Polinômios de Bernstein para Análise de Elementos Finitos

Lucas B. Andrade

Novembro 2018

Conteúdo

1 Introdução

Neste trabalho são estudados os **polinômios de Bernstein** para utilização no método de elementos finitos. Este estudo se baseia principalmente nos desenvolvimentos realizados em [3].

Um estudo sobre o método de elementos finitos em uma dimensão e a utilização destes polinômios neste método foi realizado em [5].

O intuito de utilizar os polinômios de Bernstein para o método de elementos finitos seria de aumentar a eficiência dos algoritmos que envolvem o cálculo das matrizes que levarão à solução dada pelo método.

Tipicamente a matriz de um elemento, sobre $\mathbb{P}_d^n$, possui $\mathcal{O}(n^{2d})$ entradas. Desta maneira, se considerarmos integração numérica para calcular cada uma dessas entradas, com $\mathcal{O}(n^d)$ pontos de quadratura, ficamos com o custo de $\mathcal{O}(n^{3d})$ para calcular a matriz de um elemento.

Neste estudo construiremos algoritmos que fazem o cálculo da matriz de um elemento com complexidade $\mathcal{O}(n^{2d})$ para o caso em que $d = 2$, em elementos triangulares e quadriláteros. Para o caso em que d é arbitrário, veja [3].

2 Os Polinômios de Bernstein

Os polinômios de Bernstein em $[0,1]$ são definidos como

$$b_i^n(x) = \binom{n}{i} x^i \cdot (1-x)^{n-i} \tag{1}$$

Em que $\binom{n}{i} = \frac{n!}{i!(n-i)!}$ é o coeficiente binomial. Vejamos algumas de suas propriedades que utilizaremos aqui:

- P1. Multiplicação:

$$b_i^n(x) b_j^m(x) = \frac{\binom{n}{i}\binom{n}{j}}{\binom{n+m}{i+j}} b_{i+j}^{n+m}(x) \tag{2}$$

- P2. Derivada:

$$\frac{d}{dx} b_i^n(x) = n\big(b_{i-1}^{n-1}(x) - b_i^{n-1}(x)\big) \tag{3}$$

Além disso pode-se provar que o conjunto $\beta := \{B_i^n(x); i \in 0, \ldots, n\}$ é uma base para o espaço de polinômios de grau n, $\mathbb{P}_n$.

Podemos estender o domínio à $[0,1]^2$, por exemplo, com um produto tensorial, obtendo

$$B_{\vec{\alpha}}^{\vec{n}}(\vec{x}) = b_{\alpha_1}^{n_1}(x_1) b_{\alpha_2}^{n_2}(x_2) \tag{4}$$

Em que $\vec{\alpha} \in I_n^2 := \{\vec{\alpha};\ \alpha_i \in \{0, \ldots, n\} \text{ e } i \in \{1,2\}\}$.

Nos interessa também a definição desses polinômios em um *simplex* bidimensional (triângulo), para isso, primeiramente definimos $\mathcal{I}_n^2 := \{\vec{\alpha};\ \alpha_i \in \{0, \ldots, n - \sum_{k=1}^{i-1} \alpha_i\} \text{ e } i \in \{1,2,3\}\}$, um espaço de índices. E, dado um triângulo $\mathbf{T}$, não degenerado, de vértices $\vec{v_1},\ \vec{v_2},\ \vec{v_3}$, definimos as coordenadas baricêntricas com relação a este triângulo, como sendo $\vec{\lambda} = (\lambda_1,\ \lambda_2,\ \lambda_3)$, em que $\sum_{i=1}^{3} \lambda_i = 1$, de tal forma que um ponto $\vec{x}$ no plano possa ser escrito como

$$\vec{x} = \lambda_1 \vec{v_1} + \lambda_2 \vec{v_2} + \lambda_3 \vec{v_2} \tag{5}$$

Observe que a cardinalidade de $\mathcal{I}_n^2$ é $\binom{n+2}{2}$.

Desta forma define-se os polinômios de Bernstein em um triângulo como sendo:

$$\mathbf{B}_{\vec{\alpha}}^n(\vec{x}) = \prod_{i=1}^{3} \binom{n}{\alpha_i} \lambda_i^{\alpha_i} \tag{6}$$

Tranformação de Duffy e regra de Stroud

Para trabalhar com elementos triangulares utilizamos da transformação de Duffy, definida como sendo a transformação que leva pontos de $[0,1]^2$ em um triângulo $\mathbf{T}$ de vértices $\vec{v_1}, \vec{v_2}, \vec{v_3}$.

$$\vec{x}(\xi, \eta) = \gamma_1 \vec{v_1} + \gamma_2 \vec{v_2} + \gamma_3 \vec{v_3} \tag{7}$$

em que $\xi, \eta \in [0,1]$ e

$$\begin{aligned} \gamma_1 &= \xi \\ \gamma_2 &= (1-\gamma_1)\eta \\ \gamma_3 &= 1 - \gamma_2 - \gamma_1 \end{aligned}$$

Através da transformação de Duffy podemos escrever o polinômio de Bernstein definido no triângulo como função dos polinômios de Bernstein unidimensionais. [3]

Lema 1 *Seja $\vec{\alpha} \in \mathcal{I}_n^2$ e $\vec{x}(\xi, \eta)$ a transformação de Duffy, então*

$$\mathbf{B}_{\vec{\alpha}}^n(\vec{x}(\xi, \eta)) = b_{\alpha_1}^n(\xi) b_{\alpha_2}^{n-\alpha_1}(\eta)$$

Verifique a prova deste lema em [3].

