Carla Maria Alves Ferreira

Métodos para Inclusão de Raízes Simples de uma Equação Não Linear

Dissertação submetida à Universidade do Minho para obtenção do grau de Mestre em Matemática Computacional elaborada sob orientação da Professora Doutora Maria Raquel Valença.

Escola de Ciências
Universidade do Minho
Braga - 1999

À minha mãe.

Agradecimentos

Gostaria muito especialmente de exprimir o meu agradecimento à Prof. Doutora Raquel Valença, minha orientadora neste trabalho, pela sua inigualável colaboração e pelo tempo que sempre disponibilizou às minhas solicitações.

À Prof. Doutora Joana Soares agradeço o valioso apoio que me dispensou desde que entrei no Departamento de Matemática da Universidade do Minho.

Aos meus colegas do Departamento de Matemática o meu obrigado pelas variadas formas com que me ajudaram durante a realização deste trabalho.

Àqueles que me são mais íntimos não esquecerei a compreensão constante que me manifestaram.

Índice

Introdução

Numerosas e importantes questões em assuntos diversos conduzem à resolução de uma equação não linear. O problema de encontrar os zeros reais de uma dada função foi intensivamente estudado durante muitos anos e existem vários métodos descritos na literatura para a sua resolução. São em geral métodos que envolvem conceitos relativamente simples e que levam a interpretações gráficas bastante interessantes. Recentemente em Alefeld e Potra [1] e em Alefeld, Potra e Yixun Shi [2] e [3] foram apresentados alguns *métodos intervalares* para a resolução numérica de uma única equação de uma variável real. As propriedades de convergência que estes métodos exibem são bastante apelativas e o seu desempenho prático é comparável ao desempenho de eficientes métodos há muito conhecidos, tais como os métodos de Dekker [9] e Brent [7]. Neste trabalho faremos a descrição daqueles métodos e uma análise detalhada das suas propriedades de convergência.

A interpolação polinomial constitui um processo simples de obter aproximações para uma função e é fundamento de vários métodos numéricos para calcular zeros de funções. Se aliarmos a técnicas de interpolação a segurança do método da bissecção conseguem-se construir métodos bastante eficientes. Os métodos que descreveremos no Capítulo 1, designados por **métodos I a VII**, caracterizam-se, em cada iteração, por passos de interpolação linear e interpolação quadrática ou interpolação cúbica inversa, seguidos de passos *tipo* bissecção. Dada a equação $f(x) = 0$, sendo $f : [a, b] \to \mathbb{R}$ uma função contínua com um zero x^* em $[a, b]$, os métodos I a VII constroem uma sucessão de intervalos $\{[a_n, b_n]\}_{n=0}^{\infty}$ tal que

$$x^* \in [a_{n+1}, b_{n+1}] \subset [a_n, b_n] \subset \ldots \subset [a_0, b_0]$$

e

$$\lim_{n\to\infty} (b_n - a_n) = 0,$$

em que o intervalo inicial $[a_0, b_0] = [a, b]$ satisfaz a condição de $f(a)$ e $f(b)$ terem sinais contrários.

Sem uma análise cuidada do efeito dos erros de arredondamento não é possível garantir a inclusão do zero x^* num intervalo final $[a_n, b_n]$ com $b_n - a_n \leq 2\delta$, sendo δ uma tolerância especificada. No final do primeiro

capítulo faremos referência ao critério de paragem usado e apresentaremos um breve estudo sobre o efeito de alguns erros de arredondamento.

No capítulo 2 estudaremos as propriedades de convergência da sucessão $\{[a_n, b_n]\}_{n=0}^{\infty}$ gerada pelos métodos I a VII. Começaremos por introduzir os conceitos de *Q-ordem* e de *R-ordem* de convergência de uma sucessão, optando por definições inteiramente semelhantes às dadas em Potra [15]. Devidos a Schmidt [24], seguem-se importantes resultados relacionados com a R-ordem de convergência de uma sucessão. Aceitando uma sugestão dada em Ralston [23], veremos igualmente que o *índice de eficiência computacional* de um método iterativo se pode definir por $\sqrt[p]{q}$, onde q é a Q-ordem ou R-ordem de convergência do método e p é o número de avaliações da função f (ou das suas derivadas) necessário assimptoticamente em cada iteração.

Exigindo condições comuns de suavidade à função f, apresentaremos resultados que mostram que a sucessão $\{(b_n - a_n)\}_{n=0}^{\infty}$ converge para zero

- com Q-ordem de convergência pelo menos 2 e 4, respectivamente para os métodos I e II, e

- com R-ordem de convergência pelo menos igual a 3.3027... no método III, 2.4142... no método IV, 4.2360... no método V, 2.7320... no método VI e finalmente 4.6457... no método VII.

Os índices de eficiência computacional dos métodos I a VII serão respectivamente: 1.4142..., 1.5874..., 1.4892..., 1.5537..., 1.6180..., 1.6528... e 1.6685.... Veremos assim que o **método VII** apresenta o maior índice de eficiência computacional entre todos os métodos e, por último, mostraremos que este método, tal como o método V, é um método óptimo dentro de uma certa família de métodos.

O capítulo 3 destina-se à apresentação de alguns problemas numéricos cuja resolução exemplifica o comportamento prático dos métodos I a VII. Faremos primeiro referência às conclusões dos autores relativas às suas experiências numéricas que evidenciam o que se esperava: o método VII é de facto o método mais eficiente entre todos os métodos, apresentando em alguns problemas vantagem sobre os métodos de Dekker e Brent.

Para as nossas próprias experiências realizámos uma implementação dos algoritmos no programa *Mathematica*. Iremos referir algumas características deste programa, nomeadamente a capacidade de representação arbitrária de números reais que usaremos para testar os exemplos numéricos de resolução de equações não lineares apresentados.

No último capítulo apresentaremos breves conclusões a respeito das condições de aplicação e propriedades de convergência e eficiência dos métodos I a VII.

Reservámos três apêndices para incluir um resumo sobre interpolação de Neville, alguns resultados relativos a *matrizes não negativas* e a listagem do programa que desenvolvemos no *Mathematica*.

Capítulo 1

Inclusão de zeros de funções contínuas

1.1 Introdução

O problema de determinar os zeros reais de uma dada função contínua aparece frequentemente em vários domínios da matemática. A resolução numérica de uma equação não linear é assim um dos problemas fundamentais em análise numérica.

Neste capítulo consideraremos o cálculo de uma aproximação $\widehat{x}$ para a raiz da equação não linear,

$$f(x) = 0, \tag{1.1}$$

onde f denota uma função real de variável real contínua. Por raiz de (1.1), ou zero da função f, entende-se um número x^* tal que $f(x^*) = 0$.

Os diferentes métodos iterativos que têm sido estudados podem ser classificados em dois grupos - *métodos de intervalo aberto* e *métodos de encaixe*. O primeiro grupo engloba métodos localmente convergentes como o método do *ponto fixo simples* e métodos de *interpolação linear inversa* como o de *Newton* e o da *secante*. A maior parte dos métodos do segundo grupo são métodos globalmente convergentes e caracterizam-se por definir, em cada iteração, um intervalo *encaixado* no intervalo anterior e que contém a raiz. Como exemplos temos o método da *bissecção*, da *falsa posição* e modificações deste último.

O mais simples método de encaixe é o método da bissecção. Trata-se de um método que é aplicável à classe de todas as funções que mudam de sinal num certo intervalo $[a, b]$. Contudo, embora sendo absolutamente seguro que não falhe, este algoritmo é muito lento. Para outras classes de funções como, por exemplo, funções suficientemente suaves com duas ou mais derivadas contínuas e com zeros simples, é normalmente possível aplicar um método de convergência mais rápida como o método de Newton ou o da secante.

Um dos melhores algoritmos para determinar um zero real de uma função contínua combina de forma interessante num eficiente esquema computacional a garantia de convergência do método da bissecção com a maior rapidez de convergência dos métodos da secante e interpolação quadrática inversa. Este algoritmo foi inicialmente publicado por Dekker [9] em 1969 e melhorado mais tarde por Brent [7]. Trata-se de um algoritmo um pouco complicado podendo, no entanto, as suas principais características ser vistas numa versão simplificada, método de Dekker-Brent, apresentada por Vandergraft [28]. Uma implementação em Fortran da versão completa desenvolvida por Brent pode ser encontrada em Forsythe [12]. Para este último algoritmo temos a garantia de que nunca é mais lento do que o da bissecção e de que é mesmo mais rápido nos casos em que f é uma função suave.

As ideias que estão na base dos algoritmos de Dekker e Brent serviram de motivação para outros mais recentes desenvolvidos por Alefeld e Potra [1] e Alefeld, Potra e Yixun Shi [2] e [3]. Neste capítulo faremos a descrição de sete métodos iterativos de encaixe propostos por estes autores os quais designaremos simplesmente por **método I** a **método VII.** Os passos envolvidos em cada iteração são relativamente simples - os métodos I a V baseiam-se num *passo secante duplo* e no uso de técnicas de interpolação quadrática (excepto o primeiro) associados a reduções apropriadas dos sucessivos intervalos de inclusão de x^*. Os métodos IV e V constituem melhores versões do método II. Aplicar interpolação cúbica inversa em vez de interpolação quadrática sempre que possível é o que se acresce na implementação dos métodos VI e VII, constituindo versões últimas melhoradas dos métodos anteriores.

As experiências numéricas já realizadas pelos autores evidenciam que estes métodos têm um desempenho tão bom e, em certos casos, mesmo melhor que os métodos de Dekker e Brent. O **método VII** apresenta o melhor comportamento de todos, especialmente quando a tolerância exigida para o erro é *pequena.* Verifica-se assim que com um pequeno acréscimo na complexidade dos algoritmos se vão obtendo melhorias na rapidez de convergência.

O critério de paragem aplicado assim como uma pequena análise do efeito dos erros de arredondamento serão apresentados no final do capítulo.

Será apenas no próximo capítulo que apresentaremos os resultados relativos à ordem de convergência e eficiência computacional dos métodos referidos. Veremos então mais tarde que, sob certas condições de suavidade da função f, estes métodos atingem índices de eficiência muito bons.

1.2 Métodos intervalares

Os métodos de encaixe para a resolução da equação (1.1), também designados por métodos *intervalares* ou *de inclusão,* tentam determinar uma

sucessão de intervalos que converge para a solução x^*. Isto é, conhecido um intervalo inicial $[a,b]$ a que pertença x^*, os sucessivos intervalos devem conter x^* e a sua amplitude convergir para zero.

Uma importante vantagem dos métodos intervalares é que limites rigorosos para o erro $|\widehat{x} - x^*|$ estão sempre disponíveis. De facto, se $[a_n, b_n]$ é um intervalo que contém a solução x^* e $\widehat{x} \in [a_n, b_n]$, então $|\widehat{x} - x^*| \leq b_n - a_n$. Além disso, com um intervalo inicial apropriado, a maior parte dos métodos intervalares tem garantia de convergência e poderá ser encontrado um intervalo de inclusão de x^* tão pequeno quanto permita o sistema de numeração de vírgula flutuante usado.

A principal desvantagem de um método intervalar tem a ver com a necessidade de um trabalho prévio de localização da raiz x^*. De facto, a exigência de um intervalo inicial que contém a raiz pode limitar um pouco a aplicabilidade destes métodos.

Em todos os métodos que iremos descrever, será assumido que para uma dada função contínua $f : [a,b] \to \mathbb{R}$ a condição

$$f(a)f(b) < 0 \tag{1.2}$$

é sempre satisfeita. Sendo f uma função contínua no intervalo $[a,b]$, pelo teorema do valor intermédio esta condição garante a existência de pelo menos um zero de f em (a,b). O objectivo destes métodos será assim procurar não por um zero de f mas por um intervalo estreito no qual f mude de sinal.

1.2.1 Alguns métodos recentes - métodos I a VII

Seja $f : [a,b] \to \mathbb{R}$ uma função contínua com um zero simples[1] x^* em $[a,b]$. Começando com um intervalo inicial $[a_0, b_0] = [a,b]$ satisfazendo (1.2), consideraremos o problema de construir uma sucessão de intervalos $\{[a_n, b_n]\}_{n=0}^{\infty}$ tal que

$$x^* \in [a_{n+1}, b_{n+1}] \subset [a_n, b_n] \subset \ldots \subset [a_0, b_0] = [a,b] \tag{1.3}$$

e

$$\lim_{n\to\infty} (b_n - a_n) = 0. \tag{1.4}$$

Nas secções seguintes apresentaremos métodos desenvolvidos recentemente para cuja aplicação não se exigem condições de convexidade e que satisfazem (1.3) e (1.4). Embora estes métodos sejam igualmente aplicáveis a problemas mais genéricos envolvendo funções ou derivadas descontínuas, funções com zeros múltiplos, etc., as propriedades de convergência só se verificam no caso de f ser suficientemente suave e ter um zero simples x^* em $[a,b]$.

[1] x^* é um zero de multiplicidade m de $f \in C^{(m)}[x^* - r, x^* + r]$ se e só se $f(x^*) = 0$, $f'(x^*) = f''(x^*) = \cdots = f^{(m-1)}(x^*) = 0$ e $f^{(m)}(x^*) \neq 0$.
Se $m = 1$ x^* diz-se um zero simples.

Os primeiros três métodos, que passaremos a designar por **métodos I, II e III,** foram propostos por Alefeld e Potra [1]. Começaremos pela sua descrição.

Método I

Seja $[a,b]$ um intervalo de inclusão do zero x^* e considere-se um ponto $c \in [a,b]$. Tendo em conta apenas o sinal de $f(c)$, podemos facilmente "partir" o intervalo $[a,b]$. É o que implementa o procedimento *partir_intervalo* - constrói um novo intervalo $[\overline{a},\overline{b}] \subset [a,b]$ que continua a conter x^*. Consiste no seguinte:

partir_intervalo $([a,b],c)$
se $f(c) = 0$ **então** escrever c;
se $f(a)f(c) < 0$ **então** $[\overline{a},\overline{b}] = [a,c]$;
senão $[\overline{a},\overline{b}] = [c,b]$;
devolver $[\overline{a},\overline{b}]$.

Se os valores de f em a e em b forem conhecidos então cada chamada a *partir_intervalo* requer apenas a avaliação de um valor de f.

No método da bissecção este procedimento é repetido sucessivamente escolhendo-se $c = \frac{1}{2}(a+b)$ até se obter um intervalo de inclusão tão estreito quanto se deseje. O objectivo de se realizarem repetidas chamadas a *partir_intervalo* exige, portanto, incluir um critério de paragem.

Designando por δ a tolerância que se espera obter para o erro absoluto na aproximação de x^*, poderemos então construir um novo procedimento, *partir_intervalo*1, que implementará também o teste de convergência. Será da forma:

*partir_intervalo*1 $([a,b],c)$
se $f(c) = 0$ **então** escrever c e parar;
se $f(a)f(c) < 0$ **então** $[\overline{a},\overline{b}] = [a,c]$;
senão $[\overline{a},\overline{b}] = [c,b]$;
se $\overline{b} - \overline{a} \leq 2\delta$ **então** escrever $[\overline{a},\overline{b}]$ e parar;
devolver $[\overline{a},\overline{b}]$.

Pormenores sobre a implementação deste critério de paragem serão analisados numa secção mais à frente.

Consideremos a primeira iteração do **método I** iniciada com o intervalo $[a_0, b_0] = [a, b]$. O primeiro passo consiste no uso da fórmula do método da *secante* para determinar um ponto $c \in [a, b]$, isto é, calcular

$$c = a - f\,[a, b]^{-1} f(a),$$

onde a notação $f\,[x_1, x_2]$ designa a *diferença dividida* de primeira ordem de f em x_1 e x_2 que é definida por

$$f\,[x_1, x_2] = \frac{f(x_2) - f(x_1)}{x_2 - x_1}. \tag{1.5}$$

Efectua-se de seguida *partir_intervalo*1 $([a, b], c)$ e obtém-se um novo intervalo de inclusão $[\overline{a}, \overline{b}] \subset [a_0, b_0]$.

No próximo passo tenta-se obter um melhor intervalo de inclusão da raiz usando um ponto $\overline{c}$ calculado através de um *passo secante duplo*, isto é, determinando $\overline{c}$ da forma

$$\begin{cases} \textbf{se } |f(\overline{a})| < \left|f(\overline{b})\right| \textbf{ então } u = \overline{a} \textbf{ senão } u = \overline{b}; \\ \overline{c} = u - 2f\,[\overline{a}, \overline{b}]^{-1} f(u). \end{cases} \tag{1.6}$$

Deve notar-se que $\overline{c}$ é a abcissa do ponto de intersecção com o eixo dos xx da recta que passa no ponto $(u, f(u))$ e tem declive $\frac{1}{2}\, f\,[\overline{a}, \overline{b}]$ - a recta de equação

$$y = f(u) + \frac{1}{2} f\,[\overline{a}, \overline{b}]\,(x - u), \tag{1.7}$$

e é importante observar que $\overline{c}$ pertence sempre ao intervalo $[\overline{a}, \overline{b}]$. Com efeito,

- se $|f(\overline{a})| < \left|f(\overline{b})\right|$, a equação em (1.7) fica $y = f(\overline{a}) + \frac{1}{2} f\,[\overline{a}, \overline{b}]\,(x - \overline{a})$ e os pontos $(\overline{a}, f(\overline{a}))$ e $\left(\overline{b}, \frac{f(\overline{a}) + f(\overline{b})}{2}\right)$ pertencem a esta recta; $\frac{f(\overline{a}) + f(\overline{b})}{2}$ terá sempre o sinal de $f(\overline{b})$ e, uma vez que $f(\overline{a}) f(\overline{b}) < 0$, o ponto $\overline{c}$ pertence a $[\overline{a}, \overline{b}]$;

- se $|f(\overline{a})| > \left|f(\overline{b})\right|$, a equação em (1.7) fica $y = f(\overline{b}) + \frac{1}{2} f\,[\overline{a}, \overline{b}]\,(x - \overline{b})$ e os pontos $\left(\overline{a}, \frac{f(\overline{a}) + f(\overline{b})}{2}\right)$ e $(\overline{b}, f(\overline{b}))$ pertencem a esta recta; analogamente, $\frac{f(\overline{a}) + f(\overline{b})}{2}$ terá o sinal de $f(\overline{a})$ e, dado que $f(\overline{a}) f(\overline{b}) < 0$, o ponto $\overline{c}$ pertence a $[\overline{a}, \overline{b}]$;

- se $|f(\overline{a})| = \left|f(\overline{b})\right|$, $u = \overline{b}$ e, como $f(\overline{a})$ e $f(\overline{b})$ têm sinais contrários, $y(\overline{a}) = 0$, isto é, $\overline{c} = \overline{a}$.

Segue-se também de (1.6) e de (1.5) que

$$\overline{c} = u - 2\frac{\overline{b} - \overline{a}}{f(\overline{b}) - f(\overline{a})} f(u),$$

e sendo

$$f(\overline{c}) = \frac{f(\overline{c}) - f(u)}{\overline{c} - u}(\overline{c} - u) + f(u),$$

vem

$$\begin{aligned} f(\overline{c}) &= -2\frac{f(\overline{c}) - f(u)}{\overline{c} - u}\frac{\overline{b} - \overline{a}}{f(\overline{b}) - f(\overline{a})} f(u) + f(u) \\ &= f(u)\left[1 - 2\frac{f(\overline{c}) - f(u)}{\overline{c} - u}\frac{\overline{b} - \overline{a}}{f(\overline{b}) - f(\overline{a})}\right]. \end{aligned} \tag{1.8}$$

Supondo que f é continuamente diferenciável em $[a, b]$ e que os pontos $\overline{a}$, $\overline{b}$, u e $\overline{c}$ pertencem a uma pequena vizinhança do zero simples x^*, então

$$\frac{f(\overline{b}) - f(\overline{a})}{\overline{b} - \overline{a}} \approx f'(x^*) \approx \frac{f(\overline{c}) - f(u)}{\overline{c} - u}.$$

Assim, de acordo com (1.8) e notando que, sendo x^* um zero simples, se tem $f'(x^*) \neq 0$, virá

$$f(\overline{c}) \approx f(u)\left[1 - 2f'(x^*)\frac{1}{f'(x^*)}\right] = -f(u),$$

o que mostra que x^* está entre $\overline{c}$ e u.

Teremos então que, se $[\overline{a}, \overline{b}]$ é um intervalo suficientemente pequeno contendo um zero simples de f, a fórmula em (1.6) pode ser usada para obter um melhor intervalo de inclusão de x^*, fornecendo um ponto $\overline{c} \in [\overline{a}, \overline{b}]$ tal que

$$x^* \in [\min\{u, \overline{c}\}, \max\{u, \overline{c}\}]. \tag{1.9}$$

Dado que $|f(u)| = \min\left\{|f(\overline{a})|, |f(\overline{b})|\right\}$, podemos admitir que x^* está mais perto de u e passar para um intervalo de inclusão em que um dos extremos seja u parece ser o mais eficiente.

O passo seguinte seria invocar *partir_intervalo*1 $\left([\overline{a}, \overline{b}], \overline{c}\right)$. Mas antes os autores consideram ainda determinar $\widehat{c}$ da forma

$$\begin{aligned} \textbf{se } |\overline{c} - u| > 0.5(\overline{b} - \overline{a}) \quad &\textbf{então } \widehat{c} = 0.5\left(\overline{b} + \overline{a}\right); \\ &\textbf{senão } \widehat{c} = \overline{c}, \end{aligned}$$

seguindo-se então *partir_intervalo*1 $\left([\overline{a}, \overline{b}], \widehat{c}\right)$ de que resulta $[\widehat{a}, \widehat{b}] \subset [\overline{a}, \overline{b}]$.

Se ocorrer a situação em (1.9), definir $\widehat{c}$ da forma apresentada garante que $[\widehat{a}, \widehat{b}]$ tenha sempre amplitude não superior a $|\overline{c} - u|$ e no máximo igual a metade da amplitude de $[\overline{a}, \overline{b}]$. De facto, se $|\overline{c} - u| \leq 0.5(\overline{b} - \overline{a})$ então teremos $\left\{\widehat{a}, \widehat{b}\right\} = \{\overline{c}, u\}$ e $\widehat{b} - \widehat{a} = |\overline{c} - u|$. Caso contrário, obteremos melhor fazendo uma bissecção de $[\overline{a}, \overline{b}]$, caso em que $\widehat{b} - \widehat{a} = 0.5(\overline{b} - \overline{a}) < |\overline{c} - u|$. Assim, acontecerá sempre

$$\widehat{b} - \widehat{a} \leq |\overline{c} - u|,$$

quando já se está suficientemente perto da raiz x^*.

O último passo desta iteração consiste em determinar o intervalo de inclusão $[a_1, b_1]$ da forma seguinte:

$$\begin{aligned} \textbf{se } \widehat{b} - \widehat{a} \;<\; & \mu\,(b_0 - a_0) \qquad (1.10) \\ & \textbf{então } [a_1, b_1] = \left[\widehat{a}, \widehat{b}\right]; \\ & \textbf{senão } [a_1, b_1] = \textit{partir\_intervalo1}\left(\left[\widehat{a}, \widehat{b}\right], 0.5\left(\widehat{b} + \widehat{a}\right)\right), \end{aligned}$$

onde μ é um parâmetro que deve satisfazer $0 < \mu < 1$. Normalmente, os autores consideram a escolha $\mu = 0.5$.

A próxima iteração começaria com o intervalo $[a_1, b_1]$ repetindo-se os mesmos passos.

Realizar um último passo como descrito acima, com $\mu \leq 0.5$, garante que o método I produz intervalos que satisfazem pelo menos

$$b_{n+1} - a_{n+1} \leq 0.5\,(b_n - a_n), \qquad n = 0, 1, \ldots,$$

o mesmo que em cada iteração do método da bissecção.

De qualquer forma, no caso em que já se está suficientemente próximo de x^*, a condição em (1.10) é sempre satisfeita (veja-se secção 2.4.1, teorema 2.4.3) e, portanto, a última bissecção não se realizará, ficando sempre $[a_{n+1}, b_{n+1}] = \left[\widehat{a}_n, \widehat{b}_n\right]$. Teremos, assim, que o método I requer por iteração,

- no pior dos casos, 3 avaliações da função f, correspondentes a três chamadas a *partir_intervalo1*, e
- assimptoticamente, 2 avaliações da função f, dado não se realizar a chamada a *partir_intervalo1* no último passo.

Em resumo, podemos dizer que o método I garante sempre convergência desde que se inicie com um intervalo que inclua a raiz e, no pior dos casos, por iteração, pode precisar de três vezes mais avaliações de f do que requer o método da bissecção.

A presentamos de seguida uma descrição pormenorizada do algoritmo do método I.

Algoritmo I (Alefeld-Potra, 1992)

$[a_0, b_0] = [a, b]$;

para $n = 1, 2, \ldots$ **até** $n = nmax$ **fazer**

1.1 $c_n = a_n - f\,[a_n, b_n]^{-1} f(a_n)$;

1.2 $\left[\overline{a}_n, \overline{b}_n\right] = partir\_intervalo1\left([a_n, b_n], c_n\right)$;

1.3 **se** $|f(\overline{a}_n)| < |f(\overline{b}_n)|$ **então** $u_n = \overline{a}_n$;

senão $u_n = \overline{b}_n$;

1.4 $\overline{c}_n = u_n - 2f\left[\overline{a}_n, \overline{b}_n\right]^{-1} f(u_n)$;

1.5 **se** $|\overline{c}_n - u_n| > 0.5(\overline{b}_n - \overline{a}_n)$ **então** $\widehat{c}_n = 0.5\left(\overline{b}_n + \overline{a}_n\right)$;

senão $\widehat{c}_n = \overline{c}_n$;

1.6 $\left[\widehat{a}_n, \widehat{b}_n\right] = partir\_intervalo1\left(\left[\overline{a}_n, \overline{b}_n\right], \widehat{c}_n\right)$;

1.7 **se** $\widehat{b}_n - \widehat{a}_n < \mu\,(b_n - a_n)$

então $[a_{n+1}, b_{n+1}] = \left[\widehat{a}_n, \widehat{b}_n\right]$;

senão $[a_{n+1}, b_{n+1}] = partir\_intervalo1\left(\left[\widehat{a}_n, \widehat{b}_n\right], 0.5\left(\widehat{b}_n + \widehat{a}_n\right)\right)$.

fpara

Deve observar-se que em casos de convergência o algoritmo I terminará sempre nalguma chamada a *partir_intervalo*1, no passo 1.2, 1.6 ou 1.7. Ao especificar-se um número máximo de iterações com o parâmetro *nmax* pretende-se apenas precaver contra situações em que, por algum motivo, não esteja a verificar-se convergência.

A figura 1.1 ilustra a interpretação geométrica do método I.

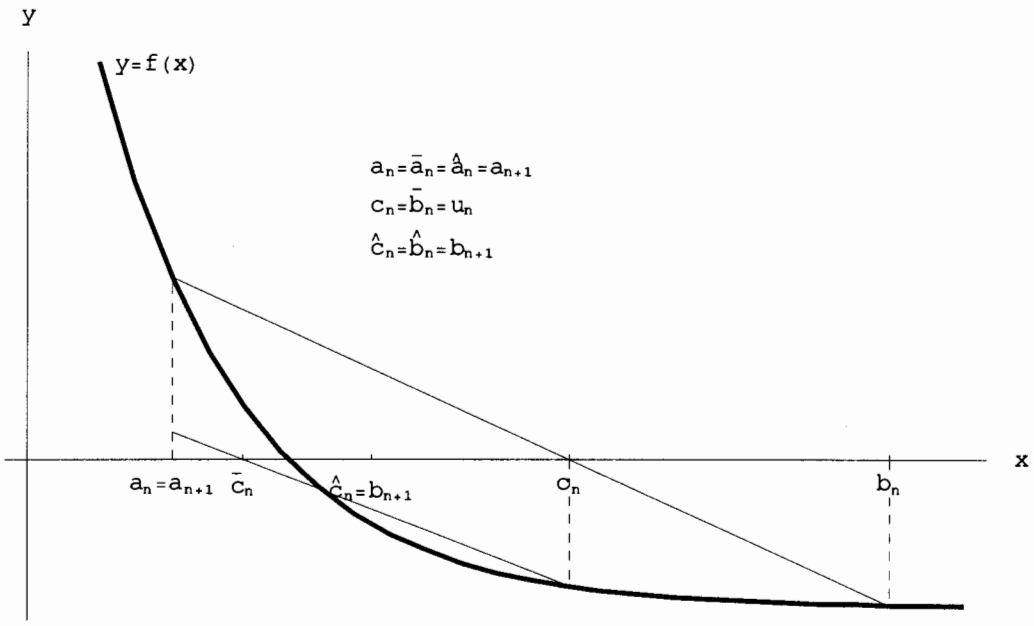

Figura 1.1 Método I

Método II

Para o intervalo $[a_n, b_n]$ considerem-se os primeiros dois passos de uma iteração do método I,

$$\begin{aligned} &c_n = a_n - f\left[a_n, b_n\right]^{-1} f(a_n); \\ &\left[\bar{a}_n, \bar{b}_n\right] = \textit{partir\_intervalo1}\left(\left[a_n, b_n\right], c_n\right). \end{aligned} \tag{1.11}$$

Uma iteração do **método II** começa igualmente por realizar estes passos.

Uma vez que f foi avaliada em três pontos distintos, a_n, b_n e c_n, parece razoável utilizar essa informação. Uma forma de o fazer será através do uso de interpolação quadrática.

Seja então $p_{(a_n,b_n,c_n)}$ o polinómio quadrático interpolador da função f nos pontos a_n, b_n e c_n. Uma vez que $[\overline{a}_n, \overline{b}_n] = [a_n, c_n]$ ou $[\overline{a}_n, \overline{b}_n] = [c_n, b_n]$, segue-se que

$$p_{(a_n,b_n,c_n)}(\overline{a}_n)\, p_{(a_n,b_n,c_n)}(\overline{b}_n) = f(\overline{a}_n) f(\overline{b}_n) < 0,$$

ou seja, este polinómio tem uma mudança de sinal em $[\overline{a}_n, \overline{b}_n]$ e, portanto, tem um único zero em $[\overline{a}_n, \overline{b}_n]$. Com a resolução exacta da equação quadrática

$$p_{(a_n,b_n,c_n)}(x) = 0, \tag{1.12}$$

determinamos este zero, obtendo assim um novo ponto $\overline{c}_n \in [\overline{a}_n, \overline{b}_n]$.

Após o cálculo de $\overline{c}_n$, o passo seguinte será reduzir o intervalo $[\overline{a}_n, \overline{b}_n]$ para $\left[\widehat{a}_n, \widehat{b}_n\right]$ ao fazer *partir_intervalo*1 $\left([\overline{a}_n, \overline{b}_n], \overline{c}_n\right)$. Teremos assim que, no método II, a (1.11) se seguem os passos

$$\begin{array}{l} \overline{c}_n = \text{único zero de } p_{(a_n,b_n,c_n)}(x) \text{ pertencente a } [\overline{a}_n, \overline{b}_n]\,; \\ \left[\widehat{a}_n, \widehat{b}_n\right] = \textit{partir\_intervalo}1\left([\overline{a}_n, \overline{b}_n], \overline{c}_n\right), \end{array}$$

e prossegue-se de forma semelhante aos passos 1.3 - 1.7 do método I.

Vejam-se os detalhes do algoritmo na página seguinte.

Da mesma forma que no método I, a condição no passo 2.9,

$$\widetilde{b}_n - \widetilde{a}_n < \mu(b_n - a_n),$$

com $0 < \mu < 1$, é sempre verificada a partir de um certo n (veja-se secção 2.4.1, teorema 2.4.4). Assim, neste último passo, quando já se estiver perto da raiz x^*, *partir_intervalo*1 não será invocado e ficaremos sempre com $[a_{n+1}, b_{n+1}] = \left[\widetilde{a}_n, \widetilde{b}_n\right]$. Sendo assim, o método II exige em cada iteração mais uma avaliação da função f do que o método I, isto é,

- o cálculo de 4 valores da função f, no pior dos casos, e
- assimptoticamente, o cálculo de 3 valores de f.

Com o método II a convergência está também assegurada. Em cada iteração a amplitude do intervalo de inclusão $[a_{n+1}, b_{n+1}]$ é pelo menos reduzida para metade da do intervalo $[a_n, b_n]$ e, portanto, relativamente ao método da bissecção, o método II, pode ser, no pior dos casos, quatro vezes mais lento por iteração.

Segue-se a apresentação do algoritmo do método II.

Algoritmo II (Alefeld-Potra, 1992)

$[a_0, b_0] = [a, b]$;

para $n = 1, 2, \ldots$ **até** $n = nmax$ **fazer**

2.1 $c_n = a_n - f[a_n, b_n]^{-1} f(a_n)$;

2.2 $[\overline{a}_n, \overline{b}_n] = partir\_intervalo1([a_n, b_n], c_n)$;

2.3 $\overline{c}_n =$ único zero de $p_{(a_n, b_n, c_n)}(x)$ pertencente a $[\overline{a}_n, \overline{b}_n]$;

2.4 $\left[\widehat{a}_n, \widehat{b}_n\right] = partir\_intervalo1\left(\left[\overline{a}_n, \overline{b}_n\right], \overline{c}_n\right)$;

2.5 **se** $|f(\widehat{a}_n)| < \left|f(\widehat{b}_n)\right|$ **então** $u_n = \widehat{a}_n$;

senão $u_n = \widehat{b}_n$;

2.6 $\widehat{c}_n = u_n - 2f\left[\widehat{a}_n, \widehat{b}_n\right]^{-1} f(u_n)$;

2.7 **se** $|\widehat{c}_n - u_n| > 0.5(\widehat{b}_n - \widehat{a}_n)$ **então** $\widetilde{c}_n = 0.5(\widehat{b}_n + \widehat{a}_n)$;

senão $\widetilde{c}_n = \widehat{c}_n$;

2.8 $\left[\widetilde{a}_n, \widetilde{b}_n\right] = partir\_intervalo1\left(\left[\widehat{a}_n, \widehat{b}_n\right], \widetilde{c}_n\right)$;

2.9 **se** $\widetilde{b}_n - \widetilde{a}_n < \mu(b_n - a_n)$

então $[a_{n+1}, b_{n+1}] = \left[\widetilde{a}_n, \widetilde{b}_n\right]$;

senão $[a_{n+1}, b_{n+1}] = partir\_intervalo1\left(\left[\widetilde{a}_n, \widetilde{b}_n\right], 0.5\left(\widetilde{b}_n + \widetilde{a}_n\right)\right)$.

fpara

Na figura 1.2 está representada graficamente uma iteração do método II.

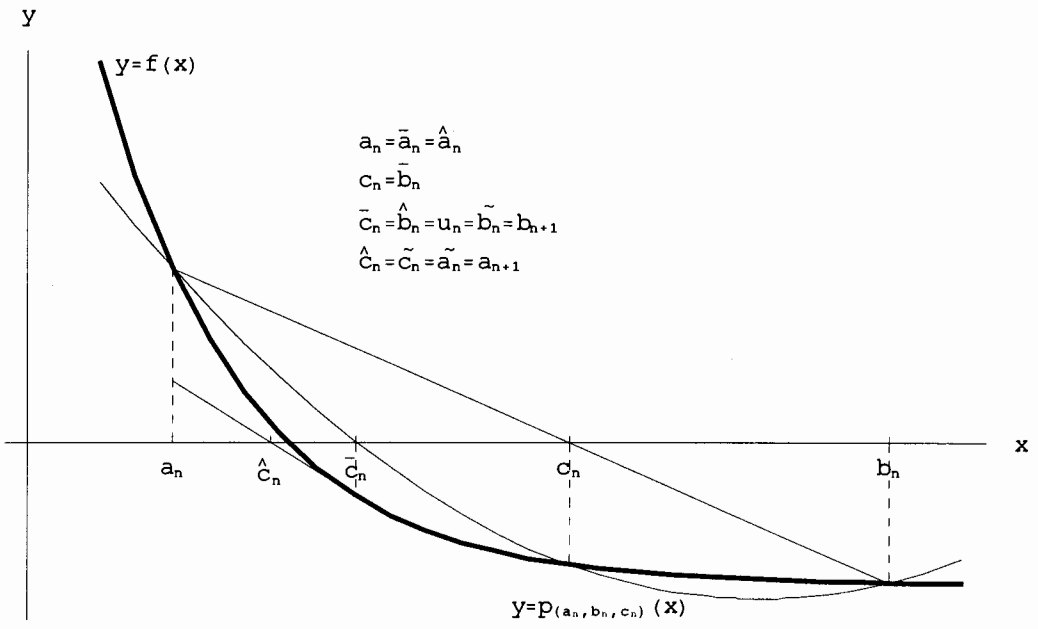

Figura 1.2 Método II

Método III

No **método III** o primeiro ponto c_n é escolhido como sendo simplesmente o ponto médio do intervalo $[a_n, b_n]$. Os primeiro passos de cada iteração serão então,

$$c_n = 0.5(a_n + b_n);$$
$$\left[\bar{a}_n, \bar{b}_n\right] = partir\_intervalo1\left([a_n, b_n], c_n\right),$$

isto é, começa-se com uma bissecção do intervalo inicial $[a_n, b_n]$.

Os passos seguintes são como em 2.3 - 2.6 do método II e termina-se com

$$[a_{n+1}, b_{n+1}] = partir\_intervalo1\left(\left[\widehat{a}_n, \widehat{b}_n\right], \widehat{c}_n\right). \tag{1.13}$$

Isto significa que, em relação ao método II, os passos 2.7 a 2.9 se substituem por (1.13).

Veremos que esta simplificação do método II (secção 2.4.1, teorema 2.4.5) resulta numa perda de rapidez de convergência. No entanto, o método III requer sempre três avaliações da função f por iteração, enquanto que o método II, embora apenas no pior dos casos, mas pode exigir quatro avaliações de f numa iteração.

Assim, uma vez que se garante que em ambos os métodos acontece $b_{n+1} - a_{n+1} \leq 0.5\,(b_n - a_n)$ $(n = 0, 1, \ldots)$, o método III poderá ser, por iteração, apenas três vezes mais lento do que o método da bissecção, enquanto que o método II poderá ser quatro vezes mais lento.

Os pormenores do algoritmo são apresentados a seguir.

Algoritmo III (Alefeld-Potra, 1992)

$[a_0, b_0] = [a, b]$;

para $n = 1, 2, \ldots$ **até** $n = nmax$ **fazer**

3.1 $c_n = 0.5(a_n + b_n)$;

3.2 $\left[\overline{a}_n, \overline{b}_n\right] = partir\_intervalo1\left([a_n, b_n], c_n\right)$;

3.3 $\overline{c}_n =$ único zero de $p_{(a_n, b_n, c_n)}(x)$ pertencente a $\left[\overline{a}_n, \overline{b}_n\right]$;

3.4 $\left[\widehat{a}_n, \widehat{b}_n\right] = partir\_intervalo1\left(\left[\overline{a}_n, \overline{b}_n\right], \overline{c}_n\right)$;

3.5 **se** $|f(\widehat{a}_n)| < \left|f(\widehat{b}_n)\right|$ **então** $u_n = \widehat{a}_n$;

senão $u_n = \widehat{b}_n$;

3.6 $\widehat{c}_n = u_n - 2f\left[\widehat{a}_n, \widehat{b}_n\right]^{-1} f(u_n)$;

3.7 $[a_{n+1}, b_{n+1}] = partir\_intervalo1\left(\left[\widehat{a}_n, \widehat{b}_n\right], \widehat{c}_n\right)$.

fpara

Pode ver-se na figura 1.3 a interpretação gráfica do método III.

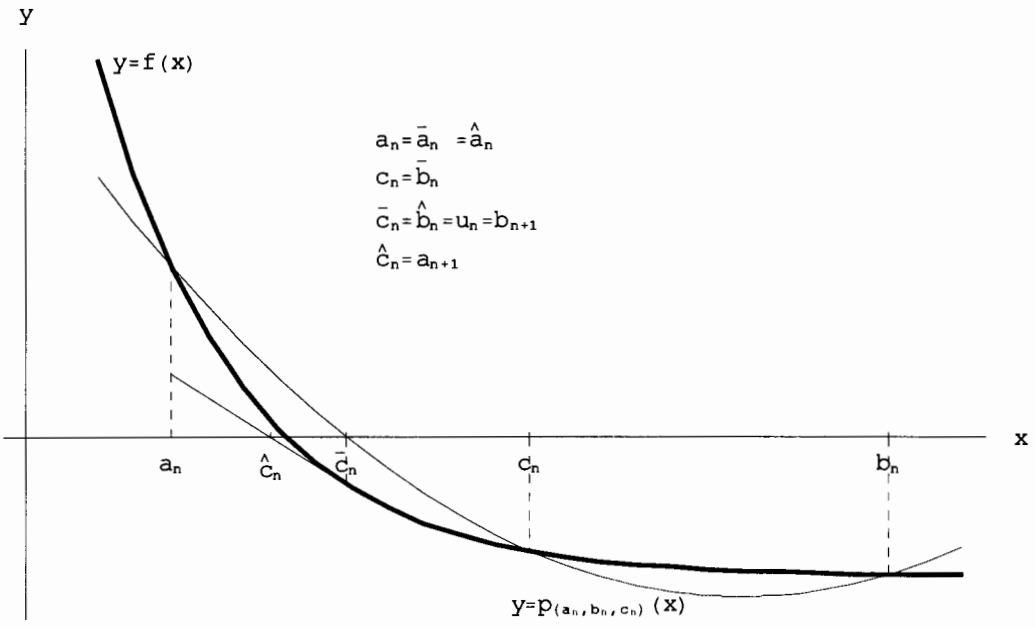

Figura 1.3 Método III

Os métodos que iremos apresentar nas duas secções seguintes podem considerar-se versões melhoradas do método II. São devidos a Alefeld, Potra e Yixun Shi [2] e passarão a ser designados por **método IV** e **método V.**

Método IV

Cada iteração do algoritmo II exige a resolução exacta de uma equação quadrática, equação (1.12), e apenas é escolhida a raiz que pertence ao intervalo $[\overline{a}_n, \overline{b}_n]$. Podemos assim considerar que existe algum desperdício de cálculo.

O **método IV** em vez de determinar exactamente os zeros do polinómio interpolador $p_{(a_n,b_n,c_n)}$ usa duas iterações do método de *Newton* para obter uma aproximação conveniente apenas do zero existente em $[\overline{a}_n, \overline{b}_n]$. Esta modificação, do ponto de vista computacional, evita o cálculo de raízes quadradas e torna mais simples a implementação do algoritmo.

Começaremos por descrever dois procedimentos que permitirão introduzir a alteração apresentada.

