INPE-6338-TDI/590

AVALIAÇÃO DO INTERPOLADOR AKIMA SOBRE MODELOS NUMÉRICOS DE ELEVAÇÃO

Ricardo Rodrigues Rangel

Dissertação de Mestrado em Computação Aplicada, orientada pelo Dr. Luiz Alberto Vieira Dias, aprovada em fevereiro de 1996.

INPE
São José dos Campos
Fevereiro de 1996

MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

INPE-6338-TDI/590

AVALIAÇÃO DO INTERPOLADOR AKIMA SOBRE MODELOS NUMÉRICOS DE ELEVAÇÃO

Ricardo Rodrigues Rangel

Dissertação de Mestrado em Computação Aplicada, orientada pelo Dr. Luiz Alberto Vieira Dias, aprovada em fevereiro de 1996.

INPE
São José dos Campos
Fevereiro de 1996

681.3.06:519.674

RANGEL, R.R.
Avaliação do interpolador Akima sobre modelos numéricos de elevação / R. R. Rangel. - São José dos Campos: INPE, 1996.
219 p. - (INPE-6338-TDI/590)

1. Interpolação. 2. Modelos numéricos de elevação. 3. Grades computacionais. 5. Altimetria. I. Título.

Aprovada pela Banca Examinadora em cumprimento a requisito exigido para a obtenção do Título de Mestre em Computação Aplicada

Dr. José Antonio Gonçalves Pereira
Presidente

Dr. Luiz Alberto Vieira Dias
Orientador

Dr. Haroldo Fraga de Campos Velho
Membro da Banca

Dr. Antonio Hugo Pereira Chaves
Membro da Banca
- Convidado -

Candidato: Ricardo Rodrigues Rangel

São José dos Campos, 14 de fevereiro de 1996

Dedico este trabalho a minha querida esposa Simone, pelo intenso apoio e carinho ao longo destes anos que caminhamos juntos, e a minha filha Jéssica, pela alegria que nos proporciona.

AGRADECIMENTOS

Ao Dr. Luiz Alberto Vieira Dias, pelos ensinamentos nas áreas de Computação Gráfica e Sensoriamento Remoto, pela orientação e pela amizade demonstrada durante os longos períodos dispendidos no desenvolvimento deste trabalho.

A minha esposa Simone, pela confiança e apoio nas horas difíceis.

A minha filha Jéssica, pelos momentos de alegria e diversão.

Aos meus pais, pela boa educação proporcionada durante a infância e a adolescência.

A todos aqueles que acreditaram e contribuíram para a conclusão deste trabalho.

PREFÁCIO

Cumpre ao autor esclarecer que todos os algoritmos apresentados neste trabalho foram desenvolvidos em Borland C++ 3.0 para ambiente Windows e que os resultados obtidos foram analisados estatisticamente com o auxílio do programa Microsoft Excel 5.0. Microsoft Windows e Microsoft Excel são marcas registradas da Microsoft Corporation, Borland C++ é marca registrada da Borland International Inc. e IBM é marca registrada da International Business Machines Corporation.

RESUMO

Este trabalho apresenta uma comparação estatística entre diversos métodos de interpolação tridimensional aplicados a Modelos Numéricos de Elevação. Para o estudo, foi considerado como área teste um terreno real localizado nas coordenadas 41° 25' N e 74° 59' W, perto de West Point e Hudson Valley, USA. Este terreno, representado por uma grade composta de 31x31 pontos com 69,81 metros de espaçamento na direção E-W e 92,51 metros na direção N-S, foi subamostrado, gerando uma nova grade de 11x11 pontos com 209,43 metros de espaçamento na direção E-W e 277,53 metros na direção N-S. A partir dos dados de altimetria da grade regular subamostrada, o terreno original foi reconstituído através das interpolações Linear, Lagrange, Bézier, B-Spline, β-Spline e Akima. As superfícies de erro geradas foram analisadas estatisticamente e os resultados desta análise apresentados graficamente e através de tabelas. Desta maneira, foi possível determinar que, nas condições do presente estudo, a Interpolação Akima Para Ajuste de Superfícies Suaves Baseado em Procedimentos Locais (Akima, 1974) foi o método que apresentou melhor desempenho.

AKIMA INTERPOLATOR VALUATION ON DIGITAL ELEVATION MODELS

ABSTRACT

This work presents a statistic comparison between several tridimensional interpolation methods applied to Digital Elevation Models. For the study, was considered as test area a real terrain localized at 41° 25' N and 74° 59' W, near West Point and Hudson Valley, USA. This terrain, represented by a 31x31 points grid with 229.04 ft E-W spacing and 303.51 ft N-S spacing, was subsampled, generating a new 11x11 points grid with 687.11 ft E-W spacing and 910.53 ft N-S spacing. From subsampled regular grid altimetry data, the original terrain was recreated through Linear, Lagrange, Bézier, B-Spline, β-Spline and Akima Interpolations. The obtained Error Surfaces was statistically analysed and the analysis results presented graphically and by tables. Thereafter, was possible to determine that, on actual study conditions, the Bivariate Interpolation and Smooth Surface Fitting Based on Local Procedures (Akima, 1974) was the method which demonstrated better performance.

SUMÁRIO

Pág.

LISTA DE FIGURAS xix

LISTA DE TABELAS xxiii

CAPÍTULO 1 - INTRODUÇÃO 1

CAPÍTULO 2 - INTERPOLADORES 3

2.1 - Introdução 3
2.2 - Interpolação Linear 3
2.3 - Interpolação Polinomial de Lagrange 8
2.4 - Interpolação Spline 13
2.5 - Interpolação Bézier 15
2.6 - Interpolação B-Spline 18
2.7 - Interpolação β-Spline 22
2.8 - Interpolação Akima 29

CAPÍTULO 3 - INTERPOLADOR AKIMA PARA AJUSTE DE CURVAS SUAVES BASEADO EM PROCEDIMENTOS LOCAIS . 33

3.1 - Introdução 33
3.2 - Determinação das derivadas nos pontos da curva 33
3.2.1 - Determinação das derivadas para funções de valores únicos 34
3.2.2 - Determinação das derivadas para funções de valores múltiplos 41
3.3 - Extrapolação de pontos para cálculo das derivadas 42
3.3.1 - Extrapolação de pontos para funções de valores únicos 43

3.3.2 - Extrapolação de pontos para funções de valores múltiplos 45
3.4 - Determinação da função fragmentada 46
3.4.1 - Determinação dos polinômios para funções de valores únicos 47
3.4.2 - Determinação dos polinômios para funções de valores múltiplos 48
3.4.2.1 - Determinação paramétrica conforme o Método de Akima 50
3.4.2.2 - Determinação paramétrica conforme a Curva de Hermite 51
3.5 - Análise dos resultados 52
3.5.1 - Resultados obtidos na Interpolação Akima para funções de valores únicos 52
3.5.2 - Resultados obtidos na Interpolação Akima para funções de valores múltiplos 53

CAPÍTULO 4 - INTERPOLADOR AKIMA PARA AJUSTE DE SUPERFÍCIES SUAVES BASEADO EM PROCEDIMENTOS LOCAIS 61

4.1 - Introdução 61
4.2 - Determinação das derivadas nos pontos da superfície 62
4.3 - Extrapolação de pontos para cálculo das derivadas 64
4.4 - Determinação da função fragmentada 68
4.5 - Análise dos resultados 72

CAPÍTULO 5 - AVALIAÇÃO DO INTERPOLADOR AKIMA SOBRE MODELOS NUMÉRICOS DE ELEVAÇÃO 75

5.1 - Introdução 75
5.2 - Interpolação sobre Modelo Numérico de Elevação 75
5.3 - Superfícies de erro 80

5.4 - Análise e avaliação 88
5.5 - Resultados da Análise 123

CAPÍTULO 6 - CONCLUSÃO 125

REFERÊNCIAS BIBLIOGRÁFICAS 127

APÊNDICE A - BIBLIOGRAFIA COMPLEMENTAR 129

APÊNDICE B - LISTAGEM DA INTERPOLAÇÃO LINEAR PARAMÉTRICA PARA CURVAS 131

APÊNDICE C - LISTAGEM DA INTERPOLAÇÃO LINEAR EXPLÍCITA PARA CURVAS 133

APÊNDICE D - LISTAGEM DA INTERPOLAÇÃO LINEAR EXPLÍCITA PARA SUPERFÍCIES 135

APÊNDICE E - LISTAGEM DA INTERPOLAÇÃO POLINOMIAL DE LAGRANGE PARA CURVAS 137

APÊNDICE F - LISTAGEM DA INTERPOLAÇÃO POLINOMIAL DE LAGRANGE PARA SUPERFÍCIES 141

APÊNDICE G - LISTAGEM DA INTERPOLAÇÃO SPLINE PARA CURVAS 145

APÊNDICE H - LISTAGEM DA INTERPOLAÇÃO BÉZIER PARA CURVAS 149

APÊNDICE I - LISTAGEM DA INTERPOLAÇÃO BÉZIER PARA SUPERFÍCIES 153

APÊNDICE J - LISTAGEM DA INTERPOLAÇÃO B-SPLINE PARA CURVAS 157

APÊNDICE K - LISTAGEM DA INTERPOLAÇÃO B-SPLINE PARA SUPERFÍCIES 161

APÊNDICE L - LISTAGEM DA INTERPOLAÇÃO β-SPLINE PARA CURVAS 165

APÊNDICE M - LISTAGEM DA INTERPOLAÇÃO β-SPLINE PARA SUPERFÍCIES 169

APÊNDICE N - LISTAGEM DA INTERPOLAÇÃO AKIMA PARA AJUSTE DE CURVAS SUAVES SOBRE FUNÇÕES DE VALORES ÚNICOS 173

APÊNDICE O - LISTAGEM DA INTERPOLAÇÃO AKIMA PARA AJUSTE DE CURVAS SUAVES SOBRE FUNÇÕES DE VALORES MÚLTIPLOS 177

APÊNDICE P - LISTAGEM DA INTERPOLAÇÃO AKIMA PARA AJUSTE DE CURVAS SUAVES SOBRE FUNÇÕES DE VALORES MÚLTIPLOS PARAMETRICAMENTE CONFORME A CURVA DE HERMITE E COM MÉTODO PROPOSTO PARA CÁLCULO DOS MÓDULOS DAS TANGENTES 181

APÊNDICE Q - LISTAGEM DA INTERPOLAÇÃO AKIMA PARA AJUSTE DE SUPERFÍCIES SUAVES BASEADA EM PROCEDIMENTOS LOCAIS 187

LISTA DE FIGURAS

Pág.

2.1 - Interpolação Linear sobre curvas..... 4
2.2 - Retalho de uma grade retangular..... 5
2.3 - Interpolação Linear sobre superfícies..... 6
2.4 - Interpolação Linear com diferentes amostragens. 7
2.5 - Interpolação Polinomial de Lagrange sobre curvas..... 9
2.6 - Interpolação Polinomial de Lagrange sobre superfícies..... 10
2.7 - Interpolação Polinomial de Lagrange com diferentes quantidades de pontos amostrados de uma uma mesma equação..... 11
2.8 - Interpolação Polinomial de Lagrange com diferentes quantidades de pontos não amostrados de uma mesma equação 12
2.9 - Interpolação Spline sobre curvas..... 15
2.10 - Interpolação Bézier sobre curvas 16
2.11 - Interpolação Bézier sobre superfícies 18
2.12 - Interpolação B-Spline sobre curvas 20
2.13 - Interpolação B-Spline sobre superfícies 22
2.14 - Interpolação β-Spline sobre curvas 24
2.15 - Interpolação β-Spline com diferentes valores de β_1..... 25
2.16 - Interpolação β-Spline com diferentes valores de β_2..... 26
2.17 - Interpolação β-Spline sobre superfícies 29
2.18 - Interpolação Akima sobre curvas 30
2.19 - Interpolação Akima sobre superfícies 31
3.1 - Coordenadas dos pontos para cálculo da derivada 34
3.2 - Condição para determinação da derivada da curva 35
3.3 - Coordenadas dos pontos para extrapolação na extremidade da direita 43

3.4 - Coordenadas dos pontos para extrapolação na extremidade da esquerda 45
3.5 - Comparação entre alguns métodos de interpolação 53
3.6 - Curva resultante de uma Interpolação Akima para funções de valores múltiplos parametrizada conforme o Método de Akima 54
3.7 - Comparação entre os métodos de interpolação para funções de valores únicos e múltiplos 54
3.8 - Família de curvas obtidas variando-se os valores dos módulos dos vetores das tangentes 56
3.9 - Curva resultante de uma Interpolação Akima parametrizada conforme a Curva de Hermite e com o método proposto para determinação dos módulos das tangentes 57
3.10 - Comparação entre os três métodos de Interpolação Akima para Ajuste de Curvas Suaves 58
4.1 - Célula de uma grade retangular 62
4.2 - Pontos necessários para o cálculo das derivadas parciais 63
4.3 - Células de uma grade regular 65
4.4 - Grade regular extrapolada em x e em z 67
4.5 - Grade regular totalmente extrapolada 67
4.6 - Célula de uma grade regular 68
4.7 - Grade regular escolhida arbitrariamente 73
4.8 - Comparação entre alguns métodos de interpolação sobre superfícies 74
5.1 - Grade regular original 76
5.2 - Grade regular subamostrada 76
5.3 - Grades refinadas obtidas com alguns métodos de interpolação sobre superfícies 80
5.4 - Superfícies de erro entre as grades refinadas por interpolação e a superfície real 88
5.5 - Polígono de Freqüência de Erros da Interpolação Linear 97

5.6 - Polígono de Freqüência de Erros da Interpolação Polinomial de Lagrange 98
5.7 - Polígono de Freqüência de Erros da Interpolação Bézier 99
5.8 - Polígono de Freqüência de Erros da Interpolação B-Spline 100
5.9 - Polígono de Freqüência de Erros da Interpolação β-Spline 101
5.10 - Polígono de Freqüência de Erros da Interpolação Akima 102
5.11 - Polígonos de Freqüência de Erros das Interpolações 103
5.12 - Valores Máximo e Mínimo dos Erros das Interpolações 109
5.13 - Valor Máximo dos Erros Absolutos das Interpolações 110
5.14 - Amplitude Total dos Erros das Interpolações .. 111
5.15 - Mediana dos Erros das Interpolações 112
5.16 - Moda dos Erros das Interpolações 113
5.17 - Média dos Erros das Interpolações 114
5.18 - Desvio Médio dos Erros das Interpolações 115
5.19 - Desvio Padrão dos Erros das Interpolações 116
5.20 - Variância dos Erros das Interpolações 117
5.21 - Tempo Gasto no Processamento das Interpolações 118
5.22 - Memória Temporária Utilizada no Processamento das Interpolações 119
5.23 - Número de Linhas das Implementações das Interpolações 120
5.24 - Ordem de Continuidade das Interpolações 121

LISTA DE TABELAS

Pág.

5.1 - Valores Altimétricos da Grade Regular Original (metros) 77
5.2 - Valores Altimétricos da Grade Regular Subamostrada (metros) 78
5.3 - Valores Altimétricos da Grade Refinada por Interpolação Linear (metros) 81
5.4 - Valores Altimétricos da Grade Refinada por Interpolação Lagrange (metros) 82
5.5 - Valores Altimétricos da Grade Refinada por Interpolação Bézier (metros) 83
5.6 - Valores Altimétricos da Grade Refinada por Interpolação B-Spline (metros) 84
5.7 - Valores Altimétricos da Grade Refinada por Interpolação β-Spline (metros) 85
5.8 - Valores Altimétricos da Grade Refinada por Interpolação Akima (metros) 86
5.9 - Valores da Superfície de Erro Obtida com a Interpolação Linear (metros) 89
5.10 - Valores da Superfície de Erro Obtida com a Interpolação Lagrange (metros) 90
5.11 - Valores da Superfície de Erro Obtida com a Interpolação Bézier (metros) 91
5.12 - Valores da Superfície de Erro Obtida com a Interpolação B-Spline (metros) 92
5.13 - Valores da Superfície de Erro Obtida com a Interpolação β-Spline (metros) 93
5.14 - Valores da Superfície de Erro Obtida com a Interpolação Akima (metros) 94
5.15 - Distribuição de Freqüência de Erros das Interpolações de -∞ até 0,00 95

5.16 - Distribuição de Freqüência de Erros das Interpolações de 0,01 até +∞ 96
5.17 - Comparação Entre as Interpolações 122

CAPÍTULO 1

INTRODUÇÃO

As técnicas utilizadas em Computação Gráfica se baseiam freqüentemente em cálculos numéricos aplicados à computação. Esta matemática aplicada é essencial na geração de desenhos, na representação e no processamento de imagens.

A geração de desenhos tridimensionais em computação envolve a utilização de retas, curvas e superfícies. Normalmente, são utilizados métodos de interpolação para representar curvas e superfícies que não possam ser descritas por fórmulas matemáticas.

As superfícies freqüentemente representam dados de altimetria de um terreno. Estes dados são obtidos através de amostras e armazenados em arquivos como grades, regulares ou irregulares. Através de interpolação, estas grades podem ser refinadas de modo a se obter informações de altimetria em qualquer posição dentro da grade. A precisão destas informações depende da qualidade das amostras e do método de interpolação utilizado.

Dentre os métodos de interpolação existentes, um dos que apresentam características apropriadas à interpolação de superfícies é o proposto por Hiroshi Akima em 1974. Dessa forma, o objetivo deste trabalho é avaliar os resultados obtidos com a Interpolação Akima sobre Modelos Numéricos de Elevação em relação aos obtidos por outros métodos e aos dados reais de altitude do terreno.

Para alcançar o objetivo proposto, o trabalho está dividido em seis capítulos.

No Capítulo 2, será apresentada uma revisão bibliográfica sobre Interpolação Linear, Polinomial de Lagrange, Spline, Bézier, B-Spline, β-Spline e Akima.

O método de Interpolação Akima para Ajuste de Curvas Suaves Baseado em Procedimentos Locais será detalhado no Capítulo 3.

O método de Interpolação Akima para Ajuste de Superfícies Suaves Baseado em Procedimentos Locais será tratado no Capítulo 4.

No Capítulo 5, os resultados obtidos com a Interpolação Akima sobre Modelos Numéricos de Elevação serão avaliados em relação aos dados reais de altimetria do terreno e comparados com os resultados obtidos através das Interpolações Linear, Polinomial de Lagrange, Spline, Bézier, B-Spline e β-Spline.

No Capítulo 6, serão analisados os resultados obtidos e apresentada a conclusão do trabalho.

CAPÍTULO 2

INTERPOLADORES

2.1 - INTRODUÇÃO

Interpolação é o processo utilizado para se determinar valores intermediários reconstituídos a partir de pontos amostrados. Na interpolação de curvas, obtém-se coordenadas de pontos intermediários dos segmentos e, na interpolação de superfícies, reconstitui-se valores internos de cada retalho da grade.

As interpolações são necessárias porque não existem fórmulas matemáticas exatas que possam descrever qualquer tipo de curva ou superfície. Alguns métodos de interpolação serão apresentados superficialmente neste capítulo, pois assume-se que sejam de conhecimento geral.

Os diferentes métodos de interpolação possuem características e aplicações diferentes. Dessa forma, os interpoladores serão analisados de modo a definir a aplicabilidade de cada método no refinamento de grades regulares.

2.2 - INTERPOLAÇÃO LINEAR

Na Interpolação Linear, o resultado depende exclusivamente das coordenadas dos pontos que definem cada intervalo. Estas coordenadas são utilizadas numa média ponderada para a interpolação dos pontos intermediários.

Os pesos dos pontos na média ponderada são obtidos através de um parâmetro t que se altera durante todo o intervalo, assumindo o valor 0 no ponto inicial e 1

no ponto final. Dessa forma, a curva resultante é representada por funções explícitas em função de t, onde qualquer ponto do intervalo pode ser obtido através das seguintes equações paramétricas:

$$x = x(t) = (1-t).x_1 + (t).x_2, \tag{2.1}$$

$$y = y(t) = (1-t).y_1 + (t).y_2, \tag{2.2}$$

$$z = z(t) = (1-t).z_1 + (t).z_2, \tag{2.3}$$

onde t = (0,...,1) e o intervalo é definido pelos pontos (x_1, y_1, z_1) e (x_2, y_2, z_2). A Figura 2.1 apresenta um conjunto aleatório de pontos interpolado linearmente.

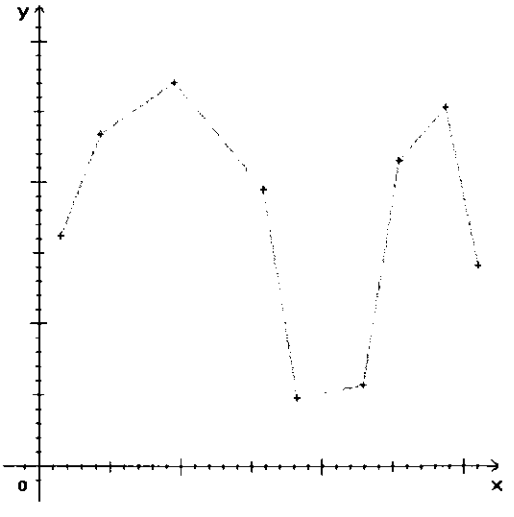


Fig. 2.1 - Interpolação Linear sobre curvas.

Na interpolação de superfícies, os pontos internos de cada retalho são obtidos através das coordenadas dos quatro pontos que o definem. A Figura 2.2 apresenta um retalho de uma grade regular.

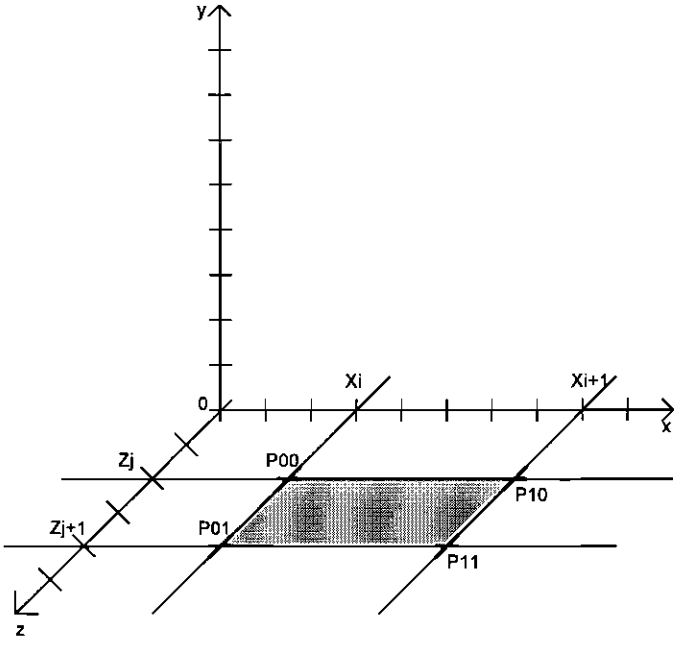


Fig. 2.2 - Retalho de uma grade retangular.

Dessa forma, um retalho definido pelos pontos P_{00}, P_{10}, P_{01} e P_{11} pode ser interpolado explicitamente através da seguinte equação:

$$y(x,z) = a + b.dx + c.dz + d.dx.dz, \tag{2.4}$$

onde:

$$dx = x - x_{0,0}, \tag{2.5}$$

$$dz = z - z_{0,0}, \tag{2.6}$$

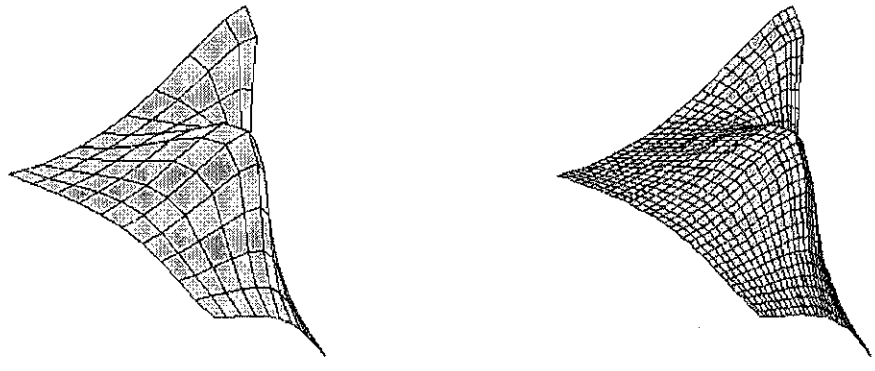
$$a = y_{0,0}, \tag{2.7}$$

$$b = \frac{y_{1,0} - a}{x_{1,1} - x_{0,0}}, \tag{2.8}$$

$$c = \frac{y_{0,1} - a}{z_{1,1} - z_{0,0}}, \tag{2.9}$$

$$d = \frac{y_{1,1} - a - b(x_{1,1} - x_{0,0}) - c(z_{1,1} - z_{0,0})}{(x_{1,1} - x_{0,0}).(z_{1,1} - z_{0,0})}. \tag{2.10}$$

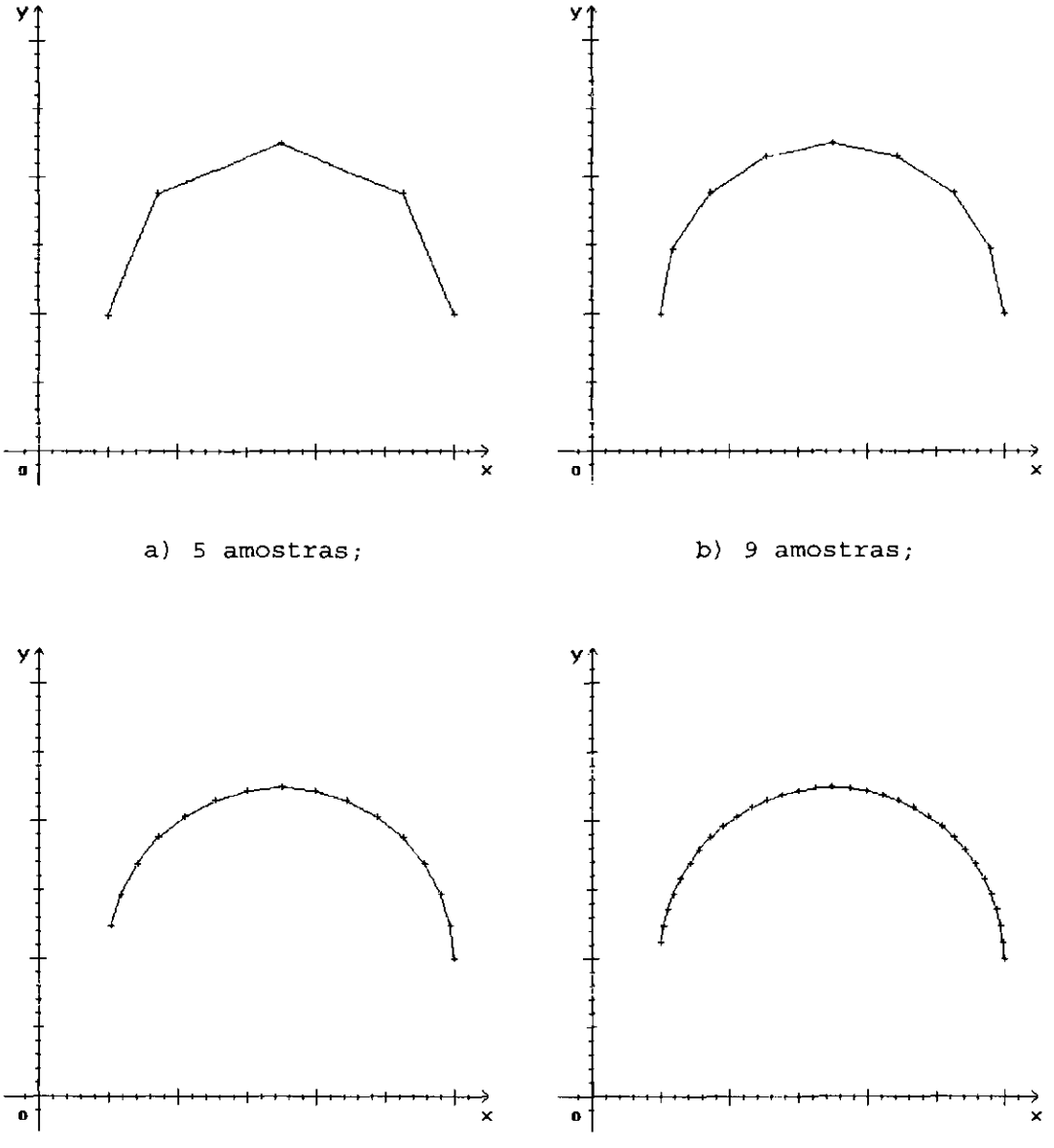
A Figura 2.3 apresenta uma grade refinada obtida através da Interpolação Linear sobre uma grade regular aleatória.



a) Grade regular aleatória; b) Grade regular refinada.

Fig. 2.3 - Interpolação Linear sobre superfícies.

A qualidade dos resultados obtidos com a Interpolação Linear depende diretamente da quantidade de pontos amostrados: quanto mais amostras forem obtidas, mais próximos estarão os resultados de uma superfície real. A Figura 2.4 apresenta uma comparação entre os resultados alcançados com diferentes amostragens.



a) 5 amostras;

b) 9 amostras;

c) 16 amostras;

d) 33 amostras.

Fig. 2.4 - Interpolação Linear com diferentes amostragens.

De uma forma geral, quanto maior for a amostragem, menor é a necessidade de interpolação. Nestas condições, os resultados praticamente independem do método utilizado. Por outro lado, como a capacidade de armazenamento de dados normalmente é limitada, as grades regulares originais não são tão finas a ponto de não

permitirem o aparecimento, nos resultados de uma Interpolação Linear, de "bicos" e "achatamentos" que dificilmente seriam encontrados num terreno real.

As implementações da Interpolação Linear Paramétrica para Curvas, da Interpolação Linear Explícita para Curvas e da Interpolação Linear Explícita para Superfícies estão listadas nos Apêndices B, C e D, respectivamente.

2.3 - INTERPOLAÇÃO POLINOMIAL DE LAGRANGE

O Teorema Fundamental da Álgebra, enunciado por D'Allambert (1746) e demonstrado por Gauss (1799), diz: "Toda equação polinomial do grau n possui exatamente n raízes". Uma equação polinomial de grau n pode ser representada por

$$f(x) = a_0 + a_1.x + a_2.x^2 + \cdots + a_n.x^n, \qquad (2.11)$$

onde a equação possui n+1 coeficientes.

A Interpolação Polinomial de Lagrange é na realidade a determinação do único polinômio p(x) de grau n-1 no máximo que passa exatamente pelos n pontos a serem interpolados. Utilizando a fórmula de Lagrange (Hildebrand, 1956), tem-se que o polinômio p(x) de grau n no máximo é obtido por

$$p(x) = \sum_{i=0}^{n} \left[\left(\prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{(x - x_j)}{(x_i - x_j)} \right) . y_i \right], \qquad (2.12)$$

onde x_i e y_i são os valores das coordenadas na posição i.

A Figura 2.5 apresenta um conjunto aleatório de pontos interpolado polinomialmente por Lagrange.

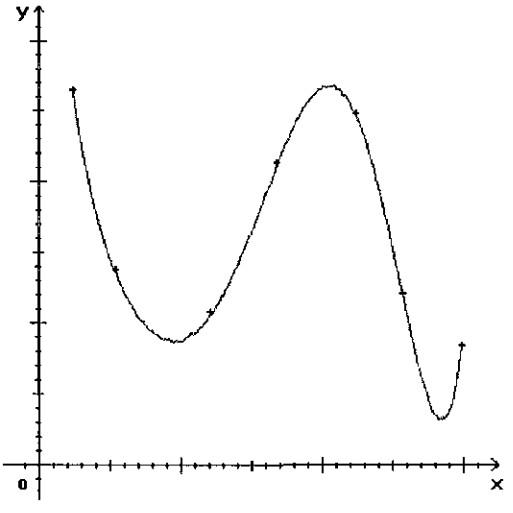


Fig. 2.5 - Interpolação Polinomial de Lagrange sobre curvas.

A extensão da Interpolação Polinomial de Lagrange para a interpolação de superfícies permite que seja determinado um único polinômio $x^{\alpha}.z^{\beta}$, onde $\alpha = (0,...,nx-1)$ e $\beta = (0,...,nz-1)$. O polinômio $p(x,z)$ passa exatamente pelos nx.nz pontos da grade a ser interpolada, onde nx representa o número de pontos na direção do eixo x e nz o número de pontos na direção do eixo z. Estendendo a fórmula de Lagrange (Hildebrand, 1956), tem-se que o polinômio $p(x,z)$ é obtido por

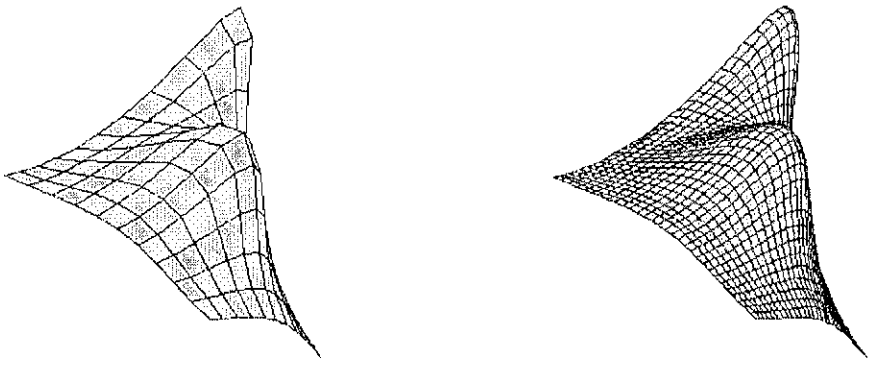
$$p(x,z) = \sum_{i=0}^{nx} \left\{ \sum_{j=0}^{nz} \left[\left(\prod_{\substack{l=0 \\ l \neq j}}^{nz} \frac{(z - z_l)}{(z_j - z_l)} \right) . y_{i,j} \right] \right\} . B , \qquad (2.13)$$

onde

$$B = \prod_{\substack{k=0 \\ k \neq i}}^{nx} \frac{(x - x_k)}{(x_i - x_k)}, \qquad (2.14)$$

e x_i, $y_{i,j}$ e z_j são os valores das coordenadas na posição i,j.

A Figura 2.6 apresenta uma grade refinada obtida através da Interpolação Polinomial de Lagrange sobre uma grade regular aleatória.

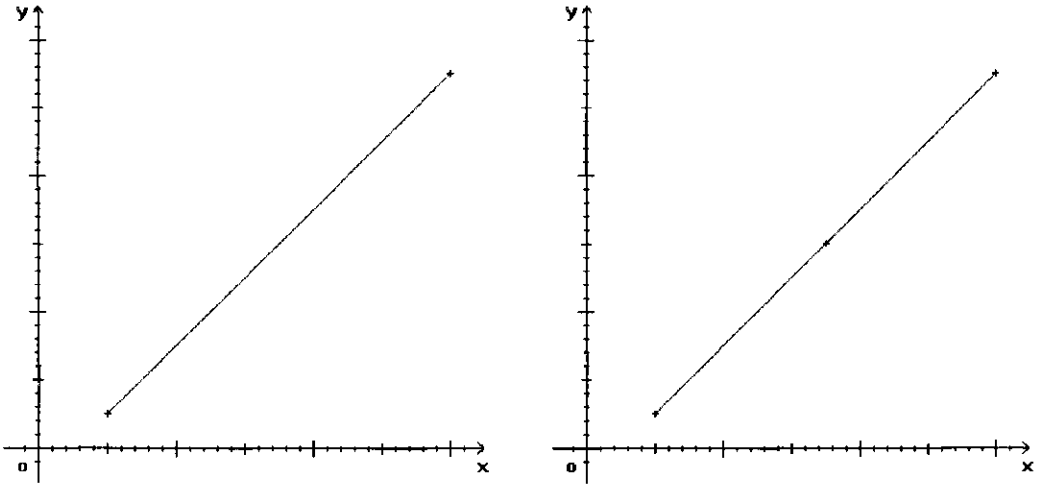


a) Grade regular aleatória; b) Grade regular refinada.

Fig. 2.6 - Interpolação Polinomial de Lagrange sobre superfícies.

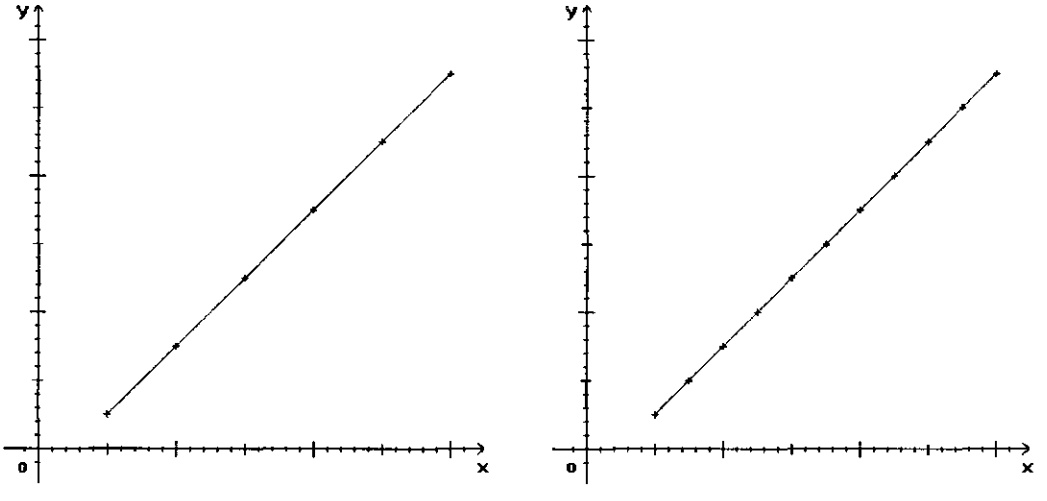
O tempo gasto no processamento da Interpolação Polinomial de Lagrange depende do número de pontos a serem interpolados. Para um grande número de pontos, além da interpolação ficar mais lenta, a curva obtida tende a ficar oscilante. Não se pode afirmar que a curva será mais oscilante com o aumento do número de pontos pois estes pontos podem ser amostras de uma mesma equação. A Figura 2.7 apresenta uma comparação entre os resultados

alcançados com diferentes quantidades de pontos amostrados de uma mesma equação.



a) 2 pontos;

b) 3 pontos;



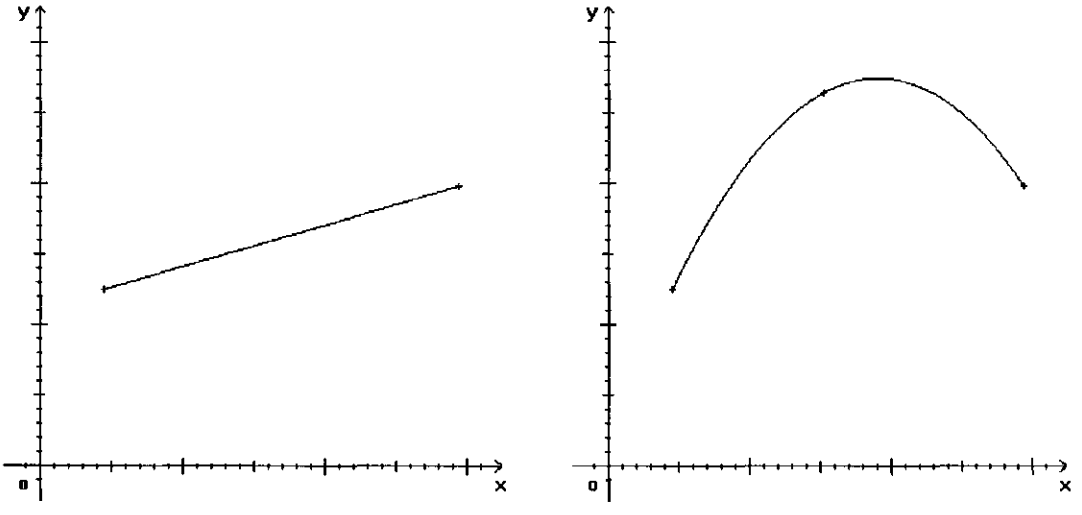
c) 6 pontos;

d) 11 pontos.

Fig. 2.7 - Interpolação Polinomial de Lagrange com diferentes quantidades de pontos amostrados de uma mesma equação.

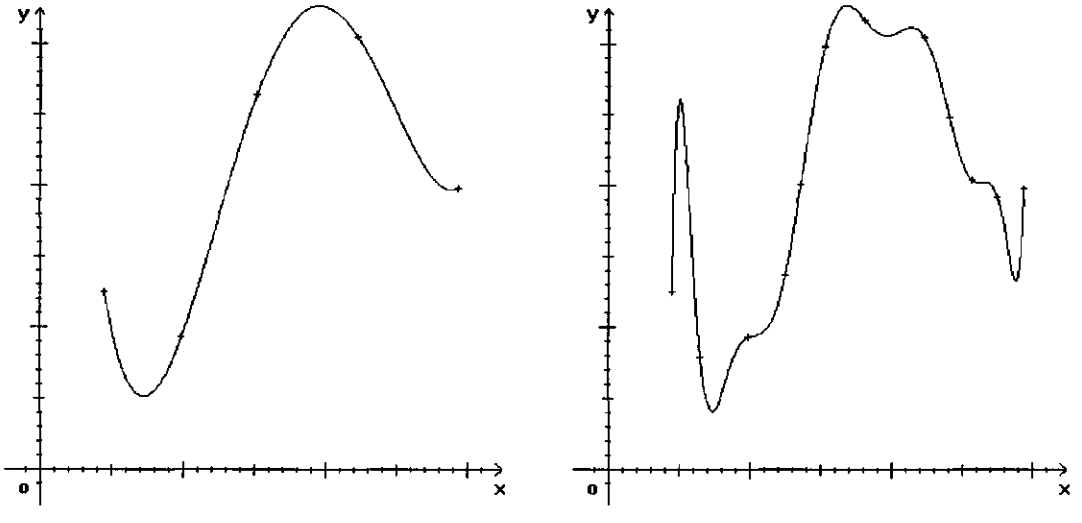
No caso do aumento do número de pontos não amostrados de uma mesma equação, a curva obtida com a

Interpolação Polinomial de Lagrange fica mais oscilante pois o grau da equação aumenta. A Figura 2.8 apresenta uma comparação entre os resultados alcançados com diferentes quantidades de pontos não amostrados de uma mesma equação.



a) 2 pontos;

b) 3 pontos;



c) 5 pontos;

d) 12 pontos.

Fig. 2.8 - Interpolação Polinomial de Lagrange com diferentes quantidades de pontos não amostrados de uma mesma equação.

Portanto, como os dados de altimetria do terreno não são amostras de uma equação matemática exata, a oscilação excessiva praticamente inviabiliza a Interpolação Polinomial de Lagrange no refinamento de grades regulares, mesmo que haja continuidade de curva nos pontos intermediários.

As implementações da Interpolação Polinomial de Lagrange para Curvas e da Interpolação Polinomial de Lagrange para Superfícies estão listadas nos Apêndices E e F, respectivamente.

2.4 - INTERPOLAÇÃO SPLINE

Na Interpolação Spline é definido um polinômio do terceiro grau para cada intervalo de pontos consecutivos. Cada polinômio é calculado com o uso de quatro pontos, de modo que estes quatro pontos delimitem três subintervalos. A expressão que representa um polinômio do terceiro grau em cada subintervalo i é

$$P_i(x) = a_i.x^3 + b_i.x^2 + c_i.x + d_i . \qquad (2.15)$$

Sendo (x_1,y_1), (x_2,y_2), (x_3,y_3) e (x_4,y_4) quatro pontos consecutivos a serem interpolados, as 3 equações a serem determinadas são:

$$P_1(x) = a_1.x^3 + b_1.x^2 + c_1.x + d_1 , \qquad (2.16)$$

$$P_2(x) = a_2.x^3 + b_2.x^2 + c_2.x + d_2 , \qquad (2.17)$$

$$P_3(x) = a_3.x^3 + b_3.x^2 + c_3.x + d_3 . \qquad (2.18)$$

As condições matemáticas para a determinação dos 12 coeficientes são as seguintes:

$$P_1(x_1) = y_1, \tag{2.19}$$

$$P_1(x_2) = y_2, \tag{2.20}$$

$$P_2(x_2) = y_2, \tag{2.21}$$

$$P_2(x_3) = y_3, \tag{2.22}$$

$$P_3(x_3) = y_3, \tag{2.23}$$

$$P_3(x_4) = y_4, \tag{2.24}$$

$$P_1'(x_2) = P_2'(x_2), \tag{2.25}$$

$$P_2'(x_3) = P_3'(x_3), \tag{2.26}$$

$$P_1''(x_2) = P_2''(x_2), \tag{2.27}$$

$$P_2''(x_3) = P_3''(x_3), \tag{2.28}$$

$$P_1''(x_1) = 0, \tag{2.29}$$

$$P_3''(x_4) = 0. \tag{2.30}$$

O sistema de equações, composto por 12 equações e 12 incógnitas, pode ser resolvido por diferentes métodos de solução de sistemas.

A Figura 2.9 apresenta um conjunto aleatório de pontos interpolado pelo método Spline e a implementação da Interpolação Spline para Curvas está listada no Apêndice G.

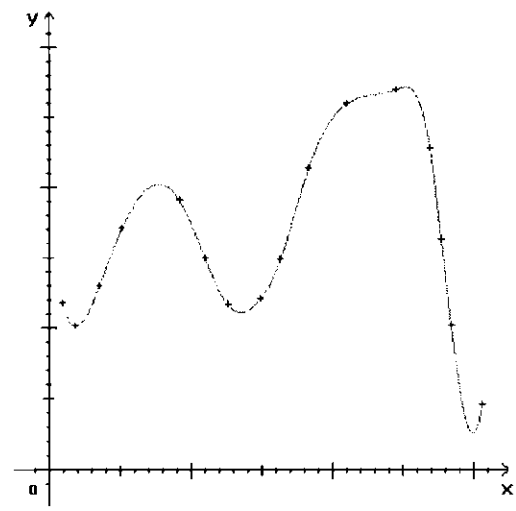


Fig. 2.9 - Interpolação Spline sobre curvas.

A curva resultante de uma Interpolação Spline passa pelos pontos e possui continuidade de segunda ordem nos pontos intermediários. É possível estender este método para a interpolação de superfícies, mas o volume de cálculos aumenta muito e o torna não muito popular. O excesso de cálculos para três dimensões torna o método praticamente inviável para o refinamento de grades.

2.5 - INTERPOLAÇÃO BÉZIER

A curva resultante de uma Interpolação Bézier passa exatamente pelos pontos extremos da curva, mas não passa pelos intermediários. Os pontos intermediários servem como pesos e influenciam o resultado global da interpolação.

Segundo Foley et. al (1992), a Curva de Bézier é obtida a partir da Curva de Hermite e pode ser expressa parametricamente por

$$x(t) = \sum_{i=0}^{n} \left(\frac{n!}{(n-i)!i!} . (1-t)^{n-i} . t^{i} . x_i \right) , \qquad (2.31)$$

$$y(t) = \sum_{i=0}^{n} \left(\frac{n!}{(n-i)!i!} . (1-t)^{n-i} . t^{i} . y_i \right) , \qquad (2.32)$$

$$z(t) = \sum_{i=0}^{n} \left(\frac{n!}{(n-i)!i!} . (1-t)^{n-i} . t^{i} . z_i \right) , \qquad (2.33)$$

onde n é igual ao número de pontos menos 1, t é um parâmetro que varia de 0 até 1, e x_i, y_i e z_i são os valores das coordenadas na posição i.

A Figura 2.10 apresenta um conjunto aleatório de pontos interpolado pelo método Bézier.

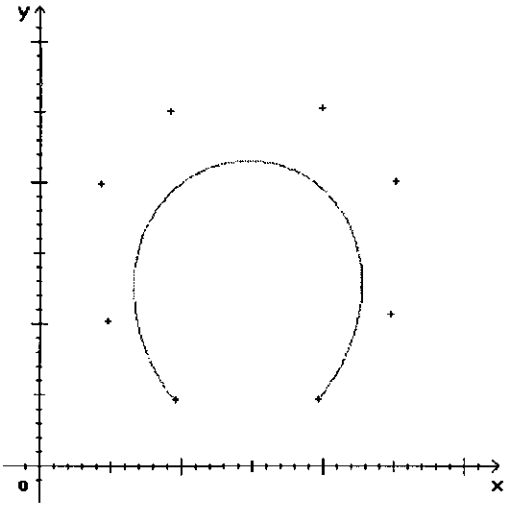


Fig. 2.10 - Interpolação Bézier sobre curvas.

A Superfície de Bézier Q(s,t) é uma extensão natural da Curva de Bézier e pode ser definida por

$$Q(s,t) = \sum_{i=0}^{m} \left[\sum_{j=0}^{n} P_{i,j} . B_{j,n}(t) \right] . B_{i,m}(s) , \qquad (2.34)$$

onde

$$Q(s,t) = [x(s,t) \quad y(s,t) \quad z(s,t)] \,, \tag{2.35}$$

$$P_{i,j} = [x_{i,j} \quad y_{i,j} \quad z_{i,j}] \,, \tag{2.36}$$

$$B_{j,n}(t) = C_{j,n}.(1-t)^{n-j}.t^{j}, \tag{2.37}$$

$$B_{i,m}(s) = C_{i,m}.(1-s)^{m-i}.s^{i}, \tag{2.38}$$

s é um parâmetro que varia de 0 até 1 na direção do eixo x, t é um parâmetro que varia de 0 até 1 na direção do eixo z, m é o número de pontos da grade em x menos 1, n é o número de pontos da grade em z menos 1,

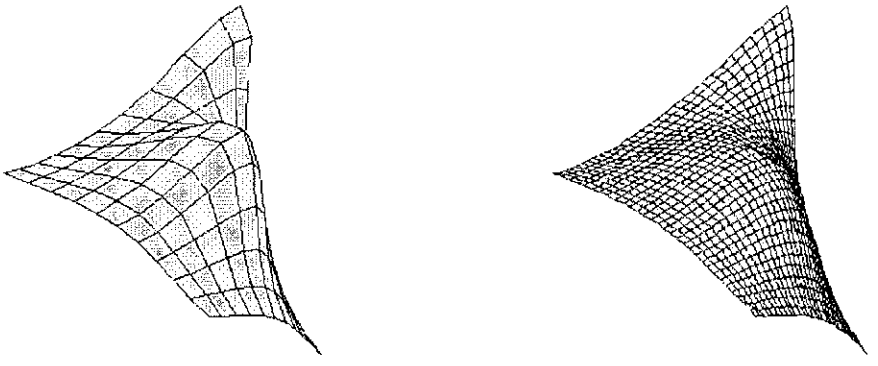
$$C_{j,n} = \frac{n!}{(n-j)!\,j!}, \tag{2.39}$$

$$C_{i,m} = \frac{m!}{(m-i)!\,i!}, \tag{2.40}$$

e $x_{i,j}$, $y_{i,j}$ e $z_{i,j}$ são os valores das coordenadas na posição i,j.

A Figura 2.11 apresenta uma grade refinada obtida através da Interpolação Bézier sobre uma grade regular aleatória.

O método apresenta um custo computacional elevado pois todos os pontos do conjunto influenciam no resultado global da interpolação. Entretanto, as curvas e superfícies resultantes não oscilam e possuem alto grau de continuidade nos pontos intermediários.



a) Grade regular aleatória; b) Grade regular refinada.

Fig. 2.11 - Interpolação Bézier sobre superfícies.

A Interpolação Bézier não é aplicável no refinamento de grades porque o resultado da interpolação não passa pelos pontos intermediários, que são valores corretos amostrados do solo.

As implementações da Interpolação Bézier para Curvas e da Interpolação Bézier para Superfícies estão listadas nos Apêndices H e I, respectivamente.

2.6 - INTERPOLAÇÃO B-SPLINE

A Interpolação Spline Natural permite que sejam determinados polinômios cúbicos que passam exatamente pelos pontos de controle, entretanto, este método possui um custo computacional muito elevado.

A Interpolação B-Spline consiste de segmentos de curvas cujos coeficientes polinomiais dependem exclusivamente de alguns pontos de controle. Dessa forma, o

tempo gasto para determinar os coeficientes é extremamente reduzido.

Na Interpolação B-Spline, assim como na Spline Natural, o controle é local e a continuidade da curva resultante é de segunda ordem, entretanto, os pontos de controle não são interpolados pois servem apenas como pesos para a determinação da curva.

Cada segmento de curva é definido por um polinômio cúbico calculado através do uso de quatro pontos, de modo que estes quatro pontos delimitem três subintervalos. A expressão que representa o polinômio do terceiro grau em cada intervalo i é obtida por

$$Q(u) = b_{-3}(u).P_{i-3} + b_{-2}(u).P_{i-2} + b_{-1}(u).P_{i-1} + b_0(u).P_i\,, \tag{2.41}$$

onde

$$Q(u) = [x(u) \quad y(u) \quad z(u)]\,, \tag{2.42}$$

$$P_i = [x_i \quad y_i \quad z_i]\,, \tag{2.43}$$

$$b_{-3}(u) = \frac{-u^3 + 3u^2 - 3u + 1}{6}\,, \tag{2.44}$$

$$b_{-2}(u) = \frac{3u^3 - 6u^2 + 4}{6}\,, \tag{2.45}$$

$$b_{-1}(u) = \frac{-3u^3 + 3u^2 + 3u + 1}{6}\,, \tag{2.46}$$

$$b_0(u) = \frac{u^3}{6}\,, \tag{2.47}$$

u varia de 0 até 1 para i = (3,...,n-1) e n é o número de pontos do conjunto. A Figura 2.12 apresenta um conjunto aleatório de pontos interpolado por B-Spline.

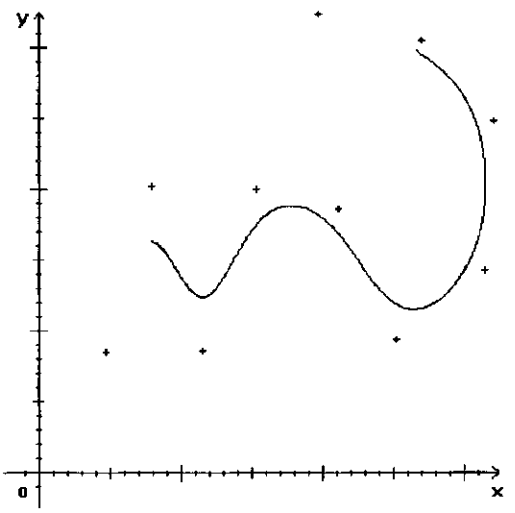


Fig. 2.12 - Interpolação B-Spline sobre curvas.

Ao contrário da Spline Natural, é possível estender este método para a interpolação de superfícies, sem que o volume dos cálculos inviabilize a sua utilização. A extensão da Interpolação B-Spline permite que seja determinado um único polinômio $x^{\alpha}.z^{\beta}$ para cada retalho da grade a ser interpolada, onde α = (0,1,2,3) e β = (0,1,2,3). A expressão que representa o polinômio em cada retalho (i,j) é obtida por

$$Q(s,t) = \sum_{m=-3}^{0}\left[\sum_{n=-3}^{0} P_{i+m,j+n}.b_m(s)\right].c_n(t) \ , \qquad (2.48)$$

onde

$$Q(s,t) = [x(s,t) \quad y(s,t) \quad z(s,t)] \ , \qquad (2.49)$$

$$P_{i,j} = [x_{i,j} \quad y_{i,j} \quad z_{i,j}] \ , \qquad (2.50)$$

$$b_{-3}(s) = \frac{-s^3 + 3s^2 - 3s + 1}{6}, \tag{2.51}$$

$$b_{-2}(s) = \frac{3s^3 - 6s^2 + 4}{6}, \tag{2.52}$$

$$b_{-1}(s) = \frac{-3s^3 + 3s^2 + 3s + 1}{6}, \tag{2.53}$$

$$b_0(s) = \frac{s^3}{6}, \tag{2.54}$$

$$c_{-3}(t) = \frac{-t^3 + 3t^2 - 3t + 1}{6}, \tag{2.55}$$

$$c_{-2}(t) = \frac{3t^3 - 6t^2 + 4}{6}, \tag{2.56}$$

$$c_{-1}(t) = \frac{-3t^3 + 3t^2 + 3t + 1}{6}, \tag{2.57}$$

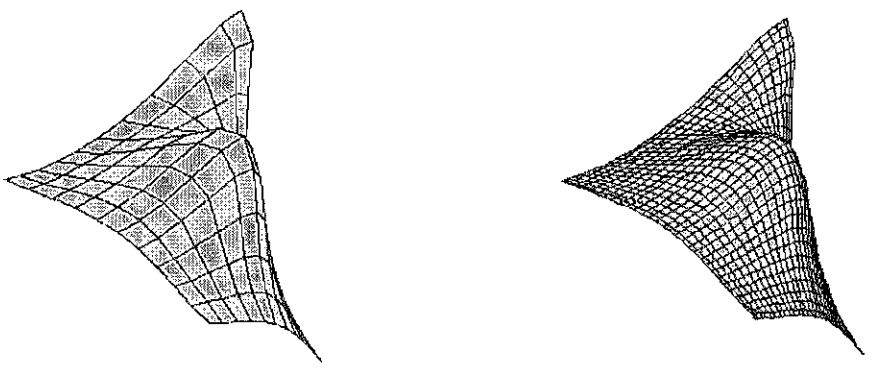
$$c_0(t) = \frac{t^3}{6}, \tag{2.58}$$

s varia de 0 até 1 para i = (3,...,nx-1), t varia de 0 até 1 para j = (3,...,nz-1), nx é o número de pontos na direção do eixo x e nz é o número de pontos na direção do eixo z.

A Figura 2.13 apresenta uma grade refinada obtida através da Interpolação B-Spline sobre uma grade regular aleatória.

Embora a superfície resultante de uma Interpolação B-Spline não passe pelos pontos da grade regular original, o método pode ser aplicado no refinamento de grades desde que haja uma grande interação entre o operador e o programa que efetuará a interpolação. Dessa

forma, com algumas alterações nas coordenadas da grade regular grossa, pode-se obter uma grade refinada capaz de representar satisfatoriamente o terreno sem que haja oscilações excessivas.



a) Grade regular aleatória; b) Grade regular refinada.

Fig. 2.13 - Interpolação B-Spline sobre superfícies.

As implementações da Interpolação B-Spline para Curvas e da Interpolação B-Spline para Superfícies estão listadas nos Apêndices J e K, respectivamente.

2.7 - INTERPOLAÇÃO β-SPLINE

A Interpolação β-Spline é uma generalização da Interpolação B-Spline, onde dois novos parâmetros de controle são adicionados, β_1 e β_2. A variação de β_1 implica na translação de toda a curva e a alteração do parâmetro β_2 implica na modificação da tensão exercida pelos pesos de controle.

Na Interpolação β-Spline, assim como na Spline Natural, o controle é local e a continuidade da curva

resultante é de segunda ordem. O custo computacional para a determinação dos coeficientes polinomiais de cada segmento de curva é relativamente maior do que na B-Spline, entretanto, continua sendo extremamente reduzido.

Cada segmento de curva é definido por um polinômio cúbico calculado através do uso de quatro pontos. A expressão que representa este polinômio em cada intervalo i é obtida por

$$Q(u) = b_{-3}(u).P_{i-3} + b_{-2}(u).P_{i-2} + b_{-1}(u).P_{i-1} + b_0(u).P_i\,, \tag{2.59}$$

onde

$$Q(u) = [x(u) \quad y(u) \quad z(u)]\,, \tag{2.60}$$

$$P_i = [x_i \quad y_i \quad z_i]\,, \tag{2.61}$$

$$b_{-3}(u) = \frac{(2\beta_1^3) - (6\beta_1^3)u + (6\beta_1^3)u^2 - (2\beta_1^3)u^3}{\gamma}\,, \tag{2.62}$$

$$b_{-2}(u) = \frac{(\beta_2 + 4\beta_1^2 + 4\beta_1) + (6\beta_1^3 - 6\beta_1)u}{\gamma} + \frac{-(3\beta_2 + 6\beta_1^3 + 6\beta_1^2)u^2 + (2\beta_2 + 2\beta_1^3 + 2\beta_1^2 + 2\beta_1)u^3}{\gamma}\,, \tag{2.63}$$

$$b_{-1}(u) = \frac{2 + (6\beta_1)u + (3\beta_2 + 6\beta_1^2)u^2 - (2\beta_2 + 2\beta_1^2 + 2\beta_1 + 2)u^3}{\gamma}\,, \tag{2.64}$$

$$b_0(u) = \frac{2u^3}{\gamma}\,, \tag{2.65}$$

u varia de 0 até 1 para i = (3,...,n-1), n é o número de pontos do conjunto e

$$\gamma = \beta_2 + 2\beta_1^3 + 4\beta_1^2 + 4\beta_1 + 2\,. \tag{2.66}$$

A Figura 2.14 apresenta um conjunto aleatório de pontos interpolado por β-Spline com $\beta_1 = 1$ e $\beta_2 = 8$.