Além disso, utilizando a transformação de Duffy, podemos calcular a integral de uma função f sobre um triângulo $\mathbf{T}$ pela regra de quadratura de Stroud [3]

$$\int_{\mathbf{T}} f(\vec{x}) d\vec{x} = \frac{|\mathbf{T}|}{2} \int_0^1 (1-\xi) \int_0^1 f\big(\vec{x}(\xi, \eta)\big) \, d\eta d\xi \tag{8}$$

Desta forma se torna interessante utilizar a quadratura de Gauss-Jacobi

$$\int_0^1 (1-x)^a x^b f(x) \, dx \approx \sum_{i=0}^{q} \omega_i^{(a,b)} \xi_i^{(a,b)} \tag{9}$$

A qual, com q pontos de integração, tem precisão $2q-1$, pesos $\omega_i^{(a,b)}$ positivos e pontos $\xi_i^{(a,b)}$ dentro do intervalo $[0, 1]$, o caso $a = b = 0$ se torna equivalente à quadratura de Gauss-Legendre. Podemos aproximar a integral de Stroud (8) utilizando a quadratura de Gauss-Jacobi fazendo

$$\int_{\mathbf{T}} f(\vec{x}) \, d\vec{x} \approx \frac{|\mathbf{T}|}{2} \sum_{i=0}^{q} \omega_i^{(1,0)} \sum_{j=0}^{q} \omega_j^{(0,0)} f\big(\vec{x}(\xi_i^{(1,0)}, \xi_j^{(0,0)})\big) \tag{10}$$

3 Momentos

Dada uma função ϕ definimos como **momento** de ϕ o valor $\mu_\alpha^n(\phi)$, resultado da integral

$$\int_{\hat{\Omega}} B_{\vec{\alpha}}^n(\vec{x}) \hat{\phi}(\vec{x}) \, d\vec{x} \tag{11}$$

em que $\hat{\Omega}$ é o **elemento mestre** [2] (e.g. $[0, 1]^2$) e $B_{\vec{\alpha}}^n$ representa o polinômio de Bernstein correspondente ao domínio $\hat{\Omega}$.

Veja a seguir os algoritmos para calcular os momentos para elementos triangulares e quadrilaterais

Algoritmo 1: MomentoTriangulo(ϕ, n, q)

Entrada: ϕ: Função $\hat{\phi}$ avaliada nos pontos de integração mapeada ao elemento mestre; n: grau dos polinômios de Bernstein; q: número de pontos de integração;

Saída : Vetor μ com os momentos $\mu^n_\alpha(\hat{K})$ do elemento para todo $\alpha \in I^2_n$

1 **for** $i \leftarrow 1$ **to** q **do**
2 $\quad s \leftarrow 1 - \xi_i^{(1,0)}$; $r \leftarrow \xi_i^{(1,0)}/s$;
3 $\quad w \leftarrow \omega_i^{(1,0)} * s^n$ **for** $\alpha_1 \leftarrow 0$ **to** n **do**
4 $\quad\quad$ **for** $j \leftarrow 1$ **to** q **do**
5 $\quad\quad\quad \alpha' = (\alpha_1, j) \in \mathcal{I}^2_k,\ k = max(n, q)$;
6 $\quad\quad\quad \mu'_\alpha \leftarrow \mu'_\alpha + w * \phi[i,\ j]$;
7 $\quad\quad w \leftarrow w * r * \frac{n-\alpha_1}{1+\alpha_1}$;
8 **for** $i \leftarrow 1$ **to** q **do**
9 $\quad s \leftarrow 1 - \xi_i^{(0,0)}$; $r \leftarrow \xi_i(0,0)/s$;
10 $\quad w \leftarrow \omega_i^{(0,0)} * s^{n-\alpha_1}$;
11 $\quad$ **for** $\alpha_1 \leftarrow 0$ **to** n **do**
12 $\quad\quad$ **for** $\alpha_2 \leftarrow 0$ **to** $n - \alpha_1$ **do**
13 $\quad\quad\quad \alpha' = (\alpha_1, i) \in \mathcal{I}^2_k,\ k = max(n, q)$;
14 $\quad\quad\quad \alpha = (\alpha_1, \alpha_2) \in \mathcal{I}^2_n$;
15 $\quad\quad\quad \mu_\alpha \leftarrow \mu_\alpha + w * \mu'_{\alpha'}$;
16 $\quad\quad w \leftarrow w * r * \frac{n-\alpha_1-\alpha_2}{1+\alpha_2}$
17 **return** μ

Observe que o Algoritmo 1 tem número de operações da ordem de $\mathcal{O}(nq^2 + n^2q)$, se tomamos $q \in \mathcal{O}(n)$ temos o gasto esperado de $\mathcal{O}(n^3)$.

