UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE CIÊNCIAS EXATAS E DA NATUREZA

PÓS-GRADUAÇÃO EM MATEMÁTICA COMPUTACIONAL

INTERPOLAÇÃO RESTRITA USANDO TETRAEDROS QUÁRTICOS DE BÉZIER

MIRELE MOUTINHO LIMA

Tese de Doutorado

Recife
31 de agosto de 2009

UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE CIÊNCIAS EXATAS E DA NATUREZA

MIRELE MOUTINHO LIMA

INTERPOLAÇÃO RESTRITA USANDO TETRAEDROS QUÁRTICOS DE BÉZIER

Trabalho apresentado ao Programa de PÓS-GRADUAÇÃO EM MATEMÁTICA COMPUTACIONAL do CENTRO DE CIÊNCIAS EXATAS E DA NATUREZA da UNIVERSIDADE FEDERAL DE PERNAMBUCO como requisito parcial para obtenção do grau de Doutor em MATEMÁTICA COMPUTACIONAL.

Orientador: *Silvio de Barros Melo*

Recife
31 de agosto de 2009

Lima, Mirele Moutinho
Interpolação restrita usando tetraedros quárticos de Bézier / Mirele Moutinho Lima. - Recife: O Autor, 2009.
xvi, 161 p. : il., fig., tab.

Tese (doutorado) – Universidade Federal de Pernambuco. CCEN. Matemática Computacional, 2009.

Inclui bibliografia e apêndice.

1. Interpolação. 2. Tetraedros quárticos de Bézier. 3. Tetrangulação. I. Título.

511.42 CDD (22. ed.) MEI2010 – 048

UFPE 60 ANOS

UNIVERSIDADE FEDERAL DE PERNAMBUCO
Centro de Ciências Exatas e da Natureza
Doutorado em Matemática Computacional

PARECER DA BANCA EXAMINADORA DE DEFESA DE TESE DE DOUTORADO

MIRELE MOUTINHO LIMA

"INTERPOLAÇÃO RESTRITA USANDO TETRAEDROS QUÁRTICOS DE BÉZIER"

A Banca Examinadora composta pelos professores SÍLVIO DE BARROS MELO (Presidente), CÉSAR AUGUSTO RODRIGUES CASTILHO e MARCÍLIA ANDRADE CAMPOS, todos da Universidade Federal de Pernambuco; e BORKO DARKO STOSIC e CLÁUDIO TADEU CRISTINO, ambos da Universidade Federal Rural de Pernambuco; considera a Tese da candidata:

(X) APROVADA ( ) APROVADA EM EXIGÊNCIA ( ) REPROVADA

Secretaria do Programa de Doutorado em Matemática Computacional do Centro de Ciências Exatas e da Natureza da Universidade Federal de Pernambuco, aos 31 dias do mês de agosto de 2009.

PROF. SÍLVIO DE BARROS MELO
PRESIDENTE E 1º EXAMINADOR

PROF. CÉSAR AUGUSTO RODRIGUES CASTILHO
2º EXAMINADOR

PROFA. MARCÍLIA ANDRADE CAMPOS
3º EXAMINADOR

PROF. BORKO DARKO STOSIC
4º EXAMINADOR

PROF. CLÁUDIO TADEU CRISTINO
5º EXAMINADOR

Av. Prof. Luis Barros Freire, s/nº, Cidade Universitária - CEP: 50670-901 - Recife - PE
FONE/FAX: (81) 2126-8412 - E-mail:ppgmc@ufpe.br - Home Page: www.ppgmc.ufpe.br

A minha mãe,
Teresinha Moutinho

Agradecimentos

A DEUS, por permitir que meus sonhos se realizem.

A minha mãe Teresinha que sempre me apoiou e esteve ao meu lado todos os anos da minha vida; que tanto se sacrificou para que eu pudesse ter uma boa formação e que sempre deu força quando os obstáculos pareciam enormes, vibrou a cada vitória e, acima de tudo, me deu uma mão quando as minhas duas já não eram suficientes. Te amo, mamãe.

As minhas queridas irmãs, Karina e Jamile Moutinho, pela solidariedade, união e respeito. Pelos encontros de família, as conversas, os planos pra vencer as dificuldades. Por tudo que passamos juntas e pelo amor que sempre houve entre nós.

A meu marido e companheiro, Tony, por tantos anos trabalhando e estudando juntos, pelo respeito, admiração e cumplicidade. Pela tranquilidade nos meus momentos de nervosismo; pela alegria nos meus momentos de tristeza; pelas palavras firmes quando a fraqueza era tão grande e pelo amor, presente em todos os momentos...

Ao professor Silvio Melo, um verdadeiro orientador, no melhor sentido da palavra, agradeço profundamente por acreditar em mim, no nosso trabalho, por se dedicar e colaborar do início ao fim para que ele fosse realizado. Minha sincera admiração, professor.

Agradeço também aos professores Francisco Cribari pelos ensinamentos, esclarecimentos e disposição sempre que precisei. Ao professor Alejandro Frery, uma excelente pessoa e com quem comecei todo esse trabalho. Ao professor Manoel Lemos, um mestre em ensinar. Ao professor Klaus Vasconcelos, pelas ótimas aulas de probabilidade. Ao professor César Castilho, pela perspectiva da análise numérica e ao professor Sóstenes Lins por sua sabedoria.

Agradeço à minha grande amiga e colega, Graça, com quem tive inúmeros encontros de estudo, tornando-os produtivos, divertidos e incomparáveis.

A Glória, por sua tranquilidade e sabedoria constantes, obrigada.

Obrigada Alex, Calitéia, Liliam, Lauro, Ademakson, Marta Miranda, companheiros de dificuldades, mas que valorizaram esse momento.

Agradeço aos nossos secretários Esaú e José Carlos e especialmente Valéria, querida funcionária, pelo cuidado e dedicação, todos sempre dispostos a esclarecer nossas dúvidas e resolver nossos problemas administrativos.

A minha tia, prima, amiga, irmã, Eveline, que está tão distante fisicamente, mas não sai do meu coração, e que acompanha e vibra com cada vitória, obrigada.

Um agradecimento especial à família Moutinho, primeiramente à minha avó Marina (*in memorian*), aos meus primos e tios, companheiros de tantas alegrias, e especialmente a Marleide, Osmar, Mário, Inês (*in memorian*), Ivonete, Agostinho (*in memorian*), Múcio e Lúcia, que nos receberam com carinho e nos deram apoio na etapa mais difícil na vida da minha família. Minha eterna gratidão a vocês, meus queridos.

Ao meu cunhado Rômulo, por entrar em nossas vidas, cuidar tão bem de nossa família e nos presentear com Julia, uma linda menina que completa esse trio tão amado.

A minha cunhada, Ana Abranches, que não desiste nunca, e a minha sogra, Gilva Ribeiro, a ambas sou grata pelo maior presente que me deram e a felicidade que me trazem sempre.

A meus amigos Mery, Arthur, Nilza, Poly, Lenylda, Karine, Eijiro, Rogério e Maura, que sempre torceram por mim e que dão um elevado valor à palavra amizade. Amo vocês, meus queridos.

Ao meu amigo e colega Luciano Barboza, a quem admiro, pelas divertidas e filosóficas conversas.

Aos meus colegas da Universidade de Pernambuco, ao professor Diretor Pedro Falcão, que merece nossa gratidão e admiração, e à professora Adelina. Em especial, agradeço a Raimundo Praxedes por tamanha dedicação e disposição pra ajudar em meio a tantas obrigações.

Aos meus queridos alunos, pelo respeito e estímulo que sempre me deram, por tornarem tão ricos nossos encontros e transformarem em prazer o meu trabalho, ao qual tanto me dedico.

Ao meu filho Thales, o meu maior amor.

"A leitura torna o homem completo; a conversação torna-o ágil; e o escrever dá-lhe precisão."

—FRANCIS BACON

Resumo

O problema de interpolação de dados espalhados trivariados e não-negativos consiste em construir uma função contínua de três variáveis independentes, não-negativa, partindo de alguns dados conhecidos, irregularmente distribuídos.

Muito se tem feito para dados bivariados e regulares, mas pouco para interpolação de dados trivariados espalhados e não-negativos, objetivo desse estudo. No entanto a necessidade de interpolação com essas características ocorre em muitas áreas diferentes do mundo real. Da medicina à economia, da engenharia à oceanografia, onde os dados são dispostos de forma aleatória, a interpolação de pontos irregularmente espaçados e trivariados é fundamental. Por exemplo, em meteorologia, medições meteorológicas estão disponíveis a partir de observações de estações posicionadas irregularmente.

Este trabalho apresenta a construção de uma interpolante C^1 trivariada de pontos espalhados, a qual é não negativa em todo lugar desde que os pontos a serem interpolados sejam não negativos. Cada tetraedro num domínio tetrangulado é dividido em quatro mini-tetraedros e a superfície interpolante sobre cada um deles é um tetraedro quártico de Bézier. Condições suficientes são derivadas para a não-negatividade desses tetraedros quárticos e elas são expressas como limites inferiores das ordenadas de controle de Bézier.

Alguns exemplos gráficos são ilustrados e podemos verificar a eficiência do algoritmo proposto, pela sua localidade, evitando a dependência de dados distantes do interpolado, pela sua fácil implementação e finalmente, por atingir rapidamente o objetivo sugerido, uma superfície C^1 e não-negativa.

Palavras-chave: Pontos espalhados, tetrangulação, interpolação, tetraedros quárticos de Bézier, preservando a positividade.

Abstract

The problem of interpolating trivariate non-negative scattered data consists of a construction of continuous function of three independent variables, non-negative, from some irregularly distributed known data.

Much has been done for regular and bivariate data, but little for interpolation of non-negative trivariate scattered data, objective of this study. Nevertheless the need for interpolation with these characteristics occurs in many different areas in the real world. From economics to medicine, from engineering to oceanography, where data are arranged spatially at random, the interpolation of irregularly spaced points is essential. For example, in meteorology, meteorological measurements are available from readings of sites in which data are general positioned irregularly.

This research presents the construction of a C^1 trivariate interpolant of scattered data, which is non negative at every point provided that the interpolating data points are non-negative. Each tetrahedron in a tetrangulated convex hull of data points is divided into four mini-tetrahedra and the interpolant surface on each one of them is a quartic Bezier tetrahedron. Sufficient conditions are derived for the non-negativity of quartic tetrahedra and they are expressed as lower limits for the Bézier control ordinates.

Experimental results, some of them graphically illustrated, are presented, in order to adress the algorithm's accuracy and locality, avoiding dependence on data points far from the point of interest. It also presents the advantages of a first order smoothness with the use of relatively low degree polynomials, as well as a low effort programming and finally, by quickly achieving its main purpose: a non-negative C^1 interpolant.

Keywords: scattered data, tetrangulation, interpolation, quartic Bézier tetrahedra, positivity preserving.

Sumário

Lista de Figuras

Lista de Símbolos

Na tabela abaixo estão listadas as notações utilizadas no texto, seguidas das páginas onde elas são introduzidas.

Símbolo	Definição	Página
$\mathbf{b}_i$	Ponto de controle	13,14
$\mathbf{b}_i^r(t)$	Ponto de controle gerado pelo algoritmo de Casteljau	13,14
$\mathbf{b}^n$	Curva de Bézier de grau n	13
$\mathbf{b}^2$	Curva de Bézier (quadrática)	13,14
$B_i^n(t)$	Polinômio de Bernstein	15
$\mathbf{b}_{\mathbf{i}}^r(t)$	Ponto de controle numa malha triangular	19
$\mathbf{b}^n$	Triângulo de Bézier de grau n	20
$\mathbf{b}^3$	Superfície de Bézier (cúbica)	21
$D_{\mathbf{d}}\mathbf{b}(\mathbf{u})$	Derivada direcional sobre uma malha de Bézier, na direção do vetor $\mathbf{d}$	22
b^3	Superfície cúbica de Bézier	24
b_{ijk}	Valor de controle (ordenada de Bézier) de b^3	25
$\mathbf{b}_{ijk}$	Ponto de controle sobre b^3	28
b^4	Superfície quártica de Bézier	41
b_{ijkp}	Valor de controle (ordenada de Bézier) de b^4	42
$\mathbf{b}_{ijkp}$	Ponto de controle sobre b^4	45

Capítulo 1

Introdução

Suponha que para uma determinada função f, a única informação que temos dela se trata dos n pontos do seu gráfico $(x_i, y_i, z_i, f(x_i, y_i, z_i))$, sendo $i = 1, \cdots n$, e necessitemos descobrir o valor de $f(p)$, para algum p que está no invólucro convexo dos pontos (x_i, y_i, z_i).

Dá-se o nome de *interpolação* ao processo de determinação desse valor por um funcional $F : R^3 \to R$ contínuo, tal que:

$$F(x_i, y_i, z_i) = f(x_i, y_i, z_i), \text{ para todo } i \in \{1, \cdots n\}. \tag{1.1}$$

Em suma, interpolação é o processo de modelagem de um dado fenômeno via uma função contínua, a partir de sua amostragem num número finito de pontos, de forma a reproduzir exatamente esses valores amostrados. Uma *aproximação* é o processo de modelagem de um dado fenômeno via uma função contínua, a partir de sua amostragem num número finito de pontos, de forma que os valores amostrados sejam reproduzidos a menos de uma tolerância. Em alguns artigos na literatura aplicada o processo de interpolação é visto como um caso particular do processo de aproximação.

Quando a amostragem dos dados é sabidamente livre de ruídos, então o processo de interpolação é mais recomendado e utilizado. Os processos de interpolação são em geral mais caros e apresentam geometria mais complexa que os de aproximação. Estes últimos são mais robustos e baratos, e são mais utilizados quando há tolerância a imprecisões na amostragem.

Neste trabalho focalizamos em um processo de interpolação por partes que satisfaz alguns critérios importantes num certo conjunto de aplicações: a suavidade C^1 e a não-negatividade.

Uma característica dos pontos a serem interpolados, (x_i, y_i, z_i) para todo $i \in \{1, \cdots n\}$, é que eles podem estar dispostos numa grade, igualmente espaçados. Mas certamente uma configuração mais comum é esses pontos estarem dispersos, ou melhor, espalhados no espaço. E essa é uma outra característica dos dados utilizados nesse trabalho.

Sendo assim, suponha que tenhamos diversos pontos espalhados no espaço tridimensional e desejamos interpolá-los, através de uma função suave. Métodos variados são analisados com o objetivo de determinar essa função e de forma que ela satisfaça alguns requisitos importantes.

1.1 Modelando Pontos Espalhados

Em 1993, Gregory M. Nielson escreveu sobre vários métodos de modelagem $3D$ [8]. A seguir narraremos alguns aspectos de seu trabalho.

Considere amostras na forma $(x_i, y_i, z_i; F_i)$, $i = 1, \cdots, N$ onde $P_i = (x_i, y_i, z_i)$ representa a variável independente (pontos espalhados ou nós de grade) e F_i é a variável dependente. Esse

tipo de dado surge frequentemente em estudos científicos. Vejamos alguns exemplos:

1. Temperaturas medidas em diversas posições de um forno.
2. Concentrações minerais conhecidas em vários níveis de profundidade de buracos abertos espalhados.
3. Densidades medidas em várias posições dentro de um corpo humano.
4. Valores de pressão calculados e medidos em diversos pontos de uma superfície.
5. Precipitações medidas em várias estações de tempo.
6. Eletroencefalogramas medidos por eletrodos presos a um couro cabeludo.

Para analisar ou visualizar as relações implicadas pelos dados podemos obter uma função de modelagem matemática, $F(x,y,z)$ tal que $F(x_i,y_i,z_i)$ coincide ou se aproxima de F_i. Nesse estudo Nielson discutiu métodos que levam a interpolações, $F(x,y,z)$ que têm, no mínimo, a continuidade da primeira derivada (F é C^1). Enquanto todos os exemplos citados acima têm amostras de dados com a mesma representação, existem diferenças fundamentais entre eles. Há casos onde o domínio é uma região $3D$ e outros onde o domínio é uma região $2D$ que é restrição de um espaço $3D$. O primeiro é dito *pontos volumétricos espalhados* e os métodos citados abaixo configuram nesse tipo.

1.1.1 Pontos Volumétricos Espalhados

Os métodos descritos em [8] são considerados como representantes de todos os métodos de interpolação de dados trivariados espalhados, até então.

Começamos com o *Método de Shepard Básico*, uma aproximação da distância inversa ponderada que é fácil de descrever e implementar. Podemos escrevê-lo da forma

$$S(P) = \frac{\displaystyle\sum_{i=1}^{N} \frac{F_i}{\|P-P_i\|^2}}{\displaystyle\sum_{i=1}^{N} \frac{1}{\|P-P_i\|^2}}, \text{ se } P \neq P_i,$$

onde $\| P-P_i \|$ representa a distância de P a P_i, mostrando que ele depende de todos os pontos dados. Embora simples, este método tem algumas deficiências que o inviabilizam em aplicações práticas. Sua principal deficiência é que ele não reproduz nenhuma propriedade local porque ele tipicamente tem um extremo em cada ponto dado (*bull's-eye*).

No caso dos dados bivariados, Franke e Nielson [8] desenvolveram uma modificação que elimina a deficiência do método básico de Shepard. Este outro método é chamado *Método de Shepard Quadrático Modificado* (MQS). Foi modificada a função distância $\| P-P_i \|$ e F_i foi trocada pela aproximação local conveniente $Q_i(x,y)$. Este método tem a forma geral

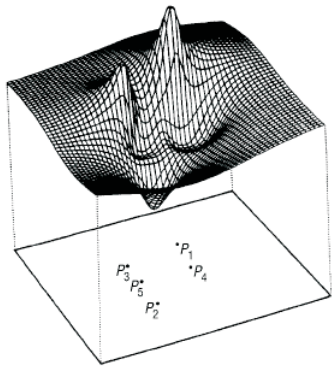

Figura 1.1 Método de Shepard Básico Bivariado: pontos críticos em P_i.

$$Q(P) = \frac{\sum_{i=1}^{N} \frac{Q_i(P)}{\rho_i^2(P)}}{\sum_{i=1}^{N} \frac{1}{\rho_i^2(P)}}$$

onde

$$\frac{1}{\rho_i(P)} = \frac{(R_w - \|P - P_i\|)_+}{R_w \|P - P_i\|},$$

para alguma constante R_w. Aqui o subscrito + denota a função potência truncada, onde o peso é zero em distâncias do ponto dado maiores que R_w. Toma-se $Q_i(P)$ como polinômios quadráticos, obtidos através da aplicação do método dos mínimos quadrados ponderado e forçado a interpolar os valores de F_i em P_i. Os pesos no processo dos mínimos quadrados são da mesma forma que a função peso da interpolante, mas no lugar de R_w um outro valor, R_q. As idéias do método MQS bivariado se estendem diretamente para o caso trivariado. Uma descrição formal do método para o caso trivariado consiste de primeiramente selecionar N_q e N_w para definir

$$\frac{1}{\rho_i(P)} = \frac{(R_w - \|P - P_i\|)_+}{R_w \|P - P_i\|}, \quad R_w = \frac{D}{2}\sqrt{\frac{N_w}{N}}$$

$$\frac{1}{\mu_i(P)} = \frac{(R_q - \|P - P_i\|)_+}{R_q \|P - P_i\|}, \quad R_q = \frac{D}{2}\sqrt{\frac{N_q}{N}}$$

onde $D = max_{i,j} \| P_i - P_j \|$ e os valores padrão de N_q e N_w são 54 e 27, respectivamente.

O próximo método é o *Spline de Distância Volumétrico*. A forma da sua função de modelagem é

$$V(P) = \sum_{i=1}^{N} c_i \|P - P_i\|^3 + a + bx + cy + dz.$$

São obtidos os coeficientes desconhecidos a partir do seguinte sistema de equações, que representa os requisitos para a interpolação e as limitações análogas para condições de contorno

$$\begin{pmatrix} & & & & 1 & P_1 \\ & \|P_i - P_j\|^3 & & & 1 & P_2 \\ & & & & & \\ . & . & . & . & . & . \\ & & & & & \\ & & & & 1 & P_N \\ 1 & 1 & & 1 & 0 & 0 \\ P'_1 & P'_2 & & P'_N & 0 & 0 \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ . \\ c_N \\ a \\ b \\ c \\ d \end{pmatrix} = \begin{pmatrix} F_1 \\ F_2 \\ . \\ F_N \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

Para este método ser implementado é necessário apenas uma rotina para resolver um sistema linear de equações. Contanto que $\mathbf{P}_i, i = 1 \cdots N$, sejam distintos, a matriz de coeficientes acima é não-singular e, portanto, o sistema pode ser resolvido. No entanto, resolver um sistema de equações lineares é um processo lento, e dificilmente é possível obter resultados para grandes conjuntos de dados em questão de minutos.

O próximo método, o *Multiquádrico*, é similar ao Spline de Distância Volumétrico. Sua função de modelagem é

$$H(P) = \sum_{i=1}^{N} c_i \sqrt{R^2 + \|P - P_i\|^2}.$$

Os requisitos da interpolação levam ao sistema de equações

$$\left(\sqrt{R^2 + \|P_i - P_j\|^2} \right) \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_N \end{pmatrix} = \begin{pmatrix} F_1 \\ F_2 \\ \vdots \\ F_N \end{pmatrix}.$$

O próximo método é consideravelmente mais complicado de descrever que os anteriores. Ele é a generalização do método da *Rede de Norma Mínima* (MNN) que interpola dados bivariados espalhados. Vejamos agora rapidamente como este método funciona e então o estenderemos para o caso de dados volumétricos. Temos três passos:

1. Triangularizar a região do invólucro convexo definida pelos pontos $P_i(x_i, y_i)$.

2. Determinar uma curva interpolante definida sobre as arestas que tenha certas propriedades de minimização.

3. Preencher a superfície triangular cujos bordos são segmenos da curva de rede a fim de completar a definição de função modeladora pelo uso de uma interpolante triangular C^1.

Uma triangulação do invólucro convexo do conjunto de pontos $P_i = (x_i, y_i)$, $i = 1, \cdots N$, consiste de um trio $(i, j, k) \in N_t$. Cada trio (i, j, k) representa um triângulo T_{ijk} com vértices P_i, P_j, P_k. Assuma que dois a dois esses triângulos se intersectam apenas nas arestas e que

a união de todos os triângulos é o invólucro convexo (ou seja, uma partição). Existem muitas triangulações possíveis para um invólucro convexo. Normalmente é preferível um tipo de triangulação ótima que não possua "triângulos longos e finos". Uma escolha comum é a triangulação *max-min*, cujo ângulo mínimo é o maior possível.

Para o segundo passo deste método, no caso bivariado do MNN uma rede é definida sobre a coleção de todas as arestas. Esta rede é caracterizada de maneira similar à norma mínima padrão para splines cúbicas interpolantes. Observe que uma spline cúbica é o único mínimo de

$$\int_a^b [F''(x)]^2 \, dx,$$

sujeito a requisitos de interpolação.

No caso de MNN,

$$\sigma(F) = \sum_{ij \in N_e} \int_{e_{ij}} \left(\frac{d^2 F}{d^2 e_{ij}} \right)^2 ds_{ij}$$

é minimizado, onde ds_{ij} representa o elemento do comprimento de arco no segmento de reta e_{ij} e

$$N_e = \{ij \text{ ou } ji \text{ (mas não ambos)} : V_i \text{ para } V_j \text{ é uma aresta da triangulação}\}.$$

Estas idéias se estendem para o caso de dados volumétricos. Minimiza-se uma quantidade similar a $\sigma(F)$, sendo que agora N_e é a coleção de todas as arestas das faces triangulares da decomposição tetraédrica do invólucro convexo. Uma vez tendo determinado a rede de curvas, é possível seguir para o terceiro passo, preenchendo o modelo com um interpolante C^1 tetraédrico. Define-se um interpolante em cada tetraedro que combinará a posição dada e a informação derivada em todas as arestas.

Chegamos ao último método, chamado de *Splines de Volume Local*. Como dito anteriormente, existem limites definidos para o tamanho do conjunto de dados para splines volumétricas, uma vez que sendo global, todos os pontos influenciam na interpolação, prejudicando assim seu resultado. Frequentemente os problemas envolvendo dados volumétricos espalhados excedem estes limites.

Para construir um método particular, útil e aplicável para um grande conjunto de dados, podemos torná-lo local. Isto significa tornar a função F_i local e, dessa forma, apenas pontos próximos são utilizados na interpolação, aumentando sua exatidão. Podemos facilmente entender a idéia básica, considerando o caso univariado (veja Figura 1.2). As funções w_k são não nulas apenas em dois intervalos e têm a propriedade que $\sum w_k(x) = 1$ para qualquer x do domínio. Determinamos a função interpolante local de pontos espalhados F_k, tal que $F_k(x_i) = F_i$ para todo conjunto de pontos x_i no suporte (região não nula) de w_k. Disso, podemos ver que

$$R(x) = \sum w_k(x) F_k(x)$$

tem a propriedade que $R(x_i) = F_i$ para todo x_i na união dos suportes de w_k, que serve como domínio da aproximação. Em geral, podemos usar qualquer método para obter a interpolante local

F_k, mas para esse método particular usamos a spline de volume citada anteriormente. Podemos facilmente construir uma função local w_k com pedaços cúbicos (bicúbicos ou tricúbicos).

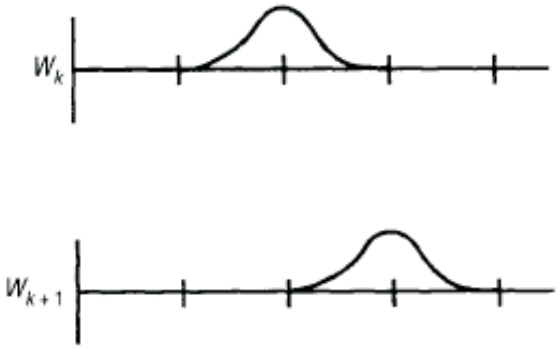

Figura 1.2 Função de peso para tornar local o interpolante.

1.1.2 Comparações

Dado um conjunto de pontos independentes $(x_i, y_i, z_i), i = 1, \cdots, N$ e uma função de teste $F(x, y, z)$, é usado um método particular de interpolação de pontos espalhados para produzir uma aproximação $A(x, y, z)$, que é comparada a $F(x, y, z)$. Essas comparações consistem em avaliações estatísticas e subjetivas baseadas em análises gráficas de F e A.

Dessa forma, as seguintes conclusões foram tiradas com respeito aos métodos citados anteriormente.

Segundo [8], o método de Shepard Quadrático Modificado, que é C^1, é uma extensão de um método bivariado bem conhecido e eficaz, para dados volumétricos. Ele "normalmente reproduz muito bem as características qualitativas da função de teste"; tais características se referem às da forma gráfica, que se inferem após uma inspeção visual. É preciso ressaltar a dependência desta afirmação com as funções de teste escolhidas, bem como da granularidade da tetrangulação do invólucro convexo. Perto da fronteira o método tende a gerar oscilações não implicadas pelos dados. Podemos aplicá-lo a conjuntos bem grandes de pontos (acima de 1000) e ele está entre os mais rápidos dos métodos testados. Em geral, a implementação deste método exige mais esforço de programação do que os demais. O usuário deve prover os parâmetros N_q e N_w ou aceitar os valores padrão.

O método de Splines Volumétrica é uma generalização direta da spline de interpolação cúbica univariada para dados volumétricos, quando representada com funções de distância. Na maior parte dos casos ele dá resultados semelhantes aos do multiquádrico, em termos de medida de erros em algumas funções de teste, mas às vezes seu desempenho é inferior. A implementação necessita apenas de uma rotina para resolver um sistema de equações lineares. O método reproduz funções lineares e é C^2. Em geral, e sem modificações, o método é limitado a conjuntos de dados menores do que $N = 500$.

Como no caso bivariado, o método Multiquádrico geralmente reproduz muito bem as carac-

terísticas qualitativas da função de teste. A implementação consiste em resolver um sistema de equações lineares. O condicionamento da matriz de coeficientes limita o tamanho do conjunto de dados usado. Tipicamente, conjuntos de dados de 300 para 500 ou mais pontos comprometerão a exatidão de precisão única. O usuário deve prover o parâmetro R^2.

O método da Rede de Norma Mínima é uma generalização volumétrica de um método bivariado eficiente e popular. A generalização não é como se esperava, e o método é mais difícil para implementar do que a maior parte dos outros métodos. O lado positivo é que ele é um método global que pode ser aplicado a conjuntos muito grandes de pontos. Em geral tem uma boa performance.

Podemos usar o método Splines de Volume Local para conjuntos com mais de 1000 pontos. A localização da função F_i às vezes deteriora o esquema quando comparado ao método global. Às vezes parece ter a mesma performance que o método multiquádrico. O usuário deve prover os valores que particionam o domínio ou aceitar valores padrão uniformemente espaçados. Isto pode ser um problema em que um subdomínio poderia não ter pontos suficientes para determinar uma spline volumétrica. Uma implementação robusta reconheceria esta situação e tomaria uma atitude evasiva e apropriada. Em geral, a implementação é um pouco mais complexa do que o Multiquádrico ou Spline Volumétrica, mas em geral é bastante fácil de usar.

Por fim, o trabalho conclui que não existe um método melhor. Todos têm lado positivo e lado negativo, e alguns são mais importantes que outros, dependendo de onde é aplicado.

1.2 Outros Métodos de Interpolação

A Distância Inversa a uma Certa Potência é um interpolador de média ponderada, que pode ser também exato ou aproximado. Neste método, os dados são ponderados durante a interpolação, de forma que a influência de um ponto, relativo a outro, diminui com a distância do nó de grade. A ponderação é feita através da potência da distância, que controla os fatores de ponderação. Quanto maior for a potência de ponderação, menor é o efeito causado pelos pontos distantes que estão na grade, durante a interpolação. À medida que as potências aumentam, o valor do nó da grade (z) se aproxima do valor do ponto mais próximo. E à medida que as potências diminuem, os pesos são mais igualmente distribuídos ao longo de uma vizinhança de pontos. O método de Shepard é um caso particular do método da Distância Inversa a uma Certa Potência e, como vimos na seção anterior, seu conhecido problema de ocorrência de ponto crítico nos nós de grade também se manifesta nessa classe de métodos

O *Método de Kriging*, muitas vezes traduzido como "Krigagem", é um método de regressão usado em geoestatística para aproximar ou interpolar dados. A teoria de Kriging foi desenvolvida a partir dos trabalhos do seu inventor Daniel G. Krige e pelo matemático francês Georges Matheron, no começo dos anos sessenta. Na comunidade estatística, também é conhecido como "Processo Gaussiano de Regressão"(veja [17]). Kriging pode ser entendido como uma predição linear ou uma forma da Inferência Bayesiana. Ele parte do princípio que pontos próximos no espaço tendem a ter valores mais parecidos do que pontos mais afastados.

Considere, para o cálculo do Kriging, a seguinte fórmula:

$$F(x,y) = \sum_{i=1}^{n} w_i f_i,$$

onde n é o número de amostras obtidas, f_i é o valor obtido no ponto i e w_i é o peso designado ao ponto i. A fim de obter os pesos de cada um dos n pontos, é resolvido um sistema de equações. Tal procedimento depende do tipo de Kriging que está sendo utilizado (Ordinário ou Simples).

Ao obtermos os valores de $w_1, w_2, ..., w_n$, calcula-se o valor de f_p:

$$f_p = w_1 f_1 + w_2 f_2 + ... + w_n f_n.$$

Desta maneira, calcula-se o valor interpolado para todos os pontos desejados.

O método do *Vizinho Natural* é bastante popular em algumas áreas. Ele é um método local. Como é interpolar por esse método? Considere um conjunto de partições do plano/espaço, onde cada uma delas destaca-se um ponto chamado centro, de forma que se um ponto P qualquer do plano/espaço está numa certa partição, então o centro desta partição é o mais próximo de P dentre todos os centros. Essas partições são ditas diagramas de Voronoi (veja Figura 1.3), e para o método aqui analisado, seus centros são os próprios pontos espalhados. Para se determinar o valor de função para um ponto arbitrário, esse ponto será primeiramente inserido como se fosse um centro de diagrama, seguindo o algoritmo de Sibson [28, 7], criando-se um novo diagrama sobreposto aos existentes (Figura 1.4). De fato, alguns desses diagramas iriam encolher no tamanho. A área associada ao novo diagrama e que foi retirada de um já existente é chamada de "área emprestada."O algoritmo de interpolação do Vizinho Natural usa uma média ponderada das observações dos pontos vizinhos, onde os pesos são proporcionais à área emprestada. O método do Vizinho Natural não permite extrapolação do invólucro convexo dos centros.

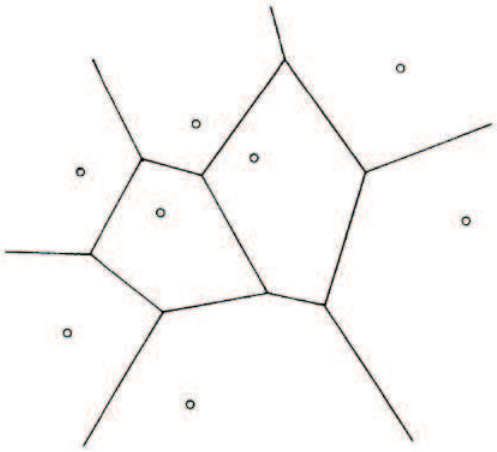

Figura 1.3 Diagramas de Voronoi com os centros sendo os pontos dados.

O método do *Vizinho Mais Próximo* atribui o valor do ponto mais próximo a cada nó da grade. Este método é útil quando os pontos já estão igualmente espaçados. Por outro lado, em casos onde os pontos estão próximos de serem uma grade, com apenas alguns valores

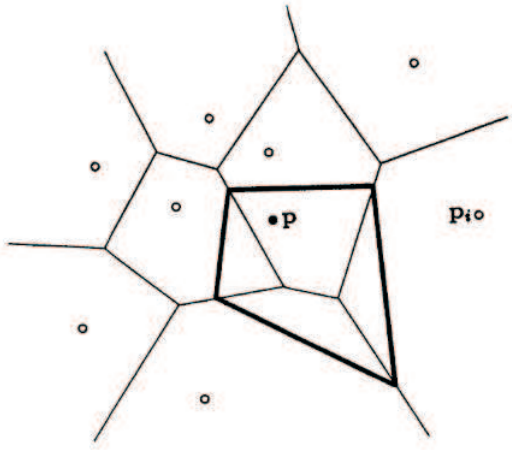

Figura 1.4 Diagramas de Voronoi com o novo centro inserido.

faltando, este método é eficaz para preencher o vazio deixado. Uma vez que ele é um método descontínuo, é considerado de baixa qualidade.

O método da *Triangulação com Interpolação Linear* ou *Interpolação Linear por Partes* em geral usa a triangulação de Delaunay [3]. Este algoritmo cria triângulos desenhando retas entre os pontos dados. Os pontos originais são conectados de tal modo que nenhuma aresta do triângulo é intersectada por outros triângulos. O resultado é uma malha de faces triangulares por cima da extensão da grade. Cada triângulo define um plano por cima da grade de nós que está dentro do triângulo, com o nível de elevação do triângulo determinado por três pontos originais que definem o triângulo. Triangulação com Interpolação Linear trabalha melhor quando os seus pontos são igualmente distribuídos por cima da área da grade. É um método eficiente, mas tem continuidade C^0.

Outros métodos de interpolação podem ser citados, como *Função de Base Radial*, o da *Média Móvel*, o da *Métrica de Dados* e o do *Polinômio Local* [13, 18, 15, 14, 12].

Os métodos de interpolação de dados são muito úteis em diversas áreas. Podemos citar, por exemplo, a Geoestatística como uma área de grande aplicação desse estudo. De fato, os modelos digitais de terreno permitem que se vá além da representação cartográfica e possam ser realizadas análises numéricas de superfícies contínuas.

Há muitos softwares de interpolação, contudo a maioria é projetada para resolver problemas específicos e têm versatilidade limitada. Um software bastante utilizado nessa área é o *Surfer*, particularmente interessado em relevo de terras e oceanos. Sua versão mais nova é a 9.x, de acordo com [16]. Em 2004 Yang *et al* escreveram sobre a versão 8.0 do Surfer e, nesse trabalho, eles apresentam métodos de interpolação, juntamente com aplicações sobre sua eficiência e exatidão [12].

Os métodos citados até agora não garantem a propriedade da não negatividade, ou até mesmo que a superfície resultante seja positiva. Ou seja, se os pontos espalhados fornecidos possuírem valores não-negativos (positivos) a superfície resultante pela interpolação desses da-

dos deve também ser não-negativa (positiva). Há diversas aplicações onde esse resultado é fundamental. Por exemplo, quando os dados de um experimento físico são medidos na forma de concentração ou pressão ou dados meteorológicos como o montante de uma enchente onde os valores negativos não fazem sentido, é importante para o interpolante conservar a positividade. Podemos citar também

1. Medicina: cálculo da densidade mineral óssea [19, 20, 21],
2. Oceanografia: concentração de nutrientes numa dada profundidade [22],
3. Medicina: modelagem de próteses [23],
4. Economia: cálculo da taxa interna de retorno [24],
5. Mecânica dos Sólidos: análise de tensões em estruturas [25].

Claramente os modelos citados acima, cujos dados são interpolados, não fazem sentido para funções de valores negativos. Dessa forma, as seções seguintes estabelecem estudos onde a função interpoladora seja positiva (não-negativa).

1.3 Interpolação G^1 de Pontos Espalhados Preservando a Positividade

Em 2006, Piah, Saaban e Majid elaboraram um estudo sobre a construção de um interpolante de pontos bivariados espalhados, G^1, com valores positivos em todo lugar se os pontos originais forem positivos [9]. Vejamos um pouco desse trabalho.

As propriedades que são frequentemente usadas para quantificar "a forma"em interpolações que preservam a forma são a positividade, a convexidade e a monotonicidade. Este problema pode surgir se os pontos dados estiverem de um lado do plano, e desejar-se ter uma superfície interpolante que está também do mesmo lado deste plano. O trabalho forneceu condições suficientes sobre uma função quártica bivariada após uma triangulação dos pontos, a fim de visualizar os dados espalhados positivos que podem surgir em certos fenômenos científicos.

Dados pontos espalhados (x_i, y_i, z_i), com $z_i \geq 0$ e $i = 1, 2, \cdots, N$, desejamos construir uma superfície $z = F(x, y)$ que interpole esses pontos, que preserve a positividade e seja G^1, isto é, geometricamente semelhante a C^1, mas sem restrições com relação à parametrização que esta impõe. O invólucro convexo desses pontos é triangulado usando o trabalho desenvolvido por Delaunay [3], e as derivadas parciais de primeira ordem de F com respeito a x e a y são calculadas usando [4].

Considere um triângulo T, com vértices V_1, V_2 e V_3, e coordenadas baricêntricas u, v, w tal que qualquer ponto V no triângulo pode ser expresso por

$$V = uV_1 + vV_2 + wV_3, \text{ onde } u + v + w = 1 \text{ e } u, v, w \geq 0.$$

Uma malha triangular quártica P é definida sobre T. Seja ela:

$$\begin{aligned} P(u,v,w) &= b_{400}u^4 + b_{040}v^4 + b_{004}w^4 + 4b_{310}u^3v + 4b_{301}u^3w + 4b_{130}uv^3 + \\ &+ 4b_{031}v^3w + 4b_{103}uw^3 + 4b_{013}vw^3 + 6b_{220}u^2v^2 + 6b_{202}u^2w^2 + \\ &+ 6b_{022}v^2w^2 + 12b_{211}u^2vw + 12b_{121}uv^2w + 12b_{112}uvw^2, \end{aligned} \tag{1.2}$$

onde b_{ijk} são ordenadas de Bézier de P (veja Seção 2.3).

A seguinte proposição estabelece condições para as ordenadas de Bézier.

Proposição 1.1. *Considere a malha triangular quártica de Bézier $P(u,v,w)$ com $b_{400} = A, b_{040} = B, b_{004} = C$, onde $A,B,C > 0$. Se $b_{rst} \geq -r_0$, para $b_{rst} \neq A,B,C$, onde r_0 é a única solução de*

$$\frac{1}{\sqrt[3]{\frac{A}{r}+1}} + \frac{1}{\sqrt[3]{\frac{B}{r}+1}} + \frac{1}{\sqrt[3]{\frac{C}{r}+1}} = 1,$$

então $P(u,v,w) \geq 0, \forall u,v,w \geq 0, u+v+w = 1$.

Essas condições sobre as ordenadas de Bézier são suficientes para assegurar que a superfície que compreende a malha triangular quártica de Bézier seja positiva. Sendo assim, a ordenadas de Bézier são determinadas e, além disso, são satisfeitas as condições de continuidade G^1. As derivadas são calculadas (e modificadas, se necessário) para assegurar que todas as ordenadas de Bézier satisfaçam as condições da proposição acima, a superfície interpolante final seja gerada por (1.2) e o objetivo seja atingido.

1.4 Interpolante de Dados Espalhados C^2 e Restrito

Vários métodos relacionados com visualização de dados de superfície, preservando a positividade e usando funções bivariadas, concentram-se apenas em gerar uma superfície resultante suave e C^1. Tanto quanto se sabe, muito pouco tem-se feito para permitir a visualização de superfícies positivas, partindo de dados espalhados positivos, e que seja C^2. Em 2007 Saaban, Piah e Majid [11] escreveram um trabalho que propõe condições suficientes sobre os pontos de Bézier a fim de garantir que a superfície que compreende a malha triangular quíntica de Bézier seja sempre positiva e satisfaça as condições de continuidade C^2. Vejamos as similaridades com trabalhos anteriores de alguns dos autores e as idéias novas usadas nesse trabalho [11].

Desejamos construir uma superfície $F(x,y)$, C^2 e que preserve a positividade ao interpolar (x_i,y_i,z_i), $i = 1,2,\cdots,N$, onde $z_i \geq 0$. É usada a triangulação de Delaunay [3] para triangularizar o invólucro convexo dos pontos (x_i,y_i), e as derivadas parciais de primeira ordem de F com respeito a x e a y são calculadas usando [4], mas a derivada parcial de segunda ordem será estimada através da aproximação quadrática do método dos mínimos quadrados.

$$min \sum_i \left(ax_i^2 + bx_iy_i + cy_i^2 + dx_i + ey_i + f - z_i\right)^2,$$

onde a,b,c,d e f são os coeficientes a serem determinados.

Considere um triângulo T, com vértices $V_1(x_1, y_1), V_2(x_2, y_2)$ e $V_3(x_3, y_3)$, e coordenadas baricêntricas u, v, w tal que qualquer ponto V no triângulo pode ser expresso por

$$V = uV_1 + vV_2 + wV_3, \text{ onde } u + v + w = 1 \text{ e } u, v, w \geq 0.$$

Uma malha triangular quíntica de Bézier, P, em T é definida como:

$$P(u, v, w) = \sum_{i+j+k=5} b_{ijk} B^5_{ijk}(u, v, w), i \geq 0, j \geq 0, k \geq 0, \tag{1.3}$$

onde $B^5_{ijk}(u, v, w) = \frac{5!}{i!j!k!} u^i v^j w^k$ e b_{ijk} são ordenadas de Bézier de P.

Semelhantemente ao que foi feito em [9] supõe-se que $b_{500}, b_{050}, b_{005}$ são positivos e condições suficientes sobre as ordenadas de Bézier restantes são determinadas a fim de garantir uma malha de Bézier positiva. A seguinte proposição estabelece essas condições.

Proposição 1.2. *Considere a malha triangular quíntica de Bézier $P(u, v, w)$ com $b_{500} = A, b_{050} = B, b_{005} = C$, onde $A, B, C > 0$. Se $b_{ijk} \geq -r_0$, para $b_{ijk} \neq A, B, C$, onde r_0 é a única solução de*

$$\frac{1}{\sqrt[3]{\frac{A}{r} + 1}} + \frac{1}{\sqrt[3]{\frac{B}{r} + 1}} + \frac{1}{\sqrt[3]{\frac{C}{r} + 1}} = 1,$$

então $P(u, v, w) \geq 0, \forall u, v, w \geq 0, u + v + w = 1$.

As derivadas parciais de primeira e segunda ordem, juntamente com as condições de continuidade C^2 determinam as ordenadas de Bézier. Se necessário, ajustes são feitos, a fim de satisfazer a condições estabelecidas pela proposição acima. Dessa forma, todas as ordenadas de Bézier são determinadas, a superfície interpolante final é gerada por (1.3) e o objetivo é atingido.

Esse método, naturalmente, gera superfícies de melhor qualidade, visto que ele é de classe C^2. Em contrapartida, o fato de se usar uma malha triangular quíntica, torna o trabalho computacionalmente mais caro.

É vasta a quantidade de trabalhos similares, no que diz respeito à quantidade de variáveis (bivariado) no domínio triangulado. Podemos citar especialmente um estudo realizado em 2004, por Kong, Ong e Saw [1] onde é tratado um esquema de interpolação de dados bivariados espalhados, com a geração de uma superfície interpolante C^1 e não-negativa (esse estudo é detalhado no Capítulo 3). No entanto, pouco se tem feito a respeito da interpolação não-negativa de dados **trivariados**, onde a continuidade C^1 é preservada. E este, por sua vez, é o foco do nosso estudo.

Os Capítulos seguintes estão distribuídos da seguinte forma: no Capítulo 2 é feita uma abordagem dos aspectos básicos envolvendo a curva de Bézier, a forma de Bernstein e sua derivada, e os triângulos de Bézier são apresentados. No Capítulo 3 o trabalho de Kong, Ong e Saw é apresentado com detalhes, estimulando assim, o nosso estudo, que é desenvolvido no Capítulo 4, a construção da malha quártica tetraédrica de Bézier e a superfície não-negativa e C^1. Aplicações são realizadas no Capítulo 5 e as conclusões, juntamente com os trabalhos futuros, se encontram no Capítulo 6.

Capítulo 2

Preliminares

Sejam $\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2$ quaisquer três pontos do espaço euclideano $\mathbb{E}^3$ e tome $t \in \mathbb{R}$. As seguintes retas são formadas, ao interpolarmos linearmente $\mathbf{b}_0, \mathbf{b}_1$ e $\mathbf{b}_1, \mathbf{b}_2$, respectivamente

$$\mathbf{b}_0^1(t) = (1-t)\mathbf{b}_0 + t\mathbf{b}_1$$
$$\mathbf{b}_1^1(t) = (1-t)\mathbf{b}_1 + t\mathbf{b}_2.$$

Ao substituirmos essas equações em

$$\mathbf{b}_0^2(t) = (1-t)\mathbf{b}_0^1(t) + t\mathbf{b}_1^1(t),$$

obtemos

$$\mathbf{b}_0^2(t) = (1-t)^2\mathbf{b}_0 + 2t(1-t)\mathbf{b}_1 + t^2\mathbf{b}_2.$$

Essa expressão quadrática em t representa uma parábola, denotada por $\mathbf{b}^2$, com t variando de $-\infty$ a $+\infty$. Essa construção consiste em uma *interpolação linear repetida*.

Para t entre 0 e 1, $\mathbf{b}^2$ está dentro do triângulo formado por $\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2$; em particular, $\mathbf{b}^2(0) = \mathbf{b}_0$ e $\mathbf{b}^2(1) = \mathbf{b}_2$.

2.1 A Curva de Bézier

2.1.1 O Algoritmo de Casteljau

Foi visto anteriormente a construção de uma curva no plano, a parábola, através de três pontos. O algoritmo a seguir gera curvas paramétricas polinomiais de grau n qualquer, através de $n+1$ pontos.

Algoritmo de Casteljau
Entrada: Conjunto de pontos, $\mathbf{b}_0, \mathbf{b}_1, \cdots \mathbf{b}_n \in \mathbb{E}^3$ e $t \in \mathbb{R}$,
Saída: Curva de grau n denotada por $\mathbf{b}^n$, citada abaixo:

$$\mathbf{b}_i^r(t) = (1-t)\mathbf{b}_i^{r-1}(t) + t\mathbf{b}_{i+1}^{r-1}(t) \qquad \begin{cases} r = 1, \cdots, n \\ i = 0, \cdots, n-r \end{cases};$$

onde $\mathbf{b}_i^0(t) = \mathbf{b}_i$. Então $\mathbf{b}_0^n(t)$ é o ponto com parâmetro t na *curva de Bézier de grau n*, denotada por $\mathbf{b}^n$.

A poligonal P formada por $\mathbf{b}_0, \cdots, \mathbf{b}_n$ é chamada *poligonal de Bézier* ou *poligonal de controle* da curva $\mathbf{b}^n$. Similarmente, os vértices do polígono $\mathbf{b}_i$ são chamados *pontos de controle* ou *pontos de Bézier*.

Na Figura 2.1 está ilustrado o caso $n = 2$, onde $\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2$, através do algoritmo de Casteljau, geram uma quadrática de Bézier. Para $t \in [0,1]$, $\mathbf{b}^2(t)$ está dentro do triângulo formado por $\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2$. Em particular, $\mathbf{b}_0(t) = \mathbf{b}_0$ e $\mathbf{b}_1(t) = \mathbf{b}_1$.

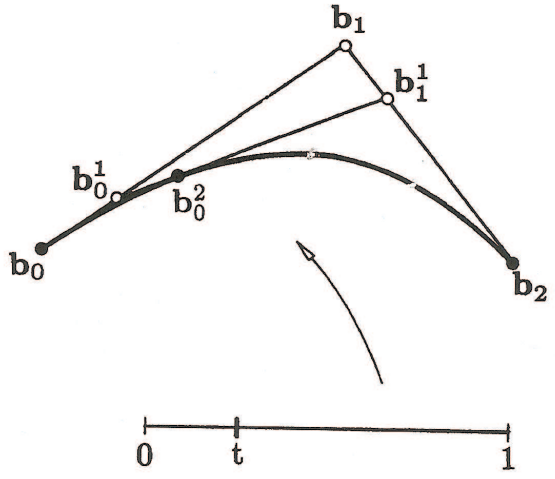

Figura 2.1 Quadrática de Bézier.

O que ocorreria numa cúbica, quando temos os pontos $\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3$? Vejamos isso: Na primeira iteração, onde $r = 1$, temos, para $0 \leq t \leq 1$, os segmentos de reta

$$\mathbf{b}_0^1(t) = (1-t)\mathbf{b}_0 + t\mathbf{b}_1,$$
$$\mathbf{b}_1^1(t) = (1-t)\mathbf{b}_1 + t\mathbf{b}_2,$$
$$\mathbf{b}_2^1(t) = (1-t)\mathbf{b}_2 + t\mathbf{b}_3.$$

Na segunda iteração, onde $r = 2$, temos outros dois segmentos de reta, agora entre os pontos determinados na iteração anterior:

$$\mathbf{b}_0^2(t) = (1-t)\mathbf{b}_0^1 + t\mathbf{b}_1^1,$$
$$\mathbf{b}_1^2(t) = (1-t)\mathbf{b}_1^1 + t\mathbf{b}_2^1.$$

Na terceira e última iteração, $r = 3$, temos o último segmento de reta:

$$\mathbf{b}_0^3(t) = (1-t)\mathbf{b}_0^2 + t\mathbf{b}_1^2.$$

Se considerarmos essa análise para t específico, por exemplo $t = 1/4$, como na Figura 2.2, encontramos o ponto final, $\mathbf{b}_0^3(1/4)$, um ponto da curva de Bézier. Fazendo isso para todo t entre 0 e 1, determinamos a cúbica que se inicia em $\mathbf{b}_0$ e finaliza em $\mathbf{b}_3$.

As curvas de Bézier possuem uma importante propriedade, a do *invólucro convexo*. Para $t \in [0,1]$, $\mathbf{b}^n(t)$ está no invólucro convexo do polígono de controle. Isto acontece porque todo ponto intermediário do algoritmo $\mathbf{b}_i^r(t)$ é obtido como uma combinação baricêntrica convexa dos $\mathbf{b}_j^{r-1}$

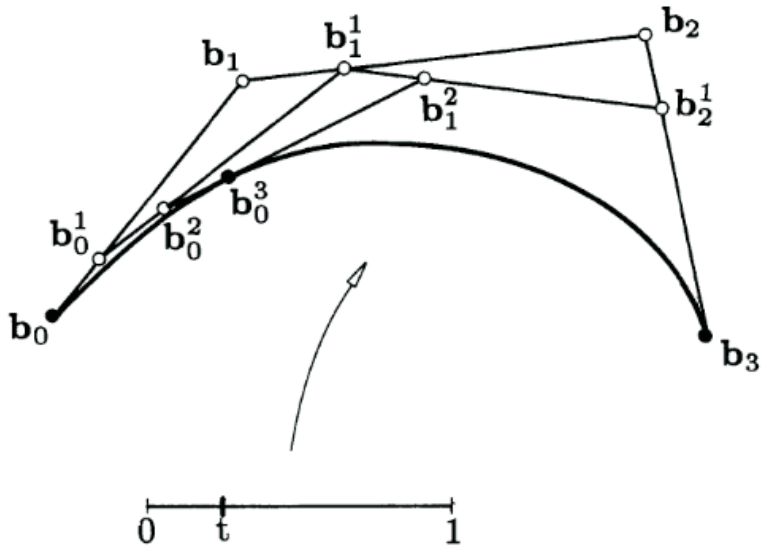

Figura 2.2 Cúbica de Bézier.

anteriores - em nenhum passo do algoritmo de Casteljau serão determinados pontos fora do invólucro convexo de $\mathbf{b}_i$. Uma importante consequência dessa propriedade é que polígonos de controle planares sempre gerarão uma curva planar.

2.1.2 A Forma de Bernstein de uma Curva de Bézier

As curvas de Bézier podem ser definidas pelo algoritmo recursivo, que foi como Casteljau o desenvolveu primeiramente. No entanto, é necessário ter uma representação explícita pra elas. Sendo assim, as curvas de Bézier serão expressas em termos dos *polinômios de Bernstein*, definidos explicitamente por

$$B_i^n(t) = \begin{pmatrix} n \\ i \end{pmatrix} t^i (1-t)^{n-i},$$

onde os coeficientes binomiais são dados por

$$\begin{pmatrix} n \\ i \end{pmatrix} = \begin{cases} \frac{n!}{i!(n-i)!} & \text{se } 0 \leq i \leq n \\ 0 & \text{caso contrário.} \end{cases}$$

Os polinômios de Bernstein satisfazem uma importante propriedade recursiva, seja ela:

$$B_i^n(t) = (1-t)B_i^{n-1}(t) + tB_{i-1}^{n-1}(t), \tag{2.1}$$

com

$$\begin{aligned} B_0^0(t) &\equiv 1, \\ B_j^n(t) &\equiv 0, j \notin \{0, \cdots, n\}. \end{aligned} \tag{2.2}$$

Outra propriedade importante é que os polinômios de Bernstein formam uma partição da unidade

$$\sum_{j=0}^{n} B_j^n(t) \equiv 1.$$

E, finalmente, os pontos intermediários do algoritmo de Casteljau, $\mathbf{b}_i^r$, podem ser expressos em termos dos polinômios de Bernstein de grau r:

$$\mathbf{b}_i^r(t) = \sum_{j=0}^{r} \mathbf{b}_{i+j} B_j^r(t), \tag{2.3}$$

com

$$r \in \{0, \cdots, n\}, \quad i \in \{0, \cdots, n-r\}.$$

Isso mostra claramente como os pontos intermediários, $\mathbf{b}_i^r$, dependem dos pontos de controle dados, $\mathbf{b}_i$.

