ANTONIO FLAVIO LICARIÃO NOGUEIRA

TECNICAS DE OTIMIZAÇÃO PARA MINIMIZAÇÃO UNIDIMENSIONAL

Dissertação apresentada à Coordenação dos Cursos de Pós-Graduação em Engenharia Elétrica da Universidade Federal da Paraíba, em cumprimento às exigências para obtenção do Grau de Mestre em Engenharia Elétrica.

AREA DE CONCENTRAÇÃO: PROCESSAMENTO DA ENERGIA

ORIENTADOR: JANUSZ S. LIPOWSKI

CO-ORIENTADOR: MARIO T. HATTORI

CAMPINA GRANDE

MARÇO - 1986

N778t Nogueira, Antonio Flavio Licariao
Tecnicas de otimizacao para minimizacao unidimensional / Antonio Flavio Licariao Nogueira. - Campina Grande, 1986.
119 f.

Dissertacao (Mestrado em Engenharia Eletrica) - Universidade Federal da Paraiba, Centro de Ciencias e Tecnologia.

1. Engenharia Eletrica 2. Minimizacao Unidimensional (Funcoes) 3. Algoritmos Hibridos 4. Processamento de Energia 5. Dissertacao I. Lipowski, Janusz S., Prof. II. Hattori, Mario T., Prof. III. Universidade Federal da Paraiba - Campina Grande (PB) IV. Título

CDU 621.3(043)

TECNICAS DE OTIMIZAÇÃO PARA MINIMIZAÇÃO UNIDIMENSIONAL

ANTONIO FLAVIO LICARIÃO NOGUEIRA

DISSERTAÇÃO APROVADA EM 12/03/86

JANUSZ S. LIPOWSKI

Orientador

MARIO T. HATTORI

Co-Orientador

WELINGTON S. MOTA

Componente da Banca

CAMPINA GRANDE

MARÇO - 1986

AGRADECIMENTOS A:

Mario T. Hattori

Janusz S. Lipowski

Antônio do Nascimento Epaminondas

RESUMO

Neste trabalho é feita uma discussão dos métodos de minimização unidimensional. São descritos vários algoritmos, desde os mais simples e básicos, até aqueles mais refinados que lidam com dificuldades adicionais como a não diferenciabilidade da função a ser minimizada. São descritos quatro algoritmos híbridos para a minimização de funções continuamente diferenciáveis. Foi feito um estudo mais detalhado de um algoritmo para a minimização de uma classe de funções que não são continuamente diferenciáveis. São apresentados, em um relatório de testes, os resultados de testes de desempenho dos algoritmos híbridos com proteções. Por fim, é apresentada uma introdução teórica sobre os algoritmos que fazem a busca do mínimo ao longo de um percurso curvilinear, usando direções de curvatura negativa.

ÍNDICE

INTRODUÇÃO

1.1 Objeto do trabalho

Os métodos descendentes que minimizam uma função $F(\tilde{x})$, $\tilde{x} \in E^n$, normalmente constroem uma sequência de estimativas $\{\tilde{x}^{(k)}\}$ do mínimo de $F(\tilde{x})$ de forma que

$$\tilde{x}^{(k+1)} = \tilde{x}^{(k)} + \alpha^{(k)} \tilde{p}^{(k)}$$

com

$$F(\tilde{x}^{(k+1)}) < F(\tilde{x}^{(k)}),$$

onde $\tilde{p}^{(k)}$ é o vetor de direção de pesquisa. Técnicas de minimização unidimensional são usadas para se determinar um escalar $\alpha^{(k)}$ que minimize $F(\tilde{x})$ para um dado ponto de pesquisa $\tilde{x}^{(k)}$ e uma direção de pesquisa $\tilde{p}^{(k)}$.

O algoritmo para minimização unidimensional influi sobremaneira no desempenho do algoritmo mais geral de minimização multidimensional e, por isso, o aspecto referente à determinação do escalar $\alpha^{(k)}$ é mais complexo do que pode parecer. O novo ponto obtido $\tilde{x}^{(k+1)}$ deve ser tal que $F(\tilde{x}^{(k+1)})$ seja suficientemente menor que $F(\tilde{x}^{(k)})$, a fim

de que a convergência do processo de minimização multidimensional seja garantida. Quando da escolha de um algoritmo para minimização unidimensional devem ser levadas em consideração as características da função objetivo ao longo da direção de pesquisa $\tilde{p}^{(k)}$, no ponto de pesquisa $\tilde{x}^{(k)}$: unimodalidade, continuidade, diferenciabilidade, etc.

Gill & Murray (1974) desenvolveram algoritmos confiáveis e eficientes para a determinação do escalar $\alpha^{(k)}$ no caso em que $F(\tilde{x})$ é continuamente diferenciável. Murray & Overton (1978) desenvolveram algoritmos, igualmente eficientes, quando se trata da minimização de certas classes de funções que não são continuamente diferenciáveis.

Existe uma grande quantidade de algoritmos para minimização unidimensional de funções continuamente diferenciáveis. No caso dos algoritmos para minimização de funções que não são continuamente diferenciáveis, pode-se encontrar somente um limitado número de trabalhos, daí o estudo de algoritmos desse grupo merecer uma ênfase especial.

No presente trabalho é feito um estudo de alguns algoritmos para minimização unidimensional, agrupados da seguinte forma:

- algoritmos básicos que utilizam o conceito de comparação de função;
- algoritmos básicos baseados em interpolação polinomial;

- algoritmos híbridos com proteções (que utilizam interpolação polinomial e comparação de função);
- algoritmos que fazem a busca do mínimo ao longo de um percurso curvilinear, usando direções de curvatura negativa.

Os algoritmos híbridos com proteções desenvolvidos por Brent (1973) e Gill & Murray (1974) são analisados com maior profundidade, especialmente porque muitas de suas características e propriedades são comuns aos algoritmos específicos para a minimização de funções que não são continuamente diferenciáveis.

Murray & Overton (1978) desenvolveram um algoritmo específico para a minimização de uma classe de funções que não são continuamente diferenciáveis. Trata-se das funções $F(\tilde{x})$ que apresentam a seguinte forma

$$F_S(\tilde{x}) = \sum_{i=1}^{m} \max(0 , f_i(\tilde{x}))$$

onde as funções $\{f_i\}$ são continuamente diferenciáveis e da forma

$$f_i : E^n \rightarrow E^1.$$

Fiacco e Mc Cormick (1968) introduziram a idéia de usar direções de curvatura negativa nos algoritmos de minimização unidimensional. Trata-se de um grupo de algoritmos

que fazem a busca do mínimo ao longo de um percurso curvilinear, usando direções de curvatura negativa. Esses algoritmos fazem a minimização unidimensional de forma diferente dos algoritmos clássicos que buscam um ponto melhor ao longo de um percurso retilíneo.

1.2 Alguns conceitos matemáticos

i) Funções unimodais

Uma função que tem somente um extremo é uma função unimodal.

No caso de funções f(x) unidimensionais, a definição de unimodalidade é a seguinte:

$f(x_1)<f(x_2)<f(x^*)>f(x_3)>f(x_4)$ se $x_1<x_2<x^*<x_3<x_4$
e o extremo é um ponto de máximo

ou

$f(x_1)>f(x_2)>f(x^*)<f(x_3)<f(x_4)$ se $x_1<x_2<x^*<x_3<x_4$
e o extremo é um ponto de mínimo.

Uma função, para ser unimodal, não precisa ser diferenciável e, nem mesmo, contínua.

ii) Continuidade

Uma dada função f(x), definida em uma vizinhança de um ponto x_0, é contínua em x_0 se $\lim_{x \to x_0} f(x)$ existe e

$$\lim_{x \to x_0} f(x) = f(x_0)$$

iii) Diferenciabilidade

Uma dada função f(x), definida em uma vizinhança de um ponto x_0, é diferenciável em x_0 se $\lim_{x \to x_0} \frac{f(x) - f(x_0)}{x - x_0}$ existe.

Se esse limite existe, tem-se

$$f'(x_0) = \lim_{x \to x_0} \frac{f(x) - f(x_0)}{x - x_0}$$

iv) Propriedade de convergência global

A propriedade de convergência global é a propriedade de gerar uma sequência de pontos $x^{(k)}$, k=2,3,... que converge para a solução x^* quando o ponto inicial da sequência $x^{(1)}$ é escolhido arbitrariamente.

v) Taxa de convergência local e ordem de convergência

A taxa de convergência local é a taxa (velocidade) na qual a sequência de pontos $\{x^{(k)}\}$ converge para a solução x^*.

A taxa de convergência local de uma sequência de

pontos gerada por um algoritmo pode ser medida pela ordem de convergência.

A ordem de convergência de uma sequência de pontos $\{x^{(k)}\}$ é definida como o menor dos valores $P > 0$ que satisfaz à relação:

$$\lim_{k \to \infty} \frac{\| x^{(k+1)} - x^* \|}{\| x^{(k)} - x^* \|^P} = \beta < \infty$$

onde $\| . \|$ é uma norma vetorial qualquer e β uma constante assintótica de erro. Valores maiores da ordem de convergência P significam convergência mais rápida.

vi) Taxa de convergência linear

É a taxa com ordem de convergência $P = 1$ e com uma constante assintótica de erro $\beta < 1$.

vii) Taxa de convergência superlinear

É a taxa de convergência com:

- ordem de convergência $P = 1$ e constante assintótica de erro $\beta = 0$;
- ordem de convergência $P = 1$ e constante assintótica de erro $\beta < \infty$.

1.3 Introdução matemática

1.3.1 Minimização unidimensional de funções continuamente diferenciáveis

A seguir, são discutidas as idéias básicas dos algo-

ritmos para minimização unidimensional de funções continuamente diferenciáveis. A função a ser minimizada é da forma

$$\phi(\alpha) = F(\tilde{x} + \alpha\tilde{p})$$

A função $\phi(\alpha)$ é continuamente diferenciável e, por questão de conveniência, serão omitidos os índices que denotam o número da iteração e $F(\tilde{x} + \alpha\tilde{p})$ será expresso como uma função unidimensional da variável α.

Na análise e discussão dos algoritmos, algumas características dos mesmos precisam ser levadas em consideração:

i) localização do mínimo
   - minimização com o mínimo ainda não localizado;
   - minimização com o mínimo localizado.

ii) precisão da pesquisa
   - minimização exata;
   - minimização não exata (para determinação de um ponto suficientemente melhor que um ponto de partida).

iii) tipo de função a ser minimizada
   - função unimodal ou não;
   - função continuamente diferenciável ou não.

iv) informações utilizadas
   - valores da função;
   - valores da primeira derivada.

v) existência ou não de propriedade de convergência global

vi) eficiência do algoritmo

- esforço computacional por iteração
  (número de avaliações do valor da função e, possivelmente, da primeira derivada).
- velocidade de convergência
  (característica que influi no esforço computacional por iteração).

Algoritmos para minimização unidimensional são frequentemente aplicados no processo de solução de um problema mais geral de minimização multidimensional e, em alguns casos, são usados para minimizar funções de uma variável que surgem naturalmente. Na primeira aplicação, é mais viável o uso de algoritmos que, além dos valores da função, fazem uso de informações extras (por exemplo, estimativas da primeira e segunda derivadas) e que não fazem uma minimização exata. Em geral, os critérios de convergência dos processos de minimização não definem um único ponto como solução e sim uma faixa de valores (ver figura 1.1). Embora todos os pontos de uma determinada faixa de valores possam satisfazer ao critério de convergência, isso não significa que todos sejam igualmente viáveis com respeito à eficiência do processo de minimização. Usualmente, se dois pontos satisfazem a um critério de convergência, aquele que corresponde a uma maior redução no valor da função é preferível. Quanto melhor a escolha do escalar α, nesse sentido, menor o número de iterações requeridas para obter uma aproximação satisfatória para o mínimo. Portanto,

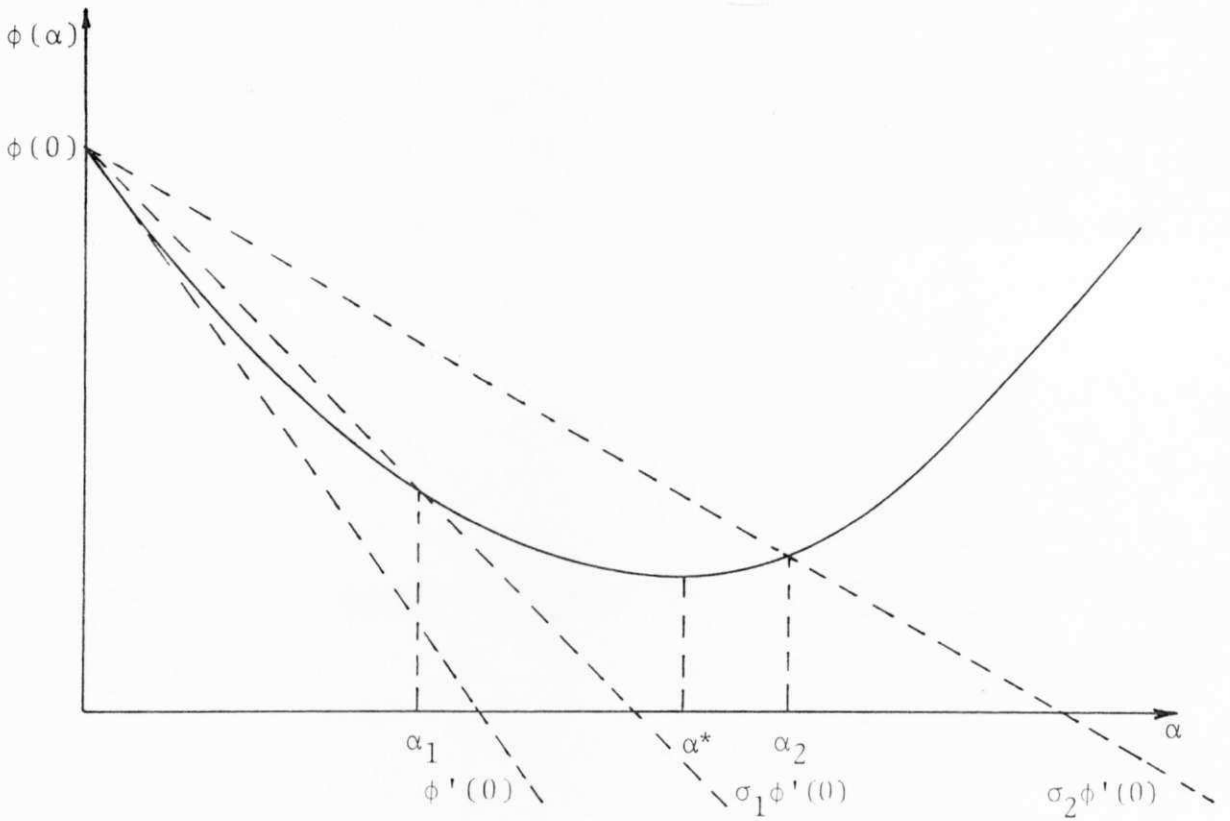


Figura 1.1 : Ilustra o processo de minimização não exata.
Para $0 < \sigma_2 < \sigma_1 \leq 1$, o processo de minimização não exata define a faixa de valores $\alpha_1 < \alpha < \alpha_2$ que satisfazem ao critério de convergência.

a escolha de um escalar α que satisfaça ao critério de convergência não é tudo que se espera de um algoritmo de minimização unidimensional; é preciso fazer uma boa escolha dentre os pontos que satisfazem ao critério. Ao mesmo tempo, entretanto, associado a essa exigência adicional de se fazer uma boa escolha, está o aspecto do esforço computacional exigido. Tanto nos algoritmos de Gill & Murray (1974) como nos algoritmos de Murray & Overton (1978), existe a facilidade de variar esse esforço computacional que pode requerer desde uma simples avaliação de função até uma minimização exata.

No processo de minimização unidimensional, duas classes de métodos são extensivamente usadas e conhecidas: métodos que utilizam o conceito de comparação de função e métodos que utilizam aproximações polinomiais.

Nos métodos que utilizam o conceito de comparação de função, os valores da função em dois pontos internos ao intervalo de pesquisa [a,b] são comparados com o objetivo de reduzir o intervalo que contém o mínimo. A partir dessa comparação, o intervalo inicial é reduzido pela supressão de uma região que não contém o mínimo (ver figura 1.2). Dessa classe de métodos, os mais conhecidos são os métodos de Fibonacci, da Seção Aurea ("Golden Section") e da Dicotomia (Bazarad & Shetty (1974), Adby & Dempster (1974), Gottfried & Weisman (1973)). A principal vantagem dos métodos dessa classe é que sua convergência é garantida para a classe de funções unimodais, mesmo que se trate de fun-

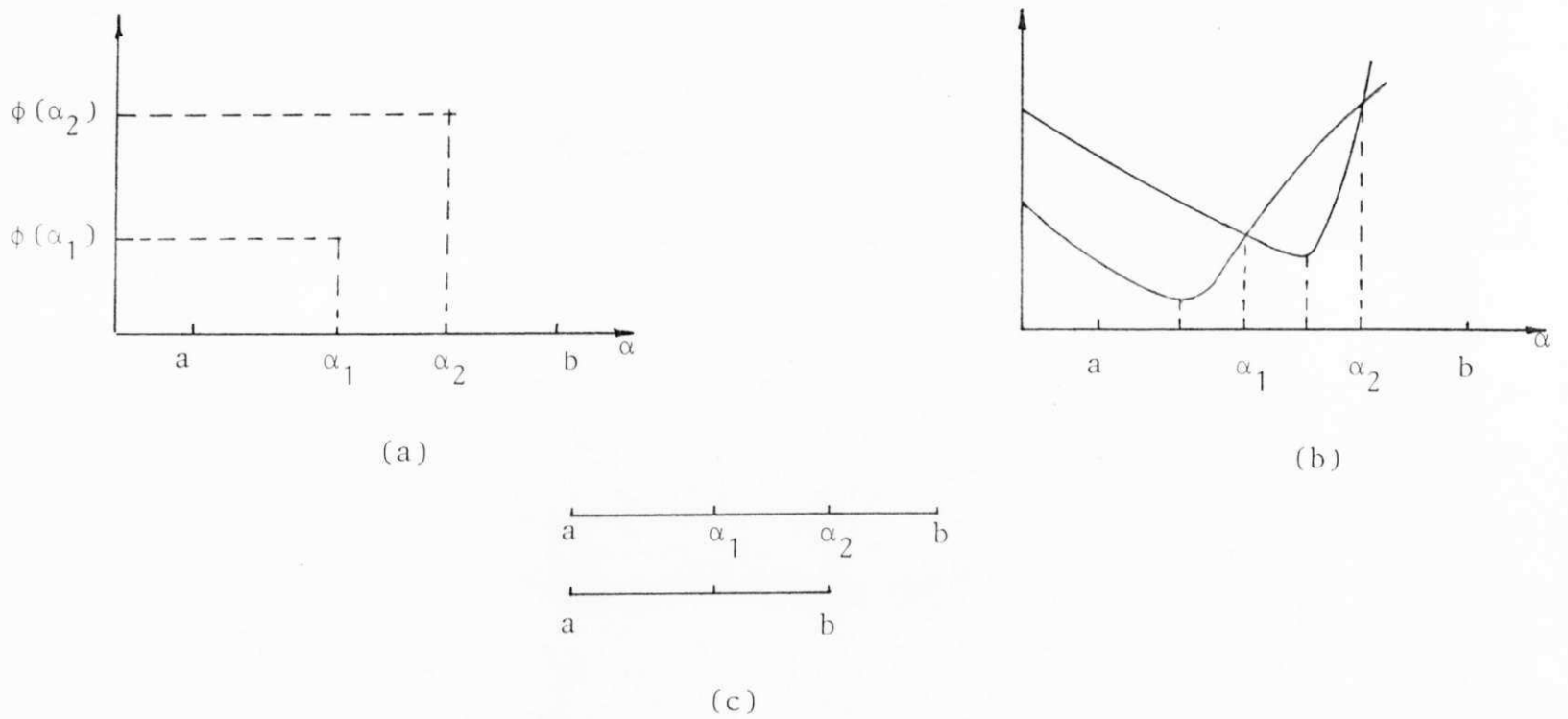


Figura 1.2: Ilustra o processo de comparação de função.

(a) Mostra que, para uma função unimodal, a partir do conhecimento dos valores da função em dois pontos α_1 e α_2, é possível reduzir o intervalo de pesquisa pela supressão de uma região do intervalo que não contém o mínimo. Para $\alpha > \alpha_2$, a função é monotomicamente crescente e, por isso, o mínimo não pode estar nessa região.

(b) Mostra duas possibilidade para a localização do ponto de mínimo.

(c) Mostra o intervalo de pesquisa, antes e depois de ser reduzido.

ções não diferenciáveis e de funções descontínuas. A principal desvantagem desses métodos é a sua lenta convergência.

Nos métodos que utilizam aproximação polinomial, a função a ser minimizada $\phi(\alpha)$ é aproximada por uma função $I(\alpha)$ que coincide com $\phi(\alpha)$ em valor de função ou em valor de função e valor de algumas derivadas em um certo número de pontos a priori escolhidos. $I(\alpha)$ deve ser uma função cujo ponto de mínimo seja facilmente determinado e, por isso, é normalmente escolhida como um polinômio do segundo ou terceiro grau.

Quando a aproximação é feita por polinômios do segundo grau, são necessárias três informações, tais como valores da função e da primeira derivada. Uma prática frequentemente aceita é tomar como informações os valores da funçao em três pontos e, a partir dessas informações, determinar o ponto de mínimo do polinômio. O valor de α que corresponde ao ponto de mínimo do polinômio de aproximação $I(\alpha)$ é usado como aproximação para o mínimo de $\phi(\alpha)$.

Quando a aproximação é feita por polinômios do terceiro grau, são necessárias quatro informações, escolhidas, na maioria das vezes, como os valores da função e da primeira derivada em dois pontos (Interpolação de Hermite).

A principal vantagem dos métodos que utilizam aproximação polinomial é que sua ordem de convergência é superlinear. Teoricamente, a ordem de convergência P é

superlinear com $1 < P < 2$ no caso de aproximações por polinômios do segundo grau e superlinear com $P = 2$ no caso de aproximações por polinômios do terceiro grau. No entanto, existem muitas dificuldades associadas à escolha dos pontos que devem ser usados na aproximação polinomial. Esse aspecto será discutido no parágrafo 2.3.

Diante das desvantagens que apresentam, tanto os métodos de comparação de função como os de aproximação polinomial, conclui-se que um processo de minimização unidimensional baseado somente em um desses métodos é ineficiente ou pouco confiável. Esse problema, no entanto, é contornado por meio de técnicas que fazem uso - ao mesmo tempo - das idéias básicas dos dois métodos.

As desvantagens dos métodos que utilizam comparação de função ou aproximação polinomial, quando usados separadamente, podem ser eliminadas pela combinação dos dois métodos. O método resultante tem propriedade de convergência global garantida - como nos métodos de comparação de função - e sua ordem de convergência é superlinear - como nos métodos de aproximação polinomial -.

Brent (1973) desenvolveu um algoritmo híbrido que utiliza aproximações por polinômios do segundo grau em conjunto com passos de comparação de função baseados no método da Seção Aurea. Análogo ao algoritmo de Brent - quando derivadas são usadas - é o algoritmo que utiliza aproximações por polinômios do terceiro grau em conjunto

com passos de comparação de função baseados no método da Bissecção.

