatrix}$

1.7 Definição do Problema na Forma Matriz-vetor

Dado matrizes $\mathcal{M}$, $\mathcal{K}$, vetor $\mathcal{F}^{n-\frac{1}{2}}$ e uma função $\mathcal{G}(c)$, queremos encontrar C^n tal que:

$$\mathcal{M}\frac{C^n - C^{n-1}}{\tau} + \mathcal{K}\frac{C^n + C^{n-1}}{2} + \mathcal{G}\left(\frac{3C^{n-1} - C^{n-2}}{2}\right) = \mathcal{F}^{n-\frac{1}{2}}$$

Em breve falaremos melhor sobre os casos $n = 0$ e $n = 1$.

Continuando as contas:

$$\mathcal{M}\frac{C^n - C^{n-1}}{\tau} + \mathcal{K}\frac{C^n + C^{n-1}}{2} + \mathcal{G}\left(\frac{3C^{n-1} - C^{n-2}}{2}\right) = \mathcal{F}^{n-\frac{1}{2}}$$

$$\mathcal{M}2\tau\frac{C^n - C^{n-1}}{\tau} + \mathcal{K}2\tau\frac{C^n + C^{n-1}}{2} + 2\tau\mathcal{G}\left(\frac{3C^{n-1} - C^{n-2}}{2}\right) = 2\tau\mathcal{F}^{n-\frac{1}{2}}$$

$$\mathcal{M}2\cancel{\tau}\frac{C^n - C^{n-1}}{\cancel{\tau}} + \mathcal{K}\cancel{2}\tau\frac{C^n + C^{n-1}}{\cancel{2}} + 2\tau\mathcal{G}\left(\frac{3C^{n-1} - C^{n-2}}{2}\right) = 2\tau\mathcal{F}^{n-\frac{1}{2}}$$

$$\mathcal{M}2[C^n - C^{n-1}] + \mathcal{K}\tau[C^n + C^{n-1}] + 2\tau\mathcal{G}\left(\frac{3C^{n-1} - C^{n-2}}{2}\right) = 2\tau\mathcal{F}^{n-\frac{1}{2}}$$

$$2\mathcal{M}C^n - 2\mathcal{M}C^{n-1} + \tau\mathcal{K}C^n + \tau\mathcal{K}C^{n-1} + 2\tau\mathcal{G}\left(\frac{3C^{n-1} - C^{n-2}}{2}\right) = 2\tau\mathcal{F}^{n-\frac{1}{2}}$$

Colocando as equações em um formato que seja resolvível por sistema linear:

$$2\mathcal{M}C^n + \tau\mathcal{K}C^n = 2\tau\mathcal{F}^{n-\frac{1}{2}} + 2\mathcal{M}C^{n-1} - \tau\mathcal{K}C^{n-1} - 2\tau\mathcal{G}\left(\frac{3C^{n-1} - C^{n-2}}{2}\right)$$

$$\left[2\mathcal{M} + \tau\mathcal{K}\right]C^n = 2\tau\mathcal{F}^{n-\frac{1}{2}} + \left[2\mathcal{M} - \tau\mathcal{K}\right]C^{n-1} - 2\tau\mathcal{G}\left(\frac{3C^{n-1} - C^{n-2}}{2}\right)$$

$$\left[\mathcal{M} + \left(\frac{\tau}{2}\right)\mathcal{K}\right]C^n = \tau\mathcal{F}^{n-\frac{1}{2}} + \left[\mathcal{M} - \left(\frac{\tau}{2}\right)\mathcal{K}\right]C^{n-1} - \tau\mathcal{G}\left(\frac{3C^{n-1} - C^{n-2}}{2}\right)$$

$$\left[\mathcal{M} + \left(\frac{\tau}{2}\right)\mathcal{K}\right]C^n = \tau\mathcal{F}^{n-\frac{1}{2}} + \left[\mathcal{M} - \left(\frac{\tau}{2}\right)\mathcal{K}\right]C^{n-1} - \tau\mathcal{G}\left(\frac{3C^{n-1} - C^{n-2}}{2}\right)$$

Seja $\mathcal{A} = \left[\mathcal{M} + \left(\frac{\tau}{2}\right)\mathcal{K}\right]$.

Seja $\mathcal{B} = \left[\mathcal{M} - \left(\frac{\tau}{2}\right)\mathcal{K}\right]$.