O algoritmo para obter-se os momentos no quadrilátero, pode ser obtido trocando-se o parâmetro $n - \alpha_1$ na linha 11 por n, e $\mathcal{I}^2_m$ por I^2_m

4 Matriz de Massa

A matriz de massa é a matriz que possui os elementos resultantes de

$$\int_{\hat{\Omega}} B^n_{\vec{\alpha}}(\xi,\eta) B^n_{\vec{\beta}}(\xi,\eta)\hat{K}(\xi,\eta)\ d\xi d\eta \tag{12}$$

para cada possível valor de $\vec{\alpha}$ e $\vec{\beta}$, desta forma verifica-se que a matriz possui $(n+1)^4$ entradas no caso do quadrilátero e $\binom{n+2}{2}^2$ entradas no caso do triângulo.

Utilizando da propriedade (2) chegamos que a integral (12) pode ser escrita como

$$C \int_{\hat{\Omega}} B^{2n}_{\vec{\alpha}+\vec{\beta}}(\xi,\eta)\hat{K}(\xi,\eta)\ d\xi d\eta \tag{13}$$

em que

$$C = \binom{2n}{n}^{-2} \binom{\alpha_1+\beta_1}{\alpha_1} \binom{2n-\alpha_1-\beta_1}{n-\alpha_1} \binom{\alpha_2+\beta_2}{\alpha_2} \binom{2n-\alpha_2-\beta_2}{n-\alpha_2}$$

no caso do quadrilátero e

$$C = \frac{\binom{\alpha_1+\beta_1}{\alpha_1}\binom{2n-\alpha_1-\beta_1}{n-\alpha_1}\binom{\alpha_2+\beta_2}{\alpha_2}\binom{2n-\alpha_1-\alpha_2-\beta_1-\beta_2}{n-\alpha_1-\alpha_2}}{\binom{2n}{n}\binom{2n-\alpha_1-\beta_1}{n-\alpha_1}}$$

no caso do triângulo.

Desta forma podemos calcular os elementos da matriz de massa utilizando os momentos $\mu^{2n}_{\vec{\alpha}+\vec{\beta}}(K)$. Veja a seguir os algoritmos construídos para o caso do quadrilátero e do triângulo.

Algoritmo 2: MassaQuadrilatero(K, $2n$, q)

Entrada: K: Função $\hat{K}$ avaliada nos pontos de integração mapeada ao elemento mestre; n: grau dos polinômios de Bernstein; q: número de pontos de integração;

Saída : Matriz de massa A do elemento

$\mu \leftarrow$ MomentoQuadrilatero(K,n,q);
$C \leftarrow 1/\binom{2n}{n}^2$;
for $\alpha_1 \leftarrow 0$ **to** n **do**
    for $\beta_1 \leftarrow 0$ **to** n **do**
        $w_1 \leftarrow C * \binom{\alpha_1+\beta_1}{\alpha_1}\binom{2n-\alpha_1-\beta_1}{n-\alpha_1}$;
        for $\alpha_2 \leftarrow 0$ **to** n **do**
            for $\beta_2 \leftarrow 0$ **to** n **do**
                $w_2 \leftarrow w_1 * \binom{\alpha_2+\beta_2}{\alpha_2}\binom{2n-\alpha_2-\beta_2}{n-\alpha_1}$;
                $\alpha = (\alpha_1, \alpha_2)$, $\beta = (\beta_1, \beta_2) \in I_n^2$;
                $A_{\alpha,\beta} \leftarrow w_2 * \mu_{\alpha+\beta}$;
return A

Utilizando a matriz de Pascal, podemos pre-computar os valores dos coeficientes binomiais utilizados por este algoritmo e por todos os algoritmos deste estudo, fazendo com que a obtenção deste tenha gasto $\mathcal{O}(1)$. Desta forma o algoritmo tem complexidade de tempo $\mathcal{O}(n^4)$, já que o algoritmo **MomentoQuadrilatero** tem gasto $\mathcal{O}(n^3)$ (quando o número de pontos de integração numérica é da ordem de n em cada variável) e os laços executam tarefas com gasto total $\mathcal{O}(n^4)$.

Além disso o algoritmo acima pode ser modificado para se utilizar valores diferentes para o grau do polinômio em cada uma das variáveis, se tivéssemos, por exemplo,

$B^{(n,m)}_{\vec{\alpha}}(\xi,\eta)$ bastaria apenas trocar os limites dos laços de α_2 e β_2 para m.