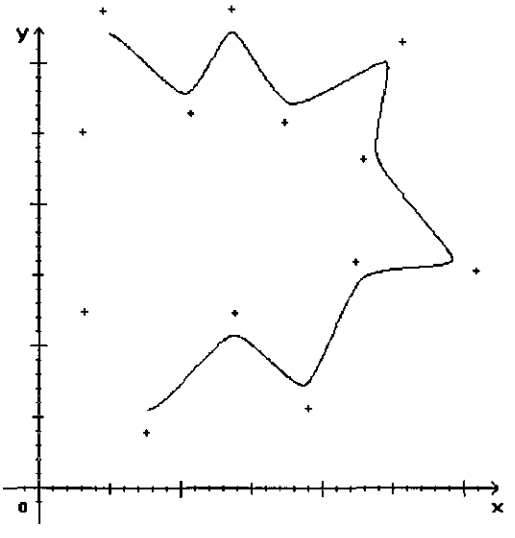
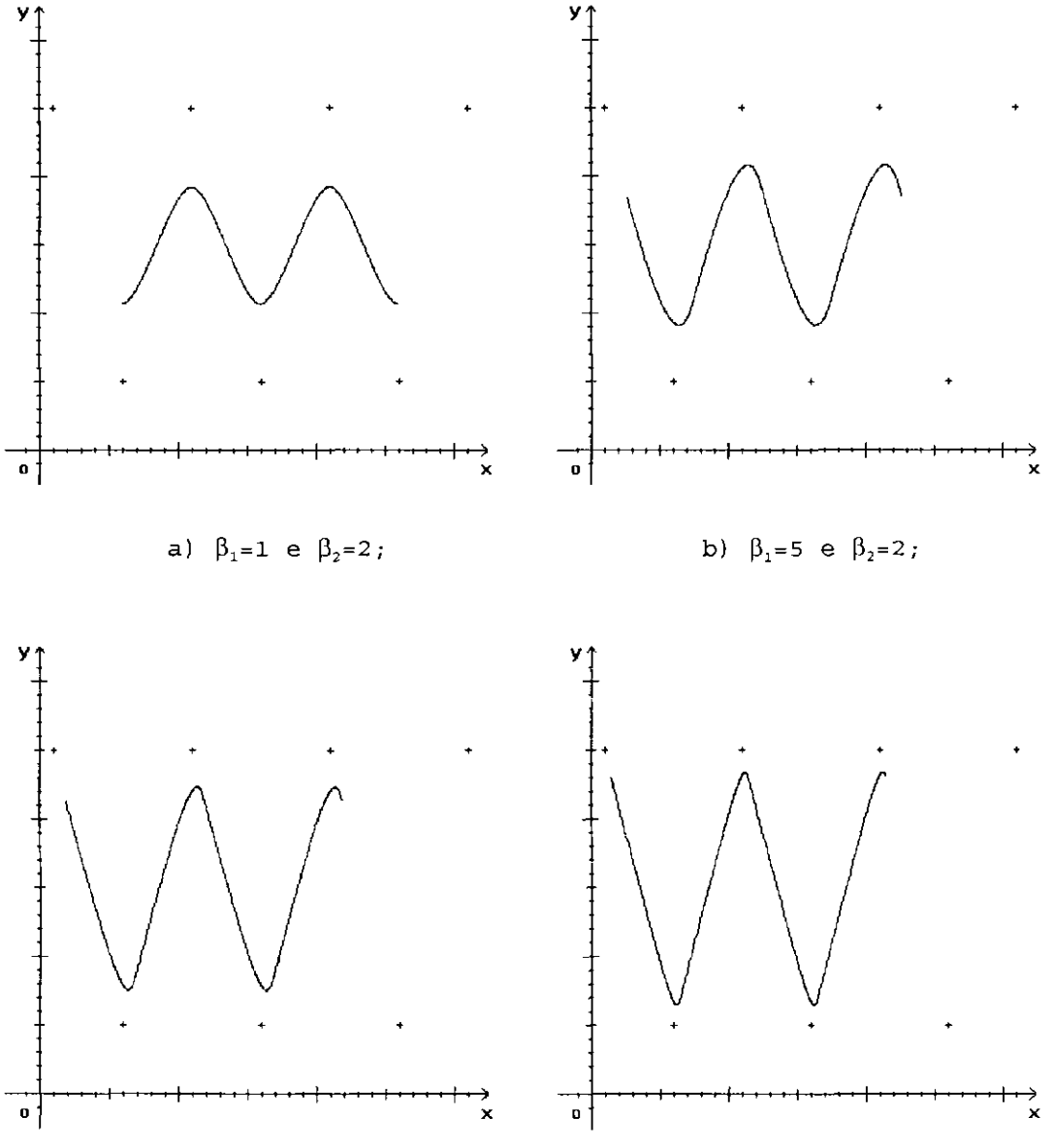


Fig. 2.14 - Interpolação β-Spline sobre curvas.

O aumento do valor do parâmetro β_1 implica no deslocamento da curva resultante na direção do primeiro ponto do conjunto de pontos a ser interpolado. Este aumento também causa um aumento na tensão exercida pelos pontos de controle.

A Figura 2.15 apresenta os resultados obtidos com a Interpolação β-Spline sobre curvas com diferentes valores de β_1.



a) $\beta_1=1$ e $\beta_2=2$;

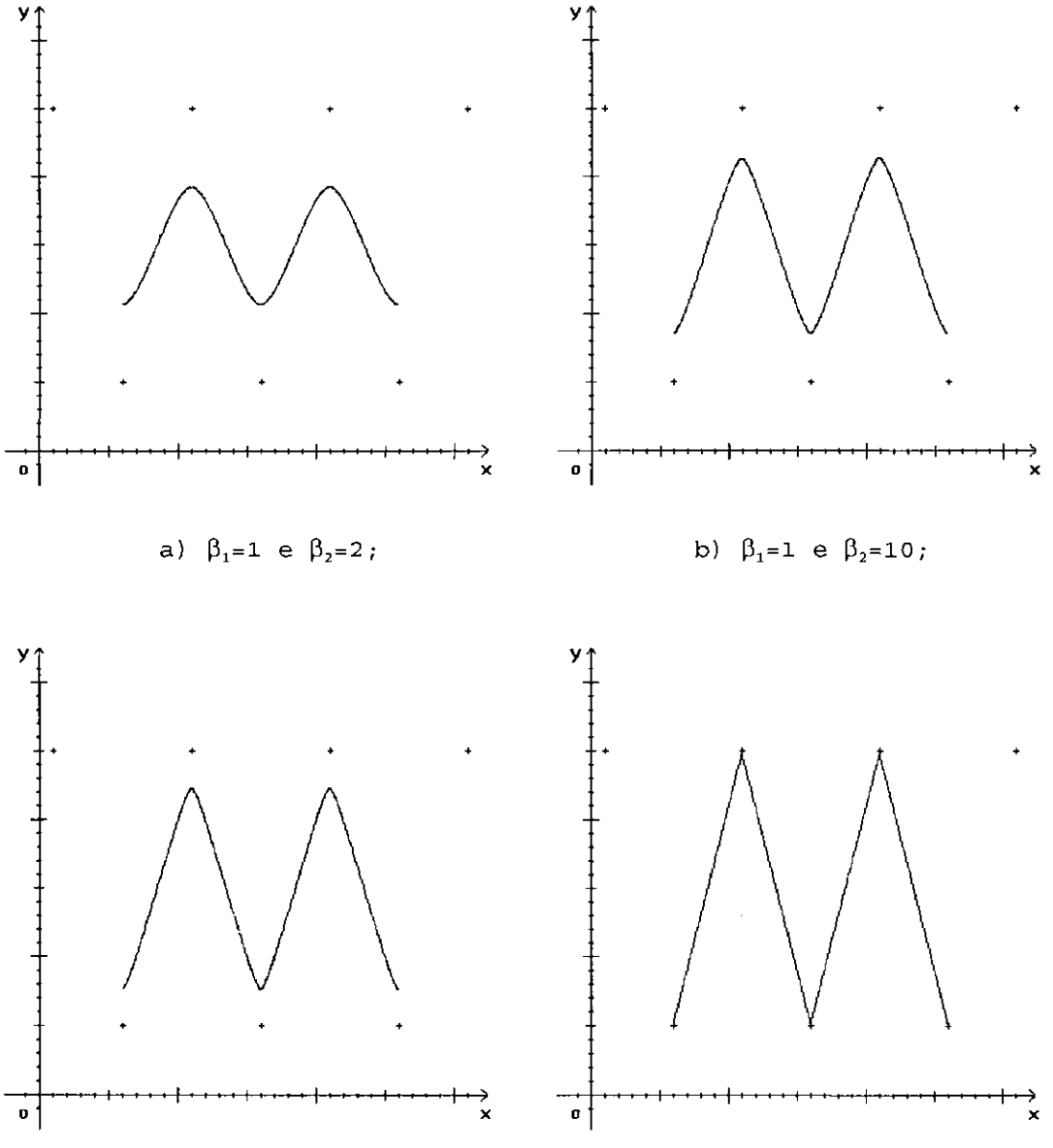
b) $\beta_1=5$ e $\beta_2=2$;

c) $\beta_1=10$ e $\beta_2=2$;

d) $\beta_1=20$ e $\beta_2=2$.

Fig. 2.15 - Interpolação β-Spline com diferentes valores de β_1.

Quanto maior for o valor de β_2, mais próximo estará o resultado de uma Interpolação Linear. A Figura 2.16 apresenta uma comparação entre os resultados alcançados com diferentes valores de β_2.



a) β_1=1 e β_2=2;

b) β_1=1 e β_2=10;

c) β_1=1 e β_2=18;

d) β_1=1 e β_2=300.

Fig. 2.16 - Interpolação β-Spline com diferentes valores de β_2.

A extensão de β-Spline para a interpolação de superfícies permite que seja determinado um único polinômio $x^{\alpha}.z^{\beta}$ para cada retalho da grade a ser interpolada, onde α = (0,1,2,3) e β = (0,1,2,3). A expressão que representa o polinômio em cada retalho (i,j) é obtida por

$$Q(s,t) = \sum_{m=-3}^{0}\left[\sum_{n=-3}^{0} P_{i+m,j+n}.b_m(s)\right].c_n(t) \quad , \tag{2.67}$$

onde

$$Q(s,t) = [x(s,t) \quad y(s,t) \quad z(s,t)] \quad , \tag{2.68}$$

$$P_{i,j} = [x_{i,j} \quad y_{i,j} \quad z_{i,j}] \quad , \tag{2.69}$$

$$b_{-3}(s) = \frac{(2\beta_{1,x}{}^3) - (6\beta_{1,x}{}^3)s + (6\beta_{1,x}{}^3)s^2 - (2\beta_{1,x}{}^3)s^3}{\gamma_x} \quad , \tag{2.70}$$

$$b_{-2}(s) = \frac{(\beta_{2,x} + 4\beta_{1,x}{}^2 + 4\beta_{1,x}) + (6\beta_{1,x}{}^3 - 6\beta_{1,x})s}{\gamma_x} +$$

$$+ \frac{-(3\beta_{2,x} + 6\beta_{1,x}{}^3 + 6\beta_{1,x}{}^2)s^2}{\gamma_x} + \quad , \tag{2.71}$$

$$+ \frac{(2\beta_{2,x} + 2\beta_{1,x}{}^3 + 2\beta_{1,x}{}^2 + 2\beta_{1,x})s^3}{\gamma_x}$$

$$b_{-1}(s) = \frac{2 + (6\beta_{1,x})s + (3\beta_{2,x} + 6\beta_{1,x}{}^2)s^2}{\gamma_x} +$$

$$+ \frac{-(2\beta_{2,x} + 2\beta_{1,x}{}^2 + 2\beta_{1,x} + 2)s^3}{\gamma_x} \quad , \tag{2.72}$$

$$b_0(s) = \frac{2s^3}{\gamma_x} \quad , \tag{2.73}$$

$$c_{-3}(t) = \frac{(2\beta_{1,z}{}^3) - (6\beta_{1,z}{}^3)t + (6\beta_{1,z}{}^3)t^2 - (2\beta_{1,z}{}^3)t^3}{\gamma_z} \quad , \tag{2.74}$$

$$c_{-2}(t) = \frac{(\beta_{2,z} + 4\beta_{1,z}^{\ 2} + 4\beta_{1,z}) + (6\beta_{1,z}^{\ 3} - 6\beta_{1,z})t}{\gamma_z} +$$

$$+ \frac{-(3\beta_{2,z} + 6\beta_{1,z}^{\ 3} + 6\beta_{1,z}^{\ 2})t^2}{\gamma_z} + \quad , \tag{2.75}$$

$$+ \frac{(2\beta_{2,z} + 2\beta_{1,z}^{\ 3} + 2\beta_{1,z}^{\ 2} + 2\beta_{1,z})t^3}{\gamma_z}$$

$$c_{-1}(t) = \frac{2 + (6\beta_{1,z})t + (3\beta_{2,z} + 6\beta_{1,z}^{\ 2})t^2}{\gamma_z} +$$

$$, \tag{2.76}$$

$$+ \frac{-(2\beta_{2,z} + 2\beta_{1,z}^{\ 2} + 2\beta_{1,z} + 2)t^3}{\gamma_z}$$

$$c_0(t) = \frac{2t^3}{\gamma_z}, \tag{2.77}$$

s varia de 0 até 1 para i = (3,...,nx-1), t varia de 0 até 1 para j = (3,...,nz-1), nx é o número de pontos na direção do eixo x, nz é o número de pontos na direção do eixo z e

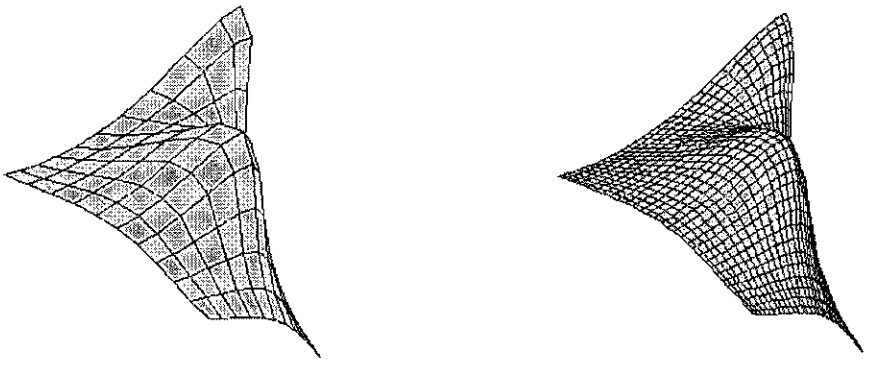
$$\gamma_x = \beta_{2,x} + 2\beta_{1,x}^{\ 3} + 4\beta_{1,x}^{\ 2} + 4\beta_{1,x} + 2, \tag{2.78}$$

$$\gamma_z = \beta_{2,z} + 2\beta_{1,z}^{\ 3} + 4\beta_{1,z}^{\ 2} + 4\beta_{1,z} + 2. \tag{2.79}$$

A Figura 2.17 apresenta uma grade refinada obtida através da Interpolação β-Spline, com $\beta_{1,x} = \beta_{1,z} = 1$ e $\beta_{2,x} = \beta_{2,z} = 2$, sobre uma grade regular aleatória.

Assim como na B-Spline, a superfície resultante de uma Interpolação β-Spline não passa pelos pontos da grade regular original. Entretanto, desde que haja uma grande interação entre o operador e o programa que efetuará a interpolação, o método pode ser aplicado no refinamento de grades. Dessa forma, com algumas alterações

nas coordenadas da grade regular grossa e nos valores dos parâmetros β_1 e β_2 , pode-se obter uma grade refinada capaz de representar satisfatoriamente o terreno sem que haja oscilações excessivas.



a) Grade regular aleatória; b) Grade regular refinada.

Fig. 2.17 - Interpolação β-Spline sobre superfícies.

As implementações da Interpolação β-Spline para Curvas e da Interpolação β-Spline para Superfícies estão listadas nos Apêndices L e M, respectivamente.

2.8 - INTERPOLAÇÃO AKIMA

Entre os métodos de interpolação existentes, o que mais se aproxima da interpolação manual é o método de Interpolação Akima (Akima, 1970). Este método, devido às suas características, possui grande aplicação em ambiente computacional.

Um dos fatores que mais contribuem para a sua utilização é o baixo custo computacional dos procedimentos implementados. Um outro fator interessante é a não

oscilação excessiva, que ocorre em outros métodos de interpolação.

Conforme pode ser observado na Figura 2.18, a curva resultante de uma Interpolação Akima passa exatamente pelos pontos de controle de maneira suave e ajustada.

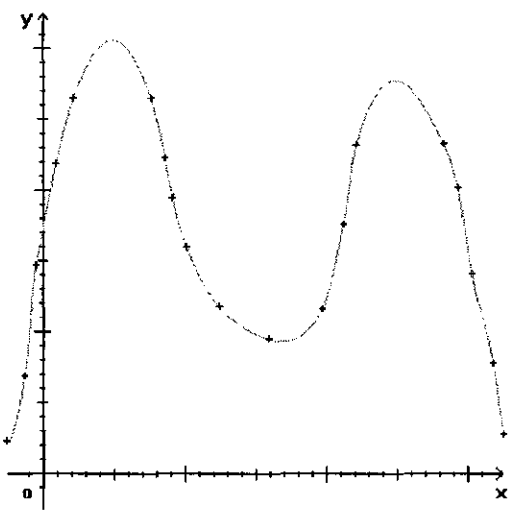
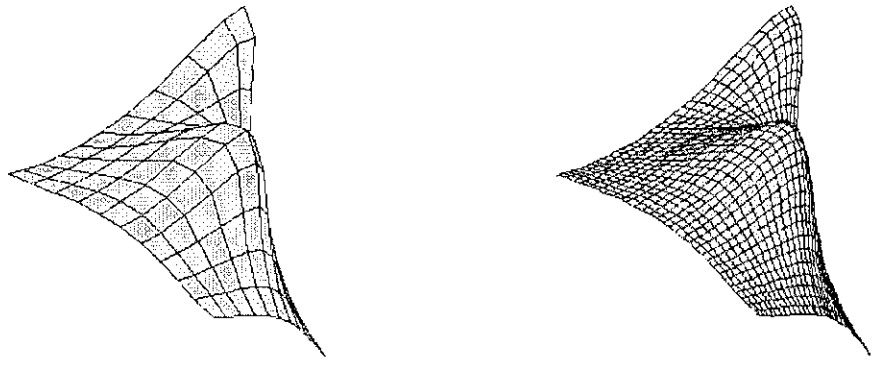


Fig. 2.18 - Interpolação Akima sobre curvas.

Observando estas características, pode-se supor que a utilização do Interpolador Akima sobre uma grade regular permite a geração de uma grade refinada com valores mais precisos e mais próximos dos dados de altimetria do terreno do que os obtidos por outros métodos.

A Figura 2.19 apresenta uma grade refinada obtida através da Interpolação Akima sobre uma grade regular aleatória.



a) Grade regular aleatória; b) Grade regular refinada.

Fig. 2.19 - Interpolação Akima sobre superfícies.

A fim de verificar e analisar as características deste interpolador, este trabalho tem como um de seus objetivos detalhar o método de Interpolação Akima para Ajuste de Curvas Suaves Baseado em Procedimentos Locais, Akima (1970), e o método de Interpolação Akima para Ajuste de Superfícies Suaves Baseado em Procedimentos Locais, Akima (1974a,b).

CAPÍTULO 3

INTERPOLADOR AKIMA PARA AJUSTE DE CURVAS SUAVES BASEADO EM PROCEDIMENTOS LOCAIS

3.1 - INTRODUÇÃO

A Interpolação Akima para Ajuste de Curvas Suaves Baseado em Procedimentos Locais é um método desenvolvido por Hiroshi Akima em 1970 para, a partir de um conjunto de pontos em um plano, ajustar uma curva suave sobre esses pontos. O método proposto será descrito neste capítulo e o seu desenvolvimento está baseado nas publicações feitas por Akima em outubro de 1970 e outubro de 1972.

Este interpolador foi planejado de modo que a curva resultante passasse exatamente sobre os pontos e que fosse suave e natural. Trata-se de uma função fragmentada composta por um conjunto de polinômios, cada qual com grau três no máximo, aplicáveis a intervalos sucessivos de pontos. Neste método, a derivada da curva é determinada em cada ponto localmente e cada polinômio é determinado pelas coordenadas e pela derivada de cada ponto.

Comparações indicam que a curva obtida pelo método proposto por Akima (1970) é muito mais parecida com a curva desenhada manualmente do que a obtida por outros métodos matemáticos.

3.2 - DETERMINAÇÃO DAS DERIVADAS NOS PONTOS DA CURVA

A derivada da curva em cada ponto é determinada localmente pelas coordenadas de cinco pontos,

sendo o ponto em questão o ponto central com dois pontos para cada lado.

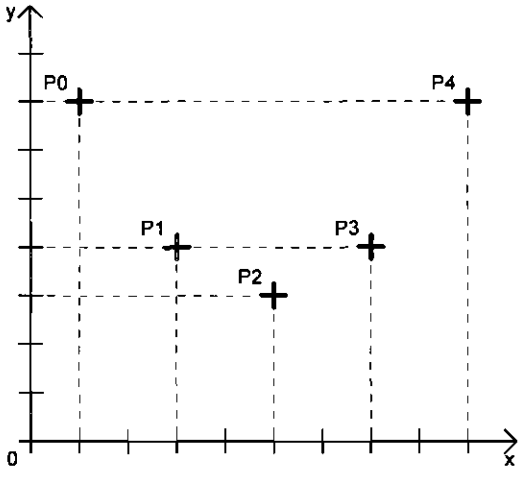


Fig. 3.1 - Coordenadas dos pontos para cálculo da derivada.

Conforme a Figura 3.1, os pontos P_0, P_1, P_2, P_3 e P_4 serão utilizados para exemplificar a determinação da derivada da curva no ponto P_2, ponto central.

3.2.1 - DETERMINAÇÃO DAS DERIVADAS PARA FUNÇÕES DE VALORES ÚNICOS

Segundo Akima (1970), pode-se assumir que a derivada no ponto P_2 deve se aproximar da declividade do segmento de reta P_1-P_2 quando a declividade de P_0-P_1 se aproximar da declividade de P_1-P_2. É altamente desejável que esta condição seja invariante sob transformação linear de escala do sistema de coordenadas, porque, dessa forma, o resultado da interpolação será o mesmo independentemente da escala utilizada.

Tendo em vista os cinco pontos apresentados na Figura 3.2, Akima (1970) previamente propôs a condição

de que a declividade do segmento de reta C-D (derivada da curva no ponto P_2) é determinada por

$$\frac{|P_1 - C|}{|C - A|} = \frac{|P_3 - D|}{|D - B|}, \qquad (3.1)$$

onde o ponto de intersecção de duas linhas retas estendidas além dos segmentos de reta P_0-P_1 e P_2-P_3 é denotado por A, o mesmo ponto correspondente aos segmentos de reta P_1-P_2 e P_3-P_4 por B, e os pontos de intersecção da tangente com duas linhas retas estendidas além dos segmentos P_0-P_1 e P_3-P_4 são denotados por C e D, respectivamente. A razão entre dois segmentos de reta, a qual é utilizada quando dois segmentos estão sobre uma linha reta, é usada aqui para representar simbolicamente a razão entre os comprimentos de dois segmentos de reta.

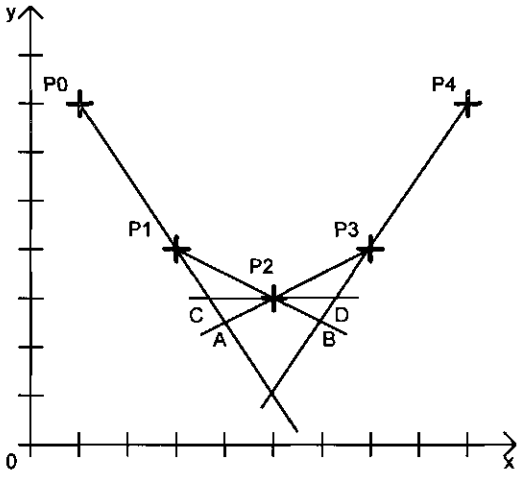


Fig. 3.2 - Condição para determinação da derivada da curva.

Sejam as coordenadas dos pontos P_0, P_1, P_2, P_3, P_4, A, B, C e D da Figura 3.2 denotadas por (x,y) e subscritas por 0, 1, 2, 3, 4, A, B, C e D, respectivamente.

Para simplificar o desenvolvimento das equações, pode-se definir

$$a_i = x_{i+1} - x_i, \tag{3.2}$$

$$b_i = y_{i+1} - y_i, \tag{3.3}$$

onde i = (0,1,2,3).

As declividades dos segmentos de reta P_0-P_1, P_1-P_2, P_2-P_3, P_3-P_4 e C-D são denotadas por m_0, m_1, m_2, m_3, m_4 e t, respectivamente, de modo que

$$m_i = \frac{y_{i+1} - y_i}{x_{i+1} - x_i} = \frac{b_i}{a_i}, \tag{3.4}$$

$$t = \frac{y_D - y_C}{x_D - x_C}, \tag{3.5}$$

onde i = (0,1,2,3).

A partir da Figura 3.2, portanto, torna-se claro que

$$\frac{y_A - y_1}{x_A - x_1} = \frac{y_C - y_1}{x_C - x_1} = \frac{b_1}{a_1}, \tag{3.6}$$

$$\frac{y_3 - y_B}{x_3 - x_B} = \frac{y_3 - y_D}{x_3 - x_D} = \frac{b_3}{a_3}, \tag{3.7}$$

$$\frac{y_2 - y_A}{x_2 - x_A} = \frac{b_2}{a_2}, \tag{3.8}$$

$$\frac{y_B - y_2}{x_B - x_2} = \frac{b_1}{a_1}, \tag{3.9}$$

$$\frac{y_2 - y_C}{x_2 - x_C} = \frac{y_D - y_2}{x_D - x_2} = t. \qquad (3.10)$$

Utilizando (3.2) e (3.3), obtém-se, a partir de (3.6),

$$\frac{(y_2 - y_A) - b_1}{(x_2 - x_A) - a_1} = \frac{b_0}{a_0}. \qquad (3.11)$$

Eliminando $(y_2 - y_A)$ de (3.8) e (3.11), tem-se

$$\frac{x_2 - x_A}{a_2} = \frac{a_0.b_1 - a_1.b_0}{a_0.b_2 - a_2.b_0}. \qquad (3.12)$$

Seguindo o mesmo procedimento,

$$\frac{x_B - x_2}{a_1} = \frac{a_2.b_3 - a_3.b_2}{a_1.b_3 - a_3.b_1}, \qquad (3.13)$$

$$(x_2 - x_C) = \frac{a_0.b_1 - a_1.b_0}{a_0.t - b_0}, \qquad (3.14)$$

$$(x_D - x_2) = \frac{a_2.b_3 - a_3.b_2}{b_3 - a_3.t}. \qquad (3.15)$$

A condição (3.1) pode ser reescrita da seguinte forma:

$$\left|\frac{(x_2 - x_C) - a_1}{(x_2 - x_A) - (x_2 - x_C)}\right| = \left|\frac{a_2 - (x_D - x_2)}{(x_D - x_2) - (x_B - x_2)}\right|, \qquad (3.16)$$

onde, por eliminação de $(x_2 - x_A)$, $(x_B - x_2)$, $(x_2 - x_C)$ e $(x_D - x_2)$ a partir de (3.12), (3.13), (3.14) e (3.15), obtém-se a equação quadrática de forma

$$|S_{01}.S_{13}|.(a_2.t - b_2)^2 = |S_{02}.S_{23}|.(a_1.t - b_1)^2, \qquad (3.17)$$

onde

$$S_{ij} = a_i.b_j - a_j.b_i\,, \qquad (\forall i \neq j)\,. \tag{3.18}$$

Com a condição de que os pontos P_1 e P_3 devem ficar do mesmo lado da tangente da curva no ponto P_2, segue-se que

$$(a_2.t - b_2).(a_1.t - b_1) \leq 0\,. \tag{3.19}$$

Portanto,

$$\sqrt{|S_{01}.S_{13}|}.(b_2 - a_2.t) = \sqrt{|S_{02}.S_{23}|}.(a_1.t - b_1)\,. \tag{3.20}$$

Resolvendo esta equação, tem-se

$$t = \frac{w_1.b_1 + w_2.b_2}{w_1.a_1 + w_2.a_2}\,, \tag{3.21}$$

onde

$$w_1 = \sqrt{|S_{02}.S_{23}|}\,, \tag{3.22}$$

$$w_2 = \sqrt{|S_{01}.S_{13}|}\,. \tag{3.23}$$

Essa é a expressão analítica da declividade do segmento de reta C-D, determinada sob a condição proposta em (3.1). Esta relação também pode ser expressa como

$$t = \frac{w_1'.m_1 + w_2'.m_2}{w_1' + w_2'}\,, \tag{3.24}$$

onde

$$w_1' = (\text{sinal de } a_1)\left|(m_2 - m_0).(m_3 - m_2)\right| , \tag{3.25}$$

$$w_2' = (\text{sinal de } a_2)\left|(m_1 - m_0).(m_3 - m_1)\right| . \tag{3.26}$$

No caso de função de valor único, pode-se sempre considerar a_1 e a_2 como positivos e, simplificando, tem-se

$$w_1' = \left|(m_2 - m_0).(m_3 - m_2)\right| , \tag{3.27}$$

$$w_2' = \left|(m_1 - m_0).(m_3 - m_1)\right| . \tag{3.28}$$

Sendo a condição (3.1) puramente geométrica, pode-se afirmar que seja invariante sob transformação linear do sistema de coordenadas, o que inclui transformação de escala e rotação. Tendo em vista (3.24), (3.25), (3.26) e a construção geométrica ditada por (3.1), nota-se que a declividade t depende exclusivamente das declividades de quatro secantes. Com isso, pode-se afirmar que a declividade t independe das larguras dos intervalos.

A partir de (3.24), (3.25) e (3.26), torna-se claro também que $m_0 = m_1$, $m_2 \neq m_0$ e $m_3 \neq m_2$ implicam em $t = m_0 = m_1$. Similarmente, $m_2 = m_3$, $m_0 \neq m_1$ e $m_3 \neq m_1$ implicam em $t = m_2 = m_3$. Quando $m_0 = m_1 = m_2$ ou quando $m_1 = m_2 = m_3$, t é indefinido por (3.24), entretanto, essa dificuldade pode ser facilmente sanada considerando $t = m_1 = m_2$ para tais casos como uma extensão lógica e natural.

Por outro lado, a condição (3.1) possui um grave problema. Sucede que, a partir de (3.24), (3.25) e (3.26), $t = m_1$ quando $m_1 = m_3$, $m_2 \neq m_0$ e $m_3 \neq m_2$. Similarmente, $t = m_2$ quando $m_2 = m_0$, $m_1 \neq m_0$ e $m_3 \neq m_1$.

Uma maneira simples de eliminar estas indesejáveis propriedades, no caso de funções de valores únicos, é modificar o peso dos coeficientes w_1' e w_2' em (3.24) para

$$w_1' = |m_3 - m_2|, \tag{3.29}$$

$$w_2' = |m_1 - m_0|. \tag{3.30}$$

Com essa modificação nos pesos, a maioria das propriedades desejáveis da condição (3.1) são mantidas. A única exceção é a perda da propriedade de invariância sob rotação do sistema de coordenadas. Trata-se de uma perda insignificante no caso de interpolação de funções de valores únicos.

Akima (1970), portanto, sugere que o valor t da derivada da curva no ponto P_2, para funções de valores únicos, seja calculado conforme

$$t = \frac{w_1'.m_1 + w_2'.m_2}{w_1' + w_2'}, \tag{3.24}$$

onde

$$w_1' = |m_3 - m_2|, \tag{3.29}$$

$$w_2' = |m_1 - m_0|. \tag{3.30}$$

Quando $m_0 = m_1 \neq m_2 = m_3$, a declividade t não pode ser definida por (3.24). A derivada t pode assumir qualquer valor entre m_1 e m_2 quando m_0 se aproxima de m_1 e m_3 se aproxima de m_2 simultaneamente. A pedra fundamental do método diz que $t = m_1$ quando $m_0 = m_1$ e, similarmente, $t = m_2$ quando $m_3 = m_2$. Entretanto, essas duas regras entram

em conflito quando $m_0 = m_1 \neq m_2 = m_3$. Como convenção para esse caso, o método define o cálculo da declividade t como sendo

$$t = \frac{m_1 + m_2}{2}, \tag{3.31}$$

onde os pesos dos coeficientes w_1' e w_2' assumem valor igual a 1. Esta convenção é também invariante sob transformação linear de escala do sistema de coordenadas.

3.2.2 - DETERMINAÇÃO DAS DERIVADAS PARA FUNÇÕES DE VALORES MÚLTIPLOS

No caso de funções de valores múltiplos, a propriedade de invariância sob rotação é essencial porque, mesmo rotacionando o sistema de coordenadas, o resultado da interpolação deve ser o mesmo. Mantendo-se esta condição, as indesejáveis propriedades descritas anteriormente podem ser eliminadas modificando-se os valores de w_1 e w_2, respectivamente (3.22) e (3.23), por

$$w_1 = |S_{23}|, \tag{3.32}$$

$$w_2 = |S_{01}|. \tag{3.33}$$

A invariância sob rotação, através do uso de (3.21) com os pesos w_1 e w_2 modificados, pode ser caracterizada pelo fato de que $|S_{ij}|$ é quantitativamente invariante, representando duas vezes a área do triângulo limitado pelos vetores (a_i, b_i) e (a_j, b_j). Dessa forma, (3.21) pode ser reescrita como

$$\frac{S_{10}}{S_{02}} = \frac{|S_{01}|}{|S_{23}|}, \tag{3.34}$$

se a_i e b_i forem escolhidos de acordo com (3.4) e (3.18). Isto modifica os pesos dos coeficientes de (3.32) e (3.33), fazendo com que

$$w_1' = a_1.|a_2.a_3.(m_3 - m_2)|, \tag{3.35}$$

$$w_2' = a_2.|a_0.a_1.(m_1 - m_0)|. \tag{3.36}$$

Observa-se, a partir de (3.35) e (3.36), que uma outra propriedade de dependência de t sobre m_0, m_1, m_2 e m_3 é perdida. Entretanto, esta propriedade não é importante no caso de funções de valores múltiplos.

Em suas demonstrações, Akima (1970) sugere, então, que o valor t da derivada da curva no ponto P_2, para funções de valores múltiplos, seja calculado conforme

$$t = \frac{w_1.b_1 + w_2.b_2}{w_1.a_1 + w_2.a_2}, \tag{3.21}$$

onde

$$w_1 = |a_2.b_3 - a_3.b_2|, \tag{3.37}$$

$$w_2 = |a_0.b_1 - a_1.b_0|. \tag{3.38}$$

3.3 - EXTRAPOLAÇÃO DE PONTOS PARA CÁLCULO DAS DERIVADAS

Tendo em vista que o cálculo da derivada da curva em cada ponto é feito localmente pelas coordenadas de cinco pontos, sendo o ponto em questão o ponto central com dois pontos para cada lado, observa-se claramente que nas extremidades há a necessidade de que dois pontos extras sejam definidos.

3.3.1 - EXTRAPOLAÇÃO DE PONTOS PARA FUNÇÕES DE VALORES ÚNICOS

Para exemplificar, o cálculo da derivada no ponto extremo da curva P_2 (com os dois pontos adjacentes P_1 e P_0) requer a definição dos pontos P_3 e P_4 extrapolados além do final da curva a ser ajustada, conforme a Figura 3.3.

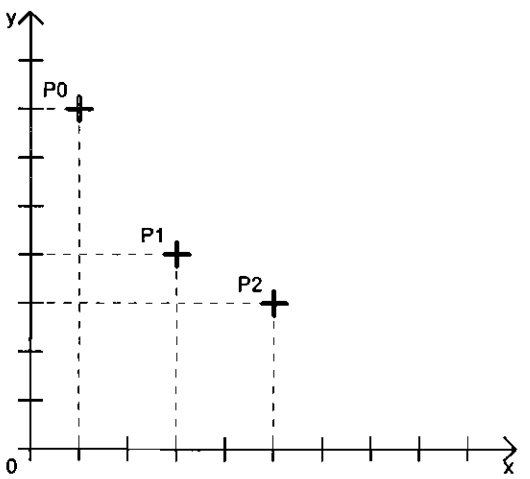


Fig. 3.3 - Coordenadas dos pontos para extrapolação na extremidade da direita.

Estes dois pontos extras, conforme Akima (1970), devem estar sobre a curva expressa por

$$y = g_0 + g_1.(x - x_2) + g_2.(x - x_2)^2 \ , \qquad (3.39)$$

onde g_0, g_1 e g_2 são constantes. Assumindo que

$$x_4 - x_2 = x_3 - x_1 = x_2 - x_0 , \qquad (3.40)$$

tem-se que

$$x_3 = x_2 + x_1 - x_0 , \qquad (3.41)$$

$$x_4 = x_3 + x_2 - x_1 . \tag{3.42}$$

As ordenadas y_3 e y_4, correspondentes a x_3 e x_4, podem ser determinadas a partir de (3.39). Disso resulta que

$$m_3 - m_2 = m_2 - m_1 = m_1 - m_0 . \tag{3.43}$$

Observa-se facilmente que, a partir de (3.43) e (3.4),

$$y_3 = y_2 + \left(2.\frac{y_2 - y_1}{x_2 - x_1} - \frac{y_1 - y_0}{x_1 - x_0} \right).(x_3 - x_2) , \tag{3.44}$$

$$y_4 = y_3 + \left(2.\frac{y_3 - y_2}{x_3 - x_2} - \frac{y_2 - y_1}{x_2 - x_1} \right).(x_4 - x_3) . \tag{3.45}$$

Analogamente, o cálculo da derivada no ponto extremo da curva P_2 (com os dois pontos adjacentes P_3 e P_4) requer a definição dos pontos P_1 e P_0 extrapolados além do final da curva a ser ajustada, conforme a Figura 3.4.

Desenvolvendo as equações, obtém-se os seguintes resultados:

$$x_1 = x_2 + x_3 - x_4 , \tag{3.46}$$

$$x_0 = x_1 + x_2 - x_3 , \tag{3.47}$$

$$y_1 = y_2 - \left(2.\frac{y_3 - y_2}{x_3 - x_2} - \frac{y_4 - y_3}{x_4 - x_3} \right).(x_2 - x_1) , \tag{3.48}$$

$$y_0 = y_1 - \left(2.\frac{y_2 - y_1}{x_2 - x_1} - \frac{y_3 - y_2}{x_3 - x_2} \right).(x_1 - x_0) . \tag{3.49}$$

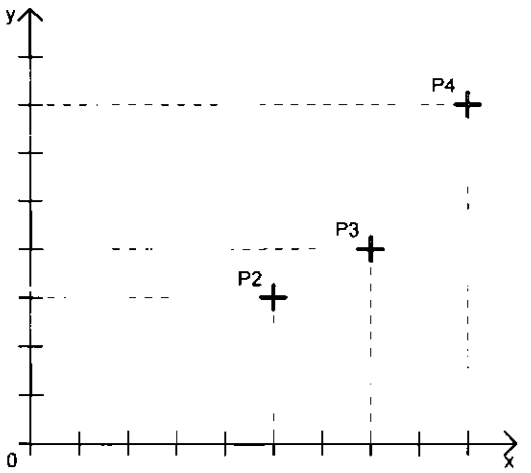


Fig. 3.4 - Coordenadas dos pontos para extrapolação na extremidade da esquerda.

3.3.2 - EXTRAPOLAÇÃO DE PONTOS PARA FUNÇÕES DE VALORES MÚLTIPLOS

Para funções de valores múltiplos, o cálculo de dois pontos extrapolados à esquerda da curva, conforme a situação descrita anteriormente pela Figura 3.4, é feito de modo que esses dois pontos extras estejam sobre a curva paramétrica expressa por

$$x = g_0 + g_1.u + g_2.u^2 \ , \qquad (3.50)$$

$$y = h_0 + h_1.u + h_2.u^2 \ , \qquad (3.51)$$

onde g_0, g_1, g_2, h_0, h_1 e h_2 são constantes e u é um parâmetro que varia de 0 até 1. Determinando-se os valores destas constantes, obtém-se

$$(x_4 - x_3) - (x_3 - x_2) = (x_3 - x_2) - (x_2 - x_1) = (x_2 - x_1) - (x_1 - x_0) \ , \qquad (3.52)$$

$$(y_4 - y_3) - (y_3 - y_2) = (y_3 - y_2) - (y_2 - y_1) = (y_2 - y_1) - (y_1 - y_0) \ . \qquad (3.53)$$

A partir de (3.52) e (3.53), tem-se que

$$x_1 = x_4 - 3x_3 + 3x_2, \tag{3.54}$$

$$x_0 = x_3 - 3x_2 + 3x_1, \tag{3.55}$$

$$y_1 = y_4 - 3y_3 + 3y_2, \tag{3.56}$$

$$y_0 = y_3 - 3y_2 + 3y_1. \tag{3.57}$$

Analogamente, para o cálculo de dois pontos extrapolados à direita da curva, conforme a situação descrita na Figura 3.3, obtém-se os seguintes resultados:

$$x_3 = 3x_2 - 3x_1 + x_0, \tag{3.58}$$

$$x_4 = 3x_3 - 3x_2 + x_1, \tag{3.59}$$

$$y_3 = 3y_2 - 3y_1 + y_0, \tag{3.60}$$

$$y_4 = 3y_3 - 3y_2 + y_1. \tag{3.61}$$

3.4 - DETERMINAÇÃO DA FUNÇÃO FRAGMENTADA

A interpolação proposta por Akima (1970) impõe que seja determinada uma função fragmentada composta por um conjunto de polinômios, cada qual com grau três no máximo. Cada polinômio expressa uma porção de curva entre um par de pontos consecutivos, de modo que a curva passe pelos dois pontos e que a declividade da curva nos pontos seja a mesma que a calculada pelo procedimento descrito na Seção 3.2.

3.4.1 - DETERMINAÇÃO DOS POLINÔMIOS PARA FUNÇÕES DE VALORES ÚNICOS

Quatro condições devem ser observadas na determinação do polinômio que representa um intervalo de pontos consecutivos. Exemplificando, no cálculo do polinômio que representa a curva entre os pontos P_0 e P_1, as condições são as seguintes:

$$y = y_0, \tag{3.62}$$

$$\frac{dy}{dx} = t_0, \tag{3.63}$$

no ponto P_0, e

$$y = y_1, \tag{3.64}$$

$$\frac{dy}{dx} = t_1, \tag{3.65}$$

no ponto P_1. Sendo t_0 e t_1 as declividades da curva nestes pontos, pode-se determinar um polinômio único de grau três.

Este polinômio pode ser expresso por muitas maneiras diferentes. Uma forma de expressá-lo é a seguinte:

$$y = b_0 + b_1.(x - x_0) + b_2.(x - x_0)^2 + b_3.(x - x_0)^3, \tag{3.66}$$

onde

$$b_0 = y_0, \tag{3.67}$$

$$b_1 = t_0, \tag{3.68}$$

$$b_2 = \frac{3.\frac{y_1 - y_0}{x_1 - x_0} - 2.t_0 - t_1}{x_1 - x_0}, \tag{3.69}$$

$$b_3 = \frac{t_0 + t_1 - 2.\frac{y_1 - y_0}{x_1 - x_0}}{(x_1 - x_0)^2}. \tag{3.70}$$

3.4.2 - DETERMINAÇÃO DOS POLINÔMIOS PARA FUNÇÕES DE VALORES MÚLTIPLOS

No caso de funções de valores múltiplos, o polinômio que representa um intervalo de pontos consecutivos é expresso parametricamente. Nessa parametrização, ao invés de se trabalhar com dy/dx, utiliza-se dx/du e dy/du, onde u é o valor paramétrico que varia de 0 até 1 dentro do intervalo. Isso implica na necessidade de se reescrever a equação (3.21) de modo a eliminar t = dy/dx. Os valores dx/du e dy/du são obtidos através do uso de sen θ e cos θ ao invés de t = tan θ:

$$\frac{dx}{du} = r.\cos\theta, \tag{3.71}$$

$$\frac{dy}{du} = r.\operatorname{sen}\theta, \tag{3.72}$$

onde θ é o ângulo formado entre a tangente t da curva e o eixo de abcissas, e r é o valor do módulo do vetor da tangente. Para esses cálculos,

$$\cos\theta = \frac{ra}{\sqrt{ra^2 + rb^2}}, \tag{3.73}$$

$$\operatorname{sen}\theta = \frac{rb}{\sqrt{ra^2 + rb^2}}, \tag{3.74}$$

onde, utilizando-se (3.2) e (3.3) e os pesos w_1 e w_2 de (3.37) e (3.38),

$$ra = w_1.a_1 + w_2.a_2, \tag{3.75}$$

$$rb = w_1.b_1 + w_2.b_2. \tag{3.76}$$

Neste caso, oito condições devem ser observadas na determinação do polinômio paramétrico que representa um intervalo de pontos consecutivos. Desde que as coordenadas do intervalo sejam (x_0,y_0) e (x_1,y_1) e as direções da curva nesses pontos sejam $(\cos\theta_0, \text{sen}\,\theta_0)$ e $(\cos\theta_1, \text{sen}\,\theta_1)$ respectivamente, as condições são as seguintes:

$$x = x_0, \tag{3.77}$$

$$y = y_0, \tag{3.78}$$

$$\frac{dx}{du} = r_0.\cos\theta_0, \tag{3.79}$$

$$\frac{dy}{du} = r_0.\text{sen}\,\theta_0, \tag{3.80}$$

quando u for igual a 0 e

$$x = x_1, \tag{3.81}$$

$$y = y_1, \tag{3.82}$$

$$\frac{dx}{du} = r_1.\cos\theta_1, \tag{3.83}$$

$$\frac{dy}{du} = r_1.\text{sen}\,\theta_1, \tag{3.84}$$

quando u for igual a 1. r_0 e r_1 são os módulos dos vetores das tangentes da curva nos pontos P_0 e P_1, respectivamente. Segundo o método proposto por Akima (1970),

$$r_0 = r_1 = r = \sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2} \quad . \tag{3.85}$$

3.4.2.1 - DETERMINAÇÃO PARAMÉTRICA CONFORME O MÉTODO DE AKIMA

Assume-se que a curva que representa o intervalo entre os pontos (x_0,y_0) e (x_1,y_1) pode ser expressa por

$$x = p_0 + p_1.u + p_2.u^2 + p_3.u^3, \tag{3.86}$$

$$y = q_0 + q_1.u + q_2.u^2 + q_3.u^3, \tag{3.87}$$

onde p_0, p_1, p_2, p_3, q_0, q_1, q_2 e q_3 são constantes e u é um parâmetro que varia de 0 até 1. Determinando-se de forma única os valores das constantes, tem-se que

$$p_0 = x_0, \tag{3.88}$$

$$p_1 = r_0.\cos\theta_0, \tag{3.89}$$

$$p_2 = 3(x_1 - x_0) - r_1.\cos\theta_1 + 2.r_0.\cos\theta_0 \quad , \tag{3.90}$$

$$p_3 = -2(x_1 - x_0) + r_1.\cos\theta_1 + r_0.\cos\theta_0 \quad , \tag{3.91}$$

$$q_0 = y_0, \tag{3.92}$$

$$q_1 = r_0.\mathrm{sen}\theta_0, \tag{3.93}$$

$$q_2 = 3(y_1 - y_0) - r_1.\mathrm{sen}\theta_1 + 2.r_0.\mathrm{sen}\theta_0 \quad , \tag{3.94}$$

$$q_3 = -2(y_1 - y_0) + r_1.\mathrm{sen}\theta_1 + r_0.\mathrm{sen}\theta_0 \quad . \tag{3.95}$$

A curva interpolada é invariante sob rotação do sistema de coordenadas, mas variante sob transformação linear de escala. Portanto, tanto as abcissas quanto as ordenadas devem ser escaladas com suas respectivas unidades, tendo um comprimento real igual no gráfico.

3.4.2.2 - DETERMINAÇÃO PARAMÉTRICA CONFORME A CURVA DE HERMITE

A forma Hermite de um segmento de curva polinomial cúbico que representa o intervalo entre os pontos (x_0,y_0) e (x_1,y_1) é determinada pelos dois pontos das extremidades P_0 e P_1 e os vetores das tangentes R_0 e R_1 nesses pontos.

Após uma série de deduções, descritas por Foley et al. (1992), a Curva de Hermite Q(u) = [x(u) y(u) z(u)] pode ser definida por

$$Q(u) = q_0.P_0 + q_1.P_1 + q_2.R_0 + q_3.R_1, \qquad (3.96)$$

onde

$$q_0 = 2u^3 - 3u^2 + 1 \ , \qquad (3.97)$$

$$q_1 = -2u^3 + 3u^2, \qquad (3.98)$$

$$q_2 = u^3 - 2u^2 + u, \qquad (3.99)$$

$$q_3 = u^3 - u^2, \qquad (3.100)$$

e u é um parâmetro que varia de 0 até 1.

Considerando os cálculos dos vetores das tangentes da curva propostos por Akima (1970), conforme (3.73), (3.74) e (3.85), pode-se expressar a curva por

$$x = q_0.x_0 + q_1.x_1 + q_2.r_0.\cos\theta_0 + q_3.r_1.\cos\theta_1 , \qquad (3.101)$$

$$y = q_0.y_0 + q_1.y_1 + q_2.r_0.\mathrm{sen}\theta_0 + q_3.r_1.\mathrm{sen}\theta_1 , \qquad (3.102)$$

onde q_0, q_1, q_2 e q_3 assumem os valores descritos anteriormente.

3.5 - ANÁLISE DOS RESULTADOS

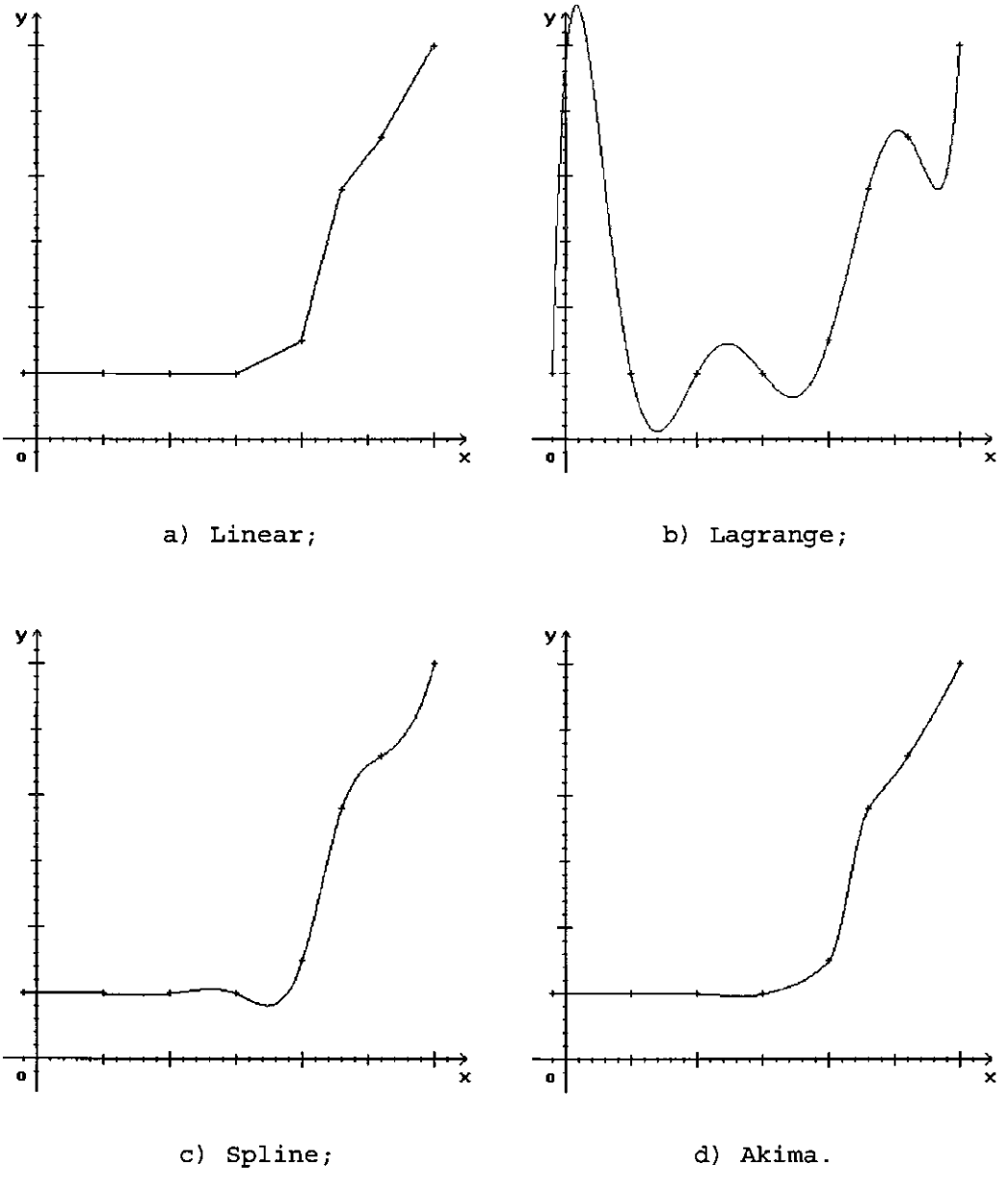
Os métodos propostos por Akima (1970) foram implementados em um sistema demonstrativo e testados exaustivamente. Os resultados obtidos nas interpolações foram analisados individualmente e as curvas resultantes, tanto no método para funções de valores únicos quanto para funções de valores múltiplos, foram comparadas.

3.5.1 - RESULTADOS OBTIDOS NA INTERPOLAÇÃO AKIMA PARA FUNÇÕES DE VALORES ÚNICOS

Os resultados obtidos com o método proposto por Akima (1970) para efetuar a interpolação sobre funções de valores únicos foram totalmente satisfatórios. A curva realmente se comporta de maneira suave e totalmente ajustada sobre os pontos.

A Figura 3.5 apresenta uma comparação entre alguns métodos de interpolação, utilizando um mesmo conjunto de pontos escolhidos arbitrariamente.

Analisando os gráficos, torna-se evidente que os resultados obtidos com a Interpolação Akima são menos oscilantes e mais próximos dos resultados de uma interpolação manual do que os obtidos por outros métodos de interpolação.



a) Linear; b) Lagrange;

c) Spline; d) Akima.

Fig. 3.5 - Comparação entre alguns métodos de interpolação.

3.5.2 - RESULTADOS OBTIDOS NA INTERPOLAÇÃO AKIMA PARA FUNÇÕES DE VALORES MÚLTIPLOS

Os resultados obtidos com o método proposto por Akima (1970) para interpolação sobre funções de valores múltiplos foram satisfatórios em quase todas as situações. A curva realmente se comporta de maneira suave e totalmente

ajustada sobre os pontos, conforme pode ser visto na Figura 3.6.

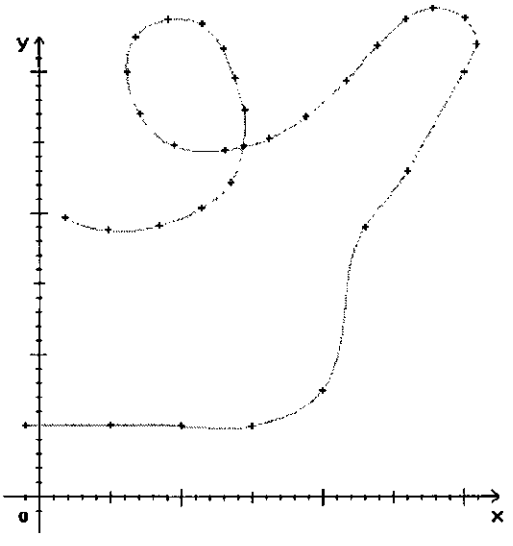


Fig. 3.6 - Curva resultante de uma Interpolação Akima para funções de valores múltiplos parametrizada conforme o Método de Akima.

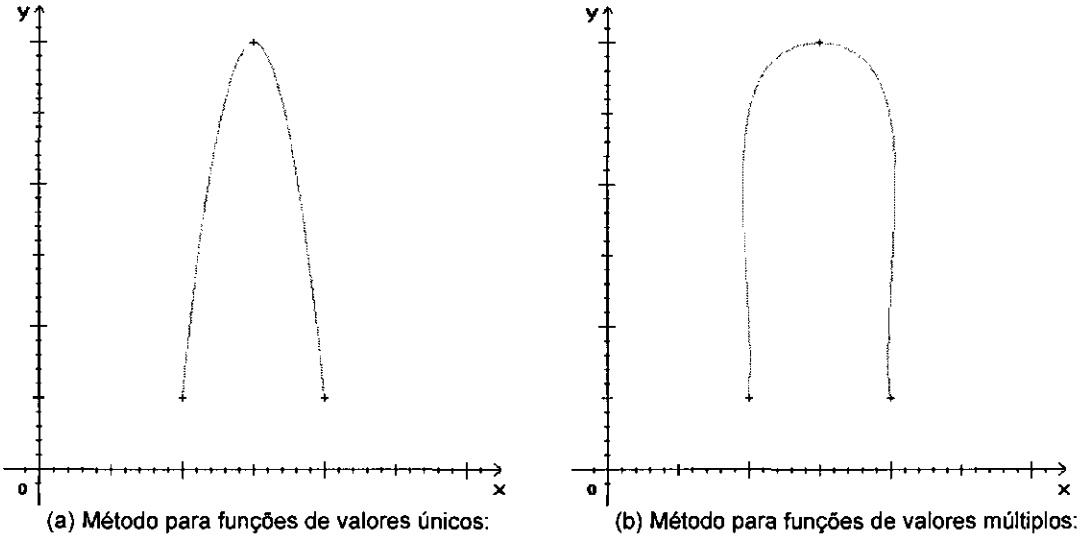


Fig. 3.7 - Comparação entre os métodos de interpolação para funções de valores únicos e múltiplos.

Entretanto, em determinadas situações, os resultados obtidos foram bastante diferentes do esperado, conforme pode ser observado na Figura 3.7.

Após uma série de análises, verificou-se que a diferença entre os resultados obtidos pelos dois métodos se deve a uma incorreção encontrada no método proposto por Akima (1970), no que diz respeito ao cálculo do módulo do vetor da tangente, em (3.85).

Segundo Foley et al. (1992), a forma de Hermite para um segmento de curva que representa um intervalo entre dois pontos depende das coordenadas dos pontos, da direção da tangente da curva em cada ponto e dos módulos dos vetores dessas tangentes. Com o método proposto por Akima (1970) para determinação dos ângulos dos vetores em termos de seno e cosseno, obtém-se valores únicos e invariáveis de direção da tangente. Entretanto, a definição imprecisa dos valores dos módulos destes vetores torna a curva resultante diferente no método proposto para funções de valores múltiplos.

A Figura 3.8 demonstra as diferentes curvas obtidas entre os pontos P_0 e P_1, mantendo-se as direções dos vetores das tangentes R_0 e R_1 e variando-se os valores dos módulos desses vetores.

O cálculo dos módulos dos vetores das tangentes, no método proposto por Akima (1970) para funções de valores múltiplos, pode ser aprimorado, fazendo com que os resultados obtidos entre os dois métodos sejam os mesmos.

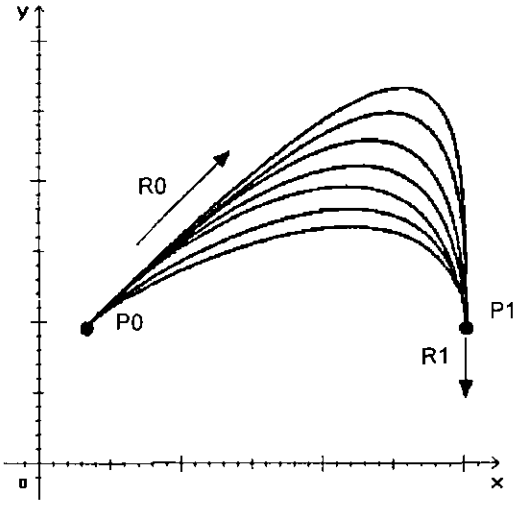


Fig. 3.8 - Família de curvas obtidas variando-se os valores dos módulos dos vetores das tangentes.

Considerando o intervalo entre os pontos P_0 e P_1 e os vetores das tangentes R_0 e R_1 nesses pontos, tem-se que

$$rv = \sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2} \quad , \tag{3.103}$$

onde rv é o módulo do vetor RV definido pelos pontos P_0 e P_1.

Com o uso da propriedade do produto escalar entre dois vetores e considerando inicialmente os vetores R_0 e R_1 como sendo os vetores unitários $(\cos\theta_0, \operatorname{sen}\theta_0)$ e $(\cos\theta_1, \operatorname{sen}\theta_1)$ respectivamente, tem-se que

$$\cos\beta_0 = \frac{\cos\theta_0.(x_1 - x_0) + \operatorname{sen}\theta_0.(y_1 - y_0)}{rv} \quad , \tag{3.104}$$

$$\cos\beta_1 = \frac{\cos\theta_1.(x_1 - x_0) + \operatorname{sen}\theta_1.(y_1 - y_0)}{rv} \quad , \tag{3.105}$$

onde θ é o ângulo formado entre a tangente da curva e o eixo de abcissas, e β é o ângulo formado entre o vetor da tangente no ponto e o vetor RV.

Dessa maneira, projetando o módulo do vetor RV, obtém-se

$$r_0 = \left| rv.\cos\beta_0 \right|, \tag{3.106}$$

$$r_1 = \left| rv.\cos\beta_1 \right|, \tag{3.107}$$

onde r é o módulo do vetor da tangente no ponto.

Esta forma de se estimar os módulos dos vetores das tangentes apresenta bons resultados, conforme pode ser observado na Figura 3.9.

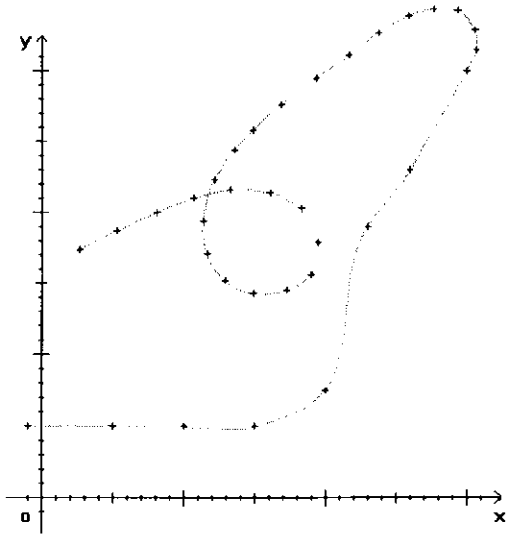
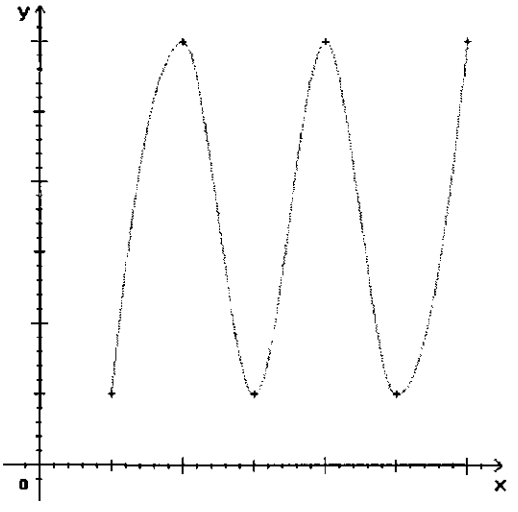
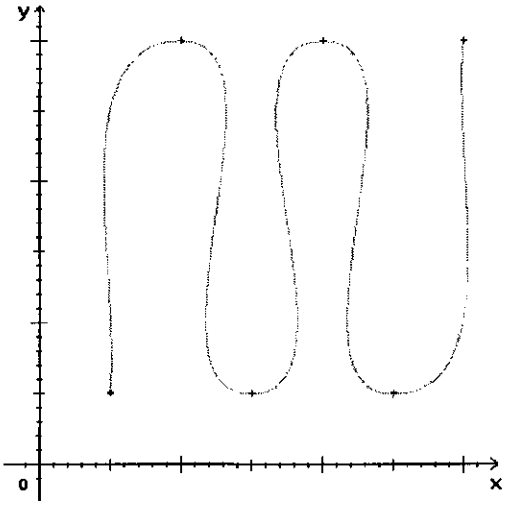


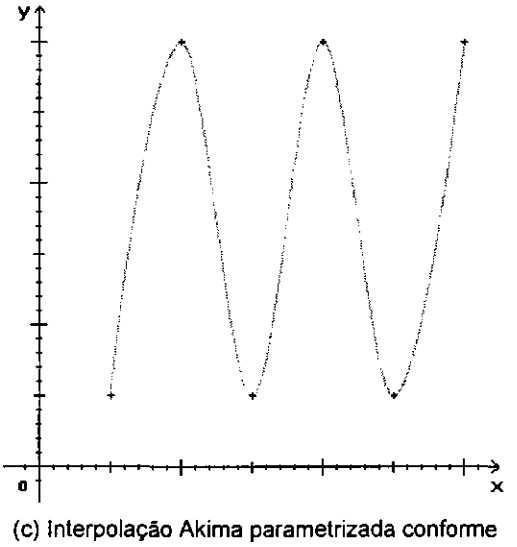
Fig. 3.9 - Curva resultante de uma Interpolação Akima parametrizada conforme a Curva de Hermite e com o método proposto para determinação dos módulos das tangentes.