Para verificarmos (2.3) basta ver que por Casteljau temos

$$\mathbf{b}_i^r(t) = (1-t)\mathbf{b}_i^{r-1}(t) + t\mathbf{b}_{i+1}^{r-1}(t)$$

e, usando o segundo princípio de indução, supõe-se que (2.3) é válido para todo $k \leq r-1$. Assim, podemos reescrever essa igualdade como

$$\mathbf{b}_i^r(t) = (1-t) \sum_{j=i}^{i+r-1} \mathbf{b}_j B_{j-i}^{r-1}(t) + t \sum_{j=i+1}^{i+r} \mathbf{b}_j B_{j-i-1}^{r-1}(t).$$

Reindexando e usando (2.2) concluimos que

$$\begin{aligned} \mathbf{b}_i^r(t) &= (1-t) \sum_{j=i}^{i+r} \mathbf{b}_j B_{j-i}^{r-1}(t) + t \sum_{j=i}^{i+r} \mathbf{b}_j B_{j-i-1}^{r-1}(t) \\ &= \sum_{j=i}^{i+r} \mathbf{b}_j [(1-t)B_{j-i}^{r-1}(t) + tB_{j-i-1}^{r-1}(t)]. \end{aligned}$$

Aplicando agora (2.1) temos que

$$\mathbf{b}_i^r(t) = \sum_{j=i}^{i+r} \mathbf{b}_j B_{j-i}^r(t) \text{ com } i \leq j \leq i+r, \quad 0 \leq j-i \leq r.$$

O que leva à igualdade definida em (2.3).

Sendo assim, para o caso $r = n$, temos uma expressão final para a curva de Bézier, em termos dos pontos de controle $\mathbf{b}_i$ e dos polinômios de Bernstein. Seja ela:

$$\mathbf{b}^n(t) = \mathbf{b}_0^n = \sum_{j=0}^{n} \mathbf{b}_j^r(t) B_j^n(t).$$

2.1.3 A Derivada da Curva de Bézier

A derivada de um polinômio de Bernstein obedece à seguinte expressão

$$\frac{d}{dt} B_i^n(t) = n[B_{i-1}^{n-1}(t) - B_i^{n-1}(t)].$$

Então podemos determinar a derivada de uma curva de Bézier $\mathbf{b}^n$ por

$$\frac{d}{dt}\mathbf{b}^n(t) = n \sum_{j=0}^{n} [B_{j-1}^{n-1}(t) - B_j^{n-1}(t)]\mathbf{b}_j.$$

Usando (2.2) podemos reescrever como

$$\frac{d}{dt}\mathbf{b}^n(t) = n \sum_{j=1}^{n} B_{j-1}^{n-1}(t)\mathbf{b}_j - n \sum_{j=0}^{n-1} B_j^{n-1}(t)\mathbf{b}_j.$$

Fazendo uma transformação no índice no primeiro termo e reagrupando temos

$$\frac{d}{dt}\mathbf{b}^n(t) = n \sum_{j=0}^{n-1} (\mathbf{b}_{j+1} - \mathbf{b}_j) B_j^{n-1}(t).$$

Definindo o *operador diferença* Δ:

$$\Delta\mathbf{b}_j = \mathbf{b}_{j+1} - \mathbf{b}_j,$$

pode-se escrever a derivada de uma curva de Bézier como

$$\frac{d}{dt}\mathbf{b}^n(t) = n \sum_{j=0}^{n-1} \Delta\mathbf{b}_j B_j^{n-1}(t); \quad \Delta\mathbf{b}_j \in \mathbb{R}^3.$$

Pode-se ver então que a derivada de uma curva de Bézier é uma outra curva de Bézier, com um grau a menos, obtida por diferenciar a poligonal de controle original. No entanto esta derivada não está mais em $\mathbb{E}^3$. Seus coeficientes são diferenças de pontos, isto é, são vetores, elementos de $\mathbb{R}^3$.

2.2 Curvas Não Paramétricas

Considere f uma curva funcional da forma $y = f(x)$, onde f denota um polinômio. Esta curva pode ser escrita na forma paramétrica

$$\mathbf{b}(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} t \\ f(t) \end{bmatrix}. \tag{2.4}$$

Nosso interesse é por funções f que são expressas em termos de bases de Bernstein:

$$f(t) = b_0 B_0^n(t) + \cdots b_n B_n^n(t),$$

onde b_j são números reais, não pontos, como na representação não-paramétrica. Sendo assim, os b_j não formam uma poligonal, mas as curvas funcionais são um subconjunto de curvas paramétricas e por isso deve possuir uma poligonal de controle. Para encontrá-lo usemos uma propriedade importante dos polinômios de Bernstein, a que eles formam uma partição da unidade:

$$\sum_{j=0}^{n} B_j^n(t) = 1.$$

Daí, chegamos a

$$\sum_{j=0}^{n} \frac{j}{n} B_j^n(t) = t.$$

Agora podemos escrever a curva funcional como

$$\mathbf{b}(t) = \sum_{j=0}^{n} \begin{bmatrix} j/n \\ b_j \end{bmatrix} B_j^n(t).$$

Então a poligonal de controle da função $f(t) = \sum_{j=0}^{n} b_j B_j^n(t)$ é dada pelos pontos $\left(\frac{j}{n}, b_j\right)$, para $j = 0, \cdots, n$. A função f é dita *função de Bézier* e b_i são as *ordenadas de Bézier*. A figura abaixo ilustra o caso de uma curva funcional para um polinômio cúbico, com suas abscissas $0, \frac{1}{3}, \frac{2}{3}, 1$.

2.3 Triângulos de Bézier

Os triângulos de Bézier são uma extensão natural das curvas de Bézier. Nesse último, tem-se pontos de controle dispostos arbitrariamente no plano ou no espaço e através de segmentos de reta que passam por eles, dois a dois, uma curva é construída, passando pelo primeiro e último pontos da sequência. Agora, os pontos de controle estão dispostos numa malha triangular e são denotados por $\mathbf{b}_{ijk}$.

A notação utilizada será:

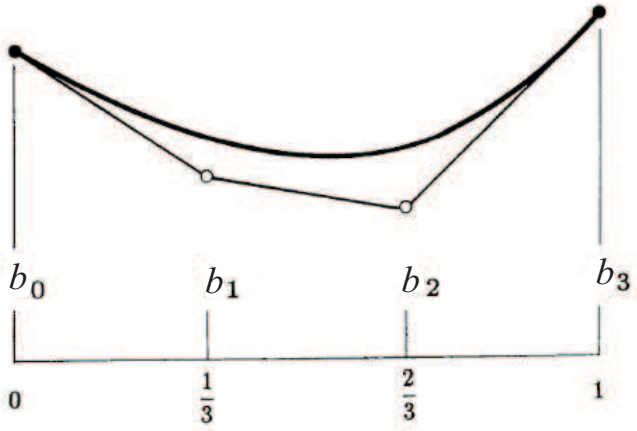

Figura 2.3 Curva de Bézier.

$$\mathbf{b_i} = \mathbf{b}_{ijk}, \text{ onde } |\mathbf{i}| = i + j + k = n \text{ e } i, j, k \geq 0,$$

e n denota o grau da superfície gerada.

Além disso

$$\mathbf{e}_1 = 100, \ \mathbf{e}_2 = 010, \ \mathbf{e}_3 = 001.$$

Veja o exemplo para o caso de $n = 3$, onde os pontos

$$\mathbf{b}_{003}, \mathbf{b}_{012}, \mathbf{b}_{021}, \mathbf{b}_{030}, \mathbf{b}_{120}, \mathbf{b}_{210}, \mathbf{b}_{300}, \mathbf{b}_{201}, \mathbf{b}_{102}, \mathbf{b}_{111}$$

estão dispostos na malha triangular da Figura 2.4.

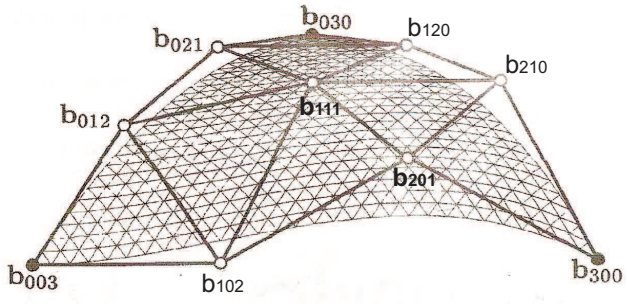

Figura 2.4 Triângulo de Bézier: malha cúbica.

O Algoritmo de Casteljau Estendido

Entrada: Uma malha triangular de pontos $\mathbf{b_i} \in \mathbb{E}^3$; $|\mathbf{i}| = n$, e um ponto em $\mathbb{E}^2$ com coordenadas baricêntricas $\mathbf{u} = (u, v, w)$.

Saída:

$$\mathbf{b}_{\mathbf{i}}^r(\mathbf{u}) = u\mathbf{b}_{\mathbf{i}+\mathbf{e}_1}^{r-1}(\mathbf{u}) + v\mathbf{b}_{\mathbf{i}+\mathbf{e}_2}^{r-1}(\mathbf{u}) + w\mathbf{b}_{\mathbf{i}+\mathbf{e}_3}^{r-1}(\mathbf{u}),$$

onde

$$r = 1, \cdots, n, \ |\mathbf{i}| = n - r \ \text{ e } \ \mathbf{b}_{\mathbf{i}}^{0}(\mathbf{u}) = \mathbf{b}_{\mathbf{i}}.$$

Então $\mathbf{b}_0^n(\mathbf{u})$ é o ponto com parâmetro $\mathbf{u}$ na superfície triangular de Bézier. Variando $\mathbf{u}$ ao longo de todo o triângulo de domínio em $\mathbb{E}^2$ obtemos a superfície dita *triângulo de Bézier de grau n*, denotada por $\mathbf{b}_0^n$ ou simplesmente de $\mathbf{b}^n$.

A idéia do algoritmo de Casteljau estendido é: tome um ponto $\mathbf{u}$, num determinado triângulo, com coordenadas baricêntricas (u, v, w), em relação aos vértices desse triângulo, e um conjunto de pontos de controle $\mathbf{b_i}$ que formam uma malha triangular. Em seguida são interpolados os vértices do triângulo formado por

$$\mathbf{b}_{\mathbf{i}+\mathbf{e}_1}, \mathbf{b}_{\mathbf{i}+\mathbf{e}_2}, \mathbf{b}_{\mathbf{i}+\mathbf{e}_3},$$

pertencente a essa malha, de forma que sejam preservadas as coordenadas baricêntricas de $\mathbf{u}$ com relação ao triângulo de domínio.

A Figura abaixo (2.5) mostra um exemplo da aplicação desse algoritmo para o caso de $n = 3$, onde na primeira iteração ($r = 1$) seriam obtidos os pontos:

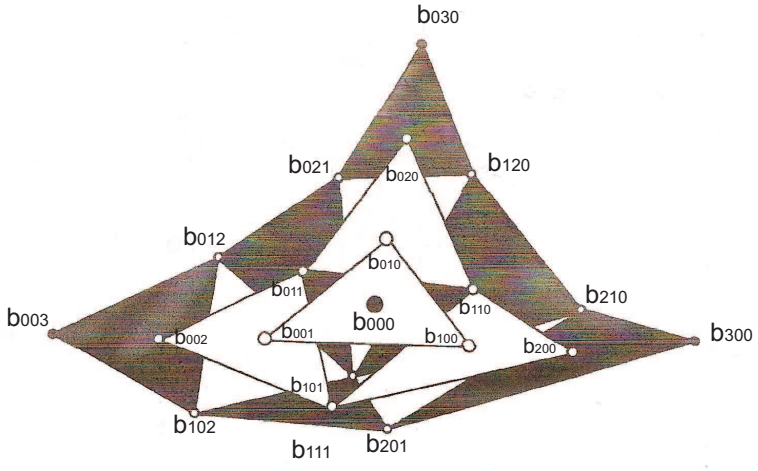

Figura 2.5 Algoritmo de Casteljau triangular.

$$\mathbf{b}_{200}^1 = u\mathbf{b}_{300} + v\mathbf{b}_{210} + w\mathbf{b}_{201},$$
$$\mathbf{b}_{020}^1 = u\mathbf{b}_{120} + v\mathbf{b}_{030} + w\mathbf{b}_{021},$$
$$\mathbf{b}_{002}^1 = u\mathbf{b}_{102} + v\mathbf{b}_{012} + w\mathbf{b}_{003},$$
$$\mathbf{b}_{110}^1 = u\mathbf{b}_{210} + v\mathbf{b}_{120} + w\mathbf{b}_{111},$$
$$\mathbf{b}_{101}^1 = u\mathbf{b}_{201} + v\mathbf{b}_{111} + w\mathbf{b}_{102},$$
$$\mathbf{b}_{011}^1 = u\mathbf{b}_{111} + v\mathbf{b}_{021} + w\mathbf{b}_{012}.$$

Na segunda iteração obteríamos mais três pontos, sejam eles:

$$\mathbf{b}^2_{100} = u\mathbf{b}^1_{200} + v\mathbf{b}^1_{110} + w\mathbf{b}^1_{101},$$
$$\mathbf{b}^2_{010} = u\mathbf{b}^1_{110} + v\mathbf{b}^1_{020} + w\mathbf{b}^1_{011},$$
$$\mathbf{b}^2_{001} = u\mathbf{b}^1_{101} + v\mathbf{b}^1_{011} + w\mathbf{b}^1_{002}.$$

Na terceira e última iteração, chegamos ao ponto desejado, que se encontra na superfície de Bézier gerada $\mathbf{b}^3$:

$$\mathbf{b}^3_{000} = u\mathbf{b}^2_{100} + v\mathbf{b}^2_{010} + w\mathbf{b}^2_{001}.$$

2.3.1 Polinômios de Bernstein

No caso bivariado os polinômios de Bernstein $B^n_{\mathbf{i}}$ são definidos por

$$B^n_{\mathbf{i}}(\mathbf{u}) = \begin{pmatrix} n \\ \mathbf{i} \end{pmatrix} u^i v^j w^k = \frac{n!}{i!\,j!\,k!} u^i v^j w^k; \quad |\mathbf{i}| = n. \tag{2.5}$$

Define-se $B^n_{\mathbf{i}}(\mathbf{u}) \equiv 0$ se $i, j, k < 0$ ou $i + j + k > n$ para algum i, j, k. Observe que, apesar de $B^n_{\mathbf{i}}(u, v, w)$ parecer trivariado, ele não é, pois temos que $u + v + w = 1$.

Sabendo que

$$\begin{pmatrix} n \\ \mathbf{i} \end{pmatrix} = \begin{pmatrix} n-1 \\ \mathbf{i}-\mathbf{e}_1 \end{pmatrix} + \begin{pmatrix} n-1 \\ \mathbf{i}-\mathbf{e}_2 \end{pmatrix} + \begin{pmatrix} n-1 \\ \mathbf{i}-\mathbf{e}_3 \end{pmatrix},$$

e que pela Definição (2.5) temos

$$uB^{n-1}_{\mathbf{i}-\mathbf{e}_1} = u\begin{pmatrix} n-1 \\ \mathbf{i}-\mathbf{e}_1 \end{pmatrix} u^{i-1} v^j w^k = \begin{pmatrix} n-1 \\ \mathbf{i}-\mathbf{e}_1 \end{pmatrix} u^i v^j w^k,$$
$$vB^{n-1}_{\mathbf{i}-\mathbf{e}_2} = v\begin{pmatrix} n-1 \\ \mathbf{i}-\mathbf{e}_2 \end{pmatrix} u^i v^{j-1} w^k = \begin{pmatrix} n-1 \\ \mathbf{i}-\mathbf{e}_2 \end{pmatrix} u^i v^j w^k,$$
$$wB^{n-1}_{\mathbf{i}-\mathbf{e}_3} = w\begin{pmatrix} n-1 \\ \mathbf{i}-\mathbf{e}_3 \end{pmatrix} u^i v^j w^{k-1} = \begin{pmatrix} n-1 \\ \mathbf{i}-\mathbf{e}_3 \end{pmatrix} u^i v^j w^k,$$

então os polinômios de Bernstein satisfazem a seguinte recursão:

$$B^n_{\mathbf{i}}(\mathbf{u}) = uB^{n-1}_{\mathbf{i}-\mathbf{e}_1}(\mathbf{u}) + vB^{n-1}_{\mathbf{i}-\mathbf{e}_2}(\mathbf{u}) + wB^{n-1}_{\mathbf{i}-\mathbf{e}_3}(\mathbf{u}), \quad |\mathbf{i}| = n.$$

Usando indução e a definição recursiva dos polinômios de Bernstein, temos que

$$\mathbf{b}^r_{\mathbf{i}}(\mathbf{u}) = \sum_{|\mathbf{j}|=r} \mathbf{b}_{\mathbf{i}+\mathbf{j}} B^r_{\mathbf{j}}(\mathbf{u}); \quad |\mathbf{i}| = n - r.$$

Fazendo $r = n$ nessa expressão acima, encontramos uma relação entre um ponto na superfície de Bézier e os pontos na malha:

$$\mathbf{b}^n(\mathbf{u}) = \mathbf{b}^n_0(\mathbf{u}) = \sum_{|\mathbf{j}|=n} \mathbf{b}_{\mathbf{j}} B^n_{\mathbf{j}}(\mathbf{u}).$$

2.3.2 Derivada

A derivada apropriada para os polinômios de Bézier é a *derivada direcional*. Sejam $\mathbf{u}_1$ e $\mathbf{u}_2$ dois pontos do domínio. A diferença entre eles, $\mathbf{d} = \mathbf{u}_2 - \mathbf{u}_1$, define um vetor. A derivada direcional de uma superfície em $\mathbf{x}(\mathbf{u})$ com respeito a $\mathbf{d}$ é dada por

$$D_{\mathbf{d}}\mathbf{x}(\mathbf{u}) = d\mathbf{x}_u(\mathbf{u}) + e\mathbf{x}_v(\mathbf{u}) + f\mathbf{x}_w(\mathbf{u}), \tag{2.6}$$

onde $\mathbf{d} = (d, e, f)$, com $d + e + f = 0$.

As derivadas parciais de uma malha de Bézier, com relação a u são calculadas da seguinte forma:

$$\begin{aligned}\frac{\partial}{\partial u}\mathbf{b}(\mathbf{u}) &= \frac{\partial}{\partial u}\sum_{|\mathbf{i}|=n}\mathbf{b}_{\mathbf{i}}B^n_{\mathbf{i}}(\mathbf{u}) \\ &= \frac{\partial}{\partial u}\sum_{|\mathbf{i}|=n}\frac{n!}{i!\,j!\,k!}u^i v^j w^k \mathbf{b}_{\mathbf{i}},\end{aligned}$$

por (2.5). Dessa última igualdade, desenvolvendo $n!$ e derivando com respeito a u temos:

$$= n\sum_{|\mathbf{i}|=n}\frac{(n-1)!}{(i-1)!\,j!\,k!}u^{i-1}v^j w^k \mathbf{b}_{\mathbf{i}}.$$

E, uma vez que $|\mathbf{i}| = n = i + j + k$, então também temos que $n - 1 = (i - 1) + j + k$. Dessa forma, podemos reindexar e escrever

$$\begin{aligned}&= n\sum_{|\mathbf{i}|=n-1}\frac{(n-1)!}{i!\,j!\,k!}u^i v^j w^k \mathbf{b}_{i+1,j,k} \\ &= n\sum_{|\mathbf{i}|=n-1}\mathbf{b}_{\mathbf{i}+\mathbf{e}_1}B^{n-1}_{\mathbf{i}}(\mathbf{u}).\end{aligned}$$

Com uma idéia análoga pode-se estabelecer as derivadas $\frac{\partial}{\partial v}\mathbf{b}(\mathbf{u})$ e $\frac{\partial}{\partial w}\mathbf{b}(\mathbf{u})$.

Juntando as três fórmulas para a derivada direcional e substituindo suas expressões encontradas, $\frac{\partial}{\partial u}\mathbf{b}(\mathbf{u})$, $\frac{\partial}{\partial v}\mathbf{b}(\mathbf{u})$, $\frac{\partial}{\partial w}\mathbf{b}(\mathbf{u})$, em (2.6) chegamos a

$$D_{\mathbf{d}}\mathbf{b}(\mathbf{u}) = d\frac{\partial}{\partial u}\mathbf{b}(\mathbf{u}) + e\frac{\partial}{\partial v}\mathbf{b}(\mathbf{u}) + f\frac{\partial}{\partial w}\mathbf{b}(\mathbf{u}). \tag{2.7}$$

E, substituindo

$$D_{\mathbf{d}}\mathbf{b}(\mathbf{u}) = n\sum_{|\mathbf{i}|=n-1}[d\mathbf{b}_{\mathbf{i}+\mathbf{e}_1} + e\mathbf{b}_{\mathbf{i}+\mathbf{e}_2} + f\mathbf{b}_{\mathbf{i}+\mathbf{e}_3}]B^{n-1}_{\mathbf{i}}(\mathbf{u}). \tag{2.8}$$

E, usando o algoritmo de Casteljau estendido, podemos concluir que

$$D_{\mathbf{d}}\mathbf{b}(\mathbf{u}) = n\sum_{|\mathbf{i}|=n-1}\mathbf{b}^1_{\mathbf{i}}(\mathbf{d})B^{n-1}_{\mathbf{i}}(\mathbf{u}). \tag{2.9}$$

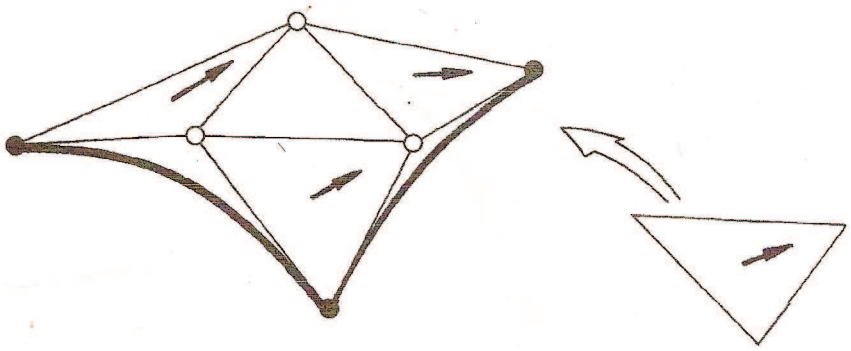

Figura 2.6 Derivada direcional: os coeficientes da derivada direcional de uma malha triangular são os vetores $\mathbf{b}_{\mathbf{i}}^{j}(\mathbf{d})$.

A derivada direcional de $\mathbf{b}^n$ é então uma malha triangular, cujos coeficientes são as imagens de $\mathbf{d}$ em cada subtriângulo de controle na rede de controle (veja a Figura 2.6).

Capítulo 3

Interpolação Restrita, Triangular de Bézier

A construção de uma superfície no computador com ajuda de programas gráficos, normalmente envolve gerar um conjunto de retalhos de superfície que são suavemente conectadas com certo grau de suavidade. Além disso, é interessante preservar algumas propriedades inerentes aos dados, como positividade, monotonicidade e convexidade, produzidas por uma interpolação por partes.

A preservação da não-negatividade refere-se à superfície interpolante gerada, que será não-negativa se os valores dados forem não-negativos.

O objetivo do estudo realizado por Kong, Ong, Saw em [1] foi interpolar dados espalhados bivariados, não-negativos e obter uma superfície C^1 e não-negativa. Para isso, seu domínio triangulado (obtido através da triangulação de Delaunay, apresentada em [3]) teve cada triângulo dividido em três mini-triângulos (a fim de garantir a continuidade C^1, seguindo o método de Clough-Tocher [5]) e em cada um desses uma malha de Bézier cúbica foi construída. Em seguida condições suficientes para a não-negatividade desses triângulos cúbicos de Bézier foram estipuladas através de limites inferiores para as ordenadas de Bézier. Finalmente a superfície foi construída, podendo haver alguns ajustes nas derivadas, caso seja necessário, e o estudo concluído.

Neste Capítulo, o trabalho de Kong *et al* [1] será revisto com o duplo propósito de apresentar o trabalho que foi utilizado para ser generalizado nesta tese, bem como de familiarizar o leitor com os conceitos específicos na literatura deste grupo de interpolações. Também será apresentada a razão pela qual não se consegue garantir a condição C^2 com polinômios cúbicos por parte, justificando em parte o uso de malhas quínticas em Saaban *et al* [11].

3.1 Continuidade C^1 entre Triângulos Cúbicos de Bézier Adjacentes

Seja T um triângulo no plano $x-y$, com vértices V_1, V_2, V_3 e coordenadas baricêntricas u, v e w, tal que qualquer ponto V do triângulo pode ser expresso de forma única como

$$V = uV_1 + vV_2 + wV_3, \text{ com } u+v+w = 1, \quad u,v,w \geq 0.$$

Seja S um funcional polinomial $S : \mathbb{E}^2 \to \mathbb{R}$ cúbico ($n = 3$) de Bézier definido em T como

$$S(u,v,w) = b^3(u,v,w) = \sum_{\substack{i+j+k=3 \\ i,j,k\geq 0}} b_{ijk} \frac{3!}{i!\,j!k!} u^i v^j w^k,$$

onde $u+v+w = 1, \quad u,v,w \geq 0$, ou seja para pontos no interior e na fronteira de T.

Ou ainda, utilizando os polinômios de Bernstein, temos

$$S(\mathbf{u}) = b^3(\mathbf{u}) = \sum_{\substack{i+j+k=3 \\ i,j,k\geq 0}} b_{ijk} B^3_{\mathbf{i}}(\mathbf{u}), \text{ onde } B^3_{\mathbf{i}}(\mathbf{u}) = \frac{3!}{i!\,j!k!} u^i v^j w^k,$$

sendo $\mathbf{u} = (u, v, w)$, com $\mathbf{i} = ijk$ e b_{ijk} denotando os valores de controle, associados a pontos igualmente espaçados de T. Se assumirmos que b_{300} é o valor de controle associado a V_1, b_{030} é associado a V_2, b_{003} é associado a V_3 então b_{ijk} é o valor de controle associado à posição $\frac{1}{3}[iV_1 + jV_2 + kV_3]$.

As ordenadas b_{ijk} (exceto b_{111}) são ditas *ordenadas de Bézier de fronteira* e b_{111} *ordenada de Bézier interna* do triângulo cúbico de Bézier S. As ordenadas de Bézier de fronteira, exceto as que não são vértices do triângulo original (b_{300}, b_{030} e b_{003}), são determinadas pela derivada direcional nos vértices ao longo da fronteira correspondente.

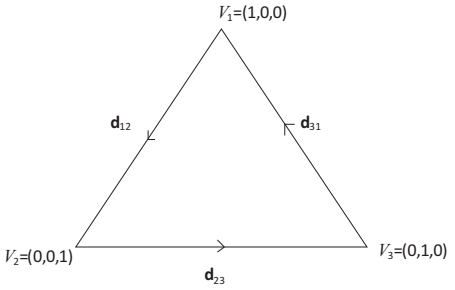

Figura 3.1 Notação no triângulo.

Vejamos como esse cálculo é feito, considerando as direções $\mathbf{d}_{12} = (-1, 1, 0)$, $\mathbf{d}_{31} = (1, 0, -1)$ e $\mathbf{d}_{23} = (0, -1, 1)$. Usando a fórmula (2.7), temos então que:

$$D_{\mathbf{d}_{12}} b(\mathbf{u}) = -1\frac{\partial b}{\partial u}(\mathbf{u}) + 1\frac{\partial b}{\partial v}(\mathbf{u}) + 0\frac{\partial b}{\partial w}(\mathbf{u}),$$

$$D_{\mathbf{d}_{31}} b(\mathbf{u}) = 1\frac{\partial b}{\partial u}(\mathbf{u}) + 0\frac{\partial b}{\partial v}(\mathbf{u}) - 1\frac{\partial b}{\partial w}(\mathbf{u}),$$

$$D_{\mathbf{d}_{23}} b(\mathbf{u}) = 0\frac{\partial b}{\partial u}(\mathbf{u}) - 1\frac{\partial b}{\partial v}(\mathbf{u}) + 1\frac{\partial b}{\partial w}(\mathbf{u}).$$

Usando (2.8), chegamos a

$$D_{\mathbf{d}_{12}}(V) = -3\sum_{|\mathbf{i}|=2} b_{\mathbf{i}+\mathbf{e}_1} B^2_{\mathbf{i}}(V) + 3\sum_{|\mathbf{i}|=2} b_{\mathbf{i}+\mathbf{e}_2} B^2_{\mathbf{i}}(V),$$

$$D_{\mathbf{d}_{31}}(V) = 3\sum_{|\mathbf{i}|=2} b_{\mathbf{i}+\mathbf{e}_1} B^2_{\mathbf{i}}(V) - 3\sum_{|\mathbf{i}|=2} b_{\mathbf{i}+\mathbf{e}_3} B^2_{\mathbf{i}}(V)$$

e

$$D_{\mathbf{d}_{23}}(V) = -3\sum_{|\mathbf{i}|=2} b_{\mathbf{i}+\mathbf{e}_2} B^2_{\mathbf{i}}(V) + 3\sum_{|\mathbf{i}|=2} b_{\mathbf{i}+\mathbf{e}_3} B^2_{\mathbf{i}}(V),$$

onde

$$\mathbf{i} \in \{110, 101, 011, 200, 020, 002\}.$$

Sabemos que, para $V = (u, v, w)$ qualquer, temos que

$$\sum_{|\mathbf{i}|=2} b_{\mathbf{i}+\mathbf{e}_1} B^2_{\mathbf{i}}(V), \sum_{|\mathbf{i}|=2} b_{\mathbf{i}+\mathbf{e}_2} B^2_{\mathbf{i}}(V) \text{ e } \sum_{|\mathbf{i}|=2} b_{\mathbf{i}+\mathbf{e}_3} B^2_{\mathbf{i}}(V)$$

são iguais a

$$b_{210}B^2_{110}(V) + b_{201}B^2_{101}(V) + b_{111}B^2_{011}(V) + b_{300}B^2_{200}(V) + b_{120}B^2_{020}(V) + b_{102}B^2_{002}(V),$$

$$b_{120}B^2_{110}(V) + b_{111}B^2_{101}(V) + b_{021}B^2_{011}(V) + b_{210}B^2_{200}(V) + b_{030}B^2_{020}(V) + b_{012}B^2_{002}(V)$$

e

$$b_{111}B^2_{110}(V) + b_{102}B^2_{101}(V) + b_{012}B^2_{011}(V) + b_{201}B^2_{200}(V) + b_{021}B^2_{020}(V) + b_{003}B^2_{002}(V),$$

respectivamente.

Portanto, usando a Definição (2.5), essas igualdades ficarão

$$\sum_{\mathbf{i}=2} b_{\mathbf{i}+\mathbf{e}_1} B^2_{\mathbf{i}}(V) = 2uvb_{210} + 2uwb_{201} + 2vwb_{111} + u^2b_{300} + v^2b_{120} + w^2b_{102},$$

$$\sum_{\mathbf{i}=2} b_{\mathbf{i}+\mathbf{e}_2} B^2_{\mathbf{i}}(V) = 2uvb_{120} + 2uwb_{111} + 2vwb_{021} + u^2b_{210} + v^2b_{030} + w^2b_{012},$$

$$\sum_{\mathbf{i}=2} b_{\mathbf{i}+\mathbf{e}_3} B^2_{\mathbf{i}}(V) = 2uvb_{111} + 2uwb_{102} + 2vwb_{012} + u^2b_{201} + v^2b_{021} + w^2b_{003}.$$

Analisando em $V = V_1 = (1,0,0)$, $V = V_2 = (0,1,0)$ e $V = V_3 = (0,0,1)$, as derivadas direcionais ficam

$$D_{\mathbf{d}_{12}}(V_1) = 3b_{210} - 3b_{300}, \quad D_{\mathbf{d}_{31}}(V_1) = 3b_{300} - 3b_{201},$$

$$D_{\mathbf{d}_{12}}(V_2) = -3b_{120} + 3b_{030}, \quad D_{\mathbf{d}_{23}}(V_2) = -3b_{030} + 3b_{021},$$

$$D_{\mathbf{d}_{31}}(V_3) = 3b_{102} - 3b_{003}, \quad D_{\mathbf{d}_{23}}(V_3) = -3b_{012} + 3b_{003}.$$

Sendo assim, pelas igualdades abaixo, verificamos que as ordenadas de Bézier de fronteira são determinadas através da derivada direcional, avaliada nos vértices da fronteira correspondente, como dito anteriormente

$$\begin{aligned}
b_{210} &= b_{300} + \tfrac{1}{3} D_{\mathbf{d}_{12}}(V_1) = S(V_1) + \tfrac{1}{3} D_{\mathbf{d}_{12}}(V_1), \\
b_{201} &= b_{300} - \tfrac{1}{3} D_{\mathbf{d}_{31}}(V_1) = S(V_1) - \tfrac{1}{3} D_{\mathbf{d}_{31}}(V_1), \\
b_{120} &= b_{030} - \tfrac{1}{3} D_{\mathbf{d}_{12}}(V_2) = S(V_2) - \tfrac{1}{3} D_{\mathbf{d}_{12}}(V_2), \\
b_{021} &= b_{030} + \tfrac{1}{3} D_{\mathbf{d}_{23}}(V_2) = S(V_2) + \tfrac{1}{3} D_{\mathbf{d}_{23}}(V_2), \\
b_{102} &= b_{003} + \tfrac{1}{3} D_{\mathbf{d}_{31}}(V_3) = S(V_3) + \tfrac{1}{3} D_{\mathbf{d}_{31}}(V_3), \\
b_{012} &= b_{003} - \tfrac{1}{3} D_{\mathbf{d}_{23}}(V_3) = S(V_3) - \tfrac{1}{3} D_{\mathbf{d}_{23}}(V_3).
\end{aligned} \tag{3.1}$$

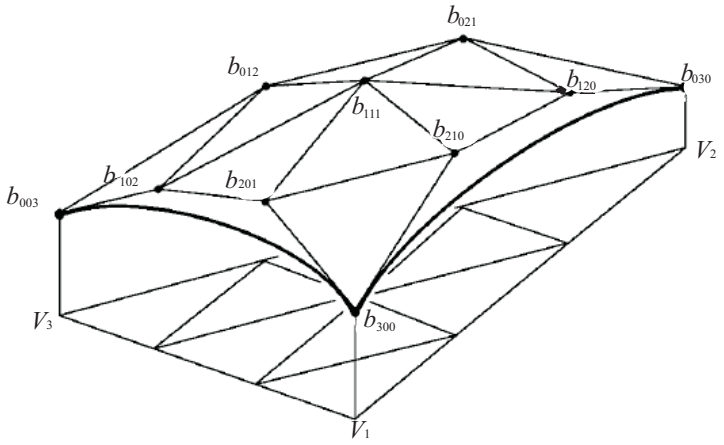

Figura 3.2 Malha cúbica de Bézier sobre seu domínio.

3.1.1 Condições de Continuidade

Considere dois triângulos $\Delta V_1 V_2 V_3$, $\Delta W_1 W_2 W_3$ no plano $x-y$, onde $V_2 = W_3$ e $V_3 = W_2$.

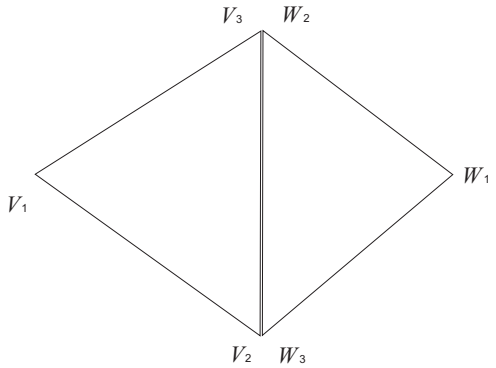

Figura 3.3 Triângulos adjacentes.

Suponha triângulos cúbicos de Bézier sobre esses dois triângulos adjacentes, com ordenadas de Bézier b_{ijk} e c_{ijk}, respectivamente. Esses dois triângulos cúbicos de Bézier têm a mesma curva de fronteira ao longo da aresta V_2V_3, então $b_{003} = c_{030}$, $b_{012} = c_{021}$, $b_{021} = c_{012}$ e $b_{030} = c_{003}$ (veja Figura 3.4). Queremos estabelecer condições para que a curva de fronteira tenha continuidade C^1.

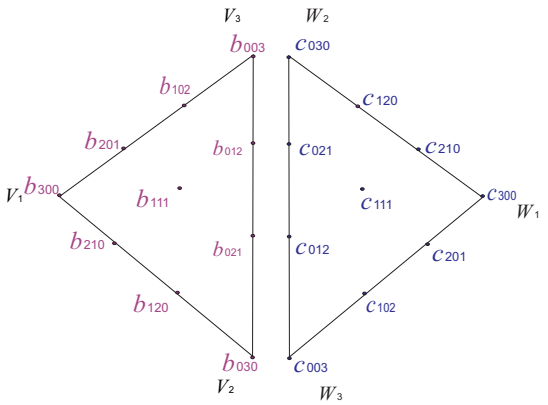

Figura 3.4 Malhas triangulares em triângulos adjacentes (visão de cima).

Associemos as ordenadas de Bézier b_{ijk} com (x_{ijk}, y_{ijk}), para $0 \leq i, j, k \leq 3$, sendo

$$(x_{ijk}, y_{ijk}) = \frac{1}{3}(iV_1 + jV_2 + kV_3),$$

e no nosso caso temos os pontos $\mathbf{b}_{ijk}$ e $\mathbf{c}_{ijk}$, onde $\mathbf{b}_{ijk} = (x_{ijk}, y_{ijk}, b_{ijk})$ e $\mathbf{c}_{ijk} = (x_{ijk}, y_{ijk}, c_{ijk})$. Para a condição C^1 ser satisfeita, geometricamente os quatro pontos de cada conjunto,

$$\{\mathbf{c}_{102}, \mathbf{b}_{120}, \mathbf{b}_{030}, \mathbf{b}_{021}\}, \{\mathbf{c}_{111}, \mathbf{b}_{111}, \mathbf{b}_{021}, \mathbf{b}_{012}\} \text{ e } \{\mathbf{c}_{120}, \mathbf{b}_{102}, \mathbf{b}_{012}, \mathbf{b}_{003}\}$$

precisam ser coplanares e a posição relativa de $\mathbf{c}_{102}$ com respeito ao triângulo $\mathbf{b}_{120}\mathbf{b}_{030}\mathbf{b}_{021}$ é a mesma posição relativa de W_1 com respeito ao triângulo $V_1V_2V_3$; e assim é também para os outros de forma análoga.

Sendo assim, sabendo que $W_1 = \alpha V_1 + \beta V_2 + \gamma V_3$, onde α, β e γ são constantes que somam 1, as condições necessárias e suficientes de continuidade C^1 entre as duas malhas são dadas pela preservação da combinação baricêntrica de V_1, V_2, V_3, ou seja:

$$\begin{aligned} c_{102} &= \alpha b_{120} + \beta b_{030} + \gamma b_{021}, \\ c_{111} &= \alpha b_{111} + \beta b_{021} + \gamma b_{012}, \\ c_{120} &= \alpha b_{102} + \beta b_{012} + \gamma b_{003}. \end{aligned} \tag{3.2}$$

Para verificarmos a primeira e última Equações acima, suponha que os triângulos vizinhos tenham as mesmas derivadas direcionais nos vértices da aresta comum. Considere que um ponto escrito com respeito à malha b tem coordenadas (u, v, w) e com respeito à malha c tem $(\bar{u}, \bar{v}, \bar{w})$, onde $u = \alpha\bar{u}$, $v = \beta\bar{u} + \bar{w}$ e $w = \gamma\bar{u} + \bar{v}$. Considere a derivada parcial com respeito à $\bar{u}$ nos vértices V_2 e W_3 as mesmas. Então

$$\frac{\partial b}{\partial \bar{u}}(V_2) = \frac{\partial c}{\partial \bar{u}}(W_3) \Rightarrow 3c_{102} = \frac{\partial b}{\partial u}\frac{du}{d\bar{u}}(V_2) + \frac{\partial b}{\partial v}\frac{dv}{d\bar{u}}(V_2) + \frac{\partial b}{\partial w}\frac{dw}{d\bar{u}}(V_2)$$

e da igualdade acima temos

$$c_{102} = \alpha b_{120} + \beta b_{030} + \gamma b_{021}.$$

Analogamente

$$\frac{\partial b}{\partial \bar{u}}(V_3) = \frac{\partial c}{\partial \bar{u}}(W_2) \Rightarrow c_{120} = \alpha b_{102} + \beta b_{012} + \gamma b_{003}.$$

Dessa forma, a primeira e terceira igualdades de (3.2) são satisfeitas.

O segundo conjunto de condições é suficiente, mas não é necessário. Como será visto adiante, ele será importante para que se alcance a propriedade da localidade, ou independência das condições no tetraedro. Ele foi estabelecido por [6] e consiste em determinar as ordenadas internas de Bézier, b_{111} e c_{111}, de forma a garantir a continuidade C^1 ao longo da fronteira comum. Ela é alcançada fazendo com que a derivada direcional com respeito à normal varie linearmente ao longo dessa fronteira.

Considere o vetor $\mathbf{d}_{ij}$ no triângulo como sendo o que se origina no vértice V_i e termina no vértice V_j.

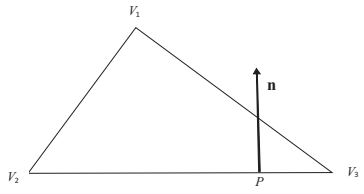

Figura 3.5 O triângulo e a normal.

Considere também o vetor normal ao vetor $\mathbf{d}_{23}$ como sendo $\mathbf{n}$ (veja Figura 3.5) e a projeção de $\mathbf{d}_{21}$ sobre $\mathbf{d}_{23}$ como $proj\ {}^{\mathbf{d}_{21}}_{\mathbf{d}_{23}}$. Sabemos que o ponto P pode ser escrito como $P(h) = hV_3 + (1-h)V_2$, onde $h\mathbf{d}_{23} = proj\ {}^{\mathbf{d}_{21}}_{\mathbf{d}_{23}}$ e este, por sua vez, satisfaz

$$\|proj\ {}^{\mathbf{d}_{21}}_{\mathbf{d}_{23}}\| = \frac{|\mathbf{d}_{21}\cdot\mathbf{d}_{23}|}{\|\mathbf{d}_{23}\|^2}\|\mathbf{d}_{23}\| = \frac{-\mathbf{d}_{12}\cdot\mathbf{d}_{23}}{\|\mathbf{d}_{23}\|^2}\|\mathbf{d}_{23}\|,$$

ou ainda

$$h = \frac{|\mathbf{d}_{21}\cdot\mathbf{d}_{23}|}{\|\mathbf{d}_{23}\|^2} = \frac{-\mathbf{d}_{12}\cdot\mathbf{d}_{23}}{\|\mathbf{d}_{23}\|^2}.$$

Como $\mathbf{n} = \overrightarrow{PV_1}$ então a combinação baricêntrica do vetor normal com respeito aos vértices do triângulo é $V_1 + (h-1)V_2 - hV_3$. Pela Definição (2.7) a derivada na direção do vetor normal fica

$$D_{\mathbf{n}}b(\mathbf{u}) = \frac{\partial b}{\partial u}(\mathbf{u}) + (h-1)\frac{\partial b}{\partial v}(\mathbf{u}) - h\frac{\partial b}{\partial w}(\mathbf{u})$$

$$= 3\sum_{|\mathbf{i}|=2} b_{\mathbf{i}+\mathbf{e}_1} B^2_{\mathbf{i}}(\mathbf{u}) + 3(h-1)\sum_{|\mathbf{i}|=2} b_{\mathbf{i}+\mathbf{e}_2} B^2_{\mathbf{i}}(\mathbf{u}) - 3h\sum_{|\mathbf{i}|=2} b_{\mathbf{i}+\mathbf{e}_3} B^2_{\mathbf{i}}(\mathbf{u})$$

e, como na fronteira comum entre os dois triângulos, temos os pontos (u, v, w), com $u+v+w = 1$, satisfazendo $u = 0$, então desenvolvendo essa derivada e usando que $w = 1 - v$ encontramos

$$3\{v^2[b_{120} + (h-1)b_{030} - hb_{021} - 2b_{111} - 2(h-1)b_{021} + 2hb_{012} + b_{102} + (h-1)b_{012} - hb_{003} +$$

$$+v[2b_{111} + 2(h-1)b_{021} - 2hb_{012} + 2hb_{003}] + b_{102} + (h-1)b_{012} - hb_{003}\}.$$

A linearidade da derivada direcional implica no coeficiente de v^2 ser nulo. Portanto, isolando a ordenada interna b_{111}, encontramos

$$b_{111} = \frac{1}{2}[b_{120} + b_{102} + h(2b_{012} - b_{021} - b_{003}) + (1-h)(2b_{021} - b_{030} - b_{012})]. \tag{3.3}$$

Analogamente, para a malha vizinha temos

$$c_{111} = \frac{1}{2}[c_{120} + c_{102} + \tilde{h}(2c_{012} - c_{021} - c_{003}) + (1-\tilde{h})(2c_{021} - c_{030} - c_{012})]. \tag{3.4}$$

Considere b_{111} e c_{111} como em (3.3) e (3.4), respectivamente, e $\tilde{h} = 1 - (\alpha h + \gamma)$. Acrescentando a essas informações que $b_{003} = c_{030}$, $b_{012} = c_{021}$, $b_{021} = c_{012}$ e $b_{030} = c_{003}$, verificamos que a segunda equação de (3.2) é verdadeira. Dessa forma, juntamente com as outras duas equações satisfeitas, temos a continuidade C^1 atingida.

Para construir uma superfície de interpolação C^1, por partes, o método de Clough-Tocher requer para cada triângulo o valor da função e as duas derivadas parciais em cada um dos três vértices, assim como a derivada na direção da normal no ponto médio de cada aresta do triângulo. Isto impõe doze condições no total sobre o polinômio cúbico definido sobre o triângulo. No entanto isso não é possível uma vez que um polinômio cúbico bivariado é determinado por apenas dez coeficientes. Dessa forma a idéia de [5] foi dividir cada triângulo do domínio triangulado (*triângulo macro*) em outros três (*mini-triângulos*), definindo uma malha cúbica de Bézier em cada um dos três. Esta construção nos fornece mais graus de liberdade, a fim de satisfazer as condições de continuidade [13].

Dessa forma o esquema adotado é dividir cada triângulo macro T, no domínio triangulado, em três mini-triângulos, dado um ponto interior G dele (veja Figura 3.7). G é escolhido como o centróide de T.

A continuidade C^1 ao longo da fronteira comum de dois triângulos macros adjacentes é obtida usando o segundo conjunto de condições, (3.3) e (3.4), enquanto que a continuidade C^1 ao longo da fronteira comum entre dois mini-triângulos adjacentes, que estão no mesmo triângulo macro, é encontrado usando o primeiro conjunto de condições (3.2). Note que a condição C^1 vincula pontos de controle na fronteira (exemplo: $b_{003} = c_{030}$, $b_{012} = c_{021}$, $b_{021} = c_{012}$ e $b_{030} = c_{003}$) e pontos de controle da próxima camada, partindo da fronteira (exemplo: $b_{120}, b_{111}, b_{102}$, num triângulo e c_{102}, c_{111} e c_{120} no outro). A condição C^2 vincula uma camada

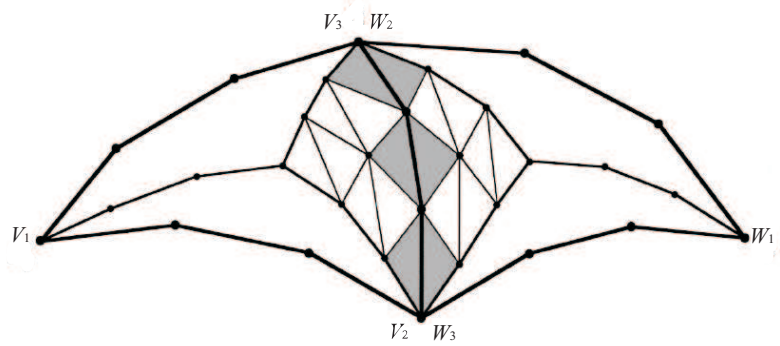

Figura 3.6 Três mini-triângulos em triângulos macros vizinhos.

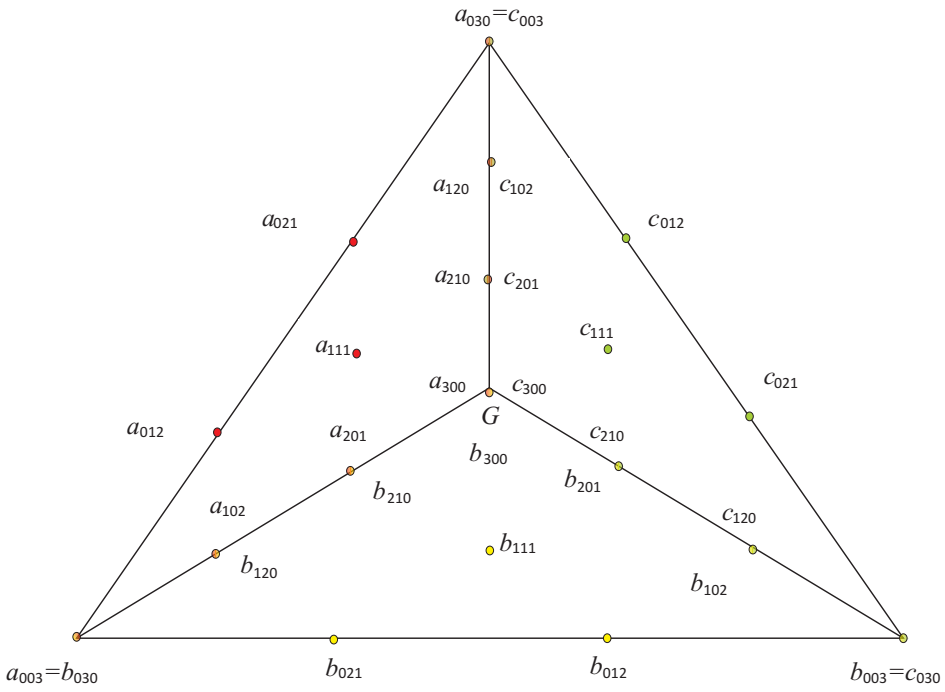

Figura 3.7 Ordenadas de Bézier dos três mini-triângulos.

adicional em cada triângulo. No caso da fronteira com o triângulo macro vizinho, a condição C^2 vincularia os pontos b_{021}, b_{012}, b_{003}, b_{111}, b_{102} e b_{201}, e os correspondentes pontos no triângulo vizinho, apenas para satisfazer C^2 no vértice b_{003}. No caso do vértice b_{030}, os pontos envolvidos incluiriam b_{021},b_{111} e b_{012}, já vinculados pela condição C^2 em b_{003}. Mesmo com uma malha quártica, haveria ainda um ponto em comum, no caso b_{022}, nas condições C^2 nos vértices b_{040} e b_{004}, e portanto uma malha quíntica seria necessária. Uma análise semelhante pode ser aplicada no caso trivariado, com conclusão ainda mais estrita com relação ao grau mínimo para comportar suavidade C^2. Neste trabalho, será adotado grau quatro, que não permite tal suavidade (ver Capítulo 4).

Suponha que cada triângulo cúbico de Bézier nos três mini-triângulos que estão no triângulo macro T têm ordenadas de Bézier

$$\{a_{ijk}\},\{b_{ijk}\} \text{ e } \{c_{ijk}\}, \text{ onde } 0 \leq i,j,k \leq 3,\ i+j+k=3,$$

respectivamente, como está ilustrado na Figura 3.7. Necessita-se que as três malhas cúbicas de Bézier tenham continuidade C^1 e as derivadas na direção da normal variando linearmente ao longo das três arestas do triângulo macro T. Pela continuidade C^0 entre esses três triângulos de Bézier cúbicos ao longo de V_i, $i = 1,2,3$, temos que

$$a_{030} = c_{003},\quad a_{120} = c_{102},\quad a_{210} = c_{201},\quad a_{003} = b_{030},\quad b_{120} = a_{102},$$
$$b_{210} = a_{201},\quad b_{003} = c_{030},\quad c_{120} = b_{102},\quad c_{210} = b_{201},$$
$$a_{030} = b_{300} = c_{300}.$$

Considere os três triângulos no triângulo macro e divida-os nos seguintes conjuntos

$$M_1 = \{a_{210} = c_{201}, b_{210} = a_{201}, c_{210} = b_{201}\},$$

$$M_2 = \{a_{111}, b_{111}, c_{111}\},$$

$$M_3 = \{c_{102} = a_{120}, a_{102} = b_{120}, b_{102} = c_{120}\},$$

$$M_4 = \{a_{021}, a_{012}, b_{021}, b_{012}, c_{021}, c_{012}\},$$

$$M_5 = \{a_{030} = c_{003}, a_{003} = b_{030}, b_{003} = c_{030}\}.$$

Deseja-se estabelecer uma sequência de passos para a determinação de todos os elementos da malha triangular de Bézier. Como vimos inicialmente, as derivadas direcionais nos vértices V_i, $i = 1,2,3$ determinarão as ordenadas de Bézier de fronteira (veja (3.1)), ou seja M_4. Em seguida, usando a continuidade C^1 sobre mini-triângulos vizinhos, os elementos de M_3 serão encontrados, usando as seguintes Equações:

$$\begin{aligned} a_{102} &= \tfrac{1}{3}(b_{021} + a_{012} + b_{030}), \\ b_{102} &= \tfrac{1}{3}(c_{021} + b_{012} + c_{030}), \\ c_{102} &= \tfrac{1}{3}(a_{021} + c_{012} + a_{030}). \end{aligned} \tag{3.5}$$

Pela continuidade C^1 nos triângulos macros, analisando em cada uma de suas arestas, a Equação (3.3) definida para cada um dos mini-triângulos determina as ordenadas internas a_{111}, b_{111}, c_{111} e, tendo isso, as ordenadas de M_1 são determinadas pela continuidade C^1, mais uma vez, que permite as Equações:

$$\begin{aligned} a_{201} &= \tfrac{1}{3}(a_{102} + b_{111} + a_{111}), \\ b_{201} &= \tfrac{1}{3}(b_{102} + c_{111} + b_{111}), \\ c_{201} &= \tfrac{1}{3}(c_{102} + a_{111} + c_{111}). \end{aligned} \tag{3.6}$$

E, finalmente, as ordenadas centrais (ponto G), serão determinadas pela Equação

$$a_{300} = \frac{1}{3}(a_{201} + b_{201} + c_{201}). \tag{3.7}$$

Seguindo esses passos, todas as ordenadas de Bézier da Figura 3.7 são determinadas.

