APROXIMAÇÃO E INTERPOLAÇÃO EM NORMAS 1, 2 E INFINITA

- UMA RESENHA -

GREICY MARA FRANÇA

GREICY MARA FRANÇA

APROXIMAÇÃO E INTERPOLAÇÃO EM NORMAS 1, 2 E INFINITA

- UMA RESENHA -

Dissertação apresentada ao curso de MESTRADO EM SISTEMA E COMPUTAÇÃO da Universidade Federal da Paraíba, em cumprimento aos requisitos necessários para a obtenção do grau de MESTRE EM SISTEMAS E COMPUTAÇÃO (M. Sc.).

MARIO TOYOTARO HATTORI

ORIENTADOR

Campina Grande - PB
Dezembro - 1988

DIGITALIZAÇÃO:

SISTEMOTECA - UFCG

APROXIMAÇÃO E INTERPOLAÇÃO EM NORMAS 1, 2 E INFINITA

- UMA RESENHA -

GREICY MARA FRANÇA

Dissertação Aprovada em 15 de dezembro de 1988

MARIO TOYOTARO HATTORI - M.Sc.

PRESIDENTE

BRUNO CORREIA DA NOBREGA QUEIROZ - M.Sc.

EXAMINADOR

CRESO SANTOS DA ROCHA - PHD.

EXAMINADOR

Campina Grande - Pb
Dezembro - 1988

PENSAMENTO

" A mais bela coragem é a confiança que devemos ter na capacidade do nosso esforço "

Coelho Neto

DEDICATÓRIA

Aos meus pais **FRANÇA** e **MIECO**,

que me deram a vida,

a minha irmã **KATIA**.

e ao meu amor **CARLOS ALBERTO**.

AGRADECIMENTOS

A minha família,

que foi presença na minha vida mas respeitou profundamente a minha maneira de Amor - a sua presença, a sua companhia, seu sorriso a sua palavra, e mesmo a sua ausência, foi expressão de amor profundo. Quando eu sofri, ela sofreu comigo e, sem me dizeres nada fizeram-me entender que o amor é irmão gêmeo da dor, e experimentei a alegria íntima de uma renúncia, alegria que jamais havia experimentado.

Ao professor Hattori,

que foi mais que um orientador, minha gratidão pelo carinho, amparo e orientação que me dispensou, fatores decisivos na concretização dos meus ideais.

RESUMO

Neste trabalho apresentamos um estudo e implementação de algoritmos para solução do problema linear em norma 1 e infinita e um estudo de ajustamento de curvas e superfícies.

Um dos campos da matemática onde o computador tem dado grande contribuição e o de aproximação de superfícies. Ela tem permitido determinar dentre uma grande família de possíveis escolhas a melhor superfície aproximante. Como um estudo de todos os algoritmos de aproximação nas normas 1, 2 e infinita seria inviável, optamos por apresentar um estudo computacional de alguns algoritmos sobre aproximação em normas 1 e infinita e uma pesquisa bibliográfica sobre aproximação em norma 2 (o progresso tem sido principalmente em norma 2) e problemas não lineares.

Entre os algoritmos estudados foram implementados :

- o algoritmos de Barrodale e Roberts (aproximação linear l_1 discreta);

- o algoritmo de Abdelmalek (solução l_1 de sistemas hiperdeterminados de equações lineares);

- o algoritmo de Duris (interpolação e aproximação suave discreta por splines) e

- o algoritmo de Akima (interpolação e aproximação de superfícies suaves para pontos distribuídos irregularmente).

ABSTRACT

In this thesis we describe a study and implementation of algorithms for linear problem of solution in 1 norms and infinite, study of curve fitting and surfaces.

One of mathematics's areas where the computer has had grand contribution is the surface fitting. It has assisted to establish inside of a big family of possibles choises, the better approximate surface. How a study of all the approximation's algorithms in 1 norms and infinite would be impracticable, we decide to exhibit a computational study some algorithms about approximation in 1 norms and infinite and a bibliography research about approximation in 2 norm (the development have been principally in 2 norm) and non linear problem.

Among the algorithms we emphasize:

- the Barrodale and Robert's algorithm (constrained l_1 Linear Approximation);

- the Abdelmalek's algorithm (L1 Solution of Overdetermined Systems of Linear);

- the Duris's algorithm (Discrete Cubic Spline Interpolation and Smoothing) and

- the Akima's algorithm (Bivariate Interpolation and Smooth Surface Fitting for Irregulary Distributed).

SUMÁRIO

CAPÍTULO III

Aproximação em uma variável - Problemas lineares

CAPÍTULO IV

Aproximação de superfícies

CAPÍTULO V

Algoritmos para problemas não lineares (PNL)

CAPÍTULO VI

Implementação de algoritmos, teste e aplicações

CAPÍTULO I

INTRODUÇÃO

1.1 - Apresentação

A teoria de aproximação é uma disciplina (área de ensino e de pesquisa) bem estabelecida na matemática e permeia na análise numérica nos tópicos mais importantes para ciência e tecnologia. Uma das mais espetaculares utilizações da análise numérica nos últimos anos foi a análise do método dos elementos finitos, inventado pelos engenheiros, aplicável na solução de problemas de contorno em equações diferenciais parciais.

A utilização de diferentes normas nos problemas de aproximação conduz a problemas numéricos e resultados teóricos diferentes.

Este trabalho é uma resenha de algoritmos sobre aproximação em normas 1 e infinita.

Algoritmos que resolvem o problema

$$\min \quad ||Ax - b||$$

sujeito ou não a restrições em diferentes normas foram estudados, adaptados e implementados. Este problema pode ser um problema linear ou um problema não linear, conforme A seja um operador linear ou não.

Nas inúmeras referências ([12], [13], [14], [15]) sobre interpolação e aproximação, verificamos que a maioria dos problemas são tratados apenas em uma dimensão.

Um espaço unidimensional é geometricamente mais simples do que espaços n-dimensionais; a teoria de aproximação e interpolação para uma dimensão é muito mais simples, mais conhecida e de maior uso.

O problema de interpolação consiste em: dado um conjunto $\{(x_i, f_i)\}$, $i = 0, \ldots, n$, encontrar um único polinômio $p_n(x)$ em P_n, um espaço linear, que satisfaça

$$p_n(x_i) = f(x_i),\ i = 0,\ldots,n,$$

sendo que este polinômio poderá ter a forma de Newton, Lagrange, fatorada ou a forma clássica

$$p_n(x) = a_0 + a_1 x + \ldots + a_n x^n.$$

Ao invés de interpolar, podemos aproximar uma curva a este mesmo conjunto de dados. Dados $\{(x_i, f_i)\}$, $i = 0, \ldots, n$, podemos aproximar uma função $f(x)$ pela combinação linear

$$\tilde{f}(x_i) = \sum_{j=0}^{n} c_j \varphi_j(x_i), \quad i = 0, \ldots, n$$

de funções mais simples $\varphi_i(x)$. Os coeficientes c_j são constantes que serão determinadas pelas condições de aproximação impostas. Se $||f - \tilde{f}||$ em alguma norma $||\ ..\ ||$ for pequeno, então $\tilde{f}$ será uma boa aproximação de f.

Para problemas em n dimensões a teoria é mais complicada e os algoritmos são mais difíceis. No caso de várias variáveis, aproximações suaves serão muito mais complicadas computacionalmente.

Especificamente, sejam $p_0, p_1, \ldots, p_n$, $n+1$ pontos distintos de Ω (conjunto de pontos num plano real, Ω conhecido como uma malha randômica), e $y_0, y_1, \ldots, y_n$, $n+1$ números reais (veja figura 1.1). O problema, então, é achar um polinômio $P(x,y)$ de grau n em duas variáveis x e y que resolve o problema de interpolação

$$P(p_i) = f(p_i), \quad p_i = (x_i, y_j) \quad 0 \leq i \leq n \ ,$$

onde um polinômio de grau n em x e y é uma função algébrica da forma

$$P(x,y) = \sum_{i,j=0}^{n} a_{ij} x^i y^i . \qquad (1.1)$$

Este problema tem uma solução (cf. Prenter [14]) e uma forma simples para escrever (1.1) seria:

$$P(x,y) = \sum_{i=0}^{n} f(p_i, l_i(p)) = \sum_{i=0}^{n} f(x_i, y_i) l_i(x,y)$$

onde os l_i's são polinômios de grau n facilmente computáveis resolvendo

$$l_i(p_j) = \delta_{i,j} \ , \quad 0 \leq i \leq n \ .$$

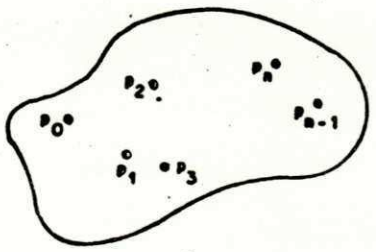

Figura 1.1 - malha randômica

A aproximação de funções tem sido construída sobre malhas retangulares e triangulares. Para a aproximação bidimensional sobre malhas triangulares e irregulares, não há algoritmos simples para construir aproximações por partes em $C^1[f(x,y)]$, $C^2[f(x,y)]$, e em conjuntos de funções mais suaves.

No caso de aproximações, há grande dificuldade em testar os poucos algoritmos existentes pois o espaço de memória disponível pode se tornar insuficiente em computadores comuns.

Devido a dimensão do assunto, nos restringimos ao estudo do problema linear.

1.2 - Importância do Trabalho

A Universidade Federal da Paraíba (UFPb) dispõe de uma Biblioteca de Algoritmos Numéricos (BITAN) implantada em 1985 e que vem sendo atualizada continuamente desde então. Os tópicos constantes da biblioteca são os seguintes:

1. Álgebra linear,
2. Equações e sistemas não lineares,
3. Integração Numérica,
4. Aproximação e interpolação,
5. Equações diferenciais ordinárias,
6. Otimização,
7. Ordenação,
8. Números aleatórios e
9. Gráficos pela impressora de linha.

Como é de interesse da UFPb difundir a BITAN como também atualizá-la, direcionamos o nosso trabalho para algoritmos que realizem aproximações em normas 1 e infinita para expandir o tópico 4 da BITAN - Aproximação e interpolação.

1.3 - Visão Global do Trabalho

Inicialmente, no segundo capítulo, apresentamos um resumo dos fundamentos da teoria de aproximação e as consequências de se trabalhar em diferentes espaços de funções.

No terceiro capítulo apresentamos a descrição de aproximação em uma variável para problemas lineares.

Apresentamos no quarto capítulo uma abordagem sobre aproximação de superfícies.

Uma breve abordagem sobre algoritmos para problemas não lineares é apresentada no capítulo cinco.

A descrição dos algoritmos implementados, testes realizados e aplicações são apresentados no sexto capítulo.

Finalmente no sétimo capítulo, concluimos fazendo comentários sobre as dificuldades encontradas para a realização do trabalho, escolha e implementação dos algoritmos e apresentamos sugestões para estudos posteriores.

CAPÍTULO II

FUNDAMENTOS DE TEORIA DE APROXIMAÇÃO

Suponhamos que nos tenham sido fornecidas certas informações sobre uma função, por exemplo, seus valores, ou os valores de suas derivadas em certos pontos. É possível reconstruir a função completamente a partir dos pontos dados e em caso afirmativo como podemos reconstruí-la ? Uma resposta parcial é dada pelo teorema da interpolação: Dados n + 1 pontos do plano real $(x_0,f_0),(x_1,f_1)\ldots,(x_n,f_n)$ existe um único polinômio $p_n(x)$ em P_n, um espaço linear, para o qual

$$p_n(x_i) = f(x_i) \quad , \quad i = 0,1,\ldots,n \ .$$

Assim se $f(x)$ for um polinômio de grau n, conseguimos reconstruí-la a partir dos pontos dados.

Podemos, também, ajustar uma curva a este mesmo conjunto de dados. Dados $n + 1$ números reais $x_0, x_1, \ldots, x_n$ onde $f_i = f(x_i)$, procuramos $\tilde{f}(x)$ impondo algumas condições para garantir a existência e unicidade da solução. Existe um número infinito de curvas que passam por $\{(x_i, f_i) \mid 0 \leq i \leq n\}$.

Como encontrar $f(x)$ exatamente é, na maiora das vezes, impossível. O problema então se transforma em outro: quais as funções $\tilde{f}(x)$ são boas aproximações dessa função ? Uma das técnicas de aproximação mais usadas é a de aproximação de $f(x)$ por uma combinação linear da forma

$$\tilde{f}(x) = c_1\varphi_1(x) + c_2\varphi_2(x) + \ldots + c_n\varphi_n(x)$$

de funções mais simples $\varphi_i(x)$ cujas propriedades são bem conhecidas e são relativamente simples de avaliar. Os coeficientes c_k são constantes que serão determinados pelas restrições que serão impostas a $\tilde{f}(x)$.

$\tilde{f}$ será uma boa aproximação de f se $||f - \tilde{f}||$ em alguma norma $||\ .\ ||$ for pequena. O problema é escolher as constantes c_i na definição de $f(x)$ que tornem

$$||f - (c_1 p_1 + \ldots + c_n p_n)||$$

pequena e que force uma solução única para esses c_i's. Existe um grande número de escolhas entre as restrições que podem ser impostas a $\tilde{f}(x)$: restrições interpoladoras, restrições puramente variacionais, mistura de restrições de interpolação e de suavização e restrições de ortogonalidade.

Precisamos, então saber em que condições se pode garantir a existência e unicidade dessa aproximação.

Problemas típicos em aproximação envolvem a seleção de um conjunto de elementos que constituem um subconjunto cuja distância a um elemento é em algum sentido o mínimo da distância de um elemento não pertencente ao conjunto. Assim, o dado conjunto pode consistir de polinômios enquanto o elemento a ser aproximado é uma função, como arc sin x, que seguramente não é um polinômio. Por outro lado, um problema de aproximação não se torna precisamente definido até decidirmos como a distância entre dois elementos será medida.

2.1 - Melhor aproximação: existência em espaços métricos

Seja o par (X,d) onde X é um conjunto e d uma função real definida para pares de pontos em X e que d satisfaz os seguintes postulados para todo x, y, z em X:

(i) $d(x,x) = 0$

(ii) $d(x,y) > 0$, se $x \neq y$

(iii) $d(x,y) = d(y,x)$

(iv) $d(x,y) \leq d(x,z) + d(z,y)$;

então dizemos que X é um espaço métrico.

Com o conceito de um espaço métrico podemos formular um problema básico da teoria de aproximação: Dado um ponto g e um conjunto M em um espaço métrico podemos determinar um ponto de M que tenha distância mínima de g?

Teorema 2.1.1 (Prenter [14]): Seja K um conjunto compacto num espaço métrico X. Para cada ponto P em X existe um ponto A em K cuja distância a P é mínima.

Isso mostra que, para o problema acima, em espaços métricos podemos determinar o ponto procurado. Mas, e em outros espaços também é possível?

2.2 - Melhor aproximação: existência em espaços lineares

Um espaço linear real X é um conjunto de objetos abstratos, chamados vetores, objetos esses que podem ser adicionados de acordo com as regras usuais da aritmética e podem ser multiplicados por números reais, também de acordo com as regras da aritmética. Mais especificamente, um espaço

linear real deve satisfazer os axiomas a seguir:

Para todo x,y,z em X e a,b em R

(i) a x em X,

(ii) x + y em X,

(iii) x + y = y + x,

(iv) (x + y) + z = x + (y + z),

(v) 1·x = x,

(vi) (a + b)x = ax + bx,

(vii) a(x + y) = ax + ay,

(viii) (ab)x = a(bx),

(ix) existe e em X tal que x + e = x e

(x) para todo x, existe -x tal que -x está em X e -x + x = e.

Se todas as propriedades acima são satisfeitas quando a multiplicação for por números complexos, o espaço será complexo.

Espaço linear normado

Realmente estamos interessados em mais do que espaços lineares. Queremos espaços lineares nos quais possamos associar um conceito de tamanho ||f|| para cada vetor f em um espaço linear X. Tais espaços lineares são conhecidos como espaços lineares normados (essas estruturas descrevem

propriedades da melhor aproximação) e o número real $||f||$ é chamado de norma de f. Uma norma é uma função real definida num espaço linear X, tendo as seguintes propriedades:

Para todo a em R e todo f e g em X

(i) $|| f || > 0$ e $|| f || = 0$ se $f = 0$

(ii) $|| a f || = | a | \cdot || f ||$

(iii) $|| f + g || \leq || f || + || g ||$

Definição 2.2.1: Sejam X um espaço normado com norma $|| \ ||$, M um subconjunto de X e p um ponto qualquer. Um ponto y_o em M é chamado de melhor aproximação de p se

$$|| p - y_o || \leq || p - y || \quad \text{para todo } y \text{ em } M$$

A melhor aproximação pode não existir e se existir, pode não ser única. É importante saber em que condições existe e é única a melhor aproximação.

Teorema 2.2.1 (Prenter [14]): Seja X um espaço normado com norma $|| \ ||$ e seja X_N um sub-espaço de X de dimensão finita. Então cada ponto x de X tem uma melhor aproximação x_N em X_N que satisfaz

$$|| x - x_N || = \min_{y \text{ em } X_N} || x - y ||$$

2.3 - Melhor aproximação: existência e unicidade em espaços de produto interno

É desejável introduzir num espaço linear X a operação de produto interno. O produto interno de 2 vetores f e g é um número real denotado por $\langle f, g \rangle$ ou (f,g). Em um espaço $C[a,b]$ de funções contínuas sobre [a, b] podemos definir

$$\langle f, g \rangle = \int_b^a f(x)\, g(x)\, dx .$$

Um espaço linear que contenha um produto interno é denominado espaço de produto interno, e a equação

$$|| f || = + \sqrt{\langle f, f \rangle}$$

define uma norma.

Para demonstrar a unicidade da melhor aproximação em espaços de produto interno precisamos dos seguintes resultados:

defina $||x|| = (x,x)^{1/2}$. Então

(a) $(x, \lambda x) = \lambda (x, x)$

(b) $||x + y|| = ||x|| + ||y|| \leftrightarrow (x,y) = ||x|| + ||y||$.

Veja demonstração em Prenter [14].

Teorema 2.3.1 (Prenter [14]): A melhor aproximação de um sub-espaço fechado de dimensão finita de um espaço de produto interno é única.

Seja X um espaço de produto interno e seja M um sub-espaço de dimensão finita de X. Seja x em X e suponha que x_o e y_o em M são duas melhores aproximações de x_o. Então

$$|| x - x_o|| = || x - y_o || = d. \text{ Se } d = 0 \longrightarrow x_o = y_o = x.$$

Suponha que $d > 0$.

Pela desigualdade do triângulo

$$|| \frac{1}{2}(x_o + y_o) - x || < \frac{1}{2} || x - x_o || + \frac{1}{2} || x - y_o || = d$$

Portanto, $\frac{1}{2}(x_o + y_o)$ é também a melhor aproximação, porque $\frac{1}{2}(x_o + y_o)$ em M e satisfaz

$$|| \frac{1}{2}(x_o + y_o) - x|| = \frac{1}{2} ||x - x_o|| + \frac{1}{2}||x - y_o||$$

Então $x - y_o = \lambda(x - x_o)$ se $\lambda = 1$. Suponha que $\lambda \neq 1$, então

$$x = \frac{y_o - x_o}{1 - \lambda}$$

e x em M, contrariando a hipótese inicial. Portanto, $\lambda = 1$.

2.4 - Melhor aproximação: existência e unicidade em Espaços de Hilbert

Um espaço de produto interno completo constitui um espaço de Hilbert.

O espaço de Hilbert é uma generalização natural para dimensões infinitas de espaços euclidianos reais ou complexos R^n e C^n. Há muitas vantagens em se trabalhar num espaço de Hilbert entre as quais:

(i) Os teoremas e processos podem ser vistos geometricamente bem como analiticamente pois nossa intuição espacial é em 1, 2 e 3 dimensões;

(ii) Todos os espaços de Hilbert são abstratamente equivalentes entre si.

A melhor aproximação de subespaços de espaços de Hilbert é conhecida como ajustamento de quadrados mínimos.

2.5 - Considerações Sobre as Conseqüências de se Trabalhar em Diferentes Espaços de Funções

Nas seções anteriores vimos em que condições a melhor aproximação existe em diferentes espaços. A melhor aproximação é caracterizada por uma distância mínima entre dois pontos, distância essa que precisa ser medida de alguma forma, isto é, precisa-se de uma métrica. Uma vez escolhida a métrica para medir distâncias resta resolver um problema de minimização de funções sem ou com restrições. Em geral precisamos escolher espaços de função que resultem em problemas de minimização que possam ser resolvidos com eficiência pelo computador e reflitam de modo fiel os objetivos almejados. Os espaços normados e os de Hilbert permitem formular problemas de aproximação que refletem situações que ocorrem frequentemente na vida real, como minimizar o desperdício, minimizar o tamanho de retalhos numa confecção ou filtrar dados espúrios. Em qualquer um desses espaços o problema a ser resolvido envolve minimização de alguma norma.

O problema geral tem a forma: encontre x^* que minimize $||b - Ax||$, em que A é um operador, b é um vetor de m componentes e x é um vetor de n componentes. Se a norma escolhida for 1, para obter x^* precisamos minimizar

$$\sum_{i=1}^{m} | r_i |$$

em que r_i é um componente do vetor resíduo $b - Ax$ e podemos impor alguma restrição como $x > 0$ e $Bx > c$. Se A e B forem lineares tem-se um problema de programação linear; se não houver restrições o problema é mais complicado (v. seção 3.2).

Se a norma escolhida for infinita, resulta um problema minimax

$$\min_{x} \max_{i} | r_i |.$$

Se a norma escolhida for definida por um produto interno é melhor minimizar

$$|| b - A x ||^2$$

e o problema será de quadrados mínimos, linear ou não linear conforme A seja linear ou não linear, podendo estar sujeito ou não a restrições.

Os dois capítulos seguintes apresentam os algoritmos de minimização de funções.

CAPÍTULO III

APROXIMAÇÃO EM UMA VARIÁVEL - PROBLEMAS LINEARES

Neste capítulo descrevemos algoritmos para resolver o problema

$$\min ||b - Ax||$$

sujeito ou não a restrições, em que A é uma matriz m por n, b e x são vetores de m e n componentes, respectivamente, quando a norma $||\cdot||$ escolhida for 1, 2 e infinita.

3.1 - Programação linear

Ao optar pela norma l_1 para resolver o problema

$$\min_{x} || b - Ax ||$$

com restrições, chega-se a um problema de programação linear conforme vimos na seção 2.5.

O método para resolução de um problema de programação linear mais conhecido é o simplex, que é um procedimento algébrico e iterativo que fornece a solução exata (na ausência de arredondamento) de qualquer problema de programação linear (PPL) em um número finito de iterações. É também capaz de indicar se um problema tem solução ilimitada, se não tem solução ou se possui infinitas soluções.

Essas características do simplex permitiram sua codificação em programas extremamente eficientes possibilitando a solução de sistemas com centenas de variáveis. Extensões posteriores, como o simplex revisado e o princípio da decomposição, aumentaram sua capacidade para dezenas de milhares e finalmente centenas de milhares de variáveis [12]. Além disso, o simplex possui não apenas uma interpretação geométrica mas também uma interpretação econômica.

Um problema de programação linear geral tem a seguinte forma:

$$\text{minimize } c_1x_1 + c_2x_2 + \ldots + c_nx_n = c^Tx \qquad (3.1.1)$$

com $x \geq 0$ e que satisfazem restrições da forma:

$$a_{i1}x_1 + a_{i2}x_2 + \ldots a_{in}x_n < b_i,\ i = 1, 2, \ldots, m_1, \qquad (3.1.2a)$$

$$a_{i1}x_1 + a_{i2}x_2 + \ldots a_{in}x_n = b_n,$$

$$i = m_1 + 1,\ m_1 + 2, \ldots, m \qquad (3.1.2b)$$

Os números c_k, a_{ik}, b_i são números reais dados. A função c^Tx a ser minimizada é chamada função objetivo. Cada x em R^n que satisfaz todas as condições é dita ser um ponto viável para o problema. Introduzindo variáveis e equações adicionais, o PPL (3.1.1) a (3.1.2) pode ser colocado numa forma em que as restrições são de igualdade ou de desigualdade elementar (da forma $x_i > 0$), quando a função objetivo c^Tx toma a forma $c^Tx = -x_p$. Para transformar um PPL para esta forma, devemos substituir cada desigualdade não elementar de (3.1.2a) por uma igualdade e uma desigualdade elementar usando uma variável de folga x_{n+i},

$$a_{i1}x_1 + \ldots + a_{in}x_n + x_{n+i} = b_i, \quad x_{n+i} \geq 0 \qquad (3.1.3)$$

Se a função objetivo $c_1x_1 + \ldots + c_nx_n$ não for elementar, introduzimos uma variável adicional x_{n+m_1+1} e incluímos uma equação de restrição adicional

$$c_1x_1 + \ldots + c_nx_n + x_{n+m_1+1} = 0 \qquad (3.1.4)$$

entre as restrições (3.1.2). A minimização de c^Tx é equivalente a maximização de x_{n+m_1+1} com este sistema estendido de restrições.

Assim podemos assumir que o PPL é sempre dado na seguinte forma padrão:

$$LP(I,p): \text{ minimize } -x_p \qquad (3.1.5)$$

para todo x em R^n com $Ax = b$ e $x_i \geq 0$ para i em I. Nessa formulação, $I = \{1,2,\ldots,n\}$, um subconjunto dos naturais, é um conjunto de índices, p é um índice fixo satisfazendo p em $N \setminus I$, $A = (a_1, a_2, \ldots, a_n)$ é uma matriz real m x n tendo colunas a_i, e b_i em R^n é um vetor dado. As variáveis x_i para as quais i em I são as variáveis com restrição enquanto que para i não em I são as variáveis livres. Por

$$P = \{x \text{ em } R^n \mid Ax = b \ \& \ x_i \geq 0 \text{ para todo } i \text{ em } I\} \quad (3.1.6)$$

denotamos o conjunto de todos os pontos viáveis de LP(I,p). x^* em P é um ponto ótimo de LP(I,p) se $x_p^* = \max \{ x_p | x_p \text{ em } P \}$.

Consideremos o sistema de equações lineares $Ax = b$ de LP(I,p). Para um vetor $J = (j_1, \ldots, j_r)$, j_i em N, $A_J = (a_{ji}, \ldots, a_{jr})$ denota a submatriz de A tendo colunas a_{ji}, x_J denota o vetor$(x_{ji}, \ldots, x_{jr})^T$. Por simplicidade denotamos o conjunto

$$\{j_i \mid i = 1, 2, \ldots, r\}$$

dos componentes de J por J e p em J se existir um i com $p = j_i$.

Um vetor $J = (j_1, \ldots, j_m)$ de m índices distintos j_i em N define uma base de $Ax = b$ (e de LP (I,p)) se A_j for não singular. A_j é também chamado uma base, as variáveis x_i para i em J são chamadas variáveis básicas, enquanto as variáveis restantes x_k, k não em J, são variáveis não básicas. Se $K = (k_1, \ldots, k_{n-m})$ for um vetor de índices contendo os índices não básicos, usaremos a notação $J \oplus K = N$.