(a) Interpolação Akima para funções de valores únicos;



(b) Interpolação Akima para funções de valores múltiplos;



(c) Interpolação Akima parametrizada conforme a Curva de Hermite e com método proposto para cálculo dos módulos das tangentes.

Fig. 3.10 - Comparação entre os três métodos de Interpolação Akima para Curvas.

As curvas resultantes são mais próximas das obtidas com o método de interpolação para funções de valores únicos. Aparentemente os resultados são

satisfatórios, mas há a necessidade de que sejam feitos estudos mais profundos neste sentido.

A Figura 3.10 apresenta uma comparação entre os dois métodos de Interpolação Akima para Ajuste de Curvas Suaves e o método de Interpolação Akima parametrizado conforme a Curva de Hermite.

As implementações da Interpolação Akima para Ajuste de Curvas Suaves Sobre Funções de Valores Únicos e da Interpolação Akima para Ajuste de Curvas Suaves Sobre Funções de Valores Múltiplos estão listadas nos Apêndices N e O, respectivamente.

A implementação da Interpolação Akima para Ajuste de Curvas Suaves Sobre Funções de Valores Múltiplos Parametricamente Conforme a Curva de Hermite e com Método Proposto para Cálculo dos Módulos das Tangentes está listada no Apêndice P.

CAPÍTULO 4

INTERPOLADOR AKIMA PARA AJUSTE DE SUPERFÍCIES SUAVES BASEADO EM PROCEDIMENTOS LOCAIS

4.1 - INTRODUÇÃO

A Interpolação Akima para Ajuste de Superfícies Suaves Baseado em Procedimentos Locais é um método desenvolvido por Hiroshi Akima em 1974 para, a partir dos pontos de uma grade retangular, ajustar uma superfície suave sobre esses pontos. O método proposto será descrito neste capítulo e o seu desenvolvimento está baseado nas publicações feitas por Akima em janeiro de 1974.

Este interpolador é uma extensão do método proposto para Ajuste de Curvas Suaves Baseado em Procedimentos Locais. Trata-se de uma função fragmentada $y = y(x,z)$ composta por um conjunto de polinômios bicúbicos $x^{\alpha}.z^{\beta}$, onde $\alpha = (0,1,2,3)$ e $\beta = (0,1,2,3)$. Os polinômios são aplicáveis a células sucessivas da grade retangular. A Figura 4.1 apresenta uma célula de uma grade retangular.

Cada polinômio é determinado pelas coordenadas e pelos valores das derivadas parciais $tx = (\partial y/\partial x)$, $tz = (\partial y/\partial z)$ e $txz = (\partial^2 y/\partial x \partial z)$ nos quatro pontos que definem a célula. Estas derivadas parciais são determinadas localmente.

A função da interpolação $y = y(x,z)$ é suave e possui continuidade na derivada parcial de primeira ordem.

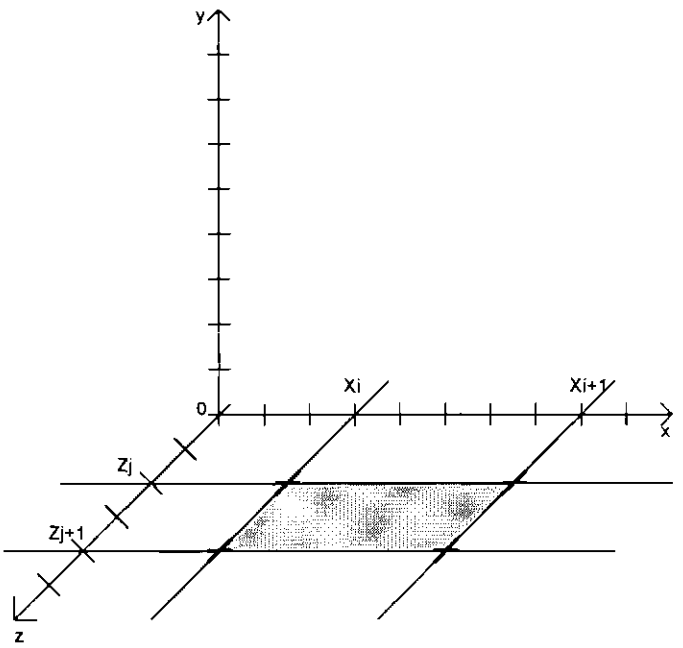


Fig. 4.1 - Célula de uma grade retangular.

4.2 - DETERMINAÇÃO DAS DERIVADAS NOS PONTOS DA SUPERFÍCIE

O cálculo das derivadas parciais é feito com o uso de vinte pontos, além dos quatro pontos que definem a célula. Conforme pode ser observado na Figura 4.2, os intervalos em x e em z podem ser diferentes, porém devem ser regulares.

A suposição utilizada por Akima (1970) no método para interpolação de curvas é estendida nesta proposta para permitir a determinação das derivadas parciais de primeira ordem tx e tz. Dessa forma, no ponto (x_i, z_j), tem-se que

$$tx_{i,j} = \frac{wx_1 . mx_{i-1,j} + wx_2 . mx_{i,j}}{wx_1 + wx_2}, \qquad (4.1)$$

$$tz_{i,j} = \frac{wz_1 . mz_{i,j-1} + wz_2 . mz_{i,j}}{wz_1 + wz_2}, \qquad (4.2)$$

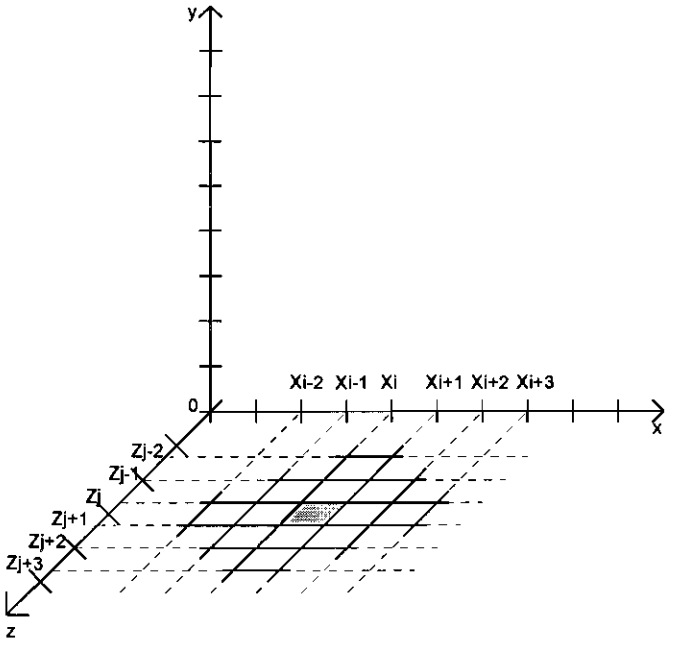


Fig. 4.2 - Pontos necessários para o cálculo das derivadas parciais.

onde os pesos wx_1, wx_2, wz_1 e wz_2 são obtidos através das seguintes equações:

$$wx_1 = \left| mx_{i+1,j} - mx_{i,j} \right|, \tag{4.3}$$

$$wx_2 = \left| mx_{i-1,j} - mx_{i-2,j} \right|, \tag{4.4}$$

$$wz_1 = \left| mz_{i,j+1} - mz_{i,j} \right|, \tag{4.5}$$

$$wz_2 = \left| mz_{i,j-1} - mz_{i,j-2} \right|, \tag{4.6}$$

onde as diferenças divididas de primeira ordem mx e mz são dadas por

$$mx_{i,j} = \frac{y_{i+1,j} - y_{i,j}}{x_{i+1} - x_i}, \tag{4.7}$$

$$mz_{i,j} = \frac{y_{i,j+1} - y_{i,j}}{z_{j+1} - z_j} . \qquad (4.8)$$

Estendendo este princípio para o caso de $y = y(x,z)$, tem-se que a derivada parcial de segunda ordem txz no ponto (x_i, z_j) é determinada por

$$txz_{i,j} = \frac{wx_1(wz_1 . mxz_{i-1,j-1} + wz_2 . mxz_{i-1,j})}{(wx_1 + wx_2).(wz_1 + wz_2)} + \frac{wx_2(wz_1 . mxz_{i,j-1} + wz_2 . mxz_{i,j})}{(wx_1 + wx_2).(wz_1 + wz_2)} , \qquad (4.9)$$

onde a diferença dividida de segunda ordem mxz é obtida por

$$mxz_{i,j} = \frac{mx_{i,j+1} - mx_{i,j}}{z_{j+1} - z_j} = \frac{mz_{i+1,j} - mz_{i,j}}{x_{i+1} - x_i} . \qquad (4.10)$$

Quando $wx_1 = wx_2 = 0$ e/ou $wz_1 = wz_2 = 0$, as derivadas parciais não podem ser determinadas por este processo. Como convenção, deve-se assumir que $wx_1 = wx_2 = 1$ e/ou $wz_1 = wz_2 = 1$ nestes casos.

4.3 - EXTRAPOLAÇÃO DE PONTOS PARA CÁLCULO DAS DERIVADAS

O cálculo das derivadas parciais em cada ponto é feito localmente com o uso de dois pontos para cada lado e um ponto para cada diagonal. Portanto, torna-se clara a necessidade de extrapolar a grade em duas linhas e duas colunas de pontos.

A Figura 4.3 apresenta uma grade regular que será usada como exemplo na extrapolação de pontos para cálculo das derivadas parciais.

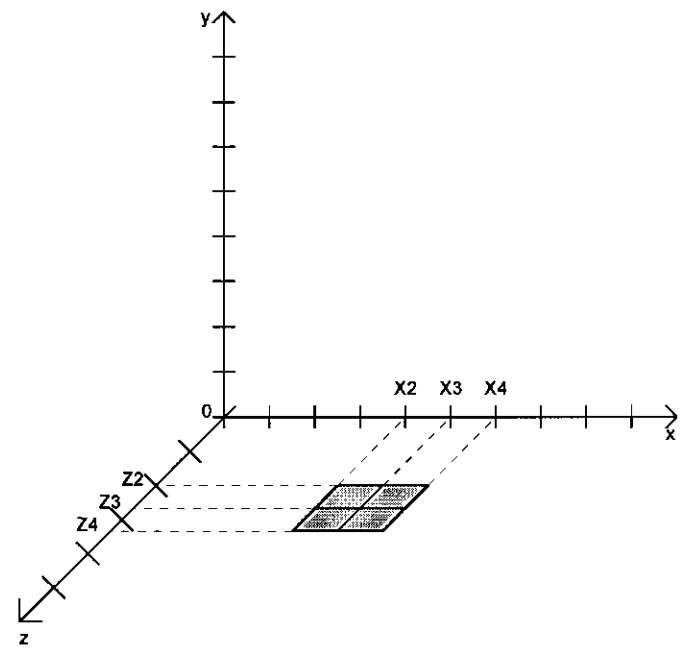


Fig. 4.3 - Células de uma grade regular.

Analogamente ao método de extrapolação proposto por Akima (1970), no eixo x tem-se que

$$x_1 = x_2 + x_3 - x_4 , \tag{4.11}$$

$$x_0 = x_1 + x_2 - x_3 , \tag{4.12}$$

$$y_{1,j} = y_{2,j} - \left(2.\frac{y_{3,j} - y_{2,j}}{x_3 - x_2} - \frac{y_{4,j} - y_{3,j}}{x_4 - x_3} \right).(x_2 - x_1) , \tag{4.13}$$

$$y_{0,j} = y_{1,j} - \left(2.\frac{y_{2,j} - y_{1,j}}{x_2 - x_1} - \frac{y_{3,j} - y_{2,j}}{x_3 - x_2} \right).(x_1 - x_0) , \tag{4.14}$$

$$x_5 = x_4 + x_3 - x_2 , \tag{4.15}$$

$$x_6 = x_5 + x_4 - x_3 , \tag{4.16}$$

$$y_{5,j} = y_{4,j} + \left(2.\frac{y_{4,j} - y_{3,j}}{x_4 - x_3} - \frac{y_{3,j} - y_{2,j}}{x_3 - x_2} \right).(x_5 - x_4) , \tag{4.17}$$

$$y_{6,j} = y_{5,j} + \left(2.\frac{y_{5,j} - y_{4,j}}{x_5 - x_4} - \frac{y_{4,j} - y_{3,j}}{x_4 - x_3} \right).(x_6 - x_5), \qquad (4.18)$$

onde j = (2,3,4) e, no eixo z,

$$z_1 = z_2 + z_3 - z_4, \qquad (4.19)$$

$$z_0 = z_1 + z_2 - z_3, \qquad (4.20)$$

$$y_{i,1} = y_{i,2} - \left(2.\frac{y_{i,3} - y_{i,2}}{z_3 - z_2} - \frac{y_{i,4} - y_{i,3}}{z_4 - z_3} \right).(z_2 - z_1), \qquad (4.21)$$

$$y_{i,0} = y_{i,1} - \left(2.\frac{y_{i,2} - y_{i,1}}{z_2 - z_1} - \frac{y_{i,3} - y_{i,2}}{z_3 - z_2} \right).(z_1 - z_0), \qquad (4.22)$$

$$z_5 = z_4 + z_3 - z_2, \qquad (4.23)$$

$$z_6 = z_5 + z_4 - z_3, \qquad (4.24)$$

$$y_{i,5} = y_{i,4} + \left(2.\frac{y_{i,4} - y_{i,3}}{z_4 - z_3} - \frac{y_{i,3} - y_{i,2}}{z_3 - z_2} \right).(z_5 - z_4), \qquad (4.25)$$

$$y_{i,6} = y_{i,5} + \left(2.\frac{y_{i,5} - y_{i,4}}{z_5 - z_4} - \frac{y_{i,4} - y_{i,3}}{z_4 - z_3} \right).(z_6 - z_5), \qquad (4.26)$$

onde i = (2,3,4). A Figura 4.4 apresenta os resultados obtidos com as equações descritas anteriormente.

Os pontos das quinas da borda são definidos como sendo a média entre os resultados de (4.13), (4.14), (4.17) e (4.18) com j = (0,1,5,6), e os obtidos a partir de (4.21), (4.22), (4.25) e (4.26) com i = (0,1,5,6). A grade regular totalmente extrapolada em duas linhas e duas colunas é apresentada na Figura 4.5.

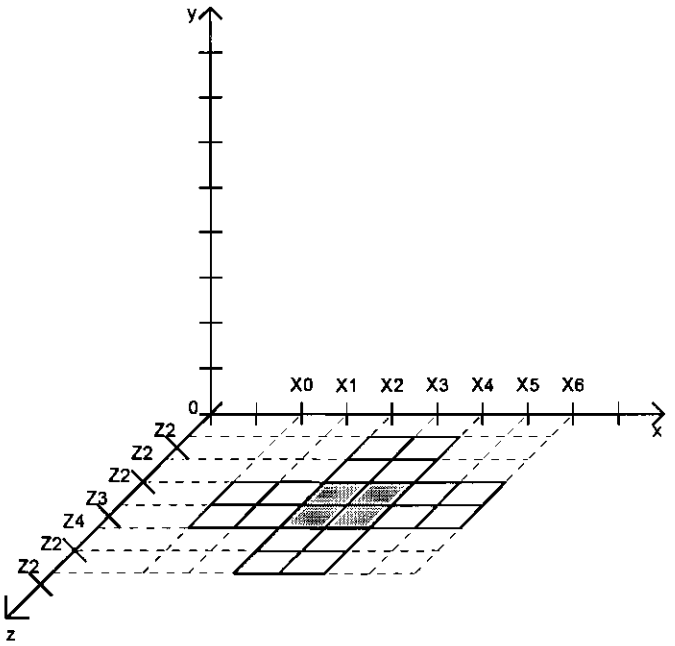


Fig. 4.4 - Grade regular extrapolada em x e em z.

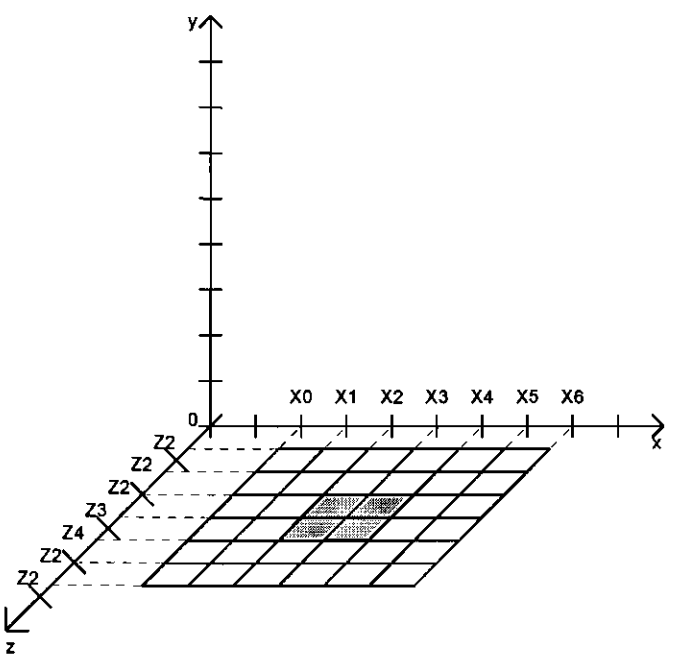


Fig. 4.5 - Grade regular totalmente extrapolada.

4.4 - DETERMINAÇÃO DA FUNÇÃO FRAGMENTADA

A interpolação proposta por Akima (1974a,b) impõe que seja determinada uma função fragmentada $y = y(x,z)$ composta por um conjunto de polinômios bicúbicos, cada qual com grau três no máximo. Cada polinômio expressa uma célula da grade retangular, de modo que a superfície passe pelos quatro pontos que delimitam a célula e que as derivadas parciais nos pontos sejam calculadas conforme o procedimento descrito na seção 4.2.

A Figura 4.6 será usada para exemplificar a determinação do polinômio bicúbico que representa uma célula definida pelos pontos $P_{0,0}$, $P_{1,0}$, $P_{0,1}$ e $P_{1,1}$.

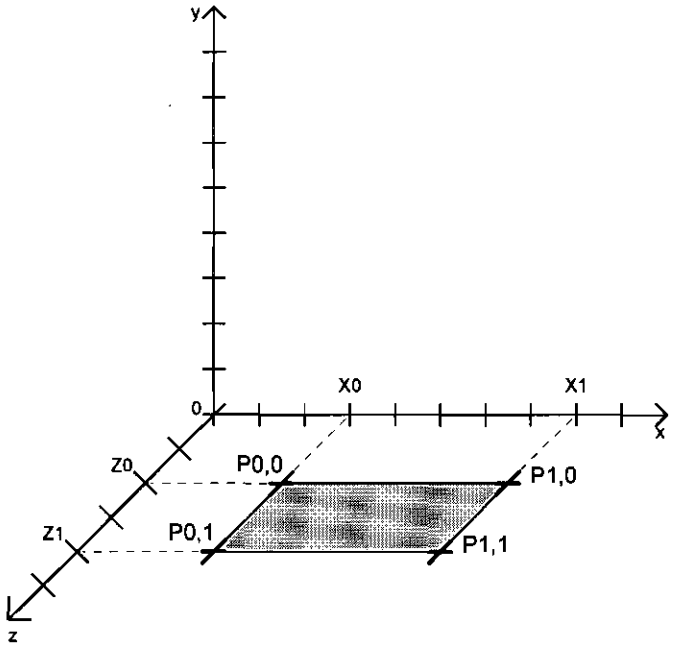


Fig. 4.6 - Célula de uma grade regular.

Dezesseis condições devem ser observadas na determinação do polinômio que representa uma célula da grade. No caso da célula da Figura 4.6, as condições são as seguintes:

$$y = y_{0,0}, \tag{4.27}$$

$$\frac{\partial y}{\partial x} = tx_{0,0}, \tag{4.28}$$

$$\frac{\partial y}{\partial z} = tz_{0,0}, \tag{4.29}$$

$$\frac{\partial^2 y}{\partial x \partial z} = txz_{0,0}, \tag{4.30}$$

no ponto $P_{0,0}$,

$$y = y_{1,0}, \tag{4.31}$$

$$\frac{\partial y}{\partial x} = tx_{1,0}, \tag{4.32}$$

$$\frac{\partial y}{\partial z} = tz_{1,0}, \tag{4.33}$$

$$\frac{\partial^2 y}{\partial x \partial z} = txz_{1,0}, \tag{4.34}$$

no ponto $P_{1,0}$,

$$y = y_{0,1}, \tag{4.35}$$

$$\frac{\partial y}{\partial x} = tx_{0,1}, \tag{4.36}$$

$$\frac{\partial y}{\partial z} = tz_{0,1}, \tag{4.37}$$

$$\frac{\partial^2 y}{\partial x \partial z} = txz_{0,1}, \tag{4.38}$$

no ponto $P_{0,1}$, e

$$y = y_{1,1}\,, \tag{4.39}$$

$$\frac{\partial y}{\partial x} = tx_{1,1}\,, \tag{4.40}$$

$$\frac{\partial y}{\partial z} = tz_{1,1}\,, \tag{4.41}$$

$$\frac{\partial^2 y}{\partial x \partial z} = txz_{1,1}\,, \tag{4.42}$$

no ponto $P_{1,1}$. Sendo tx, tz e txz as derivadas parciais da superfície nestes pontos, pode-se determinar um polinômio bicúbico único de grau três.

Este polinômio pode ser expresso por muitas maneiras diferentes. Uma forma de expressá-lo é a seguinte:

$$y(x,z) = q_0 + q_1(x - x_0) + q_2(x - x_0)^2 + q_3(x - x_0)^3\,, \tag{4.43}$$

onde

$$q_0 = b_{0,0} + b_{0,1}(z - z_0) + b_{0,2}(z - z_0)^2 + b_{0,3}(z - z_0)^3\,, \tag{4.44}$$

$$q_1 = b_{1,0} + b_{1,1}(z - z_0) + b_{1,2}(z - z_0)^2 + b_{1,3}(z - z_0)^3\,, \tag{4.45}$$

$$q_2 = b_{2,0} + b_{2,1}(z - z_0) + b_{2,2}(z - z_0)^2 + b_{2,3}(z - z_0)^3\,, \tag{4.46}$$

$$q_3 = b_{3,0} + b_{3,1}(z - z_0) + b_{3,2}(z - z_0)^2 + b_{3,3}(z - z_0)^3\,, \tag{4.47}$$

e $b_{i,j}$ são os coeficientes obtidos a partir das seguintes equações:

$$b_{0,0} = y_{0,0}\,, \tag{4.48}$$

$$b_{0,1} = tz_{0,0}\,, \tag{4.49}$$

$$b_{0,2} = \frac{2(mz_{0,0} - tz_{0,0}) + mz_{0,0} - tz_{0,1}}{dz}, \tag{4.50}$$

$$b_{0,3} = \frac{-2mz_{0,0} + tz_{0,1} + tz_{0,0}}{dz^2}, \tag{4.51}$$

$$b_{1,0} = tx_{0,0}, \tag{4.52}$$

$$b_{1,1} = txz_{0,0}, \tag{4.53}$$

$$b_{1,2} = \frac{2(tx00dz - txz_{0,0}) + tx00dz - txz_{0,1}}{dz}, \tag{4.54}$$

$$b_{1,3} = \frac{-2tx00dz + txz_{0,1} + txz_{0,0}}{dz^2}, \tag{4.55}$$

$$b_{2,0} = \frac{2(mx_{0,0} - tx_{0,0}) + mx_{0,0} - tx_{1,0}}{dx}, \tag{4.56}$$

$$b_{2,1} = \frac{2(tz00dx - txz_{0,0}) + tz00dx - txz_{1,0}}{dx}, \tag{4.57}$$

$$b_{2,2} = \frac{3(A+E)+D}{dx.dz}, \tag{4.58}$$

$$b_{2,3} = \frac{-3E - B - D}{dx.dz^2}, \tag{4.59}$$

$$b_{3,0} = \frac{-2mx_{0,0} + tx_{1,0} + tx_{0,0}}{dx^2}, \tag{4.60}$$

$$b_{3,1} = \frac{-2tz00dx + txz_{1,0} + txz_{0,0}}{dx^2}, \tag{4.61}$$

$$b_{3,2} = \frac{-3E - C - D}{dz.dx^2}, \tag{4.62}$$

$$b_{3,3} = \frac{D+2E}{dx^2.dz^2}, \tag{4.63}$$

onde tx, tz e txz são as derivadas parciais, mx, mz e mxz são as diferenças divididas, dx é o intervalo da grade na direção x, dz é o intervalo da grade na direção z, e

$$tx00dz = \frac{tx_{0,1} - tx_{0,0}}{dz}, \tag{4.64}$$

$$tx10dz = \frac{tx_{1,1} - tx_{1,0}}{dz}, \tag{4.65}$$

$$tz00dx = \frac{tz_{1,0} - tz_{0,0}}{dx}, \tag{4.66}$$

$$tz10dx = \frac{tz_{1,1} - tz_{0,1}}{dx}, \tag{4.67}$$

$$A = mxz_{0,0} - tx00dz - tz00dx + txz_{0,0}, \tag{4.68}$$

$$B = tx10dz - tx00dz - txz_{1,0} + txz_{0,0}, \tag{4.69}$$

$$C = tz10dx - tz00dx - txz_{0,1} + txz_{0,0}, \tag{4.70}$$

$$D = txz_{1,1} - txz_{1,0} - txz_{0,1} + txz_{0,0}, \tag{4.71}$$

$$E = 2A - B - C. \tag{4.72}$$

4.5 - ANÁLISE DOS RESULTADOS

O método proposto por Akima (1974a,b) foi implementado em um sistema demonstrativo e testado exaustivamente. A listagem dessa implementação se encontra no Apêndice Q.

Os resultados obtidos foram totalmente satisfatórios. As superfícies realmente se comportam de maneira suave e perfeitamente ajustadas sobre os pontos.

A Figura 4.8 apresenta uma comparação entre alguns métodos de interpolação, utilizando uma mesma grade regular escolhida arbitrariamente, conforme a Figura 4.7. A grade refinada por β-Spline, apresentada na Figura 4.8(e), foi gerada com $\beta_{1,x} = \beta_{1,z} = 1$ e $\beta_{2,x} = \beta_{2,z} = 2$.

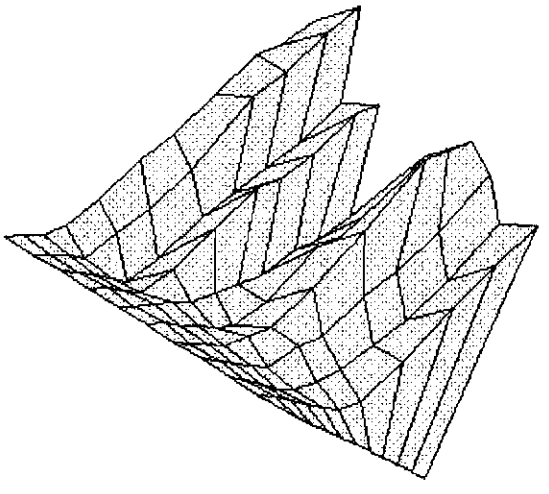
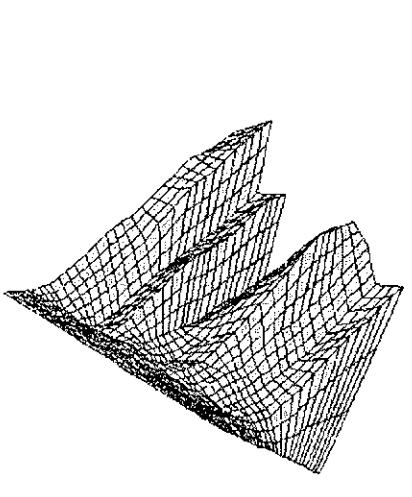
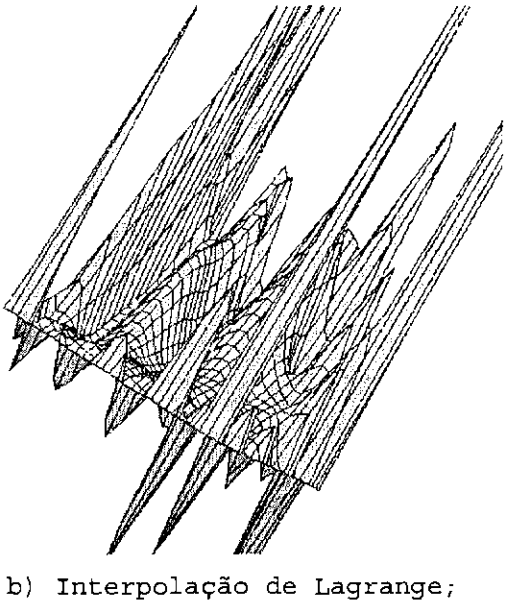


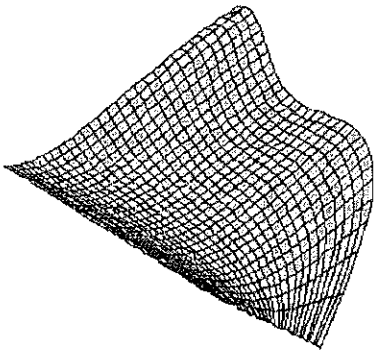
Fig. 4.7 - Grade regular escolhida arbitrariamente.



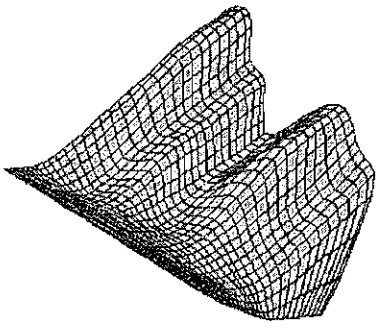
a) Interpolação Linear;



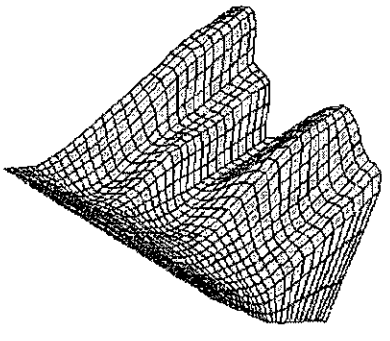
b) Interpolação de Lagrange;



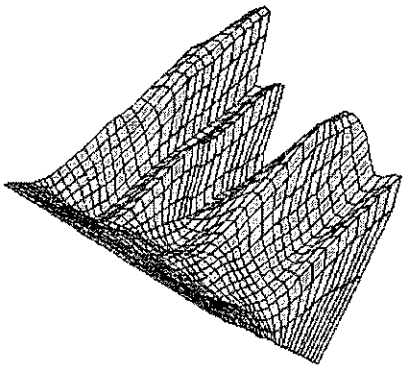
c) Interpolação Bézier;



d) Interpolação B-Spline;



e) Interpolação β-Spline;



f) Interpolação Akima.

Fig. 4.8 - Comparação entre alguns métodos de interpolação sobre superfícies.

Analisando visualmente os gráficos, torna-se evidente que os resultados obtidos com a Interpolação Akima são menos oscilantes e mais próximos dos dados altimétricos do terreno do que os obtidos por outros métodos de interpolação.

CAPÍTULO 5

AVALIAÇÃO DO INTERPOLADOR AKIMA SOBRE MODELOS NUMÉRICOS DE ELEVAÇÃO

5.1 - INTRODUÇÃO

O método de Interpolação Akima para Ajuste de Superfícies Suaves Baseado em Procedimentos Locais (Akima, 1974a) aparentemente apresenta bons resultados na interpolação de superfícies. Entretanto, a sua utilização sobre Modelos Numéricos de Elevação deve ser analisada a fim de permitir uma perfeita avaliação dos erros obtidos com esse método.

As diferenças encontradas entre os dados interpolados e os dados reais de altimetria devem ser mensuradas e os resultados comparados com os alcançados por outros métodos existentes de interpolação de superfícies.

5.2 - INTERPOLAÇÃO SOBRE MODELO NUMÉRICO DE ELEVAÇÃO

Nesta avaliação, os dados reais de altimetria são representados por amostras obtidas nas coordenadas 41° 25′ N e 74° 59′ W, perto de West Point e Hudson Valley, USA. A grade é composta de 31 x 31 pontos com 69,81 metros de espaçamento na direção E-W e 92,51 metros na direção N-S. Nos dados originais, a altitude varia de 0 até 394 metros. A Figura 5.1 apresenta o Modelo Numérico de Elevação correspondente à grade regular original.

A grade original, considerada como terreno real, foi subamostrada, gerando uma nova grade de 11 x 11 pontos com 209,43 metros de espaçamento na direção E-W e

277,53 metros na direção N-S. A Figura 5.2 apresenta a grade regular subamostrada.

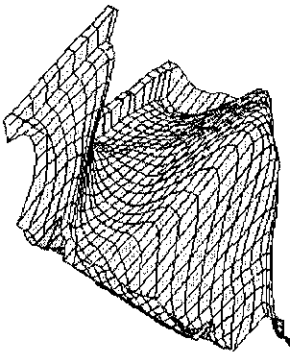


Fig. 5.1 - Grade regular original.

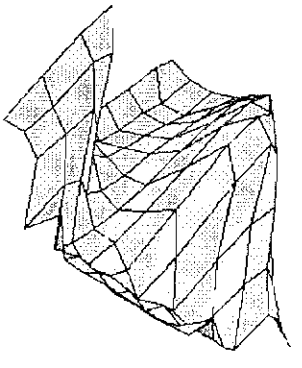


Fig. 5.2 - Grade regular subamostrada.

Os valores altimétricos da grade regular original e da grade regular subamostrada estão descritos nas Tabelas 5.1 e 5.2, respectivamente.

TABELA 5.1 - VALORES ALTIMÉTRICOS DA GRADE REGULAR ORIGINAL (METROS)

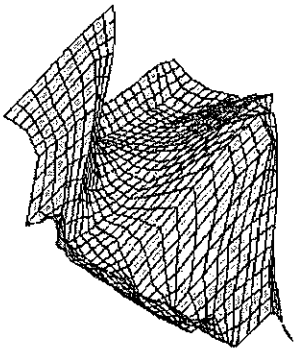
167,00	167,00	160,00	157,00	151,00	144,00	134,00	131,00	124,00	117,00	116,00	109,00	105,00	108,00	111,00	116,00	117,00	131,00	146,00	167,00	174,00	185,00	185,00	167,00	156,00	143,00	129,00	116,00	100,00	83,00	75,00
167,00	162,00	160,00	156,00	151,00	144,00	134,00	134,00	127,00	117,00	112,00	100,00	100,00	100,00	105,00	110,00	117,00	117,00	139,00	151,00	167,00	185,00	185,00	167,00	157,00	143,00	127,00	114,00	98,00	83,00	71,00
164,00	143,00	139,00	137,00	134,00	129,00	124,00	119,00	113,00	106,00	98,00	93,00	90,00	91,00	92,00	95,00	99,00	106,00	116,00	131,00	148,00	161,00	162,00	155,00	143,00	130,00	115,00	102,00	88,00	83,00	63,00
162,00	140,00	137,00	136,00	133,00	129,00	125,00	120,00	113,00	106,00	99,00	94,00	92,00	91,00	92,00	92,00	96,00	101,00	109,00	123,00	140,00	156,00	161,00	155,00	143,00	129,00	115,00	101,00	88,00	83,00	75,00
161,00	138,00	135,00	135,00	133,00	128,00	124,00	119,00	113,00	107,00	101,00	96,00	93,00	92,00	91,00	91,00	92,00	95,00	100,00	111,00	126,00	145,00	154,00	152,00	140,00	126,00	112,00	99,00	86,00	83,00	75,00
161,00	140,00	137,00	135,00	133,00	129,00	125,00	120,00	115,00	109,00	104,00	99,00	95,00	92,00	90,00	89,00	88,00	89,00	92,00	98,00	109,00	126,00	136,00	137,00	129,00	118,00	106,00	94,00	82,00	83,00	71,00
163,00	144,00	141,00	138,00	135,00	131,00	126,00	121,00	116,00	112,00	107,00	103,00	98,00	94,00	90,00	87,00	84,00	82,00	83,00	85,00	91,00	103,00	113,00	118,00	115,00	108,00	99,00	88,00	77,00	74,00	63,00
167,00	148,00	146,00	142,00	139,00	135,00	130,00	125,00	119,00	114,00	110,00	105,00	101,00	95,00	90,00	84,00	80,00	76,00	76,00	76,00	79,00	84,00	90,00	96,00	97,00	95,00	89,00	82,00	71,00	65,00	50,00
169,00	152,00	153,00	150,00	146,00	140,00	135,00	128,00	123,00	118,00	113,00	109,00	103,00	97,00	90,00	83,00	77,00	71,00	69,00	67,00	67,00	70,00	73,00	79,00	82,00	84,00	81,00	74,00	60,00	50,00	16,00
169,00	157,00	159,00	158,00	153,00	147,00	141,00	135,00	128,00	121,00	116,00	111,00	105,00	98,00	90,00	82,00	76,00	69,00	64,00	60,00	59,00	58,00	59,00	63,00	66,00	68,00	66,00	57,00	43,00	16,00	15,00
172,00	161,00	166,00	166,00	161,00	155,00	149,00	142,00	135,00	128,00	122,00	115,00	107,00	98,00	91,00	82,00	75,00	67,00	61,00	55,00	51,00	48,00	46,00	47,00	50,00	49,00	46,00	37,00	26,00	15,00	11,00
167,00	163,00	171,00	173,00	171,00	166,00	159,00	151,00	144,00	136,00	130,00	121,00	111,00	101,00	92,00	83,00	75,00	67,00	60,00	53,00	48,00	42,00	38,00	34,00	34,00	30,00	26,00	19,00	15,00	11,00	10,00
167,00	161,00	171,00	178,00	180,00	177,00	169,00	161,00	154,00	147,00	140,00	130,00	118,00	106,00	95,00	85,00	77,00	67,00	60,00	52,00	47,00	39,00	33,00	26,00	23,00	19,00	16,00	13,00	14,00	11,00	38,00
164,00	157,00	169,00	180,00	185,00	185,00	180,00	173,00	165,00	158,00	149,00	142,00	130,00	116,00	102,00	91,00	81,00	69,00	61,00	53,00	47,00	39,00	31,00	23,00	18,00	15,00	13,00	14,00	17,00	38,00	29,00
154,00	149,00	161,00	174,00	184,00	188,00	185,00	180,00	173,00	167,00	159,00	151,00	140,00	127,00	113,00	100,00	87,00	74,00	64,00	55,00	48,00	39,00	30,00	21,00	15,00	13,00	12,00	15,00	20,00	29,00	26,00
146,00	141,00	153,00	164,00	176,00	184,00	186,00	183,00	179,00	175,00	168,00	161,00	151,00	140,00	126,00	111,00	96,00	81,00	69,00	59,00	50,00	39,00	28,00	19,00	14,00	12,00	13,00	18,00	21,00	26,00	12,00
134,00	131,00	141,00	152,00	164,00	173,00	179,00	180,00	181,00	179,00	175,00	169,00	161,00	150,00	138,00	122,00	105,00	89,00	76,00	64,00	51,00	37,00	25,00	17,00	13,00	14,00	17,00	19,00	18,00	12,00	12,00
129,00	123,00	132,00	141,00	152,00	160,00	168,00	174,00	179,00	181,00	178,00	174,00	168,00	160,00	150,00	134,00	116,00	97,00	83,00	67,00	49,00	32,00	20,00	15,00	15,00	18,00	21,00	20,00	15,00	12,00	12,00
120,00	115,00	123,00	132,00	140,00	147,00	155,00	164,00	172,00	176,00	176,00	174,00	170,00	165,00	157,00	142,00	122,00	100,00	82,00	63,00	43,00	26,00	16,00	16,00	18,00	22,00	21,00	16,00	12,00	12,00	12,00
109,00	106,00	115,00	122,00	129,00	136,00	142,00	152,00	161,00	168,00	170,00	171,00	168,00	165,00	159,00	145,00	124,00	100,00	78,00	55,00	33,00	19,00	12,00	15,00	19,00	21,00	16,00	12,00	10,00	12,00	12,00
95,00	96,00	105,00	113,00	119,00	126,00	133,00	142,00	150,00	158,00	163,00	165,00	164,00	161,00	153,00	137,00	114,00	89,00	64,00	40,00	22,00	14,00	12,00	17,00	19,00	18,00	12,00	10,00	10,00	12,00	12,00
83,00	86,00	94,00	102,00	109,00	117,00	125,00	134,00	143,00	152,00	157,00	159,00	158,00	152,00	141,00	121,00	100,00	76,00	52,00	30,00	16,00	13,00	15,00	18,00	16,00	13,00	10,00	10,00	10,00	12,00	12,00
79,00	76,00	83,00	91,00	99,00	109,00	118,00	127,00	136,00	145,00	151,00	152,00	147,00	137,00	120,00	101,00	81,00	59,00	37,00	19,00	14,00	17,00	19,00	18,00	15,00	12,00	10,00	10,00	10,00	12,00	12,00
66,00	69,00	74,00	82,00	90,00	100,00	109,00	120,00	128,00	135,00	138,00	136,00	126,00	113,00	97,00	82,00	64,00	44,00	27,00	15,00	16,00	20,00	21,00	17,00	12,00	10,00	10,00	10,00	10,00	12,00	12,00
66,00	62,00	67,00	73,00	80,00	90,00	99,00	109,00	116,00	120,00	118,00	113,00	100,00	88,00	77,00	65,00	48,00	30,00	18,00	12,00	17,00	19,00	20,00	13,00	10,00	10,00	10,00	10,00	10,00	12,00	12,00
58,00	57,00	62,00	67,00	73,00	81,00	89,00	97,00	102,00	104,00	98,00	89,00	78,00	69,00	61,00	50,00	35,00	22,00	15,00	12,00	16,00	17,00	17,00	12,00	10,00	10,00	10,00	10,00	10,00	12,00	12,00
50,00	50,00	55,00	60,00	65,00	72,00	79,00	85,00	89,00	89,00	84,00	75,00	66,00	58,00	50,00	37,00	26,00	18,00	14,00	12,00	16,00	15,00	15,00	10,00	10,00	10,00	10,00	10,00	10,00	12,00	12,00
47,00	45,00	49,00	53,00	58,00	65,00	71,00	75,00	78,00	79,00	77,00	69,00	60,00	50,00	39,00	29,00	21,00	17,00	15,00	13,00	15,00	14,00	14,00	10,00	10,00	10,00	10,00	10,00	10,00	12,00	12,00
42,00	41,00	44,00	47,00	51,00	57,00	62,00	67,00	68,00	69,00	68,00	61,00	52,00	39,00	30,00	22,00	19,00	16,00	15,00	13,00	14,00	13,00	12,00	10,00	10,00	10,00	10,00	10,00	10,00	12,00	12,00
40,00	38,00	40,00	43,00	46,00	51,00	54,00	57,00	58,00	58,00	56,00	48,00	38,00	27,00	21,00	17,00	15,00	15,00	14,00	14,00	14,00	13,00	12,00	10,00	10,00	10,00	10,00	10,00	10,00	12,00	12,00
37,00	41,00	42,00	42,00	47,00	52,00	50,00	50,00	50,00	50,00	50,00	33,00	16,00	16,00	16,00	15,00	15,00	16,00	16,00	16,00	16,00	29,00	12,00	12,00	12,00	12,00	12,00	12,00	12,00	12,00	12,00

TABELA 5.2 - VALORES ALTIMÉTRICOS DA GRADE REGULAR SUBAMOSTRADA (METROS)

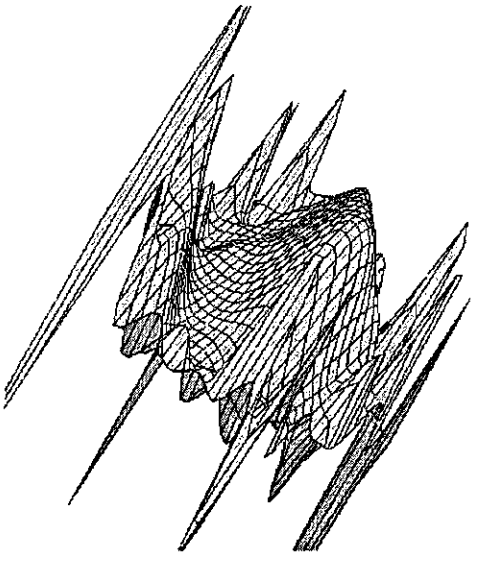
167,00	157,00	134,00	117,00	105,00	116,00	146,00	185,00	156,00	116,00	75,00
162,00	136,00	125,00	106,00	92,00	92,00	109,00	156,00	143,00	101,00	75,00
163,00	138,00	126,00	112,00	98,00	87,00	83,00	103,00	115,00	88,00	63,00
169,00	158,00	141,00	121,00	105,00	82,00	64,00	58,00	66,00	57,00	15,00
167,00	178,00	169,00	147,00	118,00	85,00	60,00	39,00	23,00	13,00	38,00
146,00	164,00	186,00	175,00	151,00	111,00	69,00	39,00	14,00	18,00	12,00
120,00	132,00	155,00	176,00	170,00	142,00	82,00	26,00	18,00	16,00	12,00
83,00	102,00	125,00	152,00	158,00	121,00	52,00	13,00	16,00	10,00	12,00
66,00	73,00	99,00	120,00	100,00	65,00	18,00	19,00	10,00	10,00	12,00
47,00	53,00	71,00	79,00	60,00	29,00	15,00	14,00	10,00	10,00	12,00
37,00	42,00	50,00	50,00	16,00	15,00	16,00	29,00	12,00	12,00	12,00

Através dos dados de altimetria da grade regular subamostrada, pode-se reconstruir a grade regular original. Este refinamento será feito utilizando-se os diversos métodos de interpolação sobre superfícies.

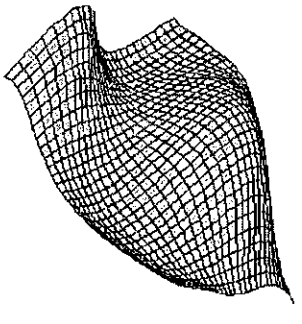
A Figura 5.3 apresenta os resultados obtidos com alguns interpoladores. A grade refinada por β-Spline, apresentada na Figura 5.3(e), foi gerada com $\beta_{1,x} = \beta_{1,z} = 1$ e $\beta_{2,x} = \beta_{2,z} = 2$.



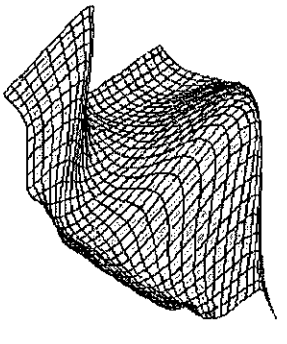
a) Interpolação Linear;



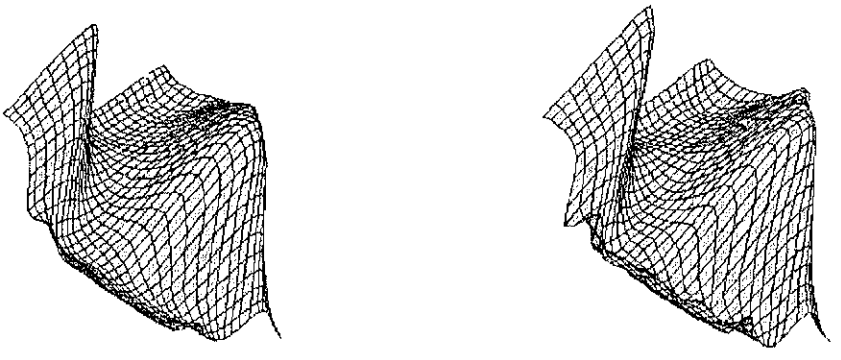
b) Interpolação de Lagrange;



c) Interpolação Bézier;



d) Interpolação B-Spline;



e) Interpolação β-Spline; f) Interpolação Akima.

Fig. 5.3 - Grades refinadas obtidas com alguns métodos de interpolação sobre superfícies.

Similarmente à grade regular original, as grades refinadas por interpolação são compostas de 31 x 31 pontos com 69,81 metros de espaçamento na direção E-W e 92,51 metros na direção N-S. As Tabelas 5.3, 5.4, 5.5, 5.6, 5.7 e 5.8 apresentam os valores altimétricos das grades refinadas por Interpolação Linear, Lagrange, Bézier, B-Spline, β-Spline e Akima, respectivamente.

5.3 - SUPERFÍCIES DE ERRO

As diferenças entre os resultados da interpolação e os dados altimétricos reais podem ser verificadas através de superfícies de erro. Estas superfícies correspondem às diferenças encontradas entre as grades refinadas e a superfície real. A Figura 5.4 apresenta as superfícies de erro correspondentes aos resultados demonstrados na Figura 5.3.

TABELA 5.3 - VALORES ALTIMÉTRICOS DA GRADE REFINADA POR INTERPOLAÇÃO LINEAR (METROS)

167,00	163,67	160,33	157,00	149,33	141,67	134,00	128,33	122,67	117,00	113,00	109,00	105,00	108,67	112,33	116,00	126,00	136,00	146,00	159,00	172,00	185,00	175,33	165,67	156,00	142,67	129,33	116,00	102,33	88,67	75,00
165,33	160,22	155,11	150,00	143,67	137,33	131,00	125,11	119,22	113,33	109,11	104,89	100,67	103,11	105,56	108,00	116,56	125,11	133,67	147,56	161,44	175,33	167,44	159,56	151,67	138,11	124,56	111,00	99,00	87,00	75,00
163,67	156,78	149,89	143,00	138,00	133,00	128,00	121,89	115,78	109,67	105,22	100,78	96,33	97,56	98,78	100,00	107,11	114,22	121,33	136,11	150,89	165,67	159,56	153,44	147,33	133,56	119,78	106,00	95,67	85,33	75,00
162,00	153,33	144,67	136,00	132,33	128,67	125,00	118,67	112,33	106,00	101,33	96,67	92,00	92,00	92,00	92,00	97,67	103,33	109,00	124,67	140,33	156,00	151,67	147,33	143,00	129,00	115,00	101,00	92,33	83,67	75,00
162,33	153,78	145,22	136,67	132,89	129,11	125,33	119,56	113,78	108,00	103,33	98,67	94,00	92,78	91,56	90,33	93,67	97,00	100,33	113,00	125,67	138,33	136,78	135,22	133,67	121,33	109,00	96,67	88,11	79,56	71,00
162,67	154,22	145,78	137,33	133,44	129,56	125,67	120,44	115,22	110,00	105,33	100,67	96,00	93,56	91,11	88,67	89,67	90,67	91,67	101,33	111,00	120,67	121,89	123,11	124,33	113,67	103,00	92,33	83,89	75,44	67,00
163,00	154,67	146,33	138,00	134,00	130,00	126,00	121,33	116,67	112,00	107,33	102,67	98,00	94,33	90,67	87,00	85,67	84,33	83,00	89,67	96,33	103,00	107,00	111,00	115,00	105,00	97,00	88,00	79,67	71,33	63,00
165,00	158,22	151,44	144,67	140,11	135,56	131,00	125,67	120,33	115,00	110,11	105,22	100,33	95,33	90,33	85,33	82,44	79,56	76,67	80,44	84,22	88,00	91,56	95,11	98,67	91,67	84,67	77,67	67,44	57,22	47,00
167,00	161,78	156,56	151,33	146,22	141,11	136,00	130,00	124,00	118,00	112,89	107,78	102,67	96,33	90,00	83,67	79,22	74,78	70,33	71,22	72,11	73,00	76,11	79,22	82,33	77,33	72,33	67,33	55,22	43,11	31,00
169,00	165,33	161,67	158,00	152,33	146,67	141,00	134,33	127,67	121,00	115,67	110,33	105,00	97,33	89,67	82,00	76,00	70,00	64,00	62,00	60,00	58,00	60,67	63,33	66,00	63,00	60,00	57,00	43,00	29,00	15,00
168,33	167,11	165,89	164,67	159,89	155,11	150,33	143,44	136,56	129,67	122,89	116,11	109,33	100,56	91,78	83,00	76,22	69,44	62,67	59,00	55,33	51,67	51,67	51,67	51,67	48,56	45,44	42,33	35,78	29,22	22,67
167,67	168,89	170,11	171,33	167,44	163,56	159,67	152,56	145,44	138,33	130,11	121,89	113,67	103,78	93,89	84,00	76,44	68,89	61,33	56,00	50,67	45,33	42,67	40,00	37,33	34,11	30,89	27,67	28,56	29,44	30,33
167,00	170,67	174,33	178,00	175,00	172,00	169,00	161,67	154,33	147,00	137,33	127,67	118,00	107,00	96,00	85,00	76,67	68,33	60,00	53,00	46,00	39,00	33,67	28,33	23,00	19,67	16,33	13,00	21,33	29,67	38,00
160,00	164,44	168,89	173,33	173,78	174,22	174,67	168,56	162,44	156,33	147,22	138,11	129,00	117,22	105,44	93,67	83,44	73,22	63,00	55,00	47,00	39,00	32,67	26,33	20,00	18,22	16,44	14,67	19,56	24,44	29,33
153,00	158,22	163,44	168,67	172,56	176,44	180,33	175,44	170,56	165,67	157,11	148,56	140,00	127,44	114,89	102,33	90,22	78,11	66,00	57,00	48,00	39,00	31,67	24,33	17,00	16,78	16,56	16,33	17,78	19,22	20,67
146,00	152,00	158,00	164,00	171,33	178,67	186,00	182,33	178,67	175,00	167,00	159,00	151,00	137,67	124,33	111,00	97,00	83,00	69,00	59,00	49,00	39,00	30,67	22,33	14,00	15,33	16,67	18,00	16,00	14,00	12,00
137,33	142,67	148,00	153,33	160,78	168,22	175,67	175,56	175,44	175,33	169,33	163,33	157,33	145,33	133,33	121,33	105,33	89,33	73,33	60,44	47,56	34,67	28,22	21,78	15,33	16,00	16,67	17,33	15,56	13,78	12,00
128,67	133,33	138,00	142,67	150,22	157,78	165,33	168,78	172,22	175,67	171,67	167,67	163,67	153,00	142,33	131,67	113,67	95,67	77,67	61,89	46,11	30,33	25,78	21,22	16,67	16,67	16,67	16,67	15,11	13,56	12,00
120,00	124,00	128,00	132,00	139,67	147,33	155,00	162,00	169,00	176,00	174,00	172,00	170,00	160,67	151,33	142,00	122,00	102,00	82,00	63,33	44,67	26,00	23,33	20,67	18,00	17,33	16,67	16,00	14,67	13,33	12,00
107,67	112,44	117,22	122,00	129,67	137,33	145,00	152,67	160,33	168,00	167,33	166,67	166,00	155,67	145,33	135,00	114,00	93,00	72,00	55,22	38,44	21,67	20,22	18,78	17,33	16,22	15,11	14,00	13,33	12,67	12,00
95,33	100,89	106,44	112,00	119,67	127,33	135,00	143,33	151,67	160,00	160,67	161,33	162,00	150,67	139,33	128,00	106,00	84,00	62,00	47,11	32,22	17,33	17,11	16,89	16,67	15,11	13,56	12,00	12,00	12,00	12,00
83,00	89,33	95,67	102,00	109,67	117,33	125,00	134,00	143,00	152,00	154,00	156,00	158,00	145,67	133,33	121,00	98,00	75,00	52,00	39,00	26,00	13,00	14,00	15,00	16,00	14,00	12,00	10,00	10,67	11,33	12,00
77,33	82,33	87,33	92,33	100,33	108,33	116,33	124,67	133,00	141,33	140,44	139,56	138,67	126,56	114,44	102,33	81,78	61,22	40,67	32,11	23,56	15,00	14,67	14,33	14,00	12,67	11,33	10,00	10,67	11,33	12,00
71,67	75,33	79,00	82,67	91,00	99,33	107,67	115,33	123,00	130,67	126,89	123,11	119,33	107,44	95,56	83,67	65,56	47,44	29,33	25,22	21,11	17,00	15,33	13,67	12,00	11,33	10,67	10,00	10,67	11,33	12,00
66,00	68,33	70,67	73,00	81,67	90,33	99,00	106,00	113,00	120,00	113,33	106,67	100,00	88,33	76,67	65,00	49,33	33,67	18,00	18,33	18,67	19,00	16,00	13,00	10,00	10,00	10,00	10,00	10,67	11,33	12,00
59,67	61,89	64,11	66,33	74,11	81,89	89,67	95,22	100,78	106,33	99,78	93,22	86,67	75,44	64,22	53,00	41,00	29,00	17,00	17,11	17,22	17,33	14,89	12,44	10,00	10,00	10,00	10,00	10,67	11,33	12,00
53,33	55,44	57,56	59,67	66,56	73,44	80,33	84,44	88,56	92,67	86,22	79,78	73,33	62,56	51,78	41,00	32,67	24,33	16,00	15,89	15,78	15,67	13,78	11,89	10,00	10,00	10,00	10,00	10,67	11,33	12,00
47,00	49,00	51,00	53,00	59,00	65,00	71,00	73,67	76,33	79,00	72,67	66,33	60,00	49,67	39,33	29,00	24,33	19,67	15,00	14,67	14,33	14,00	12,67	11,33	10,00	10,00	10,00	10,00	10,67	11,33	12,00
43,67	45,56	47,44	49,33	54,22	59,11	64,00	65,78	67,56	69,33	61,33	53,33	45,33	38,33	31,33	24,33	21,33	18,33	15,33	16,56	17,78	19,00	16,22	13,44	10,67	10,67	10,67	10,67	11,11	11,56	12,00
40,33	42,11	43,89	45,67	49,44	53,22	57,00	57,89	58,78	59,67	50,00	40,33	30,67	27,00	23,33	19,67	18,33	17,00	15,67	18,44	21,22	24,00	19,78	15,56	11,33	11,33	11,33	11,33	11,56	11,78	12,00
37,00	38,67	40,33	42,00	44,67	47,33	50,00	50,00	50,00	50,00	38,67	27,33	16,00	15,67	15,33	15,00	15,33	15,67	16,00	20,33	24,67	29,00	23,33	17,67	12,00	12,00	12,00	12,00	12,00	12,00	12,00

TABELA 5.4 - VALORES ALTIMÉTRICOS DA GRADE REFINADA POR INTERPOLAÇÃO LAGRANGE (METROS)