3.1.2 Condições Suficientes para Não-negatividade

Além de determinar uma superfície interpolada com continuidade C^1, deseja-se satisfazer o critério da não-negatividade da mesma e, para isso, é preciso estimar limitantes para os valores de cada uma das coordenadas de uma malha cúbica de Bézier.

Comecemos fixando as ordenadas de Bézier nos três vértices do triângulo T, positivas, como sendo $m \geq l$ para todo $m \in M_5$, sendo l um número positivo qualquer. É necessário estimar condições suficientes para garantir que as três malhas cúbicas definidas nos mini-triângulos de T, interpolando os valores positivos dados, sejam não-negativas, enquanto a continuidade C^1 é mantida ao longo das arestas comuns.

O seguinte Teorema garante a não-negatividade da curva de Bézier cúbica.

Teorema 3.1. *Seja $r(x) = A(1-x)^3 + 3B(1-x)^2x + 3C(1-x)x^2 + Dx^3$, $0 \leq x \leq 1$, onde A e D são positivos e ao menos B ou C é negativo. Então $r(x) < 0$ para algum $x \in (0,1)$ [respectivamente $r(x) = 0$ para algum ponto em (0,1)] se e somente se $3B^2C^2 + 6ABCD - 4(AC^3 + B^3D) - A^2D^2 > 0$ [respectivamente = 0].*

Demonstração: Considere

$$r(x) = A(1-x)^3 + 3B(1-x)^2x + 3C(1-x)x^2 + Dx^3,$$

com $0 \leq x \leq 1$, onde A e D são positivos e

$$\Phi = 3B^2C^2 + 6ABCD - 4(AC^3 + B^3D) - A^2D^2.$$

Se $B = \frac{-A}{3}$ e $C = \frac{-D}{3}$ então

$$\begin{aligned} \Phi &= 3\left(\frac{-A}{3}\right)^2\left(\frac{-D}{3}\right)^2 + 6A \cdot \frac{-A}{3} \cdot \frac{-D}{3} \cdot D - 4\left[A \cdot \left(\frac{-D}{3}\right)^3 + \left(\frac{-A}{3}\right)^3 D\right] - A^2D^2 \\ &= \frac{4AD}{27}(A-D)^2 \geq 0. \end{aligned}$$

Sendo assim

$$\Phi = 0 \text{ se e somente se } A = D \quad \text{e} \quad \Phi > 0 \text{ se e somente se } A \neq D. \tag{3.8}$$

Note que se $A = D = l > 0$ e $B = C = \frac{-l}{3a}$, onde $a > 1$, então

$$\begin{aligned} r(x) &= l(1-x)^3 + 3\cdot\frac{-l}{3a}(1-x)^2x + 3\cdot\frac{-l}{3a}(1-x)x^2 + lx^3, \\ &= \frac{l}{a}[a(1-x)^3 - x(1-x)^2 - x^2(1-x) + x^3a], \\ &= \frac{l}{a}[(3a+1)x^2 - (3a+1)x + a]. \end{aligned}$$

Estudando a equação do segundo grau acima, vemos que para $a > 1$ ela não tem solução e, o menor valor que ela assumirá será $\frac{a-1}{4}$ e, portanto $r(x) \geq \frac{l(a-1)}{4a} > 0$, para todo $x \in [0,1]$.

■

Devido ao Teorema 3.1 e a Afirmação (3.8), seguindo os passos definidos no final da Seção 3.1.1, são fixados limites inferiores para as ordenadas de fronteira de Bézier de T como

$$m_4 \geq \frac{-l}{3a}, \text{ sendo } a > 1, \forall m_4 \in M_4. \tag{3.9}$$

Pelas Equações (3.5) obtemos

$$m_3 \geq \frac{1}{3}\left(-\frac{l}{3a} - \frac{l}{3a} + l\right) = \frac{l}{3}\left(1 - \frac{2}{3a}\right), \forall m_3 \in M_3.$$

Se além disso tivermos $m_2 \geq \frac{-l}{3a}, \forall m_2 \in M_2$, então por (3.6)

$$\begin{aligned} m_1 &\geq \frac{1}{3}\left[\frac{l}{3}\left(1 - \frac{2}{3a}\right) - \frac{l}{3a} - \frac{l}{3a}\right], \\ &\geq \frac{l}{3}\left[\frac{1}{3}\left(1 - \frac{2}{3a}\right) - \frac{1}{3a} - \frac{1}{3a}\right], \\ &\geq \frac{l}{9}\left[1 - \frac{8}{3a}\right]. \end{aligned}$$

As ordenadas de Bézier $a_{300} = b_{300} = c_{300}$ no centróide têm o valor da superfície interpolada em G, então $b_{300} \geq 0$ é necessário para garantir a não-negatividade da malha triangular de Bézier. Mais ainda, é também necessário que $m_1 \geq 0$, $\forall m_1 \in M_1$ pois, do contrário, valores negativos de m_1 deixariam a derivada parcial negativa em G ao longo das arestas correspondentes e o retalho de Bézier correspondente não satisfaria a não-negatividade.

Por (3.7) e pelo limitante inferior encontrado para os elementos de M_1, temos que

$$b_{300} = \frac{1}{3}(a_{201} + b_{201} + c_{201}) \geq \frac{1}{3}\left[\frac{3l}{9}\left(1 - \frac{8}{3a}\right)\right] = \frac{l}{9}\left(1 - \frac{8}{3a}\right).$$

Ou seja, para que b_{300} seja não-negativo, é preciso que

$$1 - \frac{8}{3a} \geq 0, \text{ ou seja } a \geq \frac{8}{3}.$$

Então se $\forall m \in M_2 \cup M_4$, $m \geq \frac{-l}{3a} \geq \frac{-l}{8}$ e, sabendo que $a \geq \frac{8}{3}$, conclui-se que

$$m_3 \geq \frac{l}{3}\left(1 - \frac{2}{3a}\right) \Rightarrow m_3 \geq \frac{l}{4}, \forall m_3 \in M_3$$

e

$$m \geq 0, \forall m \in M_1 \cup \{b_{300}\}.$$

Proposição 3.1. *Seja T um triângulo no plano cortado em três mini-triângulos, através de seu centróide. Seja a malha triangular cúbica de Bézier definida em cada um desses mini-triângulos, cujas ordenadas são $\{a_{ijk}\}$, $\{b_{ijk}\}$ e $\{c_{ijk}\}$, com $0 \leq i, j, k \leq 3$ e $i + j + k = 3$. Suponha que os três mini-triângulos de Bézier formem uma malha triangular Q em T que seja C^1. Se $\forall m \in M_5$, $m \geq l$, onde $l > 0$ e $\forall m \in M_2 \cup M_4$, $m \geq -\frac{l}{3a}$, com $a \geq \frac{8}{3}$, então $Q(x, y) \geq 0$, $\forall (x, y) \in T$.*

Demonstração: Considere a malha triangular de Bézier

$$S(u, v, w) = \sum_{\substack{i+j+k=3 \\ i,j,k \geq 0}} b_{ijk} \frac{3!}{i!\, j!\, k!} u^i v^j w^k$$

no mini-triângulo GV_2V_3 do triângulo da Figura 3.8.

Tome $P(0, v, w)$ onde $v + w = 1$, um ponto na aresta V_2V_3, oposta ao vértice G. O parâmetro t varia entre 0 e 1, do vértice G ao ponto P, e as coordenadas baricêntricas para um ponto do segmento GP podem ser escritas como $(1 - t, tv, tw)$, $0 \leq t \leq 1$. Veja a Figura 3.8 que representa esse mini-triângulo.

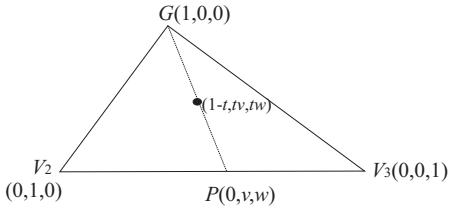

Figura 3.8 Mini-triângulo com o segmento GP.

Sendo assim, pode-se verificar que a curva na malha triangular cúbica de Bézier, S, ao longo do segmento de reta GP é dada por

$$S(1 - t, t(1 - w), tw) = A(w)B_0^3(t) + B(w)B_1^3(t) + C(w)B_2^3(t) + D(w)B_3^3(t). \tag{3.10}$$

De fato,

$S(1-t,t(1-w),tw) =$

$$\begin{aligned} & b_{300}B^3_{300}(1-t,t(1-w),tw) + b_{210}B^3_{210}(1-t,t(1-w),tw) + \\ + \; & b_{201}B^3_{201}(1-t,t(1-w),tw) + b_{120}B^3_{120}(1-t,t(1-w),tw) + \\ + \; & b_{111}B^3_{111}(1-t,t(1-w),tw) + b_{102}B^3_{102}(1-t,t(1-w),tw) + \\ + \; & b_{030}B^3_{030}(1-t,t(1-w),tw) + b_{021}B^3_{021}(1-t,t(1-w),tw) + \\ + \; & b_{012}B^3_{012}(1-t,t(1-w),tw) + b_{003}B^3_{003}(1-t,t(1-w),tw). \end{aligned}$$

E, sendo $B^3_{ijk}(u,v,w) = \frac{3!}{i!j!k!}u^i v^j w^k$, então

$S(1-t,t(1-w),tw) =$

$$\begin{aligned} & b_{300}(1-t)^3 + 3b_{210}(1-t)^2t(1-w) + 3b_{201}(1-t)^2tw + \\ + \; & 3b_{120}(1-t)t^2(1-w)^2 + 6b_{111}(1-t)t(1-w)tw) + 3b_{102}(1-t)t^2w^2 + \\ + \; & b_{030}t^3((1-w)^3 + 3b_{021}t^2(1-w)^2tw + 3b_{012}t(1-w)t^2w^2 + b_{003}t^3w^3. \end{aligned}$$

Usando que $B^3_i(t) = \frac{3!}{(3-i)!i!}(1-t)^{3-i}t^i$ e agrupando os termos semelhantes, obtem-se

$S(1-t,t(1-w),tw) =$

$$\begin{aligned} & b_{300}B^3_0(t) + [b_{210}((1-w) + b_{201}w]B^3_1(t) + \\ + \; & [b_{120}(1-w)^2 + 2b_{111}w(1-w) + b_{102}w^2]B^3_2(t) + \\ + \; & [b_{030}(1-w)^3 + 3b_{021}w(1-w)^2 + 3b_{012}(1-w)w^2 + b_{003}w^3]B^3_3(t). \end{aligned}$$

Denotando $A(w) = b_{300}$, $B(w) = b_{210}(1-w) + b_{201}w$, $C(w) = b_{120}(1-w)^2 + 2b_{111}w(1-w) + b_{102}w^2$ e $D(w) = b_{030}(1-w)^3 + 3b_{021}w(1-w)^2 + 3b_{012}(1-w)w^2 + b_{003}w^3$, tem-se que

$$S(1-t,t(1-w),tw) = A(W)B^3_0(t) + B(w)B^3_1(t) + C(w)B^3_2(t) + D(w)B^3_3(t)$$

e, para verificar sua não-negatividade, basta observar que $B^3_i(t) \geq 0$, $\forall i \in \{0,1,2,3\}$, e analisar os termos $A(w), B(w), C(w), D(w)$.

Assumindo $a \geq 8/3$ e voltando à discussão anterior, tem-se que $b_{300} \geq 0$, $b_{210} \geq 0$ e $b_{201} \geq 0$. Sendo assim $A(w)$ e $B(w)$ são não-negativos.

Desde que $b_{111} \geq -\frac{l}{8}$, $b_{120} \geq \frac{l}{4}$ e $b_{102} \geq \frac{l}{4}$, então

$$C(w) \geq \frac{l}{4}(1-w)^2 - 2\frac{l}{8}w(1-w) + \frac{l}{4}w^2 = \frac{l}{2}\left[\frac{1}{2} - w + \frac{3}{2}w^2\right] > 0, \forall w.$$

Para estudar $D(w)$ observe que supondo $b_{030} \geq 0$ e $b_{003} \geq 0$, além de $b_{012} \geq -\frac{l}{3a}$ e $b_{021} \geq -\frac{l}{3a}$, temos

$$D(w) \geq l(1-w)^3 - \frac{l}{a}w(1-w)^2 - \frac{l}{a}w^2(1-w) + lw^3.$$

E considerando o Teorema 3.1, podemos concluir que $D(w) \geq 0$.

Dessa forma a malha triangular S, que é composta pelas curvas cúbicas de Bézier univariadas ao longo do segmento GP (conforme Figura 3.8), é não-negativa. Usando o mesmo argumento, agora para os mini-triângulos com ordenadas de Bézier $\{a_{ijk}\}$ e $\{c_{ijk}\}$, estas são também não-negativas.

■

3.2 Construção da Superfície Interpolante

Dados pontos (x_i, y_i, z_i), com $z_i > 0$ e $i = 1, 2 \cdots, N$, tal que $(x_i, y_i) \neq (x_j, y_j)$ para $i \neq j$, descreve-se a construção de uma função $F(x, y)$, C^1, preservando a não-negatividade, com $F(x_i, y_i) = z_i$, $i = 1, 2, \cdots, N$. O processo de construção consiste em três etapas usuais para interpolação de pontos espalhados

1. O domínio Ω da função F é o invólucro convexo de $\{V_i = (x_i, y_i) : i = 1, \cdots, N\}$. Os pontos V_i, $i = 1, 2 \cdots, N$, são usados como vértices da triangulação do domínio Ω. A triangulação de Delaunay (conforme [3]) é usada para triangular o domínio Ω.

2. A estimação das derivadas parciais de primeira ordem, isto é F_x e F_y, em cada $V_i(x_i, y_i)$, para a superfície F é obtida usando o método desenvolvido em [4].

3. Para todo triângulo macro no domínio, uma malha triangular será gerada.

O terceiro passo começa com a subdivisão de cada triângulo macro do domínio em três mini-triângulos, através do seu centróide, e em seguida um triângulo cúbico de Bézier é construído em cada mini-triângulo. A determinação das ordenadas de Bézier destes três mini-triângulos, $\{a_{ijk}\}$, $\{b_{ijk}\}$ e $\{c_{ijk}\}$, é descrita como segue.

Após as derivadas parciais de primeira ordem, F_x e F_y, em cada vértice $V_i(x_i, y_i)$ no domínio triangulado Ω, serem estimadas (por [4]), em cada retalho Q no triângulo macro, a derivada direcional em seu vértice V_i na direção da aresta $\mathbf{d}_{ij}$, de V_i para V_j, é calculada por

$$D_{\mathbf{d}_{ij}}(V_i) = (x_j - x_i)\frac{\partial F}{\partial x}(V_i) + (y_j - y_i)\frac{\partial F}{\partial y}(V_i).$$

Baseado nos pontos dados, as ordenadas dos vértices de cada triângulo macro são determinadas. Observando que no triângulo macro T da Figura 3.7, temos que:

$$a_{030} = c_{003} = F(V_1) = z_1,$$
$$b_{030} = a_{003} = F(V_2) = z_2,$$
$$c_{030} = b_{003} = F(V_3) = z_3$$

e, tendo as derivadas direcionais estimadas em cada vértice, as ordenadas de Bézier da fronteira, representadas pelo conjunto M_4, são determinadas usando (3.1). No entanto, estas ordenadas determinadas podem não garantir que a malha resultante seja não-negativa. Para garantir isso é preciso impor condições nestas ordenadas de fronteira, isto é,

$$m_4 \geq -\frac{l}{8}, \forall m_4 \in M_4,$$

com $l = min\{F(V_1), F(V_2), F(V_3)\}$, como na Proposição 3.1. Para alcançá-la, a derivada parcial de primeira ordem em V_i, $i = 1,2,3$, deve ser modificada, se necessário. A modificação das derivadas F_x e F_y no vértice V_i é feita inserindo um escalar em cada um deles como um fator positivo α, sendo $\alpha \leq 1$, levando em consideração todos os retalhos triangulares no triângulo macro dividindo aquele vértice. Vejamos como se dará esse procedimento.

Tome O para ser o vértice em seu domínio triangulado Ω e π_i, $i = 1,2,\cdots,k$, para ser o triângulo macro na triangulação que tem O como um vértice.

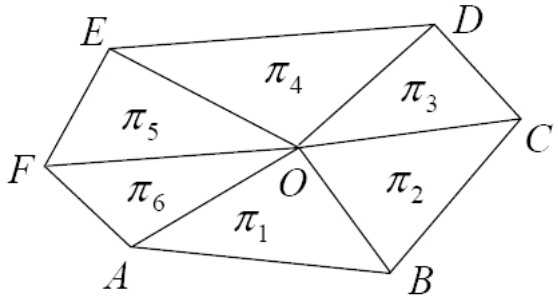

Figura 3.9 Domínio Triangulado Ω.

Considere o triângulo π_1 (veja Figura 3.9) e o limite inferior $-\frac{l_{\pi_1}}{8}$ onde

$$l_{\pi_1} = min\{F(O), F(A), F(B)\}.$$

Denote as derivadas direcionais em O na direção das arestas OA e OB por $D_{d_{OA}}$ e $D_{d_{OB}}$, respectivamente. Os fatores escalares α_{OA} e α_{OB} são definidos como segue.

Se $F(O) + \frac{1}{3}D_{d_{OA}} \geq -\frac{l_{\pi_1}}{8}$, então $\alpha_{OA} = 1$, caso contrário, α_{OA} é determinado pela equação

$$F(O) + \alpha_{OA}\frac{1}{3}D_{d_{OA}} = -\frac{l_{\pi_1}}{8}.$$

Similarmente, se $F(O) + \frac{1}{3}D_{d_{OB}} \geq -\frac{l_{\pi_1}}{8}$, então $\alpha_{OB} = 1$, caso contrário, α_{OB} é definido pela equação

$$F(O) + \alpha_{OB}\frac{1}{3}D_{d_{OB}} = -\frac{l_{\pi_1}}{8}.$$

Daí define-se $\alpha_{\pi_1} = \min\{\alpha_{OA}, \alpha_{OB}\}$. Usando o mesmo método, α_{π_i}, $i = 2,3,\cdots,k$, são encontrados. Para todas as ordenadas de Bézier adjacentes a O satisfazerem as condições de negatividade, escolhemos

$$\alpha_0 = \min\{\alpha_{\pi_1}, \alpha_{\pi_2}, \cdots, \alpha_{\pi_k}\}.$$

Se $\alpha_0 < 1$, então as derivadas parciais primeiras em O são multiplicadas pelo fator α_0 e as ordenadas de Bézier adjacentes a O são determinadas de acordo com isso. O processo citado anteriormente é repetido em todos os vértices V_i no domínio Ω. Então todas as ordenadas de Bézier ao longo das arestas dos triângulos macros podem ser determinadas da forma citada.

Pela propriedade de continuidade C^1, ordenadas em M_3 serão determinadas usando as relações em (3.5). Em seguida, as ordenadas de Bézier internas, M_2, são determinadas usando (3.3), (3.4) e uma equação análoga para a malha a. A Proposição 3.1 impõe um limite inferior nessas ordenadas de Bézier internas. Aqui, já que as ordenadas de Bézier da fronteira do triângulo macro são fixadas, poderia se usar os valores atuais deles para relaxar o limite nas ordenadas de Bézier internas sugerido na Proposição 3.1. Observe que para garantir que $C(w)$ em (3.10) seja não-negativo, é necessário que

$$b_{111} \geq -\min\{b_{120}, b_{102}\} \tag{3.11}$$

e, similarmente

$$\begin{aligned} a_{111} &\geq -\min\{a_{120}, a_{102}\}, \\ c_{111} &\geq -\min\{c_{120}, c_{102}\}. \end{aligned} \tag{3.12}$$

Ainda mais, desde que a_{201}, b_{201} e c_{201} têm que ser não-negativos, por (3.6) temos que

$$\begin{aligned} a_{102} + b_{111} + a_{111} &\geq 0, \\ b_{102} + c_{111} + b_{111} &\geq 0, \\ c_{102} + a_{111} + c_{111} &\geq 0. \end{aligned} \tag{3.13}$$

Para garantir (3.11), (3.12) e (3.13) é suficiente ter

$$\begin{aligned} a_{111} &\geq -\tfrac{1}{2}\min\{a_{120}, a_{102}\}, \\ b_{111} &\geq -\tfrac{1}{2}\min\{b_{120}, b_{102}\}, \\ c_{111} &\geq -\tfrac{1}{2}\min\{c_{120}, c_{102}\}. \end{aligned} \tag{3.14}$$

De fato, se (3.14) é verdade, claramente que (3.11) e (3.12) também serão.

Para verificarmos que (3.13) é verdadeiro, suponhamos primeiramente que $\min\{a_{120}, a_{102}\} = a_{102}$. Dessa forma

$$a_{111} \geq -\frac{1}{2}\min\{a_{120}, a_{102}\} \Rightarrow a_{102} + b_{111} + a_{111} \geq \frac{1}{2}a_{102} + b_{111}.$$

Agora temos

$$b_{111} \geq -\frac{1}{2}\min\{b_{120}, b_{102}\} \Rightarrow a_{102} + b_{111} + a_{111} \geq \frac{1}{2}(a_{102} - b_{102}),$$

para o caso de $\min\{b_{120}, b_{102}\} = b_{102}$.

Pela continuidade C^0, $a_{102} = b_{120}$ e, portanto

$$a_{102} + b_{111} + a_{111} \geq \frac{1}{2}(b_{120} - b_{102}),$$

e este último, pela hipótese de b_{102} ser o mínimo, é não-negativo. Do contrário, tem-se

$$a_{102} + b_{111} + a_{111} \geq \frac{1}{2}(a_{102} - b_{120}) = \frac{1}{2}(a_{102} - a_{102}) = 0.$$

De forma similar, supondo que $\min\{a_{120}, a_{102}\} = a_{102}$, conclui-se que $a_{102} + b_{111} + a_{111} \geq 0$.

Portanto o resultado é alcançado para a primeira Equação de (3.14) e facilmente pode-se adaptar o raciocínio para concluir as outras duas Equações. Sendo assim, para que (3.11), (3.12) e (3.13) sejam verdadeiros, é suficiente que (3.14) o seja.

Note que, por causa da Proposição 3.1, estes limites inferiores em (3.14) são menores que ou iguais a $-\frac{l}{8}$, onde $l = \min\{a_{003}, b_{003}, c_{003}\}$. Se os valores internos, a_{111}, b_{111} e c_{111} não satisfazem (3.14) , então eles são incrementados até o limite correspondente. Isto será suficiente para garantir que as ordenadas de controle $A(w), B(w), C(w), D(w)$ da curva cúbica de Bézier em (3.10) sejam não-negativas e então o triângulo de Bézier correspondente é não-negativo.

De fato, (3.13) implica, devido a (3.6), em $A(w) \geq 0$, $B(w) \geq 0$. Vejamos $C(w)$:

$$C(w) \geq (1-w)^2 b_{120} - 2w(1-w)b_{120} + w^2 b_{120} = b_{120} \geq 0,$$

por ser elemento de M_3.

O último termo por sua vez, $D(w)$, depende das ordenadas b_{030}, b_{021}, b_{012} e b_{003}, que são todas não-negativas, pela construção aqui apresentada.

Finalmente, $a_{201}, b_{201}, c_{201}$ e a_{300} são obtidos via (3.6) e (3.7), uma vez que já determinamos os elementos dos conjuntos M_2 e M_3. Quando uma ordenada interna de Bézier for modificada, a continuidade C^1 ao longo da fronteira do triângulo macro é mantida, por modificar a ordenada interna de Bézier correspondente do triângulo macro adjacente, de acordo com (3.2), e recalculando as ordenadas de Bézier que são dependentes na ordenada interna modificada. Esta adaptação para manter a continuidade C^1 não atrapalhará a propriedade de não-negatividade por causa de (3.2) e o limite inferior em (3.9).

A malha triangular no triângulo macro, consistindo de três triângulos de Bézier com suas ordenadas $\{a_{ijk}\}$, $\{b_{ijk}\}$ e $\{c_{ijk}\}$, então gerado, é não-negativo e é C^1 ao longo da curva de fronteira comum com a malha adjacente.

Denote a malha triangular construída desta forma no triângulo macro T como Q_T. Então a superfície interpolante F, que tem a continuidade C^1 e preserva a não-negatividade, pode ser definida no domínio triangulado como $F|T = Q_T$, para todo triângulo macro T no domínio.

Capítulo 4

Malha Quártica num Tetraedro

Este Capítulo tem como objetivo apresentar uma nova metodologia de construção de superfícies interpolantes, não-negativas, com continuidade C^1, uma vez dados os pontos trivariados espalhados no espaço. Vejamos de que forma esse trabalho se desenvolve.

Nas Seções 4.1 e 4.2 temos a definição do funcional polinomial quártico e um novo Teorema é apresentado, 4.1, que trata da continuidade C^1 entre dois tetraedros quárticos adjacentes. Na Seção 4.3 vemos como a derivada direcional pode determinar os valores de controle de uma malha quártica tetraédrica, enunciado no Lema 4.1. Num domínio tetrangulado se faz necessário estudar a continuidade entre tetraedros macros adjacentes, e esse resultado fica estabelecido no Teorema 4.2, que pode ser visto na Secão 4.4. Nesta Seção encontramos também a Subseção 4.4.1 que trata da propriedade da localidade, importante característica dos métodos locais de interpolação. Nas Seções 4.5 e 4.6 temos a solução do problema de determinação de todos os valores de controle de uma malha quártica tetraédrica, uma vez que a idéia usada por [1] não foi suficiente para o caso aqui estudado. Dessa forma, uma nova idéia é apresentada, a redução do grau da malha numa face do tetraedro, e o problema é então resolvido. Na Secão 4.7 um tetraedro macro é particionado em quatro mini-tetraedros e malhas quárticas são estabelecidas, juntamente com relações entre suas ordenadas de Bézier, a fim de atingir a continuidade C^1 entre as malhas. Na Secão 4.8 temos o estudo das condições suficientes de não-negatividade de uma malha quártica de Bézier, estabelecidas no Teorema principal desse trabalho, o 4.3. A Secão 4.9 conclui este estudo, com a apresentação da construção da superfície interpolante, não-negativa e com continuidade C^1.

4.1 Introdução

Considere um tetraedro T, não degenerado, no espaço com vértices conhecidos V_1, V_2, V_3 e V_4. Qualquer ponto $V \in \mathbb{E}^3$ pode ser escrito de forma única como

$$V = uV_1 + vV_2 + wV_3 + rV_4,$$

onde $u+v+w+r = 1$. Dizemos que u, v, w, r são as coordenadas baricêntricas de V com relação a T.

Seja S um funcional polinomial, $S : \mathbb{E}^3 \to \mathbb{R}$, quártico ($n = 4$) de Bézier definido em T como

$$S(u,v,w,r) = b^4(u,v,w,r) = \sum_{\substack{i+j+k+p=4 \\ i,j,k,p\geq 0}} b_{ijkp} \frac{4!}{i!\,j!k!p!} u^i v^j w^k r^p,$$

onde $u+v+w+r=1, \quad u,v,w,r \geq 0$, ou seja para pontos no interior e na fronteira de T. Ou ainda, utilizando os polinômios de Bernstein, temos

$$S(\mathbf{u}) = b^4(\mathbf{u}) = \sum_{\substack{i+j+k+p=4 \\ i,j,k,p\geq 0}} b_{ijkp} B^4_{\mathbf{i}}(\mathbf{u}), \text{ onde } B^4_{\mathbf{i}}(\mathbf{u}) = \frac{4!}{i!\,j!\,k!\,p!} u^i v^j w^k r^p,$$

sendo $\mathbf{u} = (u,v,w,r)$, com $\mathbf{i} = ijkp$ e b_{ijkp} denotando os valores de controle, associados a pontos igualmente espaçados de T. Se assumirmos que b_{4000} é o valor de controle associado a V_1, b_{0400} é associado a V_2, b_{0040} é associado a V_3 e b_{0004} é associado a V_4, então b_{ijkp} é o valor de controle associado à posição $\frac{1}{4}[iV_1 + jV_2 + kV_3 + pV_4]$.

Considere $\mathbf{i} = ijkp$ todas as possibilidades de índices para os valores de controle $b_{\mathbf{i}}$ numa tetraedrização. São eles:

$\{0004, 0103, 0202, 0301, 0400, 0013, 0112, 0211, 0310, 0022, 0121, 0220, 0031, 0130, 0040,$
$1003, 1102, 1201, 1300, 1012, 1111, 1210, 1021, 1120, 1030, 2002, 2101, 2200, 2011, 2110,$
$2020, 3001, 3100, 3010, 4000\}$.

Sendo assim, usando a função abaixo

$$f(ijkp) = \begin{cases} 5k - \frac{k(k-1)}{2} + j & \text{se } i = 0 \\ 15 + 4k - \frac{k(k-1)}{2} + j & \text{se } i = 1 \\ 25 + 3k - \frac{k(k-1)}{2} + j & \text{se } i = 2 \\ 31 + 2k + j & \text{se } i = 3 \\ 34 & \text{se } i = 4 \end{cases} \tag{4.1}$$

nosso novo conjunto de índices será $\{0, 1, 2, \cdots, 33, 34\}$.

Dessa forma, nossa malha tetraédrica quártica possui os valores de controle $b_{f(\mathbf{i})}$ (observe as Figuras 4.1 e 4.2). Daqui em diante as malhas quárticas tetraédricas serão sempre representadas com essa nomenclatura.

4.2 Continuidade C^1 entre Tetraedros Quárticos Adjacentes

Considere dois tetraedros quárticos de Bézier adjacentes, $\Delta V_1V_2V_3V_4$, $\Delta W_1W_2W_3W_4$, com face adjacente comum sendo $V_2V_3V_4$ e $W_2W_3W_4$, e $V_4 = W_4, V_2 = W_3$, $V_3 = W_2$ (veja Figura 4.3). Suponha que os tetraedros quárticos de Bézier tenham nesses dois tetraedros ordenadas de Bézier $b_{\mathbf{i}}$ e $a_{\mathbf{i}}$, para $\mathbf{i} \in \{0, 1, \cdots, 34\}$.

Como os tetraedros possuem a mesma face fronteira, temos que

$$\begin{array}{ccccc} b_0 = a_0, & b_5 = a_1, & b_9 = a_2, & b_{12} = a_3, & b_{14} = a_4, \\ b_1 = a_5, & b_2 = a_9, & b_3 = a_{12}, & b_4 = a_{14}, & b_8 = a_{13}, \\ b_{11} = a_{11}, & b_{13} = a_8, & b_{10} = a_7, & b_7 = a_{10}, & b_6 = a_6. \end{array} \tag{4.2}$$

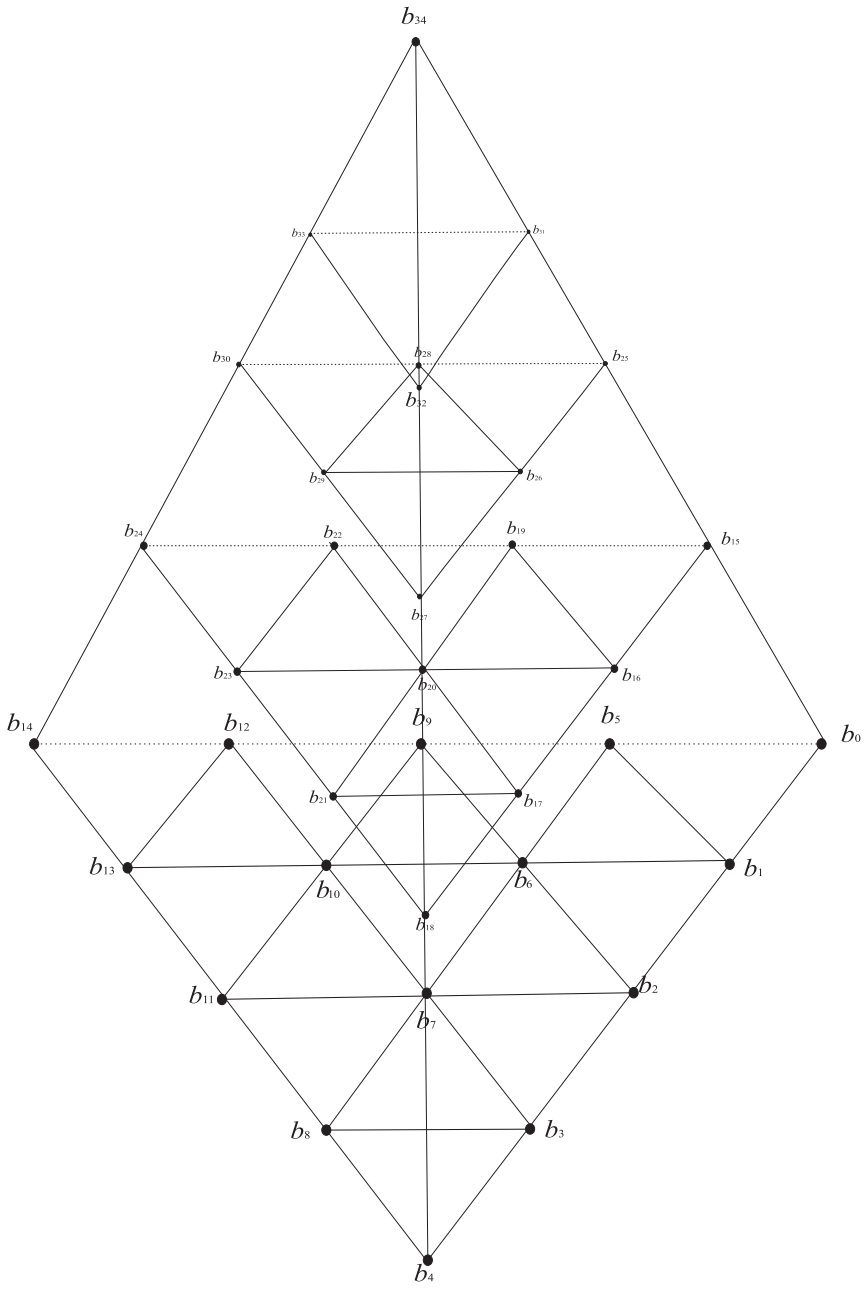

Figura 4.1 Malha quártica tetraédrica de Bézier.

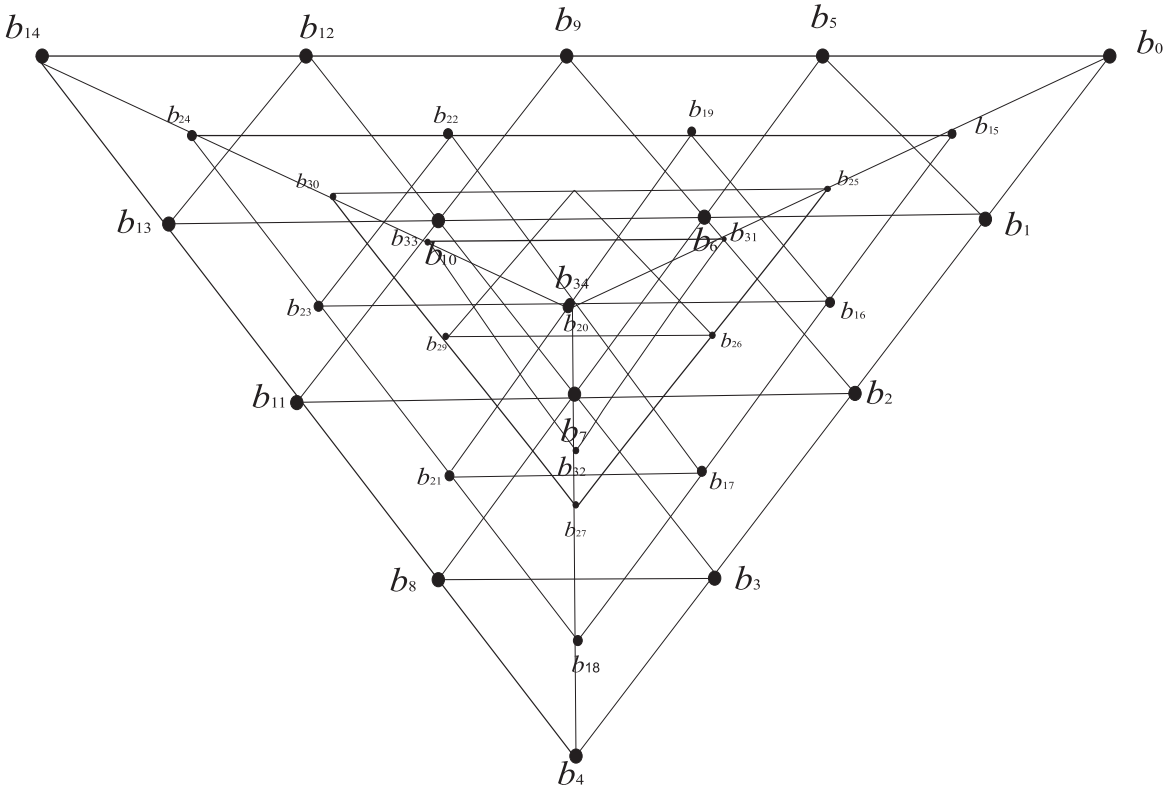

Figura 4.2 Malha quártica tetraédrica de Bézier (vista de cima).

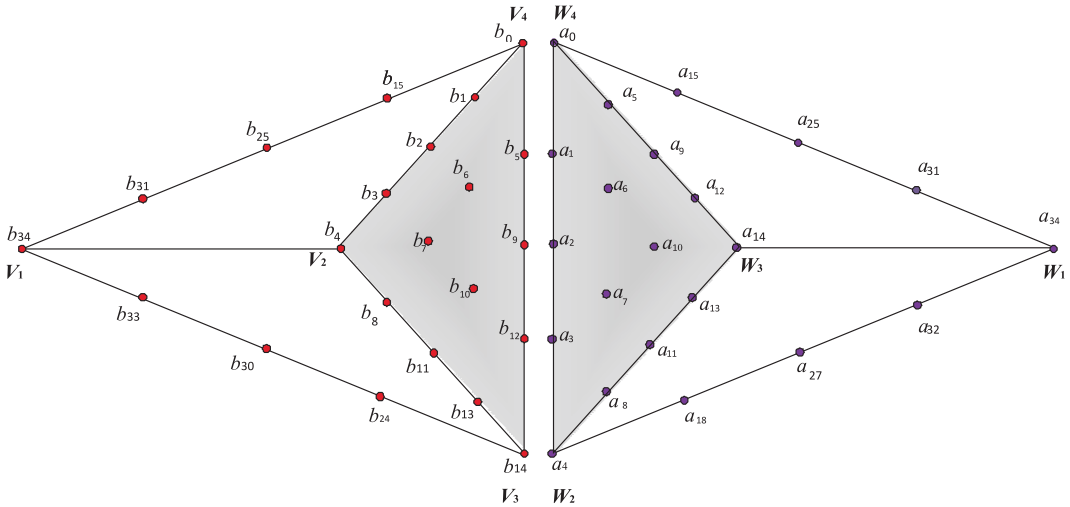

Figura 4.3 Duas malhas tetraédricas de Bézier adjacentes.

Sendo assim, as condições necessárias e suficientes para atingir a continuidade C^1 entre as duas malhas podem ser estabelecidas no Teorema abaixo.

Teorema 4.1. *Considere $F : T_1 \cup T_2 \to \mathbb{R}$ funcional composto quártico, sendo dois tetraedros com uma face comum, T_1 e T_2, tal que*

$$F(\mathbf{u}) = \begin{cases} S_1(\mathbf{u}) & se \quad \mathbf{u} \in T_1 \\ S_2(\mathbf{u}) & se \quad \mathbf{u} \in T_2, \end{cases}$$

onde S_1 e S_2 são dois funcionais quárticos de Bézier definidos sobre T_1 e T_2, respectivamente. Suponha que na face comum, $V_2V_3V_4$, valem as propriedades:

1. *S_1 e S_2 têm os mesmos pontos e valores de controle,*
2. *S_1 e S_2 têm as mesmas derivadas nos vértices V_2, V_3 e V_4.*

Então podemos concluir que F é C^1 na face comum se e somente se as seguintes Equações são satisfeitas:

$$\begin{array}{ll} a_{15} = \alpha b_{15} + \beta b_1 + \gamma b_5 + \lambda b_0, & a_{16} = \alpha b_{19} + \beta b_6 + \gamma b_9 + \lambda b_5, \\ a_{17} = \alpha b_{22} + \beta b_{10} + \gamma b_{12} + \lambda b_9, & a_{18} = \alpha b_{24} + \beta b_{13} + \gamma b_{14} + \lambda b_{12}, \\ a_{19} = \alpha b_{16} + \beta b_2 + \gamma b_6 + \lambda b_1, & a_{20} = \alpha b_{20} + \beta b_7 + \gamma b_{10} + \lambda b_6, \\ a_{21} = \alpha b_{23} + \beta b_{11} + \gamma b_{13} + \lambda b_{10}, & a_{22} = \alpha b_{17} + \beta b_3 + \gamma b_7 + \lambda b_2, \\ a_{23} = \alpha b_{21} + \beta b_8 + \gamma b_{11} + \lambda b_7, & a_{24} = \alpha b_{18} + \beta b_4 + \gamma b_8 + \lambda b_3, \end{array} \tag{4.3}$$

onde

$$W_1 = \alpha V_1 + \beta V_2 + \gamma V_3 + \lambda V_4, \text{ sendo } \alpha + \beta + \gamma + \lambda = 1. \tag{4.4}$$

Demonstração: Considere $\mathbf{p}$ um ponto do domínio (Figura 4.3), ou seja $\mathbf{p} = \bar{u}W_1 + \bar{v}W_2 + \bar{w}W_3 + \bar{r}W_4$, onde $\bar{u} + \bar{v} + \bar{w} + \bar{r} = 1$. Usando (4.4) e a igualdade entre vértices da fronteira dos tetraedros das malhas, podemos escrever $\mathbf{p}$ da seguinte forma:

$$\mathbf{p} = \bar{u}(\alpha V_1 + \beta V_2 + \gamma V_3 + \lambda V_4) + \bar{v}V_3 + \bar{w}V_2 + \bar{r}V_4,$$

$$\mathbf{p} = (\alpha\bar{u})V_1 + (\beta\bar{u} + \bar{w})V_2 + (\gamma\bar{u} + \bar{v})V_3 + (\lambda\bar{u} + \bar{r})V_4.$$

Assim, se $\mathbf{u} = (u, v, w, r)$ e seu equivalente $\bar{\mathbf{u}} = (\bar{u}, \bar{v}, \bar{w}, \bar{r})$, então

$$u = \alpha\bar{u}, v = \beta\bar{u} + \bar{w}, w = \gamma\bar{u} + \bar{v}, r = \lambda\bar{u} + \bar{r}$$

e, para pontos na borda comum entre as malhas dos tetraedros temos $u = 0 = \bar{u}$, ou seja $v = \bar{w}$, $w = \bar{v}$ e $r = \bar{r}$.

Da mesma forma, para um vetor $\mathbf{d} = (d, e, f, g)$, onde $d + e + f + g = 0$,

$$d = \alpha\bar{d}, e = \beta\bar{d} + \bar{f}, f = \gamma\bar{d} + \bar{e}, g = \lambda\bar{d} + \bar{g}. \tag{4.5}$$

Desenvolvendo a derivada na borda comum, na malha b encontramos

$$D_{\mathbf{d}}(b(\mathbf{u})) = d\frac{\partial b}{\partial u}(\mathbf{u}) + e\frac{\partial b}{\partial v}(\mathbf{u}) + f\frac{\partial b}{\partial w}(\mathbf{u}) + g\frac{\partial b}{\partial r}(\mathbf{u}),$$

$$\begin{aligned} D_{\mathbf{d}}(b(\mathbf{u})) &= 4\Bigg[d\sum_{|i|=3} b_{\mathbf{i}+\mathbf{e}_1} B^3_{\mathbf{i}}(\mathbf{u}) + e\sum_{|i|=3} b_{\mathbf{i}+\mathbf{e}_2} B^3_{\mathbf{i}}(\mathbf{u}) + f\sum_{|i|=3} b_{\mathbf{i}+\mathbf{e}_3} B^3_{\mathbf{i}}(\mathbf{u}) + \\ &\quad g\sum_{|i|=3} b_{\mathbf{i}+\mathbf{e}_4} B^3_{\mathbf{i}}(\mathbf{u})\Bigg], \end{aligned}$$

$$D_{\mathbf{d}}(b(\mathbf{u})) = 4\sum_{|i|=3} B^3_{\mathbf{i}}(\mathbf{u})\left(db_{\mathbf{i}+\mathbf{e}_1} + eb_{\mathbf{i}+\mathbf{e}_2} + fb_{\mathbf{i}+\mathbf{e}_3} + gb_{\mathbf{i}+\mathbf{e}_4}\right).$$

Utilizando a função de índices $f(\mathbf{i})$ definida em (4.1), temos a seguinte igualdade:

$$\begin{aligned} D_{\mathbf{d}}(b(\mathbf{u})) &= 4[v^3(db_{18} + eb_4 + fb_8 + gb_3) + w^3(db_{24} + eb_{13} + fb_{14} + gb_{12}) + \\ &\quad + r^3(db_{15} + eb_1 + fb_5 + gb_0) + 3v^2w(db_{21} + eb_8 + fb_{11} + gb_7) + \\ &\quad + 3v^2r(db_{17} + eb_3 + fb_7 + gb_2) + 3vw^2(db_{23} + eb_{11} + fb_{13} + gb_{10}) + \\ &\quad + 3w^2r(db_{22} + eb_{10} + fb_{12} + gb_9) + 3vr^2(db_{16} + eb_2 + fb_6 + gb_1) + \\ &\quad + 3vr^2(db_{16} + eb_2 + fb_6 + gb_1) + 3wr^2(db_{19} + eb_6 + fb_9 + gb_5) + \\ &\quad + 6vwr(db_{20} + eb_7 + fb_{10} + gb_6)], \end{aligned}$$

Por outro lado, pela malha a e as igualdades de (4.2) temos

$$\begin{aligned} D_{\bar{\mathbf{d}}}(a(\mathbf{u})) &= 4[w^3(\bar{d}a_{18} + \bar{e}b_{14} + \bar{f}a_8 + \bar{g}b_{12}) + v^3(\bar{d}a_{24} + \bar{e}a_{13} + \bar{f}b_4 + \bar{g}a_{12}) + \\ &\quad + r^3(\bar{d}a_{15} + \bar{e}b_5 + \bar{f}a_5 + \bar{g}b_0) + 3w^2v(\bar{d}a_{21} + \bar{e}a_8 + \bar{f}a_{11} + \bar{g}a_7) + \\ &\quad + 3w^2r(\bar{d}a_{17} + \bar{e}b_{12} + \bar{f}a_7 + \bar{g}b_9) + 3wv^2(\bar{d}a_{23} + \bar{e}a_{11} + \bar{f}a_{13} + \bar{g}a_{10}) + \\ &\quad + 3v^2r(\bar{d}a_{22} + \bar{e}a_{10} + \bar{f}a_{12} + \bar{g}a_9) + 3wr^2(\bar{d}a_{16} + \bar{e}b_9 + \bar{f}a_6 + \bar{g}b_5) + \\ &\quad + 3wr^2(\bar{d}a_{16} + \bar{e}b_9 + \bar{f}a_6 + \bar{g}b_5) + 3v^2r(\bar{d}a_{19} + \bar{e}a_6 + \bar{f}a_9 + \bar{g}a_5) + \\ &\quad + 6wvr(\bar{d}a_{20} + \bar{e}a_7 + \bar{f}a_{10} + \bar{g}a_6)], \end{aligned}$$

Usando que na borda comum, pela continuidade C^1, temos a derivada coincidindo, então

$$D_{\mathbf{d}}b(\mathbf{u}) = D_{\bar{\mathbf{d}}}a(\mathbf{u}).$$

Logo, por (4.5) e (4.2) a igualdade anterior é equivalente a

$$\begin{aligned} &\bar{d}\{v^3[\alpha b_{18} - a_{24} + \beta b_4 + \gamma b_8 + \lambda b_3] + w^3[\alpha b_{24} - a_{18} + \beta b_{13} + \gamma b_{14} + \lambda b_{12}] + \\ &+ r^3[\alpha b_{15} - a_{15} + \beta b_1 + \gamma b_5 + \lambda b_0] + 3v^2w[\alpha b_{21} - a_{23} + \beta b_8 + \gamma b_{11} + \lambda b_7] + \\ &+ 3v^2r[\alpha b_{17} - a_{22} + \beta b_3 + \gamma b_7 + \lambda b_2] + 3vw^2[\alpha b_{23} - a_{21} + \beta b_{11} + \gamma b_{13} + \lambda b_{10}] + \\ &+ 3rw^2[\alpha b_{22} - a_{17} + \beta b_{10} + \gamma b_{12} + \lambda b_9] + 3vr^2[\alpha b_{16} - a_{19} + \beta b_2 + \gamma b_6 + \lambda b_1] + \end{aligned}$$

$$+3wr^2[\alpha b_{19} - a_{16} + \beta b_6 + \gamma b_9 + \lambda b_5] + 6vrw[\alpha b_{20} - a_{20} + \beta b_7 + \gamma b_{10} + \lambda b_6]\} = 0.$$

Se $\bar{d} = 0$, imediatamente a igualdade acima se verifica. Do contrário, como essa igualdade tem que ser verdade para todo $v, w, r \in [0,1]$, sendo $v + w + r = 1$, então (4.3) é válida, como desejávamos.

■

Se associarmos as ordenadas de Bézier b_{ijkp} com $(x_{ijkp}, y_{ijkp}, z_{ijkp})$, para $0 \leq i, j, k, p \leq 4$, onde

$$(x_{ijkp}, y_{ijkp}, z_{ijkp}) = \frac{1}{4}(iV_1 + jV_2 + kV_3 + pV_4),$$

assim como para obter $\mathbf{b}_{ijkp}$, com $\mathbf{b}_{ijkp} = (x_{ijkp}, y_{ijkp}, z_{ijkp}, b_{ijkp})$, os pontos $\mathbf{a}_{ijkp}$ são definidos similarmente, então as condições acima têm uma interpretação geométrica, isto é os cinco pontos em cada conjunto abaixo têm que pertencer a um mesmo hiperplano

$$\{\mathbf{a}_{15}, \mathbf{b}_{15}, \mathbf{b}_1, \mathbf{b}_5, \mathbf{b}_0\}, \quad \{\mathbf{a}_{16}, \mathbf{b}_{19}, \mathbf{b}_6, \mathbf{b}_9, \mathbf{b}_5\},$$
$$\{\mathbf{a}_{17}, \mathbf{b}_{22}, \mathbf{b}_{10}, \mathbf{b}_{12}, \mathbf{b}_9\}, \quad \{\mathbf{a}_{18}, \mathbf{b}_{24}, \mathbf{b}_{13}, \mathbf{b}_{14}, \mathbf{b}_{12}\},$$
$$\{\mathbf{a}_{19}, \mathbf{b}_{16}, \mathbf{b}_2, \mathbf{b}_6, \mathbf{b}_1\}, \quad \{\mathbf{a}_{20}, \mathbf{b}_{20}, \mathbf{b}_7, \mathbf{b}_{10}, \mathbf{b}_6\},$$
$$\{\mathbf{a}_{21}, \mathbf{b}_{23}, \mathbf{b}_{11}, \mathbf{b}_{13}, \mathbf{b}_{10}\}, \quad \{\mathbf{a}_{22}, \mathbf{b}_{17}, \mathbf{b}_3, \mathbf{b}_7, \mathbf{b}_2\},$$
$$\{\mathbf{a}_{23}, \mathbf{b}_{21}, \mathbf{b}_8, \mathbf{b}_{11}, \mathbf{b}_7\}, \quad \{\mathbf{a}_{24}, \mathbf{b}_{18}, \mathbf{b}_4, \mathbf{b}_8, \mathbf{b}_3\}.$$

Sendo assim, a posição relativa de $\mathbf{a}_{15}$ com respeito ao tetraedro $\mathbf{b}_{15}, \mathbf{b}_1, \mathbf{b}_5, \mathbf{b}_0$ é a mesma posição relativa do ponto W_1 com respeito a V_1, V_2, V_3 e V_4, e assim também para os outros conjuntos (análogo ao caso bivariado).

4.3 Determinação de Valores de Controle pela Derivada Direcional

Considere uma malha tetraédrica com valores de controle $b_{\mathbf{i}}$ (Figura 4.4) e as ordenadas de Bézier vizinhas aos vértices do tetraedro V_2, V_3 e V_4, distribuídas no conjunto

$$\{b_1, b_3, b_5, b_8, b_{12}, b_{13}, b_{15}, b_{18}, b_{24}\}, \tag{4.6}$$

onde $b_0 = b(V_4), b_4 = b(V_2), b_{14} = b(V_3)$.

Como determinar as ordenadas de Bézier desconhecidas estabelecidas nesse conjunto? O Lema a seguir responde esta pergunta.

Lema 4.1. *Considere um tetraedro T no espaço com vértices V_1, V_2, V_3 e V_4 e um funcional polinomial quártico de Bézier definido em T, $S : \mathbb{E}^3 \to \mathbb{R}$. Considere o vértice V_i, a aresta V_iV_j e a derivada direcional na direção de d_{ij} , avaliada nesse vértice, $D_{ij}(V_i)$. Então a ordenada de Bézier associada ao ponto de controle vizinho a V_i pela aresta V_iV_j é determinada por $D_{ij}(V_i)$.*

Demonstração: De fato, considere os vetores $\mathbf{d}_{21}, \mathbf{d}_{23}, \mathbf{d}_{24}, \mathbf{d}_{31}, \mathbf{d}_{32}, \mathbf{d}_{34}, \mathbf{d}_{41}, \mathbf{d}_{42}, \mathbf{d}_{43}$, onde $\mathbf{d}_{ij} = \overrightarrow{V_i V_j}$, e considere a notação $D_{\mathbf{d}_{ij}} \equiv D_{ij}$.