Associado com alguma base J, $J \oplus K = N$, há uma única solução $\bar{x} = \bar{x}(J)$ de $Ax = b$, chamada solução básica, com $\bar{x}_k = 0$. Já que a solução de

$$A\bar{x} = A_j\bar{x}_j + A_k\bar{x}_k = A_j\bar{x}_j = b, \quad (3.1.7)$$

é dado por

$$\bar{x}_j = \bar{b},\ \bar{x}_k = 0 \quad \text{com } \bar{b} = A_j^{-1}b. \qquad (3.1.8)$$

Além disso, dada uma base J, cada solução x de $Ax = b$ é determinada unicamente por seus elementos x_k não básicos e pela solução x básica. Isso decorre da multiplicação de

$$Ax = A_j x_j + A_k x_k = b$$

por A_j e de (3.1.8):

$$x_j = \bar{b} - A_j^{-1} A_k x_k = \bar{x}_j - A_j^{-1} A_k x_k \qquad (3.1.9)$$

Se x_k em R^{n-m} for arbitrariamente próximo de x_j, e se x_j for definido por (3.1.9), então x é a solução de $Ax = b$. Por essa razão (3.1.9) provê uma parametrização do conjunto de soluções $\{x \mid Ax = b\}$ através dos componentes de x_k em R^{n-m}.

Se a solução básica $\bar{x}$ de $Ax = b$ associado com a base for um ponto de folga de LP(I,p), $\bar{x}$ em P, isto é, se $x_i \geq 0$ para todo i em I se e só se $x_i \geq 0$ para todo i em $I \subset J$, então J é uma base de folga de LP(I,p) e x é uma solução básica de folga. Finalmente uma base viável não é degenerada se $x_i > 0$ para todo i em $I \subset J$.

O PPL como um todo não é degenerado se todas as suas bases não forem degeneradas.

3.1.1 - A forma canônica

Um sistema canônico tem a seguinte forma:

$$x_1 + a_{1,m+1}x_{m+1} + \ldots + a_{1,n}x_n = b_1$$

$$x_2 + a_{2,m+1}x_{m+1} + \ldots + a_{2,n}x_n = b_2$$

$$x_m + a_{m,m+1}x_{m+1} + \ldots + a_{m,n}x_n = b_m$$

As variáveis básicas x_1, x_2, ..., x_m possuem coeficientes nulos em todas as equações, exceto em uma, em que o coeficiente é igual a 1. Existe uma variável básica diferente por equação. As variáveis x_{m+1}, ..., x_n são consideradas não básicas.

O fato de considerarmos que as m primeiras variáveis são básicas não afeta a generalidade, pois é sempre possível ordenar as colunas dessa maneira. Por outro lado, não é possível colocar um conjunto qualquer de m variáveis na base (isto é, colocar o sistema na forma canônica em relação a estas variáveis). Entretanto, se o sistema for não redundante (isto é,

se nenhuma equação puder ser escrita como combinação linear das outras), existe pelo menos um conjunto de variáveis que pode ser colocado na base.

3.1.2 - O algoritmo simplex

Sejam

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

e

$$Z = c(1)x_1 + c(2)x_2 + \ldots + c(n)x_n \quad e \quad \bar{c}(j) = -c(j)$$

Esquematicamente, podemos descrever o algoritmo simplex da seguinte forma:

Sejam $N = \{1, 2, 3, \ldots, n\}$ o conjunto dos índices das colunas da matriz A e $M = \{1, 2, 3, \ldots, m\}$ o conjunto dos índices das linhas de A; então:

Passo 1 - Entre com o problema e a solução básica viável inicial.

Passo 2 - Coloque o sistema na forma canônica em relação à solução básica

Passo 3 - Calcule $c(j^*) = Min(\bar{c}(j),\ j \text{ em } N)$

Passo 4 - Se $c(j^*) \geq 0$ então, encontramos a solução ótima. Pare.

Passo 5 - Calcule $j^* = Min(b_i / a^*_{ij}, a^*_{ij} > 0)$

Passo 6 - Se todos os $a^*_{ij} \leq 0$ então, a solução é ilimitada, pare.

Passo 7 - Faça o pivotamento em torno de a^*_{ij} e volte ao passo 3, ou seja, aplique o método de Gauss Jordan (Prenter[14]).

Resultados Importantes [12]

1 - O conjunto de soluções viáveis de um PPL é convexo.

2 - O número de soluções básicas é finito.

3 - Uma solução básica viável corresponde a um vértice do conjunto de soluções viáveis.

4 - Se existir uma solução viável, existe uma solução básica viável.

5 - Se existir uma solução ótima (onde uma solução ótima é uma solução básica viável quando os coeficientes $\bar{c}(j)$ conhecidos como custos relativos são todos positivos), existe pelo menos uma solução básica ótima.

Vejamos um exemplo utilizando o algoritmo simplex.

Seja minimizar $z = x_1 - 3x_2 + 2x_3$, sujeito a:

$$3x_1 - x_2 + 2x_3 \leq 7$$

$$-2x_1 + 4x_2 \leq 12$$

$$-4x_1 + 3x_2 + 8x_3 \leq 10$$

$$x_1 \geq 0,\ x_2 \geq 0,\ x_3 \geq 0.$$

Primeiramente, acrescentamos as variáveis de folga $x_4 \geq 0, x_5 \geq 0,\ x_6 \geq 0$; logo temos:

$$3x_1 - x_2 + 2x_3 + x_4 \leq 7$$

$$-2x_1 + 4x_2 + x_5 \leq 12$$

$$-4x_1 + 3x_2 + 8x_3 + x_6 \leq 10$$

e

$$-z + x_1 - 3x_2 + 2x_3 = 0.$$

A tabela 3.1.1 contém os dados iniciais para o problema e a tabela 3.1.2 apresenta os quadros simplex para o algoritmo descrito anteriormente. O pivô em cada quadro é colocado entre parênteses.

Tabela 3.1.1 - Dados iniciais

Quadro I

	x_1	x_2	x_3	x_4	x_5	x_6	b
	3	-1	2	1	0	0	7
	-2	4	0	0	1	0	12
	-4	3	(8)	0	0	1	10
-z	1	-3	2	0	0	0	0

Analisando o quadro I podemos verificar que o possível pivô se encontra na coluna onde o valor de -z é igual a 2. Calculando $c(j^*)$ encontramos o valor 5/4, portanto o pivô será igual a 8.

A partir destes resultados, para obtermos o quadro II aplicamos o método de Gauss Jordan em tórno do pivô que se resume nos seguintes passos:

Denominemos as linhas do quadro I por l_1, l_2, l_3 (linha do pivô) e l_4 respectivamente então façamos:

Passo 1 - $l_3 = l_3 / 8$;

Passo 2 - $l_1 = l_1 + (-2)l_3$;

Passo 3 - $l_4 = l_4 + (-2)l_3$.

Tabela 3.1.2 - Quadros simplex

Quadro II

	x_1	x_2	x_3	x_4	x_5	x_6	b
	(4)	$-\frac{7}{4}$	0	1	0	$-\frac{1}{4}$	$-\frac{18}{4}$
	-2	4	0	0	1	0	12
	$-\frac{1}{2}$	$\frac{3}{8}$	1	0	0	$\frac{1}{8}$	$\frac{5}{4}$
-z	2	$-\frac{15}{4}$	0	0	0	$-\frac{1}{4}$	$-\frac{10}{4}$

Quadro III

SOLUÇÃO

	x_1	x_2	x_3	x_4	x_5	x_6	b
	1	$-\frac{7}{16}$	0	$\frac{4}{16}$	0	$-\frac{1}{16}$	$\frac{18}{16}$
	0	$\frac{25}{8}$	0	$\frac{1}{2}$	1	$-\frac{1}{8}$	$\frac{57}{4}$
	0	$\frac{5}{32}$	1	$\frac{1}{8}$	0	$\frac{3}{32}$	$\frac{28}{16}$
-z	0	$-\frac{23}{8}$	0	$-\frac{4}{8}$	0	$-\frac{1}{8}$	$-\frac{28}{8}$

A solução ótima encontrada é:

$$x_1 = \frac{9}{8},\ x_2 = 0,\ x_3 = \frac{28}{16} \text{ e } z = \frac{19}{4}.$$

3.2 - Aproximação l_1 Linear Discreta e Solução l_1 de Equações Lineares Hiperdeterminadas.

Considere o sistema hiperdeterminado de equações lineares

$$Ax = b, \qquad (3.2.1)$$

onde A é uma matriz n por m, e b é um vetor de tamanho n. A solução l_1 de (3.2.1) é determinar o vetor x de tamanho m que minimize a função

$$R(x) = \sum_{i=1}^{n} |r_i|, \qquad (3.2.2)$$

onde

$$r_i = a_{i1}x_i + a_{i2}x_2 + \ldots + a_{im}x_m - b_i, \quad i = 1, \ldots, n \qquad (3.2.3)$$

O problema de programação linear

É mostrado por Wagner (cf. Abdelmalek [3]) que este problema pode ser reduzido a um problema de programação linear. A forma primal é

$$\min Z = \sum_{i=1}^{n} \varepsilon_{1i} + \sum_{i=1}^{n} \varepsilon_{2i}, \qquad (3.2.4a)$$

sujeito às restrições

$$(A\ I_n\ -\ I_n) \begin{pmatrix} x \\ \varepsilon_1 \\ \varepsilon_2 \end{pmatrix} = b, \qquad (3.2.4b)$$

os a_i sem restrições, $i = 1, \ldots, m$,

$$\varepsilon_{1i} \geq 0,\ \varepsilon_{2i} \geq 0,\ i = 1, \ldots, n \qquad (3.2.4c)$$

onde A, x e b são, respectivamente, a matriz (a_{ij}) e os vetores colunas x e b têm componentes x_i e b_i de (3.2.1). Um programa para resolver (3.2.4) usando o método simplex é dado por Barrodale & Roberts (cf. Abdelmalek [3]).

Contudo, para o dual, temos o seguinte problema

$$\max z = \sum_{i=1}^{n} b_i w_i, \qquad (3.2.5a)$$

sujeito às restrições

$$A^T w = 0, \qquad (3.2.5b)$$

$$-1 \leq w_i \leq 1, \ i = 1, \ldots, n, \qquad (3.2.5c)$$

onde o vetor $w = (w_1, w_2, \ldots, w_n)^T$ e A^T é a transposta de A de (3.2.1).

De qualquer maneira, fazendo $f_i = w_i + 1$ e $f = (f_1, \ldots, f_n)^T$, obtemos a formulação

$$\max z = \sum_{i=1}^{n} b_i (f_i - 1), \qquad (3.2.6a)$$

sujeito às restrições

$$A^T f = \begin{bmatrix} \sum_{i=1}^{n} a_{i1} \\ \sum_{i=1}^{n} a_{i2} \end{bmatrix} \qquad (3.2.6b)$$

$$0 \leq f_i \leq 2, \quad i = 1, \ldots, n. \qquad (3.2.6c)$$

Este é um problema de programação linear com variáveis positivas. Contudo, é mostrado por Hadley (cf. Abdelmalek [3]), que se um conjunto de regras simples for observado, o mesmo problema pode ser resolvido por um método simplex especial. A solução é baseada no seguinte teorema.

Teorema 3.2.1 (cf. Abdelmalek [3]) - Uma condição necessária e suficiente para o sistema (3.2.6a) ser ótimo é que os (n - m) elementos de f tenham valores nulos (limite inferior) ou 2 (limite superior), e que os outros m elementos sejam variáveis básicas.

Neste algoritmo construimos um quadro simplex para o problema (3.2.6a) como se os elementos de f não fossem limitados superiormente. Seja $k_i(A^T)$, $i = 1, \ldots, n$ a i-ésima coluna da matriz A^T. Deixe m de tais colunas formarem a base da matriz B e f_B a solução básica. Definimos o conjunto I(f) um subconjunto de (1, 2, ..., n) com a propriedade de que os vetores $(k_i(A^T \mid i \text{ em } I(f))$ formem uma base de R^n. Seja $f_{B_i} = (f_B)$, $i = 1, \ldots, m$, e sejam os conjuntos L(f) e U(f) indicadores das variáveis não básicas f_i que são respectivamente seus limites superior e inferior, isto é

$$L(f) = (i \text{ em } (1, 2, \ldots, n) \mid f_i = 0, \; i \text{ não pertencente a } I(f)),$$

e

$$U(f) = \{i \text{ em } \{1, 2, \ldots, n\} \mid f_i = 2, \ i \text{ não pertencente a } I(f)\}.$$

Então para algum $k_i(A^T)$, i não pertencente a I(f),

$$y_i = B^{-1} k_i(A^T) \qquad (3.2.7)$$

$$z_i = b_B^T y_i, \qquad (3.2.8)$$

onde os elementos de f_B são b_i, i em I(f). Assim

$$z_i = b_B^T B^{-1} k_i(A^T). \qquad (3.2.9)$$

Sejam

$$\Sigma_1 = \sum_{i \text{ em } U(f)}$$

$$\Sigma_2 = \sum_{i \text{ em } L(f)}$$

$$\Sigma_3 = \sum_{i \text{ em } I(f)}$$

Desde que alguma das variáveis não básicas sejam seus limites superiores (=2), de (3.2.6b)

$$f_B = B^{-1} \begin{bmatrix} \sum_{i=1}^{n} a_{i1} \\ \vdots \\ \sum_{i=1}^{n} a_{im} \end{bmatrix} - \sum_{1} 2y_i.$$

Fazendo $B^{-1} \begin{bmatrix} \sum_{i=1}^{n} a_{i1} \\ \vdots \\ \sum_{i=1}^{n} a_{im} \end{bmatrix} = f_{B_o}$ e por (3.2.7),

$$f_B = f_{B_o} - 2 \sum_{1} y_i. \qquad (3.2.11)$$

Também a função z em (3.2.6a) é dada por

$$z = \sum_{3} b_i (f_i - 1) - \sum_{2} b_i + \sum_{1} b_i \qquad (3.2.12)$$

O algoritmo é resumido como segue: Uma coluna não básica que pode substituir uma das colunas não básicas, pode variar de seu limite inferior até o seu limite superior ou vice-versa. A solução ótima é caracterizada pelo seguinte teorema.

Teorema 3.2.2(cf. Abdelmalek [3]) - Uma solução básica possível é máxima, se os parâmetros $(z_i - b_i)$, i não pertencente a I(f) satisfizerem as relações

$$z_i - b_i \geq 0, \ i \text{ em } L(f) \qquad (3.2.13a)$$

e

$$z_i - b_i \leq 0, \ i \text{ em } U(f). \qquad (3.2.13b)$$

O algoritmo simplex dual para o método simplex especial para resolver (3.2.6) é descrito a seguir.

O algoritmo simplex dual

Resolvemos (3.2.6) pela escolha de m colunas linearmente independentes de A^T para formar a base B. O quadro simplex é então formado avaliando (3.2.7 - 3.2.9), os vetores (y_i) e o conjunto (z_i, b_i). A limitação da solução é garantida pelo seguinte lema.

Lema 3.2.1 (cf. Abdelmalek [3]) - A solução ótima de (3.2.6) é limitada e é dada por

$$0 \leq z = Z \leq \sum_{i=1}^{n} |b_i|.$$

Os seguintes passos constituem um algoritmo simplex para o método descrito por Hadley (cf. Abdelmalek [3]). A escolha dos vetores que saem da base é feita de acordo com o método de Usow (v. Abdelmalek [3]). Para iniciar, uma variável não básica b_i tem valor zero.

Passo 1 - Para todo $z_i - b_i < 0$, i em L(f), seja $f_i = 2$ que indica a colocação de uma marca na coluna correspondente. Calcule f_B pela relação (3.2.11).

Passo 2 - Se todos os f_{B_l}, satisfizerem $0 \leq f_{B_l} \leq 2$, uma solução ótima foi encontrada, pare.

Passo 3 - Examine os f_{B_l} para $l = 1, 2, \ldots$. Considere o primeiro que for ou menor do que zero ou maior do que dois. Chame-os de f_{B_l}. A coluna correspondente na base será substituída por uma coluna não básica de acordo com os passos (3.1) - (3.4) abaixo. Se o novo f_{B_l} não for menor do que zero ou maior do que dois, esta iteração será repetida até que esta condição seja satisfeita. Na próxima iteração o exame dos f_{B_l} procederá de $f_{B_{i+1}}$ e voltará novamente para f_{B_1}. Suponha que para alguma iteração $k_j(A^T)$ associado com f_{B_i}, e $k_r(a^T)$ seja substituído por $K_j(A^T)$ na base. Consideramos os dois casos:

Caso 1 - Se $f_{B_i} \leq 0$, $k_r(A^T)$ é determinado de

$$\theta_r = \max(\theta_1, \theta_2) < 0,$$

onde

$$\theta_1 = (z_r - b_r) / y_{ir} = \max_s \{(z_s - b_s) / y_{is}\},$$

$$y_{is} < 0, \ s \text{ em } L(f)$$

e

$$\theta_2 = (z_r - b_r / y_{ir} = \max_s \{(z_s - b_s) / y_{is}\},$$

$$y_{is} > 0, \ s \text{ em } U(f).$$

Passo (3.1) - Se $\theta_r = \theta_1$, transforme o quadro simplex pelo modo usual e volte para o passo 2.

Passo (3.2) - Se $\theta_r = \theta_2$, transforme o quadro como usualmente e adicione 2 ao novo f_{B_i}. Remova a marca da coluna r indicando que b_i não é maior do que o seu limite superior. Volte ao passo 2.

Caso 2 - Se $f_{B_i} > 0$, $k_r(A^T)$ é determinado de

$$\tau_r = \min(\tau_1, \tau_2) > 0,$$

onde

$$\tau_1 = (z_r - b_r) / y_{ir} = \min_s \{(z_s - b_s) / y_{is}\},$$

$$y_{is} > 0, \ s \text{ em } L(f),$$

e

$$\tau_2 = (z_r - b_r) / y_{ir} = \min \{(z_s - b_s) / y_{is}\},$$

$$y_{is} < 0, \ s \text{ em } U(f).$$

Passo 3.3 - Se $\tau_r = \tau_1$, transforme o quadro como usualmente, marque a coluna $k_j(A^T)$ para indicar que f_j é seu limite superior. Subtraia $2y_i$ de f_B e volte ao passo 2.

Passo 3.4 - Se $\tau_r = \tau_2$, transforme o quadro como usualmente, remova a marca da coluna $k_r A(^T)$ e marque a coluna $k_j(A^T)$. Adicione 2 ao novo f_{B_i} e subtraia $2y_i$ de f_{B_i} e volte ao passo 2.

Este algoritmo (cf. Abdelmalek [3]) parece ser o mais eficiente comparado com outros métodos conhecidos. Além disso sua simplicidade e sua eficiência são devidas ao uso das técnicas do simplex dual. Variáveis artificiais não são necessárias e

como conseqüência o esforço computacional é grandemente reduzido. Se A for rank deficient, um máximo de m variáveis artificiais são necessárias para iniciar a iteração. Também neste caso o problema pode ser resolvido como um problema de duas fases.

Mostramos um exemplo de como o algoritmo é usado.

Dados

$$A = \begin{bmatrix} 1 & 1 \\ 1 & -1 \\ 1 & 2 \\ 2 & 4 \\ 2 & 1 \\ 3 & 1 \end{bmatrix}$$

e

$$b = \begin{bmatrix} 3 \\ 1 \\ 7 \\ 11.1 \\ 6.9 \\ 7.2 \end{bmatrix},$$

é requerido determinar x_1^* e x_2^* que minimizam a norma l_1 dos resíduos.

Tabela 3.2.1 - dados iniciais

Quadro I

	b ⟶	3	1	7	11.1	6.9	7.2
B^{-1}	$\sum_{j=1}^{6} k_j$	k_1	k_2	k_3	k_4	k_5	k_6
0.5 0.5	10	1	1	1	2	2	3
0.5 -0.5	8	1	-1	2	4	1	1

Tabela 3.2.2 - quadros simplex

Quadro II

b	3	1	7	11.1	6.9	7.2
f_1	k_1	k_2	k_3	k_4	k_5	k_6
-7	1	0	1.5	3	1.5	(2)
-1	0	1	-0.5	-1	0.5	1
z = 8.2	0	0	-3	-3.1	-1.9	-0.3

Quadro III

		b	3	1	7	11.1	6.9	7.2
f_B	B	f'_B	k_1	k_2	k_3	k_4	k_5	k_6
7.2	k_6	-1.5	0.5	0	0.75	1.5	0.75	1
1	k_4	4.5	-0.5	1	-1.25	(-2.5)	-0.25	0
		z = 7.5	0.1	0	-2.85	-2.8	-1.75	0

Quadro IV

		b	3	1	7	11.1	6.9	7.2
f_B	B	f'_B	k_1	k_2	k_3	k_4	k_5	k_6
7.2	k_6	0	0.2	0.6	0	0	0.6	1
1	k_4	1	0.2	-0.4	0.5	1	0.1	0
		z = 4.7	0.66	-1.12	-1.45	0	-1.47	0

A tabela 3.2.1 contém os dados iniciais para o problema e a tabela 3.2.2 apresenta os quadros simplex para o algoritmo descrito anteriormente. O pivô em cada quadro é colocado entre parênteses, por k_i subentende-se $k_i(A^T)$ e os elementos de f_B são b_i, i em I(f).

Na tabela 3.2.1, quadro I, nós tinhamos a escolha de colocar $k_6(A^T)$ ou $k_2(A^T)$ na base. O decremento de z em cada caso é 2.8. Foi escolhido $k_2(A^T)$, e isto é feito em relação ao passo 3.4 do algoritmo. Finalmente, no quadro IV, a solução ótima é encontrada, e a base final é formada pela quarta e sexta coluna. Assim, resolvendo a quarta e sexta equação desse exemplo, nós obtemos como solução $x_1^* = 1.77$, $x_2^* = 1.89$ e $z = 4.7$.

3.3 - Solução do Problema de Aproximação Linear em Norma l_1 com Restrições

Dado um sistema de m equações lineares a n incógnitas,

$$Ax = b, \qquad (3.3.1)$$

calculamos uma solução de (3.3.1) sujeito a l restrições de igualdade lineares

$$Cx = d \qquad (3.3.2)$$

e k restrições de desigualdade lineares

$$Ex \leq f, \qquad (3.3.3)$$

isto é, devemos determinar um vetor coluna x^* que minimiza

$$||b - Ax||_1 = \sum_{i=1}^{k} |b_i - a_i x| \qquad (3.3.4)$$

sujeito às restrições (3.3.2) e (3.3.3), onde b_i denota o i-ésimo componente de b e a_i denota a i-ésima linha de A.

A formulação é completamente geral no sentido de que restrições não são impostas sobre os tamanhos das matrizes A, C e E, ou sobre os sinais dos elementos de f.

Podemos usar esta formulação para resolver o problema de aproximação l_1 com restrições. Suponha que dados consistindo de k pontos (t_i, y_i) serão aproximados por uma função aproximante linear $x_1\varphi_1(t) + x_2\varphi_2(t) + \ldots + x_n\varphi_n(t)$, onde certas restrições lineares são impostas nos parâmetros $x_1, x_2, \ldots, x_n$. Isso é equivalente a achar uma solução l_1 para o sistema de equações

$$\sum_{j=1}^{n} \varphi_j(t_i)x_j = y_i, \quad i = 1, 2, \ldots, k,$$

sujeito a restrições lineares dadas. Se os valores dos dados y_i contiverem algum erro grande, então uma aproximação l_1 pode ser preferível em lugar de uma solução l_2 (quadrados mínimos).

3.3.1 - Um algoritmo melhorado para aproximação linear l_1 discreto

Descreveremos um algoritmo que segundo Barrodale & Roberts [8] parece ser o mais eficiente para resolver o problema de aproximação linear l_1 geral. O algoritmo é uma modificação do método simplex aplicado à formulação primal do problema linear l_1.

O problema de aproximação linear l_1 geral pode ser descrito como segue. Seja $f(x)$ uma função real definida sobre um subconjunto discreto $X = \{x_1, x_2, \ldots, x_m\}$ do espaço Euclidiano R^N. Dados $n(\leq m)$ funções reais $\varphi_j(x)$ definidas sobre X, formamos uma função aproximante linear $L(A,x) = \sum_{i=1}^{n} a_j \varphi_j(x)$ para algum conjunto $A = \{a_1, a_2, \ldots, a_n\}$ de números reais. O problema l_1 é determinar a melhor aproximação $L(A^*, x)$ que minimiza

$$\sum_{i=1}^{m} |f(x_i) - L(A, x_i)| .$$

Sabe-se que pelo menos uma melhor aproximação sempre existe e há vários algoritmos para calcular a melhor aproximação. Contudo, alguns desses algoritmos requerem que o conjunto de funções $\{\varphi_1(x), \varphi_2(x), \ldots, \varphi_n(x)\}$ seja linearmente independente em X, ou que estas satisfaçam a condição de Haar em X (cada elemento de X possui no máximo n-1 raízes).

Um algoritmo de aproximação de dados deveria impor o mínimo de restrições sobre a função de aproximação escolhida pelo usuário. Já que o algoritmo é baseado no método simplex pode ser usado com qualquer função de aproximação linear.

3.3.2 - O algoritmo

Uma aplicação direta do método simplex para resolver (3.3.5) não é ainda um algoritmo eficiente. De acordo com Barrodale e Young (cf. Barrodale & Roberts [8]) uma solução viável básica inicial para (3.3.5) é eficaz, e que a maioria dos vetores colunas no quadro simplex não necessita ser armazenado explicitamente.

Sejam

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix}$$

e

$$b = [b_1, \ldots, b_n]^T$$

então temos o seguinte quadro simplex :

Tabela 3.3.1 - Quadro simplex condensado inicial

BASES	b	x_1	x_2	. . . .	x_n
u_1	b_1	1,1	1,2	. . . .	n,1
.	.	.	.	. . . . .	.
.	.	.	.	. . . . .	.
.	.	.	.	. . . . .	.
u_m	b_m	1,m	2,m	. . . .	n. m
custo marginal	$\sum_{i=1}^{m} b_i$	$\sum_{i=1}^{m} \varphi_{1,i}$	$\sum_{i=1}^{m} \varphi_{2,i}$	. . .	$\sum_{i=1}^{m} \varphi_{n,i}$

onde $\varphi_{ji} = \varphi_j(x_i)$ com $1 \leq j \leq n$ e $1 \leq i \leq m$.

Algoritmo

Passo 1 - Atribuição de valores iniciais

Passo 2 - Estágio 1: Determine a coluna de pivô

Passo 3 - Determine o vetor para entrar na base, onde o vetor é escolhido de modo que tenha o maior custo marginal positivo.

Passo 4 - Se não existir custo marginal positivo então

- fim bem sucedido;
- prepare a saída e pare

Passo 5 - Determine o vetor para sair da base, escolhido entre os vetores u_i e v_i selecionando aquele que cause a máxima redução na função objetivo

Passo 6 - Se a coluna de pivô não contiver elementos positivos então

- se estágio ≠ estágio 1 então fim prematuro,
- prepare a saída e pare

Passo 7 - Efetue uma transformação simplex no quadro, que é equivalente a um movimento pelos vários vértices vizinhos

Passo 8 - Se número de iterações > N então

- Estágio 2 - volte ao passo 2

Passo 9 - Volte para o passo 2.