167,00	212,81	186,43	157,00	141,29	136,02	134,00	130,50	124,39	117,00	110,40	106,20	105,00	106,58	110,37	116,00	123,56	133,48	146,00	160,45	174,68	185,00	186,97	177,23	156,00	129,61	111,06	116,00	149,53	177,68	75,00
131,58	-126,30	-19,44	127,58	206,36	212,67	179,28	140,24	116,08	111,67	120,54	131,57	134,98	126,27	107,49	85,98	71,41	72,05	91,40	126,25	166,91	199,84	212,55	199,66	168,29	139,99	145,23	205,45	296,41	285,10	-168,48
146,61	27,66	67,26	130,07	165,76	168,68	151,48	129,56	113,50	106,96	107,90	111,28	112,07	107,48	98,07	87,52	81,24	84,33	99,36	124,85	154,80	179,99	190,88	181,96	156,19	127,76	119,98	153,86	221,40	235,60	-52,71
162,00	187,35	159,55	136,00	126,72	125,56	125,00	121,20	114,17	106,00	98,95	94,29	92,00	91,28	91,32	92,00	94,17	99,36	109,00	123,29	140,37	156,00	164,45	160,58	143,00	117,26	97,19	101,00	137,16	172,82	75,00
167,04	239,50	190,17	138,03	113,39	110,65	115,92	118,89	115,71	107,60	98,05	90,44	86,56	86,46	88,93	92,54	96,43	100,76	106,46	114,51	124,90	135,86	143,76	144,04	133,42	112,90	90,29	79,96	95,88	132,17	122,93
165,44	213,92	176,33	137,57	118,96	116,16	119,05	120,20	116,85	109,92	101,91	95,20	91,04	89,39	89,41	90,10	91,02	92,51	95,52	101,03	109,25	118,94	127,25	130,38	125,15	111,28	93,43	81,55	86,65	107,91	105,37
163,00	167,29	151,32	138,00	131,23	128,33	126,00	122,47	117,56	112,00	106,65	101,99	98,00	94,33	90,66	87,00	83,86	82,17	83,00	87,04	94,14	103,00	111,25	116,02	115,00	107,80	97,01	88,00	85,84	87,06	63,00
162,99	136,81	137,01	141,61	142,47	138,61	131,95	124,78	118,63	113,97	110,36	106,94	102,84	97,58	91,25	84,55	78,55	74,46	73,22	75,26	80,23	87,02	93,97	99,22	101,30	99,56	94,46	87,16	77,91	62,76	27,54
165,57	132,18	138,90	148,71	150,60	145,14	136,34	127,76	121,18	116,65	113,18	109,50	104,64	98,29	90,79	82,97	75,85	70,39	67,22	66,55	68,17	71,55	75,97	80,58	84,38	86,07	83,90	75,72	59,65	36,13	12,22
169,00	144,47	151,21	158,00	157,06	150,10	141,00	132,64	126,08	121,00	116,38	111,22	105,00	97,72	89,83	82,00	74,84	68,78	64,00	60,56	58,51	58,00	59,22	62,14	66,00	68,82	67,18	57,00	36,53	12,95	15,00
171,15	158,92	164,45	167,32	163,85	156,37	147,91	140,23	133,67	127,63	121,34	114,25	106,32	97,89	89,50	81,62	74,48	68,03	62,10	56,57	51,62	47,78	45,82	46,20	48,44	50,38	48,07	37,10	16,78	-0,75	26,23
170,59	164,72	171,40	174,50	171,68	165,21	157,66	150,39	143,53	136,53	128,78	120,02	110,45	100,59	91,06	82,32	74,50	67,42	60,71	54,07	47,53	41,57	36,95	34,36	33,70	33,44	30,52	21,49	6,53	-2,06	35,88
167,00	159,26	169,80	178,00	179,34	175,43	169,00	161,87	154,63	147,00	138,48	128,78	118,00	106,66	95,45	85,00	75,66	67,43	60,00	52,94	45,94	39,00	32,50	27,03	23,00	20,17	17,36	13,00	7,28	7,44	38,00
161,03	147,23	162,03	177,14	184,33	184,06	179,31	172,68	165,47	157,84	149,36	139,57	128,31	115,91	103,05	90,57	79,18	69,27	60,77	53,27	46,14	38,84	31,18	23,48	16,62	11,75	9,88	11,40	15,83	22,36	32,07
153,74	136,50	152,82	172,18	184,11	187,76	185,67	180,65	174,44	167,62	159,91	150,79	139,89	127,27	113,49	99,42	86,02	74,04	63,78	55,04	47,18	39,39	31,06	22,25	13,99	8,31	7,78	14,19	26,20	35,56	21,68
146,00	133,24	146,33	164,00	177,37	184,37	186,00	184,10	180,18	175,00	168,63	160,76	151,00	139,22	125,67	111,00	96,09	81,86	69,00	57,77	47,95	39,00	30,31	21,73	14,00	9,06	9,67	18,00	32,30	40,87	12,00
138,10	138,49	144,01	153,84	164,75	173,91	179,84	182,32	181,82	179,03	174,45	168,23	160,22	150,17	137,94	123,67	107,89	91,43	75,30	60,42	47,45	36,61	27,78	20,68	15,37	12,54	13,65	20,07	30,48	35,39	6,98
129,67	147,51	143,94	142,89	148,78	158,71	168,60	175,74	179,19	179,31	176,90	172,63	166,68	158,75	148,32	134,94	118,65	100,08	80,50	61,56	44,97	32,04	23,37	18,67	16,96	16,98	17,81	19,20	21,04	20,83	7,68
120,00	152,26	142,06	132,00	132,96	142,53	155,00	165,83	172,88	176,00	176,05	173,93	170,00	163,90	154,84	142,00	125,06	104,55	82,00	59,71	40,29	26,00	18,03	16,04	18,00	20,66	20,62	16,00	8,21	3,36	12,00
108,60	146,04	134,57	121,60	120,28	129,02	142,10	154,63	164,04	169,76	172,31	172,34	170,00	164,82	155,92	142,48	124,29	102,09	77,74	53,97	33,81	19,86	13,39	13,75	18,15	22,14	21,04	12,27	-1,38	-8,16	16,02
95,79	128,36	120,80	111,69	111,77	119,99	132,02	144,10	154,15	161,47	166,02	167,76	166,27	160,82	150,61	135,26	115,17	91,71	67,20	44,51	26,49	15,15	11,05	12,85	17,42	20,56	18,50	9,97	-1,39	-5,17	16,27
83,00	107,61	104,81	102,00	105,66	114,34	125,00	135,44	144,58	152,00	157,27	159,63	158,00	151,30	138,89	121,00	98,93	74,96	52,00	32,87	19,65	13,00	11,91	13,94	16,00	15,73	12,87	10,00	11,26	17,06	12,00
72,56	99,05	94,64	92,28	97,84	108,13	119,14	128,64	136,17	141,91	145,79	146,99	144,17	135,97	121,75	102,14	79,16	55,91	35,85	21,72	14,67	13,66	15,65	16,76	14,15	8,31	4,53	11,53	34,22	56,27	6,26
66,72	116,95	98,28	82,49	84,00	96,62	111,28	122,47	128,82	131,34	131,35	129,27	124,36	115,29	101,16	82,31	60,75	39,86	23,36	14,06	12,48	16,25	20,56	20,08	12,07	-0,13	-5,32	12,27	59,32	101,78	4,74
66,00	161,51	116,63	73,00	63,19	77,73	99,00	114,90	121,37	120,00	114,40	107,43	100,00	91,27	79,82	65,00	47,80	30,95	18,00	11,85	13,08	19,00	24,05	22,02	10,00	-7,13	-14,22	10,00	74,91	135,66	12,00
67,53	205,22	135,97	64,49	41,60	56,21	83,28	104,22	111,77	107,47	96,86	85,30	75,69	68,07	60,75	51,88	40,93	29,21	19,46	14,43	15,21	19,93	23,81	20,98	8,38	-9,60	-18,06	4,93	70,84	137,68	25,98
64,24	188,23	124,96	57,70	34,16	45,87	70,45	90,33	97,84	93,60	82,49	69,92	59,41	51,96	46,54	41,42	35,36	28,41	21,88	17,55	16,46	17,95	19,29	16,76	8,02	-4,58	-11,58	1,91	45,29	94,13	33,44
47,00	39,70	45,47	53,00	59,63	65,52	71,00	75,72	78,73	79,00	75,86	69,29	60,00	49,23	38,44	29,00	21,82	17,25	15,00	14,32	14,27	14,00	13,07	11,56	10,00	9,07	9,18	10,00	10,56	10,28	12,00
12,71	-248,63	-108,51	49,87	122,34	121,82	91,38	65,17	57,35	65,37	78,30	84,89	78,84	60,38	35,07	10,69	-6,04	-11,96	-8,30	0,29	8,20	11,64	10,79	9,91	14,72	27,28	40,36	35,55	-7,07	-75,65	-52,21
-17,36	-458,90	-222,94	46,81	171,12	169,02	112,96	62,39	43,86	55,16	78,71	95,00	91,85	67,82	30,79	-6,69	-32,92	-41,42	-32,93	-14,40	4,48	15,92	18,00	15,94	19,28	34,72	56,91	62,59	18,52	-80,22	-117,13
37,00	158,87	101,94	42,00	21,96	31,78	50,00	61,28	60,44	50,00	35,87	23,54	16,00	13,36	13,79	15,00	15,55	15,47	16,00	18,59	23,52	29,00	31,20	25,96	12,00	-5,29	-11,69	12,00	73,47	129,82	12,00

TABELA 5.5 - VALORES ALTIMÉTRICOS DA GRADE REFINADA POR INTERPOLAÇÃO BÉZIER (METROS)

167,00	163,10	158,34	153,09	147,67	142,35	137,37	132,94	129,25	126,45	124,66	123,93	124,29	125,68	128,00	131,06	134,62	138,40	142,03	145,13	147,33	148,23	147,50	144,87	140,15	133,31	124,43	113,75	101,64	88,56	75,00
165,63	160,57	155,18	149,64	144,11	138,76	133,75	129,26	125,43	122,39	120,27	119,13	119,00	119,85	121,61	124,14	127,23	130,61	133,96	136,95	139,19	140,30	139,95	137,85	133,79	127,70	119,64	109,83	98,66	86,62	74,32
164,79	158,99	153,22	147,51	141,93	136,57	131,55	126,98	122,99	119,71	117,22	115,62	114,93	115,16	116,24	118,08	120,50	123,29	126,15	128,78	130,83	131,95	131,79	130,07	126,56	121,16	113,90	104,94	94,64	83,52	72,18
164,36	158,21	152,29	146,54	140,98	135,65	130,61	125,97	121,81	118,24	115,36	113,24	111,94	111,46	111,77	112,81	114,43	116,45	118,64	120,72	122,38	123,29	123,14	121,64	118,56	113,76	107,22	99,07	89,60	79,28	68,72
164,21	158,05	152,20	146,56	141,10	135,83	130,80	126,07	121,72	117,86	114,55	111,89	109,92	108,68	108,15	108,29	109,00	110,14	111,51	112,88	113,98	114,52	114,21	112,77	109,97	105,65	99,76	92,36	83,69	74,14	64,26
164,13	158,29	152,73	147,34	142,07	136,92	131,92	127,12	122,60	118,43	114,69	111,46	108,81	106,77	105,36	104,54	104,27	104,43	104,88	105,41	105,82	105,84	105,22	103,69	101,04	97,09	91,76	85,07	77,18	68,40	59,23
163,96	158,69	153,61	148,62	143,65	138,70	133,79	128,96	124,28	119,82	115,66	111,87	108,53	105,69	103,37	101,57	100,28	99,40	98,85	98,46	98,07	97,45	96,39	94,65	92,02	88,33	83,48	77,45	70,33	62,38	53,97
163,52	159,02	154,59	150,13	145,58	140,92	136,17	131,38	126,59	121,88	117,33	113,02	109,01	105,38	102,16	99,38	97,05	95,11	93,52	92,15	90,88	89,53	87,92	85,86	83,15	79,63	75,19	69,77	63,42	56,32	48,78
162,64	159,07	155,43	151,63	147,60	143,34	138,84	134,16	129,34	124,45	119,57	114,78	110,15	105,77	101,68	97,95	94,58	91,59	88,94	86,56	84,36	82,23	80,00	77,53	74,66	71,22	67,11	62,26	56,67	50,46	43,86
161,19	158,63	155,89	152,86	149,46	145,69	141,56	137,09	132,33	127,34	122,21	117,02	111,84	106,76	101,86	97,21	92,86	88,83	85,13	81,73	78,60	75,64	72,76	69,82	66,71	63,29	59,45	55,11	50,26	44,94	39,32
159,05	157,56	155,78	153,59	150,92	147,76	144,08	139,93	135,33	130,36	125,07	119,56	113,92	108,23	102,59	97,09	91,80	86,78	82,07	77,68	73,61	69,83	66,26	62,83	59,44	55,98	52,36	48,48	44,31	39,85	35,20
156,15	155,72	154,92	153,64	151,78	149,30	146,19	142,46	138,14	133,28	127,95	122,23	116,21	110,01	103,72	97,45	91,29	85,34	79,68	74,35	69,38	64,79	60,55	56,62	52,93	49,40	45,93	42,46	38,91	35,25	31,52
152,45	153,02	153,21	152,85	151,85	150,14	147,68	144,47	140,53	135,89	130,62	124,80	118,53	111,91	105,07	98,12	91,19	84,40	77,85	71,65	65,84	60,49	55,62	51,20	47,20	43,57	40,24	37,10	34,10	31,15	28,25
147,94	149,44	150,56	151,13	151,02	150,12	148,38	145,78	142,31	137,99	132,89	127,08	120,65	113,72	106,43	98,91	91,31	83,77	76,45	69,45	62,89	56,85	51,39	46,52	42,25	38,52	35,28	32,43	29,89	27,55	25,37
142,65	144,96	146,95	148,42	149,18	149,12	148,15	146,21	143,29	139,38	134,54	128,83	122,35	115,22	107,59	99,60	91,44	83,28	75,28	67,60	60,39	53,75	47,78	42,53	38,01	34,20	31,03	28,42	26,26	24,43	22,82
136,62	139,63	142,40	144,69	146,31	147,08	146,88	145,65	143,31	139,88	135,37	129,85	123,41	116,18	108,31	99,99	91,39	82,71	74,16	65,93	58,18	51,06	44,69	39,13	34,42	30,55	27,46	25,04	23,19	21,75	20,60
129,95	133,52	136,96	140,00	142,40	143,96	144,53	143,99	142,28	139,35	135,22	129,95	123,63	116,39	108,40	99,84	90,92	81,87	72,92	64,26	56,11	48,64	41,97	36,21	31,38	27,49	24,47	22,23	20,62	19,49	18,68
122,75	126,74	130,73	134,41	137,51	139,79	141,07	141,20	140,08	137,67	133,95	128,97	122,83	115,66	107,63	98,96	89,86	80,57	71,35	62,44	54,04	46,35	39,52	33,65	28,79	24,93	22,01	19,92	18,50	17,60	17,04
115,14	119,41	123,82	128,02	131,71	134,62	136,53	137,27	136,70	134,77	131,45	126,78	120,86	113,83	105,86	97,18	88,02	78,65	69,33	60,31	51,82	44,07	37,22	31,36	26,55	22,78	19,98	18,03	16,78	16,05	15,67
107,25	111,69	116,38	120,97	125,13	128,55	130,98	132,22	132,13	130,62	127,67	123,31	117,64	110,79	102,97	94,39	85,31	76,00	66,73	57,78	49,37	41,71	34,96	29,23	24,57	20,95	18,30	16,51	15,40	14,81	14,57
99,24	103,71	108,57	113,42	117,90	121,69	124,51	126,14	126,42	125,25	122,61	118,54	113,12	106,50	98,89	90,52	81,64	72,54	63,50	54,77	46,60	39,19	32,70	27,22	22,78	19,38	16,92	15,28	14,31	13,83	13,71
91,25	95,66	100,55	105,52	110,18	114,19	117,24	119,12	119,64	118,72	116,33	112,49	107,32	100,97	93,64	85,57	77,02	68,28	59,61	51,28	43,52	36,50	30,39	25,26	21,14	18,00	15,76	14,29	13,46	13,09	13,07
83,42	87,67	92,48	97,43	102,12	106,19	109,33	111,30	111,93	111,15	108,91	105,27	100,34	94,27	87,29	79,61	71,52	63,27	55,14	47,35	40,14	33,66	28,05	23,36	19,63	16,80	14,80	13,51	12,80	12,54	12,62
75,88	79,88	84,51	89,32	93,89	97,86	100,92	102,83	103,44	102,68	100,51	97,02	92,32	86,56	79,98	72,80	65,27	57,65	50,18	43,08	36,55	30,72	25,70	21,54	18,24	15,75	14,01	12,90	12,32	12,16	12,32
68,72	72,42	76,77	81,32	85,64	89,36	92,18	93,90	94,36	93,50	91,34	87,95	83,47	78,06	71,93	65,33	58,47	51,59	44,91	38,62	32,87	27,79	23,43	19,84	17,00	14,87	13,38	12,44	11,98	11,91	12,15
62,03	65,39	69,39	73,56	77,50	80,84	83,29	84,68	84,87	83,84	81,62	78,31	74,05	69,02	63,42	57,46	51,37	45,34	39,55	34,15	29,27	24,98	21,33	18,33	15,95	14,17	12,91	12,13	11,76	11,76	12,06
55,86	58,85	62,44	66,17	69,61	72,45	74,42	75,37	75,21	73,94	71,63	68,39	64,37	59,76	54,74	49,52	44,27	39,16	34,33	29,89	25,92	22,46	19,52	17,10	15,16	13,69	12,63	11,97	11,66	11,70	12,02
50,24	52,87	56,01	59,21	62,09	64,33	65,73	66,16	65,59	64,05	61,63	58,47	54,73	50,59	46,23	41,80	37,47	33,35	29,53	26,07	23,02	20,37	18,12	16,23	14,69	13,48	12,56	11,96	11,67	11,70	12,00
45,19	47,49	50,17	52,80	55,03	56,61	57,38	57,24	56,23	54,41	51,90	48,85	45,45	41,84	38,19	34,63	31,26	28,16	25,37	22,91	20,75	18,88	17,25	15,84	14,62	13,57	12,73	12,12	11,79	11,76	12,00
40,74	42,76	44,97	47,00	48,56	49,44	49,52	48,81	47,34	45,25	42,68	39,80	36,78	33,76	30,89	28,25	25,89	23,84	22,08	20,58	19,28	18,12	17,04	16,00	15,00	14,03	13,17	12,47	12,03	11,88	12,00
37,00	38,76	40,51	41,93	42,79	42,94	42,33	41,02	39,11	36,76	34,16	31,48	28,90	26,54	24,49	22,81	21,49	20,50	19,77	19,21	18,71	18,19	17,57	16,81	15,91	14,91	13,91	13,03	12,38	12,06	12,00

TABELA 5.6 - VALORES ALTIMÉTRICOS DA GRADE REFINADA POR INTERPOLAÇÃO B-SPLINE (METROS)

164,06	161,27	157,23	152,11	146,11	139,63	133,11	126,93	121,21	116,00	111,44	108,04	106,42	107,07	110,00	115,11	122,28	131,29	141,92	153,48	163,51	169,06	168,01	161,51	151,50	139,71	126,95	113,81	100,99	89,75	81,42
162,45	158,98	154,22	148,69	142,83	136,80	130,74	124,78	119,05	113,72	109,01	105,39	103,40	103,48	105,69	109,96	116,26	124,53	134,72	146,29	156,72	162,98	162,84	157,26	147,96	136,53	123,98	111,16	98,97	88,49	80,82
160,61	156,33	150,74	144,73	139,02	133,53	128,03	122,35	116,66	111,23	106,36	102,47	100,00	99,33	100,55	103,67	108,73	115,88	125,28	136,57	147,23	154,23	155,25	150,95	142,73	131,94	119,81	107,49	96,11	86,56	79,67
159,14	154,20	147,91	141,50	135,91	130,89	125,89	120,52	114,95	109,53	104,57	100,42	97,44	95,94	96,02	97,78	101,32	106,97	115,08	125,49	135,83	143,22	145,40	142,64	135,89	126,15	114,75	103,14	92,60	83,91	77,72
158,54	153,27	146,64	140,04	134,51	129,74	125,07	120,00	114,67	109,38	104,44	100,11	96,65	94,31	93,22	93,49	95,31	99,10	105,36	114,08	123,26	130,42	133,53	132,44	127,51	119,24	108,98	98,27	88,47	80,38	74,60
158,85	153,62	147,03	140,47	134,98	130,27	125,72	120,84	115,72	110,60	105,70	101,21	97,28	94,10	91,81	90,56	90,57	92,31	96,34	102,79	110,15	116,49	120,16	120,60	117,53	110,97	102,10	92,36	83,12	75,25	69,53
160,03	155,16	148,97	142,69	137,26	132,46	127,81	122,91	117,83	112,75	107,82	103,11	98,69	94,65	91,15	88,39	86,61	86,30	88,00	91,94	97,11	102,17	105,87	107,37	105,89	100,99	93,55	84,75	75,77	67,71	61,67
161,92	157,72	152,27	146,53	141,23	136,27	131,32	126,14	120,83	115,53	110,39	105,36	100,39	95,44	90,73	86,51	83,08	80,88	80,39	81,88	84,74	88,17	91,29	93,13	92,65	89,13	83,00	75,02	66,05	57,48	50,83
164,10	160,82	156,44	151,52	146,54	141,48	136,20	130,63	124,89	119,19	113,64	108,13	102,47	96,55	90,64	85,06	80,14	76,27	73,81	73,02	73,61	75,16	77,15	78,62	78,49	75,90	70,93	63,85	55,21	46,54	39,67
166,06	163,94	160,91	157,14	152,77	147,86	142,42	136,51	130,32	124,08	117,93	111,72	105,19	98,24	91,15	84,31	78,09	72,77	68,61	65,79	64,24	63,81	64,22	64,67	64,19	62,02	58,00	52,19	44,89	37,45	31,50
167,26	166,47	165,05	162,76	159,40	155,02	149,77	143,77	137,25	130,46	123,54	116,40	108,84	100,78	92,55	84,56	77,22	70,68	65,09	60,55	57,07	54,66	53,18	52,07	50,63	48,28	45,04	41,00	36,43	32,00	28,56
167,17	167,73	168,02	167,42	165,44	162,10	157,50	151,79	145,21	138,01	130,40	122,34	113,75	104,62	95,30	86,25	77,84	70,18	63,31	57,28	52,10	47,81	44,37	41,48	38,77	36,00	33,38	31,24	29,83	29,03	28,63
165,25	167,00	168,88	170,06	169,85	168,08	164,72	159,79	153,55	146,31	138,32	129,64	120,25	110,22	99,91	89,81	80,27	71,42	63,28	55,87	49,20	43,25	38,02	33,50	29,67	26,56	24,50	23,86	24,80	26,62	28,39
161,12	163,76	166,94	169,84	171,68	172,02	170,52	166,94	161,56	154,80	146,99	138,21	128,45	117,81	106,66	95,48	84,68	74,44	64,89	56,13	48,12	40,81	34,19	28,49	23,99	20,94	19,43	19,50	21,05	23,33	25,40
155,12	158,32	162,46	166,80	170,64	173,25	173,92	172,13	168,15	162,45	155,41	147,08	137,47	126,61	114,90	102,77	90,64	78,85	67,72	57,48	48,15	39,67	32,07	25,71	21,03	18,32	17,27	17,44	18,38	19,64	20,76
147,81	151,23	155,88	161,19	166,57	171,13	173,92	174,20	172,14	168,11	162,42	155,12	146,17	135,62	123,76	110,94	97,57	84,15	71,22	59,26	48,44	38,86	30,68	24,18	19,72	17,44	16,69	16,64	16,61	16,52	16,42
139,59	142,96	147,67	153,36	159,55	165,38	169,92	172,37	172,65	170,86	167,05	161,26	153,48	143,76	132,25	119,13	104,68	89,62	74,72	60,75	48,20	37,55	29,16	23,04	19,12	17,19	16,51	16,18	15,53	14,66	13,90
130,52	133,71	138,28	143,97	150,44	157,00	162,88	167,36	170,07	170,70	169,01	164,97	158,60	149,93	138,96	125,69	110,31	93,70	76,94	61,06	46,97	35,51	27,29	21,93	18,75	17,10	16,31	15,72	14,82	13,77	12,87
120,53	123,64	128,13	133,78	140,33	147,30	154,14	160,27	165,07	167,89	168,18	165,81	160,75	152,96	142,35	128,83	112,56	94,64	76,44	59,28	44,30	32,64	25,01	20,61	18,22	16,80	15,85	15,06	14,16	13,27	12,56
109,72	112,98	117,64	123,46	130,15	137,40	144,85	152,07	158,30	162,69	164,50	163,43	159,28	151,89	141,19	127,15	110,02	91,16	72,23	54,80	40,03	29,00	22,34	18,95	17,28	16,04	14,97	14,06	13,30	12,71	12,31
98,89	102,35	107,25	113,28	120,14	127,59	135,39	143,19	150,18	155,42	158,08	157,70	153,92	146,49	135,42	120,81	103,09	83,86	65,06	48,42	34,87	25,18	19,69	17,19	16,05	14,95	13,84	12,92	12,37	12,13	12,05
88,97	92,44	97,37	103,44	110,39	117,95	125,92	133,92	141,09	146,44	149,09	148,57	144,53	136,71	125,26	110,39	92,65	73,85	56,11	41,24	29,76	21,89	17,56	15,65	14,75	13,75	12,69	11,89	11,59	11,65	11,83
80,66	83,78	88,31	94,10	100,96	108,57	116,57	124,48	131,34	136,09	137,81	136,24	131,27	122,88	111,26	96,68	79,72	62,28	46,54	34,28	25,48	19,66	16,31	14,58	13,59	12,62	11,73	11,17	11,11	11,38	11,72
73,60	76,14	80,02	85,29	91,91	99,42	107,28	114,82	121,01	124,70	124,98	121,90	115,73	106,75	95,22	81,42	65,92	50,54	37,40	28,09	22,06	18,23	15,64	13,87	12,62	11,65	11,00	10,74	10,89	11,27	11,67
67,14	69,14	72,37	77,06	83,29	90,49	97,94	104,85	110,12	112,61	111,47	107,00	99,81	90,43	79,25	66,58	53,00	40,09	29,69	23,10	19,40	17,14	15,13	13,31	11,86	10,94	10,51	10,50	10,80	11,25	11,67
60,83	62,48	65,24	69,43	75,16	81,80	88,54	94,53	98,75	100,17	98,01	92,72	85,03	75,64	65,06	53,76	42,37	32,07	24,21	19,65	17,43	16,09	14,51	12,79	11,36	10,54	10,26	10,39	10,78	11,25	11,67
54,83	56,30	58,77	62,52	67,65	73,53	79,34	84,25	87,37	87,79	84,88	79,15	71,40	62,39	52,74	43,08	34,02	26,31	20,73	17,70	16,35	15,48	14,15	12,56	11,20	10,45	10,25	10,42	10,82	11,28	11,68
49,50	50,87	53,11	56,44	60,91	65,91	70,69	74,49	76,49	75,92	72,24	66,12	58,50	50,26	42,00	34,28	27,59	22,31	18,78	17,09	16,47	15,89	14,62	12,96	11,50	10,69	10,46	10,61	10,96	11,37	11,72
45,12	46,40	48,43	51,34	55,11	59,21	62,98	65,72	66,68	65,05	60,40	53,69	46,21	39,07	32,66	27,16	22,76	19,62	17,92	17,61	17,86	17,64	16,24	14,17	12,28	11,21	10,87	10,96	11,22	11,53	11,79
41,78	42,98	44,82	47,34	50,47	53,75	56,63	58,49	58,54	55,93	50,27	42,87	35,49	29,55	25,07	21,74	19,32	17,93	17,74	18,72	19,90	20,07	18,43	15,77	13,27	11,85	11,36	11,37	11,53	11,72	11,87
39,53	40,67	42,37	44,58	47,23	49,89	52,11	53,33	52,72	49,36	42,87	34,89	27,58	22,62	19,71	18,08	17,13	16,96	17,86	19,83	21,76	22,25	20,39	17,19	14,14	12,40	11,77	11,72	11,79	11,87	11,94

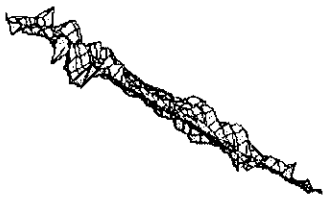
TABELA 5.7 - VALORES ALTIMÉTRICOS DA GRADE REFINADA POR INTERPOLAÇÃO β-SPLINE (METROS)

164,53	161,75	157,52	152,71	146,77	139,71	133,29	127,33	121,24	116,14	111,79	108,05	106,24	107,20	110,44	115,24	122,51	132,24	142,43	153,82	165,13	171,31	169,33	161,84	152,20	140,57	126,85	114,06	101,58	89,19	80,55
162,96	159,44	154,33	149,04	143,30	136,83	130,91	125,13	119,00	113,78	109,28	105,32	103,17	103,54	105,96	109,94	116,33	125,21	134,91	146,38	158,16	165,02	163,99	157,55	148,67	137,34	123,77	111,25	99,45	87,96	80,02
160,96	156,49	150,26	144,33	138,87	133,15	127,89	122,36	116,23	110,90	106,22	101,95	99,33	98,86	100,12	102,89	107,98	115,53	124,38	135,70	147,87	155,52	155,79	150,91	143,21	132,45	119,24	107,17	96,29	85,99	78,97
159,49	154,31	147,22	140,82	135,56	130,42	125,69	120,42	114,39	109,04	104,26	99,74	96,67	95,38	95,46	96,98	100,64	106,64	114,31	124,95	136,88	144,90	146,37	143,17	136,88	126,97	114,40	102,94	92,89	83,57	77,29
159,05	153,59	146,17	139,57	134,40	129,51	125,07	120,08	114,32	109,13	104,37	99,70	96,23	94,11	92,96	93,12	95,11	99,12	104,95	113,91	124,43	131,99	134,48	133,17	128,69	120,17	108,81	98,23	88,85	80,11	74,20
159,39	153,98	146,61	140,06	134,91	130,06	125,73	120,98	115,52	110,56	105,86	101,04	97,15	94,12	91,61	90,22	90,27	91,90	95,39	101,91	110,14	116,63	119,93	120,59	118,16	111,44	101,70	92,23	83,37	74,86	69,02
160,41	155,32	148,37	142,08	136,95	131,99	127,57	122,85	117,49	112,61	107,89	102,89	98,59	94,72	90,96	88,16	86,50	85,99	87,31	91,42	97,24	102,39	105,98	107,94	107,08	101,96	93,78	85,29	76,63	67,92	61,80
162,26	157,88	151,79	146,10	141,04	135,85	131,12	126,11	120,46	115,35	110,42	105,11	100,32	95,55	90,52	86,39	83,18	80,76	80,09	81,85	85,18	88,62	91,72	94,00	93,97	90,15	83,43	75,79	66,98	57,55	50,76
164,59	161,24	156,45	151,68	146,84	141,45	136,34	130,87	124,70	119,13	113,79	107,98	102,54	96,75	90,33	84,80	80,04	75,76	73,09	72,43	73,06	74,44	76,45	78,30	78,54	75,82	70,53	63,87	55,13	45,28	38,03
166,45	164,23	160,91	157,24	152,87	147,54	142,24	136,41	129,76	123,69	117,81	111,32	105,12	98,34	90,69	83,96	77,96	72,19	67,94	65,30	63,62	63,08	63,69	64,53	64,45	62,27	58,06	52,53	44,85	36,00	29,43
167,40	166,53	165,00	162,83	159,38	154,63	149,58	143,66	136,67	130,10	123,45	115,96	108,76	100,92	92,09	84,30	77,28	70,31	64,79	60,54	56,87	54,47	53,27	52,37	51,19	48,87	45,39	41,47	36,70	31,46	27,65
167,27	167,93	168,47	168,13	166,06	162,46	158,10	152,41	145,30	138,31	130,83	122,21	113,89	104,92	94,86	85,98	77,90	69,73	62,96	57,16	51,57	47,23	43,92	40,83	38,15	35,38	32,52	30,47	29,48	29,04	28,93
165,55	167,44	169,71	171,06	170,61	168,59	165,37	160,28	153,44	146,39	138,46	129,09	119,98	110,12	99,03	89,16	80,05	70,69	62,76	55,66	48,51	42,61	37,54	32,62	28,71	25,67	23,37	22,69	24,29	27,08	29,43
161,19	163,92	167,51	170,41	171,98	172,29	170,97	167,15	161,25	154,73	146,99	137,59	128,21	117,78	105,83	94,98	84,63	73,78	64,50	56,15	47,67	40,54	34,19	28,04	23,49	20,65	19,00	18,93	20,83	23,73	26,10
154,62	157,96	162,60	166,97	170,77	173,87	174,93	172,94	168,61	163,21	156,23	147,35	138,15	127,44	114,81	102,99	91,19	78,52	67,59	57,78	47,85	39,53	32,19	25,25	20,44	18,08	17,13	17,23	18,16	19,48	20,54
147,59	151,19	156,37	161,65	167,08	172,35	175,47	175,40	172,99	169,12	163,32	155,43	146,84	136,30	123,46	111,02	97,96	83,56	70,96	59,58	48,13	38,73	30,86	23,74	19,06	17,16	16,65	16,65	16,49	16,17	15,90
139,56	143,06	148,21	153,70	159,83	166,32	170,97	173,01	173,16	171,51	167,50	161,23	153,84	144,12	131,76	119,17	105,04	88,88	74,41	61,07	47,72	37,17	29,27	22,70	18,62	17,01	16,53	16,30	15,57	14,51	13,66
129,96	133,21	138,08	143,47	149,85	157,04	162,97	167,26	170,37	171,38	169,58	165,36	159,53	150,95	139,36	126,80	111,60	93,60	77,14	61,70	46,34	34,74	27,17	21,66	18,46	17,05	16,36	15,84	14,90	13,69	12,75
120,41	123,54	128,27	133,57	140,01	147,58	154,35	160,36	165,80	168,98	169,07	166,63	162,12	154,40	143,28	130,61	114,50	95,04	77,12	60,36	43,87	31,86	24,91	20,52	18,16	16,88	15,95	15,18	14,27	13,24	12,49
109,80	113,11	118,09	123,59	130,17	137,96	145,21	152,23	159,09	163,67	165,13	164,04	160,41	152,93	141,63	128,46	111,49	91,02	72,45	55,61	39,51	28,15	22,21	18,94	17,32	16,14	15,00	14,10	13,37	12,71	12,28
98,02	101,61	106,96	112,75	119,51	127,49	135,04	142,67	150,39	155,86	158,24	158,03	154,87	147,26	135,42	121,59	103,93	82,92	64,35	48,44	33,85	23,96	19,25	17,05	16,06	14,97	13,71	12,76	12,29	12,08	12,02
88,18	91,84	97,28	103,16	110,02	118,13	125,84	133,68	141,66	147,33	149,80	149,59	146,29	138,32	126,01	111,84	94,08	73,34	55,55	41,31	28,89	20,78	17,10	15,53	14,82	13,80	12,55	11,67	11,46	11,61	11,84
80,40	83,65	88,57	94,11	100,90	109,08	116,84	124,55	132,17	137,23	138,67	137,17	132,85	124,36	111,92	98,03	81,16	61,92	46,05	34,46	25,04	19,05	16,10	14,54	13,65	12,69	11,66	11,03	11,04	11,39	11,75
73,11	75,68	79,74	84,65	91,23	99,38	107,04	114,35	121,20	125,16	124,92	121,45	115,69	106,63	94,30	81,17	65,93	49,12	35,97	27,62	21,67	17,99	15,63	13,78	12,51	11,60	10,91	10,62	10,83	11,30	11,72
66,94	68,93	72,24	76,55	82,82	90,73	98,08	104,79	110,71	113,55	111,88	106,80	99,92	90,52	78,51	66,33	52,96	38,75	28,20	22,52	19,19	17,20	15,27	13,21	11,67	10,84	10,43	10,41	10,76	11,28	11,71
60,85	62,52	65,37	69,23	75,01	82,32	89,00	94,79	99,59	101,43	98,87	93,02	85,71	76,38	64,97	53,99	42,81	31,43	23,31	19,37	17,41	16,25	14,65	12,69	11,19	10,47	10,22	10,33	10,74	11,29	11,72
54,41	55,91	58,45	61,89	67,03	73,46	79,17	83,80	87,29	88,09	84,86	78,60	71,22	62,32	51,91	42,49	33,85	25,59	19,94	17,38	16,23	15,49	14,14	12,38	11,01	10,38	10,21	10,36	10,78	11,31	11,73
49,14	50,54	52,88	55,96	60,45	65,96	70,71	74,27	76,58	76,43	72,54	65,89	58,57	50,42	41,37	33,67	27,34	21,73	18,12	16,70	16,20	15,73	14,44	12,64	11,22	10,57	10,39	10,53	10,90	11,38	11,76
45,02	46,33	48,47	51,18	54,98	59,54	63,33	65,92	67,15	66,00	61,16	53,76	46,44	39,51	32,52	26,97	22,77	19,37	17,50	17,35	17,80	17,75	16,24	13,90	11,99	11,10	10,82	10,90	11,18	11,54	11,82
41,46	42,69	44,63	46,96	50,04	53,61	56,46	58,12	58,32	56,14	50,17	41,82	34,44	28,98	24,44	21,26	19,13	17,70	17,45	18,65	20,26	20,73	18,85	15,68	13,04	11,81	11,38	11,38	11,54	11,74	11,90
39,16	40,33	42,12	44,18	46,75	49,63	51,82	52,84	52,33	49,43	42,60	33,52	26,08	21,73	19,02	17,59	16,87	16,76	17,59	19,78	22,26	23,12	20,94	17,10	13,88	12,36	11,81	11,76	11,82	11,90	11,96

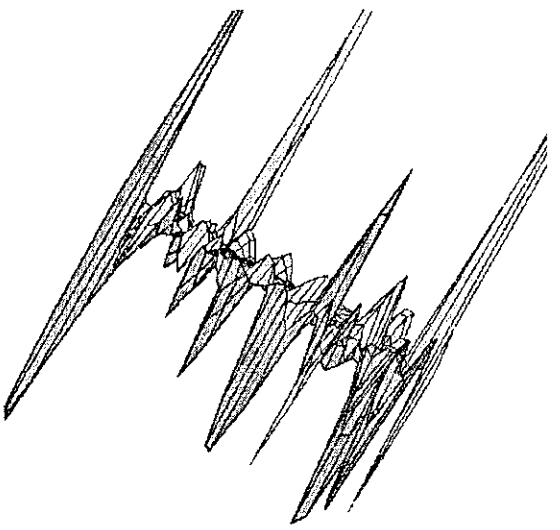
TABELA 5.8 - VALORES ALTIMÉTRICOS DA GRADE REFINADA POR INTERPOLAÇÃO AKIMA (METROS)

167,00	165,29	162,13	157,00	149,62	141,33	134,00	128,01	122,39	117,00	112,06	108,00	105,00	104,96	108,96	116,00	125,06	135,31	146,00	160,22	176,01	185,00	181,68	170,04	156,00	142,76	129,48	116,00	102,38	88,75	75,00
164,65	158,92	153,15	147,34	141,59	135,86	129,95	123,61	117,11	111,00	105,70	101,32	97,92	97,28	100,34	105,89	113,31	122,35	132,35	147,90	166,83	178,44	176,89	166,81	153,24	138,87	124,18	110,84	98,75	87,01	76,11
162,96	154,26	146,58	140,24	135,65	131,83	127,01	120,48	113,41	106,90	101,41	96,73	92,96	91,92	94,16	97,89	103,23	110,90	119,93	135,74	156,24	169,22	169,10	161,01	148,81	134,44	119,17	105,90	94,84	84,66	75,89
162,00	151,50	142,66	136,00	132,00	129,17	125,00	118,83	112,21	106,00	100,63	95,96	92,00	90,00	90,40	92,00	95,46	101,49	109,00	123,46	143,12	156,00	157,73	152,91	143,00	129,14	113,92	101,00	90,90	82,06	75,00
161,77	150,55	141,20	134,50	130,66	127,90	124,02	118,62	113,00	107,51	102,42	97,83	93,80	90,86	89,31	88,93	90,34	94,03	99,48	111,12	127,43	138,94	143,18	142,80	136,07	123,30	108,73	96,58	87,36	79,21	72,72
162,15	151,09	141,75	135,25	131,35	128,15	124,33	119,56	114,55	109,59	104,70	99,98	95,82	92,51	89,95	87,96	86,94	87,76	90,91	99,19	111,07	120,54	126,92	130,59	127,20	116,44	103,54	92,56	83,90	75,68	68,44
163,00	152,87	144,18	138,00	133,82	130,02	126,00	121,48	116,74	112,00	107,22	102,44	98,00	94,05	90,44	87,00	83,92	82,21	83,00	88,00	95,84	103,00	110,04	115,99	115,00	107,01	97,30	88,00	79,58	71,18	63,00
164,70	156,46	149,15	143,33	138,60	134,00	129,29	124,28	119,12	114,02	109,33	104,91	100,28	95,19	89,98	85,04	80,58	77,17	75,68	77,57	82,10	86,81	92,66	98,63	99,39	94,63	88,05	80,24	71,24	60,87	49,33
167,04	161,44	155,96	150,67	145,40	139,96	134,34	128,38	122,26	116,38	111,56	107,43	102,61	96,34	89,50	83,07	77,52	72,81	69,26	68,13	69,44	71,55	75,32	80,20	82,44	80,52	75,83	69,01	59,15	45,49	28,83
169,00	166,07	162,48	158,00	152,76	147,08	141,00	134,42	127,58	121,00	115,48	110,59	105,00	97,77	89,68	82,00	75,30	69,30	64,00	60,26	58,49	58,00	59,83	63,48	66,00	65,60	62,55	57,00	47,45	33,06	15,00
170,14	170,31	168,97	165,93	161,37	155,84	149,66	142,94	135,82	128,60	121,76	115,12	108,04	99,56	90,16	81,36	73,72	66,78	60,46	55,06	50,91	48,02	47,71	49,42	50,32	48,94	45,87	41,38	35,46	28,02	19,26
169,87	173,50	174,89	173,85	170,56	165,64	159,59	152,90	145,65	137,72	129,45	121,10	112,35	102,31	91,45	81,61	73,36	65,92	59,09	52,76	47,05	42,10	38,99	37,44	35,47	31,84	27,46	23,65	23,14	26,67	32,61
167,00	173,05	176,87	178,00	176,88	174,01	169,00	162,56	155,38	147,00	137,72	128,07	118,00	107,02	95,57	85,00	75,89	67,67	60,00	52,62	45,59	39,00	33,17	28,00	23,00	18,51	15,19	13,00	15,61	24,86	38,00
161,32	168,09	173,59	177,11	179,43	180,69	179,21	173,72	165,41	156,30	147,38	137,92	127,33	115,63	103,70	92,24	81,68	71,83	62,53	53,93	46,07	38,54	30,66	22,69	15,74	12,45	12,54	12,61	14,60	21,25	30,32
154,01	160,48	166,81	172,44	178,28	184,05	187,49	184,28	175,25	165,87	158,40	150,29	139,78	127,13	114,17	101,45	89,09	77,06	65,66	55,87	47,59	39,49	30,35	21,09	13,64	11,86	14,54	16,02	16,13	17,72	19,71
146,00	151,67	157,79	164,00	171,38	179,55	186,00	186,72	181,83	175,00	168,66	161,37	151,00	138,07	124,66	111,00	96,87	82,56	69,00	57,62	48,15	39,00	29,43	20,64	14,00	13,02	16,34	18,00	16,75	14,93	12,00
137,77	142,33	147,52	153,34	160,69	169,02	176,19	180,61	182,07	180,47	175,97	168,83	159,13	147,81	135,39	121,30	105,80	89,97	74,47	60,08	47,14	35,63	26,36	19,79	15,38	14,68	16,79	17,57	16,07	13,75	10,07
129,13	132,46	136,71	142,46	149,88	157,95	165,49	171,85	176,93	180,38	179,34	173,29	165,37	157,07	146,65	132,37	115,42	98,16	80,62	62,50	45,08	30,81	22,35	18,76	17,20	16,78	17,31	17,02	15,61	13,72	11,04
120,00	122,50	126,21	132,00	139,46	147,29	155,00	162,38	169,44	176,00	178,41	175,32	170,00	164,44	156,23	142,00	122,61	102,13	82,00	61,56	41,64	26,00	18,64	17,68	18,00	17,41	16,87	16,00	14,81	13,52	12,00
107,72	111,25	115,84	121,97	129,35	137,11	144,89	152,95	161,33	169,28	174,15	174,55	171,68	166,85	158,53	143,61	121,85	97,76	75,36	54,45	34,92	20,96	16,02	16,88	17,69	16,45	15,17	14,05	13,17	12,49	12,00
93,48	99,11	105,27	111,97	119,25	126,97	134,80	143,27	152,45	161,29	167,52	169,68	168,07	162,29	151,45	135,12	112,56	87,02	63,94	43,73	26,20	15,93	14,27	16,28	17,07	15,21	13,12	11,69	11,18	11,27	12,00
83,00	88,89	95,22	102,00	109,32	117,09	125,00	133,63	143,02	152,00	158,11	159,95	158,00	150,91	138,05	121,00	98,99	74,05	52,00	34,08	19,95	13,00	13,12	15,40	16,00	13,90	11,48	10,00	9,81	10,46	12,00
76,98	80,80	85,75	92,15	99,74	107,82	115,99	124,70	133,78	141,91	145,93	144,69	139,94	131,46	118,68	102,86	82,38	59,16	39,77	26,73	18,16	13,99	13,39	14,11	14,00	12,33	10,52	9,56	9,66	10,44	12,00
71,64	73,13	76,56	82,47	90,41	99,03	107,58	116,25	124,70	131,36	132,28	126,77	118,13	108,08	96,19	82,57	64,15	43,05	27,16	20,31	18,33	17,09	14,84	12,85	11,56	10,66	9,93	9,78	10,16	10,85	12,00
66,00	65,81	67,73	73,00	81,20	90,27	99,00	107,27	114,77	120,00	118,75	110,66	100,00	89,05	77,34	65,00	48,75	30,47	18,00	15,72	18,30	19,00	15,68	12,01	10,00	9,66	9,65	10,00	10,54	11,16	12,00
59,64	59,11	60,32	64,80	72,57	81,46	89,74	96,99	103,10	106,75	104,70	96,88	86,70	75,45	63,09	51,08	37,55	23,45	14,44	13,70	16,73	17,64	14,69	11,36	9,59	9,40	9,52	9,93	10,49	11,16	12,00
53,14	53,13	54,60	58,40	65,01	72,97	80,20	86,07	90,52	92,30	89,66	82,86	73,71	62,52	50,05	38,88	28,79	19,70	14,33	13,68	14,91	15,03	13,06	10,85	9,63	9,43	9,50	9,85	10,41	11,10	12,00
47,00	47,63	49,59	53,00	58,42	65,05	71,00	75,60	78,66	79,00	75,39	68,52	60,00	49,75	38,47	29,00	22,21	17,37	15,00	14,55	14,53	14,00	12,59	11,03	10,00	9,66	9,68	10,00	10,49	11,14	12,00
42,24	43,21	45,31	48,33	52,73	58,08	62,93	66,45	68,32	67,87	62,88	54,16	45,59	37,34	28,60	21,81	17,88	15,79	15,12	15,58	16,39	16,35	14,68	12,29	10,52	9,97	10,09	10,44	10,82	11,33	12,00
39,12	40,31	42,25	44,67	48,01	52,18	56,08	58,55	59,25	58,27	51,45	39,75	31,02	25,84	20,69	17,18	15,72	15,27	15,34	17,09	20,13	21,56	19,11	14,56	11,15	10,31	10,65	11,11	11,32	11,61	12,00
37,00	38,38	40,10	42,00	44,39	47,27	50,00	51,58	51,53	50,00	40,41	24,74	16,00	15,29	14,98	15,00	15,20	15,52	16,00	19,59	25,67	29,00	25,31	17,71	12,00	10,74	11,37	12,00	12,00	12,00	12,00

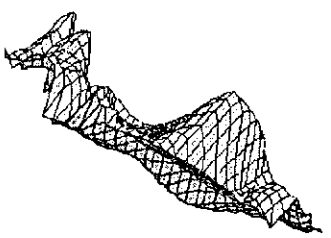
Os resultados da interpolação estarão mais próximos dos dados reais do terreno quando a superfície de erro gerada tender a ser um plano com valores constantes e nulos.



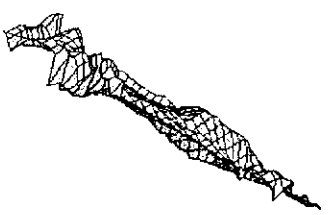
a) Interpolação Linear;



b) Interpolação de Lagrange;



c) Interpolação Bézier;



d) Interpolação B-Spline;

e) Interpolação β-Spline; f) Interpolação Akima.

Fig. 5.4 - Superfícies de erro entre as grades refinadas por interpolação e a superfície real.

Os valores das superfícies de erro apresentadas na Figura 5.4 estão descritos nas Tabelas 5.9, 5.10, 5.11, 5.12, 5.13 e 5.14.

5.4 - ANÁLISE E AVALIAÇÃO

A análise visual não é suficientemente científica para validar ou não um método ou procedimento. É necessária uma verificação mais criteriosa dos resultados obtidos. Dessa forma, através das superfícies de erro geradas, os métodos de interpolação serão analisados e avaliados, levando-se em conta algumas técnicas de estatística.

O primeiro passo é organizar os dados numéricos das superfícies de erro, distribuindo-os de acordo com as freqüências dos valores obtidos. As Tabelas 5.15 e 5.16 apresentam a distribuição de freqüência de erros das interpolações.

TABELA 5.9 - VALORES DA SUPERFÍCIE DE ERRO OBTIDA COM A INTERPOLAÇÃO LINEAR (METROS)

0,00	3,33	-0,33	0,00	1,67	2,33	0,00	2,67	1,33	0,00	3,00	0,00	0,00	-0,67	-1,33	0,00	-9,00	-5,00	0,00	8,00	2,00	0,00	9,67	1,33	0,00	0,33	-0,33	0,00	-2,33	-5,67	0,00
1,67	1,78	4,89	6,00	7,33	6,67	3,00	8,89	7,78	3,67	2,89	-4,89	-0,67	-3,11	-0,56	2,00	0,44	-8,11	5,33	3,44	5,56	9,67	17,56	7,44	5,33	4,89	2,44	3,00	-1,00	-4,00	-4,00
0,33	-13,78	-10,89	-6,00	-4,00	-4,00	-4,00	-2,89	-2,78	-3,67	-7,22	-7,78	-6,33	-6,56	-6,78	-5,00	-8,11	-8,22	-5,33	-5,11	-2,89	-4,67	2,44	1,56	-4,33	-3,56	-4,78	-4,00	-7,67	-2,33	-12,00
0,00	-13,33	-7,67	0,00	0,67	0,33	0,00	1,33	0,67	0,00	-2,33	-2,67	0,00	-1,00	0,00	0,00	-1,67	-2,33	0,00	-1,67	-0,33	0,00	9,33	7,67	0,00	0,00	0,00	0,00	-4,33	-0,67	0,00
-1,33	-15,78	-10,22	-1,67	0,11	-1,11	-1,33	-0,56	-0,78	-1,00	-2,33	-2,67	-1,00	-0,78	-0,56	0,67	-1,67	-2,00	-0,33	-2,00	0,33	6,67	17,22	16,78	6,33	4,67	3,00	2,33	-2,11	3,44	4,00
-1,67	-14,22	-8,78	-2,33	-0,44	-0,56	-0,67	-0,44	-0,22	-1,00	-1,33	-1,67	-1,00	-1,56	-1,11	0,33	-1,67	-1,67	0,33	-3,33	-2,00	5,33	14,11	13,89	4,67	4,33	3,00	1,67	-1,89	7,56	4,00
0,00	-10,67	-5,33	0,00	1,00	1,00	0,00	-0,33	-0,67	0,00	-0,33	0,33	0,00	-0,33	-0,67	0,00	-1,67	-2,33	0,00	-4,67	-5,33	0,00	6,00	7,00	0,00	2,00	2,00	0,00	-2,67	2,67	0,00
2,00	-10,22	-5,44	-2,67	-1,11	-0,56	-1,00	-0,67	-1,33	-1,00	-0,11	-0,22	0,67	-0,33	-0,33	-1,33	-2,44	-3,56	-0,67	-4,44	-5,22	-4,00	-1,56	0,89	-1,67	3,33	4,33	4,33	3,56	7,78	3,00
2,00	-9,78	-3,56	-1,33	-0,22	-1,11	-1,00	-2,00	-1,00	0,00	0,11	1,22	0,33	0,67	0,00	-0,67	-2,22	-3,78	-1,33	-4,22	-5,11	-3,00	-3,11	-0,22	-0,33	6,67	8,67	6,67	4,78	6,89	-15,00
0,00	-8,33	-2,67	0,00	0,67	0,33	0,00	0,67	0,33	0,00	0,33	0,67	0,00	0,67	0,33	0,00	0,00	-1,00	0,00	-2,00	-1,00	0,00	-1,67	-0,33	0,00	5,00	6,00	0,00	0,00	-13,00	0,00
3,67	-6,11	0,11	1,33	1,11	-0,11	-1,33	-1,44	-1,56	-1,67	-0,89	-1,11	-2,33	-2,56	-0,78	-1,00	-1,22	-2,44	-1,67	-4,00	-4,33	-3,67	-5,67	-4,67	-1,67	0,44	0,56	-5,33	-9,78	-14,22	-11,67
-0,67	-5,89	0,89	1,67	3,56	2,44	-0,67	-1,56	-1,44	-2,33	-0,11	-0,89	-2,67	-2,78	-1,89	-1,00	-1,44	-1,89	-1,33	-3,00	-2,67	-3,33	-4,67	-6,00	-3,33	-4,11	-4,89	-8,67	-13,56	-18,44	-20,33
0,00	-9,67	-3,33	0,00	5,00	5,00	0,00	-0,67	-0,33	0,00	2,67	2,33	0,00	-1,00	-1,00	0,00	0,33	-1,33	0,00	-1,00	1,00	0,00	-0,67	-2,33	0,00	-0,67	-0,33	0,00	-7,33	-18,67	0,00
4,00	-7,44	0,11	6,67	11,22	10,78	5,33	4,44	2,56	1,67	1,78	3,89	1,00	-1,22	-3,44	-2,67	-2,44	-4,22	-2,00	-2,00	0,00	0,00	-1,67	-3,33	-2,00	-3,22	-3,44	-0,67	-2,56	13,56	-0,33
1,00	-9,22	-2,44	5,33	11,44	11,56	4,67	4,56	2,44	1,33	1,89	2,44	0,00	-0,44	-1,89	-2,33	-3,22	-4,11	-2,00	-2,00	0,00	0,00	-1,67	-3,33	-2,00	-3,78	-4,56	-1,33	2,22	9,78	5,33
0,00	-11,00	-5,00	0,00	4,67	5,33	0,00	0,67	0,33	0,00	1,00	2,00	0,00	2,33	1,67	0,00	-1,00	-2,00	0,00	0,00	1,00	0,00	-2,67	-3,33	0,00	-3,33	-3,67	0,00	5,00	12,00	0,00
-3,33	-11,67	-7,00	-1,33	3,22	4,78	3,33	4,44	5,56	3,67	5,67	5,67	3,67	4,67	4,67	0,67	-0,33	-0,33	2,67	3,56	3,44	2,33	-3,22	-4,78	-2,33	-2,00	0,33	1,67	2,44	-1,78	0,00
0,33	-10,33	-6,00	-1,67	1,78	2,22	2,67	5,22	6,78	5,33	6,33	6,33	4,33	7,00	7,67	2,33	2,33	1,33	5,33	5,11	2,89	1,67	-5,78	-6,22	-1,67	1,33	4,33	3,33	-0,11	-1,56	0,00
0,00	-9,00	-5,00	0,00	0,33	-0,33	0,00	2,00	3,00	0,00	2,00	2,00	0,00	4,33	5,67	0,00	0,00	-2,00	0,00	-0,33	-1,67	0,00	-7,33	-4,67	0,00	4,67	4,33	0,00	-2,67	-1,33	0,00
1,33	-6,44	-2,22	0,00	-0,67	-1,33	-3,00	-0,67	0,67	0,00	2,67	4,33	2,00	9,33	13,67	10,00	10,00	7,00	6,00	-0,22	-5,44	-2,67	-8,22	-3,78	1,67	4,78	0,89	-2,00	-3,33	-0,67	0,00
-0,33	-4,89	-1,44	1,00	-0,67	-1,33	-2,00	-1,33	-1,67	-2,00	2,33	3,67	2,00	10,33	13,67	9,00	8,00	5,00	2,00	-7,11	-10,22	-3,33	-5,11	0,11	2,33	2,89	-1,56	-2,00	-2,00	0,00	0,00
0,00	-3,33	-1,67	0,00	-0,67	-0,33	0,00	0,00	0,00	0,00	3,00	3,00	0,00	6,33	7,67	0,00	2,00	1,00	0,00	-9,00	-10,00	0,00	1,00	3,00	0,00	-1,00	-2,00	0,00	-0,67	0,67	0,00
1,67	-6,33	-4,33	-1,33	-1,33	0,67	1,67	2,33	3,00	3,67	10,56	12,44	8,33	10,44	5,56	-1,33	-0,78	-2,22	-3,67	-13,11	-9,56	2,00	4,33	3,67	1,00	-0,67	-1,33	0,00	-0,67	0,67	0,00
-5,67	-6,33	-5,00	-0,67	-1,00	0,67	1,33	4,67	5,00	4,33	11,11	12,89	6,67	5,56	1,44	-1,67	-1,56	-3,44	-2,33	-10,22	-5,11	3,00	5,67	3,33	0,00	-1,33	-0,67	0,00	-0,67	0,67	0,00
0,00	-6,33	-3,67	0,00	-1,67	-0,33	0,00	3,00	3,00	0,00	4,67	6,33	0,00	-0,33	0,33	0,00	-1,33	-3,67	0,00	-6,33	-1,67	0,00	4,00	0,00	0,00	0,00	0,00	0,00	-0,67	0,67	0,00
-1,67	-4,89	-2,11	0,67	-1,11	-0,89	-0,67	1,78	1,22	-2,33	-1,78	-4,22	-8,67	-6,44	-3,22	-3,00	-6,00	-7,00	-2,00	-5,11	-1,22	-0,33	2,11	-0,44	0,00	0,00	0,00	0,00	-0,67	0,67	0,00
-3,33	-5,44	-2,56	0,33	-1,56	-1,44	-1,33	0,56	0,44	-3,67	-2,22	-4,78	-7,33	-4,56	-1,78	-4,00	-6,67	-6,33	-2,00	-3,89	0,22	-0,67	1,22	-1,89	0,00	0,00	0,00	0,00	-0,67	0,67	0,00
0,00	-4,00	-2,00	0,00	-1,00	0,00	0,00	1,33	1,67	0,00	4,33	2,67	0,00	0,33	-0,33	0,00	-3,33	-2,67	0,00	-1,67	0,67	0,00	1,33	-1,33	0,00	0,00	0,00	0,00	-0,67	0,67	0,00
-1,67	-4,56	-3,44	-2,33	-3,22	-2,11	-2,00	1,22	0,44	-0,33	6,67	7,67	6,67	0,67	-1,33	-2,33	-2,33	-2,33	-0,33	-3,56	-3,78	-6,00	-4,22	-3,44	-0,67	-0,67	-0,67	-0,67	-1,11	0,44	0,00
-0,33	-4,11	-3,89	-2,67	-3,44	-2,22	-3,00	-0,89	-0,78	-1,67	6,00	7,67	7,33	0,00	-2,33	-2,67	-3,33	-2,00	-1,67	-4,44	-7,22	-11,00	-7,78	-5,56	-1,33	-1,33	-1,33	-1,33	-1,56	0,22	0,00
0,00	2,33	1,67	0,00	2,33	4,67	0,00	0,00	0,00	0,00	11,33	5,67	0,00	0,33	0,67	0,00	-0,33	0,33	0,00	-4,33	-8,67	0,00	-11,33	-5,67	0,00	0,00	0,00	0,00	0,00	0,00	0,00

TABELA 5.10 - VALORES DA SUPERFÍCIE DE ERRO OBTIDA COM A INTERPOLAÇÃO LAGRANGE (METROS)

0,00	-45,81	-26,43	0,00	9,71	7,98	0,00	0,50	-0,39	0,00	5,60	2,80	0,00	1,42	0,63	0,00	-6,56	-2,48	0,00	6,55	-0,68	0,00	-1,97	-10,23	0,00	13,39	17,94	0,00	-49,53	-94,68	0,00
35,42	288,30	179,44	28,42	-55,36	-68,67	-45,28	-6,24	10,92	5,33	-8,54	-31,57	-34,98	-26,27	-2,49	24,02	45,59	44,95	47,60	24,75	0,09	-14,84	-27,55	-32,66	-11,29	3,01	-18,23	-91,45	-198,41	-202,10	239,48
17,39	115,34	71,74	6,93	-31,76	-39,68	-27,48	-10,56	-0,50	-0,96	-9,90	-18,28	-22,07	-16,48	-6,07	7,48	17,76	21,67	16,64	6,15	-6,80	-18,99	-28,88	-26,96	-13,19	2,24	-4,98	-51,86	-133,40	-152,60	115,71
0,00	-47,35	-22,55	0,00	6,28	3,44	0,00	-1,20	-1,17	0,00	0,05	-0,29	0,00	-0,28	0,68	0,00	1,83	1,64	0,00	-0,29	-0,37	0,00	-3,45	-5,58	0,00	11,74	17,81	0,00	-49,16	-89,82	0,00
-6,04	-101,50	-55,17	-3,03	19,61	17,35	8,08	0,11	-2,71	-0,60	2,95	5,56	6,44	5,54	2,07	-1,54	-4,43	-5,76	-6,46	-3,51	1,10	9,14	10,24	7,96	6,58	13,10	21,71	19,04	-9,88	-49,17	-47,93
-4,44	-73,92	-39,33	-2,57	14,04	12,84	5,95	-0,20	-1,85	-0,92	2,09	3,80	3,96	2,61	0,59	-1,10	-3,02	-3,51	-3,52	-3,03	-0,25	7,06	8,75	6,62	3,85	6,72	12,57	12,45	-4,65	-24,91	-34,37
0,00	-23,29	-10,32	0,00	3,77	2,67	0,00	-1,47	-1,56	0,00	0,35	1,01	0,00	-0,33	-0,66	0,00	0,14	-0,17	0,00	-2,04	-3,14	0,00	1,75	1,98	0,00	0,20	1,99	0,00	-8,84	-13,06	0,00
4,01	11,19	8,99	0,39	-3,47	-3,61	-1,95	0,22	0,37	0,03	-0,36	-1,94	-1,84	-2,58	-1,25	-0,55	1,45	1,54	2,78	0,74	-1,23	-3,02	-3,97	-3,22	-4,30	-4,56	-5,46	-5,16	-6,91	2,24	22,46
3,43	19,82	14,10	1,29	-4,60	-5,14	-1,34	0,24	1,82	1,35	-0,18	-0,50	-1,64	-1,29	-0,79	0,03	1,15	0,61	1,78	0,45	-1,17	-1,55	-2,97	-1,58	-2,38	-2,07	-2,90	-1,72	0,35	13,87	3,78
0,00	12,53	7,79	0,00	-4,06	-3,10	0,00	2,36	1,92	0,00	-0,38	-0,22	0,00	0,28	0,17	0,00	1,16	0,22	0,00	-0,56	0,49	0,00	-0,22	0,86	0,00	-0,82	-1,18	0,00	6,47	3,05	0,00
0,85	2,08	1,55	-1,32	-2,85	-1,37	1,09	1,77	1,33	0,37	0,66	0,75	0,68	0,11	1,50	0,38	0,52	-1,03	-1,10	-1,57	-0,62	0,22	0,18	0,80	1,56	-1,38	-2,07	-0,10	9,22	15,75	-15,23
-3,59	-1,72	-0,40	-1,50	-0,68	0,79	1,34	0,61	0,47	-0,53	1,22	0,98	0,55	0,41	0,94	0,68	0,50	-0,42	-0,71	-1,07	0,47	0,43	1,05	-0,36	0,30	-3,44	-4,52	-2,49	8,47	13,06	-25,88
0,00	1,74	1,20	0,00	0,66	1,57	0,00	-0,87	-0,63	0,00	1,52	1,22	0,00	-0,66	-0,45	0,00	1,34	-0,43	0,00	-0,94	1,06	0,00	0,50	-1,03	0,00	-1,17	-1,36	0,00	6,72	3,56	0,00
2,97	9,77	6,97	2,86	0,67	0,94	0,69	0,32	-0,47	0,16	-0,36	2,43	1,69	0,09	-1,05	0,43	1,82	-0,27	0,23	-0,27	0,86	0,16	-0,18	-0,48	1,38	3,25	3,12	2,60	1,17	15,64	-3,07
0,26	12,50	8,18	1,82	-0,11	0,24	-0,67	-0,65	-1,44	-0,62	-0,91	0,21	0,11	0,27	-0,49	0,58	0,98	-0,04	0,22	-0,04	0,82	-0,39	-1,06	-1,25	1,01	4,69	4,22	0,81	-6,20	-6,56	4,32
0,00	7,76	6,67	0,00	-1,37	-0,37	0,00	-1,10	-1,18	0,00	-0,63	0,24	0,00	0,78	0,33	0,00	-0,09	-0,86	0,00	1,23	2,05	0,00	-2,31	-2,73	0,00	2,94	3,33	0,00	-11,30	-14,87	0,00
-4,10	-7,49	-3,01	-1,84	-0,75	-0,91	-0,84	-2,32	-0,82	-0,03	0,55	0,77	0,78	-0,17	0,06	-1,67	-2,89	-2,43	0,70	3,58	3,55	0,39	-2,78	-3,68	-2,37	1,46	3,35	-1,07	-12,48	-23,39	5,02
-0,67	-24,51	-11,94	-1,89	3,22	1,29	-0,60	-1,74	-0,19	1,69	1,10	1,37	1,32	1,25	1,68	-0,94	-2,65	-3,08	2,50	5,44	4,03	-0,04	-3,37	-3,67	-1,96	1,02	3,19	0,80	-6,04	-8,83	4,32
0,00	-37,26	-19,06	0,00	7,04	4,47	0,00	-1,83	-0,88	0,00	-0,05	0,07	0,00	1,10	2,16	0,00	-3,06	-4,55	0,00	3,29	2,71	0,00	-2,03	-0,04	0,00	1,34	0,38	0,00	3,79	8,64	0,00
0,40	-40,04	-19,57	0,40	8,72	6,98	-0,10	-2,63	-3,04	-1,76	-2,31	-1,34	-2,00	0,18	3,08	2,52	-0,29	-2,09	0,26	1,03	-0,81	-0,86	-1,39	1,25	0,85	-1,14	-5,04	-0,27	11,38	20,16	-4,02
-0,79	-32,36	-15,80	1,31	7,23	6,01	0,98	-2,10	-4,15	-3,47	-3,02	-2,76	-2,27	0,18	2,39	1,74	-1,17	-2,71	-3,20	-4,51	-4,49	-1,15	0,95	4,15	1,58	-2,56	-6,50	0,03	11,39	17,17	-4,27
0,00	-21,61	-10,81	0,00	3,34	2,66	0,00	-1,44	-1,58	0,00	-0,27	-0,63	0,00	0,70	2,11	0,00	1,07	1,04	0,00	-2,87	-3,65	0,00	3,09	4,06	0,00	-2,73	-2,87	0,00	-1,26	-5,06	0,00
6,44	-23,05	-11,64	-1,28	1,16	0,87	-1,14	-1,64	-0,17	3,09	5,21	5,01	2,83	1,03	-1,75	-1,14	1,84	3,09	1,15	-2,72	-0,67	3,34	3,35	1,24	0,85	3,69	5,47	-1,53	-24,22	-44,27	5,74
-0,72	-47,95	-24,28	-0,49	6,00	3,38	-2,28	-2,47	-0,82	3,66	6,65	6,73	1,64	-2,29	-4,16	-0,31	3,25	4,14	3,64	0,94	3,52	3,75	0,44	-3,08	-0,07	10,13	15,32	-2,27	-49,32	-89,78	7,26
0,00	-99,51	-49,63	0,00	16,81	12,27	0,00	-5,90	-5,37	0,00	3,60	5,57	0,00	-3,27	-2,82	0,00	0,20	-0,95	0,00	0,15	3,92	0,00	-4,05	-9,02	0,00	17,13	24,22	0,00	-64,91	-123,66	0,00
-9,53	-148,22	-73,97	2,51	31,40	24,79	5,72	-7,22	-9,77	-3,47	1,14	3,70	2,31	0,93	0,25	-1,88	-5,93	-7,21	-4,46	-2,43	0,79	-2,93	-6,81	-8,98	1,62	19,60	28,06	5,07	-60,84	-125,68	-13,98
-14,24	-138,23	-69,96	2,30	30,84	26,13	8,55	-5,33	-8,84	-4,60	1,51	5,08	6,59	6,04	3,46	-4,42	-9,36	-10,41	-7,88	-5,55	-0,46	-2,95	-4,29	-6,76	1,98	14,58	21,58	8,09	-35,29	-82,13	-21,44
0,00	5,30	3,53	0,00	-1,63	-0,52	0,00	-0,72	-0,73	0,00	1,14	-0,29	0,00	0,77	0,56	0,00	-0,82	-0,25	0,00	-1,32	0,73	0,00	0,93	-1,56	0,00	0,93	0,82	0,00	-0,56	1,72	0,00
29,29	289,63	152,51	-2,87	-71,34	-64,82	-29,38	1,83	10,65	3,63	-10,30	-23,89	-26,84	-21,38	-5,07	11,31	25,04	27,96	23,30	12,71	5,80	1,36	1,21	0,09	-4,72	-17,28	-30,36	-25,55	17,07	87,65	64,21
57,36	496,90	262,94	-3,81	-125,12	-118,02	-58,96	-5,39	14,14	2,84	-22,71	-47,00	-53,85	-40,82	-9,79	23,69	47,92	56,42	46,93	28,40	9,52	-2,92	-6,00	-5,94	-9,28	-24,72	-46,91	-52,59	-8,52	92,22	129,13
0,00	-117,87	-59,94	0,00	25,04	20,22	0,00	-11,28	-10,44	0,00	14,13	9,46	0,00	2,64	2,21	0,00	-0,55	0,53	0,00	-2,59	-7,52	0,00	-19,20	-13,96	0,00	17,29	23,69	0,00	-61,47	-117,82	0,00