A derivada direcional de S, na direção de $\mathbf{n}$, é definida por

$$D_{\mathbf{n}}(S(\mathbf{u})) = d\frac{\partial S}{\partial u}(\mathbf{u}) + e\frac{\partial S}{\partial v}(\mathbf{u}) + f\frac{\partial S}{\partial w}(\mathbf{u}) + g\frac{\partial S}{\partial r}(\mathbf{u}), \tag{4.7}$$

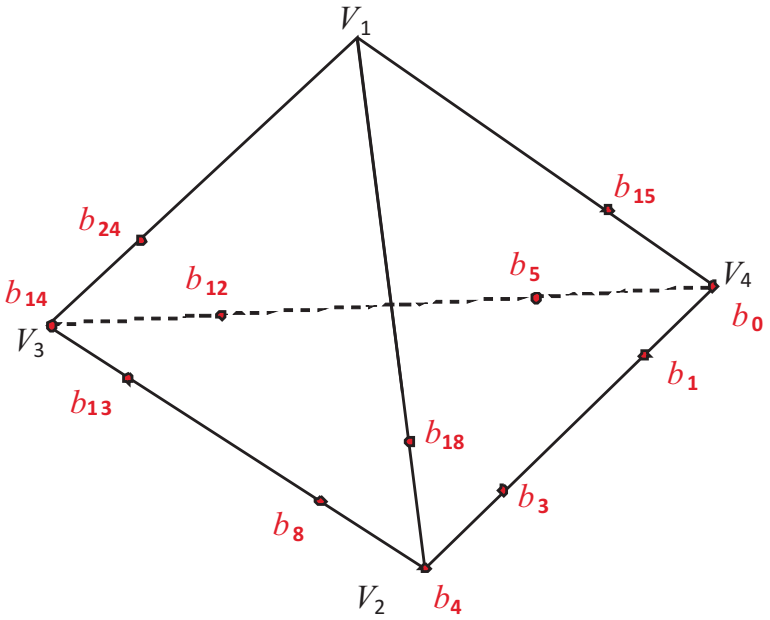

Figura 4.4 Malha com alguns valores de controle associados a seus pontos de controle.

onde $\mathbf{n}$ é o vetor (d, e, f, g), que são as coordenadas com relação a V_1, V_2, V_3, V_4 , tal que $d + e + f + g = 0$.

Sendo assim, desenvolvendo a expressão (4.7) encontramos

$$D_{\mathbf{n}}(\mathbf{b}(\mathbf{u})) = 4\sum_{|i|=3} B^3_{\mathbf{i}}(\mathbf{u})\left(db_{\mathbf{i}+\mathbf{e}_1} + eb_{\mathbf{i}+\mathbf{e}_2} + fb_{\mathbf{i}+\mathbf{e}_3} + gb_{\mathbf{i}+\mathbf{e}_4}\right),$$

e substituindo $\mathbf{u}$ em cada um dos vértices do tetraedro encontramos as igualdades

$$D_{21}(V_2) = 4(b_{18} - b_4), \quad D_{23}(V_2) = 4(b_8 - b_4), \quad D_{24}(V_2) = 4(b_3 - b_4),$$
$$D_{31}(V_3) = 4(b_{24} - b_{14}), \quad D_{32}(V_3) = 4(b_{13} - b_{14}), \quad D_{34}(V_3) = 4(b_{12} - b_{14}),$$
$$D_{41}(V_4) = 4(b_{15} - b_0), \quad D_{42}(V_4) = 4(b_1 - b_0), \quad D_{43}(V_4) = 4(b_5 - b_0).$$

Usando os valores associados aos vértices do tetraedro, podemos determinar as ordenadas do conjunto definido em (4.6) através das seguintes Equações

$$\begin{array}{lll} b_{18} = \frac{1}{4}D_{21}(V_2) + b(V_2), & b_8 = \frac{1}{4}D_{23}(V_2) + b(V_2), & b_3 = \frac{1}{4}D_{24}(V_2) + b(V_2), \\ b_{24} = \frac{1}{4}D_{31}(V_3) + b(V_3), & b_{13} = \frac{1}{4}D_{32}(V_3) + b(V_3), & b_{12} = \frac{1}{4}D_{34}(V_3) + b(V_3), \\ b_{15} = \frac{1}{4}D_{41}(V_4) + b(V_4), & b_1 = \frac{1}{4}D_{42}(V_4) + b(V_4), & b_5 = \frac{1}{4}D_{43}(V_4) + b(V_4). \end{array} \tag{4.8}$$

Da mesma forma, as malhas $\mathbf{a_i}, \mathbf{c_i}$ e $\mathbf{d_i}$ têm seus valores de controle determinados por suas respectivas derivadas direcionais nas arestas dos seus tetraedros.

■

4.4 Continuidade C^1 entre Tetraedros Macros Adjacentes

Para garantir que a condição C^1 seja satisfeita numa face, precisamos determinar restrições adequadas sobre os valores de controle próximos e/ou dentro da face para os dois tetraedros vizinhos que compartilham a face. Ao mesmo tempo, é desejável que as restrições possam ser verificadas de forma local, ou seja, que todos os ajustes em valores de controle possam ser feitos de maneira independente em cada tetraedro. Assim, podemos trabalhar um tetraedro de cada vez e quando finalizarmos um tetraedro e passarmos para seu vizinho não precisaremos armazenar suas informações. O seguinte Teorema nos ajudará nesse objetivo:

Teorema 4.2. *Considere $F : T_1 \cup T_2 \to \mathbb{R}$ funcional composto quártico, sendo dois tetraedros com uma face comum, T_1 e T_2, tal que*

$$F(\mathbf{u}) = \begin{cases} S_1(\mathbf{u}) & \text{se } \mathbf{u} \in T_1 \\ S_2(\mathbf{u}) & \text{se } \mathbf{u} \in T_2, \end{cases}$$

onde S_1 e S_2 são dois funcionais quárticos de Bézier definidos sobre T_1 e T_2, respectivamente. Suponha que na face comum, $V_2V_3V_4$, valem as propriedades:

1. *S_1 e S_2 têm os mesmos pontos e valores de controle,*
2. *S_1 e S_2 têm as mesmas derivadas nos vértices V_2, V_3 e V_4,*
3. *A derivada de F na direção de $\mathbf{n}, D_{\mathbf{n}}F$, varia linearmente.*

Então podemos concluir que F é C^1.

Demonstração: Podemos escrever um vetor qualquer de $\mathbb{R}^3$ com coordenadas baricêntricas da forma (a,b,c,d). Queremos mostrar que $D_{\mathbf{m}}(S_1(\mathbf{u})) = D_{\mathbf{m}}(S_2(\mathbf{u}))$ para $\mathbf{u} \in V_2V_3V_4$, uma vez que a continuidade C^1 significa que as superfícies resultantes tenham os mesmos planos tangentes nessa região comum.

Seja $\mathbf{n} = (c,d,e,f)$ o vetor normal à face $V_2V_3V_4$ do tetraedro T e $\mathcal{B} = \{\mathbf{n}, \mathbf{d}_{23}, \mathbf{d}_{24}\}$ uma base de $\mathbb{R}^3$, lembrando que $\mathbf{d}_{ij} = \overrightarrow{V_iV_j}$.

Suponha que $[\mathbf{m}]_{\mathcal{B}} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$, ou seja $\mathbf{m} = a\mathbf{n} + b\mathbf{d}_{23} + c\mathbf{d}_{24}$. Vale então a igualdade:

$$D_{\mathbf{m}}(F(\mathbf{u})) = aD_{\mathbf{n}}(F(\mathbf{u})) + bD_{\mathbf{d}_{23}}(F(\mathbf{u})) + cD_{\mathbf{d}_{24}}(F(\mathbf{u})),$$

para $\mathbf{u} \in V_2V_3V_4$.

Pela primeira hipótese do Teorema, observando o que foi discutido na Seção anterior, temos que para $\mathbf{u} \in V_2V_3V_4$,

$$D_{\mathbf{d}_{23}}(S_1(\mathbf{u})) = D_{\mathbf{d}_{23}}(S_2(\mathbf{u})),$$

$$D_{\mathbf{d}_{24}}(S_1(\mathbf{u})) = D_{\mathbf{d}_{24}}(S_2(\mathbf{u})).$$

Então

$$D_{\mathbf{m}}(S_1(\mathbf{u})) = aD_{\mathbf{n}}(S_1(\mathbf{u})) + bD_{\mathbf{d}_{23}}(S_1(\mathbf{u})) + cD_{\mathbf{d}_{24}}(S_1(\mathbf{u})),$$

$$D_{\mathbf{m}}(S_2(\mathbf{u})) = aD_{\mathbf{n}}(S_2(\mathbf{u})) + bD_{\mathbf{d}_{23}}(S_2(\mathbf{u})) + cD_{\mathbf{d}_{24}}(S_2(\mathbf{u}))$$

e, para concluirmos o que queremos basta provar que

$$D_{\mathbf{n}}(S_1(\mathbf{u})) = D_{\mathbf{n}}(S_2(\mathbf{u})), \text{ para } \mathbf{u} \in V_2V_3V_4.$$

Pela Definição de derivada direcional

$$D_{\mathbf{n}}(S_1(\mathbf{u})) = d\frac{\partial S_1(\mathbf{u})}{\partial v} + e\frac{\partial S_1(\mathbf{u})}{\partial w} + f\frac{\partial S_1(\mathbf{u})}{\partial r},$$

para $\mathbf{u} = (0, u_0, v_0, w_0)$.

Em consequência da segunda hipótese temos que

$$D_{\mathbf{n}}S_1(0,1,0,0) = D_{\mathbf{n}}S_2(0,1,0,0),$$
$$D_{\mathbf{n}}S_1(0,0,1,0) = D_{\mathbf{n}}S_2(0,0,1,0),$$
$$D_{\mathbf{n}}S_1(0,0,0,1) = D_{\mathbf{n}}S_2(0,0,0,1).$$

Disso, e tomando $\mathbf{u} = v_0V_2 + w_0V_3 + r_0V_4$, com $v_0 + w_0 + r_0 = 1$, temos que em consequência da terceira hipótese

$$D_{\mathbf{n}}S_1(\mathbf{u}) = v_0D_{\mathbf{n}}S_1(0,1,0,0) + w_0D_{\mathbf{n}}S_1(0,0,1,0) + r_0D_{\mathbf{n}}S_1(0,0,0,1),$$
$$D_{\mathbf{n}}S_2(\mathbf{u}) = v_0D_{\mathbf{n}}S_2(0,1,0,0) + w_0D_{\mathbf{n}}S_2(0,0,1,0) + r_0D_{\mathbf{n}}S_2(0,0,0,1)$$

e, portanto, $D_{\mathbf{n}}S_1(\mathbf{u}) = D_{\mathbf{n}}S_2(\mathbf{u})$.

■

4.4.1 A Propriedade da Localidade

Uma propriedade importante para o desempenho computacional é a chamada localidade, onde o cômputo do valor para um dado ponto depende apenas de alguns nós da grade, ao contrário do que ocorre em métodos ditos globais (ex: Shepard) em que o cômputo do valor para um ponto depende de todos os nós. Em métodos derivados da interpolação de Clough-Tocher, esta propriedade é garantida ao se tornar o cálculo dos valores de controle num dado triângulo (ou tetraedro) dependente apenas de informações concernentes ao próprio triângulo, e não depender de triângulos vizinhos. No método de Clough-Tocher, além de valores nos vértices e suas derivadas parciais, também requere-se que derivadas normais nos pontos médio de cada aresta sejam fornecidos. Naturalmente o valor da derivada normal no ponto médio de uma dada aresta é conhecido pelos dois triângulos que compartilham a aresta. Esta restrição em particular é suficiente para que os valores de controle na aresta, bem como vizinhos à aresta, sejam determinados para o caso cúbico. Kong *et al* [1] utiliza a equivalente restrição de que a derivada normal varie linearmente na aresta. Nos próximos parágrafos veremos como esta condição se generaliza para o caso trivariado quártico, e também veremos que, apesar de garantir suavidade C^1 na face de borda, ela não é suficiente para preencher os graus de liberdade da malha quártica na face de borda e na primeira camada acima dela.

Denote por $\mathbf{d}_{ij}$ o vetor que se origina no vértice V_i e termina no vértice V_j do tetraedro T. Precisamos encontrar d, e, f, g, coordenadas do vetor normal $\mathbf{n}$ à face $V_2V_3V_4$ do tetraedro (veja Figura 4.5 e Definição 4.7). Para isso podemos determinar um conjunto de vetores ortogonais, através do processo de ortogonalização de Gramm-Schmidt, a partir da base $\{\mathbf{d}_{23}, \mathbf{d}_{24}, \mathbf{d}_{21}\}$,

$$\begin{aligned}
\mathbf{d}'_{23} &= \mathbf{d}_{23}, \\
\mathbf{d}'_{24} &= \mathbf{d}_{24} - h_1 \cdot \mathbf{d}'_{23}, \\
\mathbf{d}'_{21} &= \mathbf{d}_{21} - h_2\mathbf{d}'_{23} - h_3\mathbf{d}'_{24},
\end{aligned}$$

onde

$$h_1 = \frac{<\mathbf{d}_{24}, \mathbf{d}'_{23}>}{<\mathbf{d}'_{23}, \mathbf{d}'_{23}>}, h_2 = \frac{<\mathbf{d}_{21}, \mathbf{d}'_{23}>}{<\mathbf{d}'_{23}, \mathbf{d}'_{23}>} \text{ e } h_3 = \frac{<\mathbf{d}_{21}, \mathbf{d}'_{24}>}{<\mathbf{d}'_{24}, \mathbf{d}'_{24}>}.$$

Dessa forma, o vetor desejado, $\mathbf{n} = \mathbf{d}'_{21}$, em coordenadas baricêntricas com relação a V_1, V_2, V_3 e V_4, respectivamente, vale

$$\begin{bmatrix} 1 \\ h_2 - h_3 \cdot h_1 + h_3 - 1 \\ h_3 \cdot h_1 - h_2. \\ -h_3 \end{bmatrix}$$

Assim, dado um ponto qualquer da face $V_2V_3V_4$, $(0, v, w, r)$, onde $v + w + r = 1$, temos que $D_{\mathbf{n}}((0, v, w, r))$ vale

$$1 \cdot \frac{\partial S}{\partial u}(0, v, w, r) - \alpha \cdot \frac{\partial S}{\partial v}(0, v, w, r) - \beta \cdot \frac{\partial S}{\partial w}(0, v, w, r) + (\alpha + \beta - 1)\frac{\partial S}{\partial r}(0, v, w, r),$$

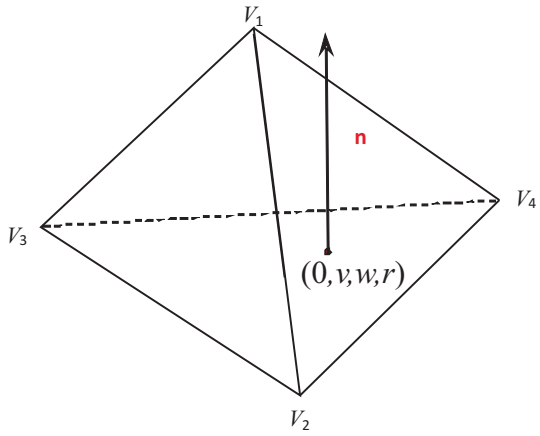

Figura 4.5 Vetor **n** normal a uma face do tetraedro.

onde

$$\frac{\partial S}{\partial u}(0,v,w,r) = 4\sum_{|\mathbf{i}|=3} b_{\mathbf{i}+\mathbf{e}_1} B^3_{\mathbf{i}}(0,v,w,r),$$

$$\frac{\partial S}{\partial v}(0,v,w,r) = 4\sum_{|\mathbf{i}|=3} b_{\mathbf{i}+\mathbf{e}_2} B^3_{\mathbf{i}}(0,v,w,r),$$

$$\frac{\partial S}{\partial w}(0,v,w,r) = 4\sum_{|\mathbf{i}|=3} b_{\mathbf{i}+\mathbf{e}_3} B^3_{\mathbf{i}}(0,v,w,r),$$

$$\frac{\partial S}{\partial r}(0,v,w,r) = 4\sum_{|\mathbf{i}|=3} b_{\mathbf{i}+\mathbf{e}_4} B^3_{\mathbf{i}}(0,v,w,r),$$

$$\alpha = h_2 - h_3 \cdot h_1 + h_3 - 1, \qquad \beta = h_3 \cdot h_1 - h_2,$$

$$\mathbf{e}_1 = 1000, \quad \mathbf{e}_2 = 0100, \quad \mathbf{e}_3 = 0010, \quad \mathbf{e}_4 = 0001$$

e $\mathbf{i} = ijkp$, sendo $i+j+k+p = 3$.

Usando que as coordenadas do ponto analisado são $(0,v,w,1-v-w)$, desenvolvendo e agrupando os termos da derivada parcial acima e usando a notação simplificada definida em (4.1), temos o resultado que segue nas expressões abaixo

Agrupando os termos de v^3 encontramos

$b_{18} - (1 - h_2 + h_3h_1 - h_3)b_4 - (h_2 - h_3h_1)b_8 + (-h_3)b_3 - b_{15} + (1 - h_2 + h_3h_1 - h_3)b_1 + (h_2 - h_3h_1)b_5 - (-h_3)b_0 - 3b_{17} + 3(1 - h_2 + h_3h_1 - h_3)b_3 + 3(h_2 - h_3h_1)b_7 - 3(-h_3)b_2 + 3b_{16} - 3(1 - h_2 + h_3h_1 - h_3)b_2 - 3(h_2 - h_3h_1)b_6 + 3(-h_3)b_1.$

Agrupando os termos de w^3 encontramos

$b_{24} - (1 - h_2 + h_3h_1 - h_3)b_{13} - (h_2 - h_3h_1)b_{14} + (-h_3)b_{12} - b_{15} + (1 - h_2 + h_3h_1 - h_3)b_1 + (h_2 - h_3h_1)b_5 + (-h_3)b_0 - 3b_{22} + 3(1 - h_2 + h_3h_1 - h_3)b_{10} + 3(h_2 - h_3h_1)b_{12} - 3(-h_3)b_9 + 3b_{19} - 3(1 - h_2 + h_3h_1 - h_3)b_6 - 3(h_2 - h_3h_1)b_9 + 3(-h_3)b_5.$

Agrupando os termos de v^2 encontramos

$3b_{15}-3(1-h_2+h_3h_1-h_3)b_1-3(h_2-h_3h_1)b_5+3(-h_3)b_0+3b_{17}-3(1-h_2+h_3h_1-h_3)b_3-3(h_2-h_3h_1)b_7+3(-h_3)b_2-6b_{16}+6(1-h_2+h_3h_1-h_3)b_2+6(h_2-h_3h_1)b_6-6(-h_3)b_1.$

Agrupando os termos de w^2 encontramos

$3b_{15}-3(1-h_2+h_3h_1-h_3)b_1-3(h_2-h_3h_1)b_5+3(-h_3)b_0+3b_{22}-3(1-h_2+h_3h_1-h_3)b_{10}-3(h_2-h_3h_1)b_{12}+3(-h_3)b_9-6b_{19}-6(1-h_2+h_3h_1-h_3)b_6+6(h_2-h_3h_1)b_9-6(-h_3)b_5.$

Agrupando os termos de v^2w encontramos

$-3b_{15}+3(1-h_2+h_3h_1-h_3)b_1+3(h_2-h_3h_1)b_5-3(-h_3)b_0+3b_{21}-3(1-h_2+h_3h_1-h_3)b_8-3(h_2-h_3h_1)b_{11}+3(-h_3)b_7-3b_{17}+3(1-h_2+h_3h_1-h_3)b_3+3(h_2-h_3h_1)b_7-3(-h_3)b_2+6b_{16}-6(1-h_2+h_3h_1-h_3)b_2-6(h_2-h_3h_1)b_6+6(-h_3)b_1+3b_{19}-3(1-h_2+h_3h_1-h_3)b_6-3(h_2-h_3h_1)b_9+3(-h_3)b_5-b_{20}+(1-h_2+h_3h_1-h_3)b_7+(h_2-h_3h_1)b_{10}-(-h_3)b_6.$

Agrupando os termos de vw^2 encontramos

$-3b_{15}+3(1-h_2+h_3h_1-h_3)b_1+3(h_2-h_3h_1)b_5-3(-h_3)b_0+3b_{23}-3(1-h_2+h_3h_1-h_3)b_{11}-3(h_2-h_3h_1)b_{13}+3(-h_3)b_{10}-3b_{22}+3(1-h_2+h_3h_1-h_3)b_{10}+3(h_2-h_3h_1)b_{12}-3(-h_3)b_9+3b_{16}-3(1-h_2+h_3h_1-h_3)b_2-3(h_2-h_3h_1)b_6+3(-h_3)b_1+6b_{19}-6(1-h_2+h_3h_1-h_3)b_6-6(h_2-h_3h_1)b_9+6(-h_3)b_5-b_{20}+(1-h_2+h_3h_1-h_3)b_7+(h_2-h_3h_1)b_{10}-(-h_3)b_6.$

Agrupando os termos de vw encontramos

$6b_{15}-6(1-h_2+h_3h_1-h_3)b_1-6(h_2-h_3h_1)b_5+6(-h_3)b_0-6b_{16}+6(1-h_2+h_3h_1-h_3)b_2+6(h_2-h_3h_1)b_6-6(-h_3)b_1-6b_{19}+6(1-h_2+h_3h_1-h_3)b_6+6(h_2-h_3h_1)b_9-6(-h_3)b_5+b_{20}-(1-h_2+h_3h_1-h_3)b_7-(h_2-h_3h_1)b_{10}+(-h_3)b_6.$

Todos esses termos agrupados citados acima precisam ser simultaneamente igualados a zero, a fim de alcançarmos a linearidade da derivada calculada na Seção 4.4. A continuidade C^1 e ao mesmo tempo a localidade na fixação de valores de controle ao longo da fronteira comum entre dois tetraedros macros adjacentes, poderão ser atingidas se forem estas condições satisfeitas neste tetraedro assim como as correspondentes condições no tetraedro vizinho. O número de graus de liberdade empregados nestas restrições é estritamente menor que o número de graus de liberdade disponíveis na malha quártica na face comum.

Com o uso de um software matemático [27], igualando todos esses termos a zero, o problema fica estabelecido, com sete equações e treze variáveis. Abaixo segue o trecho do programa desenvolvido para o cálculo desse sistema de equações.

```
solve({b18-((1-h2+h3*h1-h3+1))*b4-(h2-h3*h1)*b8+(-h3)*b3-b15+
(1-h2+h3*h1-h3+1)*b1+(h2-h3*h1)*b5-(-h3)*b0-3*b17+
3*(1-h2+h3*h1-h3+1)*b3+3*(h2-h3*h1)*b7-3*(-h3)*b2+3*b16
-3*(1-h2+h3*h1-h3+1)*b2-3*(h2-h3*h1)*b6+3*(-h3)*b1=0,

b24-(1-h2+h3*h1-h3+1)*b13-(h2-h3*h1)*b14+(-h3)*b12-b15+
```

```
(1-h2+h3*h1-h3+1)*b1+(h2-h3*h1)*b5-(-h3)*b0-3*b22+
3*(1-h2+h3*h1-h3+1)*b10+3*(h2-h3*h1)*b12-3*(-h3)*b9+3*b19-
3*(1-h2+h3*h1-h3+1)*b6-3*(h2-h3*h1)*b9+3*(-h3)*b5=0,

3*b15-3*(1-h2+h3*h1-h3+1)*b1-3*(h2-h3*h1)*b5+3*(-h3)*b0+3*b17-
3*(1-h2+h3*h1-h3+1)*b3-3*(h2-h3*h1)*b7+3*(-h3)*b2-6*b16+
6*(1-h2+h3*h1-h3+1)*b2+6*(h2-h3*h1)*b6-6*(-h3)*b1=0,

3*b15-3*(1-h2+h3*h1-h3+1)*b1-3*(h2-h3*h1)*b5+3*(-h3)*b0+3*b22-
3*(1-h2+h3*h1-h3+1)*b10-3*(h2-h3*h1)*b12+3*(-h3)*b9-6*b19+
6*(1-h2+h3*h1-h3+1)*b6+6*(h2-h3*h1)*b9-6*(-h3)*b5=0,

6*b15-6*(1-h2+h3*h1-h3+1)*b1-6*(h2-h3*h1)*b5+6*(-h3)*b0-6*b16+
6*(1-h2+h3*h1-h3+1)*b2+6*(h2-h3*h1)*b6-6*(-h3)*b1-6*b19+
6*(1-h2+h3*h1-h3+1)*b6+6*(h2-h3*h1)*b9-6*(-h3)*b5+b20
-(1-h2+h3*h1-h3)*b7-(h2-h3*h1)*b10+(-h3)*b6=0,

-3*b15+3*(1-h2+h3*h1-h3+1)*b1+3*(h2-h3*h1)*b5-3*(-h3)*b0+3*b21-
3*(1-h2+h3*h1-h3+1)*b8-3*(h2-h3*h1)*b11+3*(-h3)*b7-3*b17+
3*(1-h2+h3*h1-h3+1)*b3+3*(h2-h3*h1)*b7-3*(-h3)*b2+6*b16-
6*(1-h2+h3*h1-h3+1)*b2-6*(h2-h3*h1)*b6+6*(-h3)*b1+3*b19-
3*(1-h2+h3*h1-h3+1)*b6-3*(h2-h3*h1)*b9+3*(-h3)*b5-b20+
(1-h2+h3*h1-h3+1)*b7+(h2-h3*h1)*b10-(-h3)*b6=0,

-3*b15+3*(1-h2+h3*h1-h3+1)*b1+3*(h2-h3*h1)*b5-3*(-h3)*b0+3*b23-
3*(1-h2+h3*h1-h3+1)*b11-3*(h2-h3*h1)*b13+3*(-h3)*b10-3*b22+
3*(1-h2+h3*h1-h3+1)*b10+3*(h2-h3*h1)*b12-3*(-h3)*b9+3*b16-
3*(1-h2+h3*h1-h3+1)*b2-3*(h2-h3*h1)*b6+3*(-h3)*b1+6*b19-
6*(1-h2+h3*h1-h3+1)*b6-6*(h2-h3*h1)*b9+6*(-h3)*b5-b20+
(1-h2+h3*h1-h3+1)*b7+(h2-h3*h1)*b10-(-h3)*b6=0},
{b2,b6,b7,b9,b10,b11,b16,b17,b19,b20,b21,b22,b23});
```

O software forneceu uma solução, onde sete variáveis são determinadas em termos de valores conhecidos, como h_1, h_2 e h_3, ou então através de valores de controle, obtidos através da derivada direcional calculada em um dos vértices do tetraedro, são eles

$$b_1, b_3, b_5, b_8, b_{12}, b_{13}, b_{15}, b_{18} \text{ e } b_{24}.$$

As outras variáveis são deixadas livres, indeterminadas até o momento.

```
{ b17 = 3*h3*h1*b23-2*h3*h1*b24-h3*h1*b18-3*h3^2*h1*b10+
2*h3^2*h1*b12+h3^2*h1*b3-h3^2*h1^2*b8-3*h3^2*h1^2*b11+
```

```
5*h3^2*h1^2*b13+h3^2*h1^2*b4+2*h3*b13*h2-3*h3*b11*h2+
h3*b4*h2+3*b11*h3^2*h1-2*b13*h3^2*h1-b4*h3^2*h1-
2*h3^2*h1^2*b14+4/3*b4*h3*h1+4*b13*h3*h1+b3*h3*h1-
6*b11*h3*h1+2/3*b8*h3*h1+1/3*b5*h3*h1-1/3*b1*h3*h1+
3*h2*h3*b10-h2*h3*b3-2*h2*h3*b12+4*h3*h1*b14*h2+
2/3*b18-4/3*b4+2*b3+1/3*b15-2/3*b1-5/3*h3*b3-1/3*h3*b0+
h3*b2+1/3*h3*b1-4/3*b4*h2+2/3*b4*h3-2/3*b8*h2+1/3*b1*h2-
1/3*b5*h2-b3*h2+6*b11*h2-4*b13*h2-2*h3*h1*b4*h2+
2*h3*h1*b8*h2+6*h3*h1*b11*h2-10*h3*h1*b13*h2-2*b14*h2^2+
h2*b18+2*h2*b24-3*h2*b23+b4*h2^2-b8*h2^2-3*b11*h2^2+5*b13*h2^2,

b20 = -3*h3*h1*b23+2*h3*h1*b24+h3*h1*b18+3*h3^2*h1*b10-
2*h3^2*h1*b12-h3^2*h1*b3+h3^2*h1^2*b8+3*h3^2*h1^2*b11-
5*h3^2*h1^2*b13-h3^2*h1^2*b4-7*h3*b13*h2+6*h3*b11*h2+h3*b8*h2-
2*h3*b4*h2+2*h3*b14*h2-b8*h3^2*h1-6*b11*h3^2*h1+7*b13*h3^2*h1+
2*b4*h3^2*h1-2*b14*h3^2*h1+2*h3^2*h1^2*b14+4*b14*h3*h1-
5*b4*h3*h1-11*b13*h3*h1+9*b11*h3*h1+3*b8*h3*h1+2*b5*h3*h1-
2*b1*h3*h1-b10*h3*h1-3*h2*h3*b10-h3*b18+h2*h3*b3+2*h2*h3*b12+
b10*h2-4*b14*h2-4*h3*h1*b14*h2+3*b18-6*b4+2*b15-4*b1+4*b24-
8*b13+6*b11-3*b23-3*h3*b3-2*h3*b0+2*h3*b1-4*h3*b12+h3*b6+
3*h3*b10+5*b4*h2+5*b4*h3-3*b8*h2+2*b1*h2-2*b5*h2-9*b11*h2-
9*b11*h3+11*b13*h2+2*h3*h1*b4*h2-2*h3*h1*b8*h2-
6*h3*h1*b11*h2+10*h3*h1*b13*h2-2*h3*b24+8*h3*b13+3*h3*b23+
h3^2*b3+2*h3^2*b12-3*h3^2*b10-b4*h3^2+3*b11*h3^2+2*b14*h2^2-
h2*b18-2*h2*b24+3*h2*b23-b4*h2^2+b8*h2^2+3*b11*h2^2-
5*b13*h2^2-2*h3^2*b13,

b22 = 2/3*b14*h3*h1-2/3*b13*h3*h1+1/3*b5*h3*h1-1/3*b1*h3*h1+
b10*h3*h1-b12*h3*h1-b10*h2-2/3*b14*h2+1/3*b15-2/3*b1+2/3*b24-
4/3*b13+2*b10-1/3*h3*b0+1/3*h3*b1-2/3*h3*b12+h3*b9-h3*b10+
1/3*b1*h2-1/3*b5*h2+b12*h2+2/3*b13*h2+2/3*h3*b13,

b19 = b6*h3*h1+2*b6-b6*h2-h3*b6+b9*h2-b9*h3*h1-2/3*h3*b0+
1/3*b14*h3*h1-1/3*b13*h3*h1+h3*b5+2/3*b5*h3*h1-2/3*b1*h3*h1-
1/3*b14*h2+2/3*b15-4/3*b1+1/3*b13*h2+1/3*b24-2/3*b13+
2/3*b1*h2+2/3*h3*b1-1/3*h3*b12-2/3*b5*h2+1/3*h3*b13,

b7 = 3*h3*b10-3*b11*h3+6*b11-3*b11*h2+3*b11*h3*h1+2*b14*h3*h1-
b4*h3*h1-5*b13*h3*h1+b4*h2+b8*h3*h1-2*b14*h2+b18-2*b4+
5*b13*h2+2*b24-4*b13-3*b23-h3*b3-2*h3*b12+2*h3*b13+b4*h3-b8*h2,

b21=5*h3*b13*h2-3*h3*b11*h2-h3*b8*h2+h3*b4*h2-2*h3*b14*h2+
b8*h3^2*h1+3*b11*h3^2*h1-5*b13*h3^2*h1-b4*h3^2*h1+
```

```
2*b14*h3^2*h1-1/3*b14*h3*h1-1/3*b4*h3*h1+4/3*b13*h3*h1-
2*b11*h3*h1+4/3*b8*h3*h1+h3*b18+1/3*b14*h2+1/3*b18-2/3*b4+2*b8-
1/3*b24+2/3*b13-2*b11+b23-1/3*h3*b3+1/3*h3*b12-h3*b10+
1/3*b4*h2-5/3*b4*h3-4/3*b8*h2-b8*h3+2*b11*h2+7*b11*h3-
4/3*b13*h2+2*h3*b24-13/3*h3*b13-3*h3*b23-h3^2*b3-2*h3^2*b12+
3*h3^2*b10+b4*h3^2-3*b11*h3^2+2*h3^2*b13,

b16 = b2*h3*h1+2*b2-b2*h2-h3*b2+b6*h2-b6*h3*h1-1/3*b4*h3*h1-
2/3*b1*h3*h1+1/3*b8*h3*h1+2/3*b5*h3*h1+1/3*b4*h3-4/3*b1+1/3*b18-
2/3*b4+2/3*b15-2/3*h3*b0-1/3*b8*h2+5/3*h3*b1-1/3*h3*b3-
2/3*b5*h2+1/3*b4*h2+2/3*b1*h2,

b2 = b2, b6 = b6, b10 = b10, b9 = b9, b11 = b11, b23 = b23}
```

4.5 Determinação de Outros Valores de Controle

No final da Seção anterior vimos que alguns pontos da malha quártica de Bézier ficam livres, sem nenhum valor associado aos mesmos, o que significa que a condição C^1 e a localidade impõem menos restrições sobre os valores de controle, do que temos de graus de liberdade. Resolvemos então estabelecer uma relação entre esses valores de controle livres, sejam eles, $b_2, b_6, b_7, b_9, b_{10}, b_{11}$, e o próprio tetraedro, através de valores de outros pontos da malha.

A idéia é reduzir o grau da malha de forma a se tornar uma malha cúbica na face abaixo (veja Figura 4.6), que possui quinze pontos, dos quais nove deles possuem informação, os vértices do triângulo que estão nessa face, b_0, b_4, b_{14} e os determinados pela derivada nos vértices, $b_1, b_3, b_5, b_8, b_{12}, b_{13}$, como vimos anteriormente. Dessa forma, nos resta relacionar os outros valores $b_2, b_6, b_7, b_9, b_{10}, b_{11}$ com os citados. Vejamos isso.

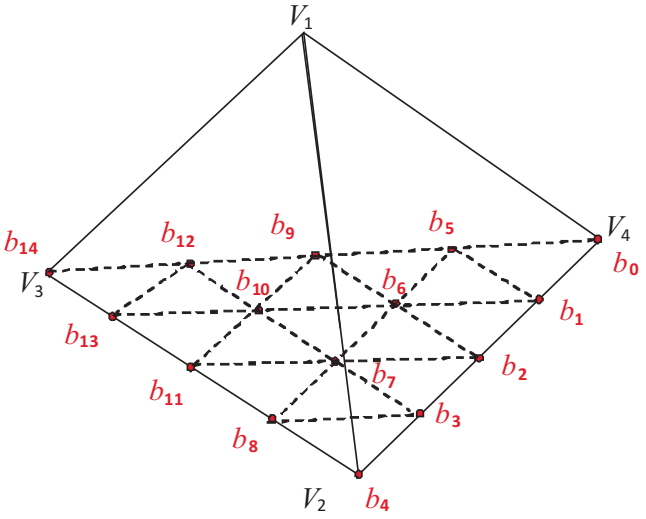

Figura 4.6 Malha quártica com alguns valores de controle associados a seus pontos de controle.

Considere uma malha triangular quártica no tetraedro, $b_{\mathbf{i}}$, distribuída na face citada. Já com a notação adaptada, estabelecida em (4.1), encontramos:

$$b^4_{000}(\mathbf{u}) = b_0u^4 + b_4v^4 + b_{14}w^4 + 4b_1u^3v + 4b_5u^3w + 4b_3uv^3 + 4b_8v^3w + 4b_{12}uw^3 +$$
$$+ 4b_{13}vw^3 + 6b_2u^2v^2 + 6b_9u^2w^2 + 6b_{11}v^2w^2 + 12b_6u^2vw + 12b_7uv^2w +$$
$$+ 12b_{10}uvw^2, \text{ onde } u+v+w = 1.$$

Agora, estabeleçamos uma malha triangular cúbica nessa mesma face:

$$c^3_{000}(\mathbf{u}) = c_0u^3 + 3c_1u^2v + 3c_2uv^2 + c_3v^3 + 3c_4v^2w + 3c_5vw^2 + c_6w^3 + 3c_7uw^2 +$$
$$+ 3c_8u^2w + 6c_9uvw, \text{ onde } u+v+w = 1.$$

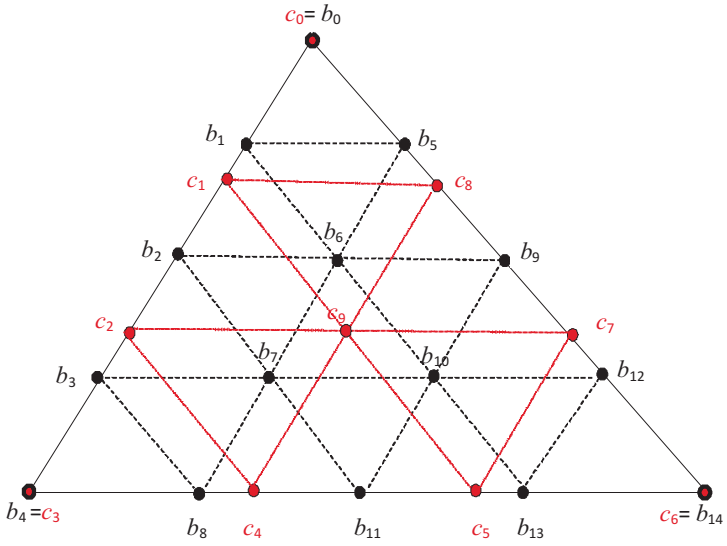

Figura 4.7 Malhas triangulares: cúbica e quártica.

Nosso objetivo é reduzir o grau da malha na fronteira, dado o excesso de graus de liberdade. Isto deve ser feito de forma a respeitar o critério da localidade.

Dada a malha triangular quártica, podemos descrever alguns de seus termos de forma conveniente, usando o fato de que $u+v+w = 1$. Seja ela:

$$b^4_{000}(\mathbf{u}) = (1-v-w)b_0u^3 + (1-u-w)b_4v^3 + (1-u-v)b_{14}w^3 + (1-v-w)4b_1u^2v +$$
$$+ (1-v-w)4b_5u^2w + (1-u-w)4b_3uv^2 + (1-u-w)4b_8v^2w +$$
$$+ (1-u-v)4b_{12}uw^2 + (1-u-v)4b_{13}vw^2 + 6b_2u^2v^2 + 6b_9u^2w^2 +$$
$$+ 6b_{11}v^2w^2 + 12uvw[(1-v-w)b_6 + (1-u-w)b_7 + (1-u-v)b_{10}].$$

Desenvolvendo a expressão acima obtemos:

$$b^4_{000}(\mathbf{u}) = b_0u^3 + b_4v^3 + b_{14}w^3 + 4b_1u^2v + 4b_5u^2w + 4b_3uv^2 + 4b_8v^2w + 4b_{12}uw^2 + 4b_{13}vw^2 +$$
$$+ 12uvw[b_6 + b_7 + b_{10}] + 6b_2u^2v^2 + 6b_9u^2w^2 + 6b_{11}v^2w^2 - b_0u^3v - b_0u^3w - b_4uv^3 -$$
$$- b_4wv^3 - b_{14}uw^3 - b_{14}vw^3 - 4b_1u^2v^2 - 4b_1u^2vw - 4b_5u^2vw - 4b_5u^2w^2 - 4b_3u^2v^2 -$$
$$- 4b_3uwv^2 - 4b_8uv^2w - 4b_8v^2w^2 - 4b_{12}u^2w^2 - 4b_{12}uvw^2 - 4b_{13}uvw^2 - 4b_{13}v^2w^2 -$$
$$- 12b_6uv^2w - 12b_6uvw^2 - 12b_7u^2vw - 12b_7uvw^2 - 12b_{10}u^2vw - 12b_{10}uv^2w.$$

Usando que $u+v+w=1$, mais termos cúbicos podem aparecer. Dessa forma, adaptando os termos, $b^4_{000}(\mathbf{u})$ é igual a

$$b_0u^3 + b_4v^3 + b_{14}w^3 + 4b_1u^2v - b_0u^2v(1-v-w) + 4b_5u^2w - b_0u^2w(1-v-w) + 4b_3uv^2 -$$
$$-b_4uv^2(1-u-w) + 4b_8v^2w - b_4wv^2(1-u-w) + 4b_{12}uw^2 - b_{14}uw^2(1-u-v) + 4b_{13}vw^2 -$$
$$-b_{14}vw^2(1-u-v) + 2(b_6+b_7+b_{10})uvw + 6b_2u^2v^2 + 6b_9u^2w^2 + 6b_{11}v^2w^2 -$$
$$-4b_1u^2v^2 - 4b_1u^2vw - 4b_5u^2vw - 4b_5u^2w^2 - 4b_3u^2v^2 - 4b_3uv^2w - 4b_8uv^2w - 4b_8v^2w^2 -$$
$$-4b_{12}u^2w^2 - 4b_{12}uvw^2 - 4b_{13}uvw^2 - 4b_{13}v^2w^2 - 12b_6uv^2w - 12b_6uvw^2 - 12b_7u^2vw -$$
$$-12b_7uvw^2 - 12b_{10}u^2vw - 12b_{10}uv^2w + (10b_6 - 2b_7 - 2b_{10})u^2vw +$$
$$+(10b_7 - 2b_6 - 2b_{10})uv^2w + (10b_{10} - 2b_7 - 2b_7)uvw^2.$$

Usando as relações entre elementos cúbicos e quárticos, de acordo com a Figura 4.7, podemos escrever

$$c_0 = b_0, \quad c_1 = \frac{4}{3}b_1 - \frac{1}{3}b_0, \quad c_2 = \frac{4}{3}b_3 - \frac{1}{3}b_4, \quad c_3 = b_4, \quad c_4 = \frac{4}{3}b_8 - \frac{1}{3}b_4,$$
$$c_5 = \frac{4}{3}b_{13} - \frac{1}{3}b_{14}, \quad c_6 = b_{14}, \quad c_7 = \frac{4}{3}b_{12} - \frac{1}{3}b_{14}, \quad c_8 = \frac{4}{3}b_5 - \frac{1}{3}b_0.$$

Identificando os termos pertencentes à cúbica, encontramos

$$b^4_{000}(\mathbf{u}) = b_0u^3 + b_4v^3 + b_{14}w^3 + 3\left(\frac{4}{3}b_1 - \frac{1}{3}b_0\right)u^2v + 3\left(\frac{4}{3}b_5 - \frac{1}{3}b_0\right)u^2w +$$
$$+ 3\left(\frac{4}{3}b_3 - \frac{1}{3}b_4\right)uv^2 + 3\left(\frac{4}{3}b_8 - \frac{1}{3}b_4\right)v^2w + 3\left(\frac{4}{3}b_{12} - \frac{1}{3}b_{14}\right)uw^2 +$$
$$+ 3\left(\frac{4}{3}b_{13} - \frac{1}{3}b_{14}\right)vw^2 + 6\left(\frac{1}{3}b_6 + \frac{1}{3}b_7 + \frac{1}{3}b_{10}\right)uvw + R.$$

Onde R é um resíduo envolvendo os termos de grau quatro que não são utilizados durante o processo de comparação com a cúbica.

Seja ele:

$$\begin{aligned}
&6b_2u^2v^2+6b_9u^2w^2+6b_{11}v^2w^2-4b_1u^2v^2-4b_1u^2vw-4b_5u^2vw-4b_5u^2w^2-4b_3u^2v^2-\\
&-4b_3uv^2w-4b_8uv^2w-4b_8v^2w^2-4b_{12}u^2w^2-4b_{12}uvw^2-4b_{13}uvw^2-4b_{13}v^2w^2-\\
&-12b_7uvw^2-12b_{10}u^2vw-12b_{10}uv^2w-12b_6uvw^2-12b_6uv^2w-12b_7u^2vw+\\
&+(10b_6-2b_7-2b_{10})u^2vw+(10b_7-2b_6-2b_{10})uv^2w+(10b_{10}-2b_7-2b_7)uvw^2+\\
&+b_0u^2v^2+b_0u^2vw+b_0u^2vw+b_0u^2w^2+b_4u^2v^2+b_4uv^2w+b_4uv^2w+b_4v^2w^2+\\
&+b_{14}u^2w^2+b_{14}uvw^2+b_{14}uvw^2+b_{14}v^2w^2.
\end{aligned}$$

Agrupando os termos semelhantes encontramos

$$\begin{aligned}
R=&(2b_0-4b_5-4b_1+10b_6-2b_7-2b_{10})u^2vw+(2b_4-4b_8-4b_3+10b_7-2b_6-2b_{10})uv^2w+\\
&+(2b_{14}-4b_{13}-4b_{12}+10b_{10}-2b_7-2b_6)uvw^2+(b_{14}+b_4-4b_{13}-4b_8+6b_{11})v^2w^2+\\
&+(b_{14}+b_0-4b_{12}-4b_5+6b_9)u^2w^2+(b_0+b_4-4b_3-4b_1+6b_2)u^2v^2.
\end{aligned}$$

Dessa forma, para que obtenhamos a cúbica desejada, é preciso anular os termos de R, ou seja

$$\begin{aligned}
2b_0-4b_5-4b_1+10b_6-2b_7-2b_{10}&=0,\\
2b_4-4b_8-4b_3+10b_7-2b_6-2b_{10}&=0,\\
2b_{14}-4b_{13}-4b_{12}+10b_{10}-2b_7-2b_6&=0,\\
b_{14}+b_4-4b_{13}-4b_8+6b_{11}&=0,\\
b_{14}+b_0-4b_{12}-4b_5+6b_9&=0,\\
b_0+b_4-4b_3-4b_1+6b_2&=0.
\end{aligned}\tag{4.9}$$

Vejamos a solução que encontramos para b_6, b_7, b_{10} em função das ordenadas de Bézier conhecidas:

$$\begin{aligned}
b_6&=\frac{2}{9}(2b_1+2b_5-b_0)+\frac{1}{18}(2b_3+2b_8-b_4)+\frac{1}{18}(2b_{12}+2b_{13}-b_{14}),\\
b_7&=\frac{2}{9}(2b_3+2b_8-b_4)+\frac{1}{18}(2b_1+2b_5-b_0)+\frac{1}{18}(2b_{12}+2b_{13}-b_{14}),\\
b_{10}&=\frac{2}{9}(2b_{12}+2b_{13}-b_{14})+\frac{1}{18}(2b_1+2b_5-b_0)+\frac{1}{18}(2b_3+2b_8-b_4).
\end{aligned}$$

Já as três últimas Equações de (4.9) são equivalentes a

$$b_2 = \frac{4}{3}\left[\frac{1}{2}(b_1+b_3)\right] - \frac{1}{3}\left[\frac{1}{2}(b_0+b_4)\right],$$
$$b_9 = \frac{4}{3}\left[\frac{1}{2}(b_5+b_{12})\right] - \frac{1}{3}\left[\frac{1}{2}(b_0+b_{14})\right],$$
$$b_{11} = \frac{4}{3}\left[\frac{1}{2}(b_8+b_{13})\right] - \frac{1}{3}\left[\frac{1}{2}(b_4+b_{14})\right],$$

o que confirma a disposição dos pontos na Figura 4.8 e estabelece uma relação entre b_2, b_9, b_{11} e valores de controle já conhecidos, $b_0, b_1, b_3, b_4, b_5, b_8, b_{12}, b_{13}, b_{14}$.

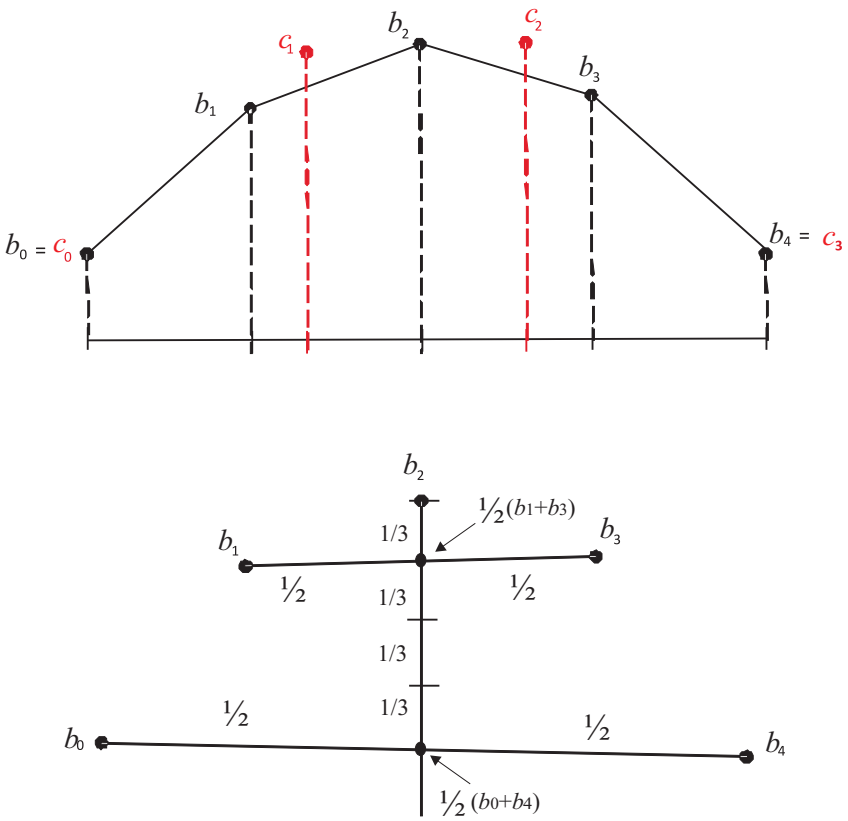

Figura 4.8 Cúbica e quártica.