Algoritmo 3: MassaTriangulo(K, $2n$, q)

Entrada: K: Função $\hat{K}$ avaliada nos pontos de integração mapeada ao elemento mestre; n: grau dos polinômios de Bernstein; q: número de pontos de integração;

Saída : Matriz de massa A do elemento

$\mu \leftarrow$ MomentoTriangulo(K,n);
$C \leftarrow 1/\binom{2n}{n}$;
for $\alpha_1 \leftarrow 0$ **to** n **do**
    for $\beta_1 \leftarrow 0$ **to** n **do**
        $w_1 \leftarrow C * \binom{\alpha_1+\beta_1}{\alpha_1}$;
        for $\alpha_2 \leftarrow 0$ **to** $n-\alpha_1$ **do**
            for $\beta_2 \leftarrow 0$ **to** $n-\beta_1$ **do**
                $w_2 \leftarrow w_1 * \binom{\alpha_2+\beta_2}{\alpha_2}\binom{2n-\alpha_1-\alpha_2-\beta_1-\beta_2}{n-\alpha_1}$;
                $\alpha = (\alpha_1, \alpha 2, n-\alpha_1-\alpha 2)$, $\beta = (\beta_1, \beta_2, n-\beta_1-\beta_2) \in \mathcal{I}^2_n$;
                $A_{\alpha,\beta} \leftarrow w_2 * \mu_{\alpha+\beta}$;
return A

Assim como o algoritmo para o quadrilátero, este algoritmo tem complexidade de tempo $\mathcal{O}(n^4)$.

5 Matriz de Rigidez

A matriz de rigidez é composta pelos elementos da seguinte integral

$$\int_{\Omega} \vec{\nabla} B^n_{\vec{\alpha}}(x,y) \vec{\nabla} B^n_{\vec{\beta}}(x,y) W(x,y)\ dxdy \tag{14}$$

Esta integral pode ser calculada com o mapeamento ao **elemento mestre** pois se tem a seguinte relação:

$$\vec{\nabla} B^n_{\vec{\alpha}}(x,y) \vec{\nabla} B^n_{\vec{\beta}}(x,y) = \left(J^{-T} \hat{\vec{\nabla}} B^n_{\vec{\alpha}}(\xi,\eta) \cdot J^{-T} \hat{\vec{\nabla}} B^n_{\vec{\beta}}(\xi,\eta)\right) \tag{15}$$

em que J é a matriz Jacobiana da transformação que mapeia $(\xi,\eta) \in \hat{\Omega}$ para $(x,y) \in \Omega$.

Desta forma podemos calcular a matriz de rigidez substituindo (15) em (14) e decompondo em três integrais:

$$\int_{\hat{\Omega}} \frac{\partial}{\partial \xi} B^n_{\vec{\alpha}}(\xi,\eta) \frac{\partial}{\partial \eta} B^n_{\vec{\beta}}(\xi,\eta) \hat{W}_1(\xi,\eta)\ d\xi d\eta \tag{16}$$

$$\int_{\hat{\Omega}} \frac{\partial}{\partial \xi} B^n_{\vec{\alpha}}(\xi,\eta) \frac{\partial}{\partial \xi} B^n_{\vec{\beta}}(\xi,\eta) \hat{W}_2(\xi,\eta)\ d\xi d\eta \tag{17}$$

$$\int_{\hat{\Omega}} \frac{\partial}{\partial \eta} B^n_{\vec{\alpha}}(\xi,\eta) \frac{\partial}{\partial \eta} B^n_{\vec{\beta}}(\xi,\eta) \hat{W}_3(\xi,\eta)\ d\xi d\eta \tag{18}$$

Assim como no caso anterior, da matriz de massa, obtemos resultados dos integrandos em função da base de Bernstein a fim de se obter um algoritmo com complexidade ótima.

5.1 Quadriláteros

Para a equação (16) ficamos com

$$\frac{\partial}{\partial\xi}B^n_{\vec{\alpha}}(\xi,\eta)\frac{\partial}{\partial\eta}B^n_{\vec{\beta}}(\xi,\eta)=\frac{n^2}{\binom{2n-1}{n-1}^2}\left(C_1B^{2n-1}_{\alpha+\beta-e_1-e_2}+C_2B^{2n-1}_{\vec{\alpha}+\vec{\beta}}-C_3B^{2n-1}_{\vec{\alpha}+\vec{\beta}-e_1}-C_4B^{2n-1}_{\vec{\alpha}+\vec{\beta}-e_2}\right) \tag{19}$$

Em que $e_1=(1,0)$, $e_2=(0,1)$, e

$$C_1=\binom{\alpha_1+\beta_1-1}{\alpha_1-1}\binom{\alpha_2+\beta_2-1}{\alpha_2-1}\binom{2n-\alpha_1-\beta_1+1}{n-\alpha_1+1}\binom{2n-\alpha_2-\beta_2+1}{n-\alpha_2+1};$$

$$C_2=\binom{\alpha_1+\beta_1}{\alpha_1}\binom{\alpha_2+\beta_2}{\alpha_2}\binom{2n-\alpha_1-\beta_1}{n-\alpha_1}\binom{2n-\alpha_2-\beta_2}{n-\alpha_2};$$

$$C_3=\binom{\alpha_1+\beta_1-1}{\alpha_1-1}\binom{\alpha_2+\beta_2}{\alpha_2}\binom{2n-\alpha_1-\beta_1+1}{n-\alpha_1+1}\binom{2n-\alpha_2-\beta_2}{n-\alpha_2};$$

$$C_4=\binom{\alpha_1+\beta_1}{\alpha_1}\binom{\alpha_2+\beta_2-1}{\beta_2-1}\binom{2n-\alpha_1-\beta_1}{n-\alpha_1}\binom{2n-\alpha_2-\beta_2+1}{n-\beta_2+1}.$$