TABELA 5.11 - VALORES DA SUPERFÍCIE DE ERRO OBTIDA COM A INTERPOLAÇÃO BÉZIER (METROS)

0,00	3,90	1,66	3,91	3,33	1,65	-3,37	-1,94	-5,25	-9,45	-8,66	-14,93	-19,29	-17,68	-17,00	-15,06	-17,62	-7,40	3,97	21,87	26,67	36,77	37,50	22,13	15,85	9,69	4,57	2,25	-1,64	-5,56	0,00
1,37	1,43	4,82	6,36	6,89	5,24	0,25	4,74	1,57	-5,39	-8,27	-19,13	-19,00	-19,85	-16,61	-14,14	-10,23	-13,61	5,04	14,05	27,81	44,70	45,05	29,15	23,21	15,30	7,36	4,17	-0,66	-3,62	-3,32
-0,79	-15,99	-14,22	-10,51	-7,93	-7,57	-7,55	-7,98	-9,99	-13,71	-19,22	-22,62	-24,93	-24,16	-24,24	-23,08	-21,50	-17,29	-10,15	2,22	17,17	29,05	30,21	24,93	16,44	8,84	1,10	-2,94	-6,64	-0,52	-9,18
-2,36	-18,21	-15,29	-10,54	-7,98	-6,65	-5,61	-5,97	-8,81	-12,24	-16,36	-19,24	-19,94	-20,46	-19,77	-20,81	-18,43	-15,45	-9,64	2,28	17,62	32,71	37,86	33,36	24,44	15,24	7,78	1,93	-1,60	3,72	6,28
-3,21	-20,05	-17,20	-11,56	-8,10	-7,83	-6,80	-7,07	-8,72	-10,86	-13,55	-15,89	-16,92	-16,68	-17,15	-17,29	-17,00	-15,14	-11,51	-1,88	12,02	30,48	39,79	39,23	30,03	20,35	12,24	6,64	2,31	8,86	10,74
-3,13	-18,29	-15,73	-12,34	-9,07	-7,92	-6,92	-7,12	-7,60	-9,43	-10,69	-12,46	-13,81	-14,77	-15,36	-15,54	-16,27	-15,43	-12,88	-7,41	3,18	20,16	30,78	33,31	27,96	20,91	14,24	8,93	4,82	14,60	11,77
-0,96	-14,69	-12,61	-10,62	-8,65	-7,70	-7,79	-7,96	-8,28	-7,82	-8,66	-8,87	-10,53	-11,69	-13,37	-14,57	-16,28	-17,40	-15,85	-13,46	-7,07	5,55	16,61	23,35	22,98	19,67	15,52	10,55	6,67	11,62	9,03
3,48	-11,02	-8,59	-8,13	-6,58	-5,92	-6,17	-6,38	-7,59	-7,88	-7,33	-8,02	-8,01	-10,38	-12,16	-15,38	-17,05	-19,11	-17,52	-16,15	-11,88	-5,53	2,08	10,14	13,85	15,37	13,81	12,23	7,58	8,68	1,22
6,36	-7,07	-2,43	-1,63	-1,60	-3,34	-3,84	-6,16	-6,34	-6,45	-6,57	-5,78	-7,15	-8,77	-11,68	-14,95	-17,58	-20,59	-19,94	-19,56	-17,36	-12,23	-7,00	1,47	7,34	12,78	13,89	11,74	3,33	-0,46	-27,86
7,81	-1,63	3,11	5,14	3,54	1,31	-0,56	-2,09	-4,33	-6,34	-6,21	-6,02	-6,84	-8,76	-11,86	-15,21	-16,86	-19,83	-21,13	-21,73	-19,60	-17,64	-13,76	-6,82	-0,71	4,71	6,55	1,89	-7,26	-28,94	-24,32
12,95	3,44	10,22	12,41	10,08	7,24	4,92	2,07	-0,33	-2,36	-3,07	-4,56	-6,92	-10,23	-11,59	-15,09	-16,80	-19,78	-21,07	-22,68	-22,61	-21,83	-20,26	-15,83	-9,44	-6,98	-6,36	-11,48	-18,31	-24,85	-24,20
10,85	7,28	16,08	19,36	19,22	16,70	12,81	8,54	5,86	2,72	2,05	-1,23	-5,21	-9,01	-11,72	-14,45	-16,29	-18,34	-19,68	-21,35	-21,38	-22,79	-22,55	-22,62	-18,93	-19,40	-19,93	-23,46	-23,91	-24,25	-21,52
14,55	7,98	17,79	25,15	28,15	26,86	21,32	16,53	13,47	11,11	9,38	5,20	-0,53	-5,91	-10,07	-13,12	-14,19	-17,40	-17,85	-19,65	-18,84	-21,49	-22,62	-25,20	-24,20	-24,57	-24,24	-24,10	-20,10	-20,15	9,75
16,06	7,56	18,44	28,87	33,98	34,88	31,62	27,22	22,69	20,01	16,11	14,92	9,35	2,28	-4,43	-7,91	-10,31	-14,77	-15,45	-16,45	-15,89	-17,85	-20,39	-23,52	-24,25	-23,52	-22,28	-18,43	-12,89	10,45	3,63
11,35	4,04	14,05	25,58	34,82	38,88	36,85	33,79	29,71	27,62	24,46	22,17	17,65	11,78	5,41	0,40	-4,44	-9,28	-11,28	-12,60	-12,39	-14,75	-17,78	-21,53	-23,01	-21,20	-19,03	-13,42	-6,26	4,57	3,18
9,38	1,37	10,60	19,31	29,69	36,92	39,12	37,35	35,69	35,12	32,63	31,15	27,59	23,82	17,69	11,01	4,61	-1,71	-5,16	-6,93	-8,18	-12,06	-16,69	-20,13	-20,42	-18,55	-14,46	-7,04	-2,19	4,25	-8,60
4,05	-2,52	4,04	12,00	21,60	29,04	34,47	36,01	38,72	39,65	39,78	39,05	37,37	33,61	29,60	22,16	14,08	7,13	3,08	-0,26	-5,11	-11,64	-16,97	-19,21	-18,38	-13,49	-7,47	-3,23	-2,62	-7,49	-6,68
6,25	-3,74	1,27	6,59	14,49	20,21	26,93	32,80	38,92	43,33	44,05	45,03	45,17	44,34	42,37	35,04	26,14	16,43	11,65	4,56	-5,04	-14,35	-19,52	-18,65	-13,79	-6,93	-1,01	0,08	-3,50	-5,60	-5,04
4,86	-4,41	-0,82	3,98	8,29	12,38	18,47	26,73	35,30	41,23	44,55	47,22	49,14	51,17	51,14	44,82	33,98	21,35	12,67	2,69	-8,82	-18,07	-21,22	-15,36	-8,55	-0,78	1,02	-2,03	-4,78	-4,05	-3,67
1,75	-5,69	-1,38	1,03	3,87	7,45	11,02	19,78	28,87	37,38	42,33	47,69	50,36	54,21	56,03	50,61	38,69	24,00	11,27	-2,78	-16,37	-22,71	-22,96	-14,23	-5,57	0,05	-2,30	-4,51	-5,40	-2,81	-2,57
-4,24	-7,71	-3,57	-0,42	1,10	4,31	8,49	15,86	23,58	32,75	40,39	46,46	50,88	54,50	54,11	46,48	32,36	16,46	0,50	-14,77	-24,60	-25,19	-20,70	-10,22	-3,78	-1,38	-4,92	-5,28	-4,31	-1,83	-1,71
-8,25	-9,66	-6,55	-3,52	-1,18	2,81	7,76	14,88	23,36	33,28	40,67	46,51	50,68	51,03	47,36	35,43	22,98	7,72	-7,61	-21,28	-27,52	-23,50	-15,39	-7,26	-5,14	-5,00	-5,76	-4,29	-3,46	-1,09	-1,07
-4,42	-11,67	-9,48	-6,43	-3,12	2,81	8,67	15,70	24,07	33,85	42,09	46,73	46,66	42,73	32,71	21,39	9,48	-4,27	-18,14	-28,35	-26,14	-16,66	-9,05	-5,36	-4,63	-4,80	-4,80	-3,51	-2,80	-0,54	-0,62
-9,88	-10,88	-10,51	-7,32	-3,89	2,14	8,08	17,17	24,56	32,32	37,49	38,98	33,68	26,44	17,02	9,20	-1,27	-13,65	-23,18	-28,08	-20,55	-10,72	-4,70	-4,54	-6,24	-5,75	-4,01	-2,90	-2,32	-0,16	-0,32
-2,72	-10,42	-9,77	-8,32	-5,64	0,64	6,82	15,10	21,64	26,50	26,66	25,05	16,53	9,94	5,07	-0,33	-10,47	-21,59	-26,91	-26,62	-15,87	-8,79	-3,43	-6,84	-7,00	-4,87	-3,38	-2,44	-1,98	0,09	-0,15
-4,03	-8,39	-7,39	-6,56	-4,50	0,16	5,71	12,32	17,13	20,16	16,38	10,69	3,95	-0,02	-2,42	-7,46	-16,37	-23,34	-24,55	-22,15	-13,27	-7,98	-4,33	-6,33	-5,95	-4,17	-2,91	-2,13	-1,76	0,24	-0,06
-5,86	-8,85	-7,44	-6,17	-4,61	-0,45	4,58	9,63	13,79	15,06	12,37	6,61	1,63	-1,76	-4,74	-12,52	-18,27	-21,16	-20,33	-17,89	-9,92	-7,46	-4,52	-7,10	-5,16	-3,69	-2,63	-1,97	-1,66	0,30	-0,02
-3,24	-7,87	-7,01	-6,21	-4,09	0,67	5,27	8,84	12,41	14,95	15,37	10,53	5,27	-0,59	-7,23	-12,80	-16,47	-16,35	-14,53	-13,07	-8,02	-6,37	-4,12	-6,23	-4,69	-3,48	-2,56	-1,96	-1,67	0,30	0,00
-3,19	-6,49	-6,17	-5,80	-4,03	0,39	4,62	9,76	11,77	14,59	16,10	12,15	6,55	-2,84	-8,19	-12,63	-12,26	-12,16	-10,37	-9,91	-6,75	-5,88	-5,25	-5,84	-4,62	-3,57	-2,73	-2,12	-1,79	0,24	0,00
-0,74	-4,76	-4,97	-4,00	-2,56	1,56	4,48	8,19	10,66	12,75	13,32	8,20	1,22	-6,76	-9,89	-11,25	-10,89	-8,84	-8,08	-6,58	-5,28	-5,12	-5,04	-6,00	-5,00	-4,03	-3,17	-2,47	-2,03	0,12	0,00
0,00	2,24	1,49	0,07	4,21	9,06	7,67	8,98	10,89	13,24	15,84	1,52	-12,90	-10,54	-8,49	-7,81	-6,49	-4,50	-3,77	-3,21	-2,71	10,81	-5,57	-4,81	-3,91	-2,91	-1,91	-1,03	-0,38	-0,06	0,00

TABELA 5.12 - VALORES DA SUPERFÍCIE DE ERRO OBTIDA COM A INTERPOLAÇÃO B-SPLINE (METROS)

2,94	5,73	2,77	4,89	4,89	4,37	0,89	4,07	2,79	1,00	4,56	0,96	-1,42	0,93	1,00	0,89	-5,28	-0,29	4,08	13,52	10,49	15,94	16,99	5,49	4,50	3,29	2,05	2,19	-0,99	-6,75	-6,42
4,55	3,02	5,78	7,31	8,17	7,20	3,26	9,22	7,95	3,28	2,99	-5,39	-3,40	-3,48	-0,69	0,04	0,74	-7,53	4,28	4,71	10,28	22,02	22,16	9,74	9,04	6,47	3,02	2,84	-0,97	-5,49	-9,82
3,39	-13,33	-11,74	-7,73	-5,02	-4,53	-4,03	-3,35	-3,66	-5,23	-8,36	-9,47	-10,00	-8,33	-8,55	-8,67	-9,73	-9,88	-9,28	-5,57	0,77	6,77	6,75	4,05	0,27	-1,94	-4,81	-5,49	-8,11	-3,56	-16,67
2,86	-14,20	-10,91	-5,50	-2,91	-1,89	-0,89	-0,52	-1,95	-3,53	-5,57	-6,42	-5,44	-4,94	-4,02	-5,78	-5,32	-5,97	-6,08	-2,49	4,17	12,78	15,60	12,36	7,11	2,85	0,25	-2,14	-4,60	-0,91	-2,72
2,46	-15,27	-11,64	-5,04	-1,51	-1,74	-1,07	-1,00	-1,67	-2,38	-3,44	-4,11	-3,65	-2,31	-2,22	-2,49	-3,31	-4,10	-5,36	-3,08	2,74	14,58	20,47	19,56	12,49	6,76	3,02	0,73	-2,47	2,62	0,40
2,15	-13,62	-10,03	-5,47	-1,98	-1,27	-0,72	-0,84	-0,72	-1,60	-1,70	-2,21	-2,28	-2,10	-1,81	-1,56	-2,57	-3,31	-4,34	-4,79	-1,15	9,51	15,84	16,40	11,47	7,03	3,90	1,64	-1,12	7,75	1,47
2,97	-11,16	-7,97	-4,69	-2,26	-1,46	-1,81	-1,91	-1,83	-0,75	-0,82	-0,11	-0,69	-0,65	-1,15	-1,39	-2,61	-4,30	-5,00	-6,94	-6,11	0,83	7,13	10,63	9,11	7,01	5,45	3,25	1,23	6,29	1,33
5,08	-9,72	-6,27	-4,53	-2,23	-1,27	-1,32	-1,14	-1,83	-1,53	-0,39	-0,36	0,61	-0,44	-0,73	-2,51	-3,08	-4,88	-4,39	-5,88	-5,74	-4,17	-1,29	2,87	4,35	5,87	6,00	6,98	4,95	7,52	-0,83
4,90	-8,82	-3,44	-1,52	-0,54	-1,48	-1,20	-2,63	-1,89	-1,19	-0,64	0,87	0,53	0,45	-0,64	-2,06	-3,14	-5,27	-4,81	-6,02	-6,61	-5,16	-4,15	0,38	3,51	8,10	10,07	10,15	4,79	3,46	-23,67
2,94	-6,94	-1,91	0,86	0,23	-0,86	-1,42	-1,51	-2,32	-3,08	-1,93	-0,72	-0,19	-0,24	-1,15	-2,31	-2,09	-3,77	-4,61	-5,79	-5,24	-5,81	-5,22	-1,67	1,81	5,98	8,00	4,81	-1,89	-21,45	-16,50
4,74	-5,47	0,95	3,24	1,60	-0,02	-0,77	-1,77	-2,25	-2,46	-1,54	-1,40	-1,84	-2,78	-1,55	-2,56	-2,22	-3,68	-4,09	-5,55	-6,07	-6,66	-7,18	-5,07	-0,63	0,72	0,96	-4,00	-10,43	-17,00	-17,56
-0,17	-4,73	2,98	5,58	5,56	3,90	1,50	-0,79	-1,21	-2,01	-0,40	-1,34	-2,75	-3,62	-3,30	-3,25	-2,84	-3,18	-3,31	-4,28	-4,10	-5,81	-6,37	-7,48	-4,77	-6,00	-7,38	-12,24	-14,83	-18,00	-18,63
1,75	-6,00	2,12	7,94	10,15	8,92	4,28	1,21	0,45	0,69	1,68	0,36	-2,25	-4,22	-4,91	-4,81	-3,27	-4,42	-3,28	-3,87	-2,20	-4,25	-5,02	-7,50	-6,67	-7,56	-8,50	-10,86	-10,80	-15,62	9,61
2,88	-6,76	2,06	10,16	13,32	12,98	9,48	6,06	3,44	3,20	2,01	3,79	1,55	-1,81	-4,66	-4,48	-3,68	-5,44	-3,89	-3,13	-1,12	-1,81	-3,19	-5,49	-5,99	-5,94	-6,43	-5,50	-4,05	14,67	3,60
-1,12	-9,32	-1,46	7,20	13,36	14,75	11,08	7,87	4,85	4,55	3,59	3,92	2,53	0,39	-1,90	-2,77	-3,64	-4,85	-3,72	-2,48	-0,15	-0,67	-2,07	-4,71	-6,03	-5,32	-5,27	-2,44	1,62	9,36	5,24
-1,81	-10,23	-2,88	2,81	9,43	12,87	12,08	8,80	6,86	6,89	5,58	5,88	4,83	4,38	2,24	0,06	-1,57	-3,15	-2,22	-0,26	1,56	0,14	-2,68	-5,18	-5,72	-5,44	-3,69	1,36	4,39	9,48	-4,42
-5,59	-11,96	-6,67	-1,36	4,45	7,62	9,08	7,63	8,35	8,14	7,95	7,74	7,52	6,24	5,75	2,87	0,32	-0,62	1,28	3,25	2,80	-0,55	-4,16	-6,04	-6,12	-3,19	0,49	2,82	2,47	-2,66	-1,90
-1,52	-10,71	-6,28	-2,97	1,56	3,00	5,12	6,64	8,93	10,30	8,99	9,03	9,40	10,07	11,04	8,31	5,69	3,30	6,06	5,94	2,03	-3,51	-7,29	-6,93	-3,75	0,90	4,69	4,28	0,18	-1,77	-0,87
-0,53	-8,64	-5,13	-1,78	-0,33	-0,30	0,86	3,73	6,93	8,11	7,82	8,19	9,25	12,04	14,65	13,17	9,44	5,36	5,56	3,72	-1,30	-6,64	-9,01	-4,61	-0,22	5,20	5,15	0,94	-2,16	-1,27	-0,56
-0,72	-6,98	-2,64	-1,46	-1,15	-1,40	-2,85	-0,07	2,70	5,31	5,50	7,57	8,72	13,11	17,81	17,85	13,98	8,84	5,77	0,20	-7,03	-10,00	-10,34	-3,95	1,72	4,96	1,03	-2,06	-3,30	-0,71	-0,31
-3,89	-6,35	-2,25	-0,28	-1,14	-1,59	-2,39	-1,19	-0,18	2,58	4,92	7,30	10,08	14,51	17,58	16,19	10,91	5,14	-1,06	-8,42	-12,87	-11,18	-7,69	-0,19	2,95	3,05	-1,84	-2,92	-2,37	-0,13	-0,05
-5,97	-6,44	-3,37	-1,44	-1,39	-0,95	-0,92	0,08	1,91	5,56	7,91	10,43	13,47	15,29	15,74	10,61	7,35	2,15	-4,11	-11,24	-13,76	-8,89	-2,56	2,35	1,25	-0,75	-2,69	-1,89	-1,59	0,35	0,17
-1,66	-7,78	-5,31	-3,10	-1,96	0,43	1,43	2,52	4,66	8,91	13,19	15,76	15,73	14,12	8,74	4,32	1,28	-3,28	-9,54	-15,28	-11,48	-2,66	2,69	3,42	1,41	-0,62	-1,73	-1,17	-1,11	0,62	0,28
-7,60	-7,14	-6,02	-3,29	-1,91	0,58	1,72	5,18	6,99	10,30	13,02	14,10	10,27	6,25	1,78	0,58	-1,92	-6,54	-10,40	-13,09	-6,06	1,77	5,36	3,13	-0,62	-1,65	-1,00	-0,74	-0,89	0,73	0,33
-1,14	-7,14	-5,37	-4,06	-3,29	-0,49	1,06	4,15	5,88	7,39	6,53	6,00	0,19	-2,43	-2,25	-1,58	-5,00	-10,09	-11,69	-11,10	-2,40	1,86	4,87	-0,31	-1,86	-0,94	-0,51	-0,50	-0,80	0,75	0,33
-2,83	-5,48	-3,24	-2,43	-2,16	-0,80	0,46	2,47	3,25	3,83	-0,01	-3,72	-7,03	-6,64	-4,06	-3,76	-7,37	-10,07	-9,21	-7,65	-1,43	0,91	2,49	-0,79	-1,36	-0,54	-0,26	-0,39	-0,78	0,75	0,33
-4,83	-6,30	-3,77	-2,52	-2,65	-1,53	-0,34	0,75	1,63	1,21	-0,88	-4,15	-5,40	-4,39	-2,74	-6,08	-8,02	-8,31	-6,73	-5,70	-0,35	-0,48	0,85	-2,56	-1,20	-0,45	-0,25	-0,42	-0,82	0,72	0,32
-2,50	-5,87	-4,11	-3,44	-2,91	-0,91	0,31	0,51	1,51	3,08	4,76	2,88	1,50	-0,26	-3,00	-5,28	-6,59	-5,31	-3,78	-4,09	-1,47	-1,89	-0,62	-2,96	-1,50	-0,69	-0,46	-0,61	-0,96	0,63	0,28
-3,12	-5,40	-4,43	-4,34	-4,11	-2,21	-0,98	1,28	1,32	3,95	7,60	7,31	5,79	-0,07	-2,66	-5,16	-3,76	-3,62	-2,92	-4,61	-3,86	-4,64	-4,24	-4,17	-2,28	-1,21	-0,87	-0,96	-1,22	0,47	0,21
-1,78	-4,98	-4,82	-4,34	-4,47	-2,75	-2,63	-1,49	-0,54	2,07	5,73	5,13	2,51	-2,55	-4,07	-4,74	-4,32	-2,93	-3,74	-4,72	-5,90	-7,07	-6,43	-5,77	-3,27	-1,85	-1,36	-1,37	-1,53	0,28	0,13
-2,53	0,33	-0,37	-2,58	-0,23	2,11	-2,11	-3,33	-2,72	0,64	7,13	-1,89	-11,58	-6,62	-3,71	-3,08	-2,13	-0,96	-1,86	-3,83	-5,76	6,75	-8,39	-5,19	-2,14	-0,40	0,23	0,28	0,21	0,13	0,06

TABELA 5.13 - VALORES DA SUPERFÍCIE DE ERRO OBTIDA COM A INTERPOLAÇÃO β-SPLINE (METROS)

2,47	5,25	2,48	4,29	4,23	4,29	0,71	3,67	2,76	0,86	4,21	0,95	-1,24	0,80	0,56	0,76	-5,51	-1,24	3,57	13,18	8,87	13,69	15,67	5,16	3,80	2,43	2,15	1,94	-1,58	-6,19	-5,55
4,04	2,56	5,67	6,96	7,70	7,17	3,09	8,87	8,00	3,22	2,72	-5,32	-3,17	-3,54	-0,96	0,06	0,67	-8,21	4,09	4,62	8,84	19,98	21,01	9,45	8,33	5,66	3,23	2,75	-1,45	-4,96	-9,02
3,04	-13,49	-11,26	-7,33	-4,87	-4,15	-3,89	-3,36	-3,23	-4,90	-8,22	-8,95	-9,33	-7,86	-8,12	-7,89	-8,98	-9,53	-8,38	-4,70	0,13	5,48	6,21	4,09	-0,21	-2,45	-4,24	-5,17	-8,29	-2,99	-15,97
2,51	-14,31	-10,22	-4,82	-2,56	-1,42	-0,69	-0,42	-1,39	-3,04	-5,26	-5,74	-4,67	-4,38	-3,46	-4,98	-4,64	-5,64	-5,31	-1,95	3,12	11,10	14,63	11,83	6,12	2,03	0,60	-1,94	-4,89	-0,57	-2,29
1,95	-15,59	-11,17	-4,57	-1,40	-1,51	-1,07	-1,08	-1,32	-2,13	-3,37	-3,70	-3,23	-2,11	-1,96	-2,12	-3,11	-4,12	-4,95	-2,91	1,57	13,01	19,52	18,83	11,31	5,83	3,19	0,77	-2,85	2,89	0,80
1,61	-13,98	-9,61	-5,06	-1,91	-1,06	-0,73	-0,98	-0,52	-1,56	-1,86	-2,04	-2,15	-2,12	-1,61	-1,22	-2,27	-2,90	-3,39	-3,91	-1,14	9,37	16,07	16,41	10,84	6,56	4,30	1,77	-1,37	8,14	1,98
2,59	-11,32	-7,37	-4,08	-1,95	-0,99	-1,57	-1,85	-1,49	-0,61	-0,89	0,11	-0,59	-0,72	-0,96	-1,16	-2,50	-3,99	-4,31	-6,42	-6,24	0,61	7,02	10,06	7,92	6,04	5,22	2,71	0,37	6,08	1,20
4,74	-9,88	-5,79	-4,10	-2,04	-0,85	-1,12	-1,11	-1,46	-1,35	-0,42	-0,11	0,68	-0,55	-0,52	-2,39	-3,18	-4,76	-4,09	-5,85	-6,18	-4,62	-1,72	2,00	3,03	4,85	5,57	6,21	4,02	7,45	-0,76
4,41	-9,24	-3,45	-1,68	-0,84	-1,45	-1,34	-2,87	-1,70	-1,13	-0,79	1,02	0,46	0,25	-0,33	-1,80	-3,04	-4,76	-4,09	-5,43	-6,06	-4,44	-3,45	0,70	3,46	8,18	10,47	10,13	4,87	4,72	-22,03
2,55	-7,23	-1,91	0,76	0,13	-0,54	-1,24	-1,41	-1,76	-2,69	-1,81	-0,32	-0,12	-0,34	-0,69	-1,96	-1,96	-3,19	-3,94	-5,30	-4,62	-5,08	-4,69	-1,53	1,55	5,73	7,94	4,47	-1,85	-20,00	-14,43
4,60	-5,53	1,00	3,17	1,62	0,37	-0,58	-1,66	-1,67	-2,10	-1,45	-0,96	-1,76	-2,92	-1,09	-2,30	-2,28	-3,31	-3,79	-5,54	-5,87	-6,47	-7,27	-5,37	-1,19	0,13	0,61	-4,47	-10,70	-16,46	-16,65
-0,27	-4,93	2,53	4,87	4,94	3,54	0,90	-1,41	-1,30	-2,31	-0,83	-1,21	-2,89	-3,92	-2,86	-2,98	-2,90	-2,73	-2,96	-4,16	-3,57	-5,23	-5,92	-6,83	-4,15	-5,38	-6,52	-11,47	-14,48	-18,04	-18,93
1,45	-6,44	1,29	6,94	9,39	8,41	3,63	0,72	0,56	0,61	1,54	0,91	-1,98	-4,12	-4,03	-4,16	-3,05	-3,69	-2,76	-3,66	-1,51	-3,61	-4,54	-6,62	-5,71	-6,67	-7,37	-9,69	-10,29	-16,08	8,57
2,81	-6,92	1,49	9,59	13,02	12,71	9,03	5,85	3,75	3,27	2,01	4,41	1,79	-1,78	-3,83	-3,98	-3,63	-4,78	-3,50	-3,15	-0,67	-1,54	-3,19	-5,04	-5,49	-5,65	-6,00	-4,93	-3,83	14,27	2,90
-0,62	-8,96	-1,60	7,03	13,23	14,13	10,07	7,06	4,39	3,79	2,77	3,65	1,85	-0,44	-1,81	-2,99	-4,19	-4,52	-3,59	-2,78	0,15	-0,53	-2,19	-4,25	-5,44	-5,08	-5,13	-2,23	1,84	9,52	5,46
-1,59	-10,19	-3,37	2,35	8,92	11,65	10,53	7,60	6,01	5,88	4,68	5,57	4,16	3,70	2,54	-0,02	-1,96	-2,56	-1,96	-0,58	1,87	0,27	-2,86	-4,74	-5,06	-5,16	-3,65	1,35	4,51	9,83	-3,90
-5,56	-12,06	-7,21	-1,70	4,17	6,68	8,03	6,99	7,84	7,49	7,50	7,77	7,16	5,88	6,24	2,83	-0,04	0,12	1,59	2,93	3,28	-0,17	-4,27	-5,70	-5,62	-3,01	0,47	2,70	2,43	-2,51	-1,66
-0,96	-10,21	-6,08	-2,47	2,15	2,96	5,03	6,74	8,63	9,62	8,42	8,64	8,47	9,05	10,64	7,20	4,40	3,40	5,86	5,30	2,66	-2,74	-7,17	-6,66	-3,46	0,95	4,64	4,16	0,10	-1,69	-0,75
-0,41	-8,54	-5,27	-1,57	-0,01	-0,58	0,65	3,64	6,20	7,02	6,93	7,37	7,88	10,60	13,72	11,39	7,50	4,96	4,88	2,64	-0,87	-5,86	-8,91	-4,52	-0,16	5,12	5,05	0,82	-2,27	-1,24	-0,49
-0,80	-7,11	-3,09	-1,59	-1,17	-1,96	-3,21	-0,23	1,91	4,33	4,87	6,96	7,59	12,07	17,37	16,54	12,51	8,98	5,55	-0,61	-6,51	-9,15	-10,21	-3,94	1,68	4,86	1,00	-2,10	-3,37	-0,71	-0,28
-3,02	-5,61	-1,96	0,25	-0,51	-1,49	-2,04	-0,67	-0,39	2,14	4,76	6,97	9,13	13,74	17,58	15,41	10,07	6,08	-0,35	-8,44	-11,85	-9,96	-7,25	-0,05	2,94	3,03	-1,71	-2,76	-2,29	-0,08	-0,02
-5,18	-5,84	-3,28	-1,16	-1,02	-1,13	-0,84	0,32	1,34	4,67	7,20	9,41	11,71	13,68	14,99	9,16	5,92	2,66	-3,55	-11,31	-12,89	-7,78	-2,10	2,47	1,18	-0,80	-2,55	-1,67	-1,46	0,39	0,16
-1,40	-7,65	-5,57	-3,11	-1,90	-0,08	1,16	2,45	3,83	7,77	12,33	14,83	14,15	12,64	8,08	2,97	-0,16	-2,92	-9,05	-15,46	-11,04	-2,05	2,90	3,46	1,35	-0,69	-1,66	-1,03	-1,04	0,61	0,25
-7,11	-6,68	-5,74	-2,65	-1,23	0,62	1,96	5,65	6,80	9,84	13,08	14,55	10,31	6,37	2,70	0,83	-1,93	-5,12	-8,97	-12,62	-5,67	2,01	5,37	3,22	-0,51	-1,60	-0,91	-0,62	-0,83	0,70	0,28
-0,94	-6,93	-5,24	-3,55	-2,82	-0,73	0,92	4,21	5,29	6,45	6,12	6,20	0,08	-2,52	-1,51	-1,33	-4,96	-8,75	-10,20	-10,52	-2,19	1,80	4,73	-0,21	-1,67	-0,84	-0,43	-0,41	-0,76	0,72	0,29
-2,85	-5,52	-3,37	-2,23	-2,01	-1,32	0,00	2,21	2,41	2,57	-0,87	-4,02	-7,71	-7,38	-3,97	-3,99	-7,81	-9,43	-8,31	-7,37	-1,41	0,75	2,35	-0,69	-1,19	-0,47	-0,22	-0,33	-0,74	0,71	0,28
-4,41	-5,91	-3,45	-1,89	-2,03	-1,46	-0,17	1,20	1,71	0,91	-0,86	-3,60	-5,22	-4,32	-1,91	-5,49	-7,85	-7,59	-5,94	-5,38	-0,23	-0,49	0,86	-2,38	-1,01	-0,38	-0,21	-0,36	-0,78	0,69	0,27
-2,14	-5,54	-3,88	-2,96	-2,45	-0,96	0,29	0,73	1,42	2,57	4,46	3,11	1,43	-0,42	-2,37	-4,67	-6,34	-4,73	-3,12	-3,70	-1,20	-1,73	-0,44	-2,64	-1,22	-0,57	-0,39	-0,53	-0,90	0,62	0,24
-3,02	-5,33	-4,47	-4,18	-3,98	-2,54	-1,33	1,08	0,85	3,00	6,84	7,24	5,56	-0,51	-2,52	-4,97	-3,77	-3,37	-2,50	-4,35	-3,80	-4,75	-4,24	-3,90	-1,99	-1,10	-0,82	-0,90	-1,18	0,46	0,18
-1,46	-4,69	-4,63	-3,96	-4,04	-2,61	-2,46	-1,12	-0,32	1,86	5,83	6,18	3,56	-1,98	-3,44	-4,26	-4,13	-2,70	-3,45	-4,65	-6,26	-7,73	-6,85	-5,68	-3,04	-1,81	-1,38	-1,38	-1,54	0,26	0,10
-2,16	0,67	-0,12	-2,18	0,25	2,37	-1,82	-2,84	-2,33	0,57	7,40	-0,52	-10,08	-5,73	-3,02	-2,59	-1,87	-0,76	-1,59	-3,78	-6,26	5,88	-8,94	-5,10	-1,88	-0,36	0,19	0,24	0,18	0,10	0,04

TABELA 5.14 - VALORES DA SUPERFÍCIE DE ERRO OBTIDA COM A INTERPOLAÇÃO AKIMA (METROS)

0,00	1,71	-2,13	0,00	1,38	2,67	0,00	2,99	1,61	0,00	3,94	1,00	0,00	3,04	2,04	0,00	-8,06	-4,31	0,00	6,78	-2,01	0,00	3,32	-3,04	0,00	0,24	-0,48	0,00	-2,38	-5,75	0,00
2,35	3,08	6,85	8,66	9,41	8,14	4,05	10,39	9,89	6,00	6,30	-1,32	2,08	2,72	4,66	4,11	3,69	-5,35	6,65	3,10	0,17	6,56	8,11	0,19	3,76	4,13	2,82	3,16	-0,75	-4,01	-5,11
1,04	-11,26	-7,58	-3,24	-1,65	-2,83	-3,01	-1,48	-0,41	-0,90	-3,41	-3,73	-2,96	-0,92	-2,16	-2,89	-4,23	-4,90	-3,93	-4,74	-8,24	-8,22	-7,10	-6,01	-5,81	-4,44	-4,17	-3,90	-6,84	-1,66	-12,89
0,00	-11,50	-5,66	0,00	1,00	-0,17	0,00	1,17	0,79	0,00	-1,63	-1,96	0,00	1,00	1,60	0,00	0,54	-0,49	0,00	-0,46	-3,12	0,00	3,27	2,09	0,00	-0,14	1,08	0,00	-2,90	0,94	0,00
-0,77	-12,55	-6,20	0,90	2,34	0,10	-0,02	0,38	0,00	-0,51	-1,42	-1,83	-0,80	1,14	1,69	2,07	1,66	0,97	0,52	-0,12	-1,43	6,06	10,82	9,20	3,93	2,70	3,27	2,42	-1,36	3,79	2,28
-1,15	-11,09	-4,75	-0,25	1,65	0,85	0,67	0,44	0,45	-0,59	-0,70	-0,98	-0,82	-0,51	0,05	1,04	1,06	1,24	1,09	-1,19	-2,07	5,46	9,08	6,41	1,80	1,56	2,46	1,44	-1,90	7,32	2,56
0,00	-8,87	-3,18	0,00	1,18	0,98	0,00	-0,48	-0,74	0,00	-0,22	0,56	0,00	-0,05	-0,44	0,00	0,08	-0,21	0,00	-3,00	-4,84	0,00	2,96	2,01	0,00	0,99	1,70	0,00	-2,58	2,82	0,00
2,30	-8,46	-3,15	-1,33	0,40	1,00	0,71	0,72	-0,12	-0,02	0,67	0,09	0,72	-0,19	0,02	-1,04	-0,58	-1,17	0,32	-1,57	-3,10	-2,81	-2,66	-2,63	-2,39	0,37	0,95	1,76	-0,24	4,13	0,67
1,96	-9,44	-2,96	-0,67	0,60	0,04	0,66	-0,38	0,74	1,62	1,44	1,57	0,39	0,66	0,50	-0,07	-0,52	-1,81	-0,26	-1,13	-2,44	-1,55	-2,32	-1,20	-0,44	3,48	5,17	4,99	0,85	4,51	-12,83
0,00	-9,07	-3,48	0,00	0,24	-0,08	0,00	0,58	0,42	0,00	0,52	0,41	0,00	0,23	0,32	0,00	0,70	-0,30	0,00	-0,26	0,51	0,00	-0,83	-0,48	0,00	2,40	3,45	0,00	-4,45	-17,06	0,00
1,86	-9,31	-2,97	0,07	-0,37	-0,84	-0,66	-0,94	-0,82	-0,60	0,24	-0,12	-1,04	-1,56	0,84	0,64	1,28	0,22	0,54	-0,06	0,09	-0,02	-1,71	-2,42	-0,32	0,06	0,13	-4,38	-9,46	-13,02	-8,26
-2,87	-10,50	-3,89	-0,85	0,44	0,36	-0,59	-1,90	-1,65	-1,72	0,55	-0,10	-1,35	-1,31	0,55	1,39	1,64	1,08	0,91	0,24	0,95	-0,10	-0,99	-3,44	-1,47	-1,84	-1,46	-4,65	-8,14	-15,67	-22,61
0,00	-12,05	-5,87	0,00	3,12	2,99	0,00	-1,56	-1,38	0,00	2,28	1,93	0,00	-1,02	-0,57	0,00	1,11	-0,67	0,00	-0,62	1,41	0,00	-0,17	-2,00	0,00	0,49	0,81	0,00	-1,61	-13,86	0,00
2,68	-11,09	-4,59	2,89	5,57	4,31	0,79	-0,72	-0,41	1,70	1,62	4,08	2,67	0,37	-1,70	-1,24	-0,68	-2,83	-1,53	-0,93	0,93	0,46	0,34	0,31	2,26	2,55	0,46	1,39	2,40	16,75	-1,32
-0,01	-11,48	-5,81	1,56	5,72	3,95	-2,49	-4,28	-2,25	1,13	0,60	0,71	0,22	-0,13	-1,17	-1,45	-2,09	-3,06	-1,66	-0,87	0,41	-0,49	-0,35	-0,09	1,36	1,14	-2,54	-1,02	3,87	11,28	6,29
0,00	-10,67	-4,79	0,00	4,62	4,45	0,00	-3,72	-2,83	0,00	-0,66	-0,37	0,00	1,93	1,34	0,00	-0,87	-1,56	0,00	1,38	1,85	0,00	-1,43	-1,64	0,00	-1,02	-3,34	0,00	4,25	11,07	0,00
-3,77	-11,33	-6,52	-1,34	3,21	3,98	2,81	-0,61	-1,07	-1,47	-0,97	0,17	1,87	2,19	2,61	0,70	-0,80	-0,97	1,53	3,92	3,86	1,37	-1,36	-2,79	-2,38	-0,68	0,21	1,43	1,93	-1,75	1,93
-0,13	-9,46	-4,71	-1,46	2,12	2,05	2,51	2,15	2,07	0,62	-1,34	0,71	2,63	2,93	3,35	1,63	0,58	-1,16	2,38	4,50	3,92	1,19	-2,35	-3,76	-2,20	1,22	3,69	2,98	-0,61	-1,72	0,96
0,00	-7,50	-3,21	0,00	0,54	-0,29	0,00	1,62	2,56	0,00	-2,41	-1,32	0,00	0,56	0,77	0,00	-0,61	-2,13	0,00	1,44	1,36	0,00	-2,64	-1,68	0,00	4,59	4,13	0,00	-2,81	-1,52	0,00
1,28	-5,25	-0,84	0,03	-0,35	-1,11	-2,89	-0,95	-0,33	-1,28	-4,15	-3,55	-3,68	-1,85	0,47	1,39	2,15	2,24	2,64	0,55	-1,92	-1,96	-4,02	-1,88	1,31	4,55	0,83	-2,05	-3,17	-0,49	0,00
1,52	-3,11	-0,27	1,03	-0,25	-0,97	-1,80	-1,27	-2,45	-3,29	-4,52	-4,68	-4,07	-1,29	1,55	1,88	1,44	1,98	0,06	-3,73	-4,20	-1,93	-2,27	0,72	1,93	2,79	-1,12	-1,69	-1,18	0,73	0,00
0,00	-2,89	-1,22	0,00	-0,32	-0,09	0,00	0,37	-0,02	0,00	-1,11	-0,95	0,00	1,09	2,95	0,00	1,01	1,95	0,00	-4,08	-3,95	0,00	1,88	2,60	0,00	-0,90	-1,48	0,00	0,19	1,54	0,00
2,02	-4,80	-2,75	-1,15	-0,74	1,18	2,01	2,30	2,22	3,09	5,07	7,31	7,06	5,54	1,32	-1,86	-1,38	-0,16	-2,77	-7,73	-4,16	3,01	5,61	3,89	1,00	-0,33	-0,52	0,44	0,34	1,56	0,00
-5,64	-4,13	-2,56	-0,47	-0,41	0,97	1,42	3,75	3,30	3,64	5,72	9,23	7,87	4,92	0,81	-0,57	-0,15	0,95	-0,16	-5,31	-2,33	2,91	6,16	4,15	0,44	-0,66	0,07	0,22	-0,16	1,15	0,00
0,00	-3,81	-0,73	0,00	-1,20	-0,27	0,00	1,73	1,23	0,00	-0,75	2,34	0,00	-1,05	-0,34	0,00	-0,75	-0,47	0,00	-3,72	-1,30	0,00	4,32	0,99	0,00	0,34	0,35	0,00	-0,54	0,84	0,00
-1,64	-2,11	1,68	2,20	0,43	-0,46	-0,74	0,01	-1,10	-2,75	-6,70	-7,88	-8,70	-6,45	-2,09	-1,08	-2,55	-1,45	0,56	-1,70	-0,73	-0,64	2,31	0,64	0,41	0,60	0,48	0,07	-0,49	0,84	0,00
-3,14	-3,13	0,40	1,60	-0,01	-0,97	-1,20	-1,07	-1,52	-3,30	-5,66	-7,86	-7,71	-4,52	-0,05	-1,88	-2,79	-1,70	-0,33	-1,68	1,09	-0,03	1,94	-0,85	0,37	0,57	0,50	0,15	-0,41	0,90	0,00
0,00	-2,63	-0,59	0,00	-0,42	-0,05	0,00	-0,60	-0,66	0,00	1,61	0,48	0,00	0,25	0,53	0,00	-1,21	-0,37	0,00	-1,55	0,47	0,00	1,41	-1,03	0,00	0,34	0,32	0,00	-0,49	0,86	0,00
-0,24	-2,21	-1,31	-1,33	-1,73	-1,08	-0,93	0,55	-0,32	1,13	5,12	6,84	6,41	1,66	1,40	0,19	1,12	0,21	-0,12	-2,58	-2,39	-3,35	-2,68	-2,29	-0,52	0,03	-0,09	-0,44	-0,82	0,67	0,00
0,88	-2,31	-2,25	-1,67	-2,01	-1,18	-2,08	-1,55	-1,25	-0,27	4,55	8,25	6,98	1,16	0,31	-0,18	-0,72	-0,27	-1,34	-3,09	-6,13	-8,56	-7,11	-4,56	-1,15	-0,31	-0,65	-1,11	-1,32	0,39	0,00
0,00	2,62	1,90	0,00	2,61	4,73	0,00	-1,58	-1,53	0,00	9,59	8,26	0,00	0,71	1,02	0,00	-0,20	0,48	0,00	-3,59	-9,67	0,00	-13,31	-5,71	0,00	1,26	0,63	0,00	0,00	0,00	0,00

TABELA 5.15 - DISTRIBUIÇÃO DE FREQÜÊNCIA DE ERROS DAS INTERPOLAÇÕES DE -∞ ATÉ 0,00

Intervalo	Linear	Lagrange	Bézier	B-Spline	β-Spline	Akima
-∞ —— -50,00	0	35	0	0	0	0
-49,99 —— -48,00	0	5	0	0	0	0
-47,99 —— -46,00	0	5	0	0	0	0
-45,99 —— -44,00	0	3	0	0	0	0
-43,99 —— -42,00	0	0	0	0	0	0
-41,99 —— -40,00	0	2	0	0	0	0
-39,99 —— -38,00	0	2	0	0	0	0
-37,99 —— -36,00	0	1	0	0	0	0
-35,99 —— -34,00	0	3	0	0	0	0
-33,99 —— -32,00	0	2	0	0	0	0
-31,99 —— -30,00	0	3	0	0	0	0
-29,99 —— -28,00	0	2	3	0	0	0
-27,99 —— -26,00	0	6	5	0	0	0
-25,99 —— -24,00	0	7	16	0	0	0
-23,99 —— -22,00	0	7	20	1	1	1
-21,99 —— -20,00	1	3	28	1	1	0
-19,99 —— -18,00	2	6	35	2	2	0
-17,99 —— -16,00	0	2	37	4	3	1
-15,99 —— -14,00	4	5	36	5	6	1
-13,99 —— -12,00	6	5	31	6	5	7
-11,99 —— -10,00	13	12	35	23	16	8
-9,99 —— -8,00	17	14	46	23	26	15
-7,99 —— -6,00	31	19	83	57	43	15
-5,99 —— -4,00	64	38	76	119	114	40
-3,99 —— -2,00	127	78	61	140	144	95
-1,99 —— 0,00	378	285	62	201	223	390

Os polígonos de freqüência de erros das interpolações Linear, Lagrange, Bézier, B-Spline, β-Spline e Akima estão apresentados nas Figuras 5.5, 5.6, 5.7, 5.8, 5.9 e 5.10, respectivamente. A Figura 5.11 apresenta uma comparação entre estes polígonos de freqüência.

TABELA 5.16 - DISTRIBUIÇÃO DE FREQÜÊNCIA DE ERROS DAS INTERPOLAÇÕES DE 0,01 ATÉ +∞

Intervalo	Linear	Lagrange	Bézier	B-Spline	β-Spline	Akima
0,01 —— 2,00	126	181	38	107	112	238
2,01 —— 4,00	70	71	31	73	73	87
4,01 —— 6,00	60	28	31	67	65	31
6,01 —— 8,00	31	28	26	44	50	16
8,01 —— 10,00	11	15	25	30	29	11
10,01 —— 12,00	10	9	25	18	16	4
12,01 —— 14,00	6	11	21	16	14	0
14,01 —— 16,00	1	8	22	14	9	0
16,01 —— 18,00	3	11	20	6	5	1
18,01 —— 20,00	0	4	7	1	3	0
20,01 —— 22,00	0	5	12	1	1	0
22,01 —— 24,00	0	4	12	2	0	0
24,01 —— 26,00	0	6	8	0	0	0
26,01 —— 28,00	0	2	13	0	0	0
28,01 —— 30,00	0	4	9	0	0	0
30,01 —— 32,00	0	2	6	0	0	0
32,01 —— 34,00	0	0	16	0	0	0
34,01 —— 36,00	0	1	8	0	0	0
36,01 —— 38,00	0	0	10	0	0	0
38,01 —— 40,00	0	0	11	0	0	0
40,01 —— 42,00	0	0	3	0	0	0
42,01 —— 44,00	0	0	5	0	0	0
44,01 —— 46,00	0	2	8	0	0	0
46,01 —— 48,00	0	3	8	0	0	0
48,01 —— 50,00	0	0	1	0	0	0
50,01 —— +∞	0	16	11	0	0	0

Analisando as tabelas de distribuição de freqüência e os polígonos de freqüência, pode-se observar claramente que os métodos de interpolação Linear e Akima foram os que tiveram uma maior freqüência de valores próximos de 0, sendo que a maior concentração foi obtida com a Interpolação Akima.

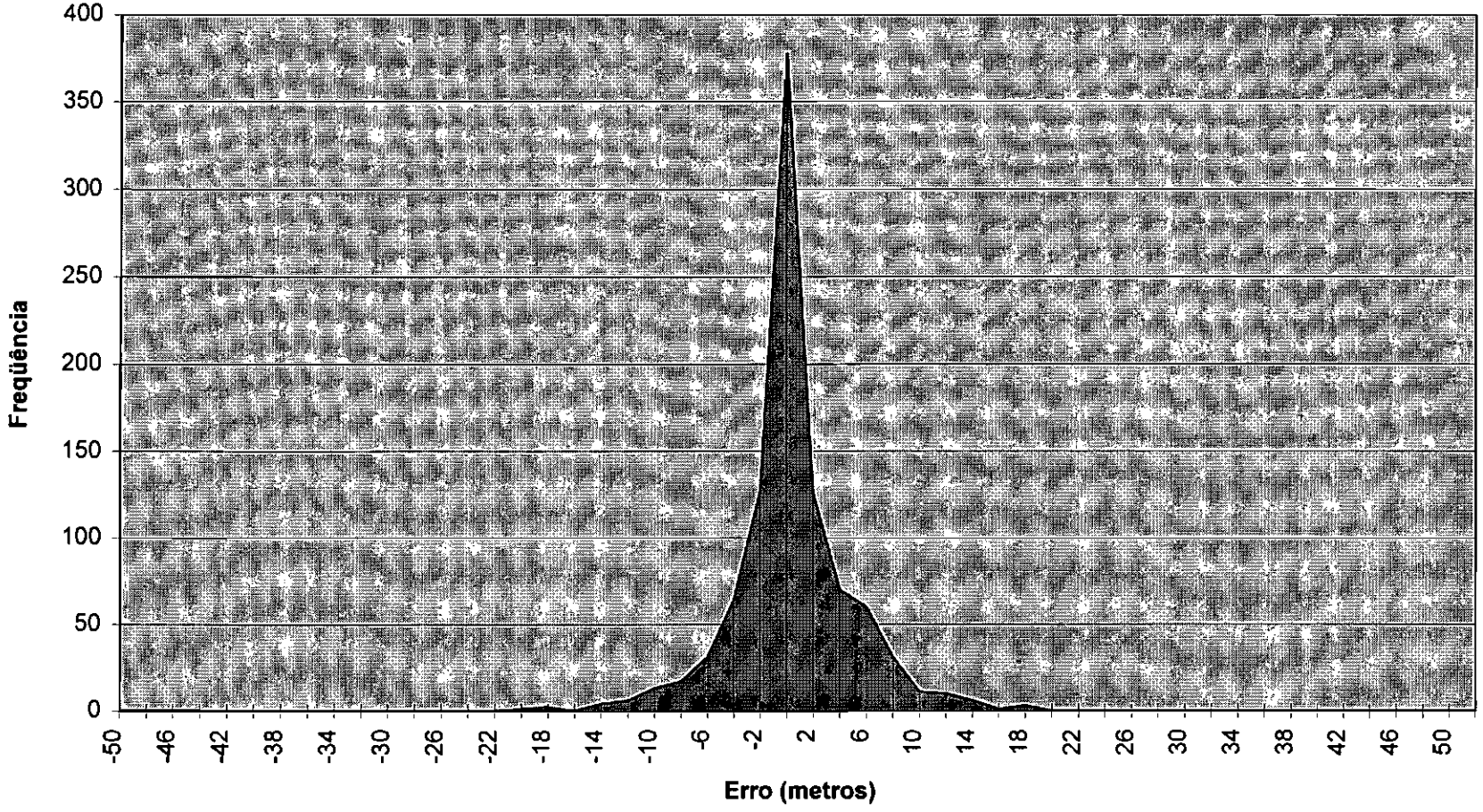


Fig. 5.5 - Polígono de Freqüência de Erros da Interpolação Linear.

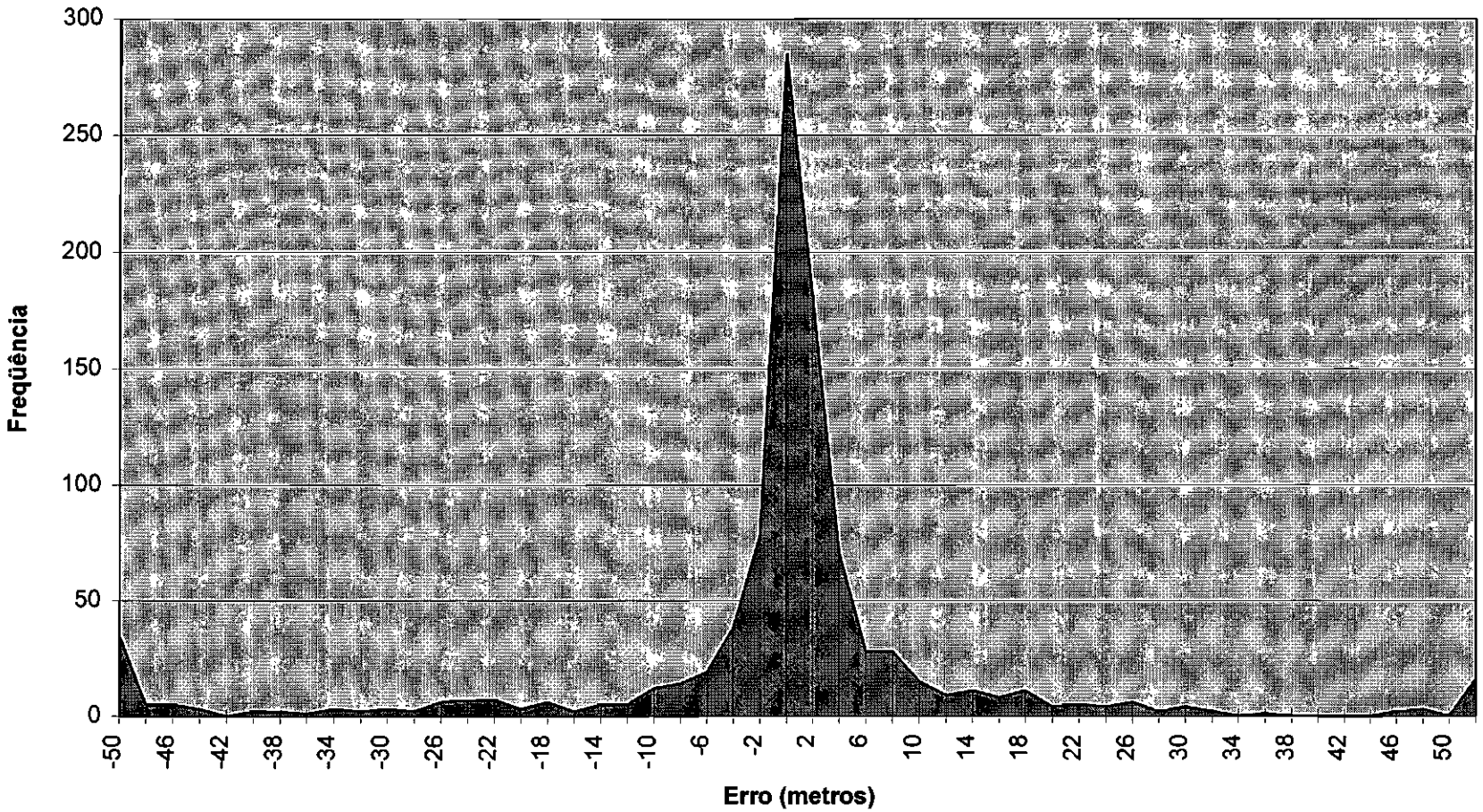


Fig. 5.6 - Polígono de Freqüência de Erros da Interpolação Polinomial de Lagrange.

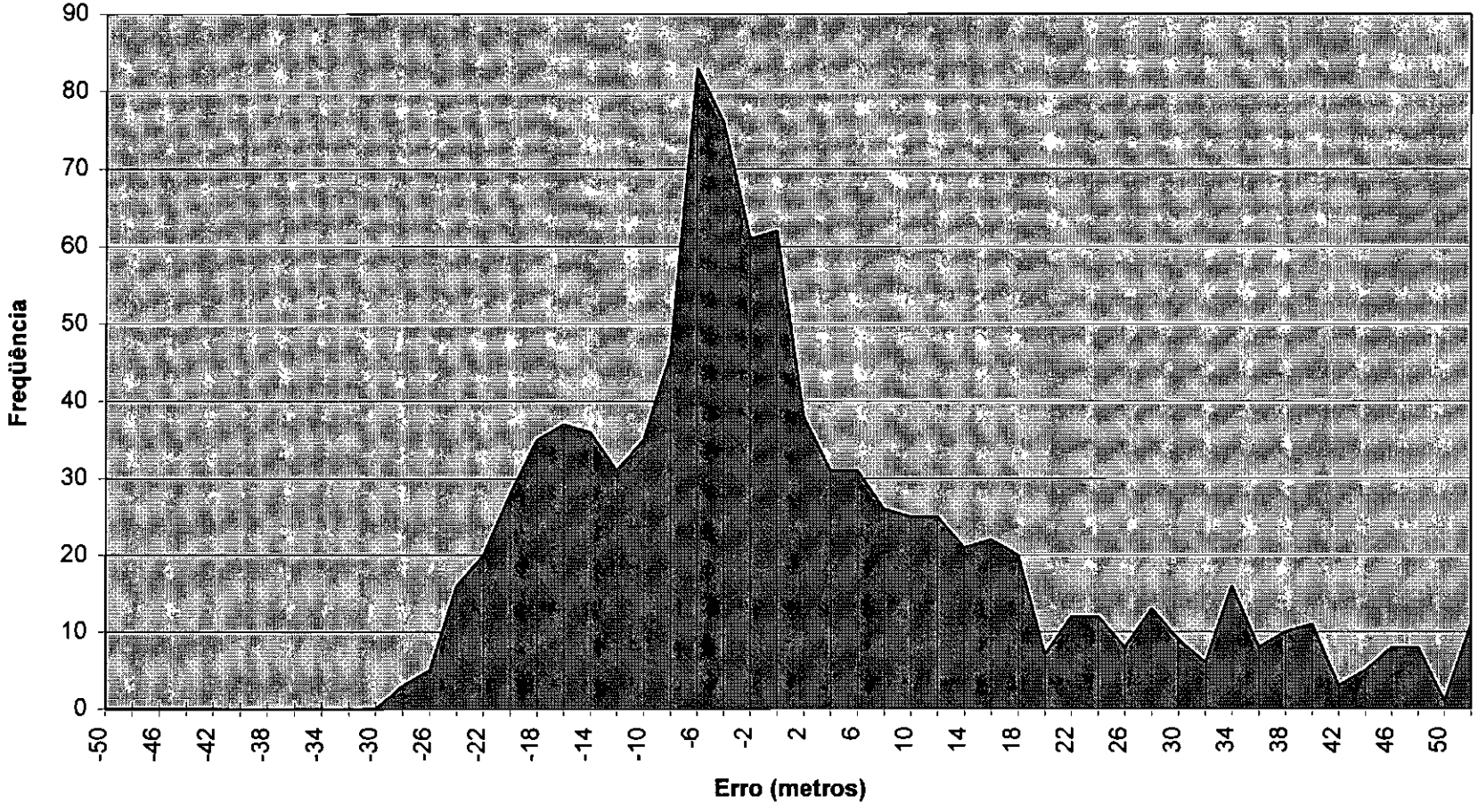


Fig. 5.7 - Polígono de Freqüência de Erros da Interpolação Bézier.