4.6 Determinação de Todos os Valores de Controle

Com as novas Equações encontradas envolvendo os termos $b_2, b_6, b_7, b_9, b_{10}, b_{11}$, (veja (4.9)), podemos melhorar o programa executado [27] e estimar todos os valores de controle. Vejamos o novo programa, agora com treze equações e treze incógnitas.

```
solve({b18-((1-h2+h3*h1-h3+1))*b4-(h2-h3*h1)*b8+(-h3)*b3-b15+
```

```
(1-h2+h3*h1-h3+1)*1+(h2-h3*h1)*b5-(-h3)*b0-3*b17+
3*(1-h2+h3*h1-h3+1)*b3+3*(h2-h3*h1)*b7-3*(-h3)*b2+3*b16
-3*(1-h2+h3*h1-h3+1)*b2-3*(h2-h3*h1)*b6+3*(-h3)*b1=0,

b24-(1-h2+h3*h1-h3+1)*b13-(h2-h3*h1)*b14+(-h3)*b12-b15+
(1-h2+h3*h1-h3+1)*b1+(h2-h3*h1)*b5-(-h3)*b0-3*b22+
3*(1-h2+h3*h1-h3+1)*b10+3*(h2-h3*h1)*b12-3*(-h3)*b9+
3*b19-3*(1-h2+h3*h1-h3+1)*b6-3*(h2-h3*h1)*b9+3*(-h3)*b5=0,

3*b15-3*(1-h2+h3*h1-h3+1)*b1-3*(h2-h3*h1)*b5+3*(-h3)*b0+3*b17-
3*(1-h2+h3*h1-h3+1)*b3-3*(h2-h3*h1)*b7+3*(-h3)*b2-6*b16+
6*(1-h2+h3*h1-h3+1)*b2+6*(h2-h3*h1)*b6-6*(-h3)*b1=0,

3*b15-3*(1-h2+h3*h1-h3+1)*b1-3*(h2-h3*h1)*b5+3*(-h3)*b0+
3*b22-3*(1-h2+h3*h1-h3+1)*b10-3*(h2-h3*h1)*b12+3*(-h3)*b9-
6*b19+6*(1-h2+h3*h1-h3+1)*b6+6*(h2-h3*h1)*b9-6*(-h3)*b5=0,

6*b15-6*(1-h2+h3*h1-h3+1)*b1-6*(h2-h3*h1)*b5+6*(-h3)*b0-6*b16+
6*(1-h2+h3*h1-h3+1)*b2+6*(h2-h3*h1)*b6-6*(-h3)*b1-6*b19+
6*(1-h2+h3*h1-h3+1)*b6+6*(h2-h3*h1)*b9-6*(-h3)*b5+b20-
(1-h2+h3*h1-h3)*b7-(h2-h3*h1)*b10+(-h3)*b6=0,

-3*b15+3*(1-h2+h3*h1-h3+1)*b1+3*(h2-h3*h1)*b5-3*(-h3)*b0+
3*b21-3*(1-h2+h3*h1-h3+1)*b8-3*(h2-h3*h1)*b11+3*(-h3)*b7-
3*b17+3*(1-h2+h3*h1-h3+1)*b3+3*(h2-h3*h1)*b7-3*(-h3)*b2+
6*b16-6*(1-h2+h3*h1-h3+1)*b2-6*(h2-h3*h1)*b6+6*(-h3)*b1+
3*b19-3*(1-h2+h3*h1-h3+1)*b6-3*(h2-h3*h1)*b9+3*(-h3)*b5-
b20+(1-h2+h3*h1-h3+1)*b7+(h2-h3*h1)*b10-(-h3)*b6=0,

 -3*b15+3*(1-h2+h3*h1-h3+1)*b1+3*(h2-h3*h1)*b5-3*(-h3)*b0+
 3*b23-3*(1-h2+h3*h1-h3+1)*b11-3*(h2-h3*h1)*b13+3*(-h3)*b10-
 3*b22+3*(1-h2+h3*h1-h3+1)*b10+3*(h2-h3*h1)*b12-3*(-h3)*b9+
 3*b16-3*(1-h2+h3*h1-h3+1)*b2-3*(h2-h3*h1)*b6+3*(-h3)*b1+
 6*b19-6*(1-h2+h3*h1-h3+1)*b6-6*(h2-h3*h1)*b9+6*(-h3)*b5-
 b20+(1-h2+h3*h1-h3+1)*b7+(h2-h3*h1)*b10-(-h3)*b6=0,

2*b0-4*b5-4*b1+10*b6-2*b7-2*b10=0,

2*b4-4*b8-4*b3+10*b7-2*b6-2*b10=0,

2*b14-4*b13-4*b12+10*b10-2*b7-2*b6=0,

b14+b4-4*b13-4*b8+6*b11=0,
```

```
b14+b0-4*b12-4*b5+6*b9=0,

b0+b4-4*b3-4*b1+6*b2=0},

{b2,b6,b7,b9,b10,b11,b16,b17,b19,b20,b21,b22,b23});
```

A solução encontrada para cada um dos valores de controle

$$b_2, b_6, b_7, b_9, b_{10}, b_{11}, b_{16}, b_{17}, b_{19}, b_{20}, b_{21}, b_{22}, b_{23},$$

foi

```
{b19 = 1/9*b3*h3*h1-2/9*b1*h3*h1-1/18*b4*h3*h1+4/9*b14*h3*h1+
4/9*b5*h3*h1-5/9*b12*h3*h1-1/18*h3*h1*b0+1/9*b8*h3*h1+
1/18*b4*h2-2/9*b13*h3*h1+5/9*b12*h2-1/9*b4+2/9*b8+2/9*b3+
2/3*b15+8/9*b5-4/9*b0-4/9*b1+1/3*b24-4/9*b13-1/9*b14+2/9*b12-
1/9*h3*b3-4/9*h3*b0+2/9*h3*b1-4/9*h3*b12+5/9*h3*b5+1/18*b4*h3-
1/9*b8*h2-4/9*b5*h2-1/9*b3*h2+2/9*b13*h2+2/9*b13*h3-
4/9*b14*h2+2/9*b1*h2-1/9*b8*h3+1/18*h2*b0+1/18*h3*b14,

b22 = 1/9*b3*h3*h1-2/9*b1*h3*h1-1/18*b4*h3*h1+4/9*b14*h3*h1+
4/9*b5*h3*h1-5/9*b12*h3*h1-1/18*h3*h1*b0+1/9*b8*h3*h1+
1/18*b4*h2-2/9*b13*h3*h1+5/9*b12*h2-1/9*b4+2/9*b8+2/9*b3+
1/3*b15+2/9*b5-1/9*b0-4/9*b1+2/3*b24-4/9*b13-4/9*b14+
8/9*b12-1/9*h3*b3-4/9*h3*b0+2/9*h3*b1-4/9*h3*b12+5/9*h3*b5+
1/18*b4*h3-1/9*b8*h2-4/9*b5*h2-1/9*b3*h2+2/9*b13*h2+
2/9*b13*h3-4/9*b14*h2+2/9*b1*h2-1/9*b8*h3+1/18*h2*b0+
1/18*h3*b14,

b7 = -1/18*b14+1/9*b13+1/9*b12-1/18*b0-2/9*b4+4/9*b8+4/9*b3
+1/9*b5+1/9*b1,

b10 = -1/18*b0+1/9*b5+1/9*b1-1/18*b4+1/9*b8+1/9*b3-2/9*b14+
4/9*b13+4/9*b12,

b6 = -1/18*b4+1/9*b8+1/9*b3-1/18*b14+1/9*b13+1/9*b12-2/9*b0+
4/9*b5+4/9*b1,

b11 = -1/6*b14-1/6*b4+2/3*b13+2/3*b8,

b9 = -1/6*b14-1/6*b0+2/3*b12+2/3*b5,

b2 = -1/6*b0-1/6*b4+2/3*b3+2/3*b1,
```

```
b23 = -1/2*b4*h3*h1+1/2*b14*h3*h1+b8*h3*h1+1/2*b4*h2-
b13*h3*h1+1/3*b18-25/27*b4+32/27*b8-4/27*b3-1/27*b5+1/54*b0-
1/27*b1+2/3*b24-1/27*b13-17/54*b14-1/27*b12-2/9*h3*b3-
1/18*h3*b0+1/9*h3*b1-2/9*h3*b12+1/9*h3*b5+4/9*b4*h3-
b8*h2+b13*h2+4/9*b13*h3-1/2*b14*h2-5/9*b8*h3-1/18*h3*b14,

b21 =-1/2*b4*h3*h1+1/2*b14*h3*h1+b8*h3*h1+1/2*b4*h2-
b13*h3*h1+2/3*b18-34/27*b4+50/27*b8-4/27*b3-1/27*b5+1/54*b0-
1/27*b1+1/3*b24-19/27*b13+1/54*b14-1/27*b12-2/9*h3*b3-
1/18*h3*b0+1/9*h3*b1-2/9*h3*b12+1/9*h3*b5+4/9*b4*h3-b8*h2+
b13*h2+4/9*b13*h3-1/2*b14*h2-5/9*b8*h3-1/18*h3*b14,

b20 = 1/3*b3*h3*h1-2*b1*h3*h1-13/6*b4*h3*h1+
13/6*b14*h3*h1+2*b5*h3*h1-1/3*b12*h3*h1+7/3*b8*h3*h1+
13/6*b4*h2-7/3*b13*h3*h1+1/3*b12*h2+2*b18-38/9*b4+4/9*b8+
4/9*b3+2*b15+1/9*b5-1/18*b0-35/9*b1+2*b24-35/9*b13-1/18*b14
+1/9*b12-7/3*h3*b3-13/6*h3*b0+7/3*h3*b1-2*h3*b12+1/3*h3*b5+
13/6*b4*h3-7/3*b8*h2-2*b5*h2-1/3*b3*h2+7/3*b13*h2+2*b13*h3-
13/6*b14*h2+2*b1*h2-1/3*b8*h3,

b17 = 5/9*b3*h3*h1-4/9*b1*h3*h1-4/9*b4*h3*h1+1/18*b14*h3*h1+
2/9*b5*h3*h1-1/9*b12*h3*h1+1/18*h3*h1*b0+2/9*b8*h3*h1+
4/9*b4*h2-1/9*b13*h3*h1+1/9*b12*h2+2/3*b18-4/3*b4+2*b3+
1/3*b15-2/3*b1-h3*b3-1/2*h3*b0+h3*b1+1/2*b4*h3-2/9*b8*h2-
2/9*b5*h2-5/9*b3*h2+1/9*b13*h2-1/18*b14*h2+4/9*b1*h2-
1/18*h2*b0,

b16 = 5/9*b3*h3*h1-4/9*b1*h3*h1-4/9*b4*h3*h1+1/18*b14*h3*h1+
2/9*b5*h3*h1-1/9*b12*h3*h1+1/18*h3*h1*b0+2/9*b8*h3*h1+
4/9*b4*h2-1/9*b13*h3*h1+1/9*b12*h2+1/3*b18-b4+4/3*b3+
2/3*b15-1/3*b0-h3*b3-1/2*h3*b0+h3*b1+1/2*b4*h3-2/9*b8*h2-
2/9*b5*h2-5/9*b3*h2+1/9*b13*h2-1/18*b14*h2+4/9*b1*h2-
1/18*h2*b0}
```

4.7 Tetraedro Macro

Dado T, um tetraedro macro, suponha que ele seja particionado em quatro mini-tetraedros quárticos, em um ponto interior G de T, cujas ordenadas de Bézier são $\{a_{ijkp}, b_{ijkp}, c_{ijkp}, d_{ijkp}\}$ com $0 \leq i, j, k, p \leq 4, i + j + k + p = 4$. G é escolhido como centróide de T, a fim de obter uma divisão de T mais homogênea. Nos quatro volumes é necessário que os seus interpolantes se encontrem com continuidade C^1, que sejam cúbicos, e cada derivada normal varie linearmente

ao longo das quatro faces do tetraedro macro T, notando que cada mini-tetraedro possui uma face de T, aqui dita face externa.

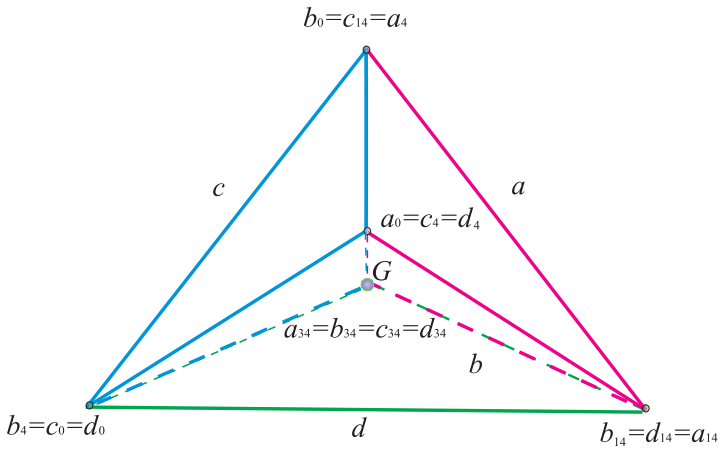

Figura 4.9 Tetraedro dividido em quatro malhas no ponto G.

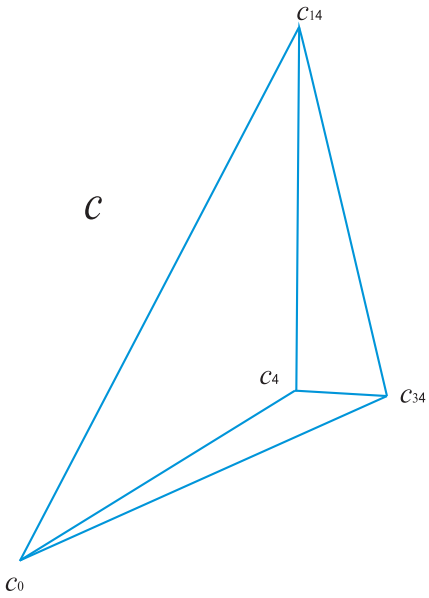

Figura 4.10 Malha c.

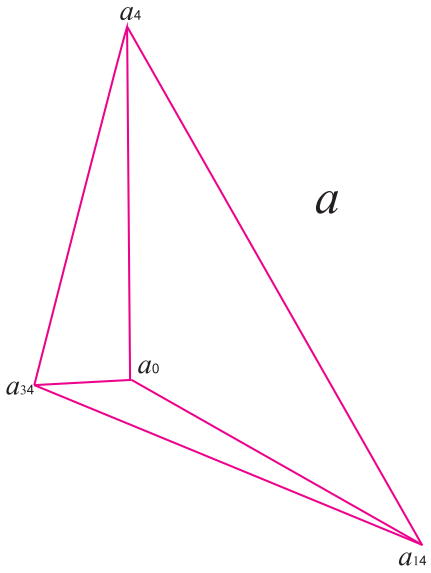

Figura 4.11 Malha a.

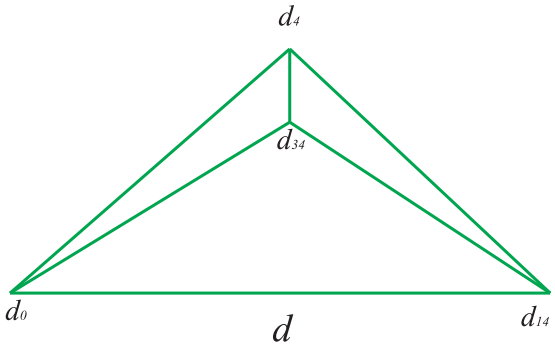

Figura 4.12 Malha d.

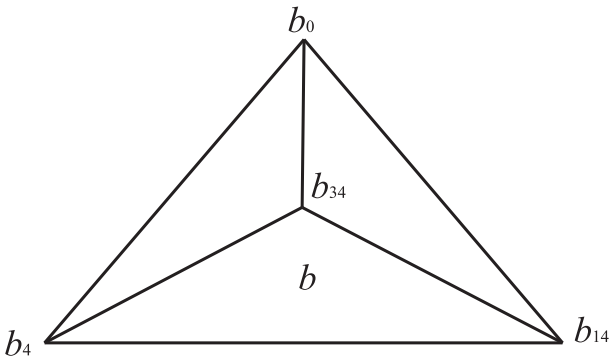

Figura 4.13 Malha b.

Pela continuidade C^0 entre esses quatro tetraedros ao longo das faces GV_iV_j, onde $i, j \in \{1,2,3,4\}$ e $j > i$, temos que

$$a_0 = c_4 = d_4,\ a_1 = c_8,\ a_2 = c_{11},\ a_3 = c_{13},\ a_4 = b_0 = c_{14},\ a_5 = d_8,$$
$$a_8 = b_5,\ a_9 = d_{11},\ a_{11} = b_9,\ a_{12} = d_{13},\ a_{13} = b_{12},\ a_{14} = b_{14} = d_{14},$$
$$a_{15} = c_{18} = d_{18},\ a_{16} = c_{21},\ a_{17} = c_{23},\ a_{18} = b_{15} = c_{24},\ a_{19} = d_{21},\ a_{21} = b_{19},$$
$$a_{22} = d_{23},\ a_{23} = b_{22},\ a_{24} = d_{24} = b_{24},\ a_{25} = c_{27} = d_{27},\ a_{26} = c_{29},\ a_{27} = b_{25} = c_{30},$$
$$a_{28} = d_{29},\ a_{29} = b_{28},\ a_{30} = b_{30} = d_{30},\ a_{31} = c_{32} = d_{32},\ a_{32} = b_{31} = c_{33},$$
$$a_{33} = b_{33} = d_{33},\ b_1 = c_{12},\ b_2 = c_9,\ b_3 = c_5,\ b_4 = c_0 = d_0,\ b_8 = d_5,\ b_{11} = d_9,$$
$$b_{13} = d_{12},\ b_{11} = d_9,\ b_{16} = c_{22},\ b_{17} = c_{19},\ b_{18} = c_{15} = d_{15},\ b_{21} = d_{19},\ b_{23} = d_{22},$$
$$b_{26} = c_{28},\ b_{27} = c_{25} = d_{25},\ b_{29} = d_{28},\ b_{32} = c_{31} = d_{31},\ c_1 = d_1,\ c_2 = d_2,\ c_3 = d_3,$$
$$c_{16} = d_{16},\ c_{17} = d_{17},\ c_{25} = d_{25},\ c_{26} = d_{26},\ c_{31} = d_{31},$$
$$a_{34} = b_{34} = c_{34} = d_{34}.$$

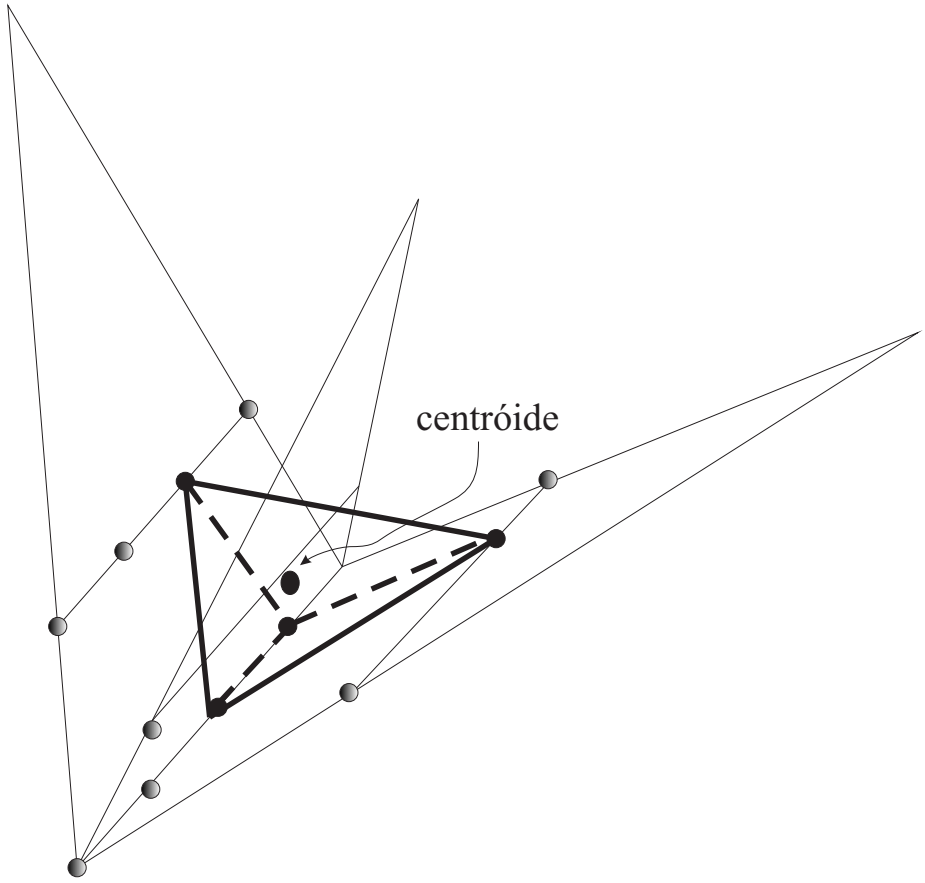

Figura 4.14 Tetraedros vizinhos.

Após estabelecer essas relações de igualdade para todos os retalhos, envolvendo os quatro tetraedros, podemos usar o Teorema 4.1 para enunciar equações que relacionem essas malhas. Para uma melhor visualização, observe a Figura 4.14.

$$
\begin{aligned}
b_{15} &= (b_0 + b_1 + b_5 + a_3)/4 & b_{16} &= (b_1 + b_2 + b_6 + c_{10})/4 \\
b_{17} &= (b_2 + b_3 + b_7 + c_6)/4 & b_{18} &= (b_3 + b_4 + b_8 + c_1)/4 \\
b_{19} &= (b_5 + b_6 + b_9 + a_7)/4 & b_{21} &= (b_7 + b_8 + b_{11} + d_6)/4 \\
b_{22} &= (b_9 + b_{10} + b_{12} + a_{10})/4 & b_{23} &= (b_{10} + b_{11} + b_{13} + d_{10})/4 \\
b_{24} &= (b_{12} + b_{13} + b_{14} + a_{12})/4 & b_{25} &= (b_{15} + b_{16} + b_{19} + a_{17})/4 \\
b_{26} &= (b_{16} + b_{17} + b_{20} + c_{20})/4 & b_{27} &= (b_{17} + b_{18} + b_{21} + c_{16})/4 \\
b_{28} &= (b_{19} + b_{20} + b_{22} + a_{20})/4 & b_{29} &= (b_{20} + b_{21} + b_{23} + d_{20})/4 \\
b_{30} &= (b_{22} + b_{23} + b_{24} + a_{22})/4 & b_{31} &= (b_{25} + b_{26} + b_{28} + a_{26})/4 \\
b_{32} &= (b_{26} + b_{27} + b_{29} + c_{26})/4 & b_{33} &= (b_{28} + b_{29} + b_{30} + a_{28})/4.
\end{aligned}
$$

$$b_{34} = (b_{31} + b_{32} + b_{33} + a_{31})/4.$$

Separemos essas Equações por níveis, de 1 a 4, de acordo com o afastamento da face externa:

Equações do nível 1:

$$\begin{aligned} b_{15} &= (b_0 + b_1 + b_5 + a_3)/4 & b_{16} &= (b_1 + b_2 + b_6 + c_{10})/4 \\ b_{17} &= (b_2 + b_3 + b_7 + c_6)/4 & b_{18} &= (b_3 + b_4 + b_8 + c_1)/4 \\ b_{19} &= (b_5 + b_6 + b_9 + a_7)/4 & b_{21} &= (b_7 + b_8 + b_{11} + d_6)/4 \\ b_{22} &= (b_9 + b_{10} + b_{12} + a_{10})/4 & b_{23} &= (b_{10} + b_{11} + b_{13} + d_{10})/4 \end{aligned} \tag{4.10}$$
$$b_{24} = (b_{12} + b_{13} + b_{14} + a_{12})/4$$

Equações do nível 2:

$$\begin{aligned} b_{25} &= (b_{15} + b_{16} + b_{19} + a_{17})/4 & b_{26} &= (b_{16} + b_{17} + b_{20} + c_{20})/4 \\ b_{27} &= (b_{17} + b_{18} + b_{21} + c_{16})/4 & b_{28} &= (b_{19} + b_{20} + b_{22} + a_{20})/4 \\ b_{29} &= (b_{20} + b_{21} + b_{23} + d_{20})/4 & b_{30} &= (b_{22} + b_{23} + b_{24} + a_{22})/4 \end{aligned} \tag{4.11}$$

Equações do nível 3:

$$b_{31} = (b_{25} + b_{26} + b_{28} + a_{26})/4 \qquad b_{32} = (b_{26} + b_{27} + b_{29} + c_{26})/4 \tag{4.12}$$
$$b_{33} = (b_{28} + b_{29} + b_{30} + a_{28})/4$$

Equação do nível 4:

$$b_{34} = (b_{31} + b_{32} + b_{33} + a_{31})/4.$$

O nível de complexidade, a quantidade de variáveis e o número de equações que as envolvem, nos leva ao uso de um software matemático [27] para solucionar nosso problema. Observe que os elementos do nível 1 são fundamentais para solucionar as Equações do nível 2. Essas por sua vez são necessárias para solucionar as do nível 3, que, claro, na Equação de nível 4, são imprescindíveis. Então cada nível é resolvido, aplicando o software em cada retalho, permitindo assim, o encontro de todos esses valores de controle.

4.8 Condições Suficientes para Não-negatividade da Malha Quártica de Bézier

Dada uma malha quártica de Bézier, $b_{\mathbf{i}}$, com $|\mathbf{i}| = 4$, num tetraedro T, desejamos estabelecer condições suficientes sobre suas ordenadas a fim de que essa malha seja não-negativa. Vejamos isso no seguinte resultado.

Teorema 4.3. *Seja T um tetraedro no espaço, dividido no seu centróide em quatro mini-tetraedros. Considere uma malha tetraédrica quártica de Bézier definida em cada um desses mini-tetraedros que têm como ordenadas de Bézier a_{ijkp}, b_{ijkp}, c_{ijkp}, d_{ijkp}, com $0 \leq i, j, k, p \leq 4$ e $i + j + k + p = 4$. Suponha que esses quatro tetraedros de Bézier formam uma malha tetraédrica Q de T e de continuidade C^1. Considerando $\mu \in \{a, b, c, d\}$, $l > 0$ e que*

1. $\forall\, m \in \{\mu_0, \mu_4, \mu_{14}\}, \quad m \geq l,$

2. $\forall\ m \in \{\mu_1, \mu_3, \mu_5, \mu_8, \mu_{12}, \mu_{13}\}, \quad m \geq \frac{l}{a}$,

3. $\mu_{20} \geq \frac{l}{3}\left(\frac{4}{a} - 1\right), \quad$ *sendo* $0 < a \leq 4$,

então

$$Q(x,y,z) \geq 0, \quad \forall (x,y,z) \in T.$$

Demonstração: Considere o tetraedro $T = GV_2V_3V_4$, onde $G = V_1 = (1,0,0,0)$, $V_2 = (0,1,0,0)$, $V_3 = (0,0,1,0)$ e $V_4 = (0,0,0,1)$. Na aresta V_3V_4 tome o ponto $P = (0,0,w,r)$ e na aresta V_2V_4 o ponto $Q = (0,w,0,r)$, onde $w + r = 1$ e $w, r \in [0,1]$. No triângulo definido por PGQ considere um ponto qualquer A, cuja combinação baricêntrica é dada por

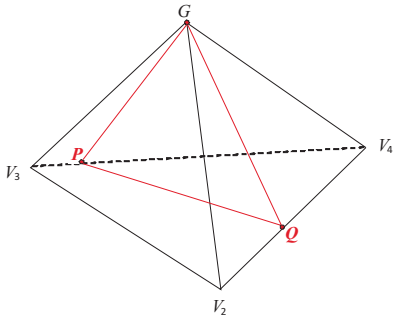

Figura 4.15 Tetraedro.

$$A = sP + tQ + lG, \text{ onde } s + t + l = 1, \text{ para } s,t,l \in [0,1]$$

e, portanto

$$A = s(0,0,w,r) + t(0,w,0,r) + (1-s-t)(1,0,0,0),$$

$$A = (1-s-t, tw, sw, (s+t)(1-w)).$$

Agora, considere o desenvolvimento da superfície $S(A)$ abaixo, usando a notação definida em (4.1) e a forma padrão de Bernstein. Além disso, podemos usar termos de $r^i_j(w)$, onde $i = 0,1,2,3,4$ e $j = 1,2,3,4,5$, de forma que,

$$\begin{aligned} S(A) &= t^4 r_1^4(w) + s^4 r_2^4(w) + 4st^3 r_3^4(w) + 4ts^3 r_4^4(w) + 6s^2t^2 r_5^4(w) + \\ &+ 4s^3(1-s-t) r_1^3(w) + 4t^3(1-s-t) r_2^3(w) + 12t(1-s-t)s^2 r_3^3(w) + \\ &+ 12st^2(1-s-t) r_4^3(w) + 6s^2(1-s-t)^2 r_1^2(w) + 6t^2(1-s-t)^2 r_2^2(w) + \\ &+ 12st(1-s-t)^2 r_3^2(w) + 4s(1-s-t)^3 r_1^0(w) + 4t(1-s-t)^3 r_2^1(w) + \\ &+ (1-s-t)^4 r_1^1(w). \end{aligned}$$

onde

$$
\begin{aligned}
r_1^4(w) &= b_4w^4 + 4b_3w^3(1-w) + 6b_2w^2(1-w)^2 + 4b_1w(1-w)^3 + b_0(1-w)^4,\\
r_2^4(w) &= b_{14}w^4 + 4b_{12}w^3(1-w) + 6b_9w^2(1-w)^2 + 4b_5w(1-w)^3 + b_0(1-w)^4,\\
r_3^4(w) &= b_0(1-w)^4 + 4(\tfrac{3}{4}b_1 + \tfrac{1}{4}b_5)w(1-w)^3 + 6(\tfrac{1}{2}b_2 + \tfrac{1}{2}b_6)w^2(1-w)^2 +\\
&\quad + 4(\tfrac{1}{4}b_3 + \tfrac{3}{4}b_7)w^3(1-w) + b_8w^4,\\
r_4^4(w) &= b_0(1-w)^4 + 4(\tfrac{3}{4}b_5 + \tfrac{1}{4}b_1)w(1-w)^3 + 6(\tfrac{1}{2}b_9 + \tfrac{1}{2}b_6)w^2(1-w)^2 +\\
&\quad + 4(\tfrac{3}{4}b_{10} + \tfrac{1}{4}b_{12})w^3(1-w) + b_{13}w^4,\\
r_5^4(w) &= b_0(1-w)^4 + 4(\tfrac{1}{2}b_1 + \tfrac{1}{2}b_5)(1-w)^3w + 6(\tfrac{1}{6}b_9 + \tfrac{1}{6}b_2 + \tfrac{2}{3}b_6)w^2(1-w)^2 +\\
&\quad + 4(\tfrac{1}{2}b_7 + \tfrac{1}{2}b_{10})w^3(1-w) + b_{11}w^4,\\
r_1^3(w) &= b_{24}w^3 + 3b_{22}w^2(1-w) + 3b_{19}w(1-w)^2 + b_{15}(1-w)^3,\\
r_2^3(w) &= b_{18}w^3 + 3b_{17}w^2(1-w) + 3b_{16}w(1-w)^2 + b_{15}(1-w)^3,\\
r_3^3(w) &= b_{23}w^3 + 3(\tfrac{1}{3}b_{22} + \tfrac{2}{3}b_{20})w^2(1-w) + 3(\tfrac{1}{3}b_{16} + \tfrac{2}{3}b_{19})w(1-w)^2 + b_{15}(1-w)^3,\\
r_4^3(w) &= b_{21}w^3 + 3(\tfrac{1}{3}b_{17} + \tfrac{2}{3}b_{20})w^2(1-w) + 3(\tfrac{1}{3}b_{19} + \tfrac{2}{3}b_{16})w(1-w)^2 + b_{15}(1-w)^3,\\
r_1^2(w) &= b_{30}w^2 + 2b_{28}w(1-w) + b_{25}(1-w)^2,\\
r_2^2(w) &= b_{27}w^2 + 2b_{26}w(1-w) + b_{25}(1-w)^2,\\
r_3^2(w) &= b_{29}w^2 + 2(\tfrac{1}{2}b_{28} + \tfrac{1}{2}b_{26})w(1-w) + b_{25}(1-w)^2,\\
r_1^1(w) &= b_{33}w + b_{31}(1-w),\\
r_2^1(w) &= b_{32}w + b_{31}(1-w),\\
r_1^0(w) &= b_{34}.
\end{aligned}
\tag{4.13}
$$

Agora, considerando

$$
B_{\mathbf{i}}^4(s,t) = \frac{4!}{i!\,j!(4-i-j)!}s^it^j(1-s-t)^{4-i-j},\ \text{onde } \mathbf{i} = ij(4-i-j),
$$

chegamos a

$$
\begin{aligned}
S(A) &= B_{040}^4(s,t)r_1^4(w) + B_{400}^4(s,t)r_2^4(w) + B_{004}^4(s,t)r_1^0(w) + B_{130}^4(s,t)r_3^4(w) +\\
&+ B_{310}^4(s,t)r_4^4(w) + B_{301}^4(s,t)r_1^3(w) + B_{220}^4(s,t)r_5^4(w) + B_{031}^4(s,t)r_2^3(w) +\\
&+ B_{211}^4(s,t)r_3^3(w) + B_{121}^4(s,t)r_4^3(w) + B_{202}^4(s,t)r_1^2(w) + B_{022}^4(s,t)r_2^2(w) +\\
&+ B_{112}^4(s,t)r_3^2(w) + B_{103}^4(s,t)r_1^1(w) + B_{013}^4(s,t)r_2^1(w).
\end{aligned}
$$

Dessa forma, como $\forall s,t \in [0,1], B_{\mathbf{i}}^4 \geq 0$, então para obter uma superfície S não-negativa é suficiente satisfazer a não-negatividade de $r_j^i(w)$, onde $i = 0,1,2,3,4$ e $j = 1,2,3,4,5$.

Considere a malha quártica com valores de controle $\mu_{\mathbf{i}}$ em T e nela tome $M_1 = \{\mu_0, \mu_4, \mu_{14}\}$, com $m \geq l$, sendo $l > 0\ \forall m \in M_1$.

Considere o conjunto $M_2 = \{\mu_1, \mu_3, \mu_5, \mu_8, \mu_{12}, \mu_{13}\}$ e suponha que

$$
m \geq \frac{l}{a}, a > 0, \forall m \in M_2.
$$

Pelas Equações relacionadas em (4.9) e os limites inferiores estabelecidos para os elementos de M_1 e M_2, temos que

$$m \geq \frac{l}{3}\left(\frac{4}{a} - 1\right), \forall m \in M_3, \text{ sendo } M_3 = \{\mu_2, \mu_6, \mu_7, \mu_9, \mu_{10}, \mu_{11}\}.$$

Pelas Equações do nível 1, estabelecidas em (4.10), e os limites inferiores considerados para M_3, os elementos de

$$M_4 = \{\mu_{15}, \mu_{18}, \mu_{24}\} \text{ e } M_5 = \{\mu_{16}, \mu_{17}, \mu_{19}, \mu_{21}, \mu_{22}, \mu_{23}\}$$

têm seus limites inferiores determinados, sejam eles:

$$m \geq \frac{l}{4}\left(\frac{3}{a} + 1\right), \forall m \in M_4,$$

$$m \geq \frac{l}{4}\left(\frac{5}{a} - 1\right), \forall m \in M_5.$$

Suponha agora que $\mu_{20} \geq \frac{l}{3}\left(\frac{4}{a} - 1\right)$. Usando as Equações de nível 2 (4.11) e os limites inferiores estabelecidos para M_4 e M_5, podemos dizer que

$$m \geq \frac{l}{8}\left(\frac{9}{a} - 1\right), \forall m \in M_6, \text{ sendo } M_6 = \{\mu_{25}, \mu_{27}, \mu_{30}\}$$

e

$$m \geq \frac{l}{4}\left(\frac{31}{6a} - \frac{7}{6}\right), \forall m \in M_7, \text{ sendo } M_7 = \{\mu_{26}, \mu_{28}, \mu_{29}\}.$$

Finalmente, usando o que foi definido para M_6 e M_7 e as Equações de nível 3 (4.12), podemos afirmar que

$$m \geq \frac{l}{4}\left(\frac{5}{a} - 1\right), \forall m \in M_8, \text{ sendo } M_8 = \{\mu_{31}, \mu_{32}, \mu_{33}\}$$

e, portanto, pela Equação de nível 4, $\mu_{34} \geq \frac{l}{4}\left(\frac{5}{a} - 1\right)$.

Como a ordenada de Bézier μ_{34} é o valor da superfície interpolante em G, μ_{34} precisa ser não-negativo para garantir a não-negatividade da malha tetraédrica de Bézier. Assim $a \leq 5$ precisa ser satisfeito.

Além do mais, se observarmos que $r_5^4(w)$ vale

$$b_0(1-w)^4 + 4\left(\frac{1}{2}b_1 + \frac{1}{2}b_5\right)(1-w)^3 w + 6\left(\frac{1}{6}b_9 + \frac{1}{6}b_2 + \frac{2}{3}b_6\right)w^2(1-w)^2 +$$

$$+4\left(\frac{1}{2}b_7 + \frac{1}{2}b_{10}\right)w^3(1-w) + b_{11}w^4,$$

e, portanto, se $b_{11} < 0$ teremos $r_5^4(w) < 0$, o que não queremos. Mas como para b_{11} ser não-negativo basta supor $a \leq 4$, então consideramos esta limitação superior e, dessa forma, podemos reestabelecer limites para os valores de controle dos conjuntos M_i, sendo $i = 2, 3, \cdots, 7$.

$$m \geq \frac{l}{4}, \forall m \in M_2, \quad m \geq 0, \forall m \in M_3, \quad m \geq \frac{7l}{16}, \forall m \in M_4, \quad m \geq \frac{l}{16}, \forall m \in M_5,$$

$$m \geq \frac{5l}{32}, \forall m \in M_6, \quad m \geq \frac{l}{32}, \forall m \in M_7, \quad m \geq \frac{l}{16}, \forall m \in M_8, \quad b_{34} \geq \frac{l}{16}.$$

Sendo assim todos os r_j^i com $i = 0,1,2,3,4$, $j = 1,2,3,4,5$ são não-negativos. Satisfeito isso, temos que $S(A)$ é não-negativa, como queríamos. A malha quártica de Bézier $S(\mathbf{u})$ é composta por essas superfícies de Bézier quárticas bivariadas, então ela será não-negativa.

Com os mesmos argumentos, os tetraedros de Bézier com ordenadas a_{ijkp}, c_{ijkp}, d_{ijkp} dos outros três mini-tetraedros são também não-negativos.

■

4.9 Construção da Superfície Interpolante C^1 e Não-Negativa

Considere os pontos (x_i, y_i, z_i, w_i), com $w_i > 0$, $i = 1,2,\cdots N$ e $(x_i, y_i, z_i) \neq (x_j, y_j, z_j)$ para $i \neq j$. Descreveremos a construção da função $F(x,y,z)$, com $F(x_i, y_i, z_i) = w_i$, que preserva a continuidade C^1 e a não-negatividade.

1. O domínio Ω da função F é o invólucro convexo de $\{V_i = (x_i, y_i, z_i) : i = 1, \cdots, N\}$. Os pontos V_i, $i = 1,2\cdots, N$, são usados como vértices da tetraedrização do domínio Ω. A triangulação de Delaunay (conforme [3]) é usada para tetrangular o domínio Ω.

2. A estimação das derivadas parciais de primeira ordem, isto é, F_x, F_y e F_z em cada $V_i(x_i, y_i, z_i)$, para a superfície F, é obtida usando o método desenvolvido em [4].

3. Para todo tetraedro macro no domínio, uma malha tetraédrica será gerada.

Esse estudo se concentrará no terceiro passo e, para isso, será analisado como construir em cada tetraedro macro uma malha tetrangular C^1 não-negativa. O processo começa com a subdivisão de cada tetraedro macro do domínio em quatro mini-tetraedros, através do seu centróide e, em seguida, um tetraedro quártico de Bézier é construído em cada mini-tetraedro. A determinação das ordenadas de Bézier destes quatro mini-tetraedros, $\{a_{ijkp}\}$, $\{b_{ijkp}\}$, $\{c_{ijkp}\}$ e $\{d_{ijkp}\}$, é descrita como segue.

Após as derivadas parciais de primeira ordem F_x, F_y e F_z em cada vértice $V_i(x_i, y_i, z_i)$ no domínio tetrangulado Ω serem estimadas, em cada retalho Q no tetraedro macro, a derivada direcional em seu vértice V_i na direção da aresta $\mathbf{d}_{ij}$, de V_i para V_j, é calculada por

$$D_{\mathbf{d}_{ij}}(V_i) = (x_j - x_i)\frac{\partial F}{\partial x}(V_i) + (y_j - y_i)\frac{\partial F}{\partial y}(V_i) + (z_j - z_i)\frac{\partial F}{\partial z}(V_i).$$

Baseado nos pontos dados, as ordenadas dos vértices de cada tetraedro macro são determinadas. Por exemplo, no tetraedro macro T da Figura 4.9:

$$a_0 = c_4 = d_4 = F(V_1) = w_1,$$
$$b_4 = c_0 = d_0 = F(V_2) = w_2,$$
$$b_{14} = a_{14} = d_{14} = F(V_3) = w_3,$$
$$b_0 = c_{14} = a_4 = F(V_4) = w_4.$$

Partindo das derivadas parciais estimadas em cada vértice e das ordenadas nos vértices, as ordenadas de Bézier pertencentes ao conjunto M_2 são determinadas através de (4.8). No entanto, estes valores podem não garantir que a malha resultante seja não-negativa. Para garantir isso é preciso impor condições nestas ordenadas, isto é,

$$m \geq \frac{l}{4}, \forall m \in M_2,$$

com $l = min\{F(V_1), F(V_2), F(V_3), F(V_4)\}$, como no Teorema 4.3. Para alcançá-la, a derivada parcial de primeira ordem em V_i, $i = 1,2,3,4$, deve ser modificada, se necessário. A modificação das derivadas F_x, F_y e F_z nos vértices V_i é feita inserindo um escalar em cada um deles como um fator positivo α, sendo $\alpha \leq 1$, levando em consideração todos os retalhos tetrangulares no tetraedro macro dividindo aquele vértice.

Vejamos como se dará esse procedimento, análogo ao caso bivariado.

Tome O para ser o vértice em seu domínio tetrangulado Ω e θ_i, $i = 1,2,\cdots,k$ para ser cada tetraedro macro na tetrangulação que tem O como um vértice.

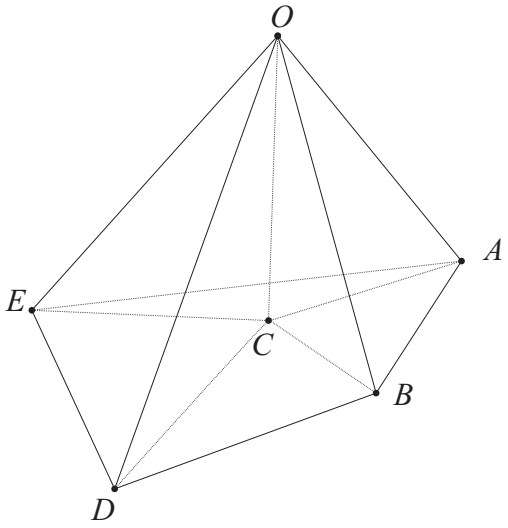

Figura 4.16 Tetraedros na tetraedrização do domínio Ω, com vértice comum O.

Considere o tetraedro θ_1 no domínio Ω (veja Figura 4.16) e o limite inferior $\frac{l_{\theta_1}}{4}$, onde

$$l_{\theta_1} = min\{F(O), F(A), F(B), F(C)\}.$$

Denote as derivadas direcionais em O na direção das arestas OA, OB e OC por $D_{d_{OA}}$, $D_{d_{OB}}$ e $D_{d_{OC}}$, respectivamente. Os fatores escalares α_{OA}, α_{OB} e α_{OC} são definidos como segue.

Se $F(O) + \frac{1}{4}D_{d_{OA}} \geq \frac{l_{\theta_1}}{4}$, então $\alpha_{OA} = 1$, caso contrário, α_{OA} é determinado pela equação

$$F(O) + \alpha_{OA}\frac{1}{4}D_{d_{OA}} = \frac{l_{\theta_1}}{4}.$$

Se $F(O) + \frac{1}{4}D_{d_{OB}} \geq \frac{l_{\theta_1}}{4}$, então $\alpha_{OB} = 1$, caso contrário, α_{OB} é definido pela equação

$$F(O) + \alpha_{OB}\frac{1}{4}D_{d_{OB}} = \frac{l_{\theta_1}}{4}.$$

Similarmente se $F(O) + \frac{1}{4}D_{d_{OC}} \geq \frac{l_{\theta_1}}{4}$, então $\alpha_{OC} = 1$, caso contrário, α_{OC} é determinado pela equação

$$F(0) + \alpha_{OC}\frac{1}{4}D_{d_{OC}} = \frac{l_{\theta_1}}{4}.$$

Daí define-se $\alpha_{\theta_1} = \min\{\alpha_{OA}, \alpha_{OB}, \alpha_{OC}\}$. Usando o mesmo método, α_{θ_i}, $i = 2, 3, \cdots, k$, são encontrados. Para todas as ordenadas de Bézier adjacentes a O satisfazerem as condições, escolhemos

$$\alpha_0 = \min\{\alpha_{\theta_1}, \alpha_{\theta_2}, \cdots, \alpha_{\theta_k}\}.$$

Se $\alpha_0 < 1$, então as derivadas parciais primeiras em O são multiplicadas pelo fator α_0 e as ordenadas de Bézier adjacentes a O são determinadas de acordo com isso. O processo citado anteriormente é repetido em todos os vértices V_i no domínio Ω. Então todas as ordenadas de Bézier ao longo das arestas dos tetraedros macros podem ser determinadas da forma citada.

Então, resolvido o problema de ajuste das derivadas, com o propósito de garantir uma malha não-negativa, voltemos à determinação das ordenadas de Bézier dos outros conjuntos.

Observe que M_4 também tem seus elementos determinados pela derivada direcional nos vértices do tetraedro, mas pela continuidade C^1, as Equações de (4.10) determinam essas ordenadas, e para isso usam o limite inferior estipulado para as ordenadas de M_2, o que nos leva a valores diferentes.

Sendo assim, usando que $m \geq l, \forall m \in M_1$ e que $m \geq \frac{l}{4}, \forall m \in M_2$, então $m \geq 0, \forall m \in M_3$, pela redução do grau na malha (veja Equações (4.9)).

Pela propriedade da continuidade C^1, as Equações de (4.10), determinamos as ordenadas de M_4 e M_5

$$m \geq \frac{7l}{16} \geq \frac{l}{4}, \forall m \in M_4 \quad \text{e} \quad m \geq \frac{l}{16}, \forall m \in M_5.$$

Desses e pelas Equações de (4.11) determinamos que

$$m \geq \frac{5l}{32}, \forall m \in M_6.$$

O Teorema 4.3 impõe um limite inferior para as ordenadas de Bézier do tipo μ_{20} ($\mu_{20} \geq 0$). Aqui, já que as ordenadas de Bézier de M_2 do tetraedro macro são fixadas, poderia se usar esse valor atual para relaxar o limite nas ordenadas de Bézier μ_{20}, sugerido no Teorema 4.3.

Considere $A = b_{15}$, $B = \frac{1}{3}b_{16} + \frac{2}{3}b_{19}$, $D = b_{23}$, todos positivos, e suponha que $C = \frac{1}{3}b_{22} + \frac{2}{3}b_{20}$ seja negativo, ou seja tome b_{20} negativo. Então pelo Teorema 3.1 e analisando $r_3^3(x)$ em (4.13), temos que

$$\Phi = 3B^2C^2 + 6ABCD - 4(AC^3 + B^3D) - A^2D^2 \leq 0 \Rightarrow r_3^3(x) \geq 0 \text{ em } [0,1].$$

Após substituir os termos de A, B, D em Φ, encontramos a seguinte expressão:

$$\Phi = -\frac{7l}{4}C^3 + \frac{3l^2}{256}C^2 + \frac{21l^3}{2048}C - \frac{53l^4}{65536}.$$

Sabendo que $C = \frac{l}{3}\left[\frac{1}{16} + \frac{2b_{20}}{l}\right]$ e usando [27], chegamos que para $b_{20} \geq \frac{-l}{6}$ temos $r_3^3(w) \geq 0$ em [0,1].

Os valores das ordenadas de Bézier de $M_7 = \{\mu_{26}, \mu_{28}, \mu_{29}\}$ foram definidas com o limite inferior $\frac{-5l}{96}$, através das Equações de (4.11), mas ao substituí-lo em $r_3^2(w)$ chegamos à conclusão que essa função assume valores negativos em [0,1]. Portanto outros testes foram feitos, elevando o valor de μ_{20} e o limite inferior $\frac{-l}{16}$ nos levou a:

$$m \geq 0, \forall m \in M_7, \quad \text{e} \quad m \geq \frac{5l}{128}, \forall m \in M_8 \text{ ou } m = b_{34}.$$

Dessa forma, $r_3^3(w), r_4^3(w), r_1^2(w), r_2^2(w), r_3^2(w), r_1^1(w), r_2^1(w), r_1^0(w)$ são todos não-negativos e a malha quártica de Bézier com suas ordenadas $\{a_\mathbf{i}\}, \{b_\mathbf{i}\}, \{c_\mathbf{i}\}, \{d_\mathbf{i}\}$ é não-negativa e é C^1 ao longo da face fronteira comum com a malha adjacente.

Denote a malha tetrangular construída desta forma no tetraedro macro T como Q_T. Então a superfície interpolante F, que tem a continuidade C^1 e preserva a não-negatividade, pode ser definida no domínio tetrangulado como $F|T = Q_T$, para todo tetraedro macro T no domínio.

Capítulo 5

Resultados Experimentais

Para ilustrar a metodologia estabelecida no Capítulo 4, denotada por Bézier nas análises abaixo, usaremos as seguintes funções de teste, $F_1, F_2, \cdots, F_6$, da mesma forma que Nielson em [8].

$$
\begin{aligned}
F_1(x,y,z) &= 0,75 \cdot \exp\left[-\frac{(9x-2)^2+(9y-2)^2+(9z-2)^2}{4}\right] + \\
&+ 0,75 \cdot \exp\left[-\frac{(9x+1)^2}{49} - \frac{(9y+2)^2}{10} - \frac{(9z+1)^2}{10}\right] + \\
&+ 0,5 \cdot \exp\left[-\frac{(9x-7)^2+(9y-3)^2+(9z-5)^2}{4}\right] - \\
&- 0,2 \cdot \exp\left[-(9x-4)^2-(9y-7)^2-(9z-5)^2\right], \\
F_2(x,y,z) &= \{\tanh(9z-9x-9y)+1\}/9, \\
F_3(x,y,z) &= [(1,25+\cos(5,4y))\cos(6z)]/\left\{6+6(3x-1)^2\right\}, \\
F_4(x,y,z) &= \exp\left\{-\frac{81}{16}\left[(x-0,5)^2+(y-0,5)^2+(z-0,5)^2\right]\right\}/3, \\
F_5(x,y,z) &= \exp\left\{-\frac{81}{4}\left[(x-0,5)^2+(y-0,5)^2+(z-0,5)^2\right]\right\}/3, \\
F_6(x,y,z) &= \sqrt{64-81\left[(x-0,5)^2+(y-0,5)^2+(z-0,5)^2\right]}/9-0,5.
\end{aligned}
$$

Um programa foi construído (Apêndice A) e é implementado com o propósito de gerar a função interpolante C^1 não-negativa e comparar seus resultados com os de outros dois métodos de interpolação, Shepard e Linear por Partes. Procederemos da forma descrita a seguir.

Tomamos quinze pontos trivariados e espalhados. Deles tomamos o invólucro convexo com trinta e quatro tetraedros, formando assim um domínio tetrangulado, com base na tetrangulação de Delaunay e através do Qhull (veja [26]). Cada tetraedro é dividido em quatro mini-tetraedros e a interpolação pelo método de Bézier é realizada. Simultaneamente as interpolações por Shepard e Linear por Partes também são feitas sobre os pontos volumétricos espalhados. Uma função é gerada por cada um desses métodos e as comparamos com os resultados fornecidos pela função verdadeira, no caso $F_1, F_2, \cdots, F_6$, citadas anteriormente. Como se dá essa comparação?

Sobre o invólucro convexo determinamos um cubo de lado 1 unidade de comprimento, que o contém e que é igualmente cortado por N planos nas direções x, y e z, gerando N^3 cubinhos

de mesmas medidas. Nessa grade, o cubinho que intersectar algum tetraedro do invólucro tem seus vértices avaliados pelas seis funções $F_1, F_2, \cdots, F_6$ e pelas funções interpolantes, geradas pelos três métodos que estão sendo comparados. Essa comparação é feita através do Erro Médio da Raiz Quadrada (RMS - Root Mean Square), que pode ser visto na Equação 5.1, onde F representa a função verdadeira, enquanto A representa a função a ser testada, a *aproximação* de F.

$$RMS = \sqrt{\frac{\sum_{i=0}^{N}\sum_{j=0}^{N}\sum_{k=0}^{N}\left[F\left(\frac{i}{N},\frac{j}{N},\frac{k}{N}\right)-A\left(\frac{i}{N},\frac{j}{N},\frac{k}{N}\right)\right]^2}{(N+1)(N+1)(N+1)}} \tag{5.1}$$

A tela inicial do programa executado pode ser visto na Figura 5.1. Observamos que o número de planos é denotado por *células* e o considerado foi $N = 10$. Pode-se ver também que na parte direita tem-se os resultados fornecidos pelo RMS para cada método. Na primeira linha, para a função F_1, temos o erro com respeito aos métodos desenvolvidos nessa análise, o de Bézier, em seguida o resultado obtido por Shepard e, naturalmente, o resultado obtido pelo método Linear por Partes.

O programa gera a diferença das imagens fornecidas por Bézier, Shepard e Linear por Partes. A cor azul representa uma diferença positiva (por exemplo, função gerada por Bézier menos a função gerada por Linear é positiva); o vermelho, caso contrário; o preto representa uma diferença nula. Uma cor vermelha pura ou azul pura significa que a diferença absoluta ficou superior em mais de 25 porcento do valor máximo das funções nos dados a serem interpolados. O invólucro convexo e tetrangulado é intersectado por um plano ortogonal a um dos três eixos e a imagem é então reproduzida.

A Tabela 5.1 resume os valores numéricos encontrados.

Função	RMS		
	Bézier	**Shepard**	**Linear**
F_1	0,0639	0,0899	0,0637
F_2	0,0303	0,0455	0,0303
F_3	0,0894	0,0732	0,0506
F_4	0,0639	0,1018	0,0779
F_5	0,0534	0,0672	0,0608
F_6	0,2833	0,1323	0,1001

Tabela 5.1 Valor do RMS obtido por função e método de interpolação.

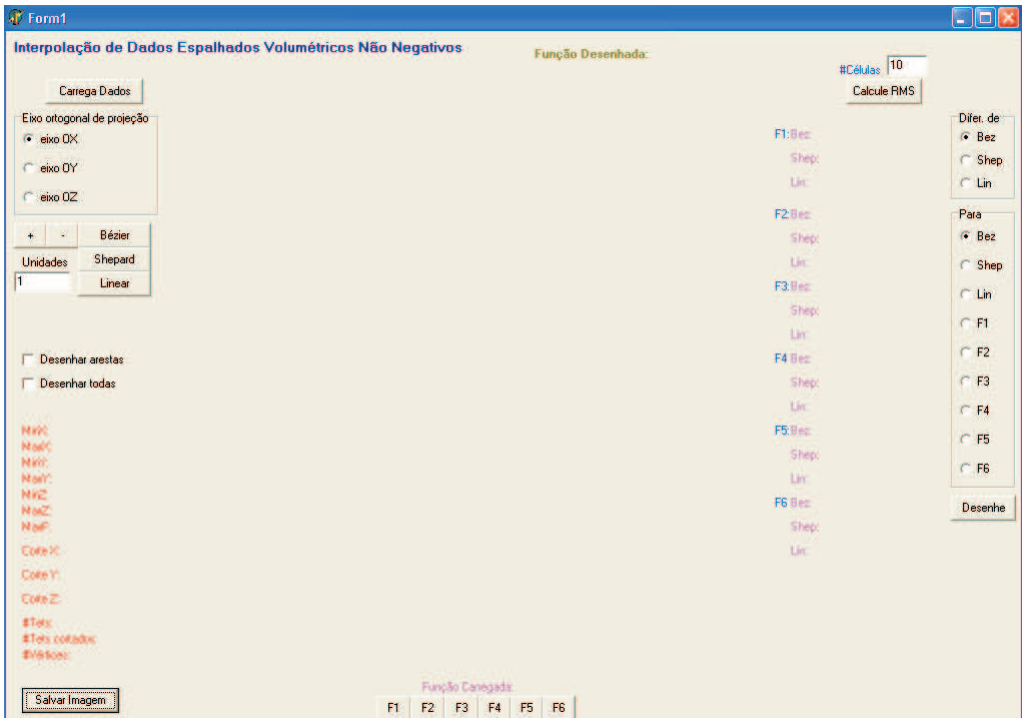

Figura 5.1 Programa para comparar funções interpoladas pelos métodos de Bézier, Shepard e Linear por Partes.

Considere a função F_1, onde o erro estimado pelo RMS para o método de Bézier foi de 0,0639, para o método de Shepard foi de 0,0899 e para o Linear por Partes, erro de 0,0637. Veja as Figuras 5.2, 5.3 e 5.4, que representam a intersecção entre a função indicada e um plano ortogonal a x.

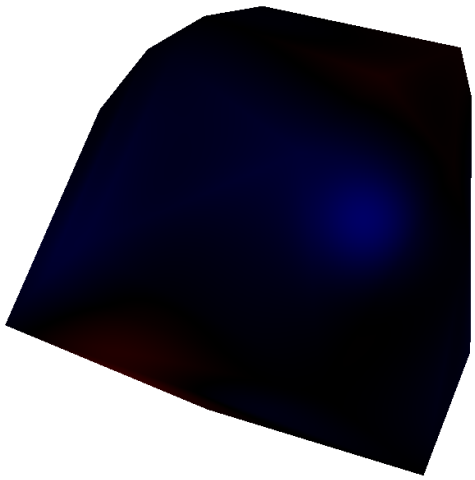

Figura 5.2 Imagem da diferença entre a função interpolada pelo método de Bézier e a função F_1 .

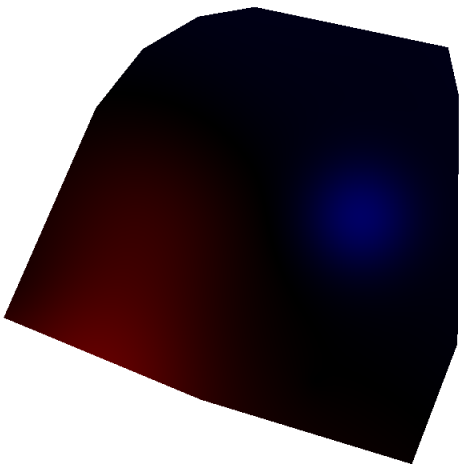

Figura 5.3 Imagem da diferença entre a função interpolada pelo método de Shepard e a função F_1.

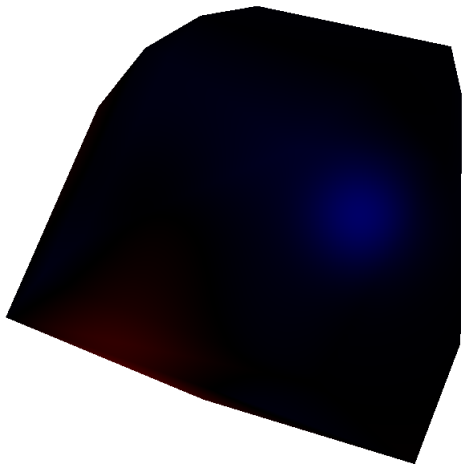

Figura 5.4 Imagem da diferença entre a função interpolada pelo método Linear por Partes e a função F_1.