Gill & Murray (1974) desenvolveram um algoritmo em duas versões (com e sem uso de derivadas) que também utiliza aproximações polinomiais em conjunto com um método de comparação de função. Esse algoritmo híbrido se constitui na base para os algoritmos específicos para a minimização de uma classe de funções que não são continuamente diferenciáveis, desenvolvidos por Murray & Overton (1978).

1.3.2 Minimização unidimensional de funções que não são continuamente diferenciáveis

Murray & Overton (1978) desenvolveram algoritmos para a minimização de certas classes de funções que não são continuamente diferenciáveis. Trata-se de algoritmos para a minimização de funções $F(\tilde{x})$ que apresentam a seguinte forma

$$F_S(\tilde{x}) = \sum_{i=1}^{m} \max(0, f_i(\tilde{x})) \qquad (1.1)$$

ou

$$F_M(\tilde{x}) = \max_{1 \leq i \leq m} f_i(\tilde{x}) \qquad (1.2)$$

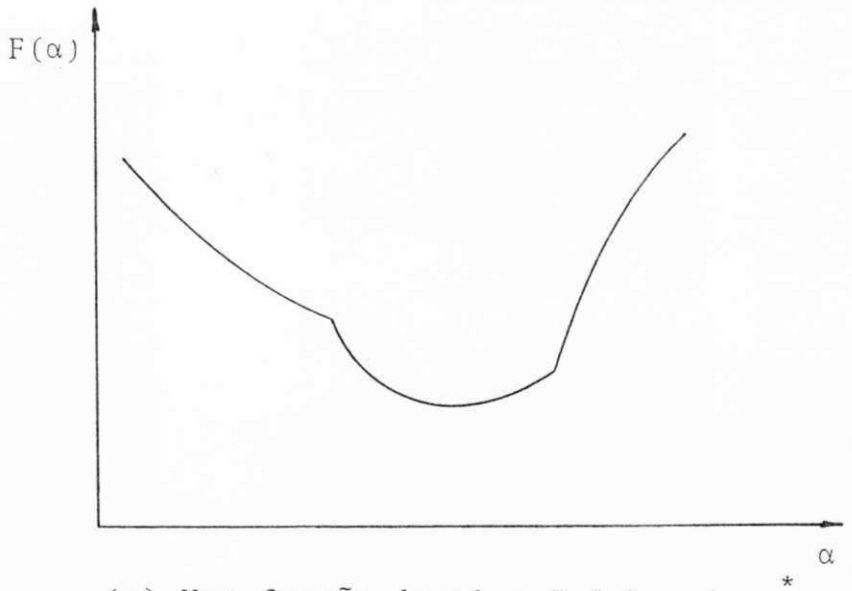
onde as funções $\{f_i\}$ são continuamente diferenciáveis e da forma

$$f_i : E^n \rightarrow E^1$$

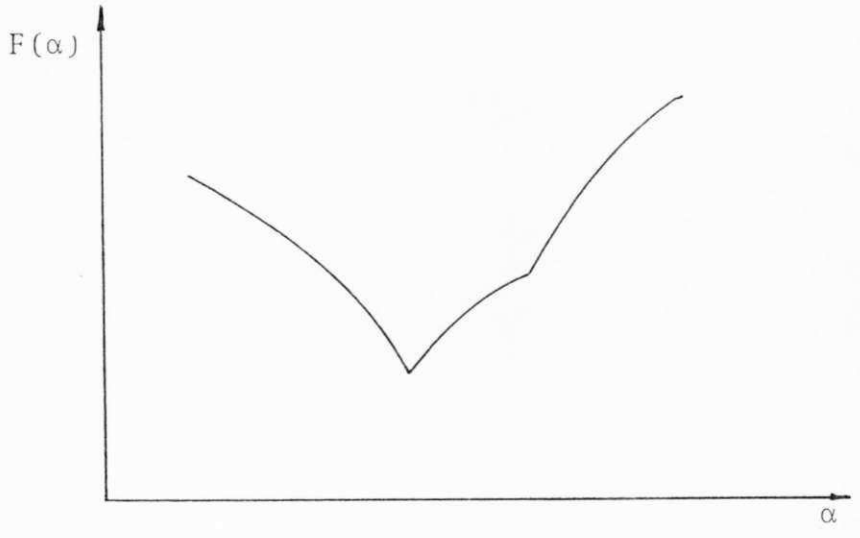
Esses algoritmos podem ser usados na construção de algoritmos para minimizar uma classe mais extensa de funções que não são continuamente diferenciáveis. No presente trabalho serão discutidos os aspectos referentes à minimização de funções do tipo $F_S(\tilde{x})$, definidas a partir da relação (1.1).

A importância da construção de algoritmos especificos para a minimização de funções que não são continuamente diferenciáveis é ilustrada na figura 1.3. Na minimização de funções continuamente diferenciáveis, os métodos mais eficientes usualmente determinam α^* iterativamente, por aproximações polinomiais sucessivas; o polinômio de aproximação coincide com a função $F(\alpha)$ em pontos que correspondem às melhores estimativas de α^* obtidas até então. No caso da figura 1.3-b, é fácil verificar que o minimo do polinômio de aproximação pode não ter qualquer relação com α^*. No caso da figura 1.3-a, o processo de aproximação é válido somente se todos os pontos usados na aproximação se situarem na porção central da curva. Tal situação é improvável de ocorrer nos primeiros estágios do processo de minimização, quando normalmente são conhecidos pontos que representam aproximações muito pobres para α^*.

A seguir, são apresentados alguns conceitos matemáticos referentes à classe de funções definidas a partir da relação (1.1).



(a) Uma função do tipo $F_S(\alpha)$ onde α^* não é uma descontinuidade



(b) Uma função do tipo $F_S(\alpha)$ onde α^* é uma descontinuidade

Figura 1.3: Ilustra a importância de algoritmos específicos para minimizar funçoes que não são continuamente diferenciáveis.

Os gradientes das m funções componentes $f_i(\tilde{x})$ são representados por $\nabla f_i(\tilde{x})$ e define-se

$$\tilde{g}(\tilde{x}) = \sum_{i:f_i(\tilde{x})>0} \nabla f_i(\tilde{x})$$

$\tilde{g}(\tilde{x})$ é o gradiente de $F(\tilde{x})$ onde o mesmo é definido ou é uma das derivadas direcionais de $F(\tilde{x})$ nos pontos onde o gradiente não é definido.

Para um dado ponto de pesquisa $\tilde{x}$ e uma dada direção de pesquisa $\tilde{p}$, pode-se representar as funções unidimensionais $f_i(\alpha)$ e $F(\alpha)$ da seguinte maneira

$$f_i(\alpha) = f_i(\tilde{x} + \alpha\tilde{p}), \quad i = 1,\ldots,m$$

e

$$F(\alpha) = F(\tilde{x} + \alpha\tilde{p}).$$

Dessa forma, tem-se

$$F(\alpha) = \sum_{i=1}^{m} \max(0, f_i(\alpha))$$

A derivada de uma função $f_i(\alpha)$, que é o gradiente projetado de $f_i(\tilde{x} + \alpha\tilde{p})$ ao longo de $\tilde{p}$, é representada por

$$f_i'(\alpha) = \nabla f_i(\tilde{x} + \alpha\tilde{p})^T \tilde{p}$$

e as derivadas direcionais à esquerda e à direita são denotadas por $F_-'(\alpha)$ e $F_+'(\alpha)$.

Define-se também

$$F'(\alpha) = \tilde{g}(\tilde{x} + \alpha\tilde{p})^T \tilde{p}$$

Portanto,

$$F'(\alpha) = \sum_{i:f_i(\alpha)>0} f_i'(\alpha)$$

$F'(\alpha)$ é a derivada de $F(\alpha)$ onde a mesma for definida ou, em caso contrário, é uma de suas derivadas direcionais.

Um ponto onde a derivada de $F(\alpha)$ não for definida é referido como um ponto de descontinuidade da derivada ou, simplesmente, uma descontinuidade. Para uma função do tipo $F_S(\alpha)$, os pontos de descontinuidade da derivada coincidem com os zeros das funções componentes $f_i(\alpha)$.

Murray & Overton (1978) desenvolveram um algoritmo específico para a minimização de funções do tipo $F_S(\alpha)$. O algoritmo apresenta duas versões, versão simplificada e versão com refinamentos. Cada versão apresenta duas variantes, dependendo do uso ou não de derivadas como informações. O principal aspecto do algoritmo é que em cada iteração do processo de minimização procura-se distinguir se α^* é ou não um ponto de descontinuidade da derivada e selecionar uma aproximação para α^* de acordo com o que parecer mais provável.

1.4 Organização dos capítulos

Os algoritmos básicos de que trata o Capítulo 2 são apresentados com o objetivo de tornar mais completa a discussão do problema da minimização unidimensional. Com isso, procurou-se assegurar à dissertação o aspecto didático, permitindo uma discussão dos métodos de minimização, desde os mais simples e básicos até aqueles mais refinados que lidam com dificuldades adicionais como a não diferenciabilidade da função a ser minimizada. Dentre os algoritmos básicos, estão incluídos os que utilizam o conceito de comparação de função, os algoritmos baseados em interpolação polinomial e os algoritmos para determinação de um intervalo fechado que contenha o mínimo.

Os algoritmos híbridos com proteções são apresentados no Capítulo 3. São descritos os algoritmos de Brent e de Gill & Murray para a minimização de funções continuamente diferenciáveis. Também é descrito o algoritmo de Murray & Overton para a minimização de uma classe de funções que não são continuamente diferenciáveis. Por fim, é apresentada uma introdução teórica sobre os algoritmos que fazem a busca do mínimo ao longo de um percurso curvilinear, usando direções de curvatura negativa. São descritos dois algoritmos desse grupo: algoritmo de segunda ordem de Armijo (Goldfarb, 1980) e algoritmo de segunda ordem de Goldstein (Goldfarb, 1980).

O Capítulo 4 trata dos testes de desempenho dos algoritmos híbridos com proteções. Esses testes foram divididos em duas categorias: testes de desempenho usando funções de teste continuamente diferenciáveis e testes de desempenho usando funções de teste que não são continuamente diferenciáveis. A princípio, São apresentados resultados do desempenho do algoritmo de Brent e do algoritmo de Gill & Murray (versão sem derivadas) e, em seguida, resultados do desempenho do algoritmo da Bissecção e do algoritmo de Gill & Murray (versão com derivadas), usados para minimizar funções continuamente diferenciáveis. No caso de funções que não são continuamente diferenciáveis, os resultados se referem ao desempenho dos algoritmos de Brent e algoritmo de Murray & Overton (versão simplificada, sem uso de derivadas) e do algoritmo da Bissecção e algoritmo de Murray & Overton (versão simplificada, com uso de derivadas), usados na minimização de funções de teste do tipo $F_S(\alpha)$, elaboradas pelo autor.

2 ALGORITMOS BASICOS

2.1 Algoritmo para localizar um intervalo fechado que contém o mínimo

Na determinação de um mínimo local da função unidimensional $\phi(\alpha) = F(\tilde{x} + \alpha\tilde{p})$ é prática usual procurar-se obter um intervalo inicial de pesquisa ("bracket") que contenha o minimizante α^* de $\phi(\alpha)$ e, a partir daí, usar técnicas de minimização unidimensional para o refinamento desse intervalo.

No processo de localização do mínimo, uma estratégia comumente usada consiste em avaliar a função $\phi(\alpha)$ em pontos afastados de um ponto inicial α_0 de múltiplos crescentes de um incremento inicial h, até que se obtenha um maior valor da função que o obtido na avaliação anterior. Essa estratégia é ilustrada na figura 2.1.

Wolfe (1978) desenvolveu um eficiente algoritmo para localização de um intervalo fechado que contém o mínimo. O algoritmo é apresentado em duas versões. Na primeira versão são usados valores da função associados a

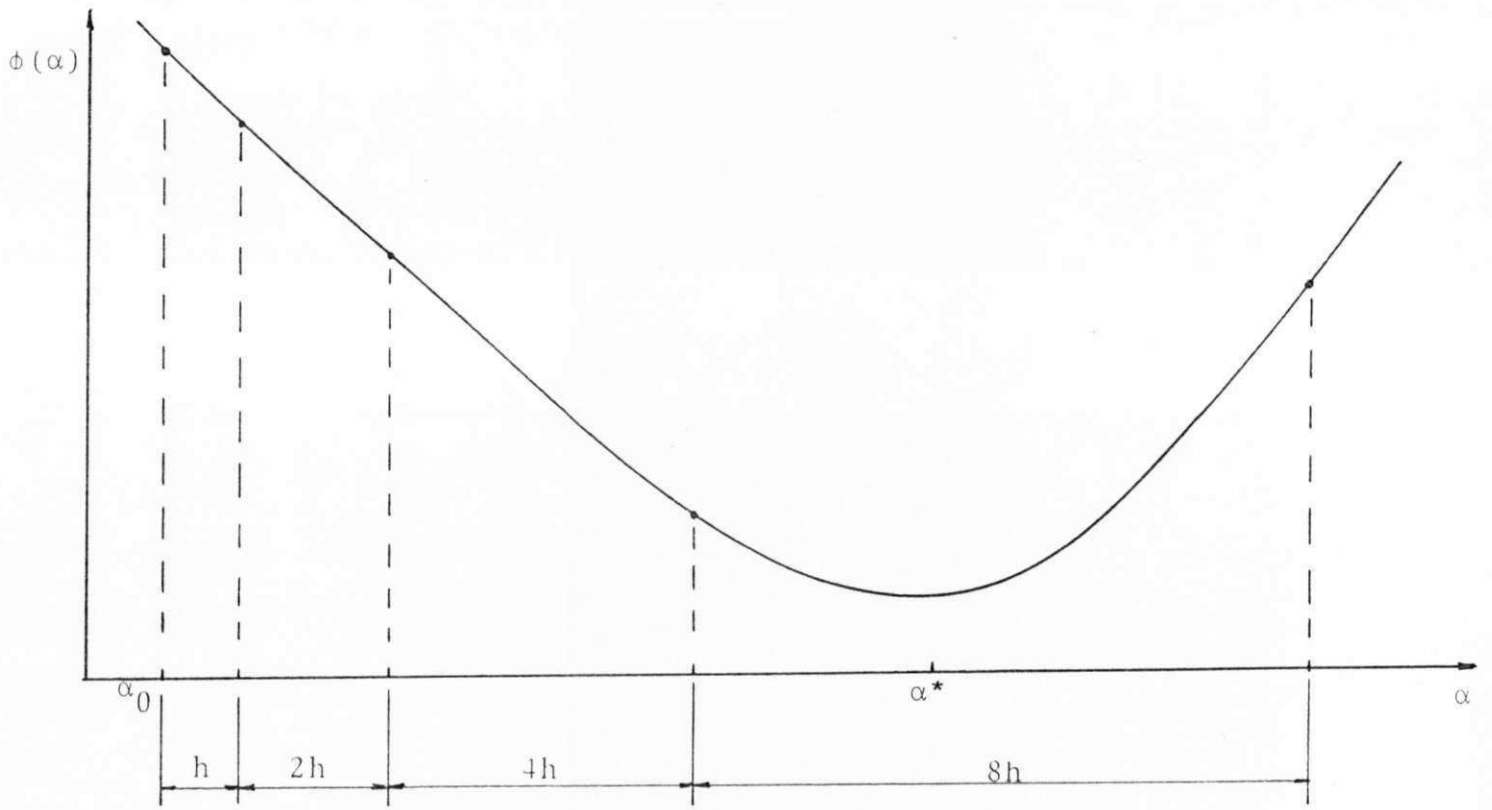


Figura 2.1: Ilustra o processo de localização do mínimo, a partir da avaliação da função $\phi(\alpha)$ em pontos afastados de um ponto inicial α_0 de múltiplos crescentes de um incremento inicial h.

três pontos. Na segunda versão são usados valores da função e da primeira derivada associados a dois pontos.

Determinação do incremento inicial h

Wolfe (1978) indica uma maneira sistemática para se obter um valor adequado do incremento h, a partir das informações disponíveis no inicio de cada iteração de um método descendente, a saber, $\phi(0) = F(\tilde{x}^{(k)})$ e $\phi'(0) = \tilde{g}(\tilde{x}^{(k)})^T \tilde{p}^{(k)}$, bem como de uma estimativa do valor da função multidimensional $F(\tilde{x})$ no seu ponto de minimo $\tilde{x}^*$.

Supondo-se que a função unidimensional $\phi(\alpha)$ seja quadrática, isto é,

$$\phi(\alpha) = \frac{1}{2} a \alpha^2 + b \alpha + c \qquad (2.1)$$

Então

$$\phi'(\alpha) = a\alpha + b \qquad (2.2)$$

de onde

$$\alpha^* = \frac{-b}{a} \qquad (2.3)$$

e

$$\phi(\alpha^*) = \frac{-b^2}{2a} + c \qquad (2.4)$$

A equação anterior pode ser reescrita como

$$\phi(\alpha^*) = (b/2)(-b/a) + c \qquad (2.5)$$

A partir de (2.3), tem-se

$$\phi(\alpha^*) = \frac{b}{2}\,\alpha^* + c \qquad (2.6)$$

e

$$\alpha^* = \frac{2(\phi(\alpha^*) - c)}{b} \qquad (2.7)$$

A partir de (2.1) e (2.2), tem-se $\phi(0) = c$ e $\phi'(0) = b$. Daí,

$$\alpha^* = \frac{2(\phi(\alpha^*) - \phi(0))}{\phi'(0)} \qquad (2.8)$$

Caso o valor de $\phi(\alpha^*)$ fosse conhecido, α^* poderia ser determinado a partir da equação (2.8). Isso sugere um método para obtenção de um valor de h que seja pelo menos da mesma ordem de grandeza do minimizante α^*. Seja f_E uma estimativa do valor da função $F(\tilde{x})$ no seu ponto de mínimo $\tilde{x}^*$. Supondo-se que $\phi(\alpha)$ seja uma função quadrática de α e que

$$\tilde{x}^* = \tilde{x}^{(k)} + \alpha^* \tilde{p}^{(k)}$$

onde α^* é o minimizante de $\phi(\alpha)$, de forma que

$$\phi(\alpha^*) = f_E \qquad (2.9)$$

Então, a partir de (2.8) e (2.9), chega-se a um valor de h dado por

$$h = \frac{2(f_E - \phi(0))}{\phi'(0)} \qquad (2.10)$$

Determinação de um intervalo fechado que contém o mínimo (caso sem derivadas)

Nessa primeira versão do algoritmo apresentado por Wolfe (1978), procura-se obter três valores $\{\alpha_1, \alpha_2, \alpha_3\}$ de forma que o minimizante α^* da função unidimensional $\phi(\alpha)$ se situe entre α_1 e α_3.

Para um valor inicial α_0 e um incremento inicial $h > 0$, obtido a partir da equação (2.10), avalia-se $\phi(\alpha_0)$ e $\phi(\alpha_0 + h)$. Caso $\phi(\alpha_0 + h) > \phi(\alpha_0)$, avalia-se $\phi(\alpha_0 - h)$. Caso $\phi(\alpha_0 - h) > \phi(\alpha_0)$, então o mínimo está localizado pelos pontos α_1, α_2 e α_3 dados por

$$\alpha_1 = \alpha_0 - h$$

$$\alpha_2 = \alpha_0 \qquad (2.11)$$

$$\alpha_3 = \alpha_0 + h$$

Essa situação é ilustrada na figura 2.2.

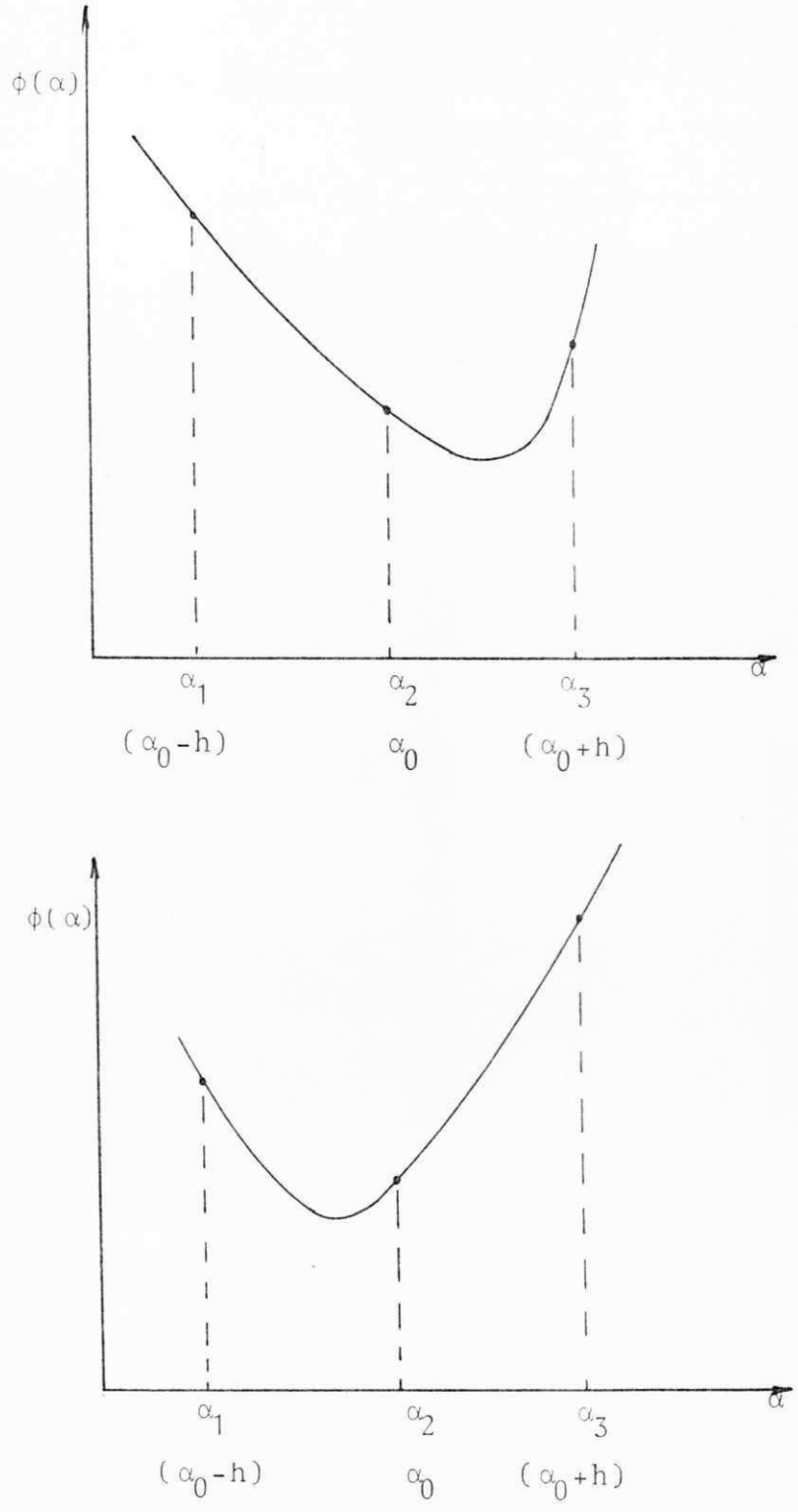


Figura 2.2: Ilustra o processo de obtenção do intervalo que contém o mínimo.