Para a equação (17) ficamos com

$$\frac{\partial}{\partial\xi}B^n_{\vec{\alpha}}(\xi,\eta)\frac{\partial}{\partial\xi}B^n_{\vec{\beta}}(\xi,\eta)=b^{2n}_{\alpha_2+\beta_2}(\eta)C_1\left(C_2b^{2n-2}_{\alpha_1+\beta_1-2}(\xi)+C_3b^{2n-2}_{\alpha_1+\beta_1}(\xi)-C_4b^{2n-2}_{\alpha_1+\beta_1-1}\right) \tag{20}$$

em que, neste caso

$$C_1=n^2\frac{\binom{\alpha_2+\beta_2}{\alpha_2}}{\binom{2n}{n}\binom{2n-2}{n-1}};$$

$$C_2=\binom{\alpha_1+\beta_1-2}{\alpha_1-1}\binom{2n-\alpha_1-\beta_1+2}{n-\alpha_1+1};$$

$$C_3=\binom{\alpha_1+\beta_1}{\alpha_1}\binom{2n-\alpha_1-\beta_1}{n-\alpha_1};$$

$$C_4=\binom{\alpha_1+\beta_1-1}{\alpha_1-1}\binom{2n-\alpha_1-\beta_1+1}{n-\alpha_1+1}+\binom{\alpha_1+\beta_1-1}{\beta_1-1}\binom{2n-\alpha_1-\beta_1+1}{n-\beta_1+1}.$$

O caso da equação (18) é similar ao acima, invertendo-se os índices de α e β. A seguir os algoritmos para cada um desses casos.

Algoritmo 4: XiEtaQuadrilatero(W, n, q)

Entrada: W: Função $\hat{W}$ avaliada nos pontos de integração mapeada ao elemento mestre; n: graus dos polinômios de Bernstein; q: número de pontos de integração;

Saída : Matriz com os valores de (16) $\forall \vec{\alpha}, \vec{\beta} \in I_n^2$

```
μ ← MomentoQuadrilatero(K,2n-1,q);
C ← n²/(2n−1 choose n−1)²;
for α₁ ← 0 to n − 1 do
    for β₁ ← 0 to n do
        w₁ ← C * (α₁+β₁ choose α₁) * (2n−α₁−β₁−1 choose n−α₁−1);
        for α₂ ← 0 to n do
            for β₂ ← 0 to n − 1 do
                α = (α₁, α₂); β = (β₁, β₂) ∈ I_n²;
                w₂ ← w₁ * (α₂+β₂ choose α₂)(2n−α₂−β₂−1 choose n−α₂−1);
                w₃ ← w₂ * μ_{α+β};
                A_{α,β} ← A_{α,β} + w₃;
                A_{α+e₁,β+e₂} ← A_{α+e₁,β+e₂} + w₃;
                A_{α+e₁,β} ← A_{α+e₁,β} − w₃;
                A_{α,β+e₂} ← A_{α,β+e₂} − w₃;
return A
```

Algoritmo 5: XiXiQuadrilatero(W, n, q)

Entrada: W: Função $\hat{W}$ avaliada nos pontos de integração mapeada ao elemento mestre; n: graus dos polinômios de Bernstein; q: número de pontos de integração;

Saída : Matriz com os valores de (17) $\forall \vec{\alpha}, \vec{\beta} \in I_n^2$

```
μ ← MomentoQuadrilatero(K,{2n₁-2,2n₂},q);
C ← n²/(2n choose n)(2n−2 choose n−1);
for α₁ ← 0 to n − 1 do
    for β₁ ← 0 to n − 1 do
        w₁ ← C * (α₁+β₁ choose α₁) * (2n−α₁−β₁−2 choose n−α₁−1);
        for α₂ ← 0 to n do
            for β₂ ← 0 to n do
                α = (α₁, α₂); β = (β₁, β₂) ∈ I_n²;
                w₂ ← w₁ * (α₂+β₂ choose α₂)(2n−α₂−β₂ choose n−α₂);
                w₃ ← w₂ * μ_{α+β};
                A_{α,β} ← A_{α,β} + w₃;
                A_{α+e₁,β+e₁} ← A_{α+e₁,β+e₁} + w₃;
                A_{α+e₁,β} ← A_{α+e₁,β} − w₃;
                A_{α,β+e₁} ← A_{α,β+e₁} − w₃;
return A
```

Algoritmo 6: EtaEtaQuadrilatero(W, n, q)