As Figuras 5.5, 5.6, 5.7 representam a intersecção entre a função indicada e um plano ortogonal a y.

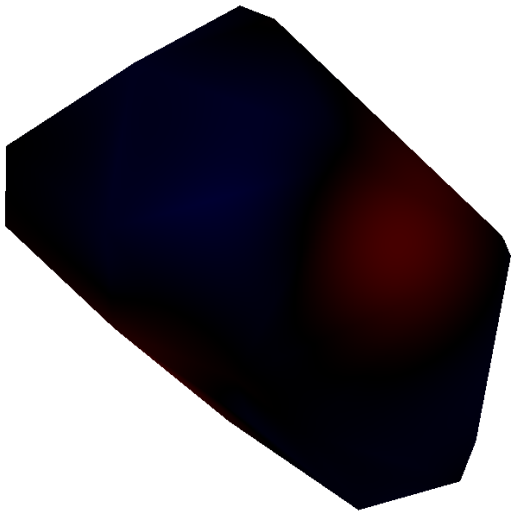

Figura 5.5 Imagem da diferença entre a função interpolada pelo método de Bézier e a função F_1 .

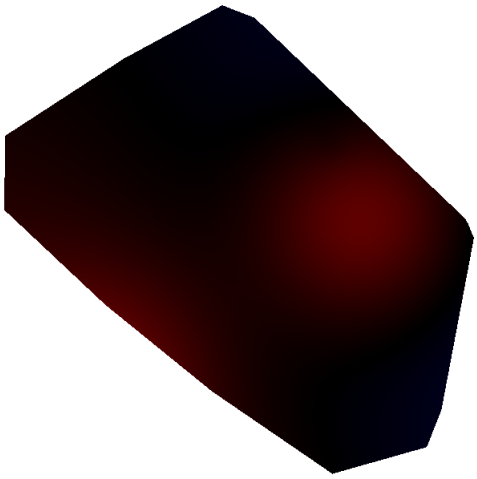

Figura 5.6 Imagem da diferença entre a função interpolada pelo método de Shepard e a função F_1.

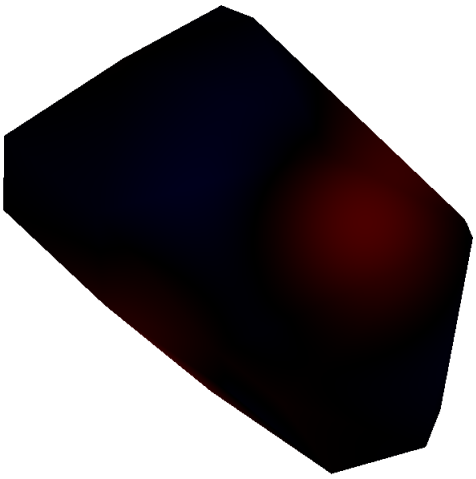

Figura 5.7 Imagem da diferença entre a função interpolada pelo método Linear por Partes e a função F_1.

As Figuras 5.8, 5.9, 5.10, representam a intersecção entre a função indicada e um plano ortogonal a z.

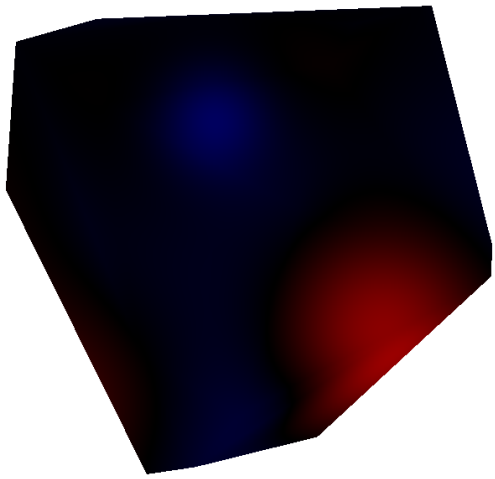

Figura 5.8 Imagem da diferença entre a função interpolada pelo método de Bézier e a função F_1 .

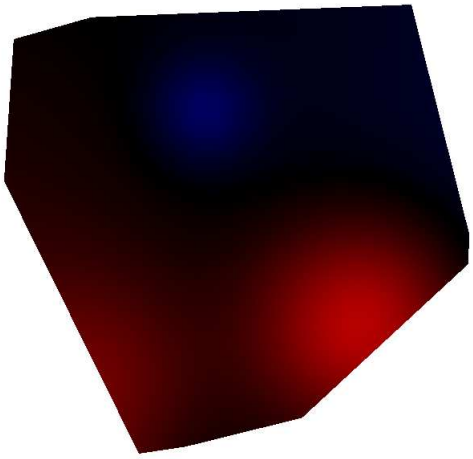

Figura 5.9 Imagem da diferença entre a função interpolada pelo método de Shepard e a função F_1.

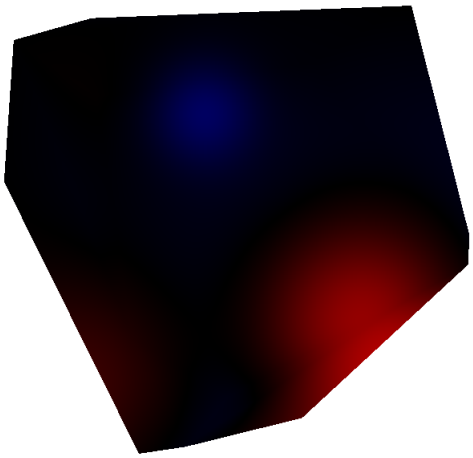

Figura 5.10 Imagem da diferença entre a função interpolada pelo método Linear por Partes e a função F_1.

Considere a função F_2, onde o erro estimado pelo RMS para o método de Bézier foi de 0,0303, para o método de Shepard foi de 0,0455 e para o Linear por Partes, também erro de 0,0303. Veja as Figuras 5.11, 5.12, 5.13, que representam a intersecção entre a função indicada e um plano ortogonal a x.

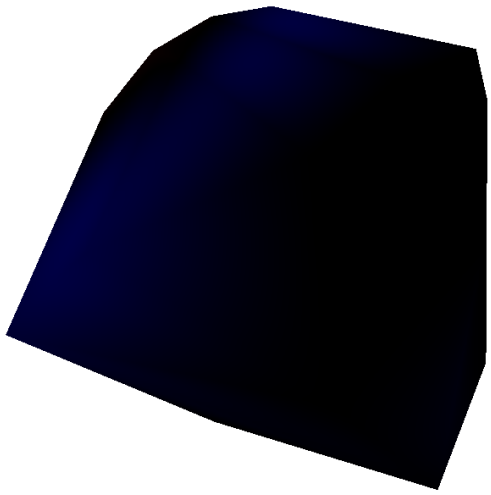

Figura 5.11 Imagem da diferença entre a função interpolada pelo método de Bézier e a função F_2 .

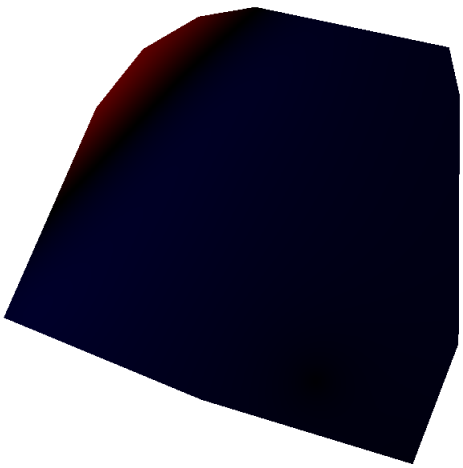

Figura 5.12 Imagem da diferença entre a função interpolada pelo método de Shepard e a função F_2.

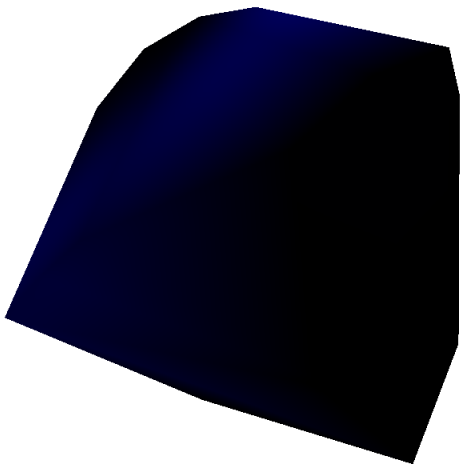

Figura 5.13 Imagem da diferença entre a função interpolada pelo método Linear por Partes e a função F_2.

As Figuras 5.14, 5.15, 5.16 representam a intersecção entre a função indicada e um plano ortogonal a y.

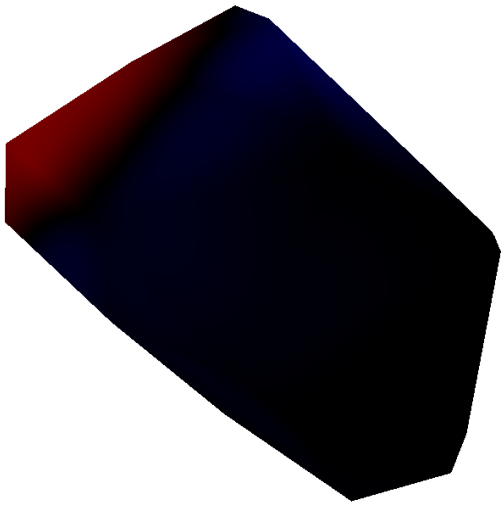

Figura 5.14 Imagem da diferença entre a função interpolada pelo método de Bézier e a função F_2 .

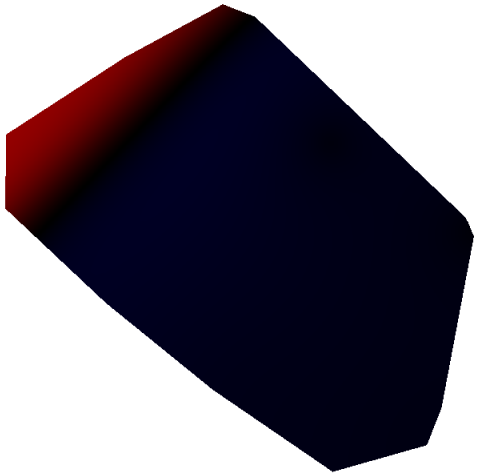

Figura 5.15 Imagem da diferença entre a função interpolada pelo método de Shepard e a função F_2.

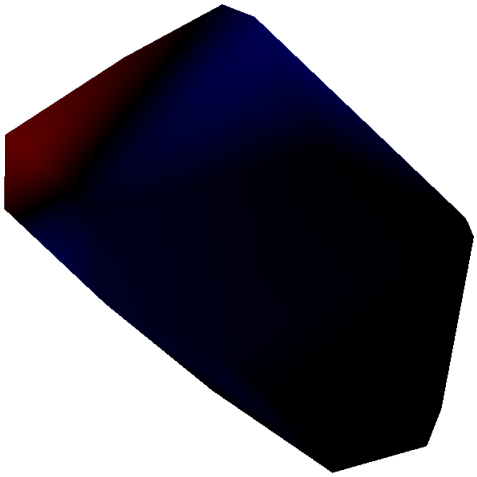

Figura 5.16 Imagem da diferença entre a função interpolada pelo método Linear por Partes e a função F_2.

As Figuras 5.17, 5.18, 5.19 representam a intersecção entre a função indicada e um plano ortogonal a z.

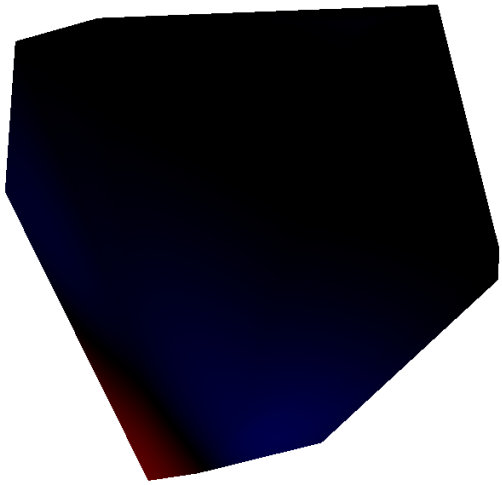

Figura 5.17 Imagem da diferença entre a função interpolada pelo método de Bézier e a função F_2 .

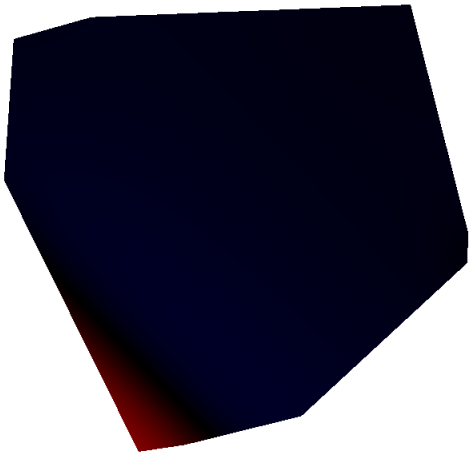

Figura 5.18 Imagem da diferença entre a função interpolada pelo método de Shepard e a função F_2.

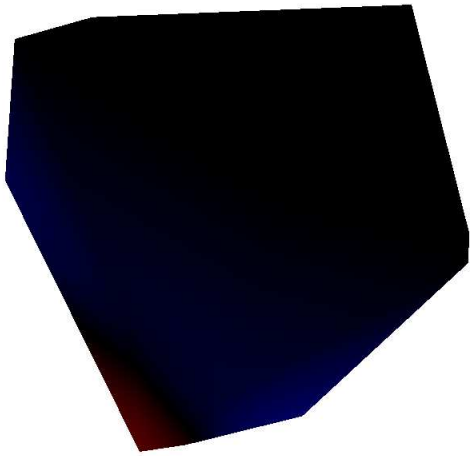

Figura 5.19 Imagem da diferença entre a função interpolada pelo método Linear por Partes e a função F_2.

Agora, considere a função F_3, onde o erro estimado pelo RMS para o método de Bézier foi de 0,0894, para o método de Shepard foi de 0,0732 e para o Linear por Partes, erro de 0,0506. Veja as Figuras 5.20, 5.21, 5.22, que representam a intersecção entre a função indicada e um plano ortogonal a x.

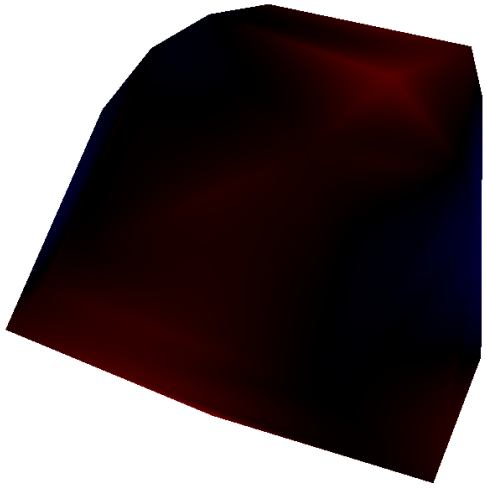

Figura 5.20 Imagem da diferença entre a função interpolada pelo método de Bézier e a função F_3 .

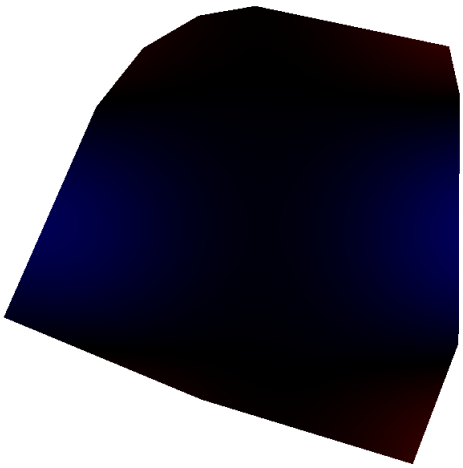

Figura 5.21 Imagem da diferença entre a função interpolada pelo método de Shepard e a função F_3.

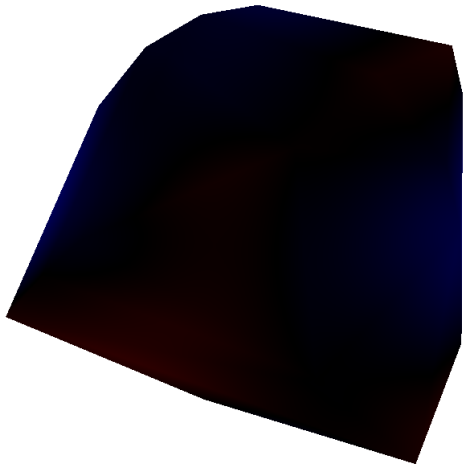

Figura 5.22 Imagem da diferença entre a função interpolada pelo método Linear por Partes e a função F_3.

Agora, as Figuras 5.23, 5.24, 5.25, que representam a intersecção entre a função indicada e um plano ortogonal a y.

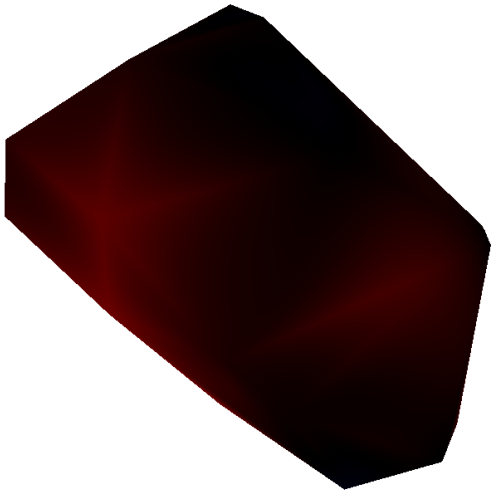

Figura 5.23 Imagem da diferença entre a função interpolada pelo método de Bézier e a função F_3 .

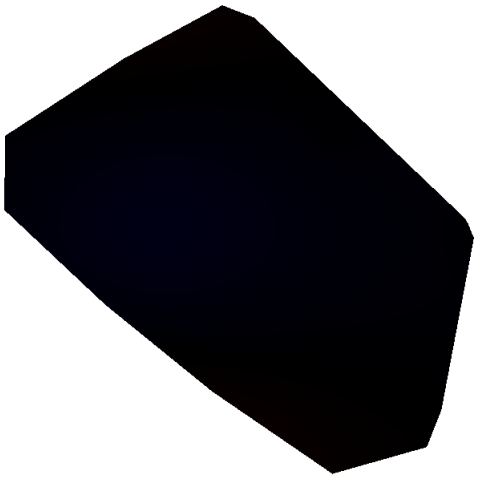

Figura 5.24 Imagem da diferença entre a função interpolada pelo método de Shepard e a função F_3.

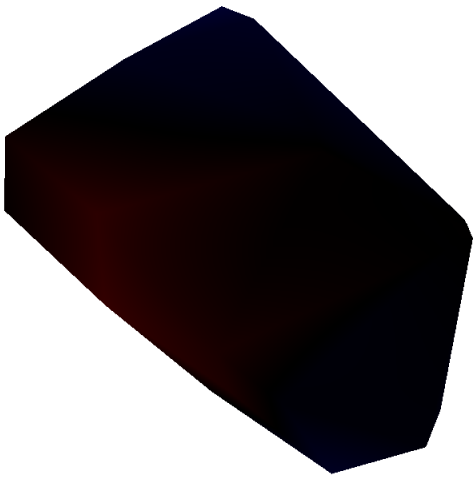

Figura 5.25 Imagem da diferença entre a função interpolada pelo método Linear por Partes e a função F_3.

Agora, as Figuras 5.26, 5.27, 5.28, que representam a intersecção entre a função indicada e um plano ortogonal a z.

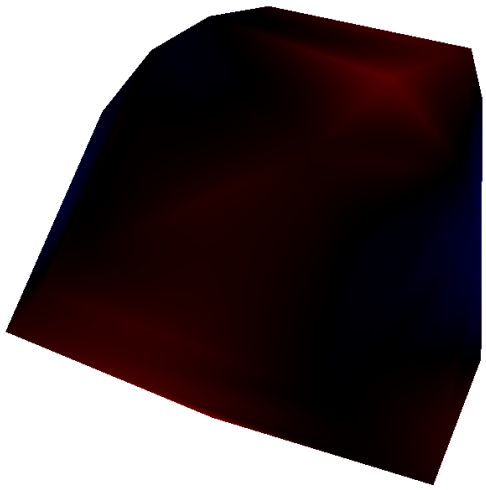

Figura 5.26 Imagem da diferença entre a função interpolada pelo método de Bézier e a função F_4 .

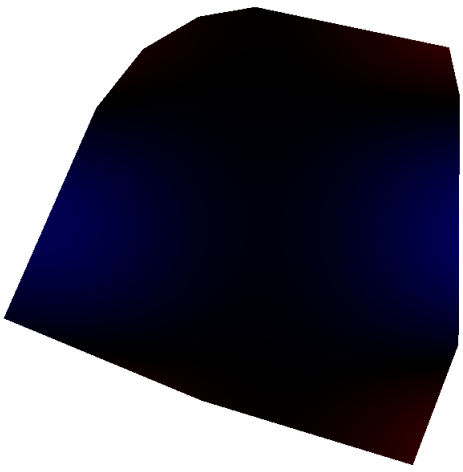

Figura 5.27 Imagem da diferença entre a função interpolada pelo método de Shepard e a função F_4.

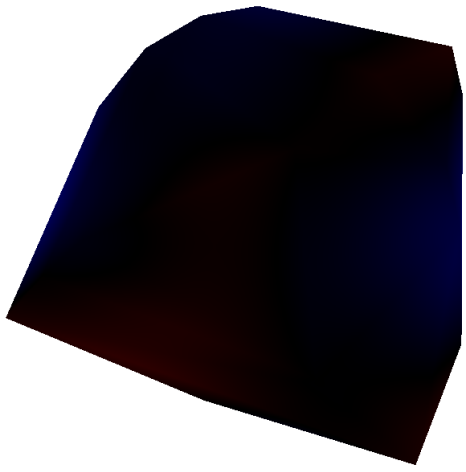

Figura 5.28 Imagem da diferença entre a função interpolada pelo método Linear por Partes e a função F_4.

Agora, considere a função F_4, onde o erro estimado pelo RMS para o método de Bézier foi de 0,0639, para o método de Shepard foi de 0,1018 e para o Linear por Partes, erro de 0,0779. Veja as Figuras 5.29, 5.30, 5.31 que representam a intersecção entre a função indicada e um plano ortogonal a x.

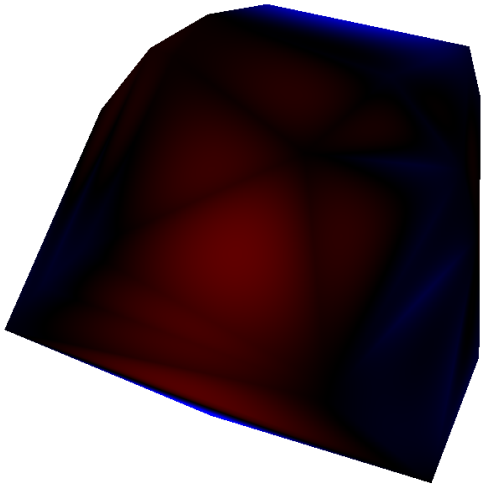

Figura 5.29 Imagem da diferença entre a função interpolada pelo método de Bézier e a função F_4 .

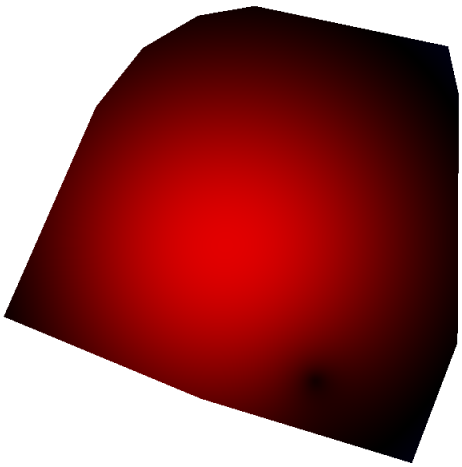

Figura 5.30 Imagem da diferença entre a função interpolada pelo método de Shepard e a função F_4.

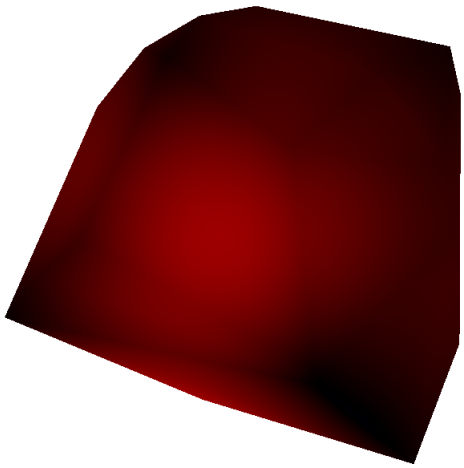

Figura 5.31 Imagem da diferença entre a função interpolada pelo método Linear por Partes e a função F_4.

Agora, as Figuras 5.32, 5.33, 5.34, que representam a intersecção entre a função indicada e um plano ortogonal a y.

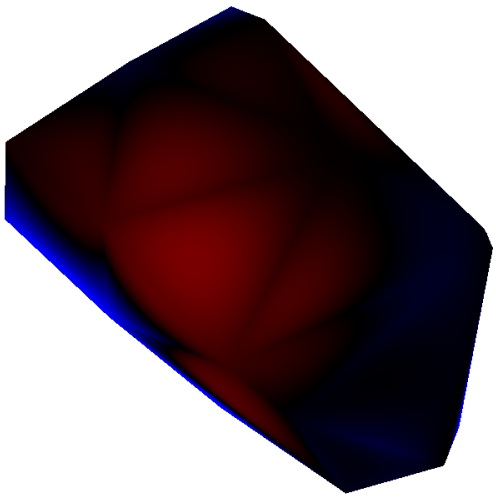

Figura 5.32 Imagem da diferença entre a função interpolada pelo método de Bézier e a função F_4 .

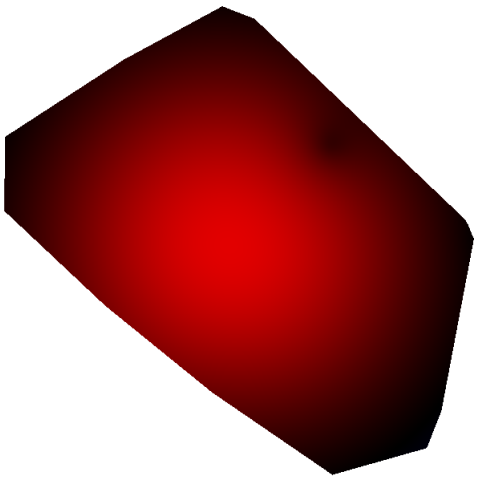

Figura 5.33 Imagem da diferença entre a função interpolada pelo método de Shepard e a função F_4.

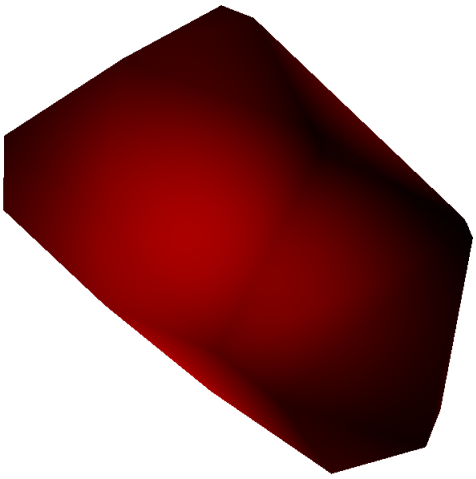

Figura 5.34 Imagem da diferença entre a função interpolada pelo método Linear por Partes e a função F_4.

As Figuras 5.35, 5.36, 5.37, representam a intersecção entre a função indicada e um plano ortogonal a z.

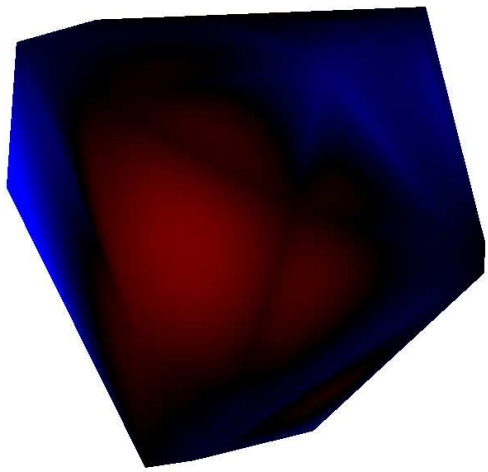

Figura 5.35 Imagem da diferença entre a função interpolada pelo método de Bézier e a função F_4 .

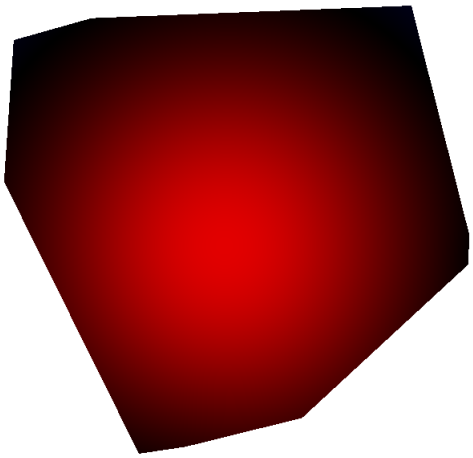

Figura 5.36 Imagem da diferença entre a função interpolada pelo método de Shepard e a função F_4.

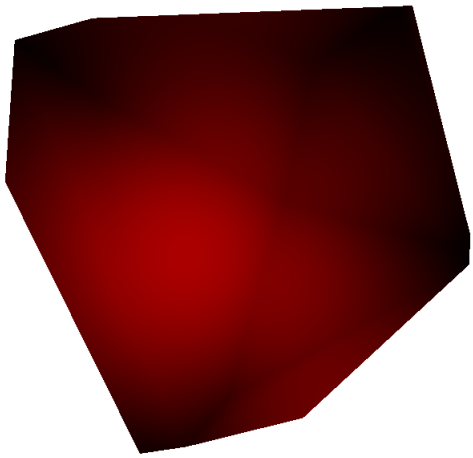

Figura 5.37 Imagem da diferença entre a função interpolada pelo método Linear por Partes e a função F_4.

Considere a função F_5, onde o erro estimado pelo RMS para o método de Bézier foi de 0,0534, para o método de Shepard foi de 0,0672 e para o Linear por Partes, erro de 0,0608. Veja as Figuras 5.38, 5.39, 5.40, que representam a intersecção entre a função indicada e um plano ortogonal a x.

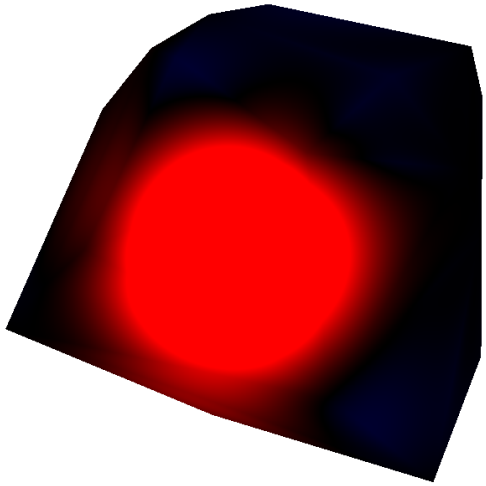

Figura 5.38 Imagem da diferença entre a função interpolada pelo método de Bézier e a função F_5 .

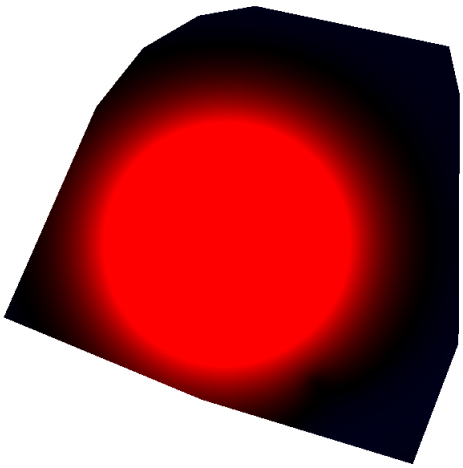

Figura 5.39 Imagem da diferença entre a função interpolada pelo método de Shepard e a função F_5.

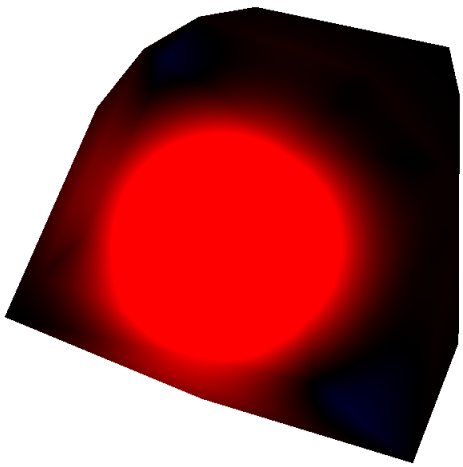

Figura 5.40 Imagem da diferença entre a função interpolada pelo método Linear por Partes e a função F_5.

As Figuras 5.41, 5.42, 5.43 representam a intersecção entre a função indicada e um plano ortogonal a y.

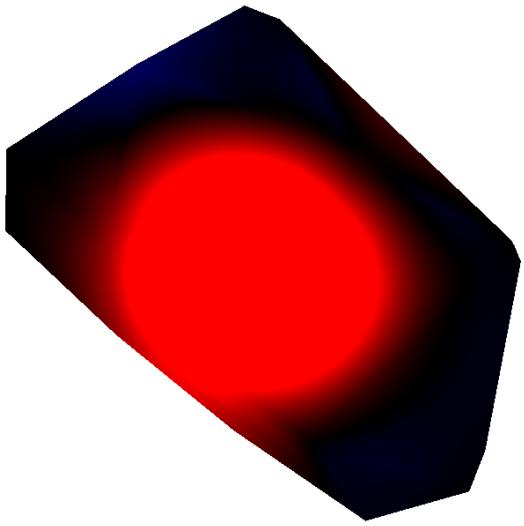

Figura 5.41 Imagem da diferença entre a função interpolada pelo método de Bézier e a função F_5 .

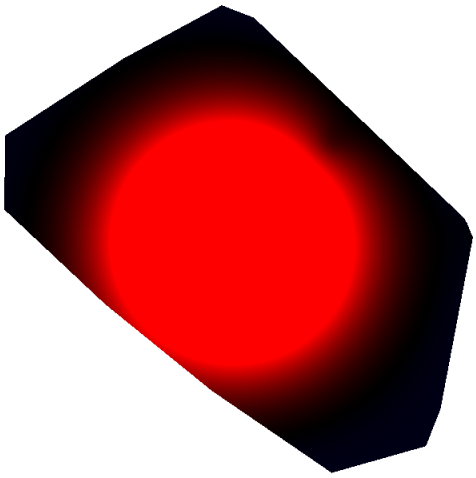

Figura 5.42 Imagem da diferença entre a função interpolada pelo método de Shepard e a função F_5.

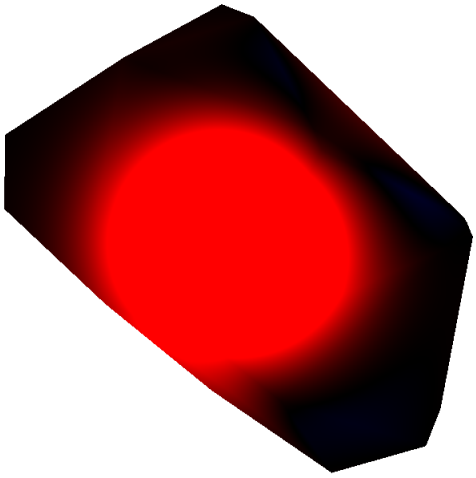

Figura 5.43 Imagem da diferença entre a função interpolada pelo método Linear por Partes e a função F_5.

As Figuras 5.44, 5.45, 5.46, representam a intersecção entre a função indicada e um plano ortogonal a z.

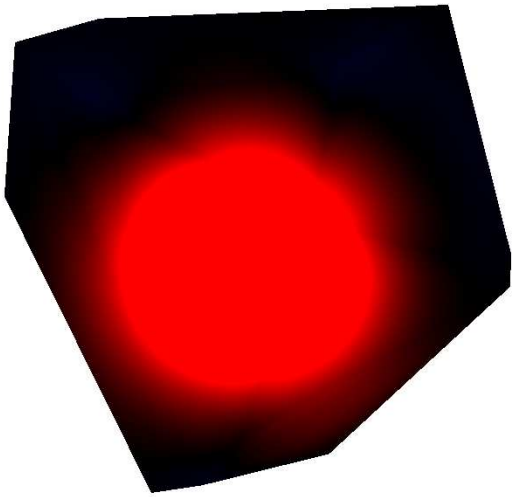

Figura 5.44 Imagem da diferença entre a função interpolada pelo método de Bézier e a função F_5 .

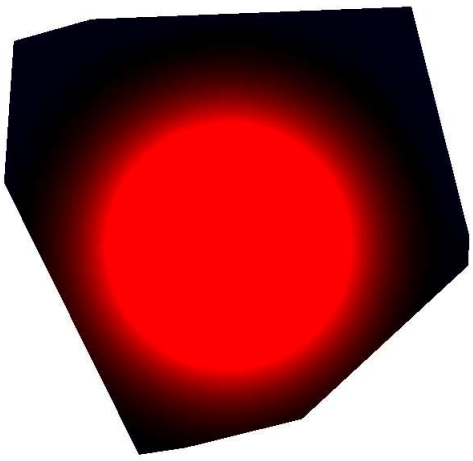

Figura 5.45 Imagem da diferença entre a função interpolada pelo método de Shepard e a função F_5.

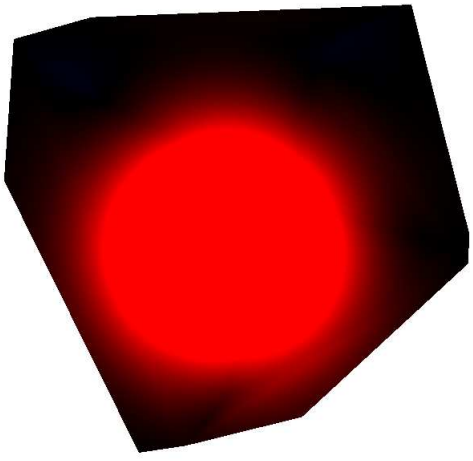

Figura 5.46 Imagem da diferença entre a função interpolada pelo método Linear por Partes e a função F_5.

Considere a função F_6. O erro estimado pelo RMS para o método de Bézier foi de 0,2833, para o método de Shepard foi de 0,1323 e para o Linear por Partes, erro de 0,1001. Foram geradas primeiramente imagens formadas pela intersecção da função indicada com um plano ortogonal ao eixo x. Veja as figuras 5.47, 5.48, 5.49.

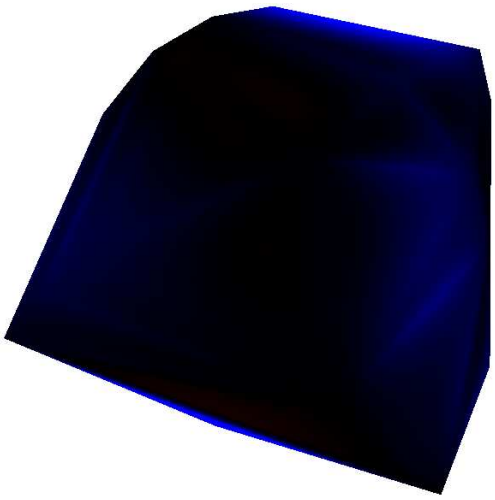

Figura 5.47 Imagem da diferença entre a função interpolada pelo método de Bézier e a função F_6 .

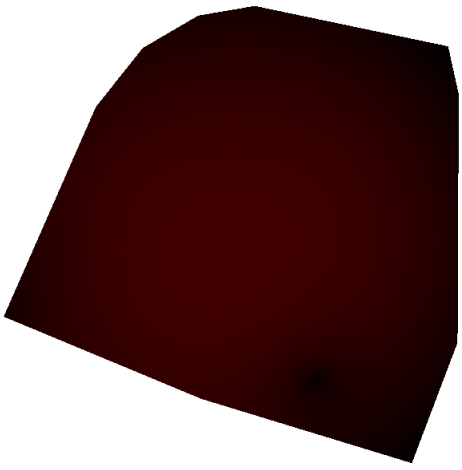

Figura 5.48 Imagem da diferença entre a função interpolada pelo método de Shepard e a função F_6.

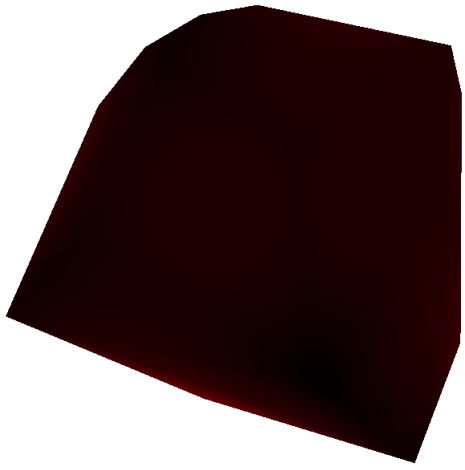

Figura 5.49 Imagem da diferença entre a função interpolada pelo método Linear por Partes e a função F_6.

As Figuras 5.50, 5.51, 5.52 representam a intersecção entre a função indicada e um plano ortogonal a y.

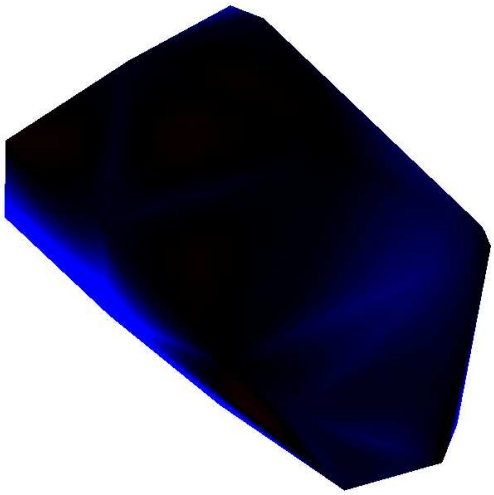

Figura 5.50 Imagem da diferença entre a função interpolada pelo método de Bézier e a função F_6 .

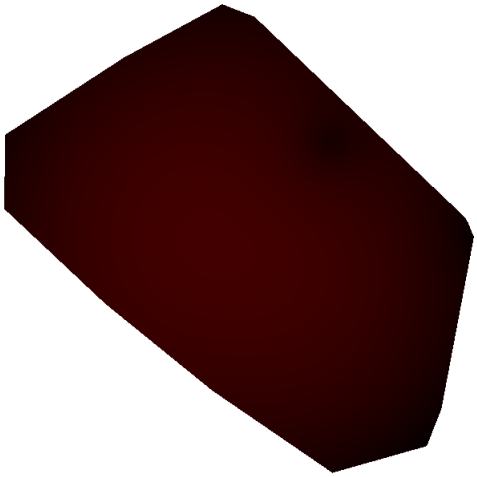

Figura 5.51 Imagem da diferença entre a função interpolada pelo método de Shepard e a função F_6.

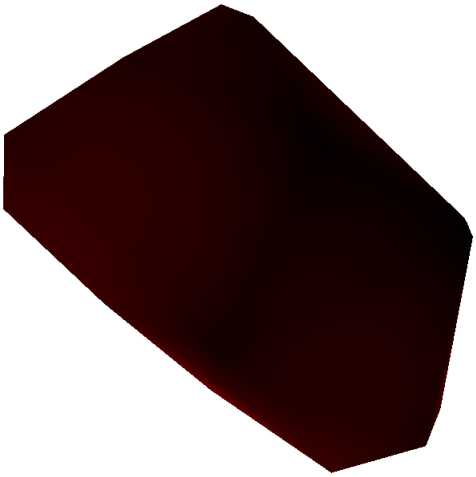

Figura 5.52 Imagem da diferença entre a função interpolada pelo método Linear por Partes e a função F_6.

As Figuras 5.53, 5.54, 5.55, representam a intersecção entre a função indicada e um plano ortogonal a z.

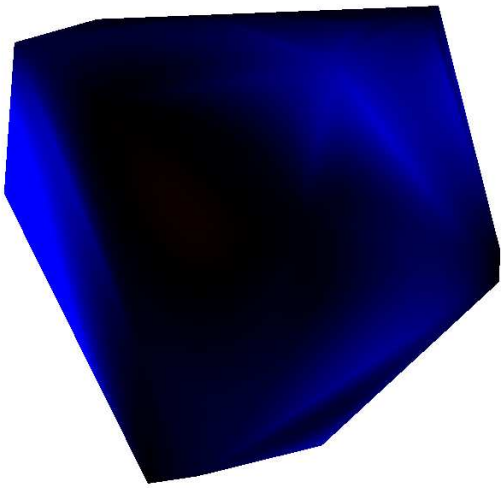

Figura 5.53 Imagem da diferença entre a função interpolada pelo método de Bézier e a função F_6 .

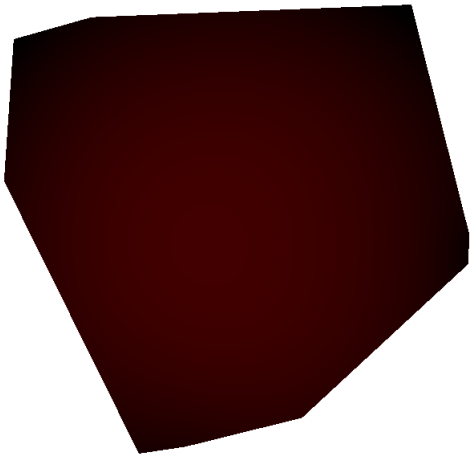

Figura 5.54 Imagem da diferença entre a função interpolada pelo método de Shepard e a função F_6.

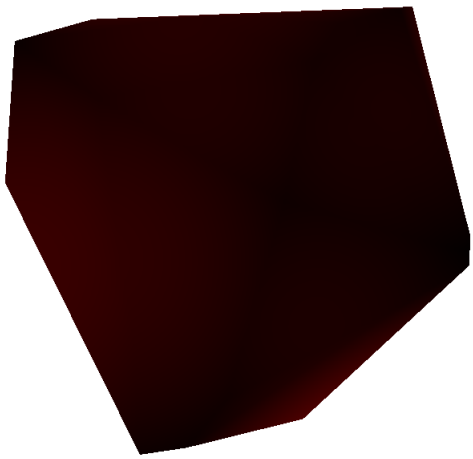

Figura 5.55 Imagem da diferença entre a função interpolada pelo método Linear por Partes e a função F_6.

CAPÍTULO 6

Conclusões

Neste trabalho desenvolvemos um modelo para geração de funções volumétricas não-negativas com continuidade C^1. Ele representa o primeiro passo nessa direção, onde dados trivariados espalhados e não-negativos são interpolados.

Técnicas já utilizadas, como a triangulação de Delaunay ([3]), a divisão de triângulos de um domínio triangulado em três mini-triângulos ([5]), com malhas cúbicas de Bézier estabelecidas sobre os mesmos, são comumente empregadas em trabalhos envolvendo interpolação C^1 de dados bivariados espalhados. Essas técnicas foram estendidas para o nosso caso, um domínio formado por pontos volumétricos dispersos, que foram tetrangulados e malhas quárticas tetraédricas de Bézier foram estabelecidas.

Juntamente com essas técnicas uma nova metodologia foi utilizada, **a redução do grau da malha quártica para uma cúbica na face externa**. Durante o processo de determinação de valores de controle, mantendo a continuidade C^1, devido ao excesso de graus de liberdade, essa técnica foi suficiente para resolver o nosso problema, respeitando a localidade, e a malha quártica tetraédrica não-negativa pôde ser desenvolvida.

O aspecto da localidade é ainda mais interessante, quando percebemos que as implementações dos algoritmos podem ser feitas de forma independente nas diversas malhas (Capítulo 4). O processo se torna paralelizável, podendo ser implementado com rapidez e eficiência.

Destacamos ainda nesse Capítulo uma discussão geral sobre modelagem, análise de experimentos realizados, limitações na forma de comparar métodos e os trabalhos futuros.

6.1 Modelagem de Dados e Análise de Experimentos

A modelagem com dados espalhados não-negativos assume apenas informações gerais sobre o fenômeno que gerou os dados, como grau de suavidade e não-negatividade. Isto nos previne de tecer julgamentos definitivos de ordem qualitativa com respeito à reprodutibilidade do fenômeno por qualquer método de interpolação. No entanto, pode-se estabelecer o que costumeiramente se aceita como "razoável" e que é compatível com alguns preceitos de teoria de informação. Numa grade com amostragem uniforme pode-se avançar mais um pouco ao se estudar o espectro de frequência da função e estabelecer limites de banda para o fenômeno original. O teorema de Nyquist ([29]) estabelece que uma dada amostragem com uma frequência f poderá representar uma função de banda limitada a uma frequência $\frac{f}{2}$; qualquer função com banda superior a este valor ou de banda ilimitada terá perdas de informação com a amostragem utilizada; e pior ainda, poderá ocorrer o fenômeno de "Aliasing", que pode destruir informações de baixa frequência. Assim, pode-se assumir que a discretização uni-

forme do fenômeno foi feita levando-se isso em conta, permitindo-nos concluir que a função interpoladora está livre dos erros de discretização ("Aliasing").

Neste contexto, em que não se garante recuperar a função ideal que representa o fenômeno, assume-se que a tendência da interpolação é seguir o que se pensa como "razoável". A interpolação linear por partes é uma boa referência do que se considera como "razoável", uma vez que é o método que menos assume condições sobre o fenômeno no interior dos tetraedros, não sendo uma função C^1, possuindo curvatura nula, e sendo governada pela expressão algébrica mais simples possível. O método de Shepard também se aproxima do "razoável" por outro aspecto: pontos próximos de um nó dado tendem a possuir valores de função próximos do valor do nó. Mas seu defeito, como já foi dito, é o problema dos pontos críticos nos nós, e da sua não-localidade. A utilização de funções para gerar valores nos nós, e então interpolá-los por algum método, para depois comparar o valor da função em pontos do interior com o valor fornecido pelo método de interpolação, serve para se ter uma idéia da ordem de grandeza do erro que usualmente se comete, bem como da sensibilidade da medida RMS com respeito a várias condições, como número de pontos, tetraedros, células, etc.

Em experimentos realizados com 100 nós, com mais de 500 tetraedros, e também com 200 nós, com mais de 1100 tetraedros, todos aleatoriamente determinados, os resultados visuais foram similares aos do experimento mostrado no capítulo anterior, mas os RMS variaram de 0,001 a valores acima de 3. Além disso, funções em que Bézier apresentava melhor resultado (veja Tabela 5.1), passaram a apresentar resultados opostos. Em parte, esta aparente instabilidade numérica é o preço que se paga por se exigir condições de suavidade acima de C^0, e que aceita mais de uma solução, podendo uma dada solução ficar muito distante de outra, que pode ser justamente a do fenômeno estudado. Além disso, ainda se paga um preço pela não negatividade, já que ajustes nas derivadas, que foram originalmente estimadas pela configuração geométrica dos nós, fazem com que as escolhas dos valores de controle se distanciem do "razoável"; esta característica justifica em parte a diferença dos valores RMS obtidos por Nielson (da ordem 0,02) dos obtidos aqui (em geral em torno de 0,05).

6.2 Trabalhos Futuros

Durante o desenvolvimento desse estudo nos deparamos com muitos obstáculos, devido a ausência de publicações que envolvam a interpolação de dados trivariados não-negativos e C^1. A quantidade de variáveis envolvidas na malha quártica tetraédrica tornou exaustiva a manipulação e análise dos dados, por conseguinte decisões independentes tiveram que ser tomadas a fim de atingir o nosso objetivo.

Dessa forma, esse estudo é apenas o início de uma série de possíveis trabalhos que podem ser tratados daqui por diante, como os citados abaixo.

1. Outras escolhas para b_{20}: estabelecemos um determinado limite inferior para essa ordenada interna, mas outras possibilidades podem ser estudadas, mais experimentos podem ser realizados e analisados.

2. Comparações diretas podem ser feitas com o método de Shepard Modificado e com outros métodos: diversas implementações podem ser feitas, com outras medidas de erro

sendo estabelecidas e análises numéricas realizadas.

3. A geração de uma malha de grau superior a quatro, que seja C^2 e não-negativa: um nível de continuidade mais elevado como o C^2 gera superfícies de melhor qualidade. Ele pode ser estudado sobre uma malha de grau cinco, por exemplo, mantendo ainda assim a não-negatividade.

4. Estudo de complexidade numérica e algorítmica.

Esses são alguns temas que surgem naturalmente após a conclusão desse trabalho.

Referências Bibliográficas

[1] Kong, V.P, Ong B.H., Saw, K.H., *Range Restricted Interpolation Using Cubic Bézier Triangles,* WSCG', Feb. 2-6, Plzen- Czech Republic (2004).

[2] Goodman, T.N.T., Ong, B.H. and Unsworth, K., *Constrained interpolation using rational cubic splines.* in Nurbs for Curve and Surface Design (Farin G., ed.), SIAM, Philadelphia, 59-74 (1991).

[3] Fang, T. P. and Piegl, L.A., *Algorithm for Delaunay triangulation and convex-hull computation using a sparse matrix.* Computer Aided Design 24, 425-436 (1992).

[4] Goodman, T.N.T., Said, H.B. and Chang, L.H.T., *Local Derivative estimation for scattered data interpolation.* 7 , 289-304 (1990).

[5] Clough, R., Tocher, J., *Finite Element Stiffness Matrices for Analysis of Plates in Blending.* in Proceedings of Conference on Matrix Methods in Structural Analysis (1965).

[6] Goodman, T.N.T. and Said, H.B., *A C^1 triangular interpolant suitable for scattered data interpolation.* Communications in Applied Numerical Methods 7, 479-485 (1991).

[7] Farin, G., *Curves and Surfaces for Computer Aided Geometric Design,* 4^{th} ed., Academic Press, (1996).

[8] Nielson, G.M., *Scattered Data Modeling.* IEEE CG & A 13, 60 - 70 (1993).

[9] Piah, A.R.M, Saaban, A., Majid, A.A., *Range Restricted Positivity-Preserving G^1 Scattered Data Interpolation.* Journal of Fundamental Sciences 2, 63 - 75 (2006).

[10] Chan, E.S. and Ong, B.H., *Range Restricted Scattered Data Interpolation Using Convex Combination of Bézier Triangles.* Journal of Computational and Applied Mathematics 136, 135-147 (2001).

[11] Saaban, A., Piah, A.R.M, Majid, A.A., *Range Restricted C^2 Interpolant to Scattered Data.* Computer Graphics, Imaging and Visualisation, 183 - 188 (2007).