Caso $\phi(\alpha_0 + h) \leq \phi(\alpha_0)$, avalia-se $\phi(\alpha_0 + 2^K h)$ $(K = 1,2,...)$ até que

$$\phi(\alpha_0 + 2^K h) > \phi(\alpha_0 + 2^{K-1} h) \tag{2.12}$$

e

$$\phi(\alpha_0 + 2^{K-1} h) < \phi(\alpha_0 + 2^{K-2} h) \tag{2.13}$$

Sejam

$$A = \alpha_0 + 2^{K-2} h$$

$$B = \alpha_0 + 2^{K-1} h \tag{2.14}$$

$$C = \alpha_0 + 2^K h$$

Então,

$$\phi(A) \geq \phi(B) < \phi(C)$$

e

$$C - B = 2(B - A)$$

Ver figura 2.3.

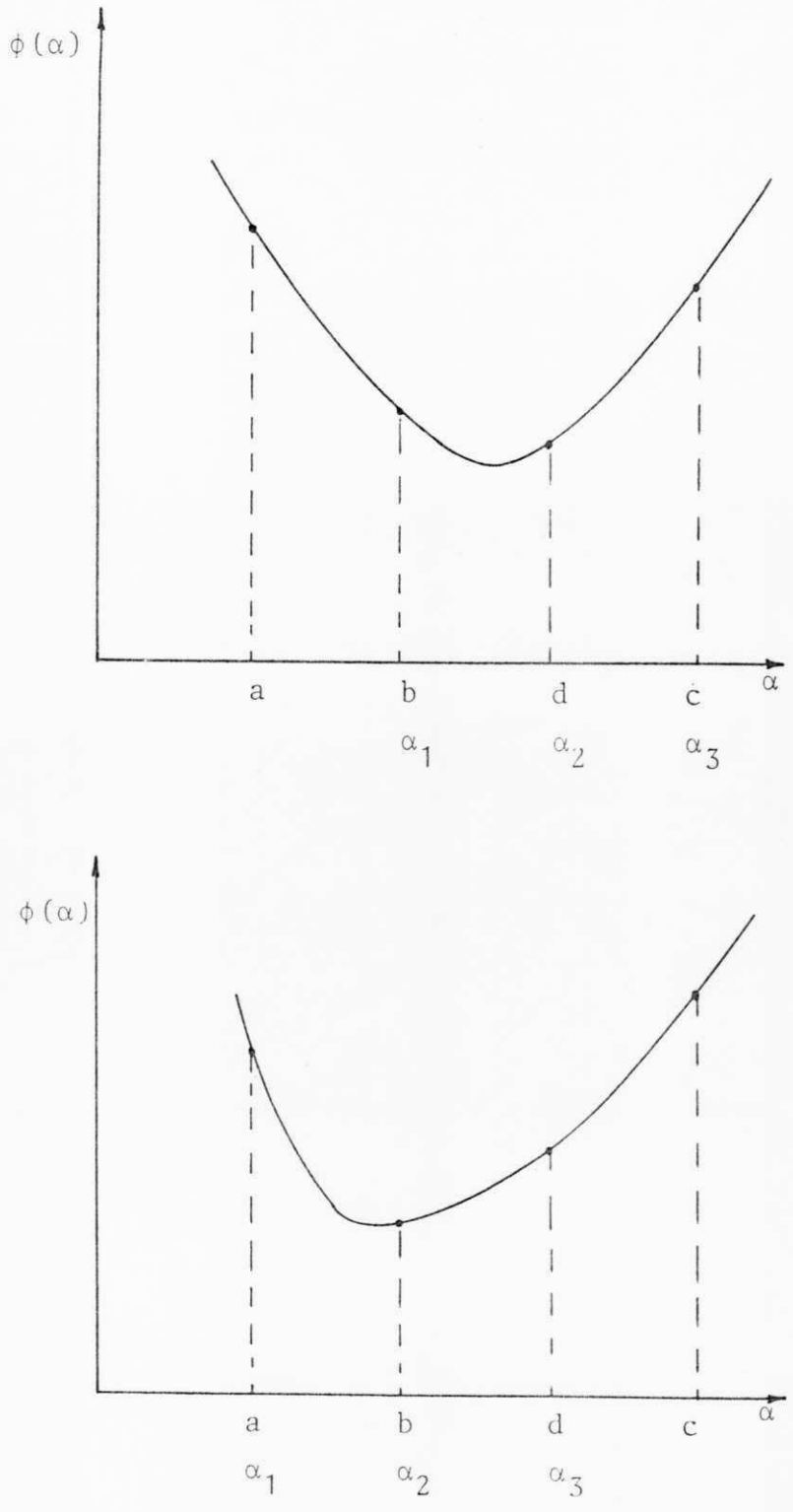


Figura 2.3: Ilustra o processo de seleção dos pontos α_1, α_2 e α_3

Seja

$$D = \frac{1}{2}(B + C)$$

Avalia-se $\phi(D)$. Caso $\phi(D) < \phi(B)$, tem-se $\alpha_1 = B$, $\alpha_2 = D$ e $\alpha_3 = C$. Caso $\phi(D) \geq \phi(B)$, tem-se $\alpha_1 = A$, $\alpha_2 = B$ e $\alpha_3 = D$.

Se, inicialmente, $\phi(\alpha_0 + h) > \phi(\alpha_0)$, então avalia-se $\phi(\alpha_0 - 2^K h)$ $(K = 1,2,...)$ até que as relações (2.12) e (2.13) sejam satisfeitas com h substituído por -h. Sejam

$$A = \alpha_0 - 2^{K-2}h$$

$$B = \alpha_0 - 2^{K-1}h$$

$$C = \alpha_0 - 2^K h$$

$$D = \frac{1}{2}(B + C)$$

Avalia-se $\phi(D)$. Caso $\phi(D) < \phi(B)$, tem-se $\alpha_1 = C$, $\alpha_2 = D$ e $\alpha_3 = A$. Ver figura 2.4.

Determinação de um intervalo fechado que contém o mínimo (caso com derivadas)

Nessa segunda versão do algoritmo, procura-se obter dois valores α_1 e α_2 de forma que o minimizante α^* da função $\phi(\alpha)$ se situe entre α_1 e α_2.

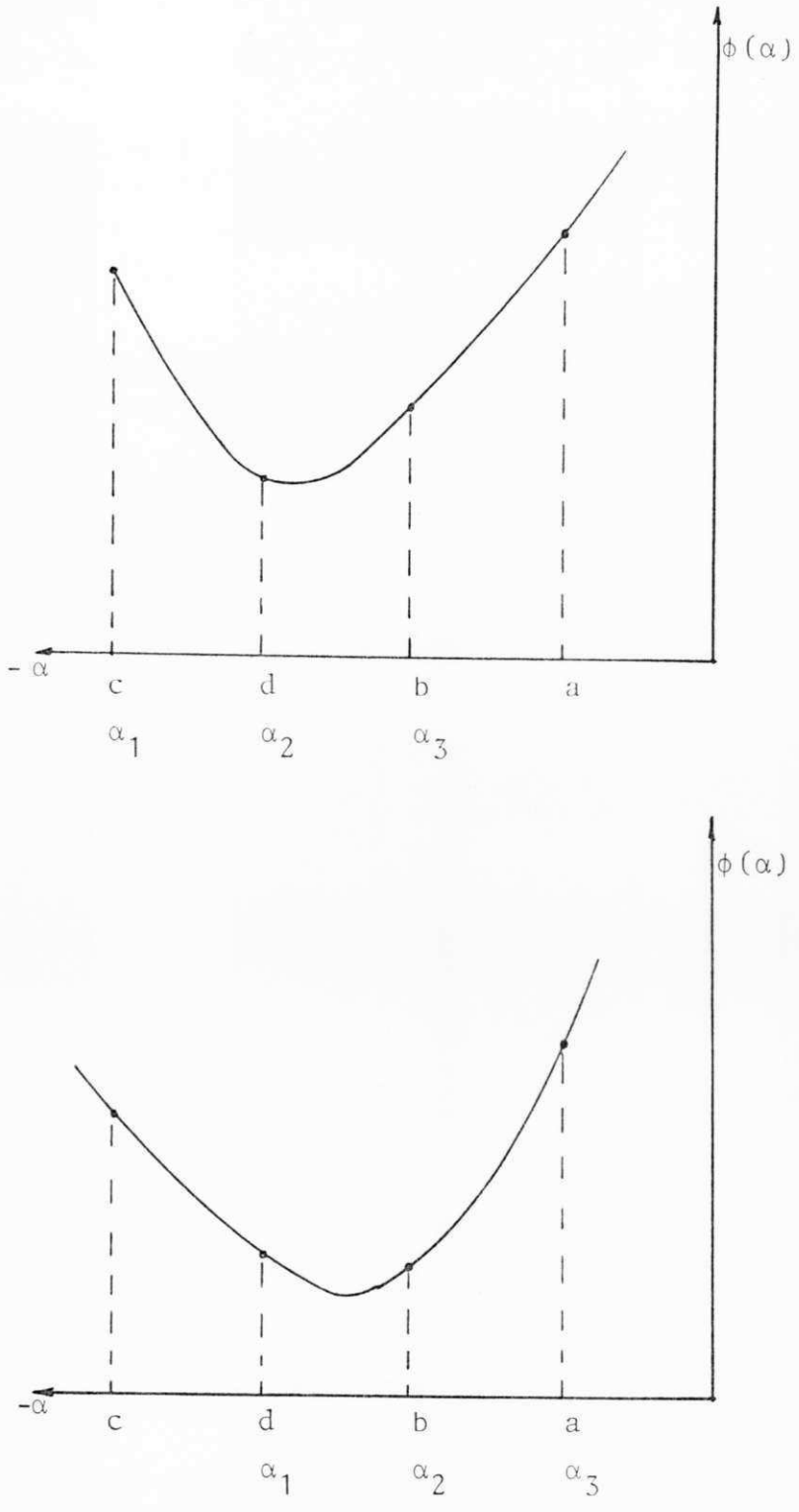


Figura 2.4: Ilustra o processo de obtenção do "bracket" quando, inicialmente, $\phi(\alpha_0 + h) > \phi(\alpha_0)$.

Para um incremento inicial $h > 0$, obtido a partir da equação (2.10), avalia-se $\phi(\alpha)$ para $\alpha = 0,\ h,\ 2h,\ 4h,\ 8h,\ \ldots,\ \alpha_1,\ \alpha_2$ onde α_1 e α_2 são os dois primeiros valores consecutivos da sequência $2^K h$ tais que $\phi'(\alpha_1) < 0$, $\phi'(\alpha_2) \geq 0$ e $\phi(\alpha_2) \geq \phi(\alpha_1)$.

2.2.1 Algoritmo de Fibonacci

O algoritmo de Fibonacci é o mais conhecido e mais eficiente dos algoritmos que utilizam o conceito de comparação de função na minimização de funções unimodais em um intervalo fechado (Bazarad & Shetty (1974)).

Seu procedimento é baseado na sequência de números de Fibonacci, definida como segue:

$$F_{k+1} = F_k + F_{k-1} \qquad k = 1, 2, \ldots \tag{2.15}$$

$$F_0 = F_1 = 1$$

A sequência é, portanto, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, ...

Seja $[a_k, b_k]$ o intervalo de pesquisa na k-ésima iteração e dois pontos λ_k e μ_k, localizados simetricamente nesse intervalo. Para n avaliações de função a serem efetuadas, tem-se

$$\lambda_k = a_k + \frac{F_{(n-k-1)}}{F_{(n-k+1)}} (b_k - a_k)$$

$$= b_k - \frac{F_{(n-k)}}{F_{(n-k+1)}} (b_k - a_k), \quad k = 1,\ldots,n-1 \qquad (2.16)$$

$$\mu_k = a_k + \frac{F_{(n-k)}}{F_{(n-k+1)}} (b_k - a_k)$$

$$= b_k - \frac{F_{(n-k-1)}}{F_{(n-k+1)}} (b_k - a_k), \quad k = 1,\ldots,n-1 \qquad (2.17)$$

Redução do intervalo de pesquisa

A seguir, procura-se mostrar como varia, de uma iteração para outra, a taxa de redução do intervalo de pesquisa. Caso $\phi(\lambda_k) > \phi(\mu_k)$, tem-se para o comprimento do novo intervalo

$$b_{(k+1)} - a_{(k+1)} = b_k - \lambda_k$$

Tomando-se a expressão de λ_k a partir de (2.16),

$$b_{k+1} - a_{k+1} = b_k - a_k - \frac{F_{(n-k-1)}}{F_{(n-k+1)}} (b_k - a_k)$$

$$= \left(1 - \frac{F_{(n-k-1)}}{F_{(n-k+1)}} \right) (b_k - a_k)$$

$$= \frac{F_{(n-k+1)} - F_{(n-k-1)}}{F_{(n-k+1)}} (b_k - a_k)$$

Pela relação (2.15), $F_{(n-k+1)} - F_{(n-k-1)} = F_{(n-k)}$. Assim,

$$b_{k+1} - a_{k+1} = \frac{F_{(n-k)}}{F_{(n-k+1)}} (b_k - a_k) \qquad (2.18)$$

No segundo caso, isto é, $\phi(\lambda_k) \leq \phi(\mu_k)$, tem-se

$$b_{k+1} - a_{k+1} = \mu_k - a_k$$

Tomando-se a expressão de μ_k a partir da relação (2.17) e procedendo de maneira análoga, obtém-se para o comprimento do novo intervalo

$$b_{k+1} - a_{k+1} = \frac{F_{(n-k)}}{F_{(n-k+1)}} (b_k - a_k)$$

Vale observar em (2.18) que a taxa de redução do intervalo de pesquisa $F_{(n-k)}/F_{(n-k+1)}$ é função do número da iteração e, por isso, varia de uma iteração para outra.

Número de avaliações da função

Na primeira iteração, dois pontos λ_1 e μ_1 são localizados no intervalo de pesquisa e duas avaliações de função são necessárias, isto é, $\phi(\lambda_1)$ e $\phi(\mu_1)$. Nas iterações subsequentes somente um ponto é localizado e uma avaliação de função é necessária, já que $\lambda_{k+1} = \mu_k$ ou $\mu_{k+1} = \lambda_k$. Ao final da iteração (n-2) foram feitas (n-1) avaliações da função.

Fazendo-se k = n-1 em (2.16) e (2.17), observa-se

que os pontos λ_{n-1} e μ_{n-1} coincidem com o ponto médio do intervalo $[a_{n-1}, b_{n-1}]$, isto é,

$$\lambda_{n-1} = \mu_{n-1} = (a_{n-1} + b_{n-1})/2.$$

Desde que $\lambda_{n-1} = \mu_{n-2}$ ou $\mu_{n-1} = \lambda_{n-2}$, o valor da função no ponto médio do intervalo na penúltima iteração é conhecido a partir da iteração anterior e, por isso, não é necessária uma avaliação adicional do valor da função.

Escolha do número de avaliações da função (n)

Diferentemente dos outros métodos que utilizam comparação de função, o método de Fibonacci requer que o número total de avaliações de função (n) seja escolhido antecipadamente. Isso porque a localização dos pontos internos ao intervalo é feita a partir das equações (2.16) e (2.17) e, portanto, depende de n.

A partir de (2.18) pode-se observar que o comprimento do intervalo de pesquisa é reduzido na k-ésima iteração por um fator $F_{(n-k)}/F_{(n-k+1)}$. Assim, ao final de (n-1) iterações, o comprimento do intervalo é reduzido de $b_1 - a_1$ para $b_n - a_n = (b_1 - a_1)/F_n$. Portanto, n deve ser escolhido de forma que $(b_1 - a_1)/F_n$ reflita a precisão requerida na pesquisa.

Essa necessidade de se especificar previamente o número de avaliações n constitui-se em uma desvantagem do método, especialmente quando aplicado a um processo de

minimização não exata. A título de ilustração, pode-se considerar o caso em que a função $\phi(\alpha)$ deva ser minimizada de forma que a solução aceitável seja uma dada fração do valor da função no ponto inicial de pesquisa e o número de avaliações não possa portanto ser previamente estabelecido.

Convergência global

O algoritmo tem a propriedade de convergência global para a classe de funções unimodais.

Taxa de convergência

O algoritmo tem taxa de convergência linear com respeito ao intervalo de pesquisa, isto é, os comprimentos dos intervalos de pesquisa consecutivos decrescem linearmente.

Sequência de passos computacionais
do algoritmo 2.2.1

1. Especifique o intervalo inicial $[a_1, b_1]$ e o comprimento do intervalo final $L > 0$.

2. Se $(b_1 - a_1) < L$, PARE.

3. Gere a sequência de números de Fibonacci até o termo F_n, de forma que a relação $F_n \geq 2(b_1 - a_1)/L$ seja satisfeita.

4. Calcule

$\lambda_1 = a_1 + [F_{n-2}/F_n](b_1 - a_1)$

$\mu_1 = b_1 - [F_{n-2}/F_n](b_1 - a_1)$

5. Avalie $\phi(\lambda_1)$ e $\phi(\mu_1)$

6. Inicialize o contador de iterações

   $k = 1$

7. Se $\phi(\lambda_k) = \phi(\mu_k)$,

   renomeie os pontos $a_1 = \lambda_k$, $b_1 = \mu_k$ e vá para 2.

8. Se $\phi(\lambda_k) < \phi(\mu_k)$,

   - renomeie os pontos $a_{k+1} = \lambda_k$; $b_{k+1} = b_k$; $\lambda_{k+1} = \mu_k$
   - calcule μ_{k+1}
   - se $k = (n-2)$, vá para 10.
   - avalie $\phi(\mu_{k+1})$
   - $k = k + 1$
   - vá para 7.

9. - Renomeie os pontos

     $a_{k+1} = a_k$; $b_{k+1} = \mu_k$; $\mu_{k+1} = \lambda_k$.

   - calcule λ_{k+1}
   - se $k = (n-2)$, vá para 10.
   - avalie $\phi(\lambda_{k+1})$
   - $k = k + 1$
   - vá para 7.

10. Tome o ponto médio do intervalo $[a_{n-1}, b_{n-1}]$ como aproximação para o mínimo e PARE.

2.2.2 Algoritmo da seção áurea (Algoritmo modificado de Fibonacci)

O algoritmo da seção áurea ("golden section algorithm") em muito se assemelha ao algoritmo de Fibonacci. Difere, basicamente, nos aspectos referentes à localização dos pontos internos ao intervalo de pesquisa e à redução do intervalo de pesquisa que, no caso do algoritmo da seção áurea, se faz a uma taxa constante.

Seja $[a_k, b_k]$ o intervalo de pesquisa na k-ésima iteração e dois pontos λ_k e μ_k, localizados simetricamente nesse intervalo. Tem-se

$$\lambda_k = a_k + (1 - R)(b_k - a_k) \tag{2.19}$$

$$\mu_k = a_k + R(b_k - a_k) \tag{2.20}$$

onde R = 0.618034 é o coeficiente da seção áurea.

O por quê do uso do coeficiente R nas relações (2.19) e (2.20) será esclarecido no próximo parágrafo. Vale observar que (1 - R) = 0.381966 e, assim, λ_k será localizado a, aproximadamente, 38% do comprimento do intervalo $[a_k, b_k]$ à direita de a_k. O mesmo ocorre com μ_k que será localizado a 38% do comprimento do intervalo à esquerda de b_k. Como após a comparação dos valores da função $\phi(\alpha)$ nos pontos λ_k e μ_k se tem $a_{k+1} = \lambda_k$ ou $b_{k+1} = \mu_k$, o novo intervalo será 61,8% menor que o anterior, isto é,

$$b_{k+1} - a_{k+1} = R(b_k - a_k)$$

Redução do intervalo de pesquisa

Sejam I_k, I_{k+1} e I_{k+2} os comprimentos do intervalo de pesquisa em três iterações sucessivas. Se esses comprimentos são tais que

$$I_k = I_{k+1} + I_{k+2} \tag{2.21}$$

e a relação dos comprimentos dos intervalos sucessivos é igual a uma constante K, isto é,

$$\frac{I_k}{I_{k+1}} = \frac{I_{k+1}}{I_{k+2}} = K \tag{2.22}$$

o valor numérico da constante K pode ser determinado a partir das relações (2.21) e (2.22). A partir de (2.22), tem-se

$$I_k = K^2 I_{k+2} \tag{2.23}$$

Dividindo-se (2.21) por I_{k+2},

$$\frac{I_k}{I_{k+2}} = \frac{I_{k+1}}{I_{k+2}} + 1 \tag{2.24}$$

Substituindo-se (2.22) e (2.23) em (2.24),

$$K^2 = K + 1 \qquad (2.25)$$

Então

$$K = (1 \pm \sqrt{5})/2$$

Tomando-se a raiz positiva, tem-se K = 1.618034.

No algoritmo da seção áurea, os sucessivos intervalos de pesquisa têm comprimentos que satisfazem às relações (2.21) e (2.22) com K = 1.618034. Vale observar que 1/K = R = 0.618034 e

$$\frac{I_{k+1}}{I_k} = \frac{I_{k+2}}{I_{k+1}} = R \qquad (2.26)$$

O coeficiente da seção áurea R = 0.618034 é relacionado com a sequencia de números de Fibonacci da seguinte maneira

$$\lim_{n\to\infty} \frac{F_{n-1}}{F_n} = \lim_{n\to\infty} \frac{(1+R)^{n-1} - (-R)^{n-1}}{(1+R)^n - (-R)^{n-1}}$$

$$\cong \frac{(1+R)^{n-1}}{(1+R)^n} = \frac{1}{1+R}$$

$$= R = 0.618034...$$

Os números da sequência de Fibonacci F_n podem ser determinados usando-se a fórmula recursiva

$$F_n = F_{n-1} + F_{n-2}, \quad n = 2, 3, \ldots$$

com condições iniciais

$$F_1 = F_0 = 1$$

ou usando a fórmula fechada

$$F_n = \frac{(1+R)^n - (-R)^n}{(1+R) - (-R)}$$

A aplicação da fórmula fechada é muito conveniente para determinação do limite da expressão F_{n-1}/F_n quando $n \to \infty$.

Número de avaliações da função

Na primeira iteração do processo dois pontos λ_1 e μ_1 são localizados e duas avaliações de função são necessárias, isto é, $\phi(\lambda_1)$ e $\phi(\mu_1)$. Nas iterações subsequentes somente um ponto é localizado e uma avaliação de função se faz necessária. O número total de avaliações necessárias para reduzir o intervalo de pesquisa até um comprimento final L não é necessariamente determinado antecipadamente, como no algoritmo de Fibonacci. Dessa forma, surge a necessidade de se introduzir um teste para verificar se o intervalo que contém o minimizante α^* é menor que o comprimento

final desejado L, quando então o processo iterativo termina.

Convergência global

O algoritmo tem a propriedade de convergência global para a classe de funções unimodais.

Taxa de convergência

O algoritmo tem taxa de convergência linear com respeito ao intervalo de pesquisa (os comprimentos dos intervalos de pesquisa consecutivos decrescem linearmente).

Sequência de passos computacionais
do algoritmo 2.2.2

1. Especifique o intervalo inicial $[a_1, b_1]$, o comprimento do intervalo final $L > 0$ e o número máximo de iterações k_{max}.