As regras para a transformação simplex do quadro são as regras normais empregadas na forma padrão do método simplex. Maiores detalhes do algoritmo podem ser encontrados em Barrodale & Roberts [8].

Mostramos, agora, um exemplo de como o algoritmo é usado.

Sejam

$$A = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \\ 1 & 5 \end{bmatrix}$$

e

$$b = [1, 1, 2, 3, 2]^T.$$

Então, temos o seguinte quadro simplex inicial:

Tabela 3.3.1

Quadro I

BASES	b	x_1	x_2	u_1	u_2	u_3	u_4	u_5	v_1	v_2	v_3	v_4	v_5
u_1	1	1	1	1					-1				
u_2	1	1	2#		1					-1			
u_3	2	1	3&			1					-1		
u_4	3	1	4				1					-1	
u_5	2	1	5*					1					-1
CUSTO MARGIN.	9	5	15	0	0	0	0	0	-2	-2	-2	-2	-2

UFPb/BIBLIOTECA/PRAI

Na primeira iteração, no Estágio 1, x_2 é introduzido na base com o custo marginal igual a 15. O pivô do simplex normal (5*) corresponde a uma aproximação no qual $x_2 = \frac{2}{5}$, $u_5 = v_5 = 0$, isto é, uma aproximação que interpola o quinto ponto dado. Entretanto, se nós incrementarmos x_2 além do valor $\frac{2}{5}$, nós podemos reduzir mais a função objetivo, isto tornaria u_5 negativo. Trocamos, aqui, u_5 na base por v_5. Isto pode ser executado subtraindo-se duas vezes a quinta linha da linha do custo marginal (assim fazemos o custo marginal de $v_5 = 0$, mudando o sinal da quinta linha e trocando o rótulo de u_5 por v_5 na base). O custo marginal de x_2 é agora igual a 5, e incrementamos, agora, x_2.

O segundo pivô (2#) corresponde a uma aproximação no qual $x_2 = \frac{1}{2}$, $u_2 = v_2 = 0$, isto é, uma aproximação que interpola o segundo ponto dado. Permutando u_2 com v_2 na base reduzimos o custo marginal para 1. Deste modo incrementamos mais x_2.

O terceiro pivô (3&) corresponde a uma aproximação na qual $x_2 = \frac{2}{3}$, $u_3 = v_3 = 0$, isto é, uma aproximação que interpola o terceiro ponto dado. A função objetivo não pode diminuir mais pelo incremento de x_2; posteriormente se permutarmos u_3 e v_3 na base, o custo marginal de x_2 torna-se -5. Neste momento passamos a usar este elemento (3&) como pivô e colocamos x_2 na base no lugar de u_3. Usando a quadro condensado, a solução completa é dada na tabela 3.3.3.

Depois de duas iterações, os custos marginais de u_1 e u_2 são -1 e 0 respectivamente, e assim os custos marginais dos vetores v_1 e v_2 são -1 e -2 respectivamente.

O quadro final (quadro IV) representa a melhor aproximação para $L(A^*,x) = \frac{1}{2} + \frac{1}{2}x$ que interpola o primeiro e o terceiro ponto dado.

Tabela 3.3.3 - Quadros condensados

QUADRO II

BASES	b	x_1	x_2
u_1	1	1	1
u_2	1	1	2
u_3	2	1	3&
u_4	3	1	4
u_5	2	1	5
CUSTO MARGINAL	9	5	15

QUADRO III

BASES	b	x_1	x_2
u_1	1/3	2/3	-1/3
v_2	1/3	-1/3	2/3
b_3	2/3	1/3	1/3
u_4	1/3	-1/3	-4/3
v_5	4/3	2/3	5/3
CUSTO MARGINAL	7/3	2/3	-1/3

QUADRO IV

BASES	b	x_1	x_2
b_1	1/2	3/2	-1/2
v_2	1/2	1/2	1/2
b_2	2/2	-1/2	1/2
u_4	1/2	1/2	-3/2
v_5	1	-1	-5
CUSTO MARGINAL	2	-1	0

3.4 - Algoritmos Que Usam Norma l_2

Se X for um espaço de produto interno e X_n é um sub-espaço de dimensão finita gerado pelos vetores $\varphi_1, \varphi_2, \ldots, \varphi_n$, então dado um vetor x em X, há um único vetor x_N em X mais próximo de x na norma $||x|| = \sqrt{(x,x)}$, isto é, x_N resolve o problema

$$||x - x_N|| = \min_{\bar{x} \text{ em } X_n} ||x - \bar{x}||,$$

e é conhecido como ajuste dos quadrados mínimos para x em X_n.

O método dos quadrados mínimos é um método variacional para aproximação de soluções de equações envolvendo operadores lineares do tipo

$$Lx = f \qquad (3.4.1)$$

de sub-espaços de X no qual o operador L é definido. Em particular se $(,)$ denota um produto interno em X, e se $X_n = \text{Span}(\varphi_1, \varphi_2, \ldots, \varphi_n)$ (espaço gerado por $\varphi_1, \ldots, \varphi_n$) e $Y_m = \text{Span}(\psi_1, \psi_2, \ldots, \psi_n)$ são sub-espaços de dimensão n de x e φ_i e ψ_j são vetores, o método geral de Galerkin (sendo o método dos quadrados mínimos um dos casos particulares deste método), procura como aproximação da solução x de (3.4.1) para um dado f obter uma função x_n em X_n satisfazendo o sistema de equações

$$(Lx_n - f, \psi_i) = 0 \qquad (3.4.2)$$

onde:

Lx_n é de dimensão $m \times 1$,

ψ_i é de dimensão $m \times 1$,

L é de dimensão $m \times n$,

$1 \leq i \leq m$ e $1 \leq j \leq n$.

Assim, se

$$x_n = c_1\varphi_1 + c_2\varphi_2 + \ldots + c_n\varphi_n$$

e se L for linear, obtemos, substituindo em (3.4.1) x_n deve satisfazer o sistema linear

$$\sum_{j=1}^{n} (L\varphi_i, \varphi_j)c_j = (f, \psi_i), \qquad 1 \leq i \leq m$$

No caso dos quadrados mínimos, com

$$\psi_i = D\,\varphi_i$$

obtém-se

$$(Lx_n - f, \psi_i) = 0 = (Lx_n - f, D\,\varphi_i).$$

Substituindo x_n, por sua expressão, obtém-se

$$(L\,\varphi_j, \varphi_i)c_j = \sum_{j=1}^{n} (L\,\varphi_j, L\,\psi_i)c_j = (f, D\,\varphi_i)$$

$$i = 1,..m$$

ou seja, transformamos o problema no problema de obtenção dos c_j.

Facilmente podemos estabelecer a existência de x_n neste caso, desde que x_n seja a solução de

$$||Lx_n - f||_2 = \min_{\tilde{x} \text{ em } X_n} ||L\tilde{x} - f||_2.$$

A solução x_n da forma $c_1\varphi_1 + \ldots + c_n\varphi_n$ satisfaz a equação

$$\sum_{j=1}^{n} (L\,\varphi_i, L\,\varphi_j)c_j = (f, D\,\varphi_i), \qquad i = 1, \ldots m.$$

Desenvolvendo o produto interno e chamando

$$b = [b_1, \ldots, b_n]^T, \text{ onde } b_i = (f, D\,\varphi_i), \quad i = 1, \ldots, m$$

chega-se ao problema

$$Ax = b. \qquad (3.4.3)$$

onde $a_{ij} = (L\,\varphi_i, L\,\varphi_j)$, $i = 1, \ldots, m$ e $j = 1, \ldots n$.

O problema de se encontrar uma solução para

$$Ax = b \qquad (3.4.4)$$

pode ser resolvido através de transformações ortogonais ou equações normais. Optamos por fazer uma rápida referência a esses métodos porque são bem conhecidos.

Equação normal

Pode-se mostrar que a solução do sistema hiperdeterminado (3.4.4) satisfaz

$$A^T Ax = A^T b$$

que é um sistema quadrado da forma

$$\tilde{A}x = \tilde{b} \qquad (3.4.5)$$

em que $\tilde{A} = A^TA$ e $\tilde{b} = A^Tb$, só poderá ser resolvido se $\tilde{A}$ for não singular, ou seja, se as colunas de A forem linearmente independentes.

O problema resume-se então em resolver um sistema quadrado e simétrico onde podemos aplicar a decomposição de Cholesky. Infelizmente A^TA é mal condicionada.

Transformações ortogonais

Neste caso podemos ter dois modos de resolução:

a) técnicas de Householder e Gram-Schmidt

Seja A uma matriz m x n (m > n) com n colunas linearmente independentes. Então existem Q, uma matriz ortogonal m x n e R uma matriz triangular superior n x n tal que

$$A = QR. \qquad (3.4.6)$$

Substituindo (3.4.6) em (3.4.4) obtemos

$$QRx = b$$

e multiplicando os dois membros da igualdade por Q^T obtemos

$$QQ^TRx = Q^Tb,$$

portanto

$$Rx = Q^Tb,$$

que é um sistema triangular que pode ser facilmente resolvido.

A obtenção de Q e R pode ser efetuada por transformações de Householder ou pelo método de Gram-Schmidt [15].

b) Decomposição por valores singulares

Se A for m x n, existem matrizes U e V ortogonais e S uma matriz diagonal n x n tais que

$$A = U \begin{bmatrix} S \\ 0 \end{bmatrix} V^T.$$

Então para resolver o problema de quadrados mínimos devemos calcular

$$q = U^Tb$$

e resolver

$$\begin{bmatrix} S \\ O \end{bmatrix} p = g \qquad e \qquad x = Vp.$$

Maiores detalhes sobre o problema de quadrados mínimos veja [12].

3.4.1 - Algoritmo para Interpolação e Aproximação Suaves por Splines Cúbicos

Interpolação e suavização discreta por splines são considerados de importância para problemas envolvendo funções definidas em pontos discretos igualmente espaçados. Possíveis áreas de aplicação são: análise de séries temporais discretas e rotinas de computador para traçar curvas, tendo um conjunto discreto de dados.

Para ambos, interpolação e aproximação suave por splines cúbicos, direcionamos nossa atenção para o problema de uma função b(t) definida no intervalo $[\tau_1, \tau_n]$. Os valores $b(\tau_i) = b_i$ são especificados para $i = 1, 2, \ldots, n$ onde $\tau_i < \tau_{i+1}$. Para muitas aplicações é conveniente assumir que b(t) seja definida num conjunto discreto de pontos $T_M = \langle t_0, \ldots, t_M \rangle$ onde $t_j = \tau_1 + jh$ para $h > 0$, e também que os τ_i,s pertençam a T_M.

Splines cúbicos definidos em $[\tau_1, \tau_n]$ com nós $\kappa_n = (\tau_1, \ldots, \tau_n)$ são funções S(T) tendo a forma $S(t) = S_i(t)$

$$S(t) = \begin{cases} S_i(t) \text{ para } t \text{ em } [\tau_i, \tau_{i+1}] \\ 0 \quad \text{caso contrário} \end{cases}$$

onde

$$S_i(t) = b_i + c_i(t - \tau_i) + x_i(t - \tau_i)^2 + d_i(t - \tau_i)^3, \qquad (3.4.7)$$

e zero fora deste intervalo, isto é, S(t) é cúbico por partes. Os $S_i(t)$ satisfazem as seguintes condições:

$$S_i(\tau_{i+1}) = S_{i+1}(\tau_{i+1}), \qquad (3.4.8)$$

(CONTINUIDADE)

$$\frac{1}{2}(\nabla + \Delta) S_i(\tau_{i+1}) = \frac{1}{2}(\nabla + \Delta) S_{i+1}(\tau_{i+1}), \qquad (3.4.9)$$

(CONTINUIDADE DA DERIVADA PRIMEIRA)

$$\nabla \Delta S_i(\tau_{i+1}) = \nabla \Delta S_{i+1}(\tau_{i+1}), \qquad (3.4.10)$$

(CONTINUIDADE DA DERIVADA SEGUNDA)

onde $\Delta f(x) = f(x + h) - f(x)$ e $\nabla f(x) = f(x) - f(x - h)$.

O spline cúbico S(t) é igual a $S_i(t)$ para t em $[\tau_i, \tau_{i+1})$ com i = 1, ..., n-1, onde $S_i(t)$ é dado por (3.4.7). Uma melhor forma para derivar equações é representar $S_i(t)$ usando polinômios fatorados como segue:

$$S_i'(t) = b_i + \bar{c}_i(t - \tau_i) + x_i(t - \tau_i)(t - \tau_{i-1}) + d_i(t - \tau_{i+h})(t - \tau_i)(t - \tau_{i-h}) \qquad (3.4.11)$$

onde b_i, x_i e d_i em (3.5.5) são como em (3.4.7), mas

$$c_i = \bar{c}_i + x_i h - d_i h^2.$$

Das condições (3.4.8), (3.4.9) e (3.4.10) com $s(\tau_n) = b_n$ resultam as equações

$$\gamma_i x_i + (\eta_i + \eta_{i+1})x_{i+1} + \gamma_{i+1} x_{i+2} = 3[b_{i+1} - b_i] \qquad (3.4.12)$$

para i = 1, ..., n-3

onde:

$$H_i = \tau_{i+1} - \tau_i, \qquad (3.4.12a)$$

$$\gamma_i = H_i - \frac{h^2}{H_i}, \qquad (3.4.12b)$$

$$\eta_i = 2H_i + \frac{h^2}{H_i}, \qquad (3.4.12c)$$

$$b_i = b[\tau_{i+1}, \tau_i] = \frac{(b_{i+1} - b_i)}{H_i},$$

o resto das equações lineares para os c_i vem das condições dos pontos finais .

O sistema de equações lineares denotado por

$$Ax = b \qquad (3.4.13)$$

se torna determinado impondo um dos três tipos de condições (I, II e III) para pontos extremos. Esses sistemas são formas modificadas dos sistemas de Lyche (cf. Duris [9]).

I - Para a condição dos pontos finais com a primeira diferença central dividida temos:

$$A = \begin{bmatrix} \eta_1 & \gamma_1 & 0 & \cdots & 0 \\ \gamma_1 & (\eta_1+\eta_2) & \gamma_2 & & \\ 0 & \gamma_2 & (\eta_2+\eta_3) & & \\ \vdots & & & & \\ & & & & 0 \\ \vdots & & & (\eta_{n-2}+\eta_{n-1}) & \\ & & \gamma_{n-2} & & \gamma_{n-1} \\ 0 & \cdots & 0 & \gamma_{n-1} & \gamma_{n-1} \end{bmatrix}$$

(3.4.14a)

$$x = [x_1, \ldots, x_n]^T, \qquad (3.4.14b)$$

$$b = 3[b_1 - S_1, b_2 - b_1, \ldots, b_{n-1} - b_{n-2}, S_n - b_{n-1}]^T. \qquad (3.4.14c)$$

Os b_n em (3.4.14b) são dados por $\nabla \Delta S(\tau_n)/2h^2$ e são introduzidos para preservar a simetria e η_i e γ_i definidos em (3.4.12b) e (3.4.12c) respectivamente.

II - Para a condição dos pontos finais com a segunda diferença central dividida temos:

$$A = \begin{bmatrix} (\eta_1+\eta_2) & \gamma_2 & 0 & \cdots & 0 \\ \gamma_2 & (\eta_2+\eta_3) & \gamma_3 & & \vdots \\ 0 & \gamma_3 & \ddots & & \vdots \\ \vdots & & & & \vdots \\ & & & & 0 \\ & & & & \gamma_{n-2} \\ 0 & & 0 & \gamma_{n-2} & (\eta_{n-2}+\eta_{n-1}) \end{bmatrix} \quad (3.4.15a)$$

$$x = [x_2, x_3, \ldots, x_{n-1}]^T, \quad (3.4.15b)$$

$$b = [b_2 - b_1 - \frac{1}{6}\gamma_1 S_1, b_3 - b_2, \ldots, b - b - \frac{1}{6}\gamma_{n-1} S]. \quad (3.4.15c)$$

III - Para a condição periódica temos:

$$A = \begin{bmatrix} (\eta_{n-1}+\eta_2) & \gamma_1 & 0 & \cdots & 0 & 0 \\ \gamma_1 & (\eta_1+\eta_2) & \gamma_2 & & & 0 \\ 0 & \gamma_2 & \ddots & & & \vdots \\ \vdots & & & & & \vdots \\ & & & & & 0 \\ & & & & & \gamma_{n-2} \\ \gamma_{n-1} & 0 & \cdots & 0 & \gamma_{n-2} & (\eta_{n-2}+\eta_{n-1}) \end{bmatrix} \quad (3.4.16a)$$

$$x = [x_1, \ldots, x_{n-1}]^T, \quad (3.4.16b)$$

$$b = 3[b_1 - b_{n-1},\ b_2 - b_1,\ \ldots,\ b_{n-1} - b_{n-2}]. \qquad (3.4.16c)$$

A matriz do sistema é tridiagonal exceto o γ_{n-1} no canto inferior. Este tipo de sistema é resolvido usando o método descrito por Bjorck e Golub (conforme Duris[9]).

Os sistemas lineares provenientes de I, II e III são todos positivos e simétricos com os elementos das matrizes A positivos. O número de condição em norma infinita para a matriz A em (3.4.15a e 3.4.16a) é limitada por [9]

$$Cond_\infty(A) \leq || A ||_\infty || A^{-1} ||_\infty \leq \frac{\max_i (H_{i+1} + H_i)}{\min_i (H_{i+1} + H_i)}$$

para (3.5.8a) o limite é

$$Cond_\infty(A) \leq \frac{\max_i (H_{i+1} + H_i)}{\min_i (H_i)}$$

Uma vez que os c_i sejam conhecidos, os b_i e d_i em (3.4.7) são dados por

$$d_i = \frac{1}{3H_i}[c_{i+1} - c_i], \qquad (3.4.17)$$

$$b_i = g_i - \frac{H_i}{3}(2c_i + c_{i+1})$$

para i = 1, ...n-1 (para o caso periódico $c_n = c_1$).

Diferenças divididas

A fórmula para as diferenças divididas geral é convenientemente obtida de uma tabela de diferenças divididas mostrada na tabela 3.4.1. As primeiras duas colunas são os dados (x_i, y_i) e o restante da tabela é dado pela seguinte fórmula

$$f[x_0, \ldots, x_n] = \frac{f[x_1, \ldots, x_n] - f[x_0, \ldots, x_n]}{x_n - x_0}$$

Tabela 3.4.1 - Tabela de diferenças divididas geral

x_i	y_i	primeira	segunda	terceira	quarta
x_0	$f[x_0]$				
		$f[x_0, x_1]$			
x_1	$f[x_1]$		$f[x_0, x_1, x_2]$		
		$f[x_1, x_2]$		$f[x_0, x_1, x_2, x_3]$	
x_2	$f[x_2]$		$f[x_1, x_2, x_3]$		$f[x_0, x_1, x_2, x_3, x_4]$
		$f[x_2, x_3]$		$f[x_1, x_2, x_3, x_4]$	
x_3	$f[x_3]$		$f[x_2, x_3, x_4]$		
		$f[x_3, x_4]$			
x_4	$f[x_4]$				

Algoritmo de interpolação

Passo 1 - Se condição ≠ III e n > 0 então

faça

- $h_2 = h * h$

- $n_1 = n - 1$

- se n ≠ 2 e condição ≠ II então

faça

- $n_2 = n_1 - 1$

senão faça

- $b_1 = 0$

- $c_1 = 0$

- $d_1 > 0$

- calcule a primeira diferença dividida

- calcule a segunda diferença dividida

- Pare

Passo 2 - Coloque no sistema linear as condições apropriadas dos pontos finais (cf. (3.4.14) ou (5.4.15) ou (3.4.16))

Passo 3 - Se condição = I então

- calcule as condições dos pontos finais da primeira diferença central dividida

$$S_1 = \frac{1}{2} \frac{(\nabla + \Delta) S(\tau_1)}{h}$$

e

$$S_n = \frac{1}{2} \frac{(\nabla + \Delta) S(\tau_n)}{h}$$

- vá para o passo 5

Passo 4 - Calcule c_i, b_i, e d_i

- calcule a primeira diferença dividida

- calcule a segunda diferença dividida

- Pare

Passo 5 - Se condição = II então

- calcule as condições dos pontos finais da segunda diferença central dividida

$$S_1 = \frac{(\nabla + \Delta) S(\tau_1)}{h^2}$$

e

$$S_n = \frac{(\nabla + \Delta)\, S(\tau_n)}{h^2}$$

- vá para o passo 4

Passo 6 - Se condição = III então

- calcule as condições finais periódicas

$$\frac{1}{2}(\nabla + \Delta)\, S(\tau_1) = \frac{1}{2}(\nabla + \Delta)\, S(\tau_n)$$

e

$$S(\tau_1) = S(\tau_n).$$

- vá para o passo 4.

Mostramos agora um exemplo de como o algoritmo de interpolação é usado para achar o spline cúbico interpolante para h = 0.1, n = 6 e os valores da função b dados abaixo.

i	τ_i	b_i
1	0.5	1.0
2	0.7	0.5
3	1.0	2.0
4	1.5	2.5
5	2.1	2.0
6	2.5	1.0

Para as três condições dos pontos finais temos:

I) Primeira diferença central dividida

$$\frac{1}{2}\,\frac{(\nabla + \Delta)\, S(\tau_1)}{h} = -1.0$$

$$\frac{1}{2}\,\frac{(\nabla + \Delta)\, S(\tau_n)}{h} = 0.0 .$$

II) Segunda diferença central dividida

$$\frac{(\nabla + \Delta)\, S(\tau_1)}{h^2} = 0.0,$$

$$\frac{(\nabla + \Delta)\, S(\tau_n)}{h^2} = 0.0 .$$

III) Periódica

$$S(\tau_1 - h) = S(\tau_n - h), \quad S(\tau_1) = S(\tau_n), \quad S(\tau_1 + h) = S(\tau_n + h).$$

SOLUÇÃO PARA I

i	intervalo	b_i	c_i	x_i	d_i
1	[0.5,0.7]	1.0	-1.751037	-18.76556	75.10376
2	[0.7,1.0]	0.5	0.930633	2.629669	-42.44052
3	[1.0,1.5]	2.0	4.736782	-11.89979	8.8852551
4	[1.5,2.1]	2.5	-0.4001307	1.378888	-3.501484
5	[2.1,2.5]	3.0	-2.697525	-4.923788	13.54401

Em particular, o spline cúbico discreto S(τ) para τ = 1.2 é dado por

$$S(1.2) = 2.0 + 4.736782 \times t - 11.89979 \times t^2 + 8.852451 \times t^3$$

onde t = (1.2 - 1.0) = 0.2.

SOLUÇÃO PARA II:

i	intervalo	b_i	c_i	x_i	d_i
1	[0.5,0.7]	1.0	-4.069061	0.00000	39.22655
2	[0.7,1.0]	0.5	1.416796	23.53593	-38.63974
3	[1.0,1.5]	2.0	4.640036	-11.23984	7.9195391
4	[1.5,2.1]	2.5	-0.5627829	0.6394691	-1.817308
5	[2.1,2.5]	3.0	-1.7982191	-2.631689	2.193076

SOLUÇÃO PARA III:

i	intervalo	b_i	c_i	x_i	d_i
1	[0.5,0.7]	1.0	-3.801780	-2.156043	43.32474
2	[0.7,1.0]	0.5	1.357985	23.83881	-38.99586
3	[1.0,1.5]	2.0	4.663832	-11.25748	7.8596271
4	[1.5,2.1]	2.5	-0.6051248	0.5319628	-1.520514
5	[2.1,2.5]	3.0	-1.6245391	-2.204966	0.0476882

Aproximação suave por spline cúbico

Duris desenvolve a teoria para aproximação suave por spline cúbico. Este tipo de aproximação foi originalmente estudado por Whittaker numa forma ligeiramente diferente. As equações para o caso cúbico podem ser derivadas do seguinte teorema.

Teorema 3.5.2 - Sejam ρ, h e W_i, i = 1, ...,n números reais positivos. Então existe uma única função S(t) definida em $T_M = (t_0, t_1, \ldots, t_m)$ que minimiza

$$\alpha(f(t)) = \rho \sum_{j=1}^{n} W_j(f(\tau_j) - b_j)^2 + \nabla \Delta [\sum_{i=1}^{m-1} f(t_i)]^2$$

Este S(t) é o único spline cúbico discreto satisfazendo:

$$S(\tau_i) = b_i - \frac{h^2}{\rho W_i}[d_i - d_{i-1}] \qquad (3.4.18)$$

onde $d_i = (x_{i+1} - x_i) / (3H_i)$ e $d_0 = d_n = 0$.

O sistema de equações lineares resultante para os x_i em (3.4.7) ou (3.4.11) é obtida de (3.4.15a - c) e (3.4.17) pela inclusão de $S_1 = S_n = 0$ e substituindo os b_i por $\tilde{b}_i = S(\tau_i)$, onde $S(\tau_i)$ é dado em (3.4.18). Esse sistema é pentadiagonal, positivo e simétrico.

Algoritmo para aproximação suave

Passo 1 - Se $n \neq 2$ então

- construa o primeiro membro do sistema linear, ou seja calcule os x_i's
- calcule as três diagonais necessárias para a construção da matriz pentadiagonal (sendo que a matriz é simétrica)
- resolva o sistema de equações lineares utilizando os x_i's calculados

senão faça

- $c_1 = (c_2 - b_1) / (\tau_2 - \tau_1)$

- $x_1 = 0$ e $d_1 = 0$

- calcule a primeira diferença dividida

- calcule a segunda diferença dividida

- Pare

Passo 2 - Se $n \leq 3$ então

- calcule x_i

- senão vá para o passo 4

Passo 3 - faça

- $x_n = 0$

- $d_n = 0$

- $x_{n-1} = x_{n-2}$

- calcule c_i, e os demais x_i e d_i

- calcule a primeira diferença dividida

- calcule a segunda diferença dividida

- Pare

Passo 4 - se n = 4 então

- calcule x_1 e x_2

- vá para o passo 3

Passo 5 - Fatore a matriz pentadiagonal

- resolva o sistema pentadiagonal e coloque a solução no vetor x.

- vá para o passo 3.