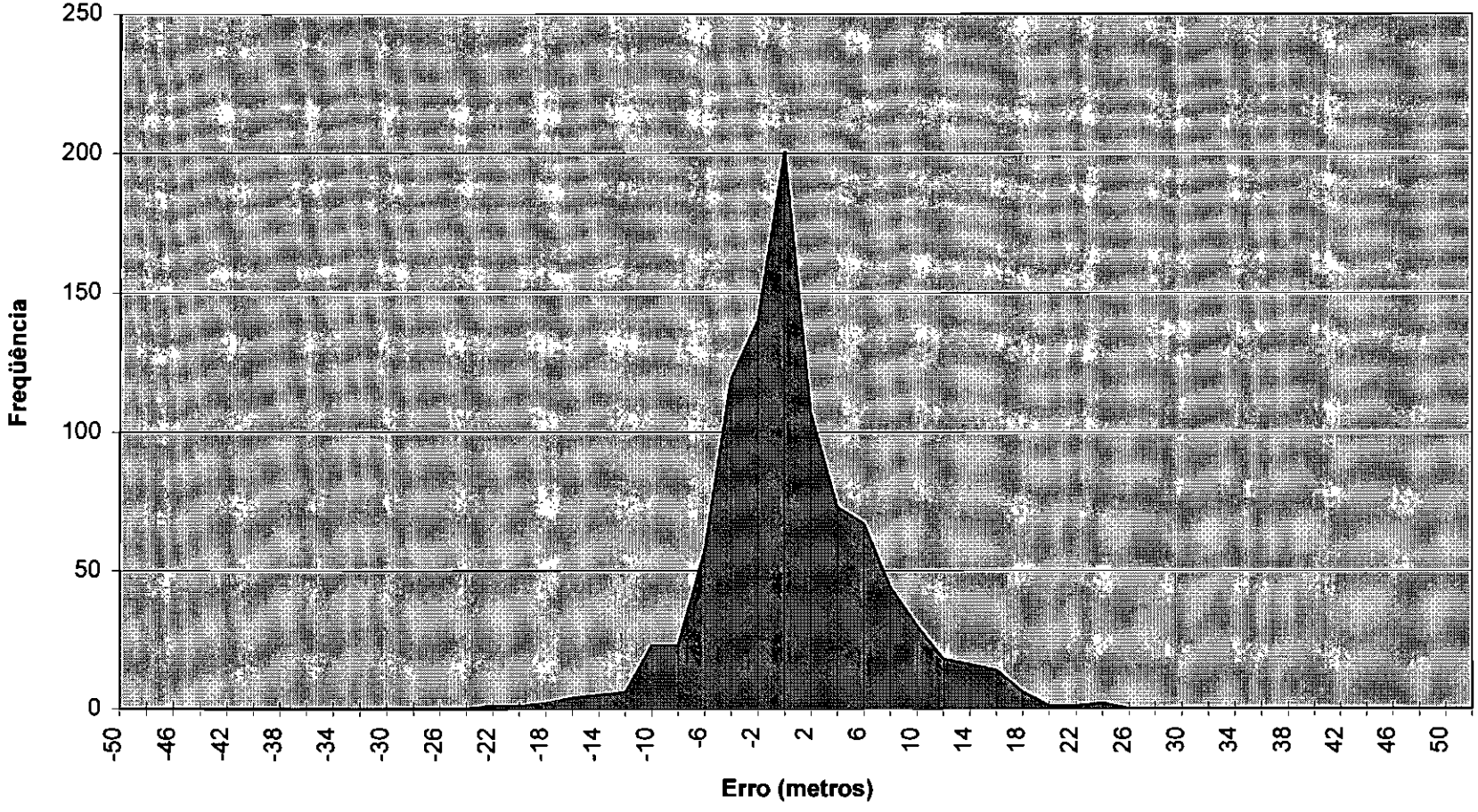


Fig. 5.8 - Polígono de Freqüência de Erros da Interpolação B-Spline.

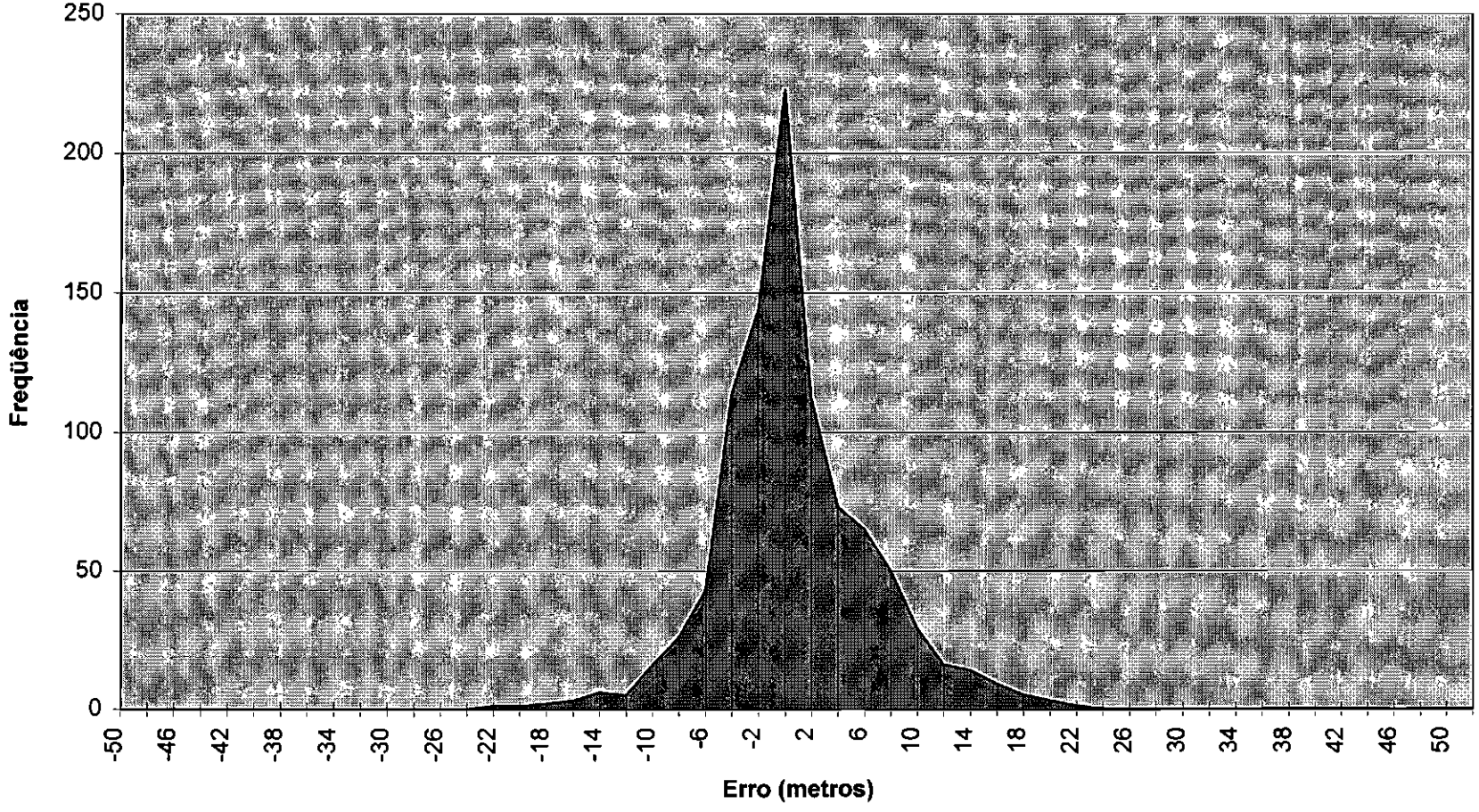


Fig. 5.9 - Polígono de Freqüência de Erros da Interpolação β-Spline.

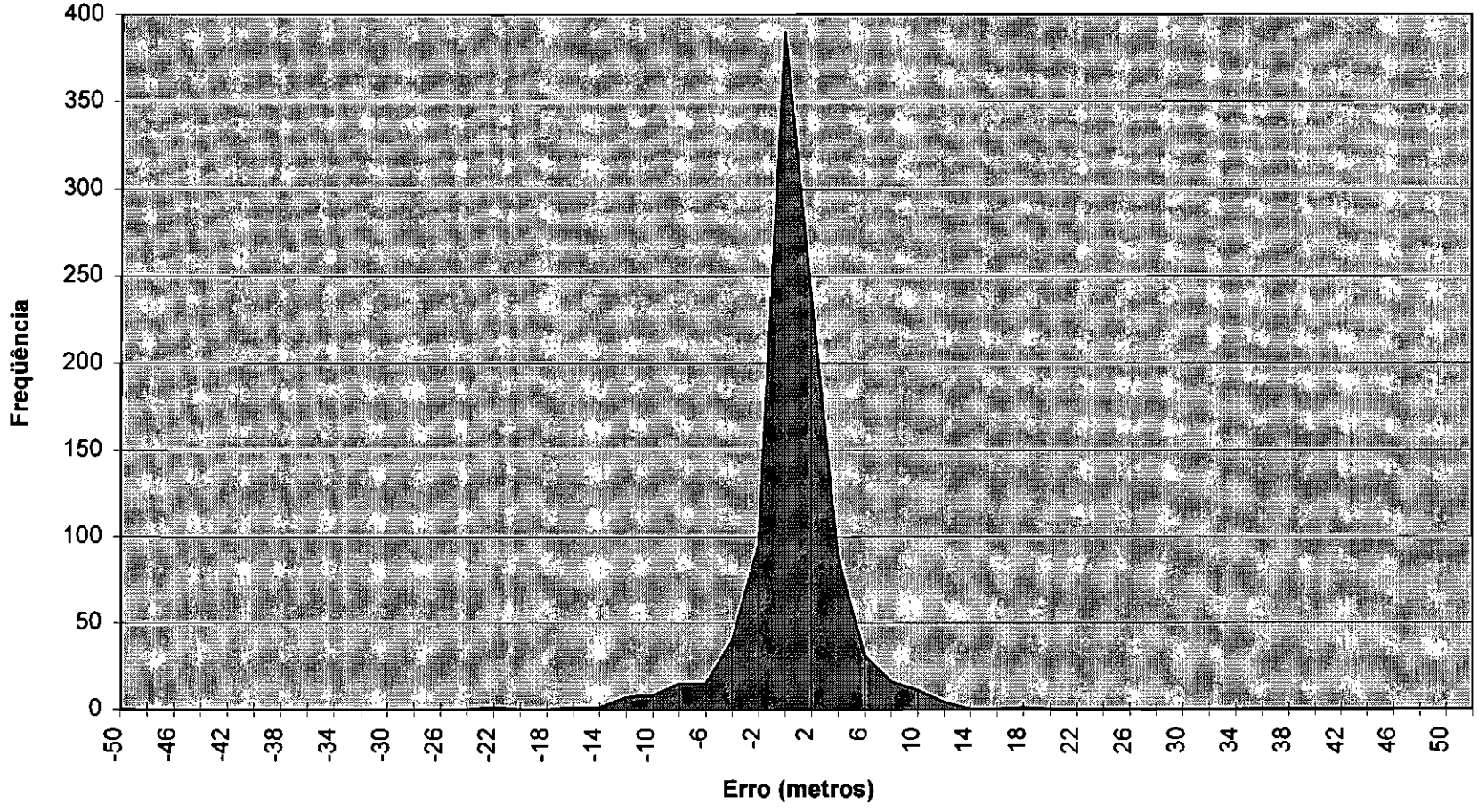


Fig. 5.10 - Polígono de Freqüência de Erros da Interpolação Akima.

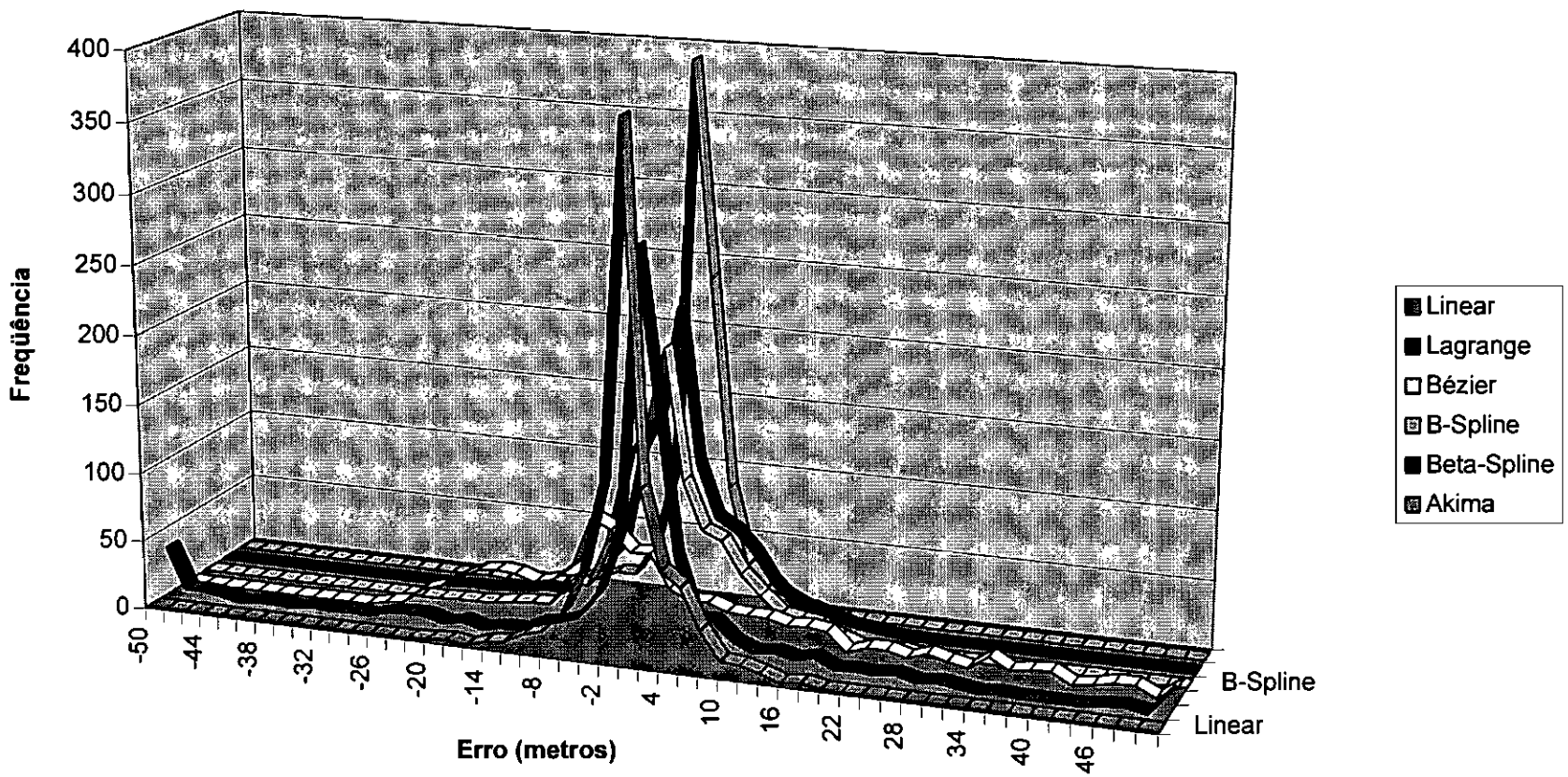


Fig. 5.11 - Polígonos de Freqüência de Erros das Interpolações.

O gráfico com a menor freqüência de valores próximos de 0 foi obtido com a Interpolação Bézier. Isto demonstra que não foi possível reconstruir a superfície original a partir da grade subamostrada com a Interpolação Bézier.

O polígono de freqüência de erros da Interpolação Polinomial de Lagrange teve uma grande freqüência de valores próximos de 0. Esta concentração foi maior que a apresentada pelos polígonos de freqüência das interpolações B-Spline e β-Spline, entretanto, com o método de Lagrange, houve uma distribuição de freqüência de erros por uma amplitude maior que a encontrada com os demais métodos de interpolação.

A Figura 5.12 apresenta os valores máximo e mínimo dos erros das interpolações e a Figura 5.13 permite visualizar o valor máximo dos erros absolutos das interpolações. A amplitude total dos erros das interpolações está representada na Figura 5.14.

Analisando estes três gráficos, pode-se notar que a Interpolação Akima apresentou um bom resultado em termos de erros máximo e mínimo, todavia, nesta comparação, os melhores resultados foram obtidos com as interpolações Linear e β-Spline. Considerando-se a amplitude total dos erros, a Interpolação Akima também apresentou um excelente resultado, somente perdendo para a Interpolação Linear que obteve a menor amplitude total.

Os maiores erros foram encontrados com as interpolações Lagrange e Bézier, sendo que a grande discrepância foi obtida com Lagrange. Nesse caso, o motivo de valores tão extremos se deve à grande oscilação

produzida pelo alto grau do polinômio calculado na Interpolação Polinomial de Lagrange.

Outras características, tais como medidas de tendência central, podem ser extraídas dos polígonos de freqüência de erros das interpolações. Algumas dessas características são a mediana, a moda e a média.

A mediana representa a média aritmética dos dois valores centrais dos erros das interpolações. Geometricamente, a mediana corresponde à vertical que divide o histograma em duas partes de áreas iguais. No gráfico da Figura 5.15, pode-se observar que, com as interpolações Akima, Linear e Lagrange, esta vertical, a que divide o polígono de freqüência em duas áreas iguais, passa exatamente pela origem. Esta mediana de valor nulo indica que não houve tendência de deslocar a superfície resultante nem para cima e nem para baixo em relação à superfície original.

Por outro lado, a mediana apresentou valores negativos com as interpolações B-Spline, β-Spline e Bézier, indicando que houve uma tendência de que a superfície resultante da interpolação ficasse abaixo da superfície original. Isto já era esperado pois, nestes três métodos, os pontos intermediários da grade subamostrada são utilizados apenas como tensores da superfície e não como amostras corretas da superfície original.

A moda de um conjunto de erros corresponde ao erro que ocorre com a maior freqüência. De acordo com o gráfico da Figura 5.16, pode-se verificar que o erro que ocorre com maior freqüência nas interpolações Linear, Lagrange, Bézier e Akima é o valor 0. Isto quer dizer que, com estes métodos, os picos dos polígonos de freqüência de

erros se encontram na origem. Observando as Tabelas 5.15 e 5.16, nota-se claramente que a maior freqüência de erros próximos do valor 0 é obtida com a Interpolação Akima.

A média aritmética, ou simplesmente média, de um conjunto de n erros é o somatório destes valores dividido por n. A Figura 5.17 apresenta a média dos erros obtidos com as interpolações Linear, Lagrange, Bézier, B-Spline, β-Spline e Akima. De acordo com este gráfico, o método de Interpolação Akima foi o que obteve a média com valor mais próximo de 0, indicando uma melhor reconstituição da superfície original.

O grau ao qual os dados numéricos tendem a dispersar-se em torno de um valor médio chama-se variação ou dispersão dos dados. Existem várias medidas de dispersão ou de variação, sendo as mais comuns o desvio médio, o desvio padrão e a variância.

O desvio médio de um conjunto de n números x_1, x_2, ..., x_n é definido por

$$D.M. = \frac{\sum_{j=1}^{n} \left| x_j - \bar{x} \right|}{n}, \qquad (5.1)$$

onde $\bar{x}$ é a média aritmética dos números e $\left| x_j - \bar{x} \right|$ é o valor absoluto do desvio de x_j em relação a $\bar{x}$. O desvio padrão deste mesmo conjunto de números é representado por s e definido por

$$s = \sqrt{\frac{\sum_{j=1}^{n} \left(x_j - \bar{x} \right)^2}{n}} \qquad (5.2)$$

e a variância é definida como o quadrado do desvio padrão e é representada por s^2.

As Figuras 5.18, 5.19 e 5.20 apresentam os desvios médios, os desvios padrões e as variâncias dos erros obtidos com os diversos métodos de interpolação, respectivamente.

Observando estes três gráficos, pode-se facilmente verificar que o conjunto dos erros obtidos com a Interpolação Akima foi o que apresentou os menores valores de dispersão de dados em termos de desvio médio, desvio padrão e variância. Estas medidas de dispersão corroboram a afirmação de que a melhor reconstituição da superfície original foi obtida com a Interpolação Akima.

Além das avaliações feitas através das técnicas de estatística, outros parâmetros devem ser verificados. Para analisar esses parâmetros, os métodos de interpolação foram implementados num sistema demonstrativo para ambiente Windows e processados num microcomputador da linha IBM PC 486 DX-2 66 MHz com 16 MBytes de memória RAM.

Uma das avaliações mais importantes se refere ao tempo gasto no processamento das interpolações. A Figura 5.21 apresenta os tempos gastos nos processamentos das interpolações Linear, Lagrange, Bézier, B-Spline, β-Spline e Akima. A Interpolação Akima demonstrou ter um processamento extremamente rápido, somente gastando 0,068 segundos a mais do que a Interpolação Linear, que foi o método que obteve o melhor resultado.

Outro parâmetro muito importante a ser analisado é a quantidade de memória temporária gasta pelo algoritmo da interpolação. A Figura 5.22 apresenta as

quantidades de memória temporária utilizadas no processamento das interpolações. Observando este gráfico, pode-se notar que a Interpolação Akima foi a que teve o pior resultado, utilizando a maior quantidade de memória temporária durante o processamento da interpolação. Esta característica foi uma das grandes desvantagens encontradas no método proposto por Akima (1974a,b).

Outra desvantagem observada na implementação do Interpolador Akima se expressa na grande quantidade de linhas de código fonte para a programação do algoritmo, conforme pode ser verificado na Figura 5.23. Esta característica não possui grande relevância pois, mesmo tendo mais linhas de código, o algoritmo da Interpolação Akima é extremamente simples e rápido, conforme a análise de custo computacional apresentada no gráfico da Figura 5.21.

Em termos de ordem de continuidade das interpolações, de acordo com a Figura 5.24, a Interpolação Akima apresentou um grau razoável de continuidade na superfície gerada. Esta ordem de continuidade do primeiro grau indica que, com o método, o resultado da interpolação se caracterizou por ser uma superfície suavemente ajustada sobre os pontos da grade subamostrada original.

A Tabela 5.17 apresenta todos os valores demonstrados nos gráficos das Figuras 5.12, 5.13, 5.14, 5.15, 5.16, 5.17, 5.18, 5.19, 5.20, 5.21, 5.22, 5.23 e 5.24.

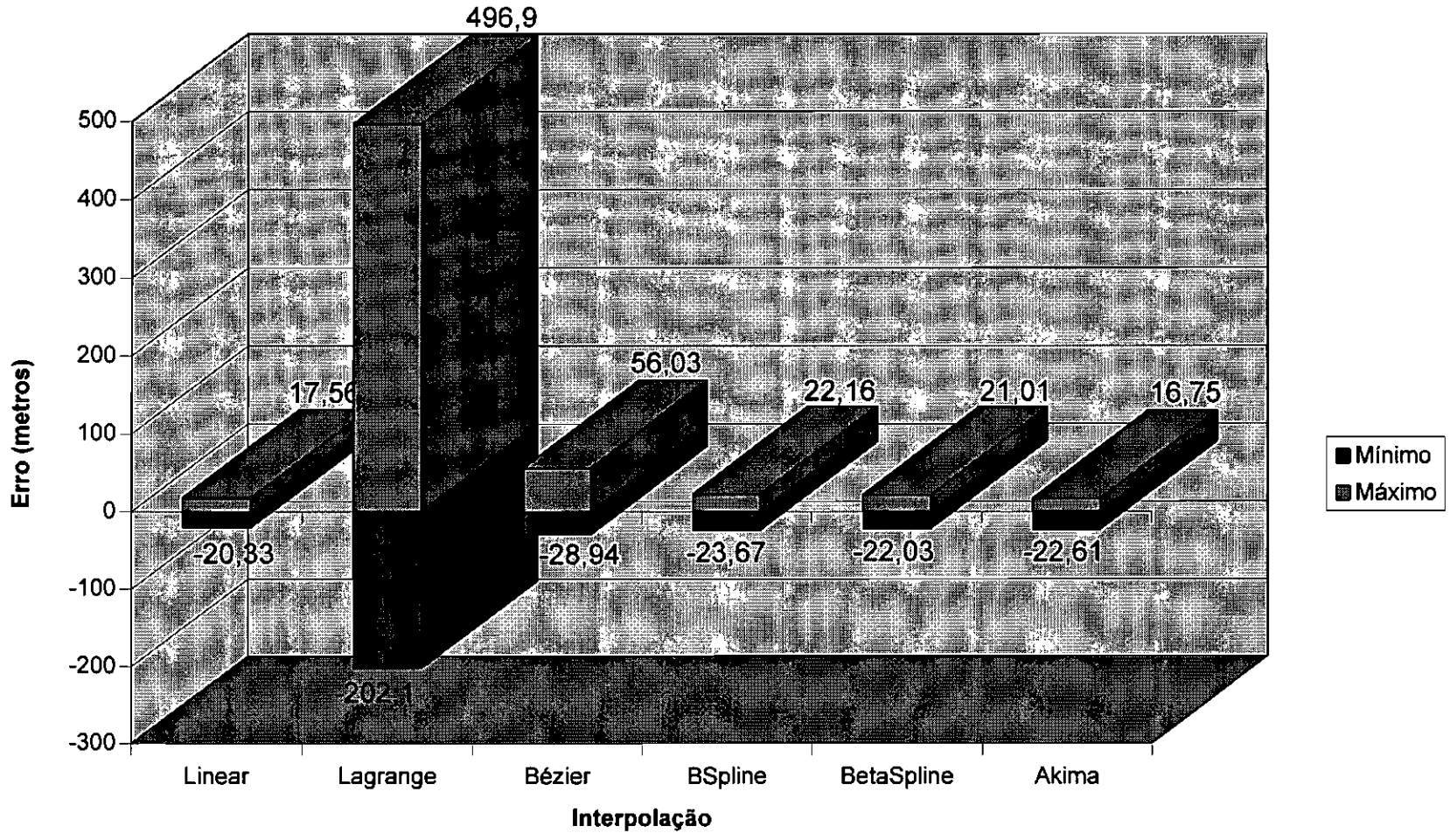


Fig. 5.12 - Valores Máximo e Mínimo dos Erros das Interpolações.

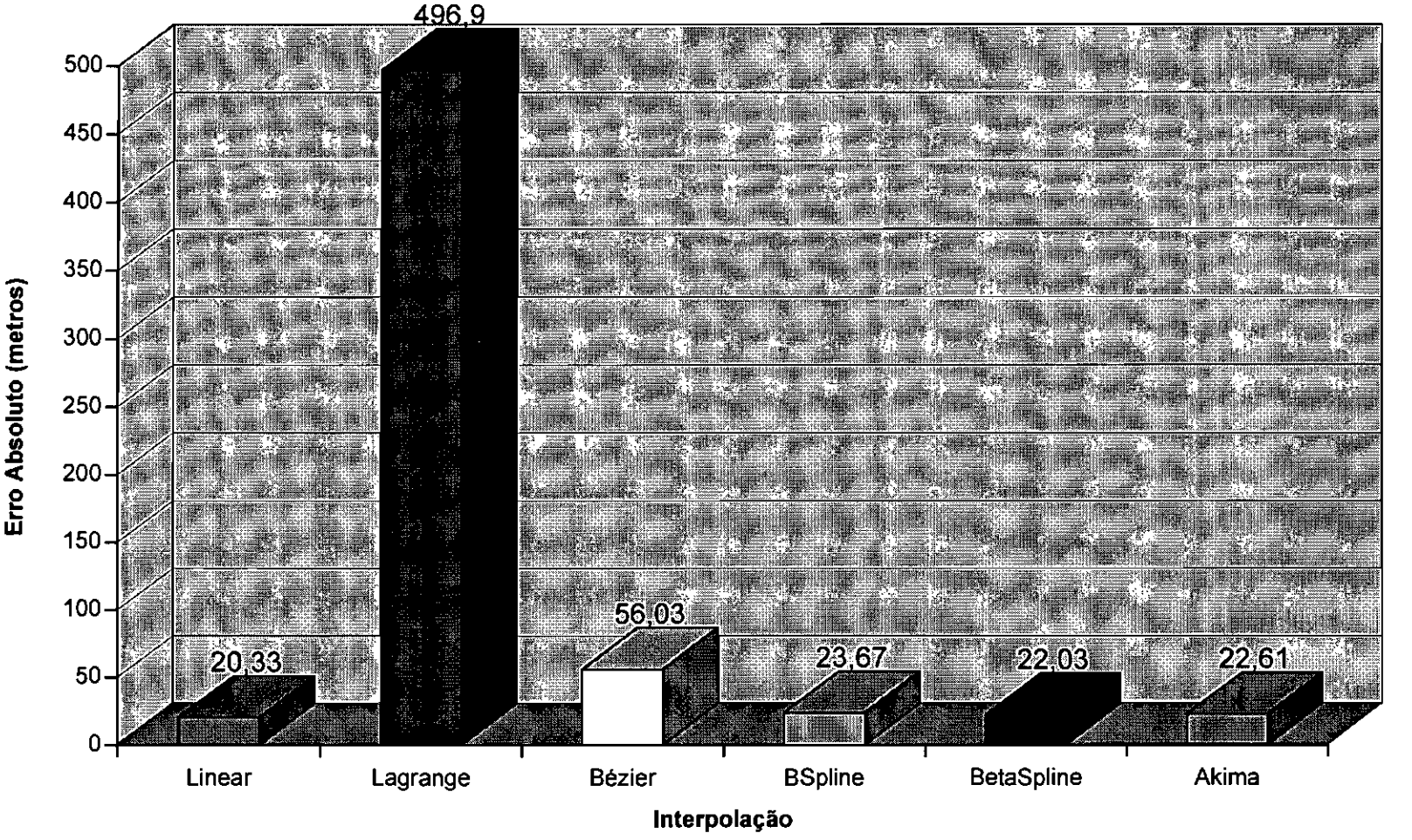


Fig. 5.13 - Valor Máximo dos Erros Absolutos das Interpolações.

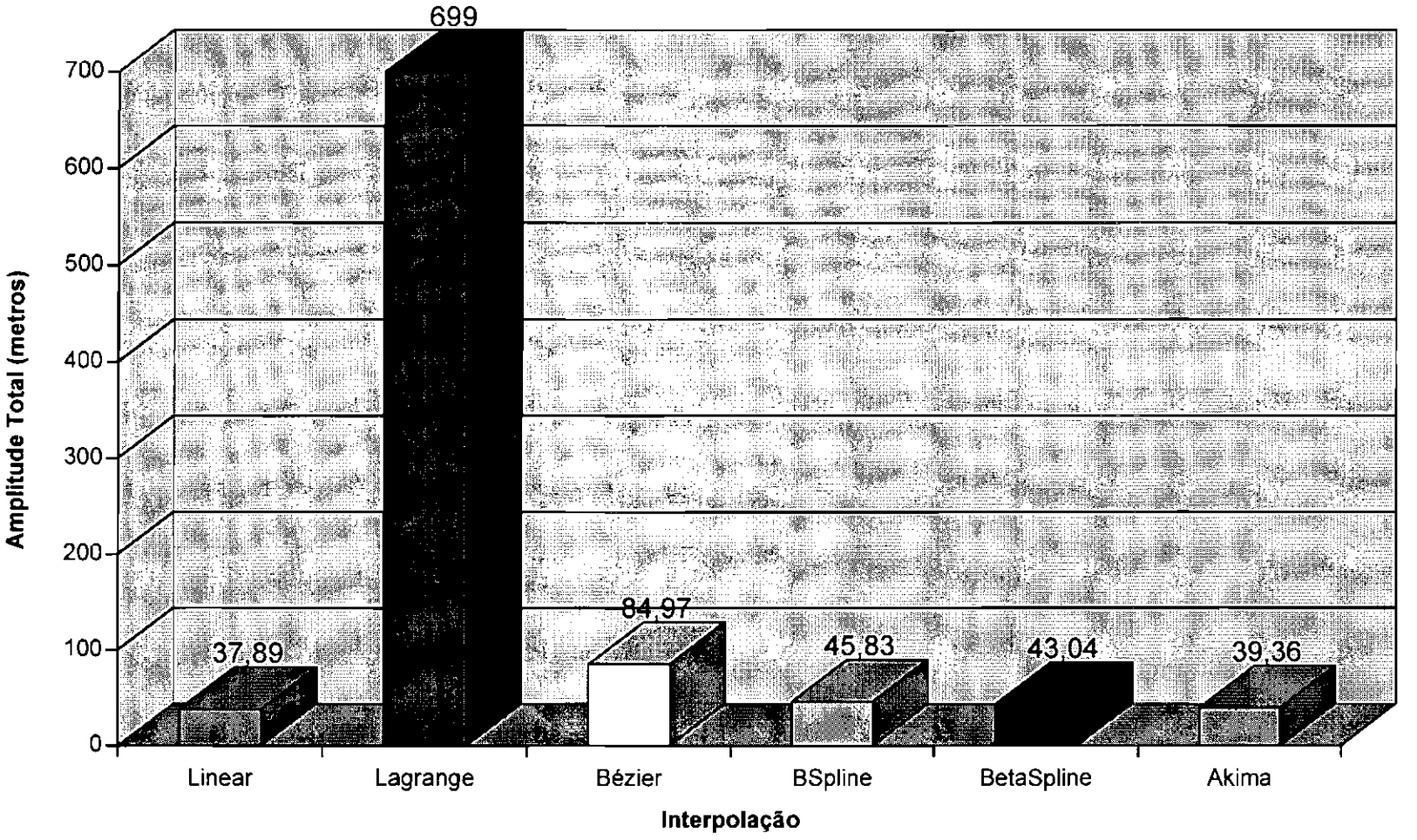


Fig. 5.14 - Amplitude Total dos Erros das Interpolações.

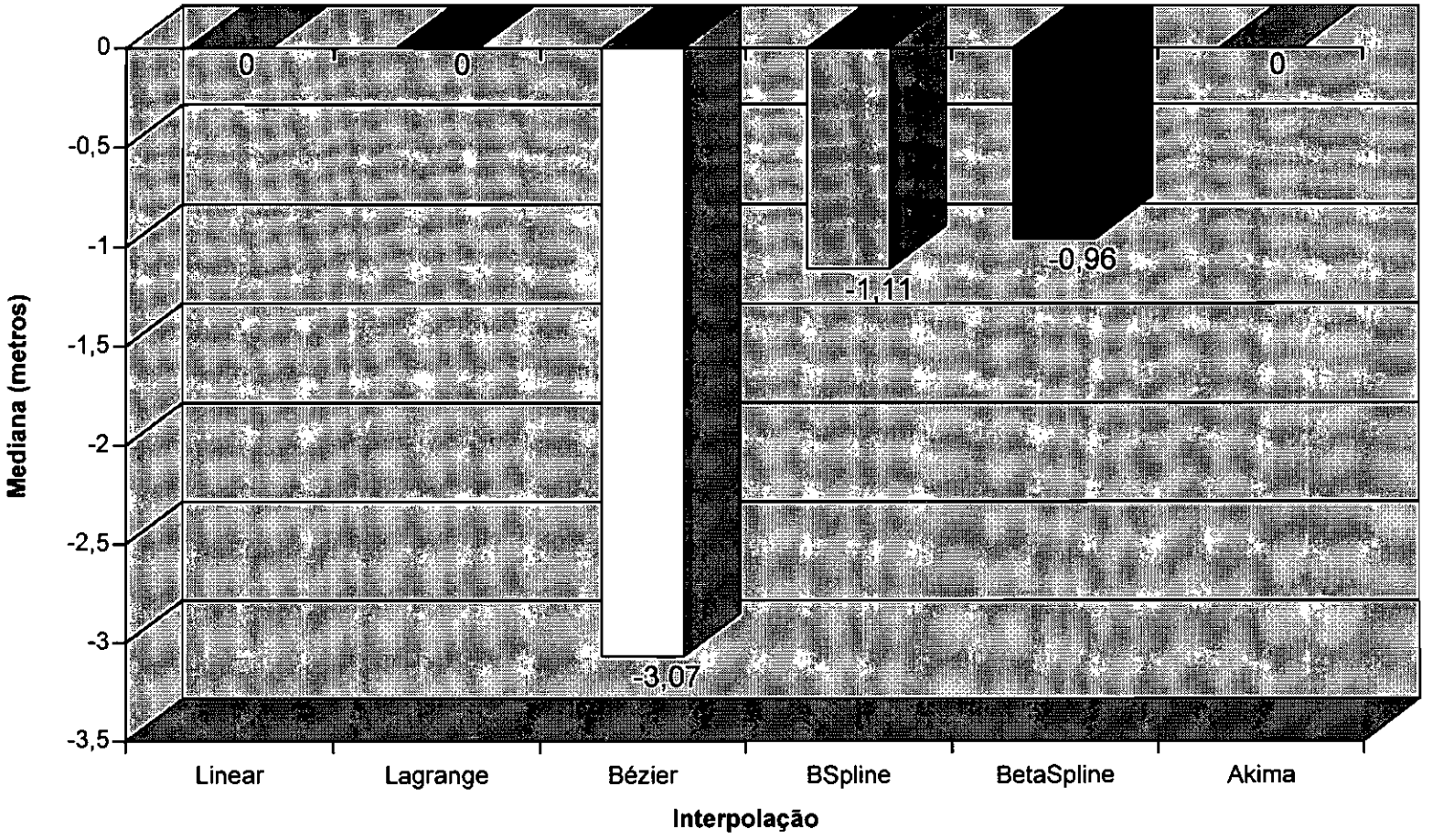


Fig. 5.15 - Mediana dos Erros das Interpolações.

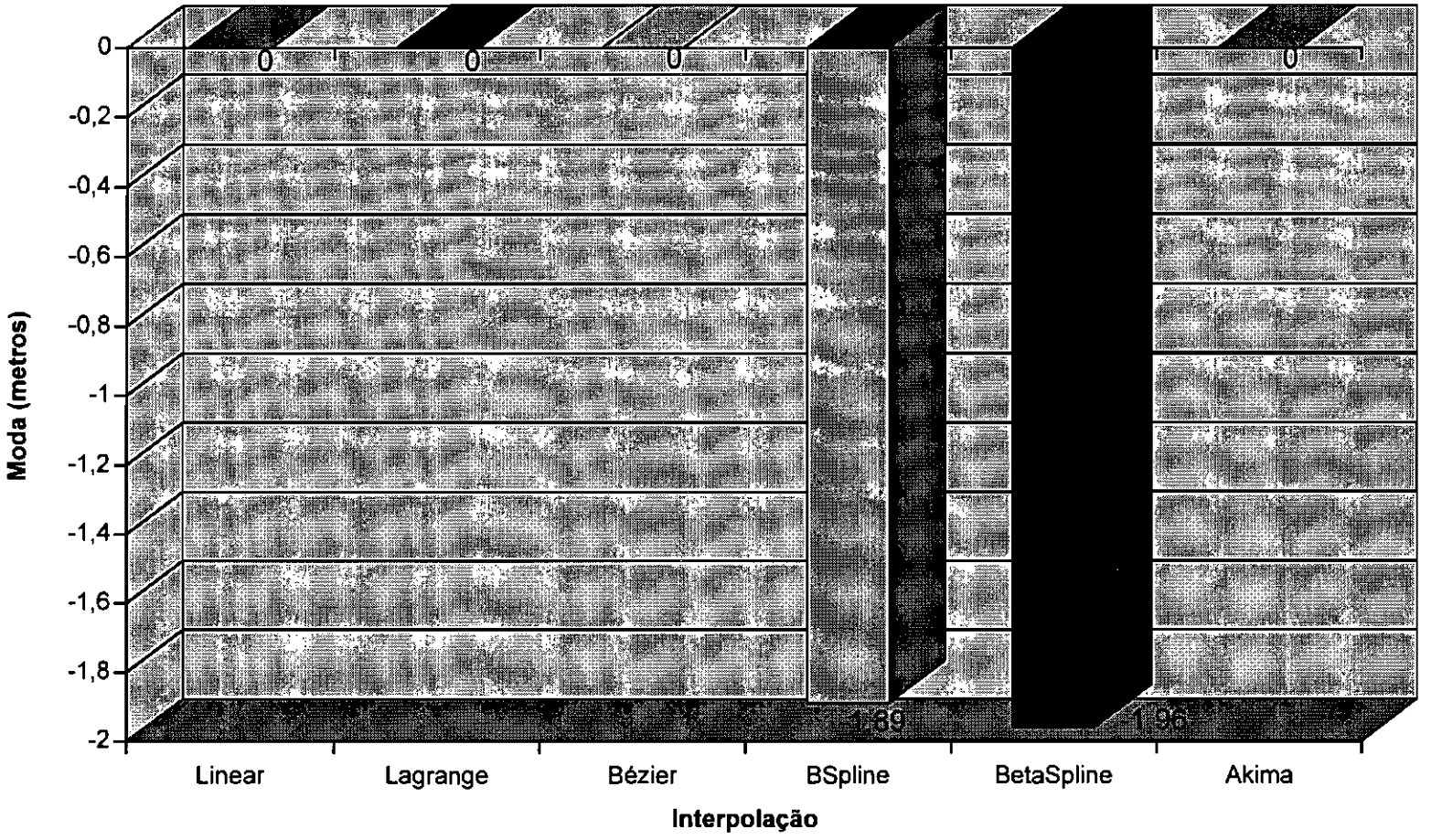


Fig. 5.16 - Moda dos Erros das Interpolações.

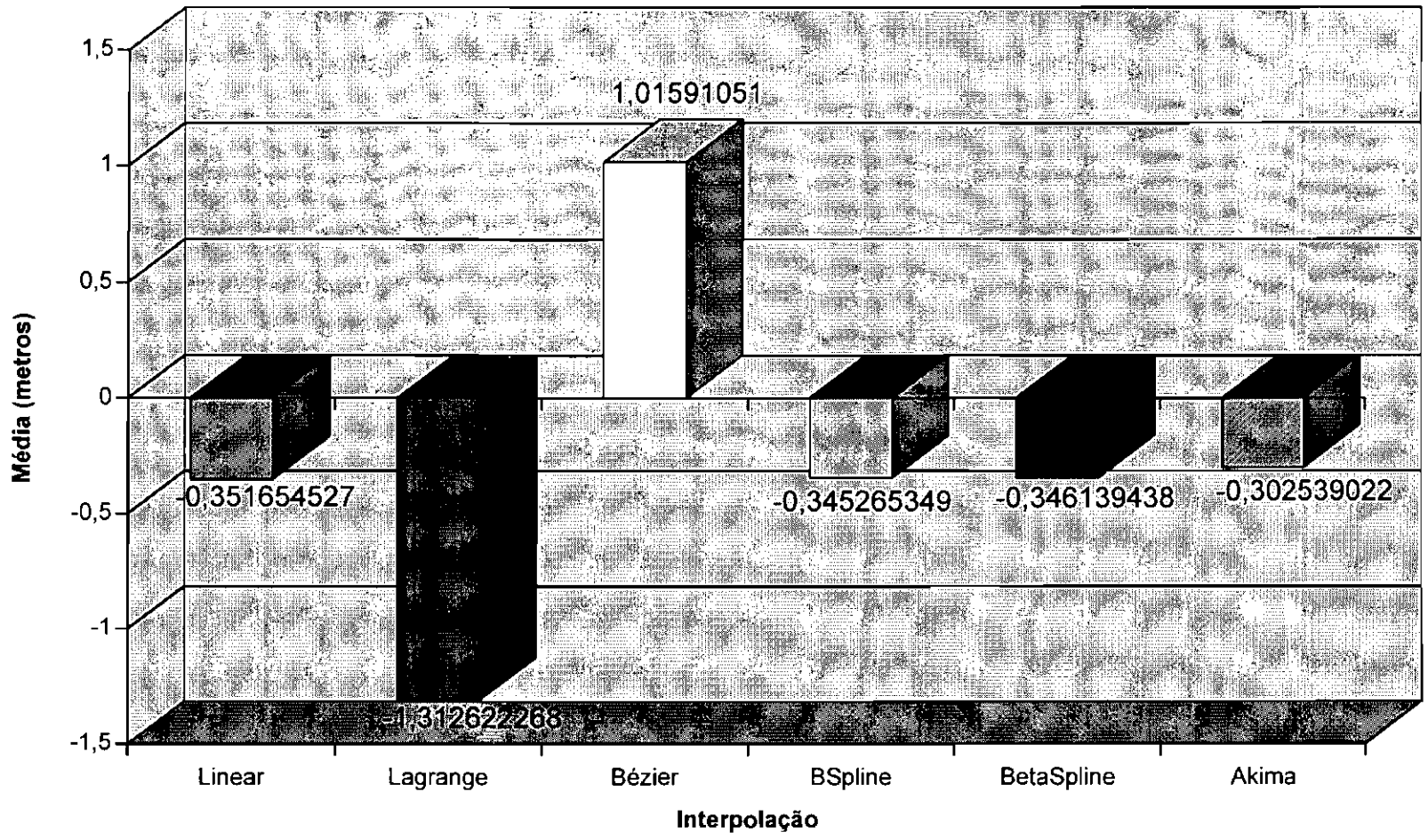


Fig. 5.17 - Média dos Erros das Interpolações.

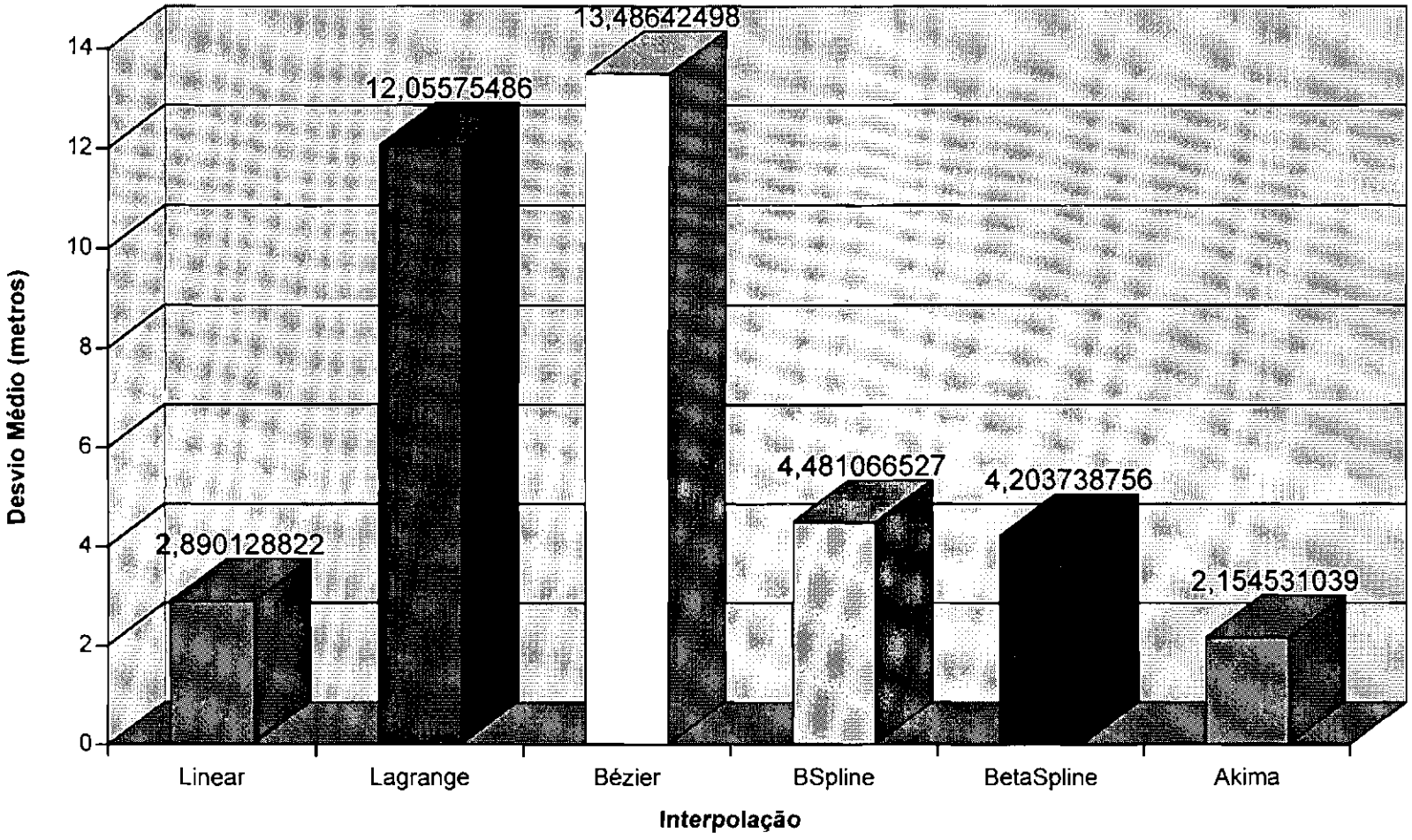


Fig. 5.18 - Desvio Médio dos Erros das Interpolações.

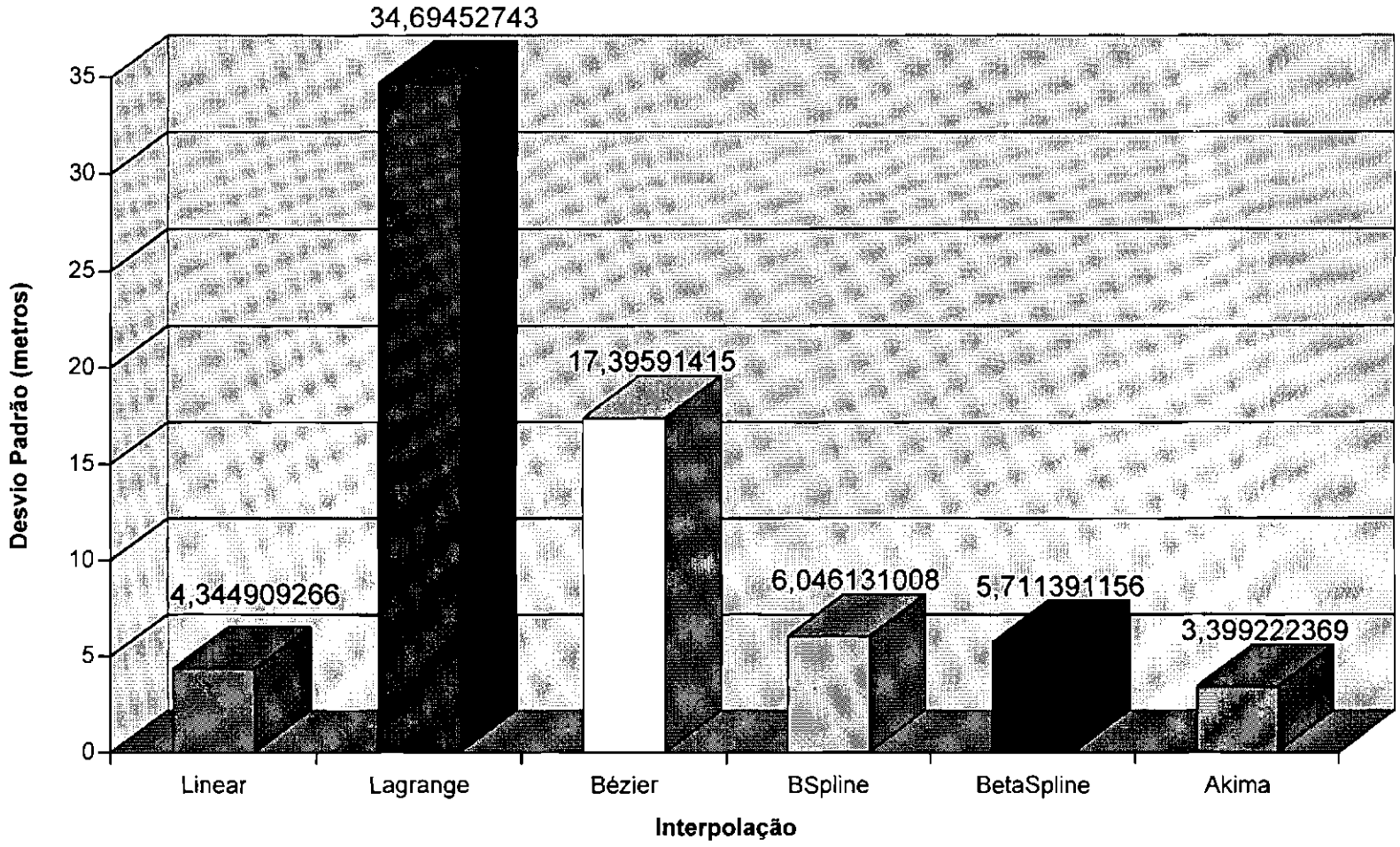


Fig. 5.19 - Desvio Padrão dos Erros das Interpolações.

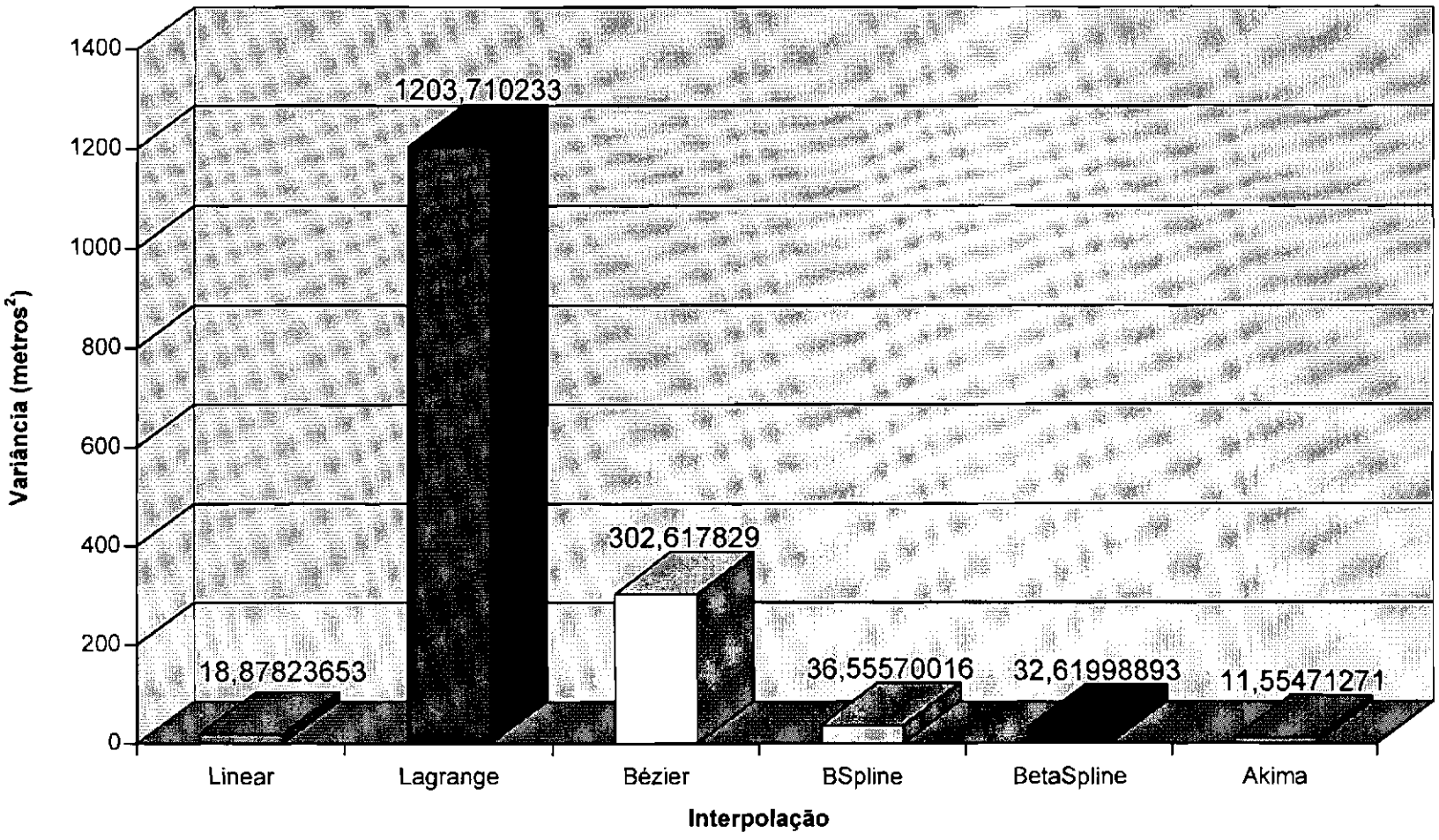


Fig. 5.20 - Variância dos Erros das Interpolações.

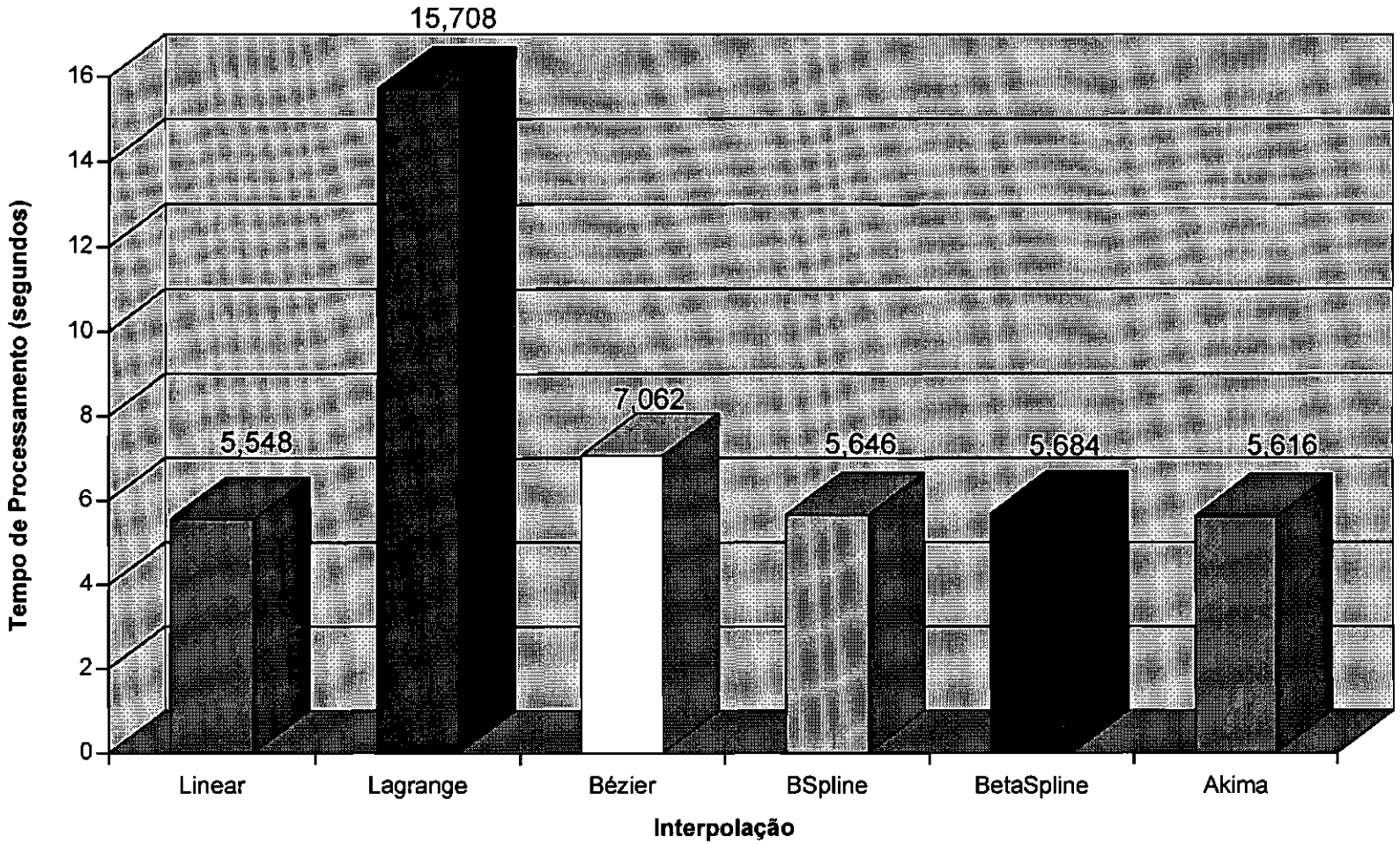


Fig. 5.21 - Tempo Gasto no Processamento das Interpolações.

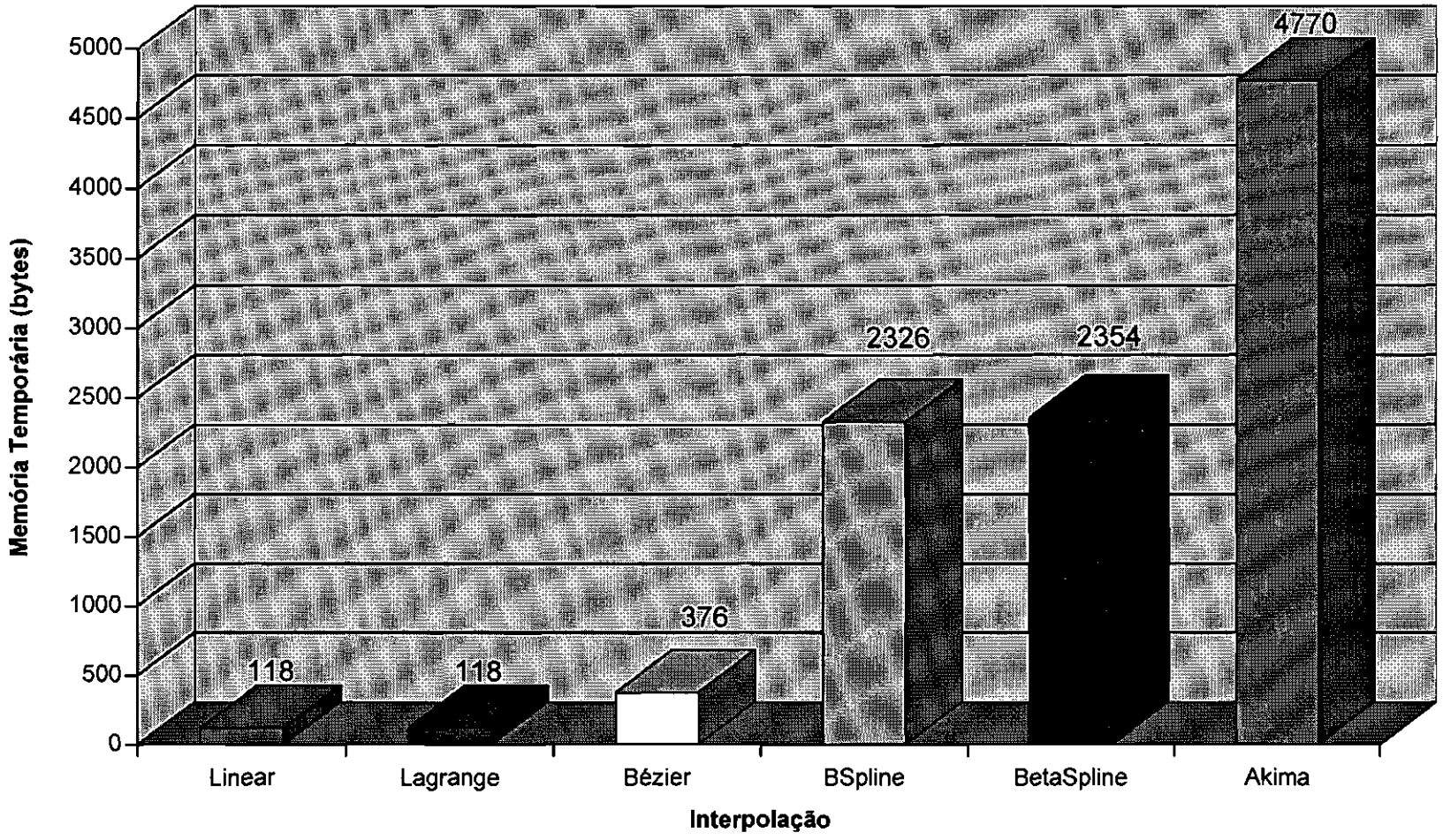


Fig. 5.22 - Memória Temporária Utilizada no Processamento das Interpolações.