Entrada: W: Função $\hat{W}$ avaliada nos pontos de integração mapeada ao elemento mestre; n: graus dos polinômios de Bernstein; q: número de pontos de integração;

Saída : Matriz com os valores de (17) $\forall\vec{\alpha},\vec{\beta} \in I_n^2$

$\mu \leftarrow$ MomentoQuadrilatero(K,{2n$_1$,2n$_2$-2},q);

$C \leftarrow n^2/\binom{2n}{n}\binom{2n-2}{n-1}$;

for $\alpha_1 \leftarrow 0$ **to** n **do**

  for $\beta_1 \leftarrow 0$ **to** n **do**

    $w_1 \leftarrow C * \binom{\alpha_1+\beta_1}{\alpha_1} * \binom{2n-\alpha_1-\beta_1}{n-\alpha_1}$;

    for $\alpha_2 \leftarrow 0$ **to** $n-1$ **do**

      for $\beta_2 \leftarrow 0$ **to** $n-1$ **do**

        $\alpha = (\alpha_1, \alpha_2)$; $\beta = (\beta_1, \beta_2) \in I_n^2$;

        $w_2 \leftarrow w_1 * \binom{\alpha_2+\beta_2}{\alpha_2}\binom{2n-\alpha_2-\beta_2-2}{n-\alpha_2-1}$;

        $w_3 \leftarrow w_2 * \mu_{\alpha+\beta}$;

        $A_{\alpha,\beta} \leftarrow A_{\alpha,\beta} + w_3$;

        $A_{\alpha+e_2,\beta+e_2} \leftarrow A_{\alpha+e_2,\beta+e_2} + w_3$;

        $A_{\alpha+e_2,\beta} \leftarrow A_{\alpha+e_2,\beta} - w_3$;

        $A_{\alpha,\beta+e_2} \leftarrow A_{\alpha,\beta+e_2} - w_3$;

return A

5.2 Triângulos

Para o caso dos elementos triangulares, pode-se utilizar as seguintes relações [3] para fazer o cálculo da Matriz de Rigidez

$$\vec{\nabla}\lambda_i = \frac{|\vec{\gamma}_i|}{2|T|}\vec{n} \tag{21}$$

$$\vec{\nabla}\mathbf{B}^n_{\vec{\alpha}}(\vec{x}) = \binom{n}{\vec{\alpha}}\alpha_1\mathbf{B}^{n-1}_{\vec{\alpha}-e_1}(\vec{x})\vec{\nabla}\lambda_1 + \alpha_2\mathbf{B}^{n-1}_{\vec{\alpha}-e_2}(\vec{x})\vec{\nabla}\lambda_2 + \alpha_3\mathbf{B}^{n-1}_{\vec{\alpha}-e_3}(\vec{x})\vec{\nabla}\lambda_3 \tag{22}$$

Em que $|\vec{\gamma}_i|$ é o comprimento do lado $\vec{\gamma}_i$ oposto ao vértice $\vec{v}_i$ do triângulo, $\vec{n}$ é o vetor normal à γ_i, e $e_i \in \{\alpha \in \mathcal{I}_1^2;\ \alpha_i = 1\}$. Utilizando as equações acima obtêm-se que

$$\vec{\nabla}\mathbf{B}^n_{\vec{\alpha}}(\vec{x}) \cdot \vec{\nabla}\mathbf{B}^n_{\vec{\beta}}(\vec{x}) = \frac{n^2}{\binom{2n-2}{n-2}}\sum_{k=1}^{3}\sum_{l=1}^{3}\vec{\nabla}\lambda_k \cdot \vec{\nabla}\lambda_l C_{k,l}\mathbf{B}^{2n-2}_{\alpha-e_k\vec{+}\beta-e_l}(\vec{x}) \tag{23}$$

Em que

$$C_{k,l} = \binom{\vec{\alpha}-e_k+\vec{\beta}-e_l}{\vec{\alpha}-e_k}$$

e se $\vec{\alpha},\vec{\beta} \in \mathcal{I}_n^d$, então

$$\binom{n}{\vec{\alpha}} = \prod_{i=1}^{d}\binom{n}{\alpha_i},\ \text{e}\ \binom{\vec{\alpha}}{\vec{\beta}} = \prod_{i=1}^{d}\binom{\alpha_i}{\beta_i}$$

Assim podemos calcular a matriz de rigidez com o algoritmo 8.