Mostramos agora um exemplo de como o algoritmo de aproximação é usado para achar o spline cúbico discreto com os seguintes dados: h = 0.1, n = 6 e os valores da função b dados abaixo.

i	τ_i	b_i
1	0.5	1.0
2	0.7	0.5
3	1.0	2.0
4	1.5	2.5
5	2.1	2.0
6	2.5	1.0

Para $\rho = 0.01$:

i	intervalo	$\tilde{b}_i$	c_i	x_i	d_i
1	[0.5,0.7]	0.7307289	1.536621	0.000000	0.8975704
2	[0.7,1.0]	01.045234	1.667059	0.5385422	-1.374241
3	[1.0,1.5]	1.556716	1.611752	-0.6982756	-0.6354344
4	[1.5,2.1]	2.108594	0.4271141	-1.651427	0.3430808
5	[2.1,2.5]	1.844454	-1.189260	-1.033881	0.8615685

Para $\rho = 1.0$:

i	intervalo	$\tilde{b}_i$	c_i	x_i	d_i
1	[0.5,0.7]	0.9166568	-2.545056	0.000000	27.78108
2	[0.7,1.0]	0.6298941	1.329906	16.66866	-26.34149
3	[1.0,1.5]	1.817824	3.915267	-7.038706	4.021049
4	[1.5,2.1]	2.518413	-0.06161821	-1.0071321	-0.5322141
5	[2.1,2.5]	1.993115	-1.921916	-2.055118	01.712599

3.5 - Algoritmos que usam norma l_∞

Os problemas minimax lineares (PMML) da análise numérica podem ser formulados como:

$$\min_{x \text{ em } \mathbb{R}^n} \max ||b - Ax||,$$

Teorema 3.5.1 (cf. Stoer [15]) - Sejam os valores a_{ij}, y_i dados para $1 \leq i \leq m$, $1 \leq j \leq n$ e $m > n$. O problema de achar o $\text{minimax } ||b_i - a_{i1}x_1 + \ldots + a_{in}x_n||$ para $1 \leq i \leq m$ tem uma solução.

Esta é a solução de um sistema hiperdeterminado de equações lineares, aceitando como resposta aqueles valores que tornam mínimo o máximo das discrepâncias individuais. O assunto é bastante conhecido, porém não foi encontrado nenhum algoritmo que use conceitos modernos.

CAPÍTULO IV

APROXIMAÇÃO DE SUPERFÍCIES

A generalização dos métodos de aproximação de funções de uma variável para funções de várias variáveis não é trivial, e somente pode ser feita para casos especiais, pois a teoria para aproximação de funções de várias variáveis ainda não atingiu o nível de desenvolvimento da teoria de aproximação a uma variável.

Há muitas aplicações de aproximações de superfícies ou aproximação de funções de duas variáveis. Algumas envolvem gráficos interativos em tempo real; essas aplicações incluem o projeto de superfícies de automóveis, aviões e barcos. Outras

aplicações envolvem a modelagem de uma superfície, como o do coração ou cérebro humano ou de depósitos minerais na terra.

Suponha que Ω é um conjunto de pontos num plano e f(P) = f(x,y) é uma função definida em Ω. f(x,y) é uma superfície tridimensional e seu gráfico será suave se f e sua derivada primeira for contínua em Ω (veja figura 4.1).

Descrevemos apenas aproximações e interpolações sobre malhas triangulares porque com elas é possível encontrar triangulações (divisão de um domínio no plano x-y) de modo a minimizar o erro cometido. A teoria sobre malhas retangulares pode ser encontrada em Prenter [14].

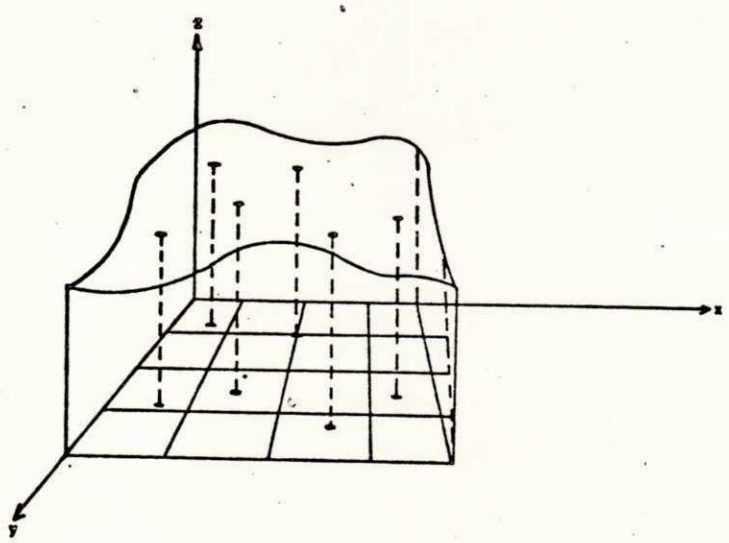

figura 4.1 - Uma função de Duas Variáveis com Uma Superfície Interpolante para Um conjunto de Pontos aleatórios.

Para essa construção necessitamos dos pontos $\langle P_0, P_1, \ldots, P_n \rangle$ da partição π a fim de gerar uma triangulação apropriada de Ω. Especificamente, um conjunto de triângulos não degenerados $\tau = \langle T_0, T_1, \ldots, T_{n-1} \rangle$ tal que

1 - o conjunto de todos os vértices de triângulos pertencentes a τ é π;

2 - cada par de triângulos (T_i, T_j) em τ intersepta em um único vértice, ou em um lado ou não se intersepta;

3 - a união dos T_i's e seus interiores é Ω,

é dito ser uma triangulação apropriada de Ω. Dada uma poligonal limitada no plano, podemos ter várias triangulações. Essas idéias são ilustradas nas figuras 4.2 e 4.3.

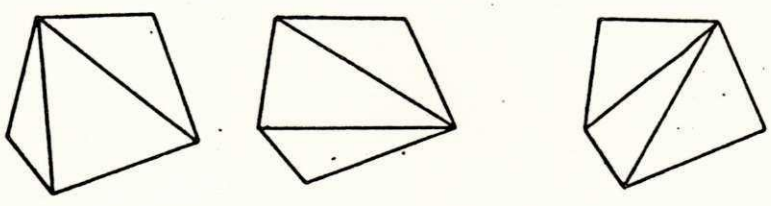

Figura 4.2 - Três Triangulações Distintas da Mesma Região.

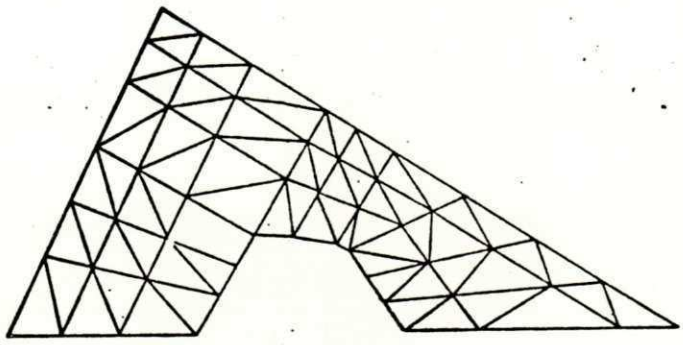

Figura 4.3 - Uma Região Ω com uma Triangulação Apropriada.

Suponha que Ω seja uma região poligonal limitada com uma triangulação apropriada τ. Damos dois métodos específicos para construir polinômios por partes em Ω com triangulação τ e indicamos como esses métodos podem e devem ser generalizados. A triangulação de uma dada região Ω é um problema de programação não trivial para o qual os engenheiros tem devotado uma parte de sua atenção procurando gerar automaticamente as malhas. Primeiro estudaremos o problema de aproximação supondo que a triangulação de nossa região tenha sido dada; na seção 4.2 trataremos do problema da triangulação.

4.1.1 - O método das placas

A idéia do método das placas (cf. Prenter [14]) é a de aproximar uma função $f(x,y)$ por planos por partes que interpolam f nos vértices de alguma triangulação apropriada do domínio Ω de f. Por exemplo, suponha que Ω é a região mostrada na figura 4.3 com a triangulação τ. Em cada triângulo T_i de Ω, construimos um plano $p_i(x,y)$

$$p_i(x,y) = a_i x + b_i y + c_i ,$$

interpolando f pelos vértices de τ_i. Definamos nosso interpolante linear por partes $S_N(x,y)$ de f

$$S_N(x,y) = p_i(x,y)$$

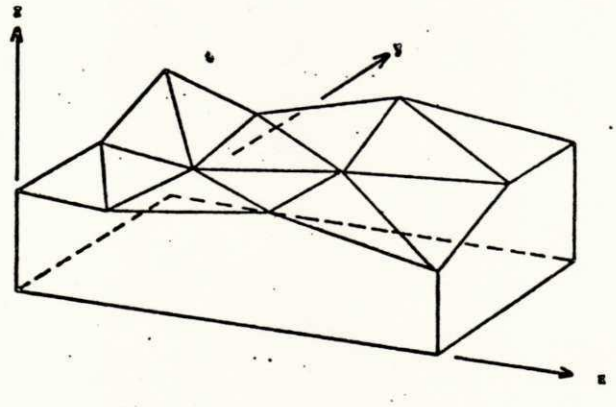

Figura 4.4 - Gráfico de um Polinômio de Lagrange Linear por Partes.

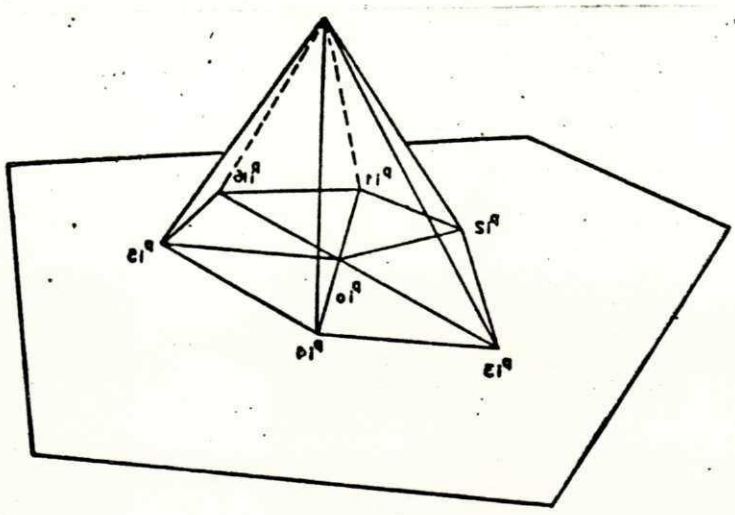

Figura 4.5 - Gráfico de uma função base $\phi_i(x,y)$

se (x,y) estiver em T_i ou em seu interior. A função resultante é um polinômio de Lagrange por partes de grau um como ilustrado na figura 4.4. Os polinômios de Lagrange por partes são constituídos pelas somas dos produtos dos polinômios base de Lagrange por partes na direção de x com polinômios base de Lagrange por partes na direção y. Seja $W(x) = (x - x_0)(x - x_1) \ldots (x - x_n)$, então o polinômio interpolador de Lagrange tem a forma

$$p_n(x) = \sum_{i=0}^{n} f(x_i)\, l_i(x),$$

em que

$$l_i(x) = \frac{W(x)}{(x - x_i)\, W'(x_i)}$$

A base canônica para computar $S_N(x,y)$ é o conjunto de funções $\langle \varphi_0, \varphi_1, \ldots, \varphi_n \rangle$, que são os polinômios lineares por partes provenientes do método das placas e que resolve o problema de interpolação

$$\varphi_i(P_j) = \delta_{ij}, \quad 0 \leq i,j \leq n.$$

O gráfico de uma típica função de forma $\varphi_i(x,y)$ correspondendo a um vértice interior P_i tem a forma dada na figura 4.5. Isto mostra que

$$S_N(x,y) = \sum_{i=0}^{n} f(P_i)\, \varphi_i(x,y)$$

e que S_n é único, suas placas unicamente determinadas por três pontos não colineares. Note que S_N é contínuo sobre Ω, desde que $p_i(x,y) = p_j(x,y)$ ao longo dos lados compartilhados dos triângulos adjacentes T_i e T_j, respectivamente.

4.1.2 - Um esquema de seis pontos sobre uma malha triangular

O método das placas consiste em construir polinômios de Lagrange linear por partes sobre uma malha triangular. Construiremos agora polinômios de Lagrange quadráticos por partes sobre a malha. Especificamente, seja $\tau = \langle T_0, T_1, \ldots, T_{n-1} \rangle$ uma

triangulação de Ω. Em cada triângulo T_i de τ construimos os pontos médios de cada lado. Nossa malha consistirá dos vértices P_{i0}, P_{i2}, P_{i4} juntamente com os pontos médios P_{i1}, P_{i3}, P_{i5} de cada triângulo T_i de τ (veja figura 4.6).

Em cada triângulo T_i computa-se a superfície quadrática

$$P_i(x,y) = a_i x^2 + b_i xy + c_i y^2 + d_i x + e_i y + f_i,$$

$$P_i(P_{ij}) = f(P_{ij}),\ 0 \le j \le 5.$$

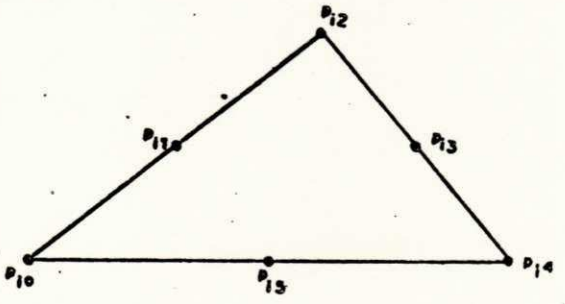

Figura 4.6 - Triângulo T_i e seus pontos de malha P_{ij}, $0 \le j \le 5$.

O polinômio interpolante quadrático por partes $S_N(x,y)$ de $f(x,y)$ com respeito a esta malha deve satisfazer

$$S_N(x,y) = P_i(x,y),$$

se (x,y) pertencer ao triângulo T_i ou ao seu interior. Isto mostra que o gráfico de $S_N(x,y)$ é uma superfície parabólica por partes (veja figura 4.7). Pode-se provar a existência e unicidade dessas funções simplesmente calculando-as.

O método preferido de computação pelos engenheiros é outra vez via base canônica ou funções de forma $\varphi_i(x,y)$ resolvendo o problema de interpolação

$$\varphi_i(P_j) = \delta_{ij}, \quad 0 \leq i,j \leq N.$$

Onde $\langle P_0, P_1, \ldots, P_N \rangle$ é o conjunto de vértices da nossa malha. Gráficos de funções de forma típicas no triângulo T_i são dadas na figura 4.8. Isto mostra que

$$S_N(x,y) = \sum_{i=0}^{n} f(x_i,y_i)\varphi_i(x,y).$$

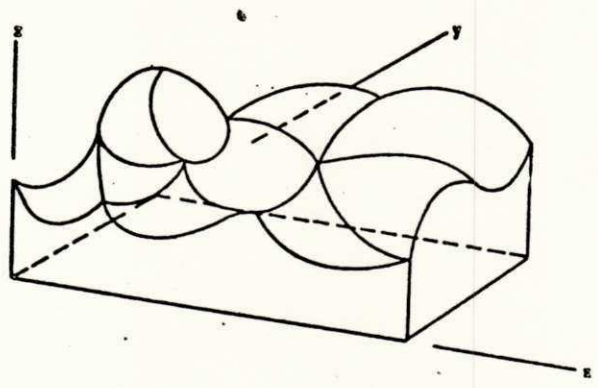

Figura 4.7 - Uma Superfície Quadrática Por Partes.

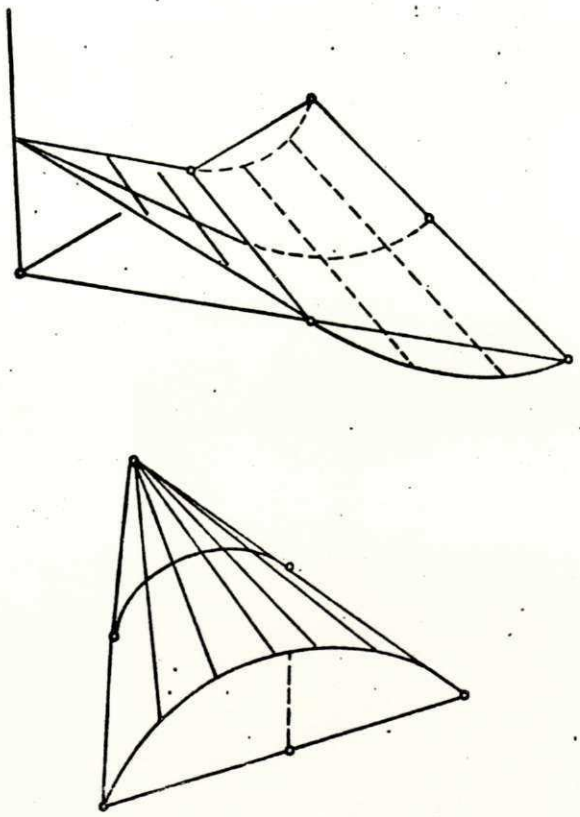

Figura 4.8 - Polinômios Base de Lagrange Quadráticos.

A computação dos φ_i's é facilmente efetuada como produtos de planos.

Note que a função $S_N(x,y)$ será contínua em todo Ω, desde que $S_N(x,y)$ se reduza a uma simples parábola ao longo dos lados adjacentes dos triângulos interseptantes (T_i, T_j).

4.2 - Geração automática de malhas

Construímos vários esquemas de aproximação sobre triangulações τ da poligonal limitada Ω no plano x - y. Para computar com tais aproximações, ao resolver equações diferenciais parciais, ou ainda em problemas diretos de aproximação, devemos efetuar uma triangulação apropriada de modo sistemático. Seria uma formidável tarefa se não pudéssemos lançar mão do computador para nos ajudar, particularmente para triangulações envolvendo um grande número de triângulos. Triangulações com a ajuda do computador e outras partições geométricas de um domínio são conhecidos como modelos de geração automática de malhas.

Dada uma poligonal limitada e conexa no plano real, como programar o computador para particionar Ω numa triangulação apropriada que nos permita usar o método das placas e outros polinômios de Lagrange por partes sobre tais triangulações? Duas triangulações de uma região limitada poligonalmente são ilustradas nas figuras 4.9 e 4.11 e uma triangulação de uma região limitada não poligonal é ilustrada na figura 4.10.

Nosso segundo problema é decidir como achar aproximações por elementos finitos (polinômios por partes) sobre domínios limitados não poligonais. Em particular, o que fazer com fronteiras curvas? Gostaríamos ainda de trabalhar sobre triangulações apropriadas de uma região Ω ou uma aproximação de Ω_N como ilustrado nas figuras 4.9, 4.11 e 4.10. Este problema foi

resolvido satisfatoriamente em parte por Zienkiewicz e Phillips, B.M. Irons, Ergatoudis, Irons, e Zienkiewicz, Ccf. Prenter [14]) usando o que são conhecidas na literatura como Transformações Isoparamétricas. Se a fronteira de Ω consiste de segmentos de polinômios ou de splines, tentamos usar splines do mesmo grau parametricamente, para mapear um quadrado em Ω de modo que as fronteiras sejam mapeadas exatamente ou aproximadamente em fronteiras de modo que a transformação resultante seja uma injeção.

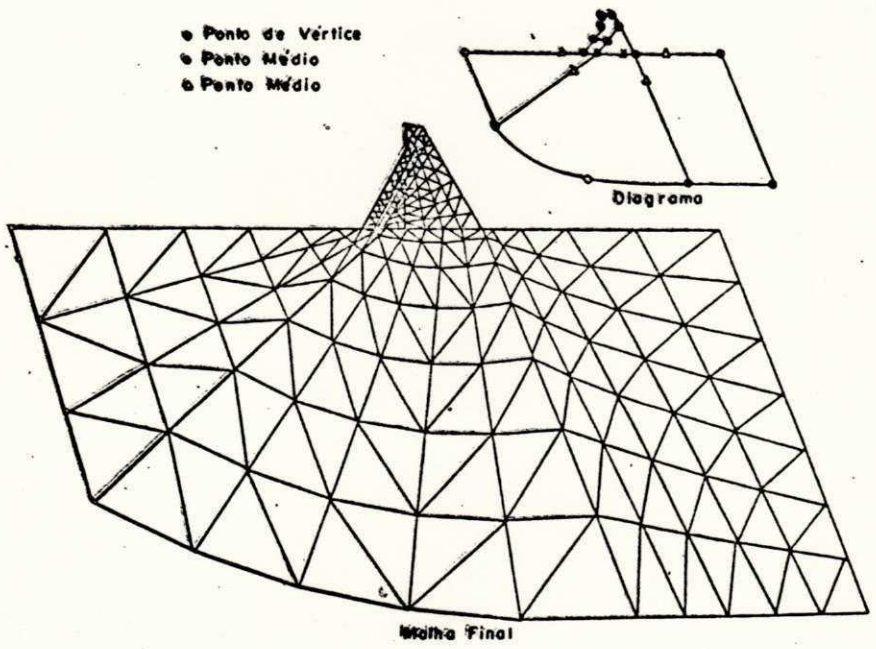

Figura 4.9 - Geração Automática de Malha Numa Região Limitada Poligonalmente.

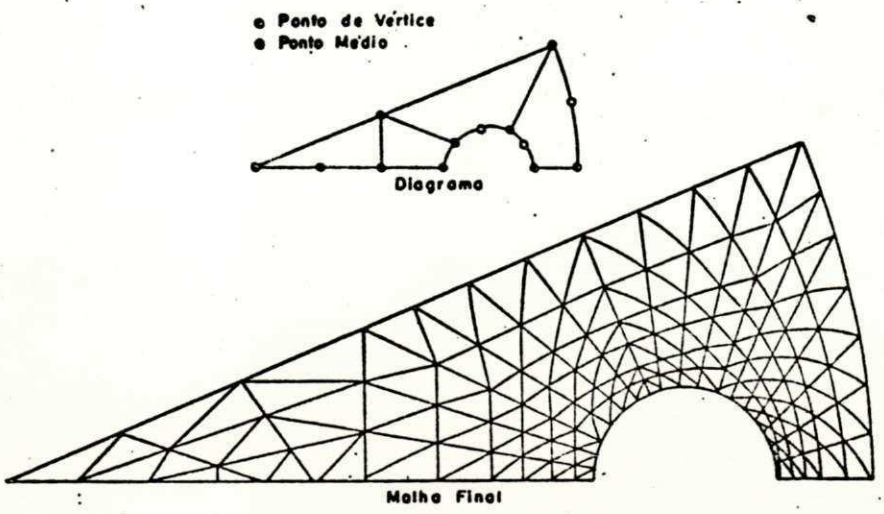

Figura 4.10 - Geração automática de malha Numa Região Limitada Não Poligonal.

Algoritmo - Elemento triangular curvado

Passo 1 - Seja $-1 = s_0 \leq s_1 \leq \ldots \leq s_n = 1$ e $-1 = t_0 \leq t_1 \leq \ldots \leq t_m = 1$ duas partições de um intervalo $[-1,1]$ nas direções s e t, respectivamente.

Passo 2 - Seja $\{T(s_i,t) \mid -1 \leq t \leq 1\}$ e $\{T(s,t_j) \mid -1 \leq s \leq 1\}$, onde $0 \leq i \leq n$ e $0 \leq j \leq m$ o conjunto de curvas coordenadas em Ω que aparecem na figura 4.12.

Passo 3 - O passo 2 particiona Ω em mn quadriláteros R_{ij} com fronteiras curvas. Escolha a diagonal mais curta de cada R_{ij}. O resultado (figura 4.13) é a triangulação de Ω em quantos triângulos possa ter a

fronteira curva.

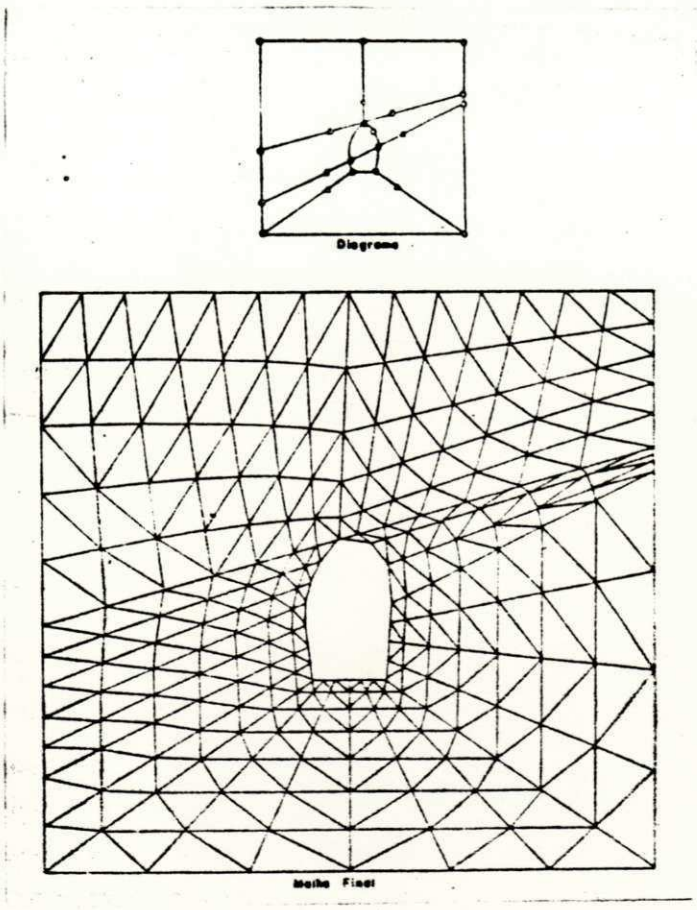

Figura 4.11 - Geração Automática de Malha Numa Região limitada Poligonalmente.

Usando este algoritmo para computar aproximações para uma dada função f(x,y) requer uma base ou um conjunto de funções de forma $\varphi_i(x,y)$, linear por partes, quadráticas, e assim por diante de polinômios de Lagrange sobre uma malha triangular. Se todos os lados de Ω fossem linhas retas, os φ_i's seriam simples funções de forma da seção anterior. Por outro lado, se alguma das fronteiras for curva, necessitamos de um conjunto de funções de forma construídas suavemente sobre esses lados curvos. Isto requeriria uma fórmula específica para T^{-1}. Em particular,

as funções de forma naturais φ_i para a triangulação descrita pelo algoritmo são as funções

$$\varphi_i(x,y) = \psi_i(T^{-1}(x,y)),$$

onde os ψ_i's são as funções de forma sobre uma triangulação análoga de Ω no plano s - t (veja figura 4.14).

A dificuldade com essa aproximação é que usualmente falta uma expressão explicita para T^{-1}. Para contornar essa dificuldade aproximamos por um segmento de reta todas as curvas para ter elementos triangulares verdadeiros. Porém, obteremos uma aproximação Ω_N para Ω.

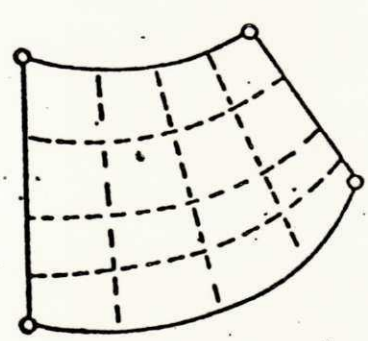

Figura 4.12 - Conjunto de Curvas Coordenadas em Ω.

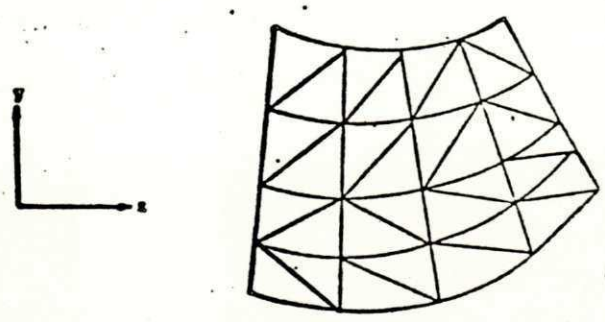

Figura 4.13 - Triangulação de Ω.

