IMPLEMENTAÇÃO DE UM GERADOR AUTOMÁTICO DE MALHAS PARA O MÉTODO DOS ELEMENTOS FINITOS

IMPLEMENTATION OF AN AUTOMATIC MESH GENERATOR FOR FINITE ELEMENT METHOD

Priscila Ligabó Murarolli*

Resumo

O objetivo deste trabalho é apresentar a implementação de um gerador automático de malhas para o método dos elementos finitos com o intuito de acelerar o processo de criação de um produto, onde através de dados confiáveis de sua qualidade pode-se verificar propriedades inesperadas em seu protótipo, determinando o comportamento térmico ou mecânico do produto em serviço, permitindo que os mapeadores gerem malhas com um modelo computacional. No mapeamento transfinito, que será utilizado, tanto as linhas que descrevem o contorno de uma superfície ou as superfícies de contorno de um volume são completamente consideradas para o gerador de malhas. O Software desenvolvido receberá informações do arquivo de origem do Software de Elementos Finitos e então gerará a malha das formas geométricas, sendo assim avaliados os resultados, verificando a qualidade da malha, quando um quadrilátero aproxima de um quadrado e quando hexaedro aproxima de um cubo e as suas características.

Palavras-Chave: Gerador de Malha. Elementos Finitos. Mapeamento Transfinito

Abstract

The objective of this paper is to present the implementation of an automatic mesh generator for finite element method in order to accelerate the process of creating a product where by reliable data quality can be observed unexpected properties in the prototype , determining the thermal or mechanical product in service, allowing mappers generate meshes with a computational model. In the transfinite mapping that will be used on the lines that describe the outline of a surface or contour surfaces of a volume are fully considered for the mesh generator. The developed software will receive information from the source finite element software and then generate the mesh of geometric shapes, and thus evaluated the results, checking the quality of the mesh, when approaching a square a square and hexahedron when approaching a cube and their characteristics.

Keywords: Mesh Generator. Finite Element Method. Transfinite mapping.

* Professora da Faculdade de Tecnologia, Ciências e Educação (FATECE – Pirassununga/SP). Coordenadora do Curso de Ciência da Computação. priscila@fatece.edu.br

Introdução

O sucesso econômico de empresas de manufatura é caracterizado pelo contínuo decréscimo do tempo para o desenvolvimento de um produto, associado ao acréscimo de sua qualidade. Parte considerável desse desenvolvimento está na criação do produto, que deve satisfazer exigências contraditórias.

A criação de um produto dá-se através de um processo interativo, pelo qual a forma (geometria) e a disposição mútua dos componentes (topologia) de um produto é aprimorada até que todos os requisitos, tais como funcionalidade, vida útil, segurança, custo, adequação ao meio ambiente, etc., sejam preenchidas. Para acelerar-se o processo de criação de um produto, necessita-se de dados confiáveis relativos à sua qualidade, muitas vezes em uma fase prematura, antes mesmo da existência de um protótipo. Como alternativa calcula-se as propriedades desse produto baseando-se em métodos numéricos.

Os cálculos referem-se basicamente à determinação do comportamento térmico ou mecânico do produto em serviço. Eles são realizados utilizando-se, entre outros, o Método dos Elementos Finitos (MEF). Assim pode-se determinar, por exemplo, a distribuição de temperaturas ou tensões resultantes sob a ação de cargas estáticas e/ou dinâmicas. Dessa forma pode-se analisar o comportamento de um componente, de um conjunto ou de uma máquina completa.

Uma das tarefas de toda análise pelo MEF é a geração de malha, ou seja, a descrição do objeto a ser analisado através de elementos finitos, do ponto de vista matemático significa subdividir o domínio de validade das formas que descrevem o comportamento do componente em subdivisões representado pelo elemento da malha de EF. No entanto antes de empregar-se os geradores automáticos de malhas, os usuários precisam definir o objeto a ser calculado como um modelo computacional. Nesse processo é necessário levar-se em conta uma série de restrições geométricas e topológicas.

Os mapeadores geram malhas com variação regular e pequena distorção dos elementos, que apresentam um comportamento muito bom em relação à convergência (WALZ et al., 1969), mas em contrapartida a descrição da geometria precisa ser bem mais detalhada. Devido a qualidade das malhas que cada vez mais são utilizados os mapeadores.

No mapeamento a discretização ocorre em dois passos. A geração de elementos é sempre feita em uma região unidimensional, bidimensional ou tridimensional unitária. Essa malha gerada na região unitária é, em um passo seguinte, projetada para a geometria, utilizando-se funções de mapeamento. O principal ponto desses métodos está na determinação e utilização das funções de mapeamento e não na geração da malha na região unitária.

Assim podemos verificar que o Mapeamento Transfinito projeta a malha gerada na região unitária para a geometria real do componente.

No mapeamento transfinito tanto as linhas que descrevem o contorno de uma superfície ou as superfícies de contorno de um volume são completamente consideradas (GORDON et al., 1973; HABER et al., 1981). Dessa forma, linhas de formas livres, inclusive com cantos vivos, podem ser utilizadas para descreverem uma superfície (SHEPHARD et al., 1981).

No mapeamento transfinito a descrição de um quadrilátero é realizada através de interpolação bilinear entre as linhas opostas do contorno. Além dos contornos podem ser utilizadas linhas internas (WU et al., 1979), que possibilitam a descrição de superfícies com maior grau de complexidade. A interpolação transfinita de corpos tridimensionais segue o mesmo princípio. No caso de um hexaedro realiza-se uma interpolação trilinear entre suas superfícies de contorno.

1 Superfícies e Volumes

O conceito de curva demonstra a idéia de um objeto, uma linha contínua no espaço. Segundo Gleicher (2004, p. 1, tradução nossa), uma curva paramétrica é um mapeamento contínuo de um espaço unidimensional para um espaço n-dimensional.

Em meados do século XX, desenhar curvas brandas para modelar uma geometria específica era considerado um trabalho difícil e fastidioso. Em 1959, Paul de Fagnet de Casteljau um funcionário da Citroën®, iniciou a tarefa para a obtenção de curvas e superfícies brandas combinando e tornando novas as malhas de polígonos para modelagem de chassis. Simultaneamente, na Renault®, Pierre Bézier também produziria curvas para o mesmo fim, que atualmente são conhecidas como curvas de Bézier. Conforme Bézier (1970) a curva ficou formalmente conhecida com a publicação de seu trabalho.

O tipo de curva que deve ser escolhida dependerá do objeto que se deseja definir, isto é, à sua aplicação. Basicamente, traçar uma curva que passe por um

conjunto de pontos caracteriza um problema de interpolação, ao passo que definir uma curva que se aproxime de um determinado conjunto de pontos é um problema de aproximação.

Para compreendermos melhor é necessário entender o conceito de interpolação que é um método que permite construir um novo conjunto de dados a partir de um conjunto discreto de dados pontuais previamente conhecidos; assim também torna-se importante conhecer um pouco sobre a aproximação que é uma representação inexata de algo que apesar de tudo ainda é suficientemente próxima para ser utilizada.

Os polinômios foram uma das primeiras escolhas para a solução de ambos os problemas acima citados, mas sua desvantagem está na sua globalidade, ou seja, quando alteramos o valor de qualquer ponto, toda a curva é afetada. Isso implica no número reduzido de aplicações que um polinômio pode ter em modelagens de curvas, restrito a pequenos intervalos ou a um número baixo de pontos de controle.

1.1 Curvas de Bézier

A fórmula para curvas desenvolvidas por Pieri Bezier, que posteriormente recebeu o seu nome ganhou grande importância quando estudada por Paul de Casteljau no fim da década de 1950, com o objetivo de criar um método eficaz para a modelagem de carros. Baseado nos princípios de Hermite, durante seus trabalhos em projetos de automóveis para a empresa francesa Renault, Bézier acrescentou dois pontos de controle às constantes de Hermite determinando os dois vetores tangentes nos pontos inicial e final.

Naquela época, Bézier trabalhava como engenheiro e concebeu uma nova formulação de curvas para representar carros em computador. De fato, a curva de Bézier é uma forma aproximada de curva polinomial, especialmente concebida para ser controlada de forma conveniente. Na sua forma bidimensional, a curva de Bézier é hoje base para vários programas gráficos como o Adobe Illustrator e Corel Draw. Algumas fontes também como TrueType e Post Script Type também são armazenadas como curvas de Bézier.

As formas mais importantes das curvas de Bézier são as quadráticas e cúbicas. Curvas de grau maior implicam em um custo mais elevado de cálculo e processamento. O método atualmente mais utilizado para avaliar as curvas de Bézier é o algoritmo de De Casteljau (FARIN, 1990) que pode-se definir da seguinte maneira

- Suponha que queiramos aproximar uma curva polinomial entre dois pontos p_0 e p_1 dados
- A solução natural é um segmento de reta que passa por $p0$ e $p1$ cuja parametrização mais comum é $p(u) = (1-u)p_0 + up_1$
- Podemos pensar em $p(u)$ como uma média ponderada entre e p_1
- Observe que os polinômios $(1-u)$ e u somam 1 para qualquer valor de u
    - São chamadas de funções de mistura (*blending functions*)

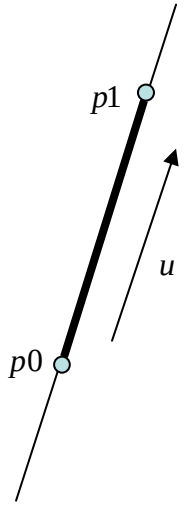


Figura 1. Curva Polinomial

- Para generalizar a idéia para três pontos p_0, p_1 e p_2 consideramos primeiramente os segmentos de reta $p_0 - p_1$ e $p_1 p_2$

$$p_{01}(u) = (1-u)p_0 + up_1$$
$$p_{11}(u) = (1-u)p_1 + up_2 \qquad (1)$$

- Podemos agora realizar uma interpolação entre $p_{01}(u)$ e $p_{12}(u)$

$$p_{02}(u) = (1-u)p_{01}(u) + up_{11}(u) = (1-u)^2 p_0 + 2u(1-u)p_1 + u^2 p_2 \qquad (2)$$