2. Se $(b_1 - a_1) < L$, PARE.

3. Calcule
   $D = R^2 (b_1 - a_1)$
   $\lambda_1 = a_1 + D$
   $\mu_1 = b_1 - D$

4. Avalie $\phi(\lambda_1)$ e $\phi(\mu_1)$

5. Inicialize o contador de iterações
   $k = 1$

6. Se $(b_k - a_k) < L$, tome o ponto médio do intervalo $[a_k, b_k]$ como aproximação para o minimo e PARE.

7. Se $\phi(\lambda_k) = \phi(\mu_k)$, renomeie os pontos $a_1 = \lambda_k$; $b_1 = \mu_k$ e vá para 2.

8. Faça D = R.D

9. Se $\phi(\lambda_k) < \phi(\mu_k)$,
   - renomeie os pontos
     $a_{k+1} = \lambda_k$; $b_{k+1} = b_k$; $\lambda_{k+1} = \mu_k$
   - calcule $\mu_{k+1} = b_{k+1} - D$
   - avalie $\phi(\mu_{k+1})$
   - k = k + 1
   - vá para 11.

10. - Renomeie os pontos
      $a_{k+1} = a_k$; $b_{k+1} = \mu_k$; $\mu_{k+1} = \lambda_k$
    - calcule $\lambda_{k+1} = a_{k+1} + D$
    - avalie $\phi(\lambda_{k+1})$
    - k = k + 1

11. Se $k < k_{max}$, vá para 6.

12. PARE.

2.2.3 Algoritmo da Dicotomia

A idéia básica do algoritmo consiste em localizar dois pontos λ_k e μ_k internamente no intervalo de pesquisa de maneira que, a partir da comparação dos valores $\phi(\lambda_k)$ e

$\phi(\mu_k)$, o novo intervalo de pesquisa seja o menor possível. Como antes de se proceder à comparação dos valores $\phi(\lambda_k)$ e $\phi(\mu_k)$ não se sabe qual dos dois é menor, procura-se localizar os pontos λ_k e μ_k de forma a minimizar o comprimento do próximo intervalo de pesquisa $[a_{k+1}, b_{k+1}]$ que será o máximo entre $(\mu_k - a_k)$ e $(b_k - \lambda_k)$ (Bazarad & Shetty (1974), Adby & Dempster (1974), Gottfried & Weisman (1973)).

De acordo com a ilustração da figura 2.5, para dois pontos λ_k e μ_k localizados simetricamente, cada um a uma distância $\varepsilon > 0$ do ponto médio, o comprimento do novo intervalo de pesquisa será $(b_k - a_k)/2 + \varepsilon$. Vale observar que esse comprimento tende a decrescer à medida que diminui a separação entre os pontos de prova λ_k e μ_k. Portanto, a estratégia ótima consiste em localizar dois pontos de prova simetricamente com respeito ao ponto médio do intervalo de pesquisa e separados entre si de uma distância que seja a menor possível.

O escalar ε deve ser suficientemente pequeno de forma que o comprimento do novo intervalo $(b_k - a_k)/2 + \varepsilon$ seja bastante próximo do valor teórico ótimo $(b_k - a_k)/2$. No entanto, não se pode reduzir o valor de ε arbitrariamente, uma vez que os valores $\phi(\lambda_k)$ e $\phi(\mu_k)$ precisam ser distinguíveis.

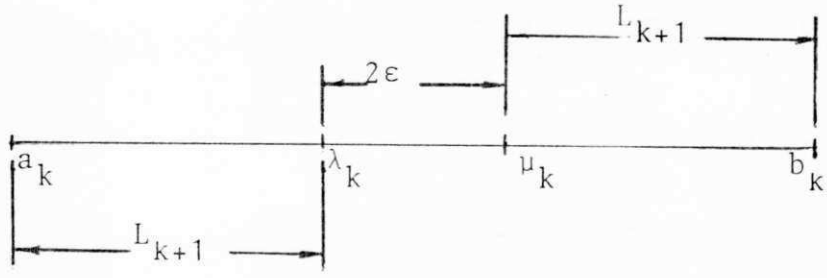


Figura 2.5: Ilustra o método da Dicotomia.

Redução do intervalo de pesquisa e
número de avaliações da função

Em cada iteração do processo dois pontos λ_k e μ_k são localizados internamente no intervalo de pesquisa e duas avaliações da função são necessárias.

O comprimento do intervalo de pesquisa no início da iteração (k+1) é dado por (Bazarad & Shetty (1974)):

$$(b_{k+1} - a_{k+1}) = \frac{1}{2}^k (b_1 - a_1) + 2\varepsilon(1 - \frac{1}{2}^k)$$

Essa fórmula pode ser usada para se determinar o número de iterações necessárias para se reduzir o intervalo até um valor especificado. Como em cada iteração são feitas duas avaliações da função, a fórmula pode também ser usada para se determinar o número de avaliações da função.

Convergência global

O algoritmo tem a propriedade de convergência glo-

bal para a classe de funções unimodais.

Taxa de convergência

O algoritmo tem taxa de convergência linear com respeito ao intervalo de pesquisa (os intervalos de pesquisa consecutivos decrescem linearmente).

Sequência de passos computacionais do algoritmo 2.2.3

1. Especifique o intervalo inicial $[a_1,b_1]$, o comprimento do intervalo final $L > 0$ e o número máximo de iterações k_{max}.

2. Inicialize o contador de iterações
   $k = 1$

3. Se $(b_k - a_k) < L$, tome o ponto médio do intervalo $[a_k,b_k]$ como aproximação para o mínimo e PARE.

4. Calcule
   $\lambda_k = (a_k + b_k)/2 - \varepsilon$
   $\mu_k = (a_k + b_k)/2 + \varepsilon$

5. Avalie $\phi(\lambda_k)$ e $\phi(\mu_k)$

6. Se $\phi(\lambda_k) < \phi(\mu_k)$,
   - renomeie os pontos $a_{k+1} = a_k$; $b_{k+1} = \mu_k$
   - $k = k + 1$
   - vá para 8.

7. - Renomeie os pontos

$a_{k+1} = \lambda_k$; $b_{k+1} = b_k$

- $k = k + 1$

8. Se $k < k_{max}$, vá para 3.

9. PARE.

2.3 Algoritmos baseados em interpolação polinomial

O principal aspecto dessa classe de algoritmos é que a função unidimensional $\phi(\alpha) : R^n \text{ --- } R^1$ a ser minimizada é aproximada por um polinômio que coincide com $\phi(\alpha)$ em valor da função ou em valor da função e valor de algumas derivadas em um certo número de pontos, a priori escolhidos. Na discussão que se segue supõe-se que $\phi(\alpha)$ é uma função unimodal.

Interpolação quadrática

Sejam x, w e v três valores distintos de α, tais que $\phi(x) \leq \phi(w) \leq \phi(v)$. Os pontos x, w e v satisfazem a uma das seguintes configurações: $w < x < v$ e $v < x < w$. Se a função $\phi(\alpha)$ é unimodal e o minimizante α^* se situa entre os pontos w e v, isto é,

$w < \alpha^* < v$

ou

$v < \alpha^* < w$

o ponto de mínimo do polinômio do segundo grau que passa pelos pontos $(x,\phi(x))$, $(w,\phi(w))$ e $(v,\phi(v))$ é dado por $\hat{u} = x + p/q$, onde

$$p = \pm[(x-v)^2\ (\phi(x)-\phi(w)) - (x-w)^2\ (\phi(x)-\phi(v))]$$

$$q = \mp 2[(x-v)\ (\phi(x)-\phi(w)) - (x-w)\ (\phi(x)-\phi(v))] \qquad (2.27)$$

Interpolação cúbica

Sejam x e w dois valores distintos de α tais que $\phi(x) \leq \phi(w)$ e que satisfazem a uma das configurações

$x < w$, com $\phi'(x) \leq 0$ e $\phi'(w) > 0$

ou

$x > w$, com $\phi'(x) > 0$ e $\phi'(w) \leq 0$

Se a função $\phi(\alpha)$ é unimodal e o minimizante α^* se situa entre os pontos x e w, o ponto de mínimo do polinômio do terceiro grau que passa pelos pontos $(x,\phi(x))$ e $(w,\phi(w))$ com derivadas $\phi'(x)$ e $\phi'(w)$, respectivamente, é dado por $x + p/q$, onde

$$p = \pm(w-x)[\phi'(x) - \gamma - \eta]$$

$$q = \frac{}{+}[\phi'(w) - \phi'(x) + 2\gamma] \qquad (2.28)$$

e

$$\gamma = \text{sinal}(w-x)[\eta^2 - \phi'(x)\,\phi'(w)]^{1/2}$$

$$\eta = 3(\phi(x)-\phi(w))/(w-x) + \phi'(x) + \phi'(w)$$

Se os erros de arredondamento são desprezíveis e o minimizante α^* está situado entre dois dos pontos usados na interpolação polinomial, o ponto de mínimo do polinômio de aproximação, obtido a partir da relação (2.27) ou (2.28) deverá também se situar entre esses dois pontos e o intervalo será reduzido. No entanto, o valor da função no novo ponto não é necessariamente menor que em todos os pontos usados na aproximação.

A principal dificuldade relacionada com o uso repetido das equações (2.27) e (2.28) é que isso implica na supressão de um dos pontos em cada estágio. Se o ponto correspondente ao maior valor da função for suprimido, o intervalo que contém o novo conjunto de pontos pode não conter o minimizante e, nesse caso, as fórmulas acima podem não levar a resultados confiáveis, uma vez que estão sendo usadas para extrapolação. Uma solução alternativa é garantir que o intervalo que contém o novo conjunto de pontos

sempre contém o minimizante α^*. No entanto, isso pode tornar a convergência do processo bastante lenta.

Nas seções 2.3.1 e 2.3.2 são apresentados dois algoritmos baseados em interpolação polinomial. Os algoritmos fazem uso das equações (2.27) e (2.28) e se prestam à minimização de funções unidimensionais $\phi(\alpha)$ em um intervalo fechado que contenha o minimizante α^*. Ao final de cada iteração, os vários pontos são renomeados de forma que o novo conjunto de pontos contenha o minimizante α^*.

2.3.1 Algoritmo baseado em interpolação quadrática com informações associadas a três pontos

Seja [a,b] um intervalo fechado que contém o minimizante α^* da função unidimensional $\phi(\alpha)$. Sejam x, w e v três valores distintos de α, tais que $\phi(x) \leq \phi(w) \leq \phi(v)$. Tem-se $a < x < b$ e duas configurações possíveis para os pontos w e v:

$w = a \quad e \quad v = b$

$w = b \quad e \quad v = a$

Escolha a tolerância tol e faça k = 0.

1. $k = k + 1$

2. Calcule o ponto médio

   $m = (a + b)/2$

3. Verifique o critério de convergência:

   Se $\max(x-a, b-x) < tol$, tome o ponto x como a solução ótima e PARE.

4. Calcule $\hat{u}$ (ponto de minimo do polinômio quadrático), fazendo

   $r = (x-w)[\phi(x) - \phi(v)]$

   $q = (x-v)[\phi(x) - \phi(w)]$

   $p = (x-v)q - (x-w)r$

   $q = 2(q-r)$

   Se $q \leq 0$, faça $q = -q$. Em caso contrário, faça

   $p = -p$.

   Faça $\hat{u} = x + p/q$.

5. Avalie $\phi(\hat{u})$.

6. Renomeie os pontos a, b, x, w, v e os correspondentes valores $\phi(x)$, $\phi(w)$ e $\phi(v)$ da seguinte maneira:

   Se $\phi(\hat{u}) \leq \phi(x)$, faça

      Se $\hat{u} \geq x$, $a = x$; $w = x$; $x = \hat{u}$

      Se $\hat{u} < x$, $b = x$; $w = x$; $x = \hat{u}$; $v = w$

   Em caso contrário, faça

Se $\hat{u} \geq x$, $b = \hat{u}$

Se $\phi(\hat{u}) < \phi(w)$, $v = w$; $w = \hat{u}$

Se $\phi(\hat{u}) \geq \phi(w)$, $v = \hat{u}$

Se $\hat{u} < x$, $a = \hat{u}$

Se $\phi(\hat{u}) < \phi(v)$, $w = \hat{u}$

Se $\phi(\hat{u}) \geq \phi(v)$, $w = v$; $v = \hat{u}$.

7. Vá para 1.

Convergência global

O algoritmo tem a propriedade de convergência global para a classe de funções unimodais.

Taxa de convergência

O algoritmo tem taxa de convergência superlinear.

2.3.2 Algoritmo baseado em interpolação cúbica de Hermite com informações associadas a dois pontos

Seja [a,b] um intervalo fechado que contém o minimizante α^* da função unidimensional $\phi(\alpha)$. Sejam x e w dois valores distintos de α tais que $\phi(x) \leq \phi(w)$ e que satisfazem a uma das configurações

$x = a$ e $w = b$, com $\phi'(x) \leq 0$ e $\phi'(w) > 0$

ou

$x = b$ e $w = a$, com $\phi'(x) > 0$ e $\phi'(w) \leq 0$.

Escolha a tolerância tol e faça $k = 0$.

1. $k = k + 1$

2. Verifique o critério de convergência:
   Se $(b-a) \leq$ tol, tome o ponto x como a solução ótima e PARE.

3. Atribua valores às variáveis auxiliares x_1, f_1, g_1, x_2, f_2 e g_2, fazendo

   Se $x=a$ e $w=b$, faça $x_1 = a$; $f_1 = \phi(x)$; $g_1 = \phi'(x)$

   $x_2 = b$; $f_2 = \phi(w)$; $g_2 = \phi'(w)$

   Se $x=b$ e $w=a$, faça $x_1 = a$; $f_1 = \phi(w)$; $g_1 = \phi'(w)$

   $x_2 = b$; $f_2 = \phi(x)$; $g_2 = \phi'(x)$

4. Calcule $\hat{u}$ (ponto de mínimo do polinômio do terceiro grau), fazendo

   $\eta = g_1 + g_2 - 3(f_1 - f_2)/(x_1 - x_2)$

   Se $|g_1| > |g_2|$, $e = |g_1|[(\eta/g_1)^2-(g_2/g_1)]^{1/2}$

   Se $|g_1| < |g_2|$, $e = |g_2|[(\eta/g_2)^2-(g_1/g_2)]^{1/2}$

   $r = g_2 - g_1 + 2e$

$s = -g_1 + \eta + e$

$t = s/r$

$\hat{u} = (1-t)x_1 + tx_2$

5. Avalie $\phi(\hat{u})$ e $\phi'(\hat{u})$.

6. Renomeie os pontos a, b, x, w e os correspondentes valores $\phi(x)$, $\phi'(x)$, $\phi(w)$ e $\phi'(w)$ da seguinte maneira:

   Se $\phi(\hat{u}) \leq \phi(x)$, faça

      Se $\phi'(\hat{u}) \leq 0$, $a = \hat{u}$; $x = \hat{u}$

      Se $\phi'(\hat{u}) > 0$, $b = \hat{u}$; $x = \hat{u}$

   Em caso contrário, faça

      Se $\hat{u} < x$, $a = \hat{u}$; $w = \hat{u}$

      Se $\hat{u} > x$, $b = \hat{u}$; $w = \hat{u}$

7. Vá para 1.

Convergência global

O algoritmo apresenta convergência global para a classe de funções unimodais que sejam continuamente diferenciáveis.

Taxa de convergência

O algoritmo tem taxa de convergência superlinear.

3 ALGORITMOS HIBRIDOS COM PROTEÇOES

3.1 Algoritmos hibridos com proteções para a minimização de funções continuamente diferenciáveis

Nesta seção são apresentados os conceitos básicos, estratégia e o conjunto de proteções dos algoritmos hibridos com proteções para minimização de funções continuamente diferenciáveis: Algoritmo de Brent, Algoritmo da Bissecção e Algoritmo de Gill & Murray em suas duas versões (com e sem uso de derivadas como informações). Também é apresentado o critério de convergência e, para cada algoritmo, é feita separadamente uma descrição do método de comparação de função.

Conceitos básicos, estratégia dos algoritmos e conjunto de proteções

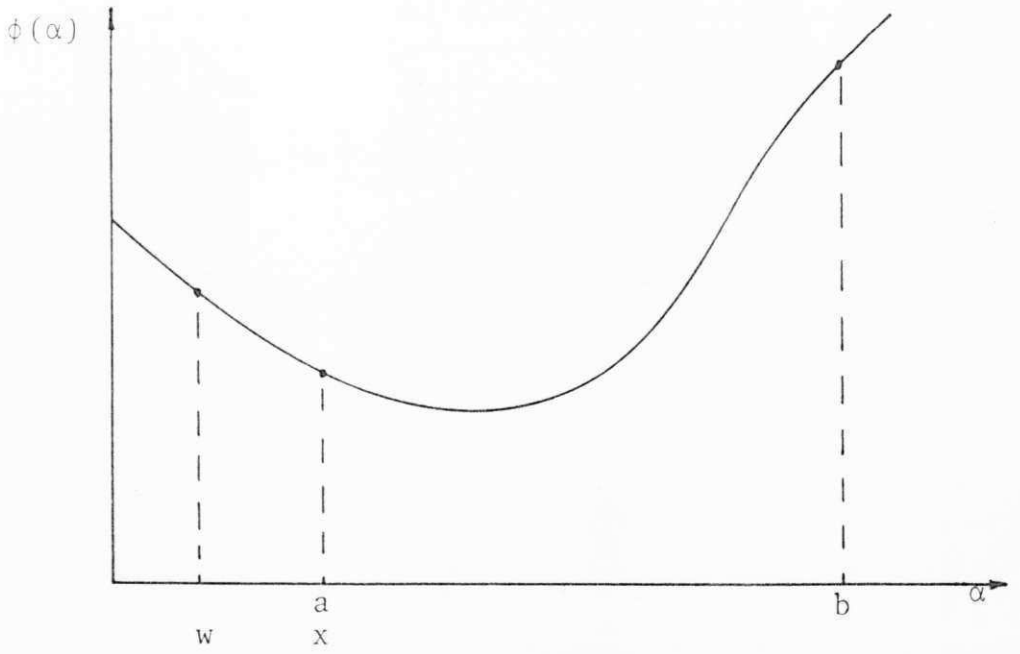
Assume-se que em cada iteração do processo de minimização é conhecido um intervalo de pesquisa [a,b] que contém o ponto de minimo α^* da função que se deseja minimizar.

Quando são usados como informações os valores da função e da primeira derivada associados a dois pontos, assume-se que dois pontos x e w são também conhecidos. O ponto x é aquele associado ao menor valor da função até então (isto é, $\phi(x) < \phi(\alpha)$ para qualquer valor de α onde a função já foi avaliada) e w é o ponto correspondente ao segundo menor valor da função ou é o último ponto onde a função foi avaliada. Ver figura 3.1-a. Existem quatro configurações possíveis para os pontos a, b, x e w:

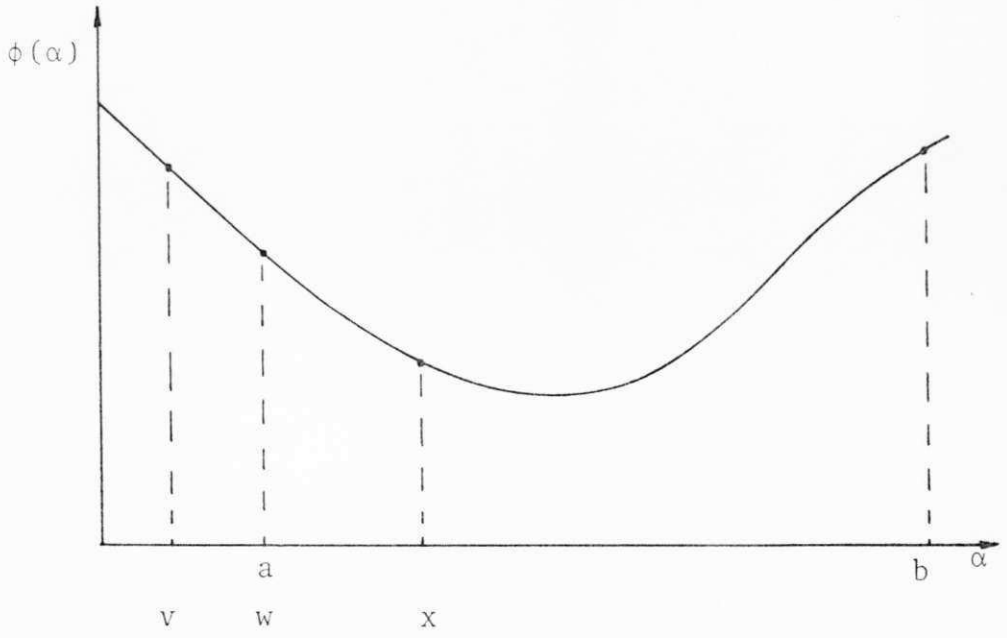
$$\begin{aligned} &\text{i)}\ x = a \quad e \quad w < a \\ &\text{ii)}\ x = a \quad e \quad w = b \\ &\text{iii)}\ x = b \quad e \quad w = a \\ &\text{iv)}\ x = b \quad e \quad w > b \end{aligned} \tag{3.1}$$

Quando são usados como informações os valores da função associados a três pontos, assume-se que três pontos x, w e v são também conhecidos; x é o ponto onde a função tem o menor valor até então, w é o ponto onde a função tem o segundo menor valor e v é o ponto onde a função tem o terceiro menor valor ou é o último ponto onde a função foi avaliada. Ver figura 3.1-b. Tem-se $a < x < b$ e quatro configurações possíveis para w e v:

$$\begin{aligned} &\text{i)}\ w = a \quad e \quad v < a \qquad &&\text{iii)}\ w = b \quad e \quad v = a \\ &\text{ii)}\ w = a \quad e \quad v = b \qquad &&\text{iv)}\ w = b \quad e \quad v > b \end{aligned} \tag{3.2}$$



(a)



(b)

Figura 3.1: (a) configuração (i) da relação (3.1)
(b) configuração (i) da relação (3.2)

A estratégia básica dos algoritmos consiste em gerar um novo ponto $\hat{u}$ (primeira previsão), a partir das informações disponíveis nos pontos x e w ou x, w e v, tais como valor da função e da primeira derivada. O novo ponto $\hat{u}$ é obtido através de interpolação polinomial com proteções (o termo interpolação é usado aqui para significar interpolação propriamente dita ou extrapolação). No caso em que derivadas são usadas, o polinômio interpolador é do terceiro grau e coincide com $\phi(\alpha)$ e $\phi'(\alpha)$ nos pontos x e w. No caso em que derivadas não são usadas, o polinômio interpolador é do segundo grau e coincide com $\phi(\alpha)$ nos pontos x, w e v. Assim, em cada iteração, o ponto de mínimo do polinômio interpolador é tomado como a previsão $\hat{u}$.

O algoritmo de Brent e o algoritmo de Gill & Murray (versão sem derivadas) utilizam interpolação quadrática em combinação com métodos de comparação de função. O algoritmo da Bissecção e o algoritmo de Gill & Murray (versão com derivadas) utilizam interpolação cúbica em combinação com métodos de comparação de função.

Em cada iteração existe um intervalo [a,b] que contém o mínimo, mas as aproximações polinomiais são sempre feitas nos pontos correspondentes aos menores valores da função, isto é, pontos x e w quando derivadas são usadas e pontos x, w e v quando derivadas não são usadas. A taxa de convergência dos algoritmos que utilizam aproximações polinomiais em combinação com comparação de função é quadrática no caso com derivadas e é superlinear com ordem 1,324...

no caso sem derivadas. No entanto, essas taxas só são efetivas caso as aproximações sejam feitas nos pontos onde a função tem os menores valores até então, sendo que esses pontos não localizam o mínimo necessariamente.