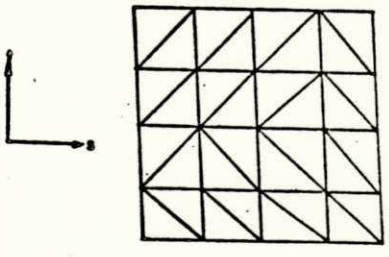

Figura 4.14 - Triangulação Análoga de Ω.

4.3 - O método de Akima

Akima desenvolveu um método de interpolação em duas variáveis e aproximação suave de superfícies. O método utiliza um meio que faz com que a superfície resultante passe por todos os pontos dados. Este método tem uma séria

desvantagem. A aplicabilidade é restrita a casos em que os valores são dados em função de coordenadas cartesianas dos pontos de um plano. O método de Akima se aplica quando os valores da função são dados em pontos arbitrariamente distribuídos no plano x-y. A função interpolante é uma função suave, isto é, ela e suas derivadas parciais de primeira ordem são contínuas.

O método utiliza o melhor critério para triangulação do plano sugerida por Lawson, e o algoritmo que implementa o método tem sido melhorado substancialmente, tanto no espaço de armazenamento requerido como também no tempo de computação. O critério max-min do ângulo sugere que quando um conjunto de quatro pontos são os vértices de um quadrilátero com cada ângulo interno menor do que π, se escolha, entre duas formas possíveis de particionamento do quadrilátero em um par de triângulos, o particionamento que maximiza o menor ângulo interno dos dois triângulos produzidos(cf. Lawson[11]).

Idéia do Algoritmo

O plano x - y é triangulado ou dividido em malhas triangulares, tendo como vértices projeções de três dados sobre o plano, e um polinômio de quinto grau em x e y é construído para cada malha triangular. Valores estimados das derivadas parciais para cada ponto dado são usados na determinação do polinômio.

Para a triangulação do plano x-y é adotado o critério max-min do ângulo. Na triangulação do plano x-y, primeiro ligamos o par de pontos mais próximos. Adicionamos, então um ponto de cada vez em ordem crescente da distância do ponto médio do par de pontos mais próximo. Essa ordenação na adição de novos pontos assegura que o novo ponto será adicionado sempre que estiver situado fora do polígono construído com os pontos antigos. O novo ponto fica fora do círculo cujo centro está no ponto médio do par de pontos mais próximo e passa pelo último ponto antigo adicionado enquanto que o polígono fica dentro do círculo. Cada vez que um novo ponto é adicionado, são construídos triângulos ligando-se o novo ponto com pontos antigos que são visíveis do novo ponto e, sempre que necessário, permutando a numeração dos triângulos.

A interpolação dos valores de z num triângulo é baseado nas três hipóteses seguintes:

(i) O valor da função no ponto (x,y) de um triângulo é interpolado pelo polinômio de quinto grau em x e y, isto é,

$$z(x,y) = \sum_{j=0}^{5} \sum_{k=0}^{5-j} q_{jk} x^j y^k$$

Note que existem 21 coeficientes para serem determinados.

(ii) Os valores da função e suas derivadas de primeira e segunda ordens (isto é, $\partial z/\partial x$, $\partial z/\partial y$, $\partial^2 z/\partial x^2$, $\partial^2 z/\partial xy$ e $\partial^2 z/\partial y^2$) são fornecidos para cada vértice do triângulo (18 condições).

(iii) A derivada parcial da função diferenciada na direção perpendicular (ou normal) a cada lado do triângulo é um polinômio de terceiro grau, no máximo. Essa hipótese fornece três condições adicionais.

A suavidade dos valores interpolados e por conseguinte a suavidade da superfície resultante ao longo do triângulo pode ser obtida como segue. É conveniente introduzir um novo sistema de coordenadas cartesianas, o qual podemos chamar de sistema s-t, no qual o eixo s é paralelo ao lado do triângulo. Já que a transformação de coordenadas entre o sistema x-y e s-t é linear, uma combinação linear dos valores $\partial z/\partial x$, $\partial z/\partial y$, $\partial z/\partial x^2$, $\partial z/\partial xy$ e $\partial z/\partial y^2$ para cada vértice unicamente determinam os valores de $\partial z/\partial s$, $\partial z/\partial t$, $\partial^2 z/\partial ss$, $\partial^2 z/\partial st$ e $\partial z/\partial t^2$ para o mesmo vértice. Então os valores de ∂z, $\partial z/\partial s$ e $\partial^2 z/\partial s^2$ em dois vértices . determinam univocamente um polinômio de quinto grau em s para valores de z pertencentes a um lado. Desde que dois polinômios de quinto grau em x e y representem valores de z em dois triângulos que compartilham um lado comum são reduzidos a um polinômio de quinto grau em s sobre esse lado, esses dois polinômios em x e y coincidem nesse lado comum. Isto prova a continuidade dos valores de z interpolados ao longo de um lado do triângulo. Analogamente

os valores de $\partial z/\partial t$ e $\partial^2 z/\partial st = (\partial z/\partial s)t$ em dois vértices determinam univocamente um polinômio de terceiro grau em s para $\partial z/\partial t$ sobre o lado. Desde que o polinômio representando $\partial z/\partial t$ é de terceiro grau em s, no máximo, dois polinômios representando $\partial z/\partial t$ em dois triângulos que compartilham um lado comum também coincidem nesse lado. Isto prova a continuidade de $\partial z/\partial t$ e assim a suavidade de z ao longo do lado do triângulo.

Para a aproximação de superfícies suaves, a triangulação é realizada do seguinte modo: o plano x-y é dividido em nove áreas retangulares e obtem-se os triângulos associados a cada retângulo. Todos os pontos das áreas retangulares obtidas são localizados e a interpolação é feita para todos os pontos da malha. Neste caso, tanto a localização dos pontos das áreas retangulares, como a interpolação de seus valores são realizadas sobre os triângulos ao invés de sobre os pontos das áreas retangulares. Organizam-se os pontos da malha para aproximação de superfícies escolhendo os pontos em relação ao número de triângulos; para cada triângulo examinam-se os pontos da área retangular onde se encontra o triângulo, buscando pontos da área retangular que estejam dentro do triângulo e armazena-os. Na aproximação de superfície, a interpolação é realizada para os pontos da malha no triângulo consecutivamente, de modo que se evita recalcular desnecessariamente os coeficientes dos polinômios.

4.4 - O método de triangulação de Lawson

Dado o conjunto S de pontos distintos, (x_i, y_i), $i = 1, \ldots, n$, a malha triangular T pode ser construída percorrendo a casca convexa desse conjunto de pontos (casca convexa de S é um polígono convexo cuja área é mínima e contém todos os pontos de S). Cada triângulo na malha tem três dos pontos dados como seus vértices e não contém nenhum outro ponto de S, no interior ou no contorno.

Conceitualmente não há dificuldade em se construir manualmente uma malha triangular.

Em geral, existem diferentes triangulações de um conjunto S. Porém, todas as possíveis triangulações de S tem o mesmo número de triângulos e o mesmo número de lados. Seja n_b o número de pontos de S da casca convexa de S e n_i o número de pontos no interior de maneira que $n = n_b + n_i$. Então o número de triângulos é

$$n_t = n_b + 2(n_i - 1) \leq 2n$$

e o número de lados é

$$n_l = 2n_b + 3(n_i - 1) \leq 3n$$

A tabela 4.1 é uma estrutura de dados representando uma malha triangular proposta por Lawson. A coluna 1 representa o número do triângulo que está sendo avaliado. Cada triângulo é representado por seis números. Como $n_t \leq 2n$ para uma malha triangular de n pontos teremos um total de 12n números para representar esta malha.

Tabela 4.1 - Estrutura de Dados Representando Uma Malha Triangular Baseada na Figura 4.15.

Triângulo Avaliado	Triângulos adjacentes zero indica a região exterior para a malha triangular			Vértices do triângulo avaliado O primeiro vértice é o ponto de contato entre o triângulo avaliado e os triângulos adjacentes		
1	2	0	4	5	8	7
2	5	1	3	5	3	8
3	6	0	2	3	1	8
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.

Este método tem a propriedade de fazer adições e mudanças nas listas de triângulos mas não a eliminação. Por esta razão há a necessidade de armazenar informações sobre os triângulos para posterior operação. O método usa o critério max-min do triângulo.

UFPb/BIBLIOTECA/PRAI

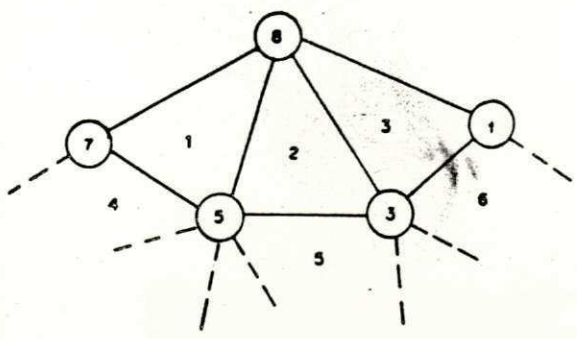

Figura 4.15 - Uma Porção de Uma Malha Triangular Descrita pela Estrutura de Dados da tabela 1.

Descrição do algoritmo

1 - Inicia-se obtendo o ponto de S que tem a menor coordenada x. O ponto P^* obtido é um ponto extremo de uma casca convexa de S. A obtenção de P^* requer O(n) operações.

2 - Os pontos de S são ordenados de acordo com a ordem crescente da distância Euclidiana a P^*. Denote os pontos ordenados por $q_1, q_2, \ldots, q_n$ com $q_1 = P^*$.

3 - O primeiro lado é obtido conectando q_1 e q_2. O próximo ponto na seqüência q_i não colinear com q_1 e q_2 é conectado com q_1 e q_2 para formar o primeiro triângulo. Se este terceiro vértice não for q_3 mas ao contrário q_k com $k > 3$, rerotulam-se os pontos q_3 até q_k de modo que q_k passa a ser q_3 e os índices(x) dos

pontos intermediários passam a ser x +1. Esses passos asseguram que q_j esteja estritamente fora da casca convexa de $\langle q_1, \ldots, q_{j-1} \rangle$ para todo j = 4,5,...,n.

Seja c o centro do triângulo $q_1\ q_2\ q_3$. Seja r o raio do círculo circunscrito. Quando uma coordenada angular for necessária para um ponto, o ângulo será medido no sentido anti-horário ao redor de c (vértice) com raio r. Note que a coordenada angular de q_1 é zero, e todos os outros pontos q_i para i > 1 tem coordenadas angulares estritamente entre 0 e 2Π.

4 - Construa uma lista inicial limitada consistindo de q_1, q_2, q_3, e q_1 junto com suas coordenadas angulares assegurando um ângulo de 2π para a segunda ocorrência de q_1. Em relação aos pontos q_k, k = 4, ..., n faz-se o seguinte para cada um deles:

5 - Determine a coordenada angular de q_k e use-a como uma chave para pesquisar um par de pontos de fronteira da casca convexa que não sejam colineares com q_k. Isto é uma pesquisa linear requerendo uma média de $n_k/2$, com $n_n = n_b n$, comparações escalares, onde n_b é o número de pontos de fronteira da casca convexa de $\langle q_1, \ldots, q_{k-1} \rangle$.

6 - Tendo achado dois pontos de fronteira aos quais q_k pode ser conectado, liga-se q_k a esses pontos de modo a formarem um novo triângulo armazenando os pontos e identificando os lados para serem testados para possível permuta de identificação. Se a

decisão for de permuta de lados, identifique os dois lados que tem q_k em comum, contrua dois novos triângulos ligando cada um destes dois lados a um novo ponto. Quando não houver mais nada armazenado, tenta-se conectar q_k com algum ponto de fronteira vizinho. Se isto for possível, então começa-se novamente a transformação e teste, iniciando com o lado oposto de q_k transformando-o em um novo triângulo. Quando q_k não puder ser conectado com nenhum dos pontos de fronteira, o processamento de q_k estará completo.

A figura 4.16 mostra um conjunto S constituído de 26 pontos num plano, as figuras 4.16a1 a 4.16d mostram alguns passos da triangulação de Lawson e, a figura 4.17 é a malha triangular construída com o conjunto S da figura 4.16.

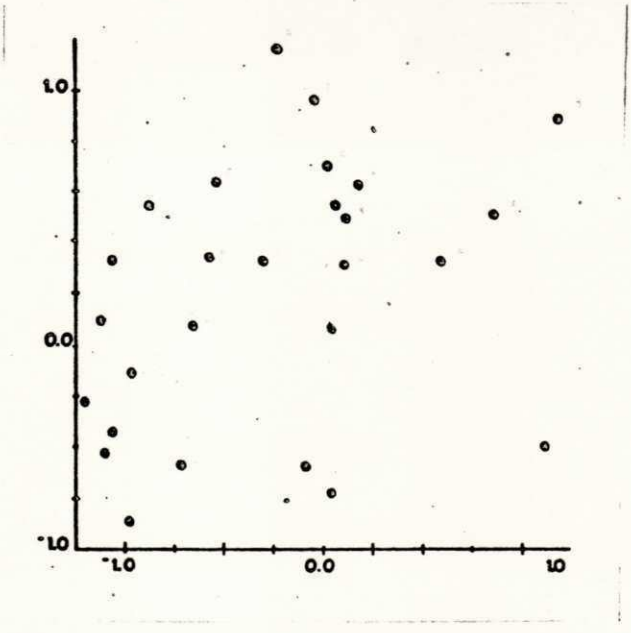

Figura 4.16 - Conjunto de 26 pontos (x_i, y_i).

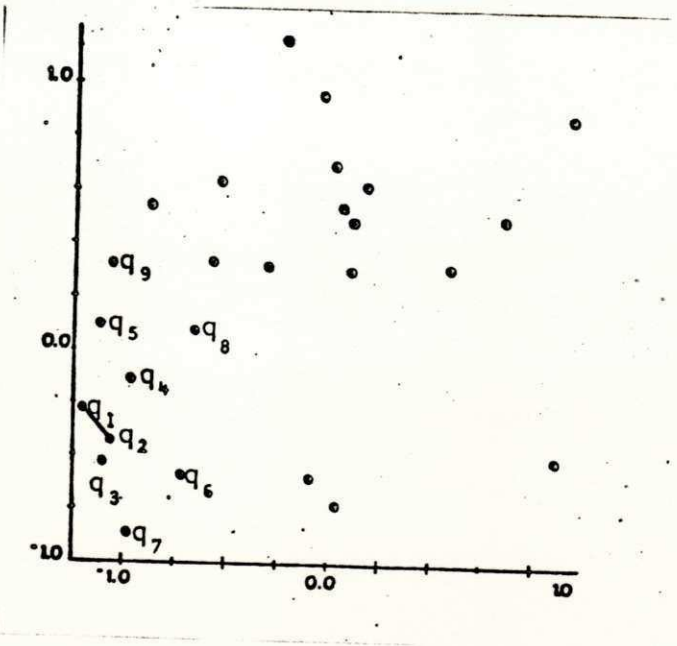

Figura 4.16-a1

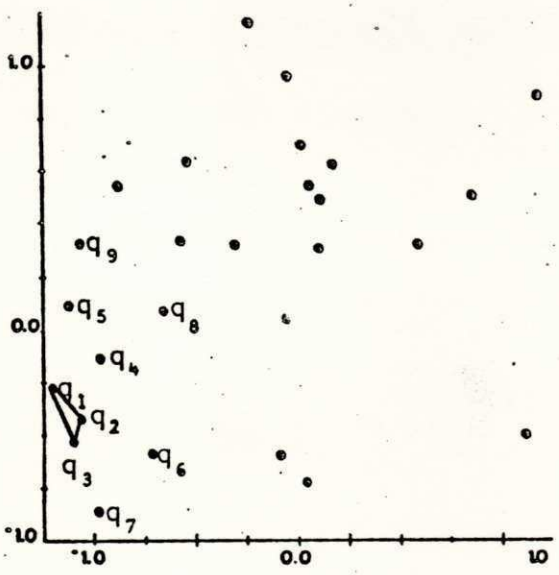

Figura 4.16-a2

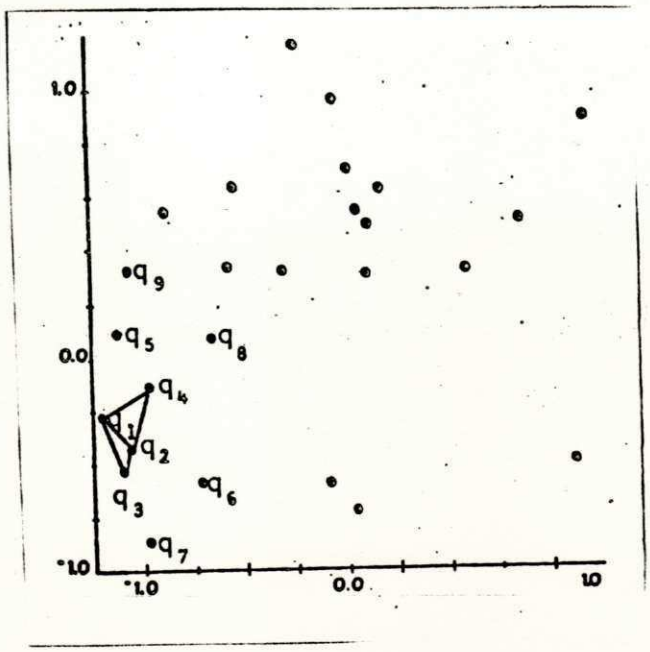

Figura 4.16-b

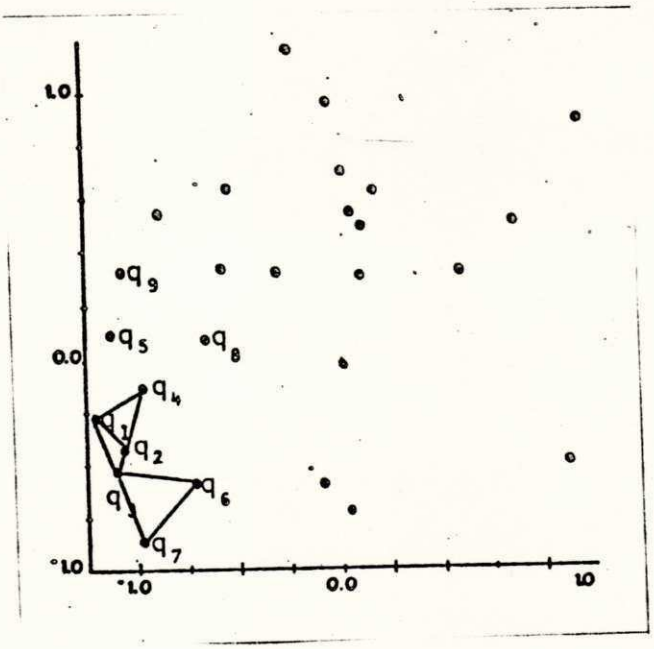

Figura 4.16-c

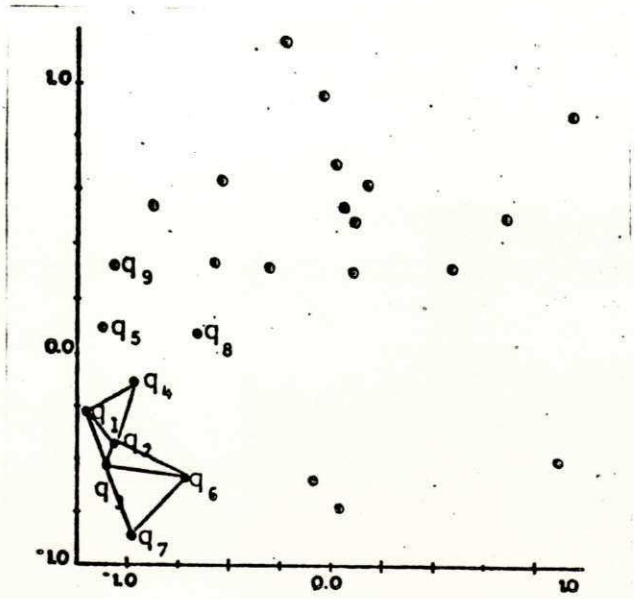

Figura 4.16-d

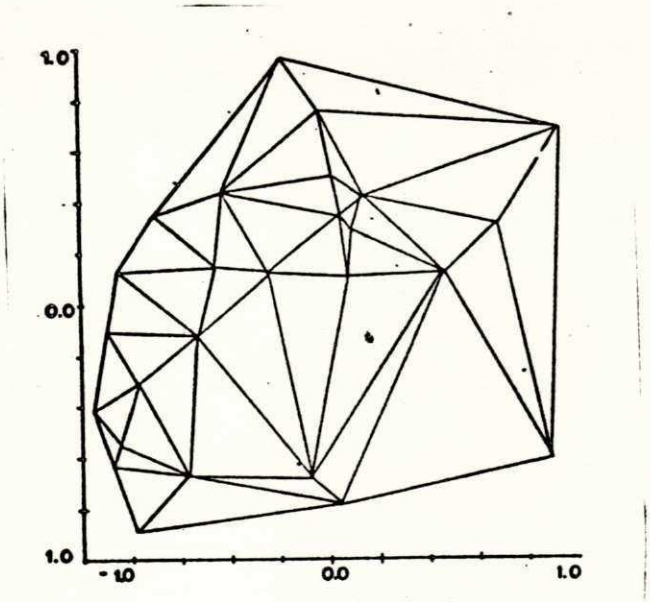

Figura 4.17 - Malha triangular de Lawson

CAPÍTULO V

ALGORITMOS PARA PROBLEMAS NÃO LINEARES (PNL)

O PNL caracteriza-se por não possuir um método geral de resolução, São muitos os algoritmos e quase sempre voltados para problemas específicos, explorando características diversas tais como continuidade, unimodalidade, diferenciabilidade de primeira ordem e/ou de segunda ordem.

Uma outra característica é a inexistência de critérios absolutos para a comparação entre os vários algoritmos existentes (Mateus e Luna [13]). O que se faz é comparar algumas características não definidas precisamente, tais como: simplicidade computacional, tempo de máquina

necessário para atingir um ponto de mínimo a partir de um ponto inicial qualquer, memória necessária, rapidez de convergência, sensibilidade a erros computacionais.

Neste capítulo apresentamos um resumo de uma pesquisa bibliográfica para tentar dar uma idéia geral de problemas de aproximação não linear.

Seja $Y : R^n \longrightarrow R^n$ uma função de l variáveis reais t_i (i = 1, ..., l) e n parâmetros reais x_j (j = 1, ..., n), $t = [t_1, \ldots, t_l]^T$ e $x = [x_1, \ldots, x_n]^T$. Suponha que são dados m variáveis t_i (i = 1, ..., m), com m números reais y_i (i = 1, ..., m).

Sejam

$$Y(x) = [Y_1(x), \ldots, Y_m(x)]^T \qquad (5.1)$$

onde $Y_i(x) = Y(t_i; x$ com (i = 1, ..., m) e $y = [y_1, y_2, \ldots]^T$.

O problema que queremos resolver é minimizar $S : R^n \longrightarrow R$ dado por:

$$S(x) = ||Y(x) - y|| \qquad (x \text{ em } R_n)$$

onde se || || for euclidiana teremos um problema de quadrados mínimos; se || || for norma l_1 será um problema de aproximação l_1 e será tratado na seção 5.2; se || || for norma infinita não será tratado pois nada foi encontrado a respeito de aproximação em norma infinita. Convém, porém, ressaltar que quase tudo o que se refere a aproximação não linear é em relação a norma 2.

5.1 - Quadrados Mínimos

Seja $f : R^n \longrightarrow R^m$ definida por

$$f(x) = Y(x) - y \qquad (x \text{ em } R^n) \tag{5.2}$$

então

$$S(x) = f(x)^T f(x) \tag{5.3}$$

e usando a notação $\partial_i f(x) = \partial f(x) / \partial x_i$

$$\partial_i S(x) = 2 \sum_{j=1}^{m} f_i(x) \partial_i f_j(x) \qquad (i = 1, \ldots, n). \tag{5.4}$$

Uma condição necessária para que x^* seja um mínimo de S é que $\partial_i S(x^*) = 0 \quad (i = 1,\ldots,n)$.

Por conseguinte x^* satisfaz o sistema de equações

$$\sum_{j=1}^{m} f_j(x)\partial_i f_j(x) = 0 \quad (i = 1, \ldots, n) \qquad (5.5)$$

O sistema de equações (5.5) é chamado sistema normal para o problema de quadrados mínimos. A matriz Jacobiana A(x) de f, função de x, é a matriz m por n com elementos $a_{ij}(x)$, dados por

$$a_{ij}(x) = \partial_j f_i(x) \qquad (i = 1, \ldots, m \; ; \; j = 1, \ldots, n).$$

Em termos da matriz Jacobiana, o sistema de equações normal é

$$A(x)^T f(x) = 0 \; .$$

Por (5.4) o gradiente F(x) de S para x é dado por

$$F(x) = 2A(x)^T f(x) \qquad (5.6)$$

Portanto o sistema de equações normal pode ser escrito como:

$$F(x) = 0 \qquad (5.7)$$

Em geral o sistema de equações normal é não linear. A maioria dos métodos para resolver o problema dos quadrados mínimos são métodos iterativos para resolver (5.7). Note que a solução de (5.7) é um ponto crítico de S mas não é necessariamente um mínimo de S. Um método para resolver o problema de quadrados mínimos para achar uma solução de (5.7) pode originar um ponto de sela e assim falhar.

O problema de resolução de sistemas de equações não lineares tem uma certa relação com o problema de quadrados mínimos. Seja $p : R^n \longrightarrow R$ e que tem um mínimo global nulo em R, e $q : R^n \longrightarrow R^n$ é dado. Seja $r : R^n \longrightarrow R$ definido por

$$r(x) = p(q(x)) \qquad (x \text{ em } R^n)$$

Se o sistema de equações $q(x) = 0$ tiver uma solução x^* então x^* é um mínimo de r. Note que $q(x) = 0$ pode ter várias soluções, cada uma das quais é um mínimo de r. Assim algum método para otimização sem restrições pode ser aplicado para a função objetivo r para obter a solução de $q(x) = 0$.

Agora se p for definido por .

$$p(u) = ||u|| \qquad (u \text{ em } R^n) \qquad (5.8)$$

onde $||.||$ é alguma norma em R^n então $p(u) \geq 0$ e $p(u) = 0$ se e somente se $u = 0$. Então p definido em (5.8) tem um mínimo global $u^* = 0$ em R^n. Por conseguinte se x^* for um zero de q então pode ser uma estimativa do mínimo r definido por

$$r(x) = ||q(x)||$$

em particular, se $||.||$ for a norma Euclidiana então temos

$$r(x) = q(x)^T q(x). \qquad (5.9)$$

Se q não tiver zeros, uma solução do problema de quadrados mínimos de $q(x) = 0$ ainda pode ser achado minimizando r definido por (5.9).