Seja novamente $[a,b]$ um intervalo em que a condição $f(a)\,f(b) < 0$ se satisfaz. Considere-se agora o polinómio quadrático $p_{(a,b,d)}$ interpolador da função f nos pontos a, b e d, com $d \notin [a,b]$ e assuma-se que se satisfazem as condições

$$\begin{array}{ll} \text{se } d < a & \text{então } f(d)f(a) > 0\,; \\ \text{se } d > b & \text{então } f(d)f(b) > 0. \end{array} \tag{1.14}$$

Se usarmos a fórmula de interpolação de *Newton*, o polinómio $p_{(a,b,d)}$ fica definido pela expressão

$$\begin{aligned} p_{(a,b,d)}(x) &= f(a) + f[a,b](x-a) + f[a,b,d](x-a)(x-b) \\ &= f(a) + B(x-a) + A(x-a)(x-b), \end{aligned}$$

onde

$$B = f[a,b] = \frac{f(b)-f(a)}{b-a} \quad \text{e} \quad A = f[a,b,d] = \frac{f[b,d]-f[a,b]}{d-a}.$$

Uma vez que $f(a)f(b) < 0$ e sendo $p_{(a,b,d)}(a) = f(a)$ e $p_{(a,b,d)}(b) = f(b)$, existe um único zero r de $p_{(a,b,d)}$ no intervalo $[a,b]$. É imediato verificar que se $A = 0$,

$$r = a - B^{-1}f(a).$$

Se $A \neq 0$, podemos obter uma aproximação para r usando um número k de iterações do método de *Newton*.

O procedimento *newton_quadrático*, que apresentamos na página seguinte, implementa o cálculo de r da forma que acabámos de expor.

Com a finalidade de garantir convergência monótona do método de Newton, a aproximação inicial r_0 é escolhida atendendo ao sinal de $p''_{(a,b,d)}(x) = 2A$ e ao valor de $p_{(a,b,d)}(a) = f(a)$.

newton_quadrático (a, b, d, k)
$A = f[a, b, d]$;
$B = f[a, b]$;
se $A = 0$ **então** $r = a - B^{-1} f(a)$;
senão
se $Af(a) > 0$ **então** $r_0 = a$;
senão $r_0 = b$;
fse
para $i = 1, 2, ..., k$ **fazer**

$$r_i = r_{i-1} - \frac{p_{(a,b,d)}(r_{i-1})}{p'_{(a,b,d)}(r_{i-1})} = r_{i-1} - \frac{p_{(a,b,d)}(r_{i-1})}{B + A(2r_{i-1} - a - b)};$$

fpara
$r = r_k$;
fse
devolver r.

Dado um ponto $c \in [a, b]$, o procedimento *partir_intervalo2* constrói um novo intervalo $[\overline{a}, \overline{b}] \subset [a, b]$ com $f(\overline{a}) f(\overline{b}) < 0$ e, além disso, fornece um ponto $d \notin [\overline{a}, \overline{b}]$ tal que

$$\begin{array}{ll} \text{se } d < \overline{a} & \text{então } f(d) f(\overline{a}) > 0 \ ; \\ \text{se } d > \overline{b} & \text{então } f(d) f(\overline{b}) > 0. \end{array} \tag{1.15}$$

Tal como o procedimento *partir_intervalo1* (veja-se pag. 4), também *partir_intervalo2* incluirá o critério de paragem. Define-se como segue:

partir_intervalo2 $([a, b], c)$
se $f(c) = 0$ **então** escrever c e parar;
se $f(a) f(c) < 0$ **então** $[\overline{a}, \overline{b}] = [a, c]$; $d = b$;
senão $[\overline{a}, \overline{b}] = [c, b]$; $d = a$;
se $\overline{b} - \overline{a} \leq 2\delta$ **então** escrever $[\overline{a}, \overline{b}]$ e parar;
devolver $\{[\overline{a}, \overline{b}], d\}$.

Convém finalmente observar que *partir_intervalo2* fornece um ponto d nas condições exigidas para a aplicação de *newton_quadrático* ao intervalo $[\overline{a}, \overline{b}]$.

Dado $[a_n, b_n]$ uma iteração do método IV começará pelos passos

$$\begin{array}{l} c_n = newton\_quadrático\,(a_n, b_n, d_n, 2)\,; \\ \left\{\left[\overline{a}_n, \overline{b}_n\right], \overline{d}_n\right\} = \; partir\_intervalo2\,([a_n, b_n], c_n)\,, \end{array} \tag{1.16}$$

isto é, com o uso de duas iterações do método de Newton para obter uma aproximação $c_n \in \left[\overline{a}_n, \overline{b}_n\right]$ do zero do polinómio interpolador $p_{(a_n, b_n, c_n)}$ ao que se segue $partir\_intervalo2\,([a_n, b_n], c_n)$.

Em comparação com o método II, exceptuando na primeira iteração, deixa de se aplicar inicialmente a fórmula do método da secante, eliminando-se assim os passos correspondentes a 2.1 e 2.2. Apenas para $n = 1$ se terá

$$\begin{array}{l} c = a - f\,[a, b]^{-1}\, f(a); \\ \{[a_1, b_1], d_1\} = \; partir\_intervalo2\,([a, b], c)\,, \end{array}$$

uma vez que é necessário determinar um primeiro ponto d_1 para se poder usar *newton_quadrático*.

Os passos que se seguem a (1.16) são os correspondentes aos passos 2.5 a 2.7 do método II. Por último teremos

$$\left\{\left[\widehat{a}_n, \widehat{b}_n\right], \widehat{d}_n\right\} = partir\_intervalo2\,\left(\left[\overline{a}_n, \overline{b}_n\right], \widehat{c}_n\right)$$

e

$$\textbf{se } \widehat{b}_n - \widehat{a}_n < \mu\,(b_n - a_n) \tag{1.17}$$

$$\textbf{então } \{[a_{n+1}, b_{n+1}], d_{n+1}\} = \left\{\left[\widehat{a}_n, \widehat{b}_n\right], \widehat{d}_n\right\};$$

$$\textbf{senão } \{[a_{n+1}, b_{n+1}], d_{n+1}\} = partir\_intervalo2\left(\left[\widehat{a}_n, \widehat{b}_n\right], 0.5\left(\widehat{b}_n + \widehat{a}_n\right)\right).$$

A iteração $n+1$ inicia-se com a aplicação de *newton_quadrático* ao intervalo $[a_{n+1}, b_{n+1}]$ com o ponto d_{n+1}.

Também para este quarto método a condição em (1.17) é sempre verdadeira a partir do momento em que o intervalo $[a_n, b_n]$ de inclusão de x^* já é suficientemente pequeno (veja-se secção 2.4.2, teorema 2.4.9). Assim, dado que acontecerá $\{[a_{n+1}, b_{n+1}], d_{n+1}\} = \left\{\left[\widehat{a}_n, \widehat{b}_n\right], \widehat{d}_n\right\}$, o método IV requer, por iteração,

- 3 valores da função f, no pior dos casos, e
- assimptoticamente, 2 valores da função f.

O método IV preserva a garantia de convergência do método II, mantém a propriedade $b_{n+1} - a_{n+1} \leq 0.5\,(b_n - a_n)$ e apresenta sensivelmente o mesmo índice de eficiência (veja-se pag. 72). O algoritmo IV é assim considerado uma melhor versão do algoritmo II uma vez que é mais simples e exige menos uma avaliação da função f por iteração.

Apresentamos de seguida o algoritmo completo do **método IV.**

Algoritmo IV (Alefeld-Potra-Yixun, 1993)

$[a_0, b_0] = [a, b]$;

$c_0 = a_0 - f[a_0, b_0]^{-1} f(a_0)$;

$\{[a_1, b_1], d_1\} = partir\_intervalo2([a_0, b_0], c_0)$;

para $n = 1, 2, \ldots$ **até** $n = nmax$ **fazer**

4.1 $c_n = newton\_quadrático(a_n, b_n, d_n, 2)$;

4.2 $\{[\overline{a}_n, \overline{b}_n], \overline{d}_n\} = partir\_intervalo2([a_n, b_n], c_n)$;

4.3 **se** $|f(\overline{a}_n)| < |f(\overline{b}_n)|$ **então** $u_n = \overline{a}_n$;

senão $u_n = \overline{b}_n$;

4.4 $\overline{c}_n = u_n - 2f[\overline{a}_n, \overline{b}_n]^{-1} f(u_n)$;

4.5 **se** $|\overline{c}_n - u_n| > 0.5(\overline{b}_n - \overline{a}_n)$ **então** $\widehat{c}_n = 0.5(\overline{b}_n + \overline{a}_n)$;

senão $\widehat{c}_n = \overline{c}_n$;

4.6 $\left\{\left[\widehat{a}_n, \widehat{b}_n\right], \widehat{d}_n\right\} = partir\_intervalo2([\overline{a}_n, \overline{b}_n], \widehat{c}_n)$;

4.7 **se** $\widehat{b}_n - \widehat{a}_n < \mu(b_n - a_n)$

então $\{[a_{n+1}, b_{n+1}], d_{n+1}\} = \left\{\left[\widehat{a}_n, \widehat{b}_n\right], \widehat{d}_n\right\}$;

senão $\{[a_{n+1}, b_{n+1}], d_{n+1}\} = partir\_intervalo2\left(\left[\widehat{a}_n, \widehat{b}_n\right], 0.5\left(\widehat{b}_n + \widehat{a}_n\right)\right)$.

fpara

Os passos do algoritmo IV podem ser acompanhados pela figura 1.4 para $n > 1$.

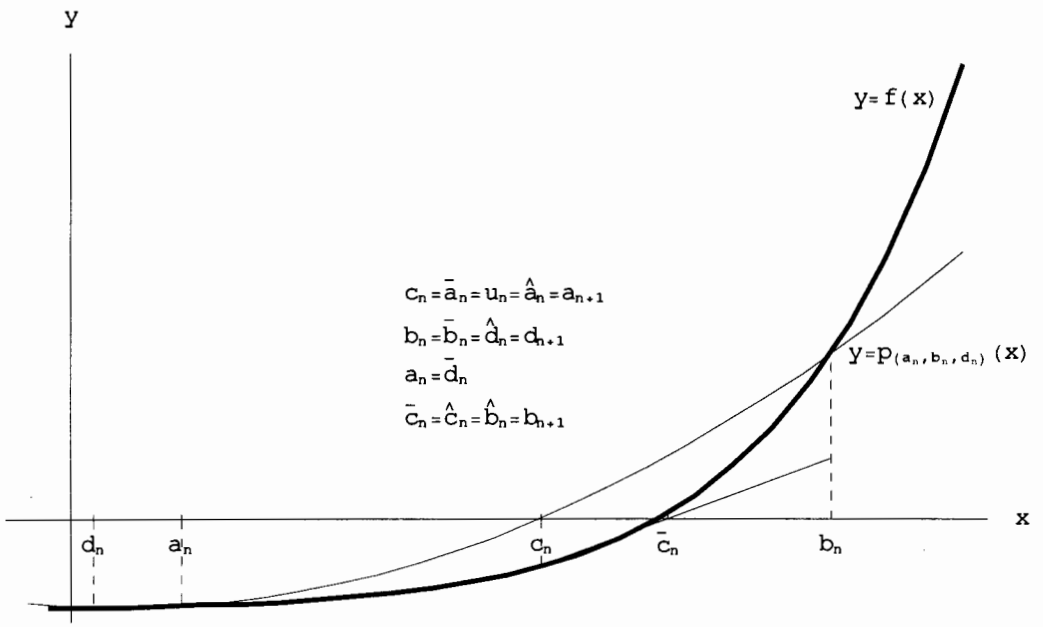

Figura 1.4 Método IV

Método V

O **método V** é construído a partir do método IV com repetição dos passos 4.1 e 4.2, o que significa que em cada iteração se usa interpolação quadrática duas vezes - uma segunda vez logo após a primeira.

Cada iteração do método V começa igualmente por considerar o polinómio quadrático $p_{(a_n,b_n,d_n)}$ interpolador de f nos pontos a_n, b_n e d_n e realiza

$$\begin{array}{l} c_n = newton\_quadrático\,(a_n, b_n, d_n, 2)\,; \\ \left\{\left[\widetilde{a}_n, \widetilde{b}_n\right], \widetilde{d}_n\right\} = \; partir\_intervalo2\,([a_n, b_n], c_n)\,. \end{array} \tag{1.18}$$

Considera-se de seguida o polinómio $p_{(\widetilde{a}_n,\widetilde{b}_n,\widetilde{d}_n)}$ interpolador de f nos pontos $\widetilde{a}_n, \widetilde{b}_n$ e $\widetilde{d}_n$, repetindo-se de novo

$$\begin{array}{l} \widetilde{c}_n = newton\_quadrático\left(\widetilde{a}_n, \widetilde{b}_n, \widetilde{d}_n, 3\right); \\ \left\{\left[\overline{a}_n, \overline{b}_n\right], \overline{d}_n\right\} = \; partir\_intervalo2\left(\left[\widetilde{a}_n, \widetilde{b}_n\right], \widetilde{c}_n\right). \end{array} \tag{1.19}$$

Note-se que em (1.18) se obtém a aproximação $c_n \in [a_n, b_n]$ para o zero do polinómio $p_{(a_n,b_n,d_n)}$ aplicando duas iterações do método de Newton, enquanto que em (1.19) para obter a aproximação $\widetilde{c}_n \in [\widetilde{a}_n, \widetilde{b}_n]$ do zero do polinómio $p_{(\widetilde{a}_n,\widetilde{b}_n,\widetilde{d}_n)}$ se aplicam três iterações.

Após (1.18) e (1.19) prossegue-se da mesma forma que nos passos 4.3 a 4.7 do algoritmo IV.

Veja-se a descrição completa do algoritmo V na página seguinte.

Introduzir esta modificação ao método IV não altera a certeza de convergência e veremos igualmente (secção 2.4.2, teorema 2.4.11) que no passo 5.9 se terá sempre $\{[a_{n+1}, b_{n+1}], d_{n+1}\} = \left\{\left[\widehat{a}_n, \widehat{b}_n\right], \widehat{d}_n\right\}$, para n suficientemente grande.

Embora seja necessário mais uma avaliação da função f em cada iteração, isto é,

- no pior dos casos, 4 avaliações da função f e
- assimptoticamente, 3 avaliações da função f,

veremos (pag. 73) que o índice de eficiência do método V é superior ao do método IV.

Mostraremos também na secção 2.6 que o método V é um *método óptimo* no sentido em que se se fizerem mais repetições dos passos (1.18) não se melhora a eficiência do método.

Segue-se a apresentação do algoritmo V e a figura 1.5 com uma descrição geométrica para $n > 1$.

Algoritmo V (Alefeld-Potra-Yixun, 1993)

$[a_0, b_0] = [a, b]$;

$c_0 = a_0 - f[a_0, b_0]^{-1} f(a_0)$;

$\{[a_1, b_1], d_1\} = partir\_intervalo2([a_0, b_0], c_0)$;

para $n = 1, 2, \ldots$ **até** $n = nmax$ **fazer**

5.1 $c_n = newton\_quadrático(a_n, b_n, d_n, 2)$;

5.2 $\left\{\left[\widetilde{a}_n, \widetilde{b}_n\right], \widetilde{d}_n\right\} = partir\_intervalo2([a_n, b_n], c_n)$;

5.3 $\widetilde{c}_n = newton\_quadrático\left(\widetilde{a}_n, \widetilde{b}_n, \widetilde{d}_n, 3\right)$;

5.4 $\left\{\left[\overline{a}_n, \overline{b}_n\right], \overline{d}_n\right\} = partir\_intervalo2\left(\left[\widetilde{a}_n, \widetilde{b}_n\right], \widetilde{c}_n\right)$;

5.5 **se** $|f(\overline{a}_n)| < |f(\overline{b}_n)|$ **então** $u_n = \overline{a}_n$;
senão $u_n = \overline{b}_n$;

5.6 $\overline{c}_n = u_n - 2f\left[\overline{a}_n, \overline{b}_n\right]^{-1} f(u_n)$;

5.7 **se** $|\overline{c}_n - u_n| > 0.5(\overline{b}_n - \overline{a}_n)$ **então** $\widehat{c}_n = 0.5\left(\overline{b}_n + \overline{a}_n\right)$;
senão $\widehat{c}_n = \overline{c}_n$;

5.8 $\left\{\left[\widehat{a}_n, \widehat{b}_n\right], \widehat{d}_n\right\} = partir\_intervalo2\left(\left[\overline{a}_n, \overline{b}_n\right], \widehat{c}_n\right)$;

5.9 **se** $\widehat{b}_n - \widehat{a}_n < \mu(b_n - a_n)$

então $\{[a_{n+1}, b_{n+1}], d_{n+1}\} = \left\{\left[\widehat{a}_n, \widehat{b}_n\right], \widehat{d}_n\right\}$;

senão $\{[a_{n+1}, b_{n+1}], d_{n+1}\} = partir\_intervalo2\left(\left[\widehat{a}_n, \widehat{b}_n\right], 0.5\left(\widehat{b}_n + \widehat{a}_n\right)\right)$.

fpara

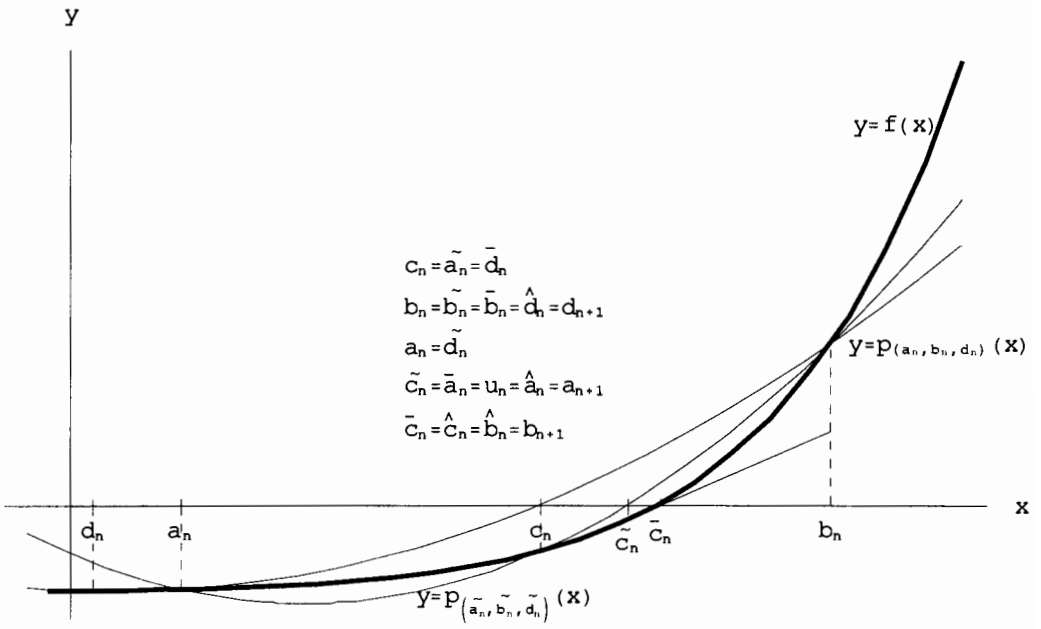

Figura 1.5 Método V

Uma outra ferramenta básica na resolução de equações não lineares é a *interpolação inversa* - considera-se um polinómio interpolador não da função f mas da função inversa f^{-1}. Alefeld, Potra e Yixun Shi [3] sugeriram ainda mais dois métodos, **método VI** e **método VII,** que fazem uso de interpolação cúbica inversa sempre que possível, em vez de interpolação quadrática - alteração que introduzem aos métodos IV e V, respectivamente.

Método VI

Suponhamos que no intervalo $[a, b]$, f satisfaz as condições do teorema da função inversa, isto é, em particular $f'(x) \neq 0$, para $x \in [a, b]$. Assim,

sendo $y = f(x)$ podemos escrever

$$x = f^{-1}(y), \quad x \in [a, b],$$

onde f^{-1} é a função inversa de f. Desta forma, determinar $f^{-1}(0)$ é equivalente a encontra o zero x^* de f em $[a, b]$.

Para estimar $f^{-1}(0)$ o que faremos é aproximar f^{-1} por um polinómio de grau três - interpolação cúbica inversa - e interpolar no ponto $y = 0$, obtendo assim uma aproximação para $x^* = f^{-1}(0)$. Embora possamos usar o processo de interpolação inversa mesmo que $f'(x) = 0$ em $[a, b]$, neste caso a precisão obtida poderá ser muito pobre dado que f^{-1} pode não existir.

Começaremos por descrever um procedimento que implementa esta técnica de interpolação.

Considerem-se então quatro pontos c, d, e e l pertencentes ao intervalo $[a, b]$. Mesmo que f não seja invertível, se $f(c)$, $f(d)$, $f(e)$ e $f(l)$ forem distintos podemos sempre construir o polinómio interpolador cúbico $p_{inverso}$ nos pontos $(f(c), c)$, $(f(d), d)$, $(f(e), e)$ e $(f(l), l)$. Pela fórmula de Newton com diferenças divididas, a expressão deste polinómio será

$$\begin{aligned} p_{inverso}(y) &= c + f^{-1}[f(c), f(d)](y - f(c)) \\ &\quad + f^{-1}[f(c), f(d), f(e)](y - f(c))(y - f(d)) \\ &\quad + f^{-1}[f(c), f(d), f(e), f(l)](y - f(c))(y - f(d))(y - f(e)) \end{aligned} \tag{1.20}$$

onde

$$f^{-1}[f(c), f(d)] = \frac{d - c}{f(d) - f(c)},$$

$$f^{-1}[f(c), f(d), f(e)] = \frac{f^{-1}[f(d), f(e)] - f^{-1}[f(c), f(d)]}{f(e) - f(c)}$$

e

$$f^{-1}[f(c), f(d), f(e), f(l)] = \frac{f^{-1}[f(d), f(e), f(l)] - f^{-1}[f(c), f(d), f(e)]}{f(l) - f(c)}.$$

Assim, poderemos sempre calcular

$$\overline{x} = p_{inverso}(0),$$

que constituirá uma solução aproximada da equação $f(x) = 0$, embora $\overline{x}$ possa não pertencer a $[a, b]$.

Estamos, portanto, interessados no caso em que $f(c)$, $f(d)$, $f(e)$ e $f(l)$ sejam distintos e $\overline{x}$ pertença a $[a, b]$.

Uma vez que temos o objectivo de determinar o valor do polinómio $p_{inverso}$ apenas em $y = 0$, é mais eficiente utilizar um algoritmo de interpolação especificamente indicado para esta tarefa e não para a determinação

explícita da expressão formal do próprio polinómio $p_{inverso}$ dada por (1.20). Um exemplo deste tipo de algoritmos é o algoritmo de *Neville* cuja descrição pode ser vista no apêndice A.

Apresentamos a seguir o procedimento $zero\_p_{inverso}$ que constituiu uma pequena modificação do algoritmo de *Neville* que permite evitar alguns erros de arredondamento desnecessários no cálculo directo de $\overline{x}$ (veja-se apêndice A, pags. 103-105).

$zero\_p_{inverso}(c, d, e, l)$

$$Q_{11} = (e-l)\frac{f(e)}{f(l)-f(e)}; \quad Q_{21} = (d-e)\frac{f(d)}{f(e)-f(d)};$$

$$D_{21} = (d-e)\frac{f(e)}{f(e)-f(d)}; \quad Q_{22} = (D_{21}-Q_{11})\frac{f(d)}{f(l)-f(d)};$$

$$Q_{31} = (c-d)\frac{f(c)}{f(d)-f(c)}; \quad D_{31} = (c-d)\frac{f(d)}{f(d)-f(c)};$$

$$Q_{32} = (D_{31}-Q_{21})\frac{f(c)}{f(e)-f(c)}; \quad D_{32} = (D_{31}-Q_{21})\frac{f(e)}{f(e)-f(c)};$$

$$Q_{33} = (D_{32}-Q_{22})\frac{f(c)}{f(l)-f(c)};$$

$\overline{x} = c + (Q_{31} + Q_{32} + Q_{33})$;

devolver $\overline{x}$.

A ideia básica do **método VI** é a de que se fará uso de interpolação inversa sempre que for possível, aplicando-se o procedimento $zero\_p_{inverso}$ em vez de *newton_quadrático* no método IV.

Considere-se então que numa dada iteração n temos o intervalo $[a_n, b_n]$ de inclusão de x^*. O passo 4.1 do método IV consiste em determinar c_n da forma

$$c_n = newton\_quadrático\,(a_n, b_n, d_n, 2)\,, \tag{1.21}$$

isto é, uma aproximação para o zero do polinómio interpolador $p_{(a_n,b_n,d_n)}$. Em substituição, sempre que haja possibilidade de o fazer, o primeiro passo do método VI efectua o cálculo de c_n da forma

$$c_n = zero\_p_{inverso}(a_n, b_n, d_n, e_n), \tag{1.22}$$

onde e_n é mais um ponto que, tal que como d_n, pertence ao intervalo $[a_{n-1}, b_{n-1}]$ mas não ao intervalo $[a_n, b_n]$. Isto significa que se considera o polinómio interpolador $p_{\{f(a_n),f(b_n),f(d_n),f(e_n)\}}$ *inverso* de f nos pontos $(f(a_n), a_n)$, $(f(b_n), b_n)$, $(f(d_n), d_n)$ e $(f(e_n), e_n)$ e a aproximação

$$c_n = p_{\{f(a_n),f(b_n),f(d_n),f(e_n)\}}(0)\,,$$

obtida com o procedimento $zero\_p_{inverso}$.

Os casos em que não será possível usar c_n calculado por (1.22), isto é, quando

- $f(a_n)$, $f(b_n)$, $f(d_n)$ e $f(e_n)$ não são todos distintos ou
- $c_n \notin [a_n, b_n]$,

resolvem-se mantendo-se a interpolação quadrática com o uso de (1.21). O mesmo se faz na primeira iteração, para $n = 1$, uma vez que ainda não se dispõe de um ponto e_1 para a interpolação cúbica inversa.

Os passos seguintes de cada iteração são os mesmos que os passos 4.2 a 4.6 do método IV. No último passo, a escolha do ponto e_{n+1} parece natural fazer-se da forma

se $\widehat{b}_n - \widehat{a}_n < \mu\,(b_n - a_n)$
então $\{[a_{n+1}, b_{n+1}]\,,\ d_{n+1}\} = \left\{\left[\widehat{a}_n, \widehat{b}_n\right]\,,\ \widehat{d}_n\right\}$;
$e_{n+1} = \overline{d}_n$;
senão $\{[a_{n+1}, b_{n+1}]\,, d_{n+1}\} = \mathit{partir\_intervalo2}\left(\left[\widehat{a}_n, \widehat{b}_n\right], 0.5\left(\widehat{b}_n + \widehat{a}_n\right)\right)$;
$e_{n+1} = \widehat{d}_n$,

existindo assim, após a iteração $n = 1$, quatro pontos a_{n+1}, b_{n+1}, d_{n+1} e e_{n+1} para a aplicação de (1.22) na iteração $n + 1$.

Passamos agora a descrever os pormenores do algoritmo e apresentamos a figura 1.6 com a respectiva ilustração gráfica para $n > 1$.

Algoritmo VI (Alefeld-Potra-Yixun, 1995)

$[a_0, b_0] = [a, b]$;

$c_0 = a_0 - f\,[a_0, b_0]^{-1}\, f(a_0)$;

$\{[a_1, b_1], d_1\} = \mathit{partir\_intervalo2}\,([a_0, b_0], c_0)$;

para $n = 1, 2, \ldots$ **até** $n = nmax$ **fazer**

6.1 **se** $n = 1$ **ou** $f(a_n)$, $f(b_n)$, $f(d_n)$ e $f(e_n)$ não são todos distintos
então $c_n = \mathit{newton\_quadrático}\,(a_n, b_n, d_n, 2)$;
senão $c_n = \mathit{zero\_p_{inverso}}(a_n, b_n, d_n, e_n)$;
se $c_n \notin [a_n, b_n]$ **então** $c_n = \mathit{newton\_quadrático}\,(a_n, b_n, d_n, 2)$;

6.2 $\left\{\left[\overline{a}_n, \overline{b}_n\right], \overline{d}_n\right\} = \mathit{partir\_intervalo2}\,([a_n, b_n], c_n)$;

6.3 **se** $|f(\overline{a}_n)| < \left|f(\overline{b}_n)\right|$ **então** $u_n = \overline{a}_n$;
senão $u_n = \overline{b}_n$;

6.4 $\bar{c}_n = u_n - 2f\left[\bar{a}_n, \bar{b}_n\right]^{-1} f(u_n)$

6.5 **se** $|\bar{c}_n - u_n| > 0.5(\bar{b}_n - \bar{a}_n)$ **então** $\hat{c}_n = 0.5\left(\bar{b}_n + \bar{a}_n\right)$;

senão $\hat{c}_n = \bar{c}_n$;

6.6 $\left\{\left[\hat{a}_n, \hat{b}_n\right], \hat{d}_n\right\} = partir\_intervalo2\left(\left[\bar{a}_n, \bar{b}_n\right], \hat{c}_n\right)$

6.7 **se** $\hat{b}_n - \hat{a}_n < \mu\left(b_n - a_n\right)$

então $\{[a_{n+1}, b_{n+1}], d_{n+1}\} = \left\{\left[\hat{a}_n, \hat{b}_n\right], \hat{d}_n\right\}$;

$e_{n+1} = \bar{d}_n$;

senão$\{[a_{n+1}, b_{n+1}], d_{n+1}\} = partir\_intervalo2\left(\left[\hat{a}_n, \hat{b}_n\right], 0.5\left(\hat{b}_n + \hat{a}_n\right)\right)$;

$e_{n+1} = \hat{d}_n$.

fpara

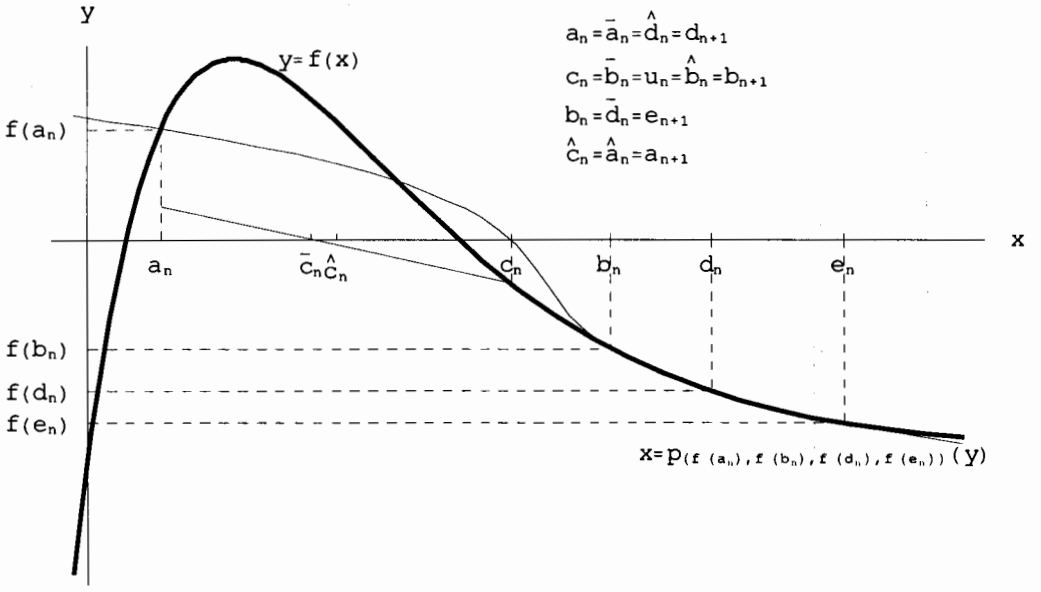

Figura 1.6 Método VI

No caso da função f ser suficientemente suave em $[a,b]$ e $x^* \in [a,b]$ ser um zero simples de f, iremos mostrar que no passo 6.1 a interpolação cúbica inversa será sempre realizada pelo algoritmo VI (veja-se secção 2.4.3, lema 2.4.14). Veremos também que, tal como para todos os métodos apresentados atrás, a condição no passo 6.7,

$$\widehat{b}_n - \widehat{a}_n < \mu\,(b_n - a_n)\,,$$

é igualmente verificada a partir de uma dada iteração n (veja-se secção 2.4.3, teorema 2.4.15), tendo-se sempre

$$\begin{aligned} &\{[a_{n+1}, b_{n+1}]\,, d_{n+1}\} = \left\{\left[\widehat{a}_n, \widehat{b}_n\right], \widehat{d}_n\right\}; \\ &e_{n+1} = \overline{d}_n. \end{aligned}$$

O método VI não altera assim o número de avaliações da função f exigidos por iteração, comparativamente com o método IV.

Embora não haja melhoria quanto à exigência no cálculo de valores da função f, veremos (pag. 73) que o método VI apresenta um índice de eficiência superior ao método IV.

Método VII

Este último método, **método VII**, obtém-se do método VI de forma comparável à que permitiu construir o método V começando com o método IV, isto é, introduzindo a repetição de dois passos - os passos 6.1 e 6.2.

Assim, em cada iteração do método VII, dado o intervalo $[a_n, b_n]$ e determinado c_n como em 6.1 do método VI, realiza-se

$$\begin{aligned} &\widetilde{e}_n = d_n; \\ &\left\{[\widetilde{a}_n, \widetilde{b}_n], \widetilde{d}_n\right\} = \mathit{partir\_intervalo2}\,([a_n, b_n], c_n)\,, \end{aligned} \tag{1.23}$$

de forma a poder repetir a interpolação cúbica inversa nos pontos $(f\,(\widetilde{a}_n)\,, \widetilde{a}_n)$, $\left(f\left(\widetilde{b}_n\right), \widetilde{b}_n\right)$, $\left(f\left(\widetilde{d}_n\right), \widetilde{d}_n\right)$ e $(f\,(\widetilde{e}_n)\,, \widetilde{e}_n)$, com a determinação de $\widetilde{c}_n$ por

$$\widetilde{c}_n = \mathit{zero\_p_{inverso}}\left(\widetilde{a}_n, \widetilde{b}_n, \widetilde{d}_n, \widetilde{e}_n\right). \tag{1.24}$$

Novamente, caso isto não seja possível, usa-se a interpolação quadrática calculando

$$\widetilde{c}_n = \mathit{newton\_quadrático}\left(\widetilde{a}_n, \widetilde{b}_n, \widetilde{d}_n, 3\right).$$

Segue-se

$$\left\{\left[\overline{a}_n, \overline{b}_n\right], \overline{d}_n\right\} = \mathit{partir\_intervalo2}\left(\left[\widetilde{a}_n, \widetilde{b}_n\right], \widetilde{c}_n\right),$$

de forma semelhante a 6.2 do método VI. E prossegue-se de maneira inteiramente idêntica aos passos 6.3 a 6.7 do método VI.

Embora exigindo mais um valor de f por iteração, veremos que esta alteração, em termos de eficiência, produz uma ligeira melhoria em relação método VI . De facto, o método VII é o método que revela um índice de eficiência mais elevado (veja-se pag. 73). Mostraremos também que se trata de um *método óptimo* dentro de uma certa classe de métodos - métodos que implementam a repetição dos passos descritos em (1.23) e (1.24) um número arbitrário k de vezes (veja-se secção 2.6).

Apresentamos de seguida o algoritmo do método VII.

Algoritmo VII (Alefeld-Potra-Yixun, 1995)

$[a_0, b_0] = [a, b]$;

$c_0 = a_0 - f[a_0, b_0]^{-1} f(a_0)$;

$\{[a_1, b_1], d_1\} = partir\_intervalo2([a_0, b_0], c_0)$;

para $n = 1, 2, \ldots$ **até** $n = nmax$ **fazer**

7.1 **se** $n = 1$ **ou** $f(a_n)$, $f(b_n)$, $f(d_n)$ e $f(e_n)$ não são todos distintos

então $c_n = newton\_quadrático(a_n, b_n, d_n, 2)$;

senão $c_n = zero\_p_{inverso}(a_n, b_n, d_n, e_n)$;

se $c_n \notin [a_n, b_n]$ **então** $c_n = newton\_quadrático(a_n, b_n, d_n, 2)$;

7.2 $\widetilde{e}_n = d_n$;

$\left\{[\widetilde{a}_n, \widetilde{b}_n], \widetilde{d}_n\right\} = partir\_intervalo2([a_n, b_n], c_n)$;

7.3 **se** $f(\widetilde{a}_n), f\left(\widetilde{b}_n\right), f\left(\widetilde{d}_n\right)$ e $f(\widetilde{e}_n)$ não são todos distintos

então $\widetilde{c}_n = newton\_quadrático\left(\widetilde{a}_n, \widetilde{b}_n, \widetilde{d}_n, 3\right)$;

senão $\widetilde{c}_n = zero\_p_{inverso}\left(\widetilde{a}_n, \widetilde{b}_n, \widetilde{d}_n, \widetilde{e}_n\right)$;

se $\widetilde{c}_n \notin \left[\widetilde{a}_n, \widetilde{b}_n\right]$ **então** $\widetilde{c}_n = newton\_quadrático\left(\widetilde{a}_n, \widetilde{b}_n, \widetilde{d}_n, 3\right)$;

7.4 $\left\{\left[\overline{a}_n, \overline{b}_n\right], \overline{d}_n\right\} = partir\_intervalo2\left(\left[\widetilde{a}_n, \widetilde{b}_n\right], \widetilde{c}_n\right)$;

7.5 **se** $|f(\overline{a}_n)| < |f(\overline{b}_n)|$ **então** $u_n = \overline{a}_n$;

senão $u_n = \overline{b}_n$;

7.6 $\overline{c}_n = u_n - 2f\left[\overline{a}_n, \overline{b}_n\right]^{-1} f(u_n)$;

7.7 **se** $|\overline{c}_n - u_n| > 0.5(\overline{b}_n - \overline{a}_n)$ **então** $\widehat{c}_n = 0.5\left(\overline{b}_n + \overline{a}_n\right)$;

senão $\widehat{c}_n = \overline{c}_n$;

7.8 $\left\{\left[\widehat{a}_n, \widehat{b}_n\right], \widehat{d}_n\right\} = partir\_intervalo2\left(\left[\overline{a}_n, \overline{b}_n\right], \widehat{c}_n\right)$;

7.9 **se** $\widehat{b}_n - \widehat{a}_n < \mu\left(b_n - a_n\right)$

então $\{[a_{n+1}, b_{n+1}], d_{n+1}\} = \left\{\left[\widehat{a}_n, \widehat{b}_n\right], \widehat{d}_n\right\}$;

$e_{n+1} = \overline{d}_n$;

senão $\{[a_{n+1}, b_{n+1}], d_{n+1}\} = partir\_intervalo2\left(\left[\widehat{a}_n, \widehat{b}_n\right], 0.5\left(\widehat{b}_n + \widehat{a}_n\right)\right)$;

$e_{n+1} = \widehat{d}_n$.

fpara

Uma interpretação geométrica do método VII é ilustrada na figura 1.7.

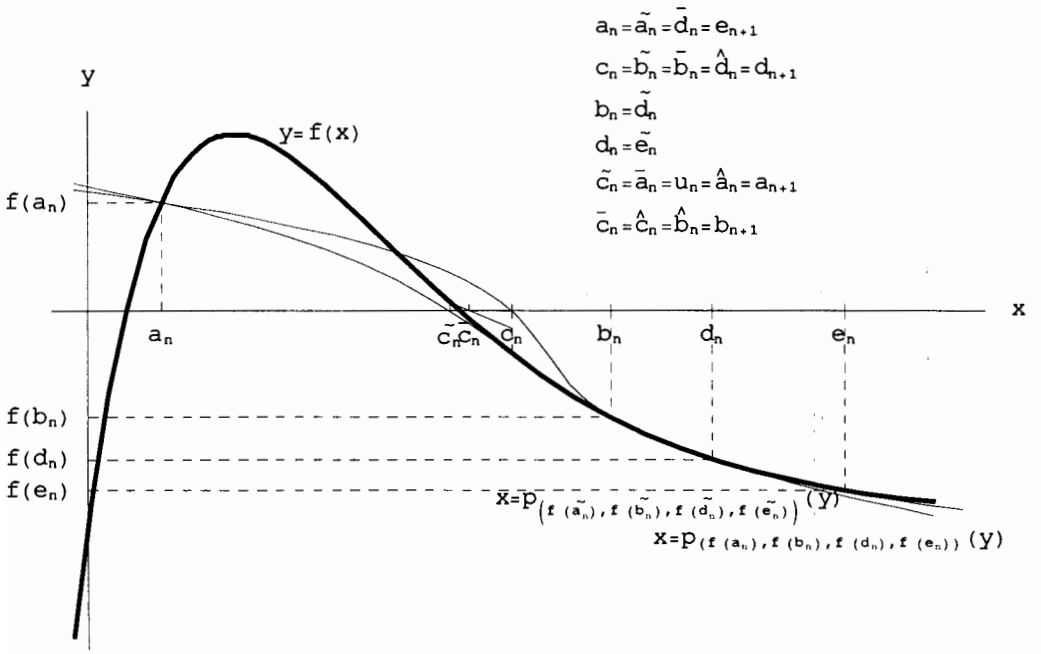

Figura 1.7 Método VII

1.2.2 Implementação do critério de paragem

Tal como é usado em Brent [7], o critério de paragem escolhido combina uma tolerância para o erro relativo (4ε) e uma tolerância para o erro absoluto ($2tol$). Em cada chamada a *partir_intervalo*1 (ou *partir_intervalo*2) considera-se o valor

$$\delta = \delta\left(\overline{a}, \overline{b}\right) = 2\varepsilon\,|u| + tol,$$

onde $u \in \left\{\overline{a}, \overline{b}\right\}$ é tal que $|f(u)| = \min\left\{\left|f(\overline{a})\right|, \left|f(\overline{b})\right|\right\}$ (melhor aproximação para x^*), ε é a *precisão relativa da máquina* (β^{1-t} ou $\frac{1}{2}\beta^{1-t}$ para uma aritmética de vírgula flutuante na base β com t dígitos, de corte ou de arredondamento, respectivamente) e tol é um parâmetro não negativo que constitui a tolerância absoluta desejada. Termina-se com o intervalo $\left[\overline{a}, \overline{b}\right]$ quando

$$\overline{b} - \overline{a} \leq 2\delta\left(\overline{a}, \overline{b}\right). \tag{1.25}$$

Esta escolha de δ previne contra as consequências de exigir valores irrealistas para a tolerância tol face à precisão da máquina usada. Em particular poderemos fazer $tol = 0$ para forçar a determinação do mais pequeno intervalo de inclusão possível.

O valor 2ε pode ser aumentado se se pretender aceitar um erro relativo mais elevado, mas não se deve tornar inferior dada a influência dos erros de arredondamento.

De acordo com este critério de paragem, na implementação do procedimento *partir_intervalo*1 (ou *partir_intervalo*2) é natural introduzir algumas modificações. A sugestão apresentada pelos autores dos algoritmos é a seguinte:

*partir_intervalo*1 $([a,b], c)$ (ou *partir_intervalo*2 $([a,b], c)$)
$\alpha = \lambda\delta\,(a,b)$;
se $b - a \leq 4\alpha$ **então** $c = \frac{a+b}{2}$;
senão
se $c \leq a + 2\alpha$ **então** $c = a + 2\alpha$;
senão
se $c \geq b - 2\alpha$ **então** $c = b - 2\alpha$;
se $f(c) = 0$ **então** escrever c e parar;
se $f(a)f(c) < 0$ **então** $\overline{a} = a$, $\overline{b} = c, (d = b)$;
senão $\overline{a} = c$, $\overline{b} = b, (d = a)$;
se $\overline{b} - \overline{a} \leq 2\delta\left(\overline{a}, \overline{b}\right)$ **então** escrever $\left[\overline{a}, \overline{b}\right]$ e parar;
devolver $\left[\overline{a}, \overline{b}\right]$.