Sendo assim, temos:

$$\mathcal{A}C^n = \tau\mathcal{F}^{n-\frac{1}{2}} + \mathcal{B}C^{n-1} - \tau\mathcal{G}\left(\frac{3C^{n-1} - C^{n-2}}{2}\right)$$

$$C^n = \mathcal{A} \setminus \tau\mathcal{F}^{n-\frac{1}{2}} + \mathcal{B}C^{n-1} - \tau\mathcal{G}\left(\frac{3C^{n-1} - C^{n-2}}{2}\right)$$

Mas veja que ao utilizarmos $u(t_{n-\frac{1}{2}}) = \dfrac{3u(t_{n-1}) - u(t_{n-2})}{2} + O(\tau^2)$ — para calcularmos C^n — estamos utilizando os valores de C^{n-1} e C^{n-2}. Mas veja que isso nos restringe a $n \geq 2$, visto que para calcular os valores de C^0 e C^1 teríamos problema. Sendo assim, faremos:

para $n = 0$: Temos diversas opções de inicialização. Veremos quatro delas em breve no documento. Por enquanto, assuma que já tenhamos tenhamos C^0.

para $n = 1$:

$$\hat{C} = \mathcal{A} \setminus \tau\mathcal{F}^{n-\frac{1}{2}} + \mathcal{B}C^{n-1} - \tau\mathcal{G}(C^{n-1})$$

Tal que $\hat{C}$ seja apenas um valor intermediário auxiliar que não utilizaremos de fato em nossa aproximação. Perceba que $\hat{C}$ apresenta ordem de erro de $O(\tau)$. por isso, no caso $n = 1$, utilizaremos $\hat{C}$ e C^{n-1} para estimar C^n visto que não temos C^{n-2} (pois não existe):

$$C^n = \mathcal{A} \setminus \tau\mathcal{F}^{n-\frac{1}{2}} + \mathcal{B}C^{n-1} - \tau\mathcal{G}\left(\frac{\hat{C} + C^{n-1}}{2}\right)$$

Dessa forma, C^n apresenta erro na ordem de $O(\tau^2)$. Ou seja, nosso método continua tendo erro na ordem de $O(\tau^2)$.

para $n \geq 2$:

$$C^n = \mathcal{A} \setminus \tau\mathcal{F}^{n-\frac{1}{2}} + \mathcal{B}C^{n-1} - \tau\mathcal{G}\left(\frac{3C^{n-1} - C^{n-2}}{2}\right)$$

Que é como calculamos anteriormente. Visto que em $n \geq 2$, já se tem os valores de ao menos 2 pontos anteriores.

1.8 Inicialização de C0

Primeira Opção

Visto que U^0 é dado de entrada e sabemos que:

$$U^0(x) = \sum_{j=1}^{m} C_j^0 \varphi_j(x)$$

Visto que nossa escolha de $\varphi_j(x)$ vale 1 nos pontos da discretização, logo:

$$C_i^0 = U^0(x)$$

$$C^0 = \begin{pmatrix} u_0(x_1) \\ \vdots \\ u_0(x_m) \end{pmatrix}$$

1.8.1 Segunda Opção

Seja $U^0 \in V_m$ tal que:

$$(U^0 - u_0, v_h) = 0 \quad \forall v_h \in V_m$$

Tomando $U^0(x) = \sum_{j=1}^{m} C_j^0 \varphi_j(x)$ e $v_h = \varphi_i$ para $i \in [1, m]$, temos:

$$(\sum_{j=1}^{m} C_j^0 \varphi_j(x) - u_0, \varphi_i) = 0$$

$$\sum_{j=1}^{m} C_j^0 (\varphi_j(x) - u_0, \varphi_i) = 0$$

$$\sum_{j=1}^{m} C_j^0 (\varphi_j(x), \varphi_i) - (u_0, \varphi_i) = 0$$

$$\sum_{j=1}^{m} C_j^0 (\varphi_j(x), \varphi_i) = (u_0, \varphi_i)$$

Que pode ser escrito na forma matriz-vetor tal que, usando a notação definida anteriormente, temos:

$$MC^0 = \begin{pmatrix} (u_0, \varphi_1) \\ \vdots \\ (u_0, \varphi_m) \end{pmatrix}$$