O ponto $\hat{u}$, gerado a partir de uma aproximação polinomial, precisa satisfazer certas condições para que venha a ser aceito como aproximação do mínimo da função $\phi(\alpha)$ naquela iteração, isto é, sob determinadas condições tem-se $u = \hat{u}$, onde u é a aproximação aceita na iteração. Quando o ponto $\hat{u}$ não satisfaz a uma das condições, um outro ponto $\bar{u}$, obtido por um método de comparação de função é aceito, isto é, $u = \bar{u}$. Essas condições são resumidas no seguinte conjunto de proteções, isto é, u é feito igual a $\hat{u}$, exceto nos seguintes casos:

i) $\hat{u}$ se situa fora do intervalo [a,b]. Isso normalmente ocorre quando se tem as configurações (i) ou (iv) das relações (3.1) ou (3.2). Entretanto, devido a erros de arredondamento, pode ocorrer mesmo quando a configuração for de interpolação. Caso $\hat{u}$ não se situe em [a,b], u é feito igual a um outro ponto $\bar{u}$, obtido por comparação de função.

ii) $\hat{u}$ é obtido por extrapolação e, mesmo que se situe em [a,b], $\bar{u}$ se situa entre $\hat{u}$ e x. Nesse caso também u é feito igual a $\bar{u}$. Essa proteção só faz parte do algoritmo de Gill & Murray.

iii) A distância $|x-\hat{u}|$ na iteração corrente é maior que

metade da distância $|x-u|$ na antepenúltima iteração. O propósito dessa restrição é assegurar a redução do incremento dado ao ponto x por um fator de pelo menos dois, a cada duas iterações. Esse procedimento evita que seja gerada uma sequência de estimativas u que oscilem em torno do mínimo, tornando lenta a redução do intervalo de pesquisa.

iv) O ponto û se situa muito próximo a um dos pontos onde a função já foi avaliada: a, x ou b. Nesse caso, u é feito igual a um outro ponto separado daqueles onde a função já foi avaliada por uma distância mínima tol(x), onde

$$tol(\alpha) = \varepsilon|\alpha| + \tau$$

ε e τ são dois parâmetros relacionados com a precisão da máquina. No caso do ponto $\bar{u}$, também pode ocorrer de o mesmo se situar muito próximo a um dos pontos a, x ou b, sendo necessário gerar um novo ponto afastado pela distância tol(x).

Após a escolha do ponto u, é feita a avaliação de (u) (e, possivelmente, de $\phi'(u)$) e os vários pontos e os correspondentes valores da função e da derivada são renomeados de acordo com o procedimento apresentado a seguir.

Renomeação dos pontos

i) Caso com derivadas

Se $\phi(\hat{u}) \leq \phi(x)$, faça

$w = x$

$x = \hat{u}$

Se $\phi'(\hat{u}) \leq 0$, $a = \hat{u}$

Se $\phi'(\hat{u}) > 0$, $b = \hat{u}$

Em caso contrário, faça

$w = \hat{u}$

Se $\hat{u} \leq x$, $a = \hat{u}$

Se $\hat{u} > x$, $b = \hat{u}$

ii) Caso sem derivadas

Se $\phi(\hat{u}) \leq \phi(x)$, faça

Se $\hat{u} \geq x$, $a = x$

Se $\hat{u} < x$, $b = x$

$v = w$

$w = x$

$x = \hat{u}$

Em caso contrário, faça

Se $\hat{u} < x$, $a = \hat{u}$

Se $\hat{u} > x$, $b = \hat{u}$

Se $\phi(\hat{u}) < \phi(w)$, $v = w$; $w = \hat{u}$

Se $\phi(\hat{u}) \geq \phi(w)$, $v = \hat{u}$

Critério de convergência

Considera-se que o processo de minimização unidi-

mensional converge quando

$max(x-a,b-x) < 2tol(x)$ (caso sem derivadas)

ou

$(b-a) < 2tol(x)$ (caso com derivadas)

Métodos de comparação de função

i) Algoritmo de Brent

No algoritmo de Brent, o método de comparação de função é baseado nas idéias do método da seção áurea. A previsão modificada (ponto $\bar{u}$) é dada por

$$\bar{u} = \begin{cases} x + \beta(a-x), & x > m \\ x + \beta(b-x), & x < m \end{cases} \qquad (3.3)$$

onde m é o ponto médio do intervalo de pesquisa e

$\beta = (3 - \sqrt{5})/2 \cong 0.381966...$

O processo de obtenção do ponto $\bar{u}$ de acordo com a relação (3.3) consiste em, inicialmente, verificar qual é o maior dentre os subintervalos [a,x] e [x,b]. O maior desses subintervalos é selecionado e o ponto $\bar{u}$ é localizado a, aproximadamente, 38% do comprimento do subintervalo, do ponto x.

Seguindo-se o procedimento do método da seção áu-

rea, dois pontos λ e μ são localizados internamente no intervalo de pesquisa (no caso, o intervalo de pesquisa para aplicação do método da seção áurea coincide com o maior dentre os subintervalos [a,x] e [x,b]). Dos pontos λ e μ, a previsão $\bar{u}$ é feita igual àquele que se situar mais próximo do ponto x.

ii) Algoritmo da Bissecção

No caso do algoritmo da Bissecção, a previsão modificada $\bar{u}$ é feita igual ao ponto médio do intervalo [a,b], isto é,

$$\bar{u} = (a + b)/2.$$

iii) Algoritmo de Gill & Murray

O método de comparação de função será descrito inicialmente para o caso sem derivadas. Consiste em estabelecer um limite artificial m_α em [a,b] dado por

$$m_\alpha = \begin{cases} x + \beta'(a-x), & w > x \\ x + \beta'(b-x), & w < x \end{cases} \qquad (3.4\text{-a})$$

onde

$$\beta' = \begin{cases} \frac{1}{2}\,(-d_1/d_2)^{1/2}, & |d_1| < |d_2| \\ \frac{5}{11}\,(0.1 - d_2/d_1), & |d_1| \geq |d_2| \end{cases} \qquad (3.4\text{-b})$$

e

$$\left.\begin{aligned} d_1 &= a - x \\ d_2 &= b - x \end{aligned}\right\} \; w < x \quad \text{ou} \quad \left.\begin{aligned} d_1 &= b - x \\ d_2 &= a - x \end{aligned}\right\} \; w > x \qquad (3.4\text{-c})$$

Caso a previsão $\hat{u}$ seja rejeitada por qualquer das proteções, a previsão modificada $\bar{u}$ é feita igual a esse limite, isto é, $\bar{u} = m_\alpha$.

A seguir, são feitos alguns comentários sobre o processo de rejeição da previsão $\hat{u}$ caso a mesma se situe além do limite artificial m_α. Vale observar que:

- x é o ponto associado ao menor valor da função na iteração corrente;
- o limite m_α sempre se situa além do ponto x, considerando a direção $w \longrightarrow x$;
- é possível que os pontos x e m_α se situem do mesmo lado do mínimo e que, à medida que se avança na direção $x \longrightarrow m_\alpha$, além de m_α, a função continue decrescendo. Ver figura 3.2.
- em uma outra situação, o ponto de mínimo pode se situar entre x e m_α e, à medida que se avança na direção $x \longrightarrow m_\alpha$, além de m_α, a função aumenta de valor. Ver figura 3.3.

Como sugere a proteção (ii), quando a previsão $\hat{u}$ se situa entre o ponto x e o limite m_α, sua aceitação é imediata. Na situação ilustrada na figura 3.2, existe a possi-

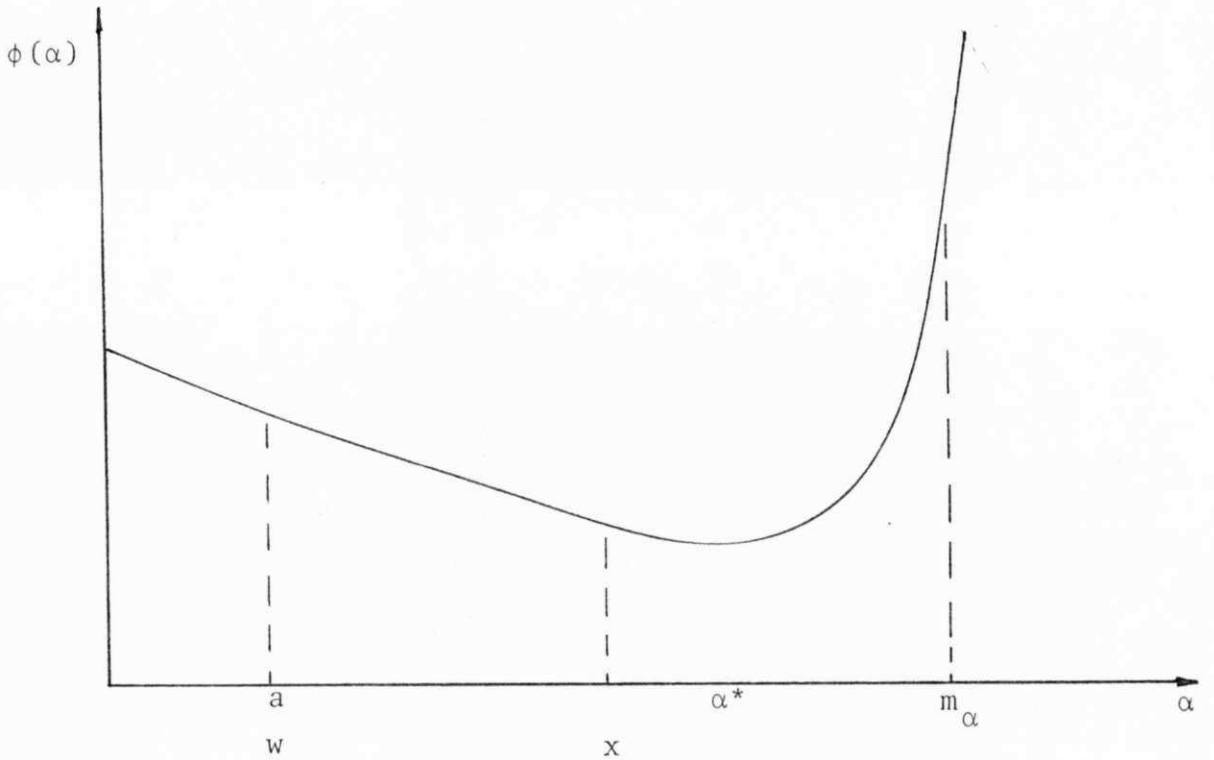


Figura 3.3: A rejeição de previsões û que se situam além do limite m_α é um aspecto importante no desempenho do algoritmo, especialmente quando se considera a possibilidade de a função crescer bruscamente além desse limite, como mostra a figura.

bilidade de o ponto $\hat{u}$ se situar em uma faixa de valores onde $\phi(\hat{u}) < \phi(m_\alpha)$ e a rejeição imediata da previsão $\hat{u}$ - como sugere a proteção (ii) - não tem sentido. Surge, pois, a necessidade de se introduzir um teste comparativo adicional. Isto é, caso a previsão $\hat{u}$ se situe além de m_α, verifica-se se $\phi(\hat{u}) > \phi(m_\alpha)$ e, em caso afirmativo, a previsão $\hat{u}$ é rejeitada e u é feito igual a m_α.

No caso com derivadas, as observações acima também são válidas. As fórmulas (3.4-a) e (3.4-b) permanecem inalteradas, mas os valores d_1 e d_2 passam a ser calculados da seguinte maneira:

$$d_1 = w - x$$

$$d_2 = \begin{cases} b - x, & w < x \\ a - x, & w > x \end{cases}$$

Vale observar que se os pontos estão em configuração de interpolação (configuração (ii) ou (iii) da relação (3.1) ou (3.2)), o limite artificial não é definido, uma vez que os pontos x e w coincidem com os extremos do intervalo de pesquisa. Nesse caso, a estratégia de comparação de função consiste em tomar o ponto médio do intervalo de pesquisa como previsão modificada, isto é,

$$\bar{u} = (a + b)/2.$$

3.2 Algoritmo de Murray & Overton para a minimização de uma classe de funções que não são continuamente diferenciáveis

A seguir, é descrito um algoritmo específico para a minimização de funções que não são diferenciáveis do tipo $F_S(\alpha)$. É apresentado em detalhes o processo de seleção da primeira previsão $\hat{u}$, uma vez que as proteções que, quando necessário, rejeitam tal previsão, o método de renomeação dos pontos, o critério de convergência e o método de comparação de função são os mesmos descritos para o algoritmo de Gill & Murray, apresentado anteriormente.

A estratégia básica desse algoritmo consiste em distinguir se o mínimo α^* é ou não um ponto de descontinuidade da derivada e, então, selecionar a previsão $\hat{u}$ de acordo com o que parecer mais provável; dessa forma, a previsão $\hat{u}$ será escolhida como a estimativa de uma descontinuidade ou como o ponto de mínimo de um polinômio que aproxima $F(\alpha)$ em torno de α^*.

3.2.1 Algoritmo de Murray & Overton - versão simplificada ("Low overhead version")

Na descrição que se segue subentende-se que em cada iteração tem-se um intervalo de pesquisa [a,b] que contém o mínimo α^* e que as relações (3.1) ou (3.2) são satisfeitas.

O processo de obtenção da previsão $\hat{u}$ pode ser dividido nas seguintes etapas:

- Estimativa de descontinuidades;
- Aproximações de $F(\alpha)$ por funções continuamente diferenciáveis.

Estimativa de descontinuidades

Nesta etapa, procura-se:

i) estimar a menor e maior descontinuidades contidas no intervalo [a,b] e atribuir seus valores às descontinuidades de teste z_L e z_R, respectivamente;

ii) estimar a média aritmética de todas as descontinuidades contidas no intervalo [a,b] e atribuir esse valor à variável $\bar{z}$;

iii) caso a configuração seja de extrapolação, verificar se existe alguma descontinuidade entre os pontos x e w (configurações (i) ou (iv) da relação (3.1)) ou entre os pontos w e v (configurações (i) ou (iv) da relação (3.2)).

A princípio, os valores $f_i(a)$ e $f_i(b)$ são comparados para as m funções componentes $f_i(\alpha)$. Se, para algum i, $f_i(a)$ e $f_i(b)$ tiverem sinais contrários, existe uma descontinuidade de $F(\alpha)$ entre a e b dada pelo zero de $f_i(\alpha)$ e esse zero passa a ser estimado. Após terem sido feitas as estimativas z_i dos zeros das funções $f_i(\alpha)$ que mudam de sinal no intervalo [a,b], deve-se ter armazenados nas descontinuidades de teste z_L e z_R os valores da menor e maior

descontinuidades contidas em [a,b]. Não é necessário armazenar os valores das estimativas z_i; à proporção que cada z_i é calculada, os valores z_L e z_R são renomeados e se procede ao cálculo da estimativa da média aritmética das descontinuidades.

Vale observar que o critério que compara os valores $f_i(a)$ e $f_i(b)$ para as m funções componentes, pode não identificar todas as descontinuidades contidas em [a,b], uma vez que uma determinada função $f_i(\alpha)$ pode apresentar um número par de zeros em [a,b] e, por isso, $f_i(a)$ e $f_i(b)$ apresentarem sinais iguais. Ver figura 3.4.

Caso não haja descontinuidades no intervalo de pesquisa (ou, mesmo que haja, o critério não identifique sua existência), z_L é feita igual a b, z_R é feita igual a a e $\bar{z}$ não é definida.

Caso a configuração seja de extrapolação, é necessário verificar se existe alguma descontinuidade entre os pontos x e w (caso com derivadas) ou entre os pontos w e v (caso sem derivadas). Se, pelo menos uma das funções componentes $f_i(\alpha)$ apresentar sinais contrários nos pontos em questão, já se tem a informação desejada, não sendo necessário fazer o teste para as funções componentes que restam. Ver figura 3.5. Vale observar que nenhuma tentativa é feita para se estimar qualquer dessas descontinuidades. Caso a configuração seja de interpolação (configuração (ii) ou (iii) da relação (3.1) ou (3.2)), essa informação é

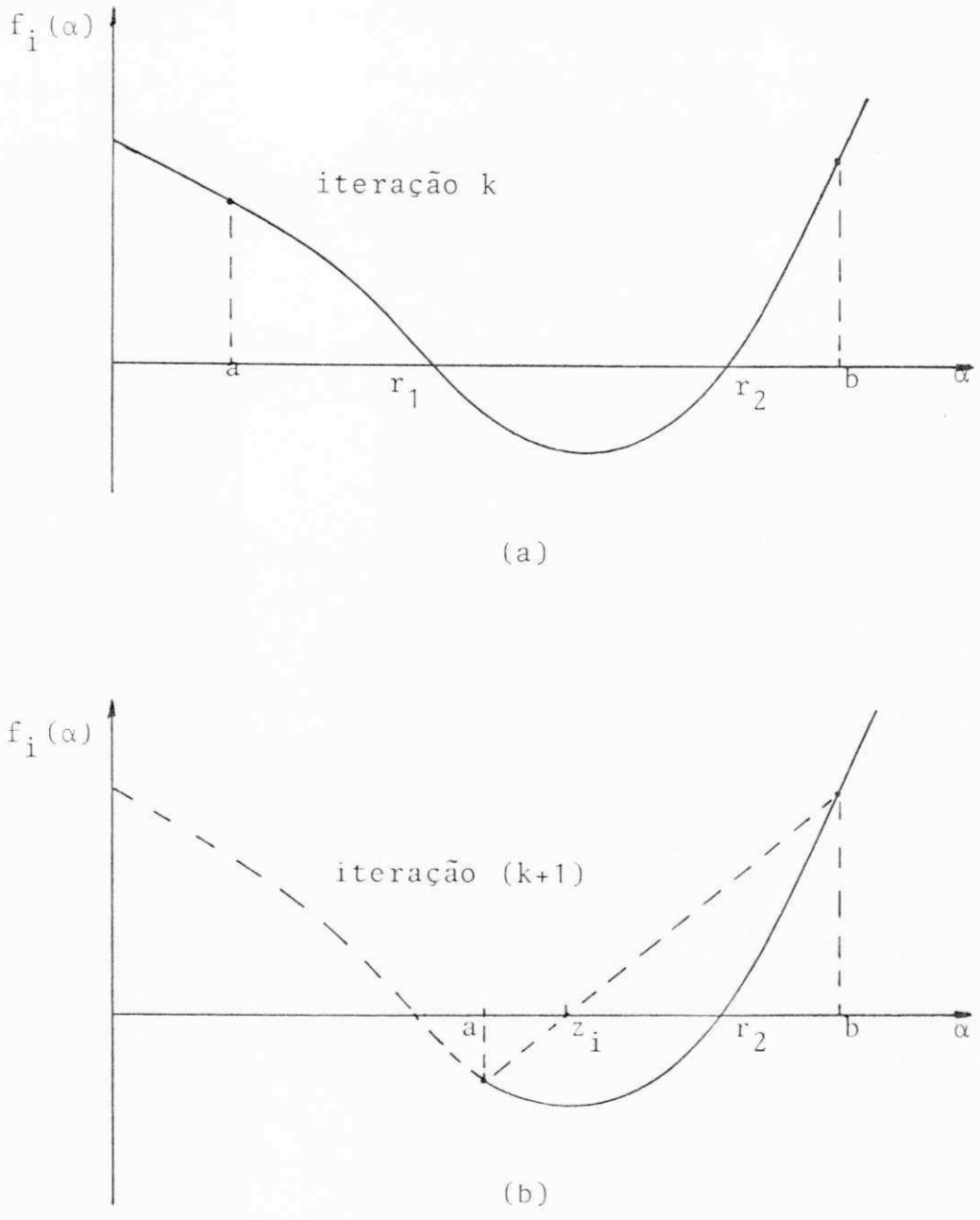


Figura 3.4: Ilustra o processo de estimativa de descontinuidades.

(a) Na iteração k, apesar de existirem dois zeros de $f_i(\alpha)$ em [a,b], o critério usado não identifica a existência de qualquer zero dessa função.

(b) Na iteração (k+1), o critério identifica a existência de um zero (r_2) que passa a ser estimado.

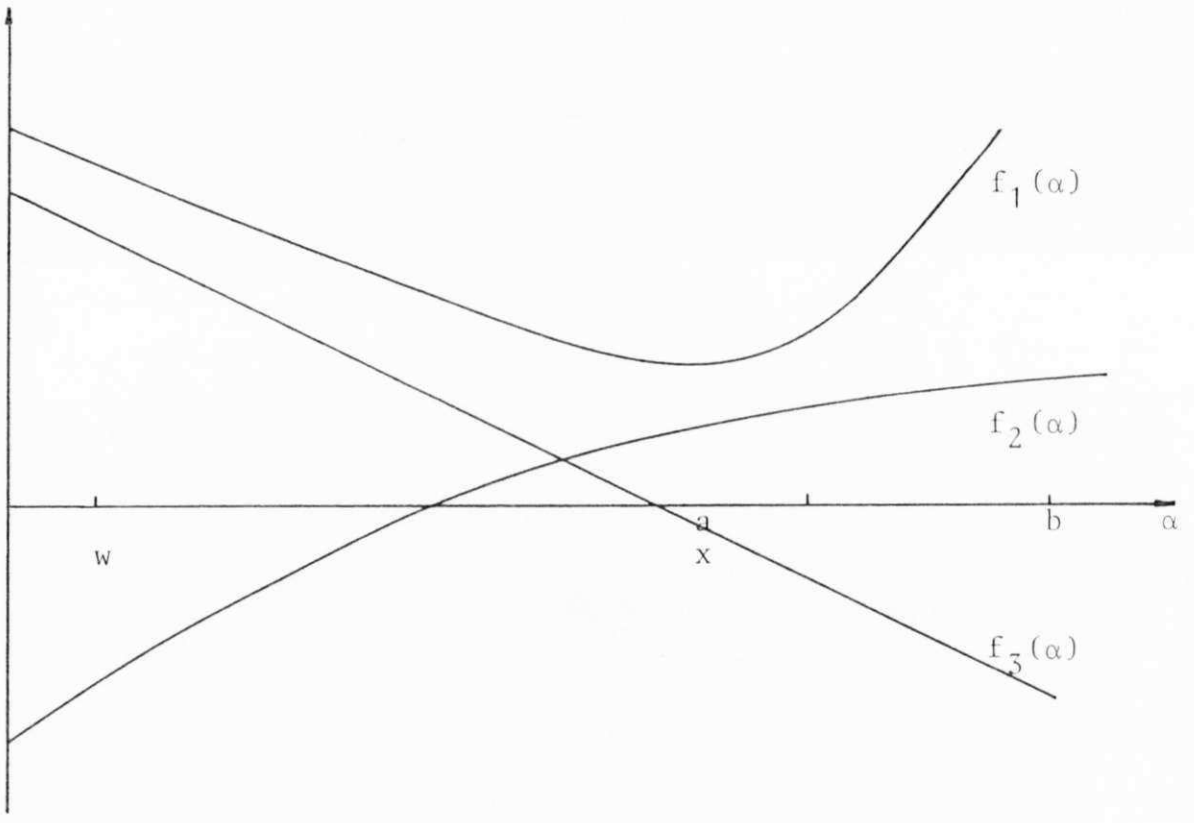


Figura 3.5: Os pontos a, b, x e w apresentam configuração de extrapolação. Os valores de $f_i(x)$ e $f_i(w)$ são comparados. Como $f_2(x)$ e $f_2(w)$ apresentam sinais contrários, já se tem a informação desejada: existe uma descontinuidade entre x e w e não é necessário comparar os valores $f_3(x)$ e $f_3(w)$.

obtida ao mesmo tempo em que são comparados os valores $f_i(a)$ e $f_i(b)$.