Neste procedimento λ é um parâmetro que deve ser escolhido de forma a satisfazer $0 < \lambda < 1$. Nas suas experiências os autores referidos utilizam

$\lambda = 0.7$. Assim, facilmente se compreende esta modificação:

- no caso em que $b-a \leq 4\lambda\delta(a,b) < 4\delta(a,b)$, ao considerarmos $c = \frac{a+b}{2}$ fica praticamente garantido que de seguida o critério de paragem se satisfaz, quer seja para o intervalo $[\overline{a},\overline{b}] = [c,b]$ ou para $[\overline{a},\overline{b}] = [a,c]$;

- quando acontece $c - a \leq 2\lambda\delta(a,b) < 2\delta(a,b)$, ao alterarmos c para $c = a + 2\lambda\delta(a,b)$ continuamos a garantir a precisão desejada no caso de $x^* \in [a,c] = [\overline{a},\overline{b}]$;

- o caso em que $c \geq b - 2\lambda\delta(a,b)$ é semelhante ao anterior, agora para o subintervalo $[c,b]$.

Quando se está já perto de verificar o critério de paragem (1.25), esta alteração dos procedimentos *partir_intervalo*1 e *partir_intervalo*2 permite assim que não sejam realizadas iterações desnecessárias.

1.2.3 Efeito dos erros de arredondamento

Quando um método iterativo para a resolução da equação $f(x) = 0$ converge, a única limitação inerente para a precisão da aproximação à raíz x^* é o número de dígitos usados na computação, isto é, a presença de erros de arredondamento.

Após um número k de iterações, quando o critério de paragem (1.25) se verifica, os algoritmos apresentados terminam com um intervalo $[a_k,b_k]$ de inclusão de x^*. Podemos considerar $\widehat{x} = \dfrac{a_k+b_k}{2}$ uma aproximação para x^* com erro

$$|x^* - \widehat{x}| \leq \delta(a_k,b_k).$$

No entanto, se tivermos em conta os erros de arredondamento, este limite para $|x^* - \widehat{x}|$ deve ser ligeiramente aumentado.

Suponhamos que num sistema de numeração de vírgula flutuante, para os números x e y, as operações aritméticas satisfazem

$$fl(x \times y) = xy(1+\varepsilon_1) \tag{1.26}$$

e

$$fl(x \pm y) = x(1+\varepsilon_2) \pm y(1+\varepsilon_3), \tag{1.27}$$

onde $fl(x)$ representa a aproximação para x no sistema considerado, sendo $|\varepsilon_i| \leq \varepsilon$, $i = 1,2,3$ com ε = unidade de erro de arredondamento ou de corte (*precisão relativa do sistema*), (veja-se por exemplo Wilkinson [30]).

Os algoritmos calculam aproximações

$$\widetilde{m} = fl\left(\overline{b} - \overline{a}\right)$$

e

$$2\widetilde{\delta}\left(\overline{a},\overline{b}\right) = fl\left(4\varepsilon|u| + 2tol\right),$$

terminando apenas quando

$$\widetilde{m} \leq 2\widetilde{\delta}\left(\overline{a}, \overline{b}\right) \tag{1.28}$$

(a menos que se tenha encontrado o zero x^* ao fim de um número finito de iterações, caso em que $\widehat{x} = x^*$).

Assumir(1.26) e (1.27) permite-nos obter

$$\begin{aligned}
\widetilde{m} &= \overline{b}(1+\varepsilon_1) - \overline{a}(1+\varepsilon_2), \qquad |\varepsilon_i| \leq \varepsilon,\ i = 1,2 \\
&= \left(\overline{b} - \overline{a}\right) - \left(\overline{a}\varepsilon_2 - \overline{b}\varepsilon_1\right) \\
&\geq \left(\overline{b} - \overline{a}\right) - \left|\overline{a}\varepsilon_2 - \overline{b}\varepsilon_1\right| \\
&\geq \left(\overline{b} - \overline{a}\right) - \left(\left|\overline{a}\varepsilon_2\right| + \left|\overline{b}\varepsilon_1\right|\right) \\
&\geq \left(\overline{b} - \overline{a}\right) - \varepsilon\left(|\overline{a}| + |\overline{b}|\right).
\end{aligned}$$

Como $0 < 1 - \varepsilon < 1$, podemos também escrever

$$\widetilde{m} \geq \left(\left(\overline{b} - \overline{a}\right) - \varepsilon\left(|\overline{a}| + |\overline{b}|\right)\right)(1-\varepsilon).$$

De forma análoga, obtém-se

$$\begin{aligned}
2\widetilde{\delta}\left(\overline{a}, \overline{b}\right) &= fl\left(4\varepsilon\,|u|\,(1+\varepsilon_3)(1+\varepsilon_4) + 2tol\,(1+\varepsilon_5)\right), \\
&= 4\varepsilon\,|u|\,(1+\varepsilon_3)(1+\varepsilon_4)(1+\varepsilon_6) + 2tol\,(1+\varepsilon_5)(1+\varepsilon_7),
\end{aligned}$$

com $|\varepsilon_i| \leq \varepsilon$, $i = 3,4,5,6,7$, donde

$$2\widetilde{\delta}\left(\overline{a}, \overline{b}\right) \leq \left(4\varepsilon\,|u| + 2tol\right)(1+\varepsilon)^3.$$

Assim, estes dois últimos resultados e (1.28) implicam que

$$\left(\left(\overline{b} - \overline{a}\right) - \varepsilon\left(|\overline{a}| + |\overline{b}|\right)\right)(1-\varepsilon) \leq \left(4\varepsilon\,|u| + 2tol\right)(1+\varepsilon)^3,$$

donde

$$\left(\overline{b} - \overline{a}\right) \leq \frac{1}{(1-\varepsilon)}\left(4\varepsilon\,|u| + 2tol\right)(1+\varepsilon)^3 + \varepsilon\left(|\overline{a}| + |\overline{b}|\right).$$

Escolhendo $\widehat{x} = \dfrac{\overline{a} + \overline{b}}{2}$ como aproximação para x^*, vem

$$|x^* - \widehat{x}| \leq \frac{1}{2}\left(\overline{b} - \overline{a}\right)$$

e, portanto,

$$|x^* - \widehat{x}| \leq \frac{1}{(1-\varepsilon)}\left(2\varepsilon\,|u| + tol\right)(1+\varepsilon)^3 + \frac{\varepsilon}{2}\left(|\overline{a}| + |\overline{b}|\right).$$

Uma vez que $\frac{(1+\varepsilon)^3}{1-\varepsilon} \leq 1+5\varepsilon$ para $0 < \varepsilon \leq 0.1$, podemos escrever

$$|x^* - \widehat{x}| \leq (2\varepsilon\,|u| + tol) + \frac{\varepsilon}{2}\left(|\overline{a}| + |\overline{b}|\right),$$

desprezando os termos de ordem εtol e $\varepsilon^2\,|u|$. Ao considerarmos $|\widehat{x}| \approx |\overline{a}| \approx |\overline{b}|$, poderemos escrever

$$|x^* - \widehat{x}| \approx 3\varepsilon\,|\widehat{x}| + tol. \tag{1.29}$$

É claro que um limite mais rigoroso para o erro $|x^* - \widehat{x}|$ deveria também incluir o efeito da propagação dos erros de arredondamento noutros cálculos, nomeadamente no cálculo da função f. Sendo assim, admitindo que os algoritmos descritos são implementados de maneira a não tornar significativa a acumulação de erros de arredondamento, o limite em (1.29) apenas garante que se encontrou um zero $\widehat{x}$ da *função computável* f com uma precisão estimada em $3\varepsilon\,|\widehat{x}| + tol$.

Capítulo 2

Convergência dos métodos I a VII

2.1 Introdução

Embora haja vários métodos de inclusão para resolver a equação

$$f(x) = 0, \tag{2.1}$$

onde f é uma função contínua num dado intervalo $[a, b]$ e com um zero simples x^* em $[a, b] = [a_0, b_0]$, na maior parte destes métodos a sucessão $\{(b_n - a_n)\}_{n=0}^{\infty}$ das amplitudes dos sucessivos intervalos de inclusão $[a_n, b_n]$ $(n = 0, 1, \ldots)$ não apresenta propriedades de convergência atractivas. É o caso do método de Dekker em que as amplitudes $b_n - a_n$ podem permanecer maiores do que uma certa quantidade positiva relativamente grande até à última iteração (veja-se um exemplo em [7, pag. 49]).

Além disso, com o método de Dekker, assim como com outros, tais como o método de Brent, o método da falsa posição ou o método de Illinois[1], apenas foi estudada a *ordem de convergência* da sucessão $\{|x_n - x^*|\}_{n=0}^{\infty}$, onde x_n representa a aproximação para x^* obtida na iteração n, e não a ordem de convergência da sucessão $\{(b_n - a_n)\}_{n=0}^{\infty}$.

Contudo, determinar a ordem de convergência da sucessão $\{(b_n - a_n)\}_{n=0}^{\infty}$ é muito importante uma vez que na implementação da maioria dos algoritmos de resolução de equações não lineares o critério de paragem é definido em termos da amplitude $b_n - a_n$ do intervalo de inclusão $[a_n, b_n]$.

Neste capítulo iremos estudar as propriedades de convergência da sucessão $\{(b_n - a_n)\}_{n=0}^{\infty}$ gerada pelos métodos I a VII, métodos descritos no capítulo anterior. Foi já anteriormente observado que para todos estes métodos

[1] Um estudo detalhado sobre as propriedades de convergência do método da falsa posição e de algumas variantes deste, como por exempo o método de Illinois, pode ser visto em [23, pags. 338-344].

acontece sempre

$$\lim_{n\to\infty}(b_n - a_n) = 0.$$

Começaremos por apresentar os conceitos de *Q-ordem* e de *R-ordem* de convergência de uma sucessão, adoptando definições análogas às usadas em Potra [15]. Extraídos de [24], seguir-se-ão resultados importantes para o estudo da *R*-ordem de convergência nos métodos III a VII.

Alefeld e Potra [1] mostram que a sucessão $\{(b_n - a_n)\}_{n=0}^{\infty}$ converge para zero pelo menos *Q-quadraticamente,* no método I, com *Q*-ordem de convergência pelo menos 4, no método II, e com *R*-ordem de convergência pelo menos igual a $0.5\left(3+\sqrt{13}\right) = 3.3027\ldots$, no método III.

Para os métodos IV e V, Alefeld, Potra e Yixun Shi [2] provam que $\{(b_n - a_n)\}_{n=0}^{\infty}$ converge para zero com *R*-ordem de convergência pelo menos igual a $1+\sqrt{2} = 2.4142\ldots$, no método IV, e pelo menos igual a $2+\sqrt{5} = 4.2360\ldots$, no método V.

Por último, em relação aos métodos VI e VII, prova-se em Alefeld, Potra e Yixun Shi [3], que a sucessão $\{(b_n - a_n)\}_{n=0}^{\infty}$ converge para zero com *R*-ordem de convergência pelo menos igual a $1+\sqrt{3} = 2.7320\ldots$, no método VI, e pelo menos igual a $2+\sqrt{7} = 4.6457\ldots$, no método VII.

Num método iterativo, além do estudo da rapidez de convergência para a solução, é fundamental considerar também o esforço de cálculo envolvido em cada iteração. Veremos que faz sentido definir *índice de eficiência computacional* de um método iterativo por

$$IndEfic = \sqrt[p]{q}, \tag{2.2}$$

onde q é a *Q*-ordem ou *R*-ordem de convergência do método e p é o número de avaliações da função f (ou das suas derivadas) necessário assimptoticamente em cada iteração.

Usando a definição em (2.2), os correspondentes índices de eficiência dos métodos I a VII serão respectivamente: $\sqrt{2} = 1.4142\ldots$, $\sqrt[3]{4} = 1.5874\ldots$, $\sqrt[3]{0.5\left(3+\sqrt{13}\right)} = 1.4892\ldots$, $\sqrt{1+\sqrt{2}} = 1.5537\ldots$, $\sqrt[3]{2+\sqrt{5}} = 1.6180\ldots$, $\sqrt{1+\sqrt{3}} = 1.6528\ldots$ e $\sqrt[3]{2+\sqrt{7}} = 1.6685\ldots$.

Por último mostraremos que os índices de eficiência dos métodos V e VII são óptimos dentro de uma certa classe de métodos.

2.2 Conceitos básicos

Um conceito fundamental que permite caracterizar a rapidez de convergência de um método iterativo é a sua *ordem de convergência.*

Existem várias noções de *ordem de convergência* de um método e nem sempre são equivalentes. Apresentaremos de seguida as definições de *Q-ordem* e de *R-ordem* de convergência que adoptamos e que serão as utilizadas no estudo sobre a convergência dos métodos I a VII.

Considere-se um espaço métrico (X, d) com $X \subseteq \mathbb{R}$. Dizemos que uma sucessão $\{x_n\}_{n=0}^{\infty}$ de pontos de X converge para um ponto $\zeta \in X$ se

$$\lim_{n\to\infty} d(x_n, \zeta) = 0.$$

As propriedades de convergência da sucessão $\{x_n\}_{n=0}^{\infty}$ irão depender da distância d em X. Mas para uma dada distância d, a rapidez de convergência da sucessão $\{x_n\}_{n=0}^{\infty}$ para ζ é caracterizada pela rapidez de convergência para zero da sucessão de números não negativos $d(x_n, \zeta)$.

Nas definições que se seguem consideraremos sucessões $\{\varepsilon_n\}_{n=0}^{\infty}$ de números não negativos convergindo para zero e que têm ordens de convergência maiores ou iguais a um.

A mais importante medida para caracterizar a rapidez de convergência de sucessões obtidas através de um processo iterativo é a *Q-ordem* de convergência.

Definição 2.2.1 *Diz-se que a sucessão $\{\varepsilon_n\}_{n=0}^{\infty}$ converge com Q-ordem pelo menos $\tau \geq 1$ se existir uma constante positiva b tal que*

$$\varepsilon_{n+1} \leq b\varepsilon_n^{\tau}, \qquad n = 0, 1, \ldots.$$

Se $\tau = 1$ deve ter-se $b < 1$.

A noção de *R-ordem* de convergência representa também uma noção alternativa importante para medir a rapidez de convergência de um método iterativo.

Definição 2.2.2 *Diz-se que a sucessão $\{\varepsilon_n\}_{n=0}^{\infty}$ converge com R-ordem pelo menos $\tau > 1$ se existirem constantes $b > 0$ e $\theta \in (0, 1)$ tais que*

$$\varepsilon_n \leq b\theta^{\tau^n}, \qquad n = 0, 1, \ldots.$$

É imediato notar que para $\tau = 1$ esta definição não revelaria interesse.

Facilmente se observa que a *R*-ordem de convergência constitui um conceito menos preciso que a *Q*-ordem de convergência.

Consideremos, por exemplo, que para um dado método iterativo de resolução de (2.1) se obtém a sucessão $\{\varepsilon_n\}_{n=0}^{\infty} = d(x_n, x^*)$, onde x^* representa a solução exacta da equação. Suponhamos que $\{\varepsilon_n\}_{n=0}^{\infty}$ converge para zero com *Q*-ordem pelo menos τ, sendo a distância $d(x_n, x^*)$ definida por

$$d(x_n, x^*) = |x_n - x^*|.$$

Poderemos assim avaliar quanto ao ganho, em termos de precisão, para a aproximação $x_{n+1} \approx x^*$ relativamente à aproximação anterior $x_n \approx x^*$.

Pela definição, podemos afirmar que o erro absoluto $\varepsilon_{n+1} = |x_{n+1} - x^*|$ é aproximadamente igual a $b\varepsilon_n^\tau = b\,|x_n - x^*|^\tau$. Isto equivale a dizer que

$$\lim_{n\to\infty} \frac{\varepsilon_{n+1}}{\varepsilon_n^\tau} \simeq b,$$

ou, de forma assimptótica,

$$\varepsilon_{n+1} \sim b\varepsilon_n^\tau.$$

Assim, cada iteração aproximadamente eleva a τ o erro absoluto ε_n, o que implica que o número de dígitos correctos que se garantem para x_{n+1} é aproximadamente multiplicado por τ em cada iteração. Se $\tau = 1$, a redução do erro ε_{n+1} é linear e tanto maior quanto menor for a constante b.

Com informação apenas sobre a R-ordem de convergência do método não se pode tirar o mesmo tipo de conclusões. Se a R-ordem de convergência da sucessão $\{\varepsilon_n\}_{n=0}^\infty$ é pelo menos τ, significa que $\{\varepsilon_n\}_{n=0}^\infty$ é comparada com uma sucessão que converge para zero com Q-ordem pelo menos τ. De facto, notando que $\theta^{\tau^{n+1}} = \left(\theta^{\tau^n}\right)^\tau$, a sucessão $\left\{\theta^{\tau^n}\right\}_{n=0}^\infty$ converge para zero com Q-ordem pelo menos τ, e a noção de R-ordem de convergência apenas garante que na iteração n o erro absoluto ε_n é não superior a $b\theta^{\tau^n}$. Assim, não é possível avaliar de forma tão rigorosa o decrescimento do erro em cada iteração.

No entanto, podemos dizer que para ambos os conceitos quanto maior τ, mais rápida a convergência.

A seguir serão apresentados resultados relativos à R-ordem de convergência de sucessões relacionadas entre si através de um determinado sistema de desigualdades.

2.3 R-Ordem de convergência de sucessões relacionadas entre si

Consideremos que para um dado método iterativo um elemento x^* é aproximado por um conjunto de s sucessões

$$\{x_{n,1}\},\ \{x_{n,2}\},\ldots,\{x_{n,s}\}.$$

Suponhamos que um sistema de desigualdades da forma

$$|x_{n+1,i} - x^*| \le \alpha_i \prod_{j=1}^{s} |x_{n,j} - x^*|^{m_{ij}}, \qquad i = 1,\ldots,s,$$

onde $m_{ij} \ge 0$ se verifica para as s sucessões referidas.

Nesta secção vamos mostrar que sob certas condições a R-ordem de convergência das sucessões $\{x_{n,1}\},\{x_{n,2}\},\ldots,\{x_{n,s}\}$ é pelo menos igual ao raio espectral $\rho(M)$ da matriz $M = (m_{ij})$, $(i,j = 1,\ldots,s)$. Recorde-se que

$$\rho(M) = \max\{|\lambda| : \lambda \text{ é valor próprio da matriz } M\}.$$

Apresentaremos também resultados relativos à R-ordem de convergência derivados das desigualdades

$$|x_{n+1} - x^*| \leq \alpha \prod_{j=0}^{s} |x_{n-j} - x^*|^{m_j},$$

que surgem como caso particular para o conjunto das $s+1$ sucessões

$$\{x_n\},\ \{x_{n-1}\}, \ldots, \{x_{n-s}\}.$$

No que se segue iremos considerar as sucessões $\{\varepsilon_{n,i}\}$ definidas por

$$\varepsilon_{n,i} = |x_{n,i} - x^*|, \qquad i = 1, \ldots, s,$$

cuja rapidez de convergência para zero caracteriza a rapidez de convergência para x^* das sucessões $\{x_{n,i}\}$ $(i = 1, \ldots, s)$.

Teorema 2.3.1 (**Schmidt** [24]) *Sejam* $\{\varepsilon_{n,1}\}, \{\varepsilon_{n,2}\}, \ldots, \{\varepsilon_{n,s}\}$ *sucessões que convergem para zero satisfazendo o sistema de desigualdades*

$$0 \leq \varepsilon_{n+1,i} \leq \alpha_i \prod_{j=1}^{s} \varepsilon_{n,j}^{m_{ij}}, \qquad i = 1, \ldots, s \quad e \quad n \geq n_0, \tag{2.3}$$

onde $\alpha_i > 0$ *e* $m_{ij} \geq 0$ *para* $i, j = 1, \ldots, s$.
Assuma-se que a matriz $M = (m_{ij})$ *tem um valor próprio* $\tau > 1$ *com um vector próprio* $\mathbf{u} > \mathbf{0}$, *isto é,* $u_1 > 0, u_2 > 0, \ldots, u_s > 0$ *para* $\mathbf{u} = (u_1, \ldots, u_s)^T$.
Então existem constantes $\theta \in (0,1)$ *e* $\gamma_1 > 0, \gamma_2 > 0, \ldots, \gamma_s > 0$ *e existe* $n_1 \geq n_0$ *tais que*

$$\varepsilon_{n,i} \leq \gamma_i \theta^{u_i \tau^n} \text{ para } n \geq n_1 \geq n_0 \text{ e } i = 1, 2, \ldots, s. \tag{2.4}$$

Antes de prosseguirmos com a demonstração deste teorema é importante ter em consideração as seguintes notas.

Nota 2.3.1 *Pela definição 2.2.2 de R-ordem de convergência,* (2.4) *assegura que as sucessões* $\{\varepsilon_{n,1}\}, \{\varepsilon_{n,2}\}, \ldots, \{\varepsilon_{n,s}\}$ *convergem com R-ordem pelo menos* τ. *De facto, como* $\theta \in (0,1)$ *e* $u_i > 0$, *tem-se* $0 < \sigma_i = \theta^{u_i} < 1$ $(i = 1, 2, \ldots, s)$ *e escolhendo* $b_i \geq \max\left\{\gamma_i, \frac{\varepsilon_{n,i}}{\sigma_i^{\tau^n}};\ n = 0, 1, \ldots, n_1 - 1\right\}$ *vem*

$$\varepsilon_{n,i} \leq b_i \sigma_i^{\tau^n} \text{ para } n = 0, 1, \ldots \quad e \quad i = 1, 2, \ldots, s.$$

Nota 2.3.2 *Sendo* M *uma matriz não negativa, isto é,* $m_{ij} \geq 0$ $(i, j = 1, 2, \ldots, s)$, *o raio espectral* $\rho(M)$ *é um valor próprio de* M *(veja-se apêndice B, teorema B.1.1). Se* $\rho(M) > 1$ *e se* $\mathbf{u} > \mathbf{0}$, *para um correspondente vector próprio* $\mathbf{u}$, *então este teorema pode ser aplicado.*

Nota 2.3.3 *Nas condições indicadas, isto é, sendo M uma matriz não negativa e $\mathbf{u} > \mathbf{0}$,*

$$M\mathbf{u} = \tau\mathbf{u},\ \mathbf{u} > \mathbf{0} \text{ implica que } \tau = \rho(M),$$

(veja-se apêndice B, corolário B.1.3) o que significa que as sucessões $\{\varepsilon_{n,1}\}$, $\{\varepsilon_{n,2}\}, \ldots, \{\varepsilon_{n,s}\}$ têm R-ordem de convergência pelo menos igual ao raio espectral da matriz M.

Usaremos indução para demonstrar este teorema.

Demonstração. Uma vez que as sucessões $\{\varepsilon_{n,1}\}$, $\{\varepsilon_{n,2}\}, \ldots, \{\varepsilon_{n,s}\}$ convergem para zero, se, por exemplo, fixarmos $\gamma_i > 0$ $(i = 1, 2, \ldots, s)$ é sempre possível encontrar $n_1 \geq n_0$ e $\theta \in (0,1)$ tais que

$$\varepsilon_{n_1,i} \leq \gamma_i \theta^{u_i \tau^{n_1}}, \qquad i = 1, \ldots, s.$$

Considere-se, então, $\gamma_i > 0$ $(i = 1, 2, \ldots, s)$ fixo e seja $0 < \delta < \gamma_i$. Dado que as sucessões $\{\varepsilon_{n,1}\}$, $\{\varepsilon_{n,2}\}, \ldots, \{\varepsilon_{n,s}\}$ convergem para zero, existe $n_1 \geq n_0$ tal que

$$\varepsilon_{n_1,i} \leq \gamma_i - \delta, \quad i = 1, 2, \ldots, s.$$

Ora, sabendo que quando $\theta \to 1$ temos $\gamma_i \theta^{u_i \tau^{n_1}} \to \gamma_i$, podemos escolher $\theta \in (0,1)$ para o qual acontece $\gamma_i \theta^{u_i \tau^{n_1}} \geq \gamma_i - \delta$ e teremos

$$\varepsilon_{n_1,i} \leq \gamma_i - \delta \leq \gamma_i \theta^{u_i \tau^{n_1}},$$

como se pretendia.

Suponhamos agora que (2.4) se verifica para $n \geq n_1 \geq n_0$, isto é,

$$\varepsilon_{n,i} \leq \gamma_i \theta^{u_i \tau^n}, \quad i = 1, 2, \ldots, s \quad \text{e} \quad n \geq n_1. \tag{2.5}$$

Então, de (2.3) e da hipótese de indução (2.5) vem

$$\begin{aligned}
\varepsilon_{n+1,i} &\leq \alpha_i \prod_{j=1}^{s} \varepsilon_{n,j}^{m_{ij}} \\
&\leq \alpha_i \prod_{j=1}^{s} \left(\gamma_j \theta^{u_j \tau^n}\right)^{m_{ij}} \\
&= \alpha_i \prod_{j=1}^{s} \gamma_j^{m_{ij}} \theta^{m_{ij} u_j \tau^n} \\
&= \alpha_i \left(\prod_{j=1}^{s} \gamma_j^{m_{ij}} \right) \theta^{\tau^n \sum_{j=1}^{s} m_{ij} u_j}.
\end{aligned}$$

Dado que τ é um valor próprio de M, $M\mathbf{u} = \tau\mathbf{u}$ equivale a

$$\sum_{j=1}^{s} m_{ij} u_j = \tau u_i, \qquad i = 1, \ldots, s.$$

Segue-se

$$\begin{aligned} \varepsilon_{n+1,i} &\leq \alpha_i \left(\prod_{j=1}^{s} \gamma_j^{m_{ij}} \right) \theta^{u_i \tau^{n+1}} \\ &\leq \gamma_i \theta^{u_i \tau^{n+1}}, \end{aligned}$$

se $\gamma_1 > 0, \gamma_2 > 0, \ldots, \gamma_s > 0$ satisfizerem

$$\alpha_i \left(\prod_{j=1}^{s} \gamma_j^{m_{ij}} \right) \leq \gamma_i, \qquad i = 1, \ldots, s. \tag{2.6}$$

De facto, existem soluções de (2.6), por exemplo, da forma

$$\gamma_i = e^{u_i \lambda}, \qquad i = 1, \ldots, s,$$

escolhendo $\lambda < 0$ de acordo com

$$\lambda \leq \min_{i=1,\ldots,s} \frac{-\ln \alpha_i}{(\tau - 1)\, u_i}.$$

Com efeito, como $\tau - 1 > 0$ e $u_i > 0$, tem-se

$$\lambda (\tau - 1)\, u_i \leq -\ln \alpha_i,$$

ou, de forma equivalente,

$$\ln \alpha_i + u_i \tau \lambda \leq u_i \lambda.$$

Donde

$$\begin{aligned} & e^{\ln \alpha_i + u_i \tau \lambda} \leq e^{u_i \lambda} \\ \Leftrightarrow \quad & \alpha_i e^{u_i \tau \lambda} \leq e^{u_i \lambda} \\ \Leftrightarrow \quad & \alpha_i e^{\sum_{j=1}^{s} m_{ij} u_j \lambda} \leq e^{u_i \lambda} \\ \Leftrightarrow \quad & \alpha_i \prod_{j=1}^{s} e^{m_{ij} u_j \lambda} \leq e^{u_i \lambda} \\ \Leftrightarrow \quad & \alpha_i \prod_{j=1}^{s} \left(e^{u_j \lambda} \right)^{m_{ij}} \leq e^{u_i \lambda}. \end{aligned}$$

Atendendo a que $\gamma_i = e^{u_i \lambda}$ $(i = 1, \ldots, s)$ temos precisamente o resultado pretendido em (2.6).

Fica assim completa a demonstração do teorema. □

Corolário 2.3.2 (**Schmidt** [24]) *Seja* $\{\varepsilon_n\}$ *uma sucessão convergente para zero e que satisfaz*

$$0 \leq \varepsilon_{n+1} \leq \alpha \prod_{j=0}^{s} \varepsilon_{n-j}^{m_j}, \qquad n \geq n_0, \tag{2.7}$$

com $\alpha > 0$, $m_0 \geq 0, \ldots, m_s \geq 0$ *e* $m_0 + m_1 + \cdots + m_s > 1$.
Então a R-ordem de convergência da sucessão $\{\varepsilon_n\}$ *é pelo menos igual a* $\tau > 1$, *onde* τ *é a única solução positiva da equação*

$$\lambda^{s+1} = m_0\lambda^s + m_1\lambda^{s-1} + \cdots + m_{s-1}\lambda + m_s. \tag{2.8}$$

Demonstração. A condição em (2.7) é equivalente a escrever

$$0 \leq \varepsilon_{n+1,i} \leq \alpha_i \prod_{j=1}^{s+1} \varepsilon_{n,j}^{m_{ij}}, \qquad i = 1, \ldots, s+1,$$

denotando

$$\varepsilon_{n,1} = \varepsilon_n,\ \varepsilon_{n,2} = \varepsilon_{n-1}, \ldots,\ \varepsilon_{n,s+1} = \varepsilon_{n-s},$$

e fazendo

$$\begin{aligned}
\alpha_1 &= \alpha, \\
m_{1j} &= m_{j-1}, \quad j = 1, \ldots, s+1, \\
\alpha_i &= 1, \quad i = 2, \ldots, s+1, \\
m_{ij} &= \begin{cases} 1 \text{ se } i = j+1 \\ 0 \text{ se } i \neq j+1 \end{cases}, \quad i = 2, \ldots, s+1;\ j = 1, \ldots, s+1.
\end{aligned}$$

De facto, ficamos com o sistema de desigualdades

$$\begin{aligned}
0 &\leq \varepsilon_{n+1,1} = \varepsilon_{n+1} \leq \alpha \prod_{j=0}^{s} \varepsilon_{n-j}^{m_j} = \alpha_1 \prod_{j=1}^{s+1} \varepsilon_{n,j}^{m_{1j}} \\
0 &\leq \varepsilon_{n+1,2} = \varepsilon_n \leq \varepsilon_n = \varepsilon_{n,1} = \alpha_2 \prod_{j=1}^{s+1} \varepsilon_{n,j}^{m_{2j}} \\
&\vdots \\
0 &\leq \varepsilon_{n+1,s+1} = \varepsilon_{n+1-s} \leq \varepsilon_{n+1-s} = \varepsilon_{n,s} = \alpha_{s+1} \prod_{j=1}^{s+1} \varepsilon_{n,j}^{m_{s+1,j}}.
\end{aligned}$$

Com esta apresentação estamos em condições de poder aplicar o teorema 2.3.1, desde que a matriz $M = (m_{ij})$, $(i, j = 1, \ldots, s+1)$ dada por

$$M = \begin{pmatrix}
m_0 & m_1 & \cdots & m_{s-1} & m_s \\
1 & 0 & \cdots & 0 & 0 \\
0 & 1 & \cdots & 0 & 0 \\
\vdots & \vdots & \cdots & \vdots & \vdots \\
0 & 0 & \cdots & 1 & 0
\end{pmatrix}$$

tenha um valor próprio $\tau > 1$ ao qual esteja associado um vector próprio $\mathbf{u} > 0$. Vamos ver que isto acontece.

É conhecido que os valores próprios λ de M são as soluções da equação

$$\lambda^{s+1} = m_0\lambda^s + m_1\lambda^{s-1} + \cdots + m_{s-1}\lambda + m_s.$$

Designando $p(\lambda) = \lambda^{s+1} - m_0\lambda^s - m_1\lambda^{s-1} - \cdots - m_{s-1}\lambda - m_s$, se usarmos a *Regra dos Sinais de Descartes* verificamos que p tem um único zero positivo. Seja τ esse zero. Como

$$p(1) = 1 - (m_0 + m_1 + \cdots + m_{s-1} + m_s) < 0$$

e

$$\lim_{\lambda \to +\infty} p(\lambda) = +\infty,$$

τ pertence a $(1, +\infty)$ e é a única solução positiva da equação em (2.8).

Dado $M = (m_{ij})$ ser uma matriz não negativa, $\rho(M)$ é um valor próprio de M ao qual está associado um vector próprio não negativo (veja-se apêndice B, teorema B.1.1). Assim, como τ é o único valor próprio positivo de M, vem que $\rho(M) = \tau > 1$.

É também fácil verificar que todo o vector próprio $\mathbf{u} \geq \mathbf{0}$ correspondente a τ satisfaz $\mathbf{u} > \mathbf{0}$, isto é, $u_1 > 0, u_2 > 0, \ldots, u_{s+1} > 0$ para $\mathbf{u} = (u_1, u_2, \ldots, u_{s+1})^T$. Se suposermos que $\mathbf{u}$ tem uma componente u_i, $i \in \{1, \ldots, s+1\}$, nula chegamos a uma contradição. De facto, de $M\mathbf{u} = \tau\mathbf{u}$ vem

$$\begin{cases} m_0u_1 + m_1u_2 + \ldots + m_su_{s+1} = \tau u_1 \\ u_1 = \tau u_2 \\ \quad \vdots \\ u_{i-1} = \tau u_i \\ u_i = \tau u_{i+1} \\ \quad \vdots \\ u_s = \tau u_{s+1}. \end{cases}$$

Sendo $u_i = 0$, vem $u_{i-1} = 0, \ldots, u_1 = 0$ e dado que $\tau \neq 0$ temos também $u_{i+1} = 0$ o que implica que $u_{i+2} = 0, \ldots, u_{s+1} = 0$, isto é, $\mathbf{u} = (0, 0, \ldots, 0)^T$, o que contradiz a definição de vector próprio.

Assim, estão satisfeitas as condições para podermos aplicar o teorema 2.3.1. Em particular, para a sucessão $\{\varepsilon_{n,1}\} = \{\varepsilon_n\}$ teremos que existem constantes $\theta \in (0,1)$ e $\gamma_1 > 0$ tais que

$$\varepsilon_n \leq b_1\sigma_1^{\tau^n} \quad \text{para } n = 0, 1, \ldots,$$

com $\sigma_1 = \theta^{u_1}$ e $b_1 \geq \max\left\{\gamma_1, \frac{\varepsilon_n}{\sigma_1^{\tau^n}};\ n = 0, 1, \ldots, n_1 - 1\right\}$. Ou seja, a sucessão $\{\varepsilon_n\}$ converge para zero com R-ordem de convergência pelo menos igual a $\tau > 1$. □

2.4 Propriedades de convergência dos métodos I a VII

Considere-se uma função real f contínua num intervalo $[a,b]$ e assuma-se que $f(a)\,f(b) < 0$.

Recordemos os **métodos I a VII** apresentados no capítulo 1. Iremos mostrar que estes métodos, começando com o intervalo inicial $[a_0,b_0] = [a,b]$, produzem uma sucessão de intervalos $\{[a_n,b_n]\}_{n=0}^{\infty}$ tal que

$$[a_{n+1},b_{n+1}] \subset [a_n,b_n] \quad \text{e} \quad f(a_{n+1})\,f(b_{n+1}) < 0.$$

Assim, a sucessão das amplitudes dos intervalos $[a_n,b_n]$,

$$\{(b_n - a_n)\}_{n=0}^{\infty},$$

convergirá para zero, o que significa que as sucessões $\{a_n\}_{n=0}^{n=\infty}$ e $\{b_n\}_{n=0}^{n=\infty}$ convergirão para x^*, sendo x^* um zero de f em $[a,b]$. E teremos também

$$\lim_{n\to\infty} |x^* - \widehat{x}_n| \leq \lim_{n\to\infty} (b_n - a_n) = 0,$$

para qualquer aproximação $\widehat{x}_n \in [a_n,b_n]$.

O teorema que de seguida apresentamos evidencia estas propriedades básicas de convergência dos métodos I a VII e reúne resultados apresentados em Alefeld e Potra [1] e Alefeld, Potra e Yixun Shi [2] e [3].

Teorema 2.4.1 *Seja f uma função real de variável real contínua no intervalo $[a,b]$ e que satisfaz $f(a)\,f(b) < 0$. Considere-se qualquer um dos algoritmos I, II, III, IV, V, VI ou VII.*
Então, ou um zero de f é encontrado ao fim de um número finito de iterações ou a sucessão de intervalos $\{[a_n,b_n]\}_{n=0}^{\infty}$ produzida satisfaz

$$x^* \in [a_{n+1},b_{n+1}] \subset [a_n,b_n] \subset \ldots \subset [a_0,b_0] = [a,b] \tag{2.9}$$

e

$$\lim_{n\to\infty} (b_n - a_n) = 0, \tag{2.10}$$

em que x^* é um zero de f em $[a,b]$.

A demonstração deste teorema é imediata e serão omitidos os pormenores.

Demonstração. Cada iteração dos algoritmos I a VII inicia-se com um intervalo $[a_n,b_n]$ $(n = 0,1,\ldots)$ tal que $f(a_n)\,f(b_n) < 0$, o que garante a existência de pelo menos um zero x^* de f em $[a_n,b_n]$. Os passos realizados numa iteração caracterizam-se pelo cálculo de determinados pontos que pertencem sempre ao intervalo $[a_n,b_n]$ e, com estes pontos, por algumas chamadas ao procedimento *partir_intervalo*1 (veja-se pag. 4). Produzem-se assim intervalos de inclusão intermédios de x^* que estão sempre contidos em $[a_n,b_n]$,

obtendo-se por fim o intervalo $[a_{n+1}, b_{n+1}]$ que satisfaz $f(a_{n+1})\, f(b_{n+1}) < 0$ e com o qual se inicia a iteração seguinte.

Analisaremos com algum detalhe apenas o exemplo do algoritmo II (veja-se pag. 11). Para este algoritmo temos

$$c_n \in [a_n, b_n],$$

dado que $c_n = a_n - f[a_n, b_n]^{-1} f(a_n)$ e $f(a_n)\, f(b_n) < 0$.

Com $[\overline{a}_n, \overline{b}_n] = partir\_intervalo1([a_n, b_n], c_n)$ virá

$$[\overline{a}_n, \overline{b}_n] \subset [a_n, b_n]$$

e $f(\overline{a}_n)\, f(\overline{b}_n) < 0$, por construção do procedimento *partir_intervalo1*.

Segue-se o cálculo do único zero $\overline{c}_n$ do polinómio $p_{(a_n, b_n, c_n)}$ existente no intervalo $[\overline{a}_n, \overline{b}_n]$ (veja-se pag. 10), tendo-se, portanto,

$$\overline{c}_n \in [\overline{a}_n, \overline{b}_n]$$

e, após $partir\_intervalo1\left([\overline{a}_n, \overline{b}_n], \overline{c}_n\right)$, obtém-se

$$[\widehat{a}_n, \widehat{b}_n] \subset [\overline{a}_n, \overline{b}_n],$$

sendo $f(\widehat{a}_n)\, f\left(\widehat{b}_n\right) < 0$.

Pelo apresentado na pag. 5, teremos que $\widehat{c}_n = u_n - 2f\left[\widehat{a}_n, \widehat{b}_n\right]^{-1} f(u_n)$, com $u_n = \widehat{a}_n$ se $|f(\widehat{a}_n)| < \left|f(\widehat{b}_n)\right|$ ou $u_n = \widehat{b}_n$ se $|f(\widehat{a}_n)| \geq \left|f(\widehat{b}_n)\right|$, é tal que

$$\widehat{c}_n \in [\widehat{a}_n, \widehat{b}_n],$$

podendo acontecer $\widehat{c}_n = \widehat{a}_n$ ou $\widehat{c}_n = \widehat{b}_n$. Fazendo $\widetilde{c}_n = \widehat{c}_n$ ou $\widetilde{c}_n = 0.5\left(\widehat{a}_n + \widehat{b}_n\right)$, teremos também

$$\widetilde{c}_n \in [\widehat{a}_n, \widehat{b}_n],$$

e com $\left[\widetilde{a}_n, \widetilde{b}_n\right] = partir_intervalo1\left(\left[\widehat{a}_n, \widehat{b}_n\right], \widetilde{c}_n\right)$acontece

$$[\widetilde{a}_n, \widetilde{b}_n] \subseteq [\widehat{a}_n, \widehat{b}_n],$$

com $f(\widetilde{a}_n)\, f\left(\widetilde{b}_n\right) < 0$.

Finalmente, poderemos terminar com $[a_{n+1}, b_{n+1}] = \left[\widetilde{a}_n, \widetilde{b}_n\right]$ ou $[a_{n+1}, b_{n+1}] = partir\_intervalo1\left(\left[\widetilde{a}_n, \widetilde{b}_n\right], 0.5\left(\widetilde{b}_n + \widetilde{a}_n\right)\right)$, pelo que

$$[a_{n+1}, b_{n+1}] \subseteq [\widetilde{a}_n, \widetilde{b}_n] \quad \text{e} \quad f(a_{n+1})\, f(b_{n+1}) < 0.$$

Assim, em cada iteração do método II é sempre verdade que

$$x^* \in [a_{n+1}, b_{n+1}] \subseteq [\widetilde{a}_n, \widetilde{b}_n] \subseteq [\widehat{a}_n, \widehat{b}_n] \subset [\overline{a}_n, \overline{b}_n] \subset [a_n, b_n], \quad n = 0, 1, \ldots.$$

Para os restantes métodos o estudo é inteiramente análogo, obtendo-se sempre o resultado

$$[a_{n+1}, b_{n+1}] \subset [a_n, b_n] \quad \text{com} \quad f(a_{n+1}) f(b_{n+1}) < 0.$$

Destas observações segue-se (2.9).

Notemos também que a sucessão de intervalos $\{[a_n, b_n]\}_{n=0}^{\infty}$ produzida pelos algoritmos I a VII satisfaz, pelo menos,

$$b_{n+1} - a_{n+1} \leq \mu_1 (b_n - a_n) \text{ para } n \geq 0, \tag{2.11}$$

onde $\mu_1 = \max\{\mu, 0.5\}$ (veja-se o primeiro passo de uma iteração no algoritmo III e o último passo nos restantes algoritmos). Teremos assim que

$$b_n - a_n \leq \mu_1^n (b - a), \quad n = 0, 1, \ldots$$

e, sendo $0 < \mu < 1$, decorre imediatamente que

$$\lim_{n \to \infty} (b_n - a_n) = 0,$$

isto é, (2.10) é sempre válido. □

Por (2.11) está garantido que a sucessão $\{(b_n - a_n)\}_{n=0}^{\infty}$ converge para zero com Q-ordem pelo menos igual a 1 (convergência linear). Impostas certas condições de suavidade à função f, consideradas comuns, obtêm-se melhores resultados.

2.4.1 Métodos I, II e III

Os teoremas que se seguem referem-se às propriedades de convergência dos algoritmos I, II e III. Iremos mostrar que a sucessão $\{(b_n - a_n)\}_{n=0}^{\infty}$ converge para zero com Q-ordem de convergência pelo menos 2 e 4,respectivamente para os algoritmos I e II, enquanto que para o algoritmo III apresenta R-ordem de convergência pelo menos igual a $0.5\left(3 + \sqrt{13}\right)$.

De acordo com o que já foi anteriormente exposto (veja-se pag. 6) podemos estabelecer o seguinte lema.

Lema 2.4.2 (Alefeld e Potra [1]) *Seja f uma função continuamente diferenciável em $[a, b]$ tal que $f(a) f(b) < 0$ e seja x^* um zero simples de f em $[a, b]$. Suponha-se que o algoritmo I (ou o algoritmo II ou o III) não termina ao fim de um número finito de iterações.*
Então existe um inteiro positivo n_1 tal que u_n e $\overline{c}_n$ definidos nos passos 1.3 *e* 1.4 *do algoritmo I (ou u_n e $\widehat{c}_n$ definidos nos passos* 2.5 *e* 2.6 *do algoritmo II ou em* 3.5 *e* 3.6 *do algoritmo III) satisfazem*

$$f(\overline{c}_n) f(u_n) < 0 \quad (\text{ou} \quad f(\widehat{c}_n) f(u_n) < 0) \quad \text{para} \quad n \geq n_1.$$

O próximo teorema estabelece a Q-ordem de convergência para o algoritmo I (veja-se pag. 8).

Teorema 2.4.3 (Alefeld e Potra [1]) *Suponhamos que f é uma função duas vezes continuamente diferenciável em $[a,b]$ com $f(a)f(b) < 0$ e seja x^* um zero simples de f nesse intervalo. Assuma-se que o algoritmo I não termina ao fim de um número finito de iterações.*
Então a sucessão das amplitudes $\{(b_n - a_n)\}_{n=0}^{\infty}$ converge para zero pelo menos Q-quadraticamente, isto é, existe uma constante $\gamma > 0$ tal que

$$b_{n+1} - a_{n+1} \leq \gamma (b_n - a_n)^2, \qquad n = 0, 1, \ldots.$$

Além disso, existe um inteiro positivo n_1 tal que para $n \geq n_1$ se tem

$$a_{n+1} = \widehat{a}_n \quad e \quad b_{n+1} = \widehat{b}_n.$$

Antes de prosseguirmos com a demonstração deste teorema, recordemos que $f[x_0, x_1]$, diferença dividida de primeira ordem de f relativa aos argumentos x_0 e x_1, é definida por

$$f[x_0, x_1] = \frac{f(x_1) - f(x_0)}{x_1 - x_0}.$$

Diferenças divididas de ordem superior de f definem-se recursivamente. A diferença dividida de ordem n relativa aos argumentos distintos $x_0, x_1, \ldots, x_n$ é representada por $f[x_0, x_1, \ldots, x_n]$ e definida pela fórmula

$$f[x_0, x_1, \ldots, x_n] = \frac{f[x_1, \ldots, x_n] - f[x_0, \ldots, x_{n-1}]}{x_n - x_0}. \tag{2.12}$$

Prova-se que as diferenças divididas de ordem n não dependem da ordem pela qual os argumentos são referidos em (2.12), isto é, são funções simétricas dos seus argumentos. Prova-se também que se f tem derivadas contínuas até à ordem n em $[a,b]$ então

$$f[x_0, x_1, \ldots, x_n] = \frac{f^{(n)}(\xi)}{n!}, \tag{2.13}$$

onde $\xi \in (\min\{x_0, x_1, \ldots, x_n\}, \max\{x_0, x_1, \ldots, x_n\})$.

Passamos agora a demonstrar o teorema apresentado.

Demonstração. Recordando que no passo 1.1 do algoritmo I se tem $c_n = a_n - f[a_n, b_n]^{-1} f(a_n)$, vem

$$(c_n - a_n) f[a_n, b_n] = -f(a_n).$$

Assim,

$$\begin{aligned} f(c_n) &= f(c_n) - f(a_n) - f[a_n, b_n](c_n - a_n) \\ &= (f[a_n, c_n] - f[a_n, b_n])(c_n - a_n) \\ &= (f[a_n, c_n] - f[b_n, a_n])(c_n - a_n) \\ &= f[b_n, a_n, c_n](c_n - b_n)(c_n - a_n). \end{aligned}$$

Por (2.13), $f[b_n, a_n, c_n] = \dfrac{f''(\xi)}{2}$ com $\xi \in (\min\{b_n, a_n, c_n\}, \max\{b_n, a_n, c_n\})$ e, designando

$$\gamma_2 = \frac{1}{2} \max_{x \in [a,b]} \left| f''(x) \right|, \tag{2.14}$$

segue-se que

$$|f(c_n)| \leq \gamma_2 |c_n - b_n| |c_n - a_n|.$$

Como $c_n \in [a_n, b_n]$ e

$$\max_{x \in [a_n, b_n]} |(x - a_n)(x - b_n)| = \frac{1}{4}(b_n - a_n)^2, \tag{2.15}$$

vem

$$|f(c_n)| \leq \frac{1}{4}\gamma_2 (b_n - a_n)^2. \tag{2.16}$$

Uma vez que

$$\lim_{n \to +\infty} a_n = \lim_{n \to +\infty} b_n = x^* \quad \text{e} \quad f'(x^*) \neq 0,$$

existe um número inteiro positivo n_1 tal que

$$\max_{x,y \in [a_n, b_n]} \left| f[x,y]^{-1} \right| \leq \gamma_1, \quad n \geq n_1, \tag{2.17}$$

com $\gamma_1 > \dfrac{1}{|f'(x^*)|}$, pois acontece $\lim\limits_{\substack{n \to +\infty \\ x,y \in [a_n, b_n]}} f[x,y]^{-1} = \dfrac{1}{f'(x^*)}$.

Se n_1 for suficientemente grande, de acordo com o lema 2.4.2, podemos assumir que

$$f(\overline{c}_n) f(u_n) < 0, \quad n \geq n_1,$$

o que significa que o zero x^* está entre $\overline{c}_n$ e u_n.

Então, dos passos 1.4 a 1.6 do algoritmo I e do facto que $u_n, \overline{c}_n \in \left[\overline{a}_n, \overline{b}_n\right]$, deduz-se que

$$\widehat{b}_n - \widehat{a}_n \leq |\overline{c}_n - u_n|, \quad n \geq n_1. \tag{2.18}$$

Por outro lado, $c_n \in \left\{\overline{a}_n, \overline{b}_n\right\}$ implica que

$$|f(c_n)| \geq |f(u_n)| \tag{2.19}$$

(vejam-se passos 1.2 e 1.3).

Desta forma, de (2.18), do passo 1.4 do algoritmo I, de (2.19), de (2.16) e (2.17) segue-se que

$$\begin{aligned} \widehat{b}_n - \widehat{a}_n &\leq |\overline{c}_n - u_n| = \left|2f\left[\overline{a}_n, \overline{b}_n\right]^{-1} f(u_n)\right| \\ &\leq 2\left|f\left[\overline{a}_n, \overline{b}_n\right]^{-1}\right| |f(c_n)| \\ &\leq \gamma_3 (b_n - a_n)^2, \quad n \geq n_1, \end{aligned} \tag{2.20}$$

com $\gamma_3 = \frac{1}{2}\gamma_1\gamma_2$.

Se n_1 for suficientemente grande também acontece

$$\widehat{b}_n - \widehat{a}_n < \mu (b_n - a_n), \quad n \geq n_1,$$

(veja-se passo 1.7), o que deriva imediatamente de (2.20) e do facto que

$$\gamma_3 (b_n - a_n) < \mu, \quad n \geq n_1,$$

uma vez que $\lim_{n \to +\infty} (b_n - a_n) = 0$. Isto prova que para $n \geq n_1$ se tem

$$a_{n+1} = \widehat{a}_n \quad \text{e} \quad b_{n+1} = \widehat{b}_n. \tag{2.21}$$

Tomando

$$\gamma \geq \max\left\{\gamma_3, \frac{(b_{i+1} - a_{i+1})}{(b_i - a_i)^2};\ i = 0, 1, \ldots, n_1 - 1\right\}$$

e usando (2.20) e (2.21) teremos

$$b_{n+1} - a_{n+1} \leq \widehat{b}_n - \widehat{a}_n \leq \gamma (b_n - a_n)^2, \quad \gamma > 0 \quad \text{e} \quad n = 0, 1, \ldots,$$

o que significa que no algoritmo I a sucessão $\{(b_n - a_n)\}_{n=0}^{\infty}$ converge para zero com Q-ordem de convergência pelo menos 2 ou, por outras palavras, converge pelo menos Q-quadraticamente para zero. □

O teorema seguinte refere-se à Q-ordem de convergência do algoritmo II (veja-se pag. 11).

Teorema 2.4.4 (Alefeld e Potra [*1*]) *Considere-se uma função f três vezes continuamente diferenciável em $[a, b]$ com $f(a) f(b) < 0$ e seja x^* um zero simples de f nesse intervalo. Assuma-se que o algoritmo II não termina ao fim de um número finito de iterações.*
Então a sucessão das amplitudes $\{(b_n - a_n)\}_{n=0}^{\infty}$ converge para zero com Q-ordem de convergência pelo menos 4, isto é, existe uma constante $\overline{\gamma} > 0$ tal que

$$b_{n+1} - a_{n+1} \leq \overline{\gamma} (b_n - a_n)^4, \qquad n = 0, 1, \ldots.$$

Além disso, existe um inteiro positivo n_2 tal que para $n \geq n_2$ se tem

$$a_{n+1} = \widetilde{a}_n \quad e \quad b_{n+1} = \widetilde{b}_n.$$

Demonstração. Tal como na demonstração do teorema 2.4.3 podemos assumir que existe um inteiro n_1 tal que (2.17) acontece. De forma correspondente, se n_1 é suficientemente grande teremos também

$$f(\widehat{c}_n)f(u_n) < 0, \quad n \geq n_1,$$

e, atendendo aos passos 2.6 a 2.8 do algoritmo II, virá

$$\widetilde{b}_n - \widetilde{a}_n \leq |\widehat{c}_n - u_n|, \quad n \geq n_1. \tag{2.22}$$

Com os passos 2.4 e 2.5 temos que $\overline{c}_n \in \left\{\widehat{a}_n, \widehat{b}_n\right\}$ e

$$|f(\overline{c}_n)| \geq |f(u_n)|. \tag{2.23}$$

A partir do passo 2.6, de (2.23) e de (2.17), segue-se que

$$\begin{aligned} |\widehat{c}_n - u_n| &= \left|2f\left[\widehat{a}_n, \widehat{b}_n\right]^{-1} f(u_n)\right| \\ &\leq 2\left|f\left[\widehat{a}_n, \widehat{b}_n\right]^{-1}\right| |f(\overline{c}_n)| \\ &\leq 2\gamma_1 |f(\overline{c}_n)|. \end{aligned} \tag{2.24}$$

Seja

$$\gamma_4 = \frac{1}{3!} \max_{x\in[a,b]} \left|f'''(x)\right|,$$

e considere-se o polinómio quadrático interpolador da função f nos pontos a_n, b_n e c_n que designamos por $p_{(a_n,b_n,c_n)}$ (veja-se passo 2.3). Atendendo à conhecida fórmula para o erro de interpolação obtém-se

$$\begin{aligned} |f(\overline{c}_n)| &= \left|f(\overline{c}_n) - p_{(a_n,b_n,c_n)}(\overline{c}_n)\right| \\ &= \frac{1}{3!}\left|f'''(\xi)\right| |\overline{c}_n - a_n| |\overline{c}_n - b_n| |\overline{c}_n - c_n|, \end{aligned} \tag{2.25}$$

com $\xi \in (\min\{a_n, b_n, c_n\}, \max\{a_n, b_n, c_n\})$.
Como

$$\begin{aligned} |\overline{c}_n - c_n| &= \left|f[c_n, \overline{c}_n]^{-1}(f(\overline{c}_n) - f(c_n))\right| \\ &\leq \left|f[c_n, \overline{c}_n]^{-1}\right| (|f(\overline{c}_n)| + |f(c_n)|), \end{aligned}$$

de (2.17) vem

$$|\overline{c}_n - c_n| \leq \gamma_1 (|f(\overline{c}_n)| + |f(c_n)|). \tag{2.26}$$

Com (2.15) e com as desigualdades (2.25) e (2.26) obtém-se

$$\begin{aligned} |f(\overline{c}_n)| &\leq \frac{1}{4}\gamma_4 (b_n - a_n)^2 \gamma_1 (|f(\overline{c}_n)| + |f(c_n)|) \\ &= \gamma_5 (b_n - a_n)^2 (|f(\overline{c}_n)| + |f(c_n)|), \end{aligned} \tag{2.27}$$

fazendo $\gamma_5 = \frac{1}{4}\gamma_4\gamma_1$.

Porque $\lim_{n\to+\infty}(b_n - a_n) = 0$, podemos assumir que existe um inteiro positivo $n_2 \geq n_1$ tal que

$$\gamma_5 (b_n - a_n)^2 \leq \frac{1}{2}, \quad n \geq n_2. \tag{2.28}$$

Assim, de (2.27) e (2.28) temos

$$\begin{aligned} |f(\overline{c}_n)| &\leq \gamma_5 (b_n - a_n)^2 |f(\overline{c}_n)| + \gamma_5 (b_n - a_n)^2 |f(c_n)| \\ &\leq \frac{1}{2}|f(\overline{c}_n)| + \gamma_5 (b_n - a_n)^2 |f(c_n)|, \end{aligned}$$

o que equivale a ter

$$|f(\overline{c}_n)| \leq 2\gamma_5 (b_n - a_n)^2 |f(c_n)|, \quad n \geq n_2. \tag{2.29}$$

Por (2.22), (2.24), (2.29) e usando, como visto em (2.16),

$$|f(c_n)| \leq \frac{1}{4}\gamma_2 (b_n - a_n)^2, \tag{2.30}$$

vem finalmente,

$$\begin{aligned} \widetilde{b}_n - \widetilde{a}_n &\leq |\widehat{c}_n - u_n| \\ &\leq 2\gamma_1 |f(\overline{c}_n)| \\ &\leq 4\gamma_1\gamma_5 (b_n - a_n)^2 |f(c_n)| \\ &\leq \gamma_1\gamma_5\gamma_2 (b_n - a_n)^4, \quad n \geq n_2. \end{aligned} \tag{2.31}$$

Novamente, se n_2 é suficientemente grande, de (2.31) e do facto que

$$\gamma_1\gamma_2\gamma_5 (b_n - a_n)^3 < \mu, \quad n \geq n_2,$$

dado que $\lim_{n\to+\infty}(b_n - a_n) = 0$, vem

$$\widetilde{b}_n - \widetilde{a}_n < \mu (b_n - a_n), \quad n \geq n_2$$

(veja-se passo 2.9). Isto prova que para $n \geq n_2$ se tem

$$a_{n+1} = \widetilde{a}_n \quad \text{e} \quad b_{n+1} = \widetilde{b}_n. \tag{2.32}$$

Se considerarmos

$$\overline{\gamma} \geq \max\left\{\gamma_1\gamma_2\gamma_5, \frac{(b_{i+1} - a_{i+1})}{(b_i - a_i)^4};\ i = 0, 1, \ldots, n_2 - 1\right\}$$

e usarmos (2.31) e (2.32) teremos

$$b_{n+1} - a_{n+1} \leq \widetilde{b}_n - \widetilde{a}_n \leq \overline{\gamma}(b_n - a_n)^4, \quad \overline{\gamma} > 0 \quad \text{e} \quad n = 0, 1, \ldots,$$

o que significa que no algoritmo II a sucessão $\{(b_n - a_n)\}_{n=0}^{\infty}$ converge para zero com Q-ordem de convergência pelo menos 4. □

Finalmente para o algoritmo III (veja-se pag. 13), prova-se a seguir que tem R-ordem de convergência pelo menos igual a $0.5\left(3+\sqrt{13}\right)$.

Teorema 2.4.5 (Alefeld e Potra [1]) *Considerem-se as mesmas condições do teorema 2.4.4 e assuma-se que o algoritmo III não termina ao fim de um número finito de iterações.*
Então a sucessão $\{(b_n-a_n)\}_{n=0}^{\infty}$ *converge para zero com R-ordem de convergência pelo menos igual a* $0.5\left(3+\sqrt{13}\right)=3.3027\ldots$.

Demonstração. As desigualdades

$$|\widehat{c}_n-u_n|\leq 2\gamma_1\,|f(\bar{c}_n)|\,,\qquad n\geq n_1,$$

e

$$|f(\bar{c}_n)|\leq 2\gamma_5\,(b_n-a_n)^2\,|f(c_n)|\,,\qquad n\geq n_2\geq n_1,$$

vistas em (2.24) e (2.29), respectivamente, não dependem de uma escolha particular de $c_n\in[a_n,b_n]$.
Então, para qualquer ponto $c_n\in[a_n,b_n]$ teremos

$$|\widehat{c}_n-u_n|\leq \gamma_6\,(b_n-a_n)^2\,|f(c_n)|\,, \tag{2.33}$$

com $\gamma_6=4\gamma_1\gamma_5$.
Usando novamente o lema 2.4.2 podemos assumir que

$$f(\widehat{c}_n)f(u_n)<0,\qquad n\geq n_3\geq n_2,$$

para n_3 suficientemente grande.
Assim, de acordo com os passos 3.5 a 3.7 do algoritmo III, temos

$$\{a_{n+1},b_{n+1}\}=\{u_n,\widehat{c}_n\}\,,\qquad n\geq n_3. \tag{2.34}$$

Então, definindo

$$c_{n+1}=\frac{a_{n+1}+b_{n+1}}{2}=\frac{u_n+\widehat{c}_n}{2},$$

vem

$$c_{n+1}-u_n=\frac{\widehat{c}_n-u_n}{2}.$$

A partir do passo 3.6 do mesmo algoritmo temos

$$(\widehat{c}_n-u_n)\,f\left[\widehat{a}_n,\widehat{b}_n\right]=-2f(u_n).$$

Assim,

$$\begin{aligned}
2f\,(c_{n+1}) &= 2f\,(c_{n+1})-2f(u_n)-(\widehat{c}_n-u_n)\,f\left[\widehat{a}_n,\widehat{b}_n\right]\\
&= 2f\,[u_n,c_{n+1}]\,(c_{n+1}-u_n)-f\left[\widehat{a}_n,\widehat{b}_n\right](\widehat{c}_n-u_n)\\
&= f\,[u_n,c_{n+1}]\,(\widehat{c}_n-u_n)-f\left[\widehat{a}_n,\widehat{b}_n\right](\widehat{c}_n-u_n)\\
&= (\widehat{c}_n-u_n)\left(f\,[u_n,c_{n+1}]-f\left[\widehat{a}_n,\widehat{b}_n\right]\right). \qquad (2.35)
\end{aligned}$$

Usando (2.13) e aplicando o teorema do valor médio, vem

$$\begin{aligned}\left|f\left[u_n, c_{n+1}\right] - f\left[\widehat{a}_n, \widehat{b}_n\right]\right| &= \left|f'(\xi_1) - f'(\xi_2)\right| \\ &= \left|f''(\xi)\right| \left|\xi_2 - \xi_1\right|,\end{aligned}$$

com $\xi_1, \xi_2 \in \left(\widehat{a}_n, \widehat{b}_n\right)$ e $\xi \in \left(\min\left\{\xi_1, \xi_2\right\}, \max\left\{\xi_1, \xi_2\right\}\right)$.
Sendo γ_2 como dado em (2.14), segue-se que

$$\left|f\left[u_n, c_{n+1}\right] - f\left[\widehat{a}_n, \widehat{b}_n\right]\right| \leq 2\gamma_2 \left(b_n - a_n\right). \tag{2.36}$$

Assim, de (2.35), (2.33) e de (2.36) vem

$$\begin{aligned}\left|f\left(c_{n+1}\right)\right| &= \frac{1}{2}\left|\widehat{c}_n - u_n\right| \left|f\left[u_n, c_{n+1}\right] - f\left[\widehat{a}_n, \widehat{b}_n\right]\right| \\ &\leq \gamma_7 \left(b_n - a_n\right)^3 \left|f(c_n)\right|, \quad n \geq n_3,\end{aligned} \tag{2.37}$$

com $\gamma_7 = \gamma_6\gamma_2$.

Defina-se

$$\varepsilon_n = b_n - a_n \quad \text{e} \quad \eta_n = \left|f(c_n)\right|, \quad n = 0, 1, \ldots,$$

e note-se que $\{\varepsilon_n\}$ e $\{\eta_n\}$ são sucessões de termos não negativos que convergem para zero.
A partir de (2.34) e (2.33) deduz-se que

$$\begin{aligned}\varepsilon_{n+1} &= b_{n+1} - a_{n+1} \\ &= \left|\widehat{c}_n - u_n\right| \\ &\leq \gamma_6 \varepsilon_n^2 \eta_n,\end{aligned}$$

e, a partir de (2.37),

$$\eta_{n+1} = \left|f(c_{n+1})\right| \leq \gamma_7 \varepsilon_n^3 \eta_n,$$

com $n \geq n_3$.
Denotando $\varepsilon_{n,1} = \varepsilon_n$ e $\varepsilon_{n,2} = \eta_n$ ficamos com o seguinte sistema de desigualdades

$$\begin{aligned}\varepsilon_{n+1,1} &\leq \gamma_6 \varepsilon_{n,1}^2 \varepsilon_{n,2} \\ \varepsilon_{n+1,2} &\leq \gamma_7 \varepsilon_{n,1}^3 \varepsilon_{n,2}.\end{aligned}$$

De acordo com o teorema 2.3.1, as sucessões $\{\varepsilon_{n,1}\}_{n=0}^{\infty}$ e $\{\varepsilon_{n,2}\}_{n=0}^{\infty}$ têm R-ordem de convergência pelo menos igual ao raio espectral τ da matriz $M = \begin{pmatrix} 2 & 1 \\ 3 & 1 \end{pmatrix}$ que é igual a $0.5\left(3 + \sqrt{13}\right) > 1$. De facto, τ é um valor próprio da matriz M ao qual está associado, por exemplo, o vector $u = \left(1, 0.5\left(\sqrt{13} - 1\right)\right)^T$, cujas componentes são todas positivas.

Em particular, fica demonstrado que a sucessão $\{\varepsilon_n\}_{n=0}^{\infty} = \{(b_n - a_n)\}_{n=0}^{\infty}$ converge para zero com R-ordem de convergência pelo menos $0.5\left(3 + \sqrt{13}\right)$. □

2.4.2 Métodos IV e V

Nesta secção consideraremos as propriedades de convergência dos algoritmos IV e V (vejam-se pags. 18 e 21). Iremos provar que a sucessão $\{(b_n - a_n)\}_{n=0}^{\infty}$ converge para zero com R-ordem de convergência pelo menos $1 + \sqrt{2} = 2.4142\ldots$, no caso do algoritmo IV, e com R-ordem de convergência pelo menos $2 + \sqrt{5} = 4.2360\ldots$, no caso do algoritmo V.

Recorde-se que entre os algoritmos IV e II a diferença básica é que o primeiro, ao contrário do segundo, não usa a solução exacta de uma equação quadrática em cada iteração. Em vez disso, usa duas iterações do método de Newton para obter uma aproximação conveniente.

Dada uma função f definida num intervalo $[a,b]$ com $f(a)\,f(b) < 0$, considere-se então o polinómio quadrático $p_{(a,b,d)}$ interpolador de f nos pontos a, b e d, com $d \notin [a,b]$. Recordemos o procedimento *newton_quadrático* (vejam-se pags. 15 e 16) que determina uma aproximação r para o único zero z do polinómio $p_{(a,b,d)}$ pertencente a $[a,b]$, usando k iterações do método de Newton.

Começaremos por apresentar um lema que estabelece um limite superior para o erro absoluto $|z - r|$.

Lema 2.4.6 (Alefeld, Potra e Yixun Shi [2]) *Considere-se o procedimento newton_quadrático e seja $z \in [a,b]$ um zero do polinómio $p_{(a,b,d)}$.*

(i) Dada a escolha feita para a aproximação inicial r_0, o valor obtido $r = r_k$ após k iterações do método de Newton satisfaz $r \in (a,b)$ e tem-se $r \approx z$.

(ii) Suponhamos que f é duas vezes continuamente diferenciável num intervalo $[e,l]$ com $f'(x) \neq 0$ para $x \in [e,l]$. Se $\{a,b,d\} \subseteq [e,l]$ e se

$$\delta = \min_{e \leq x \leq l} \left| f'(x) \right| - (b-a) \max_{e \leq x \leq l} \left| f''(x) \right| > 0,$$

então

$$|z - r| \leq \lambda^L (b-a)^{2^k},$$

onde

$$\lambda = \frac{1}{2\delta} \max_{e \leq x \leq l} \left| f''(x) \right| \quad e \quad L = 2^k - 1.$$

Demonstração. (i) resulta da convergência monótona do método de Newton para polinómios do segundo grau. Refira-se que escolher inicialmente $r_0 \in \{a,b\}$ de forma que $p''_{(a,b,d)}(r_0)\, p_{(a,b,d)}(r_0) > 0$ assegura a convergência para a raiz z, acontecendo sempre $p'_{(a,b,d)}(r_i) > 0$ ou $p'_{(a,b,d)}(r_i) < 0$ e $r_i \in [a,z]$ ou $r_i \in [z,b]$, para $i = 0, 1, \ldots k$.

Para mostrar o resultado em (**ii**) notemos que

$$\begin{aligned} p_{(a,b,d)}(x) &= f(a) + f[a,b](x-a) + f[a,b,d](x-a)(x-b) \\ &= f(a) + B(x-a) + A(x-a)(x-b), \end{aligned}$$

e, por (2.13) temos

$$|A| = |f[a,b,d]| = \left|\frac{f''(\xi)}{2}\right| \leq \frac{1}{2} \max_{e\leq x\leq l} \left|f''(x)\right|,$$

uma vez que $\xi \in (\min\{a,b,d\}, \max\{a,b,d\})$ e $\{a,b,d\} \subseteq [e,l]$.
Temos também que, para $x \in [a,b]$,

$$\begin{aligned} \left|p'_{(a,b,d)}(x)\right| &= |B + A(2x-a-b)| \\ &\geq |B| - |A|\,|2x-a-b| \\ &\geq |B| - |A|\,(b-a) \\ &= |f[a,b]| - |f[a,b,d]|\,(b-a) \\ &= \left|f'(\xi_1)\right| - \frac{1}{2}\left|f''(\xi_2)\right|(b-a), \quad \xi_1 \in (a,b) \text{ e } \xi_2 \text{ entre } a, b \text{ e } d \\ &\geq \min_{e\leq x\leq l}\left|f'(x)\right| - (b-a)\max_{e\leq x\leq l}\left|f''(x)\right| \\ &= \delta > 0, \quad \text{para } x \in [e,l] \supset [a,b]. \end{aligned}$$

Por simplicidade, no que se segue o polinómio $p_{(a,b,d)}$ será abreviado por p.

Considere-se a seguinte expressão para o polinómio p (série de Taylor de 2^a ordem de p no ponto r_{k-1}):

$$p(x) = p(r_{k-1}) + (x-r_{k-1})\,p'(r_{k-1}) + (x-r_{k-1})^2 A.$$

Como $p(z) = 0$ e $p'(r_{k-1}) \neq 0$, podemos escrever

$$r_{k-1} - \frac{p(r_{k-1})}{p'(r_{k-1})} = z + (z-r_{k-1})^2 \frac{A}{p'(r_{k-1})},$$

isto é,

$$r_k = z + (z-r_{k-1})^2 \frac{A}{p'(r_{k-1})}.$$

Assim,

$$\begin{aligned} |z-r_k| &= |z-r_{k-1}|^2 \frac{|A|}{|p'(r_{k-1})|} \\ &\leq |z-r_{k-1}|^2 \frac{1}{2}\max_{e\leq x\leq l}\left|f''(x)\right| \frac{1}{|p'(r_{k-1})|} \\ &\leq |z-r_{k-1}|^2 \frac{1}{2\delta}\max_{e\leq x\leq l}\left|f''(x)\right| \\ &= |z-r_{k-1}|^2 \lambda. \end{aligned}$$

Aplicando este resultado sucessivamente, tem-se

$$|z - r| = |z - r_k| \leq \lambda^L |z - r_0|^{2^k} \leq \lambda^L (b-a)^{2^k},$$

onde $L = 1 + 2 + \ldots + 2^{k-1} = 2^k - 1$. □

Continuaremos com mais dois lemas antes de podermos apresentar os teoremas que se referem à R-ordem de convergência dos algoritmos IV e V.

Lema 2.4.7 (Alefeld, Potra e Yixun Shi [2]) *Seja f uma função continuamente diferenciável em $[a,b]$ tal que $f(a) f(b) < 0$ e seja x^* um zero simples de f em $[a,b]$. Suponha-se que o algoritmo IV (ou o algoritmo V) não termina ao fim de um número finito de iterações.*
Então existe um inteiro positivo n_3 tal que para u_n e $\bar{c}_n$ definidos nos passos 4.3 *e* 4.4 *(ou em* 5.5 *e* 5.6*) satisfazem*

$$f(\bar{c}_n) f(u_n) < 0 \quad \textit{para} \quad n \geq n_3.$$

Este resultado, tendo sido já anteriormente mencionado para os algoritmos I, II e III, considera-se provado.

Lema 2.4.8 (Alefeld, Potra e Yixun Shi [2]) *Considerem-se as condições do lema anterior e assuma-se que f é três vezes continuamente diferenciável em $[a,b]$.*
Então

(*i*) *para o algoritmo IV, existe uma constante $r_1 > 0$ e n_1 tais que*

$$|f(c_n)| \leq r_1 (b_n - a_n)^2 (b_{n-1} - a_{n-1}), \quad n \geq n_1,$$

onde c_n está definido no passo 4.1;

(*ii*) *para o algoritmo V, existe uma constante $r_2 > 0$ e n_2 tais que*

$$|f(\tilde{c}_n)| \leq r_2 (b_n - a_n)^4 (b_{n-1} - a_{n-1}), \quad n \geq n_2,$$

onde $\tilde{c}_n$ está definido no passo 5.3.

Demonstração. Pelo teorema 2.4.1, recorde-se que se tem sempre

$$\lim_{n\to\infty} (b_n - a_n) = 0 \quad \text{e} \quad x^* \in (a_n, b_n).$$

Assim, quando n é suficientemente grande acontece

$$f'(x) \neq 0 \text{ para } x \in [a_n, b_n],$$

uma vez que f' é contínua em $[a,b]$ e $f'(x^*) \neq 0$. Por uma questão de simplicidade, vamos assumir que $f'(x) \neq 0$ para $x \in [a,b]$.

Seja $\lambda_0 = \min_{a \le x \le b} |f'(x)|$ e fixe-se $0 < \delta < \lambda_0$. Existe um inteiro n_1 tal que para $n \ge n_1$ acontece simultaneamente

$$b_n - a_n < 1 \tag{2.38}$$

e

$$(b_n - a_n) \max_{a \le x \le b} |f''(x)| < \lambda_0 - \delta.$$

Assim,

$$\delta_n = \lambda_0 - (b_n - a_n) \max_{a \le x \le b} |f''(x)| > \delta > 0, \quad n \ge n_1. \tag{2.39}$$

(i) Consideremos $n \ge n_1$. Para o algoritmo IV, suponhamos que z_n é o único zero do polinómio $p_{(a_n, b_n, d_n)}$ em $[a_n, b_n]$. Pela fórmula para o erro de interpolação, tem-se

$$\begin{aligned} |f(z_n)| &= |f(z_n) - p_{(a_n,b_n,d_n)}(z_n)| \\ &\le \lambda_1 |z_n - a_n| |z_n - b_n| |z_n - d_n|, \end{aligned}$$

onde $\lambda_1 = \frac{1}{3!} \max_{x \in [a,b]} \left| f'''(x) \right|$.
Notando que $z_n \in [a_n, b_n] \subset [a_{n-1}, b_{n-1}]$ e $d_n \in [a_{n-1}, b_{n-1}]$, temos

$$|z_n - a_n| |z_n - b_n| \le \frac{1}{4} (b_n - a_n)^2$$

e

$$|z_n - d_n| \le b_{n-1} - a_{n-1}.$$

Segue-se

$$|f(z_n)| \le \frac{1}{4} \lambda_1 (b_n - a_n)^2 (b_{n-1} - a_{n-1}). \tag{2.40}$$

Seja c_n a aproximação para o zero z_n obtida através da chamada a *newton_quadrático* com $k = 2$, isto é, com duas iterações do método de Newton (veja-se passo 4.1). Observando (2.39), se usarmos o lema 2.4.6, temos

$$\begin{aligned} |c_n - z_n| &\le \left(\frac{1}{2\delta_n} \max_{a \le x \le b} |f''(x)| \right)^3 (b_n - a_n)^4. \\ &< \lambda_2 (b_n - a_n)^4, \quad n \ge n_1, \end{aligned}$$

onde $\lambda_2 = \left(\frac{1}{2\delta} \max_{a \le x \le b} |f''(x)| \right)^3$, dado que $\delta_n > \delta$.
Por (2.38) e, dado que $[a_n, b_n] \subset [a_{n-1}, b_{n-1}]$ se tem $b_n - a_n < b_{n-1} - a_{n-1}$, podemos escrever

$$(b_n - a_n)^4 < (b_n - a_n)^2 (b_n - a_n) < (b_n - a_n)^2 (b_{n-1} - a_{n-1}),$$

e virá

$$|c_n - z_n| < \lambda_2 (b_n - a_n)^2 (b_{n-1} - a_{n-1}), \quad n \geq n_1. \tag{2.41}$$

Sendo

$$\frac{|f(c_n)| - |f(z_n)|}{|c_n - z_n|} \leq \frac{|f(c_n) - f(z_n)|}{|c_n - z_n|} = f'(\xi)_{\xi \text{ entre } c_n \text{ e } z_n} \leq \max_{a \leq x \leq b} |f'(x)|,$$

ou seja,

$$|f(c_n)| \leq \left(\max_{a \leq x \leq b} |f'(x)| \right) |c_n - z_n| + |f(z_n)|,$$

se combinarmos (2.41) e (2.40) ficamos com

$$|f(c_n)| \leq r_1 (b_n - a_n)^2 (b_{n-1} - a_{n-1}), \quad n \geq n_1, \tag{2.42}$$

onde $r_1 = \lambda_2 \max_{a \leq x \leq b} |f'(x)| + \frac{1}{4}\lambda_1 > 0$.

(ii) Consideremos novamente $n \geq n_1$. Para o algoritmo V, temos igualmente

$$|f(c_n)| \leq r_1 (b_n - a_n)^2 (b_{n-1} - a_{n-1}), \quad n \geq n_1, \tag{2.43}$$

uma vez que c_n é definido no passo 5.1 do algoritmo V que coincide com o passo 4.1 do algoritmo IV.

Suponhamos que $\widetilde{z}_n$ é o único zero do polinómio $p_{(\widetilde{a}_n, \widetilde{b}_n, \widetilde{d}_n)} \equiv p_{(a_n, b_n, c_n)}$ em $\left[\widetilde{a}_n, \widetilde{b}_n\right]$. Tal como visto para o algoritmo II na demonstração do teorema 2.4.4 em (2.29), fazendo corresponder $\widetilde{z}_n$ a $\overline{c}_n$ e sendo c_n definido no passo 5.1, existe n_2 (podemos escolher $n_2 \geq n_1$) tal que

$$|f(\widetilde{z}_n)| \leq \lambda_3 (b_n - a_n)^2 |f(c_n)|, \quad n \geq n_2, \tag{2.44}$$

onde $\lambda_3 = \frac{1}{2} \times \frac{1}{3!} \max_{x \in [a,b]} \left| f'''(x) \right| \times \max_{x \in [a,b]} \frac{1}{|f'(x)|} = \frac{1}{2} \frac{\lambda 1}{\lambda_0}$.

Seja $\widetilde{c}_n$ a aproximação para o zero $\widetilde{z}_n$ obtida através da chamada a *newton_quadrático* com $k = 3$ (veja-se passo 5.3). De forma análoga a (2.41), usando novamente o lema 2.4.6 e (2.39) podemos escrever

$$\begin{aligned} |\widetilde{c}_n - \widetilde{z}_n| &\leq \left(\frac{1}{2\widetilde{\delta}_n} \max_{a \leq x \leq b} |f''(x)| \right)^7 \left(\widetilde{b}_n - \widetilde{a}_n \right)^8 \\ &< \lambda_4 (b_n - a_n)^8, \quad n \geq n_2, \end{aligned}$$

onde $\widetilde{\delta}_n = \lambda_0 - \left(\widetilde{b}_n - \widetilde{a}_n \right) \max_{a \leq x \leq b} |f''(x)| > \delta_n > \delta$ e $\lambda_4 = \left(\frac{1}{2\delta} \max_{a \leq x \leq b} |f''(x)| \right)^7$. Por (2.38) e dado que $\widetilde{b}_n - \widetilde{a}_n < b_n - a_n < b_{n-1} - a_{n-1}$, pois $\left[\widetilde{a}_n, \widetilde{b}_n\right] \subset [a_n, b_n] \subset [a_{n-1}, b_{n-1}]$, tem-se

$$\left(\widetilde{b}_n - \widetilde{a}_n \right)^8 < (b_n - a_n)^8 < (b_n - a_n)^4 (b_n - a_n) < (b_n - a_n)^4 (b_{n-1} - a_{n-1}),$$

e

$$|\widetilde{c}_n - \widetilde{z}_n| < \lambda_4 (b_n - a_n)^4 (b_{n-1} - a_{n-1}), \quad n \geq n_2. \tag{2.45}$$

De forma semelhante a (2.42), combinando (2.45), (2.44) e (2.43) obtém-se

$$\begin{aligned} |f(\widetilde{c}_n)| &\leq \left(\max_{a\leq x\leq b} |f'(x)|\right) |\widetilde{c}_n - \widetilde{z}_n| + |f(\widetilde{z}_n)| \\ &\leq \left(\max_{a\leq x\leq b} |f'(x)|\right) \lambda_4 (b_n - a_n)^4 (b_{n-1} - a_{n-1}) + \lambda_3 (b_n - a_n)^2 |f(c_n)| \\ &\leq r_2 (b_n - a_n)^4 (b_{n-1} - a_{n-1}), \quad n \geq n_2, \end{aligned}$$

onde $r_2 = \lambda_4 \max_{a\leq x\leq b} |f'(x)| + \lambda_3 r_1 > 0$. □

O resultado sobre a R-ordem de convergência do algoritmo IV deriva como corolário do teorema seguinte.

Teorema 2.4.9 (Alefeld, Potra e Yixun Shi [2]) *Assuma-se que f é três vezes continuamente diferenciável em $[a,b]$ e suponhamos que $f(a) f(b) < 0$ sendo x^* um zero simples de f em $[a,b]$.*
Então a sucessão $\{(b_n - a_n)\}_{n=0}^{\infty}$ produzida pelo algoritmo IV converge para zero e existe uma constante $L_1 > 0$ tal que

$$b_{n+1} - a_{n+1} \leq L_1 (b_n - a_n)^2 (b_{n-1} - a_{n-1}), \quad n = 1, 2, \ldots.$$

Mais ainda, existe um inteiro N_1 tal que para $n \geq N_1$ se tem

$$a_{n+1} = \widehat{a}_n \quad e \quad b_{n+1} = \widehat{b}_n.$$

Demonstração. Tal como na demonstração do lema 2.4.8, vamos assumir, sem perda de generalidade, que $f'(x) \neq 0$ para $x \in [a,b]$.

Seja N_1 tal que $N_1 \geq \max\{n_1, n_3\}$, onde n_3 e n_1 são referidos nos lemas 2.4.7 e 2.4.8, respectivamente. A partir dos passos 4.4 a 4.6 do algoritmo IV e do facto que u_n, $\overline{c}_n \in [\overline{a}_n, \overline{b}_n]$ deduz-se que

$$\widehat{b}_n - \widehat{a}_n \leq |\overline{c}_n - u_n|, \quad n \geq N_1. \tag{2.46}$$

Do passo 4.4 obtém-se

$$\begin{aligned} |\overline{c}_n - u_n| &= \left|2 f[\overline{a}_n, \overline{b}_n]^{-1} f(u_n)\right| \\ &= 2\left|\frac{1}{f'(\xi)}\right| |f(u_n)|, \quad \xi \in (\overline{a}_n, \overline{b}_n) \\ &\leq \frac{2}{\lambda_0} |f(u_n)|, \end{aligned} \tag{2.47}$$

onde λ_0 ficou atrás definido por $\lambda_0 = \min_{a\leq x\leq b} |f'(x)|$.

Finalmente, uma vez que $c_n \in \{\overline{a}_n, \overline{b}_n\}$ e atendendo aos passos 4.3 e 4.2, temos

$$|f(u_n)| \leq |f(c_n)|. \tag{2.48}$$

Combinando (2.46), (2.47) e (2.48) vem

$$\widehat{b}_n - \widehat{a}_n \leq \frac{2}{\lambda_0} |f(c_n)|, \quad n \geq N_1.$$

Pelo lema 2.4.8(i) temos

$$|f(c_n)| \leq r_1 (b_n - a_n)^2 (b_{n-1} - a_{n-1}), \quad n \geq N_1,$$

donde

$$\widehat{b}_n - \widehat{a}_n \leq \frac{2}{\lambda_0} r_1 (b_n - a_n)^2 (b_{n-1} - a_{n-1}), \quad n \geq N_1. \tag{2.49}$$

Uma vez que a sucessão $\{(b_n - a_n)\}_{n=0}^{\infty}$ converge para zero, se N_1 é suficientemente grande teremos

$$\widehat{b}_n - \widehat{a}_n < \mu (b_n - a_n), \quad n \geq N_1,$$

pois acontecerá $\frac{2}{\lambda_0} r_1 (b_n - a_n)(b_{n-1} - a_{n-1}) < \mu$.
Assim, para $n \geq N_1$ tem-se

$$a_{n+1} = \widehat{a}_n \quad \text{e} \quad b_{n+1} = \widehat{b}_n \tag{2.50}$$

(veja-se passo 4.7).

Tomando

$$L_1 \geq \max \left\{ \frac{2}{\lambda_0} r_1, \frac{(b_{i+1} - a_{i+1})}{(b_i - a_i)^2 (b_{i-1} - a_{i-1})}; \; i = 1, \ldots, N_1 - 1 \right\}$$

e usando (2.49) e (2.50), temos o resultado pretendido

$$b_{n+1} - a_{n+1} \leq \widehat{b}_n - \widehat{a}_n \leq L_1 (b_n - a_n)^2 (b_{n-1} - a_{n-1}), \quad n = 1, 2, \ldots. \; \square \tag{2.51}$$

Corolário 2.4.10 (Alefeld, Potra e Yixun Shi [2]) *Considerem-se as condições do teorema 2.4.9. A sucessão* $\{\varepsilon_n\}_{n=0}^{\infty} = \{(b_n - a_n)\}_{n=0}^{\infty}$ *converge para zero com R-ordem de convergência pelo menos* $1 + \sqrt{2} = 2.4142\ldots$.

Demonstração. Este resultado segue directamente do corolário 2.3.2, dado que a sucessão $\{\varepsilon_n\}_{n=0}^{\infty}$ converge para zero e por (2.51) se tem

$$0 \leq \varepsilon_{n+1} \leq L_1 \varepsilon_n^2 \varepsilon_{n-1}, \quad n = 1, 2, \ldots.$$

Assim, a R-ordem de convergência de $\{\varepsilon_n\}_{n=0}^{\infty}$ é pelo menos igual à única solução positiva da equação $\lambda^2 = 2\lambda + 1$ que é igual a $1 + \sqrt{2}$. $\square$

Por fim, apresentamos o teorema que permite concluir sobre a R-ordem de convergência do algoritmo V.

Teorema 2.4.11 (Alefeld, Potra e Yixun Shi [2]) *Assumam-se as mesmas condições do teorema 2.4.9.*
Então a sucessão $\{(b_n - a_n)\}_{n=0}^{\infty}$ *produzida pelo algoritmo V converge para zero e existe uma constante* $L_2 > 0$ *tal que*

$$b_{n+1} - a_{n+1} \leq L_2 (b_n - a_n)^4 (b_{n-1} - a_{n-1}), \quad n = 1, 2, \ldots.$$

Mais ainda, existe um inteiro N_2 *tal que para* $n \geq N_2$ *se tem*

$$a_{n+1} = \widehat{a}_n \quad e \quad b_{n+1} = \widehat{b}_n.$$

Demonstração. A demonstração é semelhante à do teorema 2.4.9. Vamos igualmente assumir que $f'(x) \neq 0$ para $x \in [a, b]$.

Seja N_2 tal que $N_2 \geq \max\{n_2, n_3\}$, onde n_3 e n_2 são referidos nos lemas 2.4.7 e 2.4.8, respectivamente. Então, à semelhança do caso do algoritmo IV, tem-se

$$\widehat{b}_n - \widehat{a}_n \leq \frac{2}{\lambda_0} |f(\widetilde{c}_n)|, \quad n \geq N_2.$$

Pelo lema 2.4.8(ii) acontece

$$|f(\widetilde{c}_n)| \leq r_2 (b_n - a_n)^4 (b_{n-1} - a_{n-1}), \quad n \geq N_2,$$

e virá

$$\widehat{b}_n - \widehat{a}_n \leq \frac{2}{\lambda_0} r_2 (b_n - a_n)^4 (b_{n-1} - a_{n-1}), \quad n \geq N_2. \tag{2.52}$$

Usando o mesmo tipo de raciocínio que anteriormente, dado que a sucessão $\{(b_n - a_n)\}_{n=0}^{\infty}$ converge para zero, se N_2 é suficientemente grande teremos

$$\widehat{b}_n - \widehat{a}_n < \mu (b_n - a_n), \quad n \geq N_2,$$

e, assim, para $n \geq N_2$ ter-se-á

$$a_{n+1} = \widehat{a}_n \quad \text{e} \quad b_{n+1} = \widehat{b}_n. \tag{2.53}$$

Considerando

$$L_2 \geq \max\left\{ \frac{2}{\lambda_0} r_2, \frac{(b_{i+1} - a_{i+1})}{(b_i - a_i)^4 (b_{i-1} - a_{i-1})}; \; i = 1, \ldots, N_2 - 1 \right\}$$

e usando (2.52) e (2.53), temos o resultado desejado

$$b_{n+1} - a_{n+1} \leq \widehat{b}_n - \widehat{a}_n \leq L_2 (b_n - a_n)^4 (b_{n-1} - a_{n-1}), \quad n = 1, 2, \ldots. \; \square \tag{2.54}$$

Corolário 2.4.12 (Alefeld, Potra e Yixun Shi [2]) *Considerem-se as condições do teorema 2.4.11. A sucessão* $\{\varepsilon_n\}_{n=0}^{\infty} = \{(b_n - a_n)\}_{n=0}^{\infty}$ *converge para zero com R-ordem de convergência pelo menos* $2+\sqrt{5} = 4.2360\ldots$.

Demonstração. Uma vez que a sucessão $\{\varepsilon_n\}_{n=0}^{\infty}$ converge para zero e por (2.54) se tem

$$0 \leq \varepsilon_{n+1} \leq L_2 \varepsilon_n^4 \varepsilon_{n-1},, \quad n = 1, 2, \ldots,$$

podemos aplicar o corolário 2.3.2. Assim, a R-ordem de convergência da sucessão $\{\varepsilon_n\}_{n=0}^{\infty}$ é pelo menos igual à única solução positiva da equação $\lambda^2 = 4\lambda + 1$ que é igual a $2+\sqrt{5}$. □

2.4.3 Métodos VI e VII

Consideremos os algoritmos VI e VII (vejam-se pags. 25 e 28). Nesta secção iremos provar que a sucessão $\{(b_n - a_n)\}_{n=0}^{\infty}$ converge para zero com R-ordem de convergência pelo menos $1+\sqrt{3} = 2.7320\ldots$, no caso do algoritmo VI, e com R-ordem de convergência pelo menos $2+\sqrt{7} = 4.6457\ldots$, no caso do algoritmo VII.

Recorde-se que os métodos VI e VII constituem versões melhoradas dos métodos IV e V, respectivamente, e que as melhorias são conseguidas através do uso de interpolação cúbica inversa em vez de interpolação quadrática, sempre que possível. Vamos começar por verificar que impostas certas condições à função f, a interpolação cúbica inversa será sempre realizada a partir de uma dada iteração n.

Seja $y = f(x)$ uma função continuamente diferenciável num intervalo $[a,b]$ e considerem-se quatro pontos c, d, e e l pertencentes a $[a,b]$. Supondo que $f'(x) \neq 0$ para todo o $x \in [a,b]$, f é estritamente monótona em $[a,b]$ e existe a função inversa $x = f^{-1}(y)$. Assim, $f(c)$, $f(d)$, $f(e)$ e $f(l)$ serão todos distintos e poderemos construir o polinómio interpolador de f^{-1} em $(f(c),c)$, $(f(d),d)$, $(f(e),e)$ e $(f(l),l)$. Designando este polinómio por $p_{inverso}$, a sua expressão será dada por

$$\begin{aligned} p_{inverso}(y) \;=\; & c + f^{-1}[f(c),f(d)]\,(y-f(c)) \\ & + f^{-1}[f(c),f(d),f(e)]\,(y-f(c))(y-f(d)) \\ & + f^{-1}[f(c),f(d),f(e),f(l)]\,(y-f(c))(y-f(d))(y-f(e)). \end{aligned}$$

Se $f(a)\,f(b) < 0$ então existe um único zero simples x^* de f em $[a,b]$ e uma aproximação para x^* pode ser obtida calculando $\overline{x} = p_{inverso}(0)$. Recordando o procedimento *zero_p_inverso* (veja-se pag. 24), que foi especificamente desenvolvido para calcular o valor de um polinómio interpolador em 0, obtemos assim

$$x^* \approx \overline{x} = zero\_p_{inverso}\,(c,d,e,l) \in [a,b]\,.$$

Se assumirmos também que $f^{(4)}$ existe e é contínua em $[a,b]$, a fórmula para o erro de interpolação permite-nos escrever

$$\begin{aligned} |\overline{x} - x^*| &= \left|p_{inverso}(0) - f^{-1}(0)\right| \\ &\leq \frac{1}{4!} \max_{y \in f([a,b])} \left|\left[f^{-1}(y)\right]^{(4)}\right| |f(c)||f(d)||f(e)||f(l)| . \end{aligned} \quad (2.55)$$

Aplicando o conhecido teorema da derivada da função inversa temos $f^{-1'}(y) = \dfrac{1}{f'(x)}$ e, juntamente com outras regras de derivação adequadas, obtém-se

$$\left[f^{-1}(y)\right]^{(4)} = \frac{10f'(x)f''(x)f'''(x) - 15\left[f''(x)\right]^3 - \left[f'(x)\right]^2 f^{(4)}(x)}{\left[f'(x)\right]^7},$$

para $y \in f([a,b]) = \{z : z = f(x),\ x \in [a,b]\}$ com $x = f^{-1}(y) \in [a,b]$. Designando

$$\begin{aligned} M_1 &= \max_{x \in [a,b]} \left|f'(x)\right|, \\ M_2 &= \max_{x \in [a,b]} \left|f''(x)\right|, \\ M_3 &= \max_{x \in [a,b]} \left|f'''(x)\right|, \\ M_4 &= \max_{x \in [a,b]} \left|f^{(4)}(x)\right| \\ m_1 &= \min_{x \in [a,b]} \left|f'(x)\right|, \end{aligned} \quad (2.56)$$

deduz-se, a partir de (2.55), que

$$|\overline{x} - x^*| \leq M |f(c)||f(d)||f(e)||f(l)|, \quad (2.57)$$

onde $M = \dfrac{10M_1M_2M_3 + 15M_2^3 + M_1^2M_4}{24m_1^7}$.

Suponhamos agora que o algoritmo VI ou VII é aplicado à determinação de uma aproximação para o zero simples x^* de f em $[a,b]$. Seja $[a_n, b_n]$ o intervalo de inclusão de x^* com que se inicia a iteração n. Iremos provar a seguir que em ambos os algoritmos, a partir de n é suficientemente grande, acontecerá sempre que

$$f(a_n), f(b_n), f(d_n) \text{ e } f(e_n) \text{ são todos distintos}$$

e

$$c_n = zero\_p_{inverso}(a_n, b_n, d_n, e_n) \in [a_n, b_n],$$

onde d_n e e_n pertencem ao intervalo $[a_{n-1}, b_{n-1}]$ (vejam-se passos 6.1 do algoritmo VI e 7.1 do algoritmo VII).

Para o algoritmo VII teremos também que no passo 7.3

$$f(\widetilde{a}_n), f\left(\widetilde{b}_n\right), f\left(\widetilde{d}_n\right) \text{ e } f(\widetilde{e}_n) \text{ são todos distintos}$$

e

$$\widetilde{c}_n = zero\_p_{inverso}\left(\widetilde{a}_n, \widetilde{b}_n, \widetilde{d}_n, \widetilde{e}_n\right) \in \left[\widetilde{a}_n, \widetilde{b}_n\right].$$

Isto significa que nos passos referidos nunca se usará interpolação quadrática mas sempre interpolação cúbica inversa.

Observemos primeiro que o lema 2.4.7 referente aos algoritmos IV e V é também claramente aplicável aos algoritmos VI e VII.

Lema 2.4.13 (Alefeld, Potra e Yixun Shi [3]) *Seja f uma função continuamente diferenciável em $[a,b]$ tal que $f(a)f(b) < 0$ e seja x^* um zero simples de f em $[a,b]$. Suponha-se que o algoritmo VI (ou o algoritmo VII) não termina ao fim de um número finito de iterações.*
Então existe um inteiro positivo n_3 tal que para u_n e $\overline{c}_n$ definidos nos passos 6.3 *e* 6.4 *(ou em* 7.5 *e* 7.6*) satisfazem*

$$f(\overline{c}_n) f(u_n) < 0 \quad para \quad n \geq n_3.$$

Resultado correspondente ao lema 2.4.8 é apresentado no lema seguinte.

Lema 2.4.14 (Alefeld, Potra e Yixun Shi [3]) *Seja f uma função quatro vezes continuamente diferenciável em $[a,b]$ tal que $f(a)f(b) < 0$ e x^* um zero simples de f em $[a,b]$. Suponhamos que o algoritmo VI ou VII não termina ao fim de um número finito de iterações.*
Então

(*i*) *para o algoritmo VI, existe $l_1 > 0$ e n_1 tais que c_n no passo* 6.1 *será sempre obtido através da chamada a $zero\_p_{inverso}$ para $n \geq n_1$ e*

$$|f(c_n)| \leq l_1 (b_n - a_n)^2 (b_{n-1} - a_{n-1})^2, \quad n \geq n_1; \tag{2.58}$$

(*ii*) *para o algoritmo VII, existe $l_2 > 0$ e n_2 tais que c_n no passo* 7.1 *e $\widetilde{c}_n$ no passo* 7.3 *serão sempre obtidos através da chamada a $zero\_p_{inverso}$ para $n \geq n_2$ e*

$$|f(\widetilde{c}_n)| \leq l_2 (b_n - a_n)^4 (b_{n-1} - a_{n-1})^3, \quad n \geq n_2. \tag{2.59}$$

Demonstração. Pelos resultados anteriormente apresentados, temos

$$x^* \in [a_n, b_n] \quad \text{e} \quad \lim_{n\to\infty} (b_n - a_n) = 0.$$

(i) Tal como na demonstração do lema 2.4.8, dado que x^* é um zero simples de f , vamos assumir por simplicidade que $f'(x) \neq 0$ para todo o $x \in [a,b]$. Com isto, f é estritamente monótona em $[a,b]$ e, portanto, os valores $f(a_n)$, $f(b_n)$, $f(d_n)$ e $f(e_n)$ (veja-se passo 6.1) serão todos distintos. Assim, c_n será sempre obtido através do procedimento *zero_p*$_{inverso}$ desde que aconteça $c_n \in [a_n, b_n]$. Vamos provar que de facto existe um inteiro n_1 a partir do qual isto acontece.

A partir de (2.57) tem-se

$$|c_n - x^*| \leq M\,|f(a_n)|\,|f(b_n)|\,|f(d_n)|\,|f(e_n)|\,. \tag{2.60}$$

Como,

$$\left|\frac{f(a_n) - f(x^*)}{a_n - x^*}\right| = |f'(\xi)|\,, \quad \xi \in (a_n, b_n)\,,$$

é equivalente a escrever

$$|f(a_n)| = |f'(\xi)|\,|a_n - x^*|\,,$$

dado que $f(x^*) = 0$, vem

$$|f(a_n)| \leq M_1\,(b_n - a_n)\,, \tag{2.61}$$

onde M_1 está definido em (2.56).
De forma análoga se obtém

$$|f(b_n)| \leq M_1\,(b_n - a_n)\,. \tag{2.62}$$

Como d_n, $e_n \in [a_{n-1}, b_{n-1}]$, vem igualmente

$$\begin{aligned} |f(d_n)| &\leq M_1\,(b_{n-1} - a_{n-1}) \quad \text{e} \\ |f(e_n)| &\leq M_1\,(b_{n-1} - a_{n-1})\,. \end{aligned} \tag{2.63}$$

Desta forma, (2.60) conduz a

$$|c_n - x^*| \leq M\,(M_1)^4\,(b_n - a_n)^2\,(b_{n-1} - a_{n-1})^2\,. \tag{2.64}$$

Uma vez que $x^* \in (a,b)$, existe $\varepsilon > 0$ tal que

$$[x^* - \varepsilon, x^* + \varepsilon] \subset (a,b)$$

e, dado que $\lim_{n\to\infty} (b_n - a_n) = 0$, temos por (2.64) que existe um inteiro positivo $\overline{n}$ tal que

$$|c_n - x^*| \leq \varepsilon,$$

isto é,

$$c_n \in [x^* - \varepsilon, x^* + \varepsilon] \subset (a,b)\,, \quad n \geq \overline{n}$$

(note-se que ε fica fixo à partida, não depende de n). Assim, a desigualdade

$$|f(c_n)| \leq M_1 |c_n - x^*|\,, \tag{2.65}$$

verifica-se para $n \geq \overline{n}$. Se voltarmos a usar (2.60), pode obter-se

$$|f(c_n)| \leq M (M_1)^4 (b_n - a_n) (b_{n-1} - a_{n-1})^2 |f(a_n)|\,,$$

assim como

$$|f(c_n)| \leq M (M_1)^4 (b_n - a_n) (b_{n-1} - a_{n-1})^2 |f(b_n)|\,.$$

Novamente porque $\lim_{n\to\infty} (b_n - a_n) = 0$, podemos escolher $n_1 \geq \overline{n}$ tal que $M (M_1)^4 (b_n - a_n) (b_{n-1} - a_{n-1})^2 < 1$ para $n \geq n_1$, e virá

$$c_n \in (a,b) \quad \text{e} \quad |f(c_n)| < \min\{|f(a_n)|, |f(b_n)|\}\,, \quad n \geq n_1. \tag{2.66}$$

Uma vez que f é estritamente monótona em $[a,b]$ e $f(a_n) f(b_n) < 0$, (2.66) implica que

$$c_n \in [a_n, b_n]\,, \quad n \geq n_1,$$

o que permite concluir que c_n no passo 6.1 será sempre obtido através do procedimento *zero_p*$_{inverso}$.

O resultado em (2.58) segue imediatamente de (2.65) e de (2.64) com $l_1 = M (M_1)^5$.

(ii) Argumentos semelhantes aos acabados de apresentar podem ser aplicados para mostrar que existe um inteiro $n_2 \geq n_1$ tal que c_n no passo 7.1 e $\widetilde{c}_n$ no passo 7.3 serão sempre obtidos através do procedimento *zero_p*$_{inverso}$. Teremos, portanto, $c_n, \widetilde{c}_n \in [a_n, b_n]$ e poderemos escrever

$$\begin{aligned} |f(\widetilde{c}_n)| &\leq M_1 |\widetilde{c}_n - x^*| \\ &\leq M_1 M |f(\widetilde{a}_n)| \left|f\left(\widetilde{b}_n\right)\right| \left|f\left(\widetilde{d}_n\right)\right| |f(\widetilde{e}_n)| \\ &= M_1 M |f(a_n)| |f(b_n)| |f(c_n)| |f(d_n)|\,, \end{aligned}$$

observando que $\left\{\widetilde{a}_n, \widetilde{b}_n, \widetilde{d}_n, \widetilde{e}_n\right\} = \{a_n, b_n, c_n, d_n\}$ (veja-se passo 7.2). Usando de novo (2.61), (2.62) e (2.63) vem

$$|f(\widetilde{c}_n)| \leq M_1^4 M (b_n - a_n)^2 (b_{n-1} - a_{n-1}) |f(c_n)|\,.$$

Por fim, por (2.58),

$$|f(\widetilde{c}_n)| \leq l_2 (b_n - a_n)^4 (b_{n-1} - a_{n-1})^3\,, \quad n \geq n_2.$$

com $l_2 = M_1^4 M l_1 = (M_1)^9 M^2$. □

O teorema seguinte permitirá estabelecer que a R-ordem de convergência do algoritmo VI é pelo menos igual a $1+\sqrt{3}=2.7320\ldots$.

Teorema 2.4.15 (Alefeld, Potra e Yixun Shi [3]) *Considerem-se as mesmas condições do lema 2.4.14. A sucessão* $\{(b_n-a_n)\}_{n=0}^{\infty}$ *produzida pelo algoritmo VI converge para zero e existe uma constante* $R_1>0$ *tal que*

$$b_{n+1}-a_{n+1}\leq R_1\left(b_n-a_n\right)^2\left(b_{n-1}-a_{n-1}\right)^2,\quad n=1,2,\ldots. \tag{2.67}$$

Mais ainda, existe um inteiro N_1 *tal que para* $n\geq N_1$ *se tem*

$$a_{n+1}=\widehat{a}_n \quad e \quad b_{n+1}=\widehat{b}_n.$$

A demonstração deste teorema é praticamente o decalque da demonstração do teorema 2.4.9.

Demonstração. Assuma-se que $f'(x)\neq 0$ para todo o $x\in[a,b]$. Seja N_1 tal que $N_1\geq\max\{n_1,n_3\}$, onde n_3 e n_1 são referidos nos lemas 2.4.13 e 2.4.14, respectivamente. Seguindo exactamente a primeira parte da demonstração do teorema 2.4.9 chegamos a

$$\widehat{b}_n-\widehat{a}_n\leq\frac{2}{m_1}\left|f\left(c_n\right)\right|,\quad n\geq N_1,$$

onde m_1 ficou definido em (2.56).
Agora, pelo lema 2.4.14(i) tem-se

$$\widehat{b}_n-\widehat{a}_n\leq\frac{2}{m_1}l_1\left(b_n-a_n\right)^2\left(b_{n-1}-a_{n-1}\right)^2,\quad n\geq N_1. \tag{2.68}$$

Porque sucessão $\{(b_n-a_n)\}_{n=0}^{\infty}$ converge para zero, se N_1 é suficientemente grande teremos

$$\widehat{b}_n-\widehat{a}_n<\mu\left(b_n-a_n\right),\quad n\geq N_1,$$

pois acontecerá $\frac{2}{m_1}l_1\left(b_n-a_n\right)\left(b_{n-1}-a_{n-1}\right)^2<\mu$, com $\mu>0$ (veja-se passo 6.7 do algoritmo VI). Assim, para $n\geq N_1$ ter-se-á

$$a_{n+1}=\widehat{a}_n \quad \text{e} \quad b_{n+1}=\widehat{b}_n. \tag{2.69}$$

Escolhendo

$$R_1\geq\max\left\{\frac{2}{m_1}l_1,\frac{\left(b_{i+1}-a_{i+1}\right)}{\left(b_i-a_i\right)^2\left(b_{i-1}-a_{i-1}\right)^2};\ i=1,\ldots,N_1-1\right\}$$

e recorrendo a (2.68) e a (2.69), obtemos (2.67). □

Corolário 2.4.16 (Alefeld, Potra e Yixun Shi [3]) *Considerando as condições do teorema 2.4.15, a sucessão* $\{\varepsilon_n\}_{n=0}^{\infty}=\{(b_n-a_n)\}_{n=0}^{\infty}$ *converge para zero com R-ordem de convergência pelo menos* $1+\sqrt{3}=2.7320\ldots$.

Demonstração. Acabámos de provar com o teorema anterior que

$$0 \leq \varepsilon_{n+1} \leq R_1 \varepsilon_n^2 \varepsilon_{n-1}^2, \quad n = 1, 2, \ldots.$$

Assim, segue directamente do corolário 2.3.2 que a R-ordem de convergência de $\{\varepsilon_n\}_{n=0}^{\infty}$ é pelo menos igual a $1 + \sqrt{3}$ que é a única solução positiva da equação $\lambda^2 = 2\lambda + 2$. □

Falta apresentar os resultados que permitem concluir que a R-ordem de convergência do algoritmo VII é pelo menos $2 + \sqrt{7} = 4.6457\ldots$.

Teorema 2.4.17 (Alefeld, Potra e Yixun Shi [3]) *Considerem-se as mesmas condições do lema 2.4.14. A sucessão* $\{(b_n - a_n)\}_{n=0}^{\infty}$ *produzida pelo algoritmo VII converge para zero e existe uma constante* $R_2 > 0$ *tal que*

$$b_{n+1} - a_{n+1} \leq R_2 (b_n - a_n)^4 (b_{n-1} - a_{n-1})^3, \quad n = 1, 2, \ldots. \tag{2.70}$$

Mais ainda, existe um inteiro N_2 *tal que para* $n \geq N_2$ *se tem*

$$a_{n+1} = \widehat{a}_n \quad e \quad b_{n+1} = \widehat{b}_n.$$

A demonstração deste teorema é inteiramente idêntica à do teorema 2.4.15.

Demonstração. Vamos assumir que $f'(x) \neq 0$ para todo o $x \in [a, b]$. Seja N_2 tal que $N_2 \geq \max\{n_2, n_3\}$, onde n_3 e n_2 são referidos nos lemas 2.4.13 e 2.4.14, respectivamente. Mais uma vez, podemos afirmar que sendo

$$f(\overline{c}_n) f(u_n) < 0, \quad n \geq N_2,$$

a partir dos passos 7.7 e 7.8 e do facto que $u_n, \overline{c}_n \in [\overline{a}_n, \overline{b}_n]$ (vejam-se os passos 7.5 e 7.6), temos

$$\widehat{b}_n - \widehat{a}_n \leq |\overline{c}_n - u_n|, \quad n \geq N_2.$$

Pelos passos 7.5 e 7.6, dado que $\widetilde{c}_n \in \{\overline{a}_n, \overline{b}_n\}$, vem

$$\begin{aligned} |\overline{c}_n - u_n| &\leq \frac{2}{m_1} |f(u_n)| \\ &\leq \frac{2}{m_1} |f(\widetilde{c}_n)|, \quad n \geq N_2, \end{aligned} \tag{2.71}$$

onde m_1 ficou definido em (2.56).
Pelo lema 2.4.14(ii) segue-se

$$\widehat{b}_n - \widehat{a}_n \leq \frac{2}{m_1} l_2 (b_n - a_n)^4 (b_{n-1} - a_{n-1})^3, \quad n \geq N_2.$$

O resto da demonstração é semelhante à parte correspondente no teorema 2.4.15 e escolhendo

$$R_2 \geq \max\left\{ \frac{2}{m_1} l_2, \frac{(b_{i+1} - a_{i+1})}{(b_i - a_i)^4 (b_{i-1} - a_{i-1})^3}; \; i = 1, \ldots, N_2 - 1 \right\}$$

obtém-se o resultado pretendido em (2.70).□

Corolário 2.4.18 (Alefeld, Potra e Yixun Shi [3]) *Considerando as condições do teorema 2.4.17, a sucessão* $\{\varepsilon_n\}_{n=0}^{\infty} = \{(b_n - a_n)\}_{n=0}^{\infty}$ *converge para zero com R-ordem de convergência pelo menos* $2 + \sqrt{7} = 4.6457\ldots$.

Demonstração. Mais uma vez, dado que

$$0 \leq \varepsilon_{n+1} \leq R_2 \varepsilon_n^4 \varepsilon_{n-1}^3, , \qquad n = 1, 2, \ldots,$$

segue-se que a R-ordem de convergência de $\{\varepsilon_n\}_{n=0}^{\infty}$ é pelo menos igual a $2 + \sqrt{7}$, única solução positiva da equação $\lambda^2 = 4\lambda + 3$. □

Fica assim concluido o estudo sobre a ordem de convergência dos métodos I a VII. Na próxima secção iremos referir-nos à eficiência computacional que cada método apresenta.

2.5 Eficiência Computacional

A eficiência computacional de um método iterativo é determinada fundamentalmente pelo esforço computacional envolvido no cálculo de cada iteração e pela rapidez de convergência para a solução. Embora possamos comparar métodos iterativos com base apenas na sua ordem de convergência, faz mais sentido comparar a sua *eficiência computacional* pois, ao considerar o trabalho computacional total no cálculo da solução, é um conceito mais completo.

Com o objectivo de chegar a uma definição de eficiência computacional seguimos uma sugestão apresentada em Ralston [23, pags. 336 e 337] introduzindo, no entanto, algumas adaptações que consideramos convenientes.

2.5.1 Definição de índice de eficiência

Consideremos dois métodos iterativos 1 e 2 para determinar uma aproximação de uma raíz α da equação $f(x) = 0$, começando com o mesmo valor inicial x_0. Suponhamos que as sucessões

$$\varepsilon_n^{(1)} = |x_n - \alpha| \quad \text{e} \quad \varepsilon_n^{(2)} = |x_n - \alpha|, \qquad n = 0, 1, \ldots,$$

correspondentes aos métodos 1 e 2, convergem para zero com Q-ordem de convergência pelo menos $p_1 > 1$ e $p_2 > 1$, respectivamente, isto é, existem constantes c_1, $c_2 > 0$ tais que

$$\varepsilon_{n+1}^{(1)} \leq c_1 \left(\varepsilon_n^{(1)}\right)^{p_1}, \qquad n = 0, 1, \ldots,$$

num caso, e

$$\varepsilon_{n+1}^{(2)} \leq c_2 \left(\varepsilon_n^{(2)}\right)^{p_2}, \qquad n = 0, 1, \ldots,$$

noutro.

Para n suficientemente grande, podemos considerar válidas as aproximações

$$\varepsilon_{n+1}^{(1)} = c_1 \left(\varepsilon_n^{(1)}\right)^{p_1} \tag{2.72}$$

e

$$\varepsilon_{n+1}^{(2)} = c_2 \left(\varepsilon_n^{(2)}\right)^{p_2}, \tag{2.73}$$

dado que $\varepsilon_n^{(1)}$ e $\varepsilon_n^{(2)}$ convergem para zero.
Sejam

$$S_n = -\ln \varepsilon_n^{(1)} \quad \text{e} \quad T_n = -\ln \varepsilon_n^{(2)}, \quad n = 0, 1, \ldots.$$

Então

$$\begin{aligned} S_{n+1} &= -\ln \varepsilon_{n+1}^{(1)} \\ &= -\ln \left[c_1 \left(\varepsilon_n^{(1)}\right)^{p_1}\right] \\ &= -\ln c_1 - p_1 \ln \varepsilon_n^{(1)} \\ &= -\ln c_1 + p_1 S_n, \quad n = 0, 1, \ldots, \end{aligned} \tag{2.74}$$

e, analogamente,

$$T_{n+1} = -\ln c_2 + p_2 T_n, \quad n = 0, 1, \ldots.$$

Vamos mostrar por indução que

$$S_n = S_0 p_1^n - \ln c_1^{\frac{p_1^n - 1}{p_1 - 1}}, \quad n = 1, \ldots.. \tag{2.75}$$

Para $n = 1$, vem por (2.74)

$$S_1 = -\ln c_1 + p_1 S_0,$$

que corresponde exactamente à expressão em (2.75) quando $n = 1$.
Suponhamos agora que (2.75) é válido para n. Então,

$$\begin{aligned} S_{n+1} &= -\ln c_1 + p_1 S_n \\ &= -\ln c_1 + p_1 \left(S_0 p_1^n - \ln c_1^{\frac{p_1^n - 1}{p_1 - 1}}\right) \\ &= -\ln c_1 + S_0 p_1^{n+1} - \ln c_1^{\frac{p_1^{n+1} - p_1}{p_1 - 1}} \\ &= S_0 p_1^{n+1} - \ln c_1^{\frac{p_1^{n+1} - 1}{p_1 - 1}}, \end{aligned}$$

o que prova o resultado pretendido.

De igual forma, temos

$$T_n = T_0 p_2^n - \ln c_2^{\frac{p_2^n - 1}{p_2 - 1}}, \quad n = 1, \ldots. \tag{2.76}$$

Uma vez que estamos a considerar que ambos os métodos começam no mesmo ponto inicial x_0, vem

$$S_0 = -\ln \varepsilon_0^{(1)} = -\ln \varepsilon_0^{(2)} = T_0. \tag{2.77}$$

Sejam I e J o número de iterações necessárias para alcançar a precisão desejada, respectivamente, para os métodos 1 e 2. Então, $\varepsilon_I^{(1)}$ e $\varepsilon_J^{(2)}$ podem ser considerados *iguais* e, portanto, também S_I e T_J. Usando as fórmulas em (2.75) e em (2.76) assim como (2.77), de $S_I = T_J$ temos

$$S_0 p_1^I - \ln c_1^{\frac{p_1^I - 1}{p_1 - 1}} = S_0 p_2^J - \ln c_2^{\frac{p_2^J - 1}{p_2 - 1}},$$

ou seja,

$$S_0 \left(p_1^I - p_2^J\right) + \ln \frac{c_2^{(p_2^J - 1)/(p_2 - 1)}}{c_1^{(p_1^I - 1)/(p_1 - 1)}} = 0. \tag{2.78}$$

Se θ e ϕ representam respectivamente a quantidade de cálculo necessário por iteração nos métodos 1 e 2, então o custo computacional total é θI e ϕJ, respectivamente. Embora θ e ϕ possam ser estimados a partir das fórmulas iterativas dos métodos, (2.78) não nos dá uma forma óbvia de comparar I e J. Pode normalmente assumir-se que o segundo termo em (2.78) é pequeno comparado com o primeiro (o que acontece, por exemplo, se c_1 e c_2 são ambos próximos de 1). Neste caso, tem-se

$$S_0 \left(p_1^I - p_2^J\right) \approx 0,$$

ou

$$p_1^I \approx p_2^J,$$

ou ainda,

$$\frac{I}{1/\ln p_1} \approx \frac{J}{1/\ln p_2}. \tag{2.79}$$

Designando as quantidades em (2.79) por k , temos

$$\theta I \approx \theta \frac{1}{\ln p_1} k \quad \text{e} \quad \phi J \approx \phi \frac{1}{\ln p_2} k.$$

Assim, para um método cuja Q-ordem de convergência é pelo menos $p > 1$ e cujo custo por iteração é θ, uma medida do custo total pode ser dada por

$$\theta \frac{1}{\ln p}.$$

O **índice de eficiência**, que será o inverso do custo, pode ser definido como

$$IndEf = \frac{\ln p}{\theta},$$

ou, usando uma definição mais comum,

$$IndEf = p^{\frac{1}{\theta}},$$

já que $e^{\frac{\ln p}{\theta}} = p^{\frac{1}{\theta}}$.

É importante notar que uma vez que as aproximações em (2.72) e em (2.73) só se verificam quando já se está perto da raíz α, o índice de eficiência mede apenas o quanto o método é bom quando está próximo de convergir, o que significa que se trata de uma noção assimptótica de convergência. Na verdade, em aplicações práticas, métodos com índices de eficiência inferiores podem ter um desempenho melhor do que métodos com índices de eficiência superiores. Isto deve-se ao facto de na prática se parar ao fim de um determinado número de iterações. De qualquer forma, dados dois métodos iterativos e exigindo-se uma dada precisão na solução, podemos dizer que o método com maior índice de eficiência terá de realizar menor quantidade de cálculo para atingir essa precisão.

Em geral, a maior percentagem do esforço computacional dependerá do número de avaliações da função f e das suas derivadas que cada iteração requer, e não entrará em conta com as operações aritméticas necessárias para combinar essas quantidades de acordo com a fórmula iterativa. Desta forma, dado um método iterativo com Q-ordem de convergência pelo menos $p > 1$ e com o cálculo assimptótico de q valores de f (ou das suas derivadas) por iteração, iremos usar para índice de eficiência o valor dado por

$$IndEfic = p^{\frac{1}{q}}. \tag{2.80}$$

Conjectura-se que quando um método tem R-ordem de convergência pelo menos p, o índide de eficiência se define da mesma forma.

2.5.2 Índice de eficiência dos métodos I a VII

De acordo com a definição em (2.80), para estabelecermos os índices de eficiência dos métodos I a VII é necessário contarmos o número de avaliações da função f que ocorre assimptoticamente por iteração em cada um dos métodos.

Se em cada iteração se guardarem os valores de f que vão sendo calculados, cada chamada ao procedimento *partir_intervalo*1 ou *partir_intervalo*2 exige apenas o cálculo de um novo valor de f. Sendo assim, é fácil verificar que o cálculo assimptótico dos valores de f por iteração corresponde ao número de chamadas ao procedimento *partir_intervalo*1 ou *partir_intervalo*2

que se realiza assimptoticamente. Recorde-se que já se fez referência a este assunto quando se descreveram os métodos.

Com os resultados já conhecidos sobre a Q-ordem ou R-ordem de convergência dos métodos, fica então possível calcular o índice de eficiência dos mesmos.

Método I

Invocando o teorema 2.4.3, para $n \geq n_1$ o método I não realiza a chamada a *partir_intervalo*1 no último passo de cada iteração, pelo que requer assimptoticamente o cálculo de 2 valores da função f por iteração. Como a sucessão $\{(b_n - a_n)\}_{n=0}^{\infty}$ converge para zero pelo menos Q-quadraticamente, temos que o índice de eficiência do método I é dado por

$$IndEfic_I = \sqrt{2} = 1.4142\ldots.$$

Método II

Pelo teorema 2.4.4, o método II requer o cálculo assimptótico de três valores de f e a sucessão $\{(b_n - a_n)\}_{n=0}^{\infty}$ converge para zero com Q-ordem de convergência pelo menos 4. Assim, o índice de eficiência do método II é

$$IndEfic_{II} = \sqrt[3]{4} = 1.5874\ldots.$$

Método III

O teorema 2.4.5 estabelece que a R-ordem de convergência do método III é pelo menos $0.5\left(3+\sqrt{13}\right)$ e, sendo necessário o cálculo de três valores de f por iteração, o índice de eficiência deste método é

$$IndEfic_{III} = \sqrt[3]{0.5\left(3+\sqrt{13}\right)} = 1.4892\ldots.$$

Método IV

Usando o teorema 2.4.9 e o corolário 2.4.10, sabemos que o método IV tem R-ordem de convergência pelo menos $1+\sqrt{2}$ e exige assimptoticamente o cálculo de dois valores de f por iteração. Assim, o índice de eficiência tem o valor

$$IndEfic_{IV} = \sqrt{1+\sqrt{2}} = 1.5537\ldots.$$

Método V

Sabemos, pelo teorema 2.4.11 e pelo corolário 2.4.12, que o método V calcula assimptoticamente 3 valores de f e tem R-ordem de convergência pelo menos $2+\sqrt{5}$, pelo que o seu índice de eficiência é

$$IndEfic_V = \sqrt[3]{2+\sqrt{5}} = 1.6180\ldots.$$

Método VI

Pelo teorema 2.4.15 e corolário 2.4.16, o método VI requer assimptoticamente apenas dois valores de f por iteração e tem R-ordem de convergência pelo menos $1+\sqrt{3}$. Assim, o índice de eficiência do método VI é dado por

$$IndEfic_{VI} = \sqrt{1+\sqrt{3}} = 1.6528\ldots.$$

Método VII

Usando o teorema 2.4.17 e corolário 2.4.18, o método VII requer o cálculo assimptótico de três valores de f por iteração e possui R-ordem de convergência pelo menos $2+\sqrt{7}$, pelo que o seu índice de eficiência é

$$IndEfic_{VII} = \sqrt[3]{2+\sqrt{7}} = 1.6685\ldots.$$

2.6 Métodos V e VII - dois métodos óptimos

Nesta secção iremos mostrar que os métodos V e VII (vejam-se pags. 21 e 28) são considerados métodos óptimos no sentido em que apresentam o maior índice de eficiência possível dentro de uma determinada classe de métodos. Vamos primeiro referirmo-nos ao método V e depois ao método VII.

Como vimos, o índice de eficiência do método V é 1.6180... e superior ao do método IV que é 1.5537.... É imediato verificar que esta melhoria é conseguida através da repetição dos passos 5.1 e 5.2 nos passos 5.3 e 5.4, que constitui a única alteração introduzida ao método IV para se obter o método V. Ou seja, se em cada iteração realizarmos interpolação quadrática duas vezes e não apenas uma, calculando para isso mais um valor da função f, a eficiência computacional do método é melhorada. E se repetirmos a interpolação quadrática um número arbitrário de vezes? É de esperar que haja um número óptimo.

Apresentamos a seguir o algoritmo $\mathbf{V_k}$ que resulta do algoritmo V com a repetição dos passos 5.1 e 5.2 num total k de vezes.

Algoritmo V_k

$[a_0, b_0] = [a, b]$;

$c_0 = a_0 - f\,[a_0, b_0]^{-1} f(a_0)$;

$\{[a_1, b_1], d_1\} = partir\_intervalo2\,([a_0, b_0], c_0)$;

para $n = 1, 2, \ldots$ **até** $n = nmax$ **fazer**

1.1 $c_n = newton\_quadrático\,(a_n, b_n, d_n, 2)$;

1.2 $\left\{\left[a_n^{(1)}, b_n^{(1)}\right], d_n^{(1)}\right\} = partir\_intervalo2\,([a_n, b_n], c_n)$;

1.3 $c_n^{(1)} = newton\_quadrático\left(a_n^{(1)}, b_n^{(1)}, d_n^{(1)}, 3\right)$;

1.4 $\left\{\left[a_n^{(2)}, b_n^{(2)}\right], d_n^{(2)}\right\} = partir\_intervalo2\left([a_n^{(1)}, b_n^{(1)}], c_n^{(1)}\right)$;

$\vdots$

$\vdots$

$1.2k-1$ $\widetilde{c}_n = newton\_quadrático\left(a_n^{(k-1)}, b_n^{(k-1)}, d_n^{(k-1)}, k+1\right)$;

$1.2k$ $\left\{\left[\overline{a}_n, \overline{b}_n\right], \overline{d}_n\right\} = partir\_intervalo2\left([a_n^{(k-1)}, b_n^{(k-1)}], \widetilde{c}_n\right)$;

$1.2k+1$ **se** $|f(\overline{a}_n)| < \left|f(\overline{b}_n)\right|$ **então** $u_n = \overline{a}_n$;
senão $u_n = \overline{b}_n$;

$1.2k+2$ $\overline{c}_n = u_n - 2f\left[\overline{a}_n, \overline{b}_n\right]^{-1} f(u_n)$;

$1.2k+3$ **se** $|\overline{c}_n - u_n| > 0.5(\overline{b}_n - \overline{a}_n)$ **então** $\widehat{c}_n = 0.5\left(\overline{b}_n + \overline{a}_n\right)$;
senão $\widehat{c}_n = \overline{c}_n$;

$1.2k+4$ $\left\{\left[\widehat{a}_n, \widehat{b}_n\right], \widehat{d}_n\right\} = partir\_intervalo2\left(\left[\overline{a}_n, \overline{b}_n\right], \widehat{c}_n\right)$;

$1.2k+5$ **se** $\widehat{b}_n - \widehat{a}_n < \mu\,(b_n - a_n)$

então $\{[a_{n+1}, b_{n+1}], d_{n+1}\} = \left\{\left[\widehat{a}_n, \widehat{b}_n\right], \widehat{d}_n\right\}$;

senão $\{[a_{n+1}, b_{n+1}], d_{n+1}\} = partir\_intervalo2\left(\left[\widehat{a}_n, \widehat{b}_n\right], 0.5\left(\widehat{b}_n + \widehat{a}_n\right)\right)$;

fpara

Os métodos IV e V são casos particulares do método $\mathrm{V_k}$ em que $k = 1$ e $k = 2$, respectivamente. Com os resultados que se seguem vamos mostrar que a escolha $k = 2$ é a que conduz a um índice de eficiência mais elevado, o que significa que o método V pode ser visto como um método óptimo.

Lema 2.6.1 *Seja f uma função três vezes continuamente diferenciável em $[a,b]$ tal que $f(a)f(b) < 0$ e seja x^* um zero simples de f em $[a,b]$. Suponhamos que o algoritmo V_k não termina ao fim de um número finito de iterações.*
Então existe uma constante $r_k > 0$ e n_k tais que

$$|f(\widetilde{c}_n)| \leq r_k (b_n - a_n)^{2k} (b_{n-1} - a_{n-1}), \quad n \geq n_k \quad com \quad k = 1, 2, \ldots, \tag{2.81}$$

onde $\widetilde{c}_n$ está definido no passo $1.2k - 1$.

Este lema é uma generalização do lema 2.4.8 e estão, portanto, provados os caso em que $k = 1$ ou $k = 2$. Para obtermos uma demonstração para $k > 2$ vamos usar o mesmo tipo de argumentos que foram usados anteriormente, e por isso serão omitidos alguns detalhes.

Demonstração. Seja $k > 2$. Pelo lema 2.4.8(ii) e com os devidos ajustes de notação, temos

$$\left|f\left(c_n^{(1)}\right)\right| \leq r_2 (b_n - a_n)^4 (b_{n-1} - a_{n-1}), \quad n \geq n_2, \tag{2.82}$$

com $r_2 > 0$ e onde $c_n^{(1)}$ está definido no passo 1.3 do algoritmo $\mathrm{V_k}$.
Seguindo exactamente os mesmos passos da demonstração do lema 2.4.8(ii) com referência às mesmas constantes, podemos escrever

$$\left|f(z_n^{(2)})\right| \leq \lambda_3 (b_n - a_n)^2 \left|f(c_n^{(1)})\right|, \quad n \geq n_3, \tag{2.83}$$

onde $z_n^{(2)}$ representa o único zero do polinómio $p_{\left(a_n^{(2)}, b_n^{(2)}, d_n^{(2)}\right)} \equiv p_{\left(a_n^{(1)}, b_n^{(1)}, c_n^{(1)}\right)}$ em $\left[a_n^{(2)}, b_n^{(2)}\right]$ e $n_3 \geq n_2$.
Continuando de forma inteiramente análoga, sendo $c_n^{(2)}$ a aproximação para o zero $z_n^{(2)}$ obtida através de *newton_quadrático* $\left(a_n^{(2)}, b_n^{(2)}, d_n^{(2)}, 4\right)$ podemos obter

$$\left|c_n^{(2)} - z_n^{(2)}\right| \leq \lambda_5 (b_n - a_n)^6 (b_{n-1} - a_{n-1}), \tag{2.84}$$

onde $\lambda_5 = \left(\frac{1}{2\delta} \max_{a \leq x \leq b} |f''(x)|\right)^{15}$.
Finalmente, dado que

$$\left|f\left(c_n^{(2)}\right)\right| \leq \left(\max_{a \leq x \leq b} |f'(x)|\right) \left|c_n^{(2)} - z_n^{(2)}\right| + \left|f\left(z_n^{(2)}\right)\right|,$$

se combinarmos (2.82), (2.83) e (2.84) vem

$$\left|f\left(c_n^{(2)}\right)\right| \leq r_3 \left(b_n - a_n\right)^6 \left(b_{n-1} - a_{n-1}\right), \quad n \geq n_3,$$

onde $r_3 = \lambda_5 \max\limits_{a \leq x \leq b} |f'(x)| + \lambda_3 r_2 > 0$.
Aplicando sucessivamente estes passos, obtemos

$$\left|f\left(c_n^{(3)}\right)\right| \leq r_4 \left(b_n - a_n\right)^8 \left(b_{n-1} - a_{n-1}\right), \quad n \geq n_4 \geq n_3,$$

onde $r_4 = \lambda_6 \max\limits_{a \leq x \leq b} |f'(x)| + \lambda_3 r_3 > 0$ e $\lambda_6 = \left(\frac{1}{2\delta} \max\limits_{a \leq x \leq b} |f''(x)|\right)^{31}$,

...

$$|f(\widetilde{c}_n)| = \left|f\left(c_n^{(k-1)}\right)\right| \leq r_k \left(b_n - a_n\right)^{2+2(k-1)} \left(b_{n-1} - a_{n-1}\right), \quad n \geq n_k \geq n_{k-1},$$

onde $r_k = \lambda_{k+2} \max\limits_{a \leq x \leq b} |f'(x)| + \lambda_3 r_{k-1} > 0$ e $\lambda_{k+2} = \left(\frac{1}{2\delta} \max\limits_{a \leq x \leq b} |f''(x)|\right)^{2^{k+1}-1}$.
□

O teorema seguinte generaliza, como se esperava, o resultado apresentado no teorema 2.4.11.

Teorema 2.6.2 *Considerem-se as mesmas condições do lema anterior. A sucessão* $\{(b_n - a_n)\}_{n=0}^{\infty}$ *produzida pelo algoritmo* V_k *converge para zero e existe uma constante* $L_k > 0$ *tal que*

$$b_{n+1} - a_{n+1} \leq L_k \left(b_n - a_n\right)^{2k} \left(b_{n-1} - a_{n-1}\right), \quad n = 1, 2, \ldots.$$

Mais ainda, existe um inteiro N_3 *tal que para* $n \geq N_3$ *se tem*

$$a_{n+1} = \widehat{a}_n \quad e \quad b_{n+1} = \widehat{b}_n.$$

Demonstração. Comecemos por verificar que o lema 2.4.7 se aplica igualmente ao método V_k com $k > 2$. Seja então N_3 tal que $N_3 \geq \max\{n_k, n_3\}$, onde n_k e n_3 são referidos nos lemas 2.6.1 e 2.4.7, respectivamente. Se percorrermos os passos da demonstração do teorema 2.4.11 com as alterações convenientes e por fim aplicarmos (2.81), obteremos o resultado pretendido, considerando

$$L_k \geq \max\left\{\frac{2}{\lambda_0} r_k, \frac{(b_{i+1} - a_{i+1})}{(b_i - a_i)^{2k} (b_{i-1} - a_{i-1})};\ i = 1, \ldots, N_3 - 1\right\}.$$

O resto da demonstração é análoga à parte correspondente no teorema 2.4.11 e será omitida. □

Corolário 2.6.3 *Considerem-se as mesmas condições do teorema 2.6.2. A sucessão* $\{\varepsilon_n\}_{n=0}^{\infty} = \{(b_n - a_n)\}_{n=0}^{\infty}$ *produzida pelo algoritmo* V_k *converge para zero com R-ordem de convergência pelo menos* $k + \sqrt{1+k^2}$ *para* $k = 1, 2, \ldots$.

Demonstração. Basta observar pelo teorema anterior que sendo

$$0 \leq \varepsilon_{n+1} \leq L_k \varepsilon_n^{2k} \varepsilon_{n-1},, \quad n = 1, 2, \ldots,$$

a R-ordem de convergência de $\{\varepsilon_n\}_{n=0}^{\infty}$ é pelo menos igual a $\tau = k + \sqrt{1+k^2}$ que é a única raíz positiva da equação $\lambda^2 - 2k\lambda - 1 = 0$ $(k = 1, 2, \ldots)$. □

Uma vez que o algoritmo V_k requer assimptoticamente $k+1$ avaliações da função f por iteração, o seu índice de eficiência é dado por

$$IndEfic_{V_k} = \left(k + \sqrt{1+k^2}\right)^{\frac{1}{k+1}}, \quad k \geq 1.$$

O lema seguinte mostra que $\left(k + \sqrt{1+k^2}\right)^{\frac{1}{k+1}} < \left(2 + \sqrt{1+2^2}\right)^{\frac{1}{3}}, k \neq 2$, o que significa que o valor máximo de $IndEfic_{V_k}$ é conseguido para $k = 2$.

Lema 2.6.4 *Seja* $I_n = \left(n + \sqrt{1+n^2}\right)^{\frac{1}{n+1}}$ *para* $n \in \mathbb{N}$. *Então* $I_2 > I_n$ *para* $n \neq 2$.

A demonstração é simples e será feita por indução.

Demonstração. Sendo $I_2 = 1.618\ldots$, $I_1 = 1.553\ldots$ e $I_3 = 1.575\ldots$, temos, portanto, $I_2 > I_1$ e $I_2 > I_3$.

Suponhamos que $I_n < I_2$, para $n \geq 3$, isto é,

$$n + \sqrt{1+n^2} < I_2^{n+1}, \quad n \geq 3. \tag{2.85}$$

Como $2n < n^2$ e $n + 1 < \sqrt{2}n$ para $n \geq 3$, vem

$$\begin{aligned} n + 1 + \sqrt{1 + (n+1)^2} &= n + 1 + \sqrt{2 + n^2 + 2n} \\ &< n + 1 + \sqrt{2}\sqrt{1+n^2} \\ &< \sqrt{2}\left(n + \sqrt{1+n^2}\right). \end{aligned}$$

Utilizando a hipótese em (2.85) e dado que $\sqrt{2} < I_2$ segue-se

$$\sqrt{2}\left(n + \sqrt{1+n^2}\right) < \sqrt{2} I_2^{n+1} < I_2^{n+2}.$$

Logo $\left(n + \sqrt{1+n^2}\right)^{\frac{1}{n+1}} < I_2$ para $n \neq 2$. □

Assim, o mais elevado índice de eficiência computacional possível para este tipo de algoritmos será $\left(2+\sqrt{5}\right)^{\frac{1}{3}}$ que é alcançado com $k=2$, isto é, com a implementação do algoritmo V.

A verificação de que o método VII é também um método óptimo, segue o mesmo tipo de abordagem que acabou de ser feita para o método V. Analogamente, se compararmos os índices de eficiência dos métodos VI e VII, 1.6528... e 1.6685..., respectivamente, é imediato constatar que o método VII melhora a eficiência computacional do método VI por conter a repetição dos passos 7.1 e 7.2 nos passos 7.3 e 7.4. Isto significa que realizar interpolação cúbica inversa por duas vezes em cada iteração constitui um esquema mais eficiente do que realizar apenas uma vez, embora isso exija mais um valor da função f por iteração.

O algoritmo $\mathbf{VII_k}$ que apresentamos de seguida obtém-se do algoritmo VII repetindo um número k de vezes os passos 7.1 e 7.2. Iremos mostrar que o caso em que $k=2$ é o caso óptimo, isto é, conduz a um algoritmo com o índice de eficiência computacional mais elevado.

Algoritmo $\mathbf{VII_k}$

$[a_0,b_0]=[a,b]$;

$c_0=a_0-f[a_0,b_0]^{-1}f(a_0)$;

$\{[a_1,b_1],d_1\}=partir\_intervalo2([a_0,b_0],c_0)$;

para $n=1,2,\ldots$ **até** $n=nmax$ **fazer**

1.1 **se** $n=1$ **ou** $f(a_n)$, $f(b_n)$, $f(d_n)$ e $f(e_n)$ não são todos distintos
então $c_n=newton\_quadrático(a_n,b_n,d_n,2)$;
senão $c_n=zero\_p_{inverso}(a_n,b_n,d_n,e_n)$;
se $c_n\notin[a_n,b_n]$ **então** $c_n=newton\_quadrático(a_n,b_n,d_n,2)$;

1.2 $e_n^{(1)}=d_n$;
$\left\{[a_n^{(1)},b_n^{(1)}],d_n^{(1)}\right\}=partir\_intervalo2([a_n,b_n],c_n)$;

$\vdots$

$\vdots$

$1.2k-2$ $e_n^{(k-1)} = d_n^{(k-2)}$;

$\left\{[a_n^{(k-1)}, b_n^{(k-1)}], d_n^{(k-1)}\right\} = partir\_intervalo2\left([a_n^{(k-2)}, b_n^{(k-2)}], c_n^{(k-2)}\right)$;

$1.2k-1$ **se** $f(a_n^{(k-1)})$, $f(b_n^{(k-1)})$, $f(d_n^{(k-1)})$ e $f(e_n^{(k-1)})$ não são todos distintos

então $\widetilde{c}_n = newton\_quadrático\left(a_n^{(k-1)}, b_n^{(k-1)}, d_n^{(k-1)}, k+1\right)$;

senão $\widetilde{c}_n = zero\_p_{inverso}\left(a_n^{(k-1)}, b_n^{(k-1)}, d_n^{(k-1)}, e_n^{(k-1)}\right)$;

se $\widetilde{c}_n \notin \left[a_n^{(k-1)}, b_n^{(k-1)}\right]$

então $\widetilde{c}_n = newton\_quadrático\left(a_n^{(k-1)}, b_n^{(k-1)}, d_n^{(k-1)}, k+1\right)$;

$1.2k$ $\left\{\left[\overline{a}_n, \overline{b}_n\right], \overline{d}_n\right\} = partir\_intervalo2\left([a_n^{(k-1)}, b_n^{(k-1)}], \widetilde{c}_n\right)$

$1.2k+1$ **se** $|f(\overline{a}_n)| < |f(\overline{b}_n)|$ **então** $u_n = \overline{a}_n$;

senão $u_n = \overline{b}_n$;

$1.2k+2$ $\overline{c}_n = u_n - 2f\left[\overline{a}_n, \overline{b}_n\right]^{-1} f(u_n)$;

$1.2k+3$ **se** $|\overline{c}_n - u_n| > 0.5(\overline{b}_n - \overline{a}_n)$ **então** $\widehat{c}_n = 0.5\left(\overline{b}_n + \overline{a}_n\right)$;

senão $\widehat{c}_n = \overline{c}_n$;

$1.2k+4$ $\left\{\left[\widehat{a}_n, \widehat{b}_n\right], \widehat{d}_n\right\} = partir\_intervalo2\left(\left[\overline{a}_n, \overline{b}_n\right], \widehat{c}_n\right)$;

$1.2k+5$ **se** $\widehat{b}_n - \widehat{a}_n < \mu\left(b_n - a_n\right)$

então $\{[a_{n+1}, b_{n+1}], d_{n+1}\} = \left\{\left[\widehat{a}_n, \widehat{b}_n\right], \widehat{d}_n\right\}$;

$e_{n+1} = \overline{d}_n$;

senão $\{[a_{n+1}, b_{n+1}], d_{n+1}\} = partir\_intervalo2\left(\left[\widehat{a}_n, \widehat{b}_n\right], 0.5\left(\widehat{b}_n + \widehat{a}_n\right)\right)$;

$e_{n+1} = \widehat{d}_n$.

fpara

Como facilmente se verifica, os métodos VI e VII são casos particulares do método $\mathbf{VII_k}$, com $k = 1$ e $k = 2$, respectivamente. Os resultados a seguir evidenciam que o método VII é de facto o método óptimo.

Lema 2.6.5 *Seja f uma função quatro vezes continuamente diferenciável em $[a,b]$ tal que $f(a)f(b)<0$ e x^* uma raíz simples de $f(x)=0$ em $[a,b]$. Suponhamos que o algoritmo VII_k com $k \geq 2$ não termina ao fim de um número finito de iterações.*
Então existe $l_k > 0$ e n_k tais que c_n no passo 1.1, $c_n^{(1)}$ no passo 1.3, ..., $c_n^{(k-1)} = \widetilde{c}_n$ no passo $1.2k-1$ serão sempre obtidos através da chamada a $zero\_p_{inverso}$ para $n \geq n_k$ e

$$|f(\widetilde{c}_n)| \leq l_k (b_n - a_n)^{3k-2} (b_{n-1} - a_{n-1})^3, \quad n \geq n_k. \tag{2.86}$$

Este lema generaliza o resultado apresentado na alínea (ii) do lema 2.4.14 referente ao método VII. Assim, o caso corresponde a $k=2$ já está provado e a demonstração para $k>2$ seguirá os mesmos passos. Alguns detalhes ficarão por incluir por serem idênticos aos já apresentados.

Demonstração. Veja-se a demonstração da alínea (i) do lema 2.4.14 e assumam-se as mesmas condições. Podemos usar argumentos semelhantes para mostrar a primeira parte deste lema, isto é, que existe um inteiro $n_k \geq n_{k-1} \geq \ldots \geq n_2 \geq n_1$ tal que, para $n \geq n_k$, acontece

$$c_n \in [a_n, b_n], c_n^{(1)} \in \left[a_n^{(1)}, b_n^{(1)}\right], \ldots, c_n^{(k-1)} \in \left[a_n^{(k-1)}, b_n^{(k-1)}\right],$$

e, como se assume que f é estritamente monótona em $[a,b]$, estão reunidas as condições para que c_n no passo 1.1, $c_n^{(1)}$ no passo 1.3, ..., $c_n^{(k-1)} = \widetilde{c}_n$ no passo $1.2k-1$ sejam sempre obtidos através da chamada a $zero\_p_{inverso}$ para $n \geq n_k$.

Para obtermos (2.86) vamos seguir os mesmos passos da demonstração do lema 2.4.14 (ii), mencionando as mesmas constantes. Considere-se $k>2$ e $n \geq n_k$. Pelo apresentado no lema 2.4.14(ii) temos

$$\left|f\left(c_n^{(1)}\right)\right| \leq l_2 (b_n - a_n)^4 (b_{n-1} - a_{n-1})^3, \quad n \geq n_k, \tag{2.87}$$

com $l_2 > 0$ e $c_n^{(1)}$ definido no passo 1.3 do algoritmo $\mathrm{VII_k}$.
Como $c_n^{(2)} \in [a_n, b_n]$ temos

$$\begin{aligned}
\left|f\left(c_n^{(2)}\right)\right| &\leq M_1 \left|c_n^{(2)} - x^*\right| \\
&\leq M_1 M \left|f\left(a_n^{(2)}\right)\right| \left|f\left(b_n^{(2)}\right)\right| \left|f\left(d_n^{(2)}\right)\right| \left|f\left(e_n^{(2)}\right)\right| \\
&= M_1 M \left|f\left(a_n^{(1)}\right)\right| \left|f\left(b_n^{(1)}\right)\right| \left|f\left(c_n^{(1)}\right)\right| \left|f\left(d_n^{(1)}\right)\right|.
\end{aligned}$$

Uma vez que $a_n^{(1)}, b_n^{(1)}$ e $d_n^{(1)}$ pertencem a $[a_n, b_n]$, vem

$$\left|f\left(c_n^{(2)}\right)\right| \leq M_1^4 M (b_n - a_n)^3 \left|f\left(c_n^{(1)}\right)\right|, \quad n \geq n_k$$

(notar que $d_n \notin [a_n, b_n]$ mas $d_n^{(1)}, d_n^{(2)}, \ldots, d_n^{(k-1)} \in [a_n, b_n]$).
Agora, usando (2.87) ficamos com

$$\left|f\left(c_n^{(2)}\right)\right| \leq l_3 (b_n - a_n)^7 (b_{n-1} - a_{n-1})^3, \quad n \geq n_k,$$

onde $l_3 = M_1^4 M l_2$.
Seguindo sucessivamente estes passos, obtemos

$$\left|f\left(c_n^{(3)}\right)\right| \leq l_4 (b_n - a_n)^{10} (b_{n-1} - a_{n-1})^3, \quad n \geq n_k,$$

onde $l_4 = M_1^4 M l_3$,

$$\ldots$$

$$|f(\widetilde{c}_n)| = \left|f\left(c_n^{(k-1)}\right)\right| \leq l_k (b_n - a_n)^{[4+3(k-3)]+3} (b_{n-1} - a_{n-1})^3, \quad n \geq n_k,$$

onde $l_k = M_1^4 M l_{k-1}$. □

Do teorema seguinte poderemos concluir qual a R-ordem de convergência do algoritmo VII_k.

Teorema 2.6.6 *Considerem-se as mesmas condições do lema anterior. A sucessão* $\{(b_n - a_n)\}_{n=0}^{\infty}$ *produzida pelo algoritmo* VII_k *com* $k \geq 2$ *converge para zero e existe uma constante* $R_k > 0$ *tal que*

$$b_{n+1} - a_{n+1} \leq R_k (b_n - a_n)^{3k-2} (b_{n-1} - a_{n-1})^3, \quad n = 1, 2, \ldots.$$

Mais ainda, existe um inteiro N_3 *tal que para* $n \geq N_3$ *se tem*

$$a_{n+1} = \widehat{a}_n \quad e \quad b_{n+1} = \widehat{b}_n.$$

Demonstração. É imediato verificar que o lema 2.4.13 é aplicável também ao método VII_k para $k > 2$. Considere-se então N_3 tal que $N_3 \geq \max\{n_k, n_3\}$, onde n_k e n_3 são referidos nos lemas 2.6.5 e 2.4.13, respectivamente. Introduzindo as modificações adequadas, podemos seguir os passos da demonstração do teorema 2.4.17 e aplicar o lema 2.6.5. Escolhendo

$$R_k \geq \max\left\{\frac{2}{m_1} l_k, \frac{(b_{i+1} - a_{i+1})}{(b_i - a_i)^{3k-2} (b_{i-1} - a_{i-1})^3};\ i = 1, \ldots, N_3 - 1\right\},$$

obtém-se o resultado desejado.

O que falta para completar esta demonstração é semelhante à parte correspondente no teorema 2.4.17. □

Como corolário deste teorema segue que o algoritmo $\mathrm{VII_k}$ com $k \geq 2$ tem R-ordem de convergência pelo menos $\tau = \frac{3k-2}{2} + \sqrt{3 + \left(\frac{3k-2}{2}\right)^2}$, que é a única raíz positiva da equação $\lambda^2 - (3k-2)\,\lambda - 3 = 0$.

Corolário 2.6.7 *Considerem-se as mesmas condições do teorema 2.6.6. A sucessão* $\{\varepsilon_n\}_{n=0}^{\infty} = \{(b_n - a_n)\}_{n=0}^{\infty}$ *produzida pelo algoritmo* VII_k *converge para zero com R-ordem de convergência pelo menos* $\frac{3k-2}{2} + \sqrt{3 + \left(\frac{3k-2}{2}\right)^2}$ *para* $k = 2, 3, \ldots$.

É igualmente fácil verificar que o algoritmo $\mathrm{VII_k}$ requer assimptoticamente $k+1$ avaliações da função f por iteração. Assim, o seu índice de eficiência é dado por

$$IndEfic_{VII_k} = \left(\frac{3k-2}{2} + \sqrt{3 + \left(\frac{3k-2}{2}\right)^2}\right)^{\frac{1}{k+1}}, \quad k \geq 2.$$

Pelo lema que vamos mostrar a seguir, $IndEf_{VII_k}$ tem valor máximo quando $k = 2$.

Lema 2.6.8 *Seja* $I_n = \left(\frac{3n-2}{2} + \sqrt{3 + \left(\frac{3n-2}{2}\right)^2}\right)^{\frac{1}{n+1}}$ *para* $n \in I\!N$. *Então* $I_2 > I_n$ *para* $n > 2$.

Demonstração. Acontece $I_2 > I_3$ pois $I_2 = 1.668\ldots$ e $I_3 = 1.649\ldots$.

Suponhamos que $I_n < I_2$, para $n \geq 3$, isto é,

$$\frac{3n-2}{2} + \sqrt{3 + \left(\frac{3n-2}{2}\right)^2} < I_2^{n+1}, \quad n \geq 3. \tag{2.88}$$

Como $3n + 1 < \sqrt{2}\,(3n-2)$ é válido para $n > 3$, temos

$$\begin{aligned} \frac{3n+1}{2} + \sqrt{3 + \left(\frac{3n+1}{2}\right)^2} &< \frac{\sqrt{2}(3n-2)}{2} + \sqrt{3 + \left(\frac{\sqrt{2}(3n-2)}{2}\right)^2} \\ &< \frac{\sqrt{2}(3n-2)}{2} + \sqrt{6 + 2\left(\frac{3n-2}{2}\right)^2} \\ &= \sqrt{2}\left(\frac{3n-2}{2} + \sqrt{3 + \left(\frac{3n-2}{2}\right)^2}\right) \end{aligned}$$

Aplicando agora a hipótese em (2.88) e o facto de que $\sqrt{2} < I_2$, tem-se

$$\sqrt{2}\left(\frac{3n-2}{2} + \sqrt{3 + \left(\frac{3n-2}{2}\right)^2}\right) < \sqrt{2} I_2^{n+1} < I_2^{n+2}.$$

Assim, $\left(\frac{3n-2}{2} + \sqrt{3 + \left(\frac{3n-2}{2}\right)^2}\right)^{\frac{1}{n+1}} < I_2, \quad n > 2.$ □

O método VII é portanto, entre todos os métodos apresentados, o que tem o mais elevado índice de eficiência computacional.

Capítulo 3

Resultados Numéricos

Neste capítulo apresentaremos alguns exemplos numéricos de aplicação dos métodos I a VII à resolução de uma equação não linear. Com a apresentação destes exemplos teremos apenas o intuito de ilustrar a convergência dos métodos e não pretenderemos realizar um estudo comparativo do comportamento dos mesmos. Isso foi já exaustivamente concretizado pelos autores Alefeld, Potra e Yixun Shi em [3].

Os resultados das várias experiências numéricas exibidos em [3, pags. 342 e 343] confirmam os índices de eficiência dos métodos I a VII. Deve, no entanto, fazer-se uma ressalva em relação ao método II uma vez que com os métodos III e IV se obtêm em geral melhores resultados do que com aquele. Essas experiências mostram igualmente que o método VII evidencia o melhor comportamento de todos os métodos, especialmente quando a tolerância exigida é *pequena,* o que aliás está de acordo com o facto do índice de eficiência ser um conceito assimptótico de convergência.

O desempenho dos métodos I a VII foi também comparado com o desempenho de outros métodos tais como os eficientes métodos de Dekker e Brent. Embora possam apresentar um comportamento pior em alguns problemas, foi igualmente encorajador para os autores verificarem que o comportamento prático dos métodos I a VII pode considerar-se comparável ao comportamento dos métodos de Dekker e Brent. Poderemos mesmo até afirmar que sobretudo os métodos VI e VII apresentam alguma vantagem sobre os métodos de Dekker e Brent.

As experiências referidas foram realizadas com um conjunto de diversos problemas teste - um total de 154 problemas. Para a tolerância *tol* testaram-se diferentes valores ($tol = 10^{-7}$, 10^{-10}, 10^{-15} e 0) e para os parâmetros μ e λ foi sempre escolhido $\mu = 0.5$ e $\lambda = 0.7$. Todos os algoritmos foram implementados em Fortran e foi usado um computador AT&T 3B2-1000 Modelo 80, com precisão dupla.

Dada a estrutura dos algoritmos, claramente se compreende que não faz sentido comparar o número de iterações efectuadas por cada algoritmo. Daí que as conclusões acabadas de apresentar se baseam na comparação do número total de avaliações das funções necessário para se atingir uma determinada precisão nas soluções de todos os problemas teste.

Passaremos então de seguida a apresentar os problemas que seleccionamos para exemplificar o desempenho dos algoritmos I a VII. A implementação dos algoritmos foi feita na linguagem de programação do *Mathematica* num Pentium 233 Mhz MMX. Antes, porém, iremos referir-nos a alguns pormenores relativos a esta implementação assim como a algumas características do *Mathematica*.

3.1 Implementação dos algoritmos no programa *Mathematica*

O *Mathematica* é um sistema algébrico computacional que permite também fazer cálculo numérico, representações gráficas e inclui uma linguagem de programação própria. Todas estas capacidades podem ser usadas de modo integrado e a interacção com o este sistema é relativamente simples e bastante agradável.

3.1.1 Números e precisão no *Mathematica*

O *Mathematica* define precisão de uma quantidade numérica como o número de dígitos decimais significativos do número. No caso de inteiros, racionais e símbolos considera-os de precisão infinita.

Os números reais são representados aproximadamente e distinguem-se dois tipos de números reais aproximados: números de precisão fixa (chamada precisão de máquina) e números de precisão arbitrária. Números de precisão fixa contêm um número finito e fixo de dígitos. Números de precisão arbitrária podem conter qualquer número de dígitos.

Os reais de precisão de máquina são representados internamente no sistema de vírgula flutuante de precisão dupla. Nos cálculos com números de vírgula flutuante o *Mathematica* usa as capacidades especiais do *hardware* e a principal vantagem é a rapidez de execução.

As variáveis $MachinePrecision e $MachineEpsilon indicam respectivamente o número de dígitos decimais de precisão nos números de precisão de máquina e a precisão relativa da máquina. No computador que usamos os valores correspondentes a essas variáveis são $MachinePrecision=16 e $MachineEpsilon $= 2.220445 \times 10^{-16}$.

A representação de números e a aritmética de precisão arbitrária é implementada em *software*. Está definida uma variável global $MaxPrecision que estabelece um limite superior na precisão de números que o *Mathematica*

pode usar. O valor atribuído a esta variável é 50 000. Para definir o limite inferior na precisão arbitrária destes números existe também a variável $MinPrecision cujo valor atribuído é 0.

Em cálculos com números de precisão arbitrária o *Mathematica* controla a precisão dos resultados em cada operação apresentando apenas os dígitos que se sabem serem correctos face à precisão dos dados. Para fazer isto por vezes é necessário executar as operações intermédias internas com maior precisão. A variável global $MaxExtraPrecision especifica quantos dígitos adicionais devem ser permitidos em tais cálculos intermédios. O valor que lhe está atribuido é 50. A computação com números de precisão arbitrária é assim bastante mais lenta do que com números de precisão de máquina.

Os resultados de cálculos envolvendo somente números de precisão de máquina têm também precisão de máquina, não havendo garantia que todos os algarismos sejam significativos. Os resultados de cálculos envolvendo apenas números de precisão arbitrária são também números de precisão arbitrária. Se determinado cálculo envolve números de precisão fixa e, ou valores de precisão arbitrária o resultado é um número de precisão fixa.

Fazendo $MaxPrecision = $MinPrecision = n podemos forçar a que todos os números de precisão arbitrária tenham uma precisão fixa de n dígitos. De facto, o que isto faz é obrigar o *Mathematica* a tratar os números de precisão arbitrária do mesmo modo que trata os números de precisão de máquina, mas com mais dígitos de precisão, isto é, simulam-se os cálculos numa máquina com n dígitos de precisão. Computação com precisão fixa conseguida desta forma pode tornar os cálculos mais eficientes, mas sem uma análise cuidada não se pode ter a certeza de quantos dígitos estão correctos no resultado que se obtém.

Todas estas características foram tidas em conta na implementação dos algoritmos I a VII e tentou-se explorar o melhor possível as suas vantagens.

3.1.2 Alguns pormenores de implementação dos algoritmos

Alguns detalhes na implementação dos algoritmos I a VII merecem ser mencionados por serem aplicáveis em situações em que se pretende evitar a perda considerável de algarismos significativos e também por incluirem algumas funções específicas do *Mathematica*.

- O ponto c de bissecção do intervalo $[a, b]$ é calculado por

$$c = a + \frac{b-a}{2}$$

em vez da forma convencional

$$c = \frac{a+b}{2},$$

seguindo o príncipio geral de que é sempre preferível usar fórmulas que expressem a quantidade desejada como uma pequena correcção de outra quantidade.

- Para o cálculo do ponto secante

$$c = a - \frac{b-a}{f(b) - f(a)} f(a)$$

usamos a fórmula mais segura

$$c = a - (b-a) \frac{f(a)}{f(b)} \Big/ \left(1 - \frac{f(a)}{f(b)}\right),$$

quando $|f(a)| \leq |f(b)|$, ou

$$c = b - (a-b) \frac{f(b)}{f(a)} \Big/ \left(1 - \frac{f(b)}{f(a)}\right),$$

no caso em que $|f(b)| < |f(a)|$.

O mesmo rearranjo das fórmulas é considerado para o cálculo do *ponto secante duplo*

$$\overline{c} = u - 2\frac{\overline{b} - \overline{a}}{f(\overline{b}) - f(\overline{a})} f(u) \quad \text{para } u \in \{\overline{a}, \overline{b}\},$$

assim como para as quantidades Q_{ij} $(j = 1,2,3;\ i = j, j+1, 3)$ e D_{ij} $(j = 1,2,3;\ i = j+1, 3)$ no procedimento $zero\_p_{inverso}$.

- Para construir o polinómio interpolador quadrático p_2 no conjunto de pontos $(a, f_a), (b, f_b)$ e (c, f_c) com $f_a f_b < 0$, tal como para o cálculo exacto do zero z deste polinómio pertencente ao intervalo $[a, b]$, usamos funçoes já definidas no *Mathematica* para esse fim. Temos assim que p_2 será obtido por

$$p_2[x\_] = InterpolatingPolynomial[\{\{a, f_a\}, \{b, f_b\}, \{c, f_c\}\}, x]$$

e z através de

$$Solve[p[x] == 0, x].$$

- Para a construção do procedimento *newton_quadrático* recorremos à função *FixedPoint* também definida no *Mathematica*. Usando a aproximação inicial $x_0 \in \{a, b\}$, a aproximação $x_k \approx z$ após k iterações do método de Newton é conseguida com

$$x_k = FixedPoint\left[x - p_2[x]/p_2'[x], x_0, k\right],$$

interpertando o método de Newton como um método iterativo do Ponto Fixo.

3.2 Exemplos Numéricos

Para a resolução dos problemas que apresentaremos nesta secção optamos também pela escolha $\mu = 0.5$ e $\lambda = 0.7$. Para a tolerância *tol* usamos o valor mínimo $tol = 0$ e os cálculos foram realizados em precisão de máquina.

Exemplo 3.2.1 *O polinómio* $p(x) = 4x^{10} - 3x^6 + 4x^3 - x^4 + 10x - 3$ *tem um zero positivo simples* r *pertencente ao intervalo* $[0, 1]$. *Para a inclusão deste zero, começando com o intervalo inicial* $[a_0, b_0] = [0, 1]$ *obtiveram-se os seguintes resultados:*

Método I

$$\begin{aligned}
[a_1, b_1] &= [0.2142857142857143, 0.3233244755079274] \\
[a_2, b_2] &= [0.2904417749029658, 0.2916234305145753] \\
[a_3, b_3] &= [0.2910372720661937, 0.2910374433881488] \\
[a_4, b_4] &= [0.2910373577394956, 0.2910373577394992] \\
[a_5, b_5] &= [0.2910373577394974, 0.2910373577394974]
\end{aligned}$$

Número de avaliações de p: 12.

Método II

$$\begin{aligned}
[a_1, b_1] &= [0.2852239762556765, 0.2933859223735257] \\
[a_2, b_2] &= [0.2910373577333796, 0.2910373577456081] \\
[a_3, b_3] &= [0.2910373577394974, 0.2910373577394974]
\end{aligned}$$

Número de avaliações de p: 9.

Método III

$$\begin{aligned}
[a_1, b_1] &= [0.2666861556101159, 0.3129053428428521] \\
[a_2, b_2] &= [0.2910372692748982, 0.2910374452491657] \\
[a_3, b_3] &= [0.2910373577394974, 0.2910373577394974]
\end{aligned}$$

Número de avaliações de p: 11.

Método IV

$$\begin{aligned}
[a_1, b_1] &= [0.2904060062218650, 0.2916465111393461] \\
[a_2, b_2] &= [0.2910373529855352, 0.2910373624920377] \\
[a_3, b_3] &= [0.2910373577394974, 0.2910373577394974]
\end{aligned}$$

Número de avaliações de p: 9.

Método V

$$\begin{aligned}[a_1, b_1] &= [0.2910216669409782, 0.2910524969975664] \\ [a_2, b_2] &= [0.2910373577394974, 0.2910373577394974]\end{aligned}$$

Número de avaliações de p: 8.

Método VI

$$\begin{aligned}[a_1, b_1] &= [0.2904060062218650, 0.2916465111393461] \\ [a_2, b_2] &= [0.2910373577173989, 0.2910373577615892] \\ [a_3, b_3] &= [0.2910373577394974, 0.2910373577394974]\end{aligned}$$

Número de avaliações de p: 8.

Método VII

$$\begin{aligned}[a_1, b_1] &= [0.2910358637284803, 0.2910388513036857] \\ [a_2, b_2] &= [0.2910373577394974, 0.2910373577394974]\end{aligned}$$

Número de avaliações de p: 7.

Se aplicarmos os algoritmos usando uma precisão arbitrária de 45 dígitos verificamos que os 16 *dígitos na aproximação*

$$r \approx 0.2910373577394974$$

são todos significativos.

Exemplo 3.2.2 *A equação*

$$g(x) = \frac{1}{2}\log\left(\frac{1}{100} + x^2\right) + \arctan(10x) - \frac{\pi}{2} = 0$$

tem uma raiz simples r no intervalo $[1, 2]$. *Com o intervalo inicial* $[a_0, b_0] = [1, 2]$ *obtivemos os seguintes resultados:*

Método I

$$\begin{aligned}[a_1, b_1] &= [1.064057334871195, 1.128114669742390] \\ [a_2, b_2] &= [1.090650761011914, 1.091615775555831] \\ [a_3, b_3] &= [1.091126653444566, 1.091126881078098] \\ [a_4, b_4] &= [1.091126767234820, 1.091126767234833] \\ [a_5, b_5] &= [1.091126767234826, 1.091126767234826]\end{aligned}$$

Número de avaliações de g: 11.

Método II

$$\begin{aligned}[a_1, b_1] &= [1.090573624862357, 1.091734911747263] \\ [a_2, b_2] &= [1.091126767234809, 1.091126767234844] \\ [a_3, b_3] &= [1.091126767234826, 1.091126767234826]\end{aligned}$$

Número de avaliações de g: 9.

Método III

$$\begin{aligned}[a_1, b_1] &= [1.085985895506722, 1.096778395144883] \\ [a_2, b_2] &= [1.091126764910473, 1.091126769570936] \\ [a_3, b_3] &= [1.091126767234826, 1.091126767234826]\end{aligned}$$

Número de avaliações de g: 10.

Método IV

$$\begin{aligned}[a_1, b_1] &= [1.090575536828470, 1.091732809682600] \\ [a_2, b_2] &= [1.091126756972451, 1.091126777503284] \\ [a_3, b_3] &= [1.091126767234826, 1.091126767234826]\end{aligned}$$

Número de avaliações de g: 8.

Método V

$$\begin{aligned}[a_1, b_1] &= [1.091126158068559, 1.091127436980172] \\ [a_2, b_2] &= [1.091126767234826, 1.091126767234826]\end{aligned}$$

Número de avaliações de g: 9.

Método VI

$$\begin{aligned}[a_1, b_1] &= [1.090575536828470, 1.091732809682600] \\ [a_2, b_2] &= [1.091126767188606, 1.091126767281074] \\ [a_3, b_3] &= [1.091126767234826, 1.091126767234826]\end{aligned}$$

Número de avaliações de g: 8.

Método VII

$$\begin{aligned}[a_1, b_1] &= [1.091126710568544, 1.091126829536338] \\ [a_2, b_2] &= [1.091126767234826, 1.091126767234826]\end{aligned}$$

Número de avaliações de g: 7.

Com uma precisão arbitrária de 45 dígitos, é possível verificar que

$$r \approx 1.0911267672348262117$$

com 20 dígitos de precisão, o permite concluir que todos os dígitos nos intervalos finais acima são correctos .

Exemplo 3.2.3 *A equação seguinte, exemplo extraído de [3],*

$$f(x) = -2\sum_{i=1}^{20} \frac{(2i-5)^2}{x-i^2} = 0$$

tem 19 raízes r_i, com r_i entre i^2 e $(i+1)^2$ $(i = 1, \ldots, 19)$. Embora estas raízes sejam simples, a função f apresenta um comportamento difícil devido às singularidades nos pontos $x_i = i^2$ e ao facto da segunda derivada se anular entre i^2 e $(i+1)^2$ $(i = 1, \ldots, 19)$.

Pretendemos obter uma aproximação para a raiz r_2 entre 4 e 9. Com o intervalo inicial $[a_0, b_0] = \left[4 + 10^{-4}, 9 - 10^{-4}\right]$ obtivemos os resultados que se apresentam de seguida.

Método I

$$\begin{aligned}
[a_1, b_1] &= [6.500000000000549, 8.999900000000000] \\
[a_2, b_2] &= [6.500000000000808, 7.749950000000404] \\
[a_3, b_3] &= [6.670496766194921, 7.210223383097663] \\
[a_4, b_4] &= [6.680895960445692, 6.685369243477806] \\
[a_5, b_5] &= [6.683752636320451, 6.683754484693690] \\
[a_6, b_6] &= [6.683753560807906, 6.683753560808250] \\
[a_7, b_7] &= [6.683753560808074, 6.683753560808078]
\end{aligned}$$

Número de avaliações de f: 18.

Método II

$$\begin{aligned}
[a_1, b_1] &= [6.500000000000635, 8.999900000000000] \\
[a_2, b_2] &= [6.501506653503350, 7.750703326751675] \\
[a_3, b_3] &= [6.681491747976976, 7.216097537364325] \\
[a_4, b_4] &= [6.683753185310698, 6.683753770132070] \\
[a_5, b_5] &= [6.683753560808074, 6.683753560808078]
\end{aligned}$$

Número de avaliações de f: 19.

Método III

$$\begin{aligned}
[a_1, b_1] &= [6.500000000000260, 8.999900000000000] \\
[a_2, b_2] &= [6.665007736418061, 7.749949999998908] \\
[a_3, b_3] &= [6.665007736418061, 6.702146847146303] \\
[a_4, b_4] &= [6.683753531033791, 6.683753590353396] \\
[a_5, b_5] &= [6.683753560808077, 6.683753560808079]
\end{aligned}$$

Número de avaliações de f: 17.

Método IV

$$\begin{aligned}
[a_1, b_1] &= [6.500000000000635, 7.749950000000318] \\
[a_2, b_2] &= [6.665007736418024, 7.207478868208559] \\
[a_3, b_3] &= [6.665007736418024, 6.701520143586745] \\
[a_4, b_4] &= [6.683735119663583, 6.683771869912038] \\
[a_5, b_5] &= [6.683753560804993, 6.683753560811163] \\
[a_6, b_6] &= [6.683753560808074, 6.683753560808078]
\end{aligned}$$

Número de avaliações de f: 17.

Método V

$$\begin{aligned}
[a_1, b_1] &= [6.656573628462709, 6.813147256924958] \\
[a_2, b_2] &= [6.683753012987743, 6.683754072056632] \\
[a_3, b_3] &= [6.683753560808078, 6.683753560808082]
\end{aligned}$$

Número de avaliações de f: 12.

Método VI

$$\begin{aligned}
[a_1, b_1] &= [6.500000000000635, 7.749950000000318] \\
[a_2, b_2] &= [6.682835013783069, 7.216392506891694] \\
[a_3, b_3] &= [6.683752726478106, 6.683754025618520] \\
[a_4, b_4] &= [6.683753560808078, 6.683753560808082]
\end{aligned}$$

Número de avaliações de f: 13.

Método VII

$$\begin{aligned}
[a_1, b_1] &= [6.682839340556704, 6.684630026172250] \\
[a_2, b_2] &= [6.683753560808076, 6.683753560808080]
\end{aligned}$$

Número de avaliações de f: 9.

Se para cada intervalo final considerarmos o ponto médio, trata-se de uma aproximação para r_2 com 15 ou 16 dígitos correctos, pois, seleccionando uma precisão arbitrária de 100 dígitos, é possível saber que a aproximação

$$r_2 \approx 6.6837535608080780814$$

tem 20 algarismos significativos.

Deve observar-se que a inclusão nos intervalos finais da raiz exacta dos problemas apresentados se verifica a menos de erros de arredondamento.

O próximo exemplo pretende mostrar que embora os métodos I a VII possam ser aplicados ao cálculo de uma raiz não simples (mas de multiplicidade ímpar), nestes casos podem ter um comportamento bastante pobre.

Exemplo 3.2.4 *A seguinte função*

$$h(x) = 8\sin(x) + 8x - 8\pi$$

tem um zero em $x = \pi$ *com multiplicidade três, isto é,* $h(\pi) = 0$, $h'(\pi) = 0$, $h''(\pi) = 0$ *e* $h'''(\pi) \neq 0$. *Começando com o intervalo inicial* $[a_0, b_0] = [3.1, 3.2]$ *vejamos como se comportam os métodos.*

Método I

$$\begin{aligned}
[a_1, b_1] &= [3.129009357636714, 3.164504678818357] \\
[a_2, b_2] &= [3.136148685731457, 3.150326682274907] \\
[a_3, b_3] &= [3.139553671249750, 3.144940176762329] \\
[a_4, b_4] &= [3.140806774657345, 3.142873475709837] \\
[a_5, b_5] &= [3.141292445701568, 3.142082960705702] \\
[a_6, b_6] &= [3.141477664650806, 3.141780312678254] \\
[a_7, b_7] &= [3.141548672513368, 3.141664492595811] \\
[a_8, b_8] &= [3.141575613364345, 3.141620052980078] \\
[a_9, b_9] &= [3.141584501287491, 3.141584501287491]
\end{aligned}$$

Número de avaliações de h: 27.

Método II

$$\begin{aligned}
[a_1, b_1] &= [3.129694561114365, 3.164847280557183] \\
[a_2, b_2] &= [3.136459533347272, 3.150653406952228] \\
[a_3, b_3] &= [3.139583287582339, 3.145118347267283] \\
[a_4, b_4] &= [3.140810947578233, 3.142964647422758] \\
[a_5, b_5] &= [3.141288452832100, 3.142126550127429] \\
[a_6, b_6] &= [3.141474263592845, 3.141800406860137] \\
[a_7, b_7] &= [3.141546577274426, 3.141673492067282] \\
[a_8, b_8] &= [3.141574533305357, 3.141624012686319] \\
[a_9, b_9] &= [3.141581601788352, 3.141581601788352]
\end{aligned}$$

Número de avaliações de h: 35.

Método III

$$\begin{aligned}
[a_1, b_1] &= [3.100000000000000, 3.149026582073565] \\
[a_2, b_2] &= [3.128418259576993, 3.142749188492327] \\
[a_3, b_3] &= [3.137259769705270, 3.142544298760013] \\
[a_4, b_4] &= [3.141335406234587, 3.142544298760013] \\
[a_5, b_5] &= [3.141335406234587, 3.141797147212124] \\
[a_6, b_6] &= [3.141567473806139, 3.141797147212124] \\
[a_7, b_7] &= [3.141576948287509, 3.141659060459387] \\
[a_8, b_8] &= [3.141594994981600, 3.141594994981600]
\end{aligned}$$

Número de avaliações de h: 26.

Método IV

$$\begin{aligned}
[a_1, b_1] &= [3.129698357051064, 3.164849178525532] \\
[a_2, b_2] &= [3.129698357051064, 3.142766723215205] \\
[a_3, b_3] &= [3.136216656587453, 3.142734956123841] \\
[a_4, b_4] &= [3.140083602181071, 3.142489839392020] \\
[a_5, b_5] &= [3.141439375402337, 3.142489839392020] \\
[a_6, b_6] &= [3.141452230635881, 3.141971035013951] \\
[a_7, b_7] &= [3.141541354326400, 3.141710895206392] \\
[a_8, b_8] &= [3.141583902557472, 3.141583902557472]
\end{aligned}$$

Número de avaliações de h: 24.

Método V

$$\begin{aligned}
[a_1, b_1] &= [3.133491356819307, 3.166745678409653] \\
[a_2, b_2] &= [3.133491356819307, 3.142783430805059] \\
[a_3, b_3] &= [3.137656581487524, 3.141821806155741] \\
[a_4, b_4] &= [3.139933668601941, 3.141811880279214] \\
[a_5, b_5] &= [3.141396694347796, 3.141604287313505] \\
[a_6, b_6] &= [3.141591172420537, 3.141591172420537]
\end{aligned}$$

Número de avaliações de h: 23.

Método VI

$$\begin{aligned}
[a_1, b_1] &= [3.129698357051064, 3.164849178525532] \\
[a_2, b_2] &= [3.136039082738311, 3.150444130631922] \\
[a_3, b_3] &= [3.139552443822693, 3.144998287227307] \\
[a_4, b_4] &= [3.140797761792289, 3.142898024509798] \\
[a_5, b_5] &= [3.141288806504419, 3.142093415507109] \\
[a_6, b_6] &= [3.141475996685579, 3.141784706096344] \\
[a_7, b_7] &= [3.141547833073559, 3.141666269584952] \\
[a_8, b_8] &= [3.141575223867734, 3.141620746726343] \\
[a_9, b_9] &= [3.141581188559851, 3.141581188559851]
\end{aligned}$$

Número de avaliações de h: 28.

Método VII

$$\begin{aligned}
[a_1, b_1] &= [3.132691799119504, 3.166345899559752] \\
[a_2, b_2] &= [3.137177710959511, 3.151761805259631] \\
[a_3, b_3] &= [3.139500017489695, 3.145630911374663] \\
[a_4, b_4] &= [3.140677517360420, 3.143154214367541] \\
[a_5, b_5] &= [3.141225859822983, 3.142190037095262] \\
[a_6, b_6] &= [3.141451423237323, 3.141820730166293] \\
[a_7, b_7] &= [3.141538700503475, 3.141679715334884] \\
[a_8, b_8] &= [3.141571420984795, 3.141625568159840] \\
[a_9, b_9] &= [3.141580950935227, 3.141580950935227]
\end{aligned}$$

Número de avaliações de h: 37.

Sendo $\pi \approx 3.141592653589793$ *com* 16 *dígitos correctos, nenhum dos métodos produz aproximações com mais de* 6 *dígitos de precisão. Estes resultados são realmente insatisfatórios e devem-se ao efeito significativo da acumulação de erros durante os cálculos, sem deixar de salientar a própria instabilidade inerente do problema, que se manifestam sobretudo quando se está já perto da solução (em todos os métodos apenas na última iteração se deixa de verificar a inclusão do valor* π*). É considerável o aumento do número de iterações realizadas pelos algoritmos e o seu o comportamento não reflete os índices de eficiência.*

Se aplicarmos os algoritmos com uma precisão fixa de 40 dígitos já é possível obter uma aproximação para a solução com 16 dígitos de precisão, sendo embora necessárias 151 avaliações da função h.

Poderíamos acrescentar ainda outros exemplos que exibissem um melhor ou pior desempenho dos métodos I a VII, mas julgamos serem já representativos os exemplos que acábamos de expor.

As diferentes experiências que realizámos com os algoritmos usando as capacidades de representação arbitrária de números do *Mathematica*, não acrescem nada de novo às conclusões já apresentadas sobre a convergência dos métodos. Também não era este o objectivo quando decidimos uma implementação naquele programa. Permite-nos, isso sim, obter aproximações para a solução de uma equação real de uma variável real $f(x) = 0$ com extraordinária precisão, sendo para isso necessário apenas um pequeno acréscimo no número avaliações da função f. A título de exemplo, se para resolvermos a equação do exemplo 3.2.2 com o algoritmo VII indicarmos uma precisão arbitrária de 100 dígitos, poderemos obter uma aproximação com quase 40 dígitos de precisão, exigindo-se o cálculo de apenas mais 3 avaliações da função g .

Conclusões

Quando pretendemos resolver uma equação não linear $f(x) = 0$ dispomos geralmente *a priori* de informação suficiente sobre a raiz procurada para assegurar que a convergência não seja um problema se usarmos métodos localmente convergentes. Caso essa informação seja pobre, é geralmente aconselhável usar um método que, embora mais lento, convirja independentemente dos valores iniciais, até que uma boa aproximação seja obtida, e depois passar para um método de convergência mais rápida.

Os métodos I a VII são métodos intervalares que combinam aquelas ideias: garantia de inclusão do zero de f e rapidez de convergência. São assim métodos globalmente convergentes e dependem de se encontrar um intervalo $[a, b]$ tal que $f(a)$ e $f(b)$ tenham sinais contrários. Uma vez encontrado esse intervalo, independentemente da sua amplitude, estes métodos irão prosseguir até que a raiz seja determinada. Contudo, se f tem vários zeros em $[a, b]$, então diferentes intervalos iniciais têm de ser encontrados para determinar, se o pretendermos, cada zero separadamente, o que pode não ser fácil sem o auxílio de um programa com capacidades gráficas. Ainda, se se tratar de uma zero de multiplicidade par não há possibilidade, a não ser por acaso, destes métodos convergirem ou mesmo poderem ser aplicados.

Como exemplos de métodos localmente convergentes para a resolução de equações não lineares temos os métodos de Newton e da secante, que aplicam simplesmente interpolação linear, e o método de Muller, com o uso de interpolação quadrática. É bem conhecido que estes métodos têm índices de eficiência assimptótica muito bons: o método de Newton tem índice de eficiência $\sqrt{2} = 1.414\ldots$, o método da secante $1.618\ldots$ e o método de Muller $1.839\ldots$. Mas, tratando-se de métodos localmente convergentes, podem ter um comportamento bastante erróneo no caso da aproximação inicial não estar suficientemente próxima da raiz. Nos métodos I a VII esta dificuldade é naturalmente removida.

Tanto quanto se conhecia até ao momento, o índice de eficiência assimptótica do método V, 1.618....., era o mais elevado índice de eficiência computacional para métodos iterativos de resolução de uma equação não linear com ordem de convergência superior a 1 e que produzem uma sucessão monótona de intervalos de inclusão para zeros simples de funções suficientemente suaves e sem exigências de convexidade. Os métodos VI e

VII melhoraram ainda este índice de eficiência. Com o método VII obtém-se um índice de eficiência de 1.669....

Os resultados numéricos mostram que realmente o método VII tem um desempenho prático muito bom. É um método robusto e bastante eficiente. Sem se exigir convexidade à função f, as condições que garantem aquele índice de eficiência podem consider-se comuns. Para o cálculo de raízes não simples, sendo a multiplicidade ímpar, não se conseguem evitar os sempre difíceis problemas de instabilidade numérica.

Para a implementaçao dos algoritmos I a VII não fizemos uma análise global dos erros de arredondamento que ocorrem durante a sua execução num computador. Por isso, a inclusão da raiz de uma equação no intervalo final não é uma garantia total e apenas o poderá ser para a função computável f. No entanto, nas diversas experiências que realizámos, com o auxílio das capacidades de representação arbitrária de números reais do *Mathematica*, verificámos que, no caso de raízes simples, a acumulação de erros durante os cálculos não é significativa. Obtêm-se resultados práticos muito bons.

Métodos de inclusão baseados em ferramentas de análise intervalar foram na maior parte primeiramente desenvolvidos para lidar com os efeitos de erros de arredondamento envolvidos na computação, permitindo assim garantir correcção nos resultados. Estes métodos não pretendem substituir os métodos convencionais, pelo contrário, incorporam estes métodos de forma extensiva. Para a resolução de uma equação não linear, algumas versões intervalares do método de Newton e de métodos de interpolação lidam com sucesso com o caso de funções não convexas, garantindo a inclusão da raiz no intervalo final (veja-se [5] e [18]). Contudo, o uso de aritmética intervalar pode tornar-se muito dispendioso.

Em [4], Alefeld, Potra e Völker introduziram três novas versões intervalares do método de Newton, que têm as mesmas propriedades de convergência e inclusão. O primeiro tem também eficiência assimptótica $\sqrt{2} = 1.414\ldots$, o segundo tem eficiência $1.618\ldots$ e para o terceiro método obtêm o valor $1.839\ldots$ A diferente estrutura destes métodos não permite facilmente fazer uma comparação como os métodos I a VII e também não incluimos esse objectivo neste trabalho. Além disso, poderemos dizer que o foco da computação intervalar tem sido direccionado dos erros de arredondamento para lidar também com os erros nos dados iniciais, isto é, para o desenvolver de uma conveniente e eficiente ferramente de análise de dados, em alternativa a outros modelos. Certamente, serão estas as áreas de aplicação de análise intervalar dominantes nos próximos anos.

Apêndice A

Interpolação de Neville

Sejam $x_0, x_1, \ldots, x_n$, $n+1$ pontos pertencentes a $[a,b] \subset \mathbb{R}$ e $f_0 = f(x_0)$, $f_1 = f(x_1), \ldots, f_n = f(x_n)$ os correspondente valores de uma função real f definida em $[a,b]$.

Neste apêndice iremos analisar um algoritmo que permite construir recursivamente o polinómio p_n interpolador de f em $x_0, x_1, \ldots, x_n$, através da definição de *polinómios interpoladores parciais*, isto é, polinómios que interpolam f em alguns dos pontos $x_0, x_1, \ldots, x_n$. Este algoritmo é apresentado, por exemplo, em Stoer e Burlirsch [27, pags. 40-43] e tem como ferramenta básica um lema que nos permite representar um polinómio interpolador de grau $k+1$ em termos de dois tais polinómios de grau k.

A.1 Lema de Aitken

Seja $W = \{x_0, x_1, \ldots, x_n\}$ o conjunto de todos os pontos de interpolação. P_S denotará o polinómio interpolador de f nos pontos x_i pertencentes a S, onde S representa um subconjunto de W. Assim, se S contém $k+1$ pontos, P_S é o único polinómio de grau não superior a k que satisfaz

$$P_S(x_i) = f_i, \quad x_i \in S.$$

Lema A.1.1 *Sejam S e T dois subconjuntos próprios de W tendo todos os pontos em comum excepto os dois pontos $x_i \in S$ e $x_j \in T$. Então*

$$P_{S \cup T} = \frac{(x - x_i)P_T(x) - (x - x_j)P_S(x)}{x_j - x_i}. \tag{A.1}$$

Demonstração. Sejam S e T dois subconjuntos de W com $m+1$ pontos que satisfazem as condições do lema. Assim, P_S e P_T são polinómios interpoladores de grau $\leq m$.

Denotando por P a expressão do lado direito de $(A.1)$ é imediato verificar que P tem grau $\leq m+1$. Vamos mostrar que P interpola f em todos os pontos pertencentes a $S \cup T$.

Seja $x_k \in S \cap T$. Uma vez que

$$P_S(x_k) = P_T(x_k) = f_k,$$

vem

$$P(x_k) = \frac{(x_k - x_i)f_k - (x_k - x_j)f_k}{x_j - x_i} = f_k.$$

Para $x = x_i$, temos

$$P(x_i) = \frac{-(x_i - x_j)P_S(x_i)}{x_j - x_i} = P_S(x_i) = f_i$$

e, analogamente, para $x = x_j$ vem

$$P(x_i) = \frac{(x_j - x_i)P_T(x_j)}{x_j - x_i} = P_T(x_j) = f_j.$$

Logo, $P = P_{S \cup T}$ dada a unicidade do polinómio interpolador.□

Exemplo A.1.1 *Se* $S = \{x_i\}$ *e* $T = \{x_j\}$, $S \cap T$ *é vazio e teremos*

$$\begin{aligned} P_{\{x_i,x_j\}} &= \frac{(x - x_i)P_{\{x_j\}}(x) - (x - x_j)P_{\{x_i\}}(x)}{x_j - x_i} \\ &= \frac{(x - x_i)f_j - (x - x_j)f_i}{x_j - x_i}. \end{aligned}$$

O lema (*A*.1.1) permite-nos gerar o polinómio interpolador p_n sucessivamente a partir de polinómios de graus mais baixos. Temos ainda liberdade de escolher os conjuntos S e T usados para obter o polinómio final $P_W \equiv p_n$. Duas escolhas tornaram-se generalizadas - uma deve-se a *Aitken* e outra a *Neville*. O esquema proposto por Aitken é raramente usado na prática e tem sido substituido pelo muito usado *algoritmo de Neville*.

Ainda, este lema pode ser aplicado de duas formas. Podemos usá-lo para obter uma *representação* explícita do polinómio interpolador ou podemos usá-lo para obter o *valor* do polinómio num dado ponto x.

A.2 Algoritmo de Neville

Existem aplicações em que o conjunto de nós de interpolação tem de ser frequentemente alterado e em que se pretende unicamente o valor do polinómio interpolador num dado ponto. Nestes casos, é mais adequado resolver o problema de forma directa evitando a representação formal do polinómio interpolador.

O algoritmo de Neville tem como objectivo a determinação do valor do polinómio interpolador num dado ponto x e é menos indicado para determinar o próprio polinómio interpolador. Para esta última tarefa existem

outros algorimos mais eficientes como é o caso da fórmula de Newton com diferenças divididas.

No que se segue iremos passar a denotar por $P_{i_0 i_1 \ldots i_k}$ o polinómio interpolador $P_{\{x_{i_0}, x_{i_1}, \ldots, x_{i_k}\}}$ em que $\{i_0, i_1, \ldots, i_k\} \subseteq \{0, 1, \ldots, n\}$.

O valor do polinómio interpolador $p_n(x) = P_{012\ldots n}(x)$ para um dado ponto x fixo é calculado pelo algoritmo de Neville recursivamente com base na fórmula iterativa fornecida pelo lema A.1.1 :

$$\begin{aligned} P_i(x) &= f_i, \quad i = 0, 1, \ldots, n \\ P_{i_0 i_1 \ldots i_k} &= \frac{(x - x_{i_0}) P_{i_1 i_2 \ldots i_k}(x) - (x - x_{i_k}) P_{i_0 i_1 \ldots i_{k-1}}(x)}{x_{i_k} - x_{i_0}}. \end{aligned} \tag{A.2}$$

Este algoritmo constrói uma tabela dos valores numéricos dos polinómios $P_{i_0 i_1 \cdots i_k}$ no ponto x, que vão sendo sistematicamente calculados até que finalmente resulta $P_{012\ldots n}(x)$. A forma sugerida por Neville para construir esses polinómios é a apresentada na tabela seguinte no caso em que $n = 3$.

	$k=0$	1	2	3
x_0	$f_0 = P_0(x)$			
		$P_{01}(x)$ ↘		
x_1	$f_1 = P_1(x)$		$\boxed{P_{012}(x)}$	
		$P_{12}(x)$ ↗		$P_{0123}(x)$
x_2	$f_2 = P_2(x)$		$P_{123}(x)$	
		$P_{23}(x)$		
x_3	$f_3 = P_3(x)$			

(A.3)

A primeira coluna da tabela contém os valores f_i. As colunas seguintes são preenchidas calculando-se cada entrada a partir das duas entradas vizinhas na coluna anterior de acordo com (A.2). Por exemplo, $P_{012}(x)$ é dado por

$$P_{012}(x) = \frac{(x - x_0) P_{12}(x) - (x - x_2) P_{01}(x)}{x_2 - x_0}.$$

De seguida iremos apresentar algumas variantes do algoritmo de Neville.

A.3 Variantes do algoritmo de Neville

Uma versão do algoritmo de Neville tem por base o uso da seguinte abreviatura

$$T_{i,k} := P_{i-k, i-k+1, \ldots, i}(x), \quad k = 0, 1, \ldots, n;\ i = k, k+1, \ldots, n,$$

em que $T_{i,k}$ designará o polinómio de grau $\leq k$ interpolador nos $k+1$ nós $x_{i-k}, x_{i-k+1}, \ldots, x_i$ calculado no ponto x.

Com esta notação o valor prentendido $p_n(x) = P_{012\ldots n}(x)$ é designado por T_{nn} e a tabela em $(A.3)$ ficará

	$k=0$	1	2	3	4
x_0	$f_0 = T_{00}$				
		T_{11}			
x_1	$f_1 = T_{10}$		T_{22}		
		T_{21}		$T_{33}\searrow$	
x_2	$f_2 = T_{20}$		$T_{32}\searrow$		T_{44}
		$T_{31}\searrow$		$T_{43}\nearrow$	
x_3	$f_3 = T_{30}\searrow$		$T_{42}\nearrow$		
		$T_{41}\nearrow$			
x_4	$f_4 = T_{40}\nearrow$				

As setas indicam como uma diagonal $T_{i0}, T_{i1}, \ldots, T_{ii}$ pode ser calculada se um par (x_i, f_i) for acrescentado à tabela.

A fórmula iterativa em $(A.2)$ pode ser modificada permitindo uma avaliação computacional mais eficiente. Teremos

$$
\begin{aligned}
T_{i0} &:= f_i, \quad i = 0, 1, \ldots, n \\
T_{ik} &:= \frac{(x - x_{i-k})T_{i,k-1} - (x - x_i)T_{i-1,k-1}}{x_i - x_{i-k}} \\
&= \frac{(x - x_i + x_i - x_{i-k})T_{i,k-1} - (x - x_i)T_{i-1,k-1}}{x_i - x_{i-k}} \\
&= T_{i,k-1} + \frac{(x - x_i)\,(T_{i,k-1} - T_{i-1,k-1})}{x_i - x_{i-k}} \\
&= T_{i,k-1} + \frac{T_{i,k-1} - T_{i-1,k-1}}{\dfrac{x_i - x_{i-k} + x - x}{x - x_i}} \\
&= T_{i,k-1} + \frac{T_{i,k-1} - T_{i-1,k-1}}{\dfrac{x - x_{i-k}}{x - x_i} - 1}, \quad i = 1, 2, \ldots, n;\ k = 1, 2, \ldots, i.
\end{aligned}
\tag{A.4}
$$

Deve notar-se que a forma como se fizeram variar os índices i e k sugere a construção da tabela *em diagonal.* Também é importante observar que com esta modificação se reduziu o número de divisões/multiplicações que é necessário realizar.

Outra modificação do algoritmo de Neville permite de alguma forma melhorar a precisão do resultado final.

Sejam as quantidades Q_{ik} e D_{ik} definidas por

$$
\begin{aligned}
Q_{i0} &:= D_{i0} := f_i, \qquad i = 0, 1, \ldots, n \\
Q_{ik} &:= T_{ik} - T_{i,k-1} \\
D_{ik} &:= T_{ik} - T_{i-1,k-1}, \quad i = 1, 2, \ldots, n;\ k = 1, 2, \ldots, i.
\end{aligned}
\tag{A.5}
$$

Usando os resultados intermédios dados em $(A.4)$ poderemos escrever

$$\begin{aligned} Q_{ik} &:= T_{ik} - T_{i,k-1} \\ &= (T_{i,k-1} - T_{i-1,k-1}) \frac{x - x_i}{x_i - x_{i-k}} \\ &= (D_{i,k-1} - Q_{i-1,k-1}) \frac{x_i - x}{x_{i-k} - x_i}, \end{aligned}$$

pois

$$\begin{aligned} D_{i,k-1} - Q_{i-1,k-1} &= T_{i,k-1} - T_{i-1,k-2} - T_{i-1,k-1} + T_{i-1,k-2} \\ &= T_{i,k-1} - T_{i-1,k-1}. \end{aligned}$$

Teremos também

$$\begin{aligned} D_{ik} &:= T_{ik} - T_{i-1,k-1} \\ &= T_{i,k-1} + \frac{(x - x_i)}{x_i - x_{i-k}} (T_{i,k-1} - T_{i-1,k-1}) - T_{i-1,k-1} \\ &= T_{i,k-1} \left(1 + \frac{x - x_i}{x_i - x_{i-k}}\right) - T_{i-1,k-1} \left(\frac{x - x_i}{x_i - x_{i-k}} + 1\right) \\ &= (T_{i,k-1} - T_{i-1,k-1}) \left(\frac{x - x_i}{x_i - x_{i-k}} + 1\right) \\ &= (D_{i,k-1} - Q_{i-1,k-1}) \frac{x_{i-k} - x}{x_{i-k} - x_i}. \end{aligned}$$

Assim a fórmula iterativa para T_{ik} em $(A.4)$ dará lugar a

$$\begin{aligned} Q_{i0} &:= D_{i0} := f_i, && i = 0, 1, \ldots, n \\ Q_{ik} &= (D_{i,k-1} - Q_{i-1,k-1}) \frac{x_i - x}{x_{i-k} - x_i} \\ D_{ik} &= (D_{i,k-1} - Q_{i-1,k-1}) \frac{x_{i-k} - x}{x_{i-k} - x_i}, && i = 1, 2, \ldots, n;\ k = 1, 2, \ldots, i. \end{aligned} \tag{A.6}$$

Se pretendermos organizar as quantidades Q_{ik} e D_{ik} numa tabela ficará

	$k=0$	1	2	3
x_0	$f_0 = Q_{00}$			
	$= D_{00}$			
		Q_{11}		
		D_{11}		
x_1	$f_1 = Q_{10}$		Q_{22}	
	$= D_{10}$		D_{22}	
		Q_{21} ↘		Q_{33}
		D_{21}		D_{33}
x_2	$f_2 = Q_{20}$		$\boxed{Q_{32}}$	
	$= D_{20}$		$\boxed{D_{32}}$	
		Q_{31}		
		D_{31} ↗		
x_3	$f_3 = Q_{30}$			
	$= D_{30}$			

Deve ser observado que Q_{ik} e D_{ik} vão sendo calculadas pela fórmulas em $(A.6)$ sem que as quantidades T_{ik} sejam determinadas explicitamente. Por exemplo,

$$Q_{32} = (D_{3,1} - Q_{2,1}) \frac{x_3 - x}{x_1 - x_3}$$

e

$$D_{32} = (D_{3,1} - Q_{2,1}) \frac{x_1 - x}{x_1 - x_3}.$$

Finalmente,

$$\begin{aligned} T_{nn} &= p_n(x) \\ &= f_n + \sum_{k=1}^{n} Q_{nk}. \end{aligned}$$

Com efeito, aplicando a definição em $(A.5)$,

$$\begin{aligned} f_n + \sum_{k=1}^{n} Q_{nk} &= f_n + Q_{n1} + Q_{n2} + \ldots + Q_{nn} \\ &= f_n + T_{n1} - T_{n0} + T_{n2} - T_{n1} + \ldots + T_{nn} - T_{n,n-1} \\ &= T_{nn}. \end{aligned}$$

Se os valores $f_0, f_1, \ldots, f_n$ são próximos uns dos outros, as quantidades Q_{ik} serão pequenas comparadas com os valores f_i. Basta observar, por exemplo, que $Q_{i1}, i \geq 1$ (e D_{i1}) são obtidos a partir da diferença $f_i - f_{i-1}$ resultando valores pequenos comparados com f_i. Nos sucessivos cálculos de $Q_{ik}, k > 1$ obtêm-se também valores pequenos. Isto sugere formar a soma das *correcções* $Q_{n1}, Q_{n2}, \ldots, Q_{nn}$ primeiro (contrariamente à fórmula em $(A.4)$) e depois adicionar a f_n, evitando-se assim a acumulação de erros de arredondamento desnecessários.

Por fim, se $x = 0$ as fórmulas em $(A.4)$ e em $(A.6)$ simplificam respectivamente para

$$\begin{aligned} T_{i0} &:= f_i, \quad i = 0, 1, \ldots, n \\ T_{ik} &:= T_{i,k-1} + \frac{T_{i,k-1} - T_{i-1,k-1}}{\frac{x_{i-k}}{x_i} - 1}, \quad i = 1, 2, \ldots, n;\ k = 1, 2, \ldots, i, \end{aligned} \tag{A.7}$$

e

$$\begin{aligned} Q_{i0} &:= D_{i0} := f_i, && i = 0, 1, \ldots, n \\ Q_{ik} &= (D_{i,k-1} - Q_{i-1,k-1}) \frac{x_i}{x_{i-k} - x_i} \\ D_{ik} &= (D_{i,k-1} - Q_{i-1,k-1}) \frac{x_{i-k}}{x_{i-k} - x_i}, && i = 1, 2, \ldots, n;\ k = 1, 2, \ldots, i. \end{aligned} \tag{A.8}$$

A.4 Interpolação Inversa

O problema de encontrar o valor x tal que para um dado y temos $f(x) = y$ pode ser resolvido por interpolação inversa. Considerando o conjunto de pontos $(x_i, y_i = f(x_i))$; $i = 0, \ldots, n$, simplesmente interpolamos a função inversa $x = f^{-1}(y)$, trocando os papéis de x e y. Contudo, isto apenas faz sentido se a função f for invertível no intervalo de interpolação. O algorimo de Neville é adequado para a resolução deste problema.

Iremos analisar o procedimento *zero_pinverso*, introduzido aquando da apresentação dos método VI e VII (veja-se pag. 24), que constitui um exemplo de aplicação de interpolação inversa usando a variante do algoritmo de Neville que acabámos de expor.

Seja $y = f(x)$ uma função real invertível definida em $[a, b]$. Consideremos quatro pontos $x_0 = l$, $x_1 = e$, $x_2 = d$ e $x_3 = c$ pertencentes a $[a, b]$ e $y_0 = f(x_0)$, $y_1 = f(x_1)$, $y_2 = f(x_2)$ e $y_3 = f(x_3)$ os correspondentes valores de f. Estamos interessados em obter uma aproximação da solução da equação $f(x) = 0$ usando interpolação cúbica inversa.

Designemos por $x = q_3(y)$ o polinómio interpolador de f^{-1} nos pontos

$$(y_0, x_0),\ (y_1, x_1),\ (y_2, x_2) \text{ e } (y_3, x_3)$$

ou

$$(f(l), l),\ (f(e), e)), (f(d), d) \text{ e } (f(c), c).$$

Apenas pretendemos obter o valor do polinómio q_3 quando $y = 0$, isto é, obter uma solução aproximada $\overline{x}$ de $x = f^{-1}(0)$ dada por

$$\overline{x} = q_3(0).$$

Aplicando a fórmula simplificada dada em $(A.8)$ obtém-se

$$\begin{aligned}
Q_{i0} &= D_{i0} = x_i, \qquad i = 0, 1, 2, 3; \\
Q_{11} &= (D_{10} - Q_{00}) \frac{y_1}{y_0 - y_1} = (x_1 - x_0) \frac{y_1}{y_0 - y_1} = (e - l) \frac{f(e)}{f(l) - f(e)}; \\
D_{11} &= (D_{10} - Q_{00}) \frac{y_0}{y_0 - y_1} = (e - l) \frac{f(l)}{f(l) - f(e)}; \\
Q_{21} &= (D_{20} - Q_{10}) \frac{y_2}{y_1 - y_2} = (x_2 - x_1) \frac{y_2}{y_1 - y_2} = (d - e) \frac{f(d)}{f(e) - f(d)}; \\
D_{21} &= (D_{20} - Q_{10}) \frac{y_1}{y_1 - y_2} = (d - e) \frac{f(e)}{f(e) - f(d)}; \\
Q_{22} &= (D_{21} - Q_{11}) \frac{y_2}{y_0 - y_2} = (D_{21} - Q_{11}) \frac{f(d)}{f(l) - f(d)}; \\
D_{22} &= (D_{21} - Q_{11}) \frac{y_0}{y_0 - y_2} = (D_{21} - Q_{11}) \frac{f(l)}{f(l) - f(d)};
\end{aligned}$$

$$
\begin{aligned}
Q_{31} &= (D_{30} - Q_{20}) \frac{y_3}{y_2 - y_3} = (x_3 - x_2) \frac{y_3}{y_2 - y_3} = (c - d) \frac{f(c)}{f(d) - f(c)}; \\
D_{31} &= (D_{30} - Q_{20}) \frac{y_2}{y_2 - y_3} = (c - d) \frac{f(d)}{f(d) - f(c)}; \\
Q_{32} &= (D_{31} - Q_{21}) \frac{y_3}{y_1 - y_3} = (D_{31} - Q_{21}) \frac{f(c)}{f(e) - f(c)}; \\
D_{32} &= (D_{31} - Q_{21}) \frac{y_1}{y_1 - y_3} = (D_{31} - Q_{21}) \frac{f(e)}{f(e) - f(c)}; \\
Q_{33} &= (D_{32} - Q_{22}) \frac{y_3}{y_0 - y_3} = (D_{32} - Q_{22}) \frac{f(c)}{f(l) - f(c)}; \\
D_{33} &= (D_{32} - Q_{22}) \frac{y_0}{y_0 - y_3} = (D_{32} - Q_{22}) \frac{f(l)}{f(l) - f(c)}.
\end{aligned}
$$

Convém notar que as quantidades D_{ii}, $i = 1, 2, \ldots, n$ nunca são utilizadas e o seu cálculo deve ser omitido.

Finalmente, temos

$$
\begin{aligned}
\overline{x} &= p_3(0) \\
&= x_3 + Q_{31} + Q_{32} + Q_{33} \\
&= c + Q_{31} + Q_{32} + Q_{33}.
\end{aligned}
$$

Com este exemplo, os passos do procedimento *zero_p_inverso* ficam esclarecidos por completo.

Apêndice B

Resultados sobre matrizes não negativas

Neste apêndice iremos apresentar, sem demonstrar, apenas três resultados válidos para *matrizes não negativas* extraídos de Abraham [6] e que foram referidos no capítulo 2 para a demonstração do teorema 2.3.1 e corolário 2.3.2.

B.1

Definição B.1.1 *Sejam* $A = (a_{ij})$ *e* $B = (b_{ij})$ *duas matrizes* $n \times r$. *Então,*

$$A \geq B \quad se \quad a_{ij} \geq b_{ij}, \quad i = 1, \ldots, n;\ j = 1, \ldots, r,$$
$$A > B \quad se \quad a_{ij} > b_{ij}, \quad i = 1, \ldots, n;\ j = 1, \ldots, r.$$

Se O *é a matriz nula e* $A \geq O$ $(> O)$, *dizemos que* A *é uma matriz não negativa (positiva).*

Teorema B.1.1 *Se* A *é uma matriz quadrada não negativa, então*

(a) $\rho(A)$, *o raio espectral de* A, *é um valor próprio de* A;

(b) A *tem um vector próprio não negativo correspondente a* $\rho(A)$;

(c) A^T *tem um vector próprio não negativo correspondente a* $\rho(A)$.

Teorema B.1.2 *Para* $A \geq O$,

$\alpha x \leq Ax, \quad x \geq 0$ *(com pelo menos uma componente positiva)* $\Longrightarrow \alpha \leq \rho(A)$

e

$$Ax \leq \beta x, \quad x > 0 \Longrightarrow \rho(A) \leq \beta.$$

Corolário B.1.3 *Se* x *é um vector próprio positivo de uma matriz não negativa* A, *então* x *corresponde a* $\rho(A)$.

Apêndice C

Implementação dos algoritmos: listagem do programa e dois exemplos de utilização

```
BeginPackage["EquacoesNaoLineares`"]

Algoritmo1::usage = "Algoritmo1[f,a,b] procura um intervalo de inclusão da
   solução da equação f(x)=0 começando com o intervalo de extremos a e b.
   Assume-se que f é uma função real de uma variável
   real contínua no intervalo inicial considerado e que f(a)f(b)<0.
  A opção Tolerancia refere-se à amplitude do intervalo de inclusão final.
  Tolerancia ->
   erro significa que a amplitude do intervalo final será não superior a 2*erro.
  MaxIteracoes constitui o número máximo de iterações que serão realizadas.
  PrecisaoUsada indica a precisao que deve ser usada nos cálculos, podendo
  a aritmética ser de precisão fixa (PrecisaoFixa->True)
  ou variável (PrecisaoFixa->False).
  LambdaTol e Miu são opções referentes a parâmetros usados na implementação
   do algoritmo e devem tomar valores positivos e inferiores a 1.
  A opção MostrarIteracoes (True ou False) permite
  que todas as iterações possam ser apresentadas.";

Algoritmo2::usage = "Algoritmo2[f,a,b] procura um intervalo de inclusão da
   solução da equação f(x)=0 começando com o intervalo de extremos a e b.
   Assume-se que f é uma função real de uma variável
   real contínua no intervalo inicial considerado e que f(a)f(b)<0.
  A opção Tolerancia refere-se à amplitude do intervalo de inclusão final.
  Tolerancia ->
   erro significa que a amplitude do intervalo final será não superior a 2*erro.
  MaxIteracoes constitui o número máximo de iterações que serão realizadas.
  PrecisaoUsada indica a precisao que deve ser usada nos cálculos, podendo
  a aritmética ser de precisão fixa (PrecisaoFixa->True)
  ou variável (PrecisaoFixa->False).
  LambdaTol e Miu são opções referentes a parâmetros usados na implementação
   do algoritmo e devem tomar valores positivos e inferiores a 1.
  A opção MostrarIteracoes (True ou False)
  permite que todas as iterações possam ser apresentadas.";

Algoritmo3::usage = "Algoritmo1[f,a,b] procura um intervalo de inclusão da
   solução da equação f(x)=0 começando com o intervalo de extremos a e b.
   Assume-se que f é uma função real de uma variável
   real contínua no intervalo inicial considerado e que f(a)f(b)<0.
  A opção Tolerancia refere-se à amplitude do intervalo de inclusão final.
  Tolerancia ->
   erro significa que a amplitude do intervalo final será não superior a 2*erro.
  MaxIteracoes constitui o número máximo de iterações que serão realizadas.
  PrecisaoUsada indica a precisao que deve ser usada nos cálculos, podendo
  a aritmética ser de precisão fixa (PrecisaoFixa->True)
  ou variável (PrecisaoFixa->False).
 LambdaTol é uma opção referente a um parâmetro usado na implementação
   do algoritmo e deve tomar um valor positivo e inferior a 1.
  A opção MostrarIteracoes (True ou False)
  permite que todas as iterações possam ser apresentadas.";

Algoritmo4::usage = "Algoritmo4[f,a,b] procura um intervalo de inclusão da
   solução da equação f(x)=0 começando com o intervalo de extremos a e b.
   Assume-se que f é uma função real de uma variável
   real contínua no intervalo inicial considerado e que f(a)f(b)<0.
```

```
A opção Tolerancia refere-se à amplitude do intervalo de inclusão final.
Tolerancia ->
 erro significa que a amplitude do intervalo final será não superior a 2*erro.
MaxIteracoes constitui o número máximo de iterações que serão realizadas.
PrecisaoUsada indica a precisao que deve ser usada nos cálculos, podendo
a aritmética ser de precisão fixa (PrecisaoFixa->True)
ou variável (PrecisaoFixa->False).
LambdaTol e Miu são opções referentes a parâmetros usados na implementação
 do algoritmo e devem tomar valores positivos e inferiores a 1.
A opção MostrarIteracoes (True ou
 False) permite que todas as iterações possam ser apresentadas.";

Algoritmo5::usage = "Algoritmo5[f,a,b] procura um intervalo de inclusão da
   solução da equação f(x)=0 começando com o intervalo de extremos a e b.
   Assume-se que f é uma função real de uma variável
   real contínua no intervalo inicial considerado e que f(a)f(b)<0.
  A opção Tolerancia refere-se à amplitude do intervalo de inclusão final.
  Tolerancia ->
   erro significa que a amplitude do intervalo final será não superior a 2*erro.
  MaxIteracoes constitui o número máximo de iterações que serão realizadas.
  PrecisaoUsada indica a precisao que deve ser usada nos cálculos, podendo
  a aritmética ser de precisão fixa (PrecisaoFixa->True)
  ou variável (PrecisaoFixa->False).
  LambdaTol e Miu são opções referentes a parâmetros usados na implementação
   do algoritmo e devem tomar valores positivos e inferiores a 1.
  A opção MostrarIteracoes (True ou False)
  permite que todas as iterações possam ser apresentadas.";

Algoritmo6::usage = "Algoritmo6[f,a,b] procura um intervalo de inclusão da
   solução da equação f(x)=0 começando com o intervalo de extremos a e b.
   Assume-se que f é uma função real de uma variável
   real contínua no intervalo inicial considerado e que f(a)f(b)<0.
  A opção Tolerancia refere-se à amplitude do intervalo de inclusão final.
  Tolerancia ->
   erro significa que a amplitude do intervalo final será não superior a 2*erro.
  MaxIteracoes constitui o número máximo de iterações que serão realizadas.
  PrecisaoUsada indica a precisao que deve ser usada nos cálculos, podendo
  a aritmética ser de precisão fixa (PrecisaoFixa->True)
  ou variável (PrecisaoFixa->False).
  LambdaTol e Miu são opções referentes a parâmetros usados na implementação
   do algoritmo e devem tomar valores positivos e inferiores a 1.
  A opção MostrarIteracoes (True ou False) permite
  que todas as iterações possam ser apresentadas.";

Algoritmo7::usage = "Algoritmo6[f,a,b] procura um intervalo de inclusão da
   solução da equação f(x)=0 começando com o intervalo de extremos a e b.
   Assume-se que f é uma função real de uma variável
   real contínua no intervalo inicial considerado e que f(a)f(b)<0.
  A opção Tolerancia refere-se à amplitude do intervalo de inclusão final.
  Tolerancia ->
   erro significa que a amplitude do intervalo final será não superior a 2*erro.
  MaxIteracoes constitui o número máximo de iterações que serão realizadas.
  PrecisaoUsada indica a precisao que deve ser usada nos cálculos, podendo
  a aritmética ser de precisão fixa (PrecisaoFixa->True)
  ou variável (PrecisaoFixa->False).
```

```
  LambdaTol e Miu são opções referentes a parâmetros usados na implementação
   do algoritmo e devem tomar valores positivos e inferiores a 1.
  A opção MostrarIteracoes (True ou False) permite
  que todas as iterações possam ser apresentadas.";


Begin["`Private`"];


(* ============================== Algoritmo I ======================== *)


Clear[Algoritmo1, f, a, b, opcoes];
Options[Algoritmo1] = {
            Tolerancia -> 10^-6,
            LambdaTol -> 7 / 10,
            Miu -> 1 / 2,
            MaxIteracoes -> 15 ,
            PrecisaoUsada -> $MachinePrecision,
            PrecisaoFixa -> False,
            MostrarIteracoes -> False
          };


Algoritmo1[f_Symbol, a_, b_, opcoes___] :=
  Module[{intervalos, a0, b0, niter = 1,
   result, erro, lambdaErro, miu1, maxiter, prec, ver, fix, pmax, pmin},
   (
    {erro, lambdaErro, miu1, maxiter, prec, ver, fix, a0, b0} =
     tratarOpcoes[algoritmo1, a, b, opcoes];

    If[prec != 16 && fix, (pmax = $MaxPrecision;
     pmin = $MinPrecision; $MaxPrecision = prec; $MinPrecision = prec)];

    intervalos = {{a0, b0}};
    result = {"CONTINUAR", {a0, b0, f[a0], f[b0], f, erro, lambdaErro, miu1, prec}};

    While[result[[1]] != "TERMINAR" && niter <= maxiter,
          (niter++;
           result = iteracao1[result[[2]]];
           intervalos = Insert[intervalos, {result[[2, 1]], result[[2, 2]]}, -1]
          )] ;

    If[prec != 16 && fix, ($MaxPrecision = pmax; $MinPrecision = pmin)];

   If[ver,
         Print["\n Algoritmo I \n "];
         Print[TableForm[N[intervalos, prec]]];
         Print["\nNúmero de avaliações da função: ", nfuncoes];]

    If[niter > maxiter, Print["Não se obteve convergência
        com a tolerância desejada após ", maxiter, " iterações."]];
    N[intervalos[[niter]], prec]
   )
  ];
```

```
Clear[bracket, a, fa, b, fb, c, fc, f, erro, lambdaErro, prec];
bracket[f_Symbol, a_, fa_, b_, fb_, c_, fc_, erro_, lambdaErro_, prec_] :=
  Module[{alfa, u, mc, fmc, novoInterv, epsilon},
   (
    nfuncoes = nfuncoes + 1;
    If[Min[fa, fb] == fa, u = a, u = b];

    If[prec == 16, epsilon = $MachineEpsilon, epsilon = 5 * 10^(-prec)];
    alfa = lambdaErro * (2 * epsilon * Abs[u] + erro);

   {mc, fmc} = ajustarc[a, b, c, fc, alfa, f];

   Which[fmc == 0,
            novoInterv = {mc, mc, fmc, fmc},
         Sign[fa * fmc] == -1,
            novoInterv = {a, mc, fa, fmc},
         True,
            novoInterv = {mc, b, fmc, fb}
        ];

   If[Min[novoInterv[[3]], novoInterv[[4]]] == novoInterv[[3]],
    u = novoInterv[[1]], u = novoInterv[[2]]];
   If[ novoInterv[[2]] - novoInterv[[1]] <= 4 * epsilon * Abs[u] + 2 * erro,
           {novoInterv[[1]], novoInterv[[2]], "terminar", "terminar"},
                (*else*)
           Table[novoInterv[[i]], {i, 4}]]
   )
];
```

```
Clear[ajustarc, a, b, c, fc, alfa, f];
ajustarc[a_, b_, c_, fc_, alfa_, f_Symbol] :=
 Module[{cm},
   (If[b - a <= 4 * alfa,
       {cm = a + (b - a) / 2, f[cm]},
          (*else*)
       If[c < a + 2 * alfa,
          {cm = a + 2 * alfa, f[cm]},
             (*else*)
          If[c >= b - 2 * alfa,
             {cm = b - 2 * alfa, f[cm]},
               (*else*)
             {c, fc}]]]
   )
 ];
```

```
Clear[iteracao1, f, a, b];
iteracao1[{a_, b_, fa_, fb_, f_Symbol, erro_, lambdaErro_, miu1_, prec_}] :=
  Module[{c, a1, b1, fa1, fb1, u, aux, q},
   (
    If[Abs[fa] < Abs[fb],
            (q = fa / fb; c = a - (b - a) * q / (1 - q)),
                (*else*)
            (q = fb / fa; c = b - (a - b) * q / (1 - q))];

   {a1, b1, fa1, fb1} = bracket[f, a, fa, b, fb, c, f[c], erro, lambdaErro, prec];
    If[fa1 == "terminar", Return[{"TERMINAR", {a1, b1}}]];

    If[Abs[fa1] < Abs[fb1],
             (u = a1; q = fa1 / fb1;  c = a1 - 2 * (b1 - a1) * q / (1 - q)),
                  (*else*)
              (u = b1; q = fb1 / fa1; c = b1 - 2 * (a1 - b1) * q / (1 - q))];

    If[Abs[c - u] > (1 / 2) * (b1 - a1), c = a1 + (b1 - a1) / 2];
    {a1, b1, fa1, fb1} = bracket[f, a1, fa1, b1, fb1, c, f[c], erro, lambdaErro, prec];
    If[fa1 == "terminar", Return[{"TERMINAR", {a1, b1}}]];

    If[b1 - a1 >= miu1 * (b - a), {a1, b1, fa1, fb1} =
      bracket[f, a1, fa1, b1, fb1, aux = a1 + (b1 - a1) / 2, f[aux], erro, lambdaErro, prec];
               If[fa1 == "terminar", Return[{"TERMINAR", {a1, b1}}]];
    ];

    {"CONTINUAR", {a1, b1, fa1, fb1, f, erro, lambdaErro, miu1, prec}}
   )
  ];

Clear[tratarOpcoes, list, a, b];
tratarOpcoes[algoritmo_, a_, b_, opcoes___] :=
 Module[{erro, lambdaErro, miu2, maxiter, prec, ver, fix, a0, b0},
   (
     Clear[nfuncoes]; nfuncoes = 2;
     erro = Tolerancia /. {opcoes} /. Options[Algoritmo2];
     lambdaErro = LambdaTol /. {opcoes} /. Options[Algoritmo2];
     IF[Not[algoritmo === algoritmo3], miu2 = Miu /. {opcoes} /. Options[Algoritmo2]];
     maxiter = MaxIteracoes /. {opcoes} /. Options[Algoritmo2];
     prec = PrecisaoUsada /. {opcoes} /. Options[Algoritmo2];
     fix = PrecisaoFixa /. {opcoes} /. Options[Algoritmo2];
     ver = MostrarIteracoes /. {opcoes} /. Options[Algoritmo2];

     erro = SetPrecision[erro, prec];
     lambdaErro = SetPrecision[lambdaErro, prec];
     miu2 = SetPrecision[miu2, prec];
     If[Precision[{a, b}] < prec,
               a0 = SetPrecision[a, prec];
               b0 = SetPrecision[b, prec],
                  (*else*)
               a0 = N[a, prec];
               b0 = N[b, prec];
               ];

      If[Not[algoritmo === algoritmo3],
```

```
            {erro, lambdaErro, miu2, maxiter, prec, ver, fix, a0, b0},
            {erro, lambdaErro, maxiter, prec, ver, fix, a0, b0}]
  )
 ];


(* ================================ Algoritmo II ========================== *)


Clear[Algoritmo2, f, a, b, opcoes];
Options[Algoritmo2] = {
            Tolerancia -> 10^-6,
            LambdaTol -> 7/10,
            Miu -> 1/2,
            MaxIteracoes -> 15 ,
            PrecisaoUsada -> $MachinePrecision,
            PrecisaoFixa -> False,
            MostrarIteracoes -> False
          };


Algoritmo2[f_, a_, b_, opcoes___] :=
  Module[{intervalos, a0, b0, niter = 1,
    result, erro, lambdaErro, miu2, maxiter, prec, ver, fix, pmax, pmin},
   (
    {erro, lambdaErro, miu2, maxiter, prec, ver, fix, a0, b0} =
    tratarOpcoes[algoritmo2, a, b, opcoes];

    If[prec != 16 && fix, (pmax = $MaxPrecision;
     pmin = $MinPrecision; $MaxPrecision = prec; $MinPrecision = prec)];

    intervalos = {{a0, b0}};
    result = {"CONTINUAR", {a0, b0, f[a0], f[b0], f, erro, lambdaErro, miu2, prec, fix}};

    While[result[[1]] != "TERMINAR" && niter <= maxiter,
          (niter++;
           result = iteracao2[result[[2]]];
           intervalos = Insert[intervalos, {result[[2, 1]], result[[2, 2]]}, -1]
          )];

    If[prec != 16 && fix, ($MaxPrecision = pmax; $MinPrecision = pmin)];

    If[ver,
         Print["\n Algoritmo II  \n "];
         Print[TableForm[N[intervalos, prec]]];
        Print["\nNúmero de avaliações da função: ", nfuncoes];]

    If[niter > maxiter, Print["Não se obteve convergência
        com a tolerância desejada após ", maxiter, " iterações."]];
    N[intervalos[[niter]], prec]
   )
  ];
```

```
Clear[iteracao2, f, a, b, fa, fb, f, erro, lambdaErro, miu2, prec, fix];
iteracao2[{a_, b_, fa_, fb_, f_Symbol, erro_, lambdaErro_, miu2_, prec_, fix_}] :=
 Module[{c, fc, a1, b1, fa1, fb1, sol, tam, u, aux, q},
  (
   If[Abs[fa] < Abs[fb],
           (q = fa / fb; c = a - (b - a) * q / (1 - q)),
              (*else*)
           (q = fb / fa; c = b - (a - b) * q / (1 - q))];

   {a1, b1, fa1, fb1} = bracket[f, a, fa, b, fb, c, fc = f[c], erro, lambdaErro, prec];
   If[fa1 == "terminar", Return[{"TERMINAR", {a1, b1}}]];

   If[Not[MemberQ[{a, b, c}, a1]], c = a1; fc = f[c]];
   If[Not[MemberQ[{a, b, c}, b1]], c = b1; fc = f[c]];

   (* Interpolação quadrática *)
   sol = {};
   If[Length[Union[{a, b, c}]] == 3,
         c = Solve[InterpolatingPolynomial[{{a, fa}, {b, fb}, {c, fc}}, x] == 0, x];
         If[Not[fix] && Precision[c] < prec, c = SetPrecision[c, prec]],
               (*else*)
         c = {}];
   tam = Length[c];
   Which[tam == 2, sol = {x /. c[[1]], x /. c[[2]]} , tam == 1, sol = {x /. c[[1]]}];
   c = Select[sol, (a1 ≤ # ≤ b1 &)];
   If[c != {},
         c = c[[1]],
           (*else*)
         Print[
     "Não se fez interpolação quadrática em [" , N[a, prec], ",", N[b, prec], "]"];
         c = a1 + (b1 - a1) / 2];

   {a1, b1, fa1, fb1} = bracket[f, a1, fa1, b1, fb1, c, f[c], erro, lambdaErro, prec];
   If[fa1 == "terminar", Return[{"TERMINAR", {a1, b1}}]];

   If[Abs[fa1] < Abs[fb1],
            (u = a1; q = fa1 / fb1; c = a1 - 2 * (b1 - a1) * q / (1 - q)),
                 (*else*)
            (u = b1; q = fb1 / fa1; c = b1 - 2 * (a1 - b1) * q / (1 - q))];

   If[Abs[c - u] > (1 / 2) * (b1 - a1), c = a1 + (b1 - a1) / 2];
   {a1, b1, fa1, fb1} = bracket[f, a1, fa1, b1, fb1, c, f[c], erro, lambdaErro, prec];
   If[fa1 == "terminar", Return[{"TERMINAR", {a1, b1}}]];

   If[b1 - a1 >= miu2 * (b - a),
    {a1, b1, fa1, fb1} =
     bracket[f, a1, fa1, b1, fb1, aux = a1 + (b1 - a1) / 2, f[aux], erro, lambdaErro, prec];
    If[fa1 == "terminar", Return[{"TERMINAR", {a1, b1}}]]];

   {"CONTINUAR", {a1, b1, fa1, fb1, f, erro, lambdaErro, miu2, prec, fix} }
  )
 ];
```

```
(* ===================== Algoritmo III ======================= *)


Clear[Algoritmo3, f, a, b, opcoes];
Options[Algoritmo3] = {
            Tolerancia -> 10^-6,
            LambdaTol -> 7 / 10,
            MaxIteracoes -> 15 ,
            PrecisaoUsada -> $MachinePrecision,
            PrecisaoFixa -> False,
            MostrarIteracoes -> False
          };

Algoritmo3[f_, a_, b_, opcoes___] :=
  Module[{intervalos, a0, b0, niter = 1,
    result, erro, lambdaErro, maxiter, prec, ver, fix, pmax, pmin},
    (
     {erro, lambdaErro, maxiter, prec, ver, fix, a0, b0} =
     tratarOpcoes[algoritmo3, a, b, opcoes];

     If[prec != 16 && fix, (pmax = $MaxPrecision;
      pmin = $MinPrecision; $MaxPrecision = prec; $MinPrecision = prec)];

    intervalos = {{a0, b0}};
    result = {"CONTINUAR", {a0, b0, f[a0], f[b0], f, erro, lambdaErro, prec, fix}};

    While[result[[1]] != "TERMINAR" && niter <= maxiter,
           (niter++;
            result = iteracao3[result[[2]]];
            intervalos = Insert[intervalos, {result[[2, 1]], result[[2, 2]]}, -1]
           )];

    If[prec != 16 && fix, ($MaxPrecision = pmax; $MinPrecision = pmin)];

   If[ver,
          Print["\n Algoritmo III  \n "];
          Print[TableForm[N[intervalos, prec]]];
          Print["\nNúmero de avaliações da função: ", nfuncoes];]

     If[niter > maxiter, Print["Não se obteve convergência
         com a tolerância desejada após ", maxiter, " iteraçõe.s"]];
    N[intervalos[[niter]], prec]
   )
  ];

Clear[iteracao3, f, a, b, fa, fb, erro, lambdaErro, prec, fix];
iteracao3[{a_, b_, fa_, fb_, f_Symbol, erro_, lambdaErro_, prec_, fix_}] :=
 Module[{c, fc, a1, b1, fa1, fb1, tam, sol, u, aux, x, q},
  (
   c = a + (b - a) / 2;
   {a1, b1, fa1, fb1} = bracket[f, a, fa, b, fb, c, fc = f[c], erro, lambdaErro, prec];
   If[fa1 == "terminar", Return[{"TERMINAR", {a1, b1}}]];

   If[Not[MemberQ[{a, b, c}, a1]], c = a1; fc = f[c]];
   If[Not[MemberQ[{a, b, c}, b1]], c = b1; fc = f[c]];
```

```
    (* Interpolação quadrática *)
    sol = {};
    If[Length[Union[{a, b, c}]] == 3,
          c = Solve[InterpolatingPolynomial[{{a, fa}, {b, fb}, {c, fc}}, x] == 0, x];
          If[Not[fix] && Precision[c] < prec,
          c = SetPrecision[c, prec]],
                (*else*)
           c = {}];

    tam = Length[c];
    Which[tam == 2, sol = {x /. c[[1]], x /. c[[2]]} , tam == 1, sol = {x /. c[[1]]}];

    c = Select[sol, (a1 ≤ # ≤ b1 &)];
    If[c != {},
          c = c[[1]],
            (*else*)
          Print[
      "Não se fez interpolação quadrática em [" , N[a, prec], ",", N[b, prec], "]"];
          c = a1 + (b1 - a1) / 2];

    {a1, b1, fa1, fb1} = bracket[f, a1, fa1, b1, fb1, c, f[c], erro, lambdaErro, prec];
    If[fa1 == "terminar", Return[{"TERMINAR", {a1, b1}}]];

     If[Abs[fa1] < Abs[fb1],
             (u = a1; q = fa1 / fb1; c = a1 - 2 * (b1 - a1) * q / (1 - q)),
                  (*else*)
              (u = b1; q = fb1 / fa1; c = b1 - 2 * (a1 - b1) * q / (1 - q))];


    {a1, b1, fa1, fb1} = bracket[f, a1, fa1, b1, fb1, c, f[c], erro, lambdaErro, prec];

    If[fa1 == "terminar", Return[{"TERMINAR", {a1, b1}}]];
    {"CONTINUAR", {a1, b1, fa1, fb1, f, erro, lambdaErro, prec, fix}}
   )
  ];



(* ============================ Algoritmo IV ======================== *)


Clear[Algoritmo4, f, a, b, opcoes];
Options[Algoritmo4] = {
            Tolerancia -> 10^-6,
            LambdaTol -> 7 / 10,
            Miu -> 1 / 2,
            MaxIteracoes -> 15 ,
            PrecisaoUsada -> $MachinePrecision,
            PrecisaoFixa -> False,
            MostrarIteracoes -> False
          };


Algoritmo4[f_, a_, b_, opcoes___] :=
```

```
Module[{intervalos, a0, b0, c1, a2, b2, d2, fa2, fb2, fd2, fa0, fb0, niter = 1,
 result, erro, lambdaErro, miu, maxiter, prec, ver, fix, pmax, pmin, q},
 (
  {erro, lambdaErro, miu, maxiter, prec, ver, fix, a0, b0} =
  tratarOpcoes[algoritmo4, a, b, opcoes];

  If[prec != 16 && fix, (pmax = $MaxPrecision;
   pmin = $MinPrecision; $MaxPrecision = prec; $MinPrecision = prec)];

  intervalos = {{a0, b0}};
  fa0 = f[a0];
  fb0 = f[b0];
  If[Abs[fa0] < Abs[fb0],
        (q = fa0 / fb0; c1 = a0 - (b0 - a0) * q / (1 - q)),
           (*else*)
        (q = fb0 / fa0; c1 = b0 - (a0 - b0) * q / (1 - q))];

 {a2, b2, d2, fa2, fb2, fd2} =
  bracket2[f, a0, fa0, b0, fb0, c1, f[c1], erro, lambdaErro, prec];
 If[d2 == "terminar",
       intervalos = Insert[intervalos, {a2, b2}, -1],
            (*else*)
       result =
   {"CONTINUAR", {a2, b2, d2, fa2, fb2, fd2, f, erro, lambdaErro, miu, prec}};

       While[result[[1]] != "TERMINAR" && niter <= maxiter,
         (niter++;
          result = iteracao4[result[[2]]];

          intervalos = Insert[  intervalos, {result[[2, 1]], result[[2, 2]]}, -1]
        )];
 ];

 If[prec != 16 && fix, ($MaxPrecision = pmax; $MinPrecision = pmin)];

 If[ver,
       Print["\n Algoritmo IV  \n "];
       Print[TableForm[N[intervalos, prec]]];
       Print["\nNúmero de avaliações da função: ", nfuncoes];]

 If[niter > maxiter, Print["Não se obteve convergência
     com a tolerância desejada após ", maxiter, " iterações."]];
 N[intervalos[[niter]], prec]
 )
];


Clear[bracket2, a, fa, b, fb, c, fc, erro, lambdaErro, prec];
bracket2[f_Symbol, a_, fa_, b_, fb_, c_, fc_, erro_, lambdaErro_, prec_] :=
Module[{alfa, u, mc, fmc, novoInterv, epsilon},
  (
   nfuncoes = nfuncoes + 1;
   If[Min[fa, fb] == fa, u = a, u = b];

   If[prec == 16, epsilon = $MachineEpsilon, epsilon = 5 * 10^(-prec)];
```

```
    alfa = lambdaErro * (2 * epsilon * Abs[u] + erro);

   {mc, fmc} = ajustarc[a, b, c, fc, alfa, f];
    Which[fmc == 0,
                novoInterv = {mc, mc, mc, fmc, fmc, fmc},
          Sign[fa * fmc] == -1,
                novoInterv = {a, mc, b, fa, fmc, fb},
          True,
                novoInterv = {mc, b, a, fmc, fb, fa}
         ];

     If[Min[novoInterv[[4]], novoInterv[[5]]] == novoInterv[[4]],
     u = novoInterv[[1]], u = novoInterv[[2]]];
     If[ novoInterv[[2]] - novoInterv[[1]] <= 4 * epsilon * Abs[u] + 2 * erro,
          {novoInterv[[1]],
      novoInterv[[2]], "terminar", "terminar", "terminar", "terminar"},
           (*else*)
            Table[novoInterv[[i]], {i, 6}]]
    )
];


Clear[difDiv, n];
difDiv[{x_, y_}, {fx_, fy_}] := difDiv[{x, y}] = (fy - fx) / (y - x);
difDiv[x_List, y_List] := Module[{n = Length[x]},
 difDiv[x] = (difDiv[Drop[x, 1], Drop[y, 1]] -
       difDiv[Drop[x, -1], Drop[y, -1]]) / (x[[n]] - x[[1]])
   ] /; Length[x] > 2 && Length[x] == Length[y];


Clear[newtonQuadratic, a, b, d, fa, fb, fd, prec, k];
newtonQuadratic[a_, b_, d_, fa_, fb_, fd_,
                k_?((IntegerQ[#] && # > 0) &), prec_] :=
   Module[{A, B, r0, poli, frec},

          A = difDiv[{a, b, d}, {fa, fb, fd}];
          B = difDiv[{a, b}, {fa, fb}];
          If[A == 0, r = a - (1 / B) * fa,
                If[A * fa > 0, r0 = a, r0 = b];
                poli[x_] = fa + B (x - a) + A * (x - a) * (x - b);
                frec[x_] = N[x - poli[x] / (B + A (2 x - a - b)), prec];
                FixedPoint[frec, r0, k]
           ]
  ];


Clear[iteracao4, f, a, b, d, fa, fb, fd, f, erro, lambdaErro, miu, prec];
iteracao4[{a_, b_, d_, fa_, fb_, fd_, f_Symbol, erro_, lambdaErro_, miu_, prec_}] :=
 Module[{c, a1, b1, fa1, fb1, d1, fd1, u, aux, q},
  (
   c = newtonQuadratic[a, b, d, fa, fb, fd, 2, prec];
   {a1, b1, d1, fa1, fb1, fd1} =
    bracket2[f, a, fa, b, fb, c, f[c], erro, lambdaErro, prec];
   If[d1 == "terminar", Return[{"TERMINAR", {a1, b1}}]];
```

```
    If[Abs[fa1] < Abs[fb1],
            (u = a1; q = fa1 / fb1; c = a1 - 2 * (b1 - a1) * q / (1 - q)),
                 (*else*)
             (u = b1; q = fb1 / fa1; c = b1 - 2 * (a1 - b1) * q / (1 - q))];

    If[Abs[c - u] > (1 / 2) * (b1 - a1), c = a1 + (b1 - a1) / 2];
    {a1, b1, d1, fa1, fb1, fd1} =
     bracket2[f, a1, fa1, b1, fb1, c, f[c], erro, lambdaErro, prec];
    If[d1 == "terminar", Return[{"TERMINAR", {a1, b1}}]];

    If[(b1 - a1) >= miu * (b - a), {a1, b1, d1, fa1, fb1, fd1} =
      bracket2[f, a1, fa1, b1, fb1, aux = a1 + (b1 - a1) / 2, f[aux], erro, lambdaErro, prec]
    If[d1 == "terminar", Return[{"TERMINAR", {a1, b1}}]]];

    {"CONTINUAR", {a1, b1, d1, fa1, fb1, fd1, f, erro, lambdaErro, miu, prec}}
   )
  ];



(* ============================ Algoritmo V ======================== *)


Clear[Algoritmo5, f, a, b, opcoes];
Options[Algoritmo5] = {
            Tolerancia -> 10^-6,
            LambdaTo1 -> 7 / 10,
            Miu -> 1 / 2,
            MaxIteracoes -> 15 ,
            PrecisaoUsada -> $MachinePrecision,
            PrecisaoFixa -> False,
            MostrarIteracoes -> False
          };


Algoritmo5[f_, a_, b_, opcoes___] :=
  Module[{intervalos, a0, b0, c1, a2, b2, d2, fa2, fb2, fd2, fa0, fb0, niter = 1,
    result, erro, lambdaErro, miu, maxiter, prec, ver, fix, pmax, pmin, q},
    (
    {erro, lambdaErro, miu, maxiter, prec, ver, fix, a0, b0} =
     tratarOpcoes[algoritmo5, a, b, opcoes];

    If[prec != 16 && fix, (pmax = $MaxPrecision;
      pmin = $MinPrecision; $MaxPrecision = prec; $MinPrecision = prec)];

    intervalos = {{a0, b0}};
    fa0 = f[a0];
    fb0 = f[b0];
    If[Abs[fa0] < Abs[fb0],
            (q = fa0 / fb0; c1 = a0 - (b0 - a0) * q / (1 - q)),
                (*else*)
            (q = fb0 / fa0; c1 = b0 - (a0 - b0) * q / (1 - q))];

   {a2, b2, d2, fa2, fb2, fd2} =
    bracket2[f, a0, fa0, b0, fb0, c1, f[c1], erro, lambdaErro, prec];
```

```
    If[d2 == "terminar", intervalos = Insert[intervalos, {a2, b2}, -1],
                (*else*)
          result =
      {"CONTINUAR", {a2, b2, d2, fa2, fb2, fd2, f, erro, lambdaErro, miu, prec}};

          While[result[[1]] != "TERMINAR" && niter <= maxiter,
                 (niter++;
                  result = iteracao5[result[[2]]];
                  intervalos =
        Insert[  intervalos, {result[[2, 1]], result[[2, 2]]}, -1]
           )];
     ];

    If[prec != 16 && fix, ($MaxPrecision = pmax; $MinPrecision = pmin)];

     If[ver,
          Print["\n Algoritmo V  \n "];
          Print[TableForm[N[intervalos, prec]]];
          Print["\nNúmero de avaliações da função: ", nfuncoes];]

      If[niter > maxiter, Print["Não se obteve convergência
          com a tolerância desejada após ", maxiter, " iterações."]];
      N[intervalos[[niter]], prec]
     )
    ];


Clear[iteracao5, f, a, b, d, fa, fb, fd, erro, lambdaErro, miu, prec];
iteracao5[{a_, b_, d_, fa_, fb_, fd_, f_Symbol, erro_, lambdaErro_, miu_, prec_}] :=
 Module[{c, a1, b1, fa1, fb1, d1, fd1, u, aux, q},
   (
    c = newtonQuadratic[a, b, d, fa, fb, fd, 2, prec];
    {a1, b1, d1, fa1, fb1, fd1} =
     bracket2[f, a, fa, b, fb, c, f[c], erro, lambdaErro, prec];
    If[d1 == "terminar", Return[{"TERMINAR", {a1, b1}}]];

    c = newtonQuadratic[a1, b1, d1, fa1, fb1, fd1, 3, prec];
    {a1, b1, d1, fa1, fb1, fd1} =
     bracket2[f, a1, fa1, b1, fb1, c, f[c], erro, lambdaErro, prec];
    If[d1 == "terminar", Return[{"TERMINAR", {a1, b1}}]];

    If[Abs[fa1] < Abs[fb1],
            (u = a1; q = fa1 / fb1; c = a1 - 2 * (b1 - a1) * q / (1 - q)),
                 (*else*)
             (u = b1; q = fb1 / fa1; c = b1 - 2 * (a1 - b1) * q / (1 - q))];

    If[Abs[c - u] > (1 / 2) * (b1 - a1), c = a1 + (b1 - a1) / 2];
    {a1, b1, d1, fa1, fb1, fd1} =
     bracket2[f, a1, fa1, b1, fb1, c, f[c], erro, lambdaErro, prec];
    If[d1 == "terminar", Return[{"TERMINAR", {a1, b1}}]];

    If[(b1 - a1) >= miu * (b - a), {a1, b1, d1, fa1, fb1, fd1} =
      bracket2[f, a1, fa1, b1, fb1, aux = a1 + (b1 - a1) / 2, f[aux], erro, lambdaErro, prec]
    If[d1 == "terminar", Return[{"TERMINAR", {a1, b1}}]]];
```

```
    {"CONTINUAR", {a1, b1, d1, fa1, fb1, fd1, f, erro, lambdaErro, miu, prec}}
  )
];


(* ========================== Algoritmo VI ====================== *)


Clear[Algoritmo6, f, a, b, opcoes];
Options[Algoritmo6] = {
            Tolerancia -> 10^-6,
            LambdaTol -> 7 / 10,
            Miu -> 1 / 2,
            MaxIteracoes -> 15 ,
            PrecisaoUsada -> $MachinePrecision,
            PrecisaoFixa -> False,
            MostrarIteracoes -> False
          };

Algoritmo6[f_, a_, b_, opcoes___] :=
  Module[{intervalos, a0, b0, c1, a2, b2, d2, fa2, fb2, fd2, fa0, fb0, niter = 1,
   result, erro, lambdaErro, miu, maxiter, prec, ver, fix, pmax, pmin, q},
   (
    {erro, lambdaErro, miu, maxiter, prec, ver, fix, a0, b0} =
    tratarOpcoes[algoritmo6, a, b, opcoes];

    If[prec != 16 && fix, (pmax = $MaxPrecision;
     pmin = $MinPrecision; $MaxPrecision = prec; $MinPrecision = prec)];

    intervalos = {{a0, b0}};
    fa0 = f[a0];
    fb0 = f[b0];
    If[Abs[fa0] < Abs[fb0],
            (q = fa0 / fb0; c1 = a0 - (b0 - a0) * q / (1 - q)),
                (*else*)
            (q = fb0 / fa0; c1 = b0 - (a0 - b0) * q / (1 - q))];

   {a2, b2, d2, fa2, fb2, fd2} =
    bracket2[f, a0, fa0, b0, fb0, c1, f[c1], erro, lambdaErro, prec];
   If[d2 == "terminar",
          intervalos = Insert[intervalos, {a2, b2}, -1],
                (*else*)
          result =
     {"CONTINUAR", {a2, b2, d2, d2, fa2, fb2, fd2, fd2, f, erro, lambdaErro, miu, prec}};
          (*Para n=2 teremos sempre interp. quadrática!*)

          While[result[[1]] != "TERMINAR" && niter <= maxiter,
            (niter++;
             result = iteracao6[result[[2]]];

             intervalos = Insert[  intervalos, {result[[2, 1]], result[[2, 2]]}, -1]
           )];
   ];

  If[prec != 16 && fix, ($MaxPrecision = pmax; $MinPrecision = pmin)];
```

```
  If[ver,
        Print["\n Algoritmo VI  \n "];
        Print[TableForm[N[intervalos, prec]]];
        Print["\nNúmero de avaliações da função: ", nfuncoes];]

     If[niter > maxiter, Print["Não se obteve convergência
         com a tolerância desejada após ", maxiter, " iterações."]];
     N[intervalos[[niter]], prec]

   )
  ];


Clear[ipzero, a, b, c, d];
ipzero[a_, b_, c_, d_, fa_, fb_, fc_, fd_] :=
 Module[{q11, q21, q31, d21, d31, q22, q32, d32, q33, q, aprox},

       If[
   Abs[fc] < Abs[fd], q = fc / fd; q11 = (c - d) * q / (1 - q), q11 = (c - d) * fc / (fd - fc)];
       If[
   Abs[fb] < Abs[fc], q = fb / fc; q21 = (b - c) * q / (1 - q), q21 = (b - c) * fb / (fc - fb)];
       If[Abs[fa] < Abs[fb], q = fa / fb; q31 = (a - b) * q / (1 - q), q31 = (a - b) * fa / (fb - fa)
       If[
   Abs[fc] < Abs[fb], q = fc / fb; d21 = (c - b) * q / (1 - q), d21 = (b - c) * fc / (fc - fb)];
       If[Abs[fb] < Abs[fa], q = fb / fa; d31 = (b - a) * q / (1 - q), d31 = (a - b) * fb / (fb - fa)
       If[Abs[fb] < Abs[fd],
   q = fb / fd; q22 = (d21 - q11) * q / (1 - q), q22 = (d21 - q11) * fb / (fd - fb)];
       If[Abs[fa] < Abs[fc],
   q = fa / fc; q32 = (d31 - q21) * q / (1 - q), q32 = (d31 - q21) * fa / (fc - fa)];
       If[Abs[fc] < Abs[fa],
   q = fc / fa; d32 = (q21 - d31) * q / (1 - q), d32 = (d31 - q21) * fc / (fc - fa)];
       If[Abs[fa] < Abs[fd],
   q = fa / fd; q33 = (d32 - q22) * q / (1 - q), q33 = (d32 - q22) * fa / (fd - fa)];
       aprox = a + q31 + q32 + q33
  ];


Clear[zeroP2ouP3, a, b, d, e, fa, fb, fd, fe, prec];
zeroP2ouP3[a_, b_, d_, e_, fa_, fb_, fd_, fe_, prec_] :=
  Module[{c},
    If[Length[Union[{fa, fb, fd, fe}]] < 4, newtonQuadratic[a, b, d, fa, fb, fd, 2, prec],
          (*else*)
         c = ipzero [a, b, d, e, fa, fb, fd, fe];
         If[(c - a) (c - b) >= 0, newtonQuadratic[a, b, d, fa, fb, fd, 2, prec], c]
    ]
];



Clear[iteracao6, a, b, d, e, fa, fb, fd, fe, f, erro, lambdaErro, miu, prec];
iteracao6[
  {a_, b_, d_, e_, fa_, fb_, fd_, fe_, f_Symbol, erro_, lambdaErro_, miu_, prec_}] :=
 Module[{c, a1, b1, d1, fa1, fb1, fd1, u,
   a2, b2, d2, fa2, fb2, fd2, a3, b3, d3, e3, fa3, fb3, fd3, fe3, aux, q},
```

```
 (
   c = zeroP2ouP3[a, b, d, e, fa, fb, fd, fe, prec];
   {a1, b1, d1, fa1, fb1, fd1} = bracket2[f, a, fa, b, fb, c, f[c], erro, lambdaErro, prec]

    If[d1 == "terminar", Return[{"TERMINAR", {a1, b1}}]];

    If[Abs[fa1] < Abs[fb1],
            (u = a1; q = fa1 / fb1; c = a1 - 2 * (b1 - a1) * q / (1 - q)),
                 (*else*)
             (u = b1; q = fb1 / fa1; c = b1 - 2 * (a1 - b1) * q / (1 - q))];

   If[Abs[c - u] > (1 / 2) * (b1 - a1), c = a1 + (b1 - a1) / 2];
   {a2, b2, d2, fa2, fb2, fd2} =
    bracket2[f, a1, fa1, b1, fb1, c, f[c], erro, lambdaErro, prec];
   If[d2 == "terminar", Return[{"TERMINAR", {a2, b2}}]];

   If[(b2 - a2) < miu * (b - a),
      {a3, b3, d3, e3, fa3, fb3, fd3, fe3} = {a2, b2, d2, d1, fa2, fb2, fd2, fd1},
        (*else*)
      {a3, b3, d3, e3, fa3, fb3, fd3, fe3} = Insert[Insert[bracket2[f, a2, fa2, b2, fb2,
        aux = a2 + (b2 - a2) / 2, f[aux], erro, lambdaErro, prec], d2, 4], fd2, -1];
     If[d3 == "terminar", Return[{"TERMINAR", {a3, b3}}]]];

    {"CONTINUAR", {a3, b3, d3, e3, fa3, fb3, fd3, fe3, f, erro, lambdaErro, miu, prec}}
 )
];


 (* ============================ Algoritmo VII ======================== *)


Clear[Algoritmo7, f, a, b, opcoes];
Options[Algoritmo7] = {
            Tolerancia -> 10^-6,
            LambdaTol -> 7 / 10,
            Miu -> 1 / 2,
            MaxIteracoes -> 15 ,
            PrecisaoUsada -> $MachinePrecision,
            PrecisaoFixa -> False,
            MostrarIteracoes -> False
          };

Algoritmo7[f_, a_, b_, opcoes___] :=
   Module[{intervalos, a0, b0, c1, a2, b2, d2, fa2, fb2, fd2, fa0, fb0, niter = 1,
   result, erro, lambdaErro, miu, maxiter, prec, ver, fix, pmax, pmin, q},
   (
    {erro, lambdaErro, miu, maxiter, prec, ver, fix, a0, b0} =
    tratarOpcoes[algoritmo7, a, b, opcoes];

    If[prec != 16 && fix, (pmax = $MaxPrecision;
     pmin = $MinPrecision; $MaxPrecision = prec; $MinPrecision = prec)];

    intervalos = {{a0, b0}};
   fa0 = f[a0];
   fb0 = f[b0];
```

```
    If[Abs[fa0] < Abs[fb0],
            (q = fa0 / fb0; c1 = a0 - (b0 - a0) * q / (1 - q)),
               (*else*)
            (q = fb0 / fa0; c1 = b0 - (a0 - b0) * q / (1 - q))];

    {a2, b2, d2, fa2, fb2, fd2} =
     bracket2[f, a0, fa0, b0, fb0, c1, f[c1], erro, lambdaErro, prec];

    If[d2 == "terminar",
          intervalos = Insert[intervalos, {a2, b2}, -1],
                (*else*)
          result =
      {"CONTINUAR", {a2, b2, d2, d2, fa2, fb2, fd2, fd2, f, erro, lambdaErro, miu, prec}};
       (*Para n=2 teremos sempre interp. quadrática!*)

          While[result[[1]] != "TERMINAR" && niter <= maxiter,
            (niter++;
             result = iteracao7[result[[2]]];
             intervalos = Insert[  intervalos, {result[[2, 1]], result[[2, 2]]}, -1]
           )];
    ];

    If[prec != 16 && fix, ($MaxPrecision = pmax; $MinPrecision = pmin)];

    If[ver,
          Print["\n Algoritmo VII  \n "];
          Print[TableForm[N[intervalos, prec]]];
          Print["\nNúmero de avaliações da função: ", nfuncoes];]

    If[niter > maxiter, Print["Não se obteve convergência
         com a tolerância desejada após ", maxiter, " iterações."]];
    N[intervalos[[niter]], prec]

    )
   ];


Clear[iteracao7, a, b, d, e, fa, fb, fd, fe, f, erro, lambdaErro, miu, prec];
iteracao7[
  {a_, b_, d_, e_, fa_, fb_, fd_, fe_, f_Symbol, erro_, lambdaErro_, miu_, prec_}] :=
 Module[{c, e1, fe1, a1, b1, d1, fa1, fb1, fd1, u,
   a2, b2, d2, fa2, fb2, fd2, a3, b3, d3, e3, fa3, fb3, fd3, fe3, aux, q},
   (
   c = zeroP2ouP3[a, b, d, e, fa, fb, fd, fe, prec];
   {a1, b1, d1, fa1, fb1, fd1} = bracket2[f, a, fa, b, fb, c, f[c], erro, lambdaErro, prec]

   If[d1 == "terminar", Return[{"TERMINAR", {a1, b1}}]];
   e1 = d;
   fe1 = fd;
   c = zeroP2ouP3[a1, b1, d1, e1, fa1, fb1, fd1, fe1, prec];
   {a1, b1, d1, fa1, fb1, fd1} =
    bracket2[f, a1, fa1, b1, fb1, c, f[c], erro, lambdaErro, prec];
   If[d1 == "terminar", Return[{"TERMINAR", {a1, b1}}]];

   If[Abs[fa1] < Abs[fb1],
```

```
            (u = a1; q = fa1 / fb1; c = a1 - 2 * (b1 - a1) * q / (1 - q)),
                 (*else*)
             (u = b1; q = fb1 / fa1; c = b1 - 2 * (a1 - b1) * q / (1 - q))];

    If[Abs[c - u] > (1 / 2) * (b1 - a1), c = a1 + (b1 - a1) / 2];
    {a2, b2, d2, fa2, fb2, fd2} =
     bracket2[f, a1, fa1, b1, fb1, c, f[c], erro, lambdaErro, prec];
    If[d2 == "terminar", Return[{"TERMINAR", {a2, b2}}]];

    If[(b2 - a2) < miu * (b - a),
       {a3, b3, d3, e3, fa3, fb3, fd3, fe3} = {a2, b2, d2, d1, fa2, fb2, fd2, fd1},
          (*else*)
      {a3, b3, d3, e3, fa3, fb3, fd3, fe3} = Insert[Insert[bracket2[f, a2, fa2, b2, fb2,
        aux = a2 + (b2 - a2) / 2, f[aux], erro, lambdaErro, prec], d2, 4], fd2, -1];
     If[d3 == "terminar", Return[{"TERMINAR", {a3, b3}}]]];

    {"CONTINUAR", {a3, b3, d3, e3, fa3, fb3, fd3, fe3, f, erro, lambdaErro, miu, prec}}
  )
 ];


End[];
EndPackage[];
```

Dois exemplos de utilização do programa

```
In[3]:= Clear[g, x];
        g[x_] := 1 / 2 * Log[1 / 100 + x^2] + ArcTan[10 x] - π / 2
        Algoritmo7[g, 1, 2, Tolerancia -> 0, PrecisaoUsada -> 45]

Out[4]= {1.09112676723482621166689809745234040578 72,
         1.09112676723482621166689809745234040578 72}

In[5]:= Algoritmo7[g, 1, 2, Tolerancia -> 10^-10, MostrarIteracoes -> True]

        Algoritmo VII

        1.                    2.
        1.091126710568544     1.091126829536338
        1.091126767234826     1.091126767234826

        Número de avaliações da função: 7

Out[5]= {1.091126767234826, 1.091126767234826}

In[6]:= Clear[f, x];
        f[x_] := -2 Σ_{i=1}^{20} (2 i - 5)^2 / (x - i^2)^3;
        Algoritmo5[f, 4 + 10^-4, 9 - 10^-4, Tolerancia -> 0, PrecisaoUsada -> 100]

Out[6]= {6.68375356080807808143026265054007171669369158973368197752751794973053,
         6.68375356080807808143026265054007171669369158973368197752751794973053}

In[7]:= Algoritmo5[f, 4 + 10^-4, 9 - 10^-4, MostrarIteracoes -> True]

        Algoritmo V

        4.0001                8.9999
        6.656244756299385     6.812488112598389
        6.683753005590974     6.683754405590978

        Número de avaliações da função: 9

Out[7]= {6.683753005590974, 6.683754405590978}

In[8]:= Algoritmo7[f, 4 + 10^-4, 9 - 10^-4, Tolerancia -> 0, PrecisaoFixa -> True,
         PrecisaoUsada -> 45]

Out[8]= {6.68375356080807808143026265054007171669369 16,
         6.68375356080807808143026265054007171669369 17}
```

Bibliografia

[1] G. E. Alefeld, F. A. Potra. *Some Efficient Methods for Enclosing Simple Zeros of Nonlinear Equations.* BIT 32 (1992), 333-334.

[2] G. E. Alefeld, F. A. Potra, Yixun Shi. *On Enclosing Simple Roots of Nonlinear Equations.* Mathematics of Computation, Vol. 61, N°.204 (1993), 733-744.

[3] G. E. Alefeld, F. A. Potra, Yixun Shi. *Algorithm 748: Enclosing Zeros of Continuous Functions.* ACM Transations on Mathematical Software, Vol. 21, N°.3 (1995), 327-344.

[4] G. E. Alefeld, F. Potra, W. Völker. *Effective Improvements of the Interval-Newton-Method.* Scientific Computing and Validated Numerics. Mathematical Research, Volume 90. Akademie Verlag GmbH, Berlin (1996). G. Alefeld, A. Frommer and B. Lang (editores).

[5] G. Alefeld and J. Herzberger. *Introduction to Interval Computations.* Academic Press Inc., New York (1983).

[6] Abraham Berman and Robert J. Plemmons. *Nonnegative Matrices in the Mathematical Sciences.* Academic Press, New York (1994).

[7] R. P. Brent. *Algorithms for minimization without derivatives.* Prentice-Hall, Englewood Cliffs, NJ (1972).

[8] James L. Buchanan and Peter R. Turner. *Numerical Methods and Analysis.* McGraw-Hill, Inc. (1992).

[9] T. J. Dekker. *Finding a zero by means of successive linear interpolation.* In *Constructive Aspects of the Fundamental Theorem of Algebra.* B. Dejon and P. Henrici. Wiley Interscience (1969).

[10] Gwynne Evans. *Practical Numerical Analysis.* John Wiley & Sons Ltd. (1995).

[11] Edite Manuela da G. P. Fernandes. *Computação Numérica.* Serviços de Reprografia e Publicações da Universidade do Minho, Braga (1996).

[12] George E. Forsythe, Michael A. Malcolm and Cleve B. Moler. *Computer Methods for Mathematical Computations.* Prentice-Hall, Inc. (1977).

[13] Peter Henrici. *Elements of Numerical Analysis.* John Wiley & Sons, Inc., New York (1964).

[14] J. Herzberger (editor). *Topics in Validated Computations.* Studies in Computational Mathematics 5. Elsevier Science B. V. (1994).

[15] F. A. Potra. *On Q-Order and R-Order of Convergence.* Journal Optimization Theory and Applications 63, 415-431 (1989).

[16] Maria Raquel Valença. *Métodos Numéricos* (3^a Edição). Livraria Minho, Braga (1993).

[17] Maria Raquel Valença. *Notas sobre o Mathematica.* Universidade do Minho (1998).

[18] Ramon E. Moore. *Methods and Applications of Interval Analysis.* SIAM Studies in Applied Mathematics. SIAM, Philadelphia (1979).

[19] Arnold Neumaier. *Interval Methods for Systems of Equations.* Cambridge University Press (1990).

[20] J. M. Ortega and W. C. Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables.*Academic Press Inc., New York (1970).

[21] A. M. Ostrowski. *Solution of Equations and Systems of Equations.* Academic Press, New York (1960).

[22] A. M. Ostrowski. *Solution of Equations in Euclidean and Banach Spaces.* Academic Press Inc., New York (1973).

[23] Antony Ralston and Philip Rabinowitz. *A First Curse In Numerical Analysis.* McGraw-Hill, Inc. (1978).

[24] J. W. Schmidt. *On the R-Order of Coupled Sequences.* Computing 26, 333-342 (1981).

[25] H. R. Schwarz. *Numerical Analysis. A Comprehensive Introduction.* John Wiley & Sons Ltd. (1989).

[26] L. F. Shampine and R. C. Allen, Jr.. *Numerical Computing: an introduction.* W. B. Saunders Company, Philadelphia (1973).

[27] J. Stoer and R. Burlirsch. *Introduction to Numerical Analysis* (Second Edition). Springer-Verlag, New York (1993).

[28] James S. Vandergraft. *Introduction to Numerical Computations.* Academic Press Inc., New York (1978).

[29] Richard S. Varga. *Matrix Iterative Analysis.* Prentice-Hall Inc., EngleWood Cliffs, N. J. (1962).

[30] G. H. Wilkinson. *Rounding Errors in Algebraic Processes.* Dover Publications, Inc, New York (1994).

[31] Stephen Wolfram. *The Mathematica Book.* Third Edition, Mathematica Version 3. Cambridge, University Press (1996).

[32] R. D. Skeel and J. B. Keiper. *Elementary Numerical Computing with Mathematica.* McGraw-Hill, Inc. (1993).