1.8.2 Terceira Opção

Seja $U^0 \in V_m$ tal que:

$$((U^0 - u_0)_x, v_{hx}) = 0 \quad \forall v_h \in V_m$$

Tomando $U^0(x) = \sum_{j=1}^{m} C_j^0 \varphi_j(x)$ e $v_h = \varphi_i$ para $i \in [1, m]$, temos:

$$((\sum_{j=1}^{m} C_j^0 \varphi_j - u_0)_x, \varphi_{ix}) = 0$$

$$\sum_{j=1}^{m} C_j^0((\varphi_j - u_0)_x, \varphi_{ix}) = 0$$

$$\sum_{j=1}^{m} C_j^0(\varphi_{jx} - u_{0x}, \varphi_{ix}) = 0$$

$$\sum_{j=1}^{m} C_j^0(\varphi_{jx}, \varphi_{ix})(-u_{0x}, \varphi_{ix}) = 0$$

$$\sum_{j=1}^{m} C_j^0(\varphi_{jx}, \varphi_{ix}) = (u_{0x}, \varphi_{ix})$$

Que pode ser escrito na forma matriz-vetor tal que, usando a notação definida anteriormente, temos:

$$\begin{pmatrix} (\varphi_{1x}, \varphi_{1x}) & \cdots & (\varphi_{mx}, \varphi_{1x}) \\ \vdots & \ddots & \vdots \\ (\varphi_{1x}, \varphi_{mx}) & \cdots & (\varphi_{mx}, \varphi_{mx}) \end{pmatrix} C^0 = \begin{pmatrix} (u_{0x}, \varphi_{1x}) \\ \vdots \\ (u_{0x}, \varphi_{mx}) \end{pmatrix}$$

1.8.3 Quarta Opção

Seja $U^0 \in V_m$ tal que:

$$\kappa(U^0 - u_0, v_{hx}) = 0 \quad \forall v_h \in V_m$$

Tomando $U^0(x) = \sum_{j=1}^{m} C_j^0 \varphi_j(x)$ e $v_h = \varphi_i$ para $i \in [1, m]$, temos:

$$\kappa(\sum_{j=1}^{m} C_j^0 \varphi_j - u_0, \varphi_i) = 0$$

$$\sum_{j=1}^{m} C_j^0 \kappa(\varphi_j - u_0, \varphi_i) = 0$$

$$\sum_{j=1}^{m} C_j^0 \kappa(\varphi_j, \varphi_i) + \kappa(-u_0, \varphi_i) = 0$$

$$\sum_{j=1}^{m} C_j^0 \kappa(\varphi_j, \varphi_i) = \kappa(u_0, \varphi_i)$$

Que pode ser escrito na forma matriz-vetor tal que, usando a notação definida anteriormente, temos:

$$\mathcal{K}C^0 = \begin{pmatrix} \kappa(u_0, \varphi_1) \\ \vdots \\ \kappa(u_0, \varphi_m) \end{pmatrix}$$

Capítulo 2

Detalhes de Implementação

2.1 Inicialização de C0

2.1.1 Primeira Opção

Vimos que a primeira opção para inicializar C^0 é:

$$C^0 = \begin{pmatrix} u_0(h) \\ \vdots \\ u_0(1-h) \end{pmatrix}$$

que é facilmente implementável como veremos a seguir.

2.1.2 Segunda Opção

Vimos que a segunda opção para inicializar C^0 é:

$$MC^0 = \begin{pmatrix} (u_0, \varphi_1) \\ \vdots \\ (u_0, \varphi_m) \end{pmatrix}$$

$$C^0 = M \setminus \begin{pmatrix} (u_0, \varphi_1) \\ \vdots \\ (u_0, \varphi_m) \end{pmatrix}$$

A matriz M pode ser facilmente inicializada utilizando o inicializador da K e o vetor da parte direita da equação pode ser facilmente inicializado utilizando construtor da F, como veremos em breve.

2.1.3 Terceira Opção

Vimos que a terceira opção para inicializar C^0 é:

$$\begin{pmatrix} (\varphi_{1x}, \varphi_{1x}) & \cdots & (\varphi_{mx}, \varphi_{1x}) \\ \vdots & \ddots & \vdots \\ (\varphi_{1x}, \varphi_{mx}) & \cdots & (\varphi_{mx}, \varphi_{mx}) \end{pmatrix} C^0 = \begin{pmatrix} (u_{0x}, \varphi_{1x}) \\ \vdots \\ (u_{0x}, \varphi_{mx}) \end{pmatrix}$$

$$C^0 = \begin{pmatrix} (\varphi_{1x}, \varphi_{1x}) & \cdots & (\varphi_{mx}, \varphi_{1x}) \\ \vdots & \ddots & \vdots \\ (\varphi_{1x}, \varphi_{mx}) & \cdots & (\varphi_{mx}, \varphi_{mx}) \end{pmatrix} \setminus \begin{pmatrix} (u_{0x}, \varphi_{1x}) \\ \vdots \\ (u_{0x}, \varphi_{mx}) \end{pmatrix}$$