Quando derivadas são usadas, a estimativa das descontinuidades é feita pelo método de Newton, a partir do ponto x

$$z_i = x - f_i(x)/f_i'(x)$$

Se a estimativa obtida pelo método de Newton cair fora do intervalo [a,b], uma segunda estimativa é feita usando-se o método da Secante a partir dos pontos a e b:

$$z_i = \frac{a\ f_i(b) - b\ f_i(a)}{f_i(b) - f_i(a)}$$

Essa segunda estimativa, com certeza se situa no intervalo [a,b]. A utilização do método da Secante a partir de dois pontos onde a função apresenta sinais contrários em conjunto com o método de Newton combina as vantagens dos dois métodos: a convergência global do primeiro e a convergência superlinear do segundo. O método de Newton funciona como um procedimento de terminação, acelerando a convergência do processo nas últimas iterações, quando a estimativa inicial (ponto x) se situa próxima da solução.

No caso em que derivadas não são usadas, a estimativa das descontinuidades é feita pelo método da Secante, usando os pontos a e b quando existir alguma descontinui-

dade entre os pontos w e v e, em caso contrário, usando os pontos x e w. Ver figura 3.6. Se a estimativa obtida a partir dos pontos x e w cair fora do intervalo, deve-se usar os pontos a e b.

Aproximações de $F(\alpha)$ por funções continuamente diferenciáveis

Para um dado ponto y, define-se

$$F^{(y)}(\alpha) = \sum_{i:f_i(\alpha)>0} f_i(\alpha)$$

$F^{(y)}(\alpha)$ é uma função continuamente diferenciável que coincide com $F(\alpha)$ no intervalo que contém o ponto y e ao longo do qual $F(\alpha)$ é continuamente diferenciável. A derivada dessa função é definida por

$$F^{(y)'}(\alpha) = \sum_{i:f_i(\alpha)>0} f_i'(\alpha)$$

O processo de aproximações de $F(\alpha)$ por funções continuamente diferenciáveis consiste em escolher uma função continuamente diferenciável que coincida com $F(\alpha)$ em um certo intervalo (função de aproximação) e, usando interpolação (ou extrapolação) polinomial, estimar o mínimo da função de aproximação. Essa estimativa, dependendo de certas condições, pode ser aceita como a previsão $\hat{u}$.

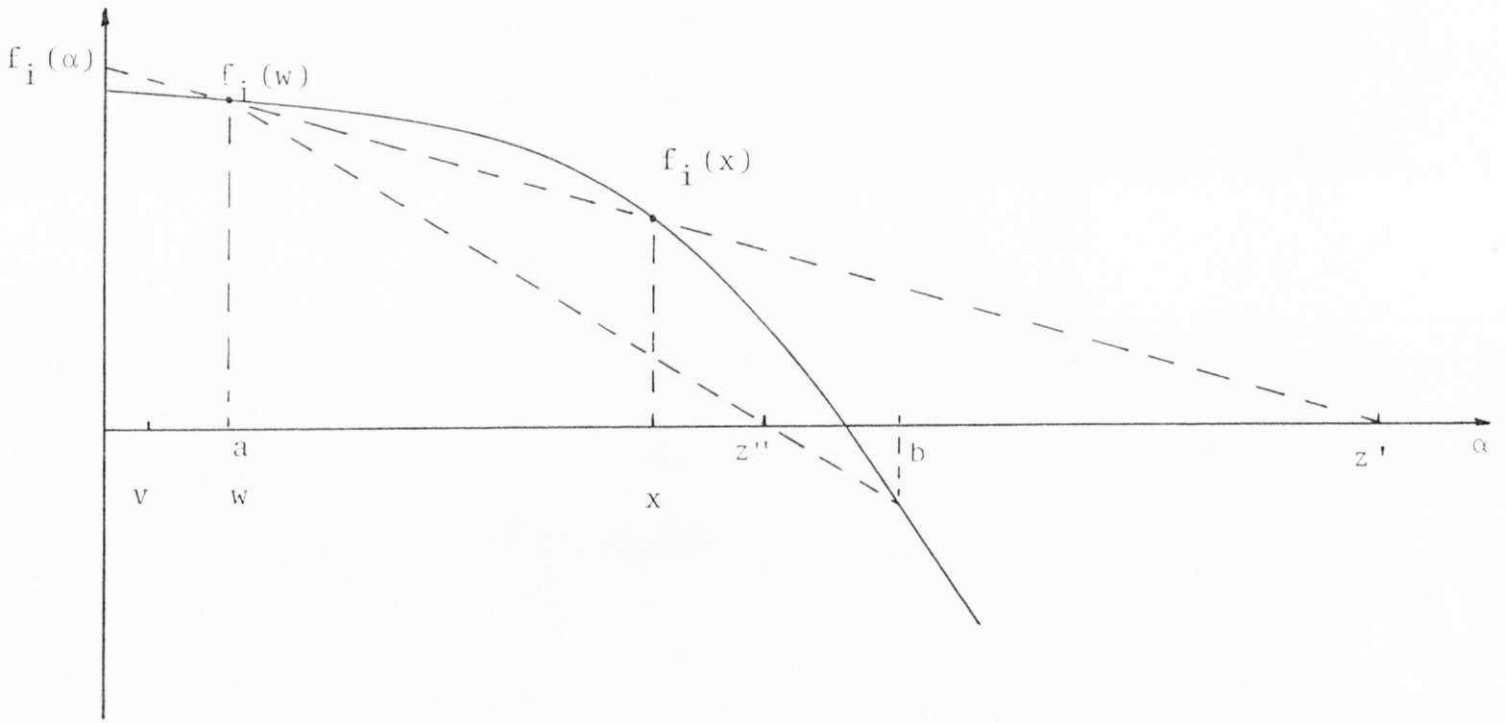


Figura 3.6: Ilustra o processo de estimativas de descontinuidades (caso sem derivadas). O método da Secante, usando os pontos x e w gera uma estimativa z' que se situa fora do intervalo [a,b]. Usando-se os pontos a e b, a estimativa z'' se situa em [a,b].

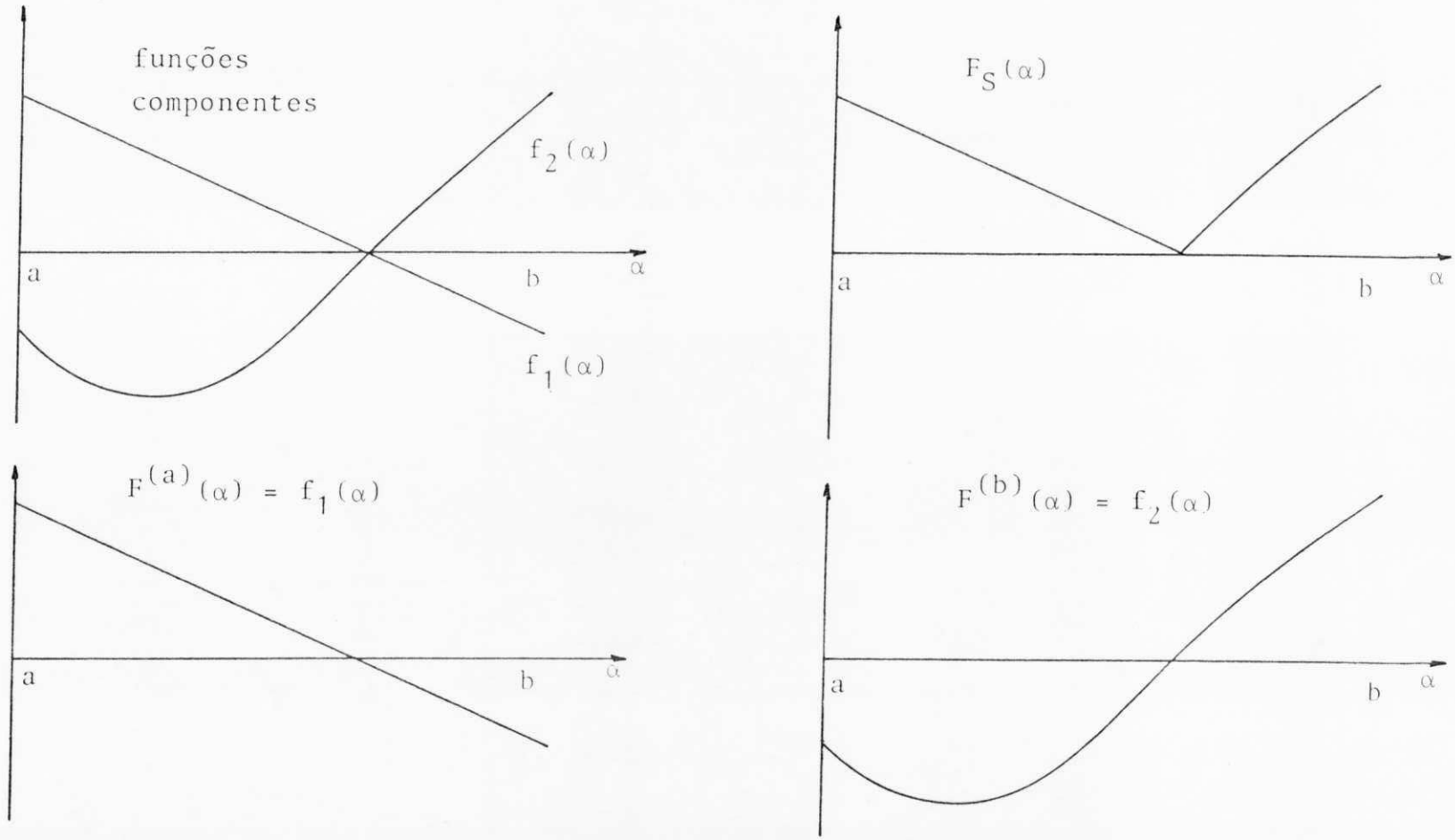


Figura 3.7: Ilustra o processo de aproximações pelas funções $F^{(a)}(\alpha)$ e $F^{(b)}(\alpha)$.

Nessa versão simplificada do algoritmo somente são consideradas duas funções de aproximação: $F^{(a)}(\alpha)$ e $F^{(b)}(\alpha)$. A função $F^{(a)}(\alpha)$ coincide com $F(\alpha)$ no intervalo $[a,z_{min}]$ onde z_{min} é a menor das descontinuidades localizadas em [a,b]. De forma análoga, $F^{(b)}(\alpha)$ coincide com $F(\alpha)$ no intervalo $[z_{max},b]$ onde z_{max} é a maior das descontinuidades localizadas em [a,b]. Ver figura 3.7

A informação obtida na primeira parte sobre a existência ou não de descontinuidades entre os pontos x e w (caso com derivadas) ou w e v (caso sem derivadas) é usada para se escolher a função de aproximação. A idéia básica é aproximar $F(\alpha)$ por funções continuamente diferenciáveis usando, quando possível, os pontos que retêm os menores valores da função, uma vez que o processo de aproximações polinomiais sucessivas somente apresenta convergência superlinear com a ordem estabelecida na seção 3.1 se os melhores pontos forem usados em cada aproximação.

No caso em que derivadas são usadas, se não for identificada qualquer descontinuidade entre x e w, é feita uma interpolação cúbica nos pontos x e w, usando como informações os valores $F(\alpha)$ e $F'(\alpha)$. Ver figura 3.8. Entretanto, se for identificada pelo menos uma descontinuidade entre x e w, é feita uma estimativa do mínimo da função $F^{(x)}(\alpha)$ (vale lembrar que x = a ou x = b) usando interpolação cúbica nos pontos a e b. Seja s_1 o mínimo do polinômio interpolador. Se s_1 se situar em [a,b] e não houver descontinuidades entre x e s_1, isto é, $s_1 \in [a,z_L]$ para x = a ou

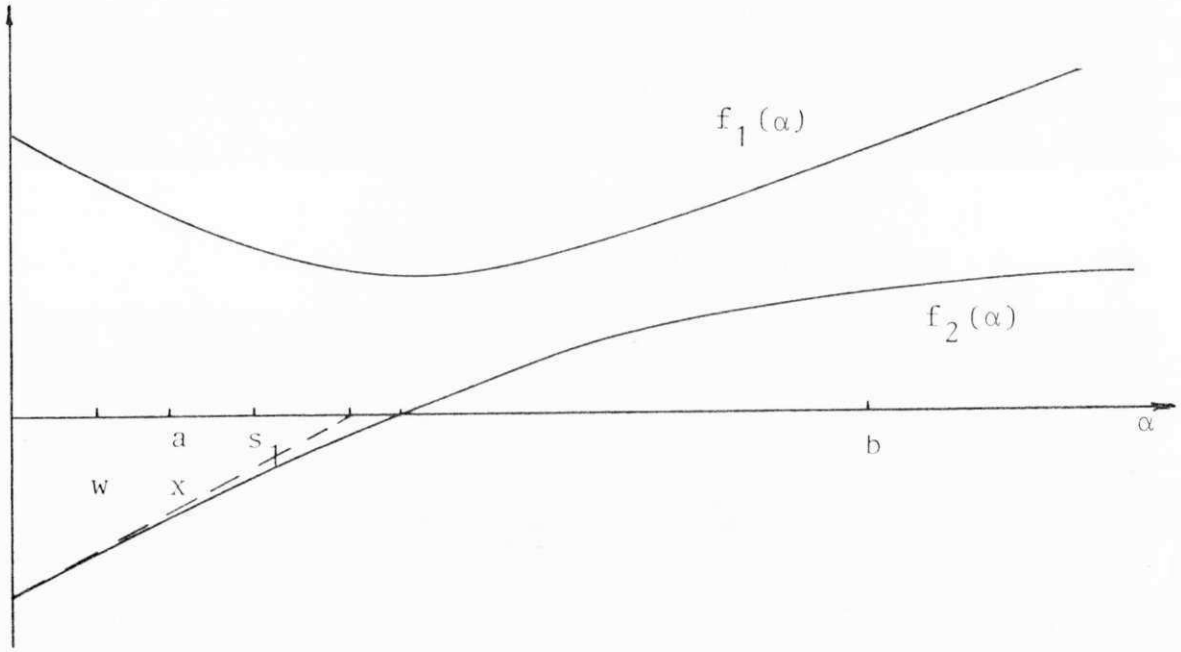


Figura 3.8: Ilustra o processo de aproximações por funções continuamente diferenciáveis. Como não é identificada a existência de descontinuidades entre x e w, é feita uma interpolação cúbica nos pontos x e w, usando como informações os valores de F(w), F'(w), F(x) e F'(x). Trata-se de um processo de extrapolação onde se tem $w < x$ e $F'(w) < 0$ e $F'(x) < 0$. O processo gera o ponto s_1 que pode ser aceito como $\hat{u}$, caso se situe entre x e z_L.

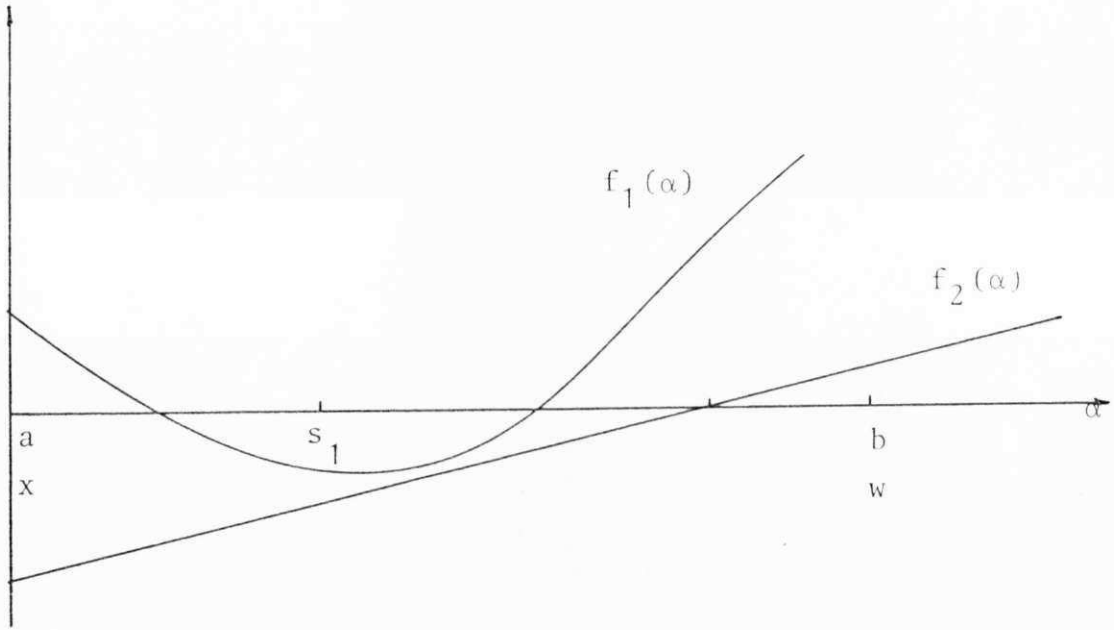


Figura 3.9: Como $f_2(x)$ e $f_2(w)$ apresentam sinais contrários, é feita uma estimativa do mínimo da função $F^{(a)}(\alpha)$, usando interpolação cúbica nos pontos a e b. Supondo-se que o ponto s_1 se situa como indicado, é importante notar que $\hat{u}$ será feito igual a s_1, apesar de existir um ponto de descontinuidade de $F(\alpha)$ entre x e s_1, isso porque o algoritmo não identifica a existência dessa descontinuidade.

$s_1 \in [z_R, b]$ para $x = b$, o ponto s_1 é aceito como a previsão $\hat{u}$. Em caso contrário, uma segunda aproximação deve ser feita. Ver figura 3.9.

Na segunda aproximação é estimado o mínimo da função $F^{(b)}(\alpha)$ caso $x = a$ ou $F^{(a)}(\alpha)$ caso $x = b$, usando interpolação cúbica nos pontos a e b. Seja s_2 o mínimo do polinômio interpolador. Caso s_2 se situe em [a,b] e $s_2 \in [a, z_L]$ para $x = b$ ou $s_2 \in [z_R, b]$ para $x = a$, o ponto s_2 é aceito como a previsão $\hat{u}$.

Caso os pontos s_1 e s_2 sejam rejeitados, isto significa que a estimativa de uma determinada descontinuidade se situa entre o ponto a e a estimativa do ponto de mínimo da função continuamente diferenciável que coincide com $F(\alpha)$ no ponto a. O mesmo é verdadeiro para a função continuamente diferenciável que coincide com $F(\alpha)$ no ponto b, isto é, a estimativa de uma determinada descontinuidade se situa entre os pontos s_2 e b. Conclui-se, portanto, que α^* pode ser um ponto de descontinuidade e, por isso, a previsão $\hat{u}$ é feita igual a $\bar{z}$. Ver figura 3.10.

No caso em que derivadas não são usadas, o procedimento é análogo. Não sendo identificada qualquer descontinuidade entre os pontos w e v, é feita uma interpolação quadrática nos pontos x, w e v usando como informações os valores de $F(\alpha)$. Em caso contrário, é feita uma estimativa do mínimo da função $F^{(w)}(\alpha)$ (vale lembrar que $w = a$ ou $w = b$), usando interpolação quadrática nos pontos a, x e b.

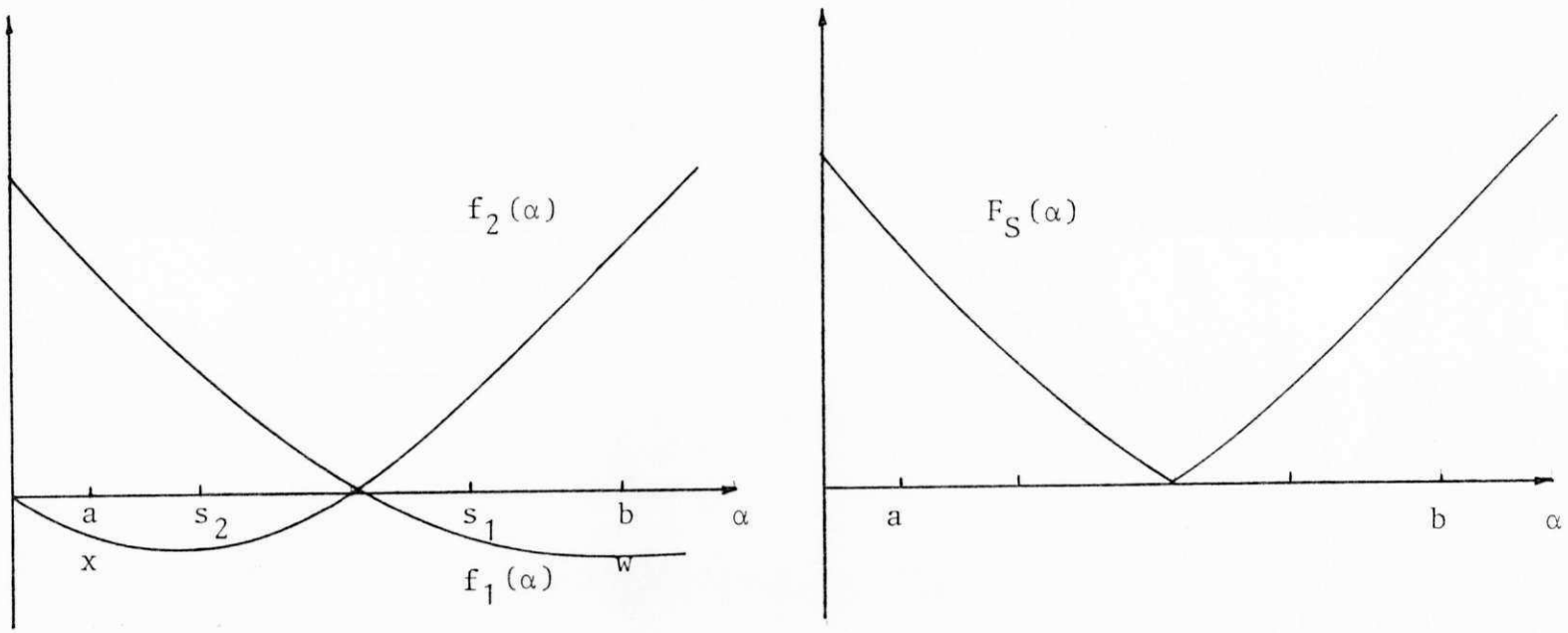


Figura 3.10: Como o algoritmo identifica a existência de descontinuidades entre x e w, é feita uma interpolação cúbica nos pontos a e b, usando como informações os valores de $F^{(a)}(\alpha)$ e $F^{(a)'}(\alpha)$. Dessa aproximação resulta o ponto s_1. Se $x = a$ e $s_1 > z_L$, uma segunda aproximação é feita, usando como informações os valores de $F^{(b)}(\alpha)$ e $F^{(b)'}(\alpha)$ nos pontos a e b. Dessa aproximação resulta o ponto s_2. Se s_2 não se situar no subintervalo $[z_R, b]$, $\hat{u}$ é feito igual a $\bar{z}$.