Utilizaremos o seguinte algoritmo como rotina auxiliar para calcular a matriz de rigidez

Algoritmo 7: ProdutosNormal($\mathbf{T}$)

Entrada: $\mathbf{T}$: Triângulo do elemento com vértices v_1, v_2, v_3
Saída : Matriz com os valores de $\vec{\nabla}\lambda_k \cdot \vec{\nabla}\lambda_l$, $1 \leq k, l \leq 3$

`// ` $\vec{n}$ `armazenará os vetores normais`

$\vec{n}_{1,x} \leftarrow v_{2,y} - v_{3,y}$;
$\vec{n}_{1,y} \leftarrow v_{3,x} - v_{2,x}$;
$\vec{n}_{2,x} \leftarrow v_{3,y} - v_{1,y}$;
$\vec{n}_{2,y} \leftarrow v_{1,x} - v_{3,x}$;
$\vec{n}_{3,x} \leftarrow v_{1,y} - v_{2,y}$;
$\vec{n}_{3,y} \leftarrow v_{2,x} - v_{1,x}$;
foreach $1 \leq k, l \leq 3$ **do**
    $N_{k,l} \leftarrow \vec{n}_k \cdot \vec{n}_l$;
return N

Algoritmo 8: RigidezTriangulo(W, n, q)

Entrada: W: Função $\hat{W}$ avaliada nos pontos de integração mapeada ao elemento mestre; n: graus dos polinômios de Bernstein; q: número de pontos de integração;
Saída : Matriz com os valores de (14) $\forall \vec{\alpha}, \vec{\beta} \in \mathcal{I}_n^2$

$\mu \leftarrow$ MomentoTriangulo(K,2n-2,q);
$N \leftarrow ProdutosNormal(\mathbf{T})$; `// ` $\mathbf{T}$ `é o triângulo do elemento para o qual a matriz está sendo calculada`
$C \leftarrow n^2 / \binom{2n-1}{n-1}^2$;
for $\alpha_1 \leftarrow 0$ **to** n **do**
    for $\beta_1 \leftarrow 0$ **to** n **do**
        $w_1 \leftarrow C * \binom{\alpha_1+\beta_1}{\alpha_1} * \binom{2n-\alpha_1-\beta_1-1}{n-\alpha_1-1}$;
        for $\alpha_2 \leftarrow 0$ **to** $n - \alpha_1$ **do**
            for $\beta_2 \leftarrow 0$ **to** $n - \beta_1$ **do**
                $\alpha = (\alpha_1, \alpha_2, n - \alpha_1 - \alpha_2)$; $\beta = (\beta_1, \beta_2, n - \beta_1 - \beta_2) \in I_n^2$;
                $w_2 \leftarrow w_1 * \binom{\alpha_2+\beta_2}{\alpha_2}\binom{2n-\alpha_2-\beta_2-1}{n-\alpha_2-1} * \mu_{\alpha+\beta}$;
                $A_{\alpha,\beta} \leftarrow A_{\alpha,\beta} + w_2 * N_{3,3}$;
                $A_{\alpha+e_1,\beta} \leftarrow A_{\alpha+e_1,\beta} + w_2 * N_{1,3}$;
                $A_{\alpha+e_1,\beta+e_1} \leftarrow A_{\alpha+e_1,\beta+e_1} + w_2 * N_{1,1}$;
                $A_{\alpha,\beta+e_1} \leftarrow A_{\alpha,\beta+e_1} + w_2 * N_{3,1}$;
                $A_{\alpha+e_2,\beta} \leftarrow A_{\alpha+e_2,\beta} + w_2 * N_{2,3}$;
                $A_{\alpha+e_2,\beta+e_2} \leftarrow A_{\alpha+e_2,\beta+e_2} + w_2 * N_{2,2}$;
                $A_{\alpha,\beta+e_2} \leftarrow A_{\alpha,\beta+e_2} + w_2 * N_{3,2}$;
                $A_{\alpha+e_1,\beta+e_2} \leftarrow A_{\alpha+e_1,\beta+e_2} + w_2 * N_{1,2}$;
                $A_{\alpha+e_2,\beta+e_1} \leftarrow A_{\alpha+e_2,\beta+e_1} + w_2 * N_{2,1}$;
return A

6 Resultados

Os algoritmos acima apresentados foram implementados em linguagem C++ com orientação a objetos (repositório no GitHub [6]) e utilizados para computar o tempo de execução a fim de comparar com os métodos implementados na biblioteca NeoPZ [7].

Foram realizadas comparações apenas sobre o tempo de execução para a construção da **Matriz de Rigidez**. Veja nas figuras 1 e 2 os gráficos de tempo de execução em função da ordem polinomial n utilizada. Veja também as tabelas 1 e 2 com a razão entre o tempo de execução para cada caso. O número de pontos de integração utilizado em ambos os casos é de $n + 1$.