Essa matriz pode ser facilmente inicializada utilizando o inicializador da K e o vetor teremos que fazer um construtor próprio. Cada termo desse vetor é:

$$\begin{aligned} \int_0^1 u_{0x}(x)\varphi_{ax}(x)\,dx &= \int_{-1}^1 u_{0x}(x(\xi,e))\varphi_{ax}^e(x(\xi,e))\frac{h}{2}\frac{2}{h}\,d\xi \\ &= \int_{-1}^1 u_{0x}(x(\xi,e))\varphi_{ax}^e(x(\xi,e))\,d\xi \\ &= \int_{-1}^1 u_{0x}(x(\xi,e))\phi_{ax}(\xi)\,d\xi \end{aligned}$$

Sendo assim, podemos fazer seu inicializador como veremos em breve.

2.1.4 Quarta Opção

Vimos que a quarta opção para inicializar C^0 é:

$$\mathcal{K}C^0 = \begin{pmatrix} \kappa(u_0, \varphi_1) \\ \vdots \\ \kappa(u_0, \varphi_m) \end{pmatrix}$$

A matriz K pde ser facilmente inicializada usando seu próprio construtor e o vetor teremos que realizar as contas. Cada um de seus termos é:

$$
\begin{aligned}
\kappa(u_0, \varphi_i) &= \alpha \int_0^1 u_{0x}(x)\varphi_{ix}(x)\,dx + \beta \int_0^1 u_0(x)\varphi_i(x)\,dx \\
&= \alpha \int_{-1}^1 u_{0x}(x(\xi,e))\varphi_{ix}((\xi,e))\frac{h}{2}\,d\xi + \beta \int_{-1}^1 u_0(x(\xi,e))\varphi_i(x(\xi,e))\frac{h}{2}\,d\xi \\
&= \alpha \int_{-1}^1 u_{0x}(x(\xi,e))\phi_{ix}(\xi)\frac{2}{h}\frac{h}{2}\,d\xi + \beta \int_{-1}^1 u_0(x(\xi,e))\phi_i(\xi)\frac{h}{2}\,d\xi \\
&= \alpha \int_{-1}^1 u_{0x}(x(\xi,e))\phi_{ix}(\xi)\,d\xi + \beta\frac{h}{2} \int_{-1}^1 u_0(x(\xi,e))\phi_i(\xi)\,d\xi
\end{aligned}
$$

Mas veja que a parte esquerda da equação é o C^0 que calculamos para a terceira opção multiplicado pela constante α. Além disso, veja que a parte direita da equação pode ser calculada utilizando o construtor da F multiplicado pela constante β.

2.1.5 Implementação

```
# Initializes the C0 third option vector
function init_C0_3rd_option_vector(u0x, ne, EQ, LG, m)

  # Initializes the C0e vector
  function init_C0e_3rd_option_vector(u0x, ne, e)
    C0e = zeros(2)
    h = 1 / ne

    for a in 1:2
      C0e[a] = gaussian_quadrature((qsi) -> u0x(qsi_to_x(qsi, e, h)) *  d_phi(a, qsi), 5)
    end

    return C0e
  end

  C0 = zeros(m+1)

  for e in 1:ne
    C0e = init_C0e_3rd_option_vector(u0x, ne, e)
    for a in 1:2
      C0[EQ[LG[a,e]]] += C0e[a]
    end
  end

  # Removes the last line
  return C0[1:m]
end

# Initializes the C0 vector
function init_C0_vector(option, u0, u0x, ne, EQ, LG, alpha, beta, gamma, m)
```

```
  if (option == 1)
    h = 1 / ne
    return u0.(h:h:1-h)

  elseif (option == 2)
    M = init_K_matrix(ne, EQ, LG, 0, 1, 0, m)
    b = init_F_vector(u0, ne, EQ, LG, m)
    return M \ b

  elseif (option == 3)
    A = init_K_matrix(ne, EQ, LG, 1, 0, 0, m)
    b = init_C0_3rd_option_vector(u0x, ne, EQ, LG, m)
    return A \ b

  elseif (option == 4)
    K = init_K_matrix(ne, EQ, LG, alpha, beta, gamma, m)
    b = (alpha .* init_C0_3rd_option_vector(u0x, ne, EQ, LG, m)) + (beta .*
        init_F_vector(u0, ne, EQ, LG, m))
    return K \ b

  else
    error("option not recognized in init_C0_vector")
  end
end
```