Caso o ponto de mínimo do polinômio interpolador (ponto s_1) se situe em [a,b] e $s_1 \in [a,z_L]$ para w = a ou $s_1 \in [z_R,b]$ para w = b, a previsão $\hat{u}$ é feita igual a s_1. Em caso contrário, uma segunda aproximação deve ser feita. Dessa vez é estimado o mínimo da função $F^{(b)}(\alpha)$ caso w = a ou $F^{(a)}(\alpha)$ caso w = b. Caso essa aproximação também não resulte em um ponto s_2 aceitável, conclui-se que α^* pode ser um ponto de descontinuidade e, por isso, a previsão $\hat{u}$ é feita igual a $\bar{z}$.

Sequência de passos computacionais para obtenção da previsão $\hat{u}$ (caso com derivadas)

1. Especifique o intervalo de pesquisa [a,b] e dois pontos x e w que satisfaçam à relação (3.1).

2. Se w = a ou w = b, vá para 4.

3. Verifique se existe alguma descontinuidade entre os pontos x e w, comparando os valores $f_i(x)$ e $f_i(w)$ para as m funções componentes.

4. Compare os valores $f_i(a)$ e $f_i(b)$ para as m funções componentes. Se não for identificada a existência de descontinuidades em [a,b], faça z_L = b, z_R = a e vá para 6.

5. Faça a estimativa das descontinuidades e atribua valores às descontinuidades de teste

   z_L = menor das estimativas

   z_R = maior das estimativas

$\bar{z}$ = média aritmética das estimativas

6. Se foi identificada alguma descontinuidade entre os pontos x e w, faça uma interpolação cúbica nos pontos a e b, usando como informações valores de $F^{(x)}(\alpha)$ e $F^{(x)'}(\alpha)$ e vá para 8.

7. Faça uma interpolação cúbica nos pontos x e w, usando como informações valores de $F(\alpha)$ e $F'(\alpha)$.

8. Atribua à variável s_1 o resultado da primeira aproximação.

9. Se $s_1 \in [a, z_L]$ para x = a ou $s_1 \in [z_R, b]$ para x = b, faça $\hat{u} = s_1$.

10. Faça um interpolação cúbica nos pontos a e b, usando como informações:
    - valores de $F^{(b)}(\alpha)$ e $F^{(b)'}(\alpha)$ se x = a;
    - valores de $F^{(a)}(\alpha)$ e $F^{(a)'}(\alpha)$ se x = b;

11. Atribua à variável s_2 o resultado da segunda aproximação.

12. Se $s_2 \in [z_R, b]$ para x = a ou $s_2 \in [a, z_L]$ para x = b, faça $\hat{u} = s_2$.

13. Faça $\hat{u} = \bar{z}$.

3.2.2 Algoritmo de Murray & Overton - versão com refinamentos ("Higher overhead version")

Essa versão do algoritmo, em comparação com a versão simplificada, requer um número maior de operações e memória, mas utiliza de forma mais completa todas as informações disponíveis. A diferença básica entre as duas versões é que na versão com refinamentos o número de aproximações de $F(\alpha)$ por funções continuamente diferenciáveis não é limitado a uma ou duas, sendo possíveis mais de m aproximações. São feitas aproximações por funções continuamente diferenciáveis entre cada duas estimativas de descontinuidades adjacentes. Outra diferença entre as duas versões é que na versão com refinamentos as descontinuidades são estimadas pelo processo de interpolação inversa.

Como na descrição da versão simplificada, supõe-se que em cada iteração é conhecido um intervalo [a,b] que contém o mínimo α^* e que as relações (3.1) ou (3.2) são satisfeitas.

Essa descrição se refere ao caso em que $F(a) < F(b)$, mas contém comentários que ilustram o processo no outro caso, isto é, $F(b) < F(a)$.

A obtenção da previsão $\hat{u}$ consiste das seguintes etapas:

- Estimativa de descontinuidades;
- Aproximações de $F(\alpha)$ por funções continuamente di-

ferenciáveis.

Estimativa de descontinuidades

Nesta parte, procura-se:

i) verificar se existe alguma descontinuidade entre os pontos x e w (caso com derivadas) ou entre os pontos w e v (caso sem derivadas), quando a configuração for de extrapolação;

ii) estimar todas as descontinuidades localizadas no intervalo [a,b];

iii) ordenar e armazenar as estimativas das descontinuidades.

Todas as descontinuidades localizadas em [a,b] são estimadas pelo processo de interpolação inversa. Como o processo de interpolações inversas sucessivas somente apresenta a velocidade de convergência superlinear estabelecida nos processos teóricos caso os melhores pontos sejam usados, a interpolação inversa deve ser feita nos pontos x e w (caso com derivadas) ou nos pontos x, w e v (caso sem derivadas), a menos que seja identificada a existência de alguma descontinuidade entre os pontos em questão, quando então devem ser usados os pontos a e b (caso com derivadas) ou pontos a, x e b (caso sem derivadas).

E possivel que a estimativa usando os pontos x e w ou x, w e v não se situe no intervalo [a,b], quando então

deve ser feita uma outra estimativa, dessa vez usando os pontos a e b ou a, x e b. Caso essa segunda estimativa também não se situe em [a,b], a estimativa passa a ser feita pelo método da Secante, usando os pontos a e b.

Quando derivadas são usadas, as estimativas são feitas por interpolação cúbica inversa, usando como informações valores da função e da primeira derivada associados a dois pontos. No caso sem derivadas, as estimativas são feitas por interpolação quadrática inversa, usando como informações valores da função associados a três pontos.

As estimativas z_i são ordenadas e armazenadas em um vetor. Como cada z_i estima o zero de uma dada função componente, é necessário armazenar, em um outro vetor, o número da função componente cujo zero é estimado pela estimativa z_i que ocupa a posição correspondente no vetor de estimativas. Ver figura 3.11.

Aproximações de $F(\alpha)$ por funções continuamente diferenciáveis

O processo consiste em fazer aproximações usando as funções continuamente diferenciáveis que coincidem com $F(\alpha)$ nos subintervalos delimitados por duas estimativas de descontinuidades adjacentes. Como se supõe que $F(a) < F(b)$, os subintervalos são analisados da esquerda para a direita. (Caso $F(b) < F(a)$ os subintervalos deveriam ser analisados da direita para a esquerda).

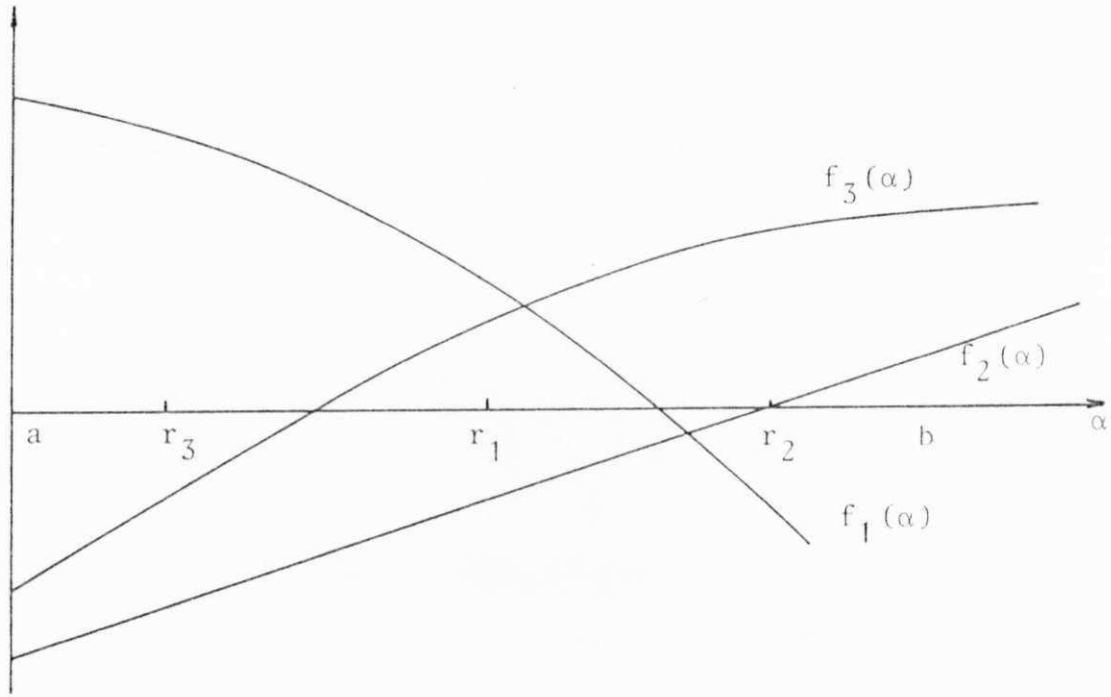


Figura 3.11: Ilustra o processo de ordenação dos vetores de estimativas (Z) e de indicação (IZ).
As estimativas dos zeros das funções $f_1(\alpha)$, $f_2(\alpha)$ e $f_3(\alpha)$ são r_1, r_2 e r_3. Após ordenados, tem-se os seguintes vetores:

$$Z = [r_3 \quad r_1 \quad r_2]^T$$

$$IZ = [3 \quad 1 \quad 2]^T$$

Sejam y_1 e y_2 os limites do subintervalo em análise e seja $h(\alpha)$ a função continuamente diferenciável que coincide com $F(\alpha)$ entre as descontinuidades estimadas por y_1 e y_2. A princípio y_1 é feito igual a a, y_2 igual ao menor dos valores $\{z_i\}$ e $h(\alpha)$ coincide com $F^{(a)}(\alpha)$. Seja s a estimativa do mínimo de $h(\alpha)$, obtida por interpolação polinomial (direta). Como na versão simplificada, o processo de interpolações diretas deve ser feito nos pontos x e w ou x, w e v caso não seja identificada a existência de descontinuidades entre os pontos em questão. Em caso contrário, devem ser usados os pontos a e b ou a, x e b.

Se $s < y_1$, a previsão $\hat{u}$ é feita igual a y_1, já que as funções continuamente diferenciáveis que coincidem com $F(\alpha)$ à esquerda e à direita de y_1, aparentemente, têm seus pontos de mínimo à direita e à esquerda de y_1, respectivamente.

Se $s > y_2$, então y_1 é feito igual a y_2, y_2 é feito igual ao próximo menor dos valores $\{z_i\}$ (ou igual a b se não houver nenhuma estimativa z_i maior que y_2) e $h(\alpha)$ é a função continuamente diferenciável que coincide com $F(\alpha)$ no novo subintervalo $[y_1, y_2]$ e o processo é repetido. A nova função $h(\alpha)$ é obtida somando-se à função $h(\alpha)$ anterior a função $\pm f_k(\alpha)$, onde z_k é o valor anterior de y_2 e o sinal é o mesmo que o de $f_k(b)$. Entretanto, se o valor anterior de y_2 for b, o processo é terminado com a previsão $\hat{u}$ feita igual a b. Ver figura 3.12.

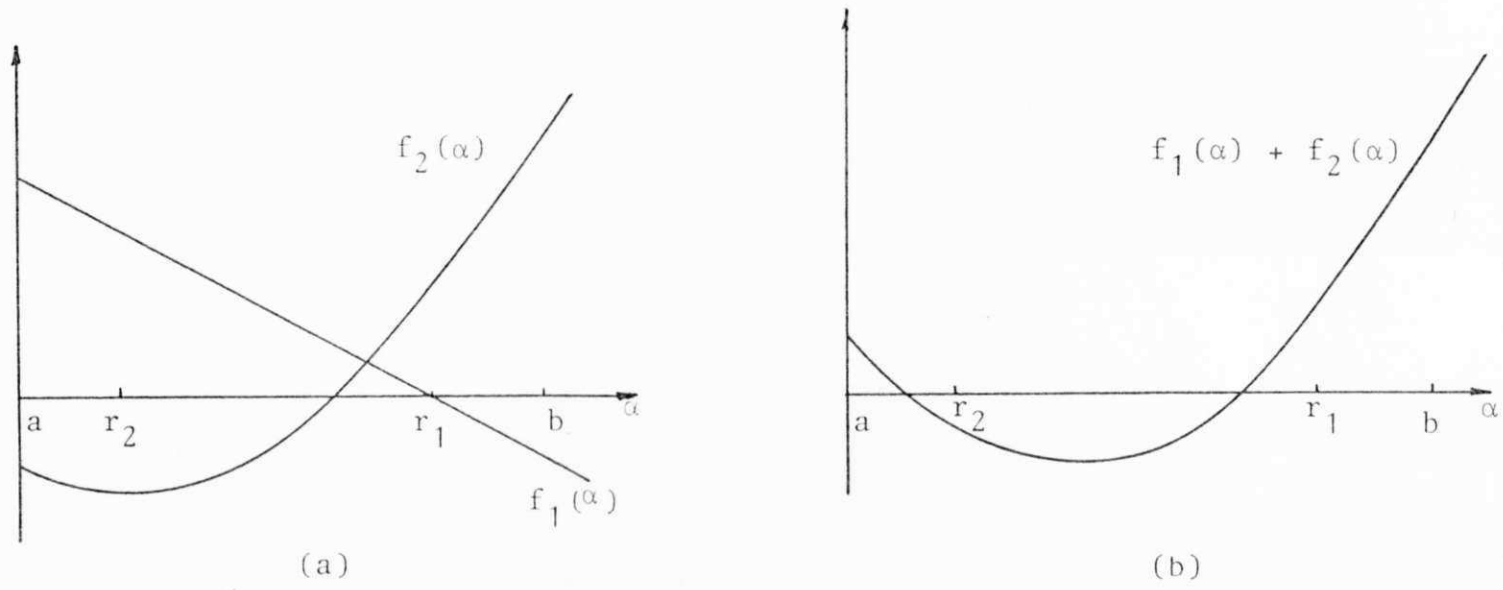


Figura 3.12: Aproximações de $F(\alpha)$ por funções continuamente diferenciáveis.

(a) As estimativas dos zeros correspondem aos pontos r_1 e r_2, indicados na figura. Na primeira aproximação, o subintervalo em análise é $[a,r_2]$ e $h(\alpha) = f_1(\alpha)$. Como $h(\alpha)$ é linear, s é indefinido e é feita uma segunda aproximação. O novo subintervalo é $[r_2,r_1]$ e, como r_2 é a estimativa do zero da função componente $f_2(\alpha)$, essa função deverá ser somada (já que $f_2(b) > 0$) à função $h(\alpha)$ anterior.

(b) Caso a estimativa do mínimo da função $f_1(\alpha) + f_2(\alpha)$ se situe em $[r_2,r_1]$, se tem $\hat{u} = s$. O algoritmo na versão simplificada geraria $\hat{u} = \bar{z}$.

Sequência de passos computacionais para obtenção da previsão $\hat{u}$

(caso com derivadas e $F(a) < F(b)$)

1. Especifique o intervalo de pesquisa [a,b] e dois pontos x e w que satisfaçam à relação (3.1)

2. Se w = a ou w = b, vá para 4.

3. Verifique se existe alguma descontinuidade entre os pontos x e w, comparando os valores $f_i(x)$ e $f_i(w)$ para as m funções componentes.

4. Compare os valores $f_i(a)$ e $f_i(b)$ para as m funções componentes. Se não foi identificada a existência de descontinuidades em [a,b], faça $y_1 = a$, $y_2 = b$, defina $h(\alpha) = F(\alpha)$ e vá para 13.

5. Faça a variável nz igual ao número de descontinuidades estimadas em [a,b].

6. Ordene o vetor de estimativas (Z) e o vetor de identificação (IZ).

7. Inicialize a variável j para contagem do número de aproximações polinomiais: j = 0

8. $y_1 = a$

9. $j = j + 1$

10. $y_2 = Z(j)$

11. Se $j = 1$, defina $h(\alpha) = F^{(a)}(\alpha)$

12. Faça $l = IZ(j-1)$ e defina $h(\alpha) = h(\alpha) + f_l(\alpha)$

13. Estime o mínimo de $h(\alpha)$ usando interpolação cúbica (direta). Atribua à variável s o valor dessa estimativa.

14. Se $y_1 \leq s \leq y_2$, faça $\hat{u} = s$

15. Se $s < y_1$, faça $\hat{u} = y_1$

16. Se $y_2 = b$, faça $\hat{u} = b$

17. $y_1 = y_2$

18. Se $j < nz$, vá para 9.

19. $j = j + 1$

20. $y_2 = b$

21. Vá para 12.

3.3 Algoritmos que fazem a busca do mínimo ao longo de um percurso curvilinear, usando direções de curvatura negativa

A idéia de usar direções de curvatura negativa nos algoritmos de minimização unidimensional foi introduzida por Fiacco & McCormick (1968) e, subsequentemente, investigada por vários pesquisadores (McCormick (1977); Moré & Sorensen (1979); Goldfarb (1980)).

Algoritmos que usam direções de curvatura negativa podem ser aplicados no processo de minimização de uma função $f(\tilde{x})$, multivariável, duas vezes diferenciável, para qualquer ponto particular de pesquisa $\tilde{x}_k$, gerado pelo processo de minimização, onde a matriz Hessiana $\nabla^2 f(\tilde{x}_k)$ tenha pelo menos um autovalor negativo (isto é, a matriz deve ser indefinida, negativa semi-definida ou negativa definida). Em especial, esses algoritmos garantem um progresso no processo de minimização quando o ponto particular de pesquisa $\tilde{x}_k$ é um ponto de inflexão (isto é, $\nabla f(\tilde{x}_k) = 0$ e $\nabla^2 f(\tilde{x}_k)$ tem pelo menos um autovalor negativo).

Esses algoritmos fazem uma minimização unidimensional (em busca de um ponto onde a função retém um menor valor) ao longo de um percurso curvilinear. Esse percurso é determinado pelos pontos $\tilde{x}(\alpha)$, definidos em termos de:

- um ponto de referência $\tilde{x}_k$;
- dois vetores: um primeiro, $\tilde{s}_k$, que representa uma direção descendente e um segundo, $\tilde{d}_k$, que representa uma direção não ascendente de curvatura negativa;
- duas funções escalares $\theta_1(\alpha)$ e $\theta_2(\alpha)$, para $\alpha > 0$.

Os pontos $\tilde{x}(\alpha)$ são definidos por

$$\tilde{x}(\alpha) = \tilde{x}_k + \theta_1(\alpha)\, \tilde{s}_k + \theta_2(\alpha)\, \tilde{d}_k$$

Esses algoritmos fazem uma minimização unidimensio-

nal de forma diferente dos algoritmos clássicos, que buscam um ponto melhor ao longo de um percurso retilíneo, determinado pelos pontos $\tilde{x}(\alpha)$, definidos por

$$\tilde{x}(\alpha) = \tilde{x}_k + \alpha \tilde{s}_k$$

onde $\tilde{x}_k$ é um ponto de referência, $\tilde{s}_k$ é um vetor que representa uma direção descendente de pesquisa e α é um escalar positivo.

Uma direção descendente $\tilde{s}_k$ satisfaz à condição $\nabla f(\tilde{x}_k)^T \tilde{s}_k < 0$ caso $\nabla f(\tilde{x}_k) \neq 0$ ou à condição $\nabla f(\tilde{x}_k)^T \tilde{s}_k = 0$ caso $\nabla f(\tilde{x}_k) = 0$. O vetor $\tilde{s}_k$ pode ser obtido, por exemplo, como a solução de

$$B_k \tilde{s}_k = - \nabla f(\tilde{x}_k)$$

onde B_k é uma matriz simétrica e positiva definida, escolhida arbitrariamente.

Uma direção não ascendente de curvatura negativa $\tilde{d}_k$ satisfaz às condições $\nabla f(\tilde{x}_k)^T \tilde{d}_k \leq 0$ e $\tilde{d}_k^T \nabla^2 f(\tilde{x}_k) \tilde{d}_k < 0$. O vetor $\tilde{d}_k$ pode ser escolhido, por exemplo, como

$$\tilde{d}_k = - \text{sinal}(\nabla f(\tilde{x}_k)^T \tilde{u}) \tilde{u}$$

onde $\tilde{u}$ é um autovetor correspondente ao autovalor mais negativo da matriz Hessiana $\nabla^2 f(\tilde{x}_k)$.

A função objetivo, unidimensional, cujo mínimo é

$\theta_2(0) = 0$, $\theta_2'(0) > 0$, $\theta_2''(0) = 0$

Essa escolha satisfaz às condições previamente especificadas caso $\nabla f(\tilde{x}_k)^T \tilde{d}_k < 0$, quando $\nabla f(\tilde{x}_k) \neq 0$.

Goldfarb (1980) escolheu $\theta_1(\alpha) = \alpha$ e $\theta_2(\alpha) = \alpha^2$. Essa escolha não satisfaz às condições previamente especificadas caso $\nabla f(\tilde{x}_k) = 0$, mas é vantajosa quando $\nabla f(\tilde{x}_k) \neq 0$ (no processo de busca do minimo, quando se tem distâncias infinitesimais entre o ponto de partida e o novo ponto gerado, a melhor direção de busca é uma direção descendente $\tilde{s}_k$; quando essas distâncias são grandes, a melhor direção é uma direção não ascendente de curvatura negativa $\tilde{d}_k$).

Dois algoritmos que usam direções de curvatura negativa são apresentados a seguir. Ambos fazem uma busca unidimensional aproximada do minimo ao longo do percurso curvilinear e garantem um decréscimo suficiente no valor da função objetivo.

Algoritmo de segunda ordem, de Armijo
(Goldfarb (1980))

Para

- uma dada função $f(\tilde{x})$, multivariável e duas vezes diferenciável;

- um dado ponto $\tilde{x}_k$
(tal que a matriz $\nabla^2 f(\tilde{x}_k)$ tenha, pelo menos, um

autovalor negativo);

- uma dada direção descendente $\tilde{s}_k$
  (caso $\nabla f(\tilde{x}_k) \neq 0$, então $\tilde{s}_k \neq 0$ e $\nabla f(\tilde{x}_k)^T \tilde{s}_k < 0$;
  caso $\nabla f(\tilde{x}_k) = 0$, então $\tilde{s}_k = 0$);

- uma dada direção não ascendente de curvatura negativa $\tilde{d}_k$
  (tal que $\nabla f(\tilde{x}_k)^T \tilde{d}_k < 0$ e $\tilde{d}_k^T \nabla^2 f(\tilde{x}_k) \tilde{d}_k < 0$);

- os parâmetros (dados) α e σ tais que $0 < \alpha < 1$, $0 < \sigma < 1$.

1. Encontrar o menor inteiro $i = 0, 1, 2, \ldots$ tal que:

$$f(\tilde{x}_k + \alpha^i \tilde{s}_k + \alpha^{2i} \tilde{d}_k) - f(\tilde{x}_k) \leq \sigma [\alpha^i \nabla f(\tilde{x}_k)^T \tilde{s}_k + 0.5 \alpha^{4i} \tilde{d}_k^T \nabla^2 f(\tilde{x}_k) \tilde{d}_k]$$

2. Fazer

$$\tilde{x}_{(k+1)} = \tilde{x}_k + \alpha^i \tilde{s}_k + \alpha^{2i} \tilde{d}_k$$

Algoritmo de segunda ordem, de Goldstein
(Goldfarb (1980))

Para

- uma dada função $f(\tilde{x})$, multivariável e duas vezes diferenciável;

- um dado ponto $\tilde{x}_k$
  (tal que a matriz $\nabla^2 f(\tilde{x}_k)$ tenha, pelo menos, um

autovalor negativo);

- uma dada direção descendente $\tilde{s}_k$
  (caso $\nabla f(\tilde{x}_k) \neq 0$, então $\tilde{s}_k \neq 0$ e $\nabla f(\tilde{x}_k)^T \tilde{s}_k < 0$;
  caso $\nabla f(\tilde{x}_k) = 0$, então $\tilde{s}_k = 0$);

- uma dada direção não ascendente de curvatura negativa $\tilde{d}_k$
  (tal que $\nabla f(\tilde{x}_k)^T \tilde{d}_k \leq 0$ e $\tilde{d}_k^T \nabla^2 f(\tilde{x}_k) \tilde{d}_k < 0$);

- os parâmetros (dados) σ_1 e σ_2, tais que $0 < \sigma_1 \leq \sigma_2 < 1$.