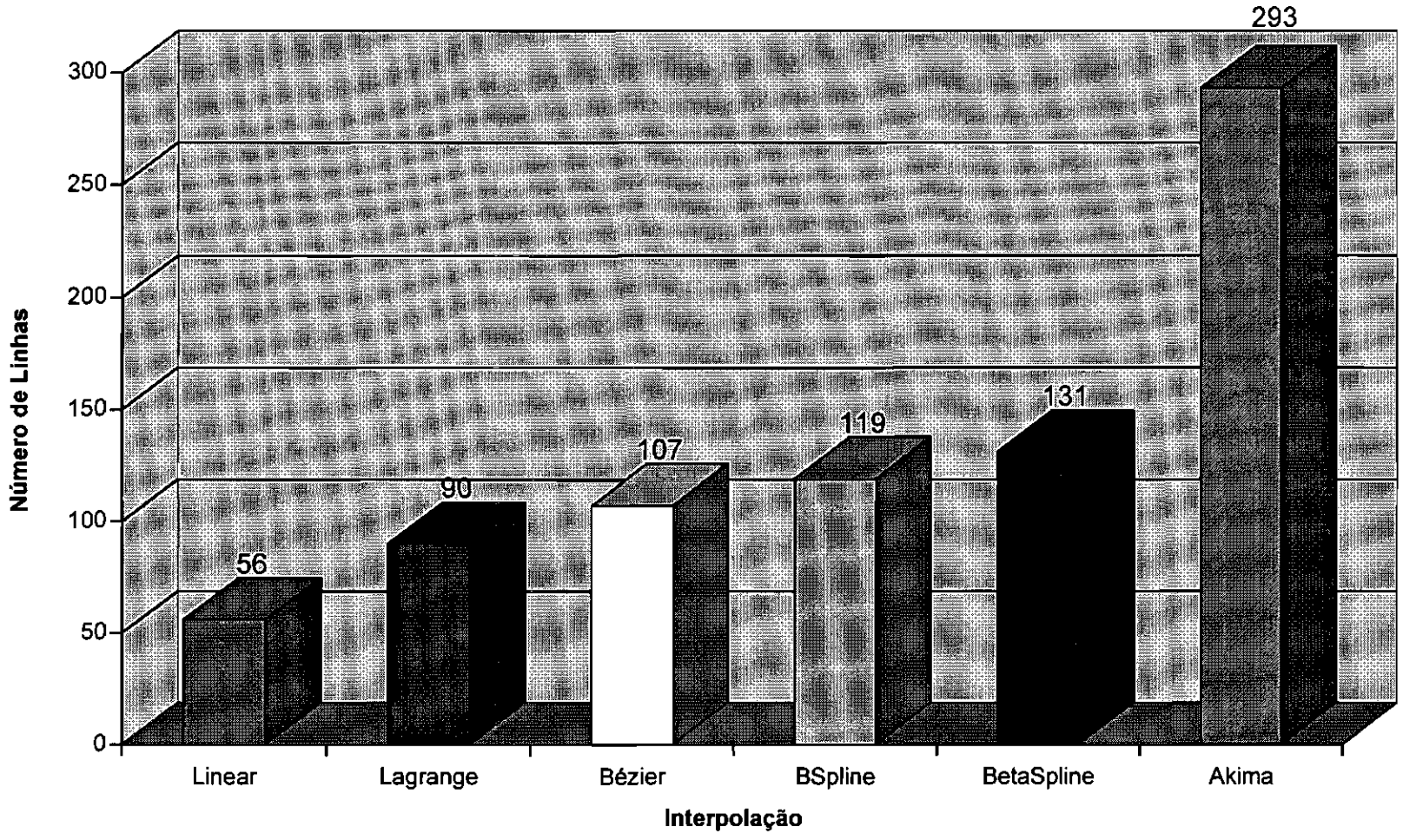


Fig. 5.23 - Número de Linhas das Implementações das Interpolações.

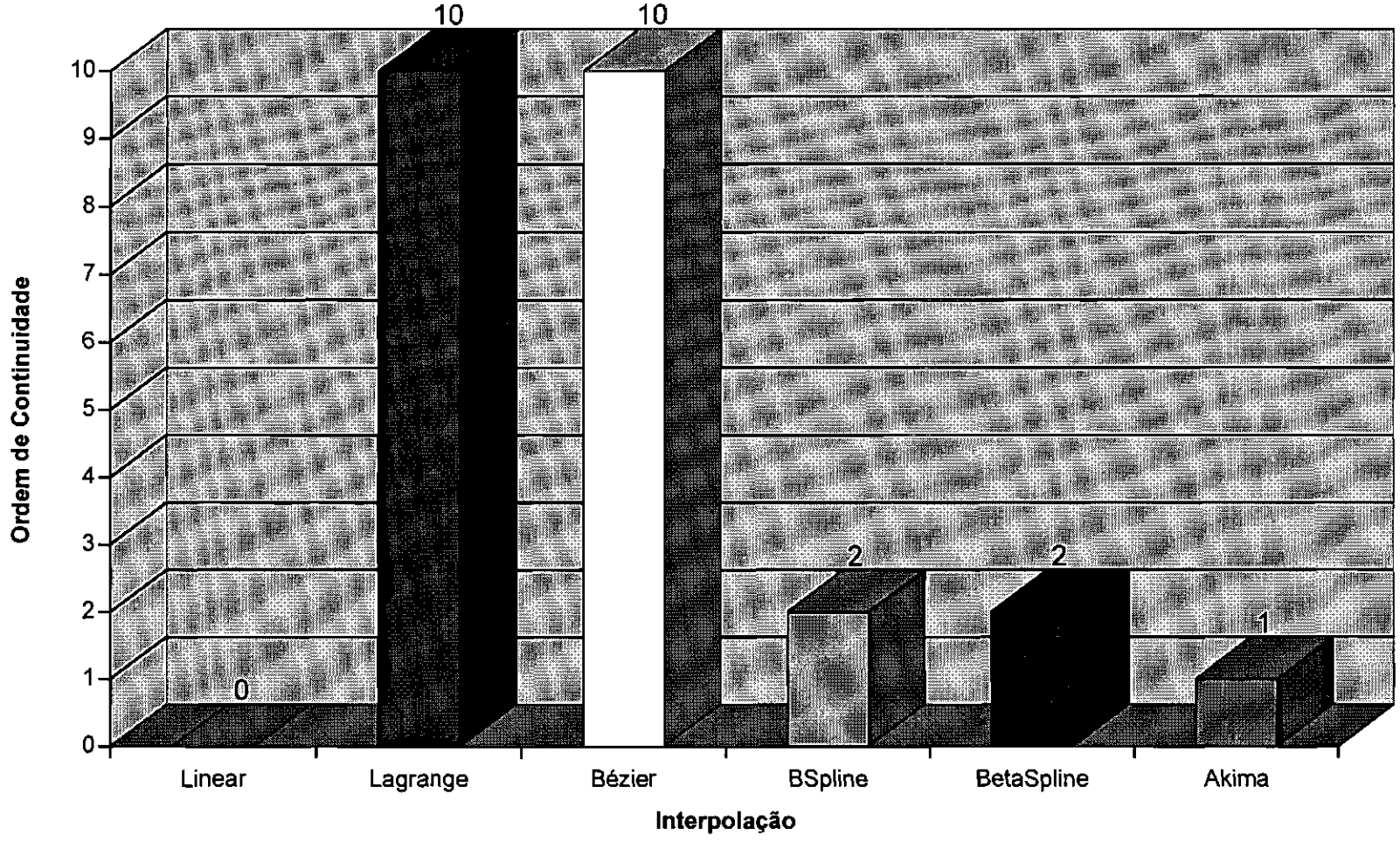


Fig. 5.24 - Ordem de Continuidade das Interpolações.

TABELA 5.17 - COMPARAÇÃO ENTRE AS INTERPOLAÇÕES

Tipo de Comparação	Linear	Lagrange	Bézier	B-Spline	β-Spline	Akima
Número de Valores da Grade Refinada	961	961	961	961	961	961
Somatório (metros)	-337,94	-1261,43	976,29	-331,8	-332,64	**-290,74**
Máximo (metros)	17,56	496,9	56,03	22,16	21,01	**16,75**
Mínimo (metros)	**-20,33**	-202,1	-28,94	-23,67	-22,03	-22,61
Máximo Absoluto (metros)	**20,33**	496,9	56,03	23,67	22,03	22,61
Amplitude Total (metros)	**37,89**	699	84,97	45,83	43,04	39,36
Mediana (metros)	**0**	**0**	-3,07	-1,11	-0,96	**0**
Moda (metros)	**0**	**0**	**0**	-1,89	-1,96	**0**
Média (metros)	-0,3516545	-1,3126223	1,0159105	-0,3452653	-0,3461394	**-0,302539**
Desvio Médio (metros)	2,8901288	12,055755	13,486425	4,4810665	4,2037388	**2,154531**
Desvio Padrão (metros)	4,3449093	34,694527	17,395914	6,046131	5,7113912	**3,3992224**
Variância (metros^2)	18,878237	1203,7102	302,61783	36,5557	32,619989	**11,554713**
Tempo Gasto no Processamento (segundos)	**5,548**	15,708	7,062	5,646	5,684	5,616
Memória Temporária Utilizada (bytes)	**118**	**118**	376	2326	2354	4770
Número de Linhas	**56**	90	107	119	131	293
Ordem de Continuidade	0	10	10	2	2	**1**
Perda de Informações	**Não**	**Não**	Sim	Sim	Sim	**Não**
Tipo de Controle	**Local**	Global	Global	**Local**	**Local**	**Local**

5.5 - RESULTADOS DA ANÁLISE

Estatisticamente, a grade regular original, considerada como superfície real, foi melhor reconstituída através da Interpolação Akima para Ajuste de Superfícies Suaves sobre a grade regular subamostrada. A grade refinada por Akima foi a que apresentou valores mais próximos da superfície original.

De acordo com os gráficos apresentados, pode-se afirmar que, com a Interpolação Akima, não houve perda de informações, pois a superfície resultante passou exatamente por todos os pontos da grade regular subamostrada, que são valores corretos amostrados da superfície original.

A superfície obtida com o método proposto por Akima (1974a,b) foi pouco oscilante, embora tenha apresentado uma suavidade de superfície proporcionada pela ordem de continuidade de grau 1.

O controle local do Interpolador Akima e a simplicidade de seus algoritmos permitiu uma interpolação extremamente rápida, somente perdendo para a Interpolação Linear, que apresenta apenas ordem de continuidade de grau zero na grade resultante.

A única desvantagem relevante encontrada no processamento do método proposto por Akima (1974a,b) se resume no gasto excessivo de memória temporária. Essa característica não inviabiliza o método pois, caso haja necessidade de implementar funções com gastos menores de memória temporária, pode-se alterar os algoritmos de modo que, ao invés de interpolar toda a grade de uma só vez, a grade original seja dividida em grades menores e, então,

interpolada. Dessa forma, embora haja um pequeno aumento no tempo gasto para o processamento da interpolação, o gasto de memória temporária será extremamente reduzido.

CAPÍTULO 6

CONCLUSÃO

A precisão de uma reconstituição de superfície a partir de amostras regulares da altimetria do terreno depende da qualidade das amostras e do interpolador utilizado no refinamento da grade.

Em termos de qualidade dos resultados, as características que mais fundamentam a utilização do método de Interpolação Akima no refinamento de grades são a precisão, a baixa oscilação, a suavidade das superfícies geradas e o ajustamento das superfícies sobre os pontos da grade.

Estatisticamente, dentre os métodos apresentados neste trabalho, a melhor reconstituição de superfície foi obtida com o Interpolador Akima. A grade refinada por Akima foi a que apresentou valores mais próximos da superfície original, produzindo uma grande freqüência de erros com valores próximos de zero.

A continuidade de primeira ordem (nos pontos intermediários) da Interpolação Akima permite gerar superfícies suaves e de baixa oscilação. Esta baixa oscilação é uma característica muito importante, pois o terreno que está sendo representado pela grade não oscila em excesso.

O ajustamento das superfícies sobre os pontos da grade permite que não se percam informações com a interpolação, pois os pontos da grade original representam amostras corretas da altimetria do terreno, desde que o processo de amostragem tenha sido feito de maneira precisa.

Em termos de custo, o fator que mais contribui para a utilização do método proposto por Akima (1974a,b) é o baixo custo computacional dos procedimentos implementados. O controle local do Interpolador Akima e a simplicidade de seus algoritmos permite uma interpolação extremamente rápida, somente perdendo para a Interpolação Linear, que apresenta apenas continuidade de superfície de ordem zero (C^0) na grade resultante.

A única desvantagem relevante encontrada no processamento da Interpolação Akima se resume no gasto excessivo de memória temporária, entretanto, esta característica não inviabiliza a utilização do método pois este gasto de memória pode ser reduzido, alterando-se os algoritmos para que a grade seja interpolada por partes.

O método de Interpolação Akima para superfícies possui também a desvantagem de ser aplicável exclusivamente em funções $y = y(x,z)$ de valores únicos. Este fator não é muito importante no caso de refinamento de grades, pois os terrenos representados possuem um valor único de altimetria para cada posição no solo.

Embora existam algumas pequenas desvantagens, a análise estatística demonstra que o método de Interpolação Akima para Ajuste de Superfícies Suaves Baseado em Procedimentos Locais (Akima, 1974a) possui excelente aplicação na modelagem numérica de superfícies através do refinamento de grades regulares com amostras reais do terreno. Os resultados obtidos com este método são mais próximos da superfície original, permitindo uma modelagem do terreno melhor do que a obtida por outros métodos de interpolação de superfícies, para um mesmo número de amostras.

REFERÊNCIAS BIBLIOGRÁFICAS

Akima, H. A new method of interpolation and smooth curve fitting based on local procedures. **Communications of ACM, 17**(4):589-602, Oct. 1970.

Akima, H. Algorithm 433, Interpolation and smooth curve fitting based on local procedures. **Communications of ACM, 15**(10):914-918, Oct. 1972.

Akima, H. A method of bivariate interpolation and smooth surface fitting based on local prodedures. **Communications of ACM, 17**(1):18-20, Jan. 1974a.

Akima, H. Algorithm 474. Bivariate interpolation and smooth surface fitting based on local procedure. **Communications of ACM, 17**(1):26-31, Jan. 1974b.

Foley, J.D. et al. **Computer graphics: principles and practice.** 2.ed. New York, Addison-Wesley, Nov. 1992, 1160 p.

Hildebrand, F.B. **Introduction to numerical analysis.** New York, McGraw-Hill, 1956, 485 p.

APÊNDICE A

BIBLIOGRAFIA COMPLEMENTAR

Ahuja, D.V. An algorithm for generating spline like curves. **IBM Systems Journal, 17**(3):206-217, 1968.

Beatty, J.C.; Barsky, B.A. **Introduction to the Use of Splines in Freeform Curve and Surface Design.** SIGGRAPH 86 COURSE, Ontário, University of Waterloo, 1986. 330 p.

Bézier, P.E. **Emploi des machines à commande numérique.** Paris, Masson et Cie., 1970 [traduzido por A.Robin Forrest como **Numerical Control - Mathematics and Aplications**, John Wiley & Sons, London, 1972].

Bézier, P.E. **Mathematical and Practical Possibilities of UNISURF, Computer Aided Geometric Design.** New York, Academic Press, New York, 1974.

Gomes, W. **Amostragem e interpolções em modelos digitais de terreno.** (Dissertação de Mestrado em Sensoriamento Remoto) - Instituto Nacional de Pesquisas Espaciais, São José dos Campos, Dez. de 1989. 96 p. (INPE-5041-TDL/404).

Greville, T.N.E. **Spline functions, interpolation, and numerical quadrature.** In: Ralston, A.; Wilf, H.S. ed. **Mathematical Methods for Digital Computers.** New York, Wiley, 1987. v. 2, part. 6, p. 156-168.

Instituto Nacional de Pesquisas Espaciais (INPE). **Manual de normas para publicações técnico-científicas.** São José dos Campos, INPE, 1990. 126 p. (INPE-5116-MAN/001).

Mortenson, M.E. **Geometric modeling.** New york, Wiley, 1985. 312 p.

Norton, P.; Yao, P.L. **Borland C++ programing for windows.** New York, Bantam Computer Books, 1992, 746 p.

Spiegel, M. R. **Estatística.** São Paulo, McGraw-Hill, 1971. 580 p. (Coleção Schaum)

Steinbruch, A. **Álgebra linear e geometria analítica.** São Paulo, McGraw-Hill, 1972. 518 p.

APÊNDICE B

LISTAGEM DA INTERPOLAÇÃO LINEAR PARAMÉTRICA PARA CURVAS

```
void TMainWindow::Linear (void)
{
   // Interpolação Linear Paramétrica para Curvas
   DesignOriented  Dsg (DisplayWindow->HWindow, wScrollOffsetX,
                        wScrollOffsetY, BitmapHandle,
                        wImageWidthInPixels, wImageHeightInPixels);
   PointSet        P;
   float           x, y, u, delta;
   int             x1, y1, x2, y2, xi, yi, xf, yf;

   SetCursor (LoadCursor (NULL, IDC_WAIT));
   Dsg.SetSolidPen(100,255,0);

   // acerta valores lidos de acordo com o eixo de coordenadas
   Points.Pointer = Points.First();
   while (Points.Pointer != Points.End())
   {
      P.Insert ( Points.ReadX(Points.Pointer)-50,
                 350-Points.ReadY(Points.Pointer));
      Points.Pointer = Points.Next(Points.Pointer);
   }

   // Equação:
   //
   //    x(u)  = x1*(1-u) + x2*(u)
   //    y(u)  = y1*(1-u) + y2*(u)
   //
   // onde:
   //
   //    u       = 0..1
   //    (x1,y1) = primeiro ponto do intervalo
   //    (x2,y2) = último ponto do intervalo

   if (P.SizeOf() >= 2)
   {
      x1 = P.ReadX(P.First());
      y1 = P.ReadY(P.First());
      P.Pointer = P.Next(P.First());
      while (P.Pointer != P.End())
      {
         x2 = P.ReadX(P.Pointer);
         y2 = P.ReadY(P.Pointer);

         // calcula variação paramétrica
         if ((abs(x2-x1))>(abs(y2-y1)))
            delta = (abs(x2-x1));
         else
            delta = (abs(y2-y1));
```

```
        // se p1 != p2
        if (delta!=0)
        {
           // varia a função paramétrica para cada
           // segmento de curva
           delta = 1/delta;
           xi = x1;
           yi = y1;
           for (u=0; u<1; u+=delta)
           {
              // determina as coordenadas
              x = x1*(1-u) + x2*(u);
              y = y1*(1-u) + y2*(u);
              xf = (int)x;
              yf = (int)y;

              // liga os pontos para desenhar a curva resultante
              Dsg.WindowLine(xi+50, yi+50, xf+50, yf+50, R2_COPYPEN);
              Dsg.BitmapLine(xi+50, yi+50, xf+50, yf+50, R2_COPYPEN);
              xi = xf;
              yi = yf;
           }
        }
        x1 = x2;
        y1 = y2;
        P.Pointer = P.Next(P.Pointer);
     }
  }
  Dsg.SetSolidPen(0,0,0);
  SetCursor (LoadCursor (NULL, IDC_ARROW));
}
```

APÊNDICE C

LISTAGEM DA INTERPOLAÇÃO LINEAR EXPLÍCITA PARA CURVAS

```
void TMainWindow::Linear (void)
{
   // Interpolação Linear para Curvas
   DesignOriented  Dsg (DisplayWindow->HWindow, wScrollOffsetX,
                        wScrollOffsetY, BitmapHandle,
                        wImageWidthInPixels, wImageHeightInPixels);
   PointSet        P;
   float           x, y;
   int             x1, y1, x2, y2, xi, yi, xf, yf, delta, i;

   SetCursor (LoadCursor (NULL, IDC_WAIT));
   Dsg.SetSolidPen(100,255,0);

   // acerta valores lidos de acordo com o eixo de coordenadas
   Points.Pointer = Points.First();
   while (Points.Pointer != Points.End())
   {
      P.Insert ( Points.ReadX(Points.Pointer)-50,
                 350-Points.ReadY(Points.Pointer));
      Points.Pointer = Points.Next(Points.Pointer);
   }

   // Equação:
   //
   //            y2-y1
   //    y(x) = -------.(x-x1) + y1
   //            x2-x1
   //
   // onde:
   //
   //    (x1,y1) = primeiro ponto do intervalo
   //    (x2,y2) = último ponto do intervalo

   if (P.SizeOf() >= 2)
   {
      x1 = P.ReadX(P.First());
      y1 = P.ReadY(P.First());
      P.Pointer = P.Next(P.First());
      while (P.Pointer != P.End())
      {
         x2 = P.ReadX(P.Pointer);
         y2 = P.ReadY(P.Pointer);

         // calcula variação
         if ((abs(x2-x1))>(abs(y2-y1)))
         {
            // maior variação em x
            delta = (abs(x2-x1));
            xi = x1;
```

```
            yi = y1;

            // se p1 != p2
            for(i=1; i<(delta+1); i++)
            {
               // determina as coordenadas
               if (x2>=x1)
                  x = x1 + i;
               else
                  x = x1 - i;
               y = ((y2-y1)*(x-x1))/(x2-x1) + y1;
               xf = (int)x;
               yf = (int)y;

               // liga os pontos para desenhar a curva resultante
               Dsg.WindowLine(xi+50, yi+50, xf+50, yf+50, R2_COPYPEN);
               Dsg.BitmapLine(xi+50, yi+50, xf+50, yf+50, R2_COPYPEN);
               xi = xf;
               yi = yf;
            }
         }
         else
         {
            // maior variação em y
            delta = (abs(y2-y1));
            xi = x1;
            yi = y1;

            // se p1 != p2
            for(i=1; i<(delta+1); i++)
            {
               // determina as coordenadas
               if (y2>=y1)
                  y = y1 + i;
               else
                  y = y1 - i;
               x = ((x2-x1)*(y-y1))/(y2-y1) + x1;
               xf = (int)x;
               yf = (int)y;

               // liga os pontos para desenhar a curva resultante
               Dsg.WindowLine(xi+50, yi+50, xf+50, yf+50, R2_COPYPEN);
               Dsg.BitmapLine(xi+50, yi+50, xf+50, yf+50, R2_COPYPEN);
               xi = xf;
               yi = yf;
            }
         }
         x1 = x2;
         y1 = y2;
         P.Pointer = P.Next(P.Pointer);
      }
   }
   Dsg.SetSolidPen(0,0,0);
   SetCursor (LoadCursor (NULL, IDC_ARROW));
}
```

APÊNDICE D

LISTAGEM DA INTERPOLAÇÃO LINEAR EXPLÍCITA PARA SUPERFÍCIES

```
void TMainWindow::Linear3D (void)
{
   // Interpolação Linear para Superfícies

   POINT3D            p00, p01, p10, p11;
   float              X, Y, Z, DX, DZ, dx, dz, deltaX, deltaZ, a, b, c, d;
   int                i, j, k ,l, Xmax, Xmin, Zmax, Zmin, calc;

   SetCursor (LoadCursor (NULL, IDC_WAIT));

   // Equação:
   //
   // para cada retalho de área:
   //
   //    ponto00 ----------------------------ponto10
   //       |                                   |
   //       |                                   |
   //       |                                   |
   //       |                                   |
   //    ponto01 ----------------------------ponto11
   //
   //
   //    Y = a + b*DX + c*DZ + d*DX*DZ;
   //
   // onde:
   //
   //    DX = X - p00.x;
   //    DZ = Z - p00.z;
   //    a  = p00.y;
   //    b  = (p10.y - a)/dx;
   //    c  = (p01.y - a)/dz;
   //    d  = (p11.y - a - b*dx - c*dz)/(dx*dz);
   //    dx = p11.x - p00.x;
   //    dz = p11.z - p00.z;

   // calcula variação da nova grade
   Xmax = aGrid[(iGX-1)*iGZ].x;
   Xmin = aGrid[0].x;
   Zmax = aGrid[iGZ-1].z;
   Zmin = aGrid[0].z;

   deltaX = (((float)Xmax-Xmin)/(iPX-1));
   deltaZ = (((float)Zmax-Zmin)/(iPZ-1));

   i = 0;
   for (k=0; k<iPX; k++)
   {
      X = Xmin + k*deltaX;
      j = 0;
```

```
      calc = 1; // indica necessidade de cálculo

      // caso ultrapasse o retalho em x
      while(X>(float)(aGrid[(i+1)*iGZ+j].x))
      {
         i++;
         calc = 1;
      }

      for (l=0; l<iPZ; l++)
      {
         Z = Zmin + l*deltaZ;

         // caso ultrapasse o retalho em z
         while(Z>(float)(aGrid[i*iGZ+(j+1)].z))
         {
            j++;
            calc = 1;
         }

         // calcula os valores para o retalho
         if (calc == 1)
         {
            // calcula os dados do retalho
            // pontos
            p00 = aGrid[i*iGZ+j];
            p01 = aGrid[i*iGZ+(j+1)];
            p10 = aGrid[(i+1)*iGZ+j];
            p11 = aGrid[(i+1)*iGZ+(j+1)];
            // calcula os valores dos coeficientes
            dx = p11.x - p00.x;
            dz = p11.z - p00.z;
            a  = p00.y;
            b  = (p10.y - a)/dx;
            c  = (p01.y - a)/dz;
            d  = (p11.y - a - b*dx - c*dz)/(dx*dz);
            // resseta a necessidade de cálculo
            calc = 0;
         }

         // calcula o polinômio linear na posição (X,Z) onde Y(X,Z)
         DX = X - p00.x;
         DZ = Z - p00.z;
         Y = a + b*DX + c*DZ + d*DX*DZ;

         // armazena os valores calculados
         aPlane[k*iPZ+l].x = (int)X;
         aPlane[k*iPZ+l].z = (int)Z;
         aPlane[k*iPZ+l].y = (int)Y;
      }
   }

   // roda a animação
   Animation (aPlane,iPX,iPZ,-PI/16,-PI/16,PI/32);
   SetCursor (LoadCursor (NULL, IDC_ARROW));
}
```

APÊNDICE E

LISTAGEM DA INTERPOLAÇÃO POLINOMIAL DE LAGRANGE PARA CURVAS

```
void TMainWindow::Lagrange (void)
{
   // Interpolação Polinomial de Lagrange para Curvas
   DesignOriented  Dsg (DisplayWindow->HWindow, wScrollOffsetX,
                        wScrollOffsetY,BitmapHandle,
                        wImageWidthInPixels, wImageHeightInPixels);
   PointSet        P;
   int             x, y, x1, y1, x2, y2;
   long double     Y, prod1, prod2, temp, tempY;

   SetCursor (LoadCursor (NULL, IDC_WAIT));
   Dsg.SetSolidPen(0,0,255);

   // acerta os valores lidos de acordo com o eixo de coordenadas
   Points.Pointer = Points.First();
   while (Points.Pointer != Points.End())
   {
      P.Insert ( Points.ReadX(Points.Pointer)-50,
                 350-Points.ReadY(Points.Pointer));
      Points.Pointer = Points.Next(Points.Pointer);
   }

   // Equação:
   //
   //            n            n        (x-xj)
   // p(x) = somatório ((produtório --------).yi)
   //           i=0          j=0       (xi-xj)
   //                        j!=i
   //

   P.ASortByX();
   if (P.SizeOf() >= 2)
   {
      x1 = P.ReadX(P.First());
      y1 = P.ReadY(P.First());
      for (x=P.ReadX(P.First())+1;x<=P.ReadX(P.Last());x++)
      {
         tempY = 0;
         P.Pointer = P.First();
         while (P.Pointer != P.End())
         {
            prod1 = 1;
            prod2 = 1;
            P.Pointer2 = P.First();
            while (P.Pointer2 != P.End())
            {
               if (P.Pointer != P.Pointer2)
               {
                  prod1 = prod1 * (x-P.ReadX(P.Pointer2));
```

```
                prod2 = prod2 * (P.ReadX(P.Pointer)-P.ReadX(P.Pointer2));
              }
              P.Pointer2 = P.Next(P.Pointer2);
            };
            tempY = tempY + ((prod1/prod2)*P.ReadY(P.Pointer));
            P.Pointer = P.Next(P.Pointer);
          };

          // testa overflow para variável float
          temp = fabsl(tempY);

          if ((temp>=3e-38)&&(temp<=3e+38))
          {
            // recebe o valor de Y
            Y = (float)tempY;
          }
          else
          {
            // recuperação de overflow para variável float
            if (temp<3e-38)
            {
              if (tempY>=0)
                Y = 3e-38;
              else
                Y = -3e-38;
            }
            else
            {
              if (tempY>=0)
                Y = 3e+38;
              else
                Y = -3e+38;
            }
          }

          // arredondamento
          y = Round((float)Y);
          x2 = x;
          y2 = y;

          // liga os pontos para desenhar a curva resultante
          Dsg.WindowLine(x1+50, y1+50, x2+50, y2+50, R2_COPYPEN);
          Dsg.BitmapLine(x1+50, y1+50, x2+50, y2+50, R2_COPYPEN);
          x1 = x2;
          y1 = y2;
        }
      }
      Dsg.SetSolidPen(0,0,0);
      SetCursor (LoadCursor (NULL, IDC_ARROW));
}
```

```
int TMainWindow::Round (float a)
{
   // arredonda um float
   int     b;
   float   c;

   b = (int)a;
   c =fabs(a) - abs(b);
   if (c>=0.5)
   {
      if (b>=0)
         b++;
      else
         b--;
   }
   return b;
}
```

APÊNDICE F

LISTAGEM DA INTERPOLAÇÃO POLINOMIAL DE LAGRANGE PARA SUPERFÍCIES

```
void TMainWindow::Lagrange3D (void)
{
   // Interpolação Polinomial de Lagrange para Superfícies

   float           X, Y, Z, deltaX, deltaZ, Xmax, Xmin, Zmax, Zmin;
   long double     prodX1, prodX2, prodZ1, prodZ2, temp, tempY1, tempY2;
   int             s, t, i, j, k ,l;

   SetCursor (LoadCursor (NULL, IDC_WAIT));

   // Equação:
   //
   //              iGX        iGZ        iGZ       (Z - zi,l)
   // p(x,z) = somatório (somatório ((produtório --------------).yi,j)) . B
   //              i=0        j=0        l=0      (zi,j - zi,l)
   //                                   l!=j
   //
   // onde:
   //
   //
   //         iGX       (X - xk,j)
   // B = produtório ----_----------
   //         k=0      (xi,j - xk,j)
   //         k!=i
   //

   // calcula variação da nova grade
   Xmax = aGrid[(iGX-1)*iGZ].x;
   Xmin = aGrid[0].x;
   Zmax = aGrid[iGZ-1].z;
   Zmin = aGrid[0].z;

   deltaX  = ((Xmax-Xmin)/(iPX-1));
   deltaZ  = ((Zmax-Zmin)/(iPZ-1));

   for (s=0; s<iPX; s++)
   {
      X = Xmin + s*deltaX;
      for (t=0; t<iPZ; t++)
      {
         Z = Zmin + t*deltaZ;

         tempY2 = 0;
         for (i=0; i<iGX; i++)
         {
            // calcula na direção Z
            tempY1 = 0;
            for (j=0; j<iGZ; j++)
            {
```

```
      prodZ1 = 1;
      prodZ2 = 1;
      for (l=0; l<iGZ; l++)
      {
         if (l!=j)
         {
            prodZ1 = prodZ1 * (Z - aGrid[l].z);
            prodZ2 = prodZ2 * (aGrid[j].z - aGrid[l].z);
         }
      }

      if (prodZ2==0)
      {
           MessageBox(HWindow,"Error: divide by zero!","",MB_OK);
           return;
      }
      else
         tempY1 = tempY1 + (prodZ1/prodZ2)*aGrid[i*iGZ+j].y;
   }

   // calcula em X e Z
   prodX1 = 1;
   prodX2 = 1;
   for (k=0; k<iGX; k++)
   {
      if (k!=i)
      {
         prodX1 = prodX1 * (X - aGrid[k*iGZ].x);
         prodX2 = prodX2 * (aGrid[i*iGZ].x - aGrid[k*iGZ].x);
      }
   }
   if (prodX2==0)
   {
      MessageBox(HWindow,"Error: divide by zero!","",MB_OK);
      return;
   }
   else
      tempY2 = tempY2 + (prodX1/prodX2)*tempY1;
}

// testa overflow para variável float
temp = fabsl(tempY2);

if ((temp>=3e-38)&&(temp<=3e+38))
{
   // recebe o valor de Y
   Y = (float)tempY2;
}
else
{
   // recuperação de overflow para variável float
   if (temp<3e-38)
   {
      if (tempY2>=0)
         Y = 3e-38;
      else
         Y = -3e-38;
   }
   else
```

```
            {
               if (tempY2>=0)
                  Y = 3e+38;
               else
                  Y = -3e+38;
            }
         }

         // armazena os valores calculados
         aPlane[s*iPZ+t].x = X;
         aPlane[s*iPZ+t].z = Z;
         aPlane[s*iPZ+t].y = Y;
      }
   }

   // roda a animação
   Animation (aPlane,iPX,iPZ,-PI/16,-PI/16,PI/32);
   SetCursor (LoadCursor (NULL, IDC_ARROW));
}
```

APÊNDICE G

LISTAGEM DA INTERPOLAÇÃO SPLINE PARA CURVAS

```
void TMainWindow::Spline (void)
{
   // Interpolação Spline para Curvas
   DesignOriented  Dsg( DisplayWindow->HWindow, wScrollOffsetX,
                        wScrollOffsetY, BitmapHandle,
                        wImageWidthInPixels, wImageHeightInPixels);
   PointSet        P;
   int             x1, y1, x2, y2, n, i, j;
   float           Y, X;
   float           *x, *y, *b, *c, *d;

   SetCursor (LoadCursor (NULL, IDC_WAIT));
   Dsg.SetSolidPen(255,0,255);
   n=Points.SizeOf();
   if (n<2)
      return;

   // aloca memória para os coeficientes dos polinômios cúbicos
   if((x=(float*)malloc(n*sizeof(float))) == NULL ||
       (y=(float*)malloc(n*sizeof(float))) == NULL ||
       (b=(float*)malloc(n*sizeof(float))) == NULL ||
       (c=(float*)malloc(n*sizeof(float))) == NULL ||
       (d=(float*)malloc(n*sizeof(float))) == NULL)
   {
      if(x) free(x);
      if(y) free(y);
      if(b) free(b);
      if(c) free(c);
      if(d) free(d);
      MessageBox(HWindow,"Memory error!","",MB_OK);
      // aborta o programa
      abort();
      return;
   }
   else
   {
      // acerta os valores lidos de acordo com o eixo de coordenadas
      Points.Pointer = Points.First();
      while (Points.Pointer != Points.End())
      {
         P.Insert ( Points.ReadX(Points.Pointer)-50,
                    350-Points.ReadY(Points.Pointer));
         Points.Pointer = Points.Next(Points.Pointer);
      }

      // ordena por abcissa
      P.ASortByX();
```

```
      // passa os valores da cadeia de 'containers' para 'buffers' estáticos
      i=0;
      P.Pointer = P.First();
      while (P.Pointer != P.End())
      {
         x[i] = P.ReadX(P.Pointer);
         y[i] = P.ReadY(P.Pointer);
         P.Pointer = P.Next(P.Pointer);
         i++;
      }

      // prepara os coeficientes das equações de cada segmento
      // total: n-1 segmentos
      CSpline(n,x,y,b,c,d);
      x1 = x[0];
      y1 = y[0];

      // varre todos os segmentos a serem interpolados
      for(i=0; i<(n-1); i++)
      {
         for(j=x[i]; j<=(x[i+1]-1); j++)
         {
            X = j;
            Y = y[i] + (X-x[i])*(b[i] + (X-x[i])*(c[i]
                      + d[i]*(X-x[i])));
            x2 = int(X);
            y2 = int(Y);

            // liga os pontos para desenhar a curva resultante
            Dsg.WindowLine(x1+50, y1+50, x2+50, y2+50, R2_COPYPEN);
            Dsg.BitmapLine(x1+50, y1+50, x2+50, y2+50, R2_COPYPEN);
            x1 = x2;
            y1 = y2;
         }
      }
      x2 = x[n-1];
      y2 = y[n-1];

      // liga o último ponto
      Dsg.WindowLine(x1+50, y1+50, x2+50, y2+50, R2_COPYPEN);
      Dsg.BitmapLine(x1+50, y1+50, x2+50, y2+50, R2_COPYPEN);

      // libera memória alocada
      if(x) free(x);
      if(y) free(y);
      if(b) free(b);
      if(c) free(c);
      if(d) free(d);
   }
   Dsg.SetSolidPen(0,0,0);
   SetCursor (LoadCursor (NULL, IDC_ARROW));
}
```

```
void TMainWindow::CSpline(int NPoints, float *x, float *y,
                          float *b, float *c, float *d)
{
   // Calcula os coeficientes dos segmentos
   //
   // Entrada:
   //
   // NPoints    = número de pontos a serem interpolados
   // x,y        = 'arrays' com as coordenadas dos pontos em ordem
   //              crescente por abcissa
   // b,c,d      = 'arrays' para armazenamento dos coeficientes
   //              de cada segmento
   //
   // Equação de cada segmento i, onde i=0..(NPoints-2):
   //
   // P(x)       = yi + bi(x-xi) + ci(x-xi)^2 + di(x-xi)^3
   //
   // Saída:
   //
   // b,c,d      = 'arrays' preenchidas com os coeficientes
   //
   // Cálculo através do uso de diferenças divididas:
   //
   // y[i]       = s(x(i))
   // b(i)       = s'(x(i))
   // c(i)       = s''(x(i))/2
   // d(i)       = s'''(x(i))/6  (derivada da direita)

   int     i, j, N;
   float   T;

   N = NPoints - 1;
   // chamada irregular
   if (NPoints<2)
      return;
   // somente dois pontos
   if (NPoints==2)
   {
      b[0] = (y[1]-y[0])/(x[1]-x[0]);
      c[0] = 0;
      d[0] = 0;
      b[1] = b[0];
      c[1] = 0;
      d[1] = 0;
      return;
   }

   // prepara a matriz tridiagonal do sistema de equações
   // valores armazenados temporariamente para cáculo de b, c e d:
   // b  = alfa
   // c  = beta
   // d  = h
   d[0] = x[1]-x[0];
   c[1] = (y[1]-y[0])/d[0];
   for(i=1;i<N;i++)
   {
      d[i]    = x[i+1]-x[i];
      b[i]    = 2.*(d[i-1]+d[i]);
      c[i+1]  = (y[i+1]-y[i])/d[i];
```

```
      c[i]    = c[i+1]-c[i];
   }

   // condições das extremidades, terceira derivada em x(0)
   // e x(NPoints-1), obtidas a partir das diferenças divididas
   b[0] = -d[0];
   b[N] = -d[N-1];
   c[0] = 0;
   c[N] = 0;
   if (NPoints!=3)
   {
      c[0] = c[2]/(x[3]-x[1])-c[1]/(x[2]-x[0]);
      c[N] = c[N-1]/(x[N]-x[N-2])-c[N-2]/(x[N-1]-x[N-3]);
      c[0] = (c[0]*d[0]*d[0])/(x[3]-x[0]);
      c[N] = (-c[N]*d[N-1]*d[N-1])/(x[N]-x[N-3]);
   }

   // eliminação para frente
   for(i=1;i<=N;i++)
   {
      T    = d[i-1]/b[i-1];
      b[i] = b[i]-T*d[i-1];
      c[i] = c[i]-T*c[i-1];
   }

   // substituição para trás
   c[N] = c[N]/b[N];
   for(j=0;j<N;j++)
   {
      i    = N-1-j;
      c[i] = (c[i]-d[i]*c[i+1])/b[i];
   }
   // c[i] = sigma(i)

   // cálculo dos coeficientes dos polinômios
   b[N] = (y[N]-y[N-1])/d[N-1]+d[N-1]*(c[N-1]+2*c[N]);
   for(i=0;i<N;i++)
   {
      b[i] = (y[i+1]-y[i])/d[i]-d[i]*(c[i+1]+2*c[i]);
      d[i] = (c[i+1]-c[i])/d[i];
      c[i] = 3*c[i];
   }
   c[N] = 3*c[N];
   d[N] = d[N-1];
}
```

APÊNDICE H

LISTAGEM DA INTERPOLAÇÃO BÉZIER PARA CURVAS

```
void TMainWindow::Bezier (void)
{
   // Interpolação Bézier para Curvas
   DesignOriented  Dsg( DisplayWindow->HWindow, wScrollOffsetX,
                        wScrollOffsetY, BitmapHandle,
                        wImageWidthInPixels, wImageHeightInPixels);
   PointSet        P;
   int             x1, y1, x2, y2, n, i;
   float           Y, X, t, delta;
   float           *x, *y, *a, *b, *c;

   SetCursor (LoadCursor (NULL, IDC_WAIT));
   Dsg.SetSolidPen(255,50,100);

   n=Points.SizeOf();
   if (n<2)
      return;

   // aloca memória
   if ((x=(float*)malloc(n*sizeof(float))) == NULL ||
       (y=(float*)malloc(n*sizeof(float))) == NULL ||
       (a=(float*)malloc(n*sizeof(float))) == NULL ||
       (b=(float*)malloc(n*sizeof(float))) == NULL ||
       (c=(float*)malloc(n*sizeof(float))) == NULL)
   {
      if(x) free(x);
      if(y) free(y);
      if(a) free(a);
      if(b) free(b);
      if(c) free(c);
      MessageBox(HWindow,"Memory error!","",MB_OK);
      abort(); // aborta o programa
      return;
   }
   else
   {
      // acerta os valores lidos de acordo com o eixo de coordenadas
      Points.Pointer = Points.First();
      while (Points.Pointer != Points.End())
      {
         P.Insert(Points.ReadX(Points.Pointer)-50,
                  350-Points.ReadY(Points.Pointer));
         Points.Pointer = Points.Next(Points.Pointer);
      }

      // passa os valores da cadeia de 'containers' para 'buffers' estáticos
      i=0;
      P.Pointer = P.First();
      while (P.Pointer != P.End())
```

```
{
   x[i] = P.ReadX(P.Pointer);
   y[i] = P.ReadY(P.Pointer);
   P.Pointer = P.Next(P.Pointer);
   i++;
}

// Equação:
//              n
// Q(t) = somatório (Pi * Bi,n(t))
//            i=0
// onde:
//
// Bi,n(t) = C(n,i) * ((1-t)^(n-i)) * (t^i)
// C(n,i)  = (n!/((n-i)!i!))
//
// de outra forma:
//
//              n
// Q(t) = Somatório((n!/((n-i)!i!)) * ((1-t)^(n-i)) * (t^i) * Pi)
//            i=0
//
// onde:
//
// n  = número de pontos - 1
// t  = 0..1
// Pi = valor do ponto na posição i

// prepara as constantes iniciais (n!/((n-i)!i!))
Binomial (n,a);

// varia a função paramétrica de 0 até 1
x1 = x[0];
y1 = y[0];

// calcula variação paramétrica
if ((fabs(x[n-1]-x[0]))>(fabs(y[n-1]-y[0])))
   delta = (fabs(x[n-1]-x[0]));
else
   delta = (fabs(y[n-1]-y[0]));
if (delta!=0)
   delta = 1/delta;
else
   delta = 0.005;

for (t=0; t<1; t+=delta)
{
   // prepara os valores de (1-t)^(n-i) e t^i
   cBezier(n,t,b,c);

   // calcula os valores de X e Y
   X = 0;
   Y = 0;
   for (i=0; i<n; i++)
   {
      X = X + a[i]*b[i]*c[i]*x[i];
      Y = Y + a[i]*b[i]*c[i]*y[i];
   }
   x2 = int(X);
```

```
        y2 = int(Y);

        // liga os pontos para desenhar a curva resultante
        Dsg.WindowLine(x1+50, y1+50, x2+50, y2+50, R2_COPYPEN);
        Dsg.BitmapLine(x1+50, y1+50, x2+50, y2+50, R2_COPYPEN);
        x1 = x2;
        y1 = y2;
      }
      x2 = x[n-1];
      y2 = y[n-1];

      // liga o último ponto
      Dsg.WindowLine(x1+50, y1+50, x2+50, y2+50, R2_COPYPEN);
      Dsg.BitmapLine(x1+50, y1+50, x2+50, y2+50, R2_COPYPEN);

      // libera a memória alocada
      if(x) free(x);
      if(y) free(y);
      if(a) free(a);
      if(b) free(b);
      if(c) free(c);
   }
   Dsg.SetSolidPen(0,0,0);
   SetCursor (LoadCursor (NULL, IDC_ARROW));
}

void TMainWindow::Binomial (int NPoints, float *a)
{
   // calcula os coeficientes para as equações de Binomial
   // (n!/((n-i)!i!)) onde n = número de pontos - 1
   // NPoints    = número de pontos
   // a          = vetor para armazenamento de (n!/((n-i)!i!))

   int     n, i;
   float   temp, nf;

   n = NPoints-1;
   a[0] = 1;
   nf = Factorial(n);
   temp = 1;
   for (i=1; i<=n; i++)
   {
      temp = temp*i;
      a[i] = (nf/((Factorial(n-i))*temp));
   }
  return;
}
```

```
void TMainWindow::cBezier (int NPoints, float t, float *b, float *c)
{
   // Calcula valores para Interpolação Bézier
   // NPoints    = número de pontos
   // t          = posição na função paramétrica (0..1)
   // b          = vetor para armazenamento de (1-t)^(n-i)
   // c          = vetor para armazenamento de t^i

   // Equação de Bézier:
   //             n
   // Q(t) = Somatório((n!/((n-i)!i!)) * ((1-t)^(n-i)) * (t^i) * Pi)
   //            i=0
   //
   // onde:
   //
   // n  = número de pontos - 1
   // t  = 0..1
   // Pi = valor do ponto na posição i

   int        n,i;
   float      s;

   // calcula os valores de (1-t)^(n-i) e t^i
   n = NPoints-1;
   s = 1-t;
   b[n] = 1;
   c[0] = 1;
   for (i=1; i<=n; i++)
   {
      b[n-i]  = b[n-i+1]*s;
      c[i]    = c[i-1]*t;
   }
  return;
}

float TMainWindow::Factorial (int a)
{
   // calcula e retorna o fatorial de a
   int     i;
   float   j=1;

   if (a<=1)
      return j;
   for (i=2; i<=a; i++)
   {
      j = j*i;
   }
   return j;
}
```

APÊNDICE I

LISTAGEM DA INTERPOLAÇÃO BÉZIER PARA SUPERFÍCIES

```
void TMainWindow::Bezier3D (void)
{
   // Interpolação Bézier para Superfícies
   float      X, Y, Z, tempX, tempY, tempZ, s, t,
              *ax, *bx, *cx, *az, *bz, *cz;
   int        i, j, k=0, l=0;

   SetCursor (LoadCursor (NULL, IDC_WAIT));

  // aloca memória
   if ((ax=(float*)malloc(iGX*sizeof(float))) == NULL ||
       (bx=(float*)malloc(iGX*sizeof(float))) == NULL ||
       (cx=(float*)malloc(iGX*sizeof(float))) == NULL ||
       (az=(float*)malloc(iGZ*sizeof(float))) == NULL ||
       (bz=(float*)malloc(iGZ*sizeof(float)))  = NULL ||
       (cz=(float*)malloc(iGZ*sizeof(float))) == NULL)
   {
      if(ax) free(ax);
      if(bx) free(bx);
      if(cx) free(cx);
      if(az) free(az);
      if(bz) free(bz);
      if(cz) free(cz);
      MessageBox(HWindow,"Memory error!","",MB_OK);
      abort(); // aborta o programa
      return;
   }

   // Equação:
   //              m          n
   // Q(s,t) = somatório (somatório (Pi,j * Bj,n(t))) . Bi,m(s)
   //             i=0        j=0
   // onde:
   //
   // Bi,m(s) = C(i,m) * ((1-s)^(m-i)) * (s^i)
   // Bj,n(t) = C(j,n) * ((1-t)^(n-j)) * (t^j)
   // C(i,m)  = (m!/((m-i)!i!))
   // C(j,n)  = (n!/((n-j)!j!))
   //
   //
   // s       = 0..1
   // t       = 0..1
   // Pi,j    = valor do ponto na posição i,j
   // m       = número de pontos da grade em x - 1
   // n       = número de pontos da grade em z - 1

   // prepara as constantes iniciais C(i,iGX) e C(j,iGZ)
   Binomial (iGX,ax);
   Binomial (iGZ,az);
```

```
   for (k=0; k<iPX; k++)
   {
      // prepara os valores de ((1-s)^(iGX-i)) * (s^i)
      s = (float)k/(iPX-1);
      cBezier(iGX,s,bx,cx);

      for (l=0; l<iPZ; l++)
      {
         // prepara os valores de ((1-t)^(iGZ-j)) * (t^j)
         t = (float)l/(iPZ-1);
         cBezier(iGZ,t,bz,cz);

         // calcula os valores de X e Y
         X = 0;
         Y = 0;
         Z = 0;
         for (i=0; i<iGX; i++)
         {
            tempX = 0;
            tempY = 0;
            tempZ = 0;
            for (j=0; j<iGZ; j++)
            {
               tempX = tempX + az[j]*bz[j]*cz[j]*aGrid[i*iGZ+j].x;
               tempY = tempY + az[j]*bz[j]*cz[j]*aGrid[i*iGZ+j].y;
               tempZ = tempZ + az[j]*bz[j]*cz[j]*aGrid[i*iGZ+j].z;
            }
            X = X + ax[i]*bx[i]*cx[i]*tempX;
            Y = Y + ax[i]*bx[i]*cx[i]*tempY;
            Z = Z + ax[i]*bx[i]*cx[i]*tempZ;
         }

         // armazena os valores calculados
         aPlane[k*iPZ+l].x = (int)X;
         aPlane[k*iPZ+l].z = (int)Z;
         aPlane[k*iPZ+l].y = (int)Y;
      }
   }

   // libera memória alocada
   if(ax) free(ax);
   if(bx) free(bx);
   if(cx) free(cx);
   if(az) free(az);
   if(bz) free(bz);
   if(cz) free(cz);

   // roda a animação
   Animation (aPlane,iPX,iPZ,-PI/16,-PI/16,PI/32);
   SetCursor (LoadCursor (NULL, IDC_ARROW));
}
```

```
void TMainWindow::Binomial (int NPoints, float *a)
{
   // calcula os coeficientes para as equações de Binomial
   // (n!/((n-i)!i!)) onde n = número de pontos - 1
   // NPoints    = número de pontos
   // a          = vetor para armazenamento de (n!/((n-i)!i!))

   int     n, i;
   float   temp, nf;

   n = NPoints-1;
   a[0] = 1;
   nf = Factorial(n);
   temp = 1;
   for (i=1; i<=n; i++)
   {
      temp = temp*i;
      a[i] = (nf/((Factorial(n-i))*temp));
   }
  return;
}

void TMainWindow::cBezier (int NPoints, float t, float *b, float *c)
{
   // Calcula valores para Interpolação Bézier
   // NPoints    = número de pontos
   // t          = posição na função paramétrica (0..1)
   // b          = vetor para armazenamento de (1-t)^(n-i)
   // c          = vetor para armazenamento de t^i

   // Equação de Bézier:
   //             n
   // Q(t) = Somatório((n!/((n-i)!i!)) * ((1-t)^(n-i)) * (t^i) * Pi)
   //            i=0
   //
   // onde:
   //
   // n  = número de pontos - 1
   // t  = 0..1
   // Pi = valor do ponto na posição i

   int        n,i;
   float      s;

   // calcula os valores de (1-t)^(n-i) e t^i
   n = NPoints-1;
   s = 1-t;
   b[n] = 1;
   c[0] = 1;
   for (i=1; i<=n; i++)
   {
      b[n-i]  = b[n-i+1]*s;
      c[i]    = c[i-1]*t;
   }
  return;
}
```

```
float TMainWindow::Factorial (int a)
{
   // calcula e retorna o fatorial de a
   int      i;
   float    j=1;

   if (a<=1)
      return j;
   for (i=2; i<=a; i++)
   {
      j = j*i;
   }
   return j;
}
```

APÊNDICE J

LISTAGEM DA INTERPOLAÇÃO B-SPLINE PARA CURVAS

```
void TMainWindow::BSpline (void)
{
   // Interpolação 'Uniform Non-periodic B-Spline' com k=3
   DesignOriented  Dsg( DisplayWindow->HWindow, wScrollOffsetX,
                        wScrollOffsetY, BitmapHandle,
                        wImageWidthInPixels, wImageHeightInPixels);
   PointSet        P;
   int             x1, y1, x2, y2, n, m, i;
   float           Y, X, u, delta, b[4];
   float           *x, *y;

   SetCursor (LoadCursor (NULL, IDC_WAIT));
   Dsg.SetSolidPen(150,200,50);

   n = Points.SizeOf();
   m = n-1;
   if (n<4)
      return;

   // aloca memória
   if ((x=(float*)malloc(n*sizeof(float))) == NULL ||
       (y=(float*)malloc(n*sizeof(float))) == NULL)
   {
      if(x) free(x);
      if(y) free(y);
      MessageBox(HWindow,"Memory error!","",MB_OK);
      abort(); // aborta o programa
      return;
   }
   else
   {
      // acerta os valores lidos de acordo com o eixo de coordenadas
      Points.Pointer = Points.First();
      while (Points.Pointer != Points.End())
      {
         P.Insert(Points.ReadX(Points.Pointer)-50,
                  350-Points.ReadY(Points.Pointer));
         Points.Pointer = Points.Next(Points.Pointer);
      }
      // passa os valores da cadeia de 'containers' para 'buffers' estáticos
      i=0;
      P.Pointer = P.First();
      while (P.Pointer != P.End())
      {
         x[i] = P.ReadX(P.Pointer);
         y[i] = P.ReadY(P.Pointer);
         P.Pointer = P.Next(P.Pointer);
         i++;
      }
```

```
// Equação:
//
// para cada segmento de curva i, onde 3 <= i <= m, (m-2 segmentos):
//
// Q(t) = ((((1-t^3)^3)*P[i-3]) + ((3*t^3-6*t^2+4)*p[i-2])
//      + ((-3*t^3+3*t^2+3*t+1)*p[i-1]) + ((t^3)*p[i])) / 6
//
// onde:
//
// t    = variação uniforme dos nós (0..1)
// m    = número de pontos - 1 , onde m >= 3
// p[i] = valor do ponto de controle na posição i

// calcula os valores iniciais de X e Y
i = 3;
cBSpline(b,0);
X = b[3]*x[i-3] + b[2]*x[i-2]   + b[1]*x[i-1] + b[0]*x[i];
Y = b[3]*y[i-3] + b[2]*y[i-2]   + b[1]*y[i-1] + b[0]*y[i];
x1 = int(X);
y1 = int(Y);

// varre a lista de pontos
for (i=3; i<=m; i++)
{
   // calcula variação paramétrica
   if ((fabs(x[i]-x[i-1]))>(fabs(y[i]-y[i-1])))
      delta = (fabs(x[i]-x[i-1]));
   else
      delta = (fabs(y[i]-y[i-1]));
   if (delta!=0)
      delta = 1/delta;
   else
      delta = 0.005;

   // uniformemente espaçada entre os nós (0..1)
   // varia a função paramétrica para cada segmento de curva
   for (u=0; u<1; u+=delta)
   {
      // calcula os valores da mistura
      cBSpline(b,u);

      // calcula os valores de X e Y
      X = b[3]*x[i-3] + b[2]*x[i-2]   + b[1]*x[i-1] + b[0]*x[i];
      Y = b[3]*y[i-3] + b[2]*y[i-2]   + b[1]*y[i-1] + b[0]*y[i];
      x2 = int(X);
      y2 = int(Y);

      // liga os pontos para desenhar a curva resultante
      Dsg.WindowLine(x1+50, y1+50, x2+50, y2+50, R2_COPYPEN);
      Dsg.BitmapLine(x1+50, y1+50, x2+50, y2+50, R2_COPYPEN);
      x1 = x2;
      y1 = y2;
   }
}
// calcula os valores finais de X e Y
i = m;
cBSpline(b,1);
X = b[3]*x[i-3] + b[2]*x[i-2]   + b[1]*x[i-1] + b[0]*x[i];
Y = b[3]*y[i-3] + b[2]*y[i-2]   + b[1]*y[i-1] + b[0]*y[i];
```

```
      x2 = int(X);
      y2 = int(Y);

      // liga o último ponto
      Dsg.WindowLine(x1+50, y1+50, x2+50, y2+50, R2_COPYPEN);
      Dsg.BitmapLine(x1+50, y1+50, x2+50, y2+50, R2_COPYPEN);


      // libera a memória alocada
      if(x) free(x);
      if(y) free(y);
   }
   Dsg.SetSolidPen(0,0,0);
   SetCursor (LoadCursor (NULL, IDC_ARROW));
}

void TMainWindow::cBSpline (float *b, float u)
{
   // calcula os valores da mistura
   float      u2, u3;

   u2 = u*u;
   u3 = u2*u;
   b[3] = (-u3+3*u2-3*u+1)/6;
   b[2] = (3*u3-6*u2+4)/6;
   b[1] = (-3*u3+3*u2+3*u+1)/6;
   b[0] = u3/6;
   return;
}
```

APÊNDICE K

LISTAGEM DA INTERPOLAÇÃO B-SPLINE PARA SUPERFÍCIES

```
void TMainWindow::BSpline3D (int ExBorder)
{
   // Interpolação B-Spline para superfícies
   //
   // Entrada:
   //
   //    ExBorder   = número de linhas e colunas a serem acrescentadas na borda
   //
   // Saída:
   //
   //    animação na tela da grade interpolada por B-Spline

   POINT3D       *aG, p00, p01, p02, p03,
                 p10, p11, p12, p13,
                 p20, p21, p22, p23,
                 p30, p31, p32, p33;
   float         X, Y, Z, temp0, temp1, temp2, temp3, b[4], c[4], s, t;
   int           calc, n, NX, NZ, i, j, k=0, l=0;

   SetCursor (LoadCursor (NULL, IDC_WAIT));

   // verifica o número de linhas e colunas a serem adicionadas
   n = ExBorder;
   if ((n>5)||(n<0))
   {
      MessageBox(HWindow,"Replicated boundary vertices error!","",MB_OK);
      abort(); // aborta o programa
      return;
   }

   NX = iGX+2*n;
   NZ = iGZ+2*n;

   // aloca memória
   if((aG=(POINT3D*)malloc(NX*NZ*sizeof(POINT3D))) == NULL)
   {
      if(aG) free(aG);
      MessageBox(HWindow,"Memory error!","",MB_OK);
      abort(); // aborta o programa
      return;
   }

   // passa os valores da grade regular para uma nova grade
   // deixando espaço para mais n pontos nas bordas
   for (i=0; i<iGX; i++)
      for (j=0; j<iGZ; j++)
         aG[(i+n)*NZ+(j+n)] = aGrid[i*iGZ+j];

   // preenche as bordas
```

```
for (j=n; j<(NZ-n); j++)
{
   for (i=0; i<n; i++)
      aG[i*NZ+j] = aG[n*NZ+j];
   for (i=(NX-n); i<NX; i++)
      aG[i*NZ+j] = aG[(NX-n-1)*NZ+j];
}
for (i=0; i<NX; i++)
{
   for (j=0; j<n; j++)
      aG[i*NZ+j] = aG[i*NZ+n];
   for (j=(NZ-n); j<NZ; j++)
      aG[i*NZ+j] = aG[i*NZ+(NZ-n-1)];
}

// Equação:
//
// para cada área i,j, onde (3<=i<=NX) e (3<=j<= NZ):
//
//                   0          0
// Qi,j(u,v) = somatório (somatório p[i+m,j+n].bm[s]) . cn[t]
//                 m=-3       n=-3
//
// onde:
//
// b-3[s]  = (-s^3+3*s^2-3*s+1)/6;
// b-2[s]  = (3*s^3-6*s^2+4)/6;
// b-1[s]  = (-3*s^3+3*s^2+3*s+1)/6;
// b-0[s]  = s^3/6;
//
// c-3[s]  = (-t^3+3*t^2-3*t+1)/6;
// c-2[s]  = (3*t^3-6*t^2+4)/6;
// c-1[s]  = (-3*t^3+3*t^2+3*t+1)/6;
// c-0[s]  = t^3/6;
//
// s       =  variação uniforme dos nós (0..1) em x
// t       =  variação uniforme dos nós (0..1) em t
// p[i,j]  =  valor do ponto de controle na posição i
// NX      = número de pontos da grade em x, onde NX >= 4
// NZ      = número de pontos da grade em z, onde NZ >= 4

i = 3;
for (k=0; k<iPX; k++)
{
   j = 3;
   calc = 1; // indica necessidade de cálculo

   // caso ultrapasse o retalho em x
   s = ((float)k*(NX-3))/(iPX-1);
   while(s > (i-2))
   {
      i++;
      calc = 1;
   }
   s = s-i+3;

   // calcula os valores da mistura em x
   cBSpline(b,s);
```

```
for (l=0; l<iPZ; l++)
{
   // caso ultrapasse o retalho em z
   t = ((float)l*(NZ-3))/(iPZ-1);
   while(t > (j-2))
   {
      j++;
      calc = 1;
   }
   t = t-j+3;

   // calcula os valores da mistura em z
   cBSpline(c,t);

   // calcula os valores para o retalho
   if (calc == 1)
   {
      // pontos
      p00 = aG[i*NZ+j];
      p01 = aG[i*NZ+(j-1)];
      p02 = aG[i*NZ+(j-2)];
      p03 = aG[i*NZ+(j-3)];
      p10 = aG[(i-1)*NZ+j];
      p11 = aG[(i-1)*NZ+(j-1)];
      p12 = aG[(i-1)*NZ+(j-2)];
      p13 = aG[(i-1)*NZ+(j-3)];
      p20 = aG[(i-2)*NZ+j];
      p21 = aG[(i-2)*NZ+(j-1)];
      p22 = aG[(i-2)*NZ+(j-2)];
      p23 = aG[(i-2)*NZ+(j-3)];
      p30 = aG[(i-3)*NZ+j];
      p31 = aG[(i-3)*NZ+(j-1)];
      p32 = aG[(i-3)*NZ+(j-2)];
      p33 = aG[(i-3)*NZ+(j-3)];
      // resseta a necessidade de cálculo
      calc = 0;
   }

   temp0 = b[3]*p33.x + b[2]*p23.x + b[1]*p13.x + b[0]*p03.x;
   temp1 = b[3]*p32.x + b[2]*p22.x + b[1]*p12.x + b[0]*p02.x;
   temp2 = b[3]*p31.x + b[2]*p21.x + b[1]*p11.x + b[0]*p01.x;
   temp3 = b[3]*p30.x + b[2]*p20.x + b[1]*p10.x + b[0]*p00.x;
   X = temp0*c[3] + temp1*c[2] + temp2*c[1] + temp3*c[0];

   temp0 = b[3]*p33.y + b[2]*p23.y + b[1]*p13.y + b[0]*p03.y;
   temp1 = b[3]*p32.y + b[2]*p22.y + b[1]*p12.y + b[0]*p02.y;
   temp2 = b[3]*p31.y + b[2]*p21.y + b[1]*p11.y + b[0]*p01.y;
   temp3 = b[3]*p30.y + b[2]*p20.y + b[1]*p10.y + b[0]*p00.y;
   Y = temp0*c[3] + temp1*c[2] + temp2*c[1] + temp3*c[0];

   temp0 = b[3]*p33.z + b[2]*p23.z + b[1]*p13.z + b[0]*p03.z;
   temp1 = b[3]*p32.z + b[2]*p22.z + b[1]*p12.z + b[0]*p02.z;
   temp2 = b[3]*p31.z + b[2]*p21.z + b[1]*p11.z + b[0]*p01.z;
   temp3 = b[3]*p30.z + b[2]*p20.z + b[1]*p10.z + b[0]*p00.z;
   Z = temp0*c[3] + temp1*c[2] + temp2*c[1] + temp3*c[0];

   // armazena os valores calculados
   aPlane[k*iPZ+l].x = (int)X;
   aPlane[k*iPZ+l].z = (int)Z;
```

```
            aPlane[k*iPZ+l].y = (int)Y;
         }
      }

      // roda a animação
      Animation (aPlane,iPX,iPZ,-PI/16,-PI/16,PI/32);
      SetCursor (LoadCursor (NULL, IDC_ARROW));
}

void TMainWindow::cBSpline (float *b, float u)
{
      // calcula os valores da mistura
      float      u2, u3;

      u2 = u*u;
      u3 = u2*u;
      b[3] = (-u3+3*u2-3*u+1)/6;
      b[2] = (3*u3-6*u2+4)/6;
      b[1] = (-3*u3+3*u2+3*u+1)/6;
      b[0] = u3/6;
      return;
}
```