O Método de Gauss-Newton

Se o sistema de equações normal (5.7) for não linear, então um método iterativo deve ser usado para resolvê-lo. Se a matriz Jacobiana B de F for conhecida, onde os elementos de B são dados por

$$b_{ij}(x) = \partial_j F_i(x) \quad (i,j = 1, \ldots, n) \qquad (5.10)$$

onde F(x) é definido por (5.6), então o método de Newton para a solução do sistema de equações não lineares pode ser usado para resolver (5.7) se uma estimativa inicial $x^{(0)}$ de uma solução x^* for conhecida. O método de Newton para resolver (5.7) consiste na geração da seqüência $(x^{(k)})$ a partir de

$$x^{(k+1)} = x^{(k)} - B(x^{(k)})^{-1} F(x^{(k)}) \qquad (k = 0, 1, \ldots)$$

Para F definida por (5.10) temos

$$\partial_j F_i(x) = 2 \sum_{l=1}^{m} (\partial_j f_l(x), \partial_i f_l(x) + f_l(x) \partial_j \gamma_i f_l(x)) \qquad (5.11)$$

$$(i, j = 1, \ldots, n)$$

de modo que para computar $B(x^{(k)})$ é necessário computar mn derivadas parciais de primeira ordem $\partial_i f_l(x^{(k)})$ $(i = 1, \ldots, n;$ $l = 1, \ldots, m)$ e $mn(n+1)/2$ derivadas parciais de segunda ordem $\partial_j \gamma_i f_l(x^{(k)})$ $(i, j = 1, \ldots, n ; l = 1, \ldots, m)$. Assim o método de Newton requer m avaliações de funções f_l, mn avaliações de funções $\partial_i f_l$, e $mn(n+1)/2$ avaliações de funções $\partial_j \gamma_i f_l$. Se o valor de $S(x^*)$ for zero ou muito pequeno, então por (5.3), o valor de $f_l(x^*)$ $(l = 1, \ldots, m)$ é também zero ou muito pequeno. Assim se x for suficientemente próximo de x^*, o termo $f_l(x)\, \partial_j \gamma_i f_l(x)$ em (5.11) é desprezível se as segundas derivadas parciais de f_l forem limitadas. Se $S(x^*)$ for muito pequeno ou for igual a zero, podemos aproximar $\partial_j F_i(x)$ por

$$\partial_j F_i(x) \approx 2 \sum_{l=1}^{m} \partial_j f_l(x)\, \partial_i f_l(x) \qquad (5.12)$$

para x suficientemente próximo de x^*.

Se

$$y = Y(t;x^*),$$

no caso dos quadrados mínimos, então $f(x^*)$ definido por (5.2) tem elementos que são muito pequenos podendo realmente ter valor zero. Se, também, uma estimativa $x^{(0)}$ próxima de x^* for conhecida, então poderia esperar que o método de Newton com aproximação de B pelo uso de (5.12) seja efetivo. Essa é a idéia do método de Gauss-Newton. Maiores detalhes sobre resolução de problemas de quadrados mínimos ver Mateus [13], Stoer [15] e Wolfe [16].

5.2 - Problemas de Aproximação l_1 Não Linear

Nós consideramos o problema de minimização em norma l_1. Este problema é definido da seguinte forma:

Problema I - Sejam $f_i : R^n \longrightarrow R$, $i = 1,\ldots,m$ funções continuamente diferenciáveis. Minimize

$$F(x) = ||f(x)|| = \sum_{i=1}^{m} |f_i(x)|,$$

Este tipo de problema aparece em uma variedade de áreas entre as quais comunicação digital e técnicas de aproximação.

Nossa atenção aqui é focada no problema de aproximação l_1 não linear que pode ser enunciado como se segue.

Problema II - Seja uma função real $f(x)$ definida num conjunto discreto $X = \langle x_1, \ldots, x_n \rangle$, que define uma função aproximação $k(A,x)$ de algum conjunto $A = \langle a_1, \ldots, a_n \rangle$ de números reais. Encontrar A^* que minimize

$$F(A) = \sum_{i=1}^{m} |f(x_i) - k(A,x_i)|.$$

Se $k(A,.)$ for uma função linear em A, tem-se um problema de aproximação l_1 linear, resolvido satisfatoriamente por Barrodale e Roberts [7]. Assim, vamos nos restringir ao caso de $k(A,.)$ ser não linear em A. Este problema não foi ainda resolvido satisfatoriamente. Note-se que o problema II é um caso especial do problema I.

5.2.1 - O algoritmo de El-Attar e Dutta

Descrevemos um algoritmo para problemas de minimização não linear em norma l_1 proposto por El-Attar et alli [10]. O algoritmo básico usa uma técnica iterativa para minimizar uma função $P(x,\varepsilon)$ com valores decrescentes de ε.

Algoritmo 5.2.1

O algoritmo básico

Passo 1 - Escolha x_0 em R^n, $\varepsilon_1 > 0$ (número pequeno), defina $k = 1$.

Passo 2 - Minimize $P(x,\varepsilon_k)$ dada pela equação

$$P(x,\varepsilon_k) = \sum_{i=1}^{m} [f_i^2(x) + \varepsilon_k]^{1/2}, \quad \varepsilon_k > 0$$

Chame a solução de x_k^*

Passo 3 - Calcule $\varepsilon_{k+1} = \varepsilon_k / L$ onde L é um número pré-especificado maior do que 1.

Passo 4 - Se $\varepsilon_{k+1} \leq \alpha$ e/ou se $||x_k^* - x_{k-1}^*||_1 \leq \beta$ onde α e β são números pequenos pré-estabelecidos dependendo da precisão desejada, PARE. Caso contrário faça $k := k+1$ e volte ao passo 2.

Para valores pequenos de ε_k, o mal condicionamento do problema $P(x,\varepsilon)$ aumenta, isto é, a função $P(x,\varepsilon)$ ainda que teoricamente diferenciável, torna-se em alguns casos não diferenciável na prática. Este mal condicionamento ocorre quando pelo menos um dos f_i's for próximo de zero ou identicamente nulo. Como resultado, a minimização da função $P(x,\varepsilon)$, emprega um método de gradiente, utilizando um grande número de avaliações de funções. Também, para alguns problemas, o algoritmo, embora direcionado para um ponto muito próximo do mínimo, não converge para o mínimo com a precisão desejada.

O Hessiano da função objetivo $P(x,\varepsilon)$ é dado por

$$\nabla^2 P(x,\varepsilon) = \sum_{i=1}^{m} \{[f_i^2(x) + \varepsilon]^{-1/2} [f_i(x)\, \nabla^2 f_i(x) + \nabla f_i(x)\, \nabla^T f_i(x)] - [f_i^2(x) + \varepsilon]^{-3/2} f_i^2(x)\, \nabla f_i(x)\, \nabla^T f_i(x)\}$$

onde $\nabla^2 = \dfrac{\partial^2}{\partial x^2} + \dfrac{\partial^2}{\partial y^2} + \dfrac{\partial^2}{\partial z^2}$.

É claro na expressão acima que se pelo menos um dos f_i's tornar-se zero para ε tendendo a 0, que a inversa do Hessiano torna-se singular.

Pode-se pensar nessas dificuldades como análoga das dificuldades ocorridas em alguns dos métodos de função penalidade para problemas de otimização com restrição onde, para um ponto ótimo, a função objetivo estendida não deve ser

necessariamente diferenciável.

A escolha de um valor inicial apropriado para ε_1 é relacionado com o escalamento do problema particular. Em geral, a experiência numérica com o algoritmo mostra que um bom valor de ε é a décima parte do maior valor absoluto dos f_i's ou um décimo da média dos valores absolutos dos f_i's, isto é,

$$\varepsilon_1 = \frac{1}{10} \max_i |f_i(x_0)| \qquad (5.13)$$

ou

$$\varepsilon_1 = \frac{1}{10m} \sum_{i=1}^{m} |f_i(x_0)|. \qquad (5.14)$$

Em alguns casos raros, a diferença dos valores de f_i, $i = 1, \ldots, m$ pode ser muito grande. Neste caso adicionamos um valor apropriado ε_{1i} com todo f_i ou todo conjunto particular de f_i's, isto é, a função $P(x,\varepsilon)$ é escolhida para ser

$$P(x,\varepsilon) = \sum_{i=1}^{m} [f_i^2(x) + \varepsilon_i]^{1/2}$$

onde

$$\varepsilon_i = \varepsilon - c_i$$

e $c_i > 0$ são as constantes de peso.

Para evitar algumas das dificuldades quando ε_k torna-se próximo de zero, usamos a técnica de extrapolação de Fiacco e MacCormick (cf. El-Attar [10]) a qual pode ser resumida como segue:

Suponha que a função $P(x,\varepsilon)$ foi minimizada para $\varepsilon_1 > \varepsilon_2 > \ldots \varepsilon_k > 0$ para $x_1^*, x_2^*, \ldots, x_k^*$. Então, todo x_j^*, $j = 1,2,\ldots,k$ pode ser expresso como um polinômio em $\varepsilon^{1/2}$ dado por

$$x_j^* = \sum_{i=0}^{k-1} a_i (\varepsilon_j^{1/2})^i,$$

onde os a_i's são vetores de n componentes. Através de algum raciocínio teórico, podemos obter uma estimativa de $\hat{x}_{k+1}$ para x^*_{k+1}, supondo que o x_k mínimo foi obtido. Esta estimativa é representada como

$$\hat{x}_{k+1} = \sum_{i=1}^{k} a_{i-1} \left(\frac{\varepsilon_1}{L^k} \right)^{(i-1)/2} \qquad (5.15)$$

Note que os a_i's são únicos e não necessitam ser calculados explicitamente. Para estimativas lineares, isto é, dados dois mínimos prévios, somente os primeiros dois termos aparecem na última equação.

Resultados numéricos (cf. El-Attar[10]) mostram que esta aproximação é geralmente mais lenta do que um método do gradiente utilizando a extrapolação. A técnica de extrapolação de Fiacco e McCormick (cf. El-Attar[10]) acelera a convergência.

Apresentamos uma versão melhorada do algoritmo básico utilizando a técnica de extrapolação de Fiacco e McCormick.

Algoritmo de Fiacco e McCormick

Algoritmo 5.2.2

Passo 1 - Escolha $\tilde{x}_1 \in R^n$, obtenha ε_1 como em (5.13) ou (5.14), e defina k = 1.

Passo 2 - Minimize $P(x, \varepsilon_k)$ com $\tilde{x}_k$ como ponto inicial. Denote a solução por x_k^*.

Passo 3 - Calcule $\varepsilon_{k+1} = \varepsilon_k / L$, onde L é um número pré-definido maior que 1.

Passo 4 - Se $\varepsilon_{k+1} \leq \alpha$ e/ou $||x_k^* - x_{k-1}^*||_1 \leq \beta$, onde α e β são números pequenos pré-definidos dependendo da precisão desejada, PARE.

Passo 5 - Se k = 1, $\hat{x}_{k+1} = x_k^*$. Se $k \geq 2$, obtenha uma estimativa $\hat{x}_{k+1}$ de x_{k+1}^* pela técnica de extrapolação (conforme (5.15)).

Passo 6 - Faça k := k + 1 e volte para o passo 2.

Uma aproximação similar a usada por Abdelmalek (cf. El-Attar [10]) para problemas de aproximação l_1 linear, pode ser aplicado ao Problema I. Seja

$$Z(x,p) = (\sum_{i=1}^{m} |f_i(x)|^p)^{1/p}, \quad p > 1,$$

onde $f_i(.)$ é definido como no problema I. Para $p > 1$ a sequência de soluções obtidas pela minimização da função diferenciável $Z(x,p)$ convergirá para a solução l_1 quando p se aproxima de 1.

Suponha que para todo i as funções f_i's são lineares em algumas das variáveis x_j, $j = 1, 2, \ldots, l$, e não linear em outras x_j, $j = l+1, \ldots, n$, e suponha que o seguinte procedimento é aplicado:

1 - Fixe as variáveis não lineares e resolva o problema l_1 usando o algoritmo de Barrodale e Roberts [8].

2 - Fixe as variáveis lineares obtidas no passo anterior e resolva o problema não linear usando o algoritmo melhorado (algoritmo 5.2.2).

3 - Repita o processo até obter a convergência.

Resultados numéricos mostram que esta aproximação conduz em todos os casos para um mínimo local.

CAPÍTULO VI

TESTES COM ALGORITMOS IMPLEMENTADOS

Neste capítulo apresentamos resultados numéricos fornecidos por alguns algoritmos para solução do problema linear em normas 1 e infinita. Não apresentamos nenhum resultado de algoritmo que utiliza norma l_2 porque os resultados em norma l_2 são bem conhecidos.

Após a investigação de algoritmos para resolver o problema linear constatamos a existência de um grande número de algoritmos para este fim.

Por esta razão nos limitamos a citar alguns exemplos esperando que estes possam servir para orientar estudos posteriores.

Os programas que usamos foram baseados naqueles já publicados e que foram transcritos para o Fortran-66 e foram testados tanto no computador IBM 4341, cujo sistema operacional é o VM 370 controlador do OS/VS1 sob o qual roda o compilador WATFIV, no computador IBM 4381 cujo sistema operacional é o VM 370 usando CMS sob o qual roda o compilador FortranVs como também, no micro computador ITAUTEC I-7000 PCxt II com o compilador FORTRAN77 da MICROSOFT versão 3.3.

6.1 - O Algoritmo de Barrodale e Roberts

O algoritmo foi descrito na seção 3.3.

O algoritmo é implementado em forma de uma subrotina principal: CL1.

A subrotina CL1 usa a modificação do método simplex da programação linear para calcular uma solução l_1 de um sistema com M equações lineares a N incógnitas

$$Ax = b$$

sujeito a l restrições lineares de igualdade

$$Cx = d$$

e k restrições lineares de desigualdade

$$Ex \leq f.$$

O algoritmo foi testado com os seguintes problemas:

Problema 1

$$A = \begin{bmatrix} 2 & 0 & 1 & 3 & 1 \\ 7 & 4 & 4 & 15 & 7 \\ 9 & 4 & 7 & 20 & 6 \\ 2 & 2 & 1 & 5 & 3 \\ 9 & 3 & 2 & 14 & 10 \\ -4 & 5 & 0 & 9 & 9 \\ 4 & 4 & 9 & 17 & -1 \\ 1 & 6 & 2 & 9 & 5 \end{bmatrix}$$

$$b = [7,4,7,4,0,4,9,6]^T$$

$$C = \begin{bmatrix} 0 & 4 & 5 & 9 & -1 \\ 3 & 2 & 7 & 12 & -2 \\ 3 & 6 & 12 & 21 & -3 \end{bmatrix}$$

$$d = [5,1,6]^T$$

$$E = \begin{bmatrix} 0 & 3 & 6 & 9 & -3 \\ 6 & 2 & 4 & 12 & 4 \end{bmatrix}$$

$$f = [5,6]^T$$

onde as matrizes A, b, C, d, E e f foram transformadas numa única matriz Q:

$$Q = \begin{bmatrix} A & b \\ C & d \\ E & f \end{bmatrix}$$

A solução obtida foi:

$$x = \begin{bmatrix} 0.0000000 \\ 1.6956520 \\ 0.0000000 \\ -0.1956521 \\ 0.0217390 \end{bmatrix}$$

com a norma do erro = 0.002665216, onde a norma do erro é a soma dos valores absolutos dos resíduos.

Problema 2

$$A = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \\ 1 & 5 \end{bmatrix}$$

$$b = \begin{bmatrix} 2 \\ 2 \\ 3 \\ 4 \\ 3 \end{bmatrix}$$

$$C = [\ 1 \quad 6\ ]$$

$$d = [\ 5\ ]$$

$$E = [\ 1 \quad 1\ ]$$

$$f = [\ 3\ ]$$

A solução obtida foi:

$$x = \begin{bmatrix} 0.1000001E+01 \\ 0.66666666E+00 \end{bmatrix}$$

com a norma erro = 0.0233333.

6.2 - O Algoritmo de Abdelmalek

O algoritmo desenvolvido por Nabih N. Abdelmalek da Divisão de Engenharia Elétrica do Conselho Nacional de Pesquisa do Canadá, Ottawa[1] foi descrito na seção 3.2.

A implementação utiliza duas subrotinas: L1 e PSLV.

A subrotina L1 resolve um sistema hiperdeterminado de equações lineares na norma l_1 pelo uso de um algoritmo simplex dual (ver seção 3.2) para programação linear.

O sistema tem a forma

$$Ax = b.$$

A solução deste sistema é um vetor x^* de tamanho m que minimiza

$$\sum_{i=1}^{n} |r_i|$$

sendo que

r_i é o i-ésimo resíduo e é dado por

$$r_i = a_{i,1}x_1 + a_{i,2}x_2 + \ldots + a_{i,m}x_m - b_i.$$

A subrotina PSLV é chamada pela subrotina L1 e resolve o sistema quadrado não singular de equações lineares

$$\mathbf{Px = b},$$

ou o sistema quadrado não singular de equações lineares

$$\mathbf{P^Tx = b},$$

onde P é uma matriz triangular superior, b e x são vetores.

A implementação foi testada com os seguintes problemas:

Problema 1

Matriz A

A matriz A é de 201 x 1 com todos os elememtos iguais a 1.

Vetor b

Elementos de b

0.00000,	0.01960,	0.03842,	0.05647,	0.07377,	0.09033,
0.10618,	0.12131,	0.13576,	0.14954,	0.16266,	0.17513,
0.18698,	0.19822,	0.20886,	0.21893,	0.22842,	0.23737,
0.24577,	0.25366,	0.26103,	0.26792,	0.27432,	0.28026,
0.28574,	0.29079,	0.29541,	0.29961,	0.30342,	0.30684,
0.30988,	0.31256,	0.31490,	0.31689,	0.31856,	0.31991,
0.32096,	0.32171,	0.32219,	0.32239,	0.32233,	0.32202,
0.32147,	0.32069,	0.31969,	0.31848,	0.31706,	0.31545,
0.31366,	0.31169,	0.30956,	0.30727,	0.30482,	0.30223,
0.29951,	0.29666,	0.29368,	0.28741,	0.28411,	0.28072,
0.27725,	0.27369,	0.27006,	0.26636,	0.26260,	0.25878,
0.25490,	0.25098,	0.24701,	0.24301,	0.23897,	0.23490,
0.23081,	0.22670,	0.22257,	0.21843,	0.21428,	0.21012,
0.20597,	0.20181,	0.19766,	0.19352,	0.18938,	0.18526,
0.18116,	0.17708,	0.17301,	0.16897,	0.16496,	0.16098,
0.15702,	0.15310,	0.14921,	0.14535,	0.14154,	0.13776,
0.13402,	0.13032,	0.12667,	0.12306,	0.11950,	0.11598,
0.11250,	0.10908,	0.10571,	0.10238,	0.09910,	0.09588,
0.09271,	0.08958,	0.08651,	0.08350,	0.08053,	0.07762,
0.07476,	0.07196,	0.06921,	0.06651,	0.06387,	0.06128,
0.05874,	0.05626,	0.05383,	0.05145,	0.04913,	0.04685,
0.04464,	0.04247,	0.04035,	0.03829,	0.03627,	0.03431,
0.03240,	0.03054,	0.02872,	0.02696,	0.02524,	0.02357,
0.02195,	0.02037,	0.01884,	0.01735,	0.01591,	0.01452,
0.01316,	0.01185,	0.01059,	0.00936,	0.00817,	0.00703,
0.00592,	0.00485,	0.00382,	0.00283,	0.00187,	0.00095,
0.00007,	-0.00078,	-0.00160,	-0.00238,	-0.00313,	-0.00385,
-0.00453,	-0.00519,	-0.00582,	-0.00642,	-0.00698,	-0.00753,
-0.00804,	-0.00583,	-0.00899,	-0.00943,	-0.00984,	-0.01023,
-0.01059,	-0.01094,	-0.01126,	-0.01155,	-0.01183,	-0.01209,
-0.01233,	-0.01255,	-0.01275,	-0.01293,	-0.01310,	-0.01325,
-0.01338,	-0.01350,	-0.01360,	-0.01369,	-0.01376,	-0.01382,
-0.01387,	-0.01390,	-0.01392,	-0.01393,	-0.01393,	-0.01392,

-0.01389, -0.01386

Vetor solução

X(I) = [0.0]

Número de iterações

ITER = 1

Norma do erro = 26.52003

onde a norma do erro é a soma dos valores absolutss dos resíduos.

Problema 2

Matriz A

dimensão da matriz - 201 por 2

onde a 1ª coluna é composta de 1's e a segunda coluna composta dos seguintes elementos:

0.08, 0.1, 0.12, 0.14, 0.16, 0.18, 0.2, 0.22, 0.24, 0.26, 0.28,
0.3, 0.32, 0.34, 0.36, 0.38, 0.4, 0.42, 0.44, 0.46, 0.48, 0.5,
0.52, 0.54, 0.56, 0.58, 0.6, 0.62, 0.64, 0.66, 0.68, 0.7, 0.72,
0.74, 0.76, 0.78, 0.8, 0.82, 0.84, 0.86, 0.88, 0.9, 0.92, 0.94,
0.96, 0.98, 1.0, 1.02, 1.04, 1.06, 1.08, 1.1, 1.12, 1.14, 1.16,
1.18, 1.2, 1.22, 1.24, 1.26, 1.28, 1.3, 1.32, 1.34, 1.36, 1.38,
1.4, 1.42, 1.44, 1.46, 1.48, 1.5, 1.52, 1.54, 1.56, 1.58, 1.6,
1.62, 1.64, 1.66, 1.68, 1.7, 1.72, 1.74, 1.76, 1.78, 1.8, 1.82,
1.84, 1.86, 1.88, 1.9, 1.92, 1.94, 1.86, 1.98, 2.0, 2.02, 2.04,
2.06, 2.08, 2.1, 2.12, 2.14, 2.16, 2.18, 2.2, 2.22, 2.24, 2.26,

2.28, 2.3, 2.32, 2.34, 2.36, 2.38, 2.4, 2.42, 2.44, 2.46, 2.48,
2.5, 2.52, 2.54, 2.56, 2.58, 2.6, 2.62, 2.64, 2.66, 2.68, 2.7,
2.72, 2.74, 2.76, 2.78, 2.8, 2.82, 2.84, 2.86, 2.88, 2.9, 2.92,
2.94, 2.96, 2.98, 3.0, 3.02, 3.04, 3.06, 3.08, 3.1, 3.12, 3.14,
3.16, 3.18, 3.2, 3.22, 3.24, 3.26, 3.28, 3.3, 3.32, 3.34, 3.36,
3.38, 3.4, 3.42, 3.44, 3.46, 3.48, 3.5, 3.52, 3.54, 3.56, 3.58,
3.6, 3.62, 3.64, 3.66, 3.68, 3.7, 3.72, 3.74, 3.76, 3.78, 3.8,
3.82, 3.84, 3.86, 3.88, 3.9, 3.92, 3.94, 3.96, 3.98, 4.0.

Vetor b

Elementos de b

0.00000, 0.01960, 0.03842, 0.05647, 0.07377, 0.09033,
0.10618, 0.12131, 0.13576, 0.14954, 0.16266, 0.17513,
0.18698, 0.19822, 0.20886, 0.21893, 0.22842, 0.23737,
0.24577, 0.25366, 0.26103, 0.26792, 0.27432, 0.28026,
0.28574, 0.29079, 0.29541, 0.29961, 0.30342, 0.30684,
0.30988, 0.31256, 0.31490, 0.31689, 0.31856, 0.31991,
0.32096, 0.32171, 0.32219, 0.32239, 0.32233, 0.32202,
0.32147, 0.32069, 0.31969, 0.31848, 0.31706, 0.31545,
0.31366, 0.31169, 0.30956, 0.30727, 0.30482, 0.30223,
0.29951, 0.29666, 0.29368, 0.29060, 0.28741, 0.28411,
0.28072, 0.27725, 0.27369, 0.27006, 0.26636, 0.26260,
0.25878, 0.25490, 0.25098, 0.24701, 0.24301, 0.23897,
0.23490, 0.23081, 0.22670, 0.22257, 0.21843, 0.21428,
0.21012, 0.20597, 0.20181, 0.19766, 0.19352, 0.18938,
0.18526, 0.18116, 0.17708, 0.17301, 0.16897, 0.16496,
0.16098, 0.15702, 0.15310, 0.14921, 0.14535, 0.14154,
0.13776, 0.13402, 0.13032, 0.12667, 0.12306, 0.11950,
0.11598, 0.11250, 0.10908, 0.10571, 0.10238, 0.09910,
0.09588, 0.09271, 0.08958, 0.08651, 0.08350, 0.08053,
0.07762, 0.07476, 0.07196, 0.06921, 0.06651, 0.06387,
0.06128, 0.05874, 0.05626, 0.05383, 0.05145, 0.04913,
0.04685, 0.04464, 0.04247, 0.04035, 0.03829, 0.03627,
0.03431, 0.03240, 0.03054, 0.02872, 0.02696, 0.02524,
0.02357, 0.02195, 0.02037, 0.01884, 0.01735, 0.01591,

0.01452, 0.01316, 0.01185, 0.01059, 0.00936, 0.00817,
0.00703, 0.00592, 0.00485, 0.00382, 0.00283, 0.00187,
0.00095, 0.00007, -0.00078, -0.00160, -0.00238, -0.00313,
-0.00385, -0.00453, -0.00519, -0.00582, -0.00642, -0.00698,
-0.00753, -0.00804, -0.00583, -0.00899, -0.00943, -0.00984,
-0.01023, -0.01059, -0.01094, -0.01126, -0.01155, -0.01183,
-0.01209, -0.01233, -0.01255, -0.01275, -0.01293, -01310,
-0.01325, -0.01338, -0.01350, -0.01360, -0.01369, -0.01378,
-0.01382, -0.01387, -0.01390, -0.01392, -0.01393, -0.01393,
-0.01392, -0.01389, -0.01386.

Vetor solução

$$X(I) = \begin{bmatrix} 0.0 \\ \\ -0.00347 \end{bmatrix}$$

Número de iterações

ITER = 2

Norma do erro = 27.04250.

6.3 - O Algoritmo de Duris

Este algoritmo foi desenvolvido por Charles S. Duris da Universidade de Drexel, Philadelphia [9] e foi descrito na seção 3.5.

A implementação é feita em duas subrotinas principais: DCSINT e DCSSMO.

A subrotina DCSINT constrói a função do spline cúbico discreto sobre o intervalo $[\tau_1, \tau_n]$ que interpola o dado (τ_j, b_j) para $j = 1, 2, \ldots, n$, e satisfaz uma das três condições dos pontos finais apresentadas na seção 3.5.1.

A subrotina DCSSMO constrói a função $S(t)$ que aproxima os dados (τ_i, b_i) para $i = 1, 2, \ldots, n$ de modo que

$$\rho(f(t)) = \rho \sum_{i=1}^{n} W_i[f(\tau_i) - b_i]^2 + \sum_{i=1}^{m-1} [\nabla \Delta f(t_j)]^2$$

é minimizado por $f(t) = S(t)$. Este $S(t)$ volta a ser um spline cúbico natural ($\nabla \Delta S(\tau_1) = \nabla \Delta S(\tau_n) = 0$). Os W_i's são pesos positivos especificando a importância relativa dos dados, e $\rho > 0$ é um parâmetro que permite o algoritmo se comportar como de aproximação ou de interpolação conforme ρ seja pequeno ou grande (para $\rho \longrightarrow 0$, tende a aproximar a derivada segunda).