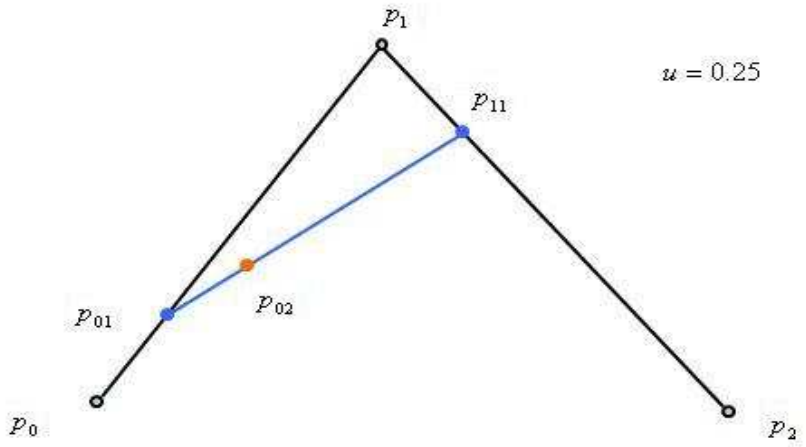


Figura 2. Algoritmo de De Casteljau $u = 0.25$

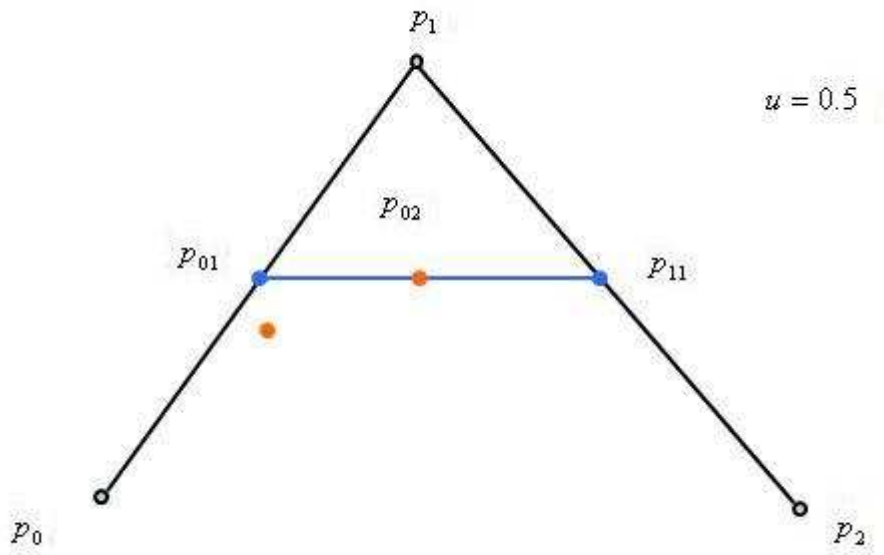


Figura 3. Algoritmo de De Casteljau $u = 0.5$

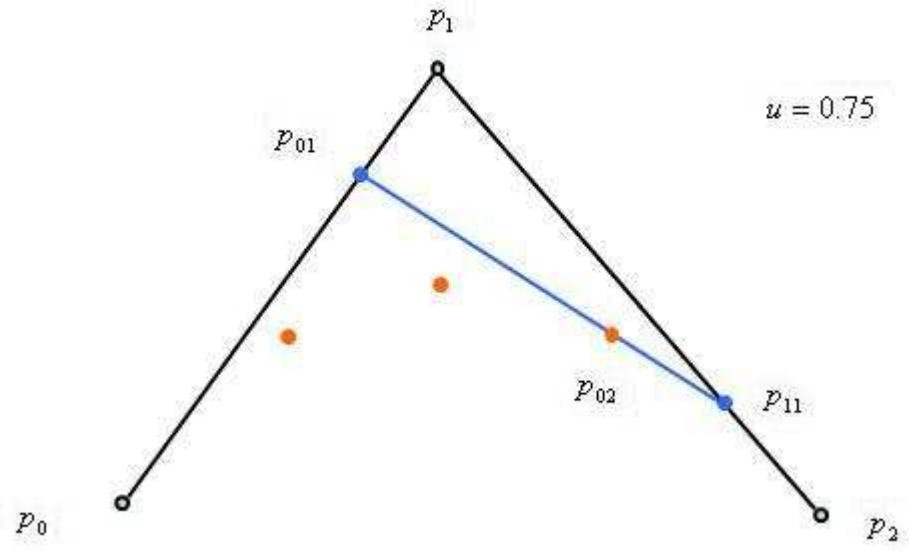


Figura 4. Algoritmo de De Casteljau $u = 0.75$

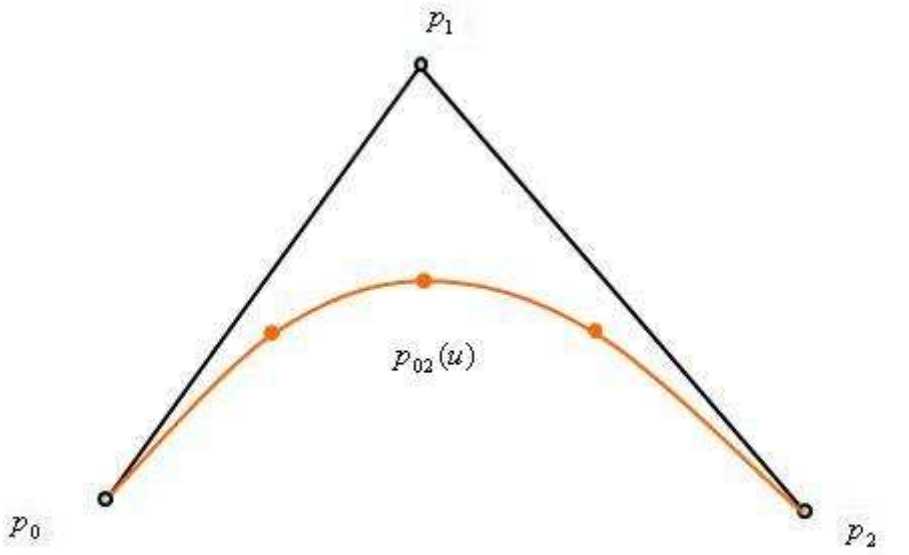


Figura 5. Algoritmo de De Casteljau

- A curva obtida pode ser entendida como a "mistura" dos pontos p_0, p_1 e p_2 por intermédio de três funções quadráticas:

$$b_{02}(u) = (1-u)^2$$
$$b_{12}(u) = 2u(1-u)$$
$$b_{22}(u) = u^2 \qquad (3)$$

- Aplicando mais uma vez a idéia podemos definir uma cúbica por 4 pontos:

$$p_{02}(u) = (1-u)^2 p_0 + 2u(1-u)p_1 + u^2 p_2$$
$$p_{12}(u) = (1-u)^2 p_1 + 2u(1-u)p_2 + u^2 p_3$$
$$p_{03}(u) = (1-u)p_{02}(u) + up_{12}(u) = (1-u)^3 p_0 + 3u(1-u)p_2 + u^3 p_3 \qquad (4)$$

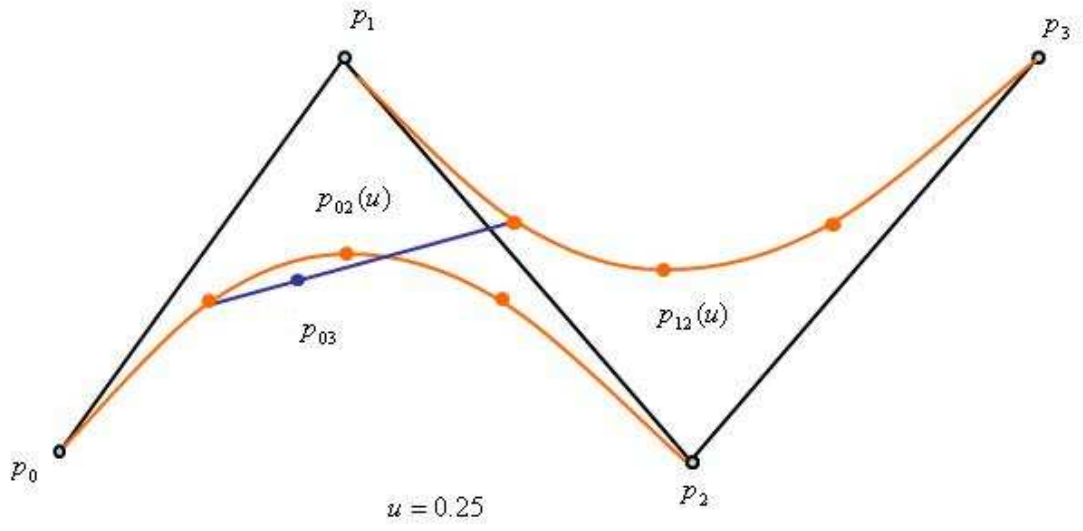


Figura 6. Reaplicação do Algoritmo de De Casteljau $u = 0.25$

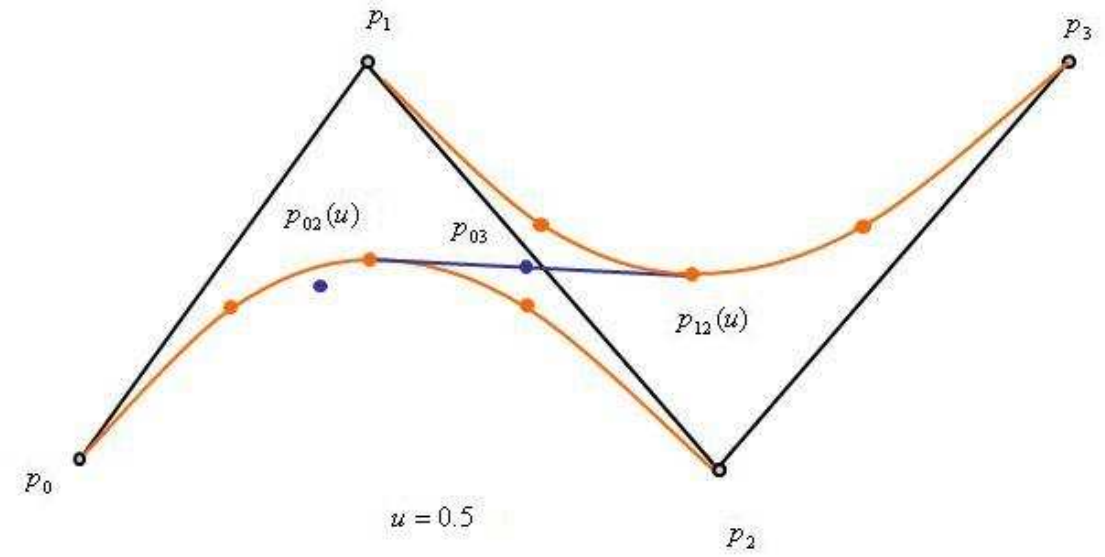


Figura 7. Reaplicação do Algoritmo de De Casteljau $u = 0.5$

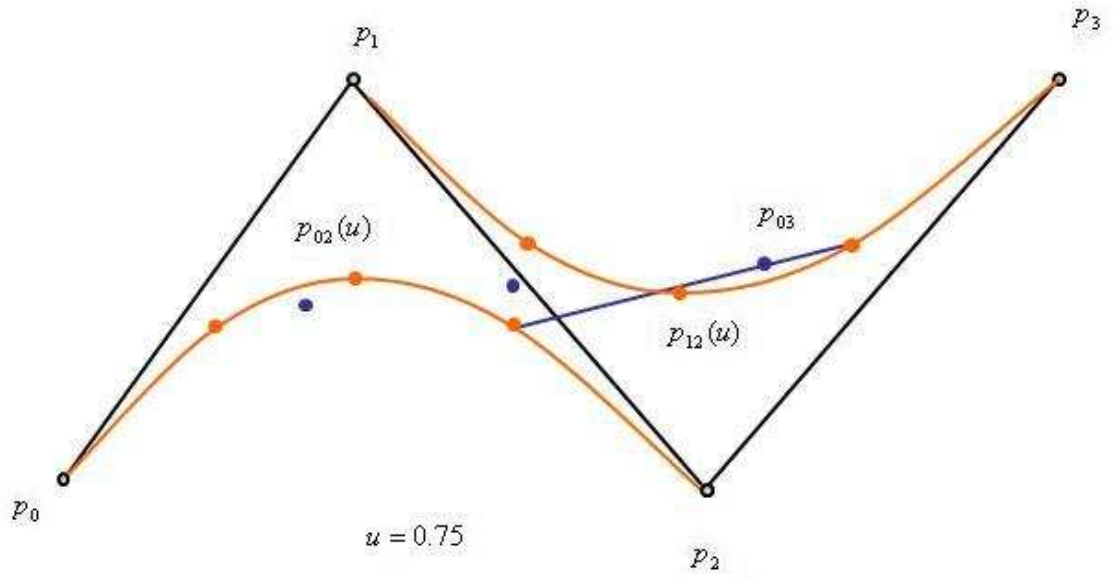


Figura 8. Reaplicação do Algoritmo de De Casteljau $u = 0.75$

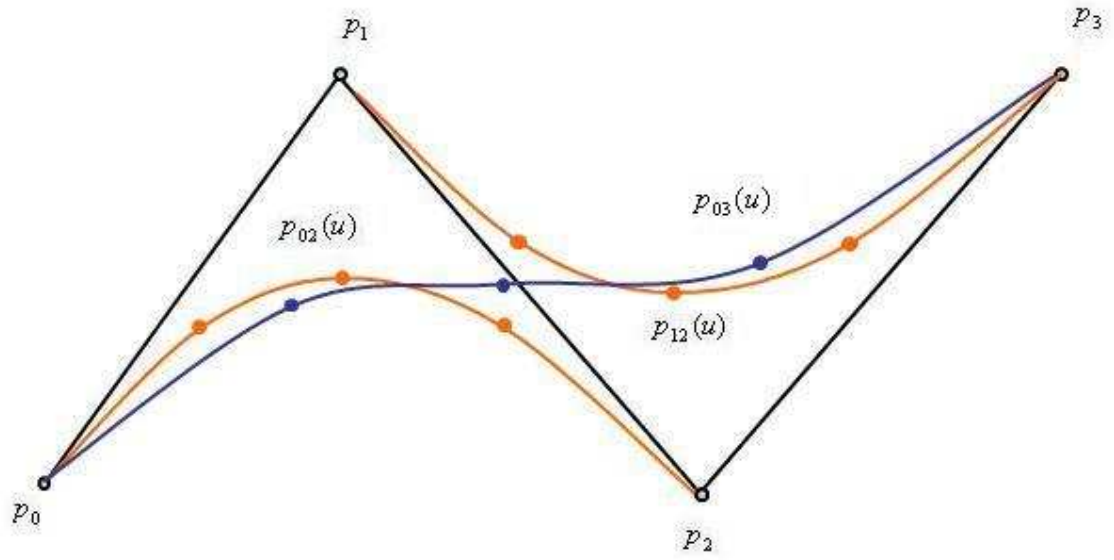


Figura 9. Reaplicação do Algoritmo de De Casteljau

- Novamente temos uma curva dada pela soma de 4 funções de mistura (agora cúbicas), cada uma multiplicada por um dos 4 pontos

$$b_{03}(u) = (1-u)^3$$
$$b_{13}(u) = 3u(1-u)^2$$
$$b_{23}(u) = 3u^2(1-u)$$
$$b_{33}(u) = u^3 \qquad (5)$$

- Em geral, uma curva de grau *n* pode ser construída desta forma e será expressa por

$$p_{ox}(u) = \sum_{j=0}^{n} b_{jn}(u)\, p_j \qquad (6)$$

Assim como a curva de Hermite, a curva de Bézier também possui pontos de controle, porém quatro (P_1, P_2, P_3 e P_4) ao invés de dois, como na fórmula de Hermite, e dois vetores tangentes (R_1 e R_4), que equivale a T_1 e T_2 em Hermite.

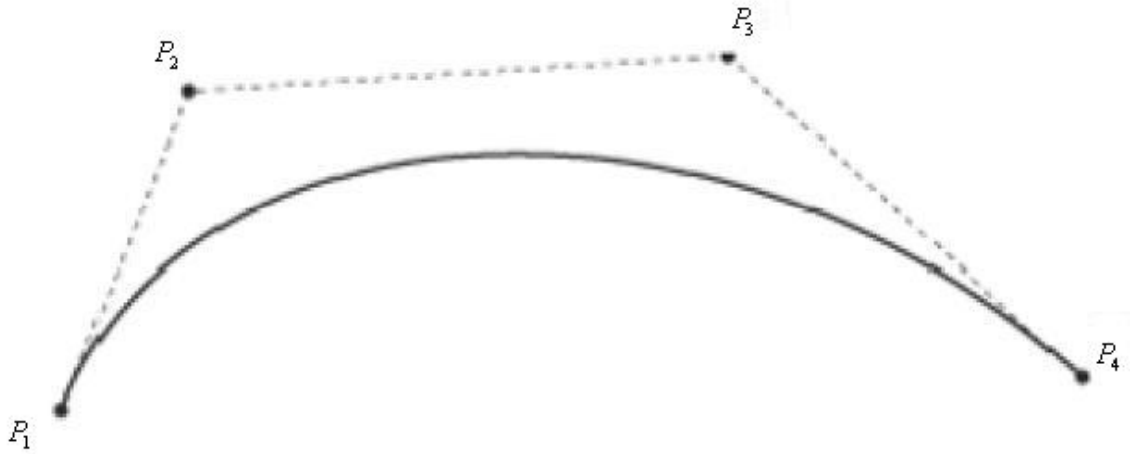


Figura 10. Curva de Bézier

Dado que P_1 e P_4 são os pontos inicial e final da curva, e os pontos P_2 e P_3 os pontos intermediários, os vetores tangentes são determinados como:

$$R_1 = 3(P_2 - P_1)$$
$$R_4 = 3(P_4 - P_3) \quad (7)$$

A de Bézier também é composta pela ponderação de polinômios, denominados polinômios de Bernstein (Figura 11).

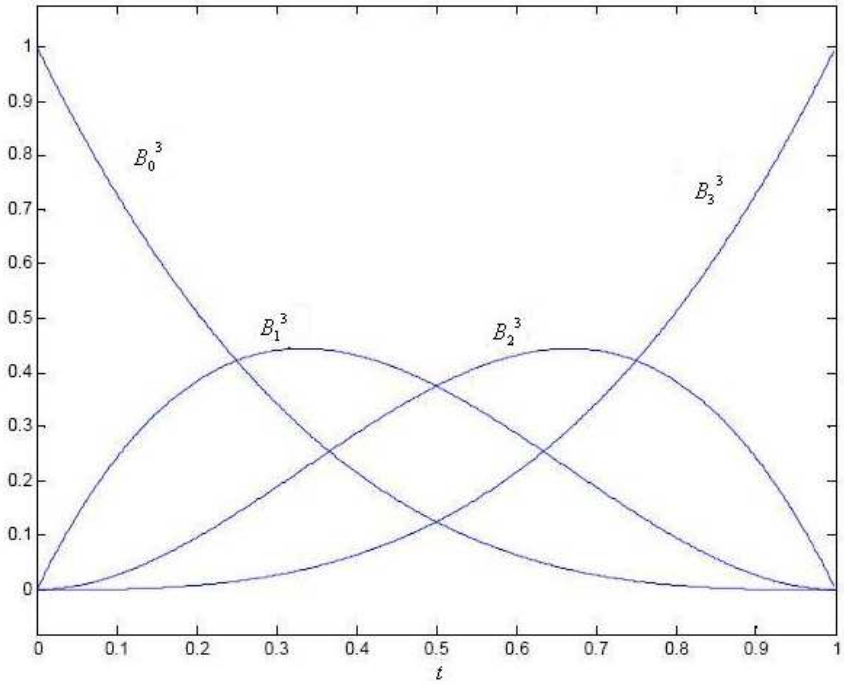


Figura 11. Polinômios de Bernstein: funções de base para a curva de Bézier

Para o caso de terceira ordem, o polinômio que descreve a curva de Bézier é:

$$x(t) = P_x = a_x t^3 + b_x t^3 + c_x t + d_x = C_X T = G_X M_B T \tag{8}$$

onde,

$$G_X = \begin{bmatrix} P_{1x} & P_{2x} & P_{3x} & P_{4x} \end{bmatrix} \tag{9}$$

e M_B é a matriz de coeficientes da base Bézier

$$M_B = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \tag{10}$$

A figura 11 monstra que cada ponto de controle possui influência sobre a curva. Para $B = 4$, a função de Bézier expande o conceito de ponderação e assume caráter polinomial, ou seja, controle não-local, onde todos os pontos de controle participam da formação ponderada da curva.

$$G_X = \begin{bmatrix} P_{1x} & P_{2x} & P_{3x} & P_{4x} & P_{5x} \end{bmatrix} \tag{11}$$

e M_4 é a matriz de coeficientes da base Bézier

$$M_4 = \begin{bmatrix} 1 & -4 & 6 & -4 & 1 \\ -4 & -12 & -12 & 4 & 0 \\ 6 & -12 & 6 & 0 & 0 \\ -4 & 4 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{12}$$

Outra formulação bem conhecida e utilizada da literatura, é a curva de Bézier paramétrica $b(t)$ considerada como a somatória dos polinômios de Bernstein multiplicados pelos pontos de controle b_i.

$$b(t) = \sum_{i=0}^{n} b_i B_i^n(t) \tag{13}$$

onde *n* é o grau da curva de Bézier. As funções $B_t^n(t)$ são os polinômios de Bernstein, e são definidos por (FARIN, 1990):

$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i}, \text{ onde } \binom{n}{i} = \begin{cases} \dfrac{n!}{i!(n-i)!} & se\ 0 \le i \le n \\ 0 & \text{caso contrário} \end{cases} \tag{14}$$

Para o caso mais comum, onde n=3 tem-se:

$$B_0^3(t) = (1-t)^3 = -t^3 + 3t^2 - 3t + 1$$
$$B_1^3(t) = 3t.(1-t)^2 = 3t^3 - 6t^2 + 3t$$
$$B_2^3(t) = 3t^2.(1-t) = -3t^3 + 3t^2$$
$$B_3^3(t) = t^3 \tag{15}$$

Observa-se que os coeficientes de *t* dos polinômios de Bernstein para o grau 3 são os menos coeficientes da matriz M_B da Equação (14). Cada polinômio $B_i^n(t)$ tem um único valor de máximo, que ocorre em $t=i/n$. Isso causa o que é chamado de controle “pseudo-local”, ou seja, ao mover-se um dos pontos de controle, a forma da curva será afetada mais fortemente na região próxima a este ponto, todavia, toda a curva é afetada (PARENTE, 2000).

Os polinômios de Bernstein possuem algumas propriedades que são passadas às curvas de Bézier. A principal delas é que qualquer polinômio é sempre positivo no intervalo de [0,1] e que:

$$B_0^n(0) = 1 \tag{16}$$
$$B_n^n(1) = 1$$

Além disso, pela Figura 11 pode-se ver que a somatória de todos os polinômios resulta na unidade:

$$\sum_{i=0}^{n} B_i^n(t) = 1 \tag{17}$$

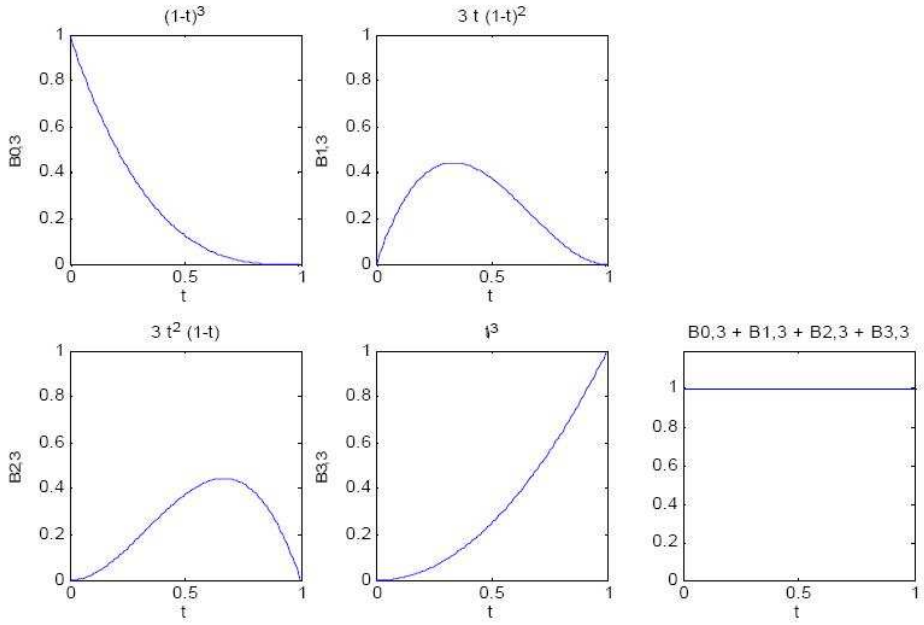


Figura 12. Polinômio cúbicos de Bernstein

Com estas propriedades, pode-se chegar a uma propriedade da curva de Bézier, denominada de “convex hull”, que diz que a curva de Bézier está completamente dentro do maior polígono convexo formado pelos pontos de controle P_1 a P_4. Qualquer transformação de escala, de translação ou de rotação da curva pode ser feita apenas realizando as transformações dos pontos de controle. Esta propriedade também é utilizada para a checagem de interferência entre curvas. Através da interferência entre curvas pode-se citar como exemplo de aplicação o cálculo de interferência de braços robóticos. Se as trajetórias de dois braços são descritas por curvas de Bézier, em vez de se realizar o cálculo de possíveis intersecções, basta fazer o teste do polígono convexo, que é muito mais simples (FARIN, 1990).

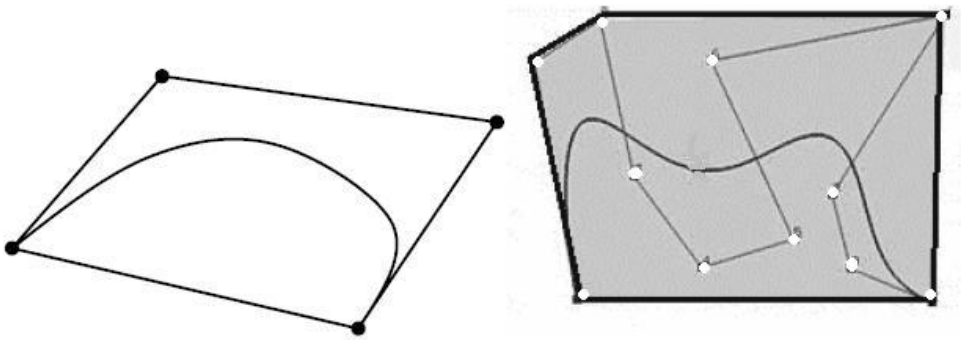


Figura 13. Curvas de Bézier

Necessitará unir duas curvas de Bézier com os seguintes pontos de controle:

Curva 1: P_1, P_2, P_3 e P_4

Curva 2: P_4, P_5, P_6, P_7, P_8, P_9, P_{10}, P_{11}, P_{12}, P_{13} e P_{14}

Então, devemos garantir a continuidade paramétrica C^1 e a continuidade geométrica G^1, já que tendo o ponto P_4 em comum, garantimos já a continuidade C^0 e G^0.

C^1 é dada por: $P_3 - P_4 = P_4 - P_5$

G^1 é dada por: $P_3 - P_4 = k(P_4 - P_5)$, onde $k > 0$

Se garantirmos que estas relações são válidas, então se tem um método para unir os segmentos de curvas.

A obtenção das curvas de Bézier para 4 pontos (P_1-P_2-P_3-P_4) segue o mesmo raciocínio descrito anteriormente, ponderando duas curvas, $C_1 \rightarrow P_1$-P_2-P_3 e $C_2 \rightarrow P_2$-P_3-P_4. O desenvolvimento desta idéia é o seguinte, com base na figura abaixo:

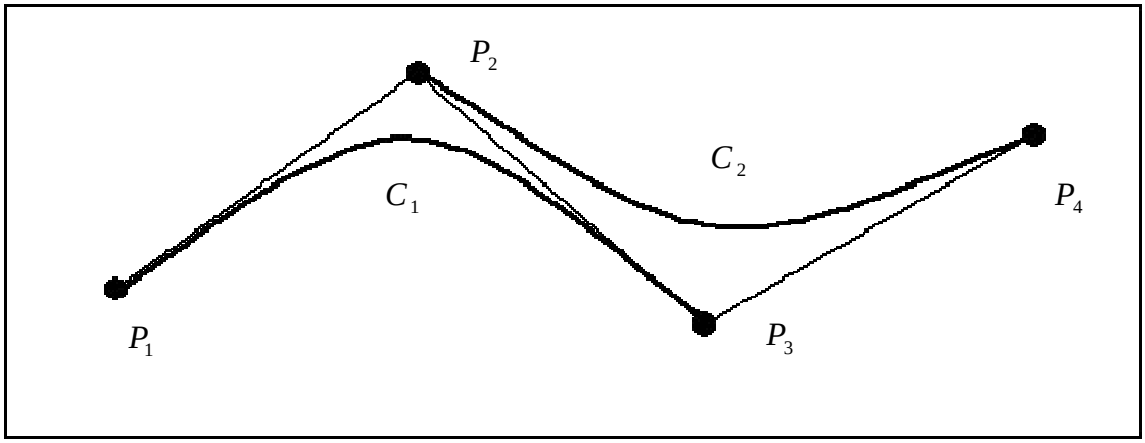


Figura 14. Curvas de Bézier com 4 pontos

Desta maneira, dadas as curvas C_1 e C_2, definidas parametricamente, em função de t, a curva C_3 pode ser expressa por:

$$C_3 = (1-t) * C_1 + t * C_2 \qquad (18)$$

Cujo desenvolvimento resulta em:

$$C_3(t) = (1-t)^3 * P_1 + 3 * t * (1-t)^2 * P_2 + 3 * t^2 * (1-t) * P_3 + t^3 * P_4 \qquad (19)$$

Para t entre 0 e 1.

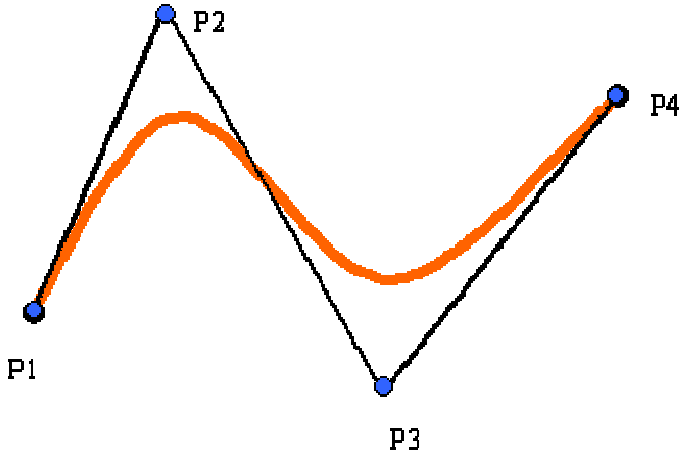


Figura 15. Curvas de Bézier por 4 pontos

Dada a equação 15 podemos dizer que para 6 pontos onde $n = 5$ temos:

$$P(t) = (1-t)^5 P_0 + +5t(1-t)^4 P_1 + 10t^2(1-t)^3 P_2 + 10t^3(1-t)^2 + 5t^4(1-t)P_4 + t^5 P_5 \quad (20)$$

1.2 Curvas B-Splines

Existem famílias de funções de mistura que podem ser utilizadas para gerar qualquer curva spline para um determinado vetor de nós. Tais famílias são chamadas de *funções base* para as splines, com o significado de que toda e qualquer curva spline pode ser obtida por uma fórmula geral, com os pontos de controle adequados. Existem muitas famílias de funções de mistura que são *funções bases*, mas existe uma em especial que oferece o melhor suporte e conseqüentemente o melhor controle local: são as *Basis-splines*, ou apenas *B-Splines*. As curvas *B-Spline* são definidas por:

$$P(t) = \sum_{k=0}^{L} p_k . N_{k,m}(t) \quad (21)$$

E as funções *B-Spline* por:

$$N_{k,m}(t) = \left(\frac{t - t_k}{t_{k+m-1} + t_k} \right) . N_{k,m-1}(t) + \left(\frac{t_{k+m} - t}{t_{k+m} - t_{k+1}} \right) . N_{k+1,m-1}(t) \quad (22)$$

sendo: t_k : os valores do parâmetro;

$T = (t_0, t_1, t_2, ...)$: vetor de nós;

$L = k + 1$: número de pontos de controle;

"*m*": é a ordem das funções *B-Splines*.

A fórmula acima é uma definição recursiva para a construção de uma função de ordem "*m*" a partir de duas funções *B-Spline* de ordem "*m*-1". A função de primeira ordem é igual a 1 dentro dos seguintes intervalos:

$$N_{k,1}(t) = \begin{cases} 1 \text{ para } t_k \le t \le t_{k+1} \\ 0 \text{ para outro intervalo} \end{cases}$$

$$N_{0.2}(t) = \frac{t}{l}.N_{0,1}(t) + \frac{2-t}{l}.N_{1,1}(t) \qquad (23)$$

Características das *B-Splines*:

- Permitem um controle local melhor porque cada ponto de controle é associado a uma função *B-Spline*. O controle local caracteriza a capacidade de modificação de um dos polinômios de uma curva sem alterar os demais. Assim, cada ponto de controle exerce influência na curva somente no intervalo de nós no qual sua respectiva função base *B-Spline* não é nula (no seu suporte);

- Possibilita a mudança na ordem da função base e conseqüentemente no grau da curva, sem modificar o número de pontos de controle.

Um exemplo de função base é a primeira função *B-spline* de ordem *m*=2, para os nós eqüidistantes, onde:

$$N_{k,1}(t) = \begin{cases} t & \text{para } 0 \le t \le 1 \\ 2-t & \text{para } 1 \le t \le 2 \\ 0 & \text{para outro intervalo} \end{cases}$$

$N_{k,1}(t)$ é um pulso triangular deslocado de uma unidade de "*t*" $N_{i,2}(t) = N_{0,2}(t).(t-i)$.

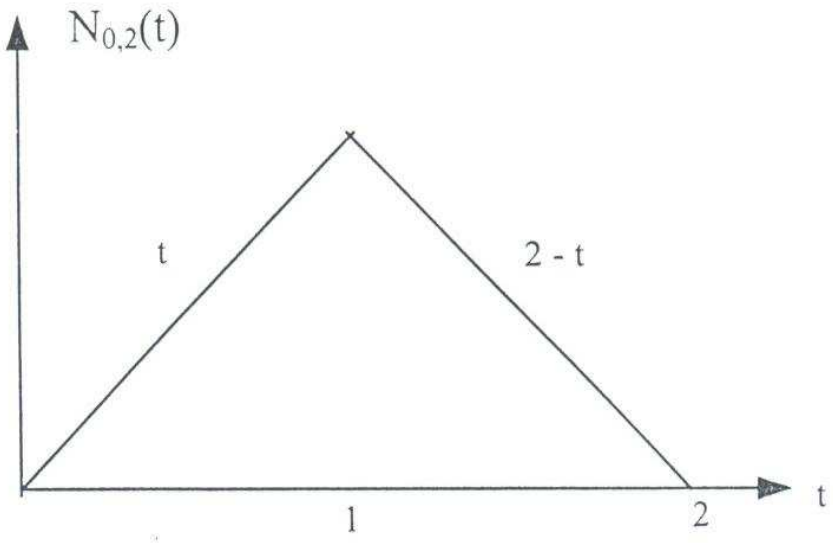


Figura 16. Função base para a spline de primeiro grau

$$N_{0,3}(t) = \frac{t}{2}.N_{0,2}(t) + \frac{3-t}{2}.N_{1,2}(t) \qquad (24)$$

Para funções *B-spline* quadráticas (*m*=3) baseadas nos mesmos nós eqüidistantes, é necessário obter somente $N_{0,3}(t)$, pois as outras funções são somente translações desta: o primeiro termo é uma rampa vezes o primeiro pulso triangular, e o segundo termo é uma rampa decrescente vezes o segundo pulso, produzindo duas parábolas coicidentes em uma extremidade. Quando as duas funções em $N_{0,3}(t)$ são somadas, as extremidades desaparecem e o pulso resultante tem uma derivada contínua. O segmento médio envolve a soma de duas quadráticas resultando:

$$N(t) = \begin{cases} 0 & \text{para outro intervalor} \\ t^2/2 & \text{para } 0 \le t \le 1 \\ 3/4 - (t-3/2)^2 & \text{para } 1 \le t \le 2 \\ (3-t)^2/2 & \text{para } 2 \le t \le 3 \end{cases}$$

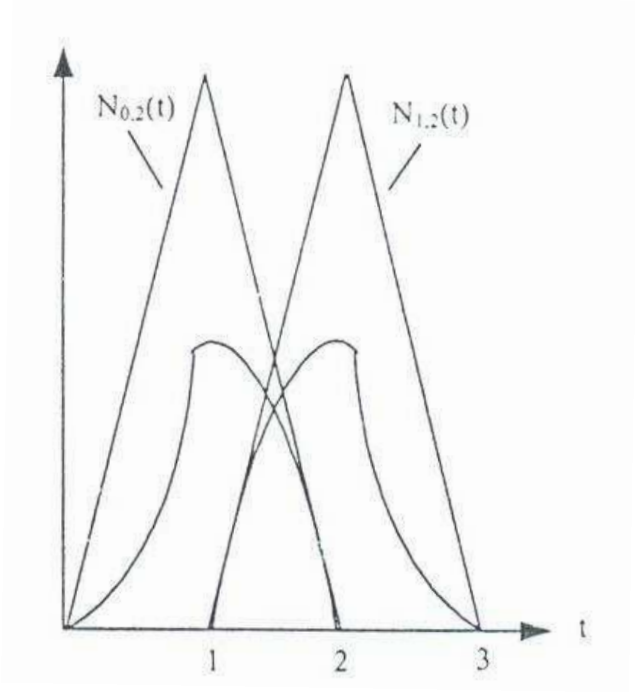


Figura 17. Funções base para a spline quadrática

As outras forma de *spline* quadráticas, $N_{k,3}(t)$ são obtidas facilmente quando os nós são eqüidistantes: se o nó $t_k = k$ então $N_{k,m}(t) = N_{0,m}(t-k)$.

A *spline* cúbica é a *spline* mais utilizada, cuja função $N_{0,4}(t)$, sendo simétrica ao redor de *t*=2 e pode ser representada compactamente como:

$$N_{0,4}(t) = \begin{cases} u.(1-t) & \text{para } 0 \le t \le 1 \\ v.(2-t) & \text{para } 1 \le t \le 2 \\ v.(t-2) & \text{para } 2 \le t \le 3 \\ u.(t-3) & \text{para } 3 \le t \le 4 \\ 0 & \text{para outro intervalo} \end{cases}$$

onde os dois segmentos $u(t)$ e $v(t)$ são dados por:

$$u(t) = \frac{1}{6}.(1-t)^3 \tag{25}$$

$$v(t) = \frac{1}{6}.(3.t^3 - 6.t^2 + 4) \tag{26}$$

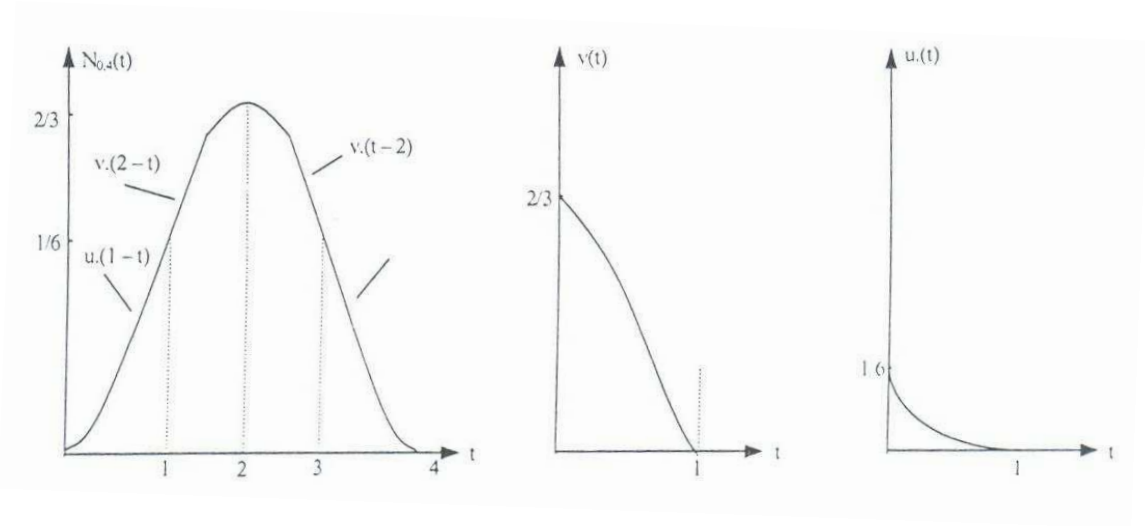


Figura 18. Funções base para a spline cúbica

O cálculo das derivadas mostra que a primeira e segunda derivada da *spline* cúbica são contínuas. Baseado no raciocínio exposto nas funções anteriores, a função $N_{k,m}(t)$ inicia em t_k e termina em t_{k+m}, seu suporte é $[t_k, t_{k+m}]$, e é positiva.

1.3 Rational B-Spline

O método *Rational B-Splineu* usado para interpolar *n* pode ser expresso da seguinte maneira:

$$S(x) = \sum_{j=1}^{n} A_j R_{j,k}(x) \tag{27}$$

$$S^{(m)}(x) = \sum_{j=1}^{n} A_j R_{j,k}^{(m)}(x) \tag{28}$$

Quando

$R_{j,k}$ = *Rational B-Spline* de ordem *k* (condição = $k-1$).

$R_{j,k}^{(m)}$ = *m*º derivada de *RAtional B-Spline* de ordem *k*.

A diferença entre o método *Rational B-Spline* e o método *B-Spline* é usar o método *B-Spline* de funções básicas substituídas pelo uso da *Rational B-Spline* de funções básicas.

Uma *Rational B-Spline*, $R_{j,k}(x)$, de ordem *k* é definida como:

$$R_{j,k}(x) = W_j N_{j,k}(x) / \sum_{j=1}^{n} W_j N_{j,k}(x) \tag{29}$$

Onde W_j é uma seqüência com valores positivos. A *B-Spline*, $N_{j,k}(x)$ da equação 76 com *knots*, T_j, T_{j+1},...,T_{j+k}, é definido pela relação de repetição como:

$$N_{j,k}(x) = \frac{x - T_j}{T_{j+k-1} - T_j} N_{j+k-1}(x) + \frac{T_{j+k} - x}{T_{j+k} - T_{j+1}} N_{j+1,k-1}(x) \tag{30}$$

Para k >1, e j=1,..., *n*.

$$N_{j,1}(x) = \begin{cases} 1, \; T_j \le x \le T_{j+1} \\ 0, \text{ outro intervalo} \end{cases} \tag{31}$$

Para k=1

A equação 29 mostra que um *Rational B-Spline* de ordem *k* pode ser obtida pela modificação da *B-Spline* de ordem *k*. As equações 29-31 produz valores de $R_{j,k}(x)$ para um ordem especificada, seqüência knot, e uma importante seqüência no domínio de interesse pela combinação linear de quantidades positivas.

Rational B-Spline tem um numero de propriedades interessantes, algumas são pertinentes para os procedimentos numerais usados em valores estimados e aplicações. O valor diferente de zero de $R_{j,k}(x)$ ocorre somente no intervalo $\lfloor T_j, T_{j+k} \rfloor$ em uma seqüência knot. Em outras palavras, uma *Rational B-Spline* de ordem *k* é diferente de zero somente de novo *k* de intervalos adjacentes entre knots. Um outra característica de *Rational B-Spline* de ordem *k* é que para qualquer ponto determinado, *x*, no domínio *k*

adjacente $R_{j,k}(x)$ é zero. Valores diferentes de zero na *Rational B-Spline* podem ser escritos como:

$$\sum_{j=1}^{n} R_{j,k}(x) = 1 \tag{32}$$

Note que, uma mudança em W_j afeta a *Rational B-Spline* sempre no intervalo de $\left[T_j, T_{j+k}\right]$. Como um resultado, o escopo pode exercer um controle local de movimentos característico pelo ajustamento particular de W_j .

1.4 NURBS

Um tipo derivado das curvas *B-Splines* são os NURBS – *Non Uniform Rational B-Splines – splines* que utilizam funções base racionais e um vetor de nós não uniforme, demonstrando que é um modelo matemático usado para gerar e representar curvas e superfícies. Uma função racional é definida como a razão de duas funções polinomiais (Choi, 1991). Uma desvantagem das curvas *B-Splines* é a dificuldade na representação de seções cônicas (hipérboles, elipses, parábola) em virtude da representação paramétrica destes tipos de curvas se dar na forma de funções racionais, possibilitando a representação de seções cônicas e curvas de formas livres utilizando funções base racionais e um vetor de nós não uniforme, além das características apresentadas pelas curvas *B-Spline*.

Uma curva NURBS $Q(u)$ de grau *p* define um pronto no espaço 2D com o parâmetro escalar *u* assumindo valores dentro de [0,1].

$$Q(u) = \frac{\sum_{i=0}^{n} N_{i,p}(u) w_i p_i}{\sum_{i=0}^{n} N_{i,p}(u) w_i} \tag{33}$$

no qual um conjunto de *n* pontos de controle p_i formam um polígono de controle e w_i são os pesos. Um aumento no peso w_i puxa a curva mais próxima para o ponto de controle p_i . $N_{i,p}(u)$ é a i-ésima função base *B-Spline* de grau *p* (ordem *p*+1).

As NURBS são curvas em R^{n+1} (porque tem coordenadas homogêneas), $Q(u) \in R^{n+1}$.

Um vetor de nós não-uniforme, com uma seqüência não-decrescente de números reais é definida como:

$$u = \left\{ \underbrace{a,...a}_{p+1}, u_{m-p-1}, \underbrace{b,...b}_{p+1} \right\} \tag{34}$$

no qual $0 \le u_i \le u_{i+1} \le 1,\ i = 0, ..., m-1\ e\ m =$ número de nós. Os nós em uma curva NURBS são os pontos em parâmetro espaçado onde as curvas polinomiais racionais são unidas juntas para formar um segmento multicurva.

Considerando a maneira de introduzir coordenadas homogêneas para uma curva *B-Spline* que deriva a definição de NURBS, através de $n+1$ pontos de controle $p_0,\ p_1,..., p_n$ e vetor de nós $u = \{u_0, u_1,..., u_m\}$ de *m*+1 nós, a curva *B-Spline* de grau *p* definida por estes parâmetros é a seguinte:

$$Q(u) = \sum_{i=0}^{n} N_{i,p}(u)\, p_i \tag{35}$$

É necessário deixar o ponto de controle p_i ser reescrito como um vetor coluna com quatro componentes com o quarto sendo 1:

$$p_i = \begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix} \tag{36}$$

Para coordenadas homogêneas, multiplicando a coordenada de um ponto com um número não-nulo não muda sua posição. Pode-se multiplicar a coordenada de p_i com w_i para obter uma nova forma em coordenadas homogêneas:

$$p_i = \begin{bmatrix} w_i x_i \\ w_i y_i \\ w_i z_i \\ w_i \end{bmatrix} \tag{84}$$

Substituindo esta nova forma homogênea na equação da curva *B-Spline* anterior, será obtido o seguinte:

$$Q(u) = \sum_{i=0}^{n} N_{i,p}(u)\, p_i = \sum_{i=0}^{n} N_{i,p}(u) \begin{bmatrix} w_i x_i \\ w_i y_i \\ w_i z_i \\ w_i \end{bmatrix} = \begin{bmatrix} \sum_{i=0}^{n} N_{i,p} w_i x_i \\ \sum_{i=0}^{n} N_{i,p} w_i y_i \\ \sum_{i=0}^{n} N_{i,p} w_i z_i \\ \sum_{i=0}^{n} N_{i,p} w_i \end{bmatrix} \tag{37}$$

Então, o ponto $Q(u)$ também está em uma forma de coordenadas homogêneas. É possível converter a fórmula anterior para coordenada cartesiana dividindo $Q(u)$ pelo quarto componente:

$$Q(u) = \begin{bmatrix} \dfrac{\sum_{i=0}^{n} N_{i,p}(u)(w_i x_i)}{\sum_{i=0}^{n} N_{i,p}(u)(w_i)} \\ \dfrac{\sum_{i=0}^{n} N_{i,p}(u)(w_i y_i)}{\sum_{i=0}^{n} N_{i,p}(u)(w_i)} \\ \dfrac{\sum_{i=0}^{n} N_{i,p}(u)(w_i z_i)}{\sum_{i=0}^{n} N_{i,p}(u)(w_i)} \\ 1 \end{bmatrix} = \frac{\sum_{i=0}^{n} N_{i,p}(u) w_i}{\sum_{i=0}^{n} N_{i,p}(u) w_j} \begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix} \tag{38}$$

Finalmente, atingi-se a fórmula a seguir:

$$Q(u) = \frac{1}{\sum_{i=0}^{n} N_{i,p}(u) w_i} \sum_{i=0}^{n} N_{i,p}(u) w_i p_i \tag{39}$$

Isto é a curva NURBS de grau *p* definida pelos pontos de controle $p_0, p_1,..., p_n$, vetor de nós $u = \{u_0, u_1,..., u_m\}$, e pesos $w_0, w_1,..., w_n$. Pode-se observar que em virtude do peso w_i ser associado com o ponto de controle p_i como seu quarto componente, o número de pesos e o número de pontos de controle devem ser iguais.

Podemos então resultar pela definição de NURBS que:

1- Se todos os pesos forem iguais a 1, uma curva NURBS reduz para uma curva *B-Spline*. Neste caso, pontos de controle em formas homogenias são idênticos para sua forma cartesiana convencional e o denominador é 1.

2- As curvas NURBS são racionais, sendo que o valor multiplicado para os pontos de controle p_i, $N_{i,p}(u)w_i$, é um polinômio de grau *p*. O denominador é a soma de todos os coeficientes e, conseqüentemente, também é um polinômio de grau *p*. Como resultado, o coeficiente do ponto de controle p_i é o quociente de dois polinômios de grau *p* e a função $Q(u)$ é racional.

Desta maneira podemos ver que as curvas *B-Spline* são casos especiais de curvas NURBS. Além disso, visto que curvas NURBS são racionais, círculos, elipses, parábolas e muitas outras curvas que são impossíveis com curvas *B-Spline* são agora possíveis como curvas NURBS.

Mas é possível questionar se as curvas NURBS são tipos especiais de curvas. Na verdade, elas são simplesmente outro tipo de curvas *B-Spline*. Considerando o ponto de controle $p_i = (w_i x_i, w_i y_i, w_i z_i, w_i)$, este ponto revela quatro componentes e pode ser tratado como um ponto no quadro espaço, e conseqüentemente, $Q(u)$ abaixo torna-se uma curva *B-Spline* em quatri dimensões:

$$Q(u) = \sum_{i=0}^{n} N_{i,p}(u)\, p_i = \sum_{i=0}^{n} N_{i,p}(u) \begin{bmatrix} w_i x_i \\ w_i y_i \\ w_i z_i \\ w_i \end{bmatrix} = \begin{bmatrix} \sum_{i=0}^{n} N_{i,p}(w_i x_i) \\ \sum_{i=0}^{n} N_{i,p}(w_i y_i) \\ \sum_{i=0}^{n} N_{i,p}(w_i z_i) \\ \sum_{i=0}^{n} N_{i,p}(w_i) \end{bmatrix} \tag{40}$$

Em uma interpolação geométrica de coordenadas homogêneas, dividindo os primeiros três componentes de coordenadas pelo quarto, adquire-se o equivalente a projetar um ponto em quatro dimensões para o pano *w*=1. Tendo em vista que a curva acima é convertida em uma curva NURBS, dividendo as três primeiras coordenadas com o quarto componente, concluí-se que uma curva NURBS no espaço tridimensional é meramente a projeção de uma curva *B-Spline* em quadri dimensões espaciais.

Determinadas propriedade importantes de NURBS são dado um conjunto de *n*=1 pontos de controle p_0, p_1,..., p_n, cada um é associado com um peso não negativo w_i (isto é, p_i tem peso $w_i \geq 0$), e um vetor de nós $u = \{u_0, u_1, ..., u_m\}$ de *m*+1 nós, a curva NURBS de grau *p* é definida como segue:

$$Q(u) = \sum_{i=0}^{n} R_{i,p}(u)\, p_i \tag{41}$$

no qual $R_{i,p}(u)$ e definida a seguir:

$$R_{i,p}(u) = \frac{N_{i,p}(u) w_i}{\sum_{j=0}^{n} N_{j,p}(u) w_j} \tag{42}$$

Desta maneira podemos destacar que NURBS é uma generalização de *B-Spline*, herdando todas as propriedades de *B-Spline*. As mas importantes para funções base NURBS são:

1- $R_{i,p}(u)$ é uma função racional de grau *p* em *u*.

2- Não negatividade – para todos *i* e *p*, $R_{i,p}(u)$ é não negativo.

3- Suporte Local - $R_{i,p}(u)$ é um não-nulo em $\lfloor u_i, u_{i+p+1} \rfloor$.

4- Em qualquer período de nó $\lfloor u_i, u_{i+p+1} \rfloor$, no máximo IPI+1 funções base de grau *p* são não-nulas, isto é: $R_{i-p,p}(u)$, $R_{i-p+1,p}(u)$, $R_{i-p+2,p}(u)$,..., e $R_{i,p}(u)$.

5- Partição de Unidade – a soma de todas funções base não-nulas de grau *p* no período $\lfloor u_i, u_{i+p+1} \rfloor$ é 1.

6- Se o número de nós é *m*+1, o grau das funções base é *p* e o número de funções base de grau *p* é *n*+1, então $m = n + p + 1$.

7- Função base $R_{i,p}(u)$ é uma curva composta de funções racionais de grau *p* com pontos de acoplamento nos nós em $\lfloor u_i, u_{i+p+1} \rfloor$.

8- em um nó de multiplicidade *k*, função base $R_{i,p}(u)$ é C^{p-k} contínuo.

9- Se $w_i = c$ para todo *i*, onde *c* é uma constante não-nula, $R_{i,p}(u) = N_{i,p}(u)$.

Podemos concluir que funções base *B-Spline* são casos especiais de funções base NURBS quando todos os pesos tornarem uma constante não-nula. Assim deve-se mencionar em especial *c*=1.

Nas importantes propriedades de curvas NURBS pode-se perceber que existe a possibilidade de uma NURBS estar aberta, semi-fechada e fechada. Como curvas *B-Spline,* se os primeiros $p+1$ nós e os últimos $p+1$ nós são iguais para o fim da esquerda e da direita do domínio, a curva é semi-fechada.

1- Curva NURBS $Q(u)$ é uma curva por partes continuadas com cada componente uma curva racional de grau *p*.

2- A igualdade $m = n + p + 1$ deve ser satisfeita.

3- Uma curva NURBS semi-fechada $Q(u)$ passa pelos dois pontos de controle finais p_0 e p_n.

4- Propriedade de Fecho Convexo Forte: a curva NURBS é contida no fecho convexo de seus pontos de controle. Além disso, se *u* está no período de nós

$\lfloor u_i, u_{i+p+1} \rangle$, então $Q(u)$ está no fecho convexo dos pontos de controle $p_{i-p}, p_{i-p+1}, \ldots, p_i$.

Esse processo tem sido executado de modo nítido, permitindo que todos os pesos devem ser não-negativos. Se alguns deles são negativos, a propriedade de fecho convexo forte ou até a propriedade de fecho convexo não suportará. A seguir, a figura 23 é uma curva NURBS de grau 2 com $n = 2$, $m = 5$ e os primeiros três e últimos três nós semi-fechados. Os pesos dos dois pontos de controle terminam em ambos 1 e o peso do ponto de controle do meio é 0.5. isto é uma arco elíptico. O segmento da curva fica contido no fecho convexo.

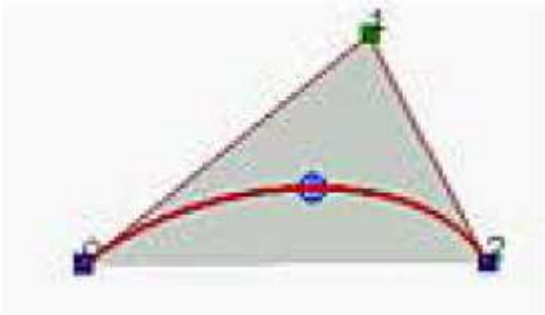


Figura 19. Curva NURBS de Grau 2

A figura 24 apresenta o peso de ponto de controle do meio configurado para zero, assim este ponto de controle não revela nenhum efeito, o resultado é o segmento da linha determinada pelas extremidades. Ainda fica contido no fecho convexo. Se o peso é mudado para -0.5, o seguimento de curva não é contido no fecho convexo e conseqüentemente a propriedade de fecho convexo falha conforme figura 25.

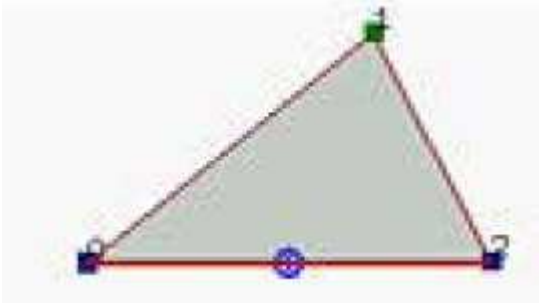


Figura 20. Curva NURBS de Grau 2

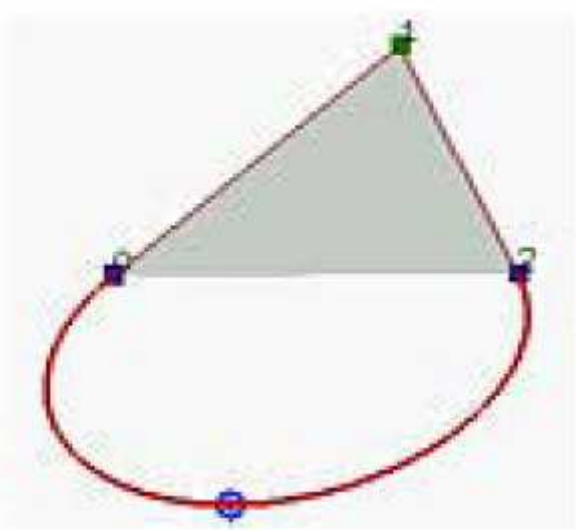


Figura 21. Curva NURBS de Grau 2

5- Esquema de modificação local: mudando a posição do ponto de controle p_i somente afeta a curva $Q(u)$ no intervalo $\lfloor u_i, u_{i+p+1} \rfloor$.

Isto segue da propriedade do esquema de modificação local de funções base *B-Spline*. Recorde que $R_{i,p}(u)$ é não-nulo no intervalo $\lfloor u_i, u_{i+p+1} \rfloor$. Se *u* não está neste intervalo, visto que $R_{i,p}(u)$ é zero e $R_{i,p}(u)p_i$ não tem nenhum efeito em computador $Q(u)$. Por outro lado, se *u* está no intervalo indicado $R_{i,p}(u)$ é não-nulo, e se $R_{i,p}(u)p_i$ é mudado, então deve-se fazer $Q(u)$.

Este esquema de modificação local é muito importante no projeto de curva, porque pode-se alterar uma curva localmente sem mudar a forma de um modo global. Além disso, se uma nova definição de forma da curva é exigida, pode se inserir mais nós (e então mais pontos de controle) de modo que a região afetada possa ser restringida para uma muito menos.

6- $Q(u)$ é C^{p-k} contínuo em um nó de multiplicidade *k*.

Se *u* não é um nós, $Q(u)$ está no meio de um segmento de curva de grau *p* e é então infinitamente diferenciável. Se *u* é um nó no domínio não-nulo de $R_{i,p}(u)$, considerando que $R_{i,p}(u)$ é somente C^{p-k} contínuo, então é necessário fazer $Q(u)$.

7- Propriedade Diminuindo variação

A propriedade diminuindo variação também suporta para curvas NURBS. Se a curva é contida em um plano, a reta não intercepta uma curva NURBS mais vezes que seu polígono de controle.

8- Curvas *B-Spline* e Curvas Bézier são casos especiais de Curvas NURBS.

Se todos os pesos forem iguais, uma curva NURBS torne-se uma curva *B-Spline*. Se, além disso, $n = p$ (isto é, o grau de uma curva *B-Spline* é igual para *n*, o número de pontos de controle menos 1) e existem $2(p+1) = 2(n+1)$ nós com p+1 deles semi-fechados em cada fim, esta curva NURBS reduz para uma curva Bézier.

9- Invariância Projetiva.

2 Mapeamento transfinito

O mapeamento transfinito é onde as curvas de contorno de uma superfície ou as superfícies de contorno de um volume são completamente consideradas (GORDON, 1973). Desta forma curvas de formas livres, inclusive com cantos vivos, podem ser utilizadas para descreverem uma superfície.

No mapeamento transfinito a descrição de um quadrilátero é realizada através de interpolação bilinear entre as curvas opostas do contorno. Além dos contornos podem ser utilizadas curvas internas, que possibilitam a descrição de superfícies curvas com maior grau de complexidade. A interpolação transfinita de corpos tridimensionais segue o mesmo princípio. No caso de um hexaedro realiza-se uma interpolação trilinear entre suas faces.

Para descrever este mapeamento é necessário introduzir o conceito de um projetor *P*. Um projetor é qualquer operador linear idempotente que mapeia uma superfície real numa superfície aproximada, sujeita a certas restrições interpoladoras. Deste modo, há uma enorme variedade de possíveis projetores.

O projetor faz uma interpolação linear entre duas curvas de contorno, Ψ_0 e Ψ_1:

$$P[F] \equiv P(s,t) = (1-t)\,\Psi_0(s) + t\,\Psi_1(s): \qquad 0 \le s \le 1,\ 0 \le t \le 1 \qquad (43)$$

onde *s* é uma coordenada paramética normalizada ao longo de Ψ_0 e Ψ_1, e *t* é uma coordenada paramétrica que tem um valor igual a zero em Ψ_0 e um em Ψ_1.

Conjuntos destes simples projetores lineares podem ser “misturados” para transformar projetores mais complexos que mapearão as curvas de *F* em todos os seus pontos. Um projetor que opera numa região *F* contornada pelas quatro curvas $S_0(t)$, $S_1(t)$, $T_0(s)$, $T_1(s)$, é mostrado na figura 26 (a). Dois projetores básicos podem ser formados, um interpolando na direção S e o outro na direção T:

$$Ps(s,t)=(1-t)T_0(s)+tT_1(s) \qquad 0\leq s\leq 1, \qquad (44)$$

$$Pt(s,t)=(1-s)S_0(t)+sS_1(t) \qquad 0\leq t\leq 1 \qquad (45)$$

Estes projetores são mostrados na figura 22 (b) e (c). O projetor produto *PsPt[F}=PtPs[F]* é mostrado na figura 22 (d). Este projetor mapeia *F* exatamente nos quatro cantos com aproximações lineares ao longo dos quatro lados. Finalmente, o projetor de soma booleana é definido tal que *F* é mapeado em toda a superfície, figura 22 (e):

$$(Ps \oplus Pt) \equiv Pst = Ps + Pt - PsPt \qquad (46)$$

Mapeamento Transfinito baseado em descrições de curvas discretas fornece uma base efetiva para geração automática de malhas de elementos finitos. Eles são gerais, simples de implementar e computacionalmente eficientes. Não há nenhuma restrição na geometria das curvas de contorno quando a forma discreta do mapeamento é usada. Restrições topológicas podem ser impostas ao mapeamento para minimizar a quantidade de dados de entrada requeridos.

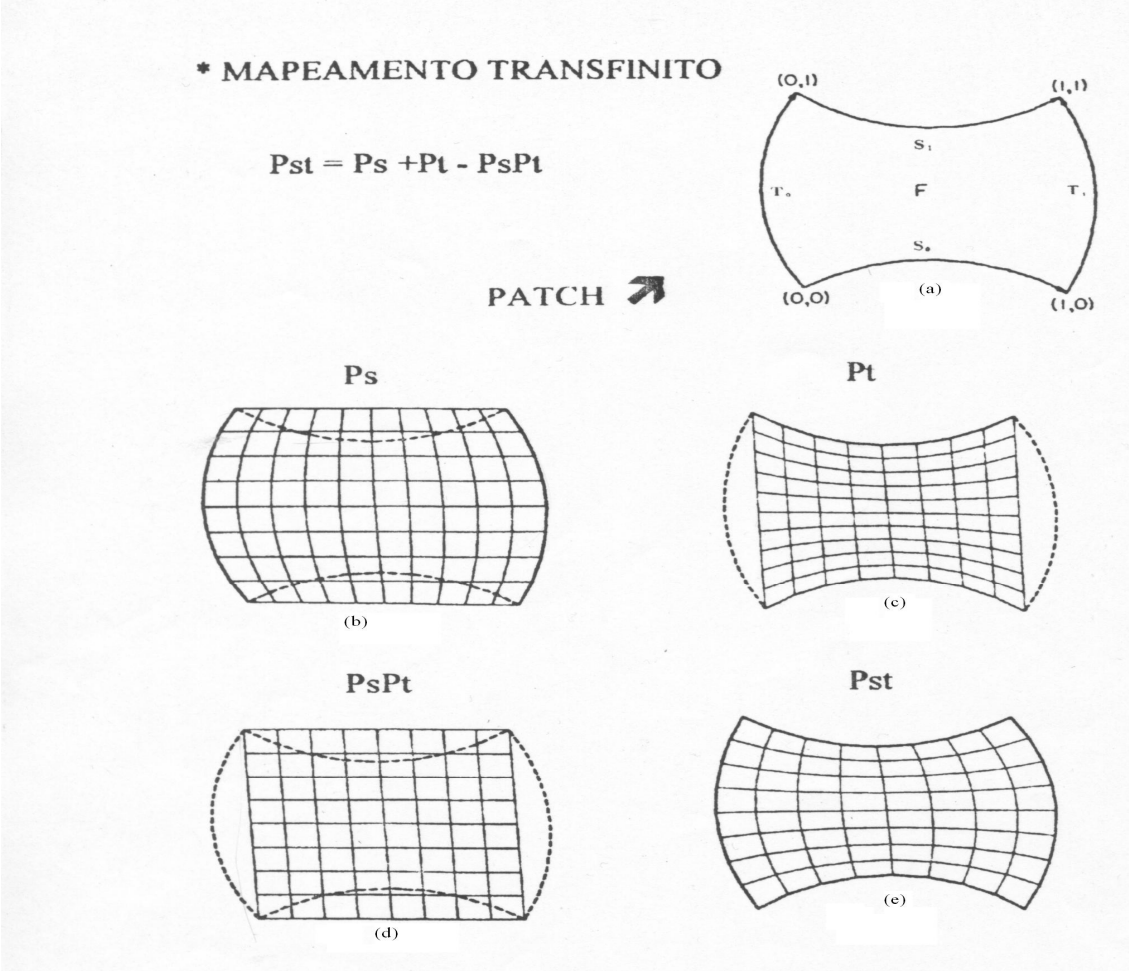
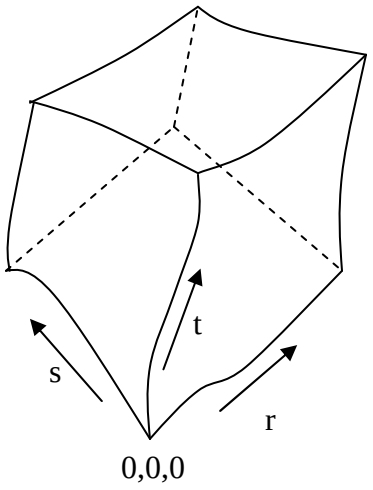


Figura 22. Mapeamento Transfinito Bilinear

No entanto o trabalho se dará em um espaço tridimensional que segue o mesmo principio acima mencionado, sendo utilizado um hexaedro o qual realiza-se uma interpolação trilinear entre suas faces.

Desta forma se verifica a necessidade de utilizar-se de métodos matemáticos, trabalhando-se no entanto com produto escalar e produto vetorial para atender as necessidades de cálculos na geometria do hexaedro.

Assim sendo:



Pode-se utilizar de cálculos como:

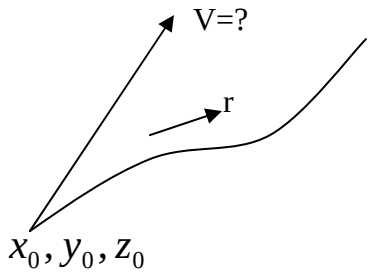
$$\begin{aligned}
&\left[\frac{dx(r)}{dr}\frac{dy(r)}{dr}\frac{dz(r)}{dr}\right]r=0\longrightarrow r'\\
&\left[\frac{dx(s)}{ds}\frac{dy(s)}{ds}\frac{dz(s)}{ds}\right]s=0\longrightarrow s'\\
&\left[\frac{dx(t)}{dt}\frac{dy(t)}{dt}\frac{dz(t)}{dt}\right]t=0\longrightarrow t'
\end{aligned} \qquad (47)$$

Assim como:

$(t'\times r').s'$

Assim sendo para se chegar nos três primeiros cálculos é preciso passar por:

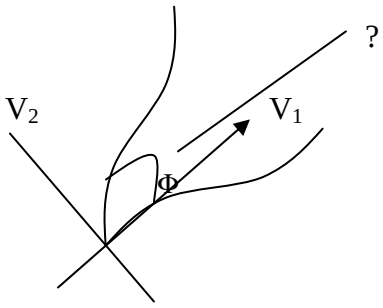
Equação paramétricas de curva



$$\left.\begin{array}{l} x = f_x(r) \\ y = f_y(r) \\ z = f_z(r) \end{array}\right\} t \in [0,1]$$

$$\begin{array}{l} x_0 = f_x(0) \\ y_0 = f_y(0) \\ z_0 = f_z(0) \end{array} \qquad (48)$$

Objetivo: qual o valor de Φ?



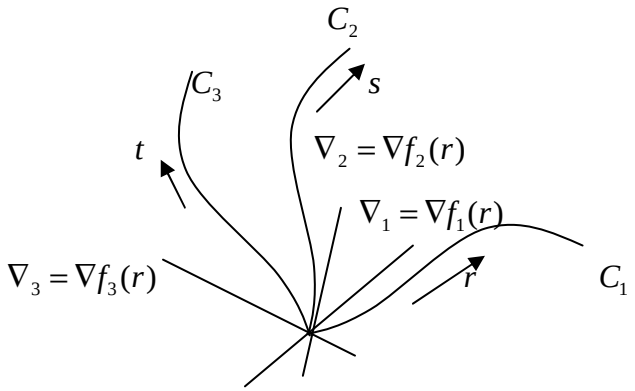
$$\Phi = \cos^{-1}\left(\frac{V_1.V_2}{|V_1||V_2|}\right)$$

$$V_1.V_2 = |V_1||V_2|\cos\Phi$$

$$\begin{array}{l} V_1 = ? \\ V_2 = ? \end{array} \qquad (49)$$

$$V_1 = \left(\frac{\partial f_{x_1}(r)}{\partial r}, \frac{\partial f_{y_1}(r)}{\partial r}, \frac{\partial f_{z_1}(r)}{\partial r}\right) r = 0$$

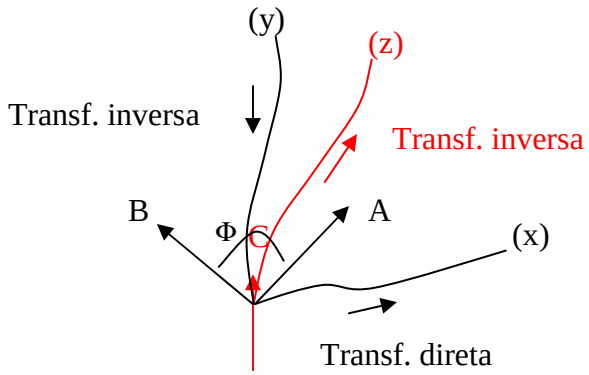
$$\nabla f_{x_1}(F)\Big|_{t=0}$$



$(\nabla_1 \times \nabla_2).\nabla_3 > 0 \longrightarrow$ regra da mão direita

$(\nabla_1 \times \nabla_2).\nabla_3 < 0 \longrightarrow$ regra da mão esquerda

Podendo assim verificar o seguinte método:



Onde:

$$C = \frac{dl_z}{d(r)}\bigg| r = 0 \qquad (50)$$

$$B = \frac{dl_2}{d(r)}\bigg| r = 1$$

$$A = \frac{dl_1}{d(r)}\bigg| r = 1$$

$$A.B = |A||B|\cos\Phi$$

$$\cos\Phi = \frac{A.B}{|A||B|}$$

Φ

0º–60º

60º–120º

120º–180º

3 Gerador de Malha

As tarefas de um gerador de malhas podem ser subdivididas em:

- Entrada e saída de Dados: necessidade de desenvolvimento de rotinas específicas capazes de ler ou escrever todos os componentes da estrutura de dados a ser desenvolvida. Dessa forma, a troca de dados entre os diversos procedimentos do

gerador de malha pode ocorrer tanto através de parâmetro de métodos, variáveis globais como também através de arquivos padrões, o que facilitará o desenvolvimento paralelo dos diversos módulos.

- Processador Geométrico: durante a preparação da geometria para a utilização da geração de malhas é preciso o emprego de processadores geométricos como do processador que faz a conversão de informações geométricas contidas no modelo para a estrutura de dados do gerador a ser desenvolvido, o processador de subdivisões que transforma uma superfície em quadrilátero e um volume em hexaedros, o processador de interferência que analisa a disposição das linhas, superfícies e volumes entre si, e o processador de densidade que determina a distribuição de elementos finitos em cada uma das entidades geométricas.
- Gerador Topológico de malhas: para cada entidade geométrica (linha, quadrilátero ou hexaedro) é realizada a geração de malha em um espaço paramétrico unitário (ordem: 1ª- reta, 2ª- quadrado ou 3ª- cubo). A distribuição de elementos ocorre levando-se em conta o resultado do processador de densidade. Esse procedimento é realizado de acordo com a hierarquia geométrica. Assim uma reta de comprimentos unitário é subdividade em diversas linhas, um quadrado unitário em quadriláteros e um cubo unitário em hexaedros.
- Mapeador de malhas: o mapeador projeta as malhas locais geradas topologicamente em linhas, superfícies ou volumes localizados no espaço. O mapeador a ser utilizado precisa garantir que pontos, linhas e superfícies de contornos da geometria unitária descrita no espaço paramétrico sejam projetados exatamente em pontos, linhas e superfícies da geometria euclidiana.
- Processador de malhas: após a geração e mapeamento de malhas locais pode-se alterar a malha localmente ou como um todo, onde o interpolador de elementos altera o grau de interpolação de elementos isoladamente ou em grupos; o montador de malhas agrupa as diversas malhas locais em uma malha global, eliminando informações redundantes, tais como dois nós localizados no mesmo ponto; o processador de cargas subdivide eventuais esforços definidos sobre linha, superfícies ou volumes em forças sobre os nós; o processador de condições de contorno relaciona eventuais restrições de movimentos definidas sobre linhas ou superfícies aos respectivos nós.

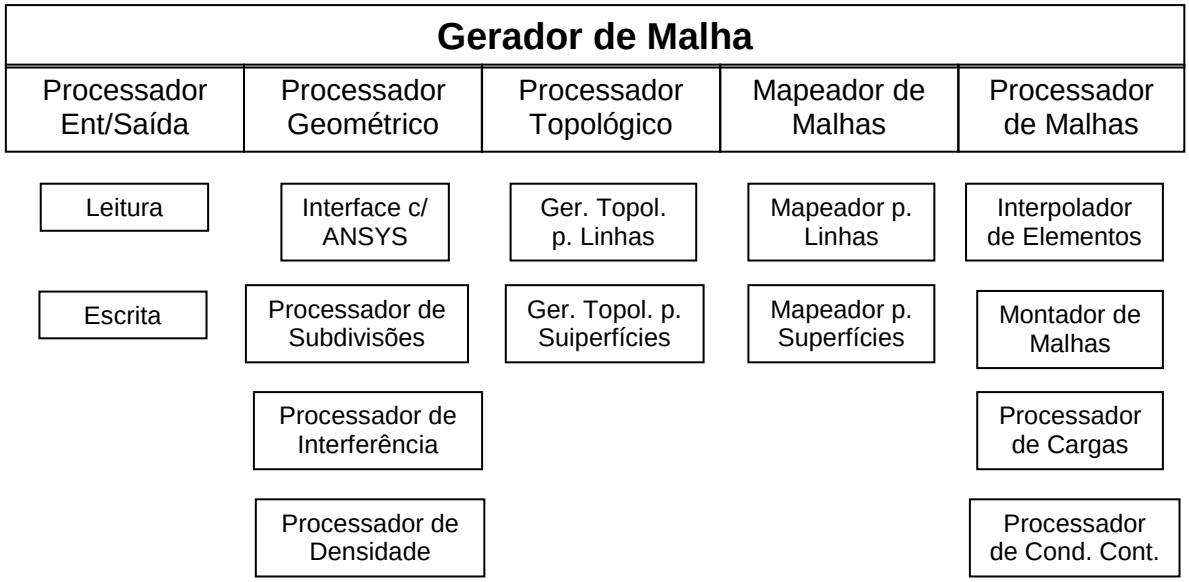


Figura 23. Subdivisão funcional de um gerado de malhas

O gerador topológico irá trabalhar em um espaço unidimensional (reta) e um espaço paramétrico bidimensional (quadrado).

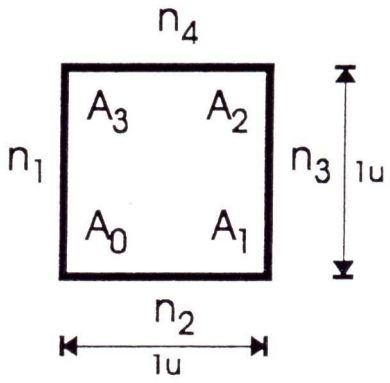


Figura 24. Espaço paramétrico Bidimensional

As variáveis n_1, n_2, n_3 e n_4 são números inteiros positivos diferentes de zero. Eles representam o número de elementos a serem gerados em cada uma das arestas do quadrado unitário. A somatória dos números de elementos por aresta deve ser um número par.

As variáveis A_0, A_1, A_2 e A_3 representam o tipo de ângulo formado pelos quadriláteros da malha permitido em cada um dos vértices do quadrado unitário. Os ângulos permitidos são: ortogonal e bissetriz.

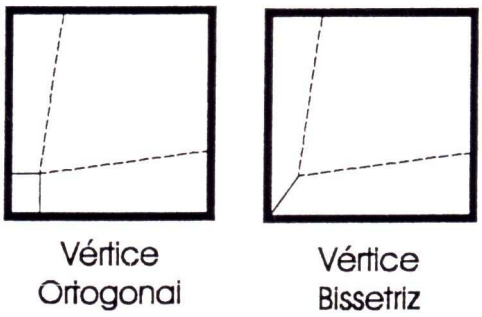


Figura 25. Restrições de ângulos para geração de malha topológica

Baseando-se nas restrições acima, obteve-se o conjunto de operações possíveis de se realizar no quadrado paramétrico.

Este conjunto de operações é ilustrado pela tabela a seguir:

Operação	
Operação "Tx" (Tabuleiro de Xadrez)	
Operação "L"	L0 L90 L180 L270
Operação "U"	U0 U90 U180 U270
Operação "LL"	LL0 LL90
Operação "LU"	LU0 LU90 LU180 LU270
Operação "UU"	UU0
Operação "LR"	LR0 LR90 LR180 LR270

Tabela 1. Operações do quadro paramétrico

Outro fato que mostrou-se importante foi perceber que o processo de geração topológica de malhas em espaços paramétricos bidimensionais é inerentemente recursivo. A cada operação realizada sobre o quadrado paramétrico gera um determinado número de elementos e um novo quadrilátero que pode ser tratado recursivamente como um novo quadrado paramétrico, como mostra a figura a seguir:

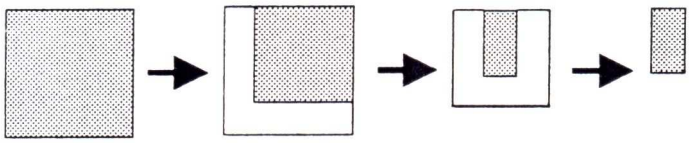


Figura 26. Geração de malha recursiva

Em primeira etapa buscou-se as regras que determinavam a geração topológica em quadrados paramétricos sem restrições de ângulos. Obteve-se a tabela abaixo:

Condição	Operação a ser realizada
Se ($n_1 = n_3$) e ($n_2 = n_4$)	Tx (n_1 x n_2 elementos)
Se ($n_1 > n_3$) e ($n_2 = n_4$)	U90
Se ($n_1 < n_3$) e ($n_2 = n_4$)	U270
Se ($n_1 = n_3$) e ($n_2 > n_4$)	U180
Se ($n_1 > n_3$) e ($n_2 > n_4$)	L180
Se ($n_1 < n_3$) e ($n_2 > n_4$)	L270
Se ($n_1 = n_3$) e ($n_2 < n_4$)	U0
Se ($n_1 = n_3$) e ($n_2 = n_4$)	L90
Se ($n_1 < n_3$) e ($n_2 < n_4$)	L0

Para ilustrar a atualização desta tabela tomemos como exemplo a geração de malhas no espaço paramétrico bidimensional com as seguintes condições: $n_1 = 5$, $n_2 = 4$, $n_3 = 8$ e $n_4 = 5$.

Em primeiro lugar constatou-se que a soma dos números de elementos a serem gerados por aresta é par, pois 5+4+8+5=22 que é par.

Temos então que $n_1 < n_3$ e $n_2 < n_4$, pela tabela a 1{ operação a ser realizada é L0. A figura abaixo ilustra esta operação.

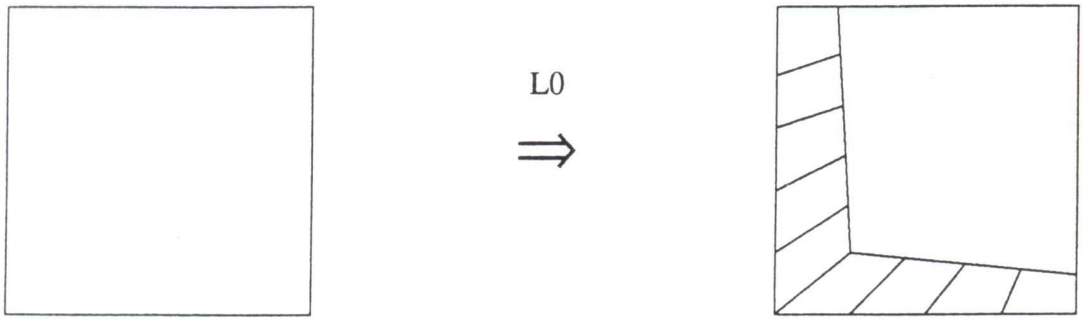


Figura 27. Operação L0

Como resultado desta operação obteve-se duas áreas distintas. Na primeira tem a malha gerada. Sobre a segunda aplicou "recursivamente" as regras da tabela sobre os novos valores, pois como pode-se observar, houve uma diminuição de ordem em n_3 e n_4 que passaram a ser respectivamente iguais a 7 e 4. Com novos valores temos $n_1 < n_3$ e $n_2 = n_4$ e portanto aplicou-se a operação U270. A figura 32 ilustra isso.

Figura 28. Operação U270

Novamente aplicou-se as regras da tabela com o novo valor de $n_3 = 5$, ou seja $n_1 = n_3$ e $n_2 = n_4$. Tendo, portanto, que a operação e Tx 5 x 4. A figura 7 mostra esta operação (operação final):

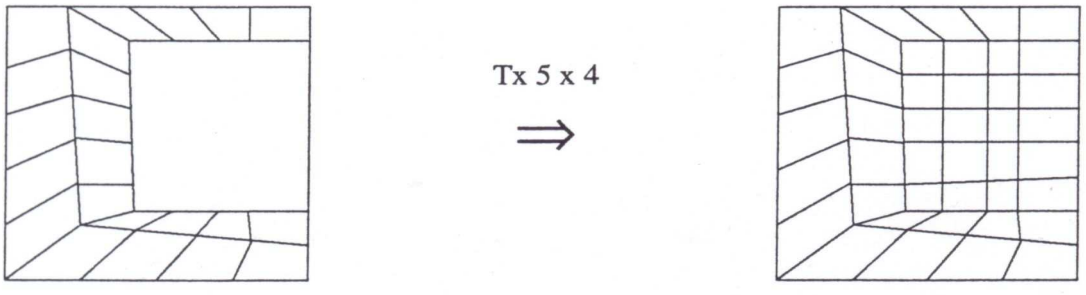


Figura 29. Operação Tx

Considerações Finais

O gerador de malhas é caracterizado pela necessidade da descrição geométrica, onde será gerada a malha, através de regiões parciais segundo regras rígidas. Cada uma dessas regiões é então preenchida pelo gerador de malhas com uma malha local. Durante a definição dessas regiões, também chamada de prediscretização, o usuário precisa considerar uma série de regras e restrições. Quanto ao posicionamento relativo das diversas regiões, deve-se observar que superfícies devem contactar-se completamente através de suas curvas e volume, através de suas superfícies de contorno. Tanto contacto parcial, como intersecção de superfícies ou de volumes não são permitidos, exigindo discretização extras, ou seja, criação de novas regiões.

Enfim pode-se observar que o Software desenvolvido receberá informações do arquivo de origem do Software de Elementos Finitos e então gerará a malha das formas geométricas, sendo assim avaliados os resultados, verificando a qualidade da malha, quando um quadrilátero aproxima de um quadrado e quando hexaedro aproxima de um cubo e as suas características.

Referências

BÉZIER, P. **Emploi des Machines a Commande Numerique.** Paris: Editora Masson, 1970.

BRONSTEIN, I. N.; SEMENDJAJEW, D. A. **Taschenbuch der Mathematik**. Leipzig: Teubner, 1981

CAI, J.; TSAI, H. M.; LIU, F. A parallel viscous flow solver on multi-block overset grids. **Computer & Fluids**. v. 35, p.1290-1301

CHOI, B. K. **Surface Modeling for CAD/CAM**. Elsevier, 1991

EDELSBRUNNER, H. **Geometry and Topology for Mesh Generation.** Cambridge: Cambridge University Press, 2001.

FARIN, G. **Curves and Surfaces for Computer Aided Geometric Design**. 2. ed. Arizona: Academic Press, 1990.

FARIN, G. **NURBS from Projective Geometry to Practical Use.** 2. ed. Arizona: AK Peters, 1999.

FOLEY, J. D. et. al. **Introduction to Computer Graphics**. USA: Addison-Wesley, 1993.

_____. **Computer Graphics principles and practice.** 2. ed. USA: Addison-Wesley, 1996.

GEORGE. P. L. **Automatic Mesh Generation:** Application to Finite Element Methods. Paris: Wiley, 1991.

GLEICHER, M. **A Curve Tutorial for Introductory Computer Graphics,** Department of Computer Sciences, University of Wiscosin, Madison, p. 1-25, Oct. 2004. Disponível em: <http://www.cs.wisc.edu/graphics/Courses/559-2004/docs/cs559-splines.pdf>. Acesso em: 17 Jan. 2009.

GORDON, W.J.; HALL, C.A. Construction of curvilinear co-ordinate systems and applications to mesh generation. **Numerische Mathematik**, v. 7, p. 461-477, 1973.
_____. Tranfinite element methods, blending-function interpolation over arbitrary curved element domains. **Numerische Mathematik**, v. 21, p.109-129,1973.
HABER, R.; SHEPHARD, M. S.; ABEL, J. F.; GALLAGHER, R. H.; GREENBERG, D. P. A general two-dimensional, graphical finite element preprocessor utilizing discrete transfinite mappings. **Int J Num Meth Eng**, v. 17, p. 1015-1044, 1971.
KNUPP, P., STEINBERG, S. **Fundamentals of Grid Generation.** USA: CRC Press, 1994.
MORTENSON, M. E. **Geometric Modeling.** Canada: John Wiley & Sons, 1985.
MYERS, R. H., MONTGOMERY, D. C. **Response Surface Methodology**: Process and Product Optimization Using Designed Experiments. Canadá: Wiley Series in Probability and Statistics, 1995.
O'ROURKE, J. **Computational Geometry in C.** 2. ed. Cambridge: Cambridge University Press, 1998.
PARENTE, E. Jr. **Análise de Sensibilidade e Otimização de Estruturas Geometricamente Não-Lineares**. 2000. Tese (Doutorado em Engenharia Civil) - Departamento de Engenharia Civil da PUC-Rio, Rio de Janeiro. 2000.
PIEGL, L., TILLER, W. **The NURBS book**. 2. ed. New York: Springer, 1997.
SHEPHARD, M. S. The finite element modelling process – will it be automated? **New and future developments in comercial finite element methods**, p. 451-468, 1981.
STOLARSKI, T., NAKASONE, Y., YOSHIMOTO, S. **Engineering Analysis with ANSYS Software.** USA: Elsevier BH, 2006.
SU, Y.; LEE, K. H.; KUMAR, A. S. Automatic mesh generation and modification techniques for mixed quadrilateral and hexahedral element meshes of non-manifold models**. Computer-Aided Design**, v. 36, p. 581-594, 2004.
TAN, S. T., LEE, C. K. Inversed Rational B-Spline for Interpolation. **Computers & Structures.** USA, v. 43, n. 5, p. 889-895, 1992.
THOMPSON, J. F., WARSI, Z. U. A., MASTIN, C. W. **Numerical Grid Generation.** New York: Noth-Holland, 1985.
TILLER, W. Rational B-Splines for Curve and Surface Representation. **Structural Dynamics Research Corporation.** USA, p. 61-69, sep. 1983.
_____. et. al. **Handbook of Grid Generation. Parallel Multibock Structured Grids**, CRC Press, 1999
TSAY, D. M., HUEY, C. O. JR. Application of Rational B-Splines to the Synthesis of Cam-Follower Motion Programs. **Journal of Mechanical Design.** USA, v. 115. p. 621-626, sep. 1993.
YAMAGUCHI, F. **Curves and Surfaces in Computer Aided Geometric Design**. Berlin: Springer-Verlag, 1988.
WALZ, J. E.; FULTON, R. E.; CYRUS, N. J.. Accuracy and convergence of finite element apporximations. **Proc sec conf matrix Meth in mech**, Wright Patterson AFB, Ohio, 1969, p. 995-1028.
ZIENKIEWICZ, O. C. **The Finite Element Method in Engineering Science.** London: McGraw-Hill, 1971.