[12] Yang, C.S, Kao S.P, Lee, F.B., *Twelve Different Interpolation Methods: A Case Study of Surfer 8.0.* XX ISPRS Congress, Istanbul, Turkey (2004).

[13] Amidror, I., *Scattered Data Interpolation Methods for Electronic Imaging Systems: a Survey.* Journal of Electronic Imaging 11 (2), 157 – 176 (2002).

[14] Fan, J., Gijbels, I., *Local polynomial modelling and its applications*, Chapman & Hall/CRC (1996).

[15] http://en.wikipedia.org/wiki/Radial_basis_function - Acessado em 22 de julho de 2009.

[16] http://www.ssg-surfer.com/ - Acessado em 16 de Julho de 2009.

[17] http://pt.wikipedia.org/wiki/Kriging - Acessado em 19 de Julho de 2009.

[18] http://www.geog.ubc.ca/courses/klink/gis.notes/ncgia/u40.html - Acessado em 22 de julho de 2009.

[19] Moura, R. C., Ramos, A. D. ; Sousa, C. S., Nascimento, T. A. S., Coelho, L. C. B. B., Valença, M. M., Melo, S. B., *Diagnosing osteoporosis: A new perspective on estimating bone density.* Physica. A, v. 381, 273-284 (2007).

[20] Melo, S. B., Dantas, C. C., Lima, E. A. O., Araujo Filho, M. C., *Distance Splines Approach to Irregularly Distributed Physical Data from the Brazilian Northeastern Coast.* Annals of the VI AMCTM, Lisboa - Portugal (2005).

[21] Castro, P.M.M. *Reconstruindo a Função de Densidade Mineral Óssea Utilizando-se Splines Cúbicos de Bézier Triangular em Dados Desestruturados para Aprimorar o Processo de Diagnóstico da Osteoporose*, Dissertação de Mestrado, Centro de Informática. Recife (2007).

[22] Novo, E., Leite, F., *O Sistema de Informações Geográficas do Reservatório da UHE Barra Bonita*, Anais VIII Simpósio Brasileiro de Sensoriamento Remoto, Salvador, Brasil, 14-19, INPE, 227-232 (1996).

[23] Moura, M.S, *Elaboração de uma ferramenta computacional para modelagem de próteses e ossos através da poroelasticidade acoplada*, Dissertação de Mestrado, Engenharia Mecânica. Campinas (2007).

[24] Haddade, I.R. *Avaliação econômica sob condições de risco em sistema produtivo de gado de leite na região Norte do estado do Rio de Janeiro*, Arq. Bras. Med. Vet. Zootec., v.57, n.3, 361-366 (2005).

[25] Barbosa, M., Fontes Junior, E. F., Vera-Tudela, C. A., Telles, J. C. *O Método de Elementos de Contorno e a Visualização Científica aliados à resolução de problemas da Mecânica Computacional*. XXXI Congresso Nacional de Matemática Aplicada e Computacional. São Paulo : SBMAC (2008).

[26] http://www.qhull.org/ - Acessado em 31 de julho de 2009.

[27] Maple 8.0.

[28] Sibson, R., *A Brief Description of the Natural Neighbour Interpolant.* In V. Barnett, editor, Interpolating Multivariate Data, John Wiley & Sons (1981).

[29] Gonzalez, Rafael C / Woods, Richard E., *Processamento de Imagens Digitais.* Editora Blucher, São Paulo (2000).