A implementação foi testada com o seguinte problema:

APROXIMAÇÃO

Seja

$$S_i = \tilde{b}_i + c_i(t - \tau_i) + x_i(t - \tau_i)^2 + d_i(t - \tau_i)^3 \qquad (7.2)$$

a função spline definida em $[\tau_i, \tau_{i+1}]$.

Dados os valores de b_i (os valores da função b definida no intervalo $[\tau_1, \tau_n]$ a ser aproximada), de τ_i (os nodos), de W_i (os pesos), o número de pontos (n), h (tamanho do passo) e ρ (parâmetro de aproximação desejada), a subrotina de suavização calcula os valores suavizados de b_i e os coeficientes c_i, x_i e d_i, para serem substituidos na equação 7.1 onde :

$$c_i = \frac{h_i}{3} (2x_i + x_{i+1})$$

$$x_i = x_i - c_i x_{i+1} - d_i x_{i+2} \quad \text{para} \quad 1 \leq i \leq n-1$$

$$x_n = \frac{x_n}{\tilde{b}_n}$$

$$d_i = \frac{(x_{i+1} - x_i)}{(3h_i)} \quad \text{e} \quad d_0 = d_n = 0$$

Problema 1

DADOS DE ENTRADA

n = 7, h = 0.1 e

i	τ_i	b_i	W_i
1	0.5	1.0	2.0
2	0.7	0.5	2.5
3	1.0	2.0	1.0
4	1.5	2.5	1.5
5	2.1	2.0	2.0
6	2.5	1.0	2.0
7	3.0	0.1	2.0

TESTE 1

$\rho = 1$

i	$\tilde{b}_i$	c_i	x_i	d_i
1	1.004575	-2.395454	0.0	-1.524938
2	0.5132843	5.575987	0.9249628	-7.060057
3	1.913113	3.028409	-7.269014	7.421034
4	2.537693	-2.518901	3.862537	-2.002262
5	1.984377	-3.032619	0.2584620	3.205337
6	1.017825	-3.220356	4.104863	-2.736575

TESTE 2

$\rho = 100$

i	$\tilde{b}_i$	c_i	x_i	d_i
1	0.9792708	-29.89449	0.0	690.97448
2	0.5281671	-76.16634	414.5847	-482.6560
3	1.959161	-38.48294	-19.80592	197.9826
4	2.514029	-112.1768	277.1680	-152.7474
5	1.994975	-5.739434	2.222279	14.72617
6	1.000839	-8.433700	19.89368	-13.26245

TESTE 3

ρ = 10000

i	$\tilde{b}_i$	c_i	x_i	d_i
1	0.9999966	-2.958673	0.0	11.46833
2	0.5000083	5.021197	6.880997	-23.17325
3	1.999980	5.653829	-13.97494	9.334736
4	2.500003	-0.8441973	0.02716711	-1.511243
5	2.000000	-2.517030	-0.00003527	0.1065174
6	1.000800	-1.842594	0.1277856	-8.519036

Problema 2

DADOS DE ENTRADA

n = 7, h = 0.1 e

i	τ_i	b_i	W_i
1	10.0	0.42	2.0
2	10.2	0.48	2.5
3	10.4	0.51	1.0
4	10.6	0.52	1.5
5	10.8	0.53	2.0
6	11.0	0.55	2.0
7	11.2	0.58	2.0

TESTE 1

$\rho = 1$

i	$\tilde{b}_i$	c_i	x_i	d_i
1	0.4220587	0.2985946	0.0	-0.6862322
2	0.4762877	0.2376834	-0.4117389	0.8605593
3	0.5142393	0.0168295	0.1045961	0.1540186
4	0.523021	0.0020926	0.1970077	-0.6012945
5	0.5265097	0.1404898	-0.1637687	0.5621337
6	0.5525540	0.1097578	0.1735112	-0.2891856

TESTE 2

$\rho = 100$

i	$\tilde{b}_i$	c_i	x_i	d_i
1	0.4201029	0.4386216	0.0	-0.0343132
2	0.4803766	0.0131477	-0.025879	-0.0019123
3	0.5079927	0.0218992	-0.001353	0.0014331
4	0.5193220	-0.2030681	-0.049335	0.0031282
5	0.5316257	-0.0176240	0.001383	-0.0022908
6	0.5493173	0.1412451	0.091093	-0.1518228

TESTE 3

ρ = 10000

i	$\tilde{b}_i$	c_i	x_i	d_i
1	0.4199999	0.2932050	0.0	0.1699139
2	0.48	0.1509314	0.101948	-0.5331357
3	0.5099999	0.0691721	-0.2179329	0.6104885
4	0.52	0.0303145	0.1483617	-0.2497324
5	0.5299999	-0.0958234	-0.0014775	0.1118329
6	0.55	0.1412498	0.0656220	-0.1093702

TESTE 4

ρ = 100000

i	$\tilde{b}_i$	c_i	x_i	d_i
1	0.42	0.3296895	0.0	-0.7422392
2	0.48	0.2323515	-0.4453431	0.1679326
3	0.51	0.0927214	-0.3445836	0.6548768
4	0.52	0.0270643	0.0483439	0.3316720
5	0.53	0.0581085	0.2473469	-0.1894390
6	0.55	0.1321755	0.1336836	-0.2228063

Neste problema, podemos verificar que, quando ρ = 100000, obtemos o valor de $\tilde{b}_i$ exatamente igual ao valor de b_i.

Problema 3

DADOS DE ENTRADA

$n = 10$, $h = 0.1$ e

i	τ_i	b_i	W_i
1	10.0	29.1	1.0
2	20.0	23.4	1.0
3	30.0	14.9	1.0
4	40.0	4.3	1.0
5	50.0	-7.4	1.0
6	60.0	-28.5	1.0
7	70.0	-36.0	1.0
8	80.0	-2.3	1.0
9	90.0	29.1	1.0
10	100.0	62.6	1.0

TESTE 1

$\rho = 1$

i	$\tilde{b}_i$	c_i	x_i	d_i
1	29.1	-0.5132368	0.0	-0.0005676
2	23.4	-0.6835385	-0.01702922	0.0000383
3	14.89999	-0.0101264	-0.01588009	0.0011144
4	4.300028	-0.9958546	0.01755368	-0.0034968
5	-7.400049	-0.0169393	-0.08735327	0.0045746
6	-28.50002	-0.0206863	0.04988660	0.0081978
7	-35.99989	0.0138862	0.02958208	-0.0097684
8	-2.300056	0.0315512	0.00276764	-0.0004279
9	29.1	0.0341714	-0.01007139	0.0003357

TESTE 2

$\rho = 100$

i	$\tilde{b}_i$	c_i	x_i	d_i
1	29.1	-0.5132360	0.0	-0.0005676
2	23.4	-0.6835374	-0.01702920	0.0000382
3	14.89999	-0.0101264	-0.01588036	0.0011145
4	4.300001	-0.9958519	0.01755475	-0.0034969
5	-7.400001	-0.0169392	-0.08735395	0.0045746
6	-28.5	-0.0206864	0.04988539	0.0081979
7	-36.0	0.0138861	0.02958232	-0.0097685
8	-2.3	0.0315512	0.00276766	-0.0004279
9	29.1	0.0341714	-0.01007154	0.0003357

TESTE 3

$\rho = 10000$

i	$\tilde{b}_i$	c_i	x_i	d_i
1	29.1	-0.5132360	0.0	-0.0005676
2	23.4	-0.6835374	-0.01702920	0.0000382
3	14.9	-0.0101264	-0.01588037	0.0011145
4	4.3	-0.9958519	0.01755476	-0.0034969
5	-7.4	-0.0169392	-0.08735395	0.0045746
6	-28.5	-0.0206864	0.04988539	0.0081979
7	-36.0	0.0138861	0.02958233	-0.0097685
8	-2.3	0.0315512	0.00276766	-0.0004279
9	29.1	0.0341714	-0.01007154	0.0003357

TESTE 4

$h = 0.01$

$\rho = 1$

i	$\tilde{b}_i$	c_i	x_i	d_i
1	29.1	-0.5132365	0.0	-0.0005676
2	23.4	-0.6835304	-0.01702907	0.0000382
3	14.9	-0.0101264	-0.01588274	0.0011147
4	4.3	-0.9958683	0.01756034	-0.0034973
5	-7.4	-0.0169386	-0.08736018	0.0045746
6	-28.5	-0.0206866	0.04988062	0.0081985
7	-36.0	0.0138852	0.02958378	-0.0097690
8	-2.3	0.0315512	0.00276764	-0.0004280
9	29.1	0.0341714	-0.01007243	0.0003357

Para este problema, com $h = 0.1$, obtemos $\tilde{b}_i = b_i$ quando $\rho = 10000$. Porém, se alterarmos o valor de h para 0.01, obtemos $\tilde{b}_i = b_i$ quando $\rho = 1$.

INTERPOLAÇÃO

Seja

$$S_i = b_i + c_i(t - \tau_i) + x_i(t - \tau_i)^2 + d_i(t - \tau_i)^3 \qquad (7.2)$$

Dado o número de pontos (n), h (parâmetro de interpolação desejado), o tipo de condição dos pontos finais que será usada (I, II, III), os valores das condições finais que serão

usadas (S_1 e S_n) onde:

para condição I

$$S_1 = \frac{1}{2} \frac{(\nabla + \Delta) S(\tau_1)}{h}$$

$$S_n = \frac{1}{2} \frac{(\nabla + \Delta) S(\tau_n)}{h},$$

para o condição II

$$S_1 = \frac{1}{2} \frac{(\nabla + \Delta) S(\tau_1)}{h^2}$$

$$S_n = \frac{1}{2} \frac{(\nabla + \Delta) S(\tau_n)}{h^2} \quad e$$

para a condição III

$$\frac{1}{2} (\nabla + \Delta) S(\tau_1) = \frac{1}{2} (\nabla + \Delta) S(\tau_n),$$

os valores de b_i (os valores da função b no intervalo $[\tau_1, \tau_n]$ a ser interpolada) e os τ_i (os nodos), a subrotina calcula os coeficientes c_i, x_i e d_i que serão substituidos na equação 7.2 onde :

$$c_i = \bar{c}_i + x_i h - d_i h^2,$$

$$x_i = x_i - c_i x_{i+1} - d_i x_{i+2} \quad \text{para} \quad 1 \leq i \leq n-1 \quad e$$

$$x_n = \frac{x_n}{b_n},$$

$$d_n = \frac{1}{3h_i}(x_{i+1} - x_i).$$

O valor de b_i permanece o mesmo.

PROBLEMA 1

DADOS DE ENTRADA

$n = 6$, $h = 0.1$, $S_1 = 2000.001$, $S_n = 0.02$ e

i	τ_i	b_i
1	0.5	1.0
2	0.7	0.5
3	1.0	2.0
4	1.5	2.5
5	2.1	2.0
6	2.5	1.0

TESTE 1

TIPO DE CONDIÇÃO = I

i	b_i	c_i	x_i	d_i
1	1.0	1731.871	-14034.42	26812.79
2	0.5	2606.557	2053.255	17875.19
3	2.0	4336.930	-362.6135	10725.11
4	2.5	5202.281	83.30513	8937.586
5	2.0	3466.240	-30.46501	13406.40

TESTE 2

TIPO DE CONDIÇÃO = II

i	b_i	c_i	x_i	d_i
1	1.0	-127.7612	1000.000	-1868.470
2	0.5	-182.8918	-121.0817	-1245.646
3	2.0	-321.1531	-13.76921	-747.3879
4	2.5	-376.6172	-5.15967	-622.8228
5	2.0	-253.0223	-0.97497	-934.2358

TESTE 3

TIPO DE CONDIÇÃO = III

i	b_i	c_i	x_i	d_i
1	1.0	-3.801781	-2.156043	43.32474
2	0.5	-3.047328	23.83881	28.88315
3	2.0	-2.254453	-11.25748	17.32990
4	2.5	-4.738678	0.5319628	14.44157
5	2.0	-5.103561	-2.204966	21.66238

PROBLEMA 2

DADOS DE ENTRADA

$n = 6$, $h = 0.1$, $S_1 = 2000.001$, $S_n = 0.02$ e

i	τ_i	b_i
1	10.0	0.42
2	10.2	0.48
3	10.4	0.51
4	10.6	0.52
5	10.8	0.53
6	11.0	0.55

TESTE 1

TIPO DE CONDIÇÃO = I

i	b_i	c_i	x_i	d_i
1	0.42	0.0530625	0.02275939	-0.0520625
2	0.48	-0.0969376	-0.8478079	-0.0520625
3	0.51	-0.1969392	-0.1890951	-0.05206225
4	0.52	-0.1969378	-0.01762926	-0.0520625
5	0.53	-0.1469375	0.2948773	-0.0520625

TESTE 2

TIPO DE CONDIÇÃO = II

i	b_i	c_i	x_i	d_i
1	0.42	0.3303851	0.0005	-0.7621207
2	0.48	0.1803849	-0.456772	-0.7621207
3	0.51	0.0803846	-0.2598738	-0.7621170
4	0.52	0.0803846	0.0160078	-0.7621207
5	0.53	0.1303851	0.1638332	-0.7621207

TESTE 3

TIPO DE CONDIÇÃO = III

i	b_i	c_i	x_i	d_i
1	0.42	-0.0908044	0.0344828	-0.0747128
2	0.48	-0.2408046	-0.0103448	-0.0747128
3	0.51	-0.3408069	-0.2413811	-0.0747124
4	0.52	-0.3408048	0.4827592	-0.0747128
5	0.53	-0.9408052	-0.0265517	-0.0747128

Interpolação polinomial é terrível para esses dados, pois este conjunto de dados é difícil para modelar matematicamente.

PROBLEMA 3

DADOS DE ENTRADA

$n = 6$, $h = 0.1$, $S_1 = 2000.001$, $S_n = 0.02$ e

i	τ_i	b_i
1	10.0	29.1
2	20.0	23.4
3	30.0	14.9
4	40.0	4.3
5	50.0	-7.4
6	60.0	-28.5
7	70.0	-36.0

8	80.0	-2.3
9	90.0	29.1
10	100.0	62.6

TESTE 1

TIPO DE CONDIÇÃO = I

i	b_i	c_i	x_i	d_i
1	10.0	0.00096807	-0.08902944	0.0031 9326
2	20.0	-0.2790318	0.00676843	0.0031 9326
3	30.0	-0.4890318	-0.02205675	-0.0031 9326
4	40.0	-0.5990319	0.01846553	-0.0031 9326
5	50.0	-0.01539032	-0.08481980	-0.0031 9326
6	60.0	-0.1790318	0.03883641	-0.0031 9326
7	70.0	0.03940968	0.3374917	-0.0031 9326
8	80.0	0.03710968	-0.1528820	-0.0031 9326
9	90.0	0.03920968	0.2051214	-0.0031 9326

TIPO DE CONDIÇÃO = II

i	b_i	c_i	x_i	d_i
1	10.0	-0.5159940	0.00052944	-0.0005900611
2	20.0	-0.7959939	-0.01720183	-0.0005900611
3	30.0	-0.01005994	-0.01569074	-0.0005900611
4	40.0	-0.01115994	0.01696793	-0.0005900611
5	50.0	-0.02055994	-0.08519450	-0.0005900611
6	60.0	-0.6959939	0.04183304	-0.0005900611
7	70.0	0.03424006	0.3258781	-0.0005900611

8	80.0	0.03194006	-0.1094172	-0.0005900611
9	90.0	0.03404006	0.04284943	-0.0005900611

TIPO DE CONDIÇÃO = III

i	b_i	c_i	x_i	d_i
1	10.0	-0.6334653	0.02083151	-0.001448499
2	20.0	-0.9134652	-0.02262346	-0.001448499
3	30.0	-0.01123465	-0.01433250	-0.001448499
4	40.0	-0.01233465	0.01696679	-0.001448499
5	50.0	-0.02173465	-0.08650415	-0.001448499
6	60.0	-0.8134652	0.04708460	-0.001448499
7	70.0	0.03306535	0.3061783	-0.001448499
8	80.0	0.03076535	-0.03585785	-0.001448499
9	90.0	-0.06346519	-0.2317322	-0.001448499

Apesar da interpolação utilizar o valor exato de b_i, a aproximação continua sendo melhor do que a interpolação. Se representarmos graficamente ambos os métodos poderemos verificar que o spline por aproximação se aproxima mais do gráfico real do que o spline por interpolação.

6.4 - O Algoritmo de Akima

Este algoritmo foi desenvolvido por Hiroshi Akima[5] do Departamento de Comércio do Instituto de Ciências de Telecomunicação, Boulder descrito na seção 4.3.

O algoritmo é implementado em dois subprogramas, IDBVIP e IDSFFT que implementam o método de interpolação e de aproximação suave de superfícies para pontos distribuídos irregularmente.

O pacote consiste de nove subprogramas, oito subrotinas e uma função. Duas subrotinas, IDBVIP e IDSFFT são subrotinas principais do pacote. A subrotina IDBVIP efetua a interpolação bivariavel; ela estima os valores de z para pontos especificados no plano x-y. A subrotina IDSFFT efetua uma aproximação suave de superfície; ela estima os valores de z para os pontos da malha retangular especificada no plano x-y e gera um conjunto contendo estes valores estimados.

As seis subrotinas restantes são subrotinas de suporte chamadas ou por IDBVIP ou IDSFFT ou por ambas: IDCLDP determina os pontos que são vizinhos aos pontos dados. IDGRID organiza os pontos da malha para ajuste de superfície ordenando em ordem crescente dos números de triângulos e dos números de lados. IDLCTN localiza um ponto, isto é, determina a que triângulo um ponto (x,y) pertence. IDPDRV estima as derivadas parciais de primeira e segunda ordens nos pontos dados. IDPTIP executa a interpolação ou extrapolação num ponto, isto é, determina o valor de z num ponto. IDTANG executa a triangulação, dividindo o plano x-y num número de triângulos de acordo com os pontos dados no plano, determina segmentos de reta que formam a casca convexa dos pontos dados e determina os números de triângulo correspondentes ao número de lados. A função IDXCHG

determina se uma permuta de dois triângulos é necessária ou não baseado no critério ângulo max-min de Lawson.

Este algoritmo foi testado com os seguintes problemas:

PROBLEMA 1

DADOS DE ENTRADA

i	x_i	y_i	z_i
1	11.16	1.24	22.15
2	24.20	16.23	2.83
3	19.85	10.72	7.97
4	10.35	4.11	22.33
5	19.32	1.39	16.83
6	0.00	20.00	34.60
7	20.87	20.00	5.74
8	19.99	4.62	14.72
9	10.28	15.16	21.59
10	4.51	20.00	17.61
11	0.00	4.48	61.77
12	16.70	19.65	6.31
13	6.08	4.58	35.74
14	25.00	11.87	4.40
15	14.90	3.12	21.70
16	0.00	0.00	58.20
17	9.66	20.00	4.73
18	5.22	14.66	40.36
19	11.77	10.47	13.62
20	15.10	17.19	12.57
21	25.00	3.87	8.74
22	25.00	0.00	12.00

UFPb/BIBLIOTECA/PRAI

23	14.59	8.71	14.81
24	15.20	0.00	21.60
25	5.23	10.72	26.50
26	2.14	15.03	53.10
27	0.51	8.37	49.43
28	25.00	20.00	0.60
29	21.67	14.36	5.52
30	3.31	0.13	44.08
31	0.00	0.00	58.20
32	5.00	0.00	39.55
33	10.00	0.00	26.90
34	15.00	0.00	21.71
35	20.00	0.00	17.68
36	25.00	0.00	12.00
37	0.00	5.00	61.58
38	5.00	5.00	39.39
39	10.00	5.00	22.04
40	15.00	5.00	21.29
41	20.00	5.00	14.36
42	25.00	5.00	8.04
43	0.00	10.00	59.18
44	5.00	10.00	27.39
45	10.00	10.00	16.78
46	15.00	10.00	13.25
47	20.00	10.00	8.59
48	25.00	10.00	5.36
49	0.00	15.00	52.82
50	5.00	15.00	40.27
51	10.00	15.00	22.76
52	15.00	15.00	16.61
53	20.00	15.00	7.40
54	25.00	15.00	2.88
55	0.00	20.00	34.60
56	5.00	20.00	14.05
57	10.00	20.00	4.17
58	15.00	20.00	3.17
59	20.00	20.00	6.31
60	25.00	20.00	0.60

i	x_i	y_i	INT	APRO
1	0.00	0.00	58.20	58.20
2	5.00	0.00	61.58	61.58
3	10.00	0.00	59.18	59.18
4	15.00	0.00	52.82	52.82
5	20.00	0.00	34.60	34.60
6	25.00	0.00	39.55	39.55
7	0.00	5.00	39.39	39.39
8	5.00	5.00	27.39	27.39
9	10.00	5.00	40.27	40.27
10	15.00	5.00	14.05	14.05
11	20.00	5.00	26.90	26.90
12	25.00	5.00	22.04	22.04
13	0.00	10.00	16.78	16.78
14	5.00	10.00	20.76	20.76
15	10.00	10.00	4.12	4.12
16	15.00	10.00	21.71	21.71
17	20.00	10.00	21.29	21.29
18	25.00	10.00	13.25	13.25
19	0.00	15.00	16.61	16.61
20	5.00	15.00	3.17	3.17
21	10.00	15.00	17.68	17.68
22	15.00	15.00	14.36	14.36
23	20.00	15.00	8.59	8.59
24	25.00	15.00	7.40	7.40
25	0.00	20.00	6.31	6.31
26	5.00	20.00	12.00	12.00
27	10.00	20.00	8.04	8.04
28	15.00	20.00	5.36	5.36
29	20.00	20.00	2.88	2.88
30	25.00	20.00	0.60	0.60

PROBLEMA 2

DADOS DE ENTRADA

i	x_i	y_i	z_i
1	0.00	1.00	-0.500000
2	0.70	1.70	-1.281667
3	1.20	2.50	-2.086667
4	2.00	3.40	-4.446667
5	2.70	4.10	-5.975000
6	3.20	4.90	-8.591666
7	4.00	5.80	-11.48667
8	4.70	6.50	-13.76167
9	5.20	7.30	-17.63167
10	6.00	8.10	-21.62000
11	6.70	8.80	-24.64167
12	7.20	9.70	-29.76500
13	8.00	10.60	-34.84667
14	8.70	11.30	-38.61500
15	9.20	12.10	-44.99166
16	10.00	13.00	-51.16667
17	10.70	13.70	-55.68167
18	11.20	14.50	-63.31167
19	12.00	15.40	-70.58000
20	12.70	16.10	-75.84167
21	13.20	16.90	-84.72500
22	14.00	17.80	-93.08667
23	14.70	18.50	-99.09500
24	15.20	19.30	-10.92317
25	0.00	1.00	-1.83
26	2.10	1.00	6.00
27	4.20	1.00	17.76
28	6.30	1.00	45.20
29	8.40	1.00	-1.14
30	10.50	1.00	6.66

31	0.00	6.50	22.65
32	2.10	6.50	46.80
33	4.20	6.50	-0.23
34	6.30	6.50	8.45
35	8.40	6.50	30.01
36	10.50	6.50	50.91
37	0.00	12.10	2.08
38	2.10	12.10	9.21
39	4.20	12.10	31.33
40	6.30	12.10	52.61
41	8.40	12.10	2.56
42	10.50	12.10	12.08
43	0.00	17.80	36.16
44	2.10	17.80	56.96
45	4.20	17.80	4.45
46	6.30	17.80	14.33
47	8.40	17.80	39.81
48	10.50	17.80	58.74

RESULTADOS

i	x_i	y_i	INT	APRO
1	0.00	1.00	-0.50	-0.50
2	2.10	1.00	314.15	314.15
3	4.20	1.00	1127.70	1127.70
4	6.30	1.00	2379.48	2349.78
5	8.40	1.00	57.02	57.02
6	10.50	1.00	70.43	70.43
7	0.00	6.50	619.65	619.65
8	2.10	6.50	1699.88	1699.88
9	4.20	6.50	233.92	233.92
10	6.30	6.50	-12.46	-12.46
11	8.40	6.50	270.84	270.84
12	10.50	6.50	1156.51	1156.51

13	0.00	12.10	489.21	489.21
14	2.10	12.10	8.05	8.05
15	4.20	12.10	60.11	60.11
16	6.30	12.10	716.60	716.60
17	8.40	12.10	766.64	766.64
18	10.50	12.10	87.08	87.08
19	0.00	17.80	-38.96	-38.96
20	2.10	17.80	318.54	318.54
21	4.20	17.80	1021.85	1021.85
22	6.30	17.80	225.21	225.21
23	8.40	17.80	-30.77	-30.77
24	10.50	17.80	38.66	38.66

Como podemos verificar nos testes acima, os resultados da interpolação e aproximação são idênticos. Isto ocorre porque o método obriga a superfície a passar pelos pontos dados. Além disso, o algoritmo de Akima para um mesmo (x,y) dado não gera, tanto para a aproximação como para a interpolação, o mesmo valor de z. O algoritmo necessita também , além dos valores de x,y e a a serem interpolados e aproximados, dos valores de saída de x e y. Esses valores são utilizados para gerar a malha e para calcular o valor final de z fazendo com que o valor resultante de z, nos pontos dados, seja diferente.

Deste modo não é possível fazer uma análise do erro cometido.

CAPÍTULO VII

CONCLUSÕES E TRABALHOS ADICIONAIS

Iniciamos este último capítulo comentando sobre as dificuldades encontradas para a realização do trabalho, e apresentamos sugestões de tópicos para estudos posteriores.

7.1 - Dificuldades Encontradas

As maiores dificuldades encontradas foram:

- a extensão do tópico que implicou em um imenso trabalho de seleção de algoritmos relevantes para constar de uma biblioteca; a ACM já publicou 25 algoritmos, o livro de Lawson & Hanson [12] contém 14 subprogramas para problemas de quadrados

mínimos, a biblioteca IMSL tem 16 algoritmos de aproximação e suavização e o NAG tem 7 algoritmos de interpolação e 22 de aproximação de superfícies e curvas.

- variedade de assuntos envolvidos: além da interpolação e aproximação, programação linear, otimização sem restrições, programação não linear, e outros.

- muito pouca referência sobre problemas não lineares em normas 1 e infinita.

- Problemas na implementação dos algoritmos pois o compilador FORTRAN do IBM 4381 da UFMS estava com problemas.

7.2 - Trabalhos Adicionais

Como o nosso trabalho foi em aproximação em normas 1 em infinita para problema linear sugerimos os seguintes temas para estudos posteriores:

- O mesmo trabalho para problema não linear;

- Aproximação adaptativa;

- Aprofundamento de estudos sobre aproximação de superfícies, comparando os diversos algoritmos existentes;

- Aprofundamento de estudos sobre splines.

7.3 - Conclusões Finais

Apesar do nosso objetivo inicial ter sido o estudo de aproximação, optamos por estudar também a interpolação pois não é possível separar um assunto do outro.