APÊNDICE L

LISTAGEM DA INTERPOLAÇÃO β-SPLINE PARA CURVAS

```
void TMainWindow::BetaSpline (int beta1, int beta2)
{
   // Interpolação 'Uniform Non-periodic Beta-Spline'
   //
   // Entrada:
   //
   //    beta1 = parâmetro 1 da BetaSpline
   //    beta2 = parâmetro 2 da BetaSpline

   DesignOriented  Dsg( DisplayWindow->HWindow, wScrollOffsetX,
                        wScrollOffsetY, BitmapHandle,
                        wImageWidthInPixels, wImageHeightInPixels);
   PointSet        P;
   int             x1, y1, x2, y2, n, m, i;
   float           Y, X, u, delta, b[4], *x, *y;

   SetCursor (LoadCursor (NULL, IDC_WAIT));
   Dsg.SetSolidPen(50,150,200);

   n = Points.SizeOf();
   m = n-1;
   if (n<4)
      return;

   // verifica o valor de gama
   if ((beta2 + 2*beta1*beta1*beta1 + 4*beta1*beta1 + 4*beta1 + 2)==0)
   {
      MessageBox(HWindow,"Beta parameters error!","",MB_OK);
      abort(); // aborta o programa
      return;
   }

   // aloca memória
   if ((x=(float*)malloc(n*sizeof(float))) == NULL ||
       (y=(float*)malloc(n*sizeof(float))) == NULL)
   {
      if(x) free(x);
      if(y) free(y);
      MessageBox(HWindow,"Memory error!","",MB_OK);
      abort(); // aborta o programa
      return;
   }
   else
   {
      // acerta os valores lidos de acordo com o eixo de coordenadas
      Points.Pointer = Points.First();
      while (Points.Pointer != Points.End())
      {
         P.Insert(Points.ReadX(Points.Pointer)-50,
```

```
            350-Points.ReadY(Points.Pointer));
   Points.Pointer = Points.Next(Points.Pointer);
}

// passa os valores da cadeia de 'containers' para 'buffers' estáticos
i=0;
P.Pointer = P.First();
while (P.Pointer != P.End())
{
   x[i] = P.ReadX(P.Pointer);
   y[i] = P.ReadY(P.Pointer);
   P.Pointer = P.Next(P.Pointer);
   i++;
}

// Equação:
//
// para cada segmento de curva i, onde 3 <= i <= m, (m-2 segmentos):
//
// Q(t) = ((b[3]*P[i-3]) + (b[2]*p[i-2]) + (b[1]*p[i-1]) + (b[0]*p[i]))
//
// onde:
//
// b1   = beta1
// b2   = beta2
// gama = b2 + 2*b1*b1*b1 + 4*b1*b1 + 4*b1 + 2
// b[3] = ((2*b1^3)-(6*b1^3)*u+(6*b1^3)*u^2-(2*b1^3)*u^3)/gama
// b[2] = ((b2+4*b1^2+4*b1)+(6*b1^3-6*b1)*u-(3*b2+6*b1^3+6*b1^2)*u2+
//          (2*b2+2*b1^3+2*b1^2+2*b1)*u^3)/gama
// b[1] = (2+(6*b1)*u+(3*b2+6*b1^2)*u^2-(2*b2+2*b1^2+2*b1+2)*u^3)/gama
// b[0] = (2*u^3)/gama
//
//  beta1=1 e beta2=0 => BetaSpline = BSpline
//
// u    = variação uniforme dos nós (0..1)
// m    = número de pontos - 1 , onde m >= 3
// p[i] = valor do ponto de controle na posição i
i = 3;

// calcula os valores da mistura
cBetaSpline(b,0,beta1,beta2);

// calcula os valores iniciais de X e Y
X = b[3]*x[i-3] + b[2]*x[i-2]   + b[1]*x[i-1] + b[0]*x[i];
Y = b[3]*y[i-3] + b[2]*y[i-2]   + b[1]*y[i-1] + b[0]*y[i];
x1 = int(X);
y1 = int(Y);

// varre os pontos do conjunto
for (i=3; i<=m; i++)
{
   // calcula variação paramétrica
   if ((fabs(x[i]-x[i-1]))>(fabs(y[i]-y[i-1])))
      delta = (fabs(x[i]-x[i-1]));
   else
      delta = (fabs(y[i]-y[i-1]));
   if (delta!=0)
      delta = 1/delta;
   else
```

```
            delta = 0.005;

          // uniformemente espaçada entre os nós (0..1)
          // varia a função paramétrica para cada segmento de curva
          for (u=0; u<1; u+=delta)
          {
             // calcula os valores da mistura
             cBetaSpline(b,u,beta1,beta2);

             // calcula os valores de X e Y
             X = b[3]*x[i-3] + b[2]*x[i-2]   + b[1]*x[i-1] + b[0]*x[i];
             Y = b[3]*y[i-3] + b[2]*y[i-2]   + b[1]*y[i-1] + b[0]*y[i];
             x2 = int(X);
             y2 = int(Y);

             // liga os pontos para desenhar a curva resultante
             Dsg.WindowLine(x1+50, y1+50, x2+50, y2+50, R2_COPYPEN);
             Dsg.BitmapLine(x1+50, y1+50, x2+50, y2+50, R2_COPYPEN);
             x1 = x2;
             y1 = y2;
          }
       }
       // calcula os valores finais de X e Y
       i = m;
       cBetaSpline(b,1,beta1,beta2);
       X = b[3]*x[i-3] + b[2]*x[i-2]   + b[1]*x[i-1] + b[0]*x[i];
       Y = b[3]*y[i-3] + b[2]*y[i-2]   + b[1]*y[i-1] + b[0]*y[i];
       x2 = int(X);
       y2 = int(Y);

       // liga o último ponto
       Dsg.WindowLine(x1+50, y1+50, x2+50, y2+50, R2_COPYPEN);
       Dsg.BitmapLine(x1+50, y1+50, x2+50, y2+50, R2_COPYPEN);
       if(x) free(x);
       if(y) free(y);
    }
    Dsg.SetSolidPen(0,0,0);
    SetCursor (LoadCursor (NULL, IDC_ARROW));
}

void TMainWindow::cBetaSpline (float *b, float u, int beta1, int beta2)
{
    // calcula os valores da mistura
    float      u2, u3, gama, b1, b12, b13, b2;

    u2 = u*u;
    u3 = u2*u;
    b1 = beta1;
    b12 = b1*b1;
    b13 = b12*b1;
    b2 = beta2;
    gama = b2 + 2*b1*b1*b1 + 4*b1*b1 + 4*b1 + 2;
    b[3] = ((2*b13)-(6*b13)*u+(6*b13)*u2-(2*b13)*u3)/gama;
    b[2] = ((b2+4*b12+4*b1)+(6*b13-6*b1)*u-(3*b2+6*b13+6*b12)*u2+
             (2*b2+2*b13+2*b12+2*b1)*u3)/gama;
    b[1] = (2+(6*b1)*u+(3*b2+6*b12)*u2-(2*b2+2*b12+2*b1+2)*u3)/gama;
    b[0] = (2*u3)/gama;
    return;
}
```

APÊNDICE M

LISTAGEM DA INTERPOLAÇÃO β-SPLINE PARA SUPERFÍCIES

```
void TMainWindow::BetaSpline3D (int beta1, int beta2, int ExBorder)
{
   // Interpolação Beta-Spline para superfícies
   //
   // Entrada:
   //
   //    beta1      = valor de beta1 para x e z
   //    beta2      = valor de beta2 para x e z
   //    ExBorder   = número de linhas e colunas a serem acrescentadas na borda
   //
   // Saída:
   //
   //    animação na tela da grade interpolada por Beta-Spline

   POINT3D       *aG, p00, p01, p02, p03,
                 p10, p11, p12, p13,
                 p20, p21, p22, p23,
                 p30, p31, p32, p33;
   float         X, Y, Z, temp0, temp1, temp2, temp3, b[4], c[4], s, t;
   int           calc, n, NX, NZ, i, j, k=0, l=0;

   SetCursor (LoadCursor (NULL, IDC_WAIT));

   // verifica o valor de gama
   if ((beta2 + 2*beta1*beta1*beta1 + 4*beta1*beta1 + 4*beta1 + 2)==0)
   {
      MessageBox(HWindow,"Beta parameters error!","",MB_OK);
      abort(); // aborta o programa
      return;
   }

   // verifica o número de linhas e colunas a serem adicionadas
   n = ExBorder;
   if ((n>5)||(n<0))
   {
      MessageBox(HWindow,"Replicated boundary vertices error!","",MB_OK);
      abort(); // aborta o programa
      return;
   }

   NX = iGX+2*n;
   NZ = iGZ+2*n;

   // aloca memória
   if((aG=(POINT3D*)malloc(NX*NZ*sizeof(POINT3D))) == NULL)
   {
      if(aG) free(aG);
      MessageBox(HWindow,"Memory error!","",MB_OK);
      abort(); // aborta o programa
```

```
   return;
}

// passa os valores da grade regular para uma nova grade
// deixando espaço para mais n pontos nas bordas
for (i=0; i<iGX; i++)
   for (j=0; j<iGZ; j++)
      aG[(i+n)*NZ+(j+n)] = aGrid[i*iGZ+j];

// preenche as bordas
for (j=n; j<(NZ-n); j++)
{
   for (i=0; i<n; i++)
      aG[i*NZ+j] = aG[n*NZ+j];
   for (i=(NX-n); i<NX; i++)
      aG[i*NZ+j] = aG[(NX-n-1)*NZ+j];
}
for (i=0; i<NX; i++)
{
   for (j=0; j<n; j++)
      aG[i*NZ+j] = aG[i*NZ+n];
   for (j=(NZ-n); j<NZ; j++)
      aG[i*NZ+j] = aG[i*NZ+(NZ-n-1)];
}

// Equação:
//
// para cada área i,j, onde (3<=i<=NX) e (3<=j<= NZ):
//
//                   0          0
// Qi,j(u,v) = somatório (somatório p[i+m,j+n].bm[s]) . cn[t]
//                  m=-3       n=-3
//
// onde:
//
// b1       = beta1
// b2       = beta2
// gama     = b2 + 2*b1*b1*b1 + 4*b1*b1 + 4*b1 + 2
//
// b-0[s]   = ((2*b1^3)-(6*b1^3)*s+(6*b1^3)*s^2-(2*b1^3)*s^3)/gama
// b-1[s]   = ((b2+4*b1^2+4*b1)+(6*b1^3-6*b1)*s-(3*b2+6*b1^3+6*b1^2)*s^2+
//              (2*b2+2*b1^3+2*b1^2+2*b1)*s^3)/gama
// b-2[s]   = (2+(6*b1)*s+(3*b2+6*b1^2)*s^2-(2*b2+2*b1^2+2*b1+2)*s^3)/gama
// b-3[s]   = (2*s^3)/gama
//
// c-0[t]   = ((2*b1^3)-(6*b1^3)*t+(6*b1^3)*t^2-(2*b1^3)*t^3)/gama
// c-1[t]   = ((b2+4*b1^2+4*b1)+(6*b1^3-6*b1)*t-(3*b2+6*b1^3+6*b1^2)*t^2+
//              (2*b2+2*b1^3+2*b1^2+2*b1)*t^3)/gama
// c-2[t]   = (2+(6*b1)*s+(3*b2+6*b1^2)*t^2-(2*b2+2*b1^2+2*b1+2)*t^3)/gama
// c-3[t]   = (2*t^3)/gama
//
//  beta1=1 e beta2=0 => BetaSpline = BSpline
//
// s        =  variação uniforme dos nós (0..1) em x
// t        =  variação uniforme dos nós (0..1) em t
// p[i,j]   =  valor do ponto de controle na posição i
// NX       = número de pontos da grade em x, onde NX >= 4
// NZ       = número de pontos da grade em z, onde NZ >= 4
```

```
i = 3;
for (k=0; k<iPX; k++)
{
   j = 3;
   calc = 1; // indica necessidade de cálculo

   // caso ultrapasse o retalho em x
   s = ((float)k*(NX-3))/(iPX-1);
   while(s > (i-2))
   {
      i++;
      calc = 1;
   }
   s = s-i+3;

   // calcula os valores da mistura
   cBetaSpline(b,s,beta1,beta2);

   for (l=0; l<iPZ; l++)
   {
      // caso ultrapasse o retalho em z
      t = ((float)l*(NZ-3))/(iPZ-1);
      while(t > (j-2))
      {
         j++;
         calc = 1;
      }
      t = t-j+3;

      // calcula os valores da mistura
      cBetaSpline(c,t,beta1,beta2);

      // calcula os valores para o retalho
      if (calc == 1)
      {
         // pontos
         p00 = aG[i*NZ+j];
         p01 = aG[i*NZ+(j-1)];
         p02 = aG[i*NZ+(j-2)];
         p03 = aG[i*NZ+(j-3)];
         p10 = aG[(i-1)*NZ+j];
         p11 = aG[(i-1)*NZ+(j-1)];
         p12 = aG[(i-1)*NZ+(j-2)];
         p13 = aG[(i-1)*NZ+(j-3)];
         p20 = aG[(i-2)*NZ+j];
         p21 = aG[(i-2)*NZ+(j-1)];
         p22 = aG[(i-2)*NZ+(j-2)];
         p23 = aG[(i-2)*NZ+(j-3)];
         p30 = aG[(i-3)*NZ+j];
         p31 = aG[(i-3)*NZ+(j-1)];
         p32 = aG[(i-3)*NZ+(j-2)];
         p33 = aG[(i-3)*NZ+(j-3)];
         // resseta a necessidade de cálculo
         calc = 0;
      }

      temp0 = b[3]*p33.x + b[2]*p23.x + b[1]*p13.x + b[0]*p03.x;
      temp1 = b[3]*p32.x + b[2]*p22.x + b[1]*p12.x + b[0]*p02.x;
      temp2 = b[3]*p31.x + b[2]*p21.x + b[1]*p11.x + b[0]*p01.x;
```

```
            temp3 = b[3]*p30.x + b[2]*p20.x + b[1]*p10.x + b[0]*p00.x;
            X = temp0*c[3] + temp1*c[2] + temp2*c[1] + temp3*c[0];

            temp0 = b[3]*p33.y + b[2]*p23.y + b[1]*p13.y + b[0]*p03.y;
            temp1 = b[3]*p32.y + b[2]*p22.y + b[1]*p12.y + b[0]*p02.y;
            temp2 = b[3]*p31.y + b[2]*p21.y + b[1]*p11.y + b[0]*p01.y;
            temp3 = b[3]*p30.y + b[2]*p20.y + b[1]*p10.y + b[0]*p00.y;
            Y = temp0*c[3] + temp1*c[2] + temp2*c[1] + temp3*c[0];

            temp0 = b[3]*p33.z + b[2]*p23.z + b[1]*p13.z + b[0]*p03.z;
            temp1 = b[3]*p32.z + b[2]*p22.z + b[1]*p12.z + b[0]*p02.z;
            temp2 = b[3]*p31.z + b[2]*p21.z + b[1]*p11.z + b[0]*p01.z;
            temp3 = b[3]*p30.z + b[2]*p20.z + b[1]*p10.z + b[0]*p00.z;
            Z = temp0*c[3] + temp1*c[2] + temp2*c[1] + temp3*c[0];

            // armazena os valores calculados
            aPlane[k*iPZ+l].x = (int)X;
            aPlane[k*iPZ+l].z = (int)Z;
            aPlane[k*iPZ+l].y = (int)Y;
         }
      }

      // roda a animação
      Animation (aPlane,iPX,iPZ,-PI/16,-PI/16,PI/32);
      SetCursor (LoadCursor (NULL, IDC_ARROW));
}

void TMainWindow::cBetaSpline (float *b, float u, int beta1, int beta2)
{
      // calcula os valores da mistura
      float      u2, u3, gama, b1, b12, b13, b2;

      u2 = u*u;
      u3 = u2*u;
      b1 = beta1;
      b12 = b1*b1;
      b13 = b12*b1;
      b2 = beta2;
      gama = b2 + 2*b1*b1*b1 + 4*b1*b1 + 4*b1 + 2;
      b[3] = ((2*b13)-(6*b13)*u+(6*b13)*u2-(2*b13)*u3)/gama;
      b[2] = ((b2+4*b12+4*b1)+(6*b13-6*b1)*u-(3*b2+6*b13+6*b12)*u2+
               (2*b2+2*b13+2*b12+2*b1)*u3)/gama;
      b[1] = (2+(6*b1)*u+(3*b2+6*b12)*u2-(2*b2+2*b12+2*b1+2)*u3)/gama;
      b[0] = (2*u3)/gama;
      return;
}
```

APÊNDICE N

LISTAGEM DA INTERPOLAÇÃO AKIMA PARA AJUSTE DE CURVAS SUAVES SOBRE FUNÇÕES DE VALORES ÚNICOS

```
void TMainWindow::Akima (void)
{
   // Interpolação Akima para Curvas
   DesignOriented  Dsg( DisplayWindow->HWindow, wScrollOffsetX,
                        wScrollOffsetY, BitmapHandle,
                        wImageWidthInPixels, wImageHeightInPixels);
   PointSet        P;
   int             x1, y1, x2, y2, n, m, i, j;
   float           Y, X, dx, dy, b[4], *x, *y, *t;

   SetCursor (LoadCursor (NULL, IDC_WAIT));
   Dsg.SetSolidPen(150,150,150);

   n = Points.SizeOf();
   m = n+4;
   if (n<3)
      return;

   // aloca memória
   if((x=(float*)malloc(m*sizeof(float))) == NULL ||
      (y=(float*)malloc(m*sizeof(float))) == NULL ||
      (t=(float*)malloc(m*sizeof(float))) == NULL)
   {
      if(x) free(x);
      if(y) free(y);
      if(t) free(t);
      MessageBox(HWindow,"Memory error!","",MB_OK);
      abort(); // aborta o programa
      return;
   }
   else
   {
      // acerta os valores lidos de acordo com o eixo de coordenadas
      Points.Pointer = Points.First();
      while (Points.Pointer != Points.End())
      {
         P.Insert(Points.ReadX(Points.Pointer)-50,
                  350-Points.ReadY(Points.Pointer));
         Points.Pointer = Points.Next(Points.Pointer);
      }
      P.ASortByX();

      // passa os valores da cadeia de 'containers' para 'buffers'
      // deixando espaço para mais dois pontos nas extremidades
      i=2;
      P.Pointer = P.First();
      while (P.Pointer != P.End())
```

```
{
   x[i] = P.ReadX(P.Pointer);
   y[i] = P.ReadY(P.Pointer);
   P.Pointer = P.Next(P.Pointer);
   i++;
}

// Equação:
//
// para cada segmento de curva i, onde 2 <= i <= m-3 (m-4 segmentos):
//
// y(x) = b[0] + b[1]*(x-x[i]) + b[2]*(x-x[i])^2 + b[3]*(x-x[i])^3
//
// onde:
//
// t[i] = valor da derivada no ponto i
// m    = número de pontos + 4, onde os dois de cada
//        extremidade são calculados pelo método de Akima
// P[i] = valor do ponto de controle na posição i
// dx   = x[i+1] - x[i]
// dy   = y[i+1] - y[i]
// b[0] = y[i]
// b[1] = t[i]
// b[2] = (3*(dy/dx) - 2*t[i] - t[i+1])/dx
// b[3] = (t[i] + t[i+1] - 2*(dy/dx))/(dx^2)

// calcula os dois pontos extras nas extremidades
// x4-x2 = x3-x1 = x2-x0 conforme Akima
x[1] = x[2] + x[3] - x[4];
x[0] = x[1] + x[2] - x[3];

// m3-m2 = m2-m1 = m1-m0 conforme Akima
y[1] = y[2] - (2*((y[3]-y[2])/(x[3]-x[2]))
     - ((y[4]-y[3])/(x[4]-x[3]))) * (x[2]-x[1]);
y[0] = y[1] - (2*((y[2]-y[1])/(x[2]-x[1]))
     - ((y[3]-y[2])/(x[3]-x[2]))) * (x[1]-x[0]);

// x4-x2 = x3-x1 = x2-x0 conforme Akima
x[m-2] = x[m-3] + x[m-4] - x[m-5];
x[m-1] = x[m-2] + x[m-3] - x[m-4];

// m3-m2 = m2-m1 = m1-m0 conforme Akima
y[m-2] = y[m-3] + (2*((y[m-3]-y[m-4])/(x[m-3]-x[m-4]))
       - ((y[m-4]-y[m-5])/(x[m-4]-x[m-5]))) * (x[m-2]-x[m-3]);
y[m-1] = y[m-2] + (2*((y[m-2]-y[m-3])/(x[m-2]-x[m-3]))
       - ((y[m-3]-y[m-4])/(x[m-3]-x[m-4]))) * (x[m-1]-x[m-2]);

// calcula as derivadas nos pontos (2 <= i <= (m-3))
cAkima(m,x,y,t);

// calcula os valores iniciais de X e Y
x1 = x[2];
y1 = y[2];

for (i=2; i<(m-3); i++)
{
   // calcula os valores da mistura
   dx = x[i+1] - x[i];
   dy = y[i+1] - y[i];
```

```
            b[0] = y[i];
            b[1] = t[i];
            b[2] = (3*(dy/dx) - 2*t[i] - t[i+1])/dx;
            b[3] = (t[i] + t[i+1] - 2*(dy/dx))/(dx*dx);

            // não-uniformemente espaçada entre os nós
            for (j=x[i]; j<(x[i+1]); j++)
            {
               // calcula os valores de X e Y
               X = j - x[i];
               Y = b[0] + X*(b[1] + X*(b[2] + X*b[3]));
               x2 = j;
               y2 = int(Y);

               // liga os pontos para desenhar a curva resultante
               Dsg.WindowLine(x1+50,y1+50,x2+50,y2+50,R2_COPYPEN);
               Dsg.BitmapLine(x1+50,y1+50,x2+50,y2+50,R2_COPYPEN);
               x1 = x2;
               y1 = y2;
            }
         }

         // calcula os valores finais de X e Y
         x2 = x[m-3];
         y2 = y[m-3];

         // liga o último ponto
         Dsg.WindowLine(x1+50, y1+50, x2+50, y2+50, R2_COPYPEN);
         Dsg.BitmapLine(x1+50, y1+50, x2+50, y2+50, R2_COPYPEN);

         // libera a memória alocada
         if(x) free(x);
         if(y) free(y);
         if(t) free(t);
      }
      Dsg.SetSolidPen(0,0,0);
      SetCursor (LoadCursor (NULL, IDC_ARROW));
}

void TMainWindow::cAkima (int NPoints, float *x, float *y, float *t)
{
      // Calcula as derivadas conforme Akima
      //
      // Entrada:
      //
      //    Npoints = número de pontos a serem interpolados
      //    x,y     = 'arrays' com as coordenadas dos pontos
      //    t       = 'array' para armazenamento das derivadas de primeira ordem
      //
      // Saída:
      //
      //    'array ' t preenchida

      int     i;
      float   w1, w2, *m;

      // aloca memória
      if((m=(float*)malloc(NPoints*sizeof(float))) == NULL)
      {
```

```
      if(m) free(m);
      MessageBox(HWindow,"Memory error!","",MB_OK);
      abort(); // aborta o programa
      return;
   }
   else
   {
      // cálculo das declividades das retas
      for (i=0; i<(NPoints-1); i++)
         m[i] = (y[i+1]-y[i])/(x[i+1]-x[i]);

      // cálculo das derivadas de primeira ordem
      for (i=2; i<=(NPoints-3); i++)
      {
         w1 = fabs(m[i+1]-m[i]);
         w2 = fabs(m[i-1]-m[i-2]);
         if ((w1&&w2)==0)
         {
            w1 = 1;
            w2 = 1;
         }

         // primeira derivada
         t[i] = (w1*m[i-1] + w2*m[i])/(w1+w2);
      }

      // libera a memória alocada
      if(m) free(m);
   }
}
```

APÊNDICE O

LISTAGEM DA INTERPOLAÇÃO AKIMA PARA AJUSTE DE CURVAS SUAVES SOBRE FUNÇÕES DE VALORES MÚLTIPLOS

```
void TMainWindow::AkimaP (void)
{
   // Interpolação Akima Parametrizada para Curvas
   DesignOriented  Dsg( DisplayWindow->HWindow, wScrollOffsetX,
                        wScrollOffsetY, BitmapHandle,
                        wImageWidthInPixels, wImageHeightInPixels);
   PointSet        P;
   int             x1, y1, x2, y2, n, m, i;
   float           Y, X, dx, dy, r, u, delta, b[4], c[4],
                   *x, *y, *tcos, *tsin;

   SetCursor (LoadCursor (NULL, IDC_WAIT));
   Dsg.SetSolidPen(150,150,150);

   n = Points.SizeOf();
   m = n+4;
   if (n<3)
      return;

   // aloca memória
   if((x=(float*)malloc(m*sizeof(float))) == NULL ||
      (y=(float*)malloc(m*sizeof(float))) == NULL ||
      (tcos=(float*)malloc(m*sizeof(float))) == NULL ||
      (tsin=(float*)malloc(m*sizeof(float))) == NULL)
   {
      if(x) free(x);
      if(y) free(y);
      if(tcos) free(tcos);
      if(tsin) free(tsin);
      MessageBox(HWindow,"Memory error!","",MB_OK);
      abort(); // aborta o programa
      return;
   }
   else
   {
      // passa os valores da cadeia de 'containers' para 'buffers'
      // deixando espaço para mais dois pontos nas extremidades
      i=2;
      Points.Pointer = Points.First();
      while (Points.Pointer != Points.End())
      {
         x[i] = Points.ReadX(Points.Pointer)-50;
         y[i] = 350-Points.ReadY(Points.Pointer);
         Points.Pointer = Points.Next(Points.Pointer);
         i++;
      }
```

```
// Equação:
//
// para cada segmento de curva i, onde 2 <= i <= m-3 (m-4 segmentos):
//
// X = b[0] + u*(b[1] + u*(b[2] + u*b[3]));
// Y = c[0] + u*(c[1] + u*(c[2] + u*c[3]));
//
// onde:
//
// m    = número de pontos + 4, onde os dois de cada
//        extremidade são calculados pelo método de Akima
// dx   = x[i+1] - x[i];
// dy   = y[i+1] - y[i];
// r    = sqrt(dx*dx + dy*dy);
// b[0] = x[i];
// b[1] = r*tcos[i];
// b[2] = 3*dx - r*(tcos[i+1] + tcos[i] + tcos[i]);
// b[3] = -2*dx + r*(tcos[i+1] +tcos[i]);
// c[0] = y[i];
// c[1] = r*tsin[i];
// c[2] = 3*dy - r*(tsin[i+1] + tsin[i] + tsin[i]);
// c[3] = -2*dy + r*(tsin[i+1] +tsin[i]);
// tcos = cosseno do vetor da derivada
// tsin = seno do vetor da derivada
// u    = variação paramétrica de 0 até 1

// calcula os dois pontos extras nas extremidades
// (x4-x3)-(x3-x2) = (x3-x2)-(x2-x1) = (x2-x1)-(x1-x0)
x[1] = x[4] - 3*x[3] + 3*x[2];
x[0] = x[3] - 3*x[2] + 3*x[1];
x[m-2] = 3*x[m-3] - 3*x[m-4] + x[m-5];
x[m-1] = 3*x[m-2] - 3*x[m-3] + x[m-4];

// (y4-y3)-(y3-y2) = (y3-y2)-(y2-y1) = (y2-y1)-(y1-y0)
y[1] = y[4] - 3*y[3] + 3*y[2];
y[0] = y[3] - 3*y[2] + 3*y[1];
y[m-2] = 3*y[m-3] - 3*y[m-4] + y[m-5];
y[m-1] = 3*y[m-2] - 3*y[m-3] + y[m-4];

// calcula os senos e cossenos nos pontos (2 <= i <= (m-3))
cAkimaP(m,x,y,tcos,tsin);

// varre os segmentos de curva parametricamente
for (i=2; i<(m-3); i++)
{
   // calcula os valores da mistura
   dx = x[i+1] - x[i];
   dy = y[i+1] - y[i];
   r  = sqrt(dx*dx + dy*dy);

   b[0] = x[i];
   b[1] = r*tcos[i];
   b[2] = 3*dx - r*(tcos[i+1] + tcos[i] + tcos[i]);
   b[3] = dx - b[1] - b[2];

   c[0] = y[i];
   c[1] = r*tsin[i];
   c[2] = 3*dy - r*(tsin[i+1] + tsin[i] + tsin[i]);
   c[3] = dy - c[1] - c[2];
```

```
        // calcula variação paramétrica
        if ((fabs(dx))>(fabs(dy)))
           delta = (fabs(dx));
        else
           delta = (fabs(dy));
        if (delta != 0)
           delta = 1/delta;
        else
           delta = 0.005;

        // varia a função paramétrica de 0 até 1
        x1 = x[i];
        y1 = y[i];
        for (u=delta; u<=1; u+=delta)
        {
           // calcula os valores de X e Y
           X = b[0] + u*(b[1] + u*(b[2] + u*b[3]));
           Y = c[0] + u*(c[1] + u*(c[2] + u*c[3]));
           x2 = int(X);
           y2 = int(Y);

           // liga os pontos para desenhar a curva resultante
           Dsg.WindowLine(x1+50,y1+50,x2+50,y2+50,R2_COPYPEN);
           Dsg.BitmapLine(x1+50,y1+50,x2+50,y2+50,R2_COPYPEN);
           x1 = x2;
           y1 = y2;
        }
     }

     // libera a memória alocada
     if(x) free(x);
     if(y) free(y);
     if(tcos) free(tcos);
     if(tsin) free(tsin);
  }
  Dsg.SetSolidPen(0,0,0);
  SetCursor (LoadCursor (NULL, IDC_ARROW));
}

void TMainWindow::cAkimaP  ( int NPoints,float *x,float *y,float *tcos,
                             float *tsin)
{
  // Calcula os valores dos senos e cossenos conforme Akima
  //
  // Entrada:
  //
  //   Npoints = número de pontos a serem interpolados
  //   x,y     = 'arrays' com as coordenadas dos pontos
  //   tcos    = 'array' para armazenamento os valores de cosseno
  //   tsin    = 'array' para armazenamento os valores de seno
  //
  // Saída:
  //
  //   'arrays ' preenchidas

  int        i;
  float      w1, w2, ra, rb, r, temp, *a, *b;
```

```
   // aloca memória
   if((a=(float*)malloc(NPoints*sizeof(float))) == NULL ||
      (b=(float*)malloc(NPoints*sizeof(float))) == NULL)
   {
      if(a) free(a);
      if(b) free(b);
      MessageBox(HWindow,"Memory error!","",MB_OK);
      abort(); // aborta o programa
      return;
   }
   else
   {
      // cálculo dos valores de a e b
      for (i=0; i<(NPoints-1); i++)
      {
         a[i] = x[i+1]-x[i];
         b[i] = y[i+1]-y[i];
      }
      // cálculo dos valores de seno e cosseno
      for (i=2; i<=(NPoints-3); i++)
      {
         w1 = fabs(a[i]*b[i+1] - a[i+1]*b[i]);
         w2 = fabs(a[i-2]*b[i-1] - a[i-1]*b[i-2]);
         if ((w1&&w2)==0)
         {
            w1 = sqrt(a[i]*a[i] + b[i]*b[i]);
            w2 = sqrt(a[i-1]*a[i-1] + b[i-1]*b[i-1]);
         }
         ra = w1*a[i-1] + w2*a[i];
         rb = w1*b[i-1] + w2*b[i];
         temp = ra*ra + rb*rb;
         if (temp != 0)
         {
            r = sqrt(temp);
            tcos[i] = ra/r;
            tsin[i] = rb/r;
         }
         else
         {
            tcos[i] = ra;
            tsin[i] = rb;
         }
      }
      // libera memória alocada
      if(a) free(a);
      if(b) free(b);
   }
}
```

APÊNDICE P

LISTAGEM DA INTERPOLAÇÃO AKIMA PARA AJUSTE DE CURVAS SUAVES SOBRE FUNÇÕES DE VALORES MÚLTIPLOS PARAMETRICAMENTE CONFORME A CURVA DE HERMITE E COM MÉTODO PROPOSTO PARA CÁLCULO DOS MÓDULOS DAS TANGENTES

```
void TMainWindow::AkimaH (void)
{
   // Interpolação Akima Parametrizada Conforme Hermite para Curvas
   DesignOriented  Dsg( DisplayWindow->HWindow, wScrollOffsetX,
                        wScrollOffsetY, BitmapHandle,
                        wImageWidthInPixels, wImageHeightInPixels);
   PointSet        P;
   int             x1, y1, x2, y2, n, m, i;
   float           Y, X, dx, dy, u, delta, b[4], temp,
                   *x, *y, *tcos, *tsin, cos0, cos1, r0, r1, rv;

   SetCursor (LoadCursor (NULL, IDC_WAIT));
   Dsg.SetSolidPen(150,150,50);

   n = Points.SizeOf();
   m = n+4;
   if (n<3)
      return;

   // aloca memória
   if((x=(float*)malloc(m*sizeof(float))) == NULL ||
      (y=(float*)malloc(m*sizeof(float))) == NULL ||
      (tcos=(float*)malloc(m*sizeof(float))) == NULL ||
      (tsin=(float*)malloc(m*sizeof(float))) == NULL)
   {
      if(x) free(x);
      if(y) free(y);
      if(tcos) free(tcos);
      if(tsin) free(tsin);
      MessageBox(HWindow,"Memory error!","",MB_OK);
      abort(); // aborta o programa
      return;
   }
   else
   {
      // passa os valores da cadeia de 'containers' para 'buffers'
      // deixando espaço para mais dois pontos nas extremidades
      i=2;
      Points.Pointer = Points.First();
      while (Points.Pointer != Points.End())
      {
         x[i] = Points.ReadX(Points.Pointer)-50;
```

```
   y[i] = 350-Points.ReadY(Points.Pointer);
   Points.Pointer = Points.Next(Points.Pointer);
   i++;
}

// Equação:
//
// para cada segmento de curva i, onde 2 <= i <= m-3 (m-4 segmentos):
//
// X = b[0]*x[i] + b[1]*x[i+1] + b[2]*r0*tcos[i] + b[3]*r1*tcos[i+1];
// Y = b[0]*y[i] + b[1]*y[i+1] + b[2]*r0*tsin[i] + b[3]*r1*tsin[i+1];
//
// onde:
//
// m    = número de pontos + 4, onde os dois de cada
//        extremidade são calculados pelo método de Akima
// b[0] = 2*u^3  - 3*u^2  + 1;
// b[1] = -2*u^3 + 3*u^2;
// b[2] = u^3 - 2*u^2 + u;
// b[3] = u^3 - u^2;
// tcos = cosseno do vetor da tangente
// tsin = seno do vetor da tangente
// r0   = módulo do vetor da tangente no ponto P0
// r1   = módulo do vetor da tangente no ponto P1
// u    = variação paramétrica de 0 até 1

// calcula os dois pontos extras nas extremidades
// (x4-x3)-(x3-x2) = (x3-x2)-(x2-x1) = (x2-x1)-(x1-x0)
x[1] = x[4] - 3*x[3] + 3*x[2];
x[0] = x[3] - 3*x[2] + 3*x[1];
x[m-2] = 3*x[m-3] - 3*x[m-4] + x[m-5];
x[m-1] = 3*x[m-2] - 3*x[m-3] + x[m-4];

// (y4-y3)-(y3-y2) = (y3-y2)-(y2-y1) = (y2-y1)-(y1-y0)
y[1] = y[4] - 3*y[3] + 3*y[2];
y[0] = y[3] - 3*y[2] + 3*y[1];
y[m-2] = 3*y[m-3] - 3*y[m-4] + y[m-5];
y[m-1] = 3*y[m-2] - 3*y[m-3] + y[m-4];

// calcula os senos e cossenos nos pontos (2 <= i <= (m-3))
cAkimaP(m,x,y,tcos,tsin);

// varre os segmentos de curva parametricamente
for (i=2; i<(m-3); i++)
{
   // cálculo do cosseno do ângulo formado pelos vetores:
   //   - tangente t: (tcos,tsin)
   //   - entre os dois pontos v: (x[i+1]-x[i],y[i+1]-y[i])
   dx = x[i+1] - x[i];
   dy = y[i+1] - y[i];
   temp = dx*dx + dy*dy;
   if (temp != 0)
   {
      rv = sqrt(temp);
      cos0 = (tcos[i]*dx + tsin[i]*dy)/rv;
      cos1 = (tcos[i+1]*dx + tsin[i+1]*dy)/rv;
      r0 = fabs(rv*cos0);
      r1 = fabs(rv*cos1);
   }
```

```
        else
        {
           r0 = 0;
           r1 = 0;
        }

        // calcula a variação paramétrica
        if ((fabs(dx))>(fabs(dy)))
           delta = (fabs(dx));
        else
           delta = (fabs(dy));
        if (delta != 0)
           delta = 1/delta;
        else
           delta = 0.005;

        // varia a função paramétrica de 0 até 1
        x1 = x[i];
        y1 = y[i];

        for (u=delta; u<=1; u+=delta)
        {
           // calcula os valores de mistura conforme Hermite
           cHermite(b,u);

           // calcula os valores de X e Y
           X = b[0]*x[i] + b[1]*x[i+1] + b[2]*r0*tcos[i] + b[3]*r1*tcos[i+1];
           Y = b[0]*y[i] + b[1]*y[i+1] + b[2]*r0*tsin[i] + b[3]*r1*tsin[i+1];
           x2 = int(X);
           y2 = int(Y);

           // liga os pontos para desenhar a curva resultante
           Dsg.WindowLine(x1+50,y1+50,x2+50,y2+50,R1_COPYPEN);
           Dsg.BitmapLine(x1+50,y1+50,x2+50,y2+50,R1_COPYPEN);
           x1 = x2;
           y1 = y2;
        }
     }

     // libera a memória alocada
     if(x) free(x);
     if(y) free(y);
     if(tcos) free(tcos);
     if(tsin) free(tsin);
  }
  Dsg.SetSolidPen(0,0,0);
  SetCursor (LoadCursor (NULL, IDC_ARROW));
}

void TMainWindow::cAkimaP  ( int NPoints,float *x,float *y,float *tcos,
                             float *tsin)
{
  // Calcula os valores dos senos e cossenos conforme Akima
  //
  // Entrada:
  //
  //   Npoints = número de pontos a serem interpolados
  //   x,y     = 'arrays' com as coordenadas dos pontos
  //   tcos    = 'array' para armazenamento os valores de cosseno
```

```
//    tsin    = 'array' para armazenamento os valores de seno
//
// Saída:
//
//    'arrays ' preenchidas

int        i;
float      w1, w2, ra, rb, r, temp, *a, *b;

// aloca memória
if((a=(float*)malloc(NPoints*sizeof(float))) == NULL ||
   (b=(float*)malloc(NPoints*sizeof(float))) == NULL)
{
   if(a) free(a);
   if(b) free(b);
   MessageBox(HWindow,"Memory error!","",MB_OK);
   abort(); // aborta o programa
   return;
}
else
{
   // cálculo dos valores de a e b
   for (i=0; i<(NPoints-1); i++)
   {
      a[i] = x[i+1]-x[i];
      b[i] = y[i+1]-y[i];
   }

   // cálculo dos valores de seno e cosseno
   for (i=2; i<=(NPoints-3); i++)
   {
      w1 = fabs(a[i]*b[i+1] - a[i+1]*b[i]);
      w2 = fabs(a[i-2]*b[i-1] - a[i-1]*b[i-2]);
      if ((w1&&w2)==0)
      {
         w1 = sqrt(a[i]*a[i] + b[i]*b[i]);
         w2 = sqrt(a[i-1]*a[i-1] + b[i-1]*b[i-1]);
      }
      ra = w1*a[i-1] + w2*a[i];
      rb = w1*b[i-1] + w2*b[i];
      temp = ra*ra + rb*rb;

      if (temp != 0)
      {
         r = sqrt(temp);
         tcos[i] = ra/r;
         tsin[i] = rb/r;
      }
      else
      {
         tcos[i] = ra;
         tsin[i] = rb;
      }
   }

   // libera a memória alocada
   if(a) free(a);
   if(b) free(b);
}
```

```
}

void TMainWindow::cHermite (float *b, float u)
{
   // calcula os valores da mistura conforme Hermite
   float   u2, u3;

   if ((u<0)||(u>1))
   {
      MessageBox(HWindow,"Hermite parameter error!","",MB_OK);
      abort(); // aborta o programa
   }
   else
   {
      u2 = u*u;
      u3 = u2*u;
      b[0] = 2*u3  - 3*u2  + 1;
      b[1] = -2*u3 + 3*u2;
      b[2] = u3 - 2*u2 + u;
      b[3] = u3 - u2;
   }
   return;
}
```

APÊNDICE Q

LISTAGEM DA INTERPOLAÇÃO AKIMA PARA AJUSTE DE SUPERFÍCIES SUAVES BASEADA EM PROCEDIMENTOS LOCAIS

```
void TMainWindow::Akima3D (void)
{
   // interpolação Akima para Ajuste de Superfícies Suaves
   POINT3D    *aG, p00, p11;
   float      *mx, *mz, *mxz, *tx, *tz, *txz,
              mx00, mz00, mxz00,
              tx00, tx01, tx10, tx11,
              tz00, tz01, tz10, tz11,
              txz00, txz01, txz10, txz11,
              b00, b01, b02, b03,
              b10, b11, b12, b13,
              b20, b21, b22, b23,
              b30, b31, b32, b33,
              Q0, Q1, Q2, Q3,
              A, B, C, D, E, DX, DZ, tx00dz, tx10dz, tz00dx, tz10dx,
              X, Y, Z, deltaX, deltaZ, dx2, dz2;
   int        i, j, k ,l, NX, NZ, dx, dz,
              Xmax, Xmin, Zmax, Zmin, calc;

   SetCursor (LoadCursor (NULL, IDC_WAIT));
   NX = iGX+4;
   NZ = iGZ+4;

   // aloca memória
   if((aG=(POINT3D*)malloc(NX*NZ*sizeof(POINT3D))) == NULL||
      (mx=(float*)malloc(NX*NZ*sizeof(float))) == NULL||
      (mz=(float*)malloc(NX*NZ*sizeof(float))) == NULL||
      (mxz=(float*)malloc(NX*NZ*sizeof(float))) == NULL||
      (tx=(float*)malloc(NX*NZ*sizeof(float))) == NULL||
      (tz=(float*)malloc(NX*NZ*sizeof(float))) == NULL||
      (txz=(float*)malloc(NX*NZ*sizeof(float))) == NULL)
   {
      if(aG) free(aG);
      if(mx) free(mx);
      if(mz) free(mz);
      if(mxz) free(mxz);
      if(tx) free(tx);
      if(tz) free(tz);
      if(txz) free(txz);
      MessageBox(HWindow,"Memory error!","",MB_OK);
      abort(); // aborta o programa
      return;
   }

   // passa os valores da grade regular para uma nova grade
   // deixando espaço para mais dois pontos nas extremidades
   for (i=0; i<iGX; i++)
```

```
   for (j=0; j<iGZ; j++)
     aG[(i+2)*NZ+(j+2)] = aGrid[i*iGZ+j];

// Equação:
//
// para cada retalho de área:
//
//    ponto00 ----------------------------ponto10
//       |                                   |
//       |                                   |
//       |                                   |
//       |                                   |
//    ponto01 ----------------------------ponto11
//
//
// Q0 = b00 + DZ*(b01 + DZ*(b02 + DZ*b03))
// Q1 = b10 + DZ*(b11 + DZ*(b12 + DZ*b13))
// Q2 = b20 + DZ*(b21 + DZ*(b22 + DZ*b23))
// Q3 = b30 + DZ*(b31 + DZ*(b32 + DZ*b33))
//
// Y(X,Z) = Q0 + DX*(Q1 + DX*(Q2 + DX*Q3))
//
// onde:
//
// dx    = p11.x - p00.x
// dz    = p11.z - p00.z
// dx2   = (float)dx*dx
// dz2   = (float)dz*dz
//
// tx00dz = (tx01 - tx00)/dz
// tx10dz = (tx11 - tx10)/dz
// tz00dx = (tz10 - tz00)/dx
// tz10dx = (tz11 - tz01)/dx
//
// A = mxz00 - tx00dz - tz00dx + txz00
// B = tx10dz - tx00dz - txz10 + txz00
// C = tz10dx - tz00dx - txz01 + txz00
// D = txz11 - txz10 - txz01 + txz00
// E = A + A - B - C
//
// b00 = p00.y
// b01 = tz00
// b02 = (2*(mz00 - tz00) + mz00 - tz01)/dz
// b03 = (-2*mz00 + tz01 + tz00)/dz2
//
// b10 = tx00
// b11 = txz00
// b12 = (2*(tx00dz - txz00) + tx00dz - txz01)/dz
// b13 = (-2*tx00dz + txz01 + txz00)/dz2
//
// b20 = (2*(mx00 - tx00) + mx00 - tx10)/dx
// b21 = (2*(tz00dx - txz00) + tz00dx - txz10)/dx
// b22 = (3*(A + E) + D)/(dx*dz)
// b23 = (-3*E - B - D)/(dx*dz2)
//
// b30 = (-2*mx00 + tx10 + tx00)/dx2
// b31 = (-2*tz00dx + txz10 + txz00)/dx2
// b32 = (-3*E - C - D)/(dz*dx2)
// b33 = (D + E + E)/(dx2*dz2)
```

```
//
// DX = X - p00.x
// DZ = Z - p00.z
//
// de maneira que:
//
// mx[i,j]    = valor da 1.diferença dividida no ponto i,j em x
// mz[i,j]    = valor da 1.diferença dividida no ponto i,j em z
// mxz[i,j]   = valor da 2.diferença dividida no ponto em x e z
// tx[i,j]    = valor da 1.derivada no ponto i,j em x
// tz[i,j]    = valor da 1.derivada no ponto i,j em z
// txz[i,j]   = valor da 2.derivada no ponto i,j em x e z
// P[i,j]     = valor(x,y,z) do ponto na posição i,j
// b00..b33   = coeficientes para cálculo da equação do retalho

// calcula os valores das bordas
Border3D (aG,NX,NZ);

// calcula as derivadas
Derivate (aG,NX,NZ,mx,mz,mxz,tx,tz,txz);

// calcula variação da nova grade
Xmax = aGrid[(iGX-1)*iGZ].x;
Xmin = aGrid[0].x;
Zmax = aGrid[iGZ-1].z;
Zmin = aGrid[0].z;

deltaX = (((float)Xmax-Xmin)/(iPX-1));
deltaZ = (((float)Zmax-Zmin)/(iPZ-1));

i = 2;
for (k=0; k<iPX; k++)
{
   X = Xmin + k*deltaX;
   j = 2;
   calc = 1; // indica necessidade de cálculo

   // caso ultrapasse o retalho em x
   while(X>(float)(aG[(i+1)*NZ+j].x))
   {
      i++;
      calc = 1;
   }

   for (l=0; l<iPZ; l++)
   {
      Z = Zmin + l*deltaZ;

      // caso ultrapasse o retalho em z
      while(Z>(float)(aG[i*NZ+(j+1)].z))
      {
         j++;
         calc = 1;
      }

      // calcula os valores para o retalho
      if (calc == 1)
      {
         // calcula os dados do retalho
```

```
// pontos
p00 = aG[i*NZ+j];
p11 = aG[(i+1)*NZ+(j+1)];
// diferenças divididas
mx00 = mx[i*NZ+j];
mz00 = mz[i*NZ+j];
mxz00 = mxz[i*NZ+j];
// primeira derivada em x
tx00 = tx[i*NZ+j];
tx01 = tx[i*NZ+(j+1)];
tx10 = tx[(i+1)*NZ+j];
tx11 = tx[(i+1)*NZ+(j+1)];
// primeira derivada em z
tz00 = tz[i*NZ+j];
tz01 = tz[i*NZ+(j+1)];
tz10 = tz[(i+1)*NZ+j];
tz11 = tz[(i+1)*NZ+(j+1)];
// segunda derivada em x e z
txz00 = txz[i*NZ+j];
txz01 = txz[i*NZ+(j+1)];
txz10 = txz[(i+1)*NZ+j];
txz11 = txz[(i+1)*NZ+(j+1)];

// calcula os valores dos coeficientes
dx = p11.x - p00.x;
dz = p11.z - p00.z;
dx2 = (float)dx*dx;
dz2 = (float)dz*dz;

tx00dz = (tx01 - tx00)/dz;
tx10dz = (tx11 - tx10)/dz;
tz00dx = (tz10 - tz00)/dx;
tz10dx = (tz11 - tz01)/dx;

A = mxz00 - tx00dz - tz00dx + txz00;
B = tx10dz - tx00dz - txz10 + txz00;
C = tz10dx - tz00dx - txz01 + txz00;
D = txz11 - txz10 - txz01 + txz00;
E = A + A - B - C;

b00 = p00.y;
b01 = tz00;
b02 = (2*(mz00 - tz00) + mz00 - tz01)/dz;
b03 = (-2*mz00 + tz01 + tz00)/dz2;

b10 = tx00;
b11 = txz00;
b12 = (2*(tx00dz - txz00) + tx00dz - txz01)/dz;
b13 = (-2*tx00dz + txz01 + txz00)/dz2;

b20 = (2*(mx00 - tx00) + mx00 - tx10)/dx;
b21 = (2*(tz00dx - txz00) + tz00dx - txz10)/dx;
b22 = (3*(A + E) + D)/(dx*dz);
b23 = (-3*E - B - D)/(dx*dz2);

b30 = (-2*mx00 + tx10 + tx00)/dx2;
b31 = (-2*tz00dx + txz10 + txz00)/dx2;
b32 = (-3*E - C - D)/(dz*dx2);
b33 = (D + E + E)/(dx2*dz2);
```

```
            // resseta a necessidade de cálculo
            calc = 0;
         }

         // calcula o polinômio na posição (X,Z) onde Y(X,Z)
         DX = X - p00.x;
         DZ = Z - p00.z;

         Q0 = b00 + DZ*(b01 + DZ*(b02 + DZ*b03));
         Q1 = b10 + DZ*(b11 + DZ*(b12 + DZ*b13));
         Q2 = b20 + DZ*(b21 + DZ*(b22 + DZ*b23));
         Q3 = b30 + DZ*(b31 + DZ*(b32 + DZ*b33));

         Y = Q0 + DX*(Q1 + DX*(Q2 + DX*Q3));

         // armazena os valores calculados
         aPlane[k*iPZ+l].x = (int)X;
         aPlane[k*iPZ+l].z = (int)Z;
         aPlane[k*iPZ+l].y = (int)Y;
      }
   }

   // roda a animação
   Animation (aPlane,iPX,iPZ,-PI/16,-PI/16,PI/32);

   // libera a memória alocada anteriormente
   if(aG) free(aG);
   if(mx) free(mx);
   if(mz) free(mz);
   if(mxz) free(mxz);
   if(tx) free(tx);
   if(tz) free(tz);
   if(txz) free(txz);
   SetCursor (LoadCursor (NULL, IDC_ARROW));
}

void TMainWindow::Border3D (POINT3D *aG, int NX, int NZ)
{
   // Calcula os valores das bordas de uma grade regular conforme Akima
   //
   // Entrada:
   //
   //    aG    =  'array' de pontos (x,y,z) da grade regular
   //    NX    =  número de pontos no eixo X da grade regular
   //    NZ    =  número de pontos no eixo Z da grade regular
   //

   // Saída:
   //
   //    'array' com as bordas preenchidas conforme Akima

   int        i, j;
   float      x0, x1, x2, x3, x4,
              y0, y1, y2, y3, y4,
              z0, z1, z2, z3, z4;

   // calcula os dois pontos extras nas extremidades do eixo x
   for (j=2; j<(NZ-2); j++)
   {
```

```
   // x4-x2 = x3-x1 = x2-x0 conforme Akima
   x2 = aG[2*NZ+j].x;
   x3 = aG[3*NZ+j].x;
   x4 = aG[4*NZ+j].x;
   x1 = x2 + x3 - x4;
   x0 = x1 + x2 - x3;
   aG[NZ+j].x = x1;
   aG[j].x = x0;

   // m3-m2 = m2-m1 = m1-m0 conforme Akima
   y2 = aG[2*NZ+j].y;
   y3 = aG[3*NZ+j].y;
   y4 = aG[4*NZ+j].y;
   y1 = y2 - (2*((y3-y2)/(x3-x2)) - ((y4-y3)/(x4-x3))) * (x2-x1);
   y0 = y1 - (2*((y2-y1)/(x2-x1)) - ((y3-y2)/(x3-x2))) * (x1-x0);
   aG[NZ+j].y = y1;
   aG[j].y = y0;

   // valores de z
   aG[NZ+j].z = aG[2*NZ+j].z;
   aG[j].z = aG[2*NZ+j].z;

   // x4-x2 = x3-x1 = x2-x0 conforme Akima
   x2 = aG[(NX-3)*NZ+j].x;
   x3 = aG[(NX-4)*NZ+j].x;
   x4 = aG[(NX-5)*NZ+j].x;
   x1 = x2 + x3 - x4;
   x0 = x1 + x2 - x3;
   aG[(NX-2)*NZ+j].x = x1;
   aG[(NX-1)*NZ+j].x = x0;

   // m3-m2 = m2-m1 = m1-m0 conforme Akima
   y2 = aG[(NX-3)*NZ+j].y;
   y3 = aG[(NX-4)*NZ+j].y;
   y4 = aG[(NX-5)*NZ+j].y;
   y1 = y2 + (2*((y2-y3)/(x2-x3)) - ((y3-y4)/(x3-x4))) * (x1-x2);
   y0 = y1 + (2*((y1-y2)/(x1-x2)) - ((y2-y3)/(x2-x3))) * (x0-x1);
   aG[(NX-2)*NZ+j].y = y1;
   aG[(NX-1)*NZ+j].y = y0;

   // valores de z
   aG[(NX-2)*NZ+j].z = aG[(NX-3)*NZ+j].z;
   aG[(NX-1)*NZ+j].z = aG[(NX-3)*NZ+j].z;
}

// calcula os dois pontos extras nas extremidades do eixo Z
for (i=0; i<NX; i++)
{
   // x4-x2 = x3-x1 = x2-x0 conforme Akima
   z2 = aG[i*NZ+2].z;
   z3 = aG[i*NZ+3].z;
   z4 = aG[i*NZ+4].z;
   z1 = z2 + z3 - z4;
   z0 = z1 + z2 - z3;
   aG[i*NZ+1].z = z1;
   aG[i*NZ+0].z = z0;

   // m3-m2 = m2-m1 = m1-m0 conforme Akima
   y2 = aG[i*NZ+2].y;
```

```
   y3 = aG[i*NZ+3].y;
   y4 = aG[i*NZ+4].y;
   y1 = y2 - (2*((y3-y2)/(z3-z2)) - ((y4-y3)/(z4-z3))) * (z2-z1);
   y0 = y1 - (2*((y2-y1)/(z2-z1)) - ((y3-y2)/(z3-z2))) * (z1-z0);
   aG[i*NZ+1].y = y1;
   aG[i*NZ+0].y = y0;

   // valores de x
   aG[i*NZ+1].x = aG[i*NZ+2].x;
   aG[i*NZ+0].x = aG[i*NZ+2].x;

   // x4-x2 = x3-x1 = x2-x0 conforme Akima
   z2 = aG[i*NZ+(NX-3)].z;
   z3 = aG[i*NZ+(NX-4)].z;
   z4 = aG[i*NZ+(NX-5)].z;
   z1 = z2 + z3 - z4;
   z0 = z1 + z2 - z3;
   aG[i*NZ+(NX-2)].z = z1;
   aG[i*NZ+(NX-1)].z = z0;

   // m3-m2 = m2-m1 = m1-m0 conforme Akima
   y2 = aG[i*NZ+(NX-3)].y;
   y3 = aG[i*NZ+(NX-4)].y;
   y4 = aG[i*NZ+(NX-5)].y;
   y1 = y2 + (2*((y2-y3)/(z2-z3)) - ((y3-y4)/(z3-z4))) * (z1-z2);
   y0 = y1 + (2*((y1-y2)/(z1-z2)) - ((y2-y3)/(z2-z3))) * (z0-z1);
   aG[i*NZ+(NX-2)].y = y1;
   aG[i*NZ+(NX-1)].y = y0;

   // valores de x
   aG[i*NZ+(NX-2)].x = aG[i*NZ+(NX-3)].x;
   aG[i*NZ+(NX-1)].x = aG[i*NZ+(NX-3)].x;
}

// calcula os valores das diagonais
j = 0;
while (j<NZ)
{
   // x4-x2 = x3-x1 = x2-x0 conforme Akima
   x2 = aG[2*NZ+j].x;
   x3 = aG[3*NZ+j].x;
   x4 = aG[4*NZ+j].x;
   x1 = x2 + x3 - x4;
   x0 = x1 + x2 - x3;

   // m3-m2 = m2-m1 = m1-m0 conforme Akima
   y2 = aG[2*NZ+j].y;
   y3 = aG[3*NZ+j].y;
   y4 = aG[4*NZ+j].y;
   y1 = y2 - (2*((y3-y2)/(x3-x2)) - ((y4-y3)/(x4-x3))) * (x2-x1);
   y0 = y1 - (2*((y2-y1)/(x2-x1)) - ((y3-y2)/(x3-x2))) * (x1-x0);

   // média
   aG[NZ+j].y = (aG[NZ+j].y + y1)/2;
   aG[j].y = (aG[j].y + y0)/2;

   // x4-x2 = x3-x1 = x2-x0 conforme Akima
   x2 = aG[(NX-3)*NZ+j].x;
   x3 = aG[(NX-4)*NZ+j].x;
```

```
      x4 = aG[(NX-5)*NZ+j].x;
      x1 = x2 + x3 - x4;
      x0 = x1 + x2 - x3;

      // m3-m2 = m2-m1 = m1-m0 conforme Akima
      y2 = aG[(NX-3)*NZ+j].y;
      y3 = aG[(NX-4)*NZ+j].y;
      y4 = aG[(NX-5)*NZ+j].y;
      y1 = y2 + (2*((y2-y3)/(x2-x3)) - ((y3-y4)/(x3-x4))) * (x1-x2);
      y0 = y1 + (2*((y1-y2)/(x1-x2)) - ((y2-y3)/(x2-x3))) * (x0-x1);

      // média
      aG[(NX-2)*NZ+j].y = (aG[(NX-2)*NZ+j].y + y1)/2;
      aG[(NX-1)*NZ+j].y = (aG[(NX-1)*NZ+j].y + y0)/2;
      j++;
      if (j==2)
         j = NZ-2;
   }
   return;
}

void TMainWindow::Derivate (POINT3D *aG, int NX, int NZ, float *mx,
                     float *mz, float *mxz, float *tx, float *tz, float *txz)
{
   // Calcula os valores das derivadas de uma grade regular
   //
   // Entrada:
   //
   //    aG    =  'array' de pontos (x,y,z) da grade regular
   //    NX    =  número de pontos no eixo X da grade regular
   //    NZ    =  número de pontos no eixo Z da grade regular
   //    mx    =  'array' para armaz. das 1.diferença dividida em x
   //    mz    =  'array' para armaz. das 1.diferença dividida em z
   //    mxz   =  'array' para armaz. das 2.dif. dividida em x e z
   //    tx    =  'array' para armaz. das 1.derivadas em relação a x
   //    tz    =  'array' para armaz. das 1.derivadas em relação a z
   //    txz   =  'array' para armaz. das 2.der. em relação a x e a z
   //
   // Saída:
   //
   //    'arrays' preenchidas conforme Akima
   //
   // Declividade:
   //
   //                              Yi+1,j - Yij
   //                   mx[ij] = --------------
   //                               Xi+1 - Xi
   //
   //                             Yi,j+1 - Yij
   //                   mz[ij] = --------------
   //                               Zi+1 - Zi

   int        i, j;
   float      wx1, wx2, wz1, wz2;

   // cálculo das declividades das retas
   // diferença dividida de primeira ordem
   for(i=0; i<(NX-1); i++)
      for(j=0; j<(NZ-1); j++)
```

```
      {
         mx[i*NZ+j] = (float)(aG[(i+1)*NZ+j].y - aG[i*NZ+j].y)/
                      (float)(aG[(i+1)*NZ+j].x - aG[i*NZ+j].x);
         mz[i*NZ+j] = (float)(aG[i*NZ+(j+1)].y - aG[i*NZ+j].y)/
                      (float)(aG[i*NZ+(j+1)].z - aG[i*NZ+j].z);
      }

   // diferença dividida de segunda ordem
   for(i=0; i<(NX-2); i++)
      for(j=0; j<(NZ-2); j++)
      {
         mxz[i*NZ+j] = ( (float)(mx[i*NZ+(j+1)] - mx[i*NZ+j])/
                         (float)(aG[i*NZ+(j+1)].z - aG[i*NZ+j].z));
      }

   // cálculo das derivadas de primeira e de segunda ordem
   for (i=2; i<=(NX-3); i++)
      for (j=2; j<=(NZ-3); j++)
      {
         wx1 = fabs(mx[(i+1)*NZ+j]-mx[i*NZ+j]);
         wx2 = fabs(mx[(i-1)*NZ+j]-mx[(i-2)*NZ+j]);
         wz1 = fabs(mz[i*NZ+(j+1)]-mz[i*NZ+j]);
         wz2 = fabs(mz[i*NZ+(j-1)]-mz[i*NZ+(j-2)]);
         if ((wx1&&wx2)==0)
         {
            wx1 = 1;
            wx2 = 1;
         }

         if ((wz1&&wz2)==0)
         {
            wz1 = 1;
            wz2 = 1;
         }

         // primeira derivada
         tx[i*NZ+j] = (wx1*mx[(i-1)*NZ+j] + wx2*mx[i*NZ+j])/(wx1+wx2);
         tz[i*NZ+j] = (wz1*mz[i*NZ+(j-1)] + wz2*mz[i*NZ+j])/(wz1+wz2);

         // segunda derivada
         txz[i*NZ+j]=(wx1*(wz1*mxz[(i-1)*NZ+(j-1)] + wz2*mxz[(i-1)*NZ+j])
                     + wx2*(wz1*mxz[i*NZ+(j-1)] + wz2*mxz[i*NZ+j]))
                     / ((wx1+wx2)*(wz1+wz2));
      }
}
```