2.2 Inicialização das Matrizes

As matrizes $\mathcal{A}$ e $\mathcal{B}$ são derivadas das matrizes $\mathcal{M}$ e $\mathcal{K}$. Veja que, na verdade, a matriz $\mathcal{M}$ pode ser construída utilizando o próprio construtor da $\mathcal{K}$. Visto que cada elemento de $\mathcal{K}$ é:

$$\kappa(\varphi_i, \varphi_j)$$

E cada elemento de $\mathcal{M}$ é:

$$(\varphi_i, \varphi_j)$$

Podemos utilizar o inicializador da $\mathcal{K}$ com parâmetros $\alpha = 0$, $\beta = 1$ e $\gamma = 0$ para inicializarmos $\mathcal{M}$ corretamente. O construtor da $\mathcal{K}$ é o mesmo do documento anterior que pode ser encontrado em [3].

2.3 Inicialização dos Vetores

Faltou mostrar como os vetores $\mathcal{F}$ e $\mathcal{G}$ são inicializados. O construtor de $\mathcal{F}$ é o mesmo encontrado em [3] (visto que as contas são as mesmas).

Pela definição de $\mathcal{G}$, cada um de seus termos é:

$$(g(\sum_{j=1}^{m} c_j\varphi_j), \varphi_i)$$

Mas dado um termo i, veja que o vetor irá resultar em 0 em grande parte das execuções do somatório visto que φ é uma base pequena. Sendo assim, podemos escrever isso como:

$$\begin{aligned}\int_0^1 g(\sum_{j=1}^{m} c_j\varphi_j(x))\varphi_i(x)\,dx &= \int_{x_1^e}^{x_2^e} g(c_{[EQ[LG[1,e]]]}\varphi_1^e(x) + c_{[EQ[LG[2,e]]]}\varphi_2^e(x))\varphi_a^e(x)\,dx \\ &= \int_{-1}^{1} g(c_{[EQ[LG[1,e]]]}\varphi_1(x(\xi,e)) + c_{[EQ[LG[2,e]]]}\varphi_2(x(\xi,e)))\varphi_a(x(\xi,e))\frac{h}{2}\,d\xi \\ &= \frac{h}{2}\int_{-1}^{1} g(c_{[EQ[LG[1,e]]]}phi_1(\xi) + c_{[EQ[LG[2,e]]]}phi_2(\xi))\phi_a(x(\xi,e))\,d\xi\end{aligned}$$

Que em código fica:

```
# Initializes the G vector
function init_G_vector(g, ne, EQ, LG, C, m, h)

  # Initializes the Ge vector
  function init_Ge_vector(g, e, EQ, LG, C, h)
    Ge = zeros(2)

    for a in 1:2
      Ge[a] = h / 2 * gaussian_quadrature((qsi) -> g(C[EQ[LG[1, e]]] * phi(1, qsi) +
          C[EQ[LG[2, e]]] * phi(2, qsi)) * phi(a, qsi), 5)
    end

    return Ge
  end

  G = zeros(m+1)
  ext_C = [C ; 0]

  for e in 1:ne
    Ge = init_Ge_vector(g, e, EQ, LG, ext_C, h)
    for a in 1:2
      G[EQ[LG[a,e]]] += Ge[a]
    end
  end

  # Removes the last line
  return G[1:m]
end
```

Bibliografia

[1] Bruno Alves do Carmo. *Elementos Finitos*. Acessado em 2 de Outubro de 2024. URL: https://github.com/bacarmo/Elementos_Finitos.

[2] Thomas J. R. Hughes. *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*. Dover Civil and Mechanical Engineering. Dover Publications, 2000. ISBN: 0486411818; 9780486411811. URL: libgen.li/file.php?md5=6ccd7300b50cdf2ec244ba5c248c4bc1.

[3] João Victor Lopez Pereira. *Finite-Elements-Method*. Acessado em 20 de Outubro de 2024. URL: https://github.com/joaovictorlopezpereira/Finite-Elements-Method.