1. Definir

$$\Psi_k(\alpha) = \frac{f(\tilde{x}_k + \alpha \tilde{s}_k + \alpha^2 \tilde{d}_k) - f(\tilde{x}_k)}{\nabla f(\tilde{x}_k)^T (\alpha \tilde{s}_k + \alpha^2 \tilde{d}_k) + \delta}$$

onde

$$\delta = \min[0.5(\alpha \tilde{s}_k + \alpha^2 \tilde{d}_k)^T \nabla^2 f(\tilde{x}_k)(\alpha \tilde{s}_k + \alpha^2 \tilde{d}_k) \ , \ 0]$$

2. Se $\Psi_k(1) \geq \sigma_1$, faça $\alpha_k = 1$
   Se $\Psi_k(1) < \sigma_1$, escolher $\alpha_k > 0$, tal que $\sigma_1 \leq \Psi_k(\alpha_k) \leq \sigma_2$

3. Fazer $\tilde{x}_{(k+1)} = \tilde{x}_k + \alpha_k \tilde{s}_k + \alpha_k^2 \tilde{d}_k$.

4 RESULTADOS NUMERICOS

4.1 Introdução

Neste capitulo é feita uma descrição da forma como foram executados os testes de desempenho dos algoritmos hibridos com proteções. Esses testes foram divididos em duas categorias: testes de desempenho usando funções de teste continuamente diferenciáveis e testes de desempenho usando funções de teste que não são continuamente diferenciáveis. Os resultados são expostos em um relatório de testes que também inclui a análise e discussão desses resultados.

A fim de se testar o desempenho dos algoritmos surgiu a necessidade de se definir as funções de teste. A escolha das funções de teste é de importância crucial na validade dos resultados de desempenho dos algoritmos de otimização. Segundo Moré et alii (1981), a maioria dos testes de desempenho de algoritmos de otimização são inadequados porque o número de funções de teste é pequeno ou

porque os pontos iniciais de pesquisa ("standard starting points") já são próximos da solução.

No caso de funções continuamente diferenciáveis foi feita uma pesquisa no sentido de se escolher um conjunto de funções de teste de uma única variável para testar os algoritmos. Essas funções deveriam levar os algoritmos a se depararem com vários aspectos computacionais complicados, a fim de proporcionar a observação de seus desempenhos frente a essas situações peculiares. Várias listas de funções de teste foram consultadas e não foi encontrada em nenhuma delas funções de teste de uma única variável. Dentre as listas consultadas estão aquelas propostas por Himmelblau (1972), Moré et alii (1981) e Wolfe (1978). Surgiu, pois, a necessidade de se usar os algoritmos híbridos com proteções em conjunto com um algoritmo para minimização de funções multidimensionais. Dessa forma, os algoritmos híbridos com proteções cujos desempenhos deveriam ser analisados foram incorporados ao algoritmo de otimização multidimensional para fazer a busca linear ("line search"), isto é, determinar o mínimo da função unidimensional $\phi(\alpha) = F(\tilde{x} + \alpha\tilde{p})$. Esse procedimento possibilita a comparação de dois algoritmos para minimização unidimensional. Se a busca linear é exata, o mesmo ponto de mínimo da função unidimensional $\phi(\alpha)$ deve ser determinado por ambos os algoritmos de busca linear e, consequentemente, o número de iterações do processo multidimensional deverá ser o mesmo em ambos os casos. Assim, pode-se comparar a eficiência dos algoritmos

usados na busca linear tendo-se por base o número de avaliações da função objetivo, número de avaliações do gradiente e esforço computacional necessários para se chegar à solução.

No caso de funções que não são continuamente diferenciáveis não foi encontrada nenhuma lista de funções de teste que servissem de padrão de avaliação para o desempenho dos algoritmos. Em vista disso, cinco funções de teste do tipo $F_S(\alpha)$ foram elaboradas pelo autor. Essas funções foram elaboradas de maneira que o algoritmo especifico para minimização de funções $F_S(\alpha)$ pudesse se deparar com situações complicadas como, por exemplo, a não identificação dos pontos de descontinuidade da derivada e a rejeição da previsão do minimo obtida pela aproximação de funções continuamente diferenciáveis. Os testes foram realizados a fim de se comparar a eficiência dos algoritmos especificos para minimização de funções do tipo $F_S(\alpha)$ em relação aos algoritmos para minimização de funções continuamente diferenciáveis, usando-se o mesmo conjunto de funções de teste.

4.2 Testes de desempenho usando funções de teste continuamente diferenciáveis

Os testes comparativos dos algoritmos hibridos com proteções usando funções de teste continuamente diferenciáveis foram separados em duas categorias, dependendo do uso ou não de derivadas como informações. Primeiramente, foram comparados os desempenhos dos algoritmos que não

utilizam derivadas como informações, a saber, algoritmo de Brent e algoritmo de Gill & Murray (versão sem derivadas). No outro caso, os testes comparativos se referem ao algoritmo da Bissecção e algoritmo de Gill & Murray (versão com derivadas).

Tomando-se por base as listas de funções de teste consultadas, foram escolhidas cinco funções, que são apresentadas no Apêndice A. Trata-se de funções frequentemente referenciadas na literatura e que apresentam dificuldades computacionais para os problemas de otimização.

Os algoritmos hibridos com proteções foram usados em conjunto com o algoritmo da descida mais ingreme ("steepest descent algorithm"). A escolha desse algoritmo se deveu à sua simplicidade e ao fato de se ter assegurada uma direção descendente de pesquisa $\tilde{p}^{(k)}$ em todas as iterações. Foi utilizada a implementação do algoritmo da descida mais ingreme feita por Pequeno (1983), bastando para isso substituir os módulos da busca linear e da verificação de convergência. O critério de convergência de Himmelblau (1972), usado na implementação original, foi substituido por um critério menos rigoroso que somente requer que $||g^{(k)}||_M < 10^{-2}$.

No critério de convergência dos algoritmos hibridos com proteções, utilizados na busca linear, os parâmetros ε e τ definidos na seção 3.1 foram feitos iguais a 10^{-6} e $10^{-6}/||\tilde{p}^{(k)}||_2$, respectivamente.

4.3 Testes de desempenho usando funções de teste que não são continuamente diferenciáveis

Esses testes também foram separados em duas categorias, dependendo do uso ou não de derivadas como informações. Primeiramente, foram comparados os desempenhos dos algoritmos que não utilizam derivadas como informações, a saber, algoritmo de Brent e algoritmo de Murray & Overton (versão simplificada, caso sem derivadas). No caso com derivadas, os testes se referem ao desempenho dos algoritmos da Bissecção e algoritmo de Murray & Overton (versão simplificada, caso com derivadas). O algoritmo de Brent foi escolhido por ter apresentado um desempenho ligeiramente melhor que o algoritmo de Gill & Murray (versão sem derivadas), nos testes usando funções continuamente diferenciáveis. No caso do algoritmo da Bissecção, a escolha se deveu à sua analogia com o algoritmo de Brent, já que seu desempenho foi idêntico ao algoritmo de Gill & Murray (versão com derivadas).

As cinco funções de teste do tipo $F_S(\alpha)$ são apresentadas no apêndice B. Em três dessas funções, o minimizante α^* é um ponto de descontinuidade da derivada; nas outras duas funções, α^* é um ponto onde a derivada é definida. Os algoritmos foram usados na minimização exata dessas funções em um intervalo fechado. Os parâmetros ε e τ definidos na seção 3.1 foram feitos iguais a 10^{-6}.

4.4 Relatório de testes

Nesta seção, são apresentados e analisados os resultados obtidos nos testes de desempenho dos algoritmos híbridos com proteções, usando-se as funções de teste relacionadas nos apêndices A e B. Os testes foram executados em um computador IBM/4341 com compilador WATFIV, usando-se dupla precisão.

4.4.1 Minimização de funções continuamente diferenciáveis

Os resultados dos testes de desempenho dos algoritmos híbridos com proteções usando-se as funções continuamente diferenciáveis relacionadas no apêndice A são apresentados nas tabelas 4.1 e 4.2. As tabelas incluem, para cada função, as seguintes informações: valor final da função, número de iterações do método da descida mais íngreme, número total de avaliações da função (nf) e número de avaliações da função no processo de refinamento do "bracket" (na). Na tabela 4.2, ng denota o número total de avaliações do gradiente.

Os resultados contidos na tabela 4.1 são relativos ao algoritmo de Brent e algoritmo de Gill & Murray (versão sem derivadas).

Tabela 4.1

Testes de desempenho usando funções continuamente diferenciáveis (caso sem derivadas)

funções de teste	Valor final de $F(\tilde{x})$	Num. de iterações	BRENT	GILL & MURRAY
FCD1	2.6×10^{-3}	14		
nf			166	167
na			93	94
FCD2	3.8×10^{-4}	741		
nf			7141	7143
na			2463	2465
FCD3	2.4×10^{-5}	1229		
nf			10864	10865
na			3491	3492
FCD4	4.2×10^{-5}	174		
nf			2028	2047
na			799	818
FCD5	-5.8×10^{-1}	7		
nf			77	77
na			34	34

Os resultados contidos na tabela 4.2 são relativos ao algoritmo da Bissecção e algoritmo de Gill & Murray (versão com derivadas).

Tabela 4.2

Testes de desempenho usando funções continuamente diferenciáveis (caso com derivadas)

funções de teste	Valor final de $F(\tilde{x})$	Num. de iterações	BISSECÇÃO	GILL & MURRAY
FCD1	2.6×10^{-3}	14		
nf			122	122
na			64	64
ng			93	93
FCD2	4.1×10^{-4}	719		
nf			4466	4466
na			1588	1588
ng			3027	3027
FCD3	2.6×10^{-5}	1165		
nf			7325	7325
na			2663	2663
ng			4994	4994
FCD4	4.3×10^{-5}	174		
nf			1467	1467
na			768	768
ng			1118	1118
FCD5	-5.8×10^{-1}	7		
nf			48	48
na			18	18
ng			33	33

Os testes de desempenho de um algoritmo usando-se um dado conjunto de funções, usualmente, mostram que o algoritmo se presta à solução do problema proposto e que, além disso, seu desempenho é melhor que o de outros algoritmos afins. Na área de otimização, é prática usual dar-se ênfase aos testes de desempenho que avaliam a eficiência dos algoritmos, o que pode ser feito, por exemplo, pela observação do número de avaliações da função objetivo, número de avaliações do gradiente e esforço computacional

necessário para se chegar a uma dada solução. Entretanto, o resultado desses testes pode não revelar diferenças significativas quanto à eficiência de dois algoritmos que são comparados, especialmente quando se trata de algoritmos similares.

Pelos resultados que constam da tabela 4.1, pode-se verificar que, para quatro das funções de teste, o algoritmo de Brent requereu um número menor de avaliações da função que o algoritmo de Gill & Murray (versão sem derivadas). Vale frisar, que não se trata de diferença significativa, uma vez que a redução do número de avaliações de função por iteração do método da descida mais íngreme é da ordem de 0,037. A obtenção de resultados bastante parecidos já poderia ser esperada uma vez que a previsão do mínimo gerada pelos dois algoritmos só é diferente se o ponto gerado pela interpolação quadrática for rejeitado.

Os resultados da tabela 4.2 mostram que nos cinco testes realizados não houve diferença no desempenho do algoritmo da Bissecção em relação ao algoritmo de Gill & Murray (versão com derivadas). Esse resultado é justificado uma vez que no algoritmo de Gill & Murray a estratégia de comparação de função quando os pontos estão em configuração de interpolação (relação (3.1)) é idêntica à do algoritmo da Bissecção que usa o ponto médio do intervalo como previsão do mínimo.

4.4.2 Minimização de funções que não são continuamente diferenciáveis

Os resultados dos testes de desempenho dos algoritmos usados para minimizar as funções relacionadas no apêndice B são apresentados nas tabelas 4.3 e 4.4. As tabelas incluem, para cada função, as seguintes informações: valor final da função, número de avaliações da função (nf) e número de iterações (nit). Na tabela 4.4, ng denota o número de avaliações do gradiente.

Os resultados contidos na tabela 4.3 são relativos a dois algoritmos que não utilizam derivadas como informações: algoritmo de Brent e algoritmo de Murray & Overton (versão simplificada, caso sem derivadas).

Tabela 4.3

Testes de desempenho usando funções que não são continuamente diferenciáveis (caso sem derivadas)

funções de teste	Valor final de $F_S(\alpha)$	BRENT	MURRAY & OVERTON
F_{S1}	4.0×10^{-7}		
nf		24	19
nit		22	17
F_{S2}	3.8×10^{-15}		
nf		48	34
nit		46	32
F_{S3}	6.6×10^{-4}		
nf		34	31
nit		32	29
F_{S4}	4.0		
nf		11	10
nit		9	8
F_{S5}	68.0		
nf		7	6
nit		5	4

A tabela 4.4 contém os resultados relativos aos algoritmos da Bissecção e algoritmo de Murray & Overton (versão simplificada, caso com derivadas).

Tabela 4.4

Testes de desempenho usando funções que não são continuamente diferenciáveis (caso com derivadas)

funções de teste	Valor final de $F_S(\alpha)$	BISSECÇÃO	MURRAY & OVERTON
F_{S1}	4.0×10^{-7}		
nf		18	8
ng		18	8
nit		16	6
F_{S2}	3.8×10^{-15}		
nf		24	17
ng		24	17
nit		22	15
F_{S3}	6.6×10^{-4}		
nf		24	10
ng		24	10
nit		22	8
F_{S4}	4.0		
nf		16	12
ng		16	12
nit		14	10
F_{S5}	68.0		
nf		24	9
ng		24	9
nit		22	7

Os resultados da tabela 4.3 mostram que nos cinco testes realizados, o algoritmo de Murray & Overton se mostrou mais eficiente que o algoritmo de Brent. O número de avaliações de função requerido pelo algoritmo de Murray & Overton foi, em média, 16% menor que aquele requerido pelo algoritmo de Brent.

Os resultados da tabela 4.4 mostram que nos cinco testes realizados, o algoritmo de Murray & Overton se mostrou mais eficiente, requerendo, em média, 46% menos avaliações da função e do gradiente em relação ao algoritmo da Bissecção.

CONCLUSÕES

Os algoritmos híbridos com proteções para minimização de funções continuamente diferenciáveis combinam a velocidade de convergência do processo de aproximações polinomiais sucessivas com a propriedade de convergência global dos processos de comparação de função. Esses algoritmos foram incorporados a um algoritmo para minimização multidimensional para fazer a busca linear exata ao longo da direção descendente $\tilde{p}$, com os parâmetros ε e τ feitos igual a 10^{-6} e $10^{-6}/\|\tilde{p}^{(k)}\|_2$. Em todos os testes, os algoritmos apresentaram desempenho satisfatório. Em muitos dos métodos descendentes, é necessário que se faça uma busca linear exata. No entanto, esse não é o caso geral e os algoritmos apresentados podem também ser usados para fazer uma busca linear não exata. Para isso, é somente necessário escolher-se adequadamente os parâmetros ε e τ ou então incluir um critério adicional de terminação como aqueles sugeridos por Murray & Overton (1978).

O algoritmo de Murray & Overton, em sua versão simplificada, foi implementado e usado na minimização de funções que não são continuamente diferenciáveis, elabora-

das pelo autor. Os resultados numéricos obtidos mostraram a potencialidade do algoritmo e, por isso, se constituem em um estímulo para que se procure conseguir uma vasta experiência no uso desse algoritmo. As duas versões do algoritmo podem ser modificadas e usadas na minimização de uma classe mais abrangente de funções não diferenciáveis.

REFERÊNCIAS BIBLIOGRAFICAS

[1] ADBY, P. R. & DEMPSTER, M. A. H., Introduction do optimization methods, Chapman and Hall, London, 1974.

[2] BRENT, R. P., Algorithms for minimization wihtout derivatives, Prentice-Hall, Englewood Cliffs, N. J., 1973.

[3] GILL, P. E. & MURRAY, W., Safeguarded steplenght algorithms for optimization using descent methods, National Physical Laboratory, Report NAC 37, 1974.

[4] GOLDFARB, D., Curvilinear path steplenght algorithms for minimization which use directions of negative curvature, Mathematical programming 18, pp 31-40, North Holland Publishing Company, 1980.

[5] GOTTFRIED, B. S., & WEISMAN, J., Introduction to optimization theory, Englewood Cliffs, Prentice-Hall, 1973.

[6] HIMMELBLAU, D. M., Apllied nonlinear programming, McGraw Hill Book Company, Austin, Texas, 1972.

[7] McCORMICK, G. P., A modification of armijo's step-size rule for negative curvature, Mathematical programming 13, pp 111-115, North-Holland Publishing Company, 1977.

[8] MORE, J. J., GARBOW, B. S. & HILLSTRON, K. E., Testing unconstrained optimization software, ACM Trans. Math. Software 7, pp 17-41, 1981.

[9] MORE, J. J., & SORENSEN, D. C. , On the use of direction or negative curvature in a odified newton method, Mathematical programming 16, pp 1-20, North Holland Publishing Company, 1979.

[10] MURRAY, W. & OVERTON, M. L., Steplenght algorithms for minimizing a class of nondifferentiable functions, Stanford University, Report n. STAN-CS-78-679, 1978.

[11] PEQUENO, M. C., Biblioteca educacional de otimização unidimensional não linear sem restrições, Universidade Federal da Paraiba, 1983.

[12] WOLFE, M. A., Numerical methods for unconstrained optimization, Van Nostrand Reinhold Company, 1978.

APÊNDICE A

FUNÇOES DE TESTE CONTINUAMENTE DIFERENCIAVEIS

Neste apêndice são apresentadas as cinco funções continuamente diferenciáveis que foram utilizadas nos testes de desempenho dos algoritmos híbridos com proteções. Para cada função é apresentada a expressão analítica, o ponto inicial de pesquisa $\tilde{x}^{(0)}$, o valor do minimizante $\tilde{x}^*$ e o valor da função no seu ponto de mínimo, isto é, $F(\tilde{x}^*)$.

1. Função de Miele e Cantrell (Wolfe (1980))

$$F(\tilde{x}) = (\exp(x_1) - x_2)^4 + 100(x_2 - x_3)^6 + [\tan(x_3 - x_4)]^4 + x_1^8$$

$$\tilde{x}^{(0)} = [1,2,2,2]^T$$

$$\tilde{x}^* = [0,1,1,1]^T$$

$$F(\tilde{x}^*) = 0$$

2. Função singular de Powell (Moré et alli (1981))

$$F(\tilde{x}) = (x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4$$

$$\tilde{x}^{(0)} = [3,-1,0,1]^T$$

$$\tilde{x}^* = [0,0,0,0]^T$$

$$F(\tilde{x}^*) = 0$$

3. Função de Fletcher e Powell (Moré et alli (1981))

$$F(\tilde{x}) = 100[(x_3 - 10\nu)^2 + (r - 1)^2] + x_3^2$$

onde

$$r = (x_1^2 + x_2^2)^{1/2}$$

e

$$\nu = \begin{cases} \frac{1}{2\pi} \tan^{-1}(x_2/x_1) & (x_1 \geq 0) \\ \frac{1}{2\pi} \tan^{-1}(x_2/x_1) + \frac{1}{2} & (x_1 < 0) \end{cases}$$

$$\tilde{x}^{(0)} = [-1,0,0]^T$$

$$\tilde{x}^* = [1,0,0]^T$$

$$F(\tilde{x}^*) = 0$$

4. Função de Dixon (Wolfe (1980))

$$F(\tilde{x}) = (1 - x_1)^2 + (1 - x_{10})^2 + \sum_{i=1}^{9} (x_i^2 - x_{i+1})^2$$

$$\tilde{x}^{(0)} = [-2,\ldots,-2]^T$$

$$\tilde{x}^* = [1,\ldots,1]^T$$

$$F(\tilde{x}^*) = 0$$

5. Função de Powell (Wolfe (1980))

$$F(\tilde{x}) = x_1^4 + x_1 x_2 + (1 + x_2)^2$$

$$\tilde{x}^{(0)} = [0,0]^T$$

$$\tilde{x}^* = [0.696, -1.348]^T$$

$$F(\tilde{x}^*) = -0.58$$

APÊNDICE B

FUNÇOES DE TESTE
QUE NÃO SÃO CONTINUAMENTE DIFERENCIAVEIS

A seguir, são apresentadas cinco funções de teste que não são continuamente diferenciáveis, elaboradas pelo autor. Trata-se das funções do tipo $F_S(\alpha)$ que foram utilizadas nos testes de desempenho dos algoritmos. Na definição de cada função constam o número de funções componentes (m), a expressão analítica das funções componentes $f_i(\alpha)$, o intervalo de pesquisa [a,b] que foi utilizado nos testes, o valor do minimizante α^* e o valor da função no seu ponto de mínimo, isto é, $F_S(\alpha^*)$.

1. $F_{S1}(\alpha)$
   (a) m = 4
   (b) $f_1(\alpha) = 10\text{Cos}(\alpha)$
       $f_2(\alpha) = -10\text{Cos}(\alpha)$
       $f_3(\alpha) = 5\alpha - 10$
       $f_4(\alpha) = 3\alpha - 9$
   (c) [a,b] = $[0,\pi]$
   (d) $\alpha^* = \pi/2$ (ponto de descontinuidade da derivada)
   (e) $F_{S1}(\alpha^*) = 0$

2. $F_{S2}(\alpha)$

(a) $m = 4$

(b) $f_1(\alpha) = -\alpha^2 + 4x10^4$

$f_i(\alpha) = 10^{-3}[\alpha - 200(i-1)]^3 \quad i = 2,3,4$

(c) $[a,b] = [0,10^3]$

(d) $\alpha^* = 200$ (ponto de descontinuidade da derivada)

(e) $F_{S2}(\alpha^*) = 0$

3. $F_{S3}(\alpha)$

(a) $m = 5$

(b) $f_1(\alpha) = -(\alpha - 400)^3 + 2.7x10^4$

$f_i(\alpha) = \alpha - [400 + 30(i - 1)] \quad i = 2,...,5$

(c) $[a,b] = [0,10^3]$

(d) $\alpha^* = 430$ (ponto de descontinuidade da derivada)

(e) $F_{S3}(\alpha^*) = 0$

4. $F_{S4}(\alpha)$

(a) $m = 2$

(b) $f_1(\alpha) = \text{Cos}(\alpha/100) + 5$

$f_2(\alpha) = -10^3\exp(-0.01\alpha) + 18$

(c) $[a,b] = [0,200]$

(d) $\alpha^* = 100$ (a função é continuamente diferenciável em α^*)

(e) $F_{S4}(\alpha^*) = 4.0$

5. $F_{S5}(\alpha)$

(a) m = 2

(b) $f_1(\alpha) = \alpha^2 - 11\alpha + 10$

$f_2(\alpha) = \alpha^2 - 45\alpha + 450$

(c) [a,b] = [0,30]

(d) $\alpha^* = 14$ (a função é continuamente diferenciável em α^*)

(e) $F_{S5}(\alpha^*) = 68.0$