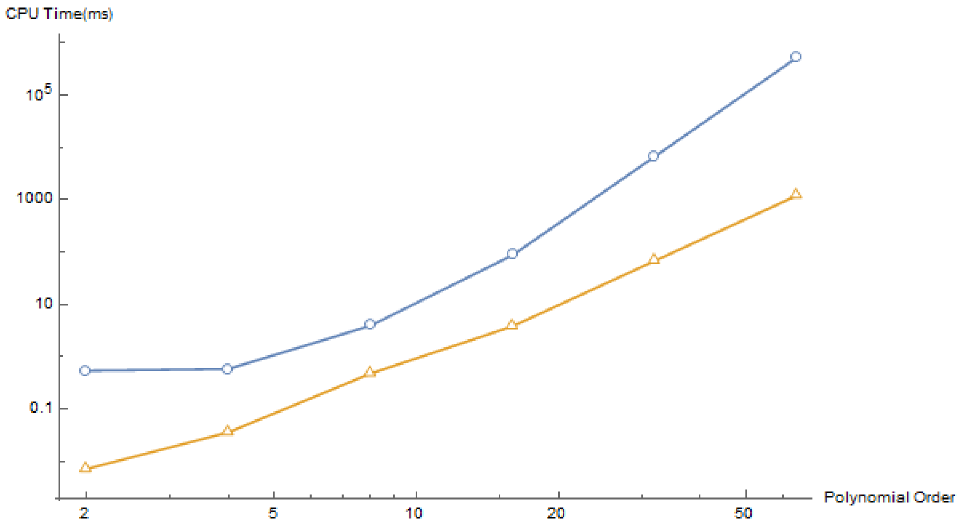

Figura 1: Gráfico comparando o tempo de execução (CPU Time) em função da ordem polinomial n utilizada nos cálculos para os elementos **Quadriláteros** no ambiente NeoPZ e a implementação dos algoritmos apresentados (Bernstein).

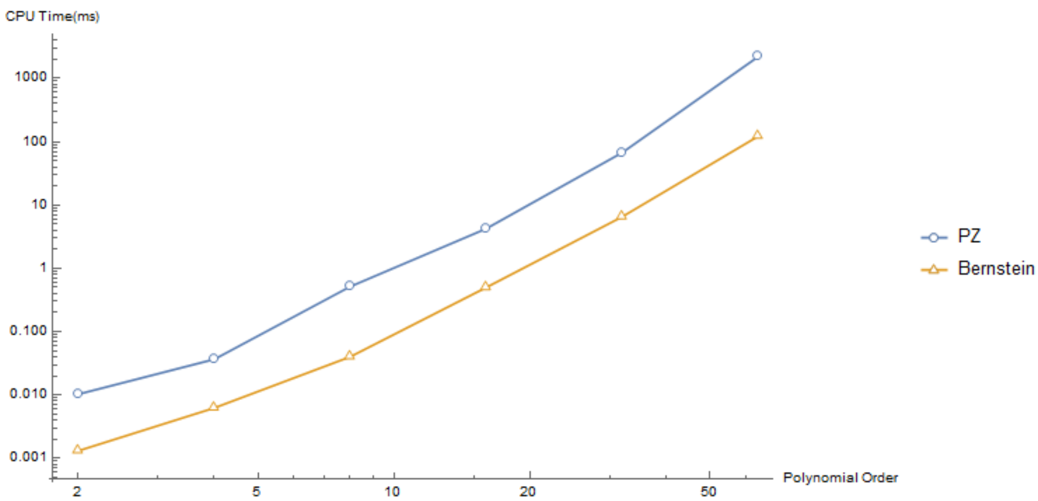

Figura 2: Gráfico comparando o tempo de execução (CPU Time) em função da ordem polinomial n utilizada nos cálculos para os elementos **Triangulares** no ambiente NeoPZ e a implementação dos algoritmos apresentados (Bernstein).

Tabela 1: Tabela com a razão entre os tempos de execução para o cálculo da Matriz de Rigidez obtidos utilizando o NeoPZ e utilizando polinômios de Bernstein para elementos Quadriláteros.

n	2	4	8	16	32	64
$\frac{t_{PZ}}{t_B}$	75.4	16.2	8.16	22.8	97.5	428

Tabela 2: Tabela com a razão entre os tempos de execução para o cálculo da Matriz de Rigidez obtidos utilizando o NeoPZ e utilizando polinômios de Bernstein para elementos Triangulares.

n	2	4	8	16	32	64
$\frac{t_{PZ}}{t_B}$	7.84	5.86	12.8	8.50	10.3	18.0

O código foi compilado utilizando o MSVC. Os testes foram realizados em máquina com processador Intel i7-7700HQ e 16GB de memória, com sistema operacional Windows 10 64 bits. Resultados também foram testados no Linux Ubuntu 18.04, sem mudanças significativas.

Referências

[1] G. Hammerlin; K-H. Hoffman, *Numerical Mathematics. Springer, 1991, XII, 425 p.*

[2] J. T. Oden; G. F. Carey, *Finite Elements: Mathematical Aspects, Prentice Hall INC., 1997*

[3] M. Ainsworth; G. Andriamaro; O. Davydov, Berntein–Bézier Finite Element of Arbitrary Order and Optimal Assembly Procedures, *SIAM J. SCI.COMPUT: Vol. 33, No. 6, pp. 3087–3109*

[4] P. Pulino; M. R. Fernandes, Resolução de Equações Diferenciais via Método de Elementos Finitos, *2002*

[5] L. B. Andrade; S. M. Gomes; P. R. Devloo, Polinômios de Bernstein para o Método de Galerkin, *2017*

[6] Repositório no GitHub contendo os códigos utilizados nos testes deste trabalho `https://github.com/AndradeL/BernsteinFEM`

[7] Página sobre o ambiente NeoPZ `http://www.labmec.org.br/pz/arquivos-html/html/index.html`