No nosso estudo sobre interpolação e aproximação e nos testes realizados nos algoritmos implementados, pudemos constatar que a aproximação é melhor do que a interpolação. De um modo geral, e nos métodos estudados não se pode garantir a precisão dos resultados da interpolação.

Mas, isto não quer dizer que a interpolação não tenha utilidade. Em muitos casos a aproximação não pode ser usada e quando usada seus resultados não são bons. No caso de aproximação de superfícies suaves, o método de Akima não faz a suavização dos dados. Em outras palavras, a superfície resultante passa por todos os pontos dados. Portanto o método é aplicado somente quando são dados os valores precisos de z (onde z é a função a ser aproximada) ou quando os erros são desprezíveis.

REFERÊNCIA BIBLIOGRÁFICA

[1] ABDELMALEK, Nabih N., Algorithm 551 - A Fortran Subroutine for the L_1 Solution of overdetermined Systems of Linear Equations. ACM Transactions on Mathematical Software, Vol. 6, N.° 2, junho de 1980, páginas 228 - 230.

[2] ABDELMALEK, Nabih N., An Efficient Method for the Discrete linear L_1 Approximation Problem. Mathematics of Computation, Vol. 19, N.° 131, julho de 1975, páginas 844 - 850.

[3] ABDELMALEK, Nabih N., On the Discrete linear L_1 Approximation and L_1 Solution of Overdetermined Systems of Linear Equations. Journal of Approximation Theory, vol.11, 1974, páginas 38-53.

[4] AKIMA, Hiroshi, A Method of Bivariate Interpolation and Smooth Surface Fitting for Irregularly Distributed Data Points. ACM Transactions on Mathematical Software, Vol.4, N.° 2, junho de 1978, páginas 148 - 159.

[5] AKIMA, Hiroshi, Algorithm 526 - Bivariate Interpolation and Smooth Surface Fitting for Irregularly Distributed Data Points. ACM Transactions on Mathematical Software, Vol. 4, N.° 2, junho de 1978, páginas 160 - 164.

[6] BARRODALE, I. & ROBERTS, F. D. K., Algorithm 552 - Solution of the Constrained l_1 Linear Approximation Problem. ACM Transactions on Mathematical Software, Vol. 6, N.° 2, junho de 1980, páginas 231 - 235.

[7] BARRODALE, I. & ROBERTS, F. D. K., An efficient Algorithm for Discrete l_1 Linear Approximation with Linear Constraints. SIAM J. Numer. Anal., Vol. 15, N.° 3, junho de 1978, páginas 603 - 611.

[8] BARRODALE, I. & ROBERTS, F. D. K., An Improved Algorithm for Discrete l_1 Linear Approximation. SIAM J. Numer. Anal., Vol. 10, 1973, páginas 839 - 848.

[9] DURIS, Charles S., Algorithm 547 - Fortran Routines for Discrete Cubic Spline Interpolation and Smoothing. ACM Transactions on Mathematical Software, Vol. 6, N.° 1, março de 1980, páginas 92 - 103.

[10] EL - ATTAR et alli, M. e DUTTA R. K., Algorithm for l_1-norm Minimization with Application to Nonlinear l_1- Approximation. SIAM J. Numer. Anal., Vol. 16, N.° 1, fevereiro de 1979, páginas 70 - 86.

[11] LAWSON, C. L. & HANSON R. J., Solving Least Squares Problems, Englewood Cliffs, Prentice-Hall, 1974.

[12] MACULAN, N. & PEREIRA, M. V. F., Programação Linear, Atlas, São Paulo, 1980.

[13] MATEUS, Geraldo R. e LUNA, Henrique P. L., Programação Não Linear. V Escola de Computação, Belo Horizonte, UFMG, 1986.

[14] PRENTER, P. M., Splines and Variational Methods, New York, Wiley Interscience, 1975.

[15] STOER, J. e BURLIRSCH, R., Introduction to Numerical Analisys, New York, Springer - Verlag, 1980.

[16] WOLFE, M.A., Numerical Methods for Unconstrained Optimization, Van Nostrand, 1978, capítulo 7.

APÊNDICE

UTILIZAÇÃO DAS SUBROTINAS

Mostramos neste apêndice como utilizar as subrotinas implementadas e apresentamos uma breve explicação do que cada subrotina faz.

INTERPOLAÇÃO

SUBROTINA IDBVIP

Esta subrotina efetua interpolação em 2 variáveis quando são dados os valores da função ZD em pontos (XD,YD) irregularmente distribuidos no plano X-Y, usando o método de Akima [5].

SEQÜÊNCIA DE CHAMADA

```
CALL IDBVIP(MD,NCP,XD,YD,ZD,NIP,XI,YI,ZI,IWK,WK)
```

PARAMÊTROS

NA CHAMADA

MD - MODO DE COMPUTAÇÃO (DEVE SER 1, 2, OU 3)

= 1 PARA UM NOVO NCP E/OU UM NOVO CONJUNTO (XD,YD),

= 2 PARA UM VELHO NCP, VELHO CONJUNTO (XD,YD), NOVO CONJUNTO (XI,YI),

= 3 PARA UM VELHO NCP, VELHO CONJUNTO (XD,YD), VELHO CONJUNTO (XI,YI).

NCP - NÚMERO DE PONTOS DE DADOS ADICIONAIS USADO PARA ESTIMAR AS DERIVADAS PARCIAIS EM CADA PONTO (DEVE SER MAIOR OU IGUAL A 2 MAS MENOR DO QUE NDP).

NDP - NÚMERO DE PONTOS DE DADOS (DEVE SER MAIOR OU IGUAL A 4).

XD - VETOR DE DIMENSÃO NDP CONTENDO AS COORDENADAS X DOS PONTOS DADOS.

YD - VETOR DE DIMENSÃO NDP CONTENDO AS COORDENADAS Y DOS PONTOS DADOS.

ZD - VETOR DE DIMENSÃO NDP CONTENDO OS VALORES DA FUNÇÃO NOS PONTOS (XD,YD) DADOS (COORDENADAS ZD.

NIP - NÚMERO DE PONTOS EM QUE A INTERPOLAÇÃO SERÁ EFETUADA (DEVE SER MAIOR OU IGUAL A 1).

XI - VETOR DE DIMENSÃO NIP CONTENDO AS COORDENADAS X DOS PONTOS DE INTERPOLAÇÃO.

YI - VETOR DE DIMENSÃO NIP CONTENDO AS COORDENADAS Y DOS PONTOS DE INTERPOLAÇÃO.

IWK - VETOR INTEIRO DE DIMENSÃO MAXO(31,27 + NCP)*NDP+NIP USADO INTERNAMENTE COMO ÁREA DE TRABALHO.

WK - VETOR REAL DE DIMENSÃO 8*NDP USADO INTERNAMENTE COMO ÁREA DE TRABALHO.

NO RETORNO

ZI - VETOR DE DIMENSÃO NIP CONTENDO OS VALORES DE Z INTERPOLADOS. DECLARADO MAS NÃO INICIALIZADO.

SUBROTINAS UTILIZADAS

IDCLDP- Esta subrotina seleciona os pontos que são vizinhos dos pontos dados.

IDLCTN- Esta subrotina localiza um ponto, isto é, determina a que triângulo um dados ponto (XII,YII) pertence. Quando um dado ponto não está situado na área dos dados, esta subrotina determina o segmento de fronteira quando o ponto está situado no exterior de uma área retangular e dois segmentos quando o ponto está situado no exterior de uma área triangular.

IDPDRV- Esta subrotina estima as derivadas parciais de primeira e segunda ordens nos pontos dados.

IDPTIP- Esta subrotina executa a interpolação ou extrapolação num ponto, isto é, determina o valor de Z num ponto.

IDTANG- Esta subrotina executa a triangulação. Divide o plano X-Y num número de triângulos de acordo com os pontos dados no plano, determina segmentos de reta que formam a casca convexa dos pontos dados e determina os números dos triângulos correspondentes aos lados dos triângulos. No término, os números de pontos

dos vértices de cada triângulo são listados no sentido anti-horário. Os números de ponto de cada um dos vértices dos lados são listados no sentido anti-horário.

APROXIMAÇÃO

SUBROTINA IDSFFT

Esta subrotina efetua a aproximação de superfícies quando os valores da função são dados em pontos irregularmente distribuidos no plano X-Y.

SEQÜÊNCIA DE CHAMADA

```
CALL IDSFFT (MD,NCP,NDP,XD,YD,ZD,NXI,NYI,XI,YI,ZI,IWK,WK)
```

PARÂMETROS

NA CHAMADA

MD - MODO DE COMPUTAÇÃO (DEVE SER 1, 2, OU 3)

= 1 PARA UM NOVO NCP E/OU UM NOVO CONJUNTO (XD,YD),

= 2 PARA UM VELHO NCP, VELHO CONJUNTO (XD,YD), NOVO CONJUNTO (XIYI),

= 3 PARA UM VELHO NCP, VELHO CONJUNTO (XD,YD), VELHO CONJUNTO (XI,YI).

NCP - NÚMERO DE PONTOS ADICIONAIS USADO PARA ESTIMAR AS DERIVADAS PARCIAIS EM CADA PONTO DADO (DEVE SER MAIOR OU IGUAL A 2 MAS, MENOR DO QUE NDP).

NDP - NÚMERO DE PONTOS DADOS (DEVE SER MAIOR OU IGUAL A 4).

XD - VETOR DE DIMENSÃO NDP CONTENDO AS COORDENDAS X DOS PONTOS DADOS.

YD - VETOR DE DIMENSÃO NDP CONTENDO AS COORDENADAS Y DOS PONTOS DADOS.

ZD - VETOR DE DIMENSÃO NDP CONTENDO OS VALORES DA FUNÇÃO NOS PONTOS DADOS (COORDENADAS ZD.

NXI - NÚMERO DE COORDENADAS X DE SAÍDA.
(DEVE SER MAIOR OU IGUAL A 1)

NYI - NÚMERO DE COORDENADAS Y DE SAÍDA.
(DEVE SER MAIOR OU IGUAL A 1)

XI - VETOR DE DIMENSÃO NXI CONTENDO AS COORDENADAS X DOS PONTOS DE SAÍDA.

YI - VETOR DE DIMENSÃO NYI CONTENDO AS COORDENADAS Y DOS PONTOS DE SAÍDA.

IWK - VETOR INTEIRO DE DIMENSÃO MAXO(31,27+NCP)*NDP+NIP USADO INTERNAMENTE COMO ÁREA DE TRABALHO.

WK - VETOR INTEIRO DE DIMENSÃO 8*NDP INTERNAMENTE COMO ÁREA DE TRABALHO.

NO RETORNO

ZI - CONJUNTO BIDIMENSIONAL DE DIMENSÃO (NXI,NYI) CONTENDO OS VALORES INTERPOLADOS DE Z NOS PONTOS DESEJADOS.

SUBROTINAS

IDCLDP- Esta subrotina seleciona os pontos que são vizinhos dos pontos dados.

IDPTIP- Esta subrotina executa a interpolação ou extrapolação num ponto, isto é, determina o valor de Z num ponto.

IDTANG- Esta subrotina executa a triangulação. Divide o plano X-Y num número de triângulos de acordo com os pontos dados no plano, determina segmentos de reta que formam a casca convexa dos pontos dados e determina os números dos triângulos correspondentes aos lados dos triângulos.

IDGRID- Esta subrotina organiza os pontos de malha para ajuste de superfície ordenado em ordem crescente dos números de triângulos e dos números de segmentos de fronteira.

IDPDRV- Esta subrotina estima as derivadas parciais de primeira e segunda ordens nos pontos dados.

2 - Aproximação de sistemas de equações lineares hiperdeterminadas

SUBROTINA L1

Esta subrotina resolve um sistema hiperdeterminado de equações lineares na norma l_1 pelo uso de um algoritmo simplex dual para a programação linear e formulação do problema dado. Nesse algoritmo, iterações intermediárias são certamente omitidas.

Para o propósito de estabilidade numérica, esta subrotina usa uma decomposição triangular para matrizes fundamentais.

O sistema de equações lineares tem a forma:

$$C * A = F$$

onde :

C - uma matriz N X M real de ordem K.I.E. M.I.E.

F - um vetor real de tamanho N

O problema está em calcular os elementos do vetor A^* de ordem M que minimiza a norma de Z.

$$Z = ABS(R(1)) + ABS(R(2)) + \ldots + ABS(R(N)),$$

onde :

R(I) é o ith resíduo e é dado por :

$$R(I) = C(I,1) * A(1) + C(I,2) * A(2) + \ldots + C(I,M) * A(M) - F(I)$$

SEQÜÊNCIA DE CHAMADA

```
   CALL L1(M1,N1,M2,M3,N2,CT,F,PREC,EPS,IC,IR,IB,UF,BP,XP,T,
  X                ALFA,P,GINV,VB,IRANK,ITER,R,A,Z,IND)
```

PARÂMETROS

NA CHAMADA

M1 - NÚMERO DE COLUNAS DA MATRIZ C.

N1 - NÚMERO DE LINHAS DA MATRIZ C.

M2 - UM INTEIRO MAIOR OU IGUAL A M1.

M3 - UM INTEIRO IGUAL A (M2 * (M2 + 3) / 2.

N2 - UM INTEIRO MAIOR OU IGUAL A N1.

CT - UMA MATRIZ DE DIMENSÃO M2 X N2. NA CHAMADA AS PRIMEIRAS M1 LINHAS E AS PRIMEIRAS N1 COLUNAS CONTÉM A TRANSPOSTA DA MATRIZ NO SISTEMA C * A = F DADO.

F - UM VETOR DE TAMANHO N2. NA CHAMADA OS PRIMEIROS N ELEMENTOS CONTÉM O LADO DIREITO DO SISTEMA C * A = F DADO.

PREC - O NÍVEL DE ARREDONDAMENTO DO COMPUTADOR.

EPS - UMA TOLERÂNCIA ESPECIFICADA TAL QUE UM NÚMERO CALCULADO X É CONSIDERADO ZERO SE ABS(X).LT. EPS.

GINV - UMA MATRIZ QUADRADA DE ORDEM M2. AS PRIMEIRAS IRANK COLUNAS E AS PRIMEIRAS IRANK LINHAS CONTÉM A INVERSA DA MATRIZ FUNDAMENTAL INICIAL E A ATUAL.

VB - UM VETOR DE TAMANHO MM. OS IRANK ELEMENTOS CONTÉM A SOLUÇÃO FUNDAMENTAL INICIAL E A ATUAL.

T - UM VETOR DE TAMANHO NN, OS PRIMEIROS N ELEMENTOS CONTÉM OS ELEMENTOS DA LINHA NO QUADRO SIMPLEX, QUE CORRESPONDE A COLUNA O QUAL RENUNCIA O BÁSICO.

ALFA - UM VETOR DE TAMANHO NN. OS PRIMEIROS N ELEMENTOS CONTÉM OS VALORES :

ALFA(J) = R(J) / T(J)

IC - UM VETOR DE TAMANHO NN. OS PRIMEIROS N ELEMENTOS CONTÉM A COLUNA DE ÍNDICES DA MATRIZ CT.

IR - UM VETOR DE TAMANHO MM. OS PRIMEIROS IRANK ELEMENTOS CONTÉM A LINHA DE ÍNDICES DAS LINHAS LINEARMENTE INDEPENDENTES DE CT.

IB - UM VETOR SINAL DE TAMANHO NN. OS PRIMEIROS N ELEMENTO TEM OS VALORES +1 OU -1. IB(J) = +1 INDICA QUE A COLUNA J NO QUADRO ESTA ABAIXO DO LIMITE 0. IB(J) = -1 INDICA QUE A COLUNA J ESTA ACIMA DO LIMITE 2.

P - UM VETOR DE TAMANHO MMM. OS PRIMEIROS ((IRANK * (IRAN + 3)) / 2) - 1 ELEMENTOS CONTÉM OS (IRANK * (IRANK + 1)) / 2 ELEMENTOS DA MATRIZ TRIANGULAR SUPERIOR MAIS (IRANK - 1) LOCAÇÕES DE TRABALHO EXTRA. VEJA OS COMENTÁRIOS NA SUBROTINA PSLV.

BP - UM VETOR DE TAMANHO MM. OS PRIMEIROS IRANK ELEMENTOS SÃO OS LADO DIREITO DAS EQUAÇÕOES TRIANGULARES COMO P * XP = BP.

XP - UM VETOR DE TAMANHO MM. OS PRIMEIROS IRANK ELEMENTOS SÃO A SOLUÇÃO DAS EQUAÇÕES TRIANGULARES COMO P * XP = B.

UF - UM VETOR DE TRABALHO DE TAMANHO MM.

NO RETORNO

IRANK- CONTERÁ A ORDEM CALCULADA DA MATRIZ C.

ITER - O NÚMERO DE ITERAÇÕES QUE A SOLUÇÃO NECESSITOU.

R - UM VETOR DE TAMANHO NN. NO RETORNO OS PRIMEIROS N ELEMENTOS CONTERÃO O RESÍDUO R = C * A - F.

A - UM VETOR DE TAMANHO NN. OS PRIMEIROS M ELEMENTOS CONTERÃO A SOLUÇÃO DO VETOR A^*.

Z - O MÍNIMO DA NORMA L1 DO VETOR RESTO R.

IND - UM INDICADOR DE RETORNO. IND = 0 INDICA QUE A SOLUÇÃO DO VETOR A^* É ÚNICA. IND = 1, INDICA QUE A^* É MUITO PROVAVELMENTE NÃO ÚNICA. IND = -1 INDICA TÉRMINO PREMATURO DO PRÓPRIO CÁLCULO POR VÁRIAS MÁ CONDIÇÕES DA MATRIZ C.

SUBROTINAS

PSLV - Esta subrotina resolve o sistema quadrado não singular de equações lineares

$$P * X = B,$$

ou o sistema quadrado não singular de equações lineares

P(TRANSPOSTA) * X = B,

onde :

P - uma matriz triangular superior,

B - um vetor, e

X - o vetor solução.

3 - Aproximação linear l_1 discreta

SUBROTINA CL1

Esta subrotina usa uma modificação do método simplex da programação linear para calcular uma solução l_1 para um sistema de equações lineares K x N

$$AX = B$$

sujeito a L igualdades lineares restritas

$$CX = D$$

e M desigualdades lineares restritas

SEQÜÊNCIA DE CHAMADA

```
      CALL CL1(K,L,M,N,KLMD,KLM2D,NKLMD,N2D,Q,KODE,TOLER,ITER,
     X           X,RES,ERROR,CU,IU,S)
```

PARÂMETROS

NA CHAMADA

K - N.º DE LINHAS DA MATRIZ A(K.GE.1).

L - N.º DE LINHAS DA MATRIZ C(L.GE.0).

M - N.º DE LINHAS DA MATRIZ F(M.GE.0).

N - N.º DE COLUNAS DAS MATRIZES A, C, E (N.GE.1).

KLMD - DEVE NO MÍNIMO SER IGUAL A K + L + M

KLM2D- DEVE NO MÍNIMO SER IGUAL A K + L + M + 2

NKLMD- DEVE NO MÍNIMO SER IGUAL A N + K + L + M

N2D - DEVE NO MÍNIMO SER IGUAL A N + 2

Q - VETOR REAL BIDIMENSIONAL COM KLM2D LINHAS E PELO MENOS N2D COLUNAS. NA ENTRADA AS MATRIZES A, C E E, E OS VETORES B, DE F DEVEM SER ARMAZENADOS NAS PRIMEIRAS K + L + M LINHAS E N + 1 COLUNAS DE Q COMO SE SEQUE:

$$Q = \begin{matrix} E & B \\ C & D \\ E & F \end{matrix}$$

ESSES VALORES SÃO DESTRUIDOS PELA SUBROTINA.

KODE - UM CÓDIGO USADO NA ENTRADA E SAIDA DA SUBROTINA. NA ENTRADA, DEVERIA NORMALMENTE SER IGUAL A 0. DE QUALQUER MANEIRA, SE RESTRIÇÕES NÃO NEGATIVAS ESTÃO PARA SER INCLUÍDAS NA RESTRIÇÃO EX.LE.F, ENTÃO CERTAMENTE KODE DEVERIA SER IGUAL A 1. E A RESTRIÇÃO NÃO NEGATIVA INCLUÍDA NOS VETORES X E RES. NA SAIDA, KODE TEM UM DOS SEGUINTES VALORES:

0 - ENCONTRADA A SOLUÇÃO ÓTIMA,
1 - SOLUÇÃO NÃO VIÁVEL PARA A RESTRIÇÃO,
2 - TÉRMINO PREMATURO DOS CÁLCULO POR ERRO DE ARREDONDAMENTO,
3 - NÚMERO MÁXIMO DE ITERAÇÕES ATINGIDO.

TOLER- UM PEQUENO NÚMERO POSITIVO. SUGERE-SE TOLER = 10**(-D * 2/3), ONDE D REPRESENTA O NÚMERO DE DIGÍTOS DECIMAIS. A SUBROTINA NÃO PODE DISTINGUIR ENTRE ZERO E ALGUMA QUANTIDADE CUJA MAGNITUDE NÃO EXCEDA TOLER.

ITER - NA ENTRADA ITER DEVE CONTER O NÚMERO MÁXIMO DE ITERAÇÕES PERMITIDO. SUGERE-SE ITER = 10*(K+L+M). NA SAIDA ITER FORNECE O NÚMERO DE ITERAÇÕES SIMPLEX.

IU - UMA MATRIZ INTEIRA COM 2 LINHAS E PELO MENOS NKLMD COLUNAS USADO COMO ÁREA DE TRABALHO.

CU - UMA MATRIZ INTEIRA COM 2 LINHAS E PELO MENOS NKLMD COLUNAS USADO COMO ÁREA DE TRABALHO.

S - VETOR INTEIRO DE PELO MENOS KLMD POSIÇÕES, USADO COMO ÁREA DE TRABALHO.

NO RETORNO

X - UMA MATRIZ REAL DE TAMANHO PELO MENOS N2D. ESTE VETOR CONTÉM UMA SOLUÇÃO PARA O PROBLEMA L1. OS VALORES -1,0 OU 1 PARA X(J) INDICA QUE A J-ÉSIMA VARIÁVEL DEVE SER OU .GE.0 OU .LE.0.

RES - UMA MATRIZ REAL DE TAMANHO PELO MENOS KLMD. OS PRIMEIROS K COMPONENTES CONTÉM AS DIFERENÇAS B - AX, NOS PRÓXIMOS L COMPONENTES AS DIFERENÇAS D - CX (ESSES SERÃO = 0), E NOS PRÓXIMOS M COMPONENTES AS DIFERENÇAS F - EX.

ERROR- FORNECE A SOMA MINIMA DOS VALORES ABSOLUTOS DOS RESIDUOS.

4 - Interpolação e Aproximação suave discreta por splines

INTERPOLAÇÃO

SUBROTINA DCSINT

Esta subrotina computa o spline cúbico natural discreto definido sobre o intervalo (TNODE(1), TNODE(N)) o qual interpola os dados (TNODE(I), G(I)), I = 1, 2, ..., N.

Nos requeremos que TNODE(I).LT.TNODE(I+1), END1 e ENDN contenham os valores das condições dos pontos finais usadas.

Se IENT = 1, as condições dos pontos finais da primeira diferença dividida central serão usadas.

Se IENT = 2, as condições dos pontos finais da segunda diferença dividida central serão usadas.

Se IENT = 3, as condições periódicas serão usadas. Para este caso, os conteúdos de G(N), END1, e ENDN são ignorados. Para as três condições finais, N deve ser maior ou igual a 2.

O spline cúbico discreto é representado por polinômios cúbicos por partes. Para T no intervalo (TNODE(I), TNODE(I+1)) o sline cúbico é

```
S(T) = G(I) + B(I) * (T - TNODE(I)) +

       C(I) * (T - TNODE(I))**2 + D(I) *

       (T - TNODE(I))**3
```

SEQÜÊNCIA DE CHAMADA

```
CALL DCSINT (IENT, H, N, TNODE, G, END1, ENDN, B, C, D)
```

PARÂMETROS

NA CHAMADA

IENT - ESPECIFICA AS CONDIÇÕES DOS PONTOS FINAIS.

H - O TAMANHO DO PASSO USADO PARA O SPLINE CÚBICO DISCRETO.

N - NÚMERO DOS NODOS(TNODE) E O VALORES DOS DADOS(G).

TNODE - ARRAY REAL CONTENDO OS NODOS (TNODE(I).LT.TNODE(I+1)).

G - VETOR REAL CONTENDO OS DADOS INTERPOLANTES.

END1 - VALOR DA CONDIÇÃO FINAL PARA TNODE(1).

ENDN - VALOR DA CONDIÇÃO FINAL PARA TNODE(N).

NO RETORNO

B - VETOR REAL CONTENDO OS COEFICIENTES DE (T - TNODE(I)) I = 1, 2, ..., N-1.

C - VETOR REAL CONTENDO OS COEFICIENTES DE (T - TNODE(I))**2, I = 1, 2, ..., N-1.

D - VETOR REAL CONTENDO OS COEFICIENTES DE (T - TNODE(I))**3, I = 1, 2, ..., N-1.

SUBROTINA DCSSMO

Esta subrotina computa o spline cúbico natural discreto definido no intervalo (TNODE(I), TNODE(N)) para aproximar uma função nos pontos (TNODE(I), G(I)), I = 1, 2, ..., N. N deve ser maior ou igual a 2.

Os nodos devem satisfazer TNODE(I) < TNODE(I+1). A solução S(T) para T no intervalo (TNODE(I), TNODE(I+1)) é dada por

$$S(T) = GSMO(I) + B(I) * (T - TNODE(I)) +$$

$$C(I) * (T - TNODE(I))**2 + D(I) *$$

$$(T - TNODE(I))**3$$

SEQÜÊNCIA DE CHAMADA

CALL DCSSMO (H, N, TNODE, G, WGS, RHO, GSMO, B, C, D)

PARÂMETROS

NA CHAMADA

H - O TAMANHO DO PASSO USADO PARA A CONSTRUÇÃO DO SPLINE CÚBICO DISCRETO.

N - NÚMERO DE NODOS(TNODE) E O VALORES DOS DADOS(G).

TNODE- VETOR REAL CONTENDO OS NODOS (TNODE(I).LT.TNODE(I+1)).

G - VETOR REAL CONTENDO OS VALORES DOS DADOS.

WSG - VETOR REAL CONTENDO OS PESOS WSG(I) CORRESPONDENDO AO DADOS (TNODE(I), G(I)).

RHO - VARIÁVEL REAL SIMPLES CONTENDO O PARÂMETRO POSITIVO PARA A VARIAÇÃO DA SUAVIDADE DOS AJUSTES. SE RHO É PEQUENO A SUAVIDADE É ENFATIZADA. SE RHO É GRANDE A APROXIMAÇÃO É ENFATIZADA.

NO RETORNO

GSMO - VETOR REAL CONTENDO OS VALORES SUAVIZADOS DOS DADOS G(I), I = 1, 2, ..., N.

B - VETOR REAL CONTENDO OS COEFICIENTES B(I) PARA OS TERMOS (T - TNODE(I)).

C - VETOR REAL CONTENDO OS COEFICIENTES C(I) PARA OS TERMOS (T - TNODE(I))**3.