APÊNDICE A

Neste Apêndice segue a listagem dos arquivos principais da implementação do programa Bézier, utilizado no Capítulo 5. Este é um modelo de interpolação trivariado de tetraedros de Bézier C^1 não negativos, na linguagem Pascal orientado a objetos e no ambiente Borland Delphi 5.

```
unit uTetEstrutura;

interface

uses Math;

procedure inicializa;
procedure reinicializa_para_RMS(func:integer);
procedure Todos_interpolantes(x,y,z:Extended;var splines,shepard,linear:
Extended);
function Splines(x,y,z:Extended):Extended;
function Shepard(x,y,z:Extended):Extended;
function Linear(x,y,z:Extended):Extended;
function F1(x,y,z:Extended):Extended;
function F2(x,y,z:Extended):Extended;
function F3(x,y,z:Extended):Extended;
function F4(x,y,z:Extended):Extended;
function F5(x,y,z:Extended):Extended;
function F6(x,y,z:Extended):Extended;
function DesenhaFi(x,y,z:Extended; tipo:integer):Extended;

Type

Tripla=record
   x,y,z:Extended;    (*coordenadas de um ponto*)
   nx,ny,nz:Extended; (*coordenadas do seu vetor normal*)
   alfa:Extended; (*valor do alfa que   o menor dos alfas das arestas
   incidentes*)
   vol:Extended;(*conter  soma dos volumes de todos os tetraedros que o
   compartilham*)
```

```
   fp:Extended; (*valor da fun  o*)
end;

subscrito=record
   i,j,k,p,cpascal:integer;
end;

Tetraedro = Class
  Public
  a:array[0..34]of Extended; (* ordenadas dos pontos de controle-
  minitet a *)
  b:array[0..34]of Extended; (* ordenadas dos pontos de controle-
  minitet b *)
  c:array[0..34]of Extended; (* ordenadas dos pontos de controle-
  minitet c *)
  d:array[0..34]of Extended; (* ordenadas dos pontos de controle-
  minitet d *)
  V1,V2,V3,V4:integer;     (* ndices dos v rtices na Lista de pontos*)
  gx,gy,gz:Extended; (*coordenadas do gradiente do hiperplano contendo
   os v rtices*)
  n1x,n1y,n1z,n2x,n2y,n2z,n3x,n3y,n3z,n4x,n4y,n4z:Extended;(*normais
   s faces *)
  volume:Extended; (*n1 normal a V1V2V3; n2 normal a V1V2V4; n3 normal
  a V1V3V4 *)
  l:Extended; (* vari vel l, que   o menor dos valores de controle dos
  vertices*)
  visitado:Boolean; (*Indica se o tetraedro j  foi visitado, ou
  desenhado*)
  procedure preenche_normais;
  function interior(x,y,z:Extended):Boolean; (*se ponto est  no
  interior *)
  procedure compute_gradiente;
  procedure coordenadas_baricentricas(px,py,pz:Extended;var alfa,beta,
  gama,delta:Extended);
  procedure gera_G;
  procedure mini_centroides;
  procedure preenche_M1;
  procedure preenche_M2inicial;
  procedure preenche_M2;
  procedure preenche_M3;
  procedure preenche_M4;
  procedure preenche_M5;
  procedure preenche_M6;
```

```
  procedure preenche_M7;
  procedure preenche_M8;
  procedure preenche_M9;
  constructor create;
end;

Const Tolerancia = 0.000001;

var subsc:array[0..34] of subscrito;
    cfat:array[0..4] of integer;
    Lpts:array of Tripla;
    Ltets:array of Tetraedro;
    nomearq,mensagem,arqerro:string;
    Npts,Ntets:integer;
    MinX,MaxX,MinY,MaxY,MinZ,MaxZ:Extended;
    deltaX,deltaY,deltaZ:Extended;
    planox,planoy,planoz:Extended;
    MaxF:Extended;
    F_jah_inicializado:Boolean;


implementation

(*-----------------------Rotinas b sicas ---------------------------*)

constructor Tetraedro.create;
begin
  inherited create;
end;

function f(i,j,k,p:integer):integer;(*fun  o de convers o de  ndices
(ijkp)->f*)
begin
   f:=0;
   case i of
      0: f:=5*k - k*(k-1) div 2 + j;
      1: f:=15 + 4*k - k*(k-1) div 2 + j;
      2: f:=25 + 3*k - k*(k-1) div 2 + j;
      3: f:=31 + 2*k + j;
      4: f:=34;
   end;
end;
```

```
procedure preenche_subsc;
var i,j,k,p,l:integer;
begin
   for i:=0 to 4 do
    for j:=0 to 4 do
     for k:=0 to 4 do
      for p:=0 to 4 do
       if i+j+k+p=4 then begin
        l:=f(i,j,k,p);
        subsc[l].i:=i;
        subsc[l].j:=j;
        subsc[l].k:=k;
        subsc[l].p:=p;
        subsc[l].cpascal:=24 div( cfat[i]*cfat[j]*cfat[k]*cfat[p]);
        end;
end;


function det(x1,y1,z1,x2,y2,z2,x3,y3,z3:Extended):Extended;
begin
   det:=x1*y2*z3-x1*y3*z2+y1*x3*z2-y1*x2*z3+z1*x2*y3-z1*x3*y2;
end;

procedure produto_vetorial(x1,y1,z1,x2,y2,z2:Extended;
var x,y,z:Extended);
begin
   x:=y1*z2 - y2*z1;
   y:=x2*z1 - x1*z2;
   z:=x1*y2 - x2*y1;
end;


procedure Tetraedro.preenche_normais;
begin
   produto_vetorial(Lpts[V2].x-Lpts[V1].x,Lpts[V2].y-Lpts[V1].y,
                    Lpts[V2].z-Lpts[V1].z,
                    Lpts[V3].x-Lpts[V1].x,Lpts[V3].y-Lpts[V1].y,
                    Lpts[V3].z-Lpts[V1].z,
                    n1x,n1y,n1z);
   produto_vetorial(Lpts[V4].x-Lpts[V1].x,Lpts[V4].y-Lpts[V1].y,
                    Lpts[V4].z-Lpts[V1].z,
                    Lpts[V2].x-Lpts[V1].x,Lpts[V2].y-Lpts[V1].y,
```

```
                    Lpts[V2].z-Lpts[V1].z,
                    n2x,n2y,n2z);
   produto_vetorial(Lpts[V3].x-Lpts[V1].x,Lpts[V3].y-Lpts[V1].y,
                    Lpts[V3].z-Lpts[V1].z,
                    Lpts[V4].x-Lpts[V1].x,Lpts[V4].y-Lpts[V1].y,
                    Lpts[V4].z-Lpts[V1].z,
                    n3x,n3y,n3z);
   produto_vetorial(Lpts[V4].x-Lpts[V2].x,Lpts[V4].y-Lpts[V2].y,
                    Lpts[V4].z-Lpts[V2].z,
                    Lpts[V3].x-Lpts[V2].x,Lpts[V3].y-Lpts[V2].y,
                    Lpts[V3].z-Lpts[V2].z,
                    n4x,n4y,n4z);
end;

function Tetraedro.interior(x,y,z:Extended):Boolean;
var vx,vy,vz:Extended;
begin
   interior:=False;
   vx:=x-Lpts[V1].x; vy:=y-Lpts[V1].y; vz:=z-Lpts[V1].z;
   if vx*n1x + vy*n1y + vz*n1z>=0 then
      if vx*n2x + vy*n2y + vz*n2z>=0 then
         if vx*n3x + vy*n3y + vz*n3z>=0 then begin
            vx:=x-Lpts[V2].x; vy:=y-Lpts[V2].y; vz:=z-Lpts[V2].z;
            if vx*n4x + vy*n4y + vz*n4z>=0 then
               interior:=True;
         end;
end;


procedure Tetraedro.compute_gradiente;
begin
   gx:=det(Lpts[V2].fp-Lpts[V1].fp,Lpts[V3].fp-Lpts[V1].fp,
   Lpts[V4].fp-Lpts[V1].fp,
           Lpts[V2].y-Lpts[V1].y,Lpts[V3].y-Lpts[V1].y,
           Lpts[V4].y-Lpts[V1].y,
           Lpts[V2].z-Lpts[V1].z,Lpts[V3].z-Lpts[V1].z,
           Lpts[V4].z-Lpts[V1].z);
   gy:=det(Lpts[V2].x-Lpts[V1].x,Lpts[V3].x-Lpts[V1].x,
   Lpts[V4].x-Lpts[V1].x,
           Lpts[V2].fp-Lpts[V1].fp,Lpts[V3].fp-Lpts[V1].fp,
           Lpts[V4].fp-Lpts[V1].fp,
           Lpts[V2].z-Lpts[V1].z,Lpts[V3].z-Lpts[V1].z,
           Lpts[V4].z-Lpts[V1].z);
```

```
   gz:=det(Lpts[V2].x-Lpts[V1].x,Lpts[V3].x-Lpts[V1].x,
   Lpts[V4].x-Lpts[V1].x,
           Lpts[V2].y-Lpts[V1].y,Lpts[V3].y-Lpts[V1].y,
           Lpts[V4].y-Lpts[V1].y,
           Lpts[V2].fp-Lpts[V1].fp,Lpts[V3].fp-Lpts[V1].fp,
           Lpts[V4].fp-Lpts[V1].fp);
   volume:=det(Lpts[V2].x-Lpts[V1].x,Lpts[V3].x-Lpts[V1].x,
   Lpts[V4].x-Lpts[V1].x,
               Lpts[V2].y-Lpts[V1].y,Lpts[V3].y-Lpts[V1].y,
               Lpts[V4].y-Lpts[V1].y,
               Lpts[V2].z-Lpts[V1].z,Lpts[V3].z-Lpts[V1].z,
               Lpts[V4].z-Lpts[V1].z);
   gx:=gx/volume;
   gy:=gy/volume;
   gz:=gz/volume;
   volume:=abs(volume/6);
end;


procedure Tetraedro.coordenadas_baricentricas(px,py,pz:Extended;
var alfa,beta,gama,delta:Extended);
var vol:Extended;
begin
   vol:=6*volume;
   alfa:=det(px-Lpts[V4].x,py-Lpts[V4].y,pz-Lpts[V4].z,
             Lpts[V3].x-Lpts[V4].x,Lpts[V3].y-Lpts[V4].y,
             Lpts[V3].z-Lpts[V4].z,
             Lpts[V2].x-Lpts[V4].x,Lpts[V2].y-Lpts[V4].y,
             Lpts[V2].z-Lpts[V4].z)
             /vol;
   beta:=det(px-Lpts[V4].x,py-Lpts[V4].y,pz-Lpts[V4].z,
             Lpts[V1].x-Lpts[V4].x,Lpts[V1].y-Lpts[V4].y,
             Lpts[V1].z-Lpts[V4].z,
             Lpts[V3].x-Lpts[V4].x,Lpts[V3].y-Lpts[V4].y,
             Lpts[V3].z-Lpts[V4].z)
             /vol;
   gama:=det(px-Lpts[V4].x,py-Lpts[V4].y,pz-Lpts[V4].z,
             Lpts[V2].x-Lpts[V4].x,Lpts[V2].y-Lpts[V4].y,
             Lpts[V2].z-Lpts[V4].z,
             Lpts[V1].x-Lpts[V4].x,Lpts[V1].y-Lpts[V4].y,
             Lpts[V1].z-Lpts[V4].z)
             /vol;
   delta:=1-alfa-beta-gama;
```

```
end;

function pot(u:Extended;i:integer):Extended;
var func:Extended;
begin
   case i of
      0:pot:=1;
      1:pot:=u;
      2:pot:=u*u;
      3:pot:=u*u*u;
      4:pot:=u*u*u*u;
   end;
end;



(*--Rotinas de preenchimento das malhas tetrangulares-- *)

procedure Tetraedro.preenche_M1;
begin
   a[0]:=Lpts[V1].fp;
   c[4]:=Lpts[V1].fp;
   d[4]:=Lpts[V1].fp;
   b[4]:=Lpts[V2].fp;
   c[0]:=Lpts[V2].fp;
   d[0]:=Lpts[V2].fp;
   a[14]:=LPts[V3].fp;
   b[14]:=Lpts[V3].fp;
   d[14]:=Lpts[V3].fp;
   b[0]:=Lpts[V4].fp;
   c[14]:=Lpts[V4].fp;
   a[4]:=Lpts[V4].fp;
end;

procedure Tetraedro.gera_G;
var G:Extended;
begin
   G:=(Lpts[V1].fp+Lpts[V2].fp+Lpts[V3].fp+Lpts[V4].fp)/4;
   a[34]:=G;
   b[34]:=G;
   c[34]:=G;
   d[34]:=G;
end;
```

```
procedure Tetraedro.preenche_M2inicial;
var Ddij,alfa1:Extended;
begin
   alfa1:=Lpts[V1].alfa;
   Ddij:=(Lpts[V2].x-Lpts[V1].x)*Lpts[V1].nx+
   (Lpts[V2].y-Lpts[V1].y)*Lpts[V1].ny
        +(Lpts[V2].z-Lpts[V1].z)*Lpts[V1].nz;
   if Lpts[V1].fp+Ddij*alfa1/4<l/4 then
      alfa1:=(l-4*Lpts[V1].fp)/Ddij;
   Ddij:=(Lpts[V3].x-Lpts[V1].x)*Lpts[V1].nx+
   (Lpts[V3].y-Lpts[V1].y)*Lpts[V1].ny
        +(Lpts[V3].z-Lpts[V1].z)*Lpts[V1].nz;
   if Lpts[V1].fp+Ddij*alfa1/4<l/4 then
      alfa1:=(l-4*Lpts[V1].fp)/Ddij;
   Ddij:=(Lpts[V4].x-Lpts[V1].x)*Lpts[V1].nx+
   (Lpts[V4].y-Lpts[V1].y)*Lpts[V1].ny
        +(Lpts[V4].z-Lpts[V1].z)*Lpts[V1].nz;
   if Lpts[V1].fp+Ddij*alfa1/4<l/4 then
      alfa1:=(l-4*Lpts[V1].fp)/Ddij;
   Lpts[V1].alfa:=alfa1;
   alfa1:=Lpts[V2].alfa;
   Ddij:=(Lpts[V1].x-Lpts[V2].x)*Lpts[V2].nx+
   (Lpts[V1].y-Lpts[V2].y)*Lpts[V2].ny
        +(Lpts[V1].z-Lpts[V2].z)*Lpts[V2].nz;
   if Lpts[V2].fp+Ddij*alfa1/4<l/4 then
      alfa1:=(l-4*Lpts[V2].fp)/Ddij;
   Ddij:=(Lpts[V3].x-Lpts[V2].x)*Lpts[V2].nx+
   (Lpts[V3].y-Lpts[V2].y)*Lpts[V2].ny
        +(Lpts[V3].z-Lpts[V2].z)*Lpts[V2].nz;
   if Lpts[V2].fp+Ddij*alfa1/4<l/4 then
      alfa1:=(l-4*Lpts[V2].fp)/Ddij;
   Ddij:=(Lpts[V4].x-Lpts[V2].x)*Lpts[V2].nx+
   (Lpts[V4].y-Lpts[V2].y)*Lpts[V2].ny
        +(Lpts[V4].z-Lpts[V2].z)*Lpts[V2].nz;
   if Lpts[V2].fp+Ddij*alfa1/4<l/4 then
      alfa1:=(l-4*Lpts[V2].fp)/Ddij;
   Lpts[V2].alfa:=alfa1;
   alfa1:=Lpts[V3].alfa;
   Ddij:=(Lpts[V1].x-Lpts[V3].x)*Lpts[V3].nx+
   (Lpts[V1].y-Lpts[V3].y)*Lpts[V3].ny
        +(Lpts[V1].z-Lpts[V3].z)*Lpts[V3].nz;
   if Lpts[V3].fp+Ddij*alfa1/4<l/4 then
```

```
      alfa1:=(l-4*Lpts[V3].fp)/Ddij;
   Ddij:=(Lpts[V2].x-Lpts[V3].x)*Lpts[V3].nx+
   (Lpts[V2].y-Lpts[V3].y)*Lpts[V3].ny
        +(Lpts[V2].z-Lpts[V3].z)*Lpts[V3].nz;
   if Lpts[V3].fp+Ddij*alfa1/4<l/4 then
      alfa1:=(l-4*Lpts[V3].fp)/Ddij;
   Ddij:=(Lpts[V4].x-Lpts[V3].x)*Lpts[V3].nx+
   (Lpts[V4].y-Lpts[V3].y)*Lpts[V3].ny
        +(Lpts[V4].z-Lpts[V3].z)*Lpts[V3].nz;
   if Lpts[V3].fp+Ddij*alfa1/4<l/4 then
      alfa1:=(l-4*Lpts[V3].fp)/Ddij;
   Lpts[V3].alfa:=alfa1;
   alfa1:=Lpts[V4].alfa;
   Ddij:=(Lpts[V1].x-Lpts[V4].x)*Lpts[V4].nx+
   (Lpts[V1].y-Lpts[V4].y)*Lpts[V4].ny
        +(Lpts[V1].z-Lpts[V4].z)*Lpts[V4].nz;
   if Lpts[V4].fp+Ddij*alfa1/4<l/4 then
      alfa1:=(l-4*Lpts[V4].fp)/Ddij;
   Ddij:=(Lpts[V2].x-Lpts[V4].x)*Lpts[V4].nx+
   (Lpts[V2].y-Lpts[V4].y)*Lpts[V4].ny
        +(Lpts[V2].z-Lpts[V4].z)*Lpts[V4].nz;
   if Lpts[V4].fp+Ddij*alfa1/4<l/4 then
      alfa1:=(l-4*Lpts[V4].fp)/Ddij;
   Ddij:=(Lpts[V3].x-Lpts[V4].x)*Lpts[V4].nx+
   (Lpts[V3].y-Lpts[V4].y)*Lpts[V4].ny
        +(Lpts[V3].z-Lpts[V4].z)*Lpts[V4].nz;
   if Lpts[V4].fp+Ddij*alfa1/4<l/4 then
      alfa1:=(l-4*Lpts[V4].fp)/Ddij;
   Lpts[V4].alfa:=alfa1;
end;

procedure Tetraedro.preenche_M2;
var Ddij,alfa1:Extended;
begin
   alfa1:=Lpts[V1].alfa;
   Ddij:=(Lpts[V2].x-Lpts[V1].x)*Lpts[V1].nx+
   (Lpts[V2].y-Lpts[V1].y)*Lpts[V1].ny
        +(Lpts[V2].z-Lpts[V1].z)*Lpts[V1].nz;
   c[3]:=Lpts[V1].fp+Ddij*alfa1/4;
   d[3]:=c[3];
   Ddij:=(Lpts[V3].x-Lpts[V1].x)*Lpts[V1].nx+
   (Lpts[V3].y-Lpts[V1].y)*Lpts[V1].ny
        +(Lpts[V3].z-Lpts[V1].z)*Lpts[V1].nz;
```

```
a[5]:=Lpts[V1].fp+Ddij*alfa1/4;
d[8]:=a[5];
Ddij:=(Lpts[V4].x-Lpts[V1].x)*Lpts[V1].nx+
(Lpts[V4].y-Lpts[V1].y)*Lpts[V1].ny
     +(Lpts[V4].z-Lpts[V1].z)*Lpts[V1].nz;
a[1]:=Lpts[V1].fp+Ddij*alfa1/4;
c[8]:=a[1];

alfa1:=Lpts[V2].alfa;
Ddij:=(Lpts[V1].x-Lpts[V2].x)*Lpts[V2].nx+
(Lpts[V1].y-Lpts[V2].y)*Lpts[V2].ny
     +(Lpts[V1].z-Lpts[V2].z)*Lpts[V2].nz;
c[1]:=Lpts[V2].fp+Ddij*alfa1/4;
d[1]:=c[1];
Ddij:=(Lpts[V3].x-Lpts[V2].x)*Lpts[V2].nx+
(Lpts[V3].y-Lpts[V2].y)*Lpts[V2].ny
     +(Lpts[V3].z-Lpts[V2].z)*Lpts[V2].nz;
b[8]:=Lpts[V2].fp+Ddij*alfa1/4;
d[5]:=b[8];
Ddij:=(Lpts[V4].x-Lpts[V2].x)*Lpts[V2].nx+
(Lpts[V4].y-Lpts[V2].y)*Lpts[V2].ny
     +(Lpts[V4].z-Lpts[V2].z)*Lpts[V2].nz;
b[3]:=Lpts[V2].fp+Ddij*alfa1/4;
c[5]:=b[3];

alfa1:=Lpts[V3].alfa;
Ddij:=(Lpts[V1].x-Lpts[V3].x)*Lpts[V3].nx+
(Lpts[V1].y-Lpts[V3].y)*Lpts[V3].ny
     +(Lpts[V1].z-Lpts[V3].z)*Lpts[V3].nz;
a[12]:=Lpts[V3].fp+Ddij*alfa1/4;
d[13]:=a[12];
Ddij:=(Lpts[V2].x-Lpts[V3].x)*Lpts[V3].nx+
(Lpts[V2].y-Lpts[V3].y)*Lpts[V3].ny
     +(Lpts[V2].z-Lpts[V3].z)*Lpts[V3].nz;
b[13]:=Lpts[V3].fp+Ddij*alfa1/4;
d[12]:=b[13];
Ddij:=(Lpts[V4].x-Lpts[V3].x)*Lpts[V3].nx+
(Lpts[V4].y-Lpts[V3].y)*Lpts[V3].ny
     +(Lpts[V4].z-Lpts[V3].z)*Lpts[V3].nz;
a[13]:=Lpts[V3].fp+Ddij*alfa1/4;
b[12]:=a[13];

alfa1:=Lpts[V4].alfa;
```

```
   Ddij:=(Lpts[V1].x-Lpts[V4].x)*Lpts[V4].nx+
   (Lpts[V1].y-Lpts[V4].y)*Lpts[V4].ny
        +(Lpts[V1].z-Lpts[V4].z)*Lpts[V4].nz;
   a[3]:=Lpts[V4].fp+Ddij*alfa1/4;
   c[13]:=a[3];
   Ddij:=(Lpts[V2].x-Lpts[V4].x)*Lpts[V4].nx+
   (Lpts[V2].y-Lpts[V4].y)*Lpts[V4].ny
        +(Lpts[V2].z-Lpts[V4].z)*Lpts[V4].nz;
   b[1]:=Lpts[V4].fp+Ddij*alfa1/4;
   c[12]:=b[1];
   Ddij:=(Lpts[V3].x-Lpts[V4].x)*Lpts[V4].nx+
   (Lpts[V3].y-Lpts[V4].y)*Lpts[V4].ny
        +(Lpts[V3].z-Lpts[V4].z)*Lpts[V4].nz;
   a[8]:=Lpts[V4].fp+Ddij*alfa1/4;
   b[5]:=a[8];
end;


procedure Tetraedro.preenche_M3; (*Equa  es 5.7 da tese*)
var p,q,r:Extended;
begin
   p:=4*a[5]+4*a[1]-2*a[0];
   q:=4*a[8]+4*a[3]-2*a[4];
   r:=4*a[13]+4*a[12]-2*a[14];
   a[6]:=(4*p+q+r)/36;
   a[7]:=(p+4*q+r)/36;
   a[10]:=(p+q+4*r)/36;
   a[2]:=(4/3)*(1/2)*(a[1]+a[3])-(1/3)*(1/2)*(a[0]+a[4]);
   a[9]:=(4/3)*(1/2)*(a[5]+a[12])-(1/3)*(1/2)*(a[0]+a[14]);
   a[11]:=(4/3)*(1/2)*(a[8]+a[13])-(1/3)*(1/2)*(a[4]+a[14]);

   p:=4*b[5]+4*b[1]-2*b[0];
   q:=4*b[8]+4*b[3]-2*b[4];
   r:=4*b[13]+4*b[12]-2*b[14];
   b[6]:=(4*p+q+r)/36;
   b[7]:=(p+4*q+r)/36;
   b[10]:=(p+q+4*r)/36;
   b[2]:=(4/3)*(1/2)*(b[1]+b[3])-(1/3)*(1/2)*(b[0]+b[4]);
   b[9]:=(4/3)*(1/2)*(b[5]+b[12])-(1/3)*(1/2)*(b[0]+b[14]);
   b[11]:=(4/3)*(1/2)*(b[8]+b[13])-(1/3)*(1/2)*(b[4]+b[14]);

   p:=4*c[5]+4*c[1]-2*c[0];
   q:=4*c[8]+4*c[3]-2*c[4];
```

```
   r:=4*c[13]+4*c[12]-2*c[14];
   c[6]:=(4*p+q+r)/36;
   c[7]:=(p+4*q+r)/36;
   c[10]:=(p+q+4*r)/36;
   c[2]:=(4/3)*(1/2)*(c[1]+c[3])-(1/3)*(1/2)*(c[0]+c[4]);
   c[9]:=(4/3)*(1/2)*(c[5]+c[12])-(1/3)*(1/2)*(c[0]+c[14]);
   c[11]:=(4/3)*(1/2)*(c[8]+c[13])-(1/3)*(1/2)*(c[4]+c[14]);

   p:=4*d[5]+4*d[1]-2*d[0];
   q:=4*d[8]+4*d[3]-2*d[4];
   r:=4*d[13]+4*d[12]-2*d[14];
   d[6]:=(4*p+q+r)/36;
   d[7]:=(p+4*q+r)/36;
   d[10]:=(p+q+4*r)/36;
   d[2]:=(4/3)*(1/2)*(d[1]+d[3])-(1/3)*(1/2)*(d[0]+d[4]);
   d[9]:=(4/3)*(1/2)*(d[5]+d[12])-(1/3)*(1/2)*(d[0]+d[14]);
   d[11]:=(4/3)*(1/2)*(d[8]+d[13])-(1/3)*(1/2)*(d[4]+d[14]);
end;

procedure Tetraedro.preenche_M4; (*Equa  es 5.8 da tese*)
begin
   b[15]:=(b[0]+b[1]+b[5]+a[3])/4;
   b[18]:=(b[3]+b[4]+b[8]+c[1])/4;
   b[24]:=(b[12]+b[13]+b[14]+a[12])/4;
   a[15]:=(a[0]+a[1]+a[5]+c[3])/4;
   a[18]:=b[15];
   a[24]:=b[24];
   c[15]:=b[18];
   c[18]:=a[15];
   c[24]:=b[15];
   d[15]:=b[18];
   d[18]:=a[15];
   d[24]:=b[24];
end;

procedure Tetraedro.preenche_M5; (*Equa  es 5.8 da tese*)
begin
   a[16]:=(a[1]+a[2]+a[6]+c[7])/4;
   a[17]:=(a[2]+a[3]+a[7]+c[10])/4;
   a[19]:=(a[5]+a[6]+a[9]+d[7])/4;
   a[21]:=(a[7]+a[8]+a[11]+b[6])/4;
   a[22]:=(a[9]+a[10]+a[12]+d[10])/4;
   a[23]:=(a[10]+a[11]+a[13]+b[10])/4;
```

```
   b[16]:=(b[1]+b[2]+b[6]+c[10])/4;
   b[17]:=(b[2]+b[3]+b[7]+c[6])/4;
   b[19]:=(b[5]+b[6]+b[9]+a[7])/4;
   b[21]:=(b[7]+b[8]+b[11]+d[6])/4;
   b[22]:=(b[9]+b[10]+b[12]+a[10])/4;
   b[23]:=(b[10]+b[11]+b[13]+d[10])/4;

   c[16]:=(c[1]+c[2]+c[6]+d[6])/4;
   c[17]:=(c[2]+c[3]+c[7]+d[7])/4;
   c[19]:=(c[5]+c[6]+c[9]+b[7])/4;
   c[21]:=(c[7]+c[8]+c[11]+a[6])/4;
   c[22]:=(c[9]+c[10]+c[12]+b[6])/4;
   c[23]:=(c[10]+c[11]+c[13]+a[7])/4;

   d[16]:=(d[1]+d[2]+d[6]+c[6])/4;
   d[17]:=(d[2]+d[3]+d[7]+c[7])/4;
   d[19]:=(d[5]+d[6]+d[9]+b[7])/4;
   d[21]:=(d[7]+d[8]+d[11]+a[6])/4;
   d[22]:=(d[9]+d[10]+d[12]+b[10])/4;
   d[23]:=(d[10]+d[11]+d[13]+a[10])/4;
end;


procedure Tetraedro.preenche_M6;
begin
   a[25]:=(a[15]+a[16]+a[19]+d[17])/4;
   a[27]:=(a[17]+a[18]+a[21]+c[22])/4;
   a[30]:=(a[22]+a[23]+a[24]+d[22])/4;

   b[25]:=(b[15]+b[16]+b[19]+a[17])/4;
   b[27]:=(b[17]+b[18]+b[21]+c[16])/4;
   b[30]:=(b[22]+b[23]+b[24]+a[22])/4;

   c[25]:=(c[15]+c[16]+c[19]+d[19])/4;
   c[27]:=(c[17]+c[18]+c[21]+d[21])/4;
   c[30]:=(c[22]+c[23]+c[24]+b[19])/4;

   d[25]:=(d[15]+d[16]+d[19]+b[17])/4;
   d[27]:=(d[17]+d[18]+d[21]+a[16])/4;
   d[30]:=(d[22]+d[23]+d[24]+b[22])/4;
end;
```

```
procedure Tetraedro.mini_centroides;
begin
   a[20]:=(5*a[0]+5*a[4]+5*a[14]+b[4])/16;
   b[20]:=(5*b[0]+5*b[4]+5*b[14]+a[0])/16;
   c[20]:=(5*c[0]+5*c[4]+5*c[14]+b[14])/16;
   d[20]:=(5*d[0]+5*d[4]+5*d[14]+b[0])/16;    (*
   a[20]:=(a[16]+a[17]+a[19]+a[22]+a[21]+a[23])/6;
   b[20]:=(b[16]+b[17]+b[19]+b[22]+b[21]+b[23])/6;
   c[20]:=(c[16]+c[17]+c[19]+c[22]+c[21]+c[23])/6;
   d[20]:=(d[16]+d[17]+d[19]+d[22]+d[21]+d[23])/6;  *)

end;

procedure Tetraedro.preenche_M7;
begin
   a[26]:=(a[16]+a[17]+a[20]+c[20])/4;
   a[28]:=(a[19]+a[20]+a[22]+d[20])/4;
   a[29]:=(a[20]+a[21]+a[23]+b[20])/4;

   b[26]:=(b[16]+b[17]+b[20]+c[20])/4;
   b[28]:=(b[19]+b[20]+b[22]+a[20])/4;
   b[29]:=(b[20]+b[21]+b[23]+d[20])/4;

   c[26]:=(c[16]+c[17]+c[20]+d[20])/4;
   c[28]:=(c[19]+c[20]+c[22]+b[20])/4;
   c[29]:=(c[20]+c[21]+c[23]+a[20])/4;

   d[26]:=(d[16]+d[17]+d[20]+c[20])/4;
   d[28]:=(d[19]+d[20]+d[22]+b[20])/4;
   d[29]:=(d[20]+d[21]+d[23]+a[20])/4;
end;


procedure Tetraedro.preenche_M8;
begin
   a[31]:=(a[25]+a[26]+a[28]+c[26])/4;
   a[32]:=(a[26]+a[27]+a[29]+b[26])/4;
   a[33]:=(a[28]+a[29]+a[30]+b[29])/4;

   b[31]:=(b[25]+b[26]+b[28]+a[26])/4;
   b[32]:=(b[26]+b[27]+b[29]+c[26])/4;
   b[33]:=(b[28]+b[29]+b[30]+d[29])/4;
```

```
   c[31]:=(c[25]+c[26]+c[28]+b[29])/4;
   c[32]:=(c[26]+c[27]+c[29]+a[28])/4;
   c[33]:=(c[28]+c[29]+c[30]+b[28])/4;

   d[31]:=(d[25]+d[26]+d[28]+b[26])/4;
   d[32]:=(d[26]+d[27]+d[29]+c[29])/4;
   d[33]:=(d[28]+d[29]+d[30]+b[28])/4;
end;

procedure Tetraedro.preenche_M9;
begin
   b[34]:=(b[31]+b[32]+b[33]+a[31])/4;
   a[34]:=b[34];
   c[34]:=a[34];
   d[34]:=c[34];
end;


procedure inicializa;
var Farq:textfile;
    erro,i,V1,V2,V3,V4:integer;
    x,y,z,fp:Extended;
begin

   MaxX:=-100000;
   MaxY:=-100000;
   MaxZ:=-100000;
   MinX:=100000;
   MinY:=100000;
   MinZ:=100000;
   MaxF:=0;
   F_jah_inicializado:=False;
   cfat[0]:=1; cfat[1]:=1; cfat[2]:=2; cfat[3]:=6; cfat[4]:=24;
   {$I-}
   AssignFile(Farq,nomearq);
   Reset(Farq);
   erro:=IOResult;
   if erro=0 then begin
      read(Farq,Npts,Ntets);
      SetLength(Lpts,Npts);
      SetLength(Ltets,Ntets);
      for i:=0 to Npts-1 do begin
```

```
   read(Farq,x,y,z,fp);
   if MaxX<x then MaxX:=x;
   if MaxY<y then MaxY:=y;
   if MaxZ<z then MaxZ:=z;
   if MinX>x then MinX:=x;
   if MinY>y then MinY:=y;
   if MinZ>z then MinZ:=z;
   if MaxF<fp then MaxF:=fp;
   Lpts[i].x:=x;
   Lpts[i].y:=y;
   Lpts[i].z:=z;
   Lpts[i].fp:=fp;
   Lpts[i].vol:=0;
   Lpts[i].nx:=0;
   Lpts[i].ny:=0;
   Lpts[i].nz:=0;
   Lpts[i].alfa:=1.0;
end;
for i:=0 to Ntets-1 do begin
   Ltets[i]:=Tetraedro.create;
   read(Farq,V1,V2,V3,V4);
   Ltets[i].V1:=V1;
   Ltets[i].V2:=V2;
   Ltets[i].V3:=V3;
   Ltets[i].V4:=V4;
   Ltets[i].l:=Lpts[Ltets[i].V1].fp;
   Ltets[i].preenche_normais;
   Ltets[i].visitado:=False;
   if Ltets[i].l>Lpts[Ltets[i].V2].fp then
      Ltets[i].l:=Lpts[Ltets[i].V2].fp;
   if Ltets[i].l>Lpts[Ltets[i].V3].fp then
      Ltets[i].l:=Lpts[Ltets[i].V3].fp;
   if Ltets[i].l>Lpts[Ltets[i].V4].fp then
      Ltets[i].l:=Lpts[Ltets[i].V4].fp;
   Ltets[i].compute_gradiente;
   Lpts[Ltets[i].V1].nx:=Lpts[Ltets[i].V1].nx+
   Ltets[i].gx/Ltets[i].volume;
   Lpts[Ltets[i].V1].ny:=Lpts[Ltets[i].V1].ny+
   Ltets[i].gy/Ltets[i].volume;
   Lpts[Ltets[i].V1].nz:=Lpts[Ltets[i].V1].nz+
   Ltets[i].gz/Ltets[i].volume;
   Lpts[Ltets[i].V2].nx:=Lpts[Ltets[i].V2].nx+
   Ltets[i].gx/Ltets[i].volume;
```

```
         Lpts[Ltets[i].V2].ny:=Lpts[Ltets[i].V2].ny+
         Ltets[i].gy/Ltets[i].volume;
         Lpts[Ltets[i].V2].nz:=Lpts[Ltets[i].V2].nz+
         Ltets[i].gz/Ltets[i].volume;
         Lpts[Ltets[i].V3].nx:=Lpts[Ltets[i].V3].nx+
         Ltets[i].gx/Ltets[i].volume;
         Lpts[Ltets[i].V3].ny:=Lpts[Ltets[i].V3].ny+
         Ltets[i].gy/Ltets[i].volume;
         Lpts[Ltets[i].V3].nz:=Lpts[Ltets[i].V3].nz+
         Ltets[i].gz/Ltets[i].volume;
         Lpts[Ltets[i].V4].nx:=Lpts[Ltets[i].V4].nx+
         Ltets[i].gx/Ltets[i].volume;
         Lpts[Ltets[i].V4].ny:=Lpts[Ltets[i].V4].ny+
         Ltets[i].gy/Ltets[i].volume;
         Lpts[Ltets[i].V4].nz:=Lpts[Ltets[i].V4].nz+
         Ltets[i].gz/Ltets[i].volume;
         Lpts[Ltets[i].V1].vol:=Lpts[Ltets[i].V1].vol +
         1/Ltets[i].volume;
         Lpts[Ltets[i].V2].vol:=Lpts[Ltets[i].V2].vol +
         1/Ltets[i].volume;
         Lpts[Ltets[i].V3].vol:=Lpts[Ltets[i].V3].vol +
         1/Ltets[i].volume;
         Lpts[Ltets[i].V4].vol:=Lpts[Ltets[i].V4].vol +
         1/Ltets[i].volume;
      end;
      for i:=0 to Npts-1 do begin
         Lpts[i].nx:=Lpts[i].nx/Lpts[i].vol;
         Lpts[i].ny:=Lpts[i].ny/Lpts[i].vol;
         Lpts[i].nz:=Lpts[i].nz/Lpts[i].vol;
      end;
      for i:=0 to Ntets-1 do begin
         Ltets[i].gera_G;
         Ltets[i].preenche_M1;
         Ltets[i].mini_centroides;
         Ltets[i].preenche_M2inicial;
      end;
      for i:=0 to Ntets-1 do begin
         Ltets[i].preenche_M2;
         Ltets[i].preenche_M3;
         Ltets[i].preenche_M4;
         Ltets[i].preenche_M5;
//         Ltets[i].mini_centroides;
         Ltets[i].preenche_M6;
```

```
         Ltets[i].preenche_M7;
         Ltets[i].preenche_M8;
         Ltets[i].preenche_M9;
      end;
      preenche_subsc;
      closefile(Farq);
   end
   else mensagem:='Problemas no arquivo';

   deltaX:=MaxX-MinX;
   deltaY:=MaxY-MinY;
   deltaZ:=MaxZ-MinZ;

{$I+}
end;

(*--------------M todos Interpoladores----------------------*)

(*conven  o da tese para as sub-malhas (todo v rtice fora da face
externa tem
   ndices 4000). Face externa de cada sub-malha:
     sub-malha a: V1=a0=a0004; V3=a14=a0040; V4=a4=a0400;
     sub-malha b: V2=b4=b0400; V3=b14=b0040; V4=b0=b0004;
     sub-malha c: V1=c4=c0400; V2=c0=c0004; V4=c14=c0040;
     sub-malha d: V1=d4=d0400; V2=d0=d0004; V3=d14=d0040;
     alfa est  associada a V1; beta a V2; gama a V3 e delta a V4*)



function Splines(x,y,z:Extended):Extended; (*calcula a fun  o em malha
subdividida*)
var l,i,j,k,p:integer;  (*retorna -1000 se o ponto cair fora do *)
    controle:integer;   (* inv lucro convexo*)
    alfa,beta,gama,delta,func:Extended;
    fora:boolean;
begin
   func:=0;
   fora:=True;
   for l:=0 to Ntets-1 do if Ltets[l].interior(x,y,z) then begin
      Ltets[l].coordenadas_baricentricas(x,y,z,alfa,beta,gama,delta);
      if (alfa<=beta)and(alfa<=delta)and(alfa<=gama) then begin
         beta:=beta-alfa; delta:=delta-alfa; gama:=gama-alfa;
         alfa:=4*alfa;
```

```
      for controle:=0 to 34 do begin
         i:=subsc[controle].i; j:=subsc[controle].j;
         k:=subsc[controle].k; p:=subsc[controle].p;
         func:=func+Ltets[l].b[controle]*subsc[controle].cpascal*
               pot(alfa,i)*pot(beta,j)*pot(gama,k)*pot(delta,p);
      end;
   end
   else if (beta<=alfa)and(beta<=delta)and(beta<=gama) then begin
      alfa:=alfa-beta; delta:=delta-beta; gama:=gama-beta;
      beta:=4*beta;
      for controle:=0 to 34 do begin
         i:=subsc[controle].i; j:=subsc[controle].j;
         k:=subsc[controle].k; p:=subsc[controle].p;
         func:=func+Ltets[l].a[controle]*subsc[controle].cpascal*
               pot(alfa,p)*pot(beta,i)*pot(gama,k)*pot(delta,j);
      end;
   end
   else if (gama<=alfa)and(gama<=delta)and(gama<=beta) then begin
      alfa:=alfa-gama; delta:=delta-gama; beta:=beta-gama;
      gama:=4*gama;
      for controle:=0 to 34 do begin
         i:=subsc[controle].i; j:=subsc[controle].j;
         k:=subsc[controle].k; p:=subsc[controle].p;
         func:=func+Ltets[l].c[controle]*subsc[controle].cpascal*
               pot(alfa,j)*pot(beta,p)*pot(gama,i)*pot(delta,k);
      end;
   end
   else begin
      alfa:=alfa-delta; beta:=beta-delta; gama:=gama-delta;
      delta:=4*delta;
      for controle:=0 to 34 do begin
         i:=subsc[controle].i; j:=subsc[controle].j;
         k:=subsc[controle].k; p:=subsc[controle].p;
         func:=func+Ltets[l].d[controle]*subsc[controle].cpascal*
               pot(alfa,j)*pot(beta,p)*pot(gama,k)*pot(delta,i);
      end;
   end;
   fora:=False;
   Ltets[l].visitado:=True;
   break;
end;
if fora then
   Splines:=-1000
```

```
   else
      Splines:=func;
end;


function Shepard(x,y,z:Extended):Extended; (*calcula a interpola  o
de Shepard*)
var i,j:integer;          (*retorna -1000 se o ponto cair fora do *)
                          (*inv lucro convexo*)
    alfa,beta,gama,delta,func,soma_dist,dist:Extended;
    fora,pronto:boolean;
    dx,dy,dz:Extended;
begin
   func:=0;
   soma_dist:=0;
   fora:=True;
   pronto:=False;
   for i:=0 to Ntets-1 do if Ltets[i].interior(x,y,z) then begin
      Ltets[i].coordenadas_baricentricas(x,y,z,alfa,beta,gama,delta);
      for j:=0 to Npts-1 do begin
         dx:=x-Lpts[j].x;   dy:=y-Lpts[j].y;    dz:=z-Lpts[j].z;
         dist:=sqrt(dx*dx + dy*dy + dz*dz);
         if  dist<Tolerancia then begin
            func:=Lpts[j].fp;
            pronto:=True;
            break;
         end
         else begin
            func:=func+Lpts[j].fp/dist;
            soma_dist:=soma_dist+1/dist;
         end;
      end;
      fora:=False;
      Ltets[i].visitado:=True;
      if not pronto then func:=func/soma_dist;
      break;
   end;
   if fora then
      Shepard:=-1000
   else
      Shepard:=func;
end;
```

```
function Linear(x,y,z:Extended):Extended; (*calcula a interpola  o
Linear *)
var i,j:integer;          (*retorna -1000 se o ponto cair fora do *)
                          (*inv lucro convexo*)
    alfa,beta,gama,delta,func:Extended;
    fora:boolean;
    dx,dy,dz:Extended;
begin
   fora:=True;
   for i:=0 to Ntets-1 do if Ltets[i].interior(x,y,z) then begin
      Ltets[i].coordenadas_baricentricas(x,y,z,alfa,beta,gama,delta);
      func:=alfa*Lpts[Ltets[i].V1].fp + beta*Lpts[Ltets[i].V2].fp+
            gama*Lpts[Ltets[i].V3].fp + delta*Lpts[Ltets[i].V4].fp;
      fora:=False;
      Ltets[i].visitado:=True;
      break;
   end;
   if fora then
      Linear:=-1000
   else
      Linear:=func;
end;

procedure Todos_interpolantes(x,y,z:Extended;var splines,shepard,
linear:Extended);
                                  (*M todos: Splines, Shepard e
                                  Linear p/ RMS*)
var l,i,j,k,p:integer;            (*retorna -1000 se o ponto cair
fora do *)
    controle:integer;             (* inv lucro convexo*)
    alfa,beta,gama,delta,func,shepfunc,soma_dist,dist,dx,dy,dz:
    Extended;
    fora,pronto:boolean;
begin
   func:=0;
   fora:=True;
   soma_dist:=0;
   shepard:=0;
   pronto:=False;
   for l:=0 to Ntets-1 do if Ltets[l].interior(x,y,z) then begin
      Ltets[l].coordenadas_baricentricas(x,y,z,alfa,beta,gama,
      delta);
      for j:=0 to Npts-1 do begin
```

```
   dx:=x-Lpts[j].x; dy:=y-Lpts[j].y;  dz:=z-Lpts[j].z;
   dist:=sqrt(dx*dx + dy*dy + dz*dz);
   if  dist<Tolerancia then begin
      shepard:=Lpts[j].fp;
      pronto:=True;
      break;
   end
   else begin
      shepard:=shepard+Lpts[j].fp/dist;
      soma_dist:=soma_dist+1/dist;
   end;
end;
if not pronto then shepard:=shepard/soma_dist;

linear:=alfa*Lpts[Ltets[l].V1].fp + beta*Lpts[Ltets[l].V2].fp+
        gama*Lpts[Ltets[l].V3].fp + delta*Lpts[Ltets[l].V4].fp;

if (alfa<=beta)and(alfa<=delta)and(alfa<=gama) then begin
   beta:=beta-alfa; delta:=delta-alfa; gama:=gama-alfa;
   alfa:=4*alfa;
   for controle:=0 to 34 do begin
      i:=subsc[controle].i; j:=subsc[controle].j;
      k:=subsc[controle].k; p:=subsc[controle].p;
      func:=func+Ltets[l].b[controle]*subsc[controle].cpascal*
            pot(alfa,i)*pot(beta,j)*pot(gama,k)*pot(delta,p);
   end;
end
else if (beta<=alfa)and(beta<=delta)and(beta<=gama) then begin
   alfa:=alfa-beta; delta:=delta-beta; gama:=gama-beta;
   beta:=4*beta;
   for controle:=0 to 34 do begin
      i:=subsc[controle].i; j:=subsc[controle].j;
      k:=subsc[controle].k; p:=subsc[controle].p;
      func:=func+Ltets[l].a[controle]*subsc[controle].cpascal*
            pot(alfa,p)*pot(beta,i)*pot(gama,k)*pot(delta,j);
   end;
end
else if (gama<=alfa)and(gama<=delta)and(gama<=beta) then begin
   alfa:=alfa-gama; delta:=delta-gama; beta:=beta-gama;
   gama:=4*gama;
   for controle:=0 to 34 do begin
      i:=subsc[controle].i; j:=subsc[controle].j;
      k:=subsc[controle].k; p:=subsc[controle].p;
```

```
            func:=func+Ltets[l].c[controle]*subsc[controle].cpascal*
                  pot(alfa,j)*pot(beta,p)*pot(gama,i)*pot(delta,k);
         end;
      end
      else begin
         alfa:=alfa-delta; beta:=beta-delta; gama:=gama-delta;
         delta:=4*delta;
         for controle:=0 to 34 do begin
            i:=subsc[controle].i; j:=subsc[controle].j;
            k:=subsc[controle].k; p:=subsc[controle].p;
            func:=func+Ltets[l].d[controle]*subsc[controle].cpascal*
                  pot(alfa,j)*pot(beta,p)*pot(gama,k)*pot(delta,i);
         end;
      end;
      fora:=False;
      Ltets[l].visitado:=True;
      break;
   end;
   if fora then
      splines:=-1000
   else
      splines:=func;
end;

(*------Fun  es sugeridas por Nielson et al. para valida  o------*)


function F1(x,y,z:Extended):Extended;(*acrescida de 0,1 para se tornar
n o negativa*)
begin
   F1:=0.75*exp(-0.25*((9*x-2)*(9*x-2)+(9*y-2)*(9*y-2)+
   (9*z-2)*(9*z-2)))+
       0.75*exp(-(9*x+1)*(9*x+1)/49-(9*y+1)/10-(9*z+1)/10)+
       0.50*exp(-0.25*((9*x-7)*(9*x-7)+(9*y-3)*(9*y-3)+
       (9*z-5)*(9*z-5)))-
       0.25*exp(-((9*x-4)*(9*x-4)+(9*y-7)*(9*y-7)+
       (9*z-5)*(9*z-5)))+0.1;
end;


function F2(x,y,z:Extended):Extended;
begin
   F2:=(tanh(9*z-9*x-9*y)+1)/9;
```

```
end;

function F3(x,y,z:Extended):Extended;(*acrescida de 0,37 para se
tornar n o negativa*)
begin
   F3:=(1.25+cos(5.4*y))*cos(6*z)/(6+6*(3*x-1)*(3*x-1))+0.37;
end;

function F4(x,y,z:Extended):Extended;
begin
   F4:=exp((-81/16)*((x-0.5)*(x-0.5)+(y-0.5)*(y-0.5)+
   (z-0.5)*(z-0.5)))/3;
end;

function F5(x,y,z:Extended):Extended;
begin
   F5:=exp((-81/4)*((x-0.5)*(x-0.5)+(y-0.5)*(y-0.5)+
   (z-0.5)*(z-0.5)))/3;
end;

function F6(x,y,z:Extended):Extended;(*acrescida de 0,3 para se
tornar n o negativa*)
begin
   F6:=sqrt(64-81*((x-0.5)*(x-0.5)+(y-0.5)*(y-0.5)+
   (z-0.5)*(z-0.5)))/9-0.2;
end;



procedure reinicializa_para_RMS(func:integer);
var    x,y,z,fp,norma:Extended;(* todas as fun  es tomam
valores entre 0 e 1 *)
       i:integer;
begin
   MaxF:=0;
   if not F_jah_inicializado then begin
      for i:=0 to Npts-1 do begin
         Lpts[i].x:=(Lpts[i].x-MinX)/deltaX;
         Lpts[i].y:=(Lpts[i].y-MinY)/deltay;
         Lpts[i].z:=(Lpts[i].z-MinZ)/deltaZ;
      end;
      F_jah_inicializado:=True;
      planoX:=(planoX - MinX)/deltaX;
```

```
   planoY:=(planoY - MinY)/deltaY;
   planoZ:=(planoZ - MinZ)/deltaZ;
end;
for i:=0 to Npts-1 do begin
   case func of
      1:Lpts[i].fp:=F1(Lpts[i].x,Lpts[i].y,Lpts[i].z);
      2:Lpts[i].fp:=F2(Lpts[i].x,Lpts[i].y,Lpts[i].z);
      3:Lpts[i].fp:=F3(Lpts[i].x,Lpts[i].y,Lpts[i].z);
      4:Lpts[i].fp:=F4(Lpts[i].x,Lpts[i].y,Lpts[i].z);
      5:Lpts[i].fp:=F5(Lpts[i].x,Lpts[i].y,Lpts[i].z);
      6:Lpts[i].fp:=F6(Lpts[i].x,Lpts[i].y,Lpts[i].z);
   end;
   if MaxF<Lpts[i].fp then MaxF:=Lpts[i].fp;
   Lpts[i].vol:=0;
   Lpts[i].nx:=0;
   Lpts[i].ny:=0;
   Lpts[i].nz:=0;
   Lpts[i].alfa:=1.0;
end;
for i:=0 to Ntets-1 do begin
   Ltets[i].l:=Lpts[Ltets[i].V1].fp;
   Ltets[i].visitado:=False;
   if Ltets[i].l>Lpts[Ltets[i].V2].fp then
      Ltets[i].l:=Lpts[Ltets[i].V2].fp;
   if Ltets[i].l>Lpts[Ltets[i].V3].fp then
      Ltets[i].l:=Lpts[Ltets[i].V3].fp;
   if Ltets[i].l>Lpts[Ltets[i].V4].fp then
      Ltets[i].l:=Lpts[Ltets[i].V4].fp;
   Ltets[i].compute_gradiente;
   Ltets[i].preenche_normais;
   Lpts[Ltets[i].V1].nx:=Lpts[Ltets[i].V1].nx+
   Ltets[i].gx/Ltets[i].volume;
   Lpts[Ltets[i].V1].ny:=Lpts[Ltets[i].V1].ny+
   Ltets[i].gy/Ltets[i].volume;
   Lpts[Ltets[i].V1].nz:=Lpts[Ltets[i].V1].nz+
   Ltets[i].gz/Ltets[i].volume;
   Lpts[Ltets[i].V2].nx:=Lpts[Ltets[i].V2].nx+
   Ltets[i].gx/Ltets[i].volume;
   Lpts[Ltets[i].V2].ny:=Lpts[Ltets[i].V2].ny+
   Ltets[i].gy/Ltets[i].volume;
   Lpts[Ltets[i].V2].nz:=Lpts[Ltets[i].V2].nz+
   Ltets[i].gz/Ltets[i].volume;
   Lpts[Ltets[i].V3].nx:=Lpts[Ltets[i].V3].nx+
```

```
      Ltets[i].gx/Ltets[i].volume;
      Lpts[Ltets[i].V3].ny:=Lpts[Ltets[i].V3].ny+
      Ltets[i].gy/Ltets[i].volume;
      Lpts[Ltets[i].V3].nz:=Lpts[Ltets[i].V3].nz+
      Ltets[i].gz/Ltets[i].volume;
      Lpts[Ltets[i].V4].nx:=Lpts[Ltets[i].V4].nx+
      Ltets[i].gx/Ltets[i].volume;
      Lpts[Ltets[i].V4].ny:=Lpts[Ltets[i].V4].ny+

      Ltets[i].gy/Ltets[i].volume;
      Lpts[Ltets[i].V4].nz:=Lpts[Ltets[i].V4].nz+
      Ltets[i].gz/Ltets[i].volume;
      Lpts[Ltets[i].V1].vol:=Lpts[Ltets[i].V1].vol +
      1/Ltets[i].volume;
      Lpts[Ltets[i].V2].vol:=Lpts[Ltets[i].V2].vol +
      1/Ltets[i].volume;
      Lpts[Ltets[i].V3].vol:=Lpts[Ltets[i].V3].vol +
      1/Ltets[i].volume;
      Lpts[Ltets[i].V4].vol:=Lpts[Ltets[i].V4].vol +
      1/Ltets[i].volume;
   end;
   for i:=0 to Npts-1 do begin
      Lpts[i].nx:=Lpts[i].nx/Lpts[i].vol;
      Lpts[i].ny:=Lpts[i].ny/Lpts[i].vol;
      Lpts[i].nz:=Lpts[i].nz/Lpts[i].vol;
   end;
   for i:=0 to Ntets-1 do begin
      Ltets[i].gera_G;
      Ltets[i].preenche_M1;
      Ltets[i].mini_centroides;
      Ltets[i].preenche_M2inicial;
   end;
   for i:=0 to Ntets-1 do begin
      Ltets[i].preenche_M2;
      Ltets[i].preenche_M3;
      Ltets[i].preenche_M4;
      Ltets[i].preenche_M5;
//      Ltets[i].mini_centroides;
      Ltets[i].preenche_M6;
      Ltets[i].preenche_M7;
      Ltets[i].preenche_M8;
      Ltets[i].preenche_M9;
   end;
```

```
end;

(* Este procedimento retornao o valor de uma fun  o Fi dentro do
inv lucro *)
function DesenhaFi(x,y,z:Extended; tipo:integer):Extended;
var i,j:integer;             (*retorna -1000 se o ponto cair fora do *)
                             (* inv lucro convexo*)
    alfa,beta,gama,delta,func:Extended;
    fora:boolean;
    dx,dy,dz:Extended;
begin
   fora:=True;
   for i:=0 to Ntets-1 do if Ltets[i].interior(x,y,z) then begin
      case tipo of
         1:func:=F1(x,y,z);
         2:func:=F2(x,y,z);
         3:func:=F3(x,y,z);
         4:func:=F4(x,y,z);
         5:func:=F5(x,y,z);
         6:func:=F6(x,y,z);
      end;
      fora:=False;
      Ltets[i].visitado:=True;
      break;
   end;
   if fora then
      DesenhaFi:=-1000
   else
      DesenhaFi:=func;
end;


end.



unit uTetBezier;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
  Dialogs,
```

```
  StdCtrls,uTetEstrutura, uBitMap, ExtCtrls, ExtDlgs,Math;

type
  TForm1 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Button1: TButton;
    AbreArquivo: TOpenDialog;
    Image1: TImage;
    EixoOrtogonal: TRadioGroup;
    Button2: TButton;
    Button3: TButton;
    Edit1: TEdit;
    Label3: TLabel;
    SavePictureDialog1: TSavePictureDialog;
    CheckBox1: TCheckBox;
    CheckBox2: TCheckBox;
    Button4: TButton;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    Label10: TLabel;
    Label11: TLabel;
    LMinX: TLabel;
    LMaxX: TLabel;
    LMinY: TLabel;
    LMaxY: TLabel;
    LMinZ: TLabel;
    LMaxZ: TLabel;
    Label12: TLabel;
    Label13: TLabel;
    Label14: TLabel;
    LplanoX: TLabel;
    LplanoY: TLabel;
    LplanoZ: TLabel;
    Label15: TLabel;
    Label16: TLabel;
    NumTets: TLabel;
    NumTetsCort: TLabel;
    Button5: TButton;
    Button6: TButton;
    NumCels: TEdit;
```

```
Label17: TLabel;
Label18: TLabel;
Label19: TLabel;
Label20: TLabel;
Label21: TLabel;
Label22: TLabel;
Label23: TLabel;
LF1: TLabel;
LF2: TLabel;
LF3: TLabel;
LF4: TLabel;
LF5: TLabel;
LF6: TLabel;
Button7: TButton;
Button8: TButton;
Button9: TButton;
Button10: TButton;
Button11: TButton;
Button12: TButton;
Label24: TLabel;
Label4: TLabel;
LMaxF: TLabel;
LF1s: TLabel;
LF1l: TLabel;
LF2s: TLabel;
LF2l: TLabel;
LF3s: TLabel;
LF3l: TLabel;
LF4s: TLabel;
LF4l: TLabel;
LF5s: TLabel;
LF5l: TLabel;
Button13: TButton;
Button14: TButton;
LF6s: TLabel;
LF6l: TLabel;
LFuncCarregada: TLabel;
Label5: TLabel;
Label25: TLabel;
Label26: TLabel;
Label27: TLabel;
Label28: TLabel;
Label29: TLabel;
```

```
    Label30: TLabel;
    Label31: TLabel;
    Label32: TLabel;
    Label33: TLabel;
    Label34: TLabel;
    Label35: TLabel;
    Label36: TLabel;
    Label37: TLabel;
    Label38: TLabel;
    Label39: TLabel;
    Label40: TLabel;
    Label41: TLabel;
    FuncDesenhada: TLabel;
    Label42: TLabel;
    LNumPts: TLabel;
    RadioGroup1: TRadioGroup;
    RadioGroup2: TRadioGroup;
    Button17: TButton;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure EixoOrtogonalClick(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure Button5Click(Sender: TObject);
    procedure Button6Click(Sender: TObject);
    procedure Button13Click(Sender: TObject);
    procedure Button14Click(Sender: TObject);
    procedure Button7Click(Sender: TObject);
    procedure Button8Click(Sender: TObject);
    procedure Button9Click(Sender: TObject);
    procedure Button10Click(Sender: TObject);
    procedure Button11Click(Sender: TObject);
    procedure Button12Click(Sender: TObject);
    procedure Button17Click(Sender: TObject);
//    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
```

```
  passo:Extended;
  Tela:TBitMap;
  Red0,Green0,Blue0,Red1,Green1,Blue1:Extended;
  Red_1,Green_1,Blue_1:Extended;
  TetsCortados:integer;
  Desenho_atual:integer;

implementation

{$R *.DFM}

procedure desenha_tetraedro_projetado(i,eixo:integer);
var x,y,z:Extended;
    deve_desenhar:Boolean;
begin
   deve_desenhar:=False;
   if Form1.CheckBox1.Checked then deve_desenhar:=True
   else if Ltets[i].visitado then deve_desenhar:=True;
   if deve_desenhar then
     case eixo of
       0:begin
         y:=Largura*(Lpts[Ltets[i].V1].y-MinY)/deltaY;
         z:=Altura*(Lpts[Ltets[i].V1].z-MinZ)/deltaZ;
         Bitmap.Canvas.MoveTo(round(y),Altura-round(z));
         y:=Largura*(Lpts[Ltets[i].V2].y-MinY)/deltaY;
         z:=Altura*(Lpts[Ltets[i].V2].z-MinZ)/deltaZ;
         Bitmap.Canvas.LineTo(round(y),Altura-round(z));
         y:=Largura*(Lpts[Ltets[i].V3].y-MinY)/deltaY;
         z:=Altura*(Lpts[Ltets[i].V3].z-MinZ)/deltaZ;
         Bitmap.Canvas.LineTo(round(y),Altura-round(z));
         y:=Largura*(Lpts[Ltets[i].V4].y-MinY)/deltaY;
         z:=Altura*(Lpts[Ltets[i].V4].z-MinZ)/deltaZ;
         Bitmap.Canvas.LineTo(round(y),Altura-round(z));
         y:=Largura*(Lpts[Ltets[i].V2].y-MinY)/deltaY;
         z:=Altura*(Lpts[Ltets[i].V2].z-MinZ)/deltaZ;
         Bitmap.Canvas.MoveTo(round(y),Altura-round(z));
         y:=Largura*(Lpts[Ltets[i].V4].y-MinY)/deltaY;
         z:=Altura*(Lpts[Ltets[i].V4].z-MinZ)/deltaZ;
         Bitmap.Canvas.LineTo(round(y),Altura-round(z));
         y:=Largura*(Lpts[Ltets[i].V1].y-MinY)/deltaY;
         z:=Altura*(Lpts[Ltets[i].V1].z-MinZ)/deltaZ;
         Bitmap.Canvas.LineTo(round(y),Altura-round(z));
         y:=Largura*(Lpts[Ltets[i].V3].y-MinY)/deltaY;
```

```
  z:=Altura*(Lpts[Ltets[i].V3].z-MinZ)/deltaZ;
  Bitmap.Canvas.LineTo(round(y),Altura-round(z));
end;
1:begin
  x:=Largura*(Lpts[Ltets[i].V1].x-MinX)/deltaX;
  z:=Altura*(Lpts[Ltets[i].V1].z-MinZ)/deltaZ;
  Bitmap.Canvas.MoveTo(round(x),Altura-round(z));
  x:=Largura*(Lpts[Ltets[i].V2].x-MinX)/deltaX;
  z:=Altura*(Lpts[Ltets[i].V2].z-MinZ)/deltaZ;
  Bitmap.Canvas.LineTo(round(x),Altura-round(z));
  x:=Largura*(Lpts[Ltets[i].V3].x-MinX)/deltaX;
  z:=Altura*(Lpts[Ltets[i].V3].z-MinZ)/deltaZ;
  Bitmap.Canvas.LineTo(round(x),Altura-round(z));
  x:=Largura*(Lpts[Ltets[i].V4].x-MinX)/deltaX;
  z:=Altura*(Lpts[Ltets[i].V4].z-MinZ)/deltaZ;
  Bitmap.Canvas.LineTo(round(x),Altura-round(z));
  x:=Largura*(Lpts[Ltets[i].V2].x-MinX)/deltaX;
  z:=Altura*(Lpts[Ltets[i].V2].z-MinZ)/deltaZ;
  Bitmap.Canvas.MoveTo(round(x),Altura-round(z));
  x:=Largura*(Lpts[Ltets[i].V4].x-MinX)/deltaX;
  z:=Altura*(Lpts[Ltets[i].V4].z-MinZ)/deltaZ;
  Bitmap.Canvas.LineTo(round(x),Altura-round(z));
  x:=Largura*(Lpts[Ltets[i].V1].x-MinX)/deltaX;
  z:=Altura*(Lpts[Ltets[i].V1].z-MinZ)/deltaZ;
  Bitmap.Canvas.LineTo(round(x),Altura-round(z));
  x:=Largura*(Lpts[Ltets[i].V3].x-MinX)/deltaX;
  z:=Altura*(Lpts[Ltets[i].V3].z-MinZ)/deltaZ;
  Bitmap.Canvas.LineTo(round(x),Altura-round(z));
end;
2:begin
  y:=Altura*(Lpts[Ltets[i].V1].y-MinY)/deltaY;
  x:=Largura*(Lpts[Ltets[i].V1].x-MinX)/deltaX;
  Bitmap.Canvas.MoveTo(round(x),Altura-round(y));
  y:=Altura*(Lpts[Ltets[i].V2].y-MinY)/deltaY;
  x:=Largura*(Lpts[Ltets[i].V2].x-MinX)/deltaX;
  Bitmap.Canvas.LineTo(round(x),Altura-round(y));
  y:=Altura*(Lpts[Ltets[i].V3].y-MinY)/deltaY;
  x:=Largura*(Lpts[Ltets[i].V3].x-MinX)/deltaX;
  Bitmap.Canvas.LineTo(round(x),Altura-round(y));
  y:=Altura*(Lpts[Ltets[i].V4].y-MinY)/deltaY;
  x:=Largura*(Lpts[Ltets[i].V4].x-MinX)/deltaX;
  Bitmap.Canvas.LineTo(round(x),Altura-round(y));
  y:=Altura*(Lpts[Ltets[i].V2].y-MinY)/deltaY;
```

```
          x:=Largura*(Lpts[Ltets[i].V2].x-MinX)/deltaX;
          Bitmap.Canvas.MoveTo(round(x),Altura-round(y));
          y:=Altura*(Lpts[Ltets[i].V4].y-MinY)/deltaY;
          x:=Largura*(Lpts[Ltets[i].V4].x-MinX)/deltaX;
          Bitmap.Canvas.LineTo(round(x),Altura-round(y));
          y:=Altura*(Lpts[Ltets[i].V1].y-MinY)/deltaY;
          x:=Largura*(Lpts[Ltets[i].V1].x-MinX)/deltaX;
          Bitmap.Canvas.LineTo(round(x),Altura-round(y));
          y:=Altura*(Lpts[Ltets[i].V3].y-MinY)/deltaY;
          x:=Largura*(Lpts[Ltets[i].V3].x-MinX)/deltaX;
          Bitmap.Canvas.LineTo(round(x),Altura-round(y));
        end;
      end;
end;

procedure desenha;
var x,y,z,f:Extended;
    i,j,red,green,blue:integer;
    Farq:textfile;
begin
   for i:=0 to Ntets-1 do Ltets[i].visitado:=False;
   TetsCortados:=0;
   case Form1.EixoOrtogonal.ItemIndex of
      0:begin
           for i:=0 to Largura do begin
              y:=MinY+(i/Largura)*deltaY;
              for j:=0 to Altura do begin
                 z:=MinZ+(j/Altura)*deltaZ;
                 f:=Splines(planoX,y,z)/MaxF;
                 if f<0 then begin
                    if f<-1 then begin
                       red:=255; green:=255; blue:=255;
                    end
                    else begin
                       f:=-f;
                       red:=round((1-f)*Red0 + f*Red_1);
                       green:=round((1-f)*Green0 + f*Green_1);
                       blue:=round((1-f)*Blue0 + f*Blue_1);
                    end;
                 end
                 else begin
                    if f>1 then begin
                       if f>=2 then begin
```

```
                  red:=240; green:=240; blue:=0;
               end
               else begin
                  red:=round((f-1)*240 + (2-f)*Red1);
                  green:=round((f-1)*240 + (2-f)*Green1);
                  blue:=round((2-f)*Blue1);
               end;
            end
            else begin
               red:=round((1-f)*Red0 + f*Red1);
               green:=round((1-f)*Green0 + f*Green1);
               blue:=round((1-f)*Blue0 + f*Blue1);
            end
         end;
         Tela.Canvas.Pixels[i,Altura-j]:=RGB(red,green,blue);
      end;
   end;
   if Form1.CheckBox2.Checked then begin
      Bitmap.Canvas.Pen.Color:=RGB(255,255,0);
      for i:=0 to Ntets-1 do
         desenha_tetraedro_projetado(i,0);
   end;
  end;
1:begin
   for i:=0 to Largura do begin
      x:=MinX+(i/Largura)*deltaX;
      for j:=0 to Altura do begin
         z:=MinZ+(j/Altura)*deltaZ;
         f:=Splines(x,planoY,z)/MaxF;
         if f<0 then begin
            if f<-1 then begin
               red:=255; green:=255; blue:=255;
            end
            else begin
               f:=-f;
               red:=round((1-f)*Red0 + f*Red_1);
               green:=round((1-f)*Green0 + f*Green_1);
               blue:=round((1-f)*Blue0 + f*Blue_1);
            end;
         end
         else begin
            if f>1 then begin
               if f>=2 then begin
```

```
                  red:=240; green:=240; blue:=0;
               end
               else begin
                  red:=round((f-1)*240 + (2-f)*Red1);
                  green:=round((f-1)*240 + (2-f)*Green1);
                  blue:=round((2-f)*Blue1);
               end;
            end
            else begin
               red:=round((1-f)*Red0 + f*Red1);
               green:=round((1-f)*Green0 + f*Green1);
               blue:=round((1-f)*Blue0 + f*Blue1);
            end
         end;
         Tela.Canvas.Pixels[i,Altura-j]:=RGB(red,green,blue);
      end;
   end;
   if Form1.CheckBox2.Checked then begin
      Bitmap.Canvas.Pen.Color:=RGB(255,255,1);
      for i:=0 to Ntets-1 do
         desenha_tetraedro_projetado(i,1);
   end;
  end;
2:begin
   for i:=0 to Largura do begin
      x:=MinX+(i/Largura)*deltaX;
      for j:=0 to Altura do begin
         y:=MinY+(j/Altura)*deltaY;
         f:=Splines(x,y,planoZ)/MaxF;
         if f<0 then begin
            if f<-1 then begin
               red:=255; green:=255; blue:=255;
            end
            else begin
               f:=-f;
               red:=round((1-f)*Red0 + f*Red_1);
               green:=round((1-f)*Green0 + f*Green_1);
               blue:=round((1-f)*Blue0 + f*Blue_1);
            end;
         end
         else begin
            if f>1 then begin
               if f>=2 then begin
```

```
                        red:=240; green:=240; blue:=0;
                     end
                     else begin
                        red:=round((f-1)*240 + (2-f)*Red1);
                        green:=round((f-1)*240 + (2-f)*Green1);
                        blue:=round((2-f)*Blue1);
                     end;
                  end
                  else begin
                     red:=round((1-f)*Red0 + f*Red1);
                     green:=round((1-f)*Green0 + f*Green1);
                     blue:=round((1-f)*Blue0 + f*Blue1);
                  end
               end;
               Tela.Canvas.Pixels[i,Altura-j]:=RGB(red,green,blue);
            end;
         end;
         if Form1.CheckBox2.Checked then begin
            Bitmap.Canvas.Pen.Color:=RGB(255,255,2);
            for i:=0 to Ntets-1 do
               desenha_tetraedro_projetado(i,2);
         end;
      end;
   end;
   Form1.FuncDesenhada.Caption:='Interpola  o usada: B zier';
   for i:=0 to Ntets-1 do if Ltets[i].visitado then TetsCortados:=
   TetsCortados+1;
   Form1.Image1.Picture.Bitmap:=Tela;
   Form1.NumTetsCort.Caption:=inttostr(TetsCortados);
   desenho_atual:=0;

(*   AssignFile(Farq,'debug.txt');
   Rewrite(Farq);
   for i:=0 to Npts-1 do begin
      f:=Splines(Lpts[i].x,Lpts[i].y,Lpts[i].z);
      write(Farq,'(',inttostr(i),'):spl=',f);
      f:=F1(Lpts[i].x,Lpts[i].y,Lpts[i].z);
      write(Farq,' F1=',f);
      f:=F2(Lpts[i].x,Lpts[i].y,Lpts[i].z);
      write(Farq,' F2=',f);
      f:=F3(Lpts[i].x,Lpts[i].y,Lpts[i].z);
      write(Farq,' F3=',f);
      f:=F4(Lpts[i].x,Lpts[i].y,Lpts[i].z);
```

```
      write(Farq,' F4=',f);
      f:=F5(Lpts[i].x,Lpts[i].y,Lpts[i].z);
      write(Farq,' F5=',f);
      f:=F6(Lpts[i].x,Lpts[i].y,Lpts[i].z);
      write(Farq,' F6=',f);
      writeln(Farq,' Valor original',Lpts[i].fp);
   end;
   closefile(Farq);*)
end;



procedure desenha_shep;
var x,y,z,f:Extended;
    i,j,red,green,blue:integer;
begin
   for i:=0 to Ntets-1 do Ltets[i].visitado:=False;
   TetsCortados:=0;
   case Form1.EixoOrtogonal.ItemIndex of
      0:begin
           for i:=0 to Largura do begin
              y:=MinY+(i/Largura)*deltaY;
              for j:=0 to Altura do begin
                 z:=MinZ+(j/Altura)*deltaZ;
                 f:=Shepard(planoX,y,z)/MaxF;
                 if f<0 then begin
                    if f<-1 then begin
                       red:=255; green:=255; blue:=255;
                    end
                    else begin
                       f:=-f;
                       red:=round((1-f)*Red0 + f*Red_1);
                       green:=round((1-f)*Green0 + f*Green_1);
                       blue:=round((1-f)*Blue0 + f*Blue_1);
                    end;
                 end
                 else begin
                    if f>1 then begin
                       if f>=2 then begin
                          red:=240; green:=240; blue:=0;
                       end
                       else begin
                          red:=round((f-1)*240 + (2-f)*Red1);
```

```
                  green:=round((f-1)*240 + (2-f)*Green1);
                  blue:=round((2-f)*Blue1);
               end;
            end
            else begin
               red:=round((1-f)*Red0 + f*Red1);
               green:=round((1-f)*Green0 + f*Green1);
               blue:=round((1-f)*Blue0 + f*Blue1);
            end
         end;
         Tela.Canvas.Pixels[i,Altura-j]:=RGB(red,green,blue);
      end;
   end;
   if Form1.CheckBox2.Checked then begin
      Bitmap.Canvas.Pen.Color:=RGB(255,255,0);
      for i:=0 to Ntets-1 do
         desenha_tetraedro_projetado(i,0);
   end;
 end;
1:begin
   for i:=0 to Largura do begin
      x:=MinX+(i/Largura)*deltaX;
      for j:=0 to Altura do begin
         z:=MinZ+(j/Altura)*deltaZ;
         f:=Shepard(x,planoY,z)/MaxF;
         if f<0 then begin
            if f<-1 then begin
               red:=255; green:=255; blue:=255;
            end
            else begin
               f:=-f;
               red:=round((1-f)*Red0 + f*Red_1);
               green:=round((1-f)*Green0 + f*Green_1);
               blue:=round((1-f)*Blue0 + f*Blue_1);
            end;
         end
         else begin
            if f>1 then begin
               if f>=2 then begin
                  red:=240; green:=240; blue:=0;
               end
               else begin
                  red:=round((f-1)*240 + (2-f)*Red1);
```

```
                  green:=round((f-1)*240 + (2-f)*Green1);
                  blue:=round((2-f)*Blue1);
               end;
            end
            else begin
               red:=round((1-f)*Red0 + f*Red1);
               green:=round((1-f)*Green0 + f*Green1);
               blue:=round((1-f)*Blue0 + f*Blue1);
            end
         end;
         Tela.Canvas.Pixels[i,Altura-j]:=RGB(red,green,blue);
      end;
   end;
   if Form1.CheckBox2.Checked then begin
      Bitmap.Canvas.Pen.Color:=RGB(255,255,1);
      for i:=0 to Ntets-1 do
         desenha_tetraedro_projetado(i,1);
   end;
 end;
2:begin
   for i:=0 to Largura do begin
      x:=MinX+(i/Largura)*deltaX;
      for j:=0 to Altura do begin
         y:=MinY+(j/Altura)*deltaY;
         f:=Shepard(x,y,planoZ)/MaxF;
         if f<0 then begin
            if f<-1 then begin
               red:=255; green:=255; blue:=255;
            end
            else begin
               f:=-f;
               red:=round((1-f)*Red0 + f*Red_1);
               green:=round((1-f)*Green0 + f*Green_1);
               blue:=round((1-f)*Blue0 + f*Blue_1);
            end;
         end
         else begin
            if f>1 then begin
               if f>=2 then begin
                  red:=240; green:=240; blue:=0;
               end
               else begin
                  red:=round((f-1)*240 + (2-f)*Red1);
```

```
                        green:=round((f-1)*240 + (2-f)*Green1);
                        blue:=round((2-f)*Blue1);
                     end;
                  end
                  else begin
                     red:=round((1-f)*Red0 + f*Red1);
                     green:=round((1-f)*Green0 + f*Green1);
                     blue:=round((1-f)*Blue0 + f*Blue1);
                  end
               end;
               Tela.Canvas.Pixels[i,Altura-j]:=RGB(red,green,blue);
            end;
         end;
         if Form1.CheckBox2.Checked then begin
            Bitmap.Canvas.Pen.Color:=RGB(255,255,2);
            for i:=0 to Ntets-1 do
               desenha_tetraedro_projetado(i,2);
         end;
      end;
   end;
   Form1.FuncDesenhada.Caption:='Interpola  o usada: Shepard';
   for i:=0 to Ntets-1 do if Ltets[i].visitado then TetsCortados:=
   TetsCortados+1;
   Form1.Image1.Picture.Bitmap:=Tela;
   Form1.NumTetsCort.Caption:=inttostr(TetsCortados);
   desenho_atual:=-1;
end;



procedure desenha_lin; (*desenha interpola  o linear por partes*)
var x,y,z,f:Extended;
    i,j,red,green,blue:integer;
begin
   for i:=0 to Ntets-1 do Ltets[i].visitado:=False;
   TetsCortados:=0;
   case Form1.EixoOrtogonal.ItemIndex of
      0:begin
           for i:=0 to Largura do begin
              y:=MinY+(i/Largura)*deltaY;
              for j:=0 to Altura do begin
                 z:=MinZ+(j/Altura)*deltaZ;
                 f:=Linear(planoX,y,z)/MaxF;
```

```
            if f<0 then begin
               if f<-1 then begin
                  red:=255; green:=255; blue:=255;
               end
               else begin
                  f:=-f;
                  red:=round((1-f)*Red0 + f*Red_1);
                  green:=round((1-f)*Green0 + f*Green_1);
                  blue:=round((1-f)*Blue0 + f*Blue_1);
               end;
            end
            else begin
               if f>1 then begin
                  if f>=2 then begin
                     red:=240; green:=240; blue:=0;
                  end
                  else begin
                     red:=round((f-1)*240 + (2-f)*Red1);
                     green:=round((f-1)*240 + (2-f)*Green1);
                     blue:=round((2-f)*Blue1);
                  end;
               end
               else begin
                  red:=round((1-f)*Red0 + f*Red1);
                  green:=round((1-f)*Green0 + f*Green1);
                  blue:=round((1-f)*Blue0 + f*Blue1);
               end
            end;
            Tela.Canvas.Pixels[i,Altura-j]:=RGB(red,green,blue);
         end;
      end;
      if Form1.CheckBox2.Checked then begin
         Bitmap.Canvas.Pen.Color:=RGB(255,255,0);
         for i:=0 to Ntets-1 do
            desenha_tetraedro_projetado(i,0);
      end;
   end;
1:begin
      for i:=0 to Largura do begin
         x:=MinX+(i/Largura)*deltaX;
         for j:=0 to Altura do begin
            z:=MinZ+(j/Altura)*deltaZ;
            f:=Linear(x,planoY,z)/MaxF;
```

```
            if f<0 then begin
               if f<-1 then begin
                  red:=255; green:=255; blue:=255;
               end
               else begin
                  f:=-f;
                  red:=round((1-f)*Red0 + f*Red_1);
                  green:=round((1-f)*Green0 + f*Green_1);
                  blue:=round((1-f)*Blue0 + f*Blue_1);
               end;
            end
            else begin
               if f>1 then begin
                  if f>=2 then begin
                     red:=240; green:=240; blue:=0;
                  end
                  else begin
                     red:=round((f-1)*240 + (2-f)*Red1);
                     green:=round((f-1)*240 + (2-f)*Green1);
                     blue:=round((2-f)*Blue1);
                  end;
               end
               else begin
                  red:=round((1-f)*Red0 + f*Red1);
                  green:=round((1-f)*Green0 + f*Green1);
                  blue:=round((1-f)*Blue0 + f*Blue1);
               end
            end;
            Tela.Canvas.Pixels[i,Altura-j]:=RGB(red,green,blue);
         end;
      end;
      if Form1.CheckBox2.Checked then begin
         Bitmap.Canvas.Pen.Color:=RGB(255,255,1);
         for i:=0 to Ntets-1 do
            desenha_tetraedro_projetado(i,1);
      end;
   end;
2:begin
      for i:=0 to Largura do begin
         x:=MinX+(i/Largura)*deltaX;
         for j:=0 to Altura do begin
            y:=MinY+(j/Altura)*deltaY;
            f:=Linear(x,y,planoZ)/MaxF;
```

```
                if f<0 then begin
                   if f<-1 then begin
                      red:=255; green:=255; blue:=255;
                   end
                   else begin
                      f:=-f;
                      red:=round((1-f)*Red0 + f*Red_1);
                      green:=round((1-f)*Green0 + f*Green_1);
                      blue:=round((1-f)*Blue0 + f*Blue_1);
                   end;
                end
                else begin
                   if f>1 then begin
                      if f>=2 then begin
                         red:=240; green:=240; blue:=0;
                      end
                      else begin
                         red:=round((f-1)*240 + (2-f)*Red1);
                         green:=round((f-1)*240 + (2-f)*Green1);
                         blue:=round((2-f)*Blue1);
                      end;
                   end
                   else begin
                      red:=round((1-f)*Red0 + f*Red1);
                      green:=round((1-f)*Green0 + f*Green1);
                      blue:=round((1-f)*Blue0 + f*Blue1);
                   end
                end;
                Tela.Canvas.Pixels[i,Altura-j]:=RGB(red,green,blue);
             end;
          end;
          if Form1.CheckBox2.Checked then begin
             Bitmap.Canvas.Pen.Color:=RGB(255,255,2);
             for i:=0 to Ntets-1 do
                desenha_tetraedro_projetado(i,2);
          end;
       end;
    end;
    Form1.FuncDesenhada.Caption:='Interpola  o usada: Linear';
    for i:=0 to Ntets-1 do if Ltets[i].visitado then TetsCortados:=
    TetsCortados+1;
    Form1.Image1.Picture.Bitmap:=Tela;
    Form1.NumTetsCort.Caption:=inttostr(TetsCortados);
```

```
   desenho_atual:=-2;
end;

procedure desenhaFuncoes(tipo:integer);
var x,y,z,f:Extended;
    i,j,red,green,blue:integer;
begin
   for i:=0 to Ntets-1 do Ltets[i].visitado:=False;
   TetsCortados:=0;
   case Form1.EixoOrtogonal.ItemIndex of
      0:begin
           for i:=0 to Largura do begin
              y:=MinY+(i/Largura)*deltaY;
              for j:=0 to Altura do begin
                 z:=MinZ+(j/Altura)*deltaZ;
                 f:=DesenhaFi(planoX,y,z,tipo)/MaxF;
                 if f<0 then begin
                    if f<-1 then begin
                       red:=255; green:=255; blue:=255;
                    end
                    else begin
                       f:=-f;
                       red:=round((1-f)*Red0 + f*Red_1);
                       green:=round((1-f)*Green0 + f*Green_1);
                       blue:=round((1-f)*Blue0 + f*Blue_1);
                    end;
                 end
                 else begin
                    if f>1 then begin
                       if f>=2 then begin
                          red:=240; green:=240; blue:=0;
                       end
                       else begin
                          red:=round((f-1)*240 + (2-f)*Red1);
                          green:=round((f-1)*240 + (2-f)*Green1);
                          blue:=round((2-f)*Blue1);
                       end;
                    end
                    else begin
                       red:=round((1-f)*Red0 + f*Red1);
                       green:=round((1-f)*Green0 + f*Green1);
                       blue:=round((1-f)*Blue0 + f*Blue1);
                    end
```

```
            end;
            Tela.Canvas.Pixels[i,Altura-j]:=RGB(red,green,blue);
         end;
      end;
      if Form1.CheckBox2.Checked then begin
         Bitmap.Canvas.Pen.Color:=RGB(255,255,0);
         for i:=0 to Ntets-1 do
            desenha_tetraedro_projetado(i,0);
      end;
   end;
1:begin
      for i:=0 to Largura do begin
         x:=MinX+(i/Largura)*deltaX;
         for j:=0 to Altura do begin
            z:=MinZ+(j/Altura)*deltaZ;
            f:=DesenhaFi(x,planoY,z,tipo)/MaxF;
            if f<0 then begin
               if f<-1 then begin
                  red:=255; green:=255; blue:=255;
               end
               else begin
                  f:=-f;
                  red:=round((1-f)*Red0 + f*Red_1);
                  green:=round((1-f)*Green0 + f*Green_1);
                  blue:=round((1-f)*Blue0 + f*Blue_1);
               end;
            end
            else begin
               if f>1 then begin
                  if f>=2 then begin
                     red:=240; green:=240; blue:=0;
                  end
                  else begin
                     red:=round((f-1)*240 + (2-f)*Red1);
                     green:=round((f-1)*240 + (2-f)*Green1);
                     blue:=round((2-f)*Blue1);
                  end;
               end
               else begin
                  red:=round((1-f)*Red0 + f*Red1);
                  green:=round((1-f)*Green0 + f*Green1);
                  blue:=round((1-f)*Blue0 + f*Blue1);
               end
```

```
            end;
            Tela.Canvas.Pixels[i,Altura-j]:=RGB(red,green,blue);
         end;
      end;
      if Form1.CheckBox2.Checked then begin
         Bitmap.Canvas.Pen.Color:=RGB(255,255,1);
         for i:=0 to Ntets-1 do
            desenha_tetraedro_projetado(i,1);
      end;
   end;
2:begin
      for i:=0 to Largura do begin
         x:=MinX+(i/Largura)*deltaX;
         for j:=0 to Altura do begin
            y:=MinY+(j/Altura)*deltaY;
            f:=DesenhaFi(x,y,planoZ,tipo)/MaxF;
            if f<0 then begin
               if f<-1 then begin
                  red:=255; green:=255; blue:=255;
               end
               else begin
                  f:=-f;
                  red:=round((1-f)*Red0 + f*Red_1);
                  green:=round((1-f)*Green0 + f*Green_1);
                  blue:=round((1-f)*Blue0 + f*Blue_1);
               end;
            end
            else begin
               if f>1 then begin
                  if f>=2 then begin
                     red:=240; green:=240; blue:=0;
                  end
                  else begin
                     red:=round((f-1)*240 + (2-f)*Red1);
                     green:=round((f-1)*240 + (2-f)*Green1);
                     blue:=round((2-f)*Blue1);
                  end;
               end
               else begin
                  red:=round((1-f)*Red0 + f*Red1);
                  green:=round((1-f)*Green0 + f*Green1);
                  blue:=round((1-f)*Blue0 + f*Blue1);
               end
```

```
                end;
                Tela.Canvas.Pixels[i,Altura-j]:=RGB(red,green,blue);
             end;
          end;
          if Form1.CheckBox2.Checked then begin
             Bitmap.Canvas.Pen.Color:=RGB(255,255,2);
             for i:=0 to Ntets-1 do
                desenha_tetraedro_projetado(i,2);
          end;
       end;
   end;
   Form1.FuncDesenhada.Caption:='Fun  o desenhada: F'+inttostr(tipo);
   for i:=0 to Ntets-1 do if Ltets[i].visitado then TetsCortados:=
   TetsCortados+1;
   Form1.Image1.Picture.Bitmap:=Tela;
   Form1.NumTetsCort.Caption:=inttostr(TetsCortados);
   Form1.LMaxF.Caption:=floattostr(MaxF);
   desenho_atual:=tipo;
end;



function Diferenca(x,y,z:Extended; tipo1,tipo2:integer):Extended;
var f1,f2,MF:Extended;
begin
   MF:=1.25*MaxF;
   case tipo1 of
      0:f1:=Splines(x,y,z)/MF;
      1:f1:=Shepard(x,y,z)/MF;
      2:f1:=Linear(x,y,z)/MF;
   end;
   if f1<-500 then  Diferenca:=f1
   else begin
      if tipo2<=2 then
         case tipo2 of
            0:f2:=Splines(x,y,z)/MF;
            1:f2:=Shepard(x,y,z)/MF;
            2:f2:=Linear(x,y,z)/MF;
         end
      else
         f2:=DesenhaFi(x,y,z,tipo2-2)/MF;
      Diferenca:=f1-f2;
   end;
```

```
end;


//  Pinta a imagem de diferen a entre tipo1 e tipo2
procedure desenhaDiferenca(tipo1,tipo2:integer);
var x,y,z,f:Extended;
    i,j,red,green,blue:integer;
begin
   for i:=0 to Ntets-1 do Ltets[i].visitado:=False;
   TetsCortados:=0;
   case Form1.EixoOrtogonal.ItemIndex of
      0:begin
           for i:=0 to Largura do begin
              y:=MinY+(i/Largura)*deltaY;
              for j:=0 to Altura do begin
                 z:=MinZ+(j/Altura)*deltaZ;
                 f:=Diferenca(planoX,y,z,tipo1,tipo2);
                 if f<0 then begin
                    if f<-500 then begin
                       red:=255; green:=255; blue:=255;
                    end
                    else begin
                       if f<-1 then f:=-1;
                       f:=-f;
                       red:=round(f*255); green:=0; blue:=0;
                    end
                 end
                 else begin
                    if f>1 then f:=1;
                    red:=0; green:=0; blue:=round(f*255);
                 end;
                 Tela.Canvas.Pixels[i,Altura-j]:=RGB(red,green,blue);
              end;
           end;
           if Form1.CheckBox2.Checked then begin
              Bitmap.Canvas.Pen.Color:=RGB(255,255,0);
              for i:=0 to Ntets-1 do
                 desenha_tetraedro_projetado(i,0);
           end;
        end;
      1:begin
           for i:=0 to Largura do begin
              x:=MinX+(i/Largura)*deltaX;
```

```
         for j:=0 to Altura do begin
            z:=MinZ+(j/Altura)*deltaZ;
            f:=Diferenca(x,planoY,z,tipo1,tipo2);
            if f<0 then begin
               if f<-500 then begin
                  red:=255; green:=255; blue:=255;
               end
               else begin
                  if f<-1 then f:=-1;
                  f:=-f;
                  red:=round(f*255); green:=0; blue:=0;
               end
            end
            else begin
               if f>1 then f:=1;
               red:=0; green:=0; blue:=round(f*255);
            end;
            Tela.Canvas.Pixels[i,Altura-j]:=RGB(red,green,blue);
         end;
      end;
      if Form1.CheckBox2.Checked then begin
         Bitmap.Canvas.Pen.Color:=RGB(255,255,1);
         for i:=0 to Ntets-1 do
            desenha_tetraedro_projetado(i,1);
      end;
   end;
2:begin
      for i:=0 to Largura do begin
         x:=MinX+(i/Largura)*deltaX;
         for j:=0 to Altura do begin
            y:=MinY+(j/Altura)*deltaY;
            f:=Diferenca(x,y,planoZ,tipo1,tipo2);
            if f<0 then begin
               if f<-500 then begin
                  red:=255; green:=255; blue:=255;
               end
               else begin
                  if f<-1 then f:=-1;
                  f:=-f;
                  red:=round(f*255); green:=0; blue:=0;
               end
            end
            else begin
```

```
                  if f>1 then f:=1;
                  red:=0; green:=0; blue:=round(f*255);
               end;
               Tela.Canvas.Pixels[i,Altura-j]:=RGB(red,green,blue);
            end;
         end;
         if Form1.CheckBox2.Checked then begin
            Bitmap.Canvas.Pen.Color:=RGB(255,255,2);
            for i:=0 to Ntets-1 do
               desenha_tetraedro_projetado(i,2);
         end;
      end;
   end;
   Form1.FuncDesenhada.Caption:='Fun  o desenhada: ';
   case tipo1 of
      0:Form1.FuncDesenhada.Caption:=Form1.FuncDesenhada.Caption+
      'Bez-';
      1:Form1.FuncDesenhada.Caption:=Form1.FuncDesenhada.Caption+
      'Shep-';
      2:Form1.FuncDesenhada.Caption:=Form1.FuncDesenhada.Caption+
      'Lin-';
   end;
   if tipo2<=2 then
      case tipo2 of
      0:Form1.FuncDesenhada.Caption:=Form1.FuncDesenhada.Caption+
      'Bez';
      1:Form1.FuncDesenhada.Caption:=Form1.FuncDesenhada.Caption+
      'Shep';
      2:Form1.FuncDesenhada.Caption:=Form1.FuncDesenhada.Caption+
      'Lin';
      end
   else
      Form1.FuncDesenhada.Caption:=Form1.FuncDesenhada.Caption+'F'+
      inttostr(tipo2-2);
   for i:=0 to Ntets-1 do if Ltets[i].visitado then TetsCortados:=
   TetsCortados+1;
   Form1.Image1.Picture.Bitmap:=Tela;
   Form1.NumTetsCort.Caption:=inttostr(TetsCortados);
   Form1.LMaxF.Caption:=floattostr(MaxF);
   desenho_atual:=tipo1;
end;
```

```
procedure TForm1.Button1Click(Sender: TObject);
var i:integer;
begin
 Largura:=600;
 Altura:=600;
 AbreArquivo.Execute;
 nomearq:=AbreArquivo.FileName;
 Label2.Caption:=nomearq;
 Tela:=prepara_imagem;
 Red0:=200;
 Green0:=200;
 Blue0:=200;
 Red1:=20;
 Green1:=200;
 Blue1:=20;
 Red_1:=200;
 Green_1:=20;
 Blue_1:=20;
 inicializa;
 LMinX.Caption:=floattostr(MinX);
 LMinY.Caption:=floattostr(MinY);
 LMinZ.Caption:=floattostr(MinZ);
 LMaxX.Caption:=floattostr(MaxX);
 LMaxY.Caption:=floattostr(MaxY);
 LMaxZ.Caption:=floattostr(MaxZ);
 LMaxF.Caption:=floattostr(MaxF);
 planoX:=deltaX/2+MinX;
 planoY:=deltaY/2+MinY;
 planoZ:=deltaZ/2+MinZ;
 LplanoX.Caption:=floattostr(planoX);
 LplanoY.Caption:=floattostr(planoY);
 LplanoZ.Caption:=floattostr(planoZ);
 LFuncCarregada.Caption:='Original';
 NumTets.Caption:=inttostr(Ntets);
 LNumPts.Caption:=inttostr(Npts);
 desenha;
 Button7.Enabled:=False;
 Button8.Enabled:=False;
 Button9.Enabled:=False;
 Button10.Enabled:=False;
 Button11.Enabled:=False;
 Button12.Enabled:=False;
 Button1.Enabled:=False;
```

```
 Button17.Enabled:=False;
end;



procedure TForm1.Button2Click(Sender: TObject);
begin
   passo:=StrToFloat(Edit1.Text);
   case Form1.EixoOrtogonal.ItemIndex of
      0:begin
           planoX:=planoX+passo;
           LplanoX.Caption:=floattostr(planoX);
        end;
      1:begin
           planoY:=planoY+passo;
           LplanoY.Caption:=floattostr(planoY);
        end;
      2:begin
           planoZ:=planoZ+passo;
           LplanoZ.Caption:=floattostr(planoZ);
        end;
   end;
   if Desenho_atual=-2 then desenha_lin
   else if Desenho_atual=-1 then desenha_shep
   else if Desenho_atual=0 then desenha
   else desenhaFuncoes(desenho_atual);
end;

procedure TForm1.Button3Click(Sender: TObject);
begin
   passo:=StrToFloat(Edit1.Text);
   case Form1.EixoOrtogonal.ItemIndex of
      0:begin
           planoX:=planoX-passo;
           LplanoX.Caption:=floattostr(planoX);
        end;
      1:begin
           planoY:=planoY-passo;
           LplanoY.Caption:=floattostr(planoY);
        end;
      2:begin
           planoZ:=planoZ-passo;
           LplanoZ.Caption:=floattostr(planoZ);
```

```
        end;
   end;
   if Desenho_atual=-2 then desenha_lin
   else if Desenho_atual=-1 then desenha_shep
   else if Desenho_atual=0 then desenha
   else desenhaFuncoes(desenho_atual);
end;

procedure TForm1.EixoOrtogonalClick(Sender: TObject);
begin
   if Desenho_atual=-2 then desenha_lin
   else if Desenho_atual=-1 then desenha_shep
   else if Desenho_atual=0 then desenha
   else desenhaFuncoes(desenho_atual);
end;

procedure TForm1.Button4Click(Sender: TObject);
begin
   SavePictureDialog1.Execute;
   salvar_imagem(SavePictureDialog1.FileName);
end;

procedure TForm1.Button5Click(Sender: TObject);
begin
   desenha;
end;



procedure TForm1.Button6Click(Sender: TObject);
var RMS,x,y,z,fsplines,fshepard,flinear,dif,RMS2,RMS3,FF:Extended;
    N,i,j,k,NumPts:integer;
    arq:textfile;
    func:Extended;
begin
   N:=strtoint(NumCels.Text);
   LMinX.Caption:='0.0';
   LMinY.Caption:='0.0';
   LMinZ.Caption:='0.0';
   LMaxX.Caption:='1.0';
   LMaxY.Caption:='1.0';
   LMaxZ.Caption:='1.0';
   reinicializa_para_RMS(1);
```

```
MinX:=0; MinY:=0; MinZ:=0; MaxX:=1; MaxY:=1; MaxZ:=1;
NumPts:=0;
RMS:=0;
RMS2:=0;
RMS3:=0;
LplanoX.Caption:=floattostr(planoX);
LplanoY.Caption:=floattostr(planoY);
LplanoZ.Caption:=floattostr(planoZ);
deltaX:=1; deltaY:=1; deltaZ:=1;
for i:=0 to N do begin
   x:=i/N;
   for j:=0 to N do begin
      y:=j/N;
      for k:=0 to N do begin
         z:=k/N;
         Todos_Interpolantes(x,y,z,fsplines,fshepard,flinear);
         if fsplines>=0 then begin
            NumPts:=NumPts+1;
            FF:=F1(x,y,z);
            dif:=fsplines-FF;
            RMS:=RMS+dif*dif;
            dif:=fshepard-FF;
            RMS2:=RMS2+dif*dif;
            dif:=flinear-FF;
            RMS3:=RMS3+dif*dif;
         end;
      end;
   end;
end;
RMS:=sqrt(RMS/NumPts);
RMS2:=sqrt(RMS2/Numpts);
RMS3:=sqrt(RMS3/Numpts);
LF1.Caption:=floattostr(RMS);
LF1s.Caption:=floattostr(RMS2);
LF1l.Caption:=floattostr(RMS3);

reinicializa_para_RMS(2);
NumPts:=0;
RMS:=0;
RMS2:=0;
RMS3:=0;
for i:=0 to N do begin
   x:=i/N;
```

```
   for j:=0 to N do begin
      y:=j/N;
      for k:=0 to N do begin
         z:=k/N;
         Todos_Interpolantes(x,y,z,fsplines,fshepard,flinear);
         if fsplines>=0 then begin
            NumPts:=NumPts+1;
            FF:=F2(x,y,z);
            dif:=fsplines-FF;
            RMS:=RMS+dif*dif;
            dif:=fshepard-FF;
            RMS2:=RMS2+dif*dif;
            dif:=flinear-FF;
            RMS3:=RMS3+dif*dif;
         end;
      end;
   end;
end;
RMS:=sqrt(RMS/NumPts);
RMS2:=sqrt(RMS2/Numpts);
RMS3:=sqrt(RMS3/Numpts);
LF2.Caption:=floattostr(RMS);
LF2s.Caption:=floattostr(RMS2);
LF2l.Caption:=floattostr(RMS3);

reinicializa_para_RMS(3);
NumPts:=0;
RMS:=0;
RMS2:=0;
RMS3:=0;
for i:=0 to N do begin
   x:=i/N;
   for j:=0 to N do begin
      y:=j/N;
      for k:=0 to N do begin
         z:=k/N;
         Todos_Interpolantes(x,y,z,fsplines,fshepard,flinear);
         if fsplines>=0 then begin
            NumPts:=NumPts+1;
            FF:=F3(x,y,z);
            dif:=fsplines-FF;
            RMS:=RMS+dif*dif;
            dif:=fshepard-FF;
```

```
            RMS2:=RMS2+dif*dif;
            dif:=flinear-FF;
            RMS3:=RMS3+dif*dif;
         end;
      end;
   end;
end;
RMS:=sqrt(RMS/NumPts);
RMS2:=sqrt(RMS2/Numpts);
RMS3:=sqrt(RMS3/Numpts);
LF3.Caption:=floattostr(RMS);
LF3s.Caption:=floattostr(RMS2);
LF3l.Caption:=floattostr(RMS3);

reinicializa_para_RMS(4);
NumPts:=0;
RMS:=0;
RMS2:=0;
RMS3:=0;
for i:=0 to N do begin
   x:=i/N;
   for j:=0 to N do begin
      y:=j/N;
      for k:=0 to N do begin
         z:=k/N;
         Todos_Interpolantes(x,y,z,fsplines,fshepard,flinear);
         if fsplines>=0 then begin
            NumPts:=NumPts+1;
            FF:=F4(x,y,z);
            dif:=fsplines-FF;
            RMS:=RMS+dif*dif;
            dif:=fshepard-FF;
            RMS2:=RMS2+dif*dif;
            dif:=flinear-FF;
            RMS3:=RMS3+dif*dif;
         end;
      end;
   end;
end;
RMS:=sqrt(RMS/NumPts);
RMS2:=sqrt(RMS2/Numpts);
RMS3:=sqrt(RMS3/Numpts);
LF4.Caption:=floattostr(RMS);
```

```
LF4s.Caption:=floattostr(RMS2);
LF4l.Caption:=floattostr(RMS3);

reinicializa_para_RMS(5);
NumPts:=0;
RMS:=0;
RMS2:=0;
RMS3:=0;
for i:=0 to N do begin
   x:=i/N;
   for j:=0 to N do begin
      y:=j/N;
      for k:=0 to N do begin
         z:=k/N;
         Todos_Interpolantes(x,y,z,fsplines,fshepard,flinear);
         if fsplines>=0 then begin
            NumPts:=NumPts+1;
            FF:=F5(x,y,z);
            dif:=fsplines-FF;
            RMS:=RMS+dif*dif;
            dif:=fshepard-FF;
            RMS2:=RMS2+dif*dif;
            dif:=flinear-FF;
            RMS3:=RMS3+dif*dif;
         end;
      end;
   end;
end;
RMS:=sqrt(RMS/NumPts);
RMS2:=sqrt(RMS2/Numpts);
RMS3:=sqrt(RMS3/Numpts);
LF5.Caption:=floattostr(RMS);
LF5s.Caption:=floattostr(RMS2);
LF5l.Caption:=floattostr(RMS3);

reinicializa_para_RMS(6);
NumPts:=0;
RMS:=0;
RMS2:=0;
RMS3:=0;
for i:=0 to N do begin
   x:=i/N;
   for j:=0 to N do begin
```

```
         y:=j/N;
         for k:=0 to N do begin
            z:=k/N;
            Todos_Interpolantes(x,y,z,fsplines,fshepard,flinear);
            if fsplines>=0 then begin
               NumPts:=NumPts+1;
               FF:=F6(x,y,z);
               dif:=fsplines-FF;
               RMS:=RMS+dif*dif;
               dif:=fshepard-FF;
               RMS2:=RMS2+dif*dif;
               dif:=flinear-FF;
               RMS3:=RMS3+dif*dif;
            end;
         end;
      end;
   end;
   RMS:=sqrt(RMS/NumPts);
   RMS2:=sqrt(RMS2/Numpts);
   RMS3:=sqrt(RMS3/Numpts);
   LF6.Caption:=floattostr(RMS);
   LF6s.Caption:=floattostr(RMS2);
   LF6l.Caption:=floattostr(RMS3);

   LMaxF.Caption:=floattostr(MaxF);
   LFuncCarregada.Caption:='F6';
   Button6.Enabled:=False;
   Button7.Enabled:=True;
   Button8.Enabled:=True;
   Button9.Enabled:=True;
   Button10.Enabled:=True;
   Button11.Enabled:=True;
   Button12.Enabled:=True;
   Button17.Enabled:=True;
end;

procedure TForm1.Button13Click(Sender: TObject); (*Desenha Shepard*)
begin
   desenha_shep;
end;

procedure TForm1.Button14Click(Sender: TObject);(*Desenha Int.Linear*)
begin
```

```
   desenha_lin;
end;


procedure TForm1.Button7Click(Sender: TObject);
begin
  reinicializa_para_RMS(1);
  desenhaFuncoes(1);
  LFuncCarregada.Caption:='F1';
end;

procedure TForm1.Button8Click(Sender: TObject);
begin
  reinicializa_para_RMS(2);
  desenhaFuncoes(2);
  LFuncCarregada.Caption:='F2';
end;


procedure TForm1.Button9Click(Sender: TObject);
begin
  reinicializa_para_RMS(3);
  desenhaFuncoes(3);
  LFuncCarregada.Caption:='F3';
end;



procedure TForm1.Button10Click(Sender: TObject);
begin
  reinicializa_para_RMS(4);
  desenhaFuncoes(4);
  LFuncCarregada.Caption:='F4';
end;



procedure TForm1.Button11Click(Sender: TObject);
begin
  reinicializa_para_RMS(5);
  desenhaFuncoes(5);
  LFuncCarregada.Caption:='F5';
end;
```

```
procedure TForm1.Button12Click(Sender: TObject);
begin
  reinicializa_para_RMS(6);
  desenhaFuncoes(6);
  LFuncCarregada.Caption:='F6';
end;

procedure TForm1.Button17Click(Sender: TObject);
begin
   if RadioGroup2.ItemIndex>2 then begin
      reinicializa_para_RMS(RadioGroup2.ItemIndex-2);
      LFuncCarregada.Caption:='F'+inttostr(RadioGroup2.ItemIndex-2);
   end;
   desenhaDiferenca(RadioGroup1.ItemIndex,RadioGroup2.ItemIndex);
